Contrasts for lm glm lme and geeglm objects

Description

Computes linear functions (i.e. weighted sums) of the estimated regression parameters.

Can also test the hypothesis, that such a function is equal to a specific value.

Usage

```
esticon(obj, cm, beta0, conf.int = TRUE, level=0.95, joint.test = FA
```

Arguments

`obj`	Regression object (of type lm, glm, lme, geeglm)
`cm`	Matrix specifying linear functions of the regresson parameters (one linear function per row). The number of columns must match the number of fitted regression parameters in the model. See 'details' below.
`beta0`	A vector of numbers
`conf.int`	TRUE
`level`	The confidence level
`joint.test`	Logical value. If TRUE a 'joint' Wald test for the hypothesis L beta=beta0 is made. Default is that the 'row-wise' tests are made, i.e. (L beta)i=beta0i. If joint.test is TRUE, then no confidence inteval etc. is calculated.

Details

Let the estimated parameters of the model be

$$\beta\_1, \beta\_2, ...,\beta\_p$$

A linear function of the estimates is of the form

$$c=\lambda\_1 \beta\_1+\lambda\_2 \beta\_2+ ...+\lambda\_p \beta\_p$$

where $\lambda\_1, \lambda\_2, ...,\lambda\_p$ is specified by the user.

The esticon function calculates c, its standard error and by default also a 95 pct confidence interval. It is sometimes of interest to test the hypothesis *H_0: c=β_0* for some value *β_0* given by the user. A test is provided for the hypothesis *H_0: c=0* but other values of *β_0* can be specified.

In general, one can specify r such linear functions at one time by speficying cm to be an *rtimes p* matrix where each row consists of p numbers *λ_1,λ_2,..., λ_p*. Default is then that *β_0* is a p vector of 0s but other values can be given.

It is possible to test simulatneously that all speficied linear functions are equal to the corresponding values in *β_0*.

For computing contrasts among levels of a single factor, 'contrast.lm' may be more convenient.

Value

Returns a matrix with one row per linear function. Columns contain estimated coefficients, standard errors, t values, degrees of freedom, two-sided p-values, and the lower and upper endpoints of the 1-alpha confidence intervals.

Note

'esticon' works on geese/geeglm objects from the geepack package (for Generalized Estimating Equations), on 'lm' and 'glm' objects, and on 'gls' objects.

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

Examples

```
data(iris)
lm1  <- lm(Sepal.Length~Sepal.Width+Species+Sepal.Width:Species, dat
## Note that the setosa parameters are set to zero
coef(lm1)

## Estimate the intercept for versicolor
lambda1 <- c(1,0,1,0,0,0)
esticon(lm1,lambda1)

## Estimate the difference between versicolor and virgica intercept
## and test if the difference is 1
```

```
lambda2 <- c(0,1,-1,0,0,0)
esticon(lm1,lambda2,beta0=1)

## Do both estimates at one time
esticon(lm1,rbind(lambda1,lambda2),beta0=c(0,1))

## Make a combined test for that the difference between versicolor a
## and difference between versicolor and virginica slope is zero:
lambda3 <- c(0,0,0,0,1,-1)
esticon(lm1,rbind(lambda2,lambda3),joint.test=TRUE)

# Example using esticon on coxph objects (thanks to Alessandro A. Le
# Using dataset 'veteran' in the survival package
# from the Veterans' Administration Lung Cancer study

library(survival);
data(veteran)
sapply(veteran,class)
levels(veteran$celltype)
attach(veteran)
veteran.s<-Surv(time,status)
coxmod<-coxph(veteran.s~age+celltype+trt,method='breslow')
summary(coxmod)

# compare a subject 50 years old with celltype 1
# to a subject 70 years old with celltype 2
# both subjects on the same treatment
AvB<-c(-20,-1,0,0,0)

# compare a subject 40 years old with celltype 2 on treat=0
# to a subject 35 years old with celltype 3 on treat=1
CvB<-c(5,1,-1,0,-1)

esti<-esticon(coxmod,rbind(AvB,CvB))
esti
exp(esti[,c(2,7,8)])
```

Groupwise computations of summary statistics, general linear contrasts and other utilities

Help pages for package ‘doBy’ version 4.0.5

budworm	Effect of Insecticide on survivial of tobacco budworms
codstom	Diet of Atlantic cod in the Gulf of St. Lawrence (Canada)
dietox	Growth curves of pigs in a 3x3 factorial experiment
doBy	Various utilities which includes functions for creating groupwise calculations etc.
dose.LD50	Calculate LD50
esticon	Contrasts for lm, glm, lme, and geeglm objects
esticon.coxph	Contrasts for lm, glm, lme, and geeglm objects
esticon.geeglm	Contrasts for lm, glm, lme, and geeglm objects
esticon.glm	Contrasts for lm, glm, lme, and geeglm objects
esticon.gls	Contrasts for lm, glm, lme, and geeglm objects
esticon.lm	Contrasts for lm, glm, lme, and geeglm objects
esticon.lme	Contrasts for lm, glm, lme, and geeglm objects
esticon.mer	Contrasts for lm, glm, lme, and geeglm objects
firstobs	Locate the index of the first/last unique value
firstobs.default	Locate the index of the first/last unique value
firstobs.formula	Locate the index of the first/last unique value
lapplyBy	Formula based version of lapply
lastobs	Locate the index of the first/last unique value
lastobs.default	Locate the index of the first/last unique value
lastobs.formula	Locate the index of the first/last unique value
orderBy	Ordering (sorting) rows of a data frame
print.splitByData	Split a data frame
recodevar	recode levels of variable
sampleBy	Sampling from a data frame
splitBy	Split a data frame
subsetBy	Finds subsets of a dataframe which is split by variables in a formula.
summaryBy	Function to calculate groupwise summary statistics
transformBy	Function to make groupwise transformations
which.maxn	Where are the n largest or n smallest elements in a numeric

	vector ?
which.minn	Where are the n largest or n smallest elements in a numeric vector ?

Effect of Insecticide on survivial of tobacco budworms

Description

Number of killed budworms after exposure to an insecticide.

Usage

```
data(budworm)
```

Format

This data frame contains 12 rows and 4 columns:

sex:
: sex of the budworm

dose:
: dose of the insecticide trans-cypermethrin in *micro g*

ndead:
: budworms killed in a trial

ntotal:
: total number of budworms exposed per trial

Details

Mortality of the moth tobacco budworm 'Heliothis virescens' for 6 doses of the pyrethroid trans-cypermethrin differentiated with respect to sex.

Source

Collet, D. (1991) Modelling Binary Data, Chapman & Hall, London, Example 3.7

References

Venables, W.N; Ripley, B.D.(1999) Modern Applied Statistics with S-Plus, Heidelberg, Springer, 3rd edition, chapter 7.2

Examples

```
data(budworm)
## function to caclulate the empirical logits
empirical.logit<- function(nevent,ntotal) {
  y<-log ((nevent+0.5)/(ntotal-nevent+0.5))
  y
}

## plot the empirical logits against log-dose

log.dose  <- log(budworm$dose)
emp.logit <- empirical.logit(budworm$ndead,budworm$ntotal)
plot(log.dose,emp.logit,type='n',xlab='log-dose',ylab='emprirical lo
title('budworm: emprirical logits of probability to die ')

male   <- budworm$sex=='male'
female <- budworm$sex=='female'
lines(log.dose[male],emp.logit[male],type='b',lty=1,col=1)
lines(log.dose[female],emp.logit[female],type='b',lty=2,col=2)
legend(0.5,2,legend=c('male','female'),lty=c(1,2),col=c(1,2))
```

Diet of Atlantic cod in the Gulf of St. Lawrence (Canada)

Description

Stomach content data for Atlantic cod (Gadus morhua) in the Gulf of St.Lawrence, Eastern Canada. Note: many prey items were of no interest for this analysis and were regrouped into the "Other" category.

Usage

```
data(codstom)
```

Format

A data frame with 10000 observations on the following 10 variables.

`region`
: a factor with levels `SGSL` `NGSL` representing the southern and northern Gulf of St. Lawrence, respectively

`ship.type`
: a factor with levels `2` `3` `31` `34` `90` `99`

`ship.id`
: a factor with levels `11558` `11712` `136148` `136885` `136902` `137325` `151225` `151935` `99433`

`trip`
: a factor with levels `10` `11` `12` `179` `1999` `2` `2001` `20020808` `3` `4` `5` `6` `7` `8` `88` `9` `95`

`set`
: a numeric vector

`fish.id`
: a numeric vector

`fish.length`
: a numeric vector, length in mm

`prey.mass`
: a numeric vector, mass of item in stomach, in g

`prey.type`
: a factor with levels `Ammodytes_sp` `Argis_dent` `Chion_opil` `Detritus` `Empty` `Eualus_fab` `Eualus_mac` `Gadus_mor` `Hyas_aran` `Hyas_coar` `Lebbeus_gro` `Lebbeus_pol` `Leptocl_mac` `Mallot_vil` `Megan_norv` `Ophiuroidea` `Other` `Paguridae` `Pandal_bor` `Pandal_mon` `Pasiph_mult` `Sabin_sept` `Sebastes_sp` `Them_abys` `Them_comp` `Them_lib`

Details

Cod are collected either by contracted commerical fishing vessels (`ship.type` 90 or 99) or by research vessels. Commercial vessels are identified by a unique `ship.id`.

Either one research vessel or several commercial vessels conduct a survey (`trip`), during which a trawl, gillnets or hooked lines are set several times. Most trips are random stratified surveys (depth-based stratification).

Each trip takes place within one of the `region`s. The `trip` label is only guaranteed to be unique within a region and the `set` label is only guaranteed to be unique within a `trip`.

For each fish caught, the `fish.length` is recorded and the fish is allocated a `fish.id`, but the `fish.id` is only guaranteed to be unique within a `set`. A subset of the fish caught are selected for stomach analysis (stratified random selection according to fish length; unit of stratification is the set for research surveys, the combination ship.id and stratum for surveys conducted by commercial vessels, although strata are not shown in codstom).

The basic experimental unit in this data set is a cod stomach (one stomach per fish). Each stomach is uniquely identified by a combination of `region`, `ship.type`, `ship.id`, `trip`, `set`, and `fish.id`.

For each prey item found in a stomach, the species and mass of the prey item are recorded, so there can be multiple observations per stomach. There may also be several prey items with the same `prey.type` in the one stomach (for example many `prey.types` have been recoded `Other`, which produced many instances of `Other` in the same stomach).

If a stomach is empty, a single observation is recorded with `prey.type` `Empty` and a `prey.mass` of zero.

Source

Small subset from a larger dataset (more stomachs, more variables, more `prey.types`) collected by D. Chabot and M. Hanson, Fisheries & Oceans Canada (chabotd@dfo-mpo.gc.ca).

Examples

```
data(codstom)
str(codstom)
# removes multiple occurences of same prey.type in stomachs
codstom1 <- summaryBy(prey.mass ~
                      region+ship.type+ship.id+trip+set+fish.id+prey
                      data = codstom, id = ~fish.length,
                      keep.names=TRUE, FUN = sum)

# keeps a single line per stomach with the total mass of stomach con
codstom2 <- summaryBy(prey.mass ~ region+ship.type+ship.id+trip+set+
                      data = codstom, id = ~fish.length,
                      keep.names=TRUE, FUN = sum)

# mean prey mass per stomach for each trip
codstom3 <- summaryBy(prey.mass ~ region+ship.type+ship.id+trip,
                      data = codstom2, keep.names=TRUE, FUN = mean)

## Not run:

# wide version, one line per stomach, one column per prey type
library(reshape)
codstom4 <- melt(codstom, id = c(1:7, 9))
codstom5 <- cast(codstom4,
                 region+ship.type+ship.id+trip+set+fish.id+fish.leng
                 prey.type, sum)
k <- length(names(codstom5))
prey_col <- 8:k
out <- codstom5[,prey_col]
out[is.na(out)] <- 0
codstom5[,prey_col] <- out
codstom5$total.content <- rowSums(codstom5[, prey_col])
## End(Not run)
```

Growth curves of pigs in a 3x3 factorial experiment

Description

The `dietox` data frame has 861 rows and 7 columns.

Data contains weight of slaughter pigs measured weekly for 12 weeks. Data also contains the startweight (i.e. the weight at week 1). The treatments are 3 different levels of Evit = vitamin E (dose: 0, 100, 200 mg dl-alpha-tocopheryl acetat /kg feed) in combination with 3 different levels of Cu=copper (dose: 0, 35, 175 mg/kg feed) in the feed. The cumulated feed intake is also recorded. The pigs are littermates.

Usage

```
data(dietox)
```

Format

This data frame contains the following columns:

Weight
: Weight

Feed
: Cumulated feed intake

Time
: Time (in weeks) in the experiment

Pig
: Id of each pig

Evit
: Vitamin E dose

Cu
: Copper dose

Start
: Start weight in experiment, i.e. weight at week 1.

Litter
: Id of litter of each pig

Source

Lauridsen, C., Højsgaard, S.,Sørensen, M.T. C. (1999) Influence of Dietary Rapeseed Oli, Vitamin E, and Copper on Performance and Antioxidant and Oxidative Status of Pigs. J. Anim. Sci.77:906-916

Examples

```
data(dietox)
str(dietox) ;
plot(dietox)
```

Various utilities which includes functions for creating groupwise calculations etc.

Description

The core doBy functions were developed to make it easy to split data into groups (defined by the levels of a set of factors) and performing some actions on each of these groups. Thus, these functions mimic what can be achieved using the BY statement in various SAS procedures.

In addition hereto the doBy package containts various other utilities.

Details

Functions summaryBy, splitBy, orderBy, sampleBy, transformBy are the core doBy functions

There is no need for a plotBy function – the xyplot function in the lattice package already fulfills these needs

The esticon function calculates linear functions of parameter estimates under various types of models.

There are various other utility functions in the package.

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

summaryBy, orderBy, transformBy, splitBy, sampleBy

Examples

```
data(dietox)

summaryBy(Weight+Feed~Evit+Cu+Time,      data=dietox, FUN=c(mean,var
na.rm=TRUE, use="pair")
```

```
orderBy(~Time+Evit, data=dietox)

splitBy(formula = ~Evit+Cu, data = dietox)

sampleBy(formula = ~Evit+Cu, frac=.1, data = dietox)
```

Calculate LD50

Description

Calculate the LD50 (the dose at which 50 pct of the subjects die) for a model of the form logit(p)=beta1 x1 + ... + betap xp + gamma d where none of the explanatory variables x1 ... xp contains the dose d.

Usage

```
dose.LD50(x, lambda)
```

Arguments

`x`	A glm object (for logistic regression)
`lambda`	A vector of the same length as the number of parameters in x.

Details

lambda contains an NA at the entry corresponding to dose d. The other entries of lambda must be the values of the covariates x1 ... xp at which the ld50 is to be calculated.

Value

A data frame

Author(s)

Søren Højsgaard

Examples

```
data(budworm)
m1 <- glm(ndead/20 ~ sex + log(dose), data=budworm, weight=ntotal, f
coef(m1)

dose.LD50(m1,c(1,1,NA))
dose.LD50(m1,c(1,0,NA))
```

Locate the index of the first/last unique value

Description

Locate the index of the first/last unique value in i) a vector or of a variable in a data frame.

Usage

```
## S3 method for class 'formula':
firstobs(formula, data=parent.frame(), ...)
## S3 method for class 'formula':
lastobs(formula, data=parent.frame(), ...)
firstobs(x, ...)
lastobs(x, ...)
```

Arguments

`x`	A vector
`formula`	A formula (only the first term is used, see 'details').
`data`	A data frame
`...`	Currently not used

Details

If writing ~a+b+c as formula, then only a is considered.

Value

A vector.

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

Examples

```
x <- c(rep(1,5),rep(2,3),rep(3,7),rep(1,4))

firstobs(x)
lastobs(x)
```

```
data(dietox)

firstobs(~Pig, data=dietox)
lastobs(~Pig, data=dietox)
```

Formula based version of lapply

Description

This function is a wrapper for calling lapply on the list resulting from first calling splitBy.

Usage

```
lapplyBy(formula, data = parent.frame(), FUN)
```

Arguments

`formula`	A formula describing how data should be split
`data`	A dataframe
`FUN`	A function to be applied to each element in the splitted list, see 'Examples' below.

Value

A list.

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

`orderBy`, `summaryBy`, `transformBy`, `splitBy`,

Examples

```
data(dietox)

## Calculate weekwise feed efficiency = weight gain / feed intake
dietox <- orderBy(~Pig+Time, data=dietox)
v<-lapplyBy(~Pig, data=dietox, function(d) c(NA, diff(d$Weight)/diff
dietox$FE <- unlist(v)

## Technically this is the same as
dietox <- orderBy(~Pig+Time, data=dietox)
wdata <- splitBy(~Pig, data=dietox)
v <- lapply(wdata, function(d) c(NA, diff(d$Weight)/diff(d$Feed)))
```

```
dietox$FE <- unlist(v)
```

Ordering (sorting) rows of a data frame

Description

Ordering (sorting) rows of a data frame by the certain variables in the data frame. This function is essentially a wrapper for the order() function - the important difference being that variables to order by can be given by a model formula.

Usage

```
orderBy(formula, data)
```

Arguments

`formula`	The right hand side of a formula
`data`	A data frame

Details

The sign of the terms in the formula determines whether sorting should be ascending or decreasing; see examples below

Value

The ordered data frame

Author(s)

Søren Højsgaard, sorenh@agrsci.dk and Kevin Wright

See Also

`summaryBy`, `transformBy`, `splitBy`, `lapplyBy`,

Examples

```
data(dietox)
orderBy(~Time+Evit, data=dietox)
## Sort decreasingly by Time
orderBy(~-Time+Evit, data=dietox)
```

Split a data frame

Description

Split a dataframe according to the levels of variables in the dataframe. The variables to split by can be given as a formula or as a character vector.

Usage

```
splitBy(formula, data = parent.frame(), drop=TRUE, return.matrix=FAL
## S3 method for class 'splitByData':
print(x, ...)
```

Arguments

`formula`	The right hand side of a formula (or a character vector)
`data`	A data frame
`drop`	Logical indicating if levels that do not occur should be dropped
`return.matrix`	Should the returned list consist of dataframs or matrices, see 'details' below
`x`	A splitByData object (basically a list).
`...`	Additional arguments, currently not used.

Details

The funcion transform the dataframe 'data' into a numerical matrix (using the 'asNumericMatrix' function from the Hmisc package) and makes the splitting operation on this. If return.matrix is TRUE, then these matrices are returned, otherwise the matrices are turned into dataframes and then these are returned.

Value

A list of dataframes of matrices

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

orderBy, summaryBy, transformBy, lapplyBy,

Examples

```
data(dietox)
splitBy(formula = ~Evit+Cu, data = dietox)
```

---

recode levels of variable

Description

Recodes a variable with levels, say '1','2' to a variable with levels, say 'a', 'b'

Usage

```
recodevar(var, src, tgt)
```

Arguments

`var`	The variable to be recoded
`src`	The source levels: the present levels of var
`tgt`	The target levels: the new levels of var

Value

A new variable

Examples

```
x <- c("dec","jan","feb","mar","apr","may")
src1 <- list(c("dec","jan","feb"), c("mar","apr","may"))
tgt1 <- list("winter","spring")
recodevar(x,src=src1,tgt=tgt1)
```

Sampling from a data frame

Description

A data frame is split according to some variables in a formula, and a sample of a certain fraction of each is drawn.

Usage

```
sampleBy(formula, frac = 0.1, replace=FALSE, data = parent.frame(),s
```

Arguments

`formula`	A formula defining the grouping of the data frame
`frac`	The part of data to be sampled.
`replace`	Is the sampling with replacement
`data`	A data frame
`systematic`	Should sampling be systematic.

Details

If systematic=FALSE (default) then frac gives the fraction of data sampled. If systematic=TRUE and frac=.2 then every 1/.2 i.e. every 5th observation is taken out.

Value

A data frame

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

`orderBy`, `summaryBy`, `transformBy`, `splitBy`,

Examples

```
data(dietox)
sampleBy(formula = ~Evit+Cu, frac=.1, data = dietox)
```

Finds subsets of a dataframe which is split by variables in a formula.

Description

A data frame is split by a formula into groups. Then subsets are found within each group, and the result is collected into a data frame.

Usage

```
subsetBy(formula, subset, data = parent.frame(), select, drop=FALSE,
join=TRUE, ... )
```

Arguments

`formula`	A formula to split by
`subset`	logical expression indicating elements or rows to keep: missing values are taken as false.
`data`	A data frame
`select`	expression, indicating columns to select from a data frame.
`drop`	passed on to [indexing operator.
`join`	If FALSE the result is a list of data frames (as defined by 'formula'); if TRUE one data frame is returned.
`...`	further arguments to be passed to or from other methods.

Value

A data frame.

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

See Also `splitBy`

Examples

```
data(dietox)
```

```
subsetBy(~Evit, Weight < mean(Weight), data=dietox)
```

Function to calculate groupwise summary statistics

Description

Function to calculate groupwise summary statistics, much like the summary procedure of SAS

Usage

```
summaryBy(formula, data = parent.frame(), id = NULL, FUN = mean,
keep.names=FALSE, p2d=FALSE, order=TRUE, ...)
```

Arguments

`formula`	A formula object, see examples below
`data`	A data frame
`id`	A formula specifying variables which data are not grouped by but which should appear in the output. See examples below.
`FUN`	A list of functions to be applied, see examples below.
`keep.names`	If TRUE and if there is only ONE function in FUN, then the variables in the output will have the same name as the variables in the input, see 'examples'.
`p2d`	Should parentheses in output variable names be replaced by dots?
`order`	Should the resulting dataframe be ordered according to the variables on the right hand side of the formula? (using orderBy
`...`	Additional arguments to FUN. This could for example be NA actions.

Details

Extra arguments ('...') are passed onto the functions in FUN. Hence care must be taken that all functions in FUN accept these arguments - OR one can explicitly write a functions which get around this. This can particularly be an issue in connection with handling NAs. See examples below.

Some code for this function has been suggested by Jim Robison-Cox.

Value

A data frame

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

orderBy, transformBy, splitBy, lapplyBy,

Examples

```
data(dietox)
dietox12    <- subset(dietox,Time==12)

summaryBy(Weight+Feed~Evit+Cu,      data=dietox12,
   FUN=c(mean,var,length))

summaryBy(Weight+Feed~Evit+Cu+Time, data=subset(dietox,Time>1),
   FUN=c(mean,var,length))

## Calculations on transformed data:

summaryBy(log(Weight)+Feed~Evit+Cu, data=dietox12)

## Calculations on all numerical variables (not mentioned elsewhere)

summaryBy(.~Evit+Cu,                data=dietox12,
   id=~Litter, FUN=mean)

## There are missing values in the 'airquality' data, so we remove t
## before calculating mean and variance with 'na.rm=TRUE'. However t
## length function does not accept any such argument. Hence we get
## around this by defining our own summary function in which length
## not supplied with this argument while mean and var are:

sumfun <- function(x, ...){
  c(m=mean(x, ...), v=var(x, ...), l=length(x))
}
summaryBy(Ozone+Solar.R~Month, data=airquality, FUN=sumfun, na.rm=TR

## Using '.' on the right hand side of a formula means to stratify b
## all variables not used elsewhere:
```

```
data(warpbreaks)
summaryBy(breaks ~ wool+tension, warpbreaks)
summaryBy(breaks ~., warpbreaks)
summaryBy(.~ wool+tension, warpbreaks)

## Keep the names of the variables (works only if FUN only returns o
## value):

summaryBy(Ozone+Wind~Month, data=airquality,FUN=c(mean),na.rm=TRUE,
  keep.names=TRUE)
```

Function to make groupwise transformations

Description

Function to make groupwise transformations of data by applying the transform function to subsets of data.

Usage

```
transformBy(formula, data, ...)
```

Arguments

`formula`	A formula with only a right hand side, see examples below
`data`	A data frame
`...`	Further arguments of the form tag=value

Details

The ... arguments are tagged vector expressions, which are evaluated in the data frame data. The tags are matched against names(data), and for those that match, the value replace the corresponding variable in data, and the others are appended to data.

Value

The modified value of the dataframe data.

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

`orderBy`, `summaryBy`, `splitBy`, `doby.xtabs`,

Examples

```
data(dietox)
transformBy(~Pig, data=dietox, minW=min(Weight), maxW=max(Weight),
```

```
gain=sum(range(Weight)*c(-1,1)))
```

Where are the n largest or n smallest elements in a numeric vector ?

Description

Determines the locations, i.e., indices of the n largest or n smallest elements of a numeric vector.

Usage

```
which.maxn(x, n = 1)
which.minn(x, n = 1)
```

Arguments

x	numeric vector
n	integer >= 1

Value

A vector of length at most n with the indices of the n largest / smaller elements. NAs are discared and that can cause the vector to be smaller than n.

Author(s)

Søren Højsgaard, sorenh@agrsci.dk

See Also

which.max, which.min

Examples

```
x <- c(1:4,0:5,11,NA,NA)
ii <- which.minn(x,5)

x <- c(1,rep(NA,10),2)
ii <- which.minn(x,5)
```

Internal functions for the doBy package

Description

Internal functions for the doBy package
