

## Welcome to Netica's Help System

This system is designed to offer the most up-to-date documentation on = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Netica\_Application.htm');return false;">Netica Application, the world's most widely used = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayesian network development software, from [Norsys Software Corp.](#)

**Here are some tips for effectively finding the information you need:**

**Navigation Buttons:** Use the small white previous and next arrows in the side panel to sequentially page through the Help system. Use the Back button in your web browser to return to the last page you were visiting if a link takes you out of sequence.

**Styles:** We recommend becoming familiar with these hyper-link styles, as they are used throughout the documentation:

= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_popup.htm');return false;">popup link = opens popup text on your screen

[Encyclopedia link](#) = takes you to an Encyclopedia page

[general link](#) = takes you to another page within the help system

[web link](#) = opens a web browser page or enables e-mail

[Glossary link](#) = takes you to the indicated Glossary entry

**Set-up:** First, open the **Table of Contents** (with the "Show" or the "Contents" button). Then we recommend re-sizing the window to your viewing preference.

**Sequential Reading:** This help system is laid out in logical chapters,

ascending somewhat by level of expertise. You can read through the entire system by using the small white previous and next arrows in the side panel (the chapters typically build upon each other). Alternatively, you can open the **Table of Contents** and go straight to your desired topic.

**The Index:** Whenever you have a problem with Netica, or need some information on how it is working, your first step should be to check the **Index** (click the Index tab in the side panel). We have taken great care to create a very comprehensive and useful index. If there are entries that you feel should be added, please `if (typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_email.htm');return false;">inform us.`

**Other Resources:** Explore how other people are using Netica by visiting our [online library](#). You can download nets and might get some ideas for how best to structure your current net. We also encourage you to explore our [online tutorial](#) for comprehensive information at the beginner, intermediate and advanced levels.

**Contact Us:** If there is anything you can't figure out by using this Help system, feel free to contact us via [email](#) or other means of [company communications](#).

Now, it is time to [Get Started!](#)

## Getting Started

Welcome to Netica Application from = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Norsys.htm');return false;">Norsys. Netica is a versatile, fast, user-friendly program that you can use to find patterns in data, create diagrams encoding knowledge or representing decision problems, use these to answer queries and find optimal decisions, and create probabilistic expert systems. It is suitable for applications in the areas of diagnosis, prediction, decision analysis, probabilistic modeling, risk management, expert system building, sensor fusion, reliability analysis, and certain kinds of statistical analysis and data mining.

**Netica API:** This guide is for Netica Application; it is not for the Netica Programmer's Library, also known as "[Netica API](#)" (Application Program Interface). The API is a module with much of the same functionality as Netica Application, but designed for programmers to embed in their programs. For help with the API, use the [C API Webdocs](#) or other online [API manuals](#).

**Web Site:** You can visit the Norsys [website](#) for the [latest version](#) of Netica, versions of Netica for other platforms, Netica API, example nets, tutorials and more information.

**Version:** You can determine the version of this guide by clicking on the Netica icon in the upper right hand of the screen. Ideally, it should at least approximately match the version of Netica, which you can find by choosing **Help** → **About Netica** from within Netica. ([features of new version](#))

**Prerequisites:** This guide assumes that you are familiar with using the Microsoft Windows operating system. It also assumes familiarity with Bayesian belief networks (= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes nets) or influence diagrams, although it is very suitable for someone who is in the process of learning about them. To learn more about the concepts and applications of BBNs, explore our [introductory references](#), the tutorial nets in our [Bayes net library](#), and the tutorial examples that came with your

[Netica download.](#)

**Next Steps:** First you may want to read the [legalities](#), and a description of [Netica Application](#) and [Netica API](#). Then [install](#) Netica and take the [Quick Tour](#) for an introduction. For background reading, you may want to see [introductory references](#). Then you are ready for the in-depth [topics](#) of Netica.

**Feedback:** Please = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_email.htm');return false;">email Norsys with your questions and comments about Netica or this onscreen help document. To learn more about our design philosophy, click [here](#).

## Netica Application

Netica Application is a comprehensive tool for working with Bayesian = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">belief nets and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision nets (influence diagrams). It can build, learn, modify, transform and store nets, as well as answer queries or find optimal solutions using its powerful inference engine. This version has many [new features](#) and many more are currently [under development](#).

Netica can also work on a Mac. [More Info](#)

[Click here](#) for **new features**.

### Features:

- Compiles Bayes nets into junction trees of cliques for fast probabilistic reasoning.
- Can [learn probabilistic relations](#) from data (including EM and gradient descent learning).
- Generates presentation quality graphics which can be transferred to other documents, including SVG graphics.
- Allows the entry of probabilistic relations by [equation](#), with an extensive built-in library of probabilistic functions and other mathematical functions. The equations can be deterministic or probabilistic, and for discrete or continuous variables.
- Provides easy graphical editing of Bayes nets and influence diagrams, including:
  - Cutting and pasting of nodes and nets without losing their probabilistic relations.
  - Many ways of displaying nodes (bar graphs, meters, etc.)
  - Links with bends to keep complex diagrams orderly.
  - Allows entering comments to document each node, keeps track of author,

when changed, etc.

- Unlimited levels of undo/redo.
- Can create and work with sets of nodes including color-coding nodes.
- Comment windows for nodes, links and states.
- Dynamic scrolling and mouse wheel supported everywhere.
- Can find optimal decisions for sequential decision problems (i.e. later decisions are dependent on the results of earlier ones).
- Can test the performance of a Bayes net using a file of cases. Netica will print out a confusion matrix, error rate, logarithmic and quadratic (Brier) scoring rule results, calibration table and surprise indexes for each node desired.
- Can do utility-free [sensitivity analysis](#).
- Can reverse individual links and “sum out” nodes of influence diagrams or belief nets, for model exploration and refinement.
- Supports [disconnected links](#), which makes possible libraries of probabilistic relationships.
- Has facilities to enter and update individual cases, store them in their own files, and apply them to other Bayes nets.
- Has facilities for the easy discretization of continuous variables.
- Has no built-in limits on the size or complexity of nets, so they are limited only by available memory.
- Can work hand-in-hand with the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Netica_API.htm');return false;">Netica API Programmer's Library`.
- Can directly connect with a database or [Excel spreadsheet for learning](#), reading cases, testing a net, etc.
- Has binary .neta format for faster and smaller files (text-based .dne files still have complete support).
- Can obfuscate nets, and can work with encrypted Bayes nets, to protect your intellectual property.
- Can automatically read in a case, compile the Bayes net when it is read from disk, and has auto-discretization, based on case files.

## FAQs - Running Netica on a Mac

To get Netica to work on your Mac, you'll need to first [install a compatibility layer](#) for running Windows programs, and then you can [install Netica](#) in the usual way. There are several possibilities for compatibility layers (see below), but we have found that [CrossOver](#) is one of the least expensive and easiest-to-use options. It allows Netica to run with full speed and capability. Netica has been created to run well under even the most basic of conditions, so there are no serious issues using Netica under Crossover (check out their [review](#) of Netica's compatibility).

*See the bottom of this screen for the known issues and simple work-arounds for running Netica on your Mac.*

### Why am I downloading a trial version of CrossOver?

Codeweavers offers a 30 day full featured version of CrossOver, so you can try it out for free and get support along the way. Once you're satisfied with its performance on your Mac, you can purchase it for \$39.95 (which goes to support the open-source development).

### Are there alternative options to using CrossOver?

There are other options for running Windows-based programs on your Mac, such as Parallels, VMware Fusion, and Apple Bootcamp. Each solution has its advantages and disadvantages. Netica works well on all of them. These other solutions will enable you to run a wider variety of Windows programs but are more expensive and larger.

This [differentiation chart](#) may be helpful in making your decision and there are many other reviews online.

### Is there a way to get WINE on my machine for free?

If you are comfortable with a more technical approach to getting WINE on your machine, you can try out the free [Wine Binary Downloads](#), as well as the [Recommended Packages](#) for building Wine on 32bit.

### Will this download pollute my Mac?

No! CrossOver is just a compatibility layer (in other words, you are not actually installing Windows on your Mac), and it won't affect anything else on your Mac.

### Can I easily get rid of CrossOver if I decide I don't want it on my machine?

Yes! To erase all traces of CrossOver, follow [these steps](#).

### I'm having trouble installing CrossOver, help!

Go through the steps of the official [Installation Guide](#). If you're still having trouble, check out their [FAQ page](#).

### Once I get Netica running on my machine, are there any known problems with the software?

There are a few minor issues, all with quick work-arounds:

**1. Right-clicking:** Control-click, as you're probably used to with Mac programs, doesn't work that way under WINE (so far).

**Workaround:** Enable secondary click. To do this, go to **System Preferences > Keyboard & Mouse > Trackpad** and enable 'Tap trackpad using two fingers for secondary click'. You can then right-click by tapping the trackpad with two fingers.

**Update:** If you are running a current version of Xquartz, you can go into the X11.app preferences when it's running and select what you want to be modifier keys for right and middle clicks.

**2. Messages window:** When Netica first opens, the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Messages_window.htm');return false;">Messages window` appears in word format in the upper left of the screen instead of bottom left.

**Solution:** Double-click the word to open the window. Moving forward, the Messages window may appear as a blank white square in the bottom

left of the Netica screen. Again, you can double-click the white square to bring up the Messages window.

**3. Onscreen Help:** the help system built into Netica does not launch when clicked.

***Solution:*** Use this online Help system for assistance and for documentation on all the latest features.

**4. Learning from Excel files:** You will get error messages if attempting to [learn from an Excel file](#).

***Solution:*** You must convert your Excel case file into a text file and then you can proceed with the steps listed in [Learning From a Case File](#).

Help us improve Netica's functionality on Mac by sending us your [questions or feedback!](#)

## New Features

To check your version of Netica, open Netica and choose **Help** → **About Netica**. Depending on your version, you may want to check for recent pre-release versions, available from our [ftp site](#).

You can try any version of Netica in 'limited mode' (without a password). Or, you can [upgrade](#) your current Netica license.

In addition to its [standard features](#), = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Netica\_Application.htm');return false;">Netica has the following new features.

### Version 4 (-) and version 5 (•) new features:

- [Generate websites](#) from Bayes nets: After creating any Bayes net, you can select the target nodes, push a button and your browser will spring up running your Bayes net as a question/answer system or as a dashboard with sliders. Best of all, your website is already deployed to the world hosted on our system (or you can host it, or make a desktop-only version). See [Example site](#).

Full usage of this system is sold as a separate product, and complete documentation for this feature is in a separate manual. [Contact Norsys](#) for details.

- [Structure Learning](#) - Netica can now do TAN learning of link structure from data. Over the series of version 5 releases, we will be further adding to Netica's learning-from-data capability (structure, parameter, testing).
- [Custom Reports](#) - Can generate highly customizable reports on many aspects of the Bayes net, nodes, states, CPTs, cases, findings, beliefs, sensitivity results, other inference results, etc. The reports can range from basic text output to nicely formatted tables. They can be in the form of HTML, XML, text, rich text, etc, based on template files that you make or choose from our library of templates.
- [Actions](#) - Can enter actions or interventions (i.e., Pearl's do-calculus) to nature nodes instead of just findings. Also handles calibration actions (aka

randomized actions or population interventions).

- Can directly move belief bars with the mouse to enter calibration findings and calibration actions
- Major improvements to [dynamic Bayes net](#) (DBN) capability for time series. Now does a "burn-in" to generate initial-state nodes.
- [EM learning](#) now handles multiple CPT tables constrained to be the same (indicated by the user field "CPT\_ID" having the same value).
- [Build table from other net](#) is a very versatile feature that among other things can fill the CPT tables of nodes in one net based on another net with a different structure.
- Option for inference by sampling when necessary (using rejection method).
- Can now do all Bayes net operations (such as inference) when there are some [disconnected links](#).
- Capability "Split Node" converts a multi-state node into a set of Boolean nodes, one node for each state of the original node.
- Can sort the states of a node (or nodes) by name, belief, other node, standard order, reverse order. Standard order is user-definable. Other node is very interesting; it uses Bayesian inference to make the order compatible with the ordering of states of another node (the selected node).
- Added [NoisyOrMultiDist function](#), available from equations, to generalize the noisy-or to multiple states, in a way usually more fitting than noisy-max does.
- Adding/Deleting/Renaming states by right-clicking can now do many states at a time (even to [multiple nodes](#)), and can add state names, numbers and titles.
- In Node-set dialog, shift-clicking the [Set Color](#) button will auto-color all node-sets from the first one to the selected one, inclusive.
- EM learning now leaves findings in net while operating (and handles them properly; they can override the case file).
- Can now have passwords that work for all users on a machine or across a network.
- Fixed: Passwords would not stay registered on some Asian MS Windows machines.

- Greatly improved the [Combine Nets](#) feature.
- Added to the **Modify** → [Order States](#) options.
- Onscreen help now also has an [online internet help system](#), and it is available from [Netica's menu](#).

Other [smaller improvements](#)

### **From version 3.16 to version 3.25:**

- You can [change the order](#) of parents while editing a node's CPT table (and thereby the structure of the table) just by dragging its columns.
- There is now a great new way to navigate large net diagrams, called [Global Zoom](#). Also new is "[push scrolling](#)", done by holding the ALT key down and dragging on the background.
- Provides the ability to do the same operation to [many nodes at once](#), such as: adding, removing, renaming or reordering states, changing discretization thresholds, entering findings, entering user-defined fields, etc.
- When you change the [name of node](#), it automatically adjusts that node's equation and the equations of its child nodes. Also works for name changes due to duplicating nodes and [time expansions](#).
- Added much wider support for [right-clicking](#) on nodes, net, link, etc.
- Can view and edit tables of a node's '[experience](#)' and frequency counts when learning from data or connected to a database.
- Improved auto-discretizing algorithm.
- Added "Combine Nets" feature (in preliminary stage).

Other [smaller improvements](#)

### **From version 2.17 to version 3.16:**

- Can create and work with [sets of nodes](#).
- Can [color-code nodes](#).
- Can directly connect with a database or [Excel spreadsheet](#) for learning, reading cases, testing a net, etc.
- Can generate [SVG graphics](#) of Bayes nets, for quality web or print publishing.

- Has a new binary [.neta format](#) for faster and smaller Bayes net files (text-based .dne files will always continue to have complete support).
- Can [obfuscate nets](#), and can work with [encrypted](#) Bayes nets, to protect your intellectual property.
- Has [auto-discretization](#), based on case files.
- When hover cursor over [nodes](#), [states](#) or [links](#) Netica can display a comment of your choice in a floating window.
- Case files can now have uncertain findings using a very simple but powerful file format ([UVF](#)), and those uncertain findings will be properly handled by Netica's belief updating, EM learning, process cases, etc.
- [Automatically compiles](#) the Bayes net when it is read from disk, when appropriate.
- The [Meter](#) display for discretized nodes, or those with state values defined, now has a needle which shows expected value, and a band around it which displays standard deviation.
- Keeps track of when tables may need to be re-built from their equations.
- Many improvements to the CPT table editor, including better pasting, better scrolling, applying an operation to multiple cells, probabilities or percentages, experience, etc.
- Dynamic scrolling and mouse wheel supported everywhere.
- Now has a **Calibration** choice on **enter-findings** menu, to enter the likelihood finding that will result in a certain belief state.
- Fixed bugs, including:
  - When copy/pasting net graphics, on some systems, the lower right of the image was cut off.
  - In the node properties dialog, repeated error messages sometimes made it difficult to exit.
  - Belief linking to modern versions of Excel didn't update links.
  - Printing didn't work to some network printers or those with long names.
- Now also reads [Hugin 6.\\*](#) .net files (as well as 4.\* and 5.\*).
- Added **Table** → **Enter Experience** command to enter [experience tables](#) for all selected nodes at once.
- There is now a "-case" [command line](#) parameter to automatically read-in a

case.

- The expected value of a node is displayed even if one of the intervals extends to infinity.

### From version 1.12 to version 2.17:

- Has [EM learning](#) and gradient descent learning of CPT tables to better deal with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_missing\_data.htm');return false;">missing data.
- Has a [Process Cases](#) feature, to read cases from a file, process them with the Bayes net one-by-one, then output the results to a new file.
- Now has a [Recent Files](#) list on the menu.
- Added [“Go Back”](#) key command to scroll to the last place being edited on a net.
- Can now compile and update nets whose CPTs or utility tables are [absent or incomplete](#) (takes missing entries as uniform probabilities or zero utilities).
- There is a [node palette](#) window to keep commonly used nodes.
- Generating a table from an equation can use the integral of the equation, instead of sampling, for many common distributions.
- [Auto-updating](#) now works whenever it can (e.g., after changing a CPT).
- Can simultaneously add links to many nodes, or from many nodes, if they are selected when adding a link.
- Improved [time expansion](#), can now edit delays in node dialog, and there is a **Modify** → **Delay Links** menu command.
- Can now read [DXpress](#) files and BNIF (.dsc) files.
- Added Noisy-Max and Noisy-Sum capabilities (Noisy-Or and Noisy-And already exist).
- Can now observe and set node **user fields** using the [node properties box](#).
- Window title has indicator (\*) to show when it has [unsaved changes](#)
- Displays \*deterministic\* nature nodes in [Label Box style](#) using a thick border.
- [Command line](#) can accept a Netica password (-password xxx), which is used

for that session, but not entered into the registry.

- [Inference algorithm](#) takes better advantage of findings entered to speed inference.
- Equation-to-table expands the interval automatically if values are out-of-bounds.
- Added *log messages* command.
- Can set the [display of node states](#) to only the N most probable, ordered by most probable first, which changes as beliefs change.
- When enter findings, and [belief updating](#) not yet done, only those nodes affected will have their beliefs indicated invalid.
- When paste nodes into a network it expands the drawing size if necessary (instead of crunching them up).
- Added a **Select Nodes** menu which can [select nodes](#) with equations, findings, dependent on, parents, ancestors, children and descendants of selected nodes.
- Added a **Select Links** menu which can [select links](#): all, disconnected, ineffectual (zero strength), forming cycle, or entering, exiting or interconnecting selected nodes.
- `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_deterministic Updating.htm');return false;">Deterministic updating precedes belief updating, for accuracy and speed.`
- Can read or create case files with comma, space or tab delimiters, and with ?, \*, space or nothing as `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_missing_data.htm');return false;">missing data chars.`
- Case files don't normally have a [header](#) with `//~->[CASE-1]->~` and a time/author stamp any more (although still accepts that).
- Added **Layout** → **Spread Out/Compact** capability.

## Installation

**Legal:** Before installing or using Netica, be sure that you accept the [License Agreement](#) which is also provided with the software in a separate document.

**Requirements:** Netica Application requires a PC running any version of Microsoft Windows from 2000 to Windows 7 (including XP and Vista). If you need a version of Netica for an earlier version of Windows (such as Windows 95), then = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_email.htm');return false;">contact Norsys.

Installation requires less than 10 MB of hard disk space. ([Reason for Netica's small size](#))

Netica will run well even on a very slow PC (0.4 GHz), using very little RAM (about 30 MB), but working with complex Bayes nets may require much more speed and large amounts of RAM.

### Install:

1. First obtain the Netica package file. You may obtain it by downloading from the Norsys [website](#), or otherwise from Norsys. The name of the file will be Netica\_win.exe.
2. Choose "Run" from the download dialog box, or save the file to disk and then double-click its icon.
3. When a dialog box appears, enter where on your hard disk you want the Netica folder placed. You can put it anywhere you wish; popular choices are C:\Netica or C:\Program Files\Netica. Make sure there is a drive letter (e.g. "C:\") at the front of the location. You don't have to include version information in the name, because a folder called Netica ### will be created within it, where ### is the version number.
4. Click the UnZip button and then close the dialog box.

**Mac Installation:** To install Netica on a Mac, follow [these steps](#). Once you've completed installation, you'll want to become familiar with the [FAQ for running Netica on your Mac](#).

**Running:** Then, open the Netica ### folder from where you had it placed. This folder is known as the = 4 && typeof(BSPSPopupOnMouseOver) ==

```
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_Glossary_FM.htm#homedirectory');return false;">home folder. Within it will be all the files required for Netica, including the executable file called Netica (or Netica.exe), with this  icon.
```

The first time you run it you should right-click on it, and choose "Run as administrator" (if you want you can run it by just double-clicking it, but then it might not be fully registered with the system). On some systems it may take a long time to start the first time; please be patient and know that next time it will start lightening-fast.

**Password:** The enter-password dialog box will appear, so if you obtained a license *password* from Norsys by e-mail or on your invoice, you may type it in, or better yet copy and paste it in. If you later wish to remove or change the password, choose **File** → **Netica Password** from Netica's menu.

**Limited Mode:** You can use Netica without a password (i.e. 'Limited Mode'), and it will operate in a full featured and useful manner, but will not be able to do larger projects (i.e. can't save nets with more than 15 nodes, learn from more than 1000 cases at a time, etc.)

**Next:** Now that Netica is installed and running, you can get an introduction to how it works by doing the operations described in the [Quick Tour](#), or check out the [New Features](#).

If you wish Netica to be on your Start menu, you can use the normal Windows method of dragging the `Netica.exe` icon  from the Netica ### folder (as mentioned above) and dropping it on the Start button. Or drag it to the desktop if you want to make a shortcut there.

You can move the Netica folder to wherever you want on your hard drive at any time. After moving it, you will again need to run Netica as Administrator once (as described above) to register the new location.

**Multiple Versions:** If you have several different versions of Netica on your hard disk at the same time, one of them will be the "default" (e.g. it will be the one that will run when you double-click a Netica document). To change the default, simply manually run once the version of Netica as Administrator (as described above). There is never any need to uninstall any Netica version before installing any new one.

**Uninstall:** To uninstall Netica, just delete the "Netica" folder. Netica does not

add any other files anywhere in your system (except of course .neta, .dne or .cas documents that you create get placed where you save them), and Netica adds very little to the Windows Registry. If you have several versions of Netica, you can uninstall one version by simply deleting its folder; that won't delete any files needed by any other version.

## Menu Items and the Toolbar

In this guide, Netica's menu items are indicated in bold, with arrows indicating choices or submenus, so **File** → **Open** means choose “Open” from the “File” menu of the main menu bar. Most menu items have a corresponding toolbar button, and from within Netica you can discover what the buttons do by resting the cursor on them briefly. For example, it will say “Open File”, when you rest it on the  toolbar button.

Some of the toolbar buttons discussed in this guide may not appear on the toolbar when you first start Netica, so you may want to [customize the toolbar](#).

Instead of using the menu or a toolbar button, you can often use a keyboard [shortcut](#) or a [right-click](#) menu, which have a few additional features not available from the overhead menu.

## Quick Tour

This quick tour will guide you through some of the major [features](#) of Netica.

All you need to know to complete this tour is how to use Microsoft Windows.

You do not need knowledge of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes nets, = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision nets

or = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_Netica.htm');return false;">Netica, although you may not fully understand what is happening until you have that

knowledge.

First, make sure you have completed the [installation](#). You should duplicate the

= 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">Examples

folder in case you accidentally change some of the supplied files. Next, run

Netica by double-clicking on its icon . The first time you run Netica it will

ask you for a [password](#). Enter the license password supplied to you by = 4

&& typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_Norsys.htm');return false;">Norsys, or click on

the “Limited Mode” button if you don’t have one.

The workspace window will open and in its lower left corner there will be an

icon for a minimized window called [Netica Messages](#). Click on the  button

to open it up, or choose **Window** → **Messages**. In this window, Netica will

often put useful messages while it is operating and sometimes it will beep to

alert you to a new message.



**Tip:** It is a good idea to leave the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_Messages_window.htm');return
false;">Messages window open while you are learning Netica. You can copy
and paste information between this window and any text file, which may be
useful for future reference. It may also help by explaining why some
unexpected outcome has occurred.
```

**Getting Help:** If you have questions at any time while using Netica you can get help by choosing **Help** → **Context Help** or pressing **F1**. Depending on what you are working on, Netica will bring up a Help page of common subjects or will take you directly to a relevant screen within the help system. If you want more information on a certain term you can access the comprehensive index directly, by choosing **Help** → **Help**. If you are connected to the internet, you can find other kinds of help from the **Help** menu, such as: **Norsys Website**, **Online Netica Web Help** (the web version of onscreen help), **Net Library Website** (our large collection of examples nets), or **Email Norsys** (to send an email to our support team).

When you are finished with your Netica session, you can end it by choosing **File** → **Exit** from the menu. If you have created or changed some work without saving it, Netica will ask if you want to save it before terminating.

To proceed through the steps of the tour, use the navigation arrows above.

## **Activities** – PART OF [QUICK TOUR](#)

Once you click on an activity below, proceed through its pages with the navigation arrow above. Each of the activities of the [quick tour](#) may require knowledge of a previous activity.

- 1 [Probabilistic Inference](#)
- 2 [Net Construction](#)
- 3 [Net Transformation](#)
- 4 [Learning from Data](#)
- 5 [Decision-Making Nets](#)
- 6 [Equations and Time Expansions](#)

## Probabilistic Inference – PART OF [QUICK TOUR](#)

**Tutorial:** To begin, we will read a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes net stored on disk. Choose **File** → **Open** from the [menu](#), and when the standard file-opening dialog box appears, use it to open the file called “Chest Clinic” in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">Examples folder. A window will appear containing a Bayes net consisting of several connected = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node.htm');return false;">nodes. This very simple net for diagnosing patients arriving at a clinic is a classic example often used to introduce Bayes nets.

You can prepare the net for = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">inference by choosing **Network** → **Compile** from the menu, or by clicking the  [toolbar button](#) (if they are gray, the net has already been compiled). When the compilation is complete, the default display style changes so that nodes are displayed with bar graphs showing the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">beliefs for each of their states. You can enter a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">finding (also known as an “observation”, or “evidence”) for a node by clicking on the name of the finding to the left of the bar graph. You can also enter a finding by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking on the node and choosing **Enter Finding**.

When the finding is entered, the displays of all the nodes will be adjusted to account for it. For instance, putting an 'abnormal' finding for the 'XRays Result' node increases the belief that the patient has lung cancer from 5.5% to 48.9%, but then indicating that the patient has made a visit to Asia decreases that belief to 37.1%, because the abnormal XRay is partially = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_explaining\_away.htm');return false;">explained away by a greater chance of Tuberculosis (which the patient could catch in Asia).

If you choose "Unknown" from the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_findings\_menu.htm');return false;">findings menu, then any finding for that node will be retracted, and if you choose "Likelihood" you will be queried for the probabilities of an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_likelihood\_finding.htm');return false;">uncertain finding ("virtual evidence") for the node. You can also click directly on the name of the finding a second time to retract the finding.

Each time you = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">enter or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_retract\_finding.htm');return false;">retract a finding, the beliefs of all the nodes will immediately be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">updated to account for the new information. If you wish updating to only occur when you do a **Network** → **Update** command, you can turn off the auto-update feature by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">toggling

## Network → Automatic Updating.

If you want to observe or change the conditional probabilities of a node (which express its relation with its parent nodes), = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditional\_probability.htm');return false;">conditional probabilities of a node (which express its relation with its parent nodes), = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_relation.htm');return false;">relation with its parent nodes), = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the node by clicking once on it, and then choose **Table** → **View/Edit**, or click on the toolbar button with the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_relation\_symbol.htm');return false;">relation symbol: . A special window called the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_table\_dialog\_box.htm');return false;">table dialog box will open which displays the probabilities in a table. The left-hand side of the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_contingency\_table.htm');return false;">table contains a vertical list of parent configurations, and for each configuration the right-hand side has a few probabilities, expressed as percentages.

Each column of the right-hand side corresponds to a different = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state.htm');return false;">state of the node. So each number represents the conditional probability that the node takes on the state indicated by the column the number is in, given that the parents have the configuration indicated by the row the number is in. If the node is = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_deterministic_node.htm');return
false;">deterministic (e.g. the “TbOrCa” node), then the table dialog box will
display a = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_function_table.htm');return false;">function
table. This is the same as the conditional probability table, except that the
right-hand side simply has the state of the node which is the function value for
that parent configuration.
```

To work with examples of more complex Bayes nets, open the file called “Alarm” (also in the medical domain), or the file called “HailFinder” (weather prediction). You can compile them and do inference in the same way as you did with Chest Clinic. With this example, you can click directly on the name of the finding a second time to retract the finding or hold down the **SHIFT** key while clicking on the name of the finding to enter a = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_negative_finding.htm');return false;">negative
finding.
```

If you have entered a number of findings for a particular = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_case.htm');return false;">case and wish to save
them in their own file, choose Cases → Save Case As and enter the file name
in the dialog box which appears. If you want to work on a new case choose
Cases → Remove Findings, and then enter the new findings. To recover the
original case, choose Cases → Get Case and pick its name from the dialog
box which appears.
```

When you are done with a net window, you can get rid of it by clicking the  button in its title bar, or by making it the = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_active_window.htm');return false;">active
window and then choosing File → Close.
```



## Net Construction – PART OF [QUICK TOUR](#)

**Adding Nodes:** Choose **File** → **New** → **Network** from the [menu](#) to create a new window. To add a node (i.e. a “chance node” or “deterministic node”), move the cursor to the toolbar and click the  tool button. When you return the cursor to the new window, it will change to an ellipse, and when you click in the window, a node will be added at the cursor. The node will be selected when first added (i.e. displayed with negative colors), so if you just press the **ENTER** key, the node dialog box will appear which allows you to enter the name, states, etc. of the node.

Utility nodes (also known as “value nodes”) or decision nodes may be added in the same way as nature nodes, by using the  or  tool buttons respectively. [More Info](#)

**Adding Links:** Links may be added by clicking the  tool button, then clicking on the node you want the link to come from (i.e. the node you want the link to come from).

```
onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parent" node),  
and finally clicking on the node you want it to go to (i.e. the "= 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_child_node.htm');return false;">child" node).
```

Alternately, you can click down in the parent node, and while holding the mouse button down, drag the cursor to the child node, and then release it. If you double-click on a toolbar button then it will create a cursor that you can use to add several nodes or links (without it switching back to the pointer each time).

```
Another way to add nodes or links is to = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-click on  
the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_background.htm');return false;">background  
and choose Modify → New Node or right-click on a node and choose Links.  
More Info
```

**Selecting Nodes & Links:** To = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select a node or link, click once on it with the regular pointer cursor, and it will become hilited. A group of nodes may be selected by clicking down on the background within the window, and then moving the mouse with the button depressed (i.e. “dragging”), so that the selection rectangle is over them. You can add to or remove from a group of selected nodes by holding down the `CTRL` key while you select the new nodes. To delete nodes or links, select them and then press the `DELETE` key. [More Info](#)

**Undoing Operations:** After doing any operation, you can undo it with **Edit** → **Undo** (or pressing `CTRL+Z`). By repeating this you can undo operations previous to that one (at least 4 operations, and sometimes more if they don't take much memory). After undoing one or more operations you can redo them one-by-one with **Edit** → **Redo** (or pressing `CTRL+SHIFT+Z`). When exploring with Netica, it is very useful to be able to try a few operations, and then easily

undo them. [More Info](#)

**Moving Nodes:** A node can be moved by clicking down on it, dragging it to its new position, and then releasing the mouse button. To move a group of nodes, first select them, and then click down on one of the selected nodes and drag it to its new position. To change the shape of a link, first select it by clicking on it, then click down on it again and drag the cursor to the point where you want the link bend to be. [More Info](#)

**Disconnecting & Reconnecting Links:** A link can be disconnected by clicking on it to select it, then clicking down on the hilited square that forms at its non-arrow end, and dragging it away from the parent node (or by selecting the link and choosing **Modify** → **Disconnect Links**). To reconnect the link to a new node, drag the non-arrow end over the new parent and release the mouse button, or choose **Modify** → **Reconnect Links**. Disconnection/reconnection is useful to change the links of a net without losing the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">conditional probability tables of the nodes. Remember that to just delete a link you simply select it and then press the `DELETE` key. **NOTE:** If you delete a link and then re-add it, you will have lost the information in the previous CPT; thus if you want to retain the CPTs, use the disconnection command. [More Info](#)

**Cutting & Pasting Nodes:** You can cut and paste nodes and subnets within a window, or between windows. Try opening a net such as Car\_Diagnosis\_0, select part of it, press `CTRL+C` (to copy it to the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_clipboard.htm');return false;">clipboard), `CTRL+N` (to create a new net), click in the middle of the new window (to indicate where to put it), and then `CTRL+V` (to paste it into the new net). You can cut and paste other parts of the net (or other nets), add nodes, etc. and then connect up the disconnected links as described in the previous paragraph. In this way you can take knowledge from previous applications, perhaps save it in a net [fragment](#) library, and re-use it to construct a new net for a new application. [More Info](#)

**Saving Nets:** At any point you can choose **File** → **Save** from the menu to save the current version of the net to file, overwriting the previous one, or **File**

→ **Save As** to save it to a new file.

**Node Properties:** You can change the properties of a node by using a [node dialog box](#), which is obtained by double-clicking on the node,( or by right-clicking and choosing **Properties**). When you make changes in the dialog box, they won't actually be applied to the node until you click the "Apply" or "Okay" buttons. You can change a node's name or title in the obvious way by typing in the appropriate text field of the dialog box. To change the name of one of the node's states, choose the state using the down-arrow beside the "State:" label, and then type in the neighboring text field. You can add or delete states with the "New" and "Delete" buttons. Most people find it more convenient to enter or change state names using the text entry area at the bottom of the dialog box. **NOTE:** If you add or delete a state from the node dialog box, you will lose the information in state comments and in the CPT. Instead, add or delete a state using the right-click menu. [More Info](#)

The text entry area at the bottom of the dialog box can be used to enter or change several different things; you choose the thing to work on with the selector directly above it. For instance, to enter some text documenting the node, you choose "Description" with the selector, and then type in the text entry box. Some choices (such as "When Changed") can't be modified; they are for viewing only. [More Info](#)

**Node CPTs:** In order to change the probabilities of a node conditioned on the values of its parents (= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs), first open the node's [table dialog box](#) by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">selecting it, and then choosing **Table** → **View/Edit**. You can click on a number (or state name if it's a deterministic node) to change it. If no probabilities have yet been entered for this node, you can click on an empty cell to enter a number. The numbers must be entered as percentages. You can select some cells by clicking down in a cell that is not currently being edited, and dragging to enclose the cells before releasing the mouse button. You can select whole rows at a time by clicking and dragging slightly to the right of the double vertical line (this is how you must select deterministic cells, since clicking directly on them brings up the state menu). There are several items in the

**Table** menu that you can use to set the values of selected cells (such as **Uniform Probabilities**, **Fill Missing**, **Normalize** and **Randomize**). Any changes you make in the dialog box will not actually be transferred to the node until you press the **Apply** or **Okay** button. These two buttons do the same thing, but the Okay button also removes the dialog box. [More Info](#)

**Node Styles:** To change the [display style](#) of some nodes, first select them and then make a choice from the **Style** menu. For the Chest Clinic net the most useful displays are Belief-bar or Meter, although if you were creating a net for an end-user you may want to give a node like TbOrCa a style of Hidden. If you want to hide all the links, you can use **Style** → **Links** → **Hide Links**. The **Style** → **Font** choice may be used to change the font and size of the nodes. [More Info](#)

## Net Transformation – PART OF [QUICK TOUR](#)

[Node absorption](#) removes a node without affecting the overall global relationship of the rest of the nodes (i.e. the joint probability distribution).

This small demonstration of node absorption will show how it doesn't effect the beliefs of the rest of the net. Open "Car\_Diagnosis\_2" from the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">Examples folder. Compile it with **Network** → **Compile**, and notice that the belief is 25.4% that “Spark Quality” is “good”. If you enter a finding of “dim” for node “Headlights”, the belief changes to 1.47%. Now select nodes “Main Fuse”, “Battery Age”, “Voltage at Plug” and “Spark Plugs”, and then click the  toolbar button or choose **Modify** → **Absorb Nodes**. The selected nodes will be absorbed, links will be added and removed, and probability tables adjusted to maintain the global relationship. Now do **Network** → **Compile**, and observe the belief is 1.47% that “Spark Quality” is “good” as it was before the absorption. If you then do **Network** → **Remove Findings**, the belief changes to 25.4%, which is what it was in the old net before any findings were entered.

[Link reversal](#) changes the direction of a link without affecting the overall global relationship of the rest of the nodes (i.e. the joint probability distribution). Select a link and then click the  toolbar button or choose **Modify** → **Reverse Links**. Netica may have to add extra links in order to maintain the joint probability distribution (or it may be able to remove some links).

See also [Disconnecting and Reconnecting Links](#)

## Learning from Data – PART OF QUICK TOUR

**Tutorial:** Open the Bayes net called "Car\_Diagnosis\_0" from the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">Examples folder (note: do not use "Car\_Diagnosis\_2"). It is a simplified = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">example net containing nodes for a few variables of interest when diagnosing a car that is not running. The nodes are linked up in a causal manner, but the net does not contain any information other than the node names, their states, and how they are linked. If you = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click one of the nodes, and then choose **Table**, you will see in the table dialog box which appears, that the node has no = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs defined (the empty boxes in the right-hand panel). None of the nodes have any probabilities defined, so the net is not yet ready for = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">inference but in this step we will learn the probabilities from data.

To learn a probabilistic table for each of the nodes we will use a file of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case.htm');return false;">cases of cars previously arriving at a garage, called “Car Cases”, in the “Examples” folder. You may want to examine this file with a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_editor.htm');return false;">text editor. The row of headings across the top are the names of the nodes in the net, and each

possible value they can take are the state names of those nodes. “BatAge” is a continuous node, so its values are real numbers. The asterisks (\*) indicate data values which are not known (i.e. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_missing\_data.htm');return false;">missing data).

With the net window = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active, and no nodes = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_deselect\_nodes.htm');return false;">selected (otherwise the learning will only apply to the selected nodes), choose **Cases** → **Learn** → **Incorp Case File**. When you are queried for a file, choose Car Cases, and enter 1 for the degree. Netica will use the cases to learn probabilistic tables for each node of the net, with the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window displaying the fraction completed.

When it is finished, if you examine a few nodes with the table dialog box, you will see the learned probability distributions. You can click on the selector that says "% **Probability**", and choose "**Counts**" from the menu to see the number of occurrences of each possibility in the data file. From the Counts table, Netica generates the "Unnormalized" table by adding a small constant (usually 1) to each cell. Summing each row of the Unnormalized table results in the "Experience" table, which is used to normalize the unnormalized table, and produce the "Probabilities" table. Thus, Netica can learn the local probability tables in a very simple and effective way. If you have = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_latent\_node.htm');return false;">latent variables, or lots of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_missing\_data.htm');return false;">missing data, then Netica must use considerably more complex algorithms, such as EM or gradient descent to learn the tables.

Now that the local CPT tables have been learned, you can [compile](#) the net and do [inference](#), paste parts of it into a [decision net](#), [absorb nodes](#), etc.

If you want to learn the link structure of your net based on a case file, use the [structure learning](#) feature.

Netica can also be used to generate files of cases which follow the probability distribution of a Bayes net (i.e. “sample” from the Bayes net). These cases can be used as realistic examples of possible scenarios, or as synthetic data for learning experiments. Simply select those nodes of the net for which you want columns in the case file, and then do **Cases** → **Simulate Cases**. You will be queried for the number of cases to generate, the name of the file to create, and how much missing data you want (enter 0 for none, 1 for all missing).

More info on [Netica's Learning](#)

See also [Learning from an Excel File](#)

See also [Test Net with Cases](#)

## Decision-Making Nets – PART OF QUICK TOUR

With Netica, decision nets are the same as regular Bayes nets, except that two additional types of nodes are present, decision nodes and utility nodes. In a regular Bayes net, all the nodes are called nature nodes because they have only to do with modeling the nature or reality of the world, the likelihood of its being in any of its possible states. The concept of utility and the concept of decision are seen as outside of merely depicting reality, being more in the realm of goals, desires, and agendas. Netica compiles both a Bayes net and a decision net into a junction tree for efficiency.

**Tutorial:** Use **File** → **Open** to read the decision net (also known as an influence diagram) "Umbrella" from the Examples folder. By opening a table dialog box for each nature node (beige oval) you can observe its conditional probability

table, and by opening one for the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_utility\_node.htm');return false;">utility node (green hexagon), you can see its utility function. If you look at the table dialog box for the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_node.htm');return false;">decision node (blue rectangle), you will see that the node does not yet have any functional relation, which indicates that there is no decision function associated with the node. Our goal is to find a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_rule.htm');return false;">decision function which will maximize the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_expected\_value.htm');return false;">expected value of the utility node.

This is accomplished by choosing **Network** → **Compile**. If you then look at the table dialog box for the decision node, it will display the discovered decision function (you will have to click the “**Reset**” button if you didn’t freshly open the dialog box). The numbers in the “Decide\_Umbrella” give the expected utility of each choice. You can = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">enter findings for Forecast to see how they change.

If you want to try another decision net, open “Car\_Buyer\_Neapolitan”, which is described in [Neapolitan90](#), and is based on the classic “[Car Buyer](#)” influence diagram. It involves two sequential decisions about whether to do some tests and then whether to buy a certain used car. The optimal decisions turn out to be not to do any tests (D = none), but to buy the car (B = Buy when D = none).

[Here](#) is a more detailed version of this exercise.

## Equations and Time Expansions – PART OF [QUICK TOUR](#)

**Tutorial:** Open the [Dynamic Bayes Net](#) called "Bouncing" from the [Examples folder](#). It is designed to model a ball which bounces back and forth in a straight line between two parallel barriers without losing any energy at each bounce. Initially we have no knowledge of the position or velocity of the ball.

The Position and Velocity nodes stand for the position and velocity at each instant of time, and are connected together with brown [time delay links](#) (and could also have regular black links as well if desired). You can think of a time delay link as delivering to the arrow of the link the value that the tail had some period of time ago, with that period of time being the link's delay.

**Equations:** If you double-click on the Position node you can see the [equation](#) which defines the new position in terms of the old position, and the velocity. It involves the constant "dt", whose value may be changed (by right-clicking on the dt node and choosing **Enter Numeric Finding**) to provide a different amount of time between time slices.

By clicking down on the [selector](#) that originally says "Equation", you can choose "Discretization" to see how the node has been [discretized](#).

Later, you can edit this list to change the discretization.

This net has already had its probability tables (CPTs) built from the equation at each node. If it didn't have, the first step would be to generate probability tables for the current discretization by making sure no nodes are selected and

choosing **Table** → **Equation to Table** from the main menu. Enter 1000 when prompted for the number of samples per cell and press 'No' when asked about including sampling uncertainty.

**Time Expansion:** Next you generate a [time expanded](#) version of the net by doing **Network** → **Expand Time**. Enter 2 in the dialog box for the expansion time and 1 for the burn-in time. This will make a new window with a new net in it. You will probably want to resize this window to make it wider. You can use **Layout** → **Drawing Size** and make the entry blank. The new net is a regular Bayes net with each Position and Velocity node representing the position and velocity at a new point in time, with nodes to the right corresponding to later times.

Finally we can compile the Bayes net for probabilistic inference with **Network** → **Compile**. Experiment with setting the position or velocity (by clicking on the desired range) to indicate observations at certain times, and see how the beliefs for position and velocity at all other times change.

An interesting situation is when the ball is near the boundary, but you don't know which way it is going, since after a little while it will surely be moving away from the boundary (perhaps due to a bounce). Or if the positions at 2 times are known, can it infer the velocity? Or if just the velocities at two adjacent times are known, and one is the negative of the other, can it infer the position (because it must have bounced). You can also try = 4 &&  
`typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_enter_finding.htm');return false;">entering`  
some negative findings (that the ball isn't at some particular position or velocity) by holding down the SHIFT key when you click on the interval.

Remember that the numerical results will not be exact, due to the discretization and sampling error in converting the equations to probability tables. You may also want to try a finer discretization by changing the [Ranges](#) of the original unexpanded net.

When you build your own time delay nets, to make some links represent a time delay, select the links, then choose **Modify** → **Delay Links**. Or you can use the node dialog box, choosing "**Delay**" from the multi-purpose selector at the bottom.

See also: [dynamic Bayes nets](#), and the [Equations](#) chapter.

# Introductory References for Bayes Nets and Decision Nets

**Overview:** For an overview of Bayes nets and decision nets, there are two special journal issues that are suitable. The first is the winter 1991 edition of AI Magazine, which contains both “Bayesian Networks Without Tears” ([Charniak91](#)) and “Decision Analysis and Expert Systems” ([Henrion&BH91](#)). The other is volume 38 of the *Communications of the ACM*, which is devoted to Bayes nets and decision nets ([Heckerman&MW95](#)), and has introductory material and descriptions of real applications.

**Fundamentals:** In 1988 Judea Pearl wrote *Probabilistic Reasoning in Intelligent Systems* ([Pearl88](#)), which was the most influential and widely cited book in the formative development of Bayes nets. It is an excellent foundational book, but it doesn't contain the latest developments, and has a very mathematical and theoretical orientation. *Bayesian Networks and Decision Graphs* ([Jensen01](#)) has somewhat more recent results, although the earlier [Jensen96](#) may be even more useful to a beginner who is just interested in understanding and applying Bayes nets in a basic way, if it can be obtained.

The "bible" on Bayes nets right now is *Probabilistic Graphical Models* ([Koller&F09](#)). It is very comprehensive and written at a high academic level by two of the world's leading researchers on Bayes nets. In the same vein, but not as comprehensive (although covering some topics, such as inference, with greater depth) is *Modeling and Reasoning with Bayesian Networks* ([Darwiche09](#)).

**Applied:** The best book on the theory of how to apply Bayes nets to the real world is *Bayesian Networks and Influence Diagrams* ([Kjaerulff&Madsen08](#)). *Bayesian Artificial Intelligence* ([Korb&Nicholson04](#)) also contains useful material to construct models. A sampling of real-world applications can be found in *Bayesian Networks: A Practical Guide to Applications* ([Pourret&Naim&Marcot08](#)).

For the usage of Bayes nets in particular fields, see:

- *Probabilistic Methods for Bioinformatics* ([Neapolitan09](#))
- *Probabilistic Methods for Financial and Marketing Informatics*

([Neapolitan07](#))

- Ecology: Special Issue of *Canadian Journal of Forest Research* ([NRC06](#))
- *Planning Improvements in Natural Resource Management* ([Cain01](#))
- *Operational Risk: Measurement and Modelling* ([King01](#))

[Russell&N09](#) is an excellent introductory textbook for artificial intelligence that does a good job of describing Bayes nets, decision nets, dynamic decision nets, policy iteration, etc. within the overall context of intelligent agents and AI software.

**Causality:** A very accessible and easy (but worthwhile) read on causal models is *Causal Models: How People Think About the World and Its Alternatives* ([Sloman05](#)). If you are building causal Bayes nets, and haven't had a lot of experience with causal modeling, you should definitely read that book. For a more academic and advanced study on causality, read *Causality: Models, Reasoning, and Inference* ([Pearl09](#)). It is by the world's leading researcher on causality, and although it is at a very advanced level for the subject, you can selectively read it to get the most important ideas without much difficulty. The excellent epilogue can be read with no background knowledge required; read it first. For causality as applied to psychology, see *The Mind's Arrows: Bayes Nets and Graphical Causal Models in Psychology* ([Glymour01](#)).

**Research:** Although much of the literature on Bayes nets is published within the artificial intelligence (AI) community, it is really the combined work of the statistics / probability, decision analysis, operations research, and AI communities (and increasingly other communities that are applying Bayes nets, such as resource management, ecology, finance, medicine, etc.). Many of the researchers who developed the theory of Bayes nets and decision nets attended the annual "Uncertainty in Artificial Intelligence" conference from 1985 to present, and so advanced material can be found in the conference proceedings (available from the "Conference Proceedings" section of the [AUAI site](#)).

## Design Philosophy

Netica was developed using a unique software development process. Many of our clients are surprised at the small size of Netica, considering its advanced capabilities, ease of use, and reliability compared to other Bayes net software. Internally, it has very little redundancy (most modern Windows programs have a large percentage of redundant or unused code), and an elegant internal architecture designed to give maximum power and usability in the simplest manner.

Our development process emphasizes constant iterative improvement. Every time a client reports a bug or asks a question, we have a process in place to fix that bug, or make sure the question is answered by the onscreen help.

Whenever a client expresses frustration or gives us a wish-list item, we make a plan on how to address the issue, and either make an immediate change, or add it to our short-range or long-range development schedule. If you ever have any problem with Netica, frustration with how it works, or desire for a new feature, please = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_email.htm');return false;">let us know; your input will be used to make Netica a better product.

We have zero-tolerance for bugs in = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Netica\_API.htm');return false;">Netica API. It gets thoroughly tested with our extensive in-house testing and by our large client base over many years, and it has no known bugs. The fact that = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Netica\_Application.htm');return false;">Netica Application shares much of its code with Netica API means that Netica Application has a very solid, robust core.

We keep abreast of the latest developments in Bayes net theory and technology, and are constantly adding features based on new research results from our research and that published in the general literature. You will find in Netica many exciting new features not available in any other Bayes net software.

One of our design goals is to make Netica very versatile; it is a tool that in the right hands can be used for many different tasks. Clients often show us imaginative solutions they've created by combining various Netica features, and many have told us that it was "fun" to use Netica because of the power it gave them.

## Bayes Nets and Probabilistic Inference

A Bayes net (also known as a belief network or probabilistic causal network) captures believed relations (which may be uncertain, stochastic, or imprecise) between a set of variables, which are relevant to some problem. They might be relevant because they will be observable, because their value is needed to take some action or report some result, or because they are intermediate or internal variables that help express the relationships between the rest of the variables.

**Example:** A [classic example](#) of the use of Bayes nets is in the medical domain. Here each new patient typically corresponds to a new `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_case.htm');return false;">case`, and the problem is to diagnose the patient (i.e. find beliefs for the unmeasurable disease variables), predict what is going to happen to the patient, or find an optimal prescription, given the values of observable variables (symptoms). A doctor may be the expert used to define the structure of the net, and provide the initial relations between variables (often in the form of `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">conditional probabilities`), based on his medical training and experience with previous cases. Then the net probabilities may be fine-tuned by using statistics from previous cases, and from new cases as they arrive.

**Bayes Net Construction:** When the Bayes net is constructed, one `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_node.htm');return false;">node` is used for each *scalar variable*, which may be discrete, continuous, or propositional (true/false). Because of this, the words “node” and “variable” are used interchangeably throughout this document, but “variable” usually refers to the real world or the original problem, while “node” usually refers to its representation within the Bayes net.

The nodes are then connected up with directed `= 4 &&`

```

typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_link.htm');return false;">links. If there is a link
from node A to node B, then node A is sometimes called the = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parent, and
node B the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_child_node.htm');return false;">child (of
course, B could be the parent of another node). Usually a link from node A to
node B indicates that A causes B, that A partially causes or predisposes B, that
B is an imperfect observation of A, that A and B are functionally related, or
that A and B are statistically correlated. The precise definition of a link is
based on conditional independence, and is explained in detail in a reference
like Neapolitan90 or Pearl88. However, most people seem to intuitively grasp
the notion of links, and use them effectively without concentrating on the
precise definition.

```

```

Finally, = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_CPT.htm');return false;">probabilistic tables are
provided for each node, which express the probabilities of that node taking on
each of its values, conditioned on the values of its parent nodes. Some nodes
may have a deterministic relation, which means that the value of the node is
given as a direct function of the parent node values.

```

**Probabilistic Inference:** After the Bayes net is constructed, it may be applied to a particular case. For each variable you know the value of, you enter that value into its node as a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">finding (also known as “evidence”). Then Netica does probabilistic inference to find = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">beliefs for all the other variables. Suppose one of the nodes corresponds to the variable “Temperature”, and it can take on the values cold, medium and hot. Then an

example belief for temperature could be: [cold - 0.1, medium - 0.6, hot - 0.3], indicating the subjective probabilities that the temperature is cold, medium or hot.

Depending on the structure of the net, and which nodes receive findings or display beliefs, Netica may do diagnosis, prediction, classification, logic, arithmetic calculation, or any combination of these, to complete the probabilistic inference. The final beliefs are sometimes called *posterior probabilities* (with *prior probabilities* being the probabilities before any findings were entered). Probabilistic inference done using a Bayes net is called = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">belief updating.

If you want to apply the net to a different case, then all the findings can be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_retract\_finding.htm');return false;">retracted, new findings = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered, and belief updating repeated to find new beliefs for all the nodes.

Probabilistic inference only results in a set of beliefs at each node; it does not change the net (knowledge base) at all. If the findings that have been entered are a true example that might give some indication of cases that will be seen in the future, you may think that they should change the knowledge base a little bit as well, so that next time it is used, its conditional probabilities more accurately reflect the real world. To achieve this you would also do = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probability\_revision.htm');return false;">probability revision.

See also [Learning from Cases](#)

## Netica's Probabilistic Inference

Netica uses the fastest known algorithm for exact general [probabilistic inference](#) in a compiled [Bayes net](#), which is message passing in a *junction tree* (or "join tree") of cliques. The algorithms used are described in [Spiegelhalter&DLC93](#), [Jensen96](#) and [Neapolitan90](#).

In this process, Netica first = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_compile\_net.htm');return false;">compiles the Bayes net into a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_junction\_tree.htm');return false;">junction tree.

The junction tree is implemented as a complex set of data structures connected up with the original Bayes net, but invisible to you as a user of Netica. You enter = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings for one or more nodes of the original Bayes net, and then when you want to know the resultant beliefs for some of the other nodes, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_deterministic Updating.htm');return false;">deterministic updating and then belief updating is done by a message-passing algorithm operating on the underlying junction tree.

It determines the resultant beliefs for each of the nodes of the original Bayes net, which it [displays](#) as a bar graph, or a needle meter, at each node. You may then [enter](#) some more findings (to be added to the first), or remove some findings, and when you request the resultant beliefs, belief updating will be performed again to take the new findings into account.

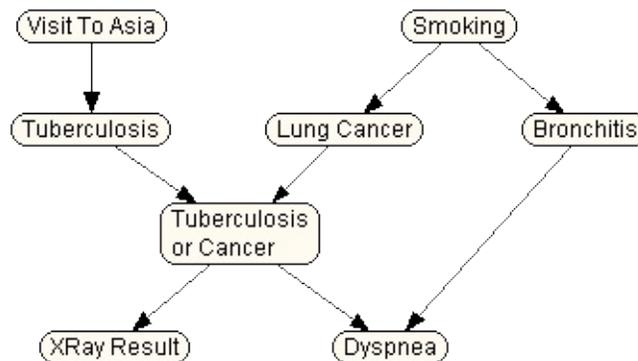
Netica can also do probabilistic inference by [link reversals](#) and [node absorption](#), but those are mainly meant for transforming a net and doing model exploration, and are usually not as fast or as convenient for general probabilistic inference as the compiled junction trees described above.

## Example Bayes Net

Let's look at an example of using Netica to do [probabilistic inference](#). In this example we will read in a simple net from a file, compile it into a form suitable for fast inference, enter some findings, and see how the beliefs of various nodes change with each finding.

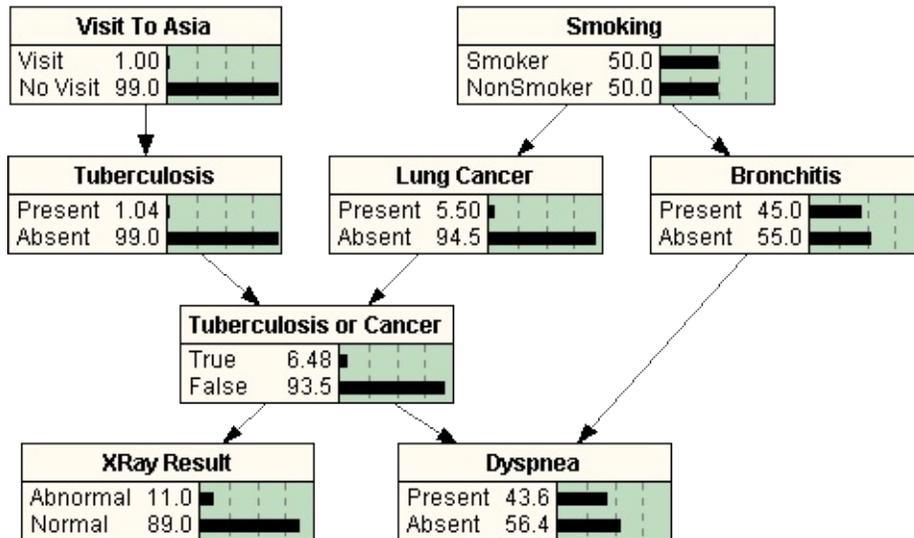
The net we will use is shown below. It is a toy medical diagnosis example from [Lauritzen&S88](#) that has historically been used for demonstration purposes. To a certain degree, the links of the net correspond to causation.

The two top nodes are “predispositions” which influence the likelihood of the diseases in the row below them. At the bottom are symptoms for the diseases.



You can read it into Netica by choosing **File** → **Open** from the menu (menu selections are indicated in bold, with arrows indicating choices or submenus, so the previous means choose “Open...” from the “File” menu). When the standard file opening dialog box appears, use it to open the file called “Chest Clinic”. This file is included with the Netica distribution in the “Examples” folder.

If you [compile](#) “Chest Clinic” by choosing **Network** → **Compile**, you will see something like the following:



The default node style has changed to [belief-bar](#), so the display looks different. Also, the appropriate data structures for fast inference have been built internally. If you [enter a finding](#) of 'abnormal' for node "X-Ray Result", by clicking on the word 'abnormal', the beliefs will be automatically updated, so that the belief that the patient has lung cancer goes from 5.5% to 48.9%. Then indicating that the patient has made a visit to Asia decreases that belief to 37.1%, because the abnormal XRay is partially [explained away](#) by a greater chance of Tuberculosis (which the patient could catch in Asia).

## Compiling a Bayes Net

To compile the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active Bayes net, choose **Network** → **Compile**, or click the  toolbar button. If they are dimmed then compiling has already been done. A “= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_junction\_tree.htm');return false;">junction tree” for fast inference will be built internally. The default display style for the nodes will be changed to [belief-bar](#), so if the nodes haven't had their [display styles](#) set, then they will be displayed in belief-bar style.

For small nets, the **Compile** command will seem to perform instantly. For large nets it may take a few seconds, in which case the progress will be displayed in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window.

**Auto-Compile:** If you save a compiled = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_auto\_update.htm');return false;">auto-updating net to file, then the next time you open it, Netica will automatically compile it and perform updating. If the net is not compiled when you save it, then you must compile it as described above when you next open it. If you wish to read in an auto-compiling net without performing any compiling, hold down the **SHIFT** key when you open it.

**Missing Tables:** It is possible to compile and update nets whose CPTs or utility tables are absent or incomplete. Netica takes missing entries as uniform probabilities or zero utilities. If there are missing entries or tables, then while compiling Netica will beep and put a message (#2760) in the Messages window.

**Optimized:** Netica also has a **Network** → **Compile Optimize** command, which spends a long time working out a very efficient structure for the internal

**junction tree**. As it is working, it displays in the Messages window the memory requirements and projected inference time to do a belief updating with the best structure it has developed so far. Press the **CTRL** key and hold down the left mouse button at the same time when you want Netica to stop and use that structure.

The memory requirements it displays are very accurate, but when considering the inference time keep in mind that it is based on doing the inference using the same computer as the one doing the compiling, it assumes the computer has enough RAM memory installed that it doesn't need to use virtual memory, and that it has a large share of the processor time under multitasking. It also doesn't include the screen redraw time, which depends on how large the Bayes net window is, and on the display style of the nodes.

If you save a Bayes net after an optimized compile, then the next time you open it the quick compile command will give it the optimized structure quickly (as long as you don't change the net nodes or links in between).

**Technical:** For those interested in a technical explanation, the quick compile does a minimum-weight search for a good elimination order, and the optimized compile searches for the best elimination order using a specialized algorithm which is a combination of minimum-weight search and stochastic search (with some facets of multi-start methods, simulated annealing and genetic algorithms).

## Entering and Retracting Findings

**Purpose:** Usually you enter findings (sometimes called “evidence”) to temporarily apply a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes net to a particular = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case.htm');return false;">case for = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">probabilistic inference.

**Entering/Right-click:** To enter a finding for a node, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click on it and choose **Enter Finding**. Each possible state of the node will be listed in alphabetical order; choose the finding from it. Menu items appear as state title (or by state name if no titles exist).

After the finding has been entered the node will be darkened (or its color otherwise changed as defined by node-sets). It does not matter in what order you enter a set of findings.

**Entering/Left-click:** When the net is = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_compile\_net.htm');return false;">compiled and the node is displayed in belief-bar or meter style, left-click directly on the name of the finding you wish to enter. If the node is displayed in belief-bar style then a solid outlined bar is drawn for the state matching the finding.

When you enter new findings, and belief updating is not yet done, only those nodes effected will have their beliefs indicated invalid.

**Retracting:** If you wish to retract (i.e. remove) a finding that you have entered for some node, choose **Unknown** from its findings menu, or click

again directly on the name of the finding of a belief-bar node. Entering a finding and retracting it is equivalent to never having entered it, even if there were other findings entered in between.

Another way of retracting the findings for some nodes is to = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select them and then choose **Cases** → **Remove Findings**, or click the toolbar button with a red cross over the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case\_symbol.htm');return false;">case symbol: 

If you wish to retract all the findings entered for all the nodes in the net, make sure all or none of them are = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_deselect\_nodes.htm');return false;">selected, and then choose **Cases** → **Remove Findings**. This is useful if you are finished with one case and wish to move on to another.

**Multiple Nodes:** You can enter the same finding for a whole set of nodes at once. First select them, then right-click on one of the selected nodes, and proceed as above. [More Info](#)

**Uncertain:** If you want to enter a finding that a node is not in a particular state, or you want to enter an uncertain finding, you would enter it as a [negative or likelihood finding](#).

**Consistency:** Netica can [check the consistency](#) of the findings you enter.

**Sets of Findings:** If you have multiple findings to repeatedly enter into a Bayes net, you can put them in a [case file](#).

**Decision Nodes:** Entering a finding into a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_node.htm');return false;">decision node does not really correspond to discovering information or making an observation, but rather to committing to a decision choice. Nevertheless, it is still often loosely referred to as "entering a finding".



## Belief Updating and AutoUpdating

When probabilistic inference is done using a Bayes net, it is called *belief updating*. New = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">beliefs (posterior probabilities) are found for each of the nodes to reflect the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings currently entered in the net.

**Auto-Updating:** Normally Netica does belief updating quickly and automatically after compiling, and every time a finding is = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered.

However, it can be slow on large nets, so you may want to enter several findings at once before updating (you also may want to use the optimizing compiler). In that case, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">toggle off **Network** → **Automatic Updating**, enter the findings, and each time you want to update the beliefs choose **Network** → **Update**, or click the  toolbar button (they will be dimmed if belief updating is not currently required, or if the net is not compiled).

**Nothing Happens:** If you enter findings into a net, and no belief updating occurs, perhaps it is because = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_auto\_update.htm');return false;">auto-updating is off (the state of auto-updating is saved with each net file) or the net is not compiled.

**Dimmed Belief Display:** If auto-updating is turned off, and you enter a new finding, or if no findings are entered but belief updating was never done, then the belief-bars or meters will not display valid results, and so they are drawn dimmed or in gray rather than in solid black. The reason they are not removed

completely is that you may want to see what the beliefs were at the point of the last belief updating.

## Negative and Likelihood Findings

Negative and likelihood findings are = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings with some uncertainty attached.

**Negative:** A *positive finding* is knowledge that some variable definitely has a particular value. However, you may know that the value of a node is **not** some = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state.htm');return false;">state without knowing what its value is.

This is called a *negative finding*. For example, say the node “Temperature” can take on the values cold, medium, and hot. You may obtain information that the temperature is not hot, although it doesn’t distinguish between medium and cold at all. That is a single negative finding.

If you receive another negative finding that the temperature is not medium, then you can conclude that it is cold. So several negative findings can be equivalent to one positive finding.

**Likelihood:** A third type of finding is a *likelihood finding* (also known as “virtual evidence”). In this case you receive uncertain information about the value of some = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete\_node.htm');return false;">discrete node. It could be from an imperfect sensor, or from a friend who is not always right.

Say you have a thermo sensor to measure “Temperature”, which is designed so that when the temperature is hot it is supposed to turn on. In actual practice you find that when the temperature is cold the sensor never goes on, when the temperature is medium it goes on 10% of time, and when it is hot it always goes on. If at a certain time you observe the sensor on, and want to = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

`onclick="BSSCPopup('X_PU_enter_finding.htm');return false;">enter this finding into the Temperature node, then you do so as a likelihood finding.`

A likelihood finding consists of one probability for each state of the node, which is the probability that the actual observation would be made if the node were in that state. For our temperature example, the likelihood finding would be (0, 0.1, 1). A common mistake is to think that the likelihood is the probability of the state given the observation made (in which case the numbers would have to add to one), but it is the other way around.

**Entering:** You can enter a negative finding for a belief-bar node of a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_compile_net.htm');return false;">compiled net`, by holding down the `SHIFT` key while you click on the name of the finding you know its not. You can enter a likelihood finding for a node by choosing "Likelihood" from its `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_findings_menu.htm');return false;">findings menu`. You will then be queried for the likelihood numbers.

You can enter a negative finding by entering a likelihood finding consisting of all 1s, except a single 0 for the state that you know it's not. By having more than a single 0 you can enter several negative findings at the same time.

**Accumulating:** Whenever you enter a positive finding for a node, all the old findings for that node are automatically `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_retract_finding.htm');return false;">retracted` first. However, if you enter more than one likelihood finding for a node, you will be queried if you first want the previous finding(s) to be removed, or if you want them to accumulate. By letting them accumulate you can enter several independent pieces of evidence (e.g. imperfect observations) for the same node. If they are not `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_conditionally_independent.htm');return false;">conditionally independent` given the observed node, and it is too inaccurate to approximate them as independent, then they should be entered by

```
adding = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_child_node.htm');return false;">
```

child nodes to the observed node (one for each observation), connecting them together to show the dependency, and then entering positive findings for the child nodes.

**Locating:** To find all the nodes with negative or likelihood findings, use **Edit** → **Select Nodes** → **Likelihood Findings**.

# Consistency of Findings

There are three ways that findings can be inconsistent:

**1. Several findings for different nodes** can be inconsistent with each other.

This is the most common form of inconsistency. When Netica alerts you that findings are inconsistent, then it means that according to this Bayes net, it is impossible to observe the set of findings you have entered. If you know that those findings are possible, then the fault lies in the net's conditional probabilities (see below).

**2. Several findings for the same node** can be inconsistent with each other.

This only applies to findings entered in accumulation mode, because otherwise each new finding will retract the previous one for that node. Usually only negative or likelihood findings are entered in accumulation mode, and the only way they can be inconsistent is to have at least one negative finding (or zero likelihood) for every state of the node.

**3. A single finding** can be inconsistent with the net itself. This is rare.

Basically the net has a zero prior probability for that finding, which means it is a finding which should never occur. Usually this indicates the net was not designed to handle cases of this type.

**When Detected:** Netica will always detect an inconsistency of type 2, and usually of type 3, as soon as you try to enter it.

onclick="BSSCPopup('X\_PU\_belief\_updating.htm');return false;">belief updating all inconsistencies will be detected, and Netica will report them (either to the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window or with a dialog box). So if the beliefs are kept current by belief updating after each finding is entered (which is one reason to use = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_auto\_update.htm');return false;">auto-updating), then you will be alerted as soon as you try to enter an inconsistent finding, otherwise you won't be alerted until the next belief updating is in progress.

**Recovery:** If Netica reports an inconsistent finding, simply = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_retract\_finding.htm');return false;">remove it and continue. Nothing within Netica will be left in a problematic state.

**Before Entering:** You can always tell if a finding will be inconsistent before you enter it by looking at the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">belief for that state (do belief updating first if necessary). If the belief is zero, the finding will be inconsistent, otherwise it will be consistent.

**Incorrect Net:** If you have problems with consistency, re-examine whether = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_relation.htm');return false;">conditional probabilities you have entered as 0% should really indicate *absolutely impossible*. If there is some very small probability that they aren't impossible, then enter that very small probability instead. Equivalently, perhaps some nodes that are = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_deterministic\_node.htm');return false;">deterministic should really be probabilistic. Also consider the possibility that the net is correct, but that the findings should be high = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_likelihood_finding.htm');return
false;">likelihood findings, rather than = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_positive_finding.htm');return false;">positive
findings.
```

**Inconsistent Net:** A net can never be inconsistent with itself. No matter what its structure, or its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">conditional probability tables, it always represents some joint probability distribution. Only findings can be inconsistent. However, a net may be incorrectly designed, as described above.

## Most Probable Explanation

Given findings for some nodes, you may want to find the most probable configuration of values for the rest of the nodes. This can be thought of as providing a plausible explanation for the observed findings, and is called the *most probable explanation* or MPE (it is a special case of the maximum a-posteriori probability, or MAP).

You cannot determine the MPE simply by taking for each node the state with highest belief after regular [belief updating](#) (which finds the marginal posterior probability for each node). For example, in a lottery for which we know there is one winner, it might be most probable for each individual person to lose, but then the overall configuration would have everybody losing, which contradicts the one winner evidence. Finding the MPE would select one representative person to win (perhaps who bought the most tickets), and the rest would be losers.

Netica can be used to find the MPE. To indicate that you wish to do MPE updating for the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active net, instead of regular belief updating, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">toggle **Network** → **Most Probable Expl.** You will have to recompile after turning MPE on or off, as is indicated by all the belief-bars turning gray. You then = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">enter findings in the usual way and it does MPE updating immediately afterwards if **Network** → **Automatic Updating** is on, or otherwise just after a **Network** → **Update** command.

After updating, each node will have a [belief-bar](#) at the 100% level, and usually some bars at lower levels. You can read off the most probable configuration by taking for each node the state with the bar at the 100% level. The shorter bars indicate the relative probabilities of the other states given that the other nodes are in the most probable configuration (scaled by the same factor used to

bring the longest bar to 100%).

Of course the bars don't add to 100%, so if you are ever using Netica and are confused that every node has a 100% bar and there are also some other nonzero bars, it is because MPE updating is on. To avoid the possibility of accidentally doing MPE updating when regular updating is expected, all nets start with the MPE feature turned off, even if MPE was on when the net was last saved.

**Multiple 100% Bars:** Sometimes two or more states of the same node have bars that are at the 100% level. This indicates that there is more than one configuration with the highest probability (i.e., the configurations have equal probability). If more than one node has this, then you should choose one of the states and enter artificial evidence that the node is in that state, to see how it changes the multiple 100% bars of other nodes. By trying each of the possibilities you can map out all the configurations that are at the highest probability level.

**Problems:** You must be careful using the MPE. Generally, it is not as good as posterior probability (i.e. regular updating) for decision problems, or providing prediction or diagnosis probabilities. Its results can change with the introduction of irrelevant variables. And, it can be deceptive in situations where even the most probable explanation is extremely unlikely.

**When to Use:** The MPE is useful for explaining and aiding understanding. If an agent finds the results of regular belief updating questionable, and asks you to provide a scenario for which the beliefs are upheld, you can use the MPE to find that scenario. People sometimes find a completely specified scenario easier to understand. And sometimes you can gain insights by putting the Bayes net in MPE mode, entering the evidence, observing the most probable configuration, and then experimenting with adding extra "evidence" to explore a set of probable configurations close to the most probable one, while seeing how changing one node effects the others.

# Creating Bayes Nets and Decision Nets

The following are actions you can take to create a new Bayes net or decision net, or to modify an existing one:

- [Opening a Window](#)
- [Adding Nodes](#)
- [Adding Links](#)
- [Undoing and Redoing](#)
- [Selecting Nodes and Links](#)
- [Moving Nodes and AutoGrid](#)
- [Reshaping a Link](#)
- [Saving a Net to File](#)
- [Deleting Nodes and Links](#)
- [Cut, Copy, Paste and Duplicate Nodes](#)
- [Right-Clicking](#)
- [Changing Node Properties](#)
- [Placing Text on Net Diagrams](#)
- [Documentation Window](#)
- [Drawing Size and Scrolling](#)
- [Zooming In and Out](#)
- [Search/Find](#)
- [Related Nodes and Links](#)
- [One Operation, Multiple Nodes](#)
- [Entering Node Probability Tables](#)

## Opening a Window

The first step in creating a new net is to bring up a window for it. Use **File** → **New** → **Network** or click the  toolbar button. A window will appear in which you can build the net. The title of the window will be set when you [save](#) it to file.

**Existing Nets:** You will often find it easier to build a net by modifying an existing one. To do so, read in the existing one with **File** → **Open** (or click the  toolbar button), save it under the name of the new net with **File** → **Save As**, and then make the desired modifications, occasionally saving it with **File** → **Save**, or by pressing . Alternatively, you can read in a recently used net by choosing **File** → **Recent Files**.

**Multiple:** You can open as many net and text windows as you wish. As with any Microsoft Windows application, the current operation always applies to the *active* window, which is the one in front with the non-dim title bar. You can make a window active by clicking on an exposed part of it, or by choosing its title from the **Window** menu.

To obtain multiple versions of the same net, right-click on the net = 4 &&  
`typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_background.htm');return false;">background`  
and choose **Duplicate in New Window**, or choose **Window** → **Duplicate Net**.

**Arranging:** From the **Window** menu you can also choose **Tile** or **Cascade**, which will arrange all the windows accordingly. If these are functions you use frequently, you can [customize](#) the toolbar by adding their buttons: , and , or simply use their [shortcut keys](#).

**Closing:** When you are done with a window, you can close it in one of three ways:

1. by clicking the  button in its title bar,
2. if you have made unsaved changes to any window, Netica will ask if you want to save them first. Make it the active window and then choose **File** → **Close All**, or

3. by pressing `CTRL+F4`, `CTRL+Q` or `CTRL+W`.

If you find there are too many windows open at once, you can choose **Window**  
→ **Close All** and begin anew.

## Adding Nodes

The toolbar buttons , , and  are used for adding = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_nature_node.htm');return false;">nature nodes`  
(i.e. “chance” or “deterministic” nodes), = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_decision_node.htm');return false;">decision`  
nodes, or = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_utility_node.htm');return false;">utility nodes`  
(also known as “value” nodes) respectively. When you click the appropriate button, and then move the cursor over the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_active_window.htm');return false;">active net`  
window, it will change to the shape of the node to be added. Next click on the place in the window where you want the node to appear, and it will be added there, with the cursor returning to normal.

**Alternate Adding Options:** Besides the toolbar, you can add new nodes to your net in the following ways:

1. = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">Right-click`  
on the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_background.htm');return false;">background`  
and choose **Modify** → **New Node** and choose the desired node type from the list
2. Using the Overhead menu, choose **Modify** → **Add** to add a **Nature**, **Decision**, **Utility**, or [constant](#) node, or to add a new [link](#).
3. Use a [shortcut](#) equivalent.

**Multiple:** To add a series of nodes, double-click on the toolbar button for the

type of nodes you wish to add, or double-press **F9** (i.e. press it twice in quick succession). The cursor will change to the shape of the node as before, but now it will be darkened indicating that a number of nodes may be added in a series.

To exit this node-adding mode, click the node-adding button again, or press the node-adding **Fx** key again, or click the  button, or press the **Pause/Break** button (or click on a different node-adding button or press a different node-adding key).

**Node Types:** Often the most useful way to add nodes is by = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking,  
because then you can choose what type of node to add, instead of setting it later (through the [dialog box](#)).

Nature - Discrete [DEFINITION](#)

Nature - Boolean [DEFINITION](#)

Nature - Numeric Continuous [CONTINUOUS VS. DISCRETE](#)

Nature - Numeric Discrete

Decision [DEFINITION](#)

Decision - Boolean

Utility [DEFINITION](#)

Constant [DEFINITION](#)

**Properties:** When a node is first added it will be = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">selected.

Whenever a node is selected you can press the **ENTER** key to bring up a [dialog box](#) for setting node properties, and when the dialog box first comes up, its field for setting the node's name is selected for changing. This makes it possible to quickly add a node with a certain name. Just click the appropriately shaped cursor where you want the node to be, press the **ENTER** key, type the name, and then press the **ENTER** key again. There is no need to think about the dialog box or wait for it to be fully drawn.

**Node Palette:** You can use the *Node Palette* to store nodes that you use

frequently. To store a node, [copy it](#) and choose **Window** → **Node Palette**, which will bring up a window to paste the node. Whenever you use Netica, you can open the node palette to retrieve your stored nodes.

**Node Groups:** To help with visual organization, you can group nodes together, to form [node-sets](#).

**Deleting Nodes:** If you want to delete a node from your net, you can simply highlight it and press the **DELETE** or the **BACKSPACE** buttons. However, if the node was linked to other nodes, there are some things to consider before deleting it entirely, as its deletion could affect your CPTs. [More Info](#)

## Adding Links

To add a link from one node to another, first click the  toolbar button, choose **Modify** → **Add** → **Link**, or press the **F10** key. When you next move the cursor to the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active net window, it will change to a link shape indicating that it is ready to add a link.

Click on the node that you wish to be the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parent, and then click on the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_child\_node.htm');return false;">child node.

Alternately, you can click down on the parent, and while holding the mouse button down, move to the child and then release it. The link will appear. If you want, you can then [change the shape](#) of the link.

**Adding-Mode:** To add a series of links, double-click the  button, or double-press the **F10** button (i.e. press it twice in quick succession). The cursor will change to a link shape as before, but now it will stay that way as you add multiple links. To exit this link-adding mode, click the  button again, or press the **F10** key again, or click the  button (or click on a node-adding tool button or press a node-adding key).

**Several at Once:** To add links from one node to many, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_link.htm');return false;">select all the nodes you want the links to go to, then draw a link from the desired parent to one of them, and all the links will be created.

To add links from many nodes to one, select all the nodes you want the links to come from, then draw a link from one of them to the desired child, and all the links will be created.

**Deleting:** To delete links, simply select all the links you wish to delete, and then press the **DELETE** key or choose **Edit** → **Delete**. Alternately, you can = 4

```
&& typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-click on a
link and choose Delete.
```

**Overlapping and Cycles:** It is possible to add more than one link between the same two nodes, or to create = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_directed\_cycle.htm');return false;">directed cycles. For most Bayes net work, you don't want to do either of these, and so Netica will beep and put a warning in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window (it doesn't stop you, since these may be valid operations when creating a [DBN](#) or [net fragment](#), but if you try to = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_compile\_net.htm');return false;">compile the net it will give you an error message). To remove an overlapping link, just click on it and press the **DELETE** key; you might not notice it disappearing, since after it is gone, you will still see the link under it.

**Right-clicking Options:** For an unselected node, you can right-click on it and choose to **Add Link To** or **From** any of the other nodes in the net.

Once you select a node or group of nodes, right-clicking will allow you to add links **To** or **From** any other node in the net. If a group of nodes is selected, you can also right-click on an unselected node and choose to **Links** → **Add Link To** or **From Selected Nodes**. [More Info](#)

## Undoing and Redoing

While **building** or changing the net, you can undo the last operation by choosing **Edit** → **Undo**, or pressing `CTRL+Z`, or clicking , or = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-clicking  
on the net = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_background.htm');return false;">background
```

and choosing **Modify** → **Undo**.

**Multiple:** To undo operations previous to that one, repeatedly invoke **Edit** → **Undo** (or `CTRL+Z`, or )

**Redo:** After a series of undos, you can “redo” one or more of them by repeatedly choosing **Edit** → **Redo**, or pressing `CTRL+SHIFT+Z`, or clicking , or right-clicking the net background and choosing **Modify** → **Redo**.

**How Many:** Netica will save a great many of the previous steps for undoing, but this will be reduced if they are steps which take large amounts of memory.

**Dimmed:** If you are undoing operations and the **Edit** → **Undo** menu item or toolbar button turns dim, it means you have undone all the operations that Netica has remembered. If you are redoing operations, and the **Edit** → **Redo** menu item or toolbar button turns dim, it means that you have restored all the operations that were previously undone.

**Past File Save:** You may even undo operations that were done previous to the last time the net was saved to file. As you are undoing operations, at the point that the net matches what was last saved to file, the = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_changed_indicator.htm');return
```

```
false;">changed-indicator * will disappear, and after undoing further
```

```
operations, it will reappear. Then, if you start redoing operations, once again it  
will disappear when the net matches the file, and then reappear.
```

**Other Window:** The operations done in each window are remembered

separately, so if you return to window A after working in another window for a while, you can still undo operations previously done in window A.

## Selecting Nodes and Links

Many operations on nodes are done by first *selecting* one or more nodes, and then choosing the operation to do. Netica indicates a selected node by drawing it with negative colors.

**Selecting Single Nodes:** Select a node by clicking once on it. You may have to click on the node's title if clicking on other parts of it cause some action to occur. Otherwise, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click over a node and choose **Select**.

**Selecting Multiple Nodes:** Click down on the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_background.htm');return false;">background and then drag the selection rectangle to include at least a part of each node.

Choosing **Edit** → **Select Nodes** → **All** (or pressing **CTRL+A**) will select all the nodes in the net. To unselect all the nodes currently selected, just click once on the net background.

**Modifying:** When you select new nodes, any nodes that were previously selected will become unselected, unless you hold down the **CTRL** key while clicking or dragging. In that case the selected status of the new node(s) will be reversed, allowing you to add to, or remove from, the collection of selected nodes. Another way is to right-click on a node whose selection status you wish to change, and choose **Select** or **Deselect**.

**Repeating:** If you often need to select the same set of nodes, you can set them as a node-set, and then select them by choosing **Edit** → **Select Nodes** → **In NodeSet**.

**Reversing:** To reverse the selection so that it consists of only the nodes currently not selected, press **CTRL+SHIFT+A** or choose **Edit** → **Select Nodes** → **Invert Selection**.

**By Properties:** Alternatively, you can select nodes and links according to their properties or relationships. For example, you can select nodes with equations, findings, dependent on, parents, ancestors, children and descendents

of selected nodes by choosing **Edit** → **Select Nodes...**

**From Text:** **Edit** → **Select Nodes** → **Listed in Clipboard** will select all the nodes whose name (not [title](#)) appears in the text currently copied to the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_clipboard.htm');return false;">clipboard. For example, if your word processing program has a document that mentions several nodes by name, you could select that text, copy it, and then do **Select Nodes** → **Listed in Clipboard** in Netica to select them.

**Save/Restore:** You can save the current selection as text with **Report** → **List of Selected** ([more info](#)), and then restore it using **Edit** → **Select Nodes** → **Listed in Clipboard**.

**Doesn't Work:** If you are selecting [belief-bar](#) or [meter](#) style nodes, then you should click on the title (or name) of the node, since clicking elsewhere may be interpreted as = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entering a finding (in which case the node will be darkened instead of drawn with negative colors).

**Links:** To select a link, click directly on it. It will become drawn in hilited outline form. As with nodes, if the **CTRL** key is held down while clicking, it will reverse the selection status of only the links being clicked on, which can be used to add or remove from your collection of selected links. You can also right-click directly on a link and choose **Select** or **Deselect**.

**Note:** Links cannot be selected by dragging the selection rectangle over them. Nodes and links cannot both be selected at the same time, so whenever you select links, any nodes that are selected will become unselected, and vice-versa.

## Moving Nodes and AutoGrid

To move a node, click down on it, drag it to its new position, and then release the mouse button. If you wish to move a set of nodes and the links between them, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the nodes you wish to move, then click down on one of them and drag the set to the new desired location.

There is an underlying “grid” of positions, and when you add or move a node, the node will be shifted slightly so that its center is directly over one of these grid positions, providing the AutoGrid is turned on. This allows you to quickly draw a net whose nodes are perfectly aligned, and with links running perfectly horizontal or perfectly vertical. To turn AutoGrid off or on, = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">toggle

**Layout** → **AutoGrid**. If you want to change the spacing of the grid, to make it finer or coarser, choose **Layout** → **Grid Spacing**, and a dialog box will appear allowing you to make the setting.

To return to the original spacing when a net is developed with AutoGrid turned off, or if changes are made to grid spacing, choose **Layout** → **Snap To Grid**.

**Nudging:** Sometimes it is useful to be able to move a node, or set of nodes, a small predictable amount. You can use the arrow keys to do this nudging.

Simply select the nodes you wish to move, and then press the key with the arrow in the direction you want them to move. The distance they will be moved with each press of the key is the grid spacing (providing the AutoGrid is turned on), so you can adjust this amount using the **Layout** → **Grid Spacing** menu entry. If the AutoGrid is turned off then they will be moved the smallest perceptible amount with each press of the key.

It is also possible to move or reshape a link by a small predictable amount ([More Info](#)).

**Alignment & Spacing:** To aid with visual comprehension of your net, or when editing a net, use the **Space Evenly** and **Align** features in the **Layout**

menu. If you are working with a large net, or are in [multiple-adding mode](#), you can quickly and evenly space out a row or column of nodes. First, decide whether you'd like the nodes to be spaced **By Gaps**, **By Centers**, **By Left/Top Edges**, or by **Using the Grid** and select that option by choosing **Layout** → **Space Evenly**. Next, `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_select_node.htm');return false;">`select the desired nodes and again choose **Layout** → **Space Evenly**, this time choosing spacing of either **Vertically** or **Horizontally**.

Similarly, you can quickly line up rows and columns of nodes. You can align nodes **By Centers**, **By Left/Top Edges**, or **By Right/Bottom Edges** when you choose **Layout** → **Align**. Then `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_select_node.htm');return false;">`select the nodes to be aligned, choose **Layout** → **Align** again and click on **Row** or **Column**.

For more info on improving a net's visual appearance, see [display style](#).

**Resizing the Net:** If you would like to change the spacing of the entire net you can choose **Layout** → **Spread Out/Compact**. Entering any number greater than 100, in the dialog box that comes up, will spread out the entire net, while numbers less than 100 will compact it. This operation only changes the spacing between things, without altering their [size](#). By selecting some nodes prior to the operation, only those nodes will be moved.

## Reshaping a Link

When a net diagram becomes large it can be very difficult to view if all the links between the nodes are straight lines. Even small diagrams can sometimes be made more legible by choosing suitable paths for the links.

**Adding Bends:** To shape a link, first click once on the link to select it. The link will be drawn with the hilite color surrounding it. Then click down again on the selected link, and drag the cursor. A bend will be placed in the link and dragged by the cursor. You can repeat this to add as many bends as you wish.

**Moving Bends:** To move the position of a bend, first select the link as before. The link will become hilited and there will be a square box at each of its bends, and at its two endpoints. To move a bend or an endpoint, click down in its hilited box, drag it to its new position, and then release the mouse button.

**Removing Bends:** Moving one bend into another, or into an endpoint, will combine them, which can be used to remove bends. If you wish to immediately remove all the bends in some links, select them and choose **Layout** → **Straighten Links**. If no links are selected when you perform this operation, all the links in the net will be straightened. Another method is to right-click directly on a link and choose Straighten.

**Moving Ends:** There are some special considerations when moving the endpoints of a link. The arrow end of a link can not be dragged very far from the node it points to (if you try to, the arrow will just bump against an invisible barrier). This prevents the creation of misleading looking diagrams (since the link is still considered to be connected to the node). If you drag the non-arrow end of a link too far from its parent node, then the link will become “disconnected” from that node. You can tell this has happened because the name of the link will suddenly appear at the endpoint you have moved ([More Info](#)).

**AutoGrid:** If the [AutoGrid](#) is on (there is a check-mark in front of the **Layout** → **AutoGrid** menu entry) then the endpoints and bends of a link will always be placed on grid positions after a move. This makes it easy to quickly draw net diagrams with aligned link segments, and to make some segments perfectly horizontal or perfectly vertical. You can turn the AutoGrid on or off by `= 4 && typeof(BSPSPopupOnMouseOver) == 'function'`

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_toggle_menu.htm');return false;">toggl
```

**Layout** → **AutoGrid**, and set the spacing of the grid with **Layout** → **Grid Spacing**.

**Nudging Bends and Ends:** If you have just been moving a link bend or endpoint, and the link is still selected, you can nudge that bend or endpoint a little to get it exactly where you want by pressing the appropriate arrow key. With every press of the key, it will move a distance of the grid spacing, if the AutoGrid is on, or the minimum perceptible distance if the AutoGrid is off.

## Saving a Net to File

To save the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active net to a file, choose **File** → **Save As** or click the  toolbar button. You will be prompted for the file name and what directory to place it in. After you have saved a net, its window title will no longer be “Untitled-x”, but instead it will be based on the name of the file. Then you can save it subsequent times by choosing **File** → **Save** or pressing **CTRL+S** without being prompted for the name.

**Changed-Indicator:** If there is a \* or + after the title of the net in its window title bar, then it means that the net has been changed since it was last read or saved, so that the screen version is different from the file version. If you exit Netica or close the window, you will be asked if you want to save the changes first. When you save the net, the changed-indicator will disappear. A \* indicator means substantial changes have been made, a + means only cosmetic changes were. (= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_changed\_indicator.htm');return false;">More Info)

**File Format:** The net normally gets saved as an efficient binary file in the NETA format (optionally encrypted), with file extension .neta. This file is machine and platform independent, so Netica Application and Netica API on Windows, Linux, MacOS, Solaris, etc. using Intel, AMD, PowerPC, ARM, etc. processors can work with it. All future versions of Netica will be able to open it, and most past versions can as well (but of course by ignoring the new Netica features that they lack).

From the dialog box for saving the file, you can choose an alternative format known as the DNET format (file extension .dne or .dnet), from the "Save as type:" choice at the bottom. It contains only = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_ASCII.htm');return false;">ASCII text, so it is useful if you want to examine or edit the file with a = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_text_editor.htm');return false;">text editor, or if
it will be read by other programs that understand DNET. All future versions of
Netica will be able to read and write DNET files, and DNET files have the
same interoperability described above for NETA files. A document precisely
describing the DNET-1 file format is available from the = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_Norsys.htm');return false;">Norsys web site
(called "DNET_File_Format.txt").
```

**Cases:** To learn about saving and reading cases, click [here](#).

## Deleting Nodes and Links

**Nodes:** To delete a node or nodes, first select them, and then press the `DELETE` key, click the  toolbar button, or choose **Edit** → **Delete**. Alternately, you can just = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click a node and choose **Delete**. This will remove the nodes, and all links from other nodes going to them. Links going from them to their child nodes will be disconnected just before the deletion. These disconnected links will be selected after the nodes are removed, so you can easily delete them as well by just pressing the `DELETE` key a second time.

If you wish to remove the nodes, but maintain the global relationship of the remaining nodes (i.e. the full = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_joint\_distribution.htm');return false;">joint probability distribution), you should = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_absorb\_node.htm');return false;">absorb the nodes.

**Links:** To delete a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link.htm');return false;">link, click on it to select it, and then press the `DELETE` key or choose **Edit** → **Delete**. To delete a series of links, select them all and then press the `DELETE` key. When a link is deleted, the conditional probabilities of the child node are collapsed to eliminate their dependence on that parent, as if the parent took on its first state. When a link is deleted the parent node is not affected in any way.

## Cut, Copy, Paste and Duplicate Nodes

**Cutting Nodes:** = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">Selected nodes  
can be removed from the net and transferred to the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_clipboard.htm');return false;">clipboard by  
choosing **Edit** → **Cut** (or pressing **CTRL+X**, clicking , or = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking  
and choosing **Delete**).

Alternately, they can be entered into the clipboard without removing them  
from the net with **Edit** → **Copy** (or pressing **CTRL+C**, or clicking ).

If no nodes are selected when the copy command is done, the whole net will  
be copied to the clipboard.

**Pasting Nodes:** Once the nodes are in the clipboard, you can duplicate them  
into any net by opening the window for that net, optionally clicking in it where  
you want them to be placed, and then choosing **Edit** → **Paste** (or pressing  
**CTRL+V**, or clicking ). When copying from one net window to another, the  
two windows must both be within the same running instance of Netica; if there  
are two separate instances of Netica running, you can't copy and paste between  
them.

All links between the transferred nodes, all their properties, and all their = 4  
&& typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs will be  
maintained during the transfers. Links that enter cut or copied nodes from a  
node not being cut or copied will appear as disconnected links in the duplicate.  
Links that go from a cut or copied node to a node not being cut or copied, will  
not be copied.

Nets may be copied and pasted into other programs, such as Microsoft Word.

## More Info

**Duplicate:** If you want to duplicate a large portion of the nodes, choose **Window → Duplicate Net** (or right-click the net background and choose **Duplicate in New Window**). In the resultant net, delete any of the nodes you don't want.

**Node Names:** Since every node name in a net must be unique, a node's name must be changed when duplicating it into a net already having that name.

Netica does this if necessary, by adding a number to the end of the old name, or incrementing the number if there already is one. Of course node titles don't have to be unique, so after duplicating or pasting you may have several nodes with the same title, which you can change manually.

## Right-Clicking

You can = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click on  
nodes, links and the net = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_background.htm');return false;">background.

Right-clicking on the background is for operations that apply to the net in general. In all cases, a menu will appear from which you can choose an operation. If you are using a Mac, read this [FAQ page](#) to find a right-clicking alternative.

**Menu Contents:** The contents of the menus which appear may vary from time-to-time depending on the status of the object being clicked on, and what other objects are selected. For instance, the **Align** item appears only if at least 2 nodes are selected, and **Space Evenly** only if 3 or more nodes are selected.

The choices for adding links depend on whether or not any nodes are selected, and whether or not you click on a selected node (you should try the three possibilities to see the different ways you can add links). If you don't see something that is usually there, try the overhead menu.

**Selection:** If you want an operation to apply to several nodes or links at once, = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select them,  
and then right-click on one of the selected ones. Even if some are selected, if you right-click on one that isn't selected, the operation will apply to that one only. However, there are a few menu entries that always apply only to the element being clicked on, even if it is part of a selected set. They are: **Properties**, **Select** and **Deselect** (which allow you to easily add or remove elements from the selected set). [More info](#) on doing operations on multiple nodes at once.

**User-Defined Fields:** Right-clicking provides a great way to examine or set the [user-defined fields](#) of a node or a net, instead of using the node properties dialog box. Right-click on the node or net background, choose **Modify** → **User Defined**, and a menu will appear listing all the fields for that node and

their values in "field = value" form. By choosing one of the items, you can change it. You can even set the fields for [several nodes at once](#).

**Network:** In general, the right-clicking menus have the same functions as the overhead ones, but there are some differences, described in the sections on [adding nodes](#), adding [titles/text](#), [duplicating the net](#) and [selecting nodes/links](#).

You can also use right-clicking to change the internal name of your net by right-clicking on the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_background.htm');return false;">background

and choosing **Modify** → **Rename**. You will be requested to enter an = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_IDname.htm');return false;">ID name that will

be stored with the net and used to identify it by = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_Netica\_API.htm');return false;">Netica API

(this is different from the name in the title bar, which is based on the file name the net is saved to, although if the internal name is not set, it will also be based on the file name). The choices in the menu **Modify** → **Default Node Style** set the default node styles for the net (the same as the overhead Style menu when no nodes are selected).

## Placing Text on Net Diagrams

You can put text directly on the net diagram, which is suitable for titles, comments, notes, copyright and other legal notices, small = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs of nodes, warnings, directions for usage, etc.

**How To:** To add a title or text to your net, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click on the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_background.htm');return false;">background and choose **Modify** → **New Title** or **New Note**. A constant node will be created.

**Don't Use:** If you have a large block of text to document a net, it is better to place it in the documentation window, and if you want to provide a description for a node, but not have it visible on the net diagram, use the node's description field, or put it in a state comment.

**Color:** By default, text elements have a light-blue background. You can modify this for the whole net by right-clicking on the net background and choosing **Modify** → **NodeSet Properties**. In the properties dialog box, click on 'Documentation' and then **Set Color**. Text elements can be added to node-sets, so if you wish to link the coloring of a text element to that of some nodes, add it to the same node-set they are in. To make text elements of varying color, create new node-sets having the desired colors.

**Font and Size:** You can also = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select some text elements, and change their font or size by right-clicking on one, and choosing **Style** → **Font**. It is not possible to have more than one coloring or font within a single text element.

You can select, [search](#), copy and paste, duplicate, edit, move and delete text elements in the natural way. For example, to duplicate an element, you hold down `CTRL` key and drag it. To change the text in it, you double-click it.



**Tips:**

- If you have some text elements that you use repeatedly, such as a copyright notice, you can copy and paste them to the **Node Palette**, found on the **Windows** menu, so that they will be available anytime you use Netica.
- There are a number of [style options](#) available for improving the overall look of your net, which is useful for printing and presentation purposes.

## Documentation Window

When you [create](#) a Bayes net or decision net, it is usually a good idea to write a short description of it, or provide other written documentation, which stays in the same file as the net itself.

With the net window = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active, choose **Window** → **Description of Net**, and a window will appear in which you can place = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_editor.htm');return false;">plain text.

This is called the net's *documentation window*, and it will automatically be closed when the net window is closed. If the documentation window was open when the net was last [saved](#), then when the net is next [opened](#), it will automatically be opened as well, and in the same place as when the net was saved.



### Tips:

- If you are distributing the net to other people, make sure the documentation window is open when you last save the net, so that they see it right away when they open your net.
- Informative text can also be placed [directly on the net diagram](#), or within a node's [description](#) box.

## Navigation and Resizing

Using the scroll bars you can move around to view or work on various areas of the net. The `HOME`, `END`, `PAGE UP` and `PAGE DOWN` keys and **arrow** keys may be used as well. While you are moving a node or a link bend, you can make the window scroll by attempting to drag them outside the window in the direction you wish to scroll. The further you go outside, the faster it scrolls.

The fastest and most popular way to navigate to a new spot is to use [Global Zoom](#).

Sometimes the easiest way to scroll to a node that you know the name of is to use [Find](#). And you can jump to the end of a link by = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-clicking it`  
and choosing **Find Parent** or **Find Child**.

You can also scroll by holding the `ALT` key down and clicking on the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_background.htm');return false;">background.`

The cursor turns into a hand, indicating that holding the mouse button down and dragging the mouse will "push" the net diagram in that direction.

**Go Back:** To have Netica automatically scroll to the place where you were previously editing (or clicked the mouse button), choose **Edit** → **Go Back**, or press `CTRL+SHIFT+G`. Pressing it again will take you to the place before that, and so on. Eventually you will return to the starting position, so you can quickly cycle through the last 6 places visited by rapidly pressing the key.

**Drawing Size:** The area within which you can create a net diagram is a rectangle of certain size. Once you scroll to the end of it you cannot scroll any further. This area is increased automatically when needed, such as when dragging or pasting nodes past its boundary, or enlarging the window (including maximizing or [zooming](#)) bigger than its size.

You can also manually change the size of this area by choosing **Layout** → **Drawing Size**. Entering blanks in the dialog box that comes up will make the size as small as possible to contain everything. The drawing size is entered as

the number of pages wide by the number of pages tall. The size of a page is based on how much your printer puts on one printed page, so it is influenced by **File** → **Printer Setup** (see [Printing](#)).

## Zooming In and Out

When a net diagram becomes large, it is handy to be able to “zoom out” to see more of it in the window at once. Press `CTRL+>` or choose **Window** → **Zoom** → **Out**. Each time you do this, the magnification of the drawing will be decreased, so you will be able to see more of it. To increase the magnification, press `CTRL+<` or choose **Window** → **Zoom** → **In**. You can also access the Zoom menu by right-clicking on the net = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_background.htm');return false;">background.
```

Whatever the zoom magnification, you can edit or use the net in the normal way.

**Global Zoom:** Netica provides a very fast and convenient way of obtaining an overview of a diagram, and quickly navigating to a new point on it, called *Global Zoom*. When a large net diagram is the = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_active_window.htm');return false;">active
```

window, you can press the `SPACE BAR` and hold it down. While it is depressed, Netica will zoom out, so that the whole net fits in the window. Move the cursor to the point on the net you wish to work on next, and then release the space bar. Netica will zoom back to normal, with the spot of interest in the center of the screen.

If you wish to use Global Zoom just to get an overview of the net, but you don't want to navigate to a new spot, then don't move the mouse cursor at all, or else move it clear out of the window, before releasing the `SPACE BAR`.

**Particular Magnifications:** You can zoom out just enough that the whole net will be displayed in the window by choosing **Window** → **Zoom** → **To Fit**.

After any sequence of zoom operations you can return to 100% magnification with **Window** → **Zoom** → **To Normal**.

**Window** → **Zoom** → **To...** allows you to choose an exact magnification amount, and **Window** → **Zoom** → **Back** returns the magnification to what it was before the last zoom operation.

**Printing:** The zoom magnification does not effect output sent to a [printer](#); to change that magnification, choose **File** → **Printer Setup** and put blank entries for the number of horizontal and vertical pages in the dialog boxes that come up afterwards. You will then be queried for a printer magnification.



**Tip:** Often nets are built using a larger font, then as the nets get bigger, developers just zoom out a bit and continue building. Instead build the net using a slightly smaller font size, and work at Normal (100%) zoom magnification, since Netica is optimized for that.

## Search/Find

To find the occurrence of some text in the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active net,  
choose **Edit** → **Find** and enter the search text. Netica will search node names,  
titles, state names and comments for the text.

When Netica finds a node containing the search text it will scroll the window  
to that node and select the node. To find the next node containing the text, use  
**Edit** → **Find Next** (or press the F3 key). **Edit** → **Find All** will select all the  
nodes containing the text.

You may only be interested in the occurrence of the search text in one type of  
field; say you are looking for the node with a certain name. As Netica finds  
each occurrence of the search text, it prints in the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return  
false;">Messages window where it found the text. For example, it might print:  
"heavy pressure" found in comment of node Depth. So you can just watch the  
Messages window until you see that Netica found the text in a node name.



### Tips:

- If you want to easily select a certain subset of nodes, you can give them a  
keyword, say "\$Unfinished", by putting that keyword in the [description](#) of  
each of them. Whenever you want to select them just do an **Edit** → **Find** on  
"\$Unfinished", and then do **Edit** → **Find All**. It is good to precede  
keywords with some special symbol, like '\$', so the search can't be confused  
by node titles, state names, etc., and you will remember it is being used as a  
keyword. However, usually a better method to work with sets of nodes is to  
[node-sets](#) and [select](#) them based on their node-set names.
- You can make a printed list of all the nodes containing certain text, by first  
doing a **Find All** search on that text, then with the nodes still selected,

choose **Report** → **List of Selected**. [More Info](#)

## Selecting Nodes/Links by Properties

In addition to the ways for [manually selecting nodes or links](#), you can make selections based on the properties or relationships of nodes or links.

By choosing **Edit** → **Select Nodes** from the overhead menu, or = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-clicking`  
the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_background.htm');return false;">background`  
and choosing **Select Nodes**, you will be presented with a list of the following options:

### Select Nodes →

**All** - All the nodes including constant nodes, and also titles and notes.

**Invert Selection** - Nodes that are currently unselected (and de-selects those that are selected).

**In NodeSet** - The nodes that are members of the chosen node-set.

**Listed in Clipboard** - Those nodes whose name (not title) appears precisely in the text you have just copied to the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_clipboard.htm');return false;">clipboard`  
(multiple nodes should be separated by commas or = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_whitespace.htm');return false;">whitespaces).`

**Match Search String** - All nodes whose name, title or state name or title, or node description contains the search text (which you will be asked for).

**With Tables** - Any nodes whose = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT table or = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"`

onclick="BSSCPopup('X\_PU\_function\_table.htm');return false;">function table is completely filled in.

**Incomplete Tables** - Any nodes whose CPT table or function table is not completely filled in (i.e., contains some blank entries).

**With Equations** - Nodes that have an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_equation.htm');return false;">equation defined.

**With Findings** - Nodes with any type of finding ("evidence"), including positive, negative or likelihood ("virtual") findings.

**Likelihood Findings** - Nodes with only a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_likelihood\_finding.htm');return false;">likelihood finding ("virtual evidence"), or multiple likelihood findings that aren't equivalent to a single = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_positive\_finding.htm');return false;">positive finding.

**Parents** - All nodes that have a link pointing to some currently selected node.

**Ancestors** - All nodes that have a path of forward-pointing links to some currently selected node.

**Children** - All nodes that have a link coming from some currently selected node.

**Descendents** - All nodes that have a path of forward-pointing links from some currently selected node.

**Connected** - Nodes that have a path of links (ignoring directions) to some currently selected node.

**Info (D-) Connected** - The nodes whose beliefs could change if a finding were obtained for a currently selected node, based on the graph connectivity (or vice-versa). These are the nodes that are **not** = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

`BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_d_separation.htm');return false;">d-separated` with the current selection. You could use *Invert Selection* after this operation to find the nodes that are d-separated.

**Markov Boundary** - Nodes in the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Markov_boundary.htm');return false;">Markov boundary` of currently selected nodes. If `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_positive_finding.htm');return false;">positive findings` are obtained for the Markov boundary nodes, then the values of the currently selected nodes will not provide any additional information about any other node in the net.

### Select Links →

**All** - All the links in the net.

**Disconnected** - All the links in the net which have been `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_disconnected_link.htm');return false;">disconnected` from their parents.

**Ineffectual** - All the links in the net which have no influence on any inference results, because they are `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_ineffectual_link.htm');return false;">ineffectual`.

**Entering** - All links directly entering a node which is currently selected (i.e. has a selected node as a child).

**Exiting** - All links directly exiting a node which is currently selected (i.e. has a selected node as a parent).

**Interconnecting** - The links which directly inter-connect the currently selected nodes (i.e. have both a child and a parent which is selected).

**Cycle containing** - If there is a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_directed\_cycle.htm');return false;">directed cycle in the net, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">belief updating will be disallowed, and this helps to find it. Select a link in the cycle (perhaps as reported by an error message), and use this operation to select the rest of the cycle.

## One Operation, Multiple Nodes

Normally you modify a node using the [node properties](#) dialog box, which allows you to change any of the properties of a single node. However, sometimes you want to make the same change to several nodes at once. With Netica, you can add, remove, rename or reorder states, change discretization thresholds, enter findings, enter user-defined fields, change node-sets, etc. to as many nodes as wish, with one operation.

**How To:** Select the desired nodes, = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">`right-click on one of them, and make a choice from the menu. Any choice you make will apply to all the nodes, except for the choices: **Select**, **Deselect** and **Properties**.

**Entering Findings:** When several nodes are selected and you right-click on one and choose **Enter Findings**, you will be presented with a menu that lists all the states of all the nodes selected, with any findings already entered having check-marks. If you choose a state from the list, then every node having that state will get it entered as a finding, while the other selected nodes will not have their finding changed. If you want the other findings removed, then you should first choose **Enter Finding** → **Unknown (Retract)**, and then the desired finding. If the selected nodes are = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_continuous.htm');return false;">`continuous nodes, then you can enter a numeric finding for them all simultaneously using **Enter Finding** → **Numeric Value**.

**Setting States:** There are 4 choices for changing the states of a node available from the Modify menu: **Add State**, **Delete State**, **Rename State** and **Set States**. The first 3 are fairly straightforward; they adjust all effected = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_CPT.htm');return false;">`CPTs to produce the least disturbance possible. In the case of **Add State** and **Rename state** = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`

`BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_probabilistic_inference.htm');return false;">inference results will not be effected at all. Set States has the more complex behaviour:`

- If the new states have exactly the same names as the old ones, and there is the same number of them, but just in a different order, it re-orders the states and all related structures (such as `= 4 && typeof(BSPSPopupOnMouseOver) == 'function'` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_state_value.htm');return false;">state real values` and CPTs), so that all inference results will be unaffected.
- Otherwise, if there are the same number of states, it just renames the states (so inference results will be unaffected).

**Warning:** If there are a different number of states, all related structures (such as `= 4 && typeof(BSPSPopupOnMouseOver) == 'function'` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_state_value.htm');return false;">state real values`, state titles and CPTs) are removed, which of course will effect inference results. If you want to keep the tables, then you need to use **Modify** → **Add State** or **Modify** → **Delete State** instead.

**User-Defined Fields:** As described on the [right-clicking](#) page, you can set user fields of a node by right-clicking and choosing **Modify** → **User Defined**. If you click on one of several selected nodes, then a menu will appear listing all the fields already defined for at least one of the selected nodes. By choosing a field, you can set its value. If the field was originally defined for only some of the nodes, Netica will ask you if you want to change the value of only those nodes, or whether you want to add the field to all the nodes. If you make a field empty, Netica will ask you whether you want it to represent an empty field, or if you want the field removed, which you can use to remove fields from a large set of nodes at once. Likewise, you can add a field to a large number of nodes by choosing **Modify** → **User Defined** → **Enter**, and then entering the name of the new field.

## Node Properties

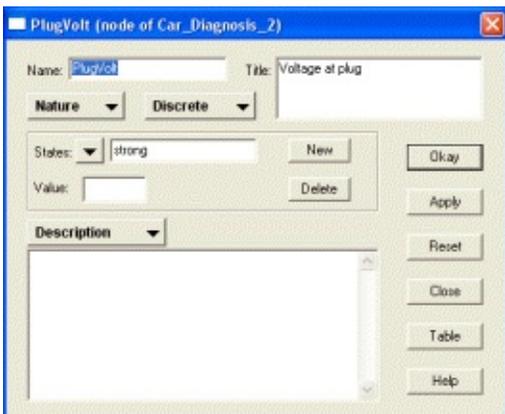
This chapter explains how to use the node dialog box to view or change the properties of a node. You can page through the entire chapter using the >> button above, see a [written description](#) of where information is, or jump directly to one of the following topics:

- [Node Dialog Box](#)
- [Buttons in the Node Dialog Box](#)
- [Node Name](#)
- [Node Title](#)
- [Node is Discrete or Continuous](#)
- [Node Kind](#)
- [Node States](#)
- [Node State Interval](#)
- [Node State Value](#)
- [User-Defined Fields](#)

## Node Dialog Box

**Purpose:** You use the *node dialog box* to change the properties of a node (e.g. its name, or how many = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state.htm');return false;">states it has). To change the way a node is related to its parents (e.g., its conditional probabilities), use the [table dialog box](#) instead, and to change the color of a node, see [Node Coloring](#).

**Obtaining:** Double-click a node to obtain its node dialog box, or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click on it and choose **Properties**, or select it and press the **ENTER** key. The latter method is very useful when adding new nodes to a net, since when a node has just been added it is already selected, and you just need to press **ENTER**.



The node dialog box can be used to set a node's:

- [Name](#)
- [Title](#)
- [States](#)
- [State numbers](#)
- [State intervals \(discretization\)](#)
- [User-defined fields](#)

**Other Fields:** At the bottom of the node dialog box is a text editing box, called the *multipurpose box*, within which you can view or change some [other node properties](#), as chosen by the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_graphic\_multipurpose\_selector.htm');return

false;">multipurpose selector.

**Buttons:** By pressing the  selector, you can choose the node [kind](#) from the options listed. If you click the  selector, you can change the node from [discrete to continuous](#). Making a change within the dialog box does not affect the node until the **Apply** or **Okay** button is pressed ([more info](#) on the right-hand buttons).

**Multiple:** You may have several node dialog boxes on the screen at the same time, and you may alternate between using different node dialog boxes and working directly in the net window. Each node dialog box pertains to a single particular node, but you can have [more than one](#) of them for the same node.

## Buttons in the Node Dialog Box

**Apply & Okay:** When you make changes in the [node dialog box](#), they don't affect the node until you click the **Apply** or **Okay** button. Then all the settings are transferred to the node in the net. The **Apply** and the **Okay** buttons have the same function, except the **Okay** button also removes the dialog box after transferring the settings.

**Reset:** Clicking the **Reset** button will make all the settings in the dialog box the same as the node in the net. If some external operation changes the node in the net, the settings in the dialog box aren't changed to reflect that until the **Reset** button is pressed.

**Close:** If you click the **Close** button or the  button in the title bar, the node dialog box will be removed. If you previously made some changes in it, and haven't pressed the **Apply** button, Netica will ask if you want to apply those changes first.

**Table:** Clicking the **Table** button will bring up the [table dialog box](#) for the same node, to view or edit its tables (= 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT, experience, etc.).`

**Help:** When you click the **Help** button, it will bring up the [Node Dialog page](#) of this Help system.

## Node Name

You can change the name of a node using the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_graphic\_node\_name.htm');return false;">“Name” text edit box of the node’s dialog box. When the dialog box first appears, the existing name of the node is selected, so if you just type a new name it will replace the existing name.

When you enter a new node name, Netica will check that it meets the restrictions of an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_IDname.htm');return false;">IDname. If you want to circumvent these restrictions, you can also give the node a title.

**Note:** When a node’s name is changed, Netica will automatically adjust the equation of the node and the equations of all it’s children nodes.

## Node Title

The [node dialog box](#) has a text edit box labeled = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_graphic_node_title.htm');return false;">"Title"`  
to provide an unrestricted alternative to the node's [name](#) (which must be an = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_IDname.htm');return false;">IDname`). You can [choose](#) whether the name, title or both are used to label a node on net diagrams.

There are no restrictions on what you may put in a title, and by using the `ENTER` key you can make the title more than one line long. Keep in mind that some unusual characters may display differently when the font is changed, or if the net is displayed on another type of computer. Node titles may contain characters from [any language](#).

Nodes are not required to have titles. If a node does not have a title, then anytime that Netica would normally use a title, it will use the node's name instead. The title is only used to label the node; anything more detailed should go in the [description](#) of the node.



### Tips:

- If the title is longer or taller than the text entry box, you can scroll it using the arrow keys.
- = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">Right-click`  
while editing a title to obtain a pop-up menu for copying, pasting, etc.
- By putting a blank line at the beginning or end of the title, or putting a space character at the beginning or end of a title line, you can effectively add some space between the outside border of a node and the writing within.



## Node is Discrete or Continuous

The [node dialog box](#) has a selector:  with the values **Continuous** and **Discrete**, which allows you to choose whether the node represents a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous or a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete.htm');return false;">discrete variable.

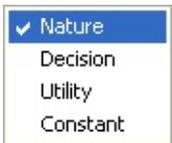
It is common to have a continuous variable that you want to break up into intervals so that you can treat it as a discrete variable, which is known as [discretizing](#) the variable. If the variable is truly continuous, it is usually best to make it a continuous node, and then [discretize it](#) with an interval list, rather than just making it a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete\_node.htm');return false;">discrete node. This provides better documentation of the node, and makes it easier if at a later time you want to discretize it another way (i.e. with a different number of states, or different cutoff points for the state intervals). Also, that way it can accept continuous values when [learning](#) from cases, or generate them when simulating cases.

Alternately, a discrete node may have numeric quantities attached to each state, so that each state can represent a number, but the variable is incapable of representing numbers between those of each state. [More Info](#).

**Note:** = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_utility\_node.htm');return false;">Utility nodes must be continuous and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_node.htm');return false;">decision nodes must be discrete.

## Node Kind

Directly below the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_graphic_node_name.htm');return false;">name` entry area of the [node dialog box](#) is a selector . Click on it to bring up the node kinds menu:



You can use it to turn the node into a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_Glossary_NR.htm#nature');return false;">nature` [node](#), [constant node](#), [decision node](#), or [utility node](#) (also known as a “value node”).

Nature nodes are called *deterministic nodes* when their relationship with their parents is deterministic, or *chance nodes* when the relationship is probabilistic. Netica will automatically determine if a nature node is deterministic and if so, will draw a thick border when in [labeled-box](#) style, and color it in the ["- Deterministic" node-set](#) color (if you haven't made other node-sets higher priority).

When a net is composed entirely of nature nodes it is called a Bayes net (also known as a “belief network”). If it also has decision and utility nodes, it is called a *decision net* (also known as an “influence diagram”).

`= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_decision_node.htm');return false;">Decision` nodes represent variables that the decision maker can control, and `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_utility_node.htm');return false;">utility` nodes represent variables the decision maker is trying to optimize.

**Alternative Ways:** You can also change node kinds straight from the net

window without using a node dialog box. To do this, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select a node or set of nodes you wish to change, and while holding down the **CTRL** key click the toolbar button indicating the kind of node you wish them to become:  for nature nodes,  for decision nodes, or  for utility nodes.

Or, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click over the node (or selected nodes) and choose **Modify** → **Kind**.

## Node States

In the [node dialog box](#), next to the label = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_graphic_states.htm');return false;">`“State”, is a down-arrow which yields a pop-up menu of the node’s = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_state.htm');return false;">`states. By choosing a state from the menu, it becomes the *current state* of the node dialog box. Its name will appear in the text edit box to the right of the pop-up menu, allowing you to modify it.

The states of a node constitute the domain of a categorical variable. They are not required to have names, but if one state is named then they all must be. It is highly recommended that a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_discrete_node.htm');return false;">`discrete node be given state names, for clarity and documentation purposes, although it is not necessary for = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_discretize.htm');return false;">`discretized continuous nodes.

**State Name:** When you enter a new state name, Netica will check that it meets the restrictions of an = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_IDname.htm');return false;">`IDname. Also, it must be different from the names of all other states of that node, although it may be the same as a state name for a different node in the net. If you need unrestricted labels, use [state titles](#).

**Adding/Removing:** To add a new state right after the current state, click the **New** button, and to delete the current state, click the **Delete** button. All discrete nodes must have at least one state, but there is no upper limit to how many they can have. Adding or removing several states to a node at once is usually done more conveniently by using the [Multi-Purpose Box](#), and adding

or removing a state from several nodes at once is best done by [right-clicking](#).

**Multiple Nodes:** Changing the states of a node, or many nodes at once, can be done quickly and easily by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking on a node, or a group of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">selected nodes, and choosing **Modify** to do one of the following operations: **Add State**, **Delete State**, **Rename State**, and **Set States** ([more info](#)).

If you are adding a state to multiple nodes, and some of them already have a state of that name, they will simply be skipped. If the existing states have = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state\_value.htm');return false;">numeric values defined, and you enter a number, then you will be given the option of adding states with that numeric value, or states titled with the number.

**See also:** [Ordering States](#)

## Ordering States

Right-click on a node (or if you want to do several, select them and then right-click on one of them), then choose from the menu **Modify** → **Order States** → **By Name**, if you want them in alphabetical order.

If you want them in some other order, choose **Modify** → **Order States** → **Standard Order**. To define what standard order is, edit the text file "State Order.txt". It is located in the same directory as Netica.exe, which can be discovered by choosing **Help** → **About Netica**, and then looking in the Messages window. You can use any text editor, such as NotePad, or you can choose from the Netica menu **File** → **Open As Text**.

In the file, the [state names](#) or titles appear one-per-line, in the order you would like them to have in the node. Viewing the existing file will make it clear.

If you want to have a different "State Order.txt" file for some particular set of Bayes nets, then you can put one in the same folder as those Bayes nets, and Netica will look for it there first.

When re-ordering the states, Netica will first try to match the state title against the entry in the "State Order" file, and if not found, then the state name. If some entries in the "State Order" file differ only by upper/lower case of some characters, Netica will properly consider the case while ordering (i.e. case sensitive). However if there is no exact match by case, but the file contains an entry that is all lower case, then Netica will do a case-insensitive match. For that reason entries in the "State Order" file often all appear in all lower case.

If you change the "State Order" file, you need to re-open any Bayes nets for which you were ordering states for the new file to take effect.

## Node State Interval

If a node dialog box is for a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous node, immediately below the state name edit box will be boxes labeled "Interval" . These are to provide = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state\_threshold.htm');return false;">threshold values for the intervals of each state in order to = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretize the continuous variable. This method of entering the thresholds is mainly just for touch-up work, or to coordinate the ranges with state names, since usually thresholds are entered using the Multi-Purpose Box.

The box to the left is the lower end of the interval for the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_current\_state\_node\_dialog.htm');return false;">current state (inclusive), and the box to the right is the upper end of the interval (exclusive).

To set or change the interval of the current state simply type a number in the appropriate box. States are arranged in a contiguous and (increasing or decreasing) monotonic way, which means that the upper end of one state interval will always be the lower end of interval of the neighboring state. As you type one of these in, Netica will fill in the other as well, so that when you change the current state you will see the number you just entered in one of the Interval boxes.

It is okay if the lower and upper bounds of an interval are the same number, to indicate a point. The numbers you enter may be integers, decimal numbers, scientific notation numbers, "infinity", or "-infinity".

## Node State Value

If a [node dialog box](#) is for a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_discrete_node.htm');return false;">`**discrete** node, immediately below the state name edit box will be a box labeled “Value”. This is to associate an integer or a real number value with each state. That number can be anything you choose, and should not be confused with the *state index*, which is an integer starting at 0 for the first state, and increasing by one for each subsequent state.

Usually you will not need to assign numbers to states, but sometimes it is useful. For example, the numbers may represent how many heads were obtained in 4 coin tosses. As another example, in a bang-bang control system, state 0 (“Off”) may map to -0.2 volts, and state 1 (“On”) may map to 6.0 volts.

If this node is the parent of a node with an `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_node_equation.htm');return false;">`equation, then the real values will be supplied to the equation.

To set or change the real number associated with the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_current_state_node_dialog.htm');return false;">`current state simply type a number in the box. You may find it easier to use the [Multi-Purpose Box](#) to enter or change the real number associated with a state. You can also change this number by right-clicking on an unselected discrete node (or a selection of nodes) and choosing **Modify** → **Numeric State Values**. [more info](#)



### Tips:

- To create a new node with node state values already assigned, `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_right_click.htm');return false;">`right-click on the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')`

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_background.htm');return false;">background
```

and then choose **Modify** → **New Node** → **Nature - Numeric Discrete**. In the dialog box that comes up you can use shorthand notation like [5,10]+1.

- If the "numbers" you are assigning as node state values have no real numeric significance (e.g. they are just identifying numbers), then you may want to make them node state titles instead, so that Netica doesn't try to interpret them as numbers.
- If you need to adjust Netica nodes to be able to read a case file in which a categorical variable is indicated with integers (for example, days of the week being 1 to 7, or gender being 1 or 2), you can assign the appropriate integer to each state of the node as the state value.

## User-Defined Fields

Sometimes it is useful to be able to define your own fields for nodes or nets, with names that you choose, and give them the kind of values you want (integers, real numbers, or text). These have sometimes been called *attribute-value pairs*, and are saved in the Bayes net file along with the rest of the information about the nodes and net.

```
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_Netica_API.htm');return false;">Netica API can also read and set user-defined fields, so they are a great way to communicate with your own program through Netica Application and then Netica API.
```

There are two ways to work with user-defined fields. The first is described in a page on the [Node Properties Dialog](#). The second is described below.

**Setting:** = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_right_click.htm');return false;">Right-click on a node and choose User Defined → Enter to create a new user field, or User
```

```
Defined → field name to change the value of an existing one. If you create a new user field, Netica will verify that its name meets the requirements of an =
```

```
4 && typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_IDname.htm');return false;">IDname. You can do the same thing to several nodes at once by = 4 &&
```

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_select_node.htm');return false;">selecting them before right-clicking one of them, and you can do it to the net itself by right-
```

```
clicking on the net's = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_background.htm');return false;">background
```

```
and choosing Modify → User Defined → field name.
```

**Observing:** To observe the values assigned to user fields of a node or net, = 4

```
&& typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

`onclick="BSSCPopup('X_PU_right_click.htm');return false;">`right-click the node or net background and choose **User Defined**. The values will appear within the menu. If they are too large for the menu, or contain special characters, you can use the [Node Properties Dialog](#) instead.

**Removing:** To remove a field from a node, right-click the node and choose **User Defined** → *field name*. When the dialog box asking for the value comes up, make it empty and press **Okay**. Then Netica will ask you if you just want an empty value, or if you want to remove the field. You can do the same thing to several nodes at once by selecting them before right-clicking one of them, and you can do it to the net itself by right-clicking on the net's background and choosing **Modify** → **User Defined** → *field name*.



### Tips:

- If you want to define the same user-field for several nodes at once, but give them each different values: Select the nodes, right-click and choose **User Defined** → **Enter**, enter the name you want for the field and press **Okay**, then leave the value box empty and press **Okay**, **No**. Now go to each node and right-click, then choose **User Defined** → *field name*. You will be able to keep track of which ones you've already entered, based on which are empty.
- For categorizing nodes, rather than user-defined fields, you may want to use [node-sets](#), since Netica has some specialized features for working with them, such as `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_select_node.htm');return false;">`selecting all the nodes in some node-set, reporting all the nodes in a node-set, and choosing display colors based on node-sets.
- If there are so many user fields that it is difficult choosing the name from a menu, instead of choosing **User Defined** → *field name*, you can choose **User Defined** → **Enter**, and then type the name of the user field you want to view or modify.

## Multi-Purpose Box

At the bottom of the [node dialog box](#) is a text edit box with a selector above it (called the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_graphic\_multipurpose\_selector.htm');return false;">multi-purpose selector). This box is used to view or enter several different properties of the node, and the selector is to choose which property to view or enter. The choices are:

- [Description](#)
- [Equation](#)
- [Thresholds](#) (for discretization)
- [State Numbers](#)
- [States](#)
- [State Titles](#)
- [State Comments](#)
- [Input \(link\) Name](#)
- [Delay](#)
- [Author](#)
- [When Changed](#)
- [User Defined](#)

To scroll in the text edit box, click down within the box and then drag the cursor in the direction you wish to scroll, or use the arrow keys to attempt to move the insertion bar past the box's boundary.

If you have a large amount of text to edit, you might find it easier to create it in a regular = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_editor.htm');return false;">text editor, and then cut and paste it into the multi-purpose box.

## Node Description – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **Description**, you can enter or view the *description* of the node. The description is an unrestricted block of plain text which can be used to store whatever information about the node you wish, such as its meaning, the meanings of its states, how it is to be measured, the origin of its probabilities, copyright information, etc. The description is currently limited to about 30 thousand characters.

Descriptive information about a node can also appear [directly on the net diagram](#). Whereas descriptive information relative to the entire net should appear in the [documentation window](#).

In addition, you can insert a comment that will come up directly in the Bayes net whenever you rest the cursor over a specific node. To do this, in the **Description** field, enclose the desired caption with square brackets and stars, for example:

[\* desired caption to be displayed \*]

## Node Equation – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **Equation**, you can view or enter the probabilistic or deterministic [equation](#) that provides the relation between the node and its parents.

For more information on equations, see: [Using Equations](#), [Equation Syntax](#) or [Built-In Functions](#)

## Node Discretization – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **Discretization**, you can enter or view the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_discretize.htm');return false;">discretization` thresholds of a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_continuous.htm');return false;">continuous` variable (if the variable is `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_discrete.htm');return false;">discrete` then the menu won't have a "Discretization" choice). You can also do this operation using the [Interval](#) setting boxes, which have the same capabilities and effect. However, usually the below method is more convenient, especially if you want to enter many evenly spaced values, or if you want to paste in the thresholds of all the states at once, which you have copied from another node or even from another program.

You simply enter all the interval thresholds into the box, separated by space(s), tabs, commas or on separate lines. There should be one more number than the desired number of states, since the first and last number specify the minimum and maximum values the variable can take (they can be "infinity" or "-infinity" if desired). If the number of states implied by the list is different from the node's current number of states, the node's number of states will be changed.

**Shortcut Notation:** If you want to create a list of evenly spaced values there are a few shortcut methods you can use. Each of the following special notations will expand into a list of numbers as described:

**[b, e] / n** will form a list beginning with b, ending with e, and having n intervals (so n + 1 numbers).

**[b, e] + d** will form a list beginning with b, ending with e, and each separated by d (except the last separation may be less if e – b is not evenly divisible by d).

**[b, e] /L n** will form a list beginning with b, ending with e, and divided logarithmically into n intervals (so n + 1 numbers).

**[b, e] +% d** will form a list beginning with b, ending with e, and each being d percent bigger than the previous (except the last may be less than d % bigger, if they don't fit evenly).

**Note:** If e is less than b then a decreasing list will be formed, but n and d should still be entered as positive numbers. The closing bracket may be replaced with a closing parenthesis if desired, to indicate excluding the endpoint e from the list formed. More than one of the above notations can be combined to form a longer list.

**Examples:** Each line below is a complete example entry:

-3.2 0 1 1e4 infinity

[0, 10] / 10

[0, 10) + 1, [10, 20) + 2, [20, 30) + 3, 33, 37

[1e6, 1] /L 6

[200, 10] +% 15

## Node States – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **States**, you can enter or view how many states the node has, and what their names are. You can also do these operations using the [State Name setting box](#), which has the same capabilities and effect. However, usually you will find this method more convenient, especially if you want to paste in the names of all the states at once, which you have copied from another node or even from another program.

You simply enter the [names](#) of all the states into the box, separated by space(s), tabs, commas or on separate lines. If the number of states in the list is different from the node's current number of states, the node's number of states will be changed.

An example entry is: low medium high

This operation can also be performed by using the [right-click menu](#). You can **Rename** or **Delete** States by right-clicking on a single or group of nodes and choosing **Modify**. A dialog box will come up asking which state you want to rename or delete.

[More Info](#)

## Node State Numbers – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **State Numbers**, you can assign a [number](#) to each state of a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_discrete.htm');return false;">discrete variable,`  
or view the current assignments (if the variable is = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_continuous.htm');return false;">continuous then`  
the menu won't have a "State Numbers" choice). You can also do this operation using the [State Value](#) setting box, which has the same capabilities and effect. However, the multi-purpose box is usually more convenient, especially if you want to enter many evenly spaced values, or if you want to paste in the numbers of all the states at once, which you have copied from another node or even from another program. ([more info](#) on state numbers)

You simply enter all the numbers into the box, separated by space(s), tabs, commas or on separate lines. There should be one number for each desired state. If the number of states implied by the list is different from the node's current number of states, the node's number of states will be changed.

**Shortcut Notation:** If you want to quickly create a list of evenly spaced values, you can use the same [shorthand notation](#) as used for discretizing a node.

## State Titles – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **State Titles**, you can enter or view the title of each node.

There are no restrictions on what you may put in a title, and by using the **ENTER** key you can make the title more than one line long. Keep in mind that some unusual characters may display differently when the font is changed, or if the net is displayed on another type of computer.

States are not required to have titles. If a state does not have a title, then anytime that Netica would normally use a title, it will use the [state's name](#) instead. The title is only used to label the state; anything more detailed should go in the [state comments](#) of the node.

## State Comments – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **State Comment**, you can enter or view comments specific to each state of the node.

For each state listed, you can enter specific comments about what that state means, or the purpose behind having the state. There are no restrictions on what you may put in a comment.

In addition, you can insert a comment that will come up within the Bayes net when you rest the cursor over a specific state. To do this, in the state comment field, enclose the desired comment with square brackets and stars, for example:

[\* desired comment to be displayed \*]

## Input Name – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **Input Name**, you can view or enter a name for each of the links going to the node. As soon as you set the selector to Input Name, an additional selector will appear to the right of it. Each choice of the new selector will correspond to a link, and will be labeled with the current link name if the link has one (in parenthesis), and/or the name of the parent node it comes from if the link is not disconnected.

Make a choice from the right-hand selector, and then enter the input-name in the box below (it must be a legal = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_IDname.htm');return false;">IDname).

Input names are used for [net libraries](#) and [equations](#).

The input name will appear on the net diagram when you hover the cursor over its link.

**Note:** You can also set Input Names by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking a link(s) and choosing **Input Name**.

## Link Delay – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **Link Delay**, you can view or enter a time-delay for each of the links going to the node. As soon as you set the selector to Link Delay, an additional selector will appear to the right of it. Each choice of the new selector will correspond to a link, and will be labeled with the link name if there is one, and/or the name of the parent node it comes from if the link is not disconnected.

The purpose of adding a link delay is for creating a [dynamic Bayes net](#).

**Note:** You can also set link delays by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_link.htm');return false;">selection of links and choosing **Delay**.

## **Author – Multi-Purpose Box**

When the [multi-purpose selector](#) of a node dialog box is set to **Author**, you can enter or view the author or source of information specific to a node.

This is a useful function when there are multiple people involved in building the Bayes net.

## When Changed – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **When Changed**, you can view the time and date the node was last changed, based on the clock in your computer. This value is similar to the one maintained by your computer operating system for when each file was last changed, but it can be more useful for Bayes nets since it actually records the time of change, not just the time of file saving, and it provides a separate value for each node. You cannot change this value; it is for observation only. If no value appears, it means that the time of the last change was not saved in the Bayes net file, and the node has not been changed since reading from file.

## User Defined – Multi-Purpose Box

When the [multi-purpose selector](#) of a node dialog box is set to **User Defined**, you can enter and view [user-defined fields](#) of the node.

**How To:** For the first field, you will need to click **New Field...** twice. A dialog box will come up asking you to enter a name for the new field (Netica will verify it meets the requirements of an = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_IDname.htm');return false;">IDname).`

Once a new field is defined, enter its value in the large text space. You can enter a number or any text; if the text is long, you might want to paste it in. Continue entering new fields or click **Okay** to exit the node dialog box.

**Removal:** If you make the entry empty, then Netica will ask you whether you want it to be a field set to empty, or whether you want the field removed.

**Better Way:** For many operations on user-defined fields, the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-click`  
menus offer a [more convenient method](#).

## Node Tables

The following chapter discusses the tables representing the relationship between nodes and their parents, and how to view or change those tables. You can page through the entire chapter using the **BROWSE** button above, see a [written description](#) of where information is, or jump directly to one of the following topics:

- [Table Dialog Box](#)
- [Meaning of the Tables](#)
- [Changing Table Entries](#)
- [Buttons in the Table Dialog Box](#)
- [Node Selector](#)
- [Deterministic/Chance Selector](#)
- [Kinds of Tables](#)
- [Scrolling and Navigating](#)
- [Empty Cells](#)
- [Cells Containing X](#)
- [Selecting, Copying and Pasting Cells](#)
- [Table Menu Commands](#)
- [Changing Column Order](#)
- [Multiple Dialog Boxes for One Node](#)

## Table Dialog Box

**Purpose:** You use the *Table Dialog Box* to enter, change or view the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_relation.htm');return false;">relationship of a node with its parents (i.e. its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs). Choosing which nodes are going to be the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents is not done with this dialog box, but rather by adding links. A node does not specify a relation with its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_child\_node.htm');return false;">child nodes, since those relations are specified at the child nodes. If you want the relation to be specified by an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_equation.htm');return false;">equation, or if you want to change the properties of a node (e.g. its name, or what states it has), use the node dialog box instead.

**Obtaining:** Obtain a table dialog box for a node by selecting it, and then choosing **Table** → **View/Edit**, clicking the tool button with the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_relation\_symbol.htm');return false;">relation symbol: , or by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking on the node and choosing **Table**.

**Table:** The relationship is expressed in the form of a table. The following describes what the table entries mean, how to change them, how to select, copy and paste them, how to scroll and resize the table, and how to reorder its columns.

**Kinds of Tables:** The table dialog box can be used to view or edit several different kinds of tables for the node, which you choose with the [table selector](#).

The node's = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_function\_table.htm');return false;">function

table or conditional probability table (= 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT) are the most

commonly used, but if you have Netica [learning from data](#) or connecting to a

database, then its = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_experience.htm');return false;">experience

table, unnormalized probability table and [counts table](#) will be of interest.

**Controls:** Making a change within the dialog box does not affect the node until the correct [button](#) is pressed. There are selectors to [choose the node](#) to work on, and to make it [deterministic or probabilistic](#). There are several [menu commands](#) that can modify the table.

**Multiple:** You may have several table dialog boxes on the screen at the same time, and you may alternate between using different table dialog boxes,

different node dialog boxes, and working directly in the net window. To use

one of the table dialog boxes it must be the = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active

window. You may even have [more than one](#) table dialog box for the same

node.

## Meaning of the Tables

For a Bayes or decision net to be fully specified, every nature and utility node must have a filled-in table. Each table expresses the value of the node in terms of its parents (or as a constant if the node has no parents). If the node is deterministic then the table will be a function which provides a value for the child for each possible configuration of parent values. If the node is probabilistic (i.e. a chance node), then the table will provide a probability for each state of the child, for each possible configuration of parent values.

For example, suppose node A (which can take on values Low, Medium or High) and node B (which can take on values True or False) are the two parents of node C (which can take on values Small, Midsize or Large). It is best to think of the relation between them as being located at node C (the child), and its [table dialog box](#) might look something like this:

The screenshot shows a dialog box titled "C Table (in net Relation\_Dialog\_Example)". It contains a dropdown menu for "Node:" set to "C", and buttons for "Apply", "Okay", "Chance", "Percentages", "Reset", and "Close". Below the controls is a table with the following data:

A	B	Small	Midsize	Large
Low	True	10.000	35.000	55.000
Low	False	17.250	82.750	0.000
Medium	True	12.100		
Medium	False	33.333	33.333	33.333
High	True	x	x	x
High	False	54.328	45.672	3.2e-012

On the left-hand side is a vertical list of all the configurations of parent values. On the right-hand side is one column for each state of C. The numbers in the table provide conditional probabilities for the values of C, given that the parents take on the configuration of their row. For example, the 10.000 (percent) in the upper left corner means that  $P(C=Small \mid A=Low, B=True) = 0.1$ . The number  $3.2e-12$  at the bottom right means  $P(C=Large \mid A=High, B=False) = 3.2 \times 10^{-14}$ .

Empty cells indicate probabilities that have not yet been specified. Examples are the two blank cells on the third row.

Cells with X indicate an impossible condition. For example the three x's on the fifth row indicate that the designer believes that the condition  $A=High$  while  $B=True$  is impossible.

## Changing Table Entries

**Purpose:** You will often bring up a [table dialog box](#) just to view the relation it represents. For example, if you have just solved a decision net, you can use it to view the optimal decision functions that Netica has found. However, its main purpose is to enter or change node relations when building a Bayes net or decision net, or doing [what-if analysis](#) with an existing net.

**Deterministic:** If the dialog box is for a deterministic relation, then you change a table entry (also known as a *cell* of the table) simply by clicking down on top of it, and then making a choice from the pop-up menu which appears.

**Probabilistic:** If the dialog box is for a probabilistic relation, then you change a probability by clicking on it to select it, and then typing in the new number. Or you can click on it to select it, then click within the selection where you want to change a few digits. You can enter the numbers as decimal fractions (e.g. 0.25) or percentages (e.g. 25) by changing the [table selector](#).

**Navigating:** When a cell is being edited, the insertion point will be flashing where new digits will enter. You can use ← or → keys to move it around within the number, and if it gets to the edge of the number, it will jump to the next cell in that direction. Further presses will jump to subsequent cells in that direction. The ↑ and ↓ arrow keys can be used in the same way to jump to cells above or below the currently selected or edited cell. Pressing the `TAB` key jumps to the “next” cell, which is the cell to the right of the current one, unless it is at the end of the line, in which case it is the first cell on the next line.

Another way to navigate, useful in large tables, is to [pick the parent value](#).

**Multiple:** Choose **Table** → **Enter Experience** to enter uniform experience tables for all = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_select_node.htm');return false;">selected nodes at once.`

## Buttons in the Table Dialog Box

**Apply & Okay:** When you make changes in the [table dialog box](#), they do not affect the node until you click the **Apply** or **Okay** button. Then all the settings are transferred to the node in the net. The **Apply** and the **Okay** buttons have the same function, except the **Okay** button also removes the dialog box after transferring the settings.

**Reset:** Clicking the **Reset** button will transfer the current values from the node in the net to the dialog box. There are two main uses for this button. The first is if you make a mistake while you are changing the table, and you haven't yet clicked the **Apply** button, you can "revert" to the original node by clicking the **Reset** button (if you have already clicked the **Apply** button, you should make the net window = `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_active_window.htm');return false;">active, do an Edit → Undo, then go back to the table dialog box and click Reset). The second use for the Reset button is to update the table dialog box after something else has changed the relation of the node.`

For example, you might be solving a decision net repeatedly, while varying some part of it. After each solution you want to see how the optimal decision function has changed. You would bring up the table dialog box for a decision node, leave it up, and after each re-solution of the net you would click the **Reset** button to observe the changes in the tables. Another example is that you are having Netica [learn](#) a Bayes net from a number of case files, and you want to observe how the conditional probabilities change as the learning process continues.

**Changed \*:** If you make any changes to the table, but have not yet pressed either the **Apply** or **Okay** button, then a \* will be displayed in the window's title bar to show that the table is currently different from that of the node in the net. If instead the node in the net is changed by something else (such as learning from data, solving an optimization, another table dialog box, etc.), then an asterisk in parenthesis is placed in the window's title bar: (\*)

**Close:** If you click the **Close** button, or the  button in the title bar, or choose **File** → **Close** while the dialog is = `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"`

`onclick="BSSCPopup('X_PU_active_window.htm');return false;">active`, then it will be removed. If you previously made some changes in it, and haven't clicked the **Apply** button, Netica will ask if you want to apply those changes first.

## Node Selector

You can switch which node a [table dialog box](#) is for, by using its pop-up menu labeled “Node”. From this menu you can choose any node in the net, and when you do, the contents of the dialog box will be completely re-adjusted for the new node.

If you have made changes in the dialog box before switching nodes, and have not yet pressed the **Apply** button, then a message will be presented asking if you want to apply the changes before switching nodes.

## Deterministic/Chance Selector

The [table dialog box](#) has a selector that allows you to make a nature node = 4  
&& typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_deterministic\_node.htm');return  
false;">deterministic or = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_chance\_node.htm');return false;">probabilistic  
(i.e. a chance node). If you use this selector before you have entered anything  
into the table, it will just modify the dialog box to be suitable for entering  
deterministic or probabilistic information. If you have already entered a  
relation, that table will be converted.

Converting a deterministic table to a probabilistic representation is simple:  
each row of the result will consist of all zeroes, except a single 100% entry at  
the state the original deterministic function mapped to.

Converting a probabilistic relationship to a deterministic one generally loses  
some information. For each parent configuration the deterministic value will  
be the state that was most likely in the probabilistic table (i.e. had the highest =  
4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_conditional\_probability.htm');return  
false;">conditional probability).

## Kinds of Tables

The table selector  allows choosing between the different types of [tables](#) that a node can have: a function table, conditional probability table (viewed with decimal or percentage numbers), experience table, unnormalized probability table and a counts table.

**Can't Set:** The only type of table that a deterministic node can have is a function table, so if the [deterministic selector](#) to the left of the table selector is set to "Deterministic", any choice of the table selector other than "Function" will just result in a beep and an explanatory message going to the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Messages_window.htm');return`  
`false;">Messages` window. If the left selector is set to "Chance" instead, then any choice from the right selector is okay, except "Function", which is then the one that results in a beep.

**Function Table:** The function table allows only a single output value for each possible set of parent values (i.e. a mathematical "function"). If the node is discrete, then there are a finite number of choices and Netica lets you enter them by clicking on a table cell and then choosing from a popup menu. If the node is continuous, not discretized (such as a utility node), then Netica lets you enter a real number in each cell.

**Conditional Probability Table (CPT):** These are the most common tables to work with, and are described in detail on the other pages of this chapter. They provide a probability for each state of the node, given the condition specified by the row (i.e. each parent node having some value), so the probabilities of each row must sum to one. You can enter them as decimal fractions or percentages, depending on whether the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_table_selector.htm');return false;">table selector` is set to **Probability** or **% Probability**.

**Experience Table:** = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_experience.htm');return false;">Experience`

tables provide one confidence number for each row of the table. Experience numbers must not be negative, and an experience of zero corresponds to an impossible condition (a row of Xs in the CPT). Experience numbers mean approximately the equivalent number of matching cases seen. Experience numbers are generated as the normalization factors of the "Unnormalized table", which in turn is the Counts table with a constant base experience added to each cell (usually 1).

**Counts Table:** These are the actual counts of matches during the learning process, so they are generally integers (e.g. there were 7 instances matching this table cell). However, they will have a fractional part if the case file had a "NumCases" column containing fractions, a learning degree other than 1 was used, the table was hardened or softened, a more advanced learning algorithm like EM or gradient descent was used or a fractional starting experience was entered before learning.

## Scrolling and Navigating

Since a node may have very many parent configurations, they often won't all be visible in the [table dialog box](#) at once. You can use the scroll bars to view whatever part of the list you are currently interested in. The `HOME`, `END`, `PAGE UP` and `PAGE DOWN` keys may be used as well.

**Resizing:** You can also resize the table dialog box to make it bigger or smaller by clicking in the box at the lower right corner, or on the window edge and dragging to the new size. When you first bring up the table dialog box, it is constructed with a size to match the tables required for the node its for. If you change which node the dialog box is for, it is sometimes convenient to resize the dialog box as well.

**Large Tables:** Tables for nodes with several parents may have a great number of rows, making it difficult to scroll to the row you want. By = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">`right-clicking on a state-name within a row, and choosing a new state name from the menu, the table will automatically scroll so that a new row replaces that same position. The new row will have all parents with the same state as the original row, except the state that was changed. By starting with the left-most column and setting a state for each column, any setting of parent values may be obtained, scrolling to reveal that row. You can change whether you want the table based on state names, state titles, or numeric levels by making a choice above the dividing line of the menu that appears when you right-click.

Remember, you can also change the [order of columns](#) if you want to.

## Empty Cells

If a cell of the [table dialog box](#) table is blank, it means that the value of the cell has not been specified. You may be building up a net over a number of sessions with Netica, and you can use the empty cells to keep track of which parts of the table have not yet been entered.

New tables start out with all empty cells, and you can set any cell or group of cells to empty by selecting the cells and then pressing the `DELETE` key or choosing **Edit** → **Delete**. If the node is deterministic, you can set a cell to empty by choosing “Unknown” from the pop-up menu which appears when you click on the cell.

If you do inference (e.g. `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_belief Updating.htm');return false;">belief updating`) with a net that has node tables with empty cells, Netica will consider them to be uniform, and give you a warning `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Messages_window.htm');return false;">message`. Netica can find and display all the nodes whose tables have one or more empty cells. ([more info](#))

## Cells Containing X

**Meaning:** A cell of the [table dialog box](#) table with just an X in it indicates that no value is required for the cell, because that configuration of parent values will never occur. It actually doesn't say anything about the relationship at hand, but just that whoever was building the Bayes net didn't feel it was necessary to enter a value because in the context of the overall net it is not required. If Netica is later doing inference and discovers that configuration of parent values can occur, it will alert you with a message in the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Messages_window.htm');return`  
`false;">messages window.`

**Importance:** If you put any arbitrary value in such a cell, all inference results will be the same, because the value will not be used. The best value to put in the cell is the true value that would be applicable if the relationship between the parents was different, but if that value doesn't exist, or you don't know it, or don't have time to determine it, it is much better to use the X feature than to just put in an arbitrary value, for three reasons. First, during inference Netica performs an important check for you as to whether the configuration is really impossible. Second, it documents your understanding at the time you were building the Bayes net for anyone else who later works with it. Third, if you later change other parts of the net, or you [copy and paste](#) this node into a different net, or put it in a net [fragment](#) library, those parent configurations may become possible.

**Setting:** You can set a cell to X by [selecting](#) the cell and then typing X (for a probabilistic table), or by choosing 'Impossible' from the cell's pop-up menu (for a deterministic table). For probabilistic tables, if one cell has an X, then all the cells in that row *must* have an X (because they all correspond to the same parent configuration).

## Getting CPTs from Text Files

If you want Netica to learn the CPTs based on data records in text files, see the [chapter on learning](#) instead; here we obtain actual probability numbers directly from text files.

There are a few ways to "import" entire CPTs to the network from files:

**1. Cut & Paste:** If the table is in tab-delimited form, with each row corresponding to the various states of the child node, you can just paste it into the table dialog box for the node. First, read the table with a text editing program (e.g. word processor, or choose **File** → **Open as Text**). Select the entire table and choose **Edit** → **Copy**.

Then open the table editor for the child node (**Table** → **View/Edit**), and select the upper left cell (by right-clicking it, or by left-clicking and dragging a tiny distance within it). The cell should turn black. Finally choose **Edit** → **Paste**, and the entire table will be filled.

Your table in the text file must use the same order of parents as in the table dialog box. If they aren't the same, you can [adjust the columns](#) beforehand just by clicking and dragging in its heading.

You can also copy and paste in a similar way from a spreadsheet program, such as Excel.

**2. Make a simulated "case file":** with 1 row for each entry of the CPT table (i.e. the [Cartesian product](#) of the parents and the child node itself). That means that if the child node has n states, there will be n times as many rows in your simulated case file as there are rows in the CPT table.

The first column of the case file should be [NumCases](#), and in that column you place the CPT probabilities.

Then choose **Cases** → **Learn** → **Incorp Case File**, and it will read the probabilities in as the frequency for each case. You will need to **Table** → **Harden** with a degree of 1 afterwards. Or, you can multiply all the probabilities in the case file by some large number (the same number for one), so it is as if you are presenting Netica with a set of cases whose probability distribution matches the CPT numbers.

**3. Use a .dne file:** Save the Bayes net as a .dne file (from the **Save As** dialog box, choose dne from the menu in the box, or make the file name end with .dne). Then open the .dne file with a text editor. It will be quite readable, and you should easily be able to find the table for the node in question. (if there is no table there, you may want to create a dummy table in Netica beforehand with **Table** → **Uniform**). Simply copy the whole table as text, and then paste to replace the table in the .dne file. The table in your text file must be comma-delimited, with each row corresponding to the states of the child node, and the rows in odometer order corresponding to the same order of parent nodes as is shown in the .dne file (you can re-order the parents as described above). Don't worry about the comments that were on each line of the .dne file (i.e., the parts starting with "//"); they aren't needed.

**4. Write a Program:** If you want it to be an automated process, you can write a program that reads the text file, and then does calls to [Netica API](#) to fill the CPT Tables.

## Selecting, Copying and Pasting Cells

To do operations on a subset of cells in the [table dialog box](#), you first select them, which will hilite them.

**Selecting:** To select cells of a numeric table, click down in one cell, and drag the mouse to another cell before releasing the mouse button. While you are moving the mouse, all the cells in the rectangle between the original cell and the current mouse position will be hilited. The cell you first click down in must not be the cell you are currently editing (i.e. the only selected cell, or the cell with the insertion point blinking in it) or Netica will think that you are doing an operation to edit just that cell. You can drag outside the window boundary to force auto scrolling. Once a selection is made, you can hold down the **SHIFT** key while you click on another cell to extend or reduce the selection.

To select whole rows at a time, click at the left edge of the row (just to the right of the double line) and drag up or down.

**Deterministic:** Selecting cells of a deterministic table is done in the same way, except only selection by rows is allowed (if you click down in a cell you will get a pop-up menu instead).

**Copy & Paste Multiple:** After you have selected some cells, you can copy their values to the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_clipboard.htm');return false;">` clipboard by choosing **Edit** → **Copy** (or pressing a [shortcut key](#), or clicking ). To place those values in some other cells, select the cells and choose **Edit** → **Paste** (or press **CTRL+V** or click ). If the paste region is larger than the copy region, the contents of the copy region will be repeated horizontally and/or vertically to fill the paste region (you will be given a notice if they are not an even integer multiple). If the paste region is smaller than the copy region, you will be given a notice and the truncated contents will be pasted. Instead of selecting the whole paste region, you may find it easier to just select the upper-leftmost cell of where you want the contents pasted.

**Copy & Paste Single:** Select a single cell by clicking down on it and slightly dragging the mouse-pointer within the cell (if you don't drag, Netica will instead prepare the cell for editing), or right click the cell. As with [nodes](#), you can cut, copy or paste single cells. Select the cell and choose a command from

the **Edit** menu, or press the equivalent shortcut key.

**Other Programs:** You can copy and paste back and forth between the Netica table and a spreadsheet, such as Excel, or a word processing program. If each row of probabilities being pasted doesn't exactly add up to one, then do a [normalize command](#) after pasting. Netica cells copied to the clipboard are entered as text, with a tab between each entry on the same line, and a carriage-return/line-feed pair at the end of every line. When copying from a word processor or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_editor.htm');return false;">text editor to Netica, the format should be the same, except space(s) and/or a comma may be substituted for each tab, and line-feed(s) may be substituted for carriage returns. Blank lines will be ignored.

**Whole Tables Including Names:** To paste whole CPT tables into Excel (or another program), make the Bayes net window = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">toggle on **Report** → **Tab Separators** and **Report** → **Copy to Clipboard** (and if desired, toggle off **Report** → **To Messages Window**). Then = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the node(s) you are interested in, choose **Report** → **CPT Tables**, click a cell in the Excel spreadsheet, and then press **CTRL+V** or **Edit** → **Paste**. You can control whether you want the names of the nodes and states included by toggling **Report** → **Include Names**.

## Table Menu Commands

While working in a [table dialog box](#), the **Table** menu is often useful. [Select](#) some cells or rows you wish to operate on, and then choose a command from the menu (if no cells are selected, the command is applied to all the cells).

**Table** → **Fill in Missing** enters a probability into empty cells so that the probabilities of each selected row add up to 1.0 (i.e. 100%). Any row having no empty cells, more than one empty cell, or [X cells](#) will simply be ignored.

The corresponding toolbar button is the one with the number entering the empty cell: .

**Table** → **Uniform Probabilities** will set the selected rows of a probability table to uniform probability vectors, which means that all states of the node are equally probable. So all the cells of these rows will contain the number  $1/(\text{number of states})$ , expressed as a percentage.

**Table** → **Randomize** works on probabilistic or deterministic tables. For deterministic tables it sets each selected cell to one picked randomly from the possible states of the node, following a uniform distribution (i.e. each state equally probable). For probabilistic tables it sets the cells of each selected row to randomly selected probabilities. Of course, the probabilities of each row will add to 1.0 (i.e. 100%). The distribution of the probabilities entered is *not* uniform (it favors numbers closer to 0 in an attempt to better match realistic distributions). The corresponding toolbar button has one die over the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`

`BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_relation_symbol.htm');return false;">relation symbol:` 

**Table** → **Remove** will convert all the selected cells to [empty cells](#). If there are selected cells, then pressing the `DELETE` key will have the same effect. The toolbar button consists of the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_relation_symbol.htm');return false;">relation symbol` covered by a red X: 

**Table** → **Build From Other Net** will build the cpts of your current net based on the cpts of another net. [More info](#)

Many of these commands can also be done from the net window, without using any table dialog box, by selecting the nodes of interest, and then choosing the menu command from the **Table** menu. It will apply to the entire table of all the selected nodes.

## Changing Column Order

You can use the [table dialog box](#) of a node to change the ordering of its states, and of its parents. This can be very useful for viewing and editing the tables, and Netica will make all required re-arrangements to tables to ensure that inference results will not be effected.

**How To:** First, bring up the table dialog box for a node (e.g. by = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">`right-clicking on it, and choosing **Table**). Click down on the name of the state or parent node that you wish to move in the gray bar of column titles. The mouse pointer will turn into a double arrow as you hold down the left mouse button. You can drag the column title to its new position, then release the mouse button.

When you change the order of the node's states in the dialog box, that change immediately effects the node in the net, without pressing the **Apply** or **Okay** button. To reverse the effect of the re-ordering, make the net window = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_active_window.htm');return false;">`active, and perform a regular "Undo", for example by choosing **Edit** → **Undo** from the menu.

When you change the order of the node's parents, the structure of the table will be suitably changed within the table dialog box. The new arrangement of probabilities can sometimes provide useful insights. If you are editing the table, the new arrangement of rows may make it easier to select a desired range of rows for some operation, or to copy and paste ranges of rows. When you press the **Apply** or **Okay** button, the new ordering will be transferred to the node in the net, so that in the future when you bring up the table dialog, it will have that ordering. As usual, you can always "undo" from the net window.

Since each table dialog box maintains its own parent ordering, you can have [more than one](#) dialog box open at a time for a single node, with a different parent ordering in each one, and work and view interchangeably between

them. In that case the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_changed\_indicator.htm');return false;">changed indicator can be very useful for keeping track of what is going on.

## Multiple Dialog Boxes for One Node

You may have more than one [table dialog box](#) (or [node dialog box](#)) for the same node. This can be confusing if you are not careful, so it is not recommended unless you need it (Netica will ask you beforehand).

However, in some situations it is very useful. Each dialog box stores all the settings for its node and doesn't lose them when you make changes in another node dialog box or to the node itself (unless you press the **Reset** button). So you can have several different settings for a node, one set in each dialog box, and alternate between them just by pressing the **Apply** button on the dialog box of choice. Of course you can do all kinds of other operations in between (such as [compiling the net](#), [saving](#) it to a file, making queries, undoing an operation, etc.).

You open multiple dialog boxes for a node in the same way as you open a single dialog box. Just do the action repetitively without closing the previous dialog boxes first, and answer 'no' to the question of whether you want to bring the existing dialog box to the front.

## Cases

The set of all = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings entered into the nodes of a single Bayes net is referred to as a *case*. A case usually provides some information about a particular object, person, event, thing, etc.

Netica has facilities for [working with cases](#), including the ability to [save a case](#) (i.e. all the current positive findings) from a Bayes net to a file. Later those findings can be [re-entered](#) into the Bayes net by reading the file.

Case files may consist of many cases (acting as a database, in which each case is a database *record*). You can use Netica to [learn](#) the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs of a Bayes net from such a file of cases. Netica can also [generate a file of cases](#) which match the probability distribution of a Bayes net, while taking account of findings currently entered. There are a number of [other ways](#) to create multi-case files.

Netica can pass through a file of cases, applying Bayes net inference to each case to generate new information about it, and then saving the case with the additional information to a new case file, which is known as [processing cases](#).

It can also use a case file to test the performance of a Bayes net, finding = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_error\_rate.htm');return false;">error rates, log loss, etc., which is known as [testing a net with cases](#).

**Example 1:** In a medical example, each case might correspond to a certain patient. When you want to work with a new patient, you save all the information gathered for the first patient into a case file before removing it from the net, perhaps using the patient's name as a file name. When it comes time to reconsider the first patient - perhaps some lab results have arrived - you just read that person's case file.

**Example 2:** For another example, each case could be a political riding. The findings would be details about that riding (such as demographics, poll

statistics, etc.), and the Bayes net could be used to predict the percentage of votes each political party will get in the next election. As new information about a riding arrived, its case file would be kept updated.

## Working with Cases

**Entering:** To enter a [case](#) in a Bayes net, simply [enter](#) each of its findings.

**Removing:** To remove the current case from a belief net, make sure no nodes

are = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_deselect\_nodes.htm');return false;">selected,

and choose **Cases** → **Remove Findings**, or click the  [toolbar button](#). It will

leave the net with no findings entered, and if it is an = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_auto\_update.htm');return false;">auto-updating

net, then its beliefs will be updated to reflect that. If some nodes are selected

when you choose **Remove Findings**, then only the findings for those nodes

will be = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_retract\_finding.htm');return false;">retracted.

**Saving:** You can [save cases](#) in files and read them back in.

**Reporting:** To create a text table showing the current case, use **Report** →

**Findings**. Using other choices on the **Report** menu you can control its

[appearance](#) and [destination](#). To locate all the nodes that have any finding, or

just a likelihood finding, you can [select](#) them.

## Saving and Reading Cases

**Saving:** To save in a file all the findings from the active net, make sure no nodes are selected, choose **Cases** → **Save Case As**, or click , and then enter the file name. If some nodes are selected, then only the findings for the selected nodes will be saved, and Netica will beep and put a notice in the Messages window that the whole case wasn't saved. Later, if you make changes to the case you can choose **Cases** → **Save Case**.

**Saving Multi-case:** Currently Netica is unable to save a case to a multi-case file, but there are other options for [creating multi-case files](#).

**Reading:** To later read the case back into its original net, make that net active, ensure that no nodes are selected, choose **Cases** → **Get Case** or click  and select the case file from the standard dialog box which appears. Any existing findings in the net will be removed, the file will be read, and the findings entered into the net. If the net is auto-updating, then belief updating will be done automatically to account for the findings of

the case.

If some nodes are selected when you choose **Cases** → **Get Case**, then Netica only removes findings and reads new ones for the selected nodes (it also beeps and puts a notice in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window that the whole case wasn't read).

You can also read a case from the [command line](#).

**Reading Multi-case:** If you read from a file with more than one case in it, then Netica will ask you which case you want. Pressing `SHIFT+F8` gets the previous case. If the case file stores [IDnums](#) then Netica will ask for one, otherwise it will ask for the position of the case.

Pressing the `F8` key generally produces the same result as choosing **Cases** → **Get Case** from the menu, except if a case has just been read from a multi-case file. Then `F8` will automatically get the next case without opening any dialog boxes, which makes it convenient to browse the cases one by one. Netica prints the case's number in the Messages window.

**Different Net:** It is possible to read a case into a different net than the one it was originally saved from. Findings from nodes of the old net will be entered into nodes of the same [name](#) in the new net (the titles of the nodes are ignored). The state names of the nodes (if present) should also be the same. Any findings not corresponding to a node in the new net will simply be ignored. All name comparisons are case-sensitive.

**Numeric Values:** Real number values that you have = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered as findings for = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretized continuous nodes are saved in case files, rather than just the state they correspond to. This enables reading the case into another net whose node for that variable has been discretized in a different way.

**Missing State:** If you read in a case, and the case file has a value that isn't

any of the states of the corresponding node in the net, then an error message will be displayed. For example, if node 'color' has the states 'red' and 'green', and in the case file the value for color is 'blue', the message will be displayed.

An exception to this occurs if one of the states of the net node is named 'other'. Then the case will be read without error, and the finding for the node will be 'other'. ([more info](#))

# Creating Case Files

If you want to create a case file containing a single [case](#), you can just enter the case as findings into a Bayes net, and then [save it to file](#) from Netica. To create a file containing many cases, there are several options.

**Word Processor:** One possibility is to use a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_text_editor.htm');return false;">text editor` or a word processing program, and manually construct it according to the [CASE-1 format](#). Be sure to save it as a “Text Only” file if you use a word processing program.

**Spreadsheet:** If the data is in a spreadsheet program such as Excel, usually you can just copy the data from the spreadsheet to the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_clipboard.htm');return false;">clipboard`, and then paste it into the text editor window, and it will come out in the required format. You will have to add the two header lines described in the [CASE-1 format](#). Or you can export it from the spreadsheet as text. In that case you may only have to add the “// ~ >[CASE 1] >~” header line. Alternatively, you can [learn from Excel](#).

**Database:** Most database programs have an option to export data in the form of “flat files” of text. Such files are suitable as case files, with the addition of the header lines described above.

**Simulation:** If you wish to create a multi-case file consisting of random cases sampled from the probability distribution represented by a Bayes net, Netica can [make it](#) automatically.

**Programming:** Another way to create a multi-case file is to write a computer program that creates it. Using the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Netica_API.htm');return false;">Netica API Programmer’s Library` is especially useful for this purpose, since it can create such a file with just a few function calls. It can also be used to update and

otherwise maintain case files, and to read them and do Bayes net learning and inference using them.

## Case File Format

**Structure:** Case files (single-case or multi-case) are pure ASCII = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_file.htm');return false;">text files. They may contain “// ~ >[CASE 1] >~” or a time-author stamp, somewhere in the first 3 lines, but that is not normally present. Then comes a line consisting of headings for the columns. Each heading corresponds to one variable of the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case.htm');return false;">case, and is the name of the node used to represent the variable (sometimes the variables are called *attributes* and the entries in the column *values*, i.e. *attribute-value*). The headings are separated by spaces and/or tabs (it doesn't matter how many).

**There should be no spaces in the names of the nodes.**

The case data is next, with one case per line (a single-case file only has one such line). The values of the variables are in the same order as the heading line, and are separated by spaces or tabs (the columns don't have to “line up” as they do in the examples below).

**Discrete:** The value of a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete.htm');return false;">discrete variable is given by its state name, or by its state number preceded by a ‘#’ character (the first state is #0). Using the state names is preferred, since the order of the states may be changed sometime, and that would render a file with state numbers invalid. The ‘#’ symbol is recommended, but may be omitted if the node has no discretization or values defined.

**Continuous:** The value of a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous variable is given by a number in integer, decimal, or scientific notation (e.g. -3.21e-7). If it has been = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretized, then

the value may be given by a state name or state number instead, but the continuous number is preferred if it is available. That way the case file can be used for different discretizations of that variable in the future. It is best if the value has the correct number of significant figures, since future versions of Netica may use this information.

**Missing:** If the values of some of the variables are unknown for some of the cases, then an asterisk \* is put in the file instead of the value. This is known as “missing data”. When reading case files, Netica can also understand a question mark ? used for missing data.

**Uncertain or Negative:** Negative, interval, Gaussian, set, etc findings can also be entered in a case file using the [UVF format](#).

**Comments:** There may be as many spaces or tabs at the end of a line as desired, and there may also be C / C++ / Java style comments (e.g. a double slash “//”, followed by any text).

**IDnum:** There are two special columns that a file may have which don’t correspond to nodes. One provides an identification number for each case, which must be an integer between 0 and 2 billion. The heading for this column is “IDnum”. Identification numbers do not have to be in order through the file. The missing data symbol \* must not appear in this column.

**NumCases:** The other special column has the heading “NumCases”, and indicates the frequency or multiplicity of the case. A multiplicity of  $m$  indicates  $m$  cases with the same variable values. It is not required to be an integer, so it can be used to represent a frequency of occurrence if desired. The missing data symbol \* must not appear in this column either.

**Examples:** [Here](#) is a listing of “Chest Clinic.cases”. It involves only discrete nodes with state names, and has an IDnum column, but no frequency column. [Here](#) is another example of a case file, this time for cars brought into a garage. It has discrete and continuous variables, state numbers and state names, and asterisks for missing entries.

**Future:** Future versions of Netica will support more advanced operations with cases, including a more efficient file representation, and a way of using Bayes nets as “indexing functions” to do the kind of lookup common in case-based reasoning. However, the above described type of file format will always be supported as well.



# Case Files with Uncertain Values – UVF Format

The [case files](#) discussed in previous pages have only had values that were completely certain (or completely missing). But Netica can also create and read case files having values that are known with limited accuracy, or only known to within some likelihood. In fact, Netica has a very elegant, practical and powerful way of expressing uncertain findings, called the *UVF format*.

When Netica reads in a case containing uncertain findings (for example, by choosing **Cases** → **Get Case**), it will enter them in the Bayes net as likelihood findings, so any probabilistic inference, node absorption, sensitivity analysis, etc. will properly account for them. Also, the operations on case files, such as learning from cases, test net with cases and process cases, will work properly on case files containing uncertain values. When learning from such cases, some learning algorithms will work better than others. For more information on that, and an example of working with case files having uncertain findings, see the [learning algorithms](#) page.

Below is a list of the different types of uncertain values, their syntax in the case file, and what they mean. Each type of uncertain value can appear anywhere in a case file where a regular value normally would. For example, a case file could be a regular `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CSV_file.htm');return false;">CSV file`, or `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_tab_delimited_file.htm');return false;">tab delimited text file`, but with some of the values replaced with entries having the syntax described below.

## Gaussian

**Syntax:** `m+-s` m and s are real numbers

**Examples:** `5+-2` `3.27+-0.03` `0+-1e-5`

This is for a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_normal_distribution.htm');return false;">Gaussian` (also known as “normal”) likelihood finding, where the m is

the mean and  $s$  is the standard deviation. Note that there cannot be any space before or after the  $\pm$ . The uncertainties in measurements from lab instruments, or polling results, are often expressed with a  $\pm$  notation, and indicate a Gaussian distribution, so they can now be easily input into Netica (although sometimes they may mean an interval distribution, as described below).

## Interval

**Syntax:** `[a, b]`  $a$  and  $b$  are real numbers, state names or indexes preceded by  $\#$

**Examples:** `[0, 10]` `[-3, 2.27]` `[lo, med]` `[#1, #3]`

There may be spaces before or after the comma or brackets. Intervals of states include both endpoints, so `[lo, med]` includes states `lo`, `med` and any states between. Intervals of numbers include the lower endpoint, but not the upper endpoint, so `[0, 10]` for variable  $X$  means  $0 \leq X < 10$ . Likelihood within the interval is one; outside the interval it is zero.

## Unbounded Interval

**Syntax:** `>m` or `<m`  $m$  is a real number, state name or state index preceded by  $\#$

**Examples:** `>4.75` `<-10` `<med` `>#2`

When  $m$  is a state, the interval includes the endpoint, and when it is a real number, the interval includes the endpoint only for `>` intervals (so `>` is really  $\geq$ ). The interval can potentially extend to infinity, but in practice will probably be limited by known maximum or minimum values for the variable. Likelihood within the interval is one; outside the interval it is zero.

## Set of Possibilities

**Syntax:** `{s1, s2, ... sn}` each  $s_i$  is a state name, state index preceded by  $\#$ , interval, unbounded interval, or Gaussian.

**Examples:** `{lo, med}` `{red, blue, green}`  
`{#1, #5, #7}` `{[0,3.5], [4.5,10]}`  
`{[#35,#122], >#500}`

There may be spaces before or after the comma or braces. The value can be considered to be a disjunction of the elements (e.g.  $X=\text{red}$  **or**  $X=\text{blue}$  **or**  $X=\text{green}$ ). The likelihood of elements in the set is one; of those not in the

set, it is zero.

## Set of Impossibilities

**Syntax:**  $\sim\{s_1, s_2, \dots, s_n\}$  each  $s_i$  is a state name, state index preceded by #, interval or unbounded interval

**Examples:**  $\sim\{lo\}$                      $\sim\{red, blue, green\}$   
 $\sim\{\#1, \#5, \#7\}$              $\sim\{[0, 3.5]\}$

There may be spaces before or after the comma or braces, but not between the tilde ( $\sim$ ) and the brace. This is the same as "Set of Possibilities" except the "possible" states are those that are *not* listed, rather than those that are listed. The likelihood of elements in the set is zero; of those not in the set, it is one.

`A = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_negative_finding.htm');return false;">negative  
finding can be represented easily by just listing the state(s) eliminated by the  
observation.`

## Likelihood

**Syntax:**  $\{s_1 p_1, s_2 p_2, \dots, s_n p_n\}$  each  $s_i$  is a state name, state index preceded by #, interval, unbounded interval, or Gaussian. Each  $p_i$  is a number between 0 and 1. Some  $p_i$  may be absent.

**Examples:**  $\{female .8, male .3\}$      $\{3+-1 0.2, 7+-2 0.4\}$   
 $\{[0,1.5] .5, [1.5,5] 0.1, [5,10] 0.02\}$

This is the same as a set of possibilities, but each possibility is weighted with a likelihood that appears after it (separated by a single space). The most common kind of likelihood vectors are for discrete variables, where each state is listed, followed by its probability. Any states that appear without a probability have a likelihood of 1, and any states that don't appear at all have a likelihood of 0.

Arbitrary likelihood distributions for continuous variables can be formed by a series of adjacent intervals, each with its own probability. Or the elements can overlap, and then their likelihoods are combined. For example  $\{[0,10] .1, [2,4] .2\}$  would be the combination of a [rect](#) function extending from 0 to 10

with height 0.1, and another rect from 2 to 4 with a height of 0.2.

Another useful distribution that is easy to form is the weighted combination of Gaussians. For example  $\{3+1\ 0.2, 7+2\ 0.4\}$  is a bi-modal distribution with peaks at 3 and 7.

It is possible to mix weighted Gaussians, intervals, and discrete states within a single  $\{ \dots \}$  likelihood vector.

## Negative Likelihood

**Syntax:**  $\sim\{s_1\ p_1, s_2\ p_2, \dots s_n\ p_n\}$  each  $s_i$  is a state name, state index preceded by #, interval, or unbounded interval. Each  $p_i$  is a positive number.

Some  $p_i$  may be absent.

**Examples:**  $\sim\{\text{red, green, teal}\ .2, \text{olive}\ .8\}$   
 $\sim\{[0,2]\ .4, [2,6]\ .2\}$

The same as a set of impossibilities, but each entry is weighted with a likelihood, which appears after it. If no number appears after it, its likelihood is 0. Entries that have numbers above 1 are indicated to be more probable than those not listed, and entries with numbers below 1 are less probable than the unlisted ones (unlisted entries have a likelihood of 1).

## Complete Uncertainty

**Syntax:**  $*$  [i.e. the syntax is just an asterisk]

If nothing is known regarding the value of this variable (i.e. `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_missing_data.htm');return false;">missing data`), then a question mark `?` or an asterisk `*` should be used to indicate that. It is equivalent to  $\sim\{ \}$  which is a likelihood of all ones.

## Example Case File - Chest Clinic

As an example of a [case file](#), here is a listing of “Chest Clinic.cases” which was produced by the [Simulating Random Cases](#) example. It involves only = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete\_node.htm');return false;">discrete nodes with state names, and has an IDnum column, but no frequency column. ([Another example](#))

```
// ~->[CASE-1]->~
```

IDnum	VisitAsia	Tuberculosis	Smoking	Cancer	TbOrCa
1	No_Visit	Present	Smoker	Absent	True
2	No_Visit	Absent	Smoker	Absent	False
3	No_Visit	Absent	Smoker	Present	True
4	No_Visit	Absent	NonSmoker	Absent	False
5	No_Visit	Absent	Smoker	Present	True
6	No_Visit	Absent	Smoker	Absent	False
...					
119	No_Visit	Absent	Smoker	Absent	False
120	No_Visit	Absent	Smoker	Present	True

## Example Case File - Car Diagnosis

Here is an example of a [case file](#) for cars brought into a garage. Notice BatAge, which is a continuous variable, Lights which are supplied by state numbers instead of names, and the asterisks for missing entries:

```
// ~->[CASE-1]->~
```

Starts	BatAge	Cranks	Lights	StMotor	SpPlug	MFuse	Alter	BatVolt	Dist	PlugVolt
False	5.9	False	#0	*	fouled	okay	*	dead	*	*
False	1.3	False	#0	*	okay	okay	*	dead	*	none
False	5.2	False	#0	Okay	okay	okay	Okay	dead	Okay	none
True	4.1	True	#2	*	okay	okay	*	strong	Okay	strong
True	2.7	*	#2	*	wide	okay	*	strong	Okay	*
*	*	True	#2	*	fouled	okay	*	*	Okay	strong
False	1.7	True	#0	Okay	okay	okay	Okay	dead	*	none
True	2.9	True	#2	*	*	*	*	strong	Okay	strong

([Another example](#))

## Simulating Random Cases

You can use Netica to generate a series of random cases whose probability distribution matches that of a particular Bayes net, which is known as *simulation* (sometimes called *sampling*). These cases can be used as example scenarios of what one should expect if the Bayes net matches reality. Or they can be manipulated and combined with other cases, and then used to learn a new net.

The sampling algorithms used are precise, so that the long-range frequencies of the cases will exactly approach the probabilities of the Bayes net, while taking account of all findings currently entered.

The cases will be stored in a file whose format matches the specification of a [case file](#). Once Netica has made the case file, you can [browse](#) it with the `F8` key to see the individual cases.

**How To:** To generate a case file for the active Bayes net, click `active` `typesof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_active_window.htm');return false;">` `compile it,` `typesof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_compile_net.htm');return false;">` `select the` `nodes for which you wish to have values in the case file, and then choose` **Cases** → **Simulate Cases**. All the nodes of the net will be used to generate the cases, but columns will only be made for the selected ones. You will be queried for how many cases to generate, the file name for the case file, where to put it, and how much `typesof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_select_node.htm');return false;">` `missing data` you want. Normally you will enter 0 for the amount of missing data, but if

you want to have a case file with asterisks for some fraction of the fields, enter that fraction (e.g. entering 0.25 means 25% of the values will be missing). If you wish to generate only a single random case (and not save to file), choose **Cases → Random Case**.

**Example:** As an example, if you do a **Cases → Simulate Cases** command with 'Chest Clinic.dne' from the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">Examples folder, and enter 120, "Chest Clinic.cases" and 0 to the dialog boxes, then you will obtain a case file similar to [this](#) (the case file you obtain may be a little different, since random numbers are involved).

**With Equations:** If one or more nodes have an [equation](#) to define the relation between a node and its parents, then you may want Netica to use those equations directly to [generate the random cases](#), instead of the [probability tables](#) which approximate the equations. In that case, don't compile the net before doing **Cases → Simulate Cases**. The sampling process will be slow if the net has an unlikely set of findings entered (a rejection method is used). In the case file generated, continuous variables (whether or not they have been = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretized) will have as values their continuous real number for each case, not just a state representing a range of values.

## Process Cases

Netica can process a file of cases. For each case in the file, Netica reads the case and enters it as findings into a Bayes net. Then Netica does = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">belief updating to find probabilities for all the nodes that didn't have findings.

Finally Netica writes the results to an output file.

**Note:** If the purpose is to grade the performance of the net, based on the cases, it is simpler to use [Test With Cases](#).

**Note:** If more control is needed, you may want to program Netica through its [COM interface](#) using Visual Basic, Java or C/C++.

**How To:** First, open the Bayes net you wish to use. Then choose **Cases** → **Process Cases** from the menu. The standard dialog box for opening a file will appear. From it choose the *control file* (described below) that you wish to use. When you click **Okay**, a new dialog box for opening case files will appear, from which you choose the case file to be processed. After you click **Okay**, the dialog box for saving a file will appear, in which you enter the name of the file you want the results written to.

Netica will then proceed to process all the cases, printing the fraction completed in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window (if that isn't obscured by the Bayes net window). If you want to halt processing before it is completed, hold down the **CTRL** key and press the left mouse button.

If there are any findings entered into the network before processing starts, those findings will be used for all belief updating, even overriding findings found in the case file. If an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered finding is going to override case file findings, you will be warned and asked if you want to = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

`BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_retract_finding.htm');return false;">retract all findings first.`

**Control File:** The *control file* is a text file which you can create by choosing **File** → **New** → **Text Edit**. You enter in this file what you wish to appear in each column of the output file (each row of the output file is for the results of one case). The choices for columns are:

IDnum()  
freq()  
finding (<node>)  
caseprob()  
bel (<node>, <state>)  
util (<node>, <state>)  
belvec (<node>)  
utilvec (<node>)  
mostprob (<node>)  
expval (<node>)  
best(<node>)  
stddev (<node>)

where <node> should be replaced with the name of a node (not its title), and <state> should be replaced with a state name for that node, or a # symbol followed by the state number (e.g., #0).

`IDnum()` and `freq()` transfer the IDnum and freq values from the case file to the output file. `finding (<node>)` transfers the finding from the case file if there is one, otherwise it places the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_missing\_data.htm');return false;">missing data symbol.

`caseprob()` displays the joint probability of all the findings of the case taken together (and including any findings entered directly in the net before processing was started).

`bel (<node>, <state>)` displays the belief that the value of <node> is <state>. In other words, it puts  $P(\text{<node>} = \text{<state>} \mid \text{findings})$ , where findings are

from the case, and those directly entered in the net before processing was started.

`mostprob (<node>)` provides the most probable state for <node>.

`util (<node>, <state>)` is for a decision node, and it displays the expected utility of making decision <state>.

`best (<node>)` provides the best decision for <node>, if that is available.

`belvec (<node>)` displays a list of numbers in parenthesis, each separated by a space, that are the beliefs for each of the states of <node>.

`utilvec (<node>)` is for a decision node, and it displays the expected utility of each decision of <node> in a parenthesized list.

`expval (<node>)` displays the expected value (i.e. mean value) of <node>, and `stddev (<node>)` puts the standard deviation of <node>. These may only be used for continuous variables, or for nodes representing discrete variables which have a real number value assigned to each state (this is done using the node dialog box).

**Example:** Here is an example control file:

```
IDnum()
bel (Color, red)
bel (Color, blue)
bel (Color, green)
expval (Cost)
```

and here is the output file it created:

IDnum	P(Color=red)	P(Color=blue)	P(Color=green)	E[Cost]
1	0.000167195	0.0117262	0.988107	6.86929
2	0.422277	0.0130726	0.56465	3
3	0.610178	0.0203665	0.369455	3
4	0.324446	0.0163193	0.659235	5.94723
5	0.000381718	0.0893132	0.910305	3

## Test Net Using Cases

The purpose of this test is to grade a Bayes net using a set of real = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_case.htm');return false;">`cases to see how well the predictions or diagnosis of the net match the actual cases. It is not for = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_decision_nets.htm');return false;">`decision nets.

First = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_select_node.htm');return false;">`select the nodes you do not wish the net to know the value of during its inference. For example, if the net is for medical diagnosis, you might select the disease node and nodes representing other unobservable internal states. These nodes are called *unobserved* nodes.

Then choose **Cases** → **Test with Cases**. You will be asked which [case file](#) to use, and after you choose one, Netica will start processing. The = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Messages_window.htm');return`  
`false;">`Messages window will come to the front and display the fraction of cases processed so far. Hold down `CTRL+SHIFT+LEFT BUTTON` at the same time if you want to stop processing cases and print the results obtained so far.

Netica will pass through the case file, [processing cases](#) one-by-one. For each case, Netica reads in the case, except for any findings for the unobserved nodes. It then does belief updating to generate beliefs for each of the unobserved nodes. It goes back and checks the true value for those nodes as supplied by the case file (if they are supplied for that case), and compares them with the beliefs it generated. It accumulates all the comparisons into summary statistics.

When Netica is done, it will print a report for each of the unobserved nodes.

These reports include a [confusion matrix](#), = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`

BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_error\_rate.htm');return false;">error rate,  
[calibration table](#), quadratic (Brier) score, [logarithmic loss score](#), spherical  
payoff score, [surprise indexes](#), and = 4 && typeof(BSPSPopupOnMouseOver)  
== 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_test\_sensitivity.htm');return false;">test  
sensitivity. For = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_binary\_node.htm');return false;">binary nodes  
it also reports = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_test\_specificity.htm');return false;">test  
specificity, predictive value and predictive value negative. For an easy way to  
produce receiver operating characteristic (ROC) plots, see [Quality of Test](#)  
topic.

**For full documentation on this function, and the reports generated, see the [Test Net with Cases](#) Chapter in Special Topics.**

## Netica's Learning

You may want to read the [introduction](#) to Bayes net learning from cases.

Netica can [learn from a file of cases](#), or it can [learn from cases one-by-one](#) as you enter them. It can also connect directly with a database, or [learn from cases in Excel](#).

The [way that Netica learns](#) relies on the concept of [experience](#).

If you are learning from a world that is constantly changing, and you want the net to adapt, [fading](#) may be useful.

After learning from some cases (or perhaps manually constructing a net) you may want to [test its performance](#) using another set of cases.

Netica [versions 5.0 and later](#) allow for basic [structure learning](#).

## Learning from Case Data

Bayes net *learning* is the process of automatically determining a representative Bayes net given data in the form of = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_case.htm');return false;">cases (called the
training cases). Each case represents an example, event, object or situation in
the world (presumably that exists or has occurred), and the case supplies
values for a set of variables which describes the event, object, etc, as specified
in the previous chapter. Each variable will become a node in the learned net
(unless you want to ignore some of them), and the possible values of that
variable will become the node's states. Learning from cases data results in = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_probability_revision.htm');return
false;">probability revision.
```

The learned net can be used to analyze a new case which comes from the same (or appropriately similar) world as the training cases did. Typically the new case will provide values for only some of the variables. These are = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_enter_finding.htm');return false;">entered as
findings, and then Netica does probabilistic inference to determine beliefs for
the values of the rest of the variables for that case. Sometimes we aren't
interested in values for all the rest of the variables, but only some of them, and
we call the nodes that correspond to these variables target nodes. If the links
of the net correspond to a causal structure, and the = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_query_node.htm');return false;">target nodes
are ancestors of the nodes with = 4 && typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_findings_node.htm');return false;">findings,
then you could say that the net has learned to do diagnosis. If the target nodes
are descendants, then the net has learned to do prediction, and if the target
```

node corresponds to a "class" variable, then the net has learned to do classification. Of course the same net could do all three, even at the same time.

The Bayes net learning task has traditionally been divided into two parts: structure learning and parameter learning. *Structure learning* determines the dependence and independence of variables and suggests a direction of causation, in other words, the placement of the links in the net. *Parameter learning* determines the conditional probability table (= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parameter\_learning.htm');return false;">parameter learning. *Structure learning* determines the dependence and independence of variables and suggests a direction of causation, in other words, the placement of the links in the net. *Parameter learning* determines the conditional probability table (= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT) at each node, given the link structures and the data.

You might not want Netica to learn the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs of all the nodes in your Bayes net. Some of the nodes may have CPTs that have already been learned well, were created manually by an expert, or are based on theoretical knowledge of the problem at hand (perhaps expressed by an equation). Netica allows you to restrict the learning process to a subset of the nodes, and those nodes are called the *learning nodes*.

If every case supplies a value with certainty for each of the variables, then the learning process is greatly simplified. If not, there are varying degrees of partial information:

If there is a variable for which none of the cases have any information, that variable is known as a *latent variable* or "hidden variable".

If some cases have values for a certain variable, and others don't, that is known as *missing data*.

Some values for variables may not be given with certainty, but only as *likelihood findings*.

It may seem strange to be learning a net that has latent variables, since none of the training cases have any information on them. You introduce a latent

variable as a parent node (or intermediate node) of multiple child nodes, and Netica uses the correlations among the children to determine relationships between the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_latent\_node.htm');return false;">latent node with others. The result may be a Bayes net that is actually simpler (has fewer CPT entries), and generalizes better (i.e. performs better on new cases seen).

For an example of using Netica to learn a latent variable, see the “Learn Latent.dne” net in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">Examples folder of Netica Application distribution, or get it from the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Norsys.htm');return false;">Norsys net library.

[More info on Netica's learning.](#)

[More info on Learning Algorithms.](#)

## Learning Algorithms

There are three main types of algorithms that Netica can use to [learn](#) CPTs: [counting](#), expectation-maximization (EM) and gradient descent. Of the three, “counting” is by far the fastest and simplest, and should be used whenever it can. It can be used whenever there are no

```
4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_latent_node.htm');return false;">latent
variables, and not much
4 && typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_missing_data.htm');return false;">missing data
or uncertain findings for the learning nodes or their parents. When learning
the
4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT of a node by
counting, Netica will only use those cases which supply a definite value for the
node and all of its parents.
```

If you can't use counting, then you must use EM learning or gradient descent. For each application area, it is usually best to try each one to see which gives the better results. Generally speaking, EM learning is more robust (i.e. gives good results in wide variety of situations), but sometimes gradient descent is faster. For all three algorithms, the order of the cases doesn't matter.

During Bayes net learning, we are trying to find the *maximum likelihood* Bayes net, which is the net that is the most likely given the data. If  $N$  is the net and  $D$  is the data, we are looking for the  $N$  which gives the highest  $P(N|D)$ .

Using Bayes rule,  $P(N|D) = P(D|N) P(N) / P(D)$ . Since  $P(D)$  will be the same for all the candidate nets, we are trying to maximize  $P(D|N) P(N)$ , which is the same as maximizing its logarithm:  **$\log(P(D|N)) + \log(P(N))$** . Below we consider each of the two terms of this equation. The more data you have, the more important the first term will be compared to the second.

There are different approaches to dealing with the second term  **$\log(P(N))$** , which is the prior probability of each net (i.e. how likely you think each net is before seeing any data). One approach is to say that each net is equally likely, in which case the term can simply be ignored, since it will contribute the same

amount for each candidate net. Another is to penalize complex nets by saying they are less likely (which is of more value when doing structure learning).

Netica bases the prior probability of each net on the experience and probability tables that exist in the net before learning starts, which appears to be a unique and elegant approach. If the net has not been given any such tables, then Netica considers all candidate nets equally likely before seeing any data.

The first term  $\log(P(D|N))$  is known as the net's *log likelihood*. If the data  $D$  consists of the  $n$  independent cases  $d_1, d_2, \dots, d_n$ , then the log likelihood is:

$\log(P(D|N)) = \log(P(d_1|N) P(d_2|N) \dots P(d_n|N)) = \log(P(d_1|N)) + \log(P(d_2|N)) + \dots + \log(P(d_n|N))$ . Each of the  $\log(P(d_i|N))$  terms is easy to calculate, since

the case is simply entered into the net as = 4 &&

`typeof(BSPSPopupOnMouseOver) == 'function')`

`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`

`onclick="BSSCPopup('X_PU_finding.htm');return false;">findings, and`

Netica's regular inference is used to determine the probability of the findings.

Both EM and gradient descent learning work by an iterative process, in which Netica starts with a candidate net, reports its log likelihood, then processes the entire case set with it to find a better net. By the nature of each algorithm the log likelihood of the new net is always as good as or better than the previous.

That process is repeated until the log likelihood numbers are no longer improving enough (according to a tolerance that you can specify), or the desired number of iterations has been reached (also a quantity you can specify). Netica uses a conjugate gradient descent, which performs much better than simple gradient descent.

**References:** To understand how each algorithm works, it is best to consult a reference, such as [Korb&Nicholson04](#), [Russell&Norvig95](#) or [Neapolitan04](#).

Briefly, EM learning repeatedly takes a Bayes net and uses it to find a better one by doing an expectation (E) step followed by a maximization (M) step. In the E step, it uses regular Bayes net inference with the existing Bayes net to compute the expected value of all the missing data, and then the M step finds the maximum likelihood Bayes net given the now extended data (i.e. original data plus expected value of missing data). Gradient descent learning searches the space of Bayes net parameters by using the negative log likelihood as an objective function it is trying to minimize. Given a Bayes net, it can find a

better one by using Bayes net inference to calculate the direction of steepest gradient to know how to change the parameters (i.e. CPTs) to go in the steepest direction of the gradient (i.e. maximum improvement). Actually, it uses a much more efficient approach than always taking the steepest path, by taking into account its previous path, which is why it's called *conjugate* gradient descent. Both algorithms can get stuck in local minima, but in actual practice do quite well, especially the EM algorithm.

Most neural network learning algorithms (such as back propagation and its improvements) are gradient descent algorithms. That invites a comparison between Bayes net learning and neural net learning, with = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_latent_node.htm');return false;">latent variables` corresponding to hidden neurons. In the case of Bayes net learning, there are generally fewer hidden nodes, the learned relationships between the nodes are generally more complex, the result of the learning has a direct physical interpretation (by probability theory) rather than just being black-box type weights, and the result of the learning is more modular (parts can be separated off and combined with other learned structures).

## Single Case Learning

Netica has the ability to revise the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs of nodes to account for the currently entered case, as well as some other learning abilities.

**Preparation:** To learn from a single case, you must first have a net constructed, including all nodes, states and links. Nodes in the net may already have their CPTs, which you entered manually or previously learned, and which you now want to improve using learning. Or there might not be any CPTs, and you want to learn them from scratch.

**Doing:** If the case is not already in the Bayes net, you enter it into the net as findings. Only = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_positive\_finding.htm');return false;">positive findings will be used; negative and likelihood findings will be ignored. Then you = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the nodes whose probabilities you would like to have revised to account for the case, if possible. Usually you would like all possible nodes to have their probabilities revised, so you would select all the nodes (or don't select any nodes, which is equivalent). Then choose **Cases** → **Learn** → **Incorporate Case**, or click the toolbar button which has an arrow pointing from the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case\_symbol.htm');return false;">case symbol to the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_relation\_symbol.htm');return false;">relation symbol: .

**Degree:** You will then be queried for a “degree”, which is normally 1. By making it 2, you can achieve the same effect as learning the same case twice, and equivalently for other numbers. By making it -1, you can exactly unlearn a case that was earlier learned with degree = 1, and so on for other negative

numbers. Don't try to unlearn cases that were never learned, or to unlearn them with greater degree than they were learned.

**What Happens:** Selected nodes for which the case provides sufficient data (i.e. findings for it and its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents) will have their probabilities revised a small amount to account for the case, and their appropriate experience levels increased slightly, according to Netica's [learning algorithm](#).

## Learning From a Case File

This describes how to learn a Bayes net from a file of cases (alternately, you can learn from cases [one-by-one](#) as you enter them, or do other [learning functions](#)). The steps involved are listed below, followed by more detailed instructions:

1. Obtain a file of cases
2. Create nodes for the variables of interest
3. Connect the nodes with links
4. Learn the conditional probability tables (CPTs)
5. You may then want to [view or modify](#) the CPTs, harden the CPTs, [absorb](#) nodes, or learn from further case files (which may require changing the names of nodes or states, or adding new nodes or links)

**1. Case File:** See [Creating Case Files](#). **Note:** If you are using [Netica on a Mac](#), it cannot learn cases from an Excel file. You must first convert the Excel file into a text file in order to successfully execute learning. Note: be sure your text file follows [proper formatting](#).

**2. Nodes:** Before learning begins, you must have a Bayes net whose nodes are the variables (i.e. attributes) of the cases. It is okay if it has additional nodes related or unrelated to the cases in the file.

If you don't already have a net constructed, or the net you have doesn't include all the variables in the case file that you wish, **Cases** → **Learn** → **Add Case File Nodes** may be helpful. It will scan through a case file and add to the current net new nodes for any variables that it discovers in the case file that aren't already in the net. The states of the new nodes will be all the possible values discovered from the case file. If your net already has a node with the same name as some variable from the case file, but that node doesn't have all the states that are mentioned in the case file for that variable, then those states will be added to the node (unless the node has a state called 'other').

After Netica has added all the nodes, you move them to the positions you want, and delete any that you aren't interested in.

**3. Links:** Add links between the nodes in the net to capture the dependencies

that you wish to learn. Try to avoid giving any node too many parents, especially if you don't have very many cases to learn from. Alternatively, you can use [TAN learning](#) to learn the link structure, given a target node.

**4. Learn:** When you choose **Cases** → **Learn** → **Incorp Case File**, Netica will ask you for a case file and a “degree”. Normally, you enter 1 for the degree, but you can enter other numbers for special effects. If you want to undo the effect of earlier learning, you can learn again from the same file, but with a degree of -1 (it doesn't matter if you have done other counting learning since then, providing you haven't hardened, softened, faded, or edited the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs). If you enter 2 for the degree, the learning will act as if it sees every case in the file twice, and similarly for other numbers (fractional numbers are okay). Netica builds up the CPTs according to its [learning algorithm](#), and as it processes the cases, it reports its progress in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window.

## Learning From Cases Using Excel

Netica can learn the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs of nodes directly from = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case.htm');return false;">cases stored in an Excel spreadsheet (aka workbook). Described elsewhere is Netica's ability to learn from text files of cases, and Netica's ability to link with Excel to export posterior beliefs.

Here are the steps to learn a Bayes net directly from an Excel spreadsheet:

1. Create an Excel Spreadsheet
2. Add Nodes to the Net
3. Discretize or Combine States
4. Add Link Structure
5. Learn CPTs
6. Use the Resulting Bayes Net

**Note:** if you are using Netica on a Mac, learning from an Excel file will not work. You must convert your case file into a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_file.htm');return false;">text file and then you can proceed with the steps listed above.

# Learning From Cases Using Excel

## 1. Create an Excel Spreadsheet

2. Add Nodes to the Net
3. Discretize or Combine States
4. Add Link Structure
5. Learn CPTs
6. Use the Resulting Bayes Net

**Structure:** The spreadsheet should be arranged with columns corresponding to the variables of interest (by [node name](#)), and each row being a case (aka "record"). At the intersection of each row and column is the cell that gives the value of the variable indicated by the column, for the case indicated by the row.

The first row **must** contain the names of the variables. Each will correspond to a node in the Bayes net, although the Excel file may have some variables that don't appear in the net and vice-versa. If desired, you can give each case an [identification number](#).

**Representing Prior Knowledge:** To simplify prior percentage knowledge in your file, you can use the [NumCases](#) function to denote the percentage of cases. For example, if you had a million cases and there are 50 cases that have the same "set of findings", you can write 10% in the row that matches that data set. This indicates that you have seen 10% of cases with these exact findings. Netica will then run that line through 10 times (or whatever represents 10% of the cases).

**Link:** The Windows database software must be able to identify the Excel worksheet as a database table. It may do this automatically, or from Excel you may have to select all the relevant cells, and then in the little box to the left of the formula bar (for defining names), enter in any name and press `ENTER`.

Finally, save the file. **Note:** if you are using Netica on a Mac, it will throw an error when learning from an Excel file. You must convert your data into a text file in order for the learning to work.

**Subset of Cells/Choosing Table:** If you already have several tables defined in

the spreadsheet, or you want Netica to just use a subset of the cells, select the set of cells you want to use, and define it with the name “ForNetica”, as described above. Whenever there is a table with that name, Netica will use it instead of any other.

>> [Next Step](#)

# Learning From Cases Using Excel

1. Create an Excel Spreadsheet
2. Add Nodes to the Net
3. Discretize or Combine States
4. Add Link Structure
5. Learn CPTs
6. Use the Resulting Bayes Net

You may already be starting with a Bayes net that has nodes with suitable names and states to match the Excel data, in which case you can go on to the next step. If not, you will have to [add](#) nodes or change their names, as described below, or let Netica do it automatically as described further below.

**Manually:** Ensure that each node's name or its title exactly matches the text in the cell at the top row of the Excel spreadsheet. Also, the state names or titles must exactly match each of the possible entries in that column (unless the column contains numeric data). **The matches are case sensitive.**

To achieve this, you may want to change the names of the nodes, or change the Excel cells, or add [node titles](#).

**Important:** If the Excel names have spaces or special characters in them, they won't be able to match the node or state *names* (since those characters are not allowed in names), so they will have to match node or [state titles](#).

**Automatically:** A great time saver is Netica's ability to examine the spreadsheet and add nodes to the Bayes net with properly matching node and state names. The Bayes net may start off empty, or it may already have some nodes. Choose **Cases** → **Learn** → **Add Case File Nodes**, and from the open-file dialog box presented, choose the Excel file with the spreadsheet. After the nodes are added, you may want to re-arrange their position, or delete ones you aren't interested in. Nodes for numeric data will be added as [= 4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\) BSPSPopupOnMouseOver\(event\);" class="BSSCPopup" onclick="BSSCPopup\('X\\_PU\\_discrete\\_node.htm'\);return false;">discrete](#) nodes, which you probably want to convert to [= 4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\)](#)

BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous  
nodes, and the next step describes an automatic way to easily do that.

**Extremely Slow:** If Excel is running, and has open the same file that you are trying to work with in Netica, the operations in Netica will be extremely slow, so close the file in Excel first.

[Prev step](#) << >> [Next step](#)

# Learning From Cases Using Excel

1. Create an Excel Spreadsheet
2. Add Nodes to the Net
3. Discretize or Combine States
4. Add Link Structure
5. Learn CPTs
6. Use the Resulting Bayes Net

You need to = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretize  
continuous nodes, and if they have already been discretized, you may want to  
adjust the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_state\_threshold.htm');return false;">thresholds  
to better capture the Excel data. If the nodes were added automatically, then  
you may want to convert discrete numeric nodes to continuous ones, and then  
discretize them.

**Automatically:** To use histogram information generated from the Excel  
dataset, select one or more nodes and choose **Modify** → **Discretize Node**  
from the menu. Any of the nodes that are = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_discrete\_node.htm');return false;">discrete will  
be converted to = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous.

Netica may ask you what file you wish to use to generate the histogram, to  
which you would normally indicate the Excel spreadsheet file. Then Netica  
will ask how many states (i.e. bins) you want each node to have, and the  
degree of rounding (20% is usually a good amount). Finally, it will  
automatically do the discretization so that the bins have approximately equal  
amounts of data samples. If you specified 0 % for the rounding, then each bin  
will have as close as possible to the same number of samples, and if you chose

a larger number, the bin amounts won't be exactly equal since Netica tries to choose somewhat round numbers for the thresholds.

**Manually:** Alternatively, you can manually change discrete nodes to continuous with the [node properties dialog](#), and for continuous nodes, you can discretize them or adjust their discretization thresholds with the [Node Discretization](#) setter.

**Combine States:** If the entries in a column are not numeric, and there are some that you don't care about, you can `DELETE` the node's states which correspond to them, and then add a state called **other**.

[Prev step](#) << >> [Next step](#)

# Learning From Cases Using Excel

1. Create an Excel Spreadsheet
2. Add Nodes to the Net
3. Discretize or Combine States
- 4. Add Link Structure**
5. Learn CPTs
6. Use the Resulting Bayes Net

Once the appropriate nodes are in place and their discretization levels chosen, add the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link\_structure.htm');return false;">link structure that you want to work with. You can do this manually as described below, or you can use [TAN Learning](#).

If there is one variable of interest that is most important (called the “target variable” or “target node”), then it is best to draw links from it to all the other nodes. That way you are directly capturing its single relationship with each of the other nodes.

Remember that each node should not have too many links entering it, since then its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT table will be too large, and there will not be enough sampling information in the spreadsheet to adequately fill all the cells.

There is no problem in having a great many links leaving a node, and since Netica will do Bayesian = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">inference on the results, it is okay for links to go in either direction.

That is why to classify, predict or diagnosis a particular variable with the best accuracy, you want to capture its relation with as many of the other variables as possible, so you put many links leaving that variable.

If the only links that are present in the net are ones that leave the target node, then although the way that each other node effects the target node has been captured, they are only being considered in isolation, and synergistic effects of the way in which they effect the target node are being lost.

**Example:** Say that we want to capture the synergy between the 3 strongest influencers of the target node. We might use “[Sensitivity to Findings](#)” (see further below) to identify these 3 nodes. Then we can put links from those 3 nodes into the target node, and from the target node to all the other nodes. In effect that says, “Consider in detail the synergy between the 3 most important influencers of the target node, and for the rest of the nodes, consider all of them, but not all the synergies between them.”

**Using Inference:** Since Netica does Bayesian inference, there is an alternate way to accomplish the above, that isn’t immediately intuitive. We can have links going from the target node to each of the other nodes, and then just put links between the strongest influencers. This method will produce the same results, and it is preferable, since it makes exploring a number of models easier. Also, it allows linking two variables whose synergistic effects we want to explore, and linking another two such variables, and yet still considering the two pairs as independently influencing the target variable.

Sometimes you want to put links simply to create a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT with certain parent nodes, so that you can later observe the conditional probabilities in that table.

If you want to take advantage of domain independence information, or you are developing a causal model, that may also influence the direction of links you choose. However, keep in mind it is possible to learn the CPTs using one link structure, then later reverse and delete links to capture the causal directions you want, as described further in this section.

In general, there are many possibilities for good link structures, depending on what you want to accomplish. You may want to experiment with a few different structures, to see the results you get from each. For example, you can use Netica to do structure learning by writing your own small program that tests a number of candidate link structures to find the best one. You write a

function which searches through some candidate link structures that are plausible and practical in your domain, perhaps also adding trial latent variables. For each structure you use Netica's parameter learning functions described in [learning from cases](#), then test the resulting net with Netica's net testing functions also described in this chapter. The net that scores the highest (perhaps penalized for complexity) is the best structure.

**Add Links Fast:** To draw links from one node to a whole set of nodes, = 4  
&& typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the set  
of nodes, then draw a single link from the desired parent node to one of the  
selected nodes, and Netica will add all the links.

[Prev step](#) << >> [Next step](#)

# Learning From Cases Using Excel

1. Create an Excel Spreadsheet
2. Add Nodes to the Net
3. Discretize or Combine States
4. Add Link Structure
5. **Learn CPTs**
6. Use the Resulting Bayes Net

The next step is to have Netica learn the conditional probability tables, or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs.

Choose **Cases** → **Learn** → **Incorp Case File** from the menu, and from the open-file dialog box presented, choose the Excel file with the spreadsheet.

When asked, enter a “degree” of 1.0 unless you are learning from more than one data source, and you want to weight each one. Netica will process the Excel spreadsheet, and fill each node’s conditional probability table.

**Note:** When learning from data sets, the top row does not necessarily need to have the node name (Netica can still learn the data if there isn't a row heading/node name). It is preferable to have a node name, but Netica can learn it without the name. If it does have a node name, it must exactly match and must not contain spaces or special characters. See [case file format](#) for more info.

If the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window is visible, you will be able to see the progress as the percent of cases processed so far. You can abort the operation by pressing **CTRL** while holding down the mouse button for a while.

[Prev step](#) << >> [Next step](#)

## Learning From Cases Using Excel

1. Create an Excel Spreadsheet
2. Add Nodes to the Net
3. Discretize or Combine States
4. Add Link Structure
5. Learn CPTs
6. Use the Resulting Bayes Net

Once the CPTs have been learned, you can use the resulting Bayes net. If necessary, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_compile\_net.htm');return false;">compile the net to do = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">probabilistic inference by choosing **Network** → **Compile**, or click the  toolbar button. You can do what-if analysis by entering = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings for some nodes (by clicking on the state names), and observing how all the probabilities change.

If you have a particular new case you wish to analyze, then you can enter everything you know about it as findings, and then observe the resulting probabilities of the target node to have Netica do classification, prediction or diagnosis.

You can open each node's = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT dialog box to view the conditional probabilities of each node given its parent nodes.

If there is one variable of particular interest (the “target variable”), then you can see how strongly each of the other variables are related to it by selecting it and choosing **Network** → **Sensitivity to Findings** from the menu. To

understand the report generated, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_email.htm');return false;">contact Norsys for the Sensitivity document.

You can copy and paste parts of the network into another network to combine knowledge learned from this Excel spreadsheet with the knowledge from the other network (which may have been learned from other data, or created by hand by an expert). If you do this, you may want to use **Modify** → **Reverse Links** and delete some extraneous links before you copy and paste, to get the link directions in the right (i.e. causal) direction first.

[Prev step](#) <<

## Experience

There has been considerable controversy over the best way to represent uncertainty, with some of the suggestions being: probability, fuzzy logic, nonmonotonic logic, belief functions, Dempster-Shafer, etc. Currently *probability* and *fuzzy logic* are the most practical methods for most applications. Of these two, probability has a much sounder theoretical basis (at least with respect to the way they are actually used). However, probability by itself does not represent the confidence one has in one's beliefs, or lack thereof (e.g. "ignorance")

**Example:** Suppose you had to draw a ball from a bag full of black and white balls, and you couldn't observe how many white balls or black balls are in the bag. If you had to supply a probability that you were going to draw a white ball, it should be 0.5, providing you had no additional information.

Contrast that with a situation in which you can count the balls in the bag beforehand (there are 10 of each), and you will shake the bag before you draw. In this situation the probability of drawing a white ball is 0.5, but whereas in the first situation you were in a state of ignorance, now you feel much more informed.

If you needed to do `&& typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_probabilistic_inference.htm');return false;">probabilistic inference` or solve [decision problems](#), then the 0.5 probability would be sufficient in either situation. In both situations you should believe and act as if there was an equal chance of drawing a white or a black ball. So the concept of experience is not required for these types of problems, and you do not need to be able to represent ignorance (ignorance is the endpoint of the experience spectrum). However, for learning and communicating knowledge, it is useful to be able to represent the degree of experience as well as the probability, as we shall see.

Suppose you then sequentially draw a number of balls from the bag. If you drew 3 white balls in a row, then in the first situation your probability that the next ball will be white should be greater than 0.5, because you are learning (perhaps incorrectly) that there seem to be a lot of white balls. In the second situation your probability of the next ball being white should be less than 0.5,

because you know that now there are more black than white balls left in the bag. Since you should arrive at different conclusions in each of the two situations, you need some more detailed way of representing the original knowledge than just  $P(\text{white}) = 0.5$ .

One way to handle this using just probabilities is to keep track of your beliefs about the ratio of white to black balls in the bag. Then you will have many probabilities, one for each possible ratio. Each of these probabilities will change as you draw a ball, and when you are asked to supply a probability that the next ball drawn will be white, they will all be involved in the calculation.

These are sometimes called *second order probabilities*, but in this example they are really just a probability distribution over possible ratios of balls. It would be easy to create a Bayes net for this, which would have an extra continuous node representing the actual ratio of balls in the bag, and beliefs for each possible ratio would be updated with each observation (for an example of this, see the "Beta Updating" net). That approach works fine for this simple problem, but you can imagine that if you had many interrelated variables, that it would become too complicated, because you would need a separate extra node for each probability number of each = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT of the original net.`

Instead Netica uses the concept of *experience*, which is a measure of the confidence that Netica has in its probabilities.

At each node Netica keeps an [experience table](#), which has a single experience number for each row of the CPT. The experience value corresponds closely to the number of cases that have been seen or its equivalent (normally it is 1 more than the number of cases). This form of experience has sometimes been called the "equivalent sample size" or "ess". To save space, Netica doesn't keep experience tables for nodes that haven't been involved in any learning and haven't had a manual entry of experience. You can view or edit experience tables with the [table editor](#).

The experience numbers are not involved in probabilistic inference or decision problems, since they aren't needed then. But whenever Netica does learning, they are involved (and that will effect the CPTs, which will effect future

probabilistic inference and decision problem results). ([More info on how experience is calculated](#))

## Counting-Learning Algorithm

This describes the simplest algorithm used by Netica for parameter [learning](#) of conditional probability tables (CPTs) from a file of [cases](#), called *counting-learning*. Although it is simple, it is a true [Bayesian learning](#) algorithm.

Before learning begins, the net starts off in a state of ignorance (providing there has been no previous learning or entry of probabilities by an expert). At each node, all `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_CPT.htm');return false;">`CPT probabilities start as uniform, and each [experience](#) starts at its lowest value (normally 1.0).

For each case to be learned the following is done. Only nodes for which the case supplies a value (finding), and supplies values for all of its parents, have their experience and conditional probabilities modified (i.e., no `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_missing_data.htm');return false;">`missing data for that node). Each of these nodes is modified as follows.

Only the single experience number, and the single probability vector, for the parent configuration which is consistent with the case is modified. The new experience number (*exper'*) is found from the old (*exper*) by:

$$exper' = exper + degree$$

where *degree* is the multiplicity of the case (set by you just before learning begins). It is normally 1, but is included so that you can make it 2 to learn two identical cases at once, or -1 to “unlearn” a case, etc. If the case file has a [NumCases](#) column, then actually *degree* would be the product of the degree you entered and the value from the NumCases column.

Within the probability vector, the probability for the node state that is consistent with the case is changed from *probc* to *probc'* as follows:

$$probc' = (probc * exper + degree) / exper'$$

The other probabilities in that vector are changed by:

$$probi' = (probi * exper) / exper'$$

which will keep the vector normalized (exper and exper' act as the old and new normalization constants).

## Bayesian Learning

Generally speaking, it is a wise idea to relate any proposed machine learning method to a Bayesian method to better understand its assumptions, strengths and weaknesses. If it can be cast, at least approximately, into a form of Bayesian learning, then you can check to see if the prior probabilities are suitable for the problem. If it does not even roughly correspond to any form of Bayesian learning, then there is little guarantee in the validity of its results, and it should only be used if it has other valuable qualities, such as being particularly simple or fast.

The Netica [learning algorithm](#) is equivalent to a system of true Bayesian learning, under the assumptions that the conditional probabilities being learned are independent of each other, and the prior distributions are Dirichlet functions (if a node has 2 states, these are “[beta functions](#)”). For more information see [Spiegelhalter&DLC93](#), section 4.1 (with the word “precision” equivalent to our “experience”).

Assuming the prior distributions to be Dirichlet generally does not result in a significant loss of accuracy, since precise priors aren't usually available, and Dirichlet functions can fairly flexibly fit a wide variety of simple functions.

Assuming the conditional probabilities to be independent generally results in poor performance when the number of usable cases isn't large compared to the number of = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
[BSPSPopupOnMouseOver\(event\);" class="BSSCPopup"](#)  
[onclick="BSSCPopup\('X\\_PU\\_parent\\_node.htm'\);return false;">parent](#)  
configurations of each node to be learned.

## Fading

When a Bayes net is supposed to capture relationships between variables in a world which is constantly changing, it is useful to treat more recent cases with a higher weight than older ones. An example might be an adaptive Bayes net that is constantly receiving new cases and doing inferences while it slowly changes to match a changing world.

Netica achieves this partial forgetting of the past by using *fading*. You do regular [learning](#) from cases as the cases arrive, and every so often you select the nodes to be faded, choose **Table** → **Fade**, and enter a *degree* from 0 to 1.

Netica will reduce the [experience](#) and smooth the probabilities of the selected nodes by an amount dictated by the *degree*, with 0 having no effect, and 1 creating uniform distributions with no experience (thereby undoing all previous learning). Then when you continue to learn new cases, they will effectively be weighted more than the cases you just faded.

Fading once with *degree* = 1 d, and again with *degree* = 1 f, is equivalent to a single fading with *degree* = 1 df. So the effects of multiple fadings accumulate as they should. To be most accurate you would fade a very small amount after each case, but for all practical purposes you can just fade a larger amount after a batch of cases.

If an occurrence time for each case is known, and the cases are learned sequentially through time, then the amount of fading to be done is:  
 $degree = 1 - r^{\Delta t}$  where  $\Delta t$  is the amount of time since the last fading was done, and r is a positive number less than (but close to) 1, and depends on the units of time and how quickly the environment is changing. Different nodes may require different values of r.

During fading, each of the probabilities in the node's conditional probability table (`= 4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT`) is modified as follows (where prob and exper are the old values of probability and experience, and prob' and exper' are the new values):

$$prob' = \text{normalize}(prob * exper * (1 - degree) + degree)$$

$exper'$  is obtained as the normalization factor from above (remember that there is one experience number per vector of probabilities). So:

$$prob' * exper' = prob * exper * (1 - degree) + degree$$

## Structure Learning

We are continually adding to [Netica's learning-from-data](#) capability (structure, parameter, testing). The first addition is the TAN Structure learning of version 5.00. TAN structure learning is the automatic method for [learning the link structure](#) of a Bayes net from a case file.

**How to:** First, select a target node that you want to diagnose/predict. Next, choose **Cases** → **Learn** → **Learn TAN Structure**. Netica will automatically determine the appropriate link structure.

As noted in the topic [learning the link structure](#), there is no problem in having a great many links leaving a node, and since Netica will do Bayesian = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">inference on the results, it is okay for links to go in either direction.

That is why to classify, predict or diagnosis a particular variable with the best accuracy, you want to capture its relation with as many of the other variables as possible, so you put many links leaving that variable.

Further documentation will be available shortly; in the meantime, you can read [Friedman, Nir et al 1997](#) to discover how Netica does TAN learning (free version available online or by [e-mailing us](#)).

If you have some data sets that you would [like us to use](#) throughout our research we would be happy to use them and send you all the results.

## Decision Nets

Decision nets are useful in solving decision problems, where the interrelations between variables are represented as in a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes net, and the goal is to find a set of decision functions which maximize a utility objective. An example decision net is “Umbrella”.

Creating a decision net is similar to creating a Bayes net. Once the decision net is constructed, it may be solved, after which you may want to do model-iteration and what-if analysis.

## Decision Nets

Bayes nets are used to determine new beliefs (in the form of probabilities) as observations are made or facts are gathered. They are composed only of *nature nodes*. To form a *decision net* (also known as an “influence diagram”), you add *decision nodes* and *utility nodes* (also known as “value” nodes).

Decision nodes, which are traditionally drawn as rectangular boxes, correspond to variables or events that you will be able to control. Utility nodes are drawn with a diamond or flattened hexagon shape, and correspond to quantities you want to maximize.

The decision net as a whole represents a [decision or planning problem](#) that you face, or that some other agent, often called “the decision maker”, faces. Netica can find values for the decision nodes that will result in the largest possible expected value for the utility node (or sum of them if there is more than one).

This is known as “solving” the decision net. Once the net is solved, you (or the decision maker) can begin to enact the plan by taking the prescribed action for each decision as it arises.

Links that go into a decision node have a special meaning and are called *informational links*. They indicate what will be known at the time the decision is to be made. That is, the decision maker will know the values of all the nodes which have links into that decision node, and will not know the values of any other nodes.

Earlier it was stated that when solving a decision net Netica finds values for each of the decision nodes. But if there are some links entering a decision node, it actually finds a decision value for each possible configuration of values of the [= 4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\) BSPSPopupOnMouseOver\(event\);" class="BSSCPopup" onclick="BSSCPopup\('X\\_PU\\_parent\\_node.htm'\);return false;">parents](#). This is a contingent plan, with a form like: “if I observe A=high and B=no, then I will do X, if I observe A=low and B=yes, then I will do Y, and so on. So, solving a net finds a *decision function* for each decision node (i.e. a function which gives a decision value for each possible set of parent values).

If there are a number of decision nodes, possibly corresponding to decisions made at different points in time, then solving the net will find a decision function for each of them, and this set of decision functions is known as a

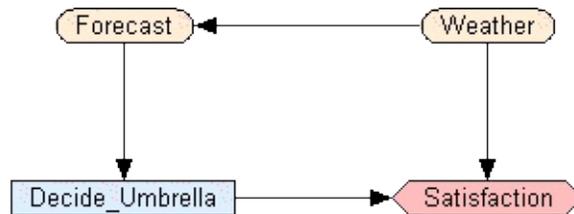
*policy*. It is a full conditional plan, specifying what to do in each possible contingency, based on the information that will be available.

Working through an [example](#) may make these ideas clearer, and then you may want to [create your own](#).

## “Umbrella” Example Decision Net

A very tiny [decision net](#) from Ross Shachter known as “Umbrella” serves as a good simple example. You can [read it](#) into Netica from the “Examples” folder.

It has 2 *nature nodes* representing the weather forecast in the morning (sunny, cloudy or rainy), and whether or not it actually rains during the day (rain or no\_rain). It has a *decision node* of whether or not to take an umbrella, and a *utility node* that measures the decision maker’s level of satisfaction. There is a link from Weather to Forecast capturing the believed correlation between the two (perhaps based on previous observations).



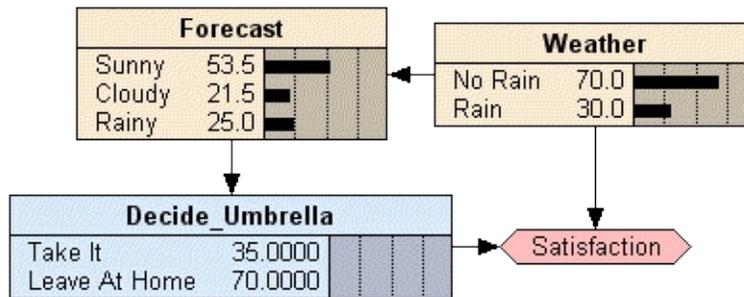
There is a link from Forecast to Umbrella indicating that the decision maker will know the forecast when he makes the decision, but no link from Weather to Umbrella; if he knew for certain what the weather was going to be, it would be easy to decide whether or not to take the umbrella.

There are links from Weather and Umbrella to Satisfaction, capturing the idea that he is most happy when it is sunny and he doesn’t take an umbrella (utility = 100), next most when it is raining and he takes an umbrella (utility = 70).

He hates carrying an umbrella on a sunny day (utility = 20), but is most unhappy if it is raining and he doesn’t have one (utility = 0). You can examine the utility values by `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_select_node.htm');return false;">selecting node Satisfaction, and then clicking `

To compile this decision net, click . [Auto-updating](#) is turned on for this decision net, so as soon as you compile it, updating will occur (i.e. it will be “solved”).

The display will change to something like:



The number beside each decision choice indicates the expected utility of making that choice. So before any information is known, deciding to take the umbrella results in an expected value of 35, while leaving it at home gives 70. Clearly the best choice given the available information is to leave the umbrella at home.

If the decision maker hears that the weather forecast is sunny, then this can be entered by clicking on the word “sunny” of the Forecast node. The expected utilities corresponding to each decision choice change. The best decision is still to leave the umbrella at home, but the expected utility has increased to 91.59, because the extra information indicates it is now more likely that the umbrella won’t be needed.

Say instead the forecast was cloudy. Click on the word “cloudy” to change the finding. Still the best decision is to leave the umbrella at home, but the expected utility has decreased to 65.12, because of the increased chance of rain.

Try a forecast of rainy. The best decision changes to “take the umbrella”, and the expected utility of that decision is 56.

With no findings = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered for node Forecast (since it is the parent node), select node “Decide Umbrella”, and click on . The table dialog will open, and there you can see the optimal decision function, which is to leave it at home unless the forecast is rainy.

For a more advanced example see [Car Buyer](#). It represents a sequential decision problem (has a later decision depending on the first), and it has multiple utility nodes.

## “Car Buyer” Example Decision Net

“Car Buyer” is an example [decision net](#) illustrating sequential decisions and multiple [utility nodes](#). For a more simple example, see the [Umbrella example](#).

Open “Car Buyer Neapolitan” from the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_examples_folder.htm');return false;">Examples folder`. It is from [Neapolitan90](#), which is a simpler version of the car buyer example of Howard62. Eliminating the rationale given there of how the numbers are arrived at, we have the following story:

Joe is going to buy a used car, which could be good with probability 0.8 or a lemon with probability 0.2. After accounting for repairs, Joe’s profit will be \$60 if the car is good, and \$-100 if it is bad. Before buying the car he has the option of having one test or two tests done on it. The first test costs \$9, and both together cost \$13. The first test has a 90% chance of returning positive if the car is good, and a 40% chance if it’s a lemon. If the first test returns positive, then the second test has a 88.89% chance of returning positive if the car is good, and a 33.33% chance if it’s a lemon. If the first test returns negative, then the second test has a 100% chance of returning positive if the car is good, and a 44.44% chance if it’s a lemon.

Joe has 2 decisions to make: whether to do the tests, and whether to buy the car. These are represented by the “Do Tests?” and “Buy It?” decision nodes. The outcome of the tests are given by nodes “First Test” and “Second Test”. The costs of the tests are represented by utility node U, and the profits after repairs (not including test costs) by utility node V. The overall utility is the sum of these two (with costs being negative).

When Joe decides whether to do the tests, he doesn’t know the value of any of these variables, so there are no links entering the “Do Tests?” node. When he decides whether to buy, he will know the outcome of both tests (the outcomes may be “not done”), and so there are links from those two nodes to “Buy It?”.

He will also know the value of “Do Tests?” since he has already made that decision, so you could put a link from that node to “Buy It?”, but it is not necessary since it is a [no-forgetting link](#) and there is already a directed path from “Do Tests?” to “Buy It?”.

You can examine the rest of the link structure, and check out the [relation tables](#), to make sure they make sense to you.

Then [compile](#) the net. Since [auto-updating](#) is turned on, it will be solved immediately. Netica adds a no-forgetting link from “Do Tests?” to “Buy It?”, indicating that “Do Tests?” may be relevant to the second decision, based only on the rest of the link structure (as it turns out, it isn’t). The expected utility of each decision choice for “Do Tests?” is printed in the node; not doing any of the tests has the highest expected utility of 28, so it is the best choice. No expected utilities are printed in the “Buy It?” node, since the first decision has not yet been made.

If you click on the “None” decision of “Do Tests?” to indicate that Joe decides not to do any tests, expected utilities appear in the “Buy It?” node. The best choice is to buy it, resulting in the highest expected utility – an overall profit of \$28. Try clicking on “First” of the “Do Tests?” nodes, indicating that Joe decides to just get the first test done. His overall expected utility is 26.2. Then click indicating that the first test outcome was “Positive”. The best decision is to buy and the expected utility is 35. If the first test comes out negative, then the best decision is to not buy, and its expected utility is -9 (which is the cost of the test). You can experiment with other combinations of findings and decision choices.

Now perhaps you want to [create](#) your own decision net?

## Creating a Decision Net

Creating a [decision net](#) is pretty much the same as creating a Bayes net, so it involves [adding the nodes and links](#), [setting node properties](#) and [entering node tables](#).

One difference, obviously, is that you must designate some nodes as = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_decision_node.htm');return false;">decision and
```

```
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_utility_node.htm');return false;">utility nodes.
```

Usually you do this when you first add them, by using the  toolbar button to add decision nodes, and the  button to add utility nodes. However, you can

change node kinds using the = 4 && typeof(BSPSPopupOnMouseOver) ==

```
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_node_dialog_box.htm');return false;">node
```

[dialog box](#), or by selecting the nodes you wish to change, and then clicking on the toolbar button for the desired kind, while holding down the `CTRL` key.

You don't enter a = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_node_relation.htm');return false;">table for
```

decision nodes, since Netica finds that when it solves the decision net.

Utility nodes must be [continuous](#), and do not need to be discretized. You may

have multiple utility nodes, and Netica will use the sum of them as the total

utility. Utility nodes should not have = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_child_node.htm');return false;">children.
```

There should be at least one = 4 && typeof(BSPSPopupOnMouseOver) ==

```
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_directed_path.htm');return false;">directed path
```

through all the decision nodes to indicate their ordering in time. You do not

need to add [no-forgetting links](#); Netica will add any necessary ones while

solving the decision net.



## No-Forgetting Links

Normally, in the process of making a decision, a decision-maker will know at least everything known for earlier decisions, as well as what those earlier decisions were.

That can be represented in the [decision net](#) by including links to a decision node from all earlier decision nodes, and from all their = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parent nodes.`

These links are called *no-forgetting* links, since they indicate that all the information the decision maker knew earlier, he still knows when making later decisions. Keep in mind that links entering a decision node are = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_informational_link.htm');return false;">informational links, and indicate what the decision maker will know when he makes the decision.`

Having more information to make decisions always leads to a policy having greater expected utility, if that information is relevant. Many of the no-forgetting links may not be relevant; they provide information about the past that does not help in the new decision. If these links are included, the optimal decision functions will simply ignore them, and the expected value of the policy will be the same, but they may result in very large relation tables for the later decisions.

When constructing a decision net, you do not need to concern yourself with no-forgetting links, as long as you put enough to create a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_directed_path.htm');return false;">directed path running through all the decision nodes (to indicate their time sequence). When Netica solves the decision net it will remove irrelevant no-forgetting links, and add any relevant ones that are not present, based on the structure of the net.`

If you wish to see all the no-forgetting links implied by a decision net, make sure there is a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`

BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_directed\_path.htm');return false;">directed path  
running through all the decision nodes and choose **Modify** → **Add No-  
Forgetting Links**. This is useful when solving the decision net using the  
alternate method of node absorption (instead of compiling).

## Solving a Decision Net

After [creating](#) a decision net, you can compile it in the same way as you [compile a Bayes net](#) (except the optimizing compiler doesn't work on decision nets yet).

Netica will attach to each decision node a deterministic function (the optimal is never probabilistic), which provides a value for the decision node for each possible configuration of `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parent` values.

Since the links into a decision node indicate what the decision maker will know when he is about to make the decision, this function provides a decision for each possible information state. Taken together, the decision functions from each of the decision nodes form a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_policy.htm');return false;">policy` which maximizes the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_expected_value.htm');return false;">expected value` of the sum of the utility nodes. You can use the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_table_dialog_box.htm');return false;">table dialog box` to examine the optimal decision table for each decision node.

To display the overall expected utility corresponding to each choice of a decision node, change the node style to [belief-bars](#). The numerical value will appear directly after each state name (although no bars will be drawn). The best decision is the one with the highest expected utility.

Findings and decision node choices may be [entered](#) in the same way as for Bayes nets, and [updating](#) done to reveal the new probabilities and expected utilities.

**Links Change:** When you compile a decision net, Netica may add or remove some links into decision nodes. If a link disappears, then it means that for all possible utilities, CPTs or findings, that link won't be relevant to the decision.

If a new link appears, then it is a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_no_forgetting_link.htm');return false;">no-forgetting link` that is relevant to the decision.

**Missing Numbers:** If there is a path from one decision node to another, it means that the ancestor decision is to be made before the descendent decision. So you must enter a choice for the ancestor decision node before utilities will be displayed at the descendent decision node (if you are familiar with [Hugin](#), you will know that it displays utilities for the descendent decision node all the time, but the Hugin documentation states that these values are not meaningful until the ancestor decision choice is entered).

**Example:** For an example of solving a decision net, see the [umbrella example](#).

## Solving a Decision Net by Node Absorption

Usually you solve a decision net by [compiling](#), but if you know about node absorption (link reversal), you may want to use that. When solving by node absorption, there must be only one utility node and it must not have any children. Also the net must have all [no-forgetting links](#) added before solving.

Choose **Network** → **Optimize Decisions**, and Netica will attach to each decision node its optimal decision function. Taken together, the decision functions from each of the decision nodes form a

```
4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_policy.htm');return false;">policy which
maximizes the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_expected_value.htm');return false;">expected
value of the utility node. In the = 4 && typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_Messages_window.htm');return
false;">Messages window Netica will print the expected utility of following
this policy. It will also put up a = 4 && typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_table_dialog_box.htm');return false;">table
dialog box showing the decision function for the first decision node. You can
use the node selector in the upper left to see the decision functions for other
nodes.
```

For the [umbrella example](#), the decision will be:

<u>Forecast</u>	<u>Best decision</u>
sunny	dont_take_umbrella
cloudy	dont_take_umbrella
rainy	take_umbrella

And “Expected utility = 77” will be printed in the Messages window.

The [node absorption](#) can also be done manually. = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

`BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_select_node.htm');return false;">Select all the nodes and then click the  toolbar button. The net will be decomposed and only the decision nodes and utility node will be left. Each decision node will have the optimal decision as its relation, but it may have some parent links removed. Any removed links are ones that are irrelevant to the decision. All the utility node's parents will be removed, and it will have the maximized expected value as its table. After = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_table_dialog_box.htm');return false;">observing the tables (and perhaps copying and pasting them to a new net), you can restore the original net by choosing Edit → Undo.`

Alternatively, you can absorb the nodes one-by-one, but you must pick a legal order (Netica will tell you if you try to use a wrong order).

## Model Iteration

For some applications of [decision nets](#) (especially in areas like gambling, control or automated decision making), it is relatively easy to set up the tables to suitably reflect reality, and the decisions that the net makes are much better than decisions you would make unaided, because of Netica's superior ability to reason precisely with probability, and to deal with many interactions in very complex situations.

However for other applications (especially in areas like business decision making or public policy), the decisions that Netica makes may not match your "gut feelings", and you may suspect that Netica is wrong. In that case, usually the decision net doesn't capture the decision problem well enough. The usual procedure is to go back to the decision net model, change it to more accurately reflect reality, and solve it again. Then repeat the process until you are satisfied with the results.

But, of what use is the decision net if all you do is try to get it to match the decisions you would make anyway? Professional decision analysts who use decision nets on a regular basis tell us that they are invaluable for disciplined decision making. The process of building the decision net clarifies the problem (often slowly changing your gut feeling as you proceed), and the iteration process forces you to re-examine assumptions you have made.

In a group decision making environment, creating a decision net is often of great benefit in fleshing out the differences in people's beliefs and values, and allows people to discuss specifics rather than arguing in generalities.

Also, once the policy is formed, the decision net acts as a living document of the beliefs and values that lead to it. If those change over time, the decision net can be changed to generate a new policy. Or parts of it can be copied and pasted into new nets for new decision problems. If a policy results in a bad outcome, it is possible to go back and determine whether it was based on bad information, and if so, to change that information for the next decision making situation.

## What-if Decision Analysis

With Netica's [undo feature](#) it is easy to solve a [decision net](#), undo it, change a utility value or a node table, and then re-solve the net to see how the optimal decisions and maximum expected utility are affected.

But what if you want to force a decision node to have a certain decision function, and see how the other decisions and maximum expected utility change? Just enter the desired decision function for the node in its [table dialog box](#) (or paste it in, since you are often obtaining it from a previous solution of a decision net), select the decision node, and change it into a nature node by clicking the  toolbar button while holding down the `CTRL` key. Then solve the net in the usual way, and observe the changes to the other decision functions and the change to the maximum expected utility. Of course you can use 'undo' to return to the original situation. In general, when you want to set a decision node to a certain decision function while you continue your analysis, you temporarily change it into a nature node.

## Display Style and Printing

The **Style** menu allows you to change the way nets are displayed and printed out, without changing what they mean or the way they behave. The [size](#), [font](#), [color](#) and [labeling](#) of the nodes can be adjusted, and individual nodes can be displayed in a [form](#) which is best for that node. The labeling and nature of the [displayed links](#) can also be changed. The [magnification](#) of the drawing can be adjusted.

Net diagrams can be [printed](#) with a printer, in color and magnified/reduced if desired. Presentation quality graphics can be created by [copying and pasting](#) from a net into other applications or by generating [SVG graphics](#).

If an end-user who is not familiar with Bayes nets is going to be using the finished net, then it can be displayed in a suitable manner, perhaps by hiding links and some nodes, and displaying the rest of the nodes as bar graphs, meters, or data entry ovals, depending on the application. You may also want to [place text](#) on the net diagram, to add a title or note.

## Node Name and Title Display

Every node has a name, but there are = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_IDname.htm');return false;">restrictions on names. To provide you with greater flexibility in labeling nodes, some or all of the nodes can also be given a [title](#), which has no restrictions. One of the [style options](#) of nodes is how the name and title are displayed on them.

To have all the nodes of the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active net displayed using their name, choose **Name** from the **Style** menu, to display them by title choose **Title**, or to display them using both, choose **Name: Title** or **Title (Name)**. The choice that you make will apply to all the nodes of the net, for both screen displays and printing displays.

To change the name or title of a node, use the [node dialog box](#).

If you are displaying nodes by title, and some nodes don't have a title, then their name will be used instead. The starting style of a new net is **Title**, but you might not notice this if you don't enter any titles, since then just the node names will appear.

## Font and Size

One of the [style options](#) of nodes is the font used for the writing within them.

The net has a default font & size for nodes, which may be changed by making

sure no nodes are = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_deselect\_nodes.htm');return false;">selected,

and then choosing **Style** → **Font** (or by = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-clicking

on the net = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_background.htm');return false;">background

and choosing **Modify** → **Default Node Style** → **Font**). A dialog box will appear showing the current font and size, which you can modify accordingly.

If you = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select some

nodes before doing this operation, the new font will apply to the selected nodes only, and will override the default.

If you change the default font, and some nodes don't change, it is because they have an overriding font. To reset them to default, select them, right-click on the selection, choose **Style** → **Font** and set them to the same font as the default.

**Node Size:** Whenever you choose a new node font, or a new font size, the nodes will be resized so that the writing within them fits nicely. In fact, the way you enlarge or shrink the nodes of a net is to choose a larger or smaller font size for them.

**Links:** To change the font for the labels of all disconnected links, use **Style** → **Links** → **Link Font**, or alternately, right-click on the = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_background.htm');return false;">background

and choose **Modify** → **Default Link Style**. [More Info](#)

**Character Sets:** Keep in mind that if your net must display [international](#)

character sets, then you must choose a = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_Unicode.htm');return false;">Unicode font that
```

includes the desired characters. "Arial Unicode MS" is usually a safe choice.

## Node Color

One of the [style options](#) of a net is the color scheme of the [node-sets](#) within the net.

### Default Node Colors:

Decision Nodes: blue

Nature Nodes: beige

Utility Nodes: pink

Note/Text Nodes: light blue

Under the **Style** menu, choose '**Colors**' to produce a node-set properties dialog box and arrange the node-set colors to produce the desired visual effect. [More Info](#)

## Node Styles

There are several different forms in which a [node](#) can be displayed, and these are known as *node styles*. There is a default style for the whole net, but each node can be set to its own style individually, which overrides the default.

**Default Style:** To set the default style for the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_active_window.htm');return false;">active net,`  
make sure that no nodes are = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_deselect_nodes.htm');return false;">selected,`  
and then choose the desired style from the **Style** menu, or = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-click on`  
the net = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_background.htm');return false;">background`  
and choose **Modify** → **Default Node Style**. If some nodes don't change, perhaps they have overriding styles (see below).

To have some nodes override the default style, = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_select_node.htm');return false;">select them`  
and then choose the desired style from the **Style** menu. If you have given some nodes an overriding style, but now you want them back to the default style, select them and choose **Style** → **Default**.

**Forms:** The various node forms are:

- **Labeled-Box:** Draws a box whose shape and color are determined by the node kind, with the [name/title](#) of the node written within it (= 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_deterministic_node.htm');return false;">deterministic nodes will be displayed using a thick border).`
- **Belief-Bar:** Draws a bar graph showing the current beliefs for each state of the node.

- Meter: Displays a needle meter showing the = 4 &&  

```

typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_mean_value.htm');return false;">mean
value for = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_continuous.htm');return
false;">continuous nodes and the current = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_belief.htm');return false;">belief for = 4
&& typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_discrete_node.htm');return
false;">discrete nodes.

```
- Hidden: Doesn't draw anything (you can also [hide the links](#)).
- Circle: Displays the node as a small circle.

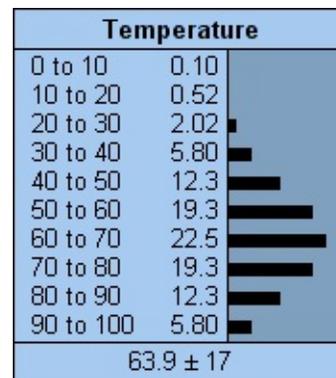
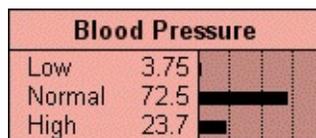
**Other Styles:** Keep in mind that you can also change other aspects of the node style, such as the size, font, color, and labeling of the nodes.

## Belief–Bar Node Style

The node form most useful to display the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">beliefs of a multi-state nature node, or expected utilities of a decision node, is the belief-bar style. Nodes can be displayed in this form by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">selecting them, and then choosing **Style** → **Belief Bars** (choose this with no nodes selected to set the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_default\_node\_style.htm');return false;">default style of the whole = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active net).

Each node will be displayed as a box labeled at the top with the node's name, title or both depending on the node's labeling.

Examples of a discrete node (left) and a discretized continuous node (right):



The name of each state is shown in the section to the left, along with a number expressing the belief (probability) of that state as a percentage. If the name is too long to fit, it will end with an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_ellipsis.htm');return false;">ellipsis. If the

percentage is too small to display numerically, but nonzero, then it will be displayed as “0+”.

**Scale:** In the section to the right are bar graphs depicting the belief amounts. Vertical dotted lines mark the 25%, 50% and 75% levels. If the node has 4 or less states, then the left edge of the box is 0% and the right edge is 100%, but if there are more states, then the scale may be expanded, so the right edge is less than 100%. If the scale is expanded so much that the right edge is less than 25%, then none of the vertical dotted lines will be visible.

**Finding:** If the net has been successfully = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_compile_net.htm');return false;">compiled, a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_positive_finding.htm');return false;">finding may be entered into a node just by clicking on the name of its state (and removed by clicking on it again). A = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_negative_finding.htm');return false;">negative finding for a node can be entered by holding down the SHIFT key and clicking on the name of the state that its known not to be.`

If the node has a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_positive_finding.htm');return false;">positive finding, then the bar for the state corresponding to the finding will be bordered above and below by black lines and the node color will darken to gray (unless you've changed the default node colors). If the color of the node indicates it has a finding, but no bar is surrounded by black lines, then a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_negative_finding.htm');return false;">negative or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_likelihood_finding.htm');return false;">likelihood finding has been entered.`

**Mean & Variance:** If a node is = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous or has = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state\_value.htm');return false;">state values defined, then its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_mean\_value.htm');return false;">mean value (i.e., = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_expected\_value.htm');return false;">expected value) will be shown in a separate area below the belief-bars. The mean value will be followed by a  $\pm$  symbol and its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_standard\_deviation.htm');return false;">standard deviation.

**Display Fewer States:** It is possible to have Netica display only the most probable states, which is very useful for nodes having many states. If you = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select some nodes, choose **Style** → **Show Only N States**, and enter a number, then the display of those nodes will be limited to the N most probable states, ordered from most probable to least. At the bottom will be an entry called "other-" that is the sum total of all the states not displayed. As you enter findings and the beliefs change, then which states are most probable also change, so which states are displayed, and their ordering, changes (if you want their ordering permanently changed by beliefs, see [here](#)).

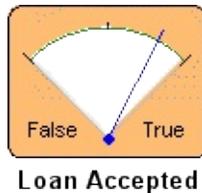
If no nodes are selected when you choose **Show Only N States** from the menu, then the number you enter will be the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_default\_node\_style.htm');return false;">default for the whole net. Nodes that have fewer states than the number you enter will

be displayed normally (i.e. with all their states in the order you defined). The default for a net before you change it is 50. So if you notice that only 50 states are being displayed for the nodes in your net, you can use this method to change that.

**Dimmed:** If the belief percentages and bar lengths are currently invalid for the net's findings (probably because the latest findings have not yet been taken into account with a `4 && typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_belief Updating.htm');return false;">belief updating`), the bars will be drawn in a grayed or dotted manner.

## Meter Node Style – Discrete Nodes

Meters are often the node form of choice for = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_binary\_node.htm');return false;">binary nodes like true/false propositions or okay/faulty conditions. Example:



Below the meter is the node's name, title or both depending on the node's labeling. The two states are printed at the bottom of the meter, on the left and right. The needle indicates the relative belief in each state. For the right-most state, the left end of the scale corresponds to a belief probability of 0% and the right end to 100%. The 50% point is when the needle is vertical, so the center of the scale is marked with a tick.

**How To:** = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">Select any discrete nature node and choose **Style** → **Meter**. If no nodes are selected when you choose **Style** → **Meter**, the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_default\_node\_style.htm');return false;">default style for the whole net will be set. If the needle has bands to the sides, or numbers appear instead of states (like this), it is because the node is continuous or has = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state\_value.htm');return false;">state values defined.

**Finding:** If a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">finding is entered,

the background color of the node will be changed to gray (unless you've changed the default [node colors](#)). If it is a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_positive_finding.htm');return false;">positive finding`, the needle will be all the way to one side, indicating certainty.

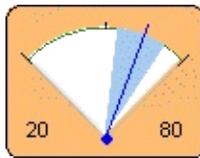
**Dimmed:** If the current position of the needle is invalid for the findings entered (probably because the latest findings have not yet been taken into account with a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_belief_updating.htm');return false;">belief updating`), the needle will be drawn gray, or not at all.

**More States:** If the meter style is used for a node with more than two states, then all the states except the first will be grouped together and will be called “Other”. The first state will be right-most on the meter, so its belief will be well represented by the needle (0% full left and 100% full right).

**Continuous:** For a continuous variable example, click [here](#).

## Meter Node Style – Continuous Nodes

Belief-meters for nodes of continuous variables, or those with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state\_value.htm');return false;">state values defined, display a needle which shows the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_expected\_value.htm');return false;">expected value of the node, and bands on the sides of the needle which show the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_standard\_deviation.htm');return false;">standard deviation, like this:



Temperature

The number to the left of the scale is the minimum value of the variable, and to the right is the maximum value. Below the meter is the node's name, title or both depending on the node's labeling.

**How To:** = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">Select any continuous nature node and choose **Style** → **Meter**. If no nodes are selected when you choose **Style** → **Meter**, the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_default\_node\_style.htm');return false;">default style for the whole net will be set. If the needle has no bands to the sides, or state names appear instead of numbers (like this), it is because the node is discrete.

**Finding:** If a = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_finding.htm');return false;">finding is entered,
the background color of the node will be changed to gray (unless you've
changed the default node colors). If it is a = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_positive_finding.htm');return false;">positive
finding, the needle will be all the way to one side, indicating certainty.
```

**Dimmed:** If the current position of the needle is invalid for the findings entered (probably because the latest findings have not yet been taken into account with a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">belief updating), the needle will be drawn gray, or not at all.

**Discrete:** For a discrete variable example, click [here](#).

## Link Styles

The **Style** → **Links** menu provides [style options](#), which allow you to choose the way all the links in the net are displayed. The menu is also available by =

```
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-clicking
on the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_background.htm');return false;">background
```

and choosing **Modify** → **Style Default Link**.

**Hidden:** = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">Toggling

**Style** → **Links** → **Hide Links** allows you to hide all the links in the net. This is especially useful when you are creating something for an end-user who does not care how nodes are linked together (then you may also want to [hide](#) some nodes). To avoid confusion, it is not possible to display some links and hide others (although if the nodes at both ends of a link are [hidden](#), the link between won't be drawn either).

**Technical:** The remaining 3 choices of the **Style** → **Links** menu are for those familiar with the process of Bayes net = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_compile_net.htm');return false;">compiling,
and wish to see how it is being done. They allow you to switch between
showing the link structure of the net which was originally constructed (Style
→ Links → Regular Dag), the link structure of the = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_Markov_network.htm');return false;">Markov
net derived from it (→ Markov Net), and the link structure of the = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_triangulated.htm');return false;">triangulated
```

net derived from the Markov network (→ **Triangulated**).

The triangulated net is used internally to determine the = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_clique.htm');return false;">cliques when
```

compiling the net, and a number is displayed with each node to indicate its

position in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_elimination_order.htm');return
```

```
false;">elimination order of the compilation.
```

The **Triangulated** and **Markov Net** styles are only for viewing the net, printing it, and copying its graphics (with no nodes selected). If you try to do any other operation the style will automatically change back to **Regular Dag**.

Before you choose these from the menu, you may want to save your net, since they will lose information about link bends.

## Copying and Pasting Graphics

After copying or cutting all or part of a net, you can paste the graphics into other Windows applications, such as Microsoft Word, PowerPoint, Works, etc.

If some nodes are = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">selected when the **Copy** command is done, then graphics for only those nodes will be placed in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_clipboard.htm');return false;">clipboard. If no nodes are = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_deselect\_nodes.htm');return false;">selected, then graphics for the whole net will be. The nodes will be drawn with the same design style as they are shown on the screen, so you may want to adjust the [style options](#) before copying to the clipboard.

**Shape:** After you paste the graphic into another program, you may want to adjust its size and shape so that the aspect ratio (i.e. height/width ratio) is similar to the Bayes net it came from, in order to achieve the best appearance. If the aspect ratio is very different from the original, some of the characters may overlap.

**Into Word:** Netica graphics are pasted into other programs as “enhanced metafiles”. When pasting into Microsoft Word it is often more convenient to have the graphic as a Microsoft Word “picture”. You can achieve this by first pasting it into Word, then while it is still selected do a **Cut**, then do an **Edit** → **Paste Special** and choose “Picture” from the dialog box (and perhaps remove the check from ‘float over text’). Sometimes Word does not do a perfect job of converting it, but just double-clicking on the graphic to bring up the picture editor, and then closing it again without making any changes, seems to fix any imperfections.

**Bitmap:** If you want to paste a net into a bitmap program, such as “Paint” (which comes with Microsoft Windows), it is recommended that you press **ALT+PRINTSCREEN** while Netica is the current application. The contents of the whole Netica window will be placed in the clipboard in bitmap form, so you

can paste it into whatever program you want, and then edit it to remove parts of the image you don't want.

## Printing

You can print a net on a printer with **File** → **Print**, or from the [command line](#).

First you may want to use **File** → **Printer Setup** to select printing options, such as page size, margins, magnification/reduction, and which printer to use.

After pressing **Okay** on the first dialog box which comes up, if the diagram is large enough, you will be queried for how many pages large you want the printing to be (default is to appear on one page). If you leave those entries blank, then you will be queried for a printer magnification instead (default is 100%).

**Multi-Page:** If the net diagram is larger than one page, then the printer will print parts of it on separate pages in such a way that the pages can be trimmed and then attached together to produce the overall diagram. If you wish to see what part of the diagram will end up on which page, you can turn on the page breaks by = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_toggle_menu.htm');return false;">toggle
```

**Layout** → **Show Page Breaks**. This will draw light blue lines at the page divisions, which can also be useful for visualizing your [drawing size](#).

When you change the magnification/reduction (or margins, etc.) using **File** → **Printer Setup**, the page break lines will shift showing how much fits on each page.

If you want really professional graphics results from Netica use [SVGs](#).

## SVG Graphics

Netica can generate very high quality graphics for print publishing or website construction. These graphics are in a format known as SVG, which stands for "scalable vector graphic". It is the main XML based format for graphics, and since it is a *vector graphic*, the images look very good at all resolutions, and the file sizes are very small. (Netica can also make [other](#) vector graphics).

SVG was jointly developed by W3C, Adobe, Sun, Apple, IBM, etc. Microsoft has stated that all their relevant products will support it (the latest version of MS Visio now does). SVG graphics can be imported into most modern Adobe products, for editing or converting to other graphics formats. The Firefox (Mozilla) and Opera browsers natively support SVG graphics, and various plug-ins are available for the Microsoft browser.

**How To:** Have Netica display your Bayes net in the desired [style](#), make sure it is the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active window, and choose **File** → **Make SVG Graphic**, or press **CTRL+G**. If you want the graphic to be of a subset of the net, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the desired nodes first. The default format that Netica will generate is SVGZ, which is a compressed SVG. SVGZ is generally a better choice because it has a significantly smaller file size, produces exactly the same image, and most programs can work with it. If you need a regular SVG, either because it is for some program that can't work with SVGZ, or because you want to examine or edit it with a text editor (or XML editor), then choose "Graphics File - XML (.svg)" from the drop-down menu at the bottom of the **Save As** dialog box that appears during generation.

**Image Editing:** Once an SVG has been made, you can open it with Microsoft Visio and edit the image. Nodes will act as whole objects, so they can be moved, resized, scaled, etc. If you want to edit some feature within a node, you must first "ungroup" the node, with **Shape** → **Grouping** → **Ungroup**. You can also add titles and graphics, change colors and the like to make sharp

presentation pieces.

**Microsoft Browser:** Since MS Internet Explorer does not yet natively support SVG, it requires a plug-in. There are several available, but currently the best is the one from Adobe, since it can do zooming and panning. That plug-in usually gets quickly and painlessly installed over the internet as soon as Explorer tries to display an SVG (after putting up a dialog box asking if you want it installed). If you don't see the dialog box, then you can get the plug-in from:

<http://www.adobe.com/svg/viewer/install/main.html>

**Examples:** If you wish to see examples of SVGs generated from Bayes nets and how to build a website using them, see:

<http://www.norsys.com/netlibrary/index.htm>

**Website Construction:** To add an SVG graphic to a web page, you can put the .svg or .svgz file in the same directory as the HTML file for the web page, and in the HTML file put:

```
<embed src="myFile.svgz"
      width="100%" height="100%"
      pluginspage="http://www.adobe.com/svg/viewer/install/">
```

The width and height attributes are not strictly necessary, but are often useful.

Note that Adobe no longer recommends using the OBJECT tag (see <http://www.adobe.com/svg/viewer/install/mainframed.htm>

For the web server, you probably don't have to do anything, since recent versions of web servers (Apache, IIS, etc.) are already properly configured, but for older versions you need to add to their mime-type table that the official MIME-type for SVG and SVGZ documents is:

```
image/svg+xml
```

The above information was valid as of Oct 2006, but since browsers often change, it may be wise to Google for the latest recommendations from Adobe, Microsoft, Mozilla, Opera, etc. Useful sites include:

<http://www.adobe.com/svg/indepth/pdfs/ReadMewin.pdf>

<http://www.adobe.com/svg/viewer/install/mainframed.htm>

[http://svg-whiz.com/wiki/index.php?title=Server\\_Configuration](http://svg-whiz.com/wiki/index.php?title=Server_Configuration)

[http://svg-whiz.com/wiki/index.php?title=MIME\\_Type](http://svg-whiz.com/wiki/index.php?title=MIME_Type)

<http://www.mozilla.org/projects/svg/faq.html>

For supporting older versions of Firefox and Opera that do not support the EMBED tag, we have found the following to work:

```
<head>
<script language="Javascript">var i.e. = false;</script> <!--[if IE]><script
type="text/javascript">i.e. = true;</script><![endif]--> </head>
<body>
<script LANGUAGE="Javascript">
if (i.e.){
    document.write('<embed src="myFile.svgz" width="100%" height="100%"
pluginspage="http://www.adobe.com/svg/viewer/install/">');
} else {
    document.write('<object data="dir/myFile.svgz" width="100%"
height="100%" type="image/svg+xml"></object>');
}
</script>
</body>
```

## Reports and Data Linking

The purpose of Netica *reports* is to: display information in = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_file.htm');return false;">text form on the screen, send it to a printer or a file, or paste it into a word processing or spreadsheet program. Some information can also be [sent by hotlinks](#) to Microsoft Excel, or Netica can be [linked to a program](#) you create using Netica API.

From the **Report** menu, you can generate reports on = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT tables of node relations, the findings currently entered (i.e. the case), beliefs for the current case (i.e. posterior probabilities), summary statistics on the whole net, the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_junction\_tree.htm');return false;">junction tree structure and the elimination order used for compiling, or a list of the currently [selected](#) nodes. Netica can also generate reports on [utility-free sensitivity](#) and how well a [Bayes net performs](#) with respect to a data set.

**How To:** The **Report** menu consists of three groups of items, separated by horizontal menu lines. To make a report you normally choose items from the second group to determine [where the report will be sent](#), and items from the third group to choose the [report options](#). Then make a choice from the first group to generate the report on the desired [subject](#).

**See also:** [User Reports](#)

# Report Subjects

Netica can generate text [reports](#) on:

## Network

Summary information on the whole net. = 4 && typeof(BSPSPopupOnMouseOver) == 'function BSPSPopupOnMouseOver(event);" class="BSS onclick="BSSCPopup('X\_PU\_report\_whole.htm false;">EXAMPLE

## Custom Report

Tailored information on a node(s) or the whole r

## Findings (Case)

The = 4 && typeof(BSPSPopupOnMouseOver) BSPSPopupOnMouseOver(event);" class="BSS onclick="BSSCPopup('X\_PU\_finding.htm');retu false;">findings (i.e. "case" or "evidence") curre including likelihood ("virtual") findings. = 4 && typeof(BSPSPopupOnMouseOver) == 'function BSPSPopupOnMouseOver(event);" class="BSS onclick="BSSCPopup('X\_PU\_report\_findings.h false;">EXAMPLE

## Equations

A list of all = 4 && typeof(BSPSPopupOnMous 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_equation.ht false;">equations of nodes. Choosing the **Horiz** option prints them in [internal form](#). = 4 && typeof(BSPSPopupOnMouseOver) == 'function BSPSPopupOnMouseOver(event);" class="BSS onclick="BSSCPopup('X\_PU\_report\_equations false;">EXAMPLE

## Beliefs

The current beliefs (i.e. posterior probabilities) f nodes, and expected utilities for decision nodes. typeof(BSPSPopupOnMouseOver) == 'function BSPSPopupOnMouseOver(event);" class="BSS onclick="BSSCPopup('X\_PU\_report\_beliefs.ht false;">EXAMPLE

## Node-Sets

The list of nodes within the requested set. = 4 & typeof(BSPSPopupOnMouseOver) == 'function BSPSPopupOnMouseOver(event);" class="BSS onclick="BSSCPopup('X\_PU\_report\_nodesets.l false;">EXAMPLE

## CPT Tables

The node relation as a conditional probability ta

```
typeof(BSPSPopupOnMouseOver) == 'function
BSPSPopupOnMouseOver(event);" class="BSS
onclick="BSSCPopup('X_PU_CPT.htm');return
or = 4 && typeof(BSPSPopupOnMouseOver) =
BSPSPopupOnMouseOver(event);" class="BSS
onclick="BSSCPopup('X_PU_function_table.ht
false;">function table. = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function
BSPSPopupOnMouseOver(event);" class="BSS
onclick="BSSCPopup('X_PU_report_CPT.htm')
false;">EXAMPLE
```

### List of Selected

A list of the names of the nodes currently select

```
typeof(BSPSPopupOnMouseOver) == 'function
BSPSPopupOnMouseOver(event);" class="BSS
onclick="BSSCPopup('X_PU_report_listselecte
false;">EXAMPLE
```

```
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_junction_tree.htm');return
false;"> Junction Tree
```

Details of the net compilation process. = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function
BSPSPopupOnMouseOver(event);" class="BSS
onclick="BSSCPopup('X_PU_report_junctiontr
false;">EXAMPLE
```

### Elimination Order

The order used during compiling. = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function
BSPSPopupOnMouseOver(event);" class="BSS
onclick="BSSCPopup('X_PU_report_eliminati
false;">EXAMPLE
```

### Links to Excel

Hot-links to the node beliefs. = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function
BSPSPopupOnMouseOver(event);" class="BSS
onclick="BSSCPopup('X_PU_report_excel.htm
false;">EXAMPLE
```

Each of the above is an item of the Report menu, and choosing it will generate that report. You can also [set options](#) or change the [destination](#) of the report.

**Which Nodes:** The **Network**, **Junction Tree** and **Elimination Order** reports always apply to the whole net. **List of Selected**, **CPT Tables** and **Links to Paste in Excel** reports apply only to the nodes currently selected. The others apply to the selected nodes if some are selected, or the whole net if none are.

If the **Findings** report is done with nodes selected, they will all be listed in the report, but their findings will be left blank if they don't have one. If no nodes are selected, only those nodes having a finding will be listed.

**Dimmed:** **Junction Tree** and **Elimination Order** will be dimmed if the net is not compiled **List of Selected**, **CPT Tables** and **Links to Paste in Excel** will be dimmed if no nodes are selected.

**List of Selected** can be used to save the current set of selected nodes, to later be restored with **Edit** → **Select Nodes** → **Listed in Clipboard**. It is also useful after an **Edit** → **Find All** search to generate a text list of the found nodes.

## Report Destinations

Reports can be sent to any of:

- To = `4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Messages_window.htm');return`  
`false;">Messages Window`
- Copy to = `4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_clipboard.htm');return false;">Clipboard`
- To File

Each of the above is an item of the **Report** menu, and choosing the item will = `4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_toggle_menu.htm');return false;">toggle it on or off. If there is a check-mark beside the item, then when a report is generated it will be sent to that destination (so it may go to multiple destinations).`

If **Copy to Clipboard** is check-marked, then after generating the report you can simply paste into a text editor, spreadsheet program, word processor, etc. Of course, you can always have the report sent to the Messages Window, where you can look at it or edit it, then copy it from there and paste it. However, if it is too large, it may not fit in the Messages Window.

If you choose **To File**, you will be asked which file to send it to. If you wish to remove the check-mark from **To File** (so output is not sent there), choose **Report** → **To File** and when the dialog box appears, click its Cancel button. You can use the report option **Report** → **Append to File** to determine whether to overwrite an existing file, or append to it.

If it is annoying you that the Messages window keeps coming to the foreground when sending a report to the clipboard or a file, then uncheck **To Messages Window**.

## Report Options

The [report](#) options are:

- Include Names
- Horizontal Format
- Tab Separators
- Append to File

Each of these is an item of the **Report** menu, and can be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">toggled on and off.

If **Include Names** is check-marked, then the names of nodes and states will be included in the reports for findings, relation CPT, beliefs and belief links, otherwise only the numbers will appear. Whether nodes are referred to by name, title or both, can be set by changing the [style](#) of the net.

**Horizontal Format** effects only the equations, findings, beliefs and belief links reports. In general it produces a more compact report, especially if nodes have the same state names. For equations, it shows the [internal representation](#) of the equation.

**Tab Separators** should be check-marked if you are going to paste the report into a spreadsheet (such as Excel). If you are pasting into a word processing program (such as MS Word), then either use space separators and change the font to a mono-space one after pasting (such as Courier), or use tab separators and adjust the tab positions of the pasted text to produce pleasing columns.

## Linking with Excel

Using [reports](#), Netica can send the beliefs it calculates to an Excel spreadsheet. Described elsewhere is Netica's ability to [learn from cases](#) using Excel.

**How To:** From the **Report** menu, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_toggle\_menu.htm');return false;">toggle the [options](#) **Horizontal Format** and **Include Names**, depending on how you want the report to look in Excel (the other options are ignored). = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">Select the nodes of interest, and then choose **Report** → **Links to Paste in Excel**. The belief links (AKA hot-links) will be placed in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_clipboard.htm');return false;">clipboard (the [destination](#) choice is ignored).

Then, click in a single cell of any Excel sheet, and paste. Once you have pasted, the belief numbers will appear in Excel, and will automatically change whenever they change in Netica (for example, as a response to entering a [finding](#)).

**Beautifying:** While the pasted cells are still selected, you may want to choose **Format** → **Cells...** from Excel, click on the Alignment tab of the dialog box which appears, and then choose Horizontal Left. It may look better to widen some of the columns. You can delete any of the pasted names or numbers you are not interested in, and cut and paste the others to move them around. They can be used as the input for any equation, etc., just like any other cell in Excel.

**Netica Exits:** If you stop the Netica program, while Excel with the links is still running, the belief numbers used will simply be the last ones Netica produced. If you restart Netica then Excel will continue to use the old numbers until you tell it to reconnect to Netica by clicking on any cell with a link in it, then clicking in the "fx" text box for editing the cells contents, and then clicking the green check-mark, or pressing the **ENTER** key.

Alternately, you can choose **Edit** → **Links...** from the menu, and in the dialog box that comes up, click on the first link and then **SHIFT** - click on the last one so they are all selected, and then click the **Update Values** button, and then the **Close** button. After the dialog box is gone, the links will be reconnected and will remain so.

**Excel Exits:** If you stop Excel while Netica is still running, and then restart Excel and open the spreadsheet with links, the reconnection will be made automatically. If some nets have to be read into Netica because Excel refers to them, and/or compiled or updated, that will happen automatically.

**Troubleshooting:** To troubleshoot Excel links with Netica, or to control their behavior, check the following:

**Edit** → **Links...** They should have "A" in the "Update" column to update automatically.

**Tools** → **Options** → **Calculation** → **Update remote references** should be checked

**Tools** → **Options** → **General** → **Ignore other applications** should NOT be checked

**Tools** → **Options** → **Edit** → **Ask to update automatic links**

**Edit** → **Paste Special** is not required when pasting the links.

## Linking with Netica API

```
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_Netica_Application.htm');return false;">Netica
Application can be used to provide a user interface for = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_Netica_API.htm');return false;">Netica API.
```

Bayes nets can be displayed in the usual way, findings can be entered and beliefs displayed, with Netica API automating Netica Application.

This is an exciting new feature that is being expanded with each new release of Netica. In many cases it allows you to write a short program that extends the functionality of Netica Application, or a short program that completely controls Netica Application to do some specific function.

Netica API (embedded in your software), runs concurrently with Netica Application, as a separate process, and they communicate with each other.

The communication is completely transparent to you; all you do is call the regular Netica API functions, and all the details are taken care of.

These capabilities are built into versions of Netica after 3.17. Contact = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Norsys.htm');return false;">Norsys to obtain documentation on using them. A version of Netica with a [COM interface](#) (ActiveX) has now been released, which provides yet another [connectivity](#) option.

## Generating User Reports

You can use Netica to display information or report windows, called *user reports*. The purpose is mainly to provide extra information, assistance or directions either to developers or to others who may be using/viewing your BN. A user report could be as simple as a text message giving a more detailed [description of a node](#). Or it could be more complex, such as the current belief probabilities of a [node-set](#), or a [sensitivity analysis](#) of questions with respect to the target node.

User reports are often used with [Netica-Web](#), as they can also be created for the user on the current case or inferred results, which can be printed out or saved to file. They can also be used from the [API](#).

The Netica mechanism to support this is called custom reports. You can [create a template file](#) or modify an [existing template](#) (usually in HTML with [special notations](#) for report elements to be inserted) which defines what you want to have displayed. You then access that template file from your Bayes net. A template file can be used to generate a report on the overall model (i.e. Bayes net), or it can apply to individual questions (i.e. nodes). In the first case, it is called a *net template file*, and the tags within it start with "NET", and in the second case it is called a *node template file*, with tags that start with "NODE".

**See also:** [Reports and Data Linking](#)

## Adding a Report

Adding and creating custom reports to your template folder is quick and simple:

1. Create or customize a report template file.
2. Edit and save the template.
3. Select the desired node(s) and choose **Report** → **Custom Report**.

**1. Create a report file:** The first step is to create the user report file. The easiest way is to use an [existing report template file](#), or modify one that is similar to what you want. To find existing template files, look in the **Report Templates** folder, which is in the same folder as the Netica executable: Netica.exe (the path to this folder is printed out in the Messages window of Netica Application when it first starts up).

**2. Edit the template:** You can open and edit an .htm file directly in Netica, by choosing **File** → **Open as Text**. However, the default templates are HTML files, so you will probably want to use a text editor that has special facilities for HTML (such as Visual Studio). You construct the template file out of normal HTML code, but you put special tags where you want Netica to insert information.

The tags consist of double opening square brackets **[ [** followed by one of the [tag names](#), possibly followed by some extra directions in parenthesis, and ending with two closing square brackets **]]**. You can refer to nodes according to their number or according to their name. For selection, the node that you select first will be considered Node(0). The second node selected will be SelNode(1).

For number, write: `[[Node(0).someInstruction]]`

For name, write: `[[Node("node name").someInstruction]]`

Then save your template file wherever you wish. If you are making htm files, the extension should be '.nsp.htm'. It is probably best not to save your templates in the same folder as the ones that come with Netica Application, just so they don't get mixed up.

**3. Generate the user report:** Next, select the desired node(s) and choose

**Report** → **Custom Report** to generate the tailored "report". From the file dialog which appears, pick the template file you have just created. In a moment your browser should display the results of your report. **Note:** the report will be displayed in whatever format it was created (e.g. htm files will be displayed in an internet browser window, doc files will be displayed in Microsoft Office Word, etc).

**Reports on Multiple Nodes:** You can run a custom report on a single node, multiple nodes, or the entire net. To run a report on multiple nodes, first select the nodes then choose **Custom Report**. Open the template file you created for those nodes, which should contain text with the prefix:  
[[SelNode(0).someInstruction]]. Remember, the node you select first will be considered Node(0).

## Available User Report Tags

The following tags may appear in any report [template file](#), in double square brackets. They will be replaced with a report according to their description. For an example of their use, see the file "Report Templates\Net All.nsp.htm". For editing tips, see item 2 of [adding a report](#).

**Net.BeliefsTable** Report on all the beliefs in the net. Can be displayed in text or html respectively, as `Net.BeliefsTable(TextFormat)` and `Net.BeliefsTable`.

**Net.CaseID** The ID number of the case currently read into the findings.

**Net.CaseProbability** The joint probability of the findings currently entered, as predicted by the net.

**Net.CaseExpectedUtility** The expected utility of the findings currently entered.

**Net.Comment** Comments made within the net or the net description

**Net.EliminationOrderList (Seperator=" , ")** The elimination order used to construct the junction tree.

**Net.Equations (Compiled)** Report on all the [equations](#) within the net. If the "Compiled" tag is left out, then the equations will appear in text as entered. With the Compiled tag, the equation appears in an internal (text) format, which is sometimes illuminating.

**Net.FileName** File name should end in .neta

**Net.FindingsList (Equals=" = ", Seperator=" , ")**  
Report on all the findings within the net in list format.

**Net.FindingsTable** Report on all the findings within the net in table format. Can be displayed in text or html respectively, as `Net.FindingsTable(TextFormat)` and `Net.FindingsTable`.  
If you want the questions listed vertically, you can use:  
`Net.FindingsTable(VerticalFormat)`

**Net.Graphic** Report on net graphics. Not available on most systems.

**Net.JunctionTreeTable (TextFormat)** Report on the net's junction tree.

**Net.ModifyDate** Date Bayes net file was last modified.

**Net.Name** Name of the net.

**Net.NodesetTable** Report on the node-sets within the net. Can be displayed in text or html, as `Net.NodesetTable (TextFormat)` or `Net.NodesetTable` respectively.

**Net.OverallTable** Report on the overall net. Can be displayed in text or html, as `Net.OverallTable (TextFormat)` or `Net.OverallTable (StyleHtml=tableOverall)`. This report contains general numeric measurements on the number of nodes of each type, number of links, number of findings, number of loops, and number of CPTs etc.

**Net.Title** Title of the net.

**Net.User (Field="..")** The value of the named user field.  
Especially useful when building [Netica-Web](#) projects.

## Node Tags

The following tags may appear, in double square brackets, in a report template file meant for a node. They will be replaced with a report according to their description. For an example of their use, see the file "Report Templates/Node All.nsp.htm".

**Node.BeliefsList (Seperator="</td><td>")**

**Node.Comment**

**Node.CPTable**

**Node.CPTable (Experience)**

**Node.CPTable (TextFormat)**

**Node.Equation (Compiled)** This nodes equation, if it has one. If the "Compiled" tag is left out, then the [equation](#) will appear in text as entered. With the Compiled tag, the equation appears in an internal (text) format, which is sometimes illuminating.

**Node.Finding**

**Node.HoverComment**

**Node.InputName (Parent=0)**

**Node.IsDeterministic**

**Node.Kind**

**Node.Label**

**Node.MutualInfo (Node="TargetNode(0)")**  
**Node.MutualInfo**  
**(Node="TargetNode(0)", Fraction=Percent)**  
**Node.MutualInfo (Node=0)**  
**Node.MutualInfo (Node=0, Fraction=Percent)**  
**Node.Name**  
**Node.NumberStates**  
**Node.Real (State=0)**  
**Node.RelativesList (Generation=-3, Seperator=", ")**  
**Node.RelativesTable (StartGeneration=-4,**  
**EndGeneration=2, StyleHtml=table)**  
**Node.SensitivityTable (MutualInfo)**  
**Node.SensitivityTable (TextFormat, MutualInfo)**  
**Node.StateComment (State=0)**  
**Node.StateLabelsList (Seperator="</td><td>")**  
**Node.StateName (State=0)**  
**Node.StateTitle (State=0)**  
**Node.ThresholdLower (State=0)**  
**Node.ThresholdUpper (State=0)**  
**Node.TitleNode.Type**  
**Node.VarianceReal**  
**(Node=0, IndicatesUnknown="unknown")**  
**Node.VisualPositionNode.VisualStyle**

## Built-In Generic Report Template Files

Netica comes with a number of pre-made template files for [common types of reports](#), and they might be just what you need. If you require a more customized approach, you can [create your own template](#) file. If so, you may want to use one of the generic ones as a starting point, or examine it for ideas on how to make your own.

**Location:** Your Netica download comes with a batch of pre-made report templates. In the main directory of your Netica installation, you will see a folder called **Report Templates** (the path to this folder is printed out in the Messages window of Netica Application when it first starts up).

The generic files `Net All.nsp.htm` and `Node All.nsp.htm` are especially useful, since they contain many possible methods for displaying information.

For example, for a report on the table of a node, you could use:

Generic/Report Templates/Node Table.nsp.htm

Here is a list of the generic reports available:

Net All.nsp.htm

Net Beliefs.nsp.htm

Net Case.nsp.htm

Net Comment.nsp.htm

Net Comment Unformatted.nsp.htm

Net Developer.nsp.htm

Net Equations.nsp.htm

Net Findings.nsp.htm

Net Graphic.nsp.htm

Net Sensitivity.nsp.htm

Net User.nsp.htm

Node All.nsp.htm

Node Description.nsp.htm

Node Developer.nsp.htm

Node Equation.nsp.htm

Node Parents.nsp.htm

Node Sensitivity.nsp.htm

Node Table.nsp.htm

Node User.nsp.htm

Node Web\_Description.nsp.htm

## Equations

The relationship between a node and its parent nodes can be entered [using an equation](#) if desired. This is possible whether the nodes are continuous or discrete, and whether the relationship is probabilistic or deterministic. Netica will convert all equations into tables (= 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_function_table.htm');return false;">function table) before compiling a net, doing net transforms, or expanding a DBN. The tables are then used in the same way as if you had entered them by hand.`

The following sections outline further information on Netica's equations:

- [Using Equations](#)
- [Syntax](#)
- [Converting to a Table](#)
- [Comparison to Java /C/C++](#)
- [Conditionals](#)
- [Link \(Input\) Names](#)
- [State Names as Constants](#)
- [Discrete Variables with State Values](#)
- [Internal Form of Equations](#)
- [Constant Nodes as Adjustable Parameters](#)
- [Examples of Equation Use](#)
- [Built-in Constants](#)
- [Built-in Functions and Distributions](#)

## Using Equations

Follow these steps to enter and use an [equation](#) to represent a node's relationship with its parents:

1. Enter the equation into the [node dialog box](#) in the [correct form](#), perhaps using [built-in functions](#). Here are some [examples](#).
2. If the node is continuous, or has any continuous parents, they must be [discretized](#).
3. [Convert](#) the equation to a table.
4. Use the net (e.g. compile it, do belief updating, absorb nodes, reverse links, or solve decision problems). Netica will use the table generated from the equation.

If you change the equation for a node, or the discretization of the node or any of its parents, or the value of a relevant [constant node](#), you must repeat step 3 above before the changes take effect.



### Tips:

- If the equations get large, it may be easier to create them in a `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_text_editor.htm');return false;">text editor`, and then paste them into the node dialog box.
- When editing an equation you can cut, copy, paste, undo, etc. using their `CTRL` key commands, or by `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-clicking`.
- If there is a mistake in the syntax of the equation, Netica will pop up an alert box with an error message. If necessary, move the alert box so that the equation is unobscured, and then click on the title bar of the Node Properties Dialog box. That will cause the Node Properties to become the active window, and a selection point will appear in the equation text close to where the error is located.
- You can [print out all the equations](#) in the net, or [select the nodes with](#)

equations.

- If you need to define intermediate variables to simplify the equations, implement them as new (intermediate) nodes.
- The tables generated by equations may result in large files (and therefore slow reading), so you may want remove the node's table with **Table** → **Remove** or  before saving the net to file. When you later read it in, do **Table** → **Equation to Table** or  (with no nodes = 4 && `typeof(BSPSPopupOnMouseOver) == 'function'`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_deselect_nodes.htm');return false;">selected)` before using it.

## Equation Syntax

Netica [equations](#) follow most of the usual standards for mathematical equations, and are similar to programming in [Java](#), [C](#) or [C++](#). The usual mathematical operators (+, -, \*, /, etc.), and the usual functions (min, abs, sin, etc.) can be used, parenthesis are used for grouping, and numeric constants are in their usual form (e.g. 3, -4.2, 5.3e-12).

**Left-Hand Side:** The part of an equation to the left-hand side of the equals symbol for a deterministic node consists of the name of the node, an open parenthesis, a list of the names of the parents separated by commas, and a close parenthesis (if you have defined [link names](#), you **must** use those instead of parent names). For instance, if the equation is for node X, and the parents of X are Vel, dt and sigma, the left-hand side could be:

$$X(\text{Vel}, \text{dt}, \text{sigma}) = \dots$$

Note that spaces are not required, and there may be more spaces if desired.

For probabilistic nodes (i.e. `"= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_chance_node.htm');return false;">chance nodes`), the left-hand side consists of a lower case "p", an open parenthesis, the name of the node, a vertical bar, a list of the names of the parents (or [link names](#)) separated by commas, and a close parenthesis. If the node mentioned above had been a probabilistic node, the left hand side of its equation could be:

$$P(X | \text{Vel}, \text{dt}, \text{sigma}) = \dots$$

**Right-Hand Side:** The right-hand side of an equation may consist of numbers, [adjustable parameters](#), [state names](#), [conditionals](#), [references](#) to the variables (i.e. parent nodes), and built-in [constants](#), [functions](#) or [operators](#).

**Nodes Allowed:** The only nodes which may be mentioned in an equation are: the node the equation describes, its parents, and any [constant node](#).

**Errors:** If there is a mistake in the syntax of the equation, Netica will pop up

an alert box with an error message. If necessary, move the alert box so that the equation is unobscured, and then click on the title bar of the Node Properties Dialog box. That will cause the Node Properties to become the active window, and a flashing cursor will appear in the equation text close to where the error is located.

**Comments:** Comments can be embedded in equations, and they will be ignored by Netica. Everything between `/*` and `*/` will be interpreted as a comment, as will everything between `//` and the end of the line. As many spaces or line breaks as desired may be placed between any two symbols.

**All Values:** If the equation is for a probabilistic node, its right-hand side **must** provide a probability for all the node's possible values (so the name of the node must appear there at least once). For example, if node `Color` (with states *red, orange, yellow*) has parent `Temp` (with states *low, med, high*), its equation could be:

```
p (Color | Temp) =
Temp == high ? (Color==yellow ? 1.0 : 0.0) :
Temp == med ? (Color==orange ? 1.0 : 0.0) :
Temp == low ? (Color==orange ? 0.2 :
Color==red ? 0.8 : 0.0) : 0
```

If you use the built-in distributions (such as [NormalDist](#)), the above is automatically taken care of. One exception to the above is if a node is `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_boolean_node.htm');return false;">boolean.`

Then only the probability for the *true* state need be given. For example, if node `It_Falls` is boolean, its equation could be:

```
p (It_Falls | Weight, Size) =
Weight/Size > 10 ? 0.10 :
Weight/Size > 5 ? 0.03 :
0.01
```

**Examples:** Here are some examples of equations:

```
X (Vel, dt, X0) = X0 + Vel * dt
```

$p(X | \text{Vel}, dt, \text{spread}) = \text{NormalDist}(X, \text{Vel} * dt, \text{spread})$

Color (Taste) =

Taste==sour? blue: Taste==sweet? red:

Taste==salty? green: gray

$p(\text{Color} | \text{Taste}) =$

(Color==red && Taste==sweet) ? 0.7 : 0.1

## More Examples

# Converting an Equation to a Table

Before doing most net operations (e.g. compiling, absorbing nodes, reversing links, or optimizing decisions), each involved node must have a relation contingency table (e.g. its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT) to express its relationship with its parents. If you have entered the relationship as an equation, then a table must be built from the equation.

To build tables for particular nodes, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the nodes, and then choose **Table** → **Equation to Table** or click the  toolbar button. If some of the selected nodes don't have equations, they will just be skipped. If no nodes are = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_deselect\_nodes.htm');return false;">selected, then Netica will build tables for all the nodes which have equations.

If a node with an equation, or any of its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents, are for continuous variables, then they must be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretized before converting the equation to a table (except a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_utility\_node.htm');return false;">utility node).

When Netica does the conversion, it must decide where within each discretized “cell” to evaluate the equation. It chooses a number of points (at truly random positions) within the cell, and uses the average of the results obtained. Just before the conversion process, Netica will put up a dialog box asking you how many random samples to make per cell. The larger this

number, the more accurate the results will be, but the longer it will take.

**Sampling Uncertainty:** Since this sampling process can't check every point within the cell, it may miss a narrow spike or ridge. Netica can represent within the created table the additional uncertainty due to the sampling, or it can assume that the sampling was truly representative. Netica will put up a dialog box asking you which it should do. It is best to ignore the sampling uncertainty when you know the equation doesn't have any narrow spikes or ridges.

When the sampling uncertainty is included, then none of the table entries will be zero, since Netica can never be sure that it hasn't missed some nonzero points within the cell. The sampling uncertainty will be smaller if you increase the number of samples per cell. Sampling uncertainty can be removed after the table is created by choosing **Table** → **Harden**, and providing a degree of 1.

**Large:** The size of the table generated is the product of the number of states of the node with the numbers of states of each of its parent nodes. So if a node has many states, or many parents, then the tables may be very large, and Netica may report that it doesn't have enough memory for the operation. Or it may just take a long time. You can alleviate the problem by eliminating unnecessary parents and using coarser discretizations (perhaps have more than one node for the same variable, with [different discretizations](#) depending on which node it is a parent for).

## Equation Syntax vs. Java or C/C++ Languages

The [equation syntax](#) is precisely the same as the Java (and C and C++) programming languages, except the part to the left of the assignment operator (=) is different, and no semicolon is required at the end of the equation.

The C/C++/Java bitwise operators (such as &, |, ~, ^) are not available in Netica, but the logical operators &&, ||, ! are. In addition Netica has a logical 'xor' function (the bitwise xor operator ^ of C/C++/Java is used for the power operator by Netica).

All of the C Standard Library math functions (sin, log, sqrt, floor, etc.) are available and use the same names.

## Equation Conditionals

Suppose continuous node  $X$  has the parents  $Y$  and  $B$ . If you wanted to give  $P(X|Y)$  a different [equation](#) involving  $X$  and  $Y$  for different values of  $B$ , you could write something like.

```
p(X|Y,B) =  
(B < 2) ? NormalDist (X, 3 + Y, 1) :  
(B < 6) ? NormalDist (X, 2 + Y, 3) :  
UniformDist (X, 0, 10)
```

The conditions are evaluated in order, so the first covers all cases where  $B < 2$ , the second covers cases  $2 \leq B < 6$ , and the last covers the remaining cases (i.e.  $B \geq 6$ ). So, if  $B$  is less than 2,  $X$  is distributed normally with mean  $3+Y$ , if it is between 2 and 6 then the mean is  $2+Y$ , and if it is over 6 then  $X$  is distributed uniformly.

If there are more parents, this sort of construct can be nested to provide a tree structure of possible contingencies.

Here are a couple more examples. They show a way to condition over the states of a `4` `&& typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_discrete_node.htm');return false;">discrete`  
`node:`

```
p(X|Y,B) =  
(B == yellow) ? NormalDist (X, 2, sqrt (Y)) :  
(B == orange) ? NormalDist (X, 4, Y) :  
(B == red) ? NormalDist (X, 6, Y ^ 2) : 0
```

```
p(X|B) =  
member (B, CA, TX, FL) ? NormalDist (X, 3, 1) :  
member (B, MA, WA) ? NormalDist (X, 5, 1) :  
member (B, NY, UT, VA) ? NormalDist (X, 7, 2) :  
UniformDist (X, 0, 10)
```

Notice that the “fall through” case of the first example is simply a 0. This indicates that the designer is counting on  $B$  to be one of yellow, orange or red. If  $B$  ever has another state, then when Netica is converting the equation to a

table it will give a warning message that “for  $n/N$  conditions, no nonzero probability was discovered by sampling” (providing no sampling uncertainty is being added).

In the last example, the fall through case gives a uniform distribution. If extra states are later added to B, then they will just fall through and use the uniform distribution.

## Link Names

When you first start working with [equations](#), you will probably use the names of the parent nodes in your equations. However, sometimes you will want a more local representation, so that you can disconnect some of the parents and hook the node up to new parents without having to change all the node names within the equation.

Or perhaps you copy and paste the node to use with new parents. Or you put the node in a net fragment library without any parents, so that it will be supplied with new parents when it is used. Or you want to copy and paste the equation from one node to another, without changing all the node names.

The solution to all these problems is to use *link names*, sometimes called *input names*. They provide an argument name for each link entering the node (and therefore a proxy for each parent node). You can [set them](#) with the node dialog box. You refer to them in your equation in exactly the same way you would the corresponding parent name. When a parent is disconnected, the link name will remain.

If link names are defined for a node, they **must** be used instead of the parent names.

## State Names as Constants

You can use the state names of a `a = 4 && typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_discrete.htm');return false;">`discrete node as constants in an [equation](#). For example, if node *Color* has states *red*, *green*, *blue* and *yellow*, and node *Temperature* has states *cool* and *warm*, you could write:

```
Temperature (Color) = member (Color, red, yellow) ? warm : cool
```

**Specifying Node:** Each state name only has meaning relative to the node it's for. Usually when you use a state name, Netica can identify that node from context. However, if Netica doesn't know which node a state name refers to (e.g. it gives an unknown value error message), you can indicate which node by following the state name with a double-dash and then the name of the node. Continuing with the above example, if node *Switch* had the states 0, 1 and 2, you could write:

```
Color (Switch) = select0 (Switch, red--Color, yellow, blue)
```

The "--Color" was not required on "yellow" and "blue", because the context was carried over from "red--Color", but it could be put there as well.

**State Numbers:** Instead of state names, you can just use the state number (numbering starts at 0), but it is highly recommended to use the names, because they are more readable and less error-prone. Also, it is not as serious if later states are added or re-ordered.

## Discrete Variables with State Values

When a node equation contains a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_discrete.htm');return false;">` discrete variable which has `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_state_value.htm');return false;">` state values defined, the numeric quantities which that variable supplies to the equation could be the state indexes of the variable, or the values associated with the state indexes. Usually a function or operator will assume it is the values that are desired, but in a few cases it will use the state indexes, for instance if the other arguments to it are state indexes.

For example, consider the equality operator, and a discrete variable Color, which has values and state names (such as blue) defined. When Color appears in the equation, it could refer to a state index of Color, or to the value associated with that state index.

Uses state index: `Color == blue`    `Color == #1`    `#Color1 == #Color2`

Uses state value: `Color == 3.2`    `Color == 1`    `Color1 == Color2`

The above is for the evaluation of equation expressions. When it comes to assigning the result of the equation to its LHS (left-hand side) variable, if the variable is discrete with values defined, the same issue arises. Has the equation calculated a state index for the variable, or a value associated with the state? The purpose of the `AsState` function is to indicate that state indexes should be used.

Continuing with the example above:

Assigns by value: `Color() = 3`    `Color1 (Color2) = Color2`

Assigns by index: `Color() = #3`    `Color1 (Color2) = AsState (Color2)`

`Color1 (X, Y) = AsState (integer (X / Y))`

You may want to verify that Netica is doing as you expect by having it calculate a few values and checking (for example, by using **Table** → **Equation to Table**). Or you can examine the [internal representation](#) of the equation by choosing **Report** → **Horizontal Format** from the menu and then **Report** → **Equation**. Check for places where Netica has inserted the function "\_levels" (maps **state index** → **real value**), "[find0](#)" (maps **real value** → **state index** of a discrete variable), or "\_discretize" (maps **real value** → **state index** of a continuous variable).

## Internal Form of Equations

Netica converts [equations](#) to an internal form for fast evaluation. Sometimes that means extra functions are invoked that weren't in the original equation; these functions always start with an underscore.

You may receive an error message that indicates there was a problem evaluating one of these functions. Generally that occurs because in your equation you supply values to a function that are out of its domain.

If you want to see what the internal representation of the equation in Netica looks like, you can generate a [report](#) by choosing **Report** → **Horizontal Format** from the menu and then **Report** → **Equation**.

Some example internal functions are:

[\\_not](#) [\\_discretize](#) [\\_find0](#)  
[\\_levels](#) [\\_select0](#) [\\_Bernoulli](#)

## “Constant” Nodes as Adjustable Parameters

You create a [constant node](#) by [adding](#) a nature node to the net, bringing up its [node dialog box](#), and choosing “Constant” from the [node kind selector](#). You can also set other characteristics of a constant node in the same way as any other node, such as giving it state names.

To set or change the value of a constant node, enter the value in the same way as you would [enter a finding](#).

**For Equations:** You can refer to the value of a constant node anywhere in any node [equation](#) by using its node name as you would a parent node. It should not appear in the argument list to the left of the = symbol. No link is required.

When you convert the equation to a table, the value of any referenced constant nodes will be used. If you change the value of a constant node, you must [rebuild](#) the table for the change to take effect.

## Examples of Equation Use

Here are example [equations](#) for some common or non-obvious situations:

**State Comparisons:** Suppose the states of node Source are CA, TX, FL, BC and NY. The states of node Dest are TX, NY, MA and UT. We want to know if cross-border travel is required to transport from Source to Dest, and that is indicated by the `boolean Travel`. The equation below works even though nodes Source and Dest have different sets of states, and in a different order.

$$\text{Travel}(\text{Source}, \text{Dest}) = (\text{Source} \neq \text{Dest})$$

**Additive Noise:** Say you want to represent something like:

$x_1 = x_2 + \text{gauss}(0, 0.2)$  which could indicate that  $x_1$  is the same as  $x_2$ , but with the addition of Gaussian noise having mean 0 and  $\sigma = 0.2$ . You could do this by defining a new node  $x_3$ , and setting the equations of  $x_1$  and  $x_3$  as:

$$x_1(x_2, x_3) = x_2 + x_3$$

$$p(x_3) = \text{NormalDist}(x_3, 0, 0.2)$$

**Multiple Discretizations:** Sometimes it is useful to use more than one node to represent a continuous variable, and discretize each differently. For example, the coarser one may be a parent for another node whose CPT would be too big with a finer discretization, while the finer one would serve as a parent for nodes requiring more accuracy. Put a link from the finer node to the coarser, and give the coarser node an equation like:

$$X_{d4}(X_{d12}) = X_{d12}$$

**Noisy-Or:** To create a noisy-or node, just [create](#) a regular `boolean` node, put links to it from the possible causes, give it a noisy-or equation, and use that to [build its CPT](#). For extra computational efficiency you

may want to select it and choose **Modify** → **Decompose Equations** before building the CPT.

For example, if C1, C2 and C3 are `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_boolean_node.htm');return false;">boolean nodes` representing causes of boolean node E, and there are links from each C<sub>i</sub> to E, then E could have the noisy-or equation:

$$P(E | C1, C2, C3) = \text{NoisyOrDist}(E, 0, C1, 0.5, C2, 0.3, C3, 0.1)$$

For its meaning, see the [NoisyOrDist](#) description. The causes, and even the link parameters, can be more complex expressions (then you won't be able to use **Modify** → **Decompose Equations** though).

**For example:**

$$P(\text{Bond} | \text{Temperature}, \text{BackTemp}, \text{Pressure}, \text{Switch}, \text{Eff}) = \text{NoisyOrDist}(\text{Bond}, 0.001, \text{Temperature} > \text{BackTemp}, 0.5, \text{Pressure} == \text{high}, 0.3, \text{Switch}, 0.9 * \text{Eff})$$

## Built-in Constants

The following constants may be used in [equations](#):

$$\text{pi} = 3.141592654$$

$$\text{deg} = \text{radian per degree} = \text{pi} / 180$$

If you wish to have the constant **e** (= 2.7182818) in your equation, use `exp(1)`.

Netica also contains an extensive library of built-in [functions](#).

**Example:** `sin (35 * deg)` returns the sine of 35 degrees

## Built-in Functions

Netica contains an extensive library of built-in functions and [constants](#) which you can use in your [equations](#).

The probability distribution functions all have a name that ends with "Dist" (e.g. NormalDist). Their first argument is always the node for which the distribution is for. So if node X has parent m, you could write:

$$P(X | m) = \text{NormalDist}(X, m, 0.2)$$

to indicate that X has a normal (Gaussian) distribution with mean given by parent m, and a standard deviation of 0.2.

## Common Operators

<u>Functional Notation</u>	<u>Operator Notation</u>
neg (x)	- x
not (b)	! b
equal (x, y)	x == y
not_equal (x, y)	x != y
approx_eq (x, y)	x ~= y
less (x, y)	x < y
greater (x, y)	x > y
less_eq (x, y)	x <= y
greater_eq (x, y)	x >= y
plus (x1, x2, ... xn)	x1 + x2 + ... + xn
minus (x, y)	x - y
mult (x1, x2, ... xn)	x1 * x2 * ... * xn
div (x, y)	x / y
mod (x, base)	x % base
power (x, y)	x ^ y
and (b1, b2, ... bn)	b1 && b2 && ... && bn
or (b1, b2, ... bn)	b1    b2    ...    bn
if (test, tval, fval)	test ? tval : fval

## Common Math

abs (x)	absolute value
sqrt (x)	square root (positive)
exp (x)	exponential ( $e^x$ )
log (x)	logarithm base e
log2 (x)	logarithm base 2
log10 (x)	logarithm base 10
sin (x)	sine
cos (x)	cosine
tan (x)	tangent
asin (x)	arc sine
acos (x)	arc cosine
atan (x)	arc tangent
atan2 (y,x)	atan(y/x) but considers quadrant
sinh (x)	hyperbolic sine
cosh (x)	hyperbolic cosine
tanh (x)	hyperbolic tangent
floor (x)	floor (highest integer $\leq x$ )
ceil (x)	ceiling (lowest integer $\geq x$ )
integer (x)	integer part of number (same sign)
frac (x)	fraction part of number (same sign)

## Special Math

- round (x)
- roundto (dx, x)
- approx\_eq (x, y)
- eqnear (reldiff, x, y)
- clip (min, max, x)
- rect (x, a, b)
- sign (x)
- xor (b1, b2, ... bn)
- increasing (x1, x2, ... xn)

- increasing\_eq (x1, x2, ... xn)
- avg (x1, x2, ... xn)
- mag (x1, x2, ... xn)
- min (x1, x2, ... xn)
- max (x1, x2, ... xn)
- argmin0/1 (x0, x1, ... xn)
- argmax0/1 (x0, x1, ... xn)
- nearest0/1 (val, c0, c1, ... cn)
- select0/1 (index, c0, c1, ... cn)
- member (elem, s1, s2, ... sn)
- factorial (n)
- logfactorial (n)
- logistic (t)
- logit (p)
- gamma (x)
- loggamma (x)
- beta (z, w)
- erf (x)
- erfc (x)
- binomial (n, k)
- multinomial (n1, n2, ... nn)

## **Continuous Probability Distributions**

- UniformDist (x, a, b)
- TriangularDist (x, m, w)
- Triangular3Dist (x, m, w1, w2)
- TriangularEnd3Dist (x, m, a, b)
- NormalDist (x,  $\mu$ ,  $\sigma$ )
- LognormalDist (x,  $\eta$ ,  $\phi$ )
- ExponentialDist (x,  $\lambda$ )
- GammaDist (x,  $\alpha$ ,  $\beta$ )

- WeibullDist ( $x, \alpha, \beta$ )
- BetaDist ( $x, \alpha, \beta$ )
- Beta4Dist ( $x, \alpha, \beta, c, d$ )
- CauchyDist ( $x, \mu, \sigma$ )
- LaplaceDist ( $x, \mu, \beta$ )
- ExtremeValueDist ( $x, \mu, \sigma$ )
- ParetoDist ( $x, a, b$ )
- ChiSquareDist ( $x, v$ )
- StudentTDist ( $x, v$ )
- FDist ( $x, v1, v2$ )

## **Discrete Probability Distributions**

- SingleDist ( $k, c$ )
- DiscUniformDist ( $k, a, b$ )
- BernoulliDist ( $b, p$ )
- BinomialDist ( $k, n, p$ )
- PoissonDist ( $k, \mu$ )
- HypergeometricDist ( $k, n, s, N$ )
- NegBinomialDist ( $k, n, p$ )
- GeometricDist ( $k, p$ )
- LogarithmicDist ( $k, p$ )
- MultinomialDist ( $bc, n, k1, p1, k2, p2, \dots km, pm$ )
- NoisyOrDist ( $e, leak, b1, p1, b2, p2, \dots bn, pn$ )
- NoisyAndDist ( $e, inh, b1, p1, b2, p2, \dots bn, pn$ )
- NoisyMaxDist (...)
- NoisySumDist (...)

## Netica-Web

Netica-Web is the newest product in the Netica family. It is a system to deploy your Bayes nets over the internet as a question-answer system. Such a system asks the user questions, or provides a dashboard to enter relevant information, and presents the user with conclusions. At each stage of the process, the system chooses the best questions to ask based on the previous answers given, using Netica's [sensitivity analysis](#). It may continuously provide conclusions, or wait until it reaches a certain level of confidence before presenting any information. [Contact](#) Norsys for full documentation and pricing information (although you may try it out now as described below).

**How to:** With Netica-Web, you can easily build an interactive website, also known as a HED system (Human- Electronic Dialog). You [construct a Bayes net](#) in the normal manner using Netica Application, and then with a single click of a button, a website is generated. First, select a node of interest (called a Target node), then choose **Netica** → **Build Website**. Alternatively, you can right-click on the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_background.htm');return false;">background` and choose **HED System** → **Add Net to Website**. In a moment your browser should spring up with your working system. You can copy/paste the URL from your browser to publish your site to the world. [Example site](#)

**Applications:** Netica-Web can be used to build diagnostic systems, such as troubleshooting for some machinery or plant, or to provide medical, environmental, corporate, etc. diagnostics. It can be used to build systems that identify some particular item(s) from a large set based on indirect criterion, such as a product to be purchased from an inventory, a movie or song selection, restaurant choice, person, etc. It can build classification systems to determine which group an item falls into, based on its characteristics, such as the personality type of a person, probable voting patterns, credit worthiness, whether a plant or product is poisonous, etc. Although these are typical applications, any Bayes net can be run on Netica-Web.

**Netica-Web versus Decision Trees:** Netica-Web is superior to old tree-based methods because it is easier to build and maintain a Bayes net than a decision tree. Bayes nets represent their knowledge in a form which is easy to

understand, and which can combine human expertise with learning from data. Furthermore, their modularity allows combining sub-models into a complete model. Any changes in probability distributions can drastically alter the structure of the optimum decision tree, whilst they produce a small change in a Bayes net. Bayes nets allow users to skip questions, answer “unknown”, or provide likelihood answers, and still maintain optimal performance. And they can recover from incorrect answers, which often throw decision trees on the wrong track.

**Constructing Nets for Netica-Web:** When constructing the Bayes net, its nodes are divided into (possibly overlapping) groups: observable, target and intermediate. Each observable node becomes a potential question in the system, and each target node will result in an answer, or have its probabilities displayed. Netica-Web determines which questions are the most relevant to determine the values of the target results. It then presents those questions first, either one at a time, or a few at a time in a scrolling list (depending on the configuration you set). The user may skip questions at any time, in which case the questions are moved to another part of the screen, to allow later answering if desired.

Since only the most relevant questions are asked, based on the previous answers, a HED session has the feel of an interactive dialog, rather than just filling out a form. Since the system is so judicious in choosing which question to ask from its pool of possible questions, it is okay to have a very large set of possible questions, further adding to its ability to reach good conclusions.

A working HED system can be built from a regular Bayes net with a single click, (**Network** → **Build Website**) although if you want to you can highly customize it. Its visual appearance, dimensions, colors, behavior, etc. can all be easily changed, and graphics can be added so that it is seamlessly integrated with a larger website.

**Hosting:** Your HED system, built as described above, is being hosted by Norsys, but if for security or other reasons you wish to host it yourself, that is also an option. Either way, if you wish, it can become part of your existing website. Alternately, the “web page” can just be local to the machine it is running on, allowing you to distribute your solution as a desktop application that does not require the internet.

To build your first HED system, be sure you have Netica 5.0 or later installed

on your machine and read the Netica-Web manual.

For more information, [email Norsys](#).

## Geo-Netica

Many Netica users have, for some time been using Netica in conjunction with GIS (Geographic Information Systems), so we decided to make it easier for you.

The result is Geo-Netica, our latest Netica product, which brings Bayes nets to the 2D world of GIS.

This product is in its early stages, but since it is already very useful to some, we have decided to make it available for sale to select clients now, with the idea that they are entitled to new upgrades as features are added. Additionally, we will listen closely to early clients, to decide which new features to add.

### **Currently it:**

- Operates on raster data, such as raster output files from ArcGIS.
- Works inside Netica window.
- Shows multiple input and output GIS images at desired resolution.
- Allows processing at lower resolution for quick exploration and model building before doing a full resolution processing.
- Can click on any point within an image to enter data from that location from each of the images into the Bayes net to see how the Bayes net processes it.
- Can hover cursor over image, to display data on that point.
- Can auto-discretize nodes based on image data.
- Can directly interact with the Bayes net on each processing cycle, for quick what-if analysis.
- Can handle input images that are not exactly the same location, and different resolutions.
- Works with any Bayes net model, including ones with equations, probabilistic tables, learned from data.
- Bayes net could be for prediction, classification, diagnosis, probabilistic processing, species models, water management, etc.
- Example input images are elevation, slope, aspect, ground vegetation, species population, water, climate, rainfall, soils, satellite images,

traversability, friend/foe positions, municipal zoning, building type, population density, pollutant levels, mineral assays, rock type, accessibility, marketing data, income levels, crime rates, property costs, political party, etc.

**Features coming soon:**

- Can learn Bayes net models from GIS data
- Easier for Bayes net models to include data from other pixels near to the one being operated on

**Other planned features:**

- Operate on vector data as well as raster data
- Interface nicely with commercially available GIS systems

If you are interested in purchasing Geo-Netica, please [= 4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\) BSPSPopupOnMouseOver\(event\);" class="BSSCPopup" onclick="BSSCPopup\('X\\_PU\\_email.htm'\);return false;">contact Norsys](#)

## Node-Sets

Sometimes it is useful to form ‘groups’ or ‘categories’ of nodes. With Netica, you can create groups of nodes, and to assign a name and color to each. **Note:** A related concept is to [group states](#) of a single node.

For instance, to clearly show the divisions between various parts of your Bayes net, you can make each subsection a different color.

Or, if there is a set of nodes that you repeatedly use for some operation (for example, a subset of nodes that you often want to remove the findings from), you can [set](#) those nodes as a node-set. Whenever you want to do the operation, you have Netica [select](#) the nodes in that node-set, and then you choose the operation from the menu.

A node may belong to several different node-sets at once, so in that case Netica needs a way to choose which node-set color to use when coloring it.

Node-sets have a priority ordering, so you can choose the most important criteria to use in [coloring](#) them, which you may want to change from time to time to view the net in different ways.

The node-set options are:

- [Creating Node-Sets](#)
- [Adding and Removing Nodes from Node-Sets](#)
- [Node Coloring](#)
- [Using Node-Sets to Select Nodes](#)
- [Node-Set Reporting](#)

# Creating Node-Sets

If there is a particular set of nodes that you work with frequently, you can select them all and give them a name, creating a [node-set](#). At any time during your work you can [select](#) this set quickly by pressing the `CTRL+SHIFT+S` buttons simultaneously, and entering the name.

There are two ways node-sets are created:

**1. By User:** Select the node(s) you want in a particular node-set, choose **Modify** → **Set NodeSet** → **New**, (the [shortcut key](#) for this is `CTRL+SHIFT+N`) and enter the name of the new set. Node-set names must follow the rules of an `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_IDname.htm');return false;">IDname`. After a node-set is set, you can later [add or remove](#) nodes from it.

**2. Built-In:** Automatically built-in definition, based on intrinsic qualities of the nodes. These nodes are denoted with a dash in front of the name (e.g. – ConstantValue).

The following is a list of the built-in node-sets, in order of their priority for [coloring](#) (*Likelihood Finding* highest priority, *Finding* second priority etc.)

LikelihoodFinding	Finding	Deterministic
Boolean	TwoState	Discrete
Continuous	Nature	Title
Documentation	ConstantValue	Constant
DecisionSolved	Decision	Adversary
Utility	Equation	HasTable
Parentless	Childless	Node

**Note:** Some of these node-sets won't always appear in the [NodeSet Properties](#) dialog box. To access them, choose **Modify** → **Set NodeSet** → **New**, and type in a name from the above list (remembering to begin the name with a dash).

## Adding and Removing Nodes from Node-Sets

As you work with your net, you may want to add or remove nodes from a previously [created node-set](#). Unlike [adding](#) and [removing](#) nodes from your Bayes net (which could affect the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">cpts`), adding and removing nodes from a node-set will only affect the categorization of the node(s) within the net.

**Adding:** `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_select_node.htm');return false;">`Select the node(s) you wish to add to the set, choose **Modify** → **Add Nodes to Set**, and click on the name of the desired node-set, or click on **Enter** and type in the name. Alternately, you can select one or multiple nodes, `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_right_click.htm');return false;">`right-click on the selection and choose **Add To Nodeset**.

If you right-click an unselected node, the menu will just contain **Nodeset** instead of **Add To Nodeset**, **Remove From Nodeset** and **Set Nodeset**, which allows you to `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_toggle_menu.htm');return false;">`toggle whether or not the node clicked-on is in the nodeset you choose.

**Removing:** Select the node(s) you wish to remove from the set, choose **Modify** → **Remove From NodeSet**, and click on the name of the desired node-set, or click on **Enter** and type in the name.

Alternately, you can use right-clicking to remove them in a manner similar to that described above for adding.

## Node-Set Dialog Box

If a node, or nodes, have previously been [created](#) as a [node-set](#), they can be assigned a color. Coloring node-sets can be very helpful visually when working with a Bayes net, and is one of the [style options](#) available for improving the presentation and comprehension of a net.

**How To:** Choose **Modify** → **NodeSet Properties** (from the overhead menu or by = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-clicking
on the net = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_background.htm');return false;">background)
```

The subsequent dialog box contains a list of all the node-sets created within the Bayes net.

### Using this dialog box you can:

**Set Color:** Before a color has been assigned to a [user-defined](#) node-set, an X will appear in the color swatch box next to its name. To choose or change the color of a node-set, click on it and click **Set Color**. A color palette will be displayed from which you can make your choice. Click **Okay** to dismiss the palette, and then click **Apply** or **Okay** in the NodeSet properties dialog box.

You can also change the [default color](#) for the other nodes of your net at this time.

**No Color:** To remove the color of a node-set, click on it, then click **No Color**.

The X that will appear in it's color box indicates that this node-set has no assigned color, and when Netica is choosing a color for nodes, it will skip this node-set and move down to the next node-set in the list. To restore the node-set's previous color, click on it and click **Set Color**. The last chosen color is remembered; hence you can just click **Okay** in the color palette which appears.

**Change Priority of Color Displayed:** If a node belongs to more than one node-set, the first node-set listed in the Properties dialog box will determine the color displayed. To change the color displayed, drag the node-set title up or down the list to the desired position.

**Rename:** To change the name of a node-set, select it's name and click

**Rename.** Type the new name into the dialog box that appears.

**Delete Node-Set:** To delete a node-set, click on it's name and press the `DELETE` key.

## Selecting Nodes in a Node-Set

To restrict an operation (e.g. clearing/setting findings, removing tables, copying nodes, absorbing nodes, modifying node kind or discretization, hiding, etc.) to a limited set of nodes, you often select them first. If you have a certain set of nodes that you often do an operation on, it is useful to [create](#) a node-set of them, so that each time you can quickly select the nodes in that node-set.

To restrict an operation (e.g. compiling, absorbing nodes, reversing links, or optimizing decisions) to a previously created [node-set](#), you can quickly select the nodes within it.

**How To:** Choose **Edit** → **Select Nodes** → **In NodeSet**, and then select the desired node-set from the list of names, or **Enter** the name. The [shortcut](#) key for this command is `CTRL+SHIFT+S`.

Once the selection has been made, it will behave like any other [node selection](#).

There is an alternate way to consistently select a certain group of nodes, if all those nodes have the same keyword in their title or description (and other nodes don't). First use **Edit** → **Find** to enter the keyword, then use **Edit** → **Find All** to select them all.

## Node-Set Reporting

Under the **Report** menu is a **NodeSets** option. Use this function to generate a [report](#) showing all nodes within a previously [created node-set](#).

**How To:** Choose **Report** → **Node Sets** → **All** to generate a detailed list of all the node-sets. If you want a report on a specific node-set, choose **Report** → **Node Sets** and click on the desired node-set name, or on **Enter** to label a set of nodes for the report.

To have all the nodes of each node-set appear on one line, first choose **Report** → **Horizontal Format**. If pasting the list into a table or cell, you may also want to choose **Report** → **Tab Separators**.

## Dynamic Bayes Nets

Dynamic Bayes nets are also known as DBNs or temporal Bayes nets. They allow you to specify a Bayes net model which has "= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_time\_delay\_link.htm');return false;">time delay" links, indicating that the value of the child node depends on the value of the parent at an earlier time. Later you expand the DBN so that some nodes in the original net become several nodes in the expanded net, indicating the value of that variable at different points in time.

DBNs can have directed cycles, as long as there is a delay link somewhere along each cycle. Delay links can be used to model feedback. Once it is expanded, it will no longer have cycles or delay links.

We improve Netica's DBN capability with each release, so if you are working with DBNs you should use at least version 5.02. Download latest versions from our [ftp site](#).

It is usually easiest to work through an example DBN first. The Bayes net called "Bouncing", in Netica's "= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_examples\_folder.htm');return false;">Examples" folder, is suitable for that purpose.

Steps for working with a Dynamic Bayes Net:

1. [Create DBN](#)
2. [Generate time expansion](#)
3. [Compile and use](#)

If you have any requests or suggestions for Netica's DBN feature, be sure to = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_email.htm');return false;">contact our support team, since we are actively improving this feature.

See also [DBN Bouncing Example](#)



# Using Dynamic Bayes Nets

## 1. Create DBN

2. Generate time expansion

3. Compile and use

Netica has unique [DBN](#) capabilities not available with any other software.

Different [links](#) may have different time delays, so that when the network is expanded, time slices may have a different structure from each other. In the expanded net, Netica will replicate nodes at a frequency that is appropriate to model the dynamic situation modeled by the DBN. Some nodes will appear only once in the expanded net (corresponding to variables whose values don't change over time), some nodes will appear a few times (for slowly changing variables), and others will appear many times (for quickly changing variables).

When entering the delay amount for a link, you can enter a number (most common is simply 1), or the value of a [constant node](#), or an [equation](#) based on the values of one or more constant nodes. Links with time delays are displayed in a reddish-brown color.

To set the time delay of several links to the same value, select the links, and choose **Modify** → **Delay Links**, or = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-click the selection and choose Delay. You will then have the chance to enter the delay you would like them all to have. If you make the delay 0, they will become regular links.
```

Another way to set link delays is from a node properties dialog. Choose **Delay** from the [multipurpose selector](#) at the bottom, and you can enter a delay amount for each link entering the node.

>> [Next step](#)

## Using Dynamic Bayes Nets

[1.](#) Create DBN

**[2.](#) Generate time expansion**

[3.](#) Compile and use

Generate a time expanded version of the net by choosing **Network** → **Expand Time**.

This will make a new window with a new net in it. You will probably want to resize this window to make it wider.

The new net is a regular Bayes net with each Position and Velocity node representing the position and velocity at a new point in time, with nodes to the right corresponding to later times.

[Prev step](#) << >> [Next step](#)

## Using Dynamic Bayes Nets

1. Create DBN
2. Generate time expansion

### 3. Compile and use

Compile the Bayes net for probabilistic inference with **Network** → **Compile**.

Turn on automatic updating, if it isn't already, by toggling **Network** →

**Automatic Updating**, so the menu item is check-marked. Experiment with setting the position or velocity (by clicking on the desired interval) to indicate observations at certain times, and see how the beliefs for position and velocity at all other times change.

You can also try `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_enter_finding.htm');return false;">entering` some negative findings by holding down the `SHIFT` key when you click on the interval.

Remember that the numerical results will not be exact, due to the discretization and sampling error in converting the equations to probability tables. You may also want to try a finer discretization by changing the [Discretization](#) of the original unexpanded net.

[Prev step](#) <<

## Testing a Net Using Cases

This section documents the menu choice **Cases** → **Test Net Using Cases** (or "**Network** → **Test Using Cases**" on older versions of Netica). The purpose of this feature is to grade a Bayes net using a set of real cases to see how well the predictions or diagnosis of the net match the actual cases. It is not for

```
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_decision_nets.htm');return false;">decision  
networks.
```

First, = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">select the  
nodes you do not wish the network to know the value of during its inference.

For example, if the network is for medical diagnosis, you might select the disease nodes and nodes representing other unobservable internal states. We will call these nodes the "unobserved" nodes. Then choose **Network** → **Test With Cases**. You will be asked which file of cases to use, and after you choose one, Netica will start processing.

```
The = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_Messages_window.htm');return  
false;">Messages window will come to the front and display the percentage of  
cases processed so far. Hold down CTRL + ALT + LEFT BUTTON at the same time if  
you want to stop processing cases (the results for the cases already processed  
will then be printed). Netica will pass through the case file, processing the  
cases one-by-one. Netica first reads in the case, except for any findings for the  
unobserved nodes. It then does = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_belief_updating.htm');return false;">belief  
updating to generate beliefs for each of the unobserved nodes. It goes back  
and checks the true value for those nodes as supplied by the case file (if they  
are supplied for that case), and compares them with the beliefs it generated. It  
accumulates all the comparisons into summary statistics.
```

When Netica is done, it will print a report for each of the unobserved nodes

(except constant nodes). Typically you are only interested in some of them, so you can ignore the rest. The report for a node named "SpkQual" (with node title "Spark quality" might look something like this:

For SpkQual: Spark quality

**Confusion:**

```

.....Predicted.....
  good      bad  very_b  Actual
-----
    253      0      0    good
    22     176      4    bad
    13      19     430  very_bad
  
```

Error rate = 6.325%

**Scoring Rule Results:**

```

Logarithmic loss = 0.2144
Quadratic loss   = 0.1099
Spherical payoff = 0.9409
  
```

**Calibration:**

good	0-0.5:	0		0.5-1:	0		1-2:	0		2-5:
0										
	5-80:	49		80-95:	87.5		95-98:	95.7		
bad	0-1:	0		1-2:	1.52		2-5:	2.4		5-10:
5.17										
	10-50:	20		50-85:	82.6		85-95:	90		95-100:
100										
very_bad	0-0.1:	0		0.1-0.5:	0		0.5-5:	6.94		5-10:
9.33										
	10-20:	16.2		20-95:	83.3		95-98:	98.9		98-99:
100										
	99-100:	100								
Total	0-0.1:	0		0.1-0.5:	0		0.5-1:	0		1-2:
0.431										
	2-5:	2.5		5-10:	6.28		10-15:	10.9		15-20:
13.3										
	20-50:	30.1		50-80:	81.5		80-90:	86		90-95:
93.7										
	95-98:	97.6		98-99:	100		99-100:	100		

**Times Surprised (percentage):**

```

.....Predicted
Probability.....
  State      < 1%      < 10%      > 90%      >
99%
-----
  
```

```
--
good          0.00 (0/312)    0.00 (0/614)    6.86 (14/204)
  0.00 (0/0)
bad           0.00 (0/225)    1.98 (13/657)    0.00 (0/69)
  0.00 (0/0)
very_bad     0.00 (0/216)    3.32 (12/361)    0.25 (1/399)
  0.00 (0/31)
Total        0.00 (0/753)    1.53 (25/1632)   2.23 (15/672)
  0.00 (0/31)
```

## Sections of the Report

[Confusion Matrix & Errors](#)

[Scoring Rule Results](#)

[Calibration & Times Surprised Table](#)

[Quality of Test](#)

### NOTES:

If you have any findings entered before choosing **Network** → **Test With Cases** they will be taken into account during all belief updating (unless the case file has a column for that node). Netica will warn you in this event, so that you don't obtain wrong results by inadvertently leaving some findings in the network. A situation in which you would want to leave a finding in the network is if the network is designed for a broader class of cases than the case file. For example, if you have a network designed to handle people of both genders (and it has a 'gender' node), but the case file contains females only, you should enter a finding of 'female' for the 'gender' node before grading the network.

If the findings for the non-unobserved nodes of a case in the case file are impossible according to the network, then an inconsistent-findings error message will be displayed, that case will be ignored, and processing will continue. If the network makes predictions for the unobserved nodes that are inconsistent with the case file, then of course no error messages will be generated, the network will simply be graded more poorly (and have a logarithmic loss of INFINITY). Depending on your application, any of the measures calculated could be the most valuable to you. However, if you want a single number to grade a network, and aren't sure which one to pick, we

suggest the logarithmic loss. This function will properly support a '[NumCases](#)' column in the case file, if one is present.

As well as grading a network, this feature can also be used to determine the usefulness of particular tests or findings in a real world environment. Often groups of findings or tests can have quite a different usefulness when considered together, than when considered one-by-one, and this feature also allows you to investigate such groups. By selecting extra nodes in the first step, you can make some possible findings from the case file unavailable to the network. Then you can see how much the results of the network are degraded by not having access to those findings. In the medical example mentioned earlier, you might additionally select the nodes 'Blood Test' and 'Smear Test', and then compare the new [confusion matrix](#) generated with the old one, to find if the number of false negatives and false positives of serious diseases changed significantly.

This feature is also available to programmers using = 4 && [typeof\(BSPSPopupOnMouseOver\) == 'function'\)](#)  
[BSPSPopupOnMouseOver\(event\);" class="BSSCPopup"](#)  
[onclick="BSSCPopup\('X\\_PU\\_Netica\\_API.htm'\);return false;">Netica API; =](#)  
[4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\)](#)  
[BSPSPopupOnMouseOver\(event\);" class="BSSCPopup"](#)  
[onclick="BSSCPopup\('X\\_PU\\_email.htm'\);return false;">contact Norsys for](#)  
more information.

## The Confusion Matrix

**Confusion Matrix:** The first part of the [Test Net with Cases](#) report is a confusion matrix titled "Confusion:" The possible states of Spark Quality are good, bad and very\_bad. For each case processed, Netica generated beliefs for each of these states. The most likely state (i.e. the one with the highest belief) was chosen as its prediction for the value of Spark Quality.

This was then compared with the true value of Spark Quality for that case, providing the case file could supply it. The confusion matrix supplies the total number of cases in each of the 9 situations: (Predicted=good, Actual=good), (Predicted=bad, Actual=good), etc. If the network is performing well then the entries along the main diagonal will be large compared to those off of it.

**Error Rate:** The next part of the report is "Error rate = 6.325%". This means that in 6.325% of the cases for which the case file supplied a Spark Quality value, the network predicted the wrong value, where the prediction was taken as the state with highest belief (same as for the confusion matrix).

### Other sections of the Test Net with Cases Report:

[Scoring Rule Results](#)

[Calibration & Times Surprised Table](#)

[Quality of Test](#)

## Scoring Rule Results & Logarithmic Loss Values

**Scoring Rule Results:** The third section of the [Test Net with Cases](#) report is titled "Scoring Rule Results:" This doesn't just take the most likely state as a prediction, but rather considers the actual belief levels of the states in determining how well they agree with the value in the case file. These results are calculated in the standard way for scoring rules.

For more information see any reference on scoring rules, such as:

Morgan, M. Granger and Max Henrion (1990) *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*, Cambridge Univ. Press, New York.

Pearl, Judea (1978) "An economic basis for certain methods of evaluating probabilistic forecasts" in *International J. of Man-Machine Studies*, 10, 175-183.

**Logarithmic loss values** were calculated using the natural log, and are between 0 and infinity inclusive, with zero indicating the best performance.

Quadratic loss (also known as the Brier score) is between 0 and 2, with 0 being best, and spherical payoff is between 0 and 1, with 1 being best.

The equations are:

$$\begin{aligned}\text{Logarithmic loss} &= \text{MOAC} [-\log(P_c)] \\ \text{Quadratic loss} &= \text{MOAC} [1 - 2 * P_c + \sum_{j=1}^n (P_j^2)] \\ \text{Spherical payoff} &= \text{MOAC} [P_c / \sqrt{\sum_{j=1}^n (P_j^2)}]\end{aligned}$$

where  $P_c$  is the probability predicted for the correct state,  $P_j$  is the probability predicted for state  $j$ ,  $n$  is the number of states, and MOAC stands for the mean (average) over all cases (i.e. all cases for which the case file provides a value for the node in question).

### Other sections of the Test Net with Cases Report:

[Confusion Matrix & Errors](#)

[Calibration & Times Surprised Table](#)

## Quality of Test

## Calibration & Times Surprised Tables

**Calibration Table:** The next part of the [Test Net with Cases](#) report is a table titled "Calibration:". It indicates whether the confidence expressed by the network is appropriate (i.e. "well calibrated"). For instance, if the network were forecasting the weather, you might want to know: Of all the times it said 30% chance of rain, what percentage of times did it rain? If there were lots of cases, the answer should be close to 30%.

For each state of the node there are a number of items separated by vertical bars (|). Each item consists of a probability percentage range R, followed by a colon (:) and then a single percentage X. It means that of all the times the belief for that state was within the range R, X percent of them the true value was that state.

**For instance:**

rain 0-10: 8.5 |

means that of all the times the belief for rain was between 0 and 10%, 8.5% of those times it rained. The reason that the probability ranges are uneven, and different from state to state, and run to run, is that they are chosen so that the X percentages are reasonably accurate. The bin sizes have to adapt, or there might not be enough cases falling in that bin. The more cases you process, the more fine will be the probability ranges.

Calibration results are often drawn as a graph (known as a "calibration curve") where ideal calibration is a straight diagonal line. For more information, see a text which discusses probability "calibration" for example, [Morgan&Henrion90,p.110](#).

**Times Surprised Table:** Following the calibration table of the report is the "Times Surprised" table. It is used to determine how often the network was quite confident in its beliefs, but was wrong. There are columns for being 90% confident and 99% confident (i.e. beliefs are greater than 90% or 99% respectively), and also for being 90% and 99% confident that the value of the node will not be a certain state (i.e. beliefs are less than 10% or 1% respectively).

The ratios indicate the number of times it was wrong out of the number of times it made such a confident prediction, and a percentage is also printed. If

the network is performing well these percentages will be low, but keep in mind that it is very reasonable to be wrong with a particular 10% or 90% prediction 10% of the time, and to be wrong with a particular 1% or 99% prediction 1% of the time. If the network rarely makes strong predictions (i.e. beliefs are rarely close to 0 or 1), then these most of these ratios will be 0/0.

## **Other sections of the Test Net with Cases Report:**

[Confusion Matrix & Errors](#)

[Scoring Rule Results](#)

[Quality of Test](#)

## Quality of A Test

**Quality of Test:** The final section of the [Test Net with Cases](#) report is the "Quality of Test" table for binary nodes, or "Test Sensitivity" table for nodes with more than 2 states. These are useful when the output of the network is going to be used to decide an action, with one action corresponding to each state of the node.

As a medical example, the node may be "Disease-A" and have the two states "Present" and "Absent". If, after updating for a case, the network reports "Present", then a particular treatment will be started, but if it reports "Absent" then the treatment won't be started. The question is, at what probability for "Present" should we say that the network is reporting Present? The confusion matrix and error rate discussed above were determined using the maximum likelihood state (i.e. the one with highest belief after updating).

For a binary variable, this means choosing the first state only if its belief is higher than 50%. But if each state has a different cost of misclassification, you may not want the cutoff probability to be 50%. In the medical example, it may be disastrous to not treat a patient who has the disease, but not that serious if he is treated unnecessarily. So you would like the network to report "Present" if the probability of the disease is above some small number, like 2%. It is a matter of trading off the rate of false positives against the rate of false negatives. Ideally you would just convert the network to a decision network, by adding a decision node for the action to be taken and a utility node for the cost of misclassification. However, at the time the network is constructed and being graded as to its usefulness, the utilities may not be known.

The Quality of Test section has performance results for a series of cutoff threshold probabilities (which run vertically in the first column). For each case, the beliefs given by the network are converted to a "prediction". The prediction is "first state" if the belief for the first state is higher than the cutoff probability, and "second state" if it's lower. You may want to change the order of the states, so that the first state is the "positive" one, to better match conventional meanings.

The meanings of the columns are:

Sensitivity = Of the cases whose actual value was the first state,

the fraction predicted correctly.  
Specificity = Of the cases whose actual value was the second state,  
the fraction predicted correctly.  
Predictive Value = Of the cases the network predicted as first state,  
the fraction predicted correctly.  
Predictive Value Negative = Of the cases the network predicted as  
second state, the fraction predicted correctly.

Often this data is summarized with a graph called the ROC (receiver operating characteristic) curve. To use Excel (available from Microsoft) to create the ROC curve from this data, select the whole table (except headings) and while holding down the <CTRL> key, type tcz. Then open the Excel file called "Graph\_ROC.xls" (available from the Norsys ftp site), paste into the indicated cell, and the graph will be drawn. If the node has more than 2 states, instead you will get a Test Sensitivity section. The first number of each "column" is the cutoff threshold probability. The second number of each column is the number of cases whose actual value was the state given at the left hand side of the row, and which the network correctly predicted to be in that state (i.e. its belief was greater than cutoff probability), divided by the total number of cases whose actual value was that state.

It may seem awkward that the cutoff probability changes in strange sized jumps. The reason is that Netica only reports on values for which it was able to gather enough data. So running the test using a greater number of cases generally results in finer divisions of the cutoff column.

### **Other sections of the Test Net with Cases Report:**

[Confusion Matrix & Errors](#)

[Scoring Rule Results](#)

[Calibration & Times Surprised Table](#)

## Sensitivity Analysis

Netica can do extensive utility-free single-finding sensitivity analysis. Select a node (called the "target node") and choose **Network** → **Sensitivity to Findings** from the menu. A [report](#) will be displayed in the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Messages_window.htm');return`  
`false;">Messages` window displaying how much the beliefs, mean value, etc. of the target node could be influenced by a single finding at each of the other nodes in the net (each is called a "findings nodes").

The first part of the report has a section for each findings node, showing how much it can effect the target node using several different sensitivity measures. The second part is a summary table which compares the [sensitivities](#) for each of the findings nodes.

If you want to limit the report to a few findings nodes, first select the target node, and then use `CTRL-select` to add the desired findings nodes to the selection. Then choose **Network** → **Sensitivity to Findings**.

Currently this sensitivity analysis will only work for Bayes nets and not = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_decision_nets.htm');return false;">decision nets.`

**Example:** Suppose you are using the net for diagnosis, and you want to determine which test is going to provide the best information about the presence of a fault or disease. Select the node for the fault or disease and choose **Sensitivity to Findings**. Use the summary list of sensitivities at the end of the report generated to identify possible findings nodes which will provide the most information about the fault/disease node. If you want more detailed information of how these findings nodes can effect the fault/disease node, look up each of them in the first part of the report.

**Single Number:** If you want a single number which best describes the degree of sensitivity of one node to another, it is recommended that you use the first column of the summary report at the end. For = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`

BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous  
nodes or nodes with = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_state\_value.htm');return false;">state values  
defined, this will be the [variance reduction](#), otherwise it will be the mutual  
information (i.e. [entropy reduction](#)).

**Findings:** When the sensitivities are calculated, the findings currently = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered into  
the net will be taken into account, which can effect the sensitivities  
significantly.

For full documentation on this function, and each of the [sensitivity measures](#)  
calculated, see [Sensitivity to Findings](#).

## Sensitivity To Findings

Of significant importance in Bayes net work is a measure of the independence between various nodes of the net. Using just the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link\_structure.htm');return false;">link structure and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_d\_separation.htm');return false;">d-separation rules, you can determine which nodes are completely independent of which other ones (see **Edit** → **Select Nodes** → **Info (D-) Connected**), and how that changes as = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings arrive. However, dependence is a matter of degree, and using Netica's [sensitivity functions](#) you can efficiently determine how much a finding at one node will likely change the beliefs at another.

During diagnosis, you may wish to know which nodes will be the most informative in crystallizing the beliefs of the most probable fault nodes. Obviously, that will change as findings arrive, so it may need to be recomputed at each stage. In a net built for classification, you can determine which features are the most valuable for performing the classification (i.e. "feature selection"). In an information gathering environment, you can identify which are the most important questions to ask at each point (to provide information on the variables of interest), based on the answers to questions already received, so as to avoid asking unnecessary or irrelevant questions.

In real-world modeling, such as environmental modeling, you can determine which parts of the model most affect the variables of interest; thereby identifying which parts should be made the most carefully and accurately.

Select a node (called the "query node") and choose **Network** → **Sensitivity to Findings** from the menu. A [report](#) will be displayed in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

`onclick="BSSCPopup('X_PU_Messages_window.htm');return false;">Messages Window` displaying how much the beliefs, expected value, etc. of the query node would be influenced by a single finding at each of the other nodes (each is called a "varying node").

The first part of the report has a section for each varying node, showing how much it can effect the query node using several different [sensitivity measures](#). The second part is a summary table which compares the sensitivities for each of the varying nodes.

If you want to limit the report to a few varying nodes, first select the query node, and then use `CTRL-SELECT` to add the desired varying nodes to the selection.

Then choose **Network** → **Sensitivity to Findings**. Currently this sensitivity analysis will only work for Bayes nets and not = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_decision_nets.htm');return false;">decision networks` (i.e. networks with decision nodes).

Here is an [example use](#) during diagnosis. Here is a description of the [measures](#) that Netica calculates.

## Sensitivity Equations

Below are descriptions of each of the utility-free [sensitivity](#) measures that Netica calculates. First are some notes for interpreting the descriptions.

**Definition:** In the definitions, "belief" means posterior probability (i.e. conditioned on all findings currently entered). In the names of the various measures "real" refers to the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_expected.htm');return false;">expected value of continuous nodes, or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete.htm');return false;">discrete nodes which have a real numeric value associated with each state "expected value" means to take the expectation over a quantity.

**Range:** The minimum and maximum values that this measure can take on.

**Compare:** A quantity which is useful to compare the value of this measure against (perhaps to express this measure as a percentage).

**Equation:** Note that all the conditionals should include all findings already entered into the network, so  $P(q)$  is really  $P(q|E)$ ,  $P(q|f)$  is really  $P(q|f,E)$ , etc.

### Notation:

Q is the query variable

F is the varying variable

q is a state of the query variable

f is a state of the varying variable

$X_q$  is the numeric real value corresponding to state q

$SUM_{\sim q}$  means the sum over all states q of Q. It applies to the whole expression following.

$MIN_{\sim q}$ ,  $MAX_{\sim q}$  are similar to  $SUM_{\sim q}$

$E(Q)$  is the expected real value of Q before any new findings

$E(Q|f)$  is the expected real value of Q after new finding f for node F

$V(Q)$  is the variance of the real value of  $Q$  before any new findings

$H(Q)$  is the entropy of  $Q$  before any new findings

RMS is "root mean square", which is the square root of the average of the values squared.

## Minimum Belief

**Definition:** Minimum belief that each state  $q$  of  $Q$  can take due to a finding at  $F$ . This provides a value for each state.

**Range:**  $[0, P(q)]$   $P(q)$  if  $Q$  is independent of  $F$

**Compare:**  $P(q)$

**Equation:**  $P_{\min}(q) = \text{MIN}_{\sim f} P(q|f)$

## Maximum Belief

**Definition:** Maximum belief that each state  $q$  of  $Q$  can take due to a finding at  $F$ . This provides a value for each state.

**Range:**  $[P(q), 1]$   $P(q)$  if  $Q$  is independent of  $F$

**Compare:**  $P(q)$

**Equation:**  $P_{\max}(q) = \text{MAX}_{\sim f} P(q|f)$

## RMS Change of Belief

**Definition:** The square root of the expected change squared of the belief of state  $q$  of  $Q$ , due to a finding at  $F$ . This provides a value for each state.

This is the standard deviation of  $P(q|f)$  about  $P(q)$  due to a finding at  $F$ , with the finding at  $F$  distributed by  $P(f)$ .

**Reference:** Spiegelhalter89 & [Neapolitan90](#),p394. They call the square of this quantity simply "variance".

**Range:**  $[0, 1]$   $0$  if  $Q$  is independent of  $F$

**Compare:**  $P(q)$

**Equation:**  $sp(q) = \text{sqrt}(Vp(q))$

$$Vp(q) = \text{SUM}_{\sim f} P(f) [P(q|f) - P(q)]^2$$

## "Variance" of Node Belief (named "Quadratic Score" in older versions of Netica)

**Definition:** The expected change squared of the beliefs of Q, taken over all of its states, due to a finding at F.

**Reference:** Spiegelhalter89 & [Neapolitan90](#),p394. They call this "variance" (for them it comes out the same as  $Vp(q)$  because they just use 2-state nodes).

**Range:**  $[0, 1]$       0 if Q is independent of F

**Equation:**  $s^2 = \sum_{\sim f} \sum_{\sim q} P(q,f) [P(q|f) - P(q)]^2$

## Minimum Real

**Definition:** The lowest that the expected real value of Q could go to, due to a finding at F.

**Requires:** Node Q is continuous, or has real number state values defined.

**Range:**  $(-\infty, E(Q)]$        $E(Q)$  if Q is independent of F

**Compare:**  $E(Q) = \sum_{\sim q} P(q) Xq$

**Equation:**  $mmin = \min_{\sim f} E(Q|f)$

## Maximum Real

**Definition:** The highest that the expected real value of Q could go to, due to a finding at F.

**Requires:** Node Q is continuous, or has real number state values defined.

**Range:**  $[E(Q), \infty)$        $E(Q)$  if Q is independent of F

**Compare:**  $E(Q)$

**Equation:**  $mmax = \max_{\sim f} E(Q|f)$

## RMS Change of Real

**Definition:** The square root of the expected change squared in the expected real value of Q, due to a finding at F. This turns out to be the same as the square root of the variance reduction of expected value.

**Requires:** Node Q is continuous, or has real number state values defined.

**Range:**  $[0, V(Q)]$  0 if Q is independent of F

**Compare:**  $E(Q)$  and maybe  $V(Q)$

**Equation:**  $sm = \sqrt{V_m}$

$$V_m = \sum_{f \sim F} P(f) [E(Q|f) - E(Q)]^2 = V_r$$

## Variance Reduction of Real

**Definition:** The expected reduction in variance of the expected real value of Q due to a finding at F. This turns out to be the square of RMS Change of Real.

**Requires:** Node Q is continuous, or has real number state values defined.

**Range:**  $[0, V(Q)]$  0 if Q is independent of F

**Reference:** Pearl88,p323. What he says is  $C(T|X)$  is actually  $C(T|X)-C(T)$ .

Var mapping:  $T \rightarrow Q, X \rightarrow F, C \rightarrow V, t \rightarrow q$  and  $X_q$

**Compare:**  $V(Q)$

**Equation:**  $V_r = V(Q) - V(Q|F) = V_m$

$$V(Q) = \sum_{q \sim Q} P(q) [X_q - E(Q)]^2$$

$$V(Q|f) = \sum_{q \sim Q} P(q|f) [X_q - E(Q|f)]^2$$

$$E(Q) = \sum_{q \sim Q} P(q) X_q$$

## Entropy Reduction (Mutual Information)

**Definition:** The mutual information between Q and F (measured in bits). The expected reduction in entropy of Q (measured in bits) due to a finding at F.

**Range:**  $[0, H(Q)]$  0 if Q is independent of F

**Reference:** [Pearl88,p321](#). He has sign of  $I(T,X)$  backwards.

Var mapping:  $T \rightarrow Q, X \rightarrow F, I(T,X) \rightarrow I$

**Compare:**  $H(Q)$

**Equation:**  $I = H(Q) - H(Q|F)$

$$= \sum_{\sim q} \sum_{\sim f} P(q,f) \log (P(q,f) / [P(q) P(f)])$$

Note that the log is base 2, which is traditional for entropy and mutual information, so that the units of the results will be "bits".

## Sensitivity - Diagnosis Example

If you want to determine which test is going to provide the best information about the presence of a fault or disease, select the node for the fault or disease and choose **Sensitivity to Findings**. Use the summary list of sensitivities at the end of the [report](#) generated to identify possible varying nodes for which a finding will provide the most information about the query node. If you want more detailed information of how these varying nodes can effect the query node, look up each of them in the first part of the report.

If you want a single number which best describes the degree of sensitivity of one node to another, it is recommended that you use the numbers provided in the first column of the summary report at the end. For continuous nodes, or nodes with real number state values defined, this will be the variance reduction, otherwise it will be the mutual information (i.e. entropy reduction).

When you do a complete sensitivity report (i.e. only the query node selected), then in the report Netica also shows the sensitivity of the query node to a finding at the query node itself. Of course, the minimum and maximum beliefs for each state will be 0 and 1 respectively, and the [maximum reductions](#) in variance and [entropy](#) will be 100%. This node is included in the report for completeness, and to quickly see what the maximum of each sensitivity value is (for example, what the full variance and entropy is).

When the sensitivities are [calculated](#), all findings currently entered into the network will be taken into account, which can effect the sensitivities significantly.

If you are trying to find the next best observation to make a diagnosis, you will probably want to combine the cost of each possible observation with its expected value as indicated by the sensitivity to that observation (finding).

If you want to consider pairs of observations, or multiple observations, the results can be quite different than if you consider observations one at a time.

To do a pair of observations, you must enter each possible finding of the first observation, and do a [sensitivity analysis](#) on the second observation, then average the results (weighted by the probability of the finding for the first observation) to find an expected value.

You will probably need [Netica API](#) to automate this. The sensitivity measures

available from Netica API, as of version 3.10, are mutual information (i.e. entropy reduction), and [RMS](#) change of real (i.e. variance reduction of real).

## Transforming a Net

There are certain ways that Netica can transform a net model which modifies its representation without modifying its meaning. Netica can remove nodes and reverse the direction of links in such a way that any inference done with the resulting net yields precisely the same results as the original net (except of course findings can't be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered for removed nodes, and their resulting beliefs are unavailable).

Some reasons to do these transforms are to: simplify the net, apply the net to more specific problems, gain understanding of the net or of the real world relations, or put the net in a form for easier probability or function = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probab\_assessment.htm');return false;">assessment.

## Link Reversals

Suppose P and C are = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete\_node.htm');return false;">discrete or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretized nodes within a net, and that P is a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parent of C.

Since there is a link going from P to C, the table for the two nodes is expressed as probabilities for the states of C, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditional\_probability.htm');return false;">conditioned on the states of P (or a function providing C's value in terms of P's value if the node is deterministic). But sometimes you might want to know what the probabilities for the states of P are, conditioned on the states of C.

You can achieve that by doing a [net transform](#) known as *link reversal*, which reverses the link from P to C. When that link is reversed, the other links and the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPTs of C and P are adjusted in such a way that any = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">probabilistic inference done after the reversal will yield exactly the same results as before the reversal. In other words, the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_joint\_distribution.htm');return false;">full joint probability distribution of the net does not change when a link is reversed.

The global relationship between the nodes remains the same; just the local expressions of it have been changed (as is the case with [node absorption](#)).

Link reversal is the probabilistic generalization of function inversion, and is a good example of *Bayes rule* in action.

**Adds Links:** During the reversal, Netica may add links to C from the parents of P, and/or add links to P from the parents of C, which will increase the complexity of the net (all links added will be confined to P and its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Markov\_boundary.htm');return false;">Markov boundary). When links are added, the size of CPTs may grow significantly, and sometimes enormously. The size of the tables is the product of the number of states of all the parents, so the size of each table can grow exponentially.

When link reversals result in a node having many parents, the operation may be slow, or Netica may report that there is not enough memory available.

**Removes Links:** Occasionally, during a reversal, Netica may remove links to C from the parents of P, and/or remove links to P from the parents of C, resulting in a simpler net. For example, if reversing a link added other links, then reversing it again will remove them (assuming all the nodes have nondegenerate CPTs, and that there are no major rounding inaccuracies).

**Simpler Net:** Usually when a net is simpler due to the direction of its links, it provides a better model of the world. For example: it usually represents true causality more accurately, it may be better at generalizing, and of course it allows for faster computation. Sometimes link reversals can be used to search for more simple nets, given a net that was originally learned from data. In that case Netica might not always automatically remove links that should be removed, because although these links will be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_weak\_link.htm');return false;">weak, they will not be completely ineffectual (perhaps because the probability tables learned are not “exact”). You would have to remove these weak links by hand.

**How To:** To reverse a link you = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_link.htm');return false;">select it and choose **Modify** → **Reverse Links**, or click the  toolbar button. If a node is = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_select_node.htm');return false;">selected when
you click the button, Netica will do all the link reversals necessary to make all
links involving the node point to it. Alternately, you can = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_right_click.htm');return false;">right-click on a
link, and choose Reverse from the menu. Right-clicking on an
__unselected__ node gives the options Links → Reverse So All Incoming
and Links → Reverse So All Outgoing, which will do all the reversals
necessary to achieve the stated effect. If all links are made to point away from
the node, then many extra links may have to be added between the ancestors of
the node (not just its Markov boundary), and Netica may report that there is
not enough memory available.
```

**Several Links:** If you have several links to reverse, you can select them all (e.g. by **CTRL**-clicking on them), and then click the  button. The amount of time and memory required to reverse a set of links depends greatly on the order in which they are reversed. Unless you know a good order to do the reversals, you should reverse them all at once rather than one-by-one so that Netica can choose a good order to do them.

See also [Disconnecting and Reconnecting Links](#)

## Node Absorption

Node *absorption* is a [net transform](#) which removes nodes from a Bayes net or decision net, and makes any necessary adjustments to the resulting net, so that any inference done with it yields the same results as before the nodes were removed (except of course you can't interact with the removed nodes). The local representation is changed, but the global relationships are not changed (as is the case with [link reversal](#)). In probability theory this is sometimes loosely called "summing out a variable". It leaves the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_joint_distribution.htm');return false;">full joint probability distribution of the remaining nodes unchanged.`

As an example, suppose you have a large net that has been constructed over time by a combination of expert assistance and probability learning. It shows the relationships between hundreds of variables, and contains much valuable information that could be used in a number of different applications.

Now you want to use it in an application where only 10 of the variables will be of interest. In every query of the new application, a particular 4 of these 10 will always have the same findings. For example, one of the nodes in the original net might be Gender, and in the restricted application the net will only be used for females, so you would like to = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_enter_finding.htm');return false;">enter a permanent finding of 'female' for the Gender node. These nodes are called context nodes. In each of the queries, you will be receiving new findings for 4 other nodes, and then you want the resulting beliefs of the remaining 2 out of 10. The nodes that will always have new findings are called findings nodes, and those whose beliefs you may want are called query nodes. The hundreds of other nodes in the net might be involved in intermediate calculations, but you don't care about their values explicitly.`

You can simplify the large net down to one with just 6 nodes using node absorption. First [enter](#) the permanent findings for the context nodes. Then select all the nodes to be absorbed (i.e. all the nodes except the findings and

query nodes), and choose **Modify** → **Absorb Nodes** or click the  toolbar button. The selected nodes will be removed, and some links may be added and/or reversed.

**Order:** If you want, you can absorb the nodes a few at a time, by selecting each group and clicking the  toolbar button. The final result of absorbing a set of nodes is not dependent on the order in which they were absorbed, but the time and memory required may be greatly affected. If you have a set of nodes to absorb and you don't know a good order to use, then it is best to absorb them all at once, so that Netica can pick a good order.

Returning to the example, the resulting 6 node net will give the same inference results as the original large one, for the restricted queries you will be making.

If you are guaranteed that there will always be findings for every findings node, then you can further simplify things by removing any links that go from findings node P to findings node C, providing C does not have a query node as a parent. This means that if you can [reverse links](#) to make all the evidence nodes ancestors of all the query nodes, then you can remove all the links between the evidence nodes. Any findings node that is left completely disconnected by this operation is irrelevant to the query, and can be deleted.

And now you can examine the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPTs` of the query nodes to see directly how they depend on the findings. You may just be able to look up the desired probabilities without doing `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_belief Updating.htm');return false;">belief updating` at all!

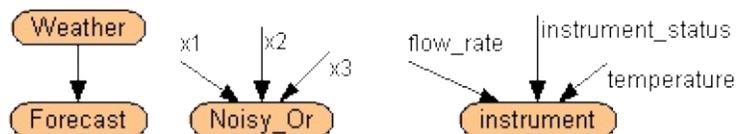
**Complexity Danger:** Even though a reduced net has fewer nodes than the original, internally it may actually be more complex, sometimes much more complex, if many links were added during node absorbing or link reversing (remember that the size of a node's CPT can be exponential in its number of parents). Generally speaking, absorbing out context nodes (i.e. nodes with findings entered) which have many ancestor nodes results in the worst increase in complexity. The next worst is absorbing out non-context nodes (i.e. nodes with no findings) which have many descendant nodes. Absorbing out context nodes with no ancestors, or non-context nodes with no descendants, will not

add any links. Of course, if the number of query and findings nodes is very small and they have few states, the resulting net must be very simple, although the transformations to generate it might temporarily require a lot of memory.

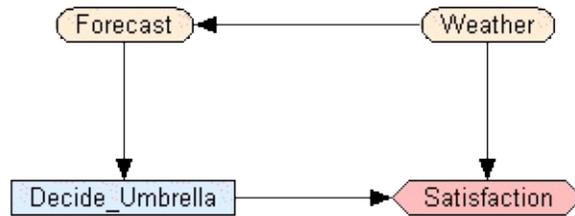
## Net Fragment Libraries

Often the probabilistic = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_relation.htm');return false;">relation between a node and its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents represents a small piece of local knowledge which may be applicable in a number of different nets to be used in different situations. That relation may have been learned from data, or entered by an expert. Each new net that it is placed in captures the global relations between such local pieces of knowledge, and belief updating combines the local and global knowledge with the details of some particular case.

You can keep pieces of local knowledge as net fragments in a net library, and later paste them into nets you design to solve a particular problem. Here is an example of a few net fragments in a library:

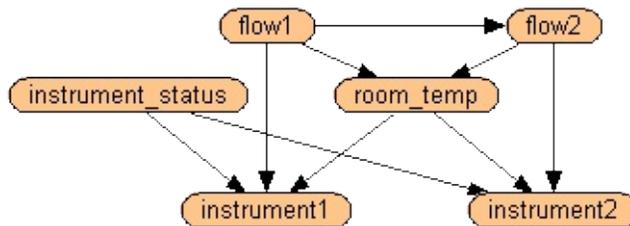


**Example:** For example, suppose that you made a simple net consisting of a node called Forecast connected to a node called Weather (what the weather turned out to be after the forecast). You could put the link between them either way, since in this situation you can't really capture causation (they are both caused by other variables, such as the weather at earlier times), but say you put the link from Weather to Forecast because often its better to put links from more immutable to less immutable variables. You could [learn](#) its probabilities from a set of cases consisting of the forecast and what the weather turned out to be. Then you could put it in a library where it might look like the leftmost fragment in the diagram above. Later, you graft it into nets for inference involving the weather and its forecast, such as the [decision problem example](#):



**Example:** As another example, suppose you have a device for measuring the flow rate in a pipe. It produces biased readings depending on the ambient temperature, and it can malfunction in a few different ways, each of them producing wrong or inaccurate readings. You can model the device with a 4 node net, consisting of one node for the reading on the device, and 3 parent nodes corresponding to: actual flow rate, ambient temperature, and device status (okay, broken 1, broken 2, etc.). You enter the probability table, and then you **disconnect** the node from its parents and place it in a library. The rightmost node in the first diagram on this page shows how it will appear.

Later, if you have a net to model a situation in which you use the instrument to make two measurements of the flows in two connected pipes located in the same room, you just duplicate the device characteristics node from the library twice into the new net, and graft it to the appropriate nodes in that net, as shown in the diagram below. Note that if the ambient temperature could be different between the two measurements, then the room\_temp node would appear as two connected nodes, similar to the flow nodes, and the same goes for the instrument\_status node if the device may have broken between measurements.



## Disconnecting and Reconnecting Links

**Purpose:** A link may be disconnected from its parent node, without the link actually being removed. That means that the child node can maintain the information on the conditional probability table it had with its parent, without actually being connected to the parent. The intention is that later it will be reconnected to the parent, or more likely to some other node, before it is used for inference. It may be reconnected soon afterwards, or the node(s) may be placed in a library of [net fragments](#) and only reconnected when the library is used to build a new Bayes net.

**How To:** To disconnect some links, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_link.htm');return false;">select them and then choose **Modify** → **Disconnect Links** or click the  toolbar button. Each link will turn into a short stub connected to its child node, and labeled with the name of the link. If you have not previously [named](#) the links, then Netica will name them with the names of the parent nodes. If you want to disconnect all the links entering some node(s), select the node(s) and choose **Modify** → **Disconnect Links**. You can also hilite individual links, right-click and choose **Disconnect**.

**By Dragging:** If you want to disconnect a link while mostly maintaining its shape, first click on the link to select it. Then click down on the hilited box at the end of the link (the end without the arrow), and drag it away from the parent node. When you release the mouse button you will know that you have dragged the box far enough if the name of the link appears beside it. All of the bends of the link will be maintained.

**Child End:** A link cannot be disconnected from its child node. That would not make sense, since a disconnected link is really just a “parent place-holder” to maintain the node’s relation (e.g. CPT) until it is connected up to a new parent. To achieve the affect of disconnecting a link at the child end, it sometimes makes sense to do a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link\_reversal.htm');return false;">link reversal, and then disconnect the link at the parent end.

**Reconnection:** To reconnect a disconnected link to a new node, first click on the link to select it. It will be outlined with the hilite color, and there will be a box of hilite color at the disconnected end of the link. Choose **Modify** → **Reconnect Links**.

You may disconnect a link from one parent and connect it to a new parent in one motion if you wish.

See also [Link Reversals](#)

See also [Deleting Nodes and Links](#)

## Creating and Using Net Libraries

**Creating:** Netica makes it easy to maintain libraries of disconnected nodes and subnets. To make a new library, just choose **File** → **New** → **Network** or click the  toolbar button. Libraries are handled internally in the same way as regular nets. Nodes and subnets can be copied to it by = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_select_node.htm');return false;">selecting nodes (there is no need to select links), and using Edit → Copy and Edit → Paste, which can transfer material (nodes, links and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPTs) from one net to another. When a node is being duplicated, but one of its parents isn't, then the duplicated node will have a disconnected link where that parent was.`

**Using:** To use nodes in the library, you use **Edit** → **Copy** and **Edit** → **Paste** again, this time to duplicate from the library into the new net. Then you connect up any disconnected links, before compiling the net or using it for inference.

**Example:** For example, to create a library with just the “instrument” node of figure 6.1, first you would make a net with the 4 nodes: flow\_rate, temperature, instrument\_status and instrument. Put links from nodes flow, temperature and instrument\_status to instrument. Enter a probability table for the node “instrument”. Now make a new net with **File** → **New** → **Network**, select the “instrument” node in the original net, do an **Edit** → **Copy**, click in the new net, do an **Edit** → **Paste**, then a **File** → **Save**. You now have file which is a library with a single node in it.

At a later session, you can use the library to construct a net in which the instrument is used to measure the flows as described at the beginning of this chapter. Make a new net with **File** → **New** → **Network**, add the nodes flow1, flow2, instrument\_status and room\_temp, link them together as shown in figure 6.2, and enter the probability table between them. Then use **File** → **Open** to open the library file with the instrument node. Select that node, do an

**Edit** → **Copy**, then click in the application net where you want the instrument node to appear. Do an **Edit** → **Paste**, click again where the other copy of it should be and do another **Edit** → **Paste**. Finally, hook up each of instrument's disconnected links to their appropriate nodes using the method described in the previous section.

Now the application net is ready for probabilistic inference (you can do a **Network** → **Compile** menu command). Perhaps you have positive findings for the “instrument” node (i.e. what you read from its dial), and you use them to determine flows and their uncertainties in a way that properly accounts for random (uncorrelated) and systematic (correlated) errors, as well as all the background knowledge about the situation.

**Netica API:** As a sister product, = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Netica\_API.htm');return false;">Netica Programmer's Library (API) can be useful for automating the process of constructing nets that are composed of nodes or subnets from a library, perhaps by using templates or rules.

## COM Interface – C# and Visual Basic Programming

Netica has APIs for several different computer languages (such as Java, C/C++, etc.), which allow it to be used as part of a computer program in one of those languages. For more information, see [Netica API](#).

One such Netica API is called Netica-COM. You can program Netica through its COM interface (also known as ActiveX or Automation) using any computer language that can access a COM interface. For example C#, Visual Basic (VB), COM enabled Java, or C++. The below is written as though VB is being used, but it pertains to programming in the other languages as well.

A great benefit of programming Netica through its COM interface is that people can interact with the GUI at the same time as your VB program is running. That can be useful for debugging, demos, and distributing products that take advantage of the Netica user interface by allowing the end-user to interact directly with the Bayes net.

If you have programmed an MS Office product using Visual Basic, you will find some similarities in style with the Netica COM interface.

To use Netica COM in its fully enabled form, you must have a Netica license [password](#) that is for both Netica Application and Netica API. First, run the [latest version](#) (must be version 3.10 or greater) of Netica Application, so it can register itself in the registry. If you have several versions of Netica on your computer, the version of Netica that VB will use will always be the one that was last run (for example, by double-clicking it). You can leave it running or exit it. Then to get access to it from Visual Studio, choose **Project** → **Add Reference from Visual Studio**, click the **COM** tab if necessary, and check-mark the entry called "Netica 1.0 Object Library" (with perhaps some other version number).

You can then use the object browser to see the Netica objects and functions available. Choose **View** → **Object Browser** or **View** → **Other Windows** → **Object Browser**, and then:

- In Visual Studio 6, set the library to Netica in the upper left choice box.
- In Visual Studio .NET, one of the top level entries in the object browser will be "Interop.Netica". You can browse that, but it won't be as good as browsing

the Netica library directly, because you won't have a description for each function. If there is no entry at the top level for the library named simply "Netica" (with the books icon), click on the **Customize** button at the top of the window, and then the **Add** button of the dialog box that appears. Choose the **COM** tab, select the Netica library from the list, click **Select** and then **Okay**. Now the Netica library should appear, and you can browse it.

In the object browser, when you click on any function or enum, then a short description will appear, which is good enough for most programming work. If you need more detail, then you will notice that within the description is the name of the equivalent function in the Netica-C API. You can look up that function in the Netica-C documentation.

- To view that documentation, see:  
<http://www.norsys.com/onLineAPIManual/index.html>
- To download it, go to: [http://www.norsys.com/download\\_api.html](http://www.norsys.com/download_api.html)

**Example Code:** Here is example code to program Netica in **C#**, **Visual Basic** and **Managed C++** (C++/CLI) using the COM interface. There is also an example Visual Studio project for each, called "Netica Demo for xx" within the "Netica\Netica xxx\Programming Examples" folder of the Netica download package.

**Password/Distribution:** If your license password enables only Netica API, then the GUI will operate in a limited mode, and if your license password enables only Netica Application, then your VB calls to Netica will operate in a limited mode. If you want to distribute your VB program that uses Netica, you need to only send Netica.exe (and perhaps Netica.hlp), but not Netica.dll. Make sure you have the required license from [= 4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\) BSPSPopupOnMouseOver\(event\);" class="BSSCPopup" onclick="BSSCPopup\('X\\_PU\\_Norsys.htm'\);return false;">Norsys](#) before you do so.

To start up Netica so that it operates with a password that is not put in the Registry, start it from your VB program with the **command line** argument - password followed by the password you want Netica to use. That password will not be entered into the registry, and its security part will not be visible to the user.

**GUI Control:** You can use the Visible property of the Netica.Application object to determine whether or not the graphical user interface (GUI) is present or not (by setting it to true or false, see the example below). If it is not visible, then it will be a minimized window appearing only as an icon on the task bar. If you wish to prevent the user from un-minimizing it, thereby preventing them from having access to the Netica GUI, set the UserAllowed property of Netica.Application to false.

**Example:** To do the above, your program might have a part looking something like this:

```
Shell ("C:\\Netica\\Netica312\\Netica.exe -password +Me/MyOrg...")
```

```
Dim app As New Netica.Application
```

```
app.Visible = False
```

```
app.UserAllowed = False
```

```
...
```

```
app.Quit
```

**Continuous/Discrete Nodes:** To make a continuous node, use BNet.NewNode and pass 0 for the number of states. To later discretize it, or set the levels of an already `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_discrete_node.htm');return false;">discrete node`, simply change its **Level** property. You just set Node.Level(0), then Node.Level(1), etc. It will adjust the number of states of the node each time. If you have very many states, you may want to start with the highest first, since that is somewhat more efficient.

## C# Example Code

Below is an example C# program that does the same thing as the "Demo" program that ships with Netica C-API. There is a Visual Studio project for it, called "Netica Demo for C#" within the "Netica\Netica xxx\Programming Examples" folder of the Netica download package. ([more info](#) on programming Netica in C#)

```
using System;

using Netica;

namespace NeticaDemo
{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Welcome to Netica API for C# !");

Netica.ApplicationClass app = new Netica.ApplicationClass();

app.Visible = true;

string net_file_name = AppDomain.CurrentDomain.BaseDirectory + "..\\..\\..\\ChestClinic.dne";

Streamer file = app.NewStream(net_file_name, null);

BNet net = app.ReadBNet(file, "");

net.Compile();

BNode TB = net.Nodes.get_Item("Tuberculosis");
```

```
double bel = TB.GetBelief("present");

Console.WriteLine("The probability of tuberculosis is " + bel.ToString("G4"));

BNode XRay = net.Nodes.get_Item("XRay");

XRay.EnterFinding("abnormal");

bel = TB.GetBelief("present");

Console.WriteLine("Given an abnormal X-Ray, the probability of tuberculosis is " + bel.ToString("G4"));

net.Nodes.get_Item("VisitAsia").EnterFinding("visit");

bel = TB.GetBelief("present");

Console.WriteLine("Given abnormal X-Ray and visit to Asia, the probability of TB is " +
bel.ToString("G4"));

net.Nodes.get_Item("Cancer").EnterFinding("present");

bel = TB.GetBelief("present");

Console.WriteLine("Given abnormal X-Ray, Asia visit, and lung cancer, the probability of TB is
" + bel.ToString("G4"));

net.Delete();

if (!app.UserControl) app.Quit();

Console.WriteLine("Press <enter> to quit.");
```

```
Console.ReadLine();
```

```
}
```

```
}
```

```
}
```

## Visual Basic Example Code

Below is an example VB program that does the same thing as the "Demo" program that ships with Netica C-API. There is a Visual Studio project for it, called "Netica Demo for VB" within the " Netica\Netica xxx\Programming Examples " folder of the Netica download package. ([more info](#) on programming Netica in VB)

Sub Main()

On Error GoTo Failed

```
Dim app As Netica.Application
app = New Netica.Application
app.Visible = True

Dim net_file_name As String
net_file_name = System.AppDomain.CurrentDomain.BaseDirectory() & "..\..\ChestClinic.dne"
Dim net As Netica.Bnet
net = app.ReadBNet(app.NewStream(net_file_name))
net.Compile()

Dim TB As Netica.BNode
TB = net.Nodes.Item("Tuberculosis")
Dim belief As Double
belief = TB.GetBelief("present")
MsgBox("The probability of tuberculosis is " & belief)

net.Nodes.Item("XRay").EnterFinding("abnormal")
belief = TB.GetBelief("present")
MsgBox("Given an abnormal X-Ray, the probability of tuberculosis is " & belief)

net.Nodes.Item("VisitAsia").EnterFinding("visit")
belief = TB.GetBelief("present")
MsgBox("Given abnormal X-Ray and visit to Asia, the probability of tuberculosis is " & belief)

net.Nodes.Item("Cancer").EnterFinding("present")
```

```

belief = TB.GetBelief("present")
MsgBox("Given abnormal X-Ray, Asia visit, and lung cancer, the probability of tuberculosis is " & belief)

net.Delete()
If Not app.UserControl Then
app.Quit()
End If

Exit Sub
Failed:

MsgBox("NeticaDemo: Error " & (Err.Number And &H7FFFS) & ": " & Err.Description)

End Sub

```

Below is an example that reads in a net from the [= 4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\) BSPSPopupOnMouseOver\(event\);" class="BSSCPopup" onclick="BSSCPopup\('X\\_PU\\_examples\\_folder.htm'\);return false;">Examples folder](#) (you may have to change the path), then reads in cases and does belief updating.

-----

Sub Main()  
On Error GoTo Failed

```

Dim app As Netica.Application
app = New Netica.Application
Dim casefile As Streamer
Dim net As Bnet
Set netfile = app.NewStream("C:\Netica Data\BNs\Car_Diagnosis_0_Learned.dne")
Set casefile = app.NewStream("C:\Netica Data\Cases\Good Cases\Car Cases 10.cas")
Set net = app.ReadBNet(netfile)
net.AutoUpdate = 1
net.Compile
Dim lights_node As Bnode

```

```

Set lights_node = net.Node("Lights")
Dim lights_dim As Long
lights_dim = lights_node.GetStateIndex("dim")
Dim id As Long
Dim fr As Double
Dim caseposn As Long
Dim done As Boolean
done = False
caseposn = FirstCase
Do
net.RetractFindings
net.ReadFindings case_posn:=caseposn, stream:=casefile, IDNum:=id, freq:=fr
net.ReadFindings case_posn:=caseposn, stream:=casefile, nodes:=net.Nodes, IDNum:=id, freq:=fr
If caseposn = NoMoreCases Then
done = True
Else
MsgBox "Belief in Lights dim = " & lights_node.GetBelief(lights_dim)
End If
caseposn = NextCase
Loop Until done
net.Delete
Exit Sub

```

Failed:

```
MsgBox "Error " & ((err.Number And &H7FFF) - 10000) & ": " & err.Description
```

```
End Sub
```

=====

## EXAMPLES ON HOW TO SET CPT TABLE ENTRIES:

-----

Here is how you could set the CPTs of the "Chest Clinic" example from the manual:

```

Dim VisitAsia As BNode, Tuberculosis As BNode, Smoking As BNode
Dim Cancer As BNode, XRay As BNode, TbOrCa As BNode
Set VisitAsia = net.Node("VisitAsia")
...
Set TbOrCa = net.Node("TbOrCa")

```

Dim p(0 To 1) As Single

p(0) = 0.01: p(1) = 0.99: VisitAsia.CPTable("") = p

p(0) = 0.05: p(1) = 0.95: Tuberculosis.CPTable(Array(0)) = p

p(0) = 0.01: p(1) = 0.99: Tuberculosis.CPTable(Array(1)) = p

p(0) = 0.5: p(1) = 0.5: Smoking.CPTable("") = p

p(0) = 0.1: p(1) = 0.9: Cancer.CPTable(Array(0)) = p:

p(0) = 0.01: p(1) = 0.99: Cancer.CPTable(Array(1)) = p

p(0) = 0.98: p(1) = 0.02: XRay.CPTable(Array(0)) = p

p(0) = 0.05: p(1) = 0.95: XRay.CPTable(Array(1)) = p

p(0) = 1: p(1) = 0: TbOrCa.CPTable(Array(0, 0)) = p:

p(0) = 1: p(1) = 0: TbOrCa.CPTable(Array(0, 1)) = p:

p(0) = 1: p(1) = 0: TbOrCa.CPTable(Array(1, 0)) = p:

p(0) = 0: p(1) = 1: TbOrCa.CPTable(Array(1, 1)) = p:

Here are 6 alternate ways to set the CPT of the TbOrCa node.

Dim p(0 To 1) As Single

Dim s(0 To 1) As Integer

s(1) = 0: s(0) = 0: p(0) = 1: p(1) = 0: TbOrCa.CPTable(s) = p:

s(0) = 1: TbOrCa.CPTable(s) = p:

s(1) = 1: s(0) = 0: TbOrCa.CPTable(s) = p:

s(0) = 1: p(0) = 0: p(1) = 1: TbOrCa.CPTable(s) = p:

Dim p(0 To 1) As Single

p(0) = 1: p(1) = 0: TbOrCa.CPTable("present,present") = p:

p(0) = 1: p(1) = 0: TbOrCa.CPTable("present,absent") = p:

p(0) = 1: p(1) = 0: TbOrCa.CPTable("absent,present") = p:

p(0) = 0: p(1) = 1: TbOrCa.CPTable("absent,absent") = p:

```
TbOrCa.StateFuncTable("present,present") = "true":  
TbOrCa.StateFuncTable("present,absent") = "true":  
TbOrCa.StateFuncTable("absent,present") = "true":  
TbOrCa.StateFuncTable("absent,absent") = "false":
```

```
TbOrCa.CPTable("*,*") = Array(1, 0)  
TbOrCa.CPTable("absent,absent") = Array(0, 1)
```

```
TbOrCa.StateFuncTable("*,*") = "true"  
TbOrCa.StateFuncTable("absent,absent") = "false"
```

```
TbOrCa.Equation = "TbOrCa (Tuberculosis, Cancer) = (Tuberculosis || Cancer)"  
TbOrCa.EquationToTable num_samples:=1, samp_unc:=False, add_exist:=False
```

## Managed C++ Example Code

Below is an example Managed C++ (i.e., "C++/CLI") program that does the same thing as the "Demo" program that ships with Netica C-API. There is a Visual Studio project for it, called "Netica Demo for CLR C++" within the "Netica\Netica xxx\Programming Examples " folder of the Netica download package. ([more info](#) on programming Netica in Managed C++)

```
using namespace System;
```

```
using namespace Netica;
```

```
int main (array<String ^> ^args)
```

```
{
```

```
    Console::WriteLine ("Welcome to Netica API for Managed C++ !");
```

```
    Netica::Application^ app = gcnew Netica::Application;
```

```
    app->Visible = true;
```

```
    String^ net_file_name = AppDomain::CurrentDomain->BaseDirectory + "..\ChestClinic.dne";
```

```
    Streamer^ file = app->NewStream (net_file_name, nullptr);
```

```
    BNet^ net = app->ReadBNet (file, "");
```

```
    net->Compile();
```

```
    BNode^ TB = net->Nodes->Item ["Tuberculosis"];
```

```
    double bel = TB->GetBelief ("present");
```

```
    Console::WriteLine ("The probability of tuberculosis is " + bel);
```

```
    BNode^ XRay = net->Nodes->Item ["XRay"];
```

```
    XRay->EnterFinding ("abnormal");
```

```
    bel = TB->GetBelief ("present");
```

```
    Console::WriteLine ("Given an abnormal X-Ray, the probability of tuberculosis is " + bel);
```

```
    net->Nodes->Item["Cancer"]->EnterFinding("present");
```

```
    bel = TB->GetBelief ("present");
```

```
    Console::WriteLine ("Given abnormal X-Ray, Asia visit, and lung cancer, the probability of TB is " + bel);
```

```
net->Delete();  
if (!app->UserControl) app->Quit();  
  
Console::WriteLine ("Press <enter> to quit.");  
Console::ReadLine();  
  
return 0;  
}
```

## Non-English Bayes Nets

Although Netica Application is in English, it can be used to build Bayes nets in any language. Netica can work with non-Latin and international character sets (such as Chinese, Japanese, Arabic, Hebrew, or Unicode), including generating [SVG graphics](#) with those characters.

The following are the places where Netica will accept text in any character set:

- [Title](#) of each node (not its [name](#)).
- [Titles](#) of all the states of each node (not their [names](#)).
- [Comment](#) (description) of each node.
- [Comments](#) for each state of each node.
- Title of the net (not its name).
- [Overall comment](#) (description) of the net.
- [User-defined fields](#) of nodes and the net.

These are sufficient for the Bayes net developer to provide the end-user with a Bayes net in their native language.

**How To:** In any of the above places, you can paste in text from another program which uses a character set based on = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Unicode.htm');return false;">Unicode`. Or you can change the character set from the Windows task bar, and then type in the characters. Or you can use = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Netica_API.htm');return false;">Netica API` to set/get them with Unicode strings.

**Font:** Make sure you choose node [fonts](#), which are capable of displaying the characters you desire. A Unicode capable font, such as Arial Unicode MS, is usually a safe choice.

**Black Rectangles:** You may just see just black rectangles where there should be non-English text. If they appear on the net diagram, the problem is that the font that you have given the nodes is not capable of displaying the characters (as described in the paragraph above). If they appear within dialog boxes,

your computer needs to be configured to display the characters (as described in the paragraph below). If the dialog box contains a row of question marks instead, that indicates an invalid place to be entering non-English text.

**Configuring Computer:** To make international character sets available from the Windows XP task bar, choose **Start** → **Settings** → **Control Panel** → **Regional and Language Options** → **Languages**. If you desire one of the languages mentioned at the bottom check-boxes, and the box is not check-marked, first check-mark the box, click **OK**, restart your computer, and return to this dialog box.

Then click **Details** → **Settings** → **Add**, choose an input language, keyboard layout and click **OK**. You can now switch between different input languages using the Language Bar, which will appear (usually on the right side of the Windows task bar).

**Websites for Characters:** For some character sets it is more convenient to generate the characters with a special program or website.

For example, see: <http://code.cside.com/>

In particular: <http://code.cside.com/3rdpage/utf-8/> and  
<http://people.w3.org/rishida/scripts/pickers/>

## Command Line Options

If Netica is run from the command line it can accept a Netica *password*, which is used for that session, but not entered into the registry.

If Netica is run with command line parameters, then on start-up, Netica will put in the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window a line showing what command line parameters were used.

**Set Path:** To run Netica from the command line, you must start it from the directory that contains the Netica Application executable (i.e. the Netica = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_NeticaHomeFolder.htm');return false;">home directory). Alternately, you can include the home directory with the Netica file name (such as typing something like C:\Program Files\Netica\Netica 317\Netica.exe all enclosed in double quotes where necessary). A third, more permanent, choice is to put the Netica home directory in the MS Windows path environment variable.

You can find the home directory from Netica by choosing **Help** → **About Netica**, and then looking in the Messages window.

You can add to the environment path variable by following:

**Start** → **Settings** → **Control Panel** → **System** → **Advanced** → **Environment Variables** → **Path** → **Edit**.

Then add a semicolon and the Netica = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_NeticaHomeFolder.htm');return false;">home directory to the end.

**Running:** Bring up the command line dialog box by following: **Start** → **Run...** Then type in the file name of Netica (e.g. Netica or "Netica 325"). Or, you can use the shell interpreter, available by choosing **Start** → **Programs** →

## Accessories → Command Prompt.

Note that you need the quotes if there is a space in the file name. When you press **okay**, Netica should come up.

**Options:** You can put command line arguments after the Netica file name to achieve certain results. The following are the options:

**<file-name>** When Netica starts up, it will open the Bayes net with the given file name. You will need to put the full path, and don't forget quotes around it if there are spaces in the name. If the net was compiled and [AutoUpdating](#) was turned on when it was last saved, then when it comes up in Netica, it will be all ready to go.

Example: "Netica 305" "C:\Netica Data\MyBN.neta"

**-print <file-name>** Prints the indicated Bayes net file on the default printer in graphical format.

**-case <file-name>** Reads the first case from the indicated file, into the Bayes net it has opened. This option must follow an option to read in a Bayes net.

Example: Netica

"C:\Data\MyBN.neta"/case"C:\Data\MyCase.cas"

**-password <pw>** Runs Netica with the given license password. Ignores any installed = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_CY\_password.htm');return false;">*password*. It uses the password for this session of Netica only (i.e. it doesn't install the password on the computer for the next time Netica is run).

**-Embedding** This is only for those using the [COM interface](#) (it has the usual meaning under COM).

**Example:** If the following is run from the directory containing Bayes net file "ChestClinic.dne" and case file "NoAsia.cas" (and the home directory is on the system path):

> Netica ChestClinic.dne -case NoAsia.cas

It will bring up Netica, open the chest clinic net, and read the NoAsia case into that net.

**Under program control:** You may want to launch Netica directly from another program. Most development systems have a function to send a command to the operating system (e.g. a shell command).

For example, if you are programming in C/C++, you could call the Win32 function `CreateProcess`, or the function `WinExec`, as follows:

```
WinExec("Netica.exe", SW_SHOWNORMAL);
```

All the command line options work, so you could call:

```
WinExec("\"C:\\Netica\\Netica 307.exe\" C:\\Data\\MyBN.neta",  
SW_SHOWNORMAL);
```

## Obfuscating a Net

If the protection of intellectual property, or proprietary or classified information is a concern for you, Netica has the ability to obfuscate your net and to work with [encrypted](#) Bayes nets. The semantic content of the net is retained, so inference results will not be affected.

**How To:** If you are going to send the net to someone else, but it has proprietary information that you don't want to reveal to other users, choose **Modify** → **Obfuscate net**. A new window will open with the obfuscated net. If you select nodes before you obfuscate, it will only apply to those nodes. If no nodes are selected, it will apply to the entire net.

**Action:** All identifying information associated with the net (including node name, node title, node comment, state name, state title, state comment, net title, net comment, net file name, link names, node-set names) will be removed or converted into non-identifying labels.

**XRef Report:** These changes are logged into the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Messages_window.htm');return false;">Messages` window for future reference, called a *cross reference report*. If you wish to paste these results into an Excel spreadsheet, select the text in the cross reference report (excluding the report heading) and choose **Modify** → **Tabify Selection**, then press `CTRL+C` to copy the table. In an Excel workbook, press `CTRL+V` to paste the table.

You can use the cross reference report as a key to translate between the full net and the obfuscated net. For instance, if the obfuscated net is to be part of a package containing software that uses [Netica API](#) to read in the net, enter some findings and perform some queries, then the key will contain the information needed to convert the familiar names to the ones needed by Netica API.

**Encryption:** Another way to protect or hide the information in a Bayes net, is to use Netica's [encryption](#) capability. In fact, a net can be both obfuscated and encrypted.

## Encrypting a Net

Netica offers strong (64 bit) encryption of Bayes nets. It may be used for security during the development of classified Bayes nets, or to protect your IP when distributing Bayes nets.

**How To:** Make the net the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_active\_window.htm');return false;">active window, choose **File** → **Encryption Password**, and enter the password you wish to use. From then on, each time you save the net, it will be in encrypted form.

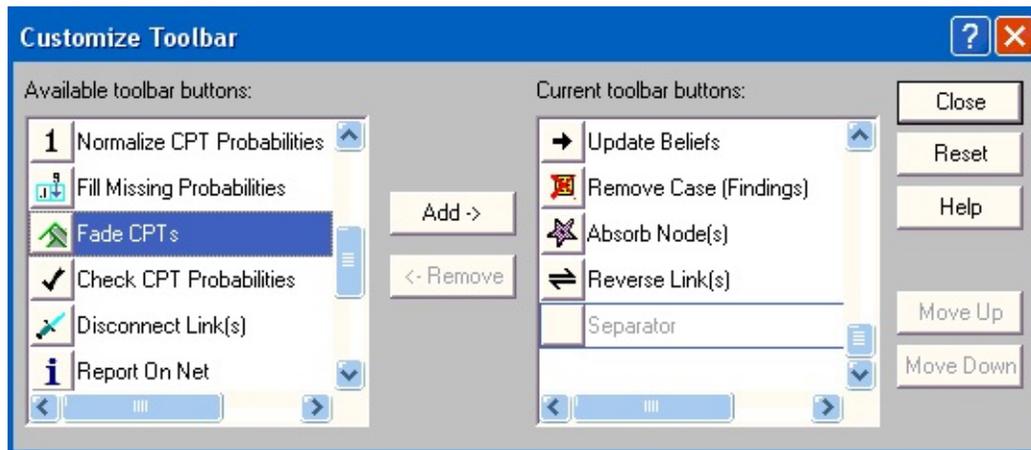
Once a net is saved in encrypted form, the only way to read it is using Netica, and to supply Netica with the same password used to encrypt it.

**For Netica API:** Encryption is often used to protect IP when distributing a Bayes net in a package which also contains [Netica API](#) and a specialized program that is going to use the net via Netica API. The specialized program has the password built into it (possibly in mutated or distributed form to evade detection), and it uses that password when it calls Netica API to read in the Bayes net.

**Obfuscation:** When you want to hide the identifying information in a Bayes net, but you can't use encryption because you have to make the net available, then [obfuscating](#) the net may be a better choice.

## Customizing the Toolbar

To change Netica's [toolbar](#), simply double-click on a blank area of the toolbar, or choose **Window** → **Customize Toolbar**, and use the resulting dialog box to make desired changes. The dialog box will look something like this:



To add buttons to the toolbar, drag them from the left list to the right list (or highlight it and click Add). Do the reverse to remove buttons from the toolbar.

To add a separator, drag it from the top of the left list to wherever you would like one on the right list. Buttons in the right list can also be dragged up and down to create an ideal order. Pressing the "Reset" button sets the toolbar to Netica's original buttons. When you are satisfied with your changes, click Close to save them.

Once they are on the toolbar, buttons can be rearranged or removed manually by holding down the **ALT** key while dragging them. Your changes will be saved when you exit Netica, so that they are available for the next session.

Many of these toolbar buttons are associated with accelerator keys. For a list of shortcut keys and their functions, click [here](#).

## Grouping Node States

Sometimes it is useful to organize the [states](#) of a node into groups, which effectively gives it a coarser resolution. Both the finer resolution and the coarser one can be used in the same Bayes net. Since the coarser node is less precise, using it instead of the fine one will result in less precise results, but the difference might be insignificant.

You have to judge what is appropriate for your particular application.

For Example: You may want to group the states of a country into regions. For example the finer node would have states CA,FL,TX,VA, etc. while the coarse one would have states West, Midwest, South, etc.

Another example is a continuous node that is `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_discretize.htm');return false;">` discretized with different resolutions. For example a coarse temperature node may have the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_state_threshold.htm');return false;">` thresholds 0,10,20, etc. while the fine one has thresholds 0,5,10,15, etc.

### Reasons for grouping states:

- Because you have probabilities for the `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">` CPTs of other nodes given the coarse node, but not given the fine node. Or perhaps you just want to elicit the probabilities relating to the coarse node because the tables are smaller, and it is easier.
- You want the Bayes net to [learn](#) from multiple data sources (e.g. two different databases), and they describe variables in different ways (perhaps one is more detailed, and provides more possible states for a variable).
- To ease the computational burden (e.g. using the coarse node instead of the fine one as the parent to those nodes that have a lot of other parents).

- You want to display results in summarized form. Perhaps the end-user viewing the [belief-bar](#) display wants to see the probabilities for a whole group of states, instead of a detailed breakdown.
- You may want to allow for the entry of findings that aren't specified precisely. For example, allowing an entry of "car" for a Vehicle\_Type node, rather than "sedan", "station wagon", etc. since that information might not be known. Essentially, this results in entering a logical disjunction over the finer states.
- You are only interested in one particular state, and you group the rest of them together as "other".

**How To:** Make two nodes and give them names that are almost the same, to indicate they are for the same variable. Perhaps the coarser one could be suffixed with "\_Approx". Sometimes it is best to append a word which describes the level of resolution, such as "Country", "Region" or "Province".

One node should be given the set of fine states, the other given the set of coarse states.

In the case of a continuous variable, the finer node would have the same threshold levels as the coarse one, but with some extra thresholds added. If the fine node is missing some of the thresholds the coarse one has, the method will still work, but will introduce extra uncertainty, since Netica must turn the coarse node into a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_chance\_node.htm');return false;">chance node when it builds the table.

Draw a link from the fine node to the coarse node. That must be the only link entering the coarse node (if you need to have other links entering it, use the "equality constraint link" method instead).

Enter a table to show the functional relationship between the fine and coarse nodes. Bring up the table editor for the coarse node, change it from **Chance** to **Deterministic**, and then go through all the rows (they correspond to the states of the fine node), entering in the right hand column the equivalent state for the coarse node (by clicking on it and choosing from the menu).

Instead you may want to show the functional relationship with an equation.

Bring up the [node properties box](#) for the coarse node, use the multi-purpose selector at the bottom to choose **Equation**, and then enter an equation like:

```
VoterRegion (VoterState) =  
  (VoterState == CA) ? West:  
  (VoterState == OR) ? West:  
  (VoterState == FL) ? South:  
  ...  
  (VoterState == VA) ? East: Other
```

or you could use the style:

```
VoterRegion (VoterState) =  
  (VoterState == CA || VoterState == OR || VoterState == WA) ? West:  
  (VoterState == FL || VoterState == TX) ? South:  
  ...  
  (VoterState == VA || VoterState == NY || VoterState == MA) ? East: Other
```

For discretized continuous nodes, you simply put an equality, such as:

```
Temperature_Approx (Temperature) = Temperature
```

After entering the equation, click **OK** on the dialog, and [convert](#) the equation to a table.

Then link the two nodes into the network the way you want. Each can be a parent or child to any other node you want, except the coarse node can not be a child. If that is required, use an "equality constraint link".

Remember, if you are using Netica to [learn from data](#), giving a node the state "other" will catch any state you haven't explicitly given the node.

# Combining Nets

The purpose of this feature is to combine the knowledge of two or more Bayes nets into a single net. The nets may have been learned from data, or created by experts.

You start with one net (the *original net*), and then add on another (the *additional net*), to form the *resultant net*. This process may be repeated to combine any number of nets.

If nodes in each of the nets have the same name (regardless of states), they will be interpreted as standing for the same variable, and will result in a single node in the resultant net. Among nodes having the same name, the resultant node will have all the states of the original node and the states of the additional node. `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPTs` will be combined accordingly.

**How to:** Open your original net, or [create](#) a new one. Choose **Modify** → **Combine Nets**, and from the standard file-open dialog box, pick the Bayes net file to be added on. If you wish to combine several nets, then after adding the first one, successively add the others in the same way.

Netica will next ask you to enter a *degree* for the additional net. It is an amount to weight the knowledge in that net relative to the first net. You will almost always accept the default of 1.0, since the experience amounts already weight the nets relative to each other, and the degree is just an extra "experience multiplier" to provide added flexibility (see below).

**Experience:** If the nets were learned from data, then they will have `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_experience.htm');return false;">experience` tables as well as `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT` tables, which will weight the nets according to how much data was used to learn each one. If the nets were created by experts rather than learned from data, they may

have CPT tables with no experience tables. In that case you will be prompted for a single experience number to be applied to the whole net. The effect of the number you enter is just to weight the nets produced by each expert, but the idea is to enter a number that is an estimate of the equivalent number of cases seen by that expert. If you wish to have more control over the experience amounts (for instance, different numbers for each node, or even a table of different numbers for each node), then [enter](#) the experience tables before doing the combining operation.

**Structure:** The structure of the original net and the additional net do not have to be the same, but they must be compatible. If in doubt, try combining them and observing the results obtained. You may want to adjust the structure of one or both nets before-hand to make them compatible. For that purpose you can use [link reversals](#), or even better, you can construct the CPTables with [Build From Other Net](#).

**Degree:** The way that Netica uses the degree as an experience multiplier is: If the degree is less than one it multiplies all experience numbers of the additional net by the degree, and if it is more than one, it multiplies all the experience numbers of the original net by  $1/\text{degree}$ . So experience numbers are never increased, and the magnitude of the degree indicates how much more reliable the additional net is compared to the original net.

**Alternative:** Another way to combine nets is to [copy](#) nodes from the additional net, and then paste them into the original net. You might have to hook up some [disconnected links](#). That method will not combine CPT tables.

# Shortcut Keys for Netica

	<b>F1</b>	Help <a href="#">INFO</a>
	<b>F3</b>	Find Next <a href="#">INFO</a>
SHIFT	<b>F4</b>	Tile Windows <a href="#">INFO</a>
CTRL	<b>F4</b>	Close Window <a href="#">INFO</a>
ALT	<b>F4</b>	Exit Netica <a href="#">INFO</a>
SHIFT	<b>F5</b>	Cascade Windows <a href="#">INFO</a>
	<b>F6</b>	Random Case <a href="#">INFO</a>
	<b>F7</b>	Paste <a href="#">INFO</a>
	<b>F8</b>	Get Next Case <a href="#">INFO</a>
SHIFT	<b>F8</b>	Get Previous Case <a href="#">INFO</a>
	<b>F9</b>	Add Node (single) <a href="#">INFO</a>
	<b>F9-F9</b>	Add Node (multiple) <a href="#">INFO</a>
	<b>F10</b>	Add Link, (single) <a href="#">INFO</a>
	<b>F10-F10</b>	Add Link (multiple) <a href="#">INFO</a>
CTRL	<b>A</b>	Select All <a href="#">INFO</a>
CTRL+SHIFT	<b>A</b>	Invert Selection <a href="#">INFO</a>
CTRL	<b>B</b>	Report Beliefs <a href="#">INFO</a>
CTRL	<b>C</b>	Copy <a href="#">INFO</a>
CTRL	<b>E</b>	Report Case (Evidence) <a href="#">INFO</a>

CTRL	<b>F</b>	Find <a href="#">INFO</a>
CTRL	<b>G</b>	Make SVG Graphic <a href="#">INFO</a>
CTRL+SHIFT	<b>G</b>	Go Back <a href="#">INFO</a>
CTRL	<b>L</b>	Learn From Case <a href="#">INFO</a>
CTRL	<b>N</b>	New Bayes Net Window <a href="#">INFO</a>
CTRL+SHIFT	<b>N</b>	New Node-Set <a href="#">INFO</a>
CTRL	<b>O</b>	Open File <a href="#">INFO</a>
CTRL	<b>P</b>	Print <a href="#">INFO</a>
CTRL	<b>R</b>	Remove Findings <a href="#">INFO</a>
CTRL	<b>S</b>	Save <a href="#">INFO</a>
CTRL+SHIFT	<b>S</b>	Select Node-Set <a href="#">INFO</a>
CTRL	<b>T</b>	Edit Table (CPT) <a href="#">INFO</a>
CTRL	<b>U</b>	Update by Sampling <a href="#">INFO</a>
CTRL	<b>V</b>	Paste <a href="#">INFO</a>
CTRL	<b>W</b>	Close Window <a href="#">INFO</a>
	<b>x</b>	Fill with impossible character, CPT Table <a href="#">.INFO</a>
CTRL	<b>X</b>	Cut <a href="#">INFO</a>
CTRL	<b>Y</b>	Redo <a href="#">INFO</a>
CTRL+SHIFT	<b>Z</b>	Redo <a href="#">INFO</a>
CTRL	<b>Z</b>	Undo <a href="#">INFO</a>
	<b>0</b>	Fill with 0, CPT Table <a href="#">INFO</a>

CTRL	<b>0</b>	Fill in Missing CPT Entries <a href="#">INFO</a>
CTRL	<b>1</b>	Normalize CPT Row or Table <a href="#">INFO</a>
CTRL	<b>8</b>	Select Incomplete CPT Table <a href="#">INFO</a>
CTRL	*	Select Incomplete CPT Table <a href="#">INFO</a>
	=	Make uniform CTP Table <a href="#">INFO</a>
CTRL	<b>Insert</b>	Copy <a href="#">INFO</a>
SHIFT	<b>Insert</b>	Paste <a href="#">INFO</a>
CTRL	<b>Tab</b>	Cycles Among Open Windows
ALT	<b>Tab</b>	Cycles Through Running Applications
ALT	<b>Enter</b>	Displays Properties of Selected Item
CTRL	>	Zoom Out <a href="#">INFO</a>
CTRL	<	Zoom In <a href="#">INFO</a>
CTRL+SHIFT	>	Zoom To Fit <a href="#">INFO</a>
CTRL+SHIFT	<	Zoom To Normal <a href="#">INFO</a>
CTRL+ALT	>	Zoom To Percentage <a href="#">INFO</a>
CTRL+ALT	<	Zoom Back <a href="#">INFO</a>
	<b>SpaceBar</b>	Zoom Global <a href="#">INFO</a>
ALT	⇧ <b>Back</b>	Undo <a href="#">INFO</a>
SHIFT	<b>Delete</b>	Cut <a href="#">INFO</a>
	<b>Delete</b>	Delete <a href="#">INFO</a>
	<b>Pause</b>	Normal Mode <a href="#">INFO</a>

## General MS Windows shortcuts:

**ALT + UNDERLINED LETTERS** in the menu does the corresponding menu command

**WINDOW + L** locks computer

**CTRL + DRAG** duplicates what is being dragged

You may want to print out the above table, to keep as a handy reference guide.

# File Formats

## Bayes Net Files

.dne	R/W	Netica Bayes net file in text form <a href="#">INFO</a>
.neta	R/W	Netica Bayes net file in binary form (possibly <a href="#">encrypted</a> ) <a href="#">INFO</a>
.net	R	Hugin text file for Bayes nets. Netica can read files created by Hugin 4.*, 5.* and 6.* <a href="#">INFO</a>
.dxp	R	Knowledge Industries file for DXpress
.dsc	R	BNIF file
.ergo	R	Noetic Systems file for Ergo <a href="#">INFO</a>

## Case Files or Data Files

.cas	R/W	Netica case file <a href="#">INFO</a>
.uvf	R/W	Netica case file in UVF (uncertain value format) <a href="#">INFO</a>
.csv	R/W	Comma separated value (may have other extensions) <a href="#">INFO</a>
.txt	R/W	Tab delimited text (may have other extensions) <a href="#">INFO</a>

## Graphics Files

.svg	W	XML scalable vector graphics file <a href="#">INFO</a>
.svgz	W	Compressed SVG file <a href="#">INFO</a>

## Text Files

.txt	R/W	Ascii text file (may have other extensions) <a href="#">INFO</a>
.dne	R/W	<a href="#">INFO</a>
.cas	R/W	<a href="#">INFO</a>
.xml	R	<a href="#">INFO</a>
.svg	W	<a href="#">INFO</a>

"R" means Netica can read the file and "W" means it can write the file.

## Bibliography

For a brief description of some of these, see [Introductory References](#). Each bibliography reference is also listed in the index, so you can find where in this document it is mentioned.

Boerlage, Brent (1994) *Link Strength in Bayesian Networks*, UBC, British Columbia, Canada. See: [Tech Report](#)

Cain, Jeremy (2001) *Planning Improvements in Natural Resource Management*, Centre for Ecology & Hydrology, Wallingford UK.

Charniak, Eugene (1991) "Bayesian networks without tears" in *AI Magazine* (Winter 1991), **12**(4), 50-63.

Clemen, Robert and Terence Reilly (2001) *Making Hard Decisions with DecisionTools*, Brooks/Cole, Pacific Grove, CA.

Cowell, Robert, A. Philip Dawid, Steffen L. Lauritzen and David J. Spiegelhalter (1999) *Probabilistic Networks and Expert Systems*, Springer, New York.

Darwiche, Adnan (2009) *Modeling and Reasoning with Bayesian Networks*, Cambridge University Press.

Friedman, Nir, Dan Geiger, and Moises Goldszmidt (1997) "Bayesian network classifiers" in *Machine Learning*, Vol 29, 131-163.

Glymour, Clark (2001) *The Mind's Arrows: Bayes Nets and Graphical Causal Models in Psychology*, The MIT Press, Cambridge, MA.

Heckerman, David and Jack Breese (1994) "A new look at causal independence" in *Uncertainty in Artificial Intelligence: Proc. of the Tenth Conf.* (July, Seattle, WA), Ramon Lopez de Mantaras and David Poole (eds.), Morgan Kaufmann, San Mateo, CA.

Heckerman, David, Abe Mamdani and Michael P. Wellman (1995) "Real-world applications of Bayesian networks" in *Communications of the ACM*, **38**(3), 24-26. Introduction to this special issue of CACM on Bayes nets.

Henrion, Max, John S. Breese and Eric J. Horvitz (1991) "Decision analysis and expert systems" in *AI Magazine* (Winter 1991), **12**(4), 64-91.

- Jensen, Finn V. (1996) *An Introduction to Bayesian Networks*, Springer-Verlag, New York.
- Jensen, Finn V. and Thomas D. Nielsen (2007) *Bayesian Networks and Decision Graphs*, 2nd Edition, Springer, New York.
- Jensen, Frank, Finn V. Jensen and Soren L. Dittmer (1994) "From influence diagrams to junction trees" in *Uncertainty in Artificial Intelligence: Proc. of the Tenth Conf.* (July, Seattle, WA), Ramon Lopez de Mantaras and David Poole (eds.), Morgan Kaufmann, San Mateo, CA.
- King, Jack L. (2001) *Operational Risk: Measurement and Modelling*, Wiley.
- Kjaerulff, Uffe B. and Anders L. Madsen (2008) *Bayesian Networks and Influence Diagrams*, Springer, New York.
- Koller, Daphne and Nir Friedman (2009) *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, Cambridge, MA.
- Korb, Kevin and Ann E. Nicholson (2004) *Bayesian Artificial Intelligence*, Chapman & Hall, London, UK.
- Lauritzen, Steffen L. (1995) "The EM algorithm for graphical association models with missing data" in *Computational Statistics and Data Analysis*, **19**(2), 191-201.
- Lauritzen, Steffen L. and David J. Spiegelhalter (1988) "Local computations with probabilities on graphical structures and their application to expert systems" in *J. Royal Statistics Society B*, **50**(2), 157-194.
- Matheson, James E. (1990) "Using Influence diagrams to value information and control" in *Influence Diagrams, Belief Nets and Decision Analysis*, Robert M. Oliver and J. Q. Smith (eds.), John Wiley & Sons, Chichester.
- Neapolitan, Richard E. (1990) *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley & Sons, New York. Currently out of print.
- Neapolitan, Richard E. (2004) *Learning Bayesian Networks*, Pearson Prentice Hall, Upper Saddle River, NJ.
- Neapolitan, Richard (2007) *Probabilistic Methods for Financial and Marketing Informatics*, Morgan Kaufmann Publishers.

- Neapolitan, Richard (2009) *Probabilistic Methods for Bioinformatics: With an Introduction to Bayesian Networks*, Morgan Kaufmann Publishers.
- NRC-CNRC (2006) *Canadian Journal of Forest Research*, Vol 36, Number 12, December 2006.
- Pearl, Judea (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA. 2nd edition 1991.
- Pearl, Judea (2009) *Causality- Models, Reasoning, and Inference*, 2nd edition, Cambridge University Press, Cambridge, UK.
- Pourret, Olivier, Patrick Naim and Bruce Marcot (2008) *Bayesian Networks: A Practical Guide to Applications*, Wiley.
- Russell, Stuart and Peter Norvig (2009) *Artificial Intelligence: A Modern Approach* (3rd edition), Prentice Hall, Englewood Cliffs, NJ.
- Shachter, Ross D. (1986) "Evaluating influence diagrams" in *Operations Research*, **34**(6), 871-882.
- Shachter, Ross D. (1988) "Probabilistic inference and influence diagrams" in *Operations Research*, **36**(4), 589-604.
- Shachter, Ross D. (1989) "Evidence absorption and propagation through evidence reversals" in *Proc. of the Fifth Workshop on Uncertainty in Artificial Intelligence* (Windsor, Ont.), 303-308. Later republished in: Henrion, Max (ed.) (1991) *Uncertainty in Artificial Intelligence 5*, North-Holland, Amsterdam.
- Shafer, Glenn (1996) *The Art of Causal Conjecture*, MIT, MA, USA.
- Sloman, Steven A. (2005) *Causal Models: How People Think about the World and Its Alternatives*, Oxford University Press, NY, USA.
- Smith, James E., Samuel Holtzman and James E. Matheson (1993) "Structuring conditional relationships in influence diagrams" in *Operations Research*, **41**(2), 280-297.
- Spiegelhalter, David J., A. Philip Dawid, Steffen L. Lauritzen and Robert G. Cowell (1993) "Bayesian analysis in expert systems" in *Statistical Science*, **8**(3), 219-283.

Spirtes, Peter, Clark Glymour and Richard Scheines (2000) *Causation, Prediction, and Search* Second Edition, The MIT Press, Cambridge, MA.

Zhang, Lianwen (Nevin), Runping Qi and David Poole (1994) “A computational theory of decision networks” in *International Journal of Approximate Reasoning*, **11**(2), 83-158.

## Glossary A - E

**Absorption:** Node *absorption* is the process of removing a node from a Bayes net or decision net, and adjusting the remaining links and node tables so that subsequent inference done on the remaining nodes will yield the same results. Usually Netica has to add some new links to maintain the global relationships between the nodes. [More Info](#)

**Active Window:** Menu and key commands generally apply to the *active window*, which is the window with the non-dim title bar, and is the front most window (except possibly for some dialog boxes and help windows). You can make a window active by clicking on an exposed part of it, or by choosing its title from the **Window** menu.

**ASCII:** *ASCII* is a text character encoding based on the English alphabet, and was first released as a standard in 1967. It represents each character with 7 bits, although there are many 8-bit extensions of it. ASCII and its extensions have been by far the dominant way to represent text in a computer, but are now being overtaken by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Unicode.htm');return false;">Unicode, which can represent many more characters. In Netica, identifiers (i.e. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_IDname.htm');return false;">IDnames) are always composed only of ASCII characters, while titles and descriptions may be in Unicode.

**Assessment:** Probability *assessment* is the process of humans determining the probabilistic or deterministic relationships between nodes and their parents (usually in the form of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditional\_probability.htm');return false;">conditional probability tables) after all the nodes and the link structure have been created. Alternatively, they can be determined automatically by some learning procedure.

**Auto-updating:** If a Bayes net is *auto-updating*, then whenever its = 4 &&

```

typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_belief.htm');return false;">beliefs become
invalid, perhaps due to = 4 && typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_enter_finding.htm');return false;">entering new
findings, they are automatically = 4 && typeof(BSPSPopupOnMouseOver)
== 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_belief_updating.htm');return
false;">recalculated (provided the net is = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_compile_net.htm');return false;">compiled).

```

Since updating may be time consuming, you may prefer to only have the net updated when you manually request it. The default for new nets is that they are auto-updating. [More Info](#)

**Background:** The *background* of a Bayes net or decision net is the area (within its window) which is not covered by a node or a link.

**Barren node:** A *barren node* is a node with no = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_child\_node.htm');return false;">children, and that is not a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_findings\_node.htm');return false;">findings node or a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_query\_node.htm');return false;">target node. During = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief\_updating.htm');return false;">belief updating and finding = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_optimal\_policy.htm');return false;">optimal decisions, nodes that are barren don't influence the results, and may simply be removed.

**Bayes net:** A *Bayes net* (also known as a belief net) is composed of a set of nodes representing variables of interest, connected by links to indicate dependencies, and containing information about the relationships between the nodes (often in the form of conditional probabilities). Usages include prediction, diagnosis, probabilistic modeling, learning from data and forming a basis for building = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision nets. [More Info](#)

**Belief:** The *belief* of a node is the set of probabilities (one for each of its possible states), taking into account the currently entered findings by using the knowledge encoded in the Bayes net. Technically speaking, it is the marginal posterior probability distribution of the node, given the findings and the Bayes net model. Sometimes the plural form “beliefs” is used to mean each of the probabilities in the set.

**Belief updating:** *Belief updating* is the process of finding new = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">beliefs for the nodes of a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes net to account for the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings that are currently known. It is a form of probabilistic inference. During belief updating the Bayes net model (in particular, the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditional\_probability.htm');return false;">conditional probability tables between the nodes) is not modified at all; for that = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probability\_revision.htm');return false;">probability revision is used. [More Info](#)

**Bernoulli process:** A *Bernoulli process* consists of a series of independent

trials, each with two possible outcomes (often labeled "success" and "failure"), with a constant probability,  $p$ , of success (such as a set of coin tosses). If there are  $n$  trials, it is also called a *binomial experiment*, and the total number of successes,  $k$ , is given by the [binomial distribution](#). The number of trials needed before getting a fixed number of successes is given by a [negative binomial distribution](#), and the number of trials needed to get one success is given by a [geometric distribution](#). If there are more than two possible outcomes, it is a multinomial experiment, and its results are given by a [multinomial distribution](#).

**Binary node:** A *binary node* is a node with exactly two states. If those states correspond to 'true' and 'false', then it is also called a *boolean node*.

```

typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_state.htm');return false;">states
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_boolean_node.htm');return false;">boolean
node.

```

**Binomial coefficient:** The *binomial coefficient*, denoted  $\text{binomial}(n,k)$ , is the number of different  $k$ -sized groups that can be drawn from a set of  $n$  distinct elements. Its value is given by:  $\text{binomial}(n,k) = n! / (k! * (n-k)!)$  Within a Netica equation, you can represent it with the [binomial](#) function.

**Boolean node:** A *boolean node* is a node with two states, and whose state names are (true, false), (yes, no), (present, absent) or (on, off). These state names can be in either order, and in lower or upper case (such as True or TRUE). Some people refer to them as "propositional nodes". They are examples of *binary nodes*.

```

typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_binary_node.htm');return false;">binary nodes.

```

**Cartesian product:** The *cartesian product* of two sets is the set of all possible pairs of elements, where the first element of each pair is taken from the first set, and the second element is taken from the second set. For example, the cartesian product of {low, medium, high} with {true, false} is {(low, true), (low, false), (medium, true), (medium, false), (high, true), (high, false)}

[More Info](#)

**Case:** A *case* is a set of findings that go together to provide information on one object, event, history, person, or other thing. [More Info](#)

**Case symbol:** The *case symbol* is a tiny yellow page with writing to depict a list of attributes and values. It looks like this:  It is used on toolbar buttons to indicate a case. [More Info](#)

For example, the  button removes the current case, the  button saves the case to a file, and the  button reads a case from file.

**Central limit theorem:** The *central limit theorem* states that the distribution of the sum of a set of random variables approaches the normal distribution as the number of variables increases, provided they are independent and some other weak conditions (such as Liapounov's conditions) are met. These conditions will always be met if they are identically distributed and their means and standard deviations exist.

**Chance node:** A *chance node* is a node whose value is probabilistic (i.e. not deterministic). If its parents values are all known, and there is no further information, then its value can only be inferred as a probability distribution over possible values. Compare with a *deterministic node*.

```
onclick="BSSCPopup('X_PU_deterministic_node.htm');return false;">deterministic node.
```

**Changed Indicator:** A *changed-indicator* (sometimes known as a "dirty indicator") is a \* or + after the title of the net in the net window's title bar, and means that the net currently displayed has been changed from the version saved to file. \* means a meaningful (i.e. "semantic") change, while + means just a change to the visual display. Entering findings will not trigger the changed-indicator, unless they are for a [constant](#) node. The changed-indicator properly responds to undo and redo. ([more info](#))

```
A = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_table_dialog_box.htm');return false;">table
dialog
```

can also have a \* changed-indicator. Or it can have a (\*) indicator, which means that something else has changed the target, which makes it different from the version in the dialog box. [More Info](#)

```
Child node: If there is a = 4 && typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_link.htm');return false;">link going from = 4
&& typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_node.htm');return false;">node A to node B,
```

then B is said to be a *child node* of A. Some people refer to it as a *direct successor*.

**Clipboard:** Whenever you do a copy operation (for example by choosing **Edit** → **Copy**, or pressing **CTRL+C**), the material you copy goes to the *clipboard*, from which you can paste it where you like.

**Clique:** A *clique* is a set of nodes in which each node is connected to all the other nodes of the set, and there isn't any other node in the net which is connected to all the nodes in the set. Some people call this a "maximal clique". When Netica compiles a Bayes net, one step is to find the cliques of

```
the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_triangulated.htm');return false;">triangulated =
4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

onclick="BSSCPopup('X\_PU\_Markov\_network.htm');return false;">Markov network.

**Compile:** When Netica *compiles* a Bayes net it takes a representation of the net similar to what you see on the screen, and from it builds a new representation called a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_junction\_tree.htm');return false;">junction tree, which it can use to do fast = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return false;">probabilistic inference. [More Info](#)

**Conditional probability:** The *conditional probability* of an event is the probability of the event occurring under certain given conditions. [More Info](#)

**Conditionally independent:** A variable X is said to be *conditionally independent* of another variable Y given knowledge Z, if obtaining knowledge about the value of X does not change your beliefs about the value of Y when you already know Z.

**Conjugate distributions:** *Conjugate distributions* are probability distributions from a family of distributions, such that if the prior distribution belongs to the family, then for any sample size and any observations, the posterior distribution also belongs to the family. Usually each distribution in the family can be specified by one or two parameters, so only these parameters need to be kept track of during probabilistic inference, which is particularly convenient.

**Constant node:** Sometimes it is useful to have something that normally acts as a fixed constant, but which you can change from time to time. That is the purpose of a *constant node*. [More Info](#)

**Contingency table:** A *contingency table* provides a value (or set of values) for each possible configuration of values for some given variables. In other words, it specifies a function of the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_cartesian\_product.htm',400,145);return false;">cartesian product of the variables. Netica's = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_table_dialog_box.htm');return false;">table
dialog box can be used to view or edit contingency tables.
```

**Continuous variable:** A *continuous* variable is one which can take on a value between any other two values, such as: indoor temperature, time spent waiting, water consumed, color wavelength, and direction of travel. A = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_discrete.htm');return false;">discrete variable
corresponds to a digital quantity, while a continuous variable corresponds to an
analog quantity. More Info
```

**CPT:** *CPT* is an abbreviation for conditional probability table (also known as “link matrix”), which is the = 4 &&

```
typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_contingency_table.htm');return
false;">contingency table of = 4 &&
```

```
typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_conditional_probability.htm');return
false;">conditional probabilities stored at each node, containing the
probabilities of the node given each configuration of = 4 &&
```

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parent values.
```

Sometimes CPT is used to refer to the deterministic = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_function_table.htm');return false;">function
table of a node, since the node's conditional probabilities can easily be found
```

```
from that. It is a form of node = 4 && typeof(BSPSPopupOnMouseOver) ==
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_node_relation.htm');return false;">relation, so
you use the table dialog box to change or view it.
```

**CSV file:** *CSV file* is a commonly used term for a form of [case file](#) in which the names of the variables appear on the first line, and then below are all the cases (i.e. records), with each case on a single line and having a value for each

of the variables, and with all the values and variables in = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_text\_file.htm');return false;">text form and separated by commas (i.e. "Comma Separated Values"). See also = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_tab\_delimited\_file.htm');return false;">tab delimited text.

**Cutset:** A *cutset* for two sets of nodes A and B is a third set C, such that if all the nodes in C are removed from the net (along with all links involving them), then there is no path from a node in A to a node in B. See also [loop cutset](#).

**d-separation:** The *d-separation* rule enables you to quickly determine whether a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">finding at one node can possibly change the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">beliefs at another node by only considering the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link\_structure.htm');return false;">link structure of a Bayes net. [More Info](#)

**Dag:** A *dag* is a net with no = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_directed\_cycle.htm');return false;">directed cycles (although it may have = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_undirected\_loop.htm');return false;">undirected loops). The word is a shortened form of “directed acyclic graph”.

**Decision net:** If = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_node.htm');return false;">decision nodes (representing variables that can be controlled) and = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_utility\_node.htm');return false;">utility nodes  
(representing variables to be optimized) are added to a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes net, then  
a *decision net* (also known as an “influence diagram”) is formed. [More Info](#)

**Decision node:** A *decision node* is a node in a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision net  
which represents a variable (or choice) under the control of the decision maker.  
When the net is solved, a = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_decision\_rule.htm');return false;">decision rule  
is found for the node which optimizes the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_expected\_value.htm');return false;">expected  
utility. Decision nodes are normally drawn as rectangles (without rounded  
corners).

**Decision rule:** A *decision rule* indicates which option to choose in making a  
certain decision, for each possible condition that may be known when the  
decision is to be made. In other words, for a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_decision\_node.htm');return false;">decision  
node, it is a function which provides a value for each member of the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_cartesian\_product.htm');return false;">cartesian  
product of the parents of the decision node.

**Decision theory:** *Decision theory* is a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_normative.htm');return false;">normative

**theory** which indicates how a single agent should best make decisions to maximize his expected utility. It considers sequences of decisions, what information the agent will have when he makes the decisions, uncertainties in the beliefs of the agent, and complex probabilistic interactions in the environment in which the agent is operating.

**Default node style:** There is one *default node style* for the whole net, which is the style to display any node which doesn't have an overriding style. It is set by choosing it from the **Style** menu when no nodes are selected. [More Info](#)

**Deselect:** To *deselect* all the nodes of a net, click on its background (i.e. within the window, but not on a node or link). [More Info](#)

**Deterministic node:** A *deterministic node* is a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_nature\_node.htm');return false;">nature node whose relationship with its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents is given as a function of the parent values (i.e. deterministic rather than probabilistic). If the parent values are all known, its value can be determined with certainty. Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_chance\_node.htm');return false;">chance node.

**Deterministic updating:** Before doing = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief\_updating.htm');return false;">belief updating, Netica does *deterministic updating* when it can, for greater speed and accuracy. If all the parents of a node have = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings, and the node has a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_function\_table.htm');return false;">function table or a deterministic equation, then its value can be found exactly (the

equation is used without = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretization) very quickly.

**Directed cycle:** A *directed cycle* (sometimes just called a “cycle”) is a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_directed\_path.htm');return false;">path through a net, following the direction of the arrows, which returns to its beginning (i.e. the first node of the path is the same as the last). Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_undirected\_loop.htm');return false;">undirected loop.

Netica can [find and display](#) directed cycles.

**Directed network:** A *directed network* is one where the links have direction (i.e. arrows). Bayes nets and decision nets are examples. Compare with an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_undirected\_network.htm');return false;">undirected network.

**Directed path:** A *directed path* is a sequence of nodes from a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_net.htm');return false;">net, such that you can get from one node of the sequence to the next node by traversing a link between them in the direction of its arrow. Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_path.htm');return false;">path.

**Dirty indicator:** see *changed indicator*.

**Disconnected link:** A *disconnected link* is one that has been disconnected from the parent node it originally came from, and = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_belief\_updating.htm');return false;">belief  
updating cannot proceed until the link is reconnected to its parent, or to  
another similar node. [More Info](#)

**Discrete node:** A *discrete node* is a node representing a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_discrete.htm');return false;">discrete variable.  
The states of a node constitute the domain of the categorical variable.

**Discrete variable:** A *discrete variable* is one with a well defined finite set of  
possible values, called = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_state.htm');return false;">states. Examples are:  
the number of dimes in a purse, the 'true' or 'false' value of a statement, which  
party will win the election, the country of origin, and the place a roulette wheel  
stops. A discrete variable corresponds to a digital quantity, while a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous  
variable corresponds to an analog quantity. [More Info](#)

**Discretize:** Often it is useful to have a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">continuous  
variable behave like a = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_discrete.htm');return false;">discrete one. To  
do this, select the node(s) and choose **Modify** → **Discretize Node**. You can  
break up the total range of the continuous variable into a number of intervals  
by supplying numbers showing where one interval ends and the next begins  
(called = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_state\_threshold.htm');return false;">thresholds).  
This is known as *discretizing* the variable. Each interval results in one state of  
the discrete version of the variable. [More Info](#)

**Elimination order:** The *elimination order* is simply an ordered list of all the nodes in the net, which specifies how to = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_triangulated.htm');return false;">triangulate the net during = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_compile\_net.htm');return false;">compiling (see [Spiegelhalter&DLC93](#)). Triangulation is the most critical step in producing an efficient compilation, so the order is included when the net is next saved to file if an “Optimized Compile” command (which finds a very good elimination order) has been done. The elimination order is indicated by the numbers following the node names when you view the net in “[triangulated style](#)”.

**Ellipsis:** An *ellipsis* is three closely spaced dots, used to indicate that there is more to follow, but that it has been left out to save space. For example: LongNameFor...

**E-mail:** The = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Norsys.htm');return false;">Norsys team very much welcomes questions and comments about Netica or this onscreen help document. All inquires should be sent to: [support@norsys.com](mailto:support@norsys.com). To do this directly from Netica, choose **Help** → **Email Norsys**.

**Entering findings:** When a Bayes net is applied to a particular situation, or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case.htm');return false;">case, then the known information about that case is entered into the Bayes net by assigning values (called "= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings", or "evidence") to the known variables (i.e. nodes), and that process is known as *entering findings* into the nodes. Entering a finding into a particular node does not = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_retract\_finding.htm');return false;">retract existing findings at that node or other nodes (but for convenience, in Netica

Application, if the new finding for a node flat-out contradicts a previously entered finding for that node, the previous finding will be retracted first).

[More Info](#)

**Equation:** Within Netica, the probabilistic or deterministic relationship between nodes may be expressed using an *equation*, as an alternative to = 4

```
&& typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_CPT.htm');return false;">CPT or = 4 &&
```

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_function_table.htm');return false;">function
```

tables. The equation follows a syntax common in mathematics and computer programming, and may use any of a large set of [special functions](#) built-in to Netica. [More Info](#)

**Ergo:** A program available from Noetic Systems Incorporated which works with Bayes nets. If you have Bayes net files in the old Ergo format (i.e. not .ent files) which you wish to use with Netica, then give them a file extension of ".ergo" and Netica will be able to read them.

**Error rate:** If a classifier is tested on a number of cases, each of known class, one can determine how many times it misclassified a case (i.e. said the case belonged to some class when in fact it belonged to a different one). That, divided by the number of classifications made, is the *error rate* (usually expressed as a percentage). The error rate is only with respect to the probability distribution of the test cases.

**Examples:** After you [install](#) Netica on your computer, within the "Netica" folder will be a folder called "Examples". It contains several Bayes nets suitable for an introduction to the field of Bayes nets, and for use in learning about Netica. Within it is a folder called "Tutorial - View these first" containing an ordered set of Bayes nets that contain some tutorial information and exercises in their net [description](#) window; if you are working through them, you might be interested in reading the [Quick Tour](#) of this onscreen help (if it mentions some file you don't see in the Example folder, don't forget to look in its Tutorial subfolder). For many more examples, be sure to see our great online Bayes net library, available by choosing **Help** → **Net Library Website**.

**Expected value:** The *expected value* (also known as *mean value*) is not the value you “expect” to see, and usually it isn’t even the value most likely to occur. This term, from probability theory, means the average value that will occur, where the average is weighted by the probability of occurrence. For example if the value will be 3 with probability 0.2 and 9 with probability 0.8, then the expected value is:  $(0.2 \times 3) + (0.8 \times 9) = 7.8$ .

**Experience:** For learning and communicating the knowledge contained within a Bayes net, it is useful to indicate a “confidence” in the conditional probabilities it contains, so that one knows how much to change them when new information about them becomes known. That confidence is called the *experience*, and it is calibrated to be equivalent to seeing a certain number of relevant cases. Sometimes this is called the “equivalent sample size”.

[More Info](#)

**Explaining-Away:** If A and B are both possible causes of E, then when we find that E occurred, providing there are no complicating factors, our beliefs that A occurred and B occurred will both increase. But if we further discover that A occurred, then our belief that B occurred will go back down a little. This is called *explaining-away*, because A provides the explanation for E, and there is no need for B to have occurred to explain E. Bayes nets automatically handle explaining away in the correct manner, while evidence-based and rule-based systems are generally poor at it.



## Glossary F - M

**Fading:** When a Bayes net is used in an environment that is changing, it may be desirable to have it adapt to its environment by constantly learning, in a process that weights cases it has seen recently more strongly than those seen in the far past. The process of discounting knowledge learned from the past is called *fading*. [More Info](#)

**Finding:** A *finding* (also known as “evidence”) is a value for one of the nodes (i.e. variables) of a Bayes net when it is applied to a particular situation. [More Info](#)

**Findings menu:** A discrete node’s *findings menu* is obtained by right-clicking on the node. It lists the states of the node, and may be used to enter a finding. [More Info](#)

**Findings node:** A *findings node* is a node which has a finding, or which

we know will receive a finding before doing = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">belief updating. Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_query\_node.htm');return false;">target node.

**Function table:** When the relationship between a node and its parents is deterministic, rather than probabilistic, then instead of a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT a node may have *function table*, in which each row corresponds to a configuration of parent values, and the row provides a single output value for the child node (i.e. the table is a function mapping parent node tuples to child node values). If a function table is converted to a CPT, then each row of the resulting CPT will consist only of zeroes, with a single 1 (or 100%) positioned at the state that was the function table's value for that row. [More Info](#)

**Home folder:** The *Netica home folder* is the folder that the Netica executable "Netica.exe" appears in, as described in [Installation](#). Normally each version of Netica on your system has its own home folder. When Netica is running, you can discover which folder is the home folder by choosing **Help** → **About** and then looking at the last line of the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Messages\_window.htm');return false;">Messages window. Also sometimes called the *Netica home directory*.

**Hugin:** A quality program available from Hugin Expert A/S which works with Bayes nets much the same way = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Netica.htm');return false;">Netica does. Netica can read .net files produced by Hugin versions 4, 5 or 6.

**IDname:** An *IDname* is any word that starts with a simple letter (a-z or A-Z), is composed only of simple letters, digits and underscores (\_), and is 30 or fewer characters long. It must not contain any spaces or punctuation.

**iff:** *iff* means “if and only if”. So the expression “A iff B” can be interpreted to mean “if A then B, and if B then A”.

**Ineffectual link:** An *ineffectual link* is one in which the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT table of its child node contain the same probabilities independent of the state of that link's parent node. In other words, the table contains replicated chunks, with an identical chunk for each state of the parent node.

Ineffectual links may be removed without effecting = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">inference results at all. When a link is first added, it is added as an ineffectual link, and remains that way until the CPT tables of its child node are modified.

Netica can [find and display](#) all the ineffectual links in a net.

**Informationally independent:** A variable X is said to be (informationally) *independent* of another variable Y if obtaining knowledge about the value of X does not change your beliefs about the value of Y. If X is informationally independent of Y, then Y is informationally independent of X. See also = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditionally\_independent.htm');return false;">conditionally independent.

**Informational link:** Any link entering a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

`onclick="BSSCPopup('X_PU_decision_node.htm');return false;">decision node` is known as an *informational link*, and indicates that the decision maker will know the value of the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parent node` when he must make that decision.

**Invertible node:** An *invertible node* is a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_deterministic_node.htm');return false;">deterministic node` whose relationship with its parents is invertible. That is, the value of any one of its parents may be expressed as a function of it and the rest of its parents.

**Joint distribution:** The *joint distribution* provides a probability for every possible outcome (i.e. every element of the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_cartesian_product.htm');return false;">cartesian product` of all the given variables). Since one, and only one, of these outcomes will eventually occur, the joint distribution probabilities add to 1.

**Joint probability:** A *joint probability* is a probability from the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_joint_distribution.htm');return false;">joint distribution`.

**Junction tree:** A *junction tree* (also known as a “join tree”) is the internal structure that Netica uses for = 4 && `typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_belief_updating.htm');return false;">belief updating`. Netica compiles a = 4 && `typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Bayes_net.htm');return false;">Bayes net` or = 4

`&& typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_decision_nets.htm');return false;">decision net  
into a junction tree for efficiency. More Info`

**Latent node:** A *latent node* (also known as a “hidden node”) is a node in a learned Bayes net which does not correspond to any variable in the input data. That is, the variable was created to more easily express the relationships between observed variables.

**Leaf node:** A *leaf node* is a = 4 `&& typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_node.htm');return false;">node with no = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_child_node.htm');return false;">children. See also = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_barren_node.htm');return false;">barren node and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_root_node.htm');return false;">root node.`

**Likelihood finding:** A *likelihood finding* (also known as “virtual evidence”) is a = 4 `&& typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_finding.htm');return false;">finding with some uncertainty attached. Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_negative_finding.htm');return false;">negative finding and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_positive_finding.htm');return false;">positive finding. More Info`

**Link:** A *link* (also known as an “arc” or an “edge”) is a connection between two nodes indicating dependence, and is usually drawn as a line with an arrow at one end.

[More Info](#)

**Link reversal:** *Link reversal* is the process of changing the direction of a link, and then making necessary changes to the rest of the net so that any subsequent inference results will not be affected. However, it may result in less efficient inference (or occasionally more efficient), since extra links may have to be added (removed) to maintain the global relationships between the nodes. [More Info](#)

**Link structure:** The *link structure* (also known as the “topological structure” or “dependency graph”) of a Bayes net or decision net is just the graph structure of the net. In other words, it consists only of the node names and links, but not of any other information about the nodes or the particular relationships the nodes have with their parents.

**Loop cutset:** A *loop cutset* for a net is a set of nodes such that if they are all removed from the net (along with all links involving them) there will be no loops in the net. See also [cutset](#).

**Markov blanket:** A set of nodes B is a *Markov blanket* of node X, if given findings for all the nodes of B, X is independent of all other nodes in the net. If it is the minimal such set, then it is called the Markov boundary (see below).

**Markov boundary:** The *Markov boundary* of a node X is the minimal set of nodes for which any set of positive

findings can be supplied and X will be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditionally\_independent.htm');return false;">conditionally independent of all other nodes in the net. For a Bayes net, this is X's parents, X's children and X's childrens' other parents.

A set of nodes Y can also have a Markov boundary. If positive findings are obtained for the Markov boundary nodes, then findings for the Y nodes will not provide any additional information about any other node in the net and vice-versa.

Netica can find and display the Markov boundary of a node or set of nodes.

**Markov network:** A *Markov network* is an = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_undirected\_network.htm');return false;">undirected network in which the nodes represent variables of interest and the connections between them represent probabilistic dependence. Netica converts a Bayes net into a Markov network as an intermediate step in producing a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_junction\_tree.htm');return false;">junction tree.  
[More Info](#)

**Mean value:** In calculus, the *mean value* theorem states, roughly, that given a section of a smooth curve, there is a point on that section at which the derivative (slope) of the curve is equal (parallel) to the "average" derivative of the section. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_mean\_value.htm');return false;">More Info

**Messages window:** The *Messages window* is a window that Netica uses to communicate textual information to you. You can open it any time using **Window** → **Messages**, and you can copy and paste information between the

Messages window and any text file. Sometimes if you make a minor mistake, Netica just beeps and places a message there. [More Info](#)

**Missing data:** If some cases have values for a certain variable, and others do not, that is known as *missing data*. [More Info](#)

**Moral graph:** A net in which each node has all its parents connected to every other of its parents (i.e. there are no “illegitimate” children). *Moralizing* a net consists of adding the required connections between parents of a node for all the nodes (i.e. “marrying” them).

**Most Probable Explanation (MPE):** The *most probable explanation*, or MPE (also known as “maximum a-posteriori probability” or “MAP”) is a set of values (one for each node) that is the most probable configuration, given the net’s current findings. Since there may be several configurations with the same probability, the MPE may not be unique. [More Info](#)

**Multiply-connected:** A *multiply-connected* network is a directed or undirected network which has more than one = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_path.htm');return false;">undirected path between some pairs of nodes. In other words, it has = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_undirected\_loop.htm');return false;">loops. [More Info](#)

**Multipurpose selector:** When you click on the *multipurpose selector*,  of a node dialog box, you will be presented with a menu of node properties that can be edited in the box below the selector. ([More Info](#))



## Glossary N - R

**Nature node:** A *nature node* in a Bayes net represents some variable of interest. It may also appear in a decision net in which case it is a variable that cannot be directly controlled by the decision maker (i.e. it is determined by nature). If a nature node has a functional relationship with its parents, it is called a *deterministic node*, whereas if the relationship is probabilistic, it is called a *chance node*. The characteristic shape for a nature node is an ellipse, or a rectangle with rounded corners.

**Negative finding:** A *negative finding* is a finding that some node is definitely **not** in some particular state. Compare with [positive finding](#) and [likelihood finding](#). [More Info](#)

**Net:** In Netica documentation, the word *net* is used to mean a Bayes net or a decision net.

**Netica:** *Netica* is a program created by Norsys for working

with = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes nets and  
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision nets.  
[More Info](#)

**Netica API:** *Netica API* (also known as “Netica Programmer’s Library”) is software that you can link with your own programs to achieve much of the functionality of = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_Netica\_Application.htm');return false;">Netica Application. It is created by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_Norsys.htm');return false;">Norsys and is designed for working with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes nets and = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision nets.  
[More Info](#)

**Netica Application:** *Netica Application* is the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_Netica.htm');return false;">Netica product with an easy-to-use graphical interface for building and working with Bayes nets and = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision nets.  
To program Netica, use [Netica API](#) instead.

**Netica-Web:** Netica-Web is a system to deploy your Bayes nets over the internet as a question-answer system. Such a system asks the user questions, or

provides a dashboard to enter relevant information, and presents the user with conclusions. [More info](#)

**No-forgetting links:** If a decision maker remembers the decisions he made at an earlier time, and also the knowledge he had available to him at that time, then in his decision net there will be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_informational\_link.htm');return false;">informational links going from earlier decision nodes and their parents, to later decision nodes. These are called *no-forgetting links*. [More Info](#)

**Node:** A *node* is a component of a Bayes net or decision net used to represent a variable (i.e. scalar quantity) of interest, and in Netica is usually drawn as a rectangle, rounded rectangle, circle or flattened hexagon. [More Info](#)

**Node dialog box:** To change or view the properties of a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node.htm');return false;">node, such as its name or the states it has, you use a *node dialog box*, which you obtain by double-clicking on the node, or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_select\_node.htm');return false;">selecting it and then pressing the **ENTER** key. To change its relation with its parent nodes, you use a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_table\_dialog\_box.htm');return false;">table dialog box. [More Info](#)

**Current state:** The *current state* of a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_dialog\_box.htm');return false;">node dialog box, is the state displayed next to the label “State:” It may be changed by using the popup menu next to the “States:” label.

The state interval thresholds or = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_state\_value.htm');return false;">state value  
displayed is for the current state. [More Info](#)

**node name:** The node name text edit box in the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_node\_dialog\_box.htm');return  
false;">node dialog box looks like = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_graphic\_node\_name.htm');return  
false;">this.

**node title:** The node title text box in the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_node\_dialog\_box.htm');return  
false;">node dialog box looks like = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_graphic\_node\_title.htm');return  
false;">this

**states label:** The **states** label in the = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_node\_dialog\_box.htm');return  
false;">node dialog box looks like = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_graphic\_states.htm');return false;">this.

**Node relationship:** A *node relationship*, or *node relation* for short, is the

relationship between a node and its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents. It may provide the value of the node as a function of its parents' values, or it may provide a probability distribution for the node depending on its parents' values. It is often expressed as a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT in which case it can be viewed or edited using the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_table\_dialog\_box.htm');return false;">table dialog box. Alternately, it may be expressed as a probabilistic or deterministic = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_equation.htm');return false;">equation.

**Non-extreme probability:** A *non-extreme probability* distribution (also known as “strictly positive”) is one where the probability is never 0. That means that it is also never 1, and that it has no points of complete “certainty”.

**Normal distribution:** The *normal distribution* (also known as “Gaussian distribution), is the most commonly used continuous distribution with infinite = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_support.htm');return false;">support. [More Info](#)

**Normative theory:** A *normative theory* does not indicate what agents usually do (which is a *descriptive theory*) or what agents ought to do (which is a *prescriptive theory*), but what agents **must** do if they wish to act optimally in a given situation, where optimally is defined in a particular way with respect to the situation.

**Norsys:** *Norsys Software Corp.* is the company which develops = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_Netica\_Application.htm');return false;">Netica Application and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Netica\_API.htm');return false;">Netica API.

You can get more information about Norsys from their web site at: [www.norsys.com](http://www.norsys.com). Questions and comments are very welcome, and may be sent by [email](#).

**Optimal policy:** The *optimal policy* (also known as the set of *optimal decisions*) is the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_policy.htm');return false;">policy which results in the greatest = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_expected\_value.htm');return false;">expected value for the sum of the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_utility\_node.htm');return false;">utility nodes (or one of those policies if there are more than one which result in the same expected utility). Finding the optimal policy is sometimes called “solving” a decision net.

**Outcome:** The *outcome* is the result of an event, or series of events, that could have turned out in one of several ways.

**Parameter learning:** *Parameter learning* is the automatic learning of the specific relationships nodes have with their = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents using case data, once it has already been determined which nodes are the parents of each node. These relationships are usually in the form of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditional\_probability.htm');return false;">conditional probabilities, or the parameters of a conditional probability

equation. Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_structure\_learning.htm');return false;">structure learning. [More Info](#)

**Parent node:** If there is a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link.htm');return false;">link going from = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node.htm');return false;">node A to node B, then A is said to be a *parent node* of B. Some people refer to it as a “direct predecessor”.

**Path:** A *path* is a sequence of nodes from a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_net.htm');return false;">net, such that you can get from one node of the sequence to the next node by traversing a link between them (but not necessarily in the direction of the arrow). Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_directed\_path.htm');return false;">directed path.

**Poisson process:** A *Poisson process* is one in which events occur randomly and independent of each other. The number of events that occur in a fixed time period is given by the [Poisson distribution](#), the time between successive events is given by the [exponential distribution](#), and the time required for the occurrence of a fixed number of events is given by the [gamma distribution](#).

**Policy:** A *policy* (also known as a “control law”) is a set of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_rule.htm');return false;">decision rules, with one for each = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_decision\_node.htm');return false;">decision node of a decision net. When Netica “optimizes decisions” it finds the policy which maximizes the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_expected\_value.htm');return false;">expected value of utility.

**Positive finding:** A *positive finding* is a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">finding that some node is definitely in some particular state. Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_negative\_finding.htm');return false;">negative finding and = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_likelihood\_finding.htm');return false;">likelihood finding. [More Info](#)

**Probabilistic inference:** *Probabilistic inference* is the process of calculating new = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief.htm');return false;">beliefs for a set of variables, given some = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">findings. Technically speaking, it is the process of finding a posterior distribution, given a prior distribution, a model and some observations. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bayes\_net.htm');return false;">Bayes nets do probabilistic inference by = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief Updating.htm');return false;">belief updating.

**Probability density function:** The *probability density function* (also known as “pdf”), is a function that provides the probability of a continuous probability distribution at each point within the distribution. It may be integrated over a region to determine the probability of that region. It is nowhere negative and its integral over the whole distribution is always 1. The integral of the pdf from negative infinity to  $x$  is known as the *cumulative density function* (cdf).

**Probability revision:** *Probability revision* is the process of adjusting the = 4

`typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_CPT.htm');return false;">conditional probability`

tables of a Bayes net to account for a new = 4

`typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_case.htm');return false;">case (i.e. set of  
findings), or more often, for a new set of cases. It is a form of = 4`

`typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_parameter_learning.htm');return  
false;">parameter learning, which generally involves learning from cases.`

Compare with = 4 `typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_belief Updating.htm');return false;">belief  
updating.`

**Prospect:** A *prospect* is the probability distribution over possible = 4

`typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_outcome.htm');return false;">outcomes, given a`

= 4 `typeof(BSPSPopupOnMouseOver) == 'function')`

`BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_policy.htm');return false;">policy and some = 4`

`typeof(BSPSPopupOnMouseOver) == 'function')`

`BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_finding.htm');return false;">findings.`

**Query node:** See [Target Node](#).

**Relation symbol:** The *relation symbol* is a green tree structure which looks like this:  It is used on toolbar buttons to indicate the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node\_relation.htm');return false;">relation a node has with its parents. For example, the  button is to view or edit a table, the  button removes a node's table, and the  button with a dice randomizes a table.

**Reports:** Netica can generate a multitude of text *reports*, useful in understanding the information in your net. These reports include:

**Report** → **Beliefs**, which lists the current beliefs (i.e. posterior probabilities) for nature nodes, and expected utilities for decision nodes. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_report\_beliefs.htm');return false;">EXAMPLE

**Report** → **CPT Tables**, which lists the node relation as a conditional probability table (= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT) or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_function\_table.htm');return false;">function table. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_report\_CPT.htm');return false;">EXAMPLE

**Report** → **Elimination**, which lists the order used during compiling. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_report\_elimination.htm');return false;">EXAMPLE

**Report** → **Equations**, which lists all = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_node_equation.htm');return`  
`false;">equations of nodes. Choosing the Horizontal Format option`  
`prints them in internal form. = 4 && typeof(BSPSPopupOnMouseOver)`  
`== 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_report_equations.htm');return`  
`false;">EXAMPLE`

**Report** → **Excel**, which hot-links to the node beliefs. = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_report_excel.htm');return false;">EXAMPLE`

**Report** → **Findings**, which lists the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_finding.htm');return false;">findings (i.e.`  
`"case" or "evidence") currently entered, including likelihood ("virtual")`  
`findings. = 4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_report_findings.htm');return`  
`false;">EXAMPLE`

**Report** → **Junction Tree**, gives details of the net compilation process. = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_report_junctiontree.htm');return`  
`false;">EXAMPLE`

**Report** → **List Selected**, generates a list of the names of the nodes currently selected. = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_report_listselected.htm');return`  
`false;">EXAMPLE`

**Report** → **Node Sets**, which lists the nodes within the requested set. = 4 && `typeof(BSPSPopupOnMouseOver) == 'function')`

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_report_nodesets.htm');return
false;">EXAMPLE
```

**Report** → **Network**, which gives a summary information on the whole net. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_report\_whole.htm');return false;">EXAMPLE

**Retracted:** Anytime after a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">finding has been = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entered into a node, that finding may be removed, or *retracted*. After doing belief updating for the net, it will be as if the finding had never been entered. [More Info](#)

**Right-click:** To *right-click* on something, place the mouse pointer ("cursor") over it, then press the right mouse button and choose an item from the menu that comes up. [More Info](#)

**Root node:** A *root node* is a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node.htm');return false;">node with no = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_parent\_node.htm');return false;">parents. See also = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_leaf\_node.htm');return false;">leaf node.



## Glossary S - Z

**Select link:** In order to do some operations on a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link.htm');return false;">link, you first *select* it by clicking once on it, and it will then be drawn using negative colors to hilite it. To select more than one link, or remove links from your selection, hold down the `CTRL` key while clicking on them. [More Info](#)

**Select node:** In order to do some operations on a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_node.htm');return false;">node, you first *select* it by clicking once on it, and it will then be drawn using negative colors to hilite it. You can select several nodes at a time by clicking on the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_background.htm');return false;">background and dragging the selection rectangle to include them. To add or remove nodes from your selection, hold down the `CTRL` key while you do the selection operation, or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_right\_click.htm');return false;">right-click on the node and choose **Select/Deselect**. [More Info](#)

**Singly-connected:** A *singly-connected* network is a directed or undirected network that has at most one = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_path.htm');return false;">undirected path between any two nodes. In other words, it has no = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_undirected\_loop.htm');return false;">loops. [More Info](#)

**Standard deviation:** The *standard deviation* is the square root of the variance. Its units are the same as those of X, and it is a measure of how “spread out” or “imprecise” the distribution is. [More Info](#)

**State level:** When = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretizing a continuous variable (i.e. node), a set of numeric borders is used to partition the range of the variable into intervals called states. Those borders are called *thresholds*. Each state of the discretized variable has a lower threshold and an upper threshold, with the upper threshold being the same as the lower threshold of the next state. If reference is made to a state's threshold without specifying lower or upper, then lower is implied, and it is also sometimes referred to as the *state's level*.

**State value:** Any = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discrete\_node.htm');return false;">discrete node may have associated with each of its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state.htm');return false;">states a number (integer or real) called its *state value*, which may be used to identify that state in databases or case files, or may be used to provide an "output value" for that state in equations. = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_continuous.htm');return false;">Continuous nodes don't have state values, but if they have been = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_discretize.htm');return false;">discretized, they have state = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_state\_threshold.htm');return false;">thresholds instead, and sometimes the term *state level* is used to refer to numbers that are

state values or thresholds. [More Info](#)

**States:** A = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_discrete.htm');return false;">discrete variable  
can take on one of several values, and these values are called *states*. For  
example the states may be “female, male”, or they might be “US, Europe,  
Japan, China”, or “True, False”. With Netica you can just let the states of a  
node be numbered, but usually you [give](#) them meaningful names.

**Structure learning:** *Structure learning* is the automatic discovery of the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_link\_structure.htm');return false;">link structure  
of a Bayes net from case data. Compare with = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_parameter\_learning.htm');return  
false;">parameter learning.

**Support:** The *support* of a probability distribution is the range of its  
variable(s) over which the = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_pdf.htm');return false;">pdf is non zero. In  
other words, those values of the variable which are possible.

**Tab delimited text file:** *Tab delimited text file* is a commonly used term for a  
form of [case file](#) in which the names of the variables appear on the first line,  
and then below are all the cases (i.e. records), with each case on a single line  
and having a value for each of the variables, and with all the values and  
variables in text form and separated by tab characters. See also = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_CSV\_file.htm');return false;">CSV file.

**Table dialog box:** To change or view the relationship of a = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_node.htm');return false;">node with its = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parents that is,
edit its conditional
```

probability table (CPT) or = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_function\_table.htm');return false;">function table, you use a *table dialog box*. To obtain = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_select_node.htm');return false;">select the
node, and then choose Table → View/Edit or click the  toolbar button. It is
not to change the node's basic properties; for that, use a = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_node_dialog_box.htm');return false;">node
dialog box instead. More Info
```

**Table selector:** Within the table dialog box, you will find a table selector, which looks like = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_table\_selector.htm');return false;">this, and which provides a menu for you to select the type of table that you view.

**Target Node:** A *target node* is a node whose beliefs we want to know after belief updating. Also known as a "query node". Compare with = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_findings\_node.htm');return false;">findings node.

**Test sensitivity:** The *test sensitivity* of an imperfect test is the percentage of cases testing positive out of those that should test positive. See also = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_test\_specificity.htm');return false;">test specificity.

**Test specificity:** The *test specificity* of an imperfect test is the percentage of cases testing negative out of those that should test negative. See also = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_test_sensitivity.htm');return false;">test
sensitivity.
```

**Text editor:** A *text editor* is a program that allows you to modify plain ASCII

```
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_text_file.htm');return false;">text. For small
files you can use Netica's File → New → Text Edit or File → Open as Text.
For larger files you can use a word processing program such as MS Word or
WordPad (save as "Text Only" if you wish to make a file).
```

**Text file:** A *text file* consists only of = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_ASCII.htm');return false;">ASCII characters. It
has no special symbols, no formatting (bold, italics, fonts or sizes), no
structure (paragraph sections, chapter sections, margins, etc.), and no special
inserts (pictures, tables, etc.). You can create or modify a text file using a = 4
```

**Time-delay link:** A *time-delay link* is a link which indicates that the child node is for a variable's value at a point in time later than the value of the = 4

```
&& typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_parent_node.htm');return false;">parent node on
which it depends. The time difference is something that you can set for the
link. A net that contains time-delay links is not a normal Bayes net, but rather
```

a meta-level representation, but it can be expanded to produce a normal Bayes net. Time delay links can be used to model feedback. [More Info](#)

**Toggled:** A menu item that can be *toggled* is like a switch that can be turned on and off. When you choose it, a checkmark will appear beside it, indicating it is **on**. Choosing it again will remove the checkmark, indicating it is **off**.

**Triangulated:** An `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_undirected_network.htm');return false;">`undirected network is *triangulated* if every loop of length 4 or more has a chord, i.e. a link joining two nonconsecutive nodes. Triangulating a net is the process of adding links until it is triangulated (which may be done in different ways, depending on where the links are added). The resulting net is composed only of triangles (each side being a link) fused together along their sides.

**Undirected loop:** An *undirected loop* (sometimes just called a “loop”) is a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_path.htm');return false;">`path through a net, not necessarily following the direction of the arrows, which returns to its beginning (i.e. the first node of the path is the same as the last). Compare with `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_directed_cycle.htm');return false;">`directed cycle.

**Undirected network:** An *undirected network* is one where the links have no direction (i.e. no arrows). A `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Markov_network.htm');return false;">`Markov network is an example. Compare with a `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_directed_network.htm');return false;">`directed

network.

**Unicode:** *Unicode* is a text character set that was designed with the goal of being able to represent all the characters of all the world's languages. It is by far the dominant international character set, and was developed by a consortium with most of the largest computer companies as members. The first version was released in 1991, and about every year a new version is released having more characters. It normally represents each character with 2 bytes, although there are a number of complexities and alternate encodings.

Netica allows some aspects of Bayes nets to be represented in Unicode, while others are required to be in = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_ASCII.htm');return false;">ASCII.

**User reports:** a Netica mechanism which displays customized information pertaining to a node, group of nodes, or to an entire net. The *user report* could be as simple as a text message giving a more detailed description of what a node means. Or it could be more complex, such as the current belief probabilities of the nodes, or a sensitivity analysis of the net. User reports can also be generated through the HED system.

**Utility node:** A *utility node* (also known as a “value node”) is a node in a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision net whose = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_expected\_value.htm');return false;">expected value is to be maximized while searching for the best = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_decision\_rule.htm');return false;">decision rule for each of the decision nodes. It is usually drawn as a flattened hexagon or a diamond.

**Weak link:** The absence of a link always indicates some = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_info_indep.htm');return false;">independence,
but even if a link is present there may nearly be independence if the = 4 &&
typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_node_relation.htm');return false;">relationship
```

that the child node has with its parents indicates that. Then it is called a *weak link*. Removing a weak link has little effect on the full joint probability distribution, and therefore little effect on inference results.

**Whitespace:** *Whitespace* is text which is composed solely of spaces, tabs, new lines, carriage returns or other [ascii](#) control characters.



# Encyclopedia

[Axioms of Probability Theory](#)

[Bayes Theorem](#)

[Cartesian Product](#)

[Conditional Probability](#)

[Connectivity](#)

[d-separation](#)

[Discrete vs. Continuous](#)

[Fundamental Theorems of Probability](#)

[Game Theory](#)

[Junction Tree](#)

[Link](#)

[Markov Network](#)

[Messages Window](#)

[Node](#)

[Password](#)

[Probabilistic Inference by Node Absorption](#)

[Standard Deviation](#)

## **Axioms of Probability Theory**

Here is a set of axioms for probability theory equivalent to those derived by Cox (and also compatible with other axiomatizations of probability, such as the Kolmogorov axioms):

1.  $P(a | a) = 1$
2.  $P(\neg a | b) = 1 - P(a | b)$
3.  $P(a, b | c) = P(a | b, c)P(b | c)$

where  $P(x, y | z)$  means the probability that propositions  $x$  and  $y$  are both true, given that proposition  $z$  is true.

## Bayes Theorem

For any two propositions, A and B,  $P(B|A) = P(A|B) \times p(B) / p(A)$ , where you read 'PP(A)' as "the probability of A", and 'P(A|B)' as "the probability of A given that B has occurred".

$$P(B|A) = P(A|B) \frac{P(B)}{P(A)}$$

$$P(A | B) = \frac{P(AB)}{P(B)} \quad \text{Definition of conditional probability}$$

$$P(B | A) = \frac{P(AB)}{P(A)} \quad \text{Same definition with B and A interchanged}$$

$$P(AB) = P(B | A)P(A) = P(A | B)P(B)$$

As shown, Bayes rule follows directly from the definition of conditional probability. This theorem is a way to translate between the probability of causes, given effects and the probability of effects given causes.

## Cartesian Product

The *cartesian product* of two sets is the set of all possible pairs of elements, where the first element of each pair is taken from the first set, and the second element is taken from the second set. For example, the cartesian product of {low, medium, high} with {true, false} is {(low, true), (low, false), (medium, true), (medium, false), (high, true), (high, false)}

The cartesian product is denoted using 'x'. It may be extended to more than two sets, as shown in this second example:

$$\{s1, s2\} \times \{s, h\} \times \{wtg, mhc\} = \{(s1, s, wtg), (s1, s, mhc), (s1, h, wtg), (s1, h, mhc), (s2, s, wtg), (s2, s, mhc), (s2, h, wtg), (s2, h, mhc)\}$$

The number of elements in the cartesian product is the multiplicative product of the sizes of the sets involved. In the first example it is  $3 * 2 = 6$ , and in the second example it is  $2 * 2 * 2 = 8$ .

Some people spell it with a capital C ("Cartesian product"), since it is named after Descartes.

## Conditional Probability

The probability that some specified variables (called the *hypothesis* variables) take specified values, given that some other specified variables (called the *given* variables) have specified values. There may be any number of given variables, including none (in which case its also called a *prior* probability). If the values of the given variables are inconsistent (i.e. their probability of occurring together is 0), then the conditional probability is undefined.

## Connectivity

The “connectivity” of a net refers to a net’s = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link\_structure.htm');return false;">link structure. High connectivity means that there are a lot of links.

You can use the link structure of a net to determine independence between nodes, by using the *d-separation* algorithm.

A **Multiply-Connected net** is a net which has more than one = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_path.htm');return false;">undirected path between some pairs of nodes. In other words, it has = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_undirected\_loop.htm');return false;">loops.

A **Singly-Connected net** is a net that has at most one path between any two nodes. In other words, it has no loops. If the link directions are ignored, it is always possible to consider it as a tree. If the link directions are not ignored, then some people call it a “polytree” (because it can be considered to be several directed trees, with arrows from root to leaves, fused together at leaf nodes).

The time it takes for Netica to do = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_belief\_updating.htm');return false;">belief updating is very dependent on the connectivity of the net. Adding a link always results in longer updating times, but where that link is added can make a big difference. If a node has many parents, then its = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_conditional\_probability.htm');return false;">conditional probability table will be very large, which results in slower overall updating times. The number of loops a net has is also very significant. If a net has no loops, then Netica can do updating extremely fast, and each

added loop will increase the updating time.

Netica implements a number of graph algorithms to make various connectivity results available to you, such as finding all parents, children, ancestors, descendents, connected, d-separated, Markov boundary, interconnecting links, cycles, etc. [More Info](#)

## d-separation

The *d-separation* rule enables you to quickly determine whether a = 4 &&

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_finding.htm');return false;">finding at one node
```

can possibly change the = 4 && typeof(BSPSPopupOnMouseOver) ==

```
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_belief.htm');return false;">beliefs at another by
```

only looking at the = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_link_structure.htm');return false;">link structure
```

of a Bayes net.

Netica uses d-separation internally, and can also find and display d-separated nodes.

## Discrete vs. Continuous

A discrete variable is one with a well defined finite set of possible values, called `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_state.htm');return false;">states`. Examples are: the number of dimes in a purse, a statement which is either “true” or “false”, which party will win the election, the country of origin, voltage output of a digital device, and the place a roulette wheel stops.

A continuous variable is one which can take on a value between any other two values, such as: indoor temperature, time spent waiting, water consumed, color wavelength, and direction of travel. A discrete variable corresponds to a digital quantity, while a continuous variable corresponds to an analog quantity.

With Netica you can [choose](#) whether you want a node to represent a discrete or continuous variable.

Often a variable will be continuous at one scale, but discrete on another. For instance the amount of water consumed might be discrete if you count individual water molecules, but it is continuous at the scale you are concerned with. Likewise, the voltage output of a digital device might be discrete at the scale you are concerned with (“high” & “low”), but continuous on a finer scale (0.7 - 3.5V), and then discrete on a very fine scale (corresponding to the number of electrons on a capacitor). You only need consider the scale of interest when setting whether a node is continuous or discrete.

Sometimes you want a continuous variable to behave like a discrete one. To do this, you break up the total range of the continuous variable into a number of intervals by supplying numbers showing where one interval ends and the next begins. This is known as *discretizing* the variable, and the numbers are called *thresholds*. Each interval corresponds to one state of the discrete version of the variable. A discretized variable is sometimes known as an *interval variable*, since its domain is composed of intervals. In Netica, a single node represents both the continuous and discrete versions of the variable, and Netica will convert a value of the continuous variable into a discrete state when appropriate. Netica allows you to [discretize](#) any continuous node, by choosing **Modify** → **Discretize Nodes**. (If the variable is `= 4 && typeof(BSPSPopupOnMouseOver) == 'function')`

[BSPSPopupOnMouseOver\(event\);" class="BSSCPopup" onclick="BSSCPopup\('X\\_PU\\_discrete.htm'\);return false;">discrete](#) then the menu won't have a "Discretization" choice).

Conversely, a discrete node may have numeric quantities attached to each state, so that each state can represent a number, but the variable is incapable of representing numbers between those of each state ([more info](#)).

See Also: [Node Discretization - Multi-Purpose Box](#)

See Also: [Node State Interval](#)

## Fundamental Theorems of Probability

*Bayes Theorem:*

$$P(a | b, c) = P(b | a, c) \frac{P(a | c)}{P(b | c)}$$

*Reasoning by cases theorem:*

$$P(a | c) = P(a | b, c)P(b | c) + P(a | \neg b, c)P(\neg b | c)$$

*Independence theorem:*

$$P(a, b | c) = P(a | c)P(b | c) \quad \text{iff } A \text{ is independent of } B \text{ given } C$$

*Where independence is defined as:*

$$P(a | b, c) = P(a | c) \quad \text{iff } A \text{ is independent of } B \text{ given } C, \text{ providing } b \text{ and } c \text{ are consistent}$$

In each of the above theorems, all the probabilities are conditioned on the proposition. Since it could be equivalent to any logical formula, they all hold with  $c$  replaced by any vector of truth values,  $c$ .

## Game Theory

Game theory usually attempts to determine what will happen when two or more agents cooperate/compete in trying to maximize their own expected utility.

Decision theory prescribes how one agent should best make decisions to maximize his expected utility, given his beliefs about the environment and how other agents will act.

If you specify the behavior (as a probabilistic policy) of the other players involved in a game, then you can use decision theory to determine how one player should best act. But it won't determine the policies for all players simultaneously like game theory sometimes will. Currently Netica only solves decision theory problems. It can still be a valuable tool for investigating game theory problems, but only from one player's point of view (rather than a global point of view).

If you want to use Netica to formulate a game, then make a decision node for a single player and a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_nature\_node.htm');return false;">nature node for each of the other players. Use the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_table\_dialog\_box.htm');return false;">table dialog box to enter a policy for each of the other players. There will be = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_link.htm');return false;">links between these nodes only if some players will know what some other players actions are before making their own decision.

Then add a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_utility\_node.htm');return false;">utility node, with links from each of the other nodes to it. You can fill the utility node table quite easily from the game theory payoff matrix (it will have one row for each cell of the matrix) by keeping in mind that you are only entering the payoff for

the single player you are analyzing. Supposedly the other payoffs were used to determine the policies for the other players, perhaps using another decision net.

You may find that you want to iterate solutions, since developing each player's decision net depends on the policies of other players, which you don't know until you have solved their decision nets, which require the policy (solution) of the first net. You may find `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_Netica_API.htm');return false;">Netica API` useful to write a program for such iterations (or to explore how strategies and policies of a population evolve over time).

## Junction Tree

Also known as a “join tree”.

A *junction tree* is the internal structure that Netica uses for

`typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_belief_updating.htm');return false;">belief  
updating. Netica compiles a typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_Bayes_net.htm');return false;">Bayes net or  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_decision_nets.htm');return false;">decision net  
into a junction tree for efficiency.`

The junction tree  $T$  of `typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_triangulated.htm');return false;">triangulated  
net  $G$  is a tree with the typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_clique.htm');return false;">cliques of  $G$  as  
nodes, such that for every node  $N$  of  $G$ , if we remove from  $T$  all cliques not  
containing  $N$ , the remaining subtree remains connected. In other words, any  
two cliques containing  $N$  are either adjacent in  $T$  or connected by a path made  
entirely of cliques that contain  $N$ .`

You can examine the junction tree that Netica creates for a Bayes net by  
making a [text report](#) using **Report** → **Junction Tree** after compiling.

## Link

The links of a Bayes net indicate which nodes (i.e. variables) are directly dependent on others. More accurately, they indicate which nodes are

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_conditionally_independent.htm');return
```

```
false;">independent of which others. If there is no link from one node to another, then their corresponding variables are conditionally independent
```

given values for their parents. Using the *d-separation* rule, you can determine which nodes effect which others during probabilistic inference, based on the

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_link_structure.htm');return false;">link structure
```

```
link of the net.
```

A link between two nodes is indicated as a line between them with an arrow at one end, there are several ways to add links between nodes.

A link into a decision node is known as an

```
typeof(BSPSPopupOnMouseOver) == 'function')
```

```
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
```

```
onclick="BSSCPopup('X_PU_informational_link.htm');return
```

```
false;">informational link, and indicates that the decision maker will know the values of the parent nodes when a decision must be made.
```

Some people refer to links as “arcs” or “edges”.

## Markov Network

A *Markov network* is an undirected net in which the nodes represent variables of interest and the connections between them represent probabilistic dependence. Netica converts a Bayes net into a Markov network as an intermediate step in producing a junction tree.

## Messages window

When you first start Netica, in the lower left corner of the workspace which opens there will be an icon for a minimized window called “Netica Messages”.

You can double click it to view its contents or you can choose **Window** → **Messages**. To return it to its minimized state click on the  button in its title bar.

**Note:** if you are using [Netica on a Mac](#), the Messages window may appear as either a blank white square in the bottom left of the Netica window, or as just a word in the upper left of the screen. Simply click on the square or word to launch the Messages window.

Netica reports textual information to you through this window. If you do **Report** → **Network**, then the report will be placed in the Messages window,

and after compiling a net, information on the `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_junction_tree.htm');return false;">junction tree` will be in the Messages window. Progress during lengthy operations is sometimes displayed in the Messages window, although it has to be visible before the operation is started.

Usually when you try to do an operation that Netica can't perform, it will display a dialog box telling you. However, for some minor operations, like clicking in the wrong place, Netica will just beep. In that case, you can look in the Netica Messages window and there will be an explanation.

You can copy and paste information between the Messages window and any text file. In addition, all information going to the messages window can also be sent to a text file by choosing **File** → **Log Messages**.

## Node

There are three kind of nodes indicating what our intention is for a variable: =

```
4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_nature_node.htm');return false;">nature nodes,
= 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_decision_node.htm');return false;">decision
nodes, and = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_utility_node.htm');return false;">utility nodes.
```

A nature node is sometimes called a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_chance\_node.htm');return false;">chance node  
when its relationship with its parents is probabilistic, and a = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_deterministic\_node.htm');return  
false;">deterministic node when it isn't. Nature nodes may also be called = 4  
&& typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_query\_node.htm');return false;">target nodes if  
we are interested in their beliefs after belief updating, or = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_findings\_node.htm');return false;">findings  
nodes if we have, or will have, a finding for them.

## Password

The password you were issued after purchasing a Netica product contains 5 components:

your name, organization, product number, version number and a security number , in the following form:

+Name/Organization/Product-Version-License/Security

The product will be either Netica Application (120), Netica API (310), or both (120,310). The version number will be an integer, and your license will be one or more of: commercial (no entry), academic (A), site (S), etc.

Here are some sample Netica license passwords:

+MyName/ABCCorp/120,310-2/12345

^ the version number is 2.

+MyName/DEFUniv/120-1-AS/54321

^ the version number is 1.

You will need to enter a password as part of the [installation](#) process.

## Standard Deviation

The *variance* of a variable  $X$  with probability distribution  $p(x)$  is given by:

$$V = \sum_x (x - \mu)^2 p(x) \quad \text{if } x \text{ is discrete}$$
$$V = \int_{-\infty}^{\infty} (x - \mu)^2 p(x) dx \quad \text{if } x \text{ is continuous}$$

where  $\mu$  is the mean value.

The variance of a probability distribution isn't exactly the same as "sample variance", which is the variance of a set of measurements.

The *standard deviation* is the square root of the variance. Its units are the same as those of  $X$ , and it is a measure of how "spread out" or "imprecise" the distribution is.

For any distribution, the mean (i.e., the expected value) and the median can never differ from each other by more than one standard deviation.

For a normal distribution, 31.8% of the probability is outside of 1 standard deviation from the mean, 4.6% is outside of 2 standard deviations, and 0.26% is outside of 3 standard deviations. For any distribution, less than  $1/(k^2)$  of the probability is outside of  $k$  standard deviations (Chebyshev's theorem).

## Netica API

The Netica API is a complete library of functions for working with Bayes nets and influence diagrams that you can call from your own program. It contains functions to build, learn, modify, transform, save and read nets, as well as a powerful inference engine. There are versions of Netica API available for many different programming languages, such as Java, C, C++, C#, Visual Basic, etc. For more information on the COM version of Netica API (mostly for C# and Visual Basic), see [Netica COM Interface](#), and for other versions see [www.norsys.com/netica\\_api.html](http://www.norsys.com/netica_api.html)

An exciting new addition to Netica API allows it to take advantage of the Netica Application user interface for displaying nets, entering findings, showing beliefs, etc. [More Info](#).

Programs that use the Netica API completely control it. For example, Netica functions will not take any action until called, Netica will not do any I/O unless requested to, and its functions will not take an unpredictable amount of time before returning.

It may be used in conjunction with other C or C++ libraries (and it won't interfere with them), but it doesn't require any other library except the Standard C library. Versions of the Netica API are available for MS Windows, Linux, Sun Sparc, Silicon Graphics, Macintosh and DOS, and each of these has an identical interface, so you can move your code between these platforms without changing anything to do with the Netica API.

Before releasing any new version of Netica API, it is put through a rigorous testing program to make sure it operates as designed. Every function is exercised in multiple ways, and its behavior carefully checked. Thousands of random nets are generated and solved in multiple ways to check the inference results, and hundreds of real nets are tested. MemCheck™ and Purify™ are used to make sure there are no memory leaks or other memory faults. In combination with a careful initial design, and base software having ten years of extensive customer usage, this results in a rock solid product.

The Netica API has been designed to be easily extended in the future without changing what already exists. Many new features are currently [under development](#), and it will continue to be extended for years to come.

## Features:

- **Dynamic Construction:** Can build and modify nets "on the fly" in memory (to support working with dynamic Bayes nets), and can save/read them to file.
- **Equations:** Probability tables may be conveniently expressed by equations, using a Java/C type syntax and taking advantage of an extensive library of built-in functions, including all the standard math functions and common probability distributions, as well as some functions and distributions specially suited to Bayes nets, such as noisy-or, noisy-max, noisy-sum, etc.
- **Learning from Data:** Probability tables can be learned from case data, even while the net is being used for probabilistic inference. Learning from data can be combined with manual construction of tables and representation by equations. It can handle missing data and latent variables or hidden nodes. Learning algorithms include: counting, sequential updating, fractional updating, EM (expectation maximization), and gradient descent.
- **Database Connectivity:** Allows direct connection to most database software.
- **Threadsafe:** Can be used safely in multi-threaded environments.
- **Encryption:** Can save and read nets to file in encrypted form, which allows deploying solutions relying on Bayes nets kept private to an organization.
- **Sensitivity:** Netica can efficiently measure the degree to which findings at any node can influence the beliefs at another node, given the findings currently entered. The measures can be in the form of mutual information (entropy reduction), or the expected reduction of real variance.
- **Advanced Decision Nets:** Can solve influence diagrams which have multiple utility and decision nodes to find optimal decisions and conditional plans, using a junction tree algorithm for speed. Handles multi-stage decision problems, where later decisions depend on the outcomes of earlier ones, and on observations not initially known. No-forgetting links need not be explicitly specified.
- **Junction Tree Algorithm:** Can compile Bayes nets and influence diagrams into a junction tree of cliques for fast probabilistic inference. An elimination order can be specified or Netica can determine one automatically, and Netica can report on the resulting junction tree.
- **Soft Evidence:** Accepts likelihood findings (i.e., "virtual evidence"), and

findings of the form that some variable is not in some state.

- **Link Reversal:** Can reverse specified links or "sum out" (absorb) nodes of a Bayes net or influence diagram while maintaining the same overall joint probability distribution, properly accounting for any findings in the removed nodes or other nodes.
- **Disconnected Links:** Links may be individually named and disconnected from parent or child nodes, thus making possible libraries of net fragments, which you may then copy and connect to other nets or node configurations.
- **Case Support:** Can save individual cases (i.e. sets of findings) to file, and manipulate files of cases. Cases may be incomplete, and may have an associated ID number and multiplicity.
- **Simulation:** Can do sampling (i.e. simulation) to generate random cases with a probability distribution matching the Bayes net. Can use a junction tree algorithm for speed, or direct sampling for nets too large to generate CPTs or a junction tree.
- **User Data:** Every node and net can store by name arbitrary data fields defined by you. These are saved to file when the object in question is being saved. As well, there are fields not saved to file, which can contain a pointer to anything you wish.
- **Error Handling:** Has a simple but powerful method for handling usage errors, which can generate very detailed error messages if desired. It won't throw exceptions (C++ version does).
- **Argument Checking:** Allows programmers to control how carefully API functions check their arguments when they are called, including a "development mode" to extensively check everything passed to an API function.
- **Compatibility:** Can work hand-in-hand with Netica Application standalone product (for example, sharing the same files), and with Netica API versions for other languages.
- **Efficient:** Is optimized for speed, and is not too large (about 500 KB to 3 MB depending on platform/usage, 1 MB typical)
- **Language Interface:** Usable by programs written in many languages, such as: C, C++, Java, Python, Perl, Visual Basic, Delphi Pascal, Lisp, CLisp, Fortran, Cobol, and Matlab.
- **Many Platforms:** Is available for a wide range of platforms including MS

Windows (95/NT to XP), Linux, Macintosh (OS X and Classic), Sun Sparc, Silicon Graphics and DOS among others.

- **Memory Limiting:** You can set a bound on how much total heap space Netica API is allowed to allocate for large tables, thereby preventing virtual memory thrashing or the memory-starving of other parts of your application.
- **More Features:** A more extensive list of features is available from: [http://www.norsys.com/netica\\_api.html](http://www.norsys.com/netica_api.html) and for those features specific to the C version: [http://www.norsys.com/netica\\_c\\_api.htm](http://www.norsys.com/netica_c_api.htm)

## Legalities

Before installing or using Netica, be sure that you accept the [License Agreement](#), which is provided with the software on a separate document (entitled License Agreement.pdf).

This entire document (consisting of all screens in Netica's help file) is:

Copyright © 1996 - 2011 by Norsys Software Corp.

This file may be copied and stored freely, provided it is duplicated in its entirety, without modification, and including the copyright notice.

Norsys Software Corp.  
3512 West 23rd Ave.  
Vancouver, BC, Canada  
V6S 1K5  
[www.norsys.com](http://www.norsys.com)

While every precaution has been taken in the preparation of this document, we assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

Netica and Norsys are registered trademarks of Norsys Software Corp.

Microsoft and Windows are registered trademarks of Microsoft, Inc.

iOS is a trademark licensed to Apple Computer, Inc. and Macintosh and Mac OS are registered trademark of Apple Computer, Inc.

UNIX is a registered trademark of The Open Group.

Linux is a registered trademark of Linus Torvalds.

Unicode is a trademark of Unicode, Inc.

Intel is a registered trademark, and Pentium is a trademark of Intel Corporation.

Other brands and product names are trademarks of their respective holders.

# Netica Application Help

To find out more about any of the subjects below, click on it and then use the Next/Previous buttons in the navigation pane above to read subsequent pages.

For a table of contents, index, or full text search click on the appropriate button above.

- [Getting Started](#)
- [Quick Tour](#)
- [Using Bayes Nets](#)
- [Creating Bayes Nets and Decision Nets](#)
- [Changing Node Properties](#)
- [Changing/Viewing Node Tables](#)
- [Cases and Case Files](#)
- [Learning From Cases](#)
- [Test Net with Cases](#)
- [Sensitivity](#)
- [Decision-Making Nets](#)
- [Display Style and Printing](#)
- [Equations to Generate Tables](#)
- [Reports and Data Linking](#)
- [Custom Reports](#)
- [Node-Sets and Coloring](#)
- [Dynamic Bayes Nets](#)
- [Net Fragment Libraries](#)
- [Transforming a Net](#)
- [Process Cases](#)
- [COM Interface – Programming Netica](#)
- [Bibliography](#)
- [Netica-Web](#)

## Installing Netica on your Mac

1. Download the trial version of [Crossover Mac](#), and save it to your hard drive.

Having trouble finding your Mac hard drive? See Navigating Folders below.

2. Add the Crossover icon to your Application directory. In the Applications directory, choose **New Folder** and name it "Crossover".

This is where you will save all Windows-based programs moving forward.

3. Download the Netica software application. You may obtain it by downloading from the Norsys [website](#), or otherwise from

Norsys. Double-click the file icon and extract it to the Crossover folder that you created in Step 2 above.

4. Double-click on the Netica icon  of Netica.exe in the directory you indicated above.

5. There are a few minor bugs with right-clicking and onscreen help, for more information see the [Mac FAQ](#) page.

### Navigating Folders in Crossover Mac

Due to the nature of running two operating systems on your machine, there are a number of drives and folders to navigate, including (Z:), (Y:), (C:), (Desktop) and (My Computer).

Thus, it can be confusing to navigate through the folders when you first install, save or open Netica files on your Mac.

*Here is a breakdown of the various drives and folders under Crossover Mac:*

**1. Desktop:** This is a Windows reference. This is NOT your Mac desktop.

There should be a separate folder in that same list of directories called "My Mac Desktop".

You need to install Netica on your Mac through the (Z:) drive.

Choose 'My Mac Desktop' if you want to access non-application files on your Mac.

**2. My Computer:** If you open 'My Computer', you will find 3 drives:

drive\_c(C:)

/Users/yourcomputername (Y:)

/(Z:) – this also shows up as an icon of a hard drive with a blank (/)

**Drive\_c:** If you are used to working on a PC, the usual way to save a file is the C drive.

This is not the case when you are working with Crossover Mac.

The C drive has a folder named "**Program Files**". Do not be fooled, these are not the Program Files of your Mac!

(Y:) takes you directly to the main account directory of your Mac, where you will see a folder for your Desktop, Downloads, Documents, etc.

/(Z:) is basically the equivalent to the c:drive in Windows.

The path to your Mac Applications is: Z/Applications/Crossover  
Save Netica in the top directory of the Crossover folder.

**Remember:** You can save your BN files anywhere on your Mac.

If you are trying to open a BN file from somewhere on your Mac, the path to your Desktop is:

Z/Users/YourUserName/Desktop. Or from "My Mac Desktop".

## Un-Installing Crossover

1. Drop your CrossOver bundle into the trash and empty it. This may first require you to stop the CrossOver CD Helper.

To stop it manually, run the **Activity Monitor** (usually found in your **Applications/Utilities folder**).

Select CrossOver CD Helper in Activity Monitor and click **Quit Process** to stop it.

2. User-specific CrossOver files are located in your home folder, under the: **Library/Application Support/CrossOver** hierarchy.

Trash that folder and its contents for each user of CrossOver on your system.

3. Then, to be truly comprehensive, remove the preferences files located in: **Library/Preferences/** called `com.codeweavers.CrossOver.plist`.

4. If you have any published bottles or CrossOver has been run using your system's administrator account,

you may have files in **/Library/Application Support/CrossOver** on your startup drive.

These may also be deleted.

## Building Tables from Other Nets

There are many ways that Netica can [learn](#) the CPTs of nodes. It learn from [case data](#), learn from a [case file](#), learn from an [Excel file](#) or learn from other nets (called *source nets*). In the latter case, you would be building a destination net that represents the same world as a source net. In other words, Netica will learn the CPT tables of a sub-net within a source net. All the inference results will be the same in both nets. The difference is that it is using a different representation, so the link structure can be different. In particular, nodes that have links in one net, may not have links in the other and the link directions might be different.

**How to:** open an existing net, or [create a new one](#). If it is a new net, you may want to [copy and paste](#) nodes from the source net into the destination net to ensure the node and state names are the same. You can then select a single node, or no nodes (in which case all nodes will be learned). Next choose **Table** → **Build From Other Net**. Netica will automatically learn the sub-set of CPTs from the source net.

### Notes:

- Nodes in the destination net must have the same name and same state names as the source net, but they can have different titles. You may want to copy & paste nodes between nets (and delete the resultant input links) to achieve this.
- Netica can't learn CPTs from the source net if there is a parent node in the destination net that is not in the source net. To resolve this, first learn the tables from the source net, then add the new parent(s) to the destination net. This is not true for children nodes.
- In both cases, Netica will learn as many of the nodes as possible.

**Applications:** when you are learning a net from data, the link structure used can significantly affect the quality of learning. For example, if you have a target node, Netica can determine a very effective [TAN structure](#). However, to combine with other sub-nets, you may want a different link structure.

So, you can create a net with the link structure you want and then build it's

CPTs using the learned net as the source net.

## Noisy-Or Distribution

(DM prob. dist. for [equations](#))

**Usage:** NoisyOrDist (e, leak, b1, p1, ... bn, pn)

**Definition:**  $P(e) = 1 - [(1 - \text{leak}) \prod_{i=1}^n (b_i ? (1 - p_i) : 1)]$

**Required:**  $0 \leq \text{leak} \leq 1$  e, bi Boolean  $0 \leq p_i \leq 1$

Use this distribution when there are several possible causes for an event, any of which can cause the event by itself, but only with a certain probability.

Also, the event can occur spontaneously (without any of the known causes being true), with probability **leak** (make this zero if it can't occur spontaneously).

Each **bi** is a `4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_boolean_node.htm');return false;">boolean variable, which may cause the event when its TRUE. e is also a boolean, which indicates whether the event occurs. Each of the pi are the probability that e will occur if bi is TRUE in isolation.`

If **leak** is zero, and only one possible cause is `TRUE`, say **bk**, then the probability for **e** is **pk**. If more possible causes are `TRUE`, **P(e)** will be greater. And if **leak** is nonzero, **P(e)** will be greater. Reducing a **pi** always results in the same or lower **P(e)**.

**pi** can be considered the “strength” of the relation between **e** and **bi**, with zero indicating independence (link could be removed), and 1 indicating maximum effect.

See [Pearl88](#), page 184 for more information (his  $q_i = 1 - p_i$ ).

**Example:**  $P(\text{Effect} \mid \text{Cause1}, \text{Cause2}) =$   
NoisyOrDist (Effect, 0.1,  
Cause1, 0.2,  
Cause2, 0.4)

## [Another Example](#)

See also [NoisyAndDist](#), [NoisyMaxDist](#) (for nodes with more than just 2

states), [NoisySumDist](#).

## Other Smaller Improvements

In addition to its major [new features](#), [Netica](#) has the following improvements.

### From version 5.00 to version 5.03

- When printing a net, you can enter the number of pages you want it to appear on, instead of just the magnification. Default is 1. But if you want to enter a magnification, just make those entries blank.
- Keeps track of starting experience used for learning (normally 1 for each state). Saves in NETA file for each node, and adjusts when doing table hardening or softening.
- Fixed: When select multiple nodes and do Sensitivity to Findings, gave a #2516 error.
- Fixed: Sometimes constant nodes weren't registered to be available to equations until after saving to file and re-reading.
- While EM learning from a case file having a case inconsistent with the Bayes net, now puts the IDnum of the case in the error message, and allows halting or continuing without it.
- Equations now recognize discrete constant nodes with a state setting.
- Names of built-in node-sets are now preceded by colon (:) instead of dash (-)
- Fixed: Process-cases didn't work when there was a finding entered in the net (gave a 2592 C0000005 error).

### From version 4.00 to version 4.16

- Greatly improved the CPTable editor, both cosmetically and functionally.
- Text editor: Handles larger text files (up to 2GB) for text editing, instead of just 30K. Helps avoid Messages Window overflow. Text entry in dialog boxes (such as user fields) can also take larger text amounts.
- Disconnecting/connecting links now works better. The tool button can reconnect as well as disconnect. When right-click on a disconnected link, menu now has "Reconnect", which reconnects link to original node (even across file save/read).

- Can now read .xlsx and .accdb database case files (as well as the old .xls and .mdb).
- When reading a case file, missing data symbols (empty, ' ', '?', '\*', N/A) will now instead be interpreted as a state, if the node has a state with an exactly matching state title.
- New onscreen help system.
- Within .dne files, it now uses a superior flat format for tables (which means the Bayes net files it creates cannot be completely read by very old versions of Netica (previous to 2.27 of 2003-05-02)). But, of course, new versions of Netica can still read all the old .dne files, created by any previous version of Netica.
- Fixed: When reading UVF files, uncertain findings for states with an \_ in their name caused error #2878.
- Fixed: When reading UVF files, extra spaces could cause errors.
- Fixed: When left and right mouse buttons in Windows were configured as left-handed, Netica couldn't select nodes.
- Fixed: Reading Hugin files didn't work as well as earlier versions of Netica.
- Can now read XML BIF 0.3 files, as described in 1998 Fabio Cozman document:  
<http://www.cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeForr>
- Mouse wheel now scrolls instead of zooms.
- Node dialog: Fixed up the node properties dialog box multipurpose box display and entry of state info (state names, state titles, state comments, state numbers) significantly (eg, allow single entry or "All" at once for each).

### **From version 3.18 to version 3.25:**

- Fixed bug: Sometimes after adding a utility node (or any undiscretized continuous node), and then trying to access its table, an internal error would result.
- Improved Unicode support (**LabelBox** display, docn nodes, setting by right-clicking, etc.)
- Fixed "Process Cases" to work with UVF files.
- Undo/redo for nodeset operations now works properly.

- Obfuscate net deletes documentation nodes.
- Improved the support for old versions of Microsoft Windows (Windows 95, 98, Me).

### **From version 3.16 to version 3.18:**

- Table editor now has a \* indicator in its title bar to show when changes have been made that are not yet applied to the node.
- Comments that you create to appear when the mouse hovers over a node or state can now be multi-line.
- Links are now always drawn underneath the nodes for diagram clarity.
- Fixed annoying situation where equation is constantly re-established when trying to delete equation from node properties dialog by backspacing.
- Fixed problems occurring when no default printer is installed, such as error message: **\*\*1223\*\*** X component of `drawing bounds` in VISUAL net 'V1' is too large (= ..., but maximum is 16383).
- Fixed bug: Database access gave an error when the first node in the net was skipped (i.e., wasn't a database column).
- Lots of small improvements to the table editor dialog box.
- Fixed bug: 'Add Case File Nodes' didn't do anything in some situations.
- Can now scroll with arrow keys.
- Can now enter empty strings for values of user-defined fields in node dialog box.
- If nodes at each end of a link are in "hidden" format, the link is not drawn.
- Other minor improvements and bug fixes.

## Doing Probabilistic Inference

After a Bayes net has been [constructed](#) and [compiled](#), you can apply it to a particular = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_case.htm');return false;">case by [entering findings](#) (also known as “evidence”), which are the values for nodes that you know. Then you use [belief updating](#) (which is a form of [probabilistic inference](#)) to determine new probabilities for the states of all the other nodes.

You can run through the process using this [example](#).

As well as finding beliefs, Netica can find the most likely configuration of the remaining variables, according to the findings entered. This is called the “[Most Probable Explanation](#)”.

You can also use Netica to do [sensitivity analysis](#) to find how tightly coupled nodes are. That can be used to determine the degree to which a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_finding.htm');return false;">finding at one node is expected to change the beliefs at other nodes, or which nodes would be the best to obtain findings for, in order to obtain maximum information on another node.

## Probabilistic Inference by Node Absorption

It is possible to do = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_probabilistic\_inference.htm');return

false;">probabilistic inference using node absorption, by = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_enter\_finding.htm');return false;">entering all

the findings, and then absorbing all the nodes except for a single query node.

The resulting probability = 4 && typeof(BSPSPopupOnMouseOver) ==

'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_CPT.htm');return false;">CPT for that node will

be a single belief vector (because the node won't have any parents), which is

the same as the beliefs that would be obtained by compiling the Bayes net and

doing belief updating. Of course, the net is destroyed in the process, but you

can recover it by choosing **Edit** → **Undo**. Normally it is better to do inference

by = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_compile\_net.htm');return false;">compiling the

net and doing = 4 && typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_belief\_updating.htm');return false;">belief

updating, but sometimes additional insights are gained by using node

absorption for inference.

It should be mentioned that node absorption will also work with = 4 &&

typeof(BSPSPopupOnMouseOver) == 'function')

BSPSPopupOnMouseOver(event);" class="BSSCPopup"

onclick="BSSCPopup('X\_PU\_decision\_nets.htm');return false;">decision nets.

To solve a decision net, select all its nodes, and click the  tool button. The

nature nodes will all be absorbed out. When a decision node is absorbed, it is

not removed from the net; instead it is completely disconnected and its

decision table set to the optimal decision function. Utility nodes are also left,

so you can see the expected utility. The algorithm used is described in

[Shachter86](#), [Shachter88](#) and [Shachter89](#).

When using node absorption to solve decision problems, the decision nodes

```
must have = 4 && typeof(BSPSPopupOnMouseOver) == 'function')
BSPSPopupOnMouseOver(event);" class="BSSCPopup"
onclick="BSSCPopup('X_PU_no_forgetting_link.htm');return false;">no-
forgetting links.
```

There must be only one utility node, with no descendants.

And if not all the nodes of the net are being absorbed, they must consist of a descendant subnet. So, if nodes are absorbed one-by-one, a suitable order must be used. These restrictions are explained in more detail in the Shachter references mentioned above. If any of these restrictions are not met, Netica will not produce an erroneous result, but will just absorb as many of the nodes as it can, and then display a message explaining why it was impossible to proceed.

**rect** (function for [equations](#))

Usage: `rect(x, a, b)`

Definition: `(a <= x && x < b) ? 1 : 0`

Required: `a ≤ b`

Returns 1 if **x** is between **a** and **b**, otherwise 0.

This function is often multiplied by another to "capture a piece" of the other function.

Also known as the "rectangular function" because of the rectangular shape of its graph.

This function is the same as the probability distribution [UniformDist](#), except height of UniformDist the is normalized so that the area under its curve is 1.

If **b** is infinity, the "unit step function" is formed.

See also [clip](#), and [sign](#).

In order to do some operations on a node, you first *select* it by clicking once on it, and it will then be drawn using negative colors to hilite it. You can select several nodes at a time by clicking on the background and dragging the selection rectangle to include them. To add or remove nodes from your selection, hold down the `CTRL` key while you do the selection operation, or right-click on the node and choose **Select/Deselect**. [More Info](#)

## Beta Distribution

(continuous probability dist for [equations](#))

Usage: BetaDist (x,  $\alpha$ ,  $\beta$ )

Definition:  $x^{\alpha-1} (1-x)^{\beta-1} / \text{beta}(\alpha, \beta)$   
where beta is the [beta function](#)

Required:  $\alpha > 0$      $\beta > 0$

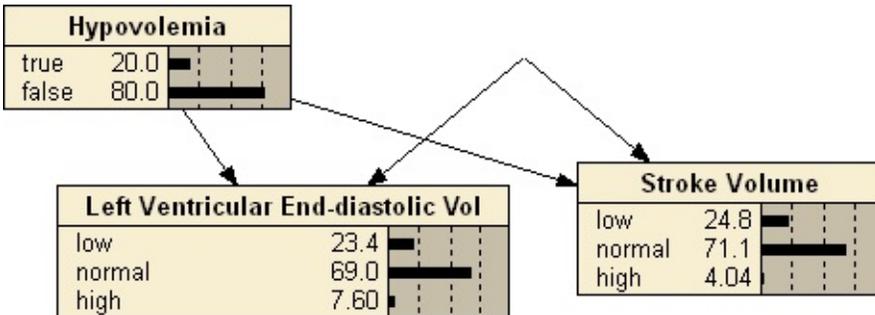
Support:  $0 \leq x \leq 1$

Moments:  $\mu = \alpha / (\alpha + \beta)$   
 $\sigma^2 = \alpha \beta / [(\alpha + \beta)^2 (\alpha + \beta + 1)]$   
 $\gamma_1 = 2 (\beta - \alpha) \text{sqrt}((\alpha + \beta + 1) / (\alpha \beta)) / (\alpha + \beta + 2)$   
 $\beta_2 = 3 (\alpha + \beta + 1) [2(\alpha + \beta)^2 + \alpha \beta (\alpha + \beta - 6)] / [\alpha \beta (\alpha + \beta + 2) (\alpha + \beta + 3)]$

Almost any reasonably smooth unimodal distribution on  $[0,1]$  can be approximated by some beta distribution (if its not on  $[0,1]$ , see [Beta4Dist](#)). An important use of the beta distribution is as a conjugate distribution for the parameter of a [Bernoulli](#) distribution.

## Hidden Node Style

Sometimes it is useful to hide nodes in your net, especially when working with [large nets](#).



### To unhide a node(s):

1. Click down on the net background (i.e. not on a node or link), and then drag with the mouse so that the dashed line rectangle goes over the node that was made hidden. The node will then be selected (even though you can't see it), so you can choose **Style** → **Default** from the menu to "bring it back" (available from both the overhead and right-click menus).
2. Alternately, if it won't harm any other nodes, you can just do **CTRL-A** to select all the nodes, then choose **Style** → **Default** from the menu.
3. If hiding the node was the last action you performed, you can simply click the **Undo** arrow or do **CTRL-Z**.

## Normal Distribution

(continuous probab dist for [equations](#))

Usage: NormalDist (x,  $\mu$ ,  $\sigma$ )

Definition:  $[1/(\sigma \sqrt{2\pi})] \exp (-[(x-\mu)/\sigma]^2 / 2)$

Required:  $\sigma > 0$

Support  $-\infty < x < \infty$

Moments: mean =  $\mu$   
standard deviation =  $\sigma$   
 $\gamma_1 = 0$        $\beta_2 = 3$

The normal distribution, or approximations of it, arise frequently in nature (this is partly explained by the `typeof(BSPSPopupOnMouseOver) == 'function'` BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_central\_limit\_theorem.htm');return false;">central limit theorem). Since it also has many convenient mathematical properties it is the most commonly used continuous distribution.

For this distribution, 68.2% of the probability is within 1 standard deviation of the mean, 95.4% is within 2 standard deviations, and 99.74% is within 3 standard deviations.

If  $\mu = 0$ ,  $\sigma = 1$  it is known as a “standard normal” distribution.

## **\_find0 Function**

This is a function that Netica uses [internally](#) to find the state of a discrete node having numeric values associated with its states, given one of those numeric values. If Netica gives you a message saying there was an error evaluating `_find0(z, x0, x1, ... xn)`, where the  $z$  and  $x_i$  are numbers, then your equation is supplying illegal values, even if you never explicitly used `_find0` in your equation.

The value of  $z$  must equal one of the  $x_i$ , or else the equation is saying that the node has a value that is not represented by any of its states. You can change the equation to only supply numbers that match those attached to the state of the node (perhaps using [nearest](#)), or you can change the numbers attached to the states. Or, if you wish it to be able to match numbers in between the  $x_i$ , then you should change the node to a continuous one, and leave the same set of attached numeric values as its discretization [thresholds](#).

## **\_Bernoulli Function**

This is a distribution that Netica uses internally to represent the Bernoulli distribution. If you get an error message saying there was an error evaluating \_Bernoulli (k, p), where k and p are numbers, then your equation is supplying illegal values, even if you never explicitly used \_Bernoulli in your equation.

For instance, if your equation for `= 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_boolean_node.htm');return false;">boolean` node B is  $P(B|x) = x/10$  and values of x can go up to 11, then \_Bernoulli (1, 1.1) will be illegal, since you are supplying 1.1 as a probability (and Netica can't normalize it, since no probability for B being false is given).

**round** (function for [equations](#))

Usage: round (x)

Definition: floor (x + 1/2)

Required: x is an unrestricted real

Rounds **x** to the nearest integer. To round off to other quantities, use [roundto](#).

**roundto** (function for [equations](#))

Usage: roundto (dx, x)

Definition:  $dx * \text{floor}((x + dx/2) / dx)$

Required:  $dx > 0$

Rounds **x** to the nearest **dx**, which may be less than or greater than 1.

For example, roundto(10,17) rounds 17 to the nearest 10, and so it returns 20.

If **dx** = 1, then this is the same as [round](#).

**approx\_eq** (function for [equations](#))

Usage: `approx_eq(x, y)`  $x \approx y$

Definition: `eqnear(2e-5, x, y)` Definition of [eqnear](#)

Required: `x` and `y` are unrestricted reals

Returns true iff  $x$  is equal to  $y$ , within a small relative tolerance.

Usually the operator form of this function is most convenient:  $x \approx y$

To have control of the tolerance, use [eqnear](#).

**eqnear** (function for [equations](#))

Usage: eqnear (reldiff, x, y)

Definition:  $|X - Y| / \max(|X|, |Y|) \leq \text{reldiff}$

Required: reldiff  $\geq 0$

Returns true iff **x** is equal to **y**, within **reldiff**.

To have Netica choose **reldiff**, use [approx\\_eq](#).

## clip

(function for [equations](#))

Usage: clip (min, max, x)

Definition:  $(x < \text{min}) ? \text{min} : (x > \text{max}) ? \text{max} : x$

Required:  $\text{min} \leq \text{max}$

Returns **x**, unless it is less than **min** (in which case it returns **min**), or more than **max** (in which case it returns **max**).

See also [max](#), [min](#), and [rect](#).

**sign** (function for [equations](#))

Usage: `sign(x)`

Definition:  $(x > 0) ? 1 : (x < 0) ? -1 : 0$

Required: `x` is an unrestricted real

Returns 1 if **x** is positive, -1 if **x** is negative, and 0 if **x** is zero.

See also [rect](#), [clip](#), and [abs](#).

**xor** (function for [equations](#))

Usage: xor (b1, b2, ... bn)

Definition: odd (NumberTrue (b1, b2, ... bn))

Required: bi are boolean

Returns the exclusive-or of **b1, b2 ... bn**.

This is also known as the parity function, and will return true iff an odd number of **bi** evaluate to true.

See also [and](#), [or](#), [not](#), and [equal](#).

**increasing** (function for [equations](#))

Usage: increasing (x1, x2, ... xn)

Definition:  $(x_2 > x_1) \ \&\& \ (x_3 > x_2) \ \&\& \ \dots \ \&\& \ (x_n > x_{n-1})$

Required: xi are unrestricted reals

Returns true iff each  $x_i$  is greater than the previous one.

If you wish the test to be “greater than or equals”, use [increasing\\_eq](#).

See also [greater](#).

**increasing\_eq** (function for [equations](#))

**Usage:** `increasing_eq (x1, x2, ... xn)`

**Definition:** `(x2 >= x1) && (x3 >= x2) && ... && (xn >= xn-1)`

**Required:** xi are unrestricted reals

Returns true iff each **xi** is greater or equal to the previous one.

If you wish it to be just greater than, use [increasing](#).

See also [greater\\_eq](#).

**avg** (function for [equations](#))

**Usage:** avg (x1, x2, ... xn)

**Definition:**  $(x_1 + x_2 + \dots + x_n) / n$

**Required:** xi are unrestricted reals

Returns the average (i.e. mean) of x1, x2, ... xn.

At least one argument must be passed.

See also [max](#).

**Example:** min (10, 6.6, 3.4, 126, 3.4) returns 3.4

**mag** (function for [equations](#))

**Usage:** mag (x1, x2, ... xn)

**Definition:**  $\text{sqrt}(x_1^2 + x_2^2 + \dots + x_n^2)$

**Required:** xi are unrestricted reals

Returns the magnitude of the vector [x1, x2, ... xn].

This is also known as the Euclidean length of the vector.

If the real and imaginary part of a complex number are passed, the magnitude of the complex number is returned.

If no arguments are passed, zero is returned. If one argument is passed, its absolute value is returned.

See also [abs](#), [max](#).

**Example:** mag (1.5, 0, -2) returns 2.5

**min** (function for [equations](#))

Usage: min (x1, x2, ... xn)

Definition: xi s.t. (xi <= xj) for all j

Required: xi are unrestricted reals

Returns the minimum of **x1, x2, ... xn**.

At least one argument must be passed.

If you just want the index of the minimum (i.e. its position in the list), use [argmin](#).

See also [max](#).

Example: min (10, 6.6, 3.4, 126, 3.4) returns 3.4

**max** (function for [equations](#))

Usage: `max (x1, x2, ... xn)`

Definition: `xi s.t. (xi >= xj) for all j`

Required: `xi are unrestricted reals`

Returns the maximum of **x1, x2, ... xn**.

At least one argument must be passed.

If you just want the index of the maximum (i.e. its position in the list), use [argmax](#).

See also [min](#).

Example: `max (-10, 6.6, 3.4, -126, 3.4)` returns 6.6

**argmin** (function for [equations](#))

**Usage:** argmin0 (x0, x1, ... xn)  
argmin1 (x1, x2, ... xn)

**Definition:** i s.t. ( $x_i \leq x_j$ ) for all j

**Required:** xi are unrestricted reals

Returns the index (position in list) of the argument with the lowest value. If there are several with the same lowest value, then the index of the first occurrence will be returned. The first argument has index 0 if argmin0 is used, or index 1 if argmin1 is used.

At least one argument must be passed.

If you want the minimum value itself, rather than its index, instead use [min](#).

See also [argmax](#).

**Examples:** argmin0 (10, 6.6, 3.4, 126, 3.4) returns 2  
argmin1 (-5e3, 6.6, 3.4, -126) returns 1

**argmax** (function for [equations](#))

**Usage:**        `argmax0 (x0, x1, ... xn)`  
                  `argmax1 (x1, x2, ... xn)`

**Definition:**    `i` s.t. ( $x_i \geq x_j$ ) for all `j`

**Required:**     `xi` are unrestricted reals

Returns the index (position in list) of the argument with the highest value. If there are several with the same highest value, then the index of the first occurrence will be returned. The first argument has index 0 if `argmax0` is used, or index 1 if `argmax1` is used.

At least one argument must be passed.

If you want the minimum value itself, rather than its index, instead use [max](#).

See also [argmin](#).

**Examples:**     `argmax0 (1, -6.6, 3.4, 1.26, 3.4)` returns 2  
                  `argmax1 (5e3, -6.6, -3.4, 126)`    returns 1

**nearest** (function for [equations](#))

**Usage:** nearest0 (val, x0, x1, ... xn)  
nearest1 (val, x1, x2, ... xn)

**Definition:** i s.t.  $(|val - x_i| \leq |val - x_j|)$  for all j

**Required:** val and xi are unrestricted reals

Returns the index (position in list) of the argument with the value closest to **val** (as measured by the absolute value of the difference). If there are several with the same smallest difference, then the index of the first occurrence will be returned. The first **x** argument has index 0 if nearest0 is used, or index 1 if nearest1 is used.

Must be passed at least 2 arguments (**val** and an **x**).

For the inverse function, see [select](#).

See also [member](#).

**Examples:** nearest0 (1, 1, 3.4, 1, 3.4) returns 0

nearest1 (5e3, -6.6, -3.4, 126) returns 3

## select

(function for [equations](#))

**Usage:** select0 (index, x0, x1, ... xn)  
select1 (index, x1, x2, ... xn)

**Definition:** xi s.t. i == index

**Required:** index is integer, xi are all the same type  
select0:  $0 \leq \text{index} < n$   
select1:  $0 < \text{index} \leq n$

Returns the value of the **x** argument at position **index**:  $x[\text{index}]$

The first **x** argument is at index 0 if select0 is used, and at index 1 if select1 is used.

Must be passed at least 2 arguments (**index** and an **x**).

For the inverse function, see [nearest](#).

**Examples:** select0 (1, -6.6, 3.4, 1.26, 3.4) returns 3.4  
select1 (1, -6.6, 3.4, 1.26) returns -6.6

**member** (function for [equations](#))

**Usage:** member (elem, s1, s2, ... sn)

**Definition:** (elem == s1) || (elem == s2) || ... || (elem == sn)

**Required:** elem and all si must be the same type

Returns true iff one of the si arguments has the same value as **elem**..

See also [nearest](#).

**Examples:** member (1, -6, 3, 1, 3) returns true

member (C, blue, red) and C = red returns true

**factorial** (function for [equations](#))

Usage: factorial (n)

Definition:  $n (n - 1) (n - 2) \dots 1$

Required:  $n \geq 0$  n is an integer

Returns the factorial of **n**, which is the product of the first **n** integers.

factorial(n) is often written as n!

factorial(0) = 1

Even fairly small values of **n** (around 170) can cause factorial to overflow. For that reason calculations with the factorial function are often done using the logarithm of the results, for which you can use [logfactorial](#).

If **n** is not an integer you may want to use the [gamma](#) function, which for integer values is related to factorial by: factorial (n) = gamma (n + 1) but which is also defined for non-integer values.

**logfactorial** (function for [equations](#))

Usage: logfactorial (n)

Definition:  $\log (n (n - 1) (n - 2) \dots 1)$

Required:  $n \geq 0$  n is an integer

Returns the natural logarithm of the [factorial](#) of **n** (i.e. **n!**).

If **n** is not an integer you may want to use the [loggamma](#) function, which for integer values is related to logfactorial by:  $\logfactorial (n) = \loggamma (n + 1)$  but which is also defined for non-integer values.

**logistic** (function for [equations](#))

**Usage:** logistic (t)

**Definition:**  $\exp(t) / (1 + \exp(t))$

**Required:** t is an unrestricted real

Logistic is also known as "expit", or "sigmoid".

This is the inverse of the logit function (also known as log odds).

See also [LogisticDist](#), [logit](#).

**Example:** expit (0) returns 0.5  
expit (1) returns 0.7310585

**logit** (function for [equations](#))

**Usage:** `logit (p)`

**Definition:**  $\log (p / (1 - p))$

**Required:**  $0 < p < 1$

logit is also known as "log odds".

This is the inverse of the logistic function (also known as "expit" or "sigmoid")

It approaches negative infinity as  $p$  approaches  $0$ , and positive infinity as  $p$  approaches  $1$ .

See also [logistic](#).

**Example:** `logit (0.5)` returns  $0$   
`logit (0.75)` returns  $1.0986123$

**gamma** (function for [equations](#))

Usage: gamma (x)

Required:  $x \geq 0$

Returns the gamma function of **x**.

The gamma function is normally defined for negative values of **x** as well, but Netica cannot compute these.

Don't confuse this function with the [gamma](#) probability distribution.

Even fairly small values of **x** (around 170) can cause gamma to overflow. For that reason calculations with the gamma function are often done using the logarithm of the results, for which you can use [loggamma](#).

For integer values of **x**, the gamma function is related to the [factorial](#) function by:  $\text{factorial}(n) = \text{gamma}(n + 1)$

**loggamma** (function for [equations](#))

Usage: loggamma (x)

Definition: log (gamma (x))

Required:  $x \geq 0$

Returns the natural logarithm of the [gamma](#) function of **x**.

The loggamma function is normally defined for negative values of x as well, but Netica cannot compute these.

**beta** (function for [equations](#))

Usage: `beta (z, w)`

Definition:  $\text{gamma}(z) \text{gamma}(w) / \text{gamma}(z + w)$

Required:  $z > 0$   $w > 0$

Returns the beta function of **z** and **w**.

[BetaDist](#) is the beta probability distribution, which is based on the beta function.

See also [gamma](#).

## erf

(function for [equations](#))

Usage: erf (x)

Definition:  $(2 / \sqrt{\pi})$  **Integral from 0 to x of**  $\exp (-t^2)$  dt

Required: x is an unrestricted real

This returns the error function of **x**. It is useful for calculating integrals of the [normal](#) distribution.

If **x** is large, you can obtain better accuracy with [erfc](#).

## erfc

(function for [equations](#))

Usage:  $\text{erfc}(x)$

Definition:  $1 - \text{erf}(x)$  (definition of [erf](#))

Required:  $x$  is an unrestricted real

This returns the complementary error function of  $x$ . It is useful for calculating an integral of a tail of a [normal](#) distribution.

It would be easy enough to just use  $1 - \text{erf}(x)$ , but this provides better numerical accuracy when  $x$  is large (so  $\text{erf}(x)$  is very close to 1).

## binomial

(function for [equations](#))

Usage: binomial (n, k)

Definition:  $n! / (k! * (n-k)!)$

Required:  $0 \leq k \leq n$  n and k are integers

Returns the binomial coefficient ( **$n \ k$** ). This is the number of different **k**-sized groups that can be drawn from a set of **n** distinct elements. See also the [multinomial](#) function.

[BinomialDist](#) is the binomial probability distribution, which is based on the binomial coefficient.

**multinomial** (function for [equations](#))

Usage: multinomial (n1, n2, ... nn)

Definition:  $(n_1 + n_2 + \dots + n_n)! / (n_1! * n_2! * \dots * n_n!)$

Required:  $n_i \geq 0$      $n_i$  are integers

Returns the number of ways an **(n1+n2+...nn)** sized set of distinct elements can be partitioned into sets of size **n1, n2, ... nn**. If partitioning into only two sets, this is the same as [binomial](#).

The probability distribution based on this function is the [multinomial distribution](#).

## Uniform Distribution

(continuous probability dist for [equations](#))

Usage: UniformDist (x, a, b)

Definition:  $1 / (b - a)$

Required:  $b > a$

Support:  $a \leq x \leq b$

Moments:  $\mu = (a + b) / 2$     $\sigma = (b - a) / \sqrt{12}$

$\gamma_1 = 0$     $\beta_2 = 1.8$

This is the distribution to use when the minimum and maximum possible values for a variable are known, but within that range there is no knowledge of which value is more likely than another.

It has a constant value from  $x = a$  to  $x = b$ , and zero value outside this range.

Also known as the "rectangular distribution", and similar to the [rect](#) function.

## Triangular Distribution (continuous probab dist for [equations](#))

Usage: TriangularDist (x, m, w)

Definition:  $(w - |x - m|) / w^2$

Required:  $w > 0$

Support:  $m - w \leq x \leq m + w$

Moments:  $\mu = m$        $\sigma = w / \sqrt{6}$   
 $\gamma_1 = 0$        $\beta_2 = 2.4$

The graph of this distribution has a triangular shape, with the highest point at  $x = a$ , and nonzero values from  $a - w$  to  $a + w$ .

See also [Triangular3Dist](#) , and [TriangularEnd3Dist](#).

## Asymmetric Triangular Distribution (continuous prob. dist. for [equations](#))

**Usage:** Triangular3Dist (x, m, w1, w2)

**Definition:**  $2(x-m+w1)/(w1(w1+w2))$  for  $m-w1 \leq x \leq m$ ,  
 $2(m+w2-x)/(w2(w1+w2))$  for  $m \leq x \leq m+w2$

**Required:**  $w1 \geq 0$   $w2 \geq 0$   $w1$  &  $w2$  can't both be 0

**Support:**  $m-w1 \leq x \leq m+w2$

**Moments:**  $\mu = m + (w2 - w1) / 3$

The = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_pdf.htm');return false;">pdf has a triangular  
shape, with the highest point at  $x=m$ , and nonzero value from  $m-w1$  to  
 $m+w2$ .

See also [TriangularDist](#) , and [TriangularEnd3Dist](#).

## Asymmetric Triangular Range Distribution (cp dist for [equations](#))

Usage: TriangularEnd3Dist (x, m, a, b)

Definition:  $2(x-a)/((b-a)(m-a))$  for  $a \leq x \leq m$ ,  
 $2(b-x)/((b-a)(b-m))$  for  $m \leq x \leq b$

Required:  $a \leq m \leq b$      $a \neq b$

Support:  $a \leq x \leq b$

Moments:  $\mu = (a + b + m) / 3$   
 $\sigma^2 = (a^2 + b^2 + m^2 - ab - am - bm) / 18$

The `typeOf(BSPSPopupOnMouseOver) == 'function'`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_pdf.htm');return false;">`pdf has a triangular shape, with the highest point at  $x=m$ , and nonzero value from **a** to **b**.

See also [Triangular3Dist](#) , and [TriangularDist](#).

## Lognormal Distribution

(continuous probability dist. for [equations](#))

Usage: LognormalDist (x,  $\xi$ ,  $\phi$ )

Definition:  $N(\log(x), \xi, \phi) / x$   
 $= (1 / [x \phi \sqrt{2\pi}]) \exp(-[(\log(x) - \xi) / \phi]^2 / 2)$   
where N is the [normal distribution](#)

Required:  $\phi > 0$

Support:  $x > 0$

Moments:  $\mu = \exp(\xi + \phi^2 / 2)$   
 $\sigma^2 = \exp(2\xi + \phi^2) [\exp(\phi^2) - 1]$   
 $\gamma_1 = [\exp(\phi^2) + 2] \sqrt{\exp(\phi^2) - 1}$   
 $\beta_2 = \exp(4\phi^2) + 2 \exp(3\phi^2) + 3 \exp(2\phi^2)$

The lognormal distribution results when the logarithm of the random variable is described by a [normal distribution](#). This is often the case for a variable which is the product of a number of random variables (by the = 4 && `typeof(BSPSPopupOnMouseOver) == 'function'` `BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_central_limit_theorem.htm');return false;">central limit theorem).`

Notice that the 'n' of Lognormal is not capitalized, indicating that this is not the same as the logarithm of the normal distribution.

## Exponential Distribution

(continuous probability dist for [equations](#))

Usage: ExponentialDist (x,  $\lambda$ )

Definition:  $\lambda \exp(-\lambda x)$

Required:  $\lambda > 0$

Support:  $x \geq 0$

Moments:  $\mu = 1 / \lambda$      $\sigma = 1 / \lambda$   
 $\gamma_1 = 2$      $\beta_2 = 9$

If events occur by a `a = 4` && `typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Poisson_process.htm');return false;">Poisson`  
`process`, then the time between successive events is described by the exponential distribution (where  $\lambda$  is the average number of events per unit time).

## Gamma Distribution

(continuous probability dist for [equations](#))

Usage: GammaDist (x,  $\alpha$ ,  $\beta$ )

Definition:  $x^{\alpha-1} \exp(-x/\beta) / (\Gamma(\alpha) \beta^\alpha)$   
 $= \exp[(\alpha-1)\log(x) - x/\beta - \log(\Gamma(\alpha)) - \alpha \log(\beta)]$

Parameters:  $\alpha$  is shape  $\beta$  is scale

Required:  $\alpha > 0$   $\beta > 0$

Support:  $x \geq 0$

Moments:  $\mu = \alpha \beta$   $\sigma = \beta \sqrt{\alpha}$   
 $\gamma_1 = 2 / \sqrt{\alpha}$   $\gamma_2 = 3 + 6 / \alpha$

If events occur by a `a = 4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Poisson_process.htm');return false;">Poisson`  
`process`, then the time required for the occurrence of  $\alpha$  events is described by the gamma distribution (where  $\beta$  is the average time between events).

For  $\alpha = 1$ , this is the [exponential distribution](#) with  $\lambda = 1 / \beta$ . For  $\beta = 2$ , this is the [chi-square](#) distribution with degrees of freedom  $\nu = 2 \alpha$ .

The Erlang distribution is a special case of the gamma distribution in which  $\beta = 1$  and  $\alpha = n$  (which is an integer).

## Weibull Distribution

(continuous probability dist for [equations](#))

Usage: WeibullDist (x,  $\alpha$ ,  $\beta$ )

Definition:  $(\alpha/\beta) (x/\beta)^{(\alpha-1)} \exp(-(x/\beta)^\alpha)$

Parameters:  $\alpha$  is shape  $\beta$  is scale

Required:  $\alpha > 0$   $\beta > 0$

Support:  $x \geq 0$

Moments:  $\mu = \beta \text{gamma}(1 + 1/\alpha)$

$$\sigma^2 = \beta^2 [\text{gamma}(1 + 2/\alpha) - \text{gamma}(1 + 1/\alpha)^2]$$

$$\gamma_1 = [\text{gamma}(1+3/\alpha) - 3 \text{gamma}(1+1/\alpha) \text{gamma}(1+2/\alpha) + 2 \text{gamma}(1+1/\alpha)^3] / [\text{gamma}(1+2/\alpha) - \text{gamma}(1+1/\alpha)^2]^{3/2}$$

$$\beta_2 = [\text{gamma}(1+4/\alpha) - 4 \text{gamma}(1+1/\alpha) \text{gamma}(1+3/\alpha) + 6 \text{gamma}(1+1/\alpha)^2 \text{gamma}(1+2/\alpha) - 3 \text{gamma}(1+1/\alpha)^4] / [\text{gamma}(1+2/\alpha) - \text{gamma}(1+1/\alpha)^2]^2$$

The Weibull distribution is often used for reliability models, since if the failure rate of an item (i.e., percent of the remaining ones which fail, as a function of time) is given as:  $Z(t) = r t^{(\alpha-1)}$ , then the distribution of item lifetimes is given by the Weibull distribution with  $r = \alpha / \beta^\alpha$ .

## Generalized Beta Distribution

(cont prob. dist for [equations](#))

Usage: Beta4Dist (x, a, b, c, d)

Definition:  $Be((x - c) / (d - c), \alpha, \beta)$

where Be is the [beta distribution](#)

Parameters: c is lower endpoint d is upper endpoint

Required:  $\alpha > 0$   $\beta > 0$   $d > c$

$c \leq x \leq d$

This is a [beta distribution](#) that has been shifted and scaled, so that the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_pdf.htm');return false;">pdf has nonzero values from  $x=c$  to  $x=d$ , instead of from  $x=0$  to  $x=1$ . This distribution has great flexibility to fit almost any smooth, unimodal distribution with no tails (i.e., only nonzero over a finite range).

## Cauchy Distribution

(continuous probab dist for [equations](#))

Usage: CauchyDist (x,  $\mu$ ,  $\sigma$ )

Definition:  $1 / (\pi \sigma (1 + ((x-\mu)/\sigma)^2))$

Parameters:  $\mu$  = location     $\sigma$  = scale

Required:  $\sigma > 0$

Moments:  $\mu$  = undefined     $\sigma$  = undefined

Although real-world data rarely follows a Cauchy distribution, it is useful because of its unusualness. For example, although it is symmetric about  $\mu$  (which is therefore its median and mode), it doesn't have a mean (or variance, etc.) because the appropriate integrals don't converge. The C(0,1) distribution is also [Student's t](#) distribution with degrees of freedom = 1.

## Laplace Distribution

(continuous probability dist for [equations](#))

Usage: LaplaceDist (x,  $\mu$ ,  $\beta$ )

Definition:  $(1/(2\beta)) \exp(-|x-\mu|/\beta)$

Required:  $b > 0$

Support:  $-\infty < x < \infty$

Moments: mean =  $\mu$

$$\sigma^2 = 2\beta^2$$

$$\gamma_1 = 0 \quad \gamma_2 = 3$$

Its = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_pdf.htm');return false;">pdf is two [exponential distributions](#) spliced together back-to-back. The difference between two iid exponential distribution random variables follows a Laplace distribution.

Also known as the "double exponential" distribution.

## Extreme Value Distribution

(continuous probab. dist. for [equations](#))

Usage: ExtremeValueDist (x,  $\alpha$ ,  $\beta$ )

Definition:  $\exp(-\exp(-(x-\alpha)/\beta) - (x-\alpha)/\beta) / \beta$

Parameters:  $\alpha$  = location       $\beta$  = scale

Required:  $\beta > 0$

Moments:  $\mu = \alpha + \gamma \beta$       ( $\gamma$  = Eulers = 0.5772156649)

$$\sigma = \pi \beta / \text{sqrt}(6)$$

$$\gamma_1 = 1.3 \quad \beta_2 = 5.4$$

This distribution is the limiting distribution for the smallest or largest values in large samples drawn from a variety of distributions, including the normal distribution.

Also known as the "Fisher-Tippet distribution", "Fisher-Tippet Type I distribution" or the "log-Weibull distribution".

## Pareto Distribution

(continuous probab dist for [equations](#))

Usage: ParetoDist (x, a, b)

Definition:  $(a/b) (b/x)^{a+1}$

Parameters: a is shape b is location

Required:  $a > 0$   $b > 0$

Support:  $x \geq b$

Moments:  $\mu = ab / (a-1)$  for  $a > 1$  ( $\infty$  for  $a \leq 1$ )

$\sigma^2 = a b^2 / [(a-1)^2 (a-2)]$  for  $a > 2$

( $\infty$  if  $1 < a \leq 2$ , non-existent if  $a \leq 1$ )

$\gamma_1 = [2(1+a)/(a-3)] \sqrt{(a-2)/a}$  for  $a > 3$

$\beta_2 = 6(a^3 + a^2 - 6a - 2) / [a(a-3)(a-4)]$  for  $a > 4$

The Pareto distribution is a power law probability distribution found in a large number of real-world situations, such as the distribution of wealth among individuals, frequencies of words, size of particles, size of towns/cities, areas burnt in forest fires, size of some fractal features etc. These are situations where there are many that are small and a few that are large (like the Pareto principle, in which 20% of the population owns 80% of the wealth).

For any value of **a**, the distribution is "scale-free", which means that no matter what range of **x** one looks at, the proportion of small to large events is the same (i.e., the slope of the curve on any section of the log-log plot is the same).

Also known as the Bradford distribution.

## Chi-Square Distribution

(continuous probability dist for [equations](#))

Usage: ChiSquareDist (x, n)

Definition:  $x^{(v/2-1)} / [\exp (x/2) 2^{(v/2)} \text{gamma} (v/2)]$

Required:  $v > 0$   $v$  is an integer

Support:  $x \geq 0$

Moments:  $\mu = v$   $\sigma^2 = 2 v$   
 $\gamma_1 = 2 \text{sqrt} (2/v)$   $\beta_2 = 3 + 12 / v$

This is the distribution of  $Z_1^2 + Z_2^2 + \dots + Z_v^2$  where  $Z_i$  are independent [standard normal](#) variates.

$v$  is usually called the “degrees of freedom” of the distribution.

**Student's t-Distribution** (continuous probability dist for [equations](#))

Usage: StudentTDist (x, v)

Definition:  $\frac{\Gamma((v+1)/2)}{\sqrt{v\pi} \Gamma(v/2)} (1+x^2/v)^{-((v+1)/2)}$

Required:  $v > 0$

Support:  $-\infty < x < \infty$

Moments:  $m = 0$

$\sigma^2 = v / (v - 2)$  for  $v > 2$

$\gamma_1 = 0$  for  $v > 3$      $\beta_2 = 6 / (v - 4)$  for  $v > 4$

The t-distribution or Student's t-distribution arises in the problem of estimating the mean of a normally distributed population when the sample size is small.

## F-Distribution

(continuous probability dist for [equations](#))

Usage:  $\text{FDist}(x, v_1, v_2)$

Definition:  $[(v_1 x / (v_2 + v_1 x))^{(1/2)}] [(v_2 / (v_2 + v_1 x))^{(v_2/2)}] / [x \text{ beta}(v_1/2, v_2/2)]$   
where beta is the [beta function](#)

Required:  $v > 0 \quad v_2 > 0$

Support:  $x \geq 0$

Moments:  $\mu = v_2 / (v_2 - 2)$  for  $v > 2$

$\sigma^2 = 2 v_2^2 (v_1 + v_2 - 2) / [v_1 (v_2 - 2)^2 (v_2 - 4)]$  for  $v_2 > 4$

The ratio of two chi-squared variates  $X_1$  and  $X_2$ , each divided by their degrees of freedom:  $(X_1/v_1)/(X_2/v_2)$  follows an F-distribution.

## Single Point Distribution

(discrete probability dist for [equations](#))

Usage: SingleDist (k, c)

Definition:  $(k == c) ? 1 : 0$

Required: k and c are integers

Support:  $k = c$

Moments:  $\mu = c$        $\sigma = 0$   
 $\gamma_1 = \text{undefined}$     $\beta_2 = \text{undefined}$

The single point distribution indicates that  $\mathbf{k} = \mathbf{c}$ . The probability that  $\mathbf{k}$  is any other value is 0.

This is the discrete version of a dirac delta.

## Discrete Uniform Distribution

(probability dist for [equations](#))

Usage: DiscUniformDist (k, a, b)

Definition:  $1 / (b - a + 1)$

Required:  $b \geq a$  k, a, b are integers

Support:  $a \leq k \leq b$

Moments:  $\mu = (a + b) / 2$

$\sigma^2 = (b-a) (b-a+2) / 12$

$\gamma_1 = 0$

$\beta_2 = (3/5) [3 - 4 / [(b-a) (b-a+2)]]$

This distribution represents the situation where **k** has an equal probability of taking on any of the integer values from **a** to **b** inclusive (where **a** and **b** are integers). If **k** were continuous, then it would be a [continuous uniform distribution](#).

## Bernoulli Distribution

(discrete probability dist for [equations](#))

Usage: BernoulliDist (b, p)

Definition:  $b \in \{0, 1\}$  ;  $0 < p < 1$

Required:  $0 \leq p \leq 1$     b boolean

Moments:  $\mu = p$   
 $\sigma^2 = p(1 - p)$   
 $\gamma_1 = (1 - 2p) / \sqrt{p(1 - p)}$   
 $\beta_2 = 3 + [1 - 6p(1-p)] / [p(1-p)]$

This is the distribution for a single "Bernoulli trial", in which **p** is the probability of an outcome labeled "success" occurring. **b** is a `boolean` that is true if the "success" occurs. An example is flipping a coin and checking for the event of heads appearing.

## Binomial Distribution

(discrete probability dist. for [equations](#))

**Usage:** BinomialDist (k, n, p)

**Definition:** binomial (n, k)  $p^k (1-p)^{(n-k)}$

```
Definition of = 4 && typeof(BSPSPopupOnMouseOver) ==  
'function') BSPSPopupOnMouseOver(event);"  
class="BSSCPopup"  
onclick="BSSCPopup('X_PU_binomial_coefficient.htm');return  
false;">binomial
```

**Required:**  $0 \leq p \leq 1$   
k and n are integers

**Support:**  $0 \leq k \leq n$

**Moments:**  $\mu = n p$   
 $\sigma^2 = n p (1 - p)$   
 $\gamma_1 = (1 - 2 p) / \sqrt{n p (1 - p)}$   
 $\beta_2 = 3 + [1 - 6 p (1-p)] / [n p (1-p)]$

```
A = 4 && typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X_PU_Bernoulli_process.htm');return false;">binomial
```

**experiment** is a series of  $n$  independent trials, each with two possible outcomes (often labeled "success" and "failure"), with a constant probability,  $p$ , of success. The total number of successes,  $k$ , is given by the binomial distribution.

If there are more than two possible outcomes, use the [multinomial distribution](#).

If the sampling is without replacement, use the [hypergeometric](#) distribution

For large  $n$ , and  $p$  not too close to 0 or 1, the binomial distribution can be approximated by a normal distribution with mean  $\mu = n p$ , and variance =  $n p (1-p)$ . For large  $n$ , and  $p$  close to 0, it can be approximated by a Poisson distribution with parameter  $\lambda = n p$ . As  $n \rightarrow$  infinity these are the limiting distributions (providing  $p =$  constant in the normal case, and  $p \rightarrow 0$ ,  $n p =$  constant in the Poisson case).

## Poisson Distribution

(discrete probab dist for [equations](#))

Usage: PoissonDist (k,  $\mu$ )

Definition:  $\exp(-\mu) * \mu^k / k!$   
 $= \exp(-\mu + k * \log(\mu) - \log(k!))$

Required:  $\mu > 0$     k is an integer

Support:  $k \geq 0$

Moments:    mean =  $\mu$              $\sigma^2 = \mu$   
 $\gamma_1 = 1 / \sqrt{\mu}$      $\beta_2 = 3 + 1/\mu$

If events occur by a `a = 4 && typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Poisson_process.htm');return false;">Poisson`  
`process`, then the number of events that occur in a fixed time interval is described by the Poisson distribution (where  $\mu$  is the average number of events per unit time).

## Hypergeometric Distribution

(discrete probability dist. for [equations](#))

**Usage:** HypergeometricDist (k, n, s, N)

**Definition:**  $\frac{\text{binomial}(s, k) \text{binomial}(N-s, n-k)}{\text{binomial}(N, n)}$

Definition of `binomial` in `Mathematica`: `binomial[n_, k_] := If[k < 0 || k > n, 0, Product[1 - s/N, {s, 1, k}]]`

`binomial[n_, k_] := If[k < 0 || k > n, 0, Product[1 - s/N, {s, 1, k}]]`

`binomial[n_, k_] := If[k < 0 || k > n, 0, Product[1 - s/N, {s, 1, k}]]`

**Required:**  $N \geq 0$   $0 \leq n \leq N$   $0 \leq s \leq N$

k, N, n and s are integers

**Support:**  $0 \leq k \leq n$

**Moments:**  $\mu = n s / N$

$\sigma^2 = n s (1-s/N) (N-n) / [N (N-1)]$

$\gamma_1 = (N-2s) (N-2n) \sqrt{N-1} / [(N-2) \sqrt{n s (N-s) (N-n)}]$

$\beta_2 = N^2 (N-1) / [(N-2)(N-3) n s (N-s)(N-n)] [N (N+1) - 6n (N-n) + 3s (N-s) / N^2 [N^2 (n-2) - N n^2 + 6n (N-n)]]$

This provides the probability that there are **k** "successes" in a random sample of size **n**, selected (without replacement) from **N** items of which **s** are labeled "success" and **N-s** labeled "failure".

It is used in place of the [binomial distribution](#) for situations which sample without replacement.

## Negative Binomial Distribution

(discrete probability dist. for [equations](#))

**Usage:** NegBinomialDist (k, n, p)

**Definition:** binomial (n+k-1, k) p^n (1-p)^k  
Definition of `4` `&& typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_binomial_coefficient.htm');return`  
`false;">binomial.`

**Required:**  $0 \leq n$   $0 < p \leq 1$  k and n are integers

**Support:**  $0 \leq k$

**Moments:**  $\mu = n(1-p) / p$   
 $\sigma^2 = n(1-p) / p^2$   
 $\gamma_1 = (2-p) / \sqrt{n(1-p)}$   
 $\beta_2 = 3 + [p^2 + 6(1-p)] / (n(1-p))$

This is the distribution of the number of failures that occur in a sequence of trials before **n** successes have occurred, in a `4` `&& typeof(BSPSPopupOnMouseOver) == 'function')`  
`BSPSPopupOnMouseOver(event);" class="BSSCPopup"`  
`onclick="BSSCPopup('X_PU_Bernoulli_process.htm');return`  
`false;">Bernoulli process (independent trials, with outcomes labeled "success" or "failure", and constant probability p of success).`

The limit of a negative binomial distribution as  $n \rightarrow \infty$ ,  $(1-p) \rightarrow 0$ ,  $n(1-p) \rightarrow \lambda$ , is a [Poisson distribution](#) with parameter  $\lambda$ .

If **n** = 1, then this distribution is just the [geometric distribution](#).

## Geometric Distribution

(discrete probability dist. for [equations](#))

Usage: GeometricDist (k, p)

Definition:  $p (1-p)^k$

Required:  $0 < p \leq 1$      $k$  is an integer

Support:  $0 \leq k$

Moments:  $\mu = (1-p) / p$   
 $s^2 = (1-p) / p^2$   
 $\gamma_1 = (2-p) / \sqrt{1-p}$   
 $\beta_2 = 3 + [p^2 + 6(1-p)] / (1-p)$

This distribution describes the number of = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_Bernoulli\_process.htm');return false;">Bernoulli trials (independent trials, with outcomes labeled "success" or "failure", and constant probability **p** of success) before the first success occurs (i.e., includes only the failure trials). An example would be the number of coin flips resulting in tails before the first head is seen.

Situations where Bernoulli trials are repeated until the  $n$ th success are called "negative binomial experiments", and the geometric distribution is a special case of the [negative binomial distribution](#) with  $n = 1$ .

Note that the geometric distribution is instead defined by some authors to have the = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X\_PU\_pdf.htm');return false;">pdf:  $p (1-p)^{(k-1)}$

## Logarithmic Distribution

(discrete probability dist. for [equations](#))

Usage: LogarithmicDist (k, p)

Definition:  $-(p^k) / (k \log (1-p))$

Required:  $0 < p < 1$     k integer

Support:  $k \geq 1$

Moments:  $\mu = -p / [(1 - p) \log (1 - p)]$   
 $\sigma^2 = -p (p + \log (1 - p)) / [(1 - p) \log (1 - p)]^2$

Also known as the "logarithmic series distribution".

## Multinomial Distribution

(discrete probability dist. for [equations](#))

**Usage:** MultinomialDist (bc, n, k1, p1, k2, p2, ... km, pm)

**Required:**  $n \geq 0$   $k_i \geq 0$   $0 \leq p_i \leq 1$   $\sum p_i \neq 0$   
bc boolean n, ki integer

**Support:**  $n = \sum k_i$

**Mean:**  $E[k_i] = n p_i$

**Covariance:**  $\text{cov}[k_i, k_j] = -n p_i p_j$  if  $i \neq j$   
if  $i=j$  then  $\text{cov}[k_i, k_i] = \text{var}[k_i] = n p_i (1-p_i)$

The multinomial distribution is a generalization of the [binomial distribution](#) to the situation where there are not just two outcomes (usually labeled "success" and "fail"), but rather **m** outcomes, each having probability **pi** ( $i=1..m$ ), and we are interested in the number of occurrences of each outcome (**ki**), given that a total of **n** trials are performed.

To create a multinomial distribution between the **ki** and **n** nodes, first add to the net a new `a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_boolean_node.htm');return false;">boolean node`, in this example called **bc**. Then add links from the nodes of all the non-fixed parameters (usually **n** and all **ki**) to node **bc**. At node **bc**, put an equation with MultinomialDist, and convert the equation to a table. Finally, give node **bc** `a = 4 && typeof(BSPSPopupOnMouseOver) == 'function') BSPSPopupOnMouseOver(event);" class="BSSCPopup" onclick="BSSCPopup('X_PU_finding.htm');return false;">finding of true`.

Normally the sum of **pi** is one, but Netica will just normalize the **pi** if that is not the case.

If **m** is 2, then **k2** is deterministically determined by **k1** (i.e.,  $k_2 = n - k_1$ ), and **k1** is distributed by [BinomialDist](#).

Each of the **ki** separately has a binomial distribution with parameters **n** and **pi**, and because of the constraint that the sum of the **ki**'s is **n**, they are negatively

correlated.

The Dirichlet distribution is the conjugate prior of the multinomial in Bayesian statistics.

For assistance on using this function, = 4 &&  
typeof(BSPSPopupOnMouseOver) == 'function')  
BSPSPopupOnMouseOver(event);" class="BSSCPopup"  
onclick="BSSCPopup('X\_PU\_email.htm');return false;">contact our support  
team.

**Noisy-And Distribution** (DM prob. dist. for [equations](#))

**Usage:** NoisyAndDist (e, inh, b1, p1, ... bn, pn)

**Definition:**  $P(e) = (1-inh) \prod_{i=1}^n (b_i ? 1 : (1-p_i))$

**Required:**  $0 \leq inh \leq 1$  e, bi Boolean  $0 \leq p_i \leq 1$

Use this distribution when there are several possible requirements for an event, and each has a probability that it will actually be necessary. Each of the necessary requirements must pass for the event to occur. Even then, there is a probability (given by **inh**) that the event may not occur (make **inh** zero to eliminate this).

Each **bi** is a [= 4 && typeof\(BSPSPopupOnMouseOver\) == 'function'\) BSPSPopupOnMouseOver\(event\);" class="BSSCPopup" onclick="BSSCPopup\('X\\_PU\\_boolean\\_node.htm'\);return false;">boolean](#) variable, which when TRUE, indicates a requirement passed. **e** is also a boolean, which indicates whether the event occurs. Each of the **pi** are the probability that **bi** will be required to cause **e**.

If **inh** is zero, and only one possible requirement is FALSE, say **bk**, then the probability for **e** is  $1-p_k$ . If more possible requirements are FALSE, the probability will be lower. And if **inh** is nonzero, the probability will be lower. Reducing a **pi** always results in the same or higher **P(e)**.

**pi** can be considered the “strength” of the relation between **e** and **bi**, with zero indicating independence (link could be removed), and 1 indicating maximum effect.

See also [NoisyOrDist](#).

**Noisy-Max Distribution** (multivariate dist. for [equations](#))

See also [NoisyOrDist](#), [NoisySumDist](#).

For information on using this function, [contact Norsys](#), and ask for the document titled "Noisy Or, Max, Sum.doc".

**Noisy-Sum Distribution** (multivariate dist. for [equations](#))

See also [NoisyMaxDist](#).

For information on using this function, [contact Norsys](#) and ask for the document titled "Noisy Or, Max, Sum.doc".

*CSV file* is a commonly used term for a form of [case file](#) in which the names of the variables appear on the first line, and then below are all the cases (i.e. records), with each case on a single line and having a value for each of the variables, and with all the values and variables in [text](#) form and separated by commas (i.e. "Comma Separated Values"). See also [tab delimited text](#).

*Tab delimited text file* is a commonly used term for a form of [case file](#) in which the names of the variables appear on the first line, and then below are all the cases (i.e. records), with each case on a single line and having a value for each of the variables, and with all the values and variables in text form and separated by tab characters. See also [CSV file](#).

A *text file* consists only of ASCII characters. It has no special symbols, no formatting (bold, italics, fonts or sizes), no structure (paragraph sections, chapter sections, margins, etc.), and no special inserts (pictures, tables, etc.). You can create or modify a text file using a text editor.

*ASCII* is a text character encoding based on the English alphabet, and was first released as a standard in 1967. It represents each character with 7 bits, although there are many 8-bit extensions of it. ASCII and its extensions have been by far the dominant way to represent text in a computer, but are now being overtaken by Unicode, which can represent many more characters. In Netica, identifiers (i.e. IDnames) are always composed only of ASCII characters, while titles and descriptions may be in Unicode.

## Logistic Distribution

(continuous probab dist for [equations](#))

Usage: LogisticDist (x,  $\mu$ ,  $\sigma$ )

Definition:  $\exp(-(x-\mu)/\sigma) / (\sigma * (1 + \exp(-(x-\mu)/\sigma))^2)$

Parameters:  $\mu$  = location     $\sigma$  = scale

Required:  $\sigma > 0$

Moments: Mean = Median = Mode =  $\mu$

variance =  $(\pi * \sigma)^2 / 3$

kurtosis = 6/5

entropy =  $\log(\sigma) + 2$

It resembles the [normal distribution](#) in shape but has heavier tails (higher kurtosis). Its cumulative distribution function (cdf) is the logistic function.