Package ‘doBy’

August 30, 2018

Version 4.6-2
Title Groupwise Statistics, LSmeans, Linear Contrasts, Utilities
Author Søren Højsgaard <sorenh@math.aau.dk> and Ulrich Halekoh
                <uhalekoh@health.sdu.dk>
Maintainer Søren Højsgaard <sorenh@math.aau.dk>
Description Contains:
        1) Facilities for working with grouped data: ‘do’ something to data
           stratified ‘by’ some variables.
        2) LSmeans (least-squares means), general linear contrasts.
        3) Miscellaneous other utilities.
Encoding UTF-8
VignetteBuilder knitr
URL http://people.math.aau.dk/~sorenh/software/doBy/
ZipData no
License GPL (>= 2)
Depends R (>= 3.2.0), methods
Imports MASS, Matrix, dplyr, plyr, magrittr
Suggests pbkrtest (>= 0.4-6), ggplot2, multcomp, geepack, lme4,
        survival, knitr
RoxygenNote 6.1.0
NeedsCompilation no
Repository CRAN
Date/Publication 2018-08-30 06:54:24 UTC

R topics documented:

     by-lapply ................................................................. 2
     by-order ................................................................. 3
     by-sample ............................................................... 4
     by-split ................................................................. 5
by-`lapply`

Formula based version of `lapply`.

**Description**

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

**Usage**

```r
lapplyBy(formula, data = parent.frame(), FUN)
```
by-order

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@math.au.dk>

See Also

lapplyBy

Examples

data(dietox)

```r
## 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$Weight) / diff(d$Feed)))
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)
```

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 dataframe.

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@math.aau.dk> and Kevin Wright.

See Also

- `transformBy`, `splitBy`

Examples

```r
orderby(~ conc + Treatment, CO2)
## Sort decreasingly by conc
orderby(~ ~ conc + Treatment, CO2)
```

---

**by-sample**

*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

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

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.
by-split

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 dataframe.

See Also
summaryBy, orderBy, splitBy, transformBy

Examples

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

by-split  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

```r
splitBy(formula, data = parent.frame(), drop = TRUE)
```

Arguments

- `formula` Variables to split data frame by, as `as.quoted` variables, a formula or character vector.
- `data` A data frame
- `drop` Logical indicating if levels that do not occur should be dropped. Deprecated; levels that do not occur are ignored.
- `...` Additional arguments, currently not used.

Value
A list of dataframes.

Author(s)
Søren Højsgaard, <sorenh@math.aau.dk>
See Also

orderBy, summaryBy, transformBy

Examples

data(dietox, package="doBy")
splitBy(formula = ~Evit + Cu, data = dietox)
splitBy(formula = c("Evit", "Cu"), data = dietox)

splitBy(~Month, data=airquality)
splitBy("Month", data=airquality)

by-subset  
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 right hand sided formula or a character vector of variables 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@math.aau.dk>
See Also

splitBy

Examples

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

Description

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

Usage

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

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`)
- **full.dimension**: If TRUE then rows of summary statistics are repeated such that the result will have the same number of rows as the input dataset.
- **var.names**: Option for user to specify the names of the variables on the left hand side.
- **fun.names**: Option for user to specify function names to apply to the variables on the left hand side.
- **...**: 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. Thanks.

Value

A dataframe.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

ave, descStat, orderBy, splitBy, transformBy

Examples

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

fun <- function(x){
  c(m=mean(x), v=var(x), n=length(x))
}

summaryBy(cbind(Weight, Feed) ~ Evit + Cu, data=dietox12, FUN=fun)

summaryBy(list(c("Weight", "Feed"), c("Evit", "Cu")), data=dietox12, FUN=fun)

## Computations on several variables is done using cbind( )
summaryBy(cbind(Weight, Feed) ~ Evit + Cu, data=subset(dietox, Time > 1), FUN=fun)

## Calculations on transformed data is possible using cbind( ), but
## the transformed variables must be named
summaryBy(cbind(lw=log(Weight), Feed) ~ Evit + Cu, data=dietox12, FUN=mean)

## There are missing values in the 'airquality' data, so we remove these
## before calculating mean and variance with 'na.rm=TRUE'. However the
## length function does not accept any such argument. Hence we get
## around this by defining our own summary function in which length is
## not supplied with this argument while mean and var are:
sumfun <- function(x, ...){
  c(m=mean(x, na.rm=TRUE, ...), v=var(x, na.rm=TRUE, ...), l=length(x))
}
by-transform

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@math.aau.dk>

See Also

orderBy, summaryBy, splitBy
Examples

data(dietox)
transformBy(~Pig, data=dietox, minW=min(Weight), maxW=max(Weight),
gain=diff(range(Weight)))

Description

Yield and sugar percentage in sugar beets from a split plot experiment. Data is obtained from a split plot experiment. There are 3 blocks and in each of these the harvest time defines the "whole plot" and the sowing time defines the "split plot". Each plot was \(25m^2\) and the yield is recorded in kg. See 'details' for the experimental layout.

Usage

beets

Format

The format is: chr "beets"

Details

Experimental plan
Sowing times
1 4. april
2 12. april
3 21. april
4 29. april
5 18. may
Harvest times
1 2. october
2 21. october
Plot allocation:

<table>
<thead>
<tr>
<th>Plot</th>
<th>1 1 1 1 1</th>
<th>2 2 2 2 2</th>
<th>1 1 1 1 1</th>
<th>Harvest time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-15</td>
<td>3 4 5 2 1</td>
<td>3 2 4 5 1</td>
<td>5 2 3 4 1</td>
<td>Sowing time</td>
</tr>
</tbody>
</table>
Plot | 2 2 2 2 2 | 1 1 1 1 1 | 2 2 2 2 2 | Harvest time |
16-30 | 2 1 5 4 3 | 4 1 3 2 5 | 1 4 3 2 5 | Sowing time  |
**data-breastcancer**

**References**


**Examples**

```r
data(beets)

beets$bh <- with(beets, interaction(block, harvest))
summary(aov(yield ~ block + sow + harvest + Error(bh), beets))
summary(aov(sugpct ~ block + sow + harvest + Error(bh), beets))
```

---

**data-breastcancer**

*Gene expression signatures for p53 mutation status in 250 breast cancer samples*

**Description**

Perturbations of the p53 pathway are associated with more aggressive and therapeutically refractory tumours. We preprocessed the data using Robust Multichip Analysis (RMA). Dataset has been truncated to the 1000 most informative genes (as selected by Wilcoxon test statistics) to simplify computation. The genes have been standardised to have zero mean and unit variance (i.e. z-scored).

**Usage**

`breastcancer`

**Format**

A data frame with 250 observations on 1001 variables. The first 1000 columns are numerical variables; the last column (named `code`) is a factor with levels `case` and `control`.

**Details**

The factor `code` defines whether there was a mutation in the p53 sequence (`code=case`) or not (`code=control`).

**Source**

Dr. Chris Holmes, c.holmes at stats dot ox dot ac dot uk

**References**

Examples

data(breastcancer)
bc <- breastcancer
pairs(bc[1:5], col=bc$code)

train <- sample(1:nrow(bc), 50)
table(bc$code[train])
library(MASS)
z <- lda(code ~ ., data=bc, prior = c(1,1)/2, subset = train)
pc <- predict(z, bc[-train, ])$class
pc
bc[-train, "code"]
table(pc, bc[-train, "code"])

---

data-budworm  
budworm data

Description

Effect of Insecticide on survival of tobacco budworms number of killed budworms exposed to an insecticide and mortality of the moth tobacco budworm 'Heliothis virens' for 6 doses of the pyrethroid trans-cypermethrin differentiated with respect to sex

Usage

budworm

Format

This data frame contains 12 rows and 4 columns:

sex: sex of the budworm
dose: dose of the insecticide trans-cypermethrin in [µg]
ndead: budworms killed in a trial
ntotal: total number of budworms exposed per trial

Source


References

data-carcass

Examples

data(budworm)

## function to calculate the empirical logits
empirical.logit<- function(y, n) {
  el <- log((y + 0.5) / (n - y + 0.5))
  el
}

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

## Not run:
* SAS example;
data budworm;
infile 'budworm.txt' firstobs=2;
input sex dose ndead ntotal;
run;

## End(Not run)

---

**data-carcass**

*Lean meat contents of 344 pig carcasses*

Description

Measurement of lean meat percentage of 344 pig carcasses together with auxiliary information collected at three Danish slaughter houses

Usage

carcass

Format

carcassall: A data frame with 344 observations on the following 17 variables.
weight  Weight of carcass
lengthc  Length of carcass from back toe to head (when the carcass hangs in the back legs)
lengthf  Length of carcass from back toe to front leg (that is, to the shoulder)
lengthp  Length of carcass from back toe to the pelvic bone
Fat02, Fat03, Fat11, Fat12, Fat13, Fat14, Fat16 Thickness of fat layer at different locations on the back of the carcass (FatXX refers to thickness at (or rather next to) rib no. XX. Notice that 02 is closest to the head
Meat11, Meat12, Meat13 Thickness of meat layer at different locations on the back of the carcass, see description above
LeanMeat  Lean meat percentage determined by dissection
slhouse  Slaughter house; a factor with levels a b c
sex  Sex of the pig; a factor with a b c. Notice that it is no an error to have three levels; the third level refers to castrates

Note
carcass: Contains only the variables Fat11, Fat12, Fat13, Meat11, Meat12, Meat13, LeanMeat

Source

Examples
data(carcass)
head(carcass)

Data-codstom  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
codstom
**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 Leptoc1_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 commercial 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 regions. 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.type,
  data = codstom,
  FUN = sum)

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

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

data-crimeRate

---

crimeRate

Description

Crime rates per 100,000 inhabitants in states of the USA for different crime types.

Usage

crimeRate
**data-dietox**

**Format**

This data frame contains:

- **State**: State of the USA
- **Murder**: crime of murder
- **Rape**: 
- **Robbery**: 
- **Assault**: 
- **Burglary**: residential theft
- **Larceny**: unlawful taking of personal property (pocket picking)
- **AutoTheft**: 

**Examples**

```r
data(crimeRate)
```

| data-dietox | Growth curves of pigs in a 3x3 factorial experiment |

**Description**

The dietox data frame has 861 rows and 7 columns.

**Usage**

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
Details
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.

Source

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

---

**data-haldCement**

*Heat development in cement under hardening.*

Description
Heat development in cement under hardening related to the chemical composition.

Usage
```
haldCement
```

Format
A data frame with 13 observations on the following 5 variables.

- x1 Percentage (weight) of [3Ca0][Al2O3]
- x2 Percentage (weight) of [3Ca0][SiO2]
- x3 Percentage (weight) of [4Ca0][Al2O3][Fe03]
- x4 Percentage (weight) of [2Ca0][SiO2]
- y Heat development measured in calories per gram cement after 180 days

References
Anders Hald (1949); Statistiske Metoder; Akademisk Forlag (in Danish), page 509.
Examples

data(haldCement)

if( interactive() ){
pairs( haldCement )
}
m <- lm( y ~ x1 + x2 + x3 + x4, data=haldCement )
summary( m )

# Notice: The model explains practically all variation in data;
# yet none of the explanatory variables appear to be statistically
# significant...

---

data-milkman  Milk yield data for manually milked cows.

Description

Milk yield data for cows milked manually twice a day (morning and evening).

Usage

milkman

Format

A data frame with 161836 observations on the following 12 variables.

cowno  a numeric vector; cow identification
lactno a numeric vector; lactation number
ampm  a numeric vector; milking time: 1: morning; 2: evening
dfc   a numeric vector; days from calving
my    a numeric vector; milk yield (kg)
fatpct a numeric vector; fat percentage
protpct a numeric vector; protein percentage
lactpct a numeric vector; lactose percentage
scce  a numeric vector; somatic cell counts
race  a factor with levels RDM Holstein Jersey
ecmy  a numeric vector; energy corrected milk
cowlact Combination of cowno and lactno; necessary because the same cow may appear more
            than once in the dataset (in different lactations)
Details

There are data for 222 cows. Some cows appear more than once in the dataset (in different lactations) and there are 288 different lactations.

References

Friggens, N. C.; Ridder, C. and Løvendahl, P. (2007). On the Use of Milk Composition Measures to Predict the Energy Balance of Dairy Cows. J. Dairy Sci. 90:5453–5467 doi:10.3168/jds.2006-821. This study was part of the Biosens project used data from the “Mælkekoens energibalance og mobilisering” project; both were funded by the Danish Ministry of Food, Agriculture and Fisheries and the Danish Cattle Association.

Examples

data(milkman)

data(NIRmilk)

Description

Near infra red light (NIR) measurements are made at 152 wavelengths on 17 milk samples. While milk runs through a glass tube, infra red light is sent through the tube and the amount of light passing through the tube is measured at different wavelengths. Each milk sample was additionally analysed for fat, lactose, protein and dry matter.

Usage

NIRmilk

Format

This data frame contains 18 rows and 158 columns. The first column is the sample number. The columns Xwww contains the infra red light amount at wavelength www. The response variables are fat, protein, lactose and dm (dry matter).

Details

PCA regression

Examples

data(NIRmilk)
data-potatoes

Weight and size of 20 potatoes

Description

Weight and size of 20 potatoes. Weight in grams; size in milimeter. There are two sizes: length is the longest length and width is the shortest length across a potato.

Usage

potatoes

Format

A data frame with 20 observations on the following 3 variables.

weight  a numeric vector
length  a numeric vector
width  a numeric vector

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

Source

My own garden; autumn 2015.

Examples

data(potatoes)
plot(potatoes)

descStat

Computing simple descriptive statistics of a numeric vector.

Description


Usage

descStat(x, na.rm = TRUE)
**Arguments**

- `x` A numeric vector
- `na.rm` Should missing values be removed

**Value**

A vector with named elements.

**Author(s)**

Gregor Gorjanc; gregor.gorjanc <at> bf.uni-lj.si

**See Also**

summaryBy

**Examples**

```r
x <- c(1, 2, 3, 4, NA, NaN)
descStat(x)
```

---

**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**

```r
esticon(obj, L, beta0, conf.int = TRUE, level = 0.95, joint.test = FALSE, ...)
```

## S3 method for class 'gls'

```r
esticon(obj, L, beta0, conf.int = TRUE, level = 0.95, joint.test = FALSE, ...)
```

## S3 method for class 'geeglm'

```r
esticon(obj, L, beta0, conf.int = TRUE, level = 0.95, joint.test = FALSE, ...)
```

## S3 method for class 'lm'

```r
esticon(obj, L, beta0, conf.int = TRUE, level = 0.95, joint.test = FALSE, ...)
```
## Arguments

- **obj** Regression object (of type lm, glm, lme, geeglm)
- **L** Matrix (or vector) specifying linear functions of the regression 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 $L \beta = \beta_0$ is made. Default is that the 'row-wise' tests are made, i.e. $(L \beta)_i = \beta_0 i$. If joint.test is TRUE, then no confidence interval etc. is calculated.
- **...** Additional arguments; currently not used.
- **object** An esticon_class object.

## Details

Let the estimated parameters of the model be

$$\beta_1, \beta_2, \ldots, \beta_p$$

A linear function of the estimates is of the form

$$l = \lambda_1 \beta_1 + \lambda_2 \beta_2 + \ldots + \lambda_p \beta_p$$

where $\lambda_1, \lambda_2, \ldots, \lambda_p$ is specified by the user.
The `esticon` function calculates $l$, its standard error and by default also a 95 pct confidence interval. It is sometimes of interest to test the hypothesis $H_0 : l = \beta_0$ for some value $\beta_0$ given by the user. A test is provided for the hypothesis $H_0 : l = 0$ but other values of $\beta_0$ can be specified.

In general, one can specify $r$ such linear functions at one time by specifying $L$ to be an $r \times p$ matrix where each row consists of $p$ numbers $\lambda_1, \lambda_2, \ldots, \lambda_p$. Default is then that $\beta_0$ is a $p$ vector of 0s but other values can be given.

It is possible to test simultaneously that all specified linear functions are equal to the corresponding values in $\beta_0$.

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.

### Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

### Examples

```r
data(iris)
lm1 <- lm(Sepal.Length ~ Sepal.Width + Species + Sepal.Width : Species, data=iris)
  ## 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, L=lambda1)

  ## Estimate the difference between versicolor and virginica intercept
  ## and test if the difference is 1
  lambda2 <- c(0, 1, -1, 0, 0, 0)
esticon(lm1, L=lambda2, beta0=1)

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

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

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

if (require(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)
est <- esticon(coxmod, L=rbind(AvB, CvB))
est
##exp(est[, c(2, 7, 8)])
```

---

### fatacid

**Fish oil in pig food**

**Description**

...

**Usage**

fatacid

**Format**

...

**Details**

A fish oil fatty acid X14 has been added in different concentrations to the food for pigs in a study. Interest is in studying how much of the fatty acid can be found in the tissue. The concentrations of x14 in the food are `verb+dose+={0.0, 4.4, 6.2, 9.3}`.

The pigs are fed with this food until their weight is 60 kg. From thereof and until they are slaughtered at 100 kg, their food does not contain the fish oil. At 60 kg (sample=1) and 100 kg (sample=2) muscle biopsies are made and the concentration of x14 is determined. Measurements on the same pig are correlated, and pigs are additionally related through litters.
References

Data courtesy of Charlotte Lauridsen, Department of Animal Science, Aarhus University, Denmark.

firstlastobs

| firstlastobs | 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

lastobs(x, ...)

firstobs(x, ...)

## Default S3 method:
lastobs(x, ...)

## Default S3 method:
firstobs(x, ...)

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

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

Arguments

x  A vector

... Currently not used

formula A formula (only the first term is used, see 'details').

data A data frame

Details

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

Value

A vector.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>
is-estimable

Examples

```r
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)
```

is-estimable  Determines if contrasts are estimable.

Description

Determines if contrasts are estimable, that is, if the contrasts can be written as a linear function of
the data.

Usage

```r
is_estimable(K, null.basis)
```

Arguments

- `K`  
  A matrix.
- `null.basis`  
  A basis for a null space (can be found with null_basis()).

Details

Consider the setting $E(Y) = Xb$. A linear function of $b$, say $l'b$ is estimable if and only if there exists an $r$ such that $r'X = l'$ or equivalently $l = X'r$. Hence $l$ must be in the column space of $X'$, i.e. in the orthogonal complement of the null space of $X$. Hence, with a basis $B$ for the null space, is_estimable() checks if each row $l$ of the matrix $K$ is perpendicular to each column basis vector in $B$.

Value

A logical vector.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>
References


See Also

null_basis

Examples

### TO BE WRITTEN

---

linest-get

Auxillary functions for computing lsmeans, contrasts etc.

Description

Auxillary functions for computing lsmeans, contrasts etc.

Usage

get_xlevels(obj)

## Default S3 method:
get_xlevels(obj)

## S3 method for class 'mer'
get_xlevels(obj)

## S3 method for class 'merMod'
get_xlevels(obj)

get_contrasts(obj)

## Default S3 method:
get_contrasts(obj)

## S3 method for class 'merMod'
get_contrasts(obj)

set_xlevels(xlev, at)

get_vartypes(obj)

set_covariate_val(xlev, covariateVal)
get_X(obj, newdata, at = NULL)

## Default S3 method:
get_X(obj, newdata, at = NULL)

## S3 method for class 'merMod'
get_X(obj, newdata, at = NULL)

Arguments

- **obj**: An R object
- **xlev**:FIXME: to be described
- **at**:FIXME: to be described
- **covariateVal**:FIXME: to be described
- **newdata**:FIXME: to be described

linest-matrix  
*Linear estimates matrix*

Description

Generate matrix specifying linear estimate.

Usage

LE_matrix(object, effect = NULL, at = NULL)

## Default S3 method:
LE_matrix(object, effect = NULL, at = NULL)

aggregate_linest_list(lel)

gelinest_list(object, effect = NULL, at = NULL)

Arguments

- **object**: Model object
- **effect**: A vector of variables. For each configuration of these the estimate will be calculated.
- **at**: A list of values of covariates (including levels of some factors) to be used in the calculations
- **lel**: Linear estimate list (as generated by get_linest_list).

Details

Check this
See Also

lsmeans, linest

Examples

## Two way anova:

data(warpbreaks)

## An additive model
m0 <- lm(breaks ~ wool + tension, data=warpbreaks)

## Estimate mean for each wool type, for tension="M":
K <- LE_matrix(m0, at=list(wool=c("A", "B"), tension="M"))
K

## Vanilla computation:
K %*% coef(m0)

## Alternative; also providing standard errors etc:
linest(m0, K)
esticon(m0, K)

## Estimate mean for each wool type when averaging over tension;
# two ways of doing this
K <- LE_matrix(m0, at=list(wool=c("A", "B")))
K
K <- LE_matrix(m0, effect="wool")
K
linest(m0, K)

## The linear estimate is sometimes called to "least squares mean"
# (LSmeans) or popputation means.
# Same as
LSmeans(m0, effect="wool")

## Without mentioning 'effect' or 'at' an average across all
#predictors are calculated:
K <- LE_matrix(m0)
K
linest(m0, K)

## Because the design is balanced (9 observations per combination
#of wool and tension) this is the same as computing the average. If
#the design is not balanced, the two quantities are in general not
#the same.
mean(warpbreaks$breaks)

## Same as
LSmeans(m0)
ls-means

## An interaction model
m1 <- lm(breaks ~ wool * tension, data=warpbreaks)

K <- LE_matrix(m1, at=list(wool=c("A", "B"), tension="M"))
K
linest(m1, K)
K <- LE_matrix(m1, at=list(wool=c("A", "B")))
K
linest(m1, K)
K
linest(m1, effect="wool")
K
LSmeans(m1, effect="wool")

K <- LE_matrix(m1)
K
linest(m1, K)
LSmeans(m1)

---

### ls-means

Compute LS-means (aka population means or marginal means)

---

**Description**

LS-means (least squares means, also known as population means and as marginal means) for a range of model types.

**Usage**

LSmeans(object, effect = NULL, at = NULL, level = 0.95, ...)

## Default S3 method:
LSmeans(object, effect = NULL, at = NULL,
level = 0.95, ...)

## S3 method for class 'lmerMod'
LSmeans(object, effect = NULL, at = NULL,
level = 0.95, adjust.df = TRUE, ...)

popMeans(object, effect = NULL, at = NULL, level = 0.95, ...)

## Default S3 method:
popMeans(object, effect = NULL, at = NULL,
level = 0.95, ...)

## S3 method for class 'lmerMod'
popMeans(object, effect = NULL, at = NULL,
level = 0.95, adjust.df = TRUE, ...)
Arguments

object     Model object
effect     A vector of variables. For each configuration of these the estimate will be calculated.
at        A list of values of covariates (including levels of some factors) to be used in the calculations
level     The level of the (asymptotic) confidence interval.
...     Additional arguments; currently not used.
adjust.df  Should denominator degrees of freedom be adjusted?

Details

There are restrictions on the formulas allowed in the model object. For example having \( y \sim \log(x) \) will cause an error. Instead one must define the variable \( \log x = \log(x) \) and do \( y \sim \log x \).

Value

A dataframe with results from computing the contrasts.

Warning

Notice that \texttt{lsmeans} and \texttt{LE_matrix} fails if the model formula contains an offset (as one would have in connection with e.g. Poisson regression. It is on the todo-list to fix this.

Note

The \texttt{LSmeans} method is a recent addition to the package, and it will eventually replace the \texttt{popMeans} method.

Some of the code has been inspired by the \texttt{lsmeans} package.

Author(s)

Søren Højsgaard, <sorenh@math.au.dk>

See Also

\texttt{LE_matrix}, \texttt{linest}

Examples

```r
## Two way anova:

data(warpbreaks)

m0 <- lm(breaks ~ wool + tension, data=warpbreaks)
m1 <- lm(breaks ~ wool * tension, data=warpbreaks)
```
ls-means

LSmeans(m0)
LSmeans(m1)

## same as:
K <- LE_matrix(m0); K
linest(m0, K)
K <- LE_matrix(m1); K
linest(m1, K)

LE_matrix(m0, effect="wool")
LSmeans(m0, effect="wool")

LE_matrix(m1, effect="wool")
LSmeans(m1, effect="wool")

LE_matrix(m0, effect=c("wool", "tension"))
LSmeans(m0, effect=c("wool", "tension"))

LE_matrix(m1, effect=c("wool", "tension"))
LSmeans(m1, effect=c("wool", "tension"))

## Regression; two parallel regression lines:
data(Puromycin)
m0 <- lm(rate ~ state + log(conc), data=Puromycin)
## Can not use LSmeans / LE_matrix here because of
## the log-transformation. Instead we must do:
Puromycin$conc <- log(Puromycin$conc )
m1 <- lm(rate ~ state + I(conc, data=Puromycin)

LE_matrix(m1)
LSmeans(m1)

LE_matrix(m1, effect="state")
LSmeans(m1, effect="state")

LE_matrix(m1, effect="state", at=list(I(conc=3))
LSmeans(m1, effect="state", at=list(I(conc=3))

## Non estimable contrasts

## ## Make balanced dataset
dat.bal <- expand.grid(list(AA=factor(1:2), BB=factor(1:3),
CC=factor(1:3)))
dat.bal$y <- rnorm(nrow(dat.bal))

## ## Make unbalanced dataset
# 'BB' is nested within 'CC' so BB=1 is only found when CC=1
# and BB=2,3 are found in each CC=2,3,4
dat.nst <- dat.bal
dat.nst$CC <- factor(c(1, 1, 2, 2, 2, 2, 1, 1, 3, 3,
null-basis

3, 3, 1, 1, 4, 4, 4, 4)

mod.bal <- lm(y ~ AA + BB * CC, data=dat.bal)
mod.nst <- lm(y ~ AA + BB : CC, data=dat.nst)

LSmeans(mod.bal, effect=c("BB", "CC"))
LSmeans(mod.nst, effect=c("BB", "CC"))
LSmeans(mod.nst, at=list(BB=1, CC=1))

LSmeans(mod.nst, at=list(BB=1, CC=2))
## Above: NA's are correct; not an estimable function

if( require( lme4 )){
  warp.mm <- lmer(breaks ~ -1 + tension + (1|wool), data=warpbreaks)
  LSmeans(warp.mm, effect="tension")
  class(warp.mm)
  fixef(warp.mm)
  coef(summary(warp.mm))
  vcov(warp.mm)
  if (require(pbkrtest))
    vcovAdj(warp.mm)
}

LSmeans(warp.mm, effect="tension")

null-basis

Finds the basis of the (right) null space.

Description

Finds the basis of the (right) null space of a matrix, a vector (a 1-column matrix) or a model object for which a model matrix can be extracted. I.e. finds basis for the (right) null space x : Mx = 0.

Usage

null_basis(object)

Arguments

object A matrix, a vector (a 1-column matrix) or a model object for which a model matrix can be extracted (using model.matrix).

Value

A matrix (possibly with zero columns if the null space consists only of the zero vector).

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>
See Also

Null

Examples

\[
M \leftarrow \text{matrix(c(1,1,1,1,1,0,0,0,0,1,1), nrow=4)}
\]
null_basis(M)
mass::null(t(M))

\[
M \leftarrow \text{c(1,1,1,1)}
\]
null_basis(M)
mass::null(t(M))

\[
m0 \leftarrow \text{lm(breaks ~ wool + tension, data=warpbreaks)}
\]
null_basis(m0)
mass::null(t(model.matrix(m0)))

## Make balanced dataset

dat.bal <- expand.grid(list(A=factor(1:2), B=factor(1:3), C=factor(1:3)))
dat.bal$y <- rnorm(nrow(dat.bal))

## Make unbalanced dataset: 'B' is nested within 'C' so B=1 is only
## found when C=1 and B=2,3 are found in each C=2,3,4

dat.nst <- dat.bal
dat.nst$C <- factor(c(1,1,2,2,2,1,1,3,3,3,3,1,1,4,4,4,4))
xtabs(y ~ C+B+A, data=dat.nst)

mod.bal <- lm(y ~ A + B*C, data=dat.bal)
mod.nst <- lm(y ~ A + B*C, data=dat.nst)

null_basis( mod.bal )
null_basis( mod.nst )

null_basis( model.matrix(mod.bal) )
null_basis( model.matrix(mod.nst) )

mass::null( t(model.matrix(mod.bal)) )
mass::null( t(model.matrix(mod.nst)) )
Usage

```
parseGroupFormula(form)
```

Arguments

- `form`: A formula of the form `y ~ x1 + ... + xn | g1 + ... + gm`

Value

If the formula is `y ~ x1 + x2 | g1 + g2` the result is

```
model   y ~ x1 + x2
groups  g1 + g2
groupFormula ~ g1 + g2
```

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

Examples

```
gf <- parseGroupFormula(y ~ x1 + x2 | g1 + g2)
gf
```

---

**recodeVar**

`Recode values of a vector`

Description

Recodes a vector with values, say 1,2 to a variable with values, say 'a', 'b'

Usage

```
recodeVar(x, src, tgt, default = NULL, keep.na = TRUE)
```

Arguments

- `x`: A vector; the variable to be recoded
- `src`: The source values: a subset of the present values of `x`
- `tgt`: The target values: the corresponding new values of `x`
- `default`: Default target value for those values of `x` not listed in 'src'. When default=NULL, values of `x` which are not given in 'src' will be kept in the output.
- `keep.na`: If TRUE then NA's in `x` will be retained in the output
Value

A vector

Warning

Care should be taken if `x` is a factor. A safe approach may be to convert `x` to a character vector using `as.character`.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

cut, factor, recodeVar

Examples

```r
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)
# [1] "winter" "winter" "winter" "spring" "spring" "spring"

x <- c(rep(1:3, 3))
# [1] 1 2 3 1 2 3

# Simple usage:
recodeVar(x, src=c(1, 2), tgt=c("A", "B"))
# [1] "A" "B" NA "A" "B" NA "A" "B" NA

# Here we need to use lists
recodeVar(x, src=list(c(1, 2)), tgt=list("A"))
recodeVar(x, src=list(c(1, 2)), tgt=list("A"), default="L")
# [1] "A" "A" "L" "A" "A" "L" "A" "A" "L"
recodeVar(x, src=list(c(1, 2)), tgt=list("A", "B"), default="L")
# [1] "A" "A" "B" "A" "A" "B" "A" "A" "B"

# Dealing with NA's in x
x<-c(NA,rep(1:3, 3),NA)
# [1] NA 1 2 3 1 2 3 1 2 3 NA
recodeVar(x, src=list(c(1, 2)), tgt=list("A"))
recodeVar(x, src=list(c(1, 2)), tgt=list("A"), default="L")
recodeVar(x, src=list(c(1, 2)), tgt=list("A"), default="L", keep.na=FALSE)
# [1] "L" "A" "A" "A" "A" "L" "A" "A" "L" "L"

x <- c("no", "yes", "not registered", "no", "yes", "no answer")
recodeVar(x, src = c("no", "yes"), tgt = c("0", "1"), default = NA)
```
renameCol

Rename columns in a matrix or a dataframe.

Description

Rename columns in a matrix or a dataframe.

Usage

renameCol(indata, src, tgt)

Arguments

- `indata`: A dataframe or a matrix
- `src`: Source: Vector of names of columns in 'indata' to be renamed. Can also be a vector of column numbers.
- `tgt`: Target: Vector with corresponding new names in the output.

Value

A dataframe if 'indata' is a dataframe; a matrix in 'indata' is a matrix.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

Examples

```r
renameCol(CO2, 1:2, c("kk", "ll"))
renameCol(CO2, c("Plant", "Type"), c("kk", "ll"))
```

# These fail - as they should:
# renameCol(CO2, c("Plant", "Type", "conc"), c("kk", "ll"))
# renameCol(CO2, c("Plant", "Type", "Plant"), c("kk", "ll"))
subSeq

Find sub-sequences of identical elements in a vector.

Description

Find sub-sequences of identical elements in a vector.

Usage

subSeq(x, item = NULL)

Arguments

x 
An atomic vector.

item 
Optionally a specific value to look for in 'x'.

Value

A dataframe.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

rle

Examples

x <- c(1, 1, 1, 0, 0, 1, 1, 1, 2, 2, 2, 1, 2, 2, 2, 3)
(ans <- subSeq(x))
ans$value
# Notice: Same results below
subSeq(x, item=1)
subSeq(x, item="1")

x <- as.character(c(1, 1, 1, 0, 0, 1, 1, 1, 2, 2, 2, 1, 2, 2, 2, 3))
(ans<-subSeq(x))
ans$value
# Notice: Same results below
subSeq(x, item="1")
subSeq(x, item=1)
Summary: Compute linear estimates

Description

Compute linear estimates for a range of models. One example of linear estimates is population means (also known as LSMEANS).

Usage

```r
## S3 method for class 'esticon_class'
summary(object, ...)
linest(object, L = NULL, level = 0.95, ...)
## S3 method for class 'linest_class'
coef(object, ...)
## S3 method for class 'linest_class'
summary(object, ...)
## S3 method for class 'linest_class'
confint(object, parm, level = 0.95, ...)
```

Arguments

- `object`: Model object
- `...`: Additional arguments; currently not used.
- `L`: Either NULL or a matrix with p columns where p is the number of parameters in the systematic effects in the model. If NULL then L is taken to be the p times p identity matrix.
- `level`: The level of the (asymptotic) confidence interval.
- `parm`: Specification of the parameters estimates for which confidence intervals are to be calculated.

Value

A dataframe with results from computing the contrasts.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

LSmeans, LE_matrix
Examples

```r
## Make balanced dataset
dat.bal <- expand.grid(list(AA=factor(1:2), BB=factor(1:3), CC=factor(1:3)))
dat.bal$y <- rnorm(nrow(dat.bal))

## Make unbalanced dataset
# 'BB' is nested within 'CC' so BB=1 is only found when CC=1
# and BB=2,3 are found in each CC=2,3,4
dat.nst <- dat.bal
dat.nst$CC <- factor(c(1,1,2,2,2,2,1,1,3,3,3,3,1,1,4,4,4))

mod.bal <- lm(y ~ AA + BB * CC, data=dat.bal)
mod.nst <- lm(y ~ AA + BB : CC, data=dat.nst)
L <- LE_matrix(mod.nst, effect=c("BB", "CC"))
linest( mod.nst, L )
```
which.maxn

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.

Note

NA’s in yvar are converted to zeros.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

subseq, rle

Examples

```r
## Events:
yvar <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
          0, 0, 0, 0, 1, 1, 0, 0, 0, 0)

## Plot results:
tse <- timeSinceEvent(yvar)
plot(sign.tse-tvar, data=tse, type="b")
grid()
rug(tse$tvar[tse$yvar==1], col=4, lwd=4)
points(scale(tse$run), col=tse$run, lwd=2)
lines(abs.tse + .2 - tvar, data=tse, type="b", col=3)

## Find times for which time since an event is at most 1:
tse$tvar[tse$abs<=1]

yvar <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
tvar <- c(207, 208, 208, 208, 209, 209, 209, 209, 210, 210, 211, 211,
          211, 212, 213, 213, 214, 214, 215, 216, 216, 216, 217, 217,
          217, 218, 218, 219, 219, 219, 219, 220, 220, 221, 221, 221,
          221, 222, 222, 222)
timeSinceEvent(yvar, tvar)
```
**which.maxn**

**Usage**

```r
which.maxn(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 discarded and that can cause the vector to be smaller than n.

**Author(s)**

Søren Højsgaard, <sorenh@math.aau.dk>

**See Also**

[which.max](#), [which.min](#)

**Examples**

```r
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)
```
## Index

*Topic **datasets**
- data-beets, 10
- data-breastcancer, 11
- data-budworm, 12
- data-carcass, 13
- data-codstom, 14
- data-crimeRate, 16
- data-dietox, 17
- data-haldCement, 18
- data-milkman, 19
- data-NIRmilk, 20
- data-potatoes, 21
- fatacid, 25

*Topic **univar**
- by-summary, 7
- by-transform, 9

*Topic **utilities**
- by-lapply, 2
- by-order, 3
- by-sample, 4
- by-split, 5
- by-subset, 6
- descStat, 21
- esticon, 22
- firstlastobs, 26
- is-estimable, 27
- linest-matrix, 29
- ls-means, 31
- null-basis, 34
- parseGroupFormula, 35
- recodeVar, 36
- subSeq, 39
- summary.esticon_class, 40
- timeSinceEvent, 41
- which.maxn, 42

*Topic **utilities**
- renameCol, 38

aggregate_linest_list (linest-matrix), 29

ave, 8

beets (data-beets), 10
breastcancer (data-breastcancer), 11
budworm (data-budworm), 12
by-lapply, 2
by-order, 3
by-sample, 4
by-split, 5
by-subset, 6
by-summary, 7
by-transform, 9
carcass (data-carcass), 13
carcassall (data-carcass), 13
codstom (data-codstom), 14
coeff.esticon_class (esticon), 22
coeff.linest_class
  (summary.esticon_class), 40
coint.linest_class
  (summary.esticon_class), 40
crimeRate (data-crimeRate), 16
cut, 37
data-beets, 10
data-breastcancer, 11
data-budworm, 12
data-carcass, 13
data-codstom, 14
data-crimeRate, 16
data-dietox, 17
data-haldCement, 18
data-milkman, 19
data-NIRmilk, 20
data-potatoes, 21
descStat, 8, 21
dietox (data-dietox), 17
esticon, 22
factor, 37
INDEX

fatacid, 25
firstlastobs, 26
firstobs (firstlastobs), 26
get_contrasts (linest-get), 28
get_linest_list (linest-matrix), 29
get_vartypes (linest-get), 28
get_X (linest-get), 28
get_xlevels (linest-get), 28

haldCement (data-haldCement), 18
is-estimable, 27
is_estimable (is-estimable), 27
lapplyBy, 3
lapplyBy (by-lapply), 2
lastobs (firstlastobs), 26
LE_matrix, 32, 40
LE_matrix (linest-matrix), 29
linest, 30, 32
linest (summary.esticon_class), 40
linest-get, 28
linest-matrix, 29
ls-means, 31
LSmeans, 30, 40
LSmeans (ls-means), 31

milkman (data-milkman), 19

NIRmilk (data-NIRmilk), 20
Null, 35
null-basis, 34
null_basis, 28
null_basis (null-basis), 34

orderBy, 5–9
orderBy (by-order), 3

parseGroupFormula, 35
popMeans (ls-means), 31
potatoes (data-potatoes), 21

recodeVar, 36, 37
renameCol, 38
rle, 39, 42

sampleBy (by-sample), 4
set_covariate_val (linest-get), 28
set_xlevels (linest-get), 28

splitBy, 4, 5, 7–9
splitBy (by-split), 5
subsetBy (by-subset), 6
summary.esticon_class, 40
summary.linest_class
  (summary.esticon_class), 40
summaryBy, 5, 6, 9, 22
summaryBy (by-summary), 7
timeSinceEvent, 41
transformBy, 4–6, 8
transformBy (by-transform), 9

which.max, 43
which.maxn, 42
which.min, 43
which.minn (which.maxn), 42