Package 'epicalc'

November 4, 2024

Version 4.0.0.0

Date 2023-10-31

Title Epidemiological Calculator

Author Virasakdi Chongsuvivatwong <cvirasak@gmail.com>

Maintainer Kittisakdi Choomalee <kittisak.c@psu.ac.th>

Depends R (>= 4.1.0), foreign, survival, MASS, nnet

Description Functions making R easy for epidemiological calculation.

License GPL (>= 2)

URL https://medipe.psu.ac.th/epicalc/

Repository medipe

Date/Publication 2023-02-09 13:09:00 UTC

NeedsCompilation no

LazyData yes

Imports readxl, knitr, kableExtra

Contents

`addMissingRecords` *Add missing records to a longitudinal data set*

Description

Add missing records to a longitudinal data set, complete the fixed parts of covariates those missings and add a variable to indicate whether the subject was present in that schedule visit

Usage

```
addMissingRecords(dataFrame = .data, id, visit, outcome, check.present = TRUE,
    present.varname = "present", update.visit.related.vars = TRUE)
```

Arguments

dataFrame	Source data frame
id	identification variable
visit	index visit
outcome	outcome variable(s)
check.present	whether a new variable should be added to indicate the presence of the subject in the particular visit
present.varname	name of the new variable indicating the presence of the subject
update.visit.related.vars	whether visit related variables among the added records should be updated

Details

This function is used with a longitudinal data set where id, visit and outcome must be specified.

If there is any duplicated visit, the function will prompt error and stop.

The records of missing visits are added together with the fixed non-outcome covariates (which do not change over time in each subject) filled up. By default, a new variable "present" is annexed on the last column of the output data frame.

Like all other Epicalc data management functions, variable descriptions are kept.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'fillin'

Examples

```
data(bacteria, package="MASS")
des(bacteria)
head(bacteria, 10) # week 6 X01 and week 4 of X02 were missing
addMissingRecords(dataFrame=bacteria, id=ID, visit=week, outcome=y) -> bacteria.new
head(bacteria.new, 10)
# Note that the missing weeks are now added 'ap', 'hilo' and 'trt' which are fixed to id
# were automatically updated.
# A variable 'present' is also added.
# Columns are reordered to have ID and week leading others variables
rm(bacteria.new)

data(Xerop)
Xerop$time[500:501] <- 5:6  # Correct the error in the dataset
Xerop[1:25,] # Note Record 19 & 20, id 121140 had only two visits
Xerop.new <- addMissingRecords(dataFrame=Xerop, id=id, visit=time, outcome=xerop)
des(Xerop.new)
Xerop.new[19:24,]
rm(Xerop.new)
# Note that 4 new records where this subject missed the followup were added.
# Id relatee variable ie. 'sex', and visit related variable ie. 'season' are updated
# and 'present; is addeded
```

adjust *Adjusted and standardized mean, proportion and rate*

Description

Computation of adjusted or standardized mean, proportion and rate after generalized linear modelling

Usage

```
adjust(adjust = NULL, by, model, standard=NULL, offset=FALSE,
type = c("response", "link"), se.fit=TRUE, alpha=.05,
ci=FALSE,  ...)
```

Arguments

adjust	expression, indicating independent variable(s) of the model to be adjusted for
by	a list of elements for the grouping variables. The elements of the list will be coerced to factors (if they are not already factors).
model	object of class 'glm' on which the adjustment computation is based
standard	a vector controlling standard values for coefficients in the model
offset	whether the predict results will include the offset term of the model being used for prediction
type	the type of prediction required. The default is "response", which uses the scale of the original response variable. For a binomial model, the default estimates are predicted probabilities. For a Poisson model the default estimates are predicted incidence rates if 'offset=FALSE', and predicted counts otherwise. The alternative "link" transforms the estimates back to the same scale as the linear predictor.
se.fit	whether standard errors to the linear predictors should be returned
alpha	significance level
ci	whether the confidence intervals should be computed
...	additional arguments passed on to other methods

Details

Crude means, proportions and rates among different groups are not readily suitable for comparison among subgroups due to potential confounding. Generalized linear modelling (glm) handles potential confounding and provides coefficients indicating the level of difference between the specific group and the referent group after adjustment for other independent variables in the model.

The current function 'adjust' adds on information to an existing 'glm' model. It gives predicted values of subgroups specified in 'by' list. The returned predicted values, if type="response", reflect the magnitude (of mean, proportion and rate) of each subgroup after adjustment for the variable(s) specified in the 'adjust' argument.

The estimated values are based on each adjusted variable being equal to its grand mean value in the dataset of the model. Variables not included in the 'adjust' argument are set to mean values of each subgroup.

Standardization is meant to fix the variable(s) with value(s) specified in the vector 'standard' instead of subgroup mean (when 'adjust' is NULL) or grand mean (when 'adjust' is specified. If there is any conflict between 'adjust' and 'standard', the latter will override the former.

For adjustment, simply give variable name(s) in the 'adjust' argument. For standardization, the argument must be a vector of the same length as the model coefficients less one (since the Intercept term is already standardized as 1). All elements of this vector except those to be standardized should be 'NA'.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com>>

See Also

'glm', 'predict.glm', 'confint'

Examples

```
library(MASS)
use(Cars93)
des()
model1 <- glm(Price ~ Origin + Horsepower + DriveTrain, family=gaussian)
table(Origin, DriveTrain)
# Crude mean price by Origin and DriveTrain
Table.crude.means <- tapply(Price, list(Origin, DriveTrain), mean)
# Adjusted mean price
adjust(Horsepower, list(Origin, DriveTrain), model=model1)
a <- adjust(Horsepower, list(Origin, DriveTrain), model=model1)
Table.adjusted.means <- xtabs(mean ~ Origin + DriveTrain, data=a)
# Compare crude means with adjusted means of subgroups
Table.crude.means
Table.adjusted.means

# Price by category of DriveTrain adjusted for Horsepower & Origina
adjust(c(Horsepower,Origin), list(DriveTrain), model=model1)

## Now for crude and adjusted probabilities of having manual transmission
manual <- Man.trans.avail =="Yes"
model2 <- glm(manual ~ Origin + Horsepower + DriveTrain, family=binomial)
Table.crude.probabilities <- tapply(manual, list(Origin, DriveTrain), mean)
adjust(Horsepower, by=list(Origin, DriveTrain), model = model2)
b <- adjust(Horsepower, list(Origin, DriveTrain), model = model2)
Table.adjusted.probabilities <- xtabs(probability ~ Origin + DriveTrain, data=b)
# What is the breakdown of probability of having manual transmission
# if all cars in each subgroup  have 180 horse power?
model2$coefficients # 'Horsepower' is the second variable.
c <- adjust(by=list(Origin, DriveTrain), model=model2, standard=c(NA,180,NA,NA))
Table.standardized.probabilities <- xtabs(probability ~ Origin + DriveTrain, data=c)

# Compare crude and adjusted probabilities
Table.crude.probabilities
Table.adjusted.probabilities
Table.standardized.probabilities

# Age-sex- standardized attack rate
data(Oswego)
```

```
use(Oswego)
sex <- factor(sex, labels=c("famale","male"))
pack()
tabpct(sex, ill, percent="row")
# Crude attack rate = 68.2 percent in males and 51.6 percent in females
agegr <- pyramid(age, sex, bin=30)$ageGroup
lr1 <- glm(ill ~ sex * agegr, family=binomial)
# Assuming a standard population have equal number of male and female
# and uniform distribution of agegr thus the probability is
# .5 in each sex, 1/2 in each agegr and 1/6  in each age-sex group.
lr1$coefficients # Coefficients of 'agegr' are 3 to 6
adjust(by=list(sex=sex), model=lr1, standard=c(.5,rep(1/3,2),rep(1/6,2)))
# Age- & sex- standardized attack rate=59.9 percent

zap()
data(Montana)
use(Montana)
agegr <- factor(agegr, labels=c("40-49","50-59","60-69","70-79"))
label.var(agegr, "age group")
period <- factor(period, labels=c("1938-1949","1950-1959","1960-1969","1970-1977"))
label.var(period, "period of working")
start <- factor(start, labels=c("pre 1925", "1925 and after"))
label.var(start, "starting year")
model3 <- glm(respdeath ~ agegr + period + start, offset=log(personyrs), family=poisson)
agg <- aggregate.data.frame(.data[,1:2], list(period=period, start=start), mean)
crude.count <- agg[,3]
Table.crude.count <- xtabs(respdeath ~ period + start, data=agg)
crude.personyrs <- agg[,4]
Table.personyrs <- xtabs(personyrs ~ period + start, data=agg)
crude.rate <- agg[,3]/agg[,4]
Table.crude.rate <- xtabs(crude.rate ~ period + start, data=agg)

adjust(adjust=agegr, by=list(period, start), model3)
c <- adjust(adjust=agegr, by=list(period, start), model3)
Table.adjusted.rate <- xtabs(rate ~ period + start, data=c)
d <- adjust(adjust=agegr, by=list(period, start), model3, offset=TRUE, ci=TRUE)
Table.adjusted.count <- xtabs(count ~ period + start, data=d)

# Compare crude and adjusted counts
Table.crude.count
Table.adjusted.count

# Compare crude and adjusted rates
Table.crude.rate
Table.adjusted.rate
```

`Age at marriage` *Dataset on age at marriage*

Description

This dataset contains data on age at first marriage of attendants at a workshop in 1997.

Usage

```
data(Marryage)
```

Format

A data frame with 27 observations on the following 7 variables.

`id` a numeric vector

`sex` a factor with levels `male` `female`

`birthyr` a numeric vector indicating year of birth

`educ` a factor with levels `bach-` `bachelor` `or` `higher`

`marital` a factor with levels `Single` `Married`

`maryr` a numeric vector indicating year of marriage

`endyr` a numeric vector indicating year of analysis

Examples

```
data(Marryage)
use(Marryage)
des()
```

`aggregate numeric` *Summary statistics of a numeric variable by group*

Description

Split the numeric variable into subsets, compute summary statistics for each, and return the results in a data frame.

Usage

```
## S3 method for class 'numeric'
aggregate(x, by, FUN=c("count","sum","mean","median","sd","se","min","max"),
na.rm=TRUE, length.warning=TRUE, ...)
```

Arguments

`x`	a numeric variable
`by`	a list of grouping elements, each as long as the variables in 'x'. Names for the grouping variables are provided if they are not given. The elements of the list will be coerced to factors (if they are not already factors).
`FUN`	scalar functions to compute the summary statistics which can be applied to all data subsets.
`na.rm`	whether missing values will be removed during the computation of the statistics.
`length.warning`	show warning if x has any missing values
`...`	additional arguments passed on to 'aggregate'

Details

This is the 'aggregate' method for objects inheriting from class 'numeric'.

If Epicalc is loaded, applying 'aggregate' to a numeric variable 'x' will call 'aggregate.numeric'. If 'x' is a data frame, 'aggregate.data.frame' will be called.

If the Epicalc package is not loaded, 'aggregate', from the stats package, coerces numeric variables (including 'ts' objects) into a data frame and calls 'aggregate.data.frame'.

The 'FUN' argument in 'aggregate.data.frame' can accept only one function.

'aggregate.numeric' takes a different approach. More than one function can be suppplied to the 'FUN' argument, however it can only be applied to one numeric variable.

'aggregate' in Epicalc is 'backward compatible' with the 'aggregate' function from the stats package. In other words, Epicalc users do not need to change basic syntax or arguments. However, the naming system of the returned object is slightly different. In addition to the ability to provide more statistics in one command, another useful feature of 'aggregate.numeric' in Epicalc is the default values of FUN. Without typing such an argument, 'aggregate.numeric' gives commonly wanted statistics in a shorter line of command.

Note that 'na.rm' set to TRUE by default to allow computation of descriptive statistics such as 'mean', and 'sd', when they are in the FUN argument, and 'length' is computed with missing records included. In standard R functions, the equivalent argument is '"na.rm"=TRUE'.

The default value of the argument 'length.warning' is TRUE. A condition where 'x' has any missing value will be noticed, which is useful during data exploration. In further analysis, after missing values have been recognized, users may change 'length.warning' to FALSE to make the output look nicer. Both 'na.rm' and 'length.,warning' will have no effect if there are no missing values in x.

'count' is an additional function specific to 'aggregate.numeric'. It displays the number of non-missing records in each subgroup.

'aggregate.plot' makes use of the above function in drawing bar plots with error lines computed from 'aggregate.numeric'. When 'FUN="mean"', the automactic choice of error values is "se". Users can also choose "sd" or "ci". 'alpha' is effective only for 'error="ci"'. If 'FUN="median"', the error values are inter-quartile range.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'aggregate', 'summ' and 'tapply'

Examples

```
data(Compaq)
use(Compaq)

## If 'x' is a data frame, the default S3 aggregate method from the stats package is called.
aggregate(data.frame(id,year), by=list(HOSPITAL=hospital, STAGE=stage),
FUN="mean")
# The two additional columns are means of 'id' and 'year'

## If 'x' is a numeric vector, 'aggregate.numeric' from Epicalc package is called.
aggregate(year, by = list(HOSPITAL = hospital, STAGE = stage),
FUN = mean)
```

```
# The above command is the same as the one below.
# However, note the difference in the name of the last column of the returned
# data frame.
aggregate.data.frame(year, by = list(HOSPITAL = hospital,
STAGE = stage), FUN = mean)

# aggregate in Epicalc can handle multiple functions
aggregate(year, by = list(HOSPITAL = hospital, STAGE = stage),
FUN = c("mean", "sd", "length"))

## Handling of missing values
.data$year[8] <- NA
use(.data)

aggregate(year, by = list(STAGE = stage), FUN = c("length", "count"))
# Note the difference between 'length' and 'count' in Stage 1
# Means of subsets in 'aggregrate.data.frame'
# have 'na.rm' set to FALSE.
aggregate.data.frame(year, by = list(STAGE = stage), FUN = "mean")

## The default value of 'na.rm' is TRUE in aggregate.numeric of Epicalc.
aggregate(year, by = list(STAGE = stage), FUN = c("mean","median"))

## It can be set to FALSE though.
aggregate(year, by = list(STAGE = stage), FUN = c("mean","median"),
"na.rm"=FALSE)

# Omitting the FUN argument produces various statistics.
options(digits=3)
aggregate(year, by = list(HOSPITAL = hospital, STAGE = stage))

# Warning of na.rm
aggregate(year, by = list(HOSPITAL = hospital, STAGE = stage), length.warning=FALSE)

# Newly defined functions can be used
p05 <- function(x) quantile(x, prob=.05, na.rm=TRUE)
p95 <- function(x) quantile(x, prob=.95, na.rm=TRUE)
aggregate(year, by = list(HOSPITAL = hospital, STAGE = stage), FUN=c("p05", "p95"))
```

`aggregate plot` *Plot summary statistics of a numeric variable by group*

Description

Split a numeric variable into subsets, plot summary statistics for each

Usage

```
## S3 method for class 'plot'
aggregate(x, by, grouping = NULL, FUN = c("mean", "median"),
    error = c("se", "ci", "sd", "none"), alpha = 0.05, lwd = 1,
    lty = "auto", line.col = "auto", bin.time = 4, bin.method = c("fixed",
        "quantile"), legend = "auto", legend.site = "topright",
    legend.bg = "white", xlim = "auto", ylim = "auto", bar.col = "auto",
    cap.size = 0.02, lagging = 0.007, main = "auto", return.output = FALSE, ...)
```

Arguments

x	a numeric variable
by	a list of grouping elements for the bar plot, or a single numeric or integer variable which will form the X axis for the time line graph
grouping	further stratification variable for the time line graph
FUN	either "mean" or "median"
error	statistic to use for error lines (either 'se' or 'sd' for barplot, or 'ci' or 'none' for time line graph). When FUN = "median", can only be 'IQR' (default) or 'none'.
alpha	level of significance for confidence intervals
lwd	relative width of the "time" lines. See 'lwd' in ?par
lty	type of the "time" lines. See 'lty' in ?par
line.col	colour(s) of the error and time lines
bin.time	number bins in the time line graph
bin.method	method to allocate the "time" variable into bins, either with 'fixed' interval or equally distributed sample sizes based on quantiles
legend	presence of automatic legend for the time line graph
legend.site	a single character string indicating location of the legend. See details of ?legend
legend.bg	background colour of the legend
xlim	X axis limits
ylim	Y axis limits
bar.col	bar colours
cap.size	relative length of terminating cross-line compared to the range of X axis
lagging	lagging value of the error bars of two adjecant categories at the same time point. The value is result of dividing this distance with the range of X axis
main	main title of the graph
return.output	whether the dataframe resulted from aggregate should be returned
...	additional graphic parameters passed on to other methods

Details

This function plots aggregated values of 'x' by a factor (barplot) or a continuous variable (time line graph).

When 'by' is of class 'factor', a bar plot with error bars is displayed.

When 'by' is a continuous variable (typically implying time), a time line graph is displayed.

Both types of plots have error arguments. Choices are 'se' and 'sd' for the bar plot and 'ci' and IQR for both bar plot and time line graph. All these can be suppressed by specifying 'error'="none".

'bin.time' and 'bin.method' are exclusively used when 'by' is a continuous variable and does not have regular values (minimum frequency of 'by' <3). This condition is automatically and silently detected by 'aggregate.plot' before 'bin.method' chooses the method for aggregation and bin.time determines the number of bins.

If 'legend = TRUE" (by default), a legend box is automatically drawn on the "topright" corner of the graph. This character string can be changed to others such as, "topleft", "center", etc (see examples).

'cap.size' can be assigned to zero to remove the error bar cap.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'aggregate.data.frame', 'aggregate.numeric', 'tapply'

Examples

```
data(Compaq)
use(Compaq)
aggregate.plot(x=year, by=list(HOSPITAL = hospital, STAGE = stage))
aggregate.plot(x=year, by=list(HOSPITAL = hospital, STAGE = stage), return = TRUE)
aggregate.plot(x=year, by=list(HOSPITAL = hospital, STAGE = stage), error="sd")
aggregate.plot(x=year, by=list(HOSPITAL = hospital, STAGE = stage), error="ci")
# moving legend and chaging bar colours
aggregate.plot(x=year, by=list(HOSPITAL = hospital, STAGE = stage), error="ci",
  legend.site = "topleft", bar.col = c("red","blue"))
# manual creation of legend
aggregate.plot(x=year, by=list(HOSPITAL = hospital, STAGE = stage), legend = FALSE)
legend(x=7,y=6,legend=c("Public","Private"), fill=grey.colors(2), cex=1.3)
aggregate.plot(x=year, by=list(HOSPITAL = hospital, STAGE = stage), FUN = "median",
  legend.site = "topleft")

# Example with regular time intervals (all frequencies > 3)
data(Sitka, package="MASS")
use(Sitka)
tab1(Time, graph=FALSE) # all frequencies > 3
aggregate.plot(x=size, by=Time)
aggregate.plot(x=size, by=Time, cap.size = 0) # Note no cap on error bars
aggregate.plot(x=size, by=Time, grouping=treat)
# For with black and white presentation
aggregate.plot(x=size, by=Time, grouping=treat, lty = 1:2, line.col = c(1,1))
aggregate.plot(x=size, by=Time, grouping=treat, FUN="median",
  line.col=3:4, lwd =2)
# Compare with boxplot below
boxplot(size ~ treat + Time, col = 3:4, las=3)

# Example with irregular time intervals (some frequencies < 3)
data(BP)
use(BP); des()
age <- as.numeric(as.Date("2008-01-01") - birthdate)/365.25
pack()
tab1(age, graph=FALSE)
aggregate.plot(x=sbp, by=age)
aggregate.plot(x=sbp, by=age, grouping=saltadd)
aggregate.plot(x=sbp, by=age, grouping=saltadd, bin.method="quantile")
aggregate.plot(x=sbp, by=age, grouping=saltadd, lwd=3,
  line.col=c("blue","green"))
aggregate.plot(x=sbp, by=age, grouping=saltadd, lwd=3,
  line.col=c("blue","green") , main = NULL)
title(main="Effect of age and salt adding on SBP", xlab="years",ylab="mm.Hg")
points(age[saltadd=="no"], sbp[saltadd=="no"], col="blue")
points(age[saltadd=="yes"], sbp[saltadd=="yes"], pch=18, col="green")

## For a binary outcome variable, aggregrated probabilities is computed
```

```
data(Outbreak)
use(Outbreak)
recode(vars = age, old.value = 99, new.value = NA)
aggregate.plot(diarrhea, by=age, bin.time=5)
diarrhea1 <- factor(diarrhea)
levels(diarrhea1) <- c("no","yes")
pack()
aggregate.plot(diarrhea1, by=age, bin.time=5)
```

`Air Pollution` *Dataset on air pollution and deaths in UK*

Description

Deaths in London from 1st-15th Dec 1952

Usage

```
data(SO2)
```

Format

A data frame with 15 observations on the following 4 variables.

`day` a numeric vector: the day in Dec 1952

`deaths` a numeric vector: number of deaths

`smoke` a numeric vector: atmospheric smoke (mg/cu.m)

`SO2` a numeric vector: atmospheric sulphur dioxide (parts/million)

Source

from John F. Osborn, Statistical Exercises in Medical Research, Blackwell 1979

`alpha` *Cronbach's alpha*

Description

Calculate reliability coefficient of items in a data frame

Usage

```
alpha (vars, dataFrame = .data, casewise = FALSE, reverse = TRUE,
    decimal = 4, vars.to.reverse = NULL, var.labels = TRUE,
    var.labels.trunc =150)
alphaBest (vars, standardized = FALSE, dataFrame = .data)
```

Arguments

vars	a vector containing at least three variables from the data frame
dataFrame	data frame where items are set as variables
casewise	whether only records with complete data will be used
reverse	whether item(s) negatively correlated with other majority will be reversed prior to computation
decimal	number of decimal places displayed
var.labels	presence of descriptions of variables in the last column of the output
var.labels.trunc	number of characters used for variable descriptions, long labels can be truncated
vars.to.reverse	variable(s) to reverse prior to computation
standardized	whether choosing the best subset of items is based on the standardized alpha coefficient, if FALSE then the unstandardized alpha coefficient is used

Details

This function is based on the 'reliability' function from package 'Rcmdr', which computes Cronbach's alpha for a composite scale.

There must be at least three items in 'vars' specified by their names or their index in the data frame.

The argument 'reverse' (default = TRUE) automatically reverses items negatively correlated with other majority into negative and reports the activities in the first column of the last result section. This can be overwritten by the argument 'vars.to.reverse'

Similar to the 'reliability' function, users can see the effect of removing each item on the coefficents and the item-rest correlation.

'alphaBest' is a variant of 'alpha' for successive removal of items aiming to reach the highest possible Cronbach alpha. The resultant values include variable indices of excluded and remaining items, which can be forwarded to 'tableStack' to achieve total and mean scores of the best selected items. However, there is no promise that this will give the highest possible alpha. Manual attemps may also be useful in making comparison.

Value

A list.

'alpha' returns an object of class "alpha"

alpha	unstandardized alpha coefficient
std.alpha	standardized alpha coefficient
sample.size	sample size
use.method	method for handling missing values
rbar	the average inter-item correlation
items.selected	names of variables included in the function
alpha.if.removed	a matrix of unstandardized and standardized alpha coefficients and correlation of each item with the rest of the items
result	as above but includes a column showing the items that were reversed (if TRUE) and a column of item description. As a matrix, it could be sent to a spreadsheet software using 'write.csv'

decimal	decimal places
item.labels	a character vector containing descriptions of the items

'apha.Best' returns a list of the following elements

best.alpha	the possible highest alpha obtained from the function
removed	indices of items removed by the function
remaining	indices of the remaining items
items.reversed	names of items reversed

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'cronbach' from 'psy' package and 'reliability' from 'Rcmdr' package and 'tableStack' and 'unclassDataframe' of Epicalc

Examples

```
data(Cars93, package="MASS")
use(Cars93)
alpha(vars=c(Min.Price:MPG.highway, EngineSize))

data(Attitudes)
use(Attitudes)

alpha(qa1:qa18)  # Needs full screen of Rconsole
alpha(qa1:qa18, var.labels.trunc=30)
                 # Fits in with default R console screen

alpha(qa1:qa18, reverse=FALSE)

alphaBest(qa1:qa18) -> best.alpha
best.alpha # .7621
tableStack(best.alpha$remaining, reverse=TRUE)

# Manual attempts by trial and error give the following
alpha(c(qa1:qa9, qa15,qa18)) # .7644
```

ANC Table — *Dataset on effect of new ANC method on mortality (as a table)*

Description

This dataset contains frequency of various combinations of methods of antenatal care in two clinics with the outcome being perinatal mortality.

Usage

```
data(ANCtable)
```

Format

A data frame with 8 observations on the following 4 variables.

`death` a numeric vector: 1=no, 2=yes

`anc` a numeric vector indicating antenatal care type: 1=old 2=new

`clinic` a numeric vector indicating clinic code: 1=clinic A, 2=clinic B

`Freq` a numeric vector of frequencies

Examples

```
data(ANCtable)
use(ANCtable)
death <- death==2
anc <- factor(anc); levels(anc) <- c("old", "new")
clinic <- factor(clinic); levels(clinic) <- c("A","B")
glm1 <- glm(death ~ anc ,weights=Freq, family=binomial)
logistic.display(glm1)
glm2 <- glm(death ~ anc + clinic ,weights=Freq,family=binomial)
logistic.display(glm2)
lrtest(glm1, glm2)
rm(death, anc, clinic)
```

Antenatal care data *Dataset on effect of new antenatal care method on mortality*

Description

This dataset contains records of high risk pregnant women under a trial on new and old methods of antenatal care in two clinics. The outcome was perinatal mortality.

Usage

```
data(ANCdata)
```

Format

A data frame with 755 observations on the following 3 variables.

`death` a factor with levels `no` `yes`

`anc` a factor with levels `old` `new`

`clinic` a factor with levels `A` `B`

`Attitudes dataset` *Dataset from an attitude survey among hospital staff*

Description

Survey on attitudes related to services among hospital staff.

Codes for the answers qa1 to qa18 are

1 = strongly disagree
2 = disagree
3 = neutral
4 = agree
5 = strong agree

Usage

```
data(Attitudes)
```

Format

A data frame with 136 observations on the following 7 variables.

`id` identifying code of repondent

`sex` gender of respondent

`dep` code of department

`qa1` I have pride in my job

`qa2` I'm happy to give service

`qa3` I feel difficulty in giving service

`qa4` I can improve my service

`qa5` A service person must have patience

`qa6` I would change my job if had the chance

`qa7` Devoting some personal time will improve oneself

`qa8` Hard work will improve oneself

`qa9` Smiling leads to trust

`qa10` I feel bad if I cannot give service

`qa11` A client is not always right

`qa12` Experienced clients should follow the procedure

`qa13` A client violating the regulation should not bargain

`qa14` Understanding colleagues will lead to understanding clients

`qa15` Clients like this place due to good service

`qa16` Clients who expect our smiling faces create pressure on us

`qa17` Clients are often self-centered

`qa18` Clients should be better served

auc *Area under time-concentration curve*

Description

Compute area under time-concentration curve for individuals

Usage

```
auc(conc, time, id=NULL)
```

Arguments

conc	concentration
time	time point where the concentration was measured
id	subject identification

Details

This function compute auc using simple trapezoid summation.

id=NULL is used when concentration (conc) and time are from only one subject.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'auc' in 'PK' package

Examples

```
# Using 'by' and 'sapply' to compute individual auc of Indometh data
tmp <- by(data=Indometh, INDICES = Indometh$Subject,
          FUN = function(x) auc(conc=x$conc,
          time=x$time, id=NULL)
          )
sapply(tmp, as.numeric)

# A better way to compute
use(Indometh)
auc(conc=conc, time=time, id=Subject)
```

Bangladesh Fertility Survey

Dataset from 1988 Bangladesh Fertility Survey

Description

The file consists of a subsample of 1934 women grouped in 60 districts.

Usage

```
data(Bang)
```

Format

A data frame with 1934 observations on the following 7 variables.

`woman` identifying code of each woman

`district` identifying code for each district

`user` `1` = using contraceptive `0` = not using

`living.children` Number of living children at time of survey

1 = none
2 = 1
3 = 2
4 = 3 or more

`age_mean` age of woman in years, centred around the mean

`urban` Type of region of residence: `1` = urban, `0` = rural

`constant` constant term = 1

Source

Huq, N. M., and Cleland, J. 1990. Bangladesh Fertility Survey 1989 (Main Report). Dhaka: *National Institute of Population Research and Training*

BE to AD

Change year in B.E. to A.D.

Description

Convert Buddhist era date to Christian era date

Usage

```
be2ad(Date.in.BE)
```

Arguments

Date.in.BE	an object of class Date

Details

This function may be useful in countries where dates are (wrongly) commonly entered in the Buddhist Era (BE). The function subtracts 543 from the year component of the argument 'Date.in.BE'. See 'note' below.

Note

Although this function is useful in converting dates in BE to AD, there is still a serious limitation.

All computers validate a date field based on the Gregorian calendar (AD). Since AD is BE less 543 years and the leap year is always with AD being a multiple of 4 (and not a multiple of 100, except if it is a multiple of 400), the computer will return an invalid date for any record with 29 February and the year in BE. Thus, any candidate dataset for this function should not have any date of 29 February. The function be2ad **cannot** retrospectively solve this problem.

If a user wants to enter data using BE, the above limitation can only be overcome by separating the three fields of BE year, month and day during data entry and then using either the existing data entry software, such as Epidata, or a statistical software, such as R, to change BE years to AD years before incorporating them into a new date variable. Thus, this date variable would have year in AD only and will not need be2ad. In order to display the correct date variable in BE format, locale must be in Thai and appropriate format must be chosen. See example.

Despite the above limitation, this function is kept in Epicalc. The reason is that there would still be a lot of (those type of faulty) datasets around in the countries that use BE that require changing BE to AD before any analysis of date variables can proceed. In doing so, the analyst must be aware of this potential problem in the dataset. It is advisable to check the data first to see whether there are any dates that fall on 29 February.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

Examples

```
Date1 <- as.Date("2543-2-28")
be2ad(Date1)

## Not run:
## One would never have to

# be2ad(as.Date("2551-2-29"))

## because as.Date("2551-2-29") is an invalid Date
## End(Not run)

# To display date and time in BE under Thai Window OS
format(Sys.Date(), "%x")
format(Sys.time(), "%c")
```

Blood pressure — *Dataset on blood pressure and determinants*

Description

This dataset contains information on the records of 100 adults from a small cross-sectional survey in 2001 investigating blood pressure and its determinants in a community.

Usage

```
data(BP)
```

Format

A data frame containing 100 observations and 6 variables with variable descriptions.

Examples

```
data(BP)
use(BP)
des()
```

BMD — *Bone Mineral Density in Postmenopausal Women*

Description

This data contains information from a clinical trial investigating three doses of a drug which is thought to affect bone mineral density levels in postmenopausal women. Treatment 1 is the lowest dosage of the drug and Treatment 3 is the highest. Subjects were postmenopausal women who were randomly allocated to one to the three doses. Bone mineral densities were measured at the start of the trial and at 12 and 24 mnths after the trial commenced.

Usage

```
data(BMD)
```

Format

A data frame with 1,439 observations on the following 9 variables.

`subj` Subject ID

`month` Months since start of trial (0, 12, 24)

`group` Treatment group (1, 2, 3)

`bmd` Bone mineral density at hip (g/cm^2)

`smoking` Smoking status (1=No, 2=Yes (<10 per day), 3=10+ per day)

`hrt` Yes or No

`bmi` Body mass index (kg/m^2)

`yrs` Number of years since menopause

`age` Age at start of trial

Source

Nand SL, Wren BG, Gross B, Heller GZ (1999). Bone Density Effects of Continuous Estrone Sulfate and Varying Doses of Medroxyprogesterone Acetate, Obstetrics and Gynecology, 93(6), 1009-1013.

Examples

```
data(BMD)
use(BMD)
des()
aggregate.plot(bmd, month, group)
boxplot(bmd~month:group, col=5:7, xaxt="n", las=1)
```

Cancer survival *Dataset on cancer survival*

Description

A dataset on cancer survival checking whether there is a survival difference between cancer patients in private and public hospitals.

Usage

```
data(Compaq)
```

Format

A data frame with 1064 observations on the following 7 variables.

`id` a numeric vector

`hospital` a factor with levels `Public hospital` `Private hospital`

`status` a numeric vector

`stage` a factor with levels `Stage 1` `Stage 2` `Stage 3` `Stage 4`

`agegr` a factor with levels `<40` `40-49` `50-59` `60+`

`ses` a factor with levels `Rich` `High-middle` `Poor-middle` `Poor`

`year` a numeric vector indicating the year of recruitment into the study

Examples

```
data(Compaq)
use(Compaq)
des()
```

cc *Odds ratio calculation and graphing*

Description

Odds ratio calculation and graphing

Usage

```
cc(outcome, exposure, decimal = 2, cctable = NULL, graph = TRUE,
original = TRUE, design = "cohort", main, xlab = "auto", ylab,
  alpha = .05, fisher.or = FALSE, exact.ci.or = fisher.or)
cci(caseexp, controlex, casenonex, controlnonex, cctable = NULL,
graph = TRUE, design = "cohort", main, xlab, ylab, xaxis, yaxis,
  alpha = .05, fisher.or = FALSE, exact.ci.or = fisher.or, decimal = 2 )
cs(outcome, exposure, cctable = NULL, decimal = 2, method="Newcombe.Wilson",
   main, xlab, ylab, cex, cex.axis)
csi(caseexp, controlex, casenonex, controlnonex, cctable = NULL,
decimal = 2, method="Newcombe.Wilson")
graph.casecontrol(caseexp, controlex, casenonex, controlnonex, decimal = 2,
                  fisher.or = FALSE, alpha=0.05)
graph.prospective(caseexp, controlex, casenonex, controlnonex, fisher.or=FALSE,
                 decimal = 2, alpha=0.05)
labelTable(outcome, exposure, cctable = NULL, cctable.dimnames = NULL)
make2x2(caseexp, controlex, casenonex, controlnonex)
```

Arguments

cctable.dimnames	Dimension names of the variables, usually omitted
decimal	number of decimal places displayed
outcome, exposure	two dichotomous variables
cctable	A 2-by-2 table. If specified, will supercede the outcome and exposure variables
graph	If TRUE (default), produces an odds ratio plot
design	Specification for graph; can be "case control","case-control", "cohort" or "prospective"
caseexp	Number of cases exposed
controlex	Number of controls exposed
casenonex	Number of cases not exosed
controlnonex	Number of controls not exposed
original	should the original table be displayed instead of standard outcome vs exposure table
main	main title of the graph
xlab	label on X axis
ylab	label on Y axis
alpha	level of significance

fisher.or	whether odds ratio should be computed by the exact method
exact.ci.or	whether confidence limite of the odds ratio should be computed by the exact method
xaxis	two categories of exposure in graph
yaxis	two categories of outcome in graph
method	method of computation for 95 percent limits of risk difference
cex.axis	character expansion factor for graph axis
cex	character expansion factor for text in the graph

Details

'cc' usually reads in two variables whereas in 'cci' four number are entered manually. However, both the variables and the numbers should be omitted if the analysis is directly on a table specified by 'cctable'.

From both functions, odds ratio and its confidence limits, chisquared test and Fisher's exact test are computed. The odds ratio calcuation is based on cross product method unless 'fisher.or' is set as TRUE. It's confidence limits are obtained by the exact method unless exact.ci.or is set as FALSE.

'cs' and 'csi' are for cohort and cross-sectional studies. It computes the absolute risk, risk difference, and risk ratio. When the exposure is a risk factor, the attributable fraction exposure, attributable fraction population and number needed to harm (NNH) are also displayed in the output. When the exposure is a protective factor, protective efficacy or percent of risk reduced and number needed to treat (NNT) are displayed instead.

If there are more than 2 exposure categories and the sample size is large enough, a graph will be plotted.

'method' in 'csi' and 'cs' chooses whether confidence limits of the risk difference should be computed by Newcomb-Wilson method. Both this and the standard method may give non-sensible values if the risk difference is not statistically significant.

'make2x2' creates a 2-by-2 table using the above orientation.

'graph.casecontrol' and 'graph.prospective' draw a graph comparing the odds of exposure between cases and controls or odds of diseased between exposed and non-exposed.

These two graphic commands are automatically chosen by 'cc' and 'cci', depending on the 'design' argument.

Alternatively, a contingency table saved from 'make2x2' can be supplied as the 'cctable' argument for the 'cc' function and so on.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'fisher.test', 'chisq.test' and 'mhor'

Examples

```
data(Oswego)
use(Oswego)
cc(ill, chocolate)
cc(ill, chocolate, design="case-control")
```

```
cs(ill, chocolate) # The outcome variable should come first.

#    For the following table
#          chocolate
#    ill     FALSE TRUE
#     FALSE     7   22
#     TRUE     20   25
#
cci(25, 22, 20, 7)
graph.casecontrol(25, 22, 20, 7)
graph.prospective(25, 22, 20, 7)
# Each of the above two lines produces untitled graph, which can be decorated
# additionally decorated

#Alternatively
table1 <- make2x2(25,22,20,7)
cc(outcome=NULL, exposure=NULL, cctable=table1)
cs(outcome=NULL, exposure=NULL, cctable=table1)
agegr <- pyramid(age, sex, bin=30)$ageGroup
cs(ill, agegr, main="Risk ratio by age group", xlab="Age (years)")
```

CI *Confidence interval of probabilty, mean and incidence*

Description

Compute confidence interval(s) of variables or values input from keyboard.

Usage

```
ci(x, ...)

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

## S3 method for class 'binomial'
ci(x, size, precision, alpha = 0.05, ...)

## S3 method for class 'numeric'
ci(x, n, sds, alpha = 0.05, ...)

## S3 method for class 'poisson'
ci(x, person.time, precision, alpha = 0.05, ...)
```

Arguments

x	a variable for 'ci', number of success for 'ci.binomial', mean(s) for 'ci.numeric', and counts for 'ci.poisson'
size	denominator for success
precision	level of precision used during computation for the confidence limits
alpha	significance level
n	sample size

sds	standard deviation
person.time	denominator for count
...	further arguments passed to or used by other methods

Details

These functions compute confidence intervals of probability, mean and incidence from variables in a dataset or values from keyboard input.

'ci' will try to identify the nature of the variable 'x' and determine the appropriate method (between 'ci.binomial' and 'ci.numeric') for computation. 'ci' without a specified method will never call 'ci.poisson'.

The specific method, ie. 'ci.binomial', 'ci.numeric' or 'ci.poisson', should be used when the values are input from the keyboard or from an aggregated data frame with columns of variables for the arguments.

'ci.binomial' and 'ci.numeric' employ exact probability computation while 'ci.numeric' is based on the t-distribution assumption.

Value

'ci.binomial' and 'ci.poisson' return a data frame containing the number of events, the denominator and the incidence rate. 'ci.numeric' returns means and standard deviations. All of these are followed by the standard error and the confidence limit, the level of which is determined by 'alpha'

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'summ'

Examples

```
data(Oswego)
use(Oswego)
# logical variable
ci(ill)
# numeric variable
ci(age)
# factor
ci(sex=="M")
ci(sex=="F")

# Example of confidence interval for means
library(MASS)
use(Cars93)
car.price <- aggregate(Price, by=list(type=Type), FUN=c("mean","length","sd"))
car.price
ci.numeric(x=car.price$mean, n=car.price$length, sds=car.price$sd.Price )

# Example of confidence interval for probabilty
data(ANCdata)
use(ANCdata)
death1 <- death=="yes"
```

```
death.by.group <- aggregate.numeric(death1,
by=list(anc=anc, clinic=clinic), FUN=c("sum","length"))
death.by.group
ci.binomial(death.by.group$sum.death1, death.by.group$length)

# Example of confidence interval for incidence
data(Montana)
des(Montana)
age.Montana <- aggregate.data.frame(Montana[,1:2],
by=list(agegr=Montana$agegr),FUN="sum")
age.Montana
ci.poisson(age.Montana$respdeath, person.time=age.Montana$personyrs)

# Keyboard input
# What is the 95 % CI of sensitivity of a test that gives all
# positive results among 40 diseased individuals
ci.binomial(40,40)

# What is the 99 % CI of incidence of a disease if the number
# of cases is 25 among 340,000 person-years
ci.poisson(25, 340000, alpha=.01) # 4.1 to 12.0 per 100,000 person-years
```

Codebook *Codebook of a data frame*

Description

Print description, summary statistics and one-way tabulation of variables

Usage

```
codebook(dataFrame=.data)
```

Arguments

dataFrame	A data frame for printing the codebook

Details

The default value of dataFrame (ie if no argument is supplied) is '.data'.

While 'summ' produces summary statistics of both numeric and factor variables, 'codebook' gives summary statistics of all numeric variables and one-way tabulation of all factors of the data frame.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'use', 'summ', 'tab1' and 'tableStack'

Examples

```
data(Familydata)
use(Familydata)
des()
codebook(Familydata)
codebook() # Same result since line #2 has created .data from Familydata
```

`Data for cleaning` *Dataset for practicing cleaning, labelling and recoding*

Description

The data come from clients of a family planning clinic.

For all variables except id: 9, 99, 99.9, 888, 999 represent missing values

Usage

```
data(Planning)
```

Format

A data frame with 251 observations on the following 11 variables.

`ID` a numeric vector: ID code

`AGE` a numeric vector

`RELIG` a numeric vector: Religion

1 = Buddhist
2 = Muslim

`PED` a numeric vector: Patient's education level

1 = none
2 = primary school
3 = secondary school
4 = high school
5 = vocational school
6 = university
7 = other

`INCOME` a numeric vector: Monthly income in Thai Baht

1 = nil
2 = < 1,000
3 = 1,000-4,999
4 = 5,000-9,999
5 = 10,000

AM a numeric vector: Age at marriage

REASON a numeric vector: Reason for family planning

1 = birth spacing
2 = enough children
3 = other

BPS a numeric vector: systolic blood pressure

BPD a numeric vector: diastolic blood pressure

WT a numeric vector: weight (Kg)

HT a numeric vector: height (cm)

Examples

```
data(Planning)
des(Planning)

# Change var. name to lowercase
names(Planning) <- tolower(names(Planning))
use(Planning)
des()

# Check for duplication of 'id'
any(duplicated(id))
duplicated(id)
id[duplicated(id)] #215

# Which one(s) are missing?
setdiff(min(id):max(id), id) # 216

# Correct the wrong one
id[duplicated(id)] <- 216
```

---

des *Desription of a data frame or a variable*

Description

Description of a data frame or a variable or wildcard for variable names

Usage

```
des(x=.data, select, exclude)
```

Arguments

x	an object such as a vector (variable), a matrix, a table, a list or a data frame
select	expression, indicating columns to select from '.data.'
exclude	expression, indicating columns to exclude

Details

The default value of x (ie if no argument is supplied) is '.data'. If 'x' is a data frame, its variable names will be listed with class and the description of each variable.

If 'x' is a variable, the environment and attached data frame containing 'x' will be described.

For a data frame containing too many variables, 'select' and 'exclude' can be specified to display fewer variable descriptions at a time. Unlike 'keepData', these two arguments do not have any permanent effect on the data frame.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'use', 'summ', 'label.var', 'subset' and 'keepData'

Examples

```
data(Oswego)
use(Oswego)
# In the tutorial, when "oswego.rec" which is an EpiInfo file is available,
# instead of typing the above two lines, one can directly type:
# use("oswego.rec")

des() # This is one of the most useful Epicalc functions!

#### Detection of variables of the same name in different data frames.
# Note that 'age' is a variable in '.data' due to the function 'use'.
des(Oswego) # Same results. Note that 'age' is also in 'Oswego'.
des(infert) # The third 'age' is in another data frame,
        # from the datasets package in R, 'infert'.
attach(infert)
search() # Show all data frames that are in the search path
des(sex) # 'sex' is found only in '.data'
des(induced)
age <- "abc" # Just a silly example for a variable
des(age)     # Shows all occurrences of 'age', wherever it is
rm(age)
detachAllData()

#### Wildcard for variables
use(Oswego)
des("c*")     # Show all variables starting with 'c'
des("?????")  # Show all variables with 5 characters in the name

agegr <- cut(age, breaks=c(0,20,40,60,80))
label.var(agegr, "age group")
# Note that the above line incoperates 'agegr' into '.data
# making it eligible to be included in the group under the following wildcard
des("age*")

#### Subset of variables in .data
des(select = 1:5) # First five variables
des(select = age:onsetdate) # Same results
```

```
des(select = c(1,2,5,20))
des(select = c(age, sex, onsetdate, fruitsalad))

des(select = sex:chocolate)

## The following six lines give the same results
des(select = -(sex:chocolate))
des(select = -sex:-chocolate)
des(select = -(2:19))
des(select = -19:-2)
des(exclude = sex:chocolate)
des(exclude = 2:19)

#### Wildcard: same effects with or without 'select'
des(select = "c*")
des("c*")

## Exclusion using wildcard, however, needs an 'exclude' argument.
des(exclude = "c*")
```

Detach all data frames

Detach all data frames

Description

Detach all data frames

Usage

```
detachAllData()
```

Details

The R command 'attach()' copies the data frame in the argument into a data frame in the search path (usually the second position) consequently making all the variables in the data frame easy to refer to. However, changing any element of the index data frame has no effect on the one in the search path unless the changed data frame is attached to the search path again. Having too many data frames in the search path subsequently causes confusion, not to mention an increase in memory usage. It is a good practice to detach the index data frame first before manipulating it and then attaching to it again. 'detachAllData()' is a self explanatory command which solves the over-attaching problem.

'detachAllData()' removes all non-function objects in the R search path.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'use', 'detach', 'search'

Examples

```
attach(CO2)
data(Hakimi)
attach(Hakimi)
search()
detachAllData()
search()
```

DHF99 *Dataset for exercise on predictors for mosquito larva infestation*

Description

Dataset from a community survey on water containers infested by mosquito larvae.

Usage

```
data(DHF99)
```

Format

A data frame with 300 observations on the following 5 variables.

`houseid` a numeric vector

`village` a numeric vector indicating village ID

`education` a factor with levels `Primary` `Secondary` `High school` `Bachelor` `Other`

`containers` a numeric vector indicating number of containers infested

`viltype` a factor with levels `rural` `urban` `slum`

References

Thammapalo, S., Chongsuwiwatwong, V., Geater, A., Lim, A., Choomalee, K. 2005. Socio-demographic and environmental factors associated with Aedes breeding places in Phuket, Thailand. *Southeast Asian J Trop Med Pub Hlth* **36(2)**: 426-33.

---

dotplot *Dot plot*

Description

Plot of frequency in dots

Usage

```
dotplot (x, bin = "auto", by = NULL, xmin = NULL, xmax = NULL,
    time.format = NULL, time.step = NULL, pch = 18, dot.col = "auto",
    main = "auto", ylab = "auto", cex.X.axis = 1, cex.Y.axis = 1, ...)
```

Arguments

x	a numeric vector. Allowed types also include "Date" and "POSIXct"
bin	number of bins for the range of 'x'
by	stratification variable
xmin	lower bound of x in the graph
xmax	upper bound of x in the graph
time.format	format for time or date at the tick marks
time.step	a character string indicating increment of the sequence of tick marks
pch	either an integer specifying a symbol or a single character to be used as the default in plotting points
dot.col	a character or a numeric vector indicating the colour of each category of 'by'
main	main title
ylab	Y axis title
cex.X.axis	character extension scale of X axis
cex.Y.axis	character extension scale of Y axis
...	graphical parameters for the dots when there is no stratification

Details

'dotplot' in Epicalc is similar to a histogram. Each dot represents one record. Attributes of the dots can be further specified in '...' when there is no strafication. Otherwise, the dots are plotted as a diamond shape and the colours are automatically chosen based on the current palette and the number of strata.

When 'bin="auto"' (by default), and the class of the vector is 'integer', 'bin' will be automatically set to max(x)-min(x)+1. This strategy is also applied to all other time and date variables. Users can try other values if the defaults are not to their liking. See the example of 'timeExposed' below.

The argument 'xmin' and 'xmax' indicate the range of x to be displayed on the graph. These two arguments are independent from the value of 'bin', which controls only the number of columns for the original data range.

Dotplot usually starts the first tick mark on the X-axis at 'xmin' (or min(x) if the 'xmin' is not specified). The argument 'time.step' is typically a character string, containing one of 'sec', 'min', 'hour', 'day', 'DSTday', 'week', 'month' or 'year'. This can optionally be preceded by an integer and a space, or followed by "s", such as "2 weeks".

Setting proper 'xmin', 'xmax' and 'time.step' can improve the location of tick marks on the X-axis. The 'time.format' argument can then be given to further improve the graph. See the last two examples for a better understanding.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'summ', 'hist', 'seq.Date' and 'seq.POSIXt'

Examples

```
a <- rep(1:2, 250)
b <- rnorm(500,mean=a)
dotplot(b)
dotplot(b, pch=1)
dotplot(b, by=a)
dotplot(b, by=a, pch=1) # You may try other values of 'pch'

# For the commands below,
# if dates in X axis are not readable,
# try omitting '#' from the next line
# Sys.setlocale("LC_ALL", "C")

# The number of dots in each column is the frequency
# of 'x' for the exact value on the X axis.
zap()
data(Outbreak)
use(Outbreak)
class(age) # numeric
dotplot(age) # 40 columns
age.as.integer <- as.integer(age)
dotplot(age.as.integer)
# 'bin' is the number of columns in the data range.
# Specifying 'min' and 'max' only expands or truncates
# the range of the X axis and has no effect on the distribution
# of the dots inside the data range.
dotplot(age.as.integer, xmin=0, xmax=150) # Just for demonstration.
dotplot(age.as.integer, xmin=0, xmax=70) # the "99"s are now out of the plot.
dotplot(age.as.integer, xmin=0, xmax=70, by=sex)

# Controlling colours of the dots
dotplot(age.as.integer, xmin=0, xmax=70, dot.col="chocolate")
sex1 <- factor(sex); levels(sex1) <- list("M"=1,"F"=0)
dotplot(age.as.integer, xmin=0, xmax=70, by=sex1, dot.col=c(2,5))
dotplot(age.as.integer, xmin=0, xmax=70, by=sex1,
  dot.col=c("brown","blue"), main="Age by sex",
  cex.X.axis=1.3, cex.Y.axis=1.5, cex.main=1.5)
```

Ectopic pregnancy *Dataset of a case-control study looking at history of abortion as a risk factor for ectopic pregnancy*

Description

This case-control study has one case series and two control groups.
The subjects were recruited based on three types of pregnancy outcome

Usage

```
data(Ectopic)
```

Format

A data frame with 723 observations on the following 4 variables.

`id` a numeric vector

`outc` a factor with levels `EP` `IA` `Deli`

EP	= ectopic pregnancy
IA	= women coming for induced abortion
Deli	= women admitted for full-term delivery

`hia` a factor with levels `never IA` `ever IA`

`gravi` a factor with levels `1-2` `3-4` `>4`

Examples

```
data(Ectopic)
library(nnet)
use(Ectopic)
multi1 <- multinom(outc ~ hia + gravi, data=.data)
summary(multi1)
mlogit.display(multi1)

# Changing referent group of outcome
outcIA <- relevel(outc, ref="IA")
pack()
multi2 <- multinom(outcIA ~ hia + gravi, data=.data)
summary(multi2)
mlogit.display(multi2)
```

---

exists.var *Check existing object*

Description

Check existing object.

Usage

```
exists.var(x)
```

Arguments

x	names to check

See Also

'exists'

Examples

```
a <- 1:10
b <- 1:20
exists.var("a")
exists.var(c("a","b"))
exists.var("age")
```

---

expand *Expand an aggregated data frame*

Description

Expand an 'aggregate'd data frame into a case-by-case format based on the values specified in a column

Usage

```
expand(aggregate.data, index.var = "Freq", retain.freq = FALSE)
```

Arguments

aggregate.data	an aggregate data frame having a variable indicating the replication of subjects having that combination of characteristics, which are indicated by other variables
index.var	name of a variable indicating frequency of replication
retain.freq	whether the index variable or frequency variable should be retained in the returned data frame

Details

An aggregated data frame has one variable (colunm) indicating the number or frequency of replication of subjects having the same values of other variables as the index record.

'expand' replicates the row using the value in 'index.var' as the number of replications.

'retain.freq' indicates whether the 'index.var', which is the frequency, should be retained.

Note

The aggregated data frame is not changed. Remember to assign the result.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'table', 'xtabs', 'aggregate'

Examples

```
## Expanding an aggregated data frame
data(ANCtable)
des(ANCtable)
a <- expand(ANCtable)
des(a)

## Aggregating a case-by-case data frame
data(ANCdata)
use(ANCdata)
des()
id <- 1:nrow(ANCdata)
aggregate.numeric(id, by=list(Death=death, Anc=anc, Clinic=clinic),
FUN="count")
```

Familydata *Dataset of a hypothetical family*

Description

Anthropometric and financial data of a hypothetical family

Usage

```
data(Familydata)
```

Format

A data frame with 11 observations on the following 6 variables.

`code` a character vector

`age` a numeric vector

`ht` a numeric vector

`wt` a numeric vector

`money` a numeric vector

`sex` a factor with levels `F` `M`

Examples

```
data(Familydata)
use(Familydata)
des()
summ()
age2 <- age^2
plot(age, money, log="y")
dots.of.age <- seq(0,80,0.01)
new.data.frame <- data.frame(age=dots.of.age, age2=dots.of.age^2)
lm1 <- lm(log(money) ~ age + age2)
dots.of.money <- predict.lm(lm1, new.data.frame)
lines(dots.of.age, exp(dots.of.money), col="blue")
```

fillin *fillin - Rectangularize a dataframe*

Description

fillin adds observations with missing data so that all combinations of the specified variables exist, thus making a complete rectangularization.

Usage

```
fillin(dataFrame=.data, select, fill=NA)
```

Arguments

dataFrame	a data frame.
select	a vector of at least 2 variables from the data frame. If missing all variables in the data frame will be used.
fill	the value used to fill in all other variables from the data frame. Defaults to NA.

Author(s)

Edward McNeil < <edward.m@psu.ac.th»

See Also

table

Examples

```
data <- data.frame(sex=c("female","male","male"),
  race=c("black","black","white"),
  x1=c(.5,.4,.1),
  x2=c(32,40,53))
data
fillin(data, select=c(sex,race))
data.new <- fillin(data, select=c(sex,race), fill=0)
data.new

data <- data.frame(x = gl(3,3),
                   y = rep(gl(3,1),3),
                   z = gl(2,6,length=9),
                   n = rpois(9,10) )
data
fillin(data, c(x,y))
fillin(data, c(x,z))
fillin(data, c(x,y,z), fill=0)
```

`Follow-up Plot` *Longitudinal followup plot*

Description

Plot longitudinal values of individuals with or without stratification

Usage

```
followup.plot(id, time, outcome, by = NULL, n.of.lines = NULL, legend = TRUE,
    legend.site = "topright", lty = "auto", line.col = "auto",
    stress = NULL, stress.labels = FALSE, label.col = 1, stress.col = NULL,
    stress.width = NULL, stress.type = NULL, lwd = 1, xlab, ylab, ...)
```

Arguments

`id`	idenfication variable of the same subject being followed up
`time`	time at each measurement
`outcome`	continuous outcome variable
`by`	stratification factor (if any)
`n.of.lines`	number of lines (or number of subjects in the data frame) randomly chosen for drawing
`legend`	whether a legend will be automatically included in the graph
`legend.site`	a single character string indicating location of the legend. See details of ?legend
`lty`	type of the "time" lines. See 'lty' in ?par
`line.col`	line colour(s) for non-stratified plot
`stress`	subset of ids to draw stressed lines
`stress.labels`	whether the stressed lines should be labelled
`label.col`	single integer indicating colour of the stressed line labels
`stress.col`	colour values used for the stressed line. Default value is '1' or black
`stress.width`	relative width of the stressed line
`stress.type`	line type code for the stressed line
`lwd`	line width
`xlab`	label for X axis
`ylab`	label for Y axis
`...`	other graphic parameters

Details

'followup.plot' plots outcome over time of the individual subjects.

If a stratification variable 'by' is specified, the levels of this variable will be used to color the lines.

'n.of.lines' is used to reduce the number of lines to allow the pattern to be seen more clearly.

'legend' is omitted if 'n.of.lines' is not NULL or the number of subjects exceeds 7 without stratification.

'line.col' works only for a non-stratified plot. It can be a single standard colour or "multicolor".

Values for 'stress.col', 'stress.width' and 'stress.type', if not NULL, should follow those for 'col', 'lwd' and 'lty', respectively

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'plot','lines'

Examples

```
use(Indometh)
followup.plot(Subject, time, conc)
followup.plot(Subject, time, conc, lty=1:6, line.col=rep("black",6))

library(MASS)
use(Sitka)
followup.plot(tree, Time, size)
followup.plot(tree, Time, size, line.col = "brown")
followup.plot(tree, Time, size, line.col = "multicolor")
followup.plot(tree, Time, size, n.of.lines=20, line.col = "multicolor")

# Breakdown of color by treatment group
followup.plot(tree, Time, size, by=treat)

# The number of lines reduced to 40
followup.plot(tree, Time, size, by=treat, n.of.lines=40)

# Stress some lines
length(table(tree)) # 79 trees followed up

# Identifying trees that sometimes became smaller
sortBy(tree, Time)
visit <- markVisits(id= tree, time=Time)
next.size <- lagVar(var=size, id=tree, visit=visit, lag.unit=-1)
pack()
smaller.trees <- tree[next.size < size]
followup.plot (tree, Time, size, line.col=5, stress=smaller.trees,
  stress.col=2, stress.width=2, stress.type=2)
followup.plot (tree, Time, size, line.col=5, stress=smaller.trees,
  stress.col=2, stress.width=2, stress.type=2, stress.labels=TRUE)
```

Hakimi's data *Dataset on effect of training personnel on neonatal mortality*

Description

Subset of a dataset from an intervention trial of education on personnel and the effect on neonatal mortality. Non-fatal records were randomly selected from the original dataset, just for practice and interpretation of interaction term.

Usage

```
data(Hakimi)
```

Format

A data frame containing 456 observations and 4 variables.

`dead` neonatal death: 1=yes, 0=no

`treatment` intervention programme: 1=yes, 2=no

`malpres` malpresentation of fetus: 1=yes, 0=no

`birthwt` birth weight for foetus in gram

Examples

```
data(Hakimi)
use(Hakimi)
des()
cc(dead, treatment)
mhor(dead, treatment, malpres)
```

`hello` *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

`Hookworm 1993` *Dataset from a study on hookworm prevalence and intensity in 1993*

Description

A dataset from a cross-sectional survey in 1993 examining hookworm infection

Usage

```
data(HW93)
```

Format

A data frame with 637 observations on the following 6 variables.

`id` a numeric vector for personal identification number

`epg` a numeric vector for eggs per gram of faeces

`age` a numeric vector for age in years

`shoes` a factor for shoe wearing with levels `no` `yes`

`intense` a factor for intensity of infection in epg. with levels `0` `1-1,999` `2,000+`

`agegr` a factor for age group with levels `<15 yrs` `15-59 yrs` `60+ yrs`

Examples

```
library(MASS)
data(HW93)
use(HW93)
intense.ord <- ordered(intense)
ord.hw <- polr(intense.ord ~ agegr + shoes)
summary(ord.hw)
```

Hookworm and blood loss

Hookworm infection and blood loss: SEAJTM 1970

Description

A study using radio-isotope to examine daily blood loss and number of hookworms infecting the patients.

Usage

```
data(Suwit)
```

Format

A data frame with 15 observations on the following 3 variables.

`id` a numeric vector

`worm` a numeric vector: number of worms

`bloss` a numeric vector: estimated daily blood loss (mg/day)

Source

Areekul, S., Devakul, K., Viravan, C., Harinasuta, C. 1970 Studies on blood loss, iron absorption and iron reabsorption in hookworm patients in Thailand. *Southeast Asian J Trop Med Pub Hlth* **1(4)**: 519-523.

References

~~ possibly secondary sources and usages ~~

Examples

```
data(Suwit)
use(Suwit)
des()
plot(worm, bloss, type="n")
text(worm, bloss, labels=id)
abline(lm(bloss ~ worm), col="red")
```

IUD trial admission data

Dataset admission of cases for IUD trials

Description

This dataset is a subset of WHO IUD trial. It should be merged with IudFollowup and IudDiscontinue

Usage

```
data(IudAdmit)
```

Format

A data frame containing 918 observations and 4 variables.

`id` a numeric vector for personal identification number

`idate` date of IUD insertion

`lmptime` time since last menstrual period

`a122` type of IUD

Examples

```
data(IudAdmit)
```

IUD trial discontinuation data

Dataset on discontinuation of the IUD trial cases

Description

This dataset is a subset of WHO IUD trial. It should be merged with IudAdmit and IudFollowup

Usage

```
data(IudDiscontinue)
```

Format

A data frame containing 398 observations and 3 variables.

`id` a numeric vector for personal identification number

`discdate` date of discontinuation

`d23` primary reason for discontinuation

Examples

```
data(IudDiscontinue)
```

IUD trial follow-up data

Dataset followup cases of IUD trials

Description

This dataset is a subset of WHO IUD trial. It should be merged with IudAdmit and IudDiscontinue

Usage

```
data(IudFollowup)
```

Format

A data frame containing 4235 observations and 6 variables.

`id` a numeric vector for personal identification number

`vlmpdate` date of last mentrual period before this visit

`vdate` date of visit

`f22` lactating

`f51` IUD threads visible

`f61` subject continuing

Examples

```
data(IudFollowup)
```

kap

Kappa statistic

Description

Measurement of agreement in categorization by 2 or more raters

Usage

```
kap(x, ...)

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

## S3 method for class 'table'
kap(x, decimal =3, wttable = c(NULL, "w", "w2"), print.wttable = FALSE, ...)

## S3 method for class '2.raters'
kap(x, rater2, decimal =3, ...)

## S3 method for class 'm.raters'
```

```
kap(x, decimal =3, ...)

## S3 method for class 'ByCategory'
kap(x, decimal =3, ...)
```

Arguments

x	an object serving the first argument for different methods

FUNCTION	'x'
'kap.table'	table
'kap.2.raters'	rater1
'kap.m.raters'	data frame with raters in column
'kap.ByCategory'	data frame with categories in column

decimal	number of decimal in the print
wttable	cross tabulation of weights of agreement among categories. Applicable only for 'kap.table' and 'kap.2.raters'
print.wttable	whether the weights table will be printed out
rater2	a vector or factor containing opinions of the second rater among two raters.
...	further arguments passed to or used by other methods.

Details

There are two different principles for the calculation of the kappa statistic. 'kap.table' and 'kap.2.raters' use two fixed raters whereas 'kap.m.raters' and 'kap.ByCategory' are based on frequency of category of rating an individual received without a requirement that the raters must be fixed.

'kap.table' analyses kappa statistics from a predefined table of agreement of two raters.

'wttable' is important only if the rating can be more than 2 levels. If this argument is left as default or 'NULL', full agreement will be weighted as 1. Partial agreement is considered as non-agreement and weighted as 0.

When 'wttable = "w"' the weights are given by

$$1 - abs(i - j)/(1 - k)$$

where i and j index the rows and columns of the ratings and k is the maximum number of possible ratings. A weight of 1 indicates an observation of perfect agreement.

When 'wttable = "w2", the weights are given by

$$1 - (abs(i - j)/(1 - k))^2.$$

In this case, weights of partial agreements will further increase.

'wttable' can otherwise be defined by the user.

'kap.2.raters' takes two vectors or factors, one for each of the two raters. Cross-tabulation of the two raters is displayed and automatically forwarded for computation of kappa statistic by 'kap.table'.

'kap.m.raters' is used for more than 2 raters. Although the variables are arranged based on columns of individual raters, only the frequency in each category rating is used. This function calculates the frequencies without any display and automatically forwards the results for computation by 'kap.ByCategory'.

'kap.ByCategory' is for the grouped data format, where each category (column) contains the counts for each individual subject being rated. As mentioned above, the frequencies can come from different sets of raters.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'table'

Examples

```
## Computation of kappa from a table
class  <- c("Normal","Benign","Suspect","Cancer")
raterA <- gl(4,4, label=class)
raterB <- gl(4,1,16, label=class)
freq   <- c(50,2,0,1,2,30,4,3,0,0,20,1,1,3,4,25)
table1 <- xtabs(freq ~ raterA + raterB)
table1
kap(table1)
wt <-c(1,.5,0,0,.5,1,0,0,0,0,1,.8,0,0,.8,1)
wttable <- xtabs(wt ~ raterA + raterB)
wttable # Agreement between benign vs normal is .5, suspect vs cancer is .8
kap(table1, wttable=wttable, print.wttable=TRUE)

# The following two lines are computational possible but inappropriate
kap(table1, wttable = "w", print.wttable=TRUE)
kap(table1, wttable = "w2", print.wttable=TRUE)

## A data set from 5 raters with 3 possible categories.
category.lab <- c("yes","no","Don't know")
rater1 <- factor(c(1,1,3,1,1,1,1,2,1,1), labels=category.lab)
rater2 <- factor(c(2,1,3,1,1,2,1,2,3,1), labels=category.lab)
rater3 <- factor(c(2,3,3,1,1,2,1,2,3,1), labels=category.lab)
rater4 <- factor(c(2,3,3,1,3,2,1,2,3,3), labels=category.lab)
rater5 <- factor(c(2,3,3,3,3,2,1,3,3,3), labels=category.lab)
kap.m.raters(data.frame(rater1,rater2,rater3,rater4,rater5))

# The above is the same as
YES <- c(1,2,0,4,3,1,5,0,1,3)
NO <- c(4,0,0,0,0,4,0,4,0,0)
DONTKNOW <- c(0,3,5,1,2,0,0,1,4,2)
kap.ByCategory(data.frame(YES,NO,DONTKNOW))

# Using 'kap.m.raters' for 2 raters is inappropriate. Kappa obtained
# from this method assumes that the agreement can come from any two raters,
# which is usually not the case.
kap.m.raters(data.frame(rater1, rater2))
# 'kap.2.raters' gives correct results
kap.2.raters(rater1, rater2)

# When there are missing values,
rater3[9] <- NA; rater4[c(1,9)] <- NA
kap.m.raters(data.frame(rater1,rater2,rater3,rater4,rater5))
# standard errors and other related statistics are not available.

# Two exclusive rating categories give only one common set of results.
# The standard error is obtainable even if the numbers of raters vary
# among individual subjects being rated.
```

```
totalRaters <- c(2,2,3,4,3,4,3,5,2,4,5,3,4,4,2,2,3,2,4,5,3,4,3,3,2)
pos <- c(2,0,2,3,3,1,0,0,0,4,5,3,4,3,0,2,1,1,1,4,2,0,0,3,2)
neg <- totalRaters - pos
kap.ByCategory(data.frame(neg, pos))
```

`Keep data` *Keep a subset of variables or records*

Description

Keep only subset of variables or records in the default data frame '.data'

Usage

```
keepData (dataFrame = .data, sample=NULL, exclude=NULL, subset, select,
 drop = FALSE, refactor = "subset.vars", ...)
```

Arguments

`dataFrame`	a data frame
`sample`	an integer indicating the size of random sample or a value < 1 indicating fraction of records to be extracted
`exclude`	an expression, indicating columns to remove from '.data'.
`subset`	a logical expression indicating elements or rows to keep: missing values are taken as false.
`select`	an expression indicating columns to select from a data frame.
`drop`	passed on to [indexing operator.
`refactor`	after subsetting, whether the levels of variable(s) with zero count should be removed
`...`	further arguments to be passed to or from other methods.

Details

'keepData' is the Epicalc version of 'subset.data.frame' which is a standard R function. It reduces the data frame to the specified subset and updates the search path accordingly.

Using 'keepData' will retain descriptions of the data, and the remaining variables, ready to be used by other Epicalc functions that can exploit them such as 'des', 'codebook', 'summ', 'tab1' etc ...

Since this command only affects the specified data frame (usually '.data'), any new variables created as free vectors will not be changed. The difference in length of variables may occur from the 'subset' argument. To avoid this, 'pack' or 'label.var' should be used to incoporate any relevant free vectors into the default data frame, '.data' so that all variables can be subsetted simultaneously, thus reducing the complications of the difference in variable lengths.

The argument 'refactor' is effective only when the argument 'subset' is specified. By default, 'refactor' is set to "subset.vars" indicating that the levels of the variables used in subset criteria will be revised to eliminate levels with zero count. If refactor="all", all factor variables in the dataFrame will be affected.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'des', 'subset', 'sample'

Examples

```
## Record sampling
data(ANCdata)
use(ANCdata)
des()
keepData(sample=500)
des() # Note reduction of sample size to 500
use(ANCdata)
keepData(sample=.1) # Select 10% or 75 records
des()

## Selecting specific record numbers
data(Compaq)
use(Compaq)
keepData(subset = 1:nrow(.data) <= 50) #First 50 records
summ()
use(Compaq)
every.seventh <- is.element(1:nrow(.data), seq(1, nrow(.data), 7))
keepData(subset = every.seventh)
head(.data)

## Selecting records under certain conditions
data(Familydata)
use(Familydata)
des()
.data
bmi <- wt/(ht/100)^2
label.var(bmi, "Body mass index (kg/m2)")
keepData(subset = ht > 120)
.data # Which record is missing?

use(Compaq)
des()
tab1(ses)
ses1 <- ses
pack()
keepData(subset=ses=="Rich")
tab1(ses)
tab1(ses1) # 'refactor' set to 'subset.vars' thus levels of ses1 not affected

## Reduction of variables
## Removal of consecutive variables
use(Familydata)
keepData(select = -(age:ht)) # Variables from 'age' to 'ht' removed
des()
## A better alternative would be:
use(Familydata)
keepData(exclude = age:ht)
des()
keepData(select = -c(1,3)) # Further removal of the first and
               # the third variables
des()
```

```
codebook()
## Targeting only a certain variables
data(Oswego)
use(Oswego)
des()
keepData(select = c(age, sex, ill, cakes:fruitsalad))
des()
keepData(select = c(1,2,5:7)) # Retain all variables except the third
                          #the the fourth
des()
# Note the repetition of '(subset)'



## Wildcard
use(Oswego)
des()
keepData(select = "c*") # The wildcard must be enclosed in quotes
des()

use(Oswego)
des()
keepData(exclude = "on*") # Variables having names starting with "on" removed
keepData(exclude = "???") # Variables having names with 3 characters removed
des() # Which are missing?
```

`lagVar` *Create a vector of lagged or subsequent value*

Description

Create a vector of lagged or subsequent value in a long form longitudinal data

Usage

```
lagVar(var, id, visit, lag.unit=-1)
```

Arguments

`var`	variable to create the lag
`id`	subject identification field
`visit`	visit of measurement
`lag.unit`	lag number of visits

Details

Data must be in long format having variable to create the lag, id and visit.

The variable 'visit' must be the number of visit, with step = 1.

The default value of lag.unit is -1. When the number is negative, the next measured is created instead.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'lag'

Examples

```
## Identification of the tree that became smaller during followup
data(Sitka, package="MASS")
use(Sitka)
table(Time)
visit <- Time
pack()
recode(visit, as.numeric(names(table(Time))), 1:5)
lag1.size <- lagVar(var=size, id=tree, visit=visit, lag=1)
data.frame(tree=tree, time=Time, visit=visit, size=size, lag1.size=lag1.size) [1:20,]
# Answer
data.frame(Time, tree, size, lag1.size) [which(lag1.size > size),]

# Alternatively
next.size <- lagVar(size, tree, visit, lag=-1)
data.frame(tree=tree, time=Time, size=size, next.size=next.size) [1:20,]
data.frame(Time, tree, size, next.size) [which(size > next.size),]
```

`List non-function objects`

List non-function objects

Description

List all objects visible in the global environment except user created functions.

Usage

```
lsNoFunction()
```

Details

Compared to standard 'ls()', this function displays only the subset of 'ls()' which are not functions.

The member of this list can be removed by 'zap()' but not the set of the functions created.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'use', 'detach', 'ls', 'rm'

Examples

```
object1 <- 1:5
object2 <- list(a=3, b=5)
function1 <- function(x) {x^3 +1}
ls()
lsNoFunction()

## To show only functions
as.character(lsf.str()[])
```

---

lookup *Recode several values of a variable*

Description

Systematic replacement of several values of a variable using an array

Usage

```
lookup(x, lookup.array)
```

Arguments

x	a variable
lookup.array	a n-by-2 array used for looking up the recoding scheme

Details

This command is used for changing more than one value of a variable using a n-by-2 look-up array. The first column of the look-up array (index column) must be unique.

If either the variable or the look-up table is character, the result vector will be character.

For changing the levels of a factor variable, 'recode(vars, "old level", "new level")' or 'levels(var) <- ' instead.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'replace', 'recode'

Examples

```
a       <- c( 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, NA)
tx      <- rbind(c(1,2),c(2,1),c(3,4),c(4,NA),c(NA,3))

# Swapping values of 1 and 2; rotating 3, 4 and NA
new.a  <- lookup(a, tx)
data.frame(a, new.a)
tableA <- table(a, new.a, exclude=NULL)
# All non-diagonal cells which are non-zero are the recoded cells.
```

```
print(tableA, zero=".")

## Character look-up table
b <- c(rep(letters[1:4],2), ".", NA)
tx1 <- cbind(c(letters[1:5], ".", NA), c("Disease A","Disease B","Disease C",
"Disease D","Disease E", NA, "Unknown"))
DiseaseName <- lookup(b, tx1)
data.frame(b, DiseaseName)
```

`lrtest` *Likelihood ratio test*

Description

Likelihood ratio test for objects of class 'glm'

Usage

```
lrtest (model1, model2)
```

Arguments

`model1`, `model2`	Two models of class "glm" having the same set of records and the same type ('family' and 'link')

Details

Likelihood ratio test checks the difference between -2*logLikelihood of the two models against the change in degrees of freedom using a chi-squared test. It is best applied to a model from 'glm' to test the effect of a factor with more than two levels. The records used in the dataset for both models MUST be the same. The function can also be used with "clogit", which does not have real logLikelihood.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'glm', 'logLik', 'deviance'

Examples

```
model0 <- glm(case ~ induced + spontaneous, family=binomial, data=infert)
model1 <- glm(case ~ induced, family=binomial, data=infert)
lrtest (model0, model1)
lrtest (model1, model0) # same result
lrtest (model1, model0) -> a
a
```

markVisits *Mark visits of subjects in a long format*

Description

Mark visits of subjects in a longitudinal data frame

Usage

```
markVisits(id, time)
```

Arguments

id	subject identification field
time	time of visit

Details

Visit numbers are essential in longitudinal data analysis. This function makes it easy for R user to do so.

The data frame must be sorted first by 'id' and 'time'

If visits marked by this function is going to be further used for calculation of lag difference, there must not be any missing visit in the data set.

Note

This was created from combination of the functions 'rle' and 'sapply'

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'rle', 'sapply', 'sortBy'

Examples

```
## Data frame
data(Sitka, package="MASS")
use(Sitka)

## Classical R methods
list1 <- rle(tree)
list1
visit1 <- unlist(sapply(X=list1$lengths, FUN=function(x) 1:x, simplify=FALSE))
visit1

## Do it again by Epicalc
sortBy(tree, Time)
visit2 <- markVisits(id=tree, time=Time)
visit2
```

Matched case-control study
Datasets on a matched case-control study of esophageal cancer

Description

Two different datasets for the same matched case-control study. VC1to6 has 1 case : varying number of controls (from 1 to 6) whereas VC1to1 has the number of control reduced to 1 for each case.

Usage

```
data(VC1to1)

data(VC1to6)
```

Format

A data frame with the following 5 variables.

`matset` a numeric vector indicating matched set number from 1 to 26

`case` a numeric vector: 1=case, 0=control

`smoking` a numeric vector: 1=smoker, 0=non-smoker

`rubber` a numeric vector: 1=exposed, 0=never exposed to rubber industry

`alcohol` a numeric vector: 1=drinker, 0=non-drinker

Source

Chongsuvivatwong, V. 1990 A case-control study of esophageal cancer in Southern Thailand. *J Gastro Hep* **5**:391–394.

See Also

'infert' in the datasets package.

Examples

```
data(VC1to6)
use(VC1to6)
des()
matchTab(case, alcohol, matset)
```

matchTab *Matched tabulation*

Description

Tabulation of outcome vs exposure from a matched case control study

Usage

```
matchTab (case, exposed, strata, decimal)
```

Arguments

case	Outcome variables where 0 = control and 1 = case
exposed	Exposure variable where 0 = non-exposed and 1 = exposed
strata	Identification number for each matched set
decimal	Number of digits displayed after the decimal point

Details

Tabulation for an unmatched case control study is based on individual records classified by outcome and exposure variables.

Matched tabulation is tallying based on each matched set. The simplest form is McNemar's table where only one case is matched with one control. 'matchTab' can handle 1:m matching where m can vary from 1 to m. A MLE method is then used to compute the conditional odds ratio.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'table', 'cc' and 'clogit'

Examples

```
.data <- infert
## Not run:

# matchTab(case, induced, stratum)
# Tabulation successful but OR not computed
# because 'induced' is not binary

## End(Not run)
attach(.data)
ia <- induced > 0  # any induced abortion
matchTab(case, ia, stratum)

# See also
clogit(case ~ ia + strata(stratum), data=infert)
detach(.data)
rm(list=ls())
```

merge with labels kept

Merge two data frames with variable labels kept

Description

Create a new data frame from merging two with variable labels of the data frame kept

Usage

```
## S3 method for class 'lab'
merge(x, y, ...)
```

Arguments

x, y	data frames to be merged to one
...	additional arguments passed on to 'merge'

Details

This is the 'merge' method exclusively used for objects of class 'data.frame'.

Epicalc can create and make use of variable labels extensively. Unfortunately, they are ignored by the function 'merge'.

The current method, 'merge.lab', carries the variable labels from both data frames into the results. If the labels from these two data frames are conflicting, that in 'x' will be used.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'label.var'

Examples

```
data1 <- data.frame(id = c("A","B"), age = c(12,25))
label.var(id, "personal id", dataFrame=data1)
label.var(age, "age in years", dataFrame=data1)
des(data1)
data2 <- data.frame(id= LETTERS, money = 1:26)
label.var(id, "Identification code", dataFrame=data2)
label.var(money, "money in dollar", dataFrame=data2)
des(data2)
merge(data1, data2) -> aa
des(aa) # No variable description
merge.lab(data1, data2) -> bb
des(bb)
merge.lab(data2, data1) -> cc
des(cc)
# Note the difference in description of 'id' between the three methods.
```

mhor *Mantel-Haenszel odds ratio*

Description

Mantel-Haenszel odds ratio calculation and graphing from a stratified case-control study

Usage

```
mhor(..., mhtable = NULL, decimal=2, graph = TRUE, design = "cohort")
```

Arguments

...	Three variables viz. 'outcome', 'exposure' and 'stratification'.
mhtable	a 2-by-2-by-s table, where s (strata) is more than one
decimal	number of decimal places displayed
graph	If TRUE (default), produces an odds ratio plot
design	Specification for graph; can be "case control","case-control", "cohort" or "prospective"

Details

'mhor' computes stratum-specific odds ratios and 95 percent confidence intervals and the Mantel-Haenszel odds ratio and chi-squared test is given as well as the homogeneity test. A stratified odds ratio graph is displayed.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'fisher.test', 'chisq.test'

Examples

```
data(Oswego)
use(Oswego)
cc(ill, chocolate)
mhor(ill, chocolate, sex)

mht1 <- table(ill, chocolate, sex)
dim(mht1)
mhor(mhtable=mht1) # same results
```

Montana — *Dataset on arsenic exposure and respiratory deaths*

Description

Dataset from a cohort study of exposure to arsenic from industry and deaths from respiratory diseases.

Usage

```
data(Montana)
```

Format

A data frame with 114 observations on the following 6 variables.

`respdeath` a numeric vector indicating number of deaths from respiratory diseases

`personyrs` a numeric vector indicating person-years of exposure

`agegr` a numeric vector: 1=40-49, 2=50-59, 3=60-69, 4=70-79)

`period` a numeric vector: 1=1938-1949, 2=1950-1959, 3=1960-1969, 4=1970-1977

`start` a numeric vector indicating starting period: 1=pre-1925, 2=1925 & after

`arsenic` a numeric vector indicating years of exposure: 1=<1 year, 2=1-4 years, 3=5-14 years, 4=15+ years

nptrend.test — *Non-parametric test for linear trend across ordered groups*

Description

Test for trend across ordered groups

Usage

```
## S3 method for class 'test'
nptrend(group, cont.var, score=NULL, alternative = "two.sided",
 decimal=3, ...)
```

Arguments

`group`	grouping variable
`cont.var`	a continuous variable
`score`	score for order of group. If not specified, the order of the group is used
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
`decimal`	number of digits to be displayed after the decimal point
`...`	further arguments passed to or used by methods.

Details

This is a variant of Kruskal-Wallis test where the group is ordinal instead of categorical. 'score' specifies relative position of each group.

Value

An object of class "nptrend" with title, z statistic, p-value, etc.

Author(s)

Jirawain Sopsuk < <jnid@hotmail.com>>

See Also

'kruskal.test'

Examples

```
group<-c(rep(1,6),rep(2,18),rep(3,8))
exposure<-c(1.4,1.4,1.4,1.6,2.3,2.3,0.9,1.0,1.1,1.1,1.2,1.2,1.5,1.9,2.2,2.6,
            2.6,2.6,2.8,2.8,3.2,3.5,4.3,5.1,0.8,1.7,1.7,1.7,3.4,7.1,8.9,13.5)
nptrend.test(group, exposure)
nptrend.test(group, exposure,score=c(1,2,5))
```

Oswego *Dataset from an outbreak of food poisoning in US*

Description

This dataset contains information on the records of 75 persons under investigation for the cause of acute food poisoning after a dinner party.

Usage

```
data(Oswego)
```

Format

A data frame containing 75 observations and 20 variables.

Source

EpiInfo package

References

See: https://www.cdc.gov/epiinfo/support/tutorials.html.

Examples

```
zap()
data(Oswego)
use(Oswego)
pyramid(age, sex)
```

`Outbreak investigation` *Dataset from an outbreak of food poisoning on a sportsday, Thailand 1990.*

Description

This dataset contains information from an outbreak investigation concerning food poisoning on a sportsday in Thailand 1990.

Dichotomous variables for exposures and symptoms were coded as follow:

0 = no
1 = yes
9 = missing or unknown

Usage

```
data(Outbreak)
```

Format

A data frame with 1094 observations on the following 13 variables.

`id` a numeric vector

`sex` a numeric vector

0 = female
1 = male

`age` a numeric vector: age in years

99 = missing

`exptime` an AsIs or character vector of exposure times

`beefcurry` a numeric vector: whether the subject had eaten beefcurry

`saltegg` a numeric vector: whether the subject had eaten salted eggs

`eclair` a numeric vector: pieces of eclair eaten

80 = ate but could not remember how much
90 = totally missing information

`water` a numeric vector: whether the subject had drunk water

`onset` an AsIs or character vector of onset times

`nausea` a numeric vector

`vomiting` a numeric vector

`abdpain` a numeric vector: abdominal pain

`diarrhea` a numeric vector

References

Thaikruea, L., Pataraarechachai, J., Savanpunyalert, P., Naluponjiragul, U. 1995 An unusual outbreak of food poisoning. *Southeast Asian J Trop Med Public Health* **26(1)**:78-85.

Examples

```
data(Outbreak)
use(Outbreak)

# Distribution of reported pieces of eclair taken
tab1(eclair)

# Defining missing value
recode(eclair, eclair>20, NA)
pieces.of.eclair <- cut(eclair, c(0,1,2,20), include.lowest=TRUE, right=FALSE)
tabpct(pieces.of.eclair, diarrhea)
```

---

poisgof *Goodness of fit test for modeling of count data*

Description

Poisson and negative binomial regression are used for modeling count data. This command tests the deviance against the degrees of freedom in the model thus determining whether there is overdispersion.

Usage

```
poisgof(model)
```

Arguments

model	A Poisson or negative binomial model

Details

To test the significance of overdispersion of the errors of a Poisson or negative binomial model, the deviance is tested against degrees of freedom using chi-squared distribution. A low P value indicates significant overdispersion.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'glm'

Examples

```
library(MASS)
quine.pois <- glm(Days ~ Sex/(Age + Eth*Lrn), data = quine, family=poisson)
poisgof(quine.pois)
quine.nb1 <- glm.nb(Days ~ Sex/(Age + Eth*Lrn), data = quine)
poisgof(quine.nb1)
```

Power *Power calculation for two sample means and proportions*

Description

Calculation of power given the results from a study

Usage

```
power.for.2p(p1, p2, n1, n2, alpha = 0.05)
power.for.2means(mu1, mu2, n1, n2, sd1, sd2, alpha = 0.05)
```

Arguments

p1, p2	probabilities of the two samples
n1, n2	sample sizes of the two samples
alpha	significance level
mu1, mu2	means of the two samples
sd1, sd2	standard deviations of the two samples

Details

These two functions compute the power of a study from the given arguments

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2means', 'n.for.2p'

Examples

```
# Suppose, in the example found in 'help(n.for.2p)',
# given the two proportions are .8 and .6 and the sample size
# for each group is 60.

power.for.2p(p1=.8, p2=.6, n1=60, n2=60) # 59 percent

# If the means of a continuous outcome variable in the same
# two groups were 50 and 60 units and the standard deviations were 30
# and 35 units, then the power to detect a statistical significance
# would be
```

```
power.for.2means(mu1=50, mu2=60, sd1=30, sd2=35, n1=60, n2=60)
# 39 percent. Note the graphic display
```

`print alpha` *Print alpha object*

Description

Print results related to Cronbach's alpha

Usage

```
## S3 method for class 'alpha'
print(x, ...)
```

Arguments

`x`	object of class 'alpha'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'tableStack'

Examples

```
data(Attitudes)
alpha(qa1:qa18, dataFrame=Attitudes) -> a
print(a)
a
```

`print cci` *Print cci results*

Description

Print results for cci and cc commands

Usage

```
## S3 method for class 'cci'
print(x, ...)
```

Arguments

`x`	object of class 'cci'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'cci'

Examples

```
cci(25, 22, 20, 7)
data(Oswego)
use(Oswego)
cc(ill, chocolate)
```

`print des` *Print 'des' results*

Description

Print description of data frame of a variable

Usage

```
## S3 method for class 'des'
print(x, ...)
```

Arguments

`x`	object of class 'des'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'des'

print kap.ByCategory *Print kap.ByCategory results*

Description

Print results for kap.Bycategory commands

Usage

```
## S3 method for class 'kap.ByCategory'
print(x, ...)
```

Arguments

x	object of class 'kap.ByCategory'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'kap.ByCategory'

print kap.table *Print kap.table results*

Description

Print results for kap.table commands

Usage

```
## S3 method for class 'kap.table'
print(x, ...)
```

Arguments

x	object of class 'kap.table'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'kap.table'

print lrtest *Print lrtest results*

Description

Print results for likelihood ratio test

Usage

```
## S3 method for class 'lrtest'
print(x, ...)
```

Arguments

x	object of class 'lrtest'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'logistic.display'

Examples

```
model0 <- glm(case ~ induced + spontaneous, family=binomial, data=infert)
model1 <- glm(case ~ induced, family=binomial, data=infert)
lrtest (model0, model1)
lrtest (model1, model0) -> a
a
```

print matchTab *Print matched tabulation results*

Description

Print matched tabulation results

Usage

```
## S3 method for class 'matchTab'
print(x, ...)
```

Arguments

x	object of class 'matchTab'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'matchTab'

`print n.for.2means` *Print n.for.2means results*

Description

Print results for sample size for hypothesis testing of 2 means

Usage

```
## S3 method for class 'n.for.2means'
print(x, ...)
```

Arguments

`x`	object of class 'n.for.2means'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2p'

Examples

```
n.for.2means(mu1 = 10, mu2 = 14, sd1=3, sd2=3.5)
n.for.2means(mu1 = 10, mu2 = 7:14, sd1=3, sd2=3.5) -> a
a
```

print n.for.2p *Print n.for.2p results*

Description

Print results for sample size for hypothesis testing of 2 proportions

Usage

```
## S3 method for class 'n.for.2p'
print(x, ...)
```

Arguments

x	object of class 'n.for.2p'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2p'

Examples

```
n.for.2p(p1=.1, p2=.2)
n.for.2p(p1=seq(1,9,.5)/10, p2=.5)
```

print n.for.cluster.2means *Print n.for.cluster.2means results*

Description

Print results for sample size for hypothesis testing of 2 means in cluster RCT

Usage

```
## S3 method for class 'n.for.cluster.2means'
print(x, ...)
```

Arguments

x	object of class 'n.for.cluster.2means'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.cluster.2means'

`print n.for.cluster.2p` *Print n.for.cluster.2p results*

Description

Print results for sample size for hypothesis testing of 2 proportions in cluster RCT

Usage

```
## S3 method for class 'n.for.cluster.2p'
print(x, ...)
```

Arguments

`x`	object of class 'n.for.cluster.2p'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.cluster.2p'

`print n.for.equi.2p` *Print n.for.equi.2p results*

Description

Print results for sample size for hypothesis testing of 2 proportions in equivalent trial

Usage

```
## S3 method for class 'n.for.equi.2p'
print(x, ...)
```

Arguments

`x`	object of class 'n.for.equi.2p'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2p'

Examples

```
n.for.equi.2p(p=.85, sig.diff=.05)
```

`print n.for.lqas` *Print n.for.lqas results*

Description

Print results for sample size for lot quality assurance sampling

Usage

```
## S3 method for class 'n.for.lqas'
print(x, ...)
```

Arguments

`x`	object of class 'n.for.lqas'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

Examples

```
n.for.lqas(p0 = 0.05, q=0)
n.for.lqas(p0 = (10:1)/100, q=0 ) -> a
a
```

print n.for.noninferior.2p *Print n.for.noninferior.2p results*

Description

Print results for sample size for hypothesis testing of 2 proportions in non-inferior trial

Usage

```
## S3 method for class 'n.for.noninferior.2p'
print(x, ...)
```

Arguments

x	object of class 'n.for.noninferior.2p'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2p'

Examples

```
n.for.noninferior.2p(p=.85, sig.inferior=.05)
```

print n.for.survey *Print n.for.survey results*

Description

Print results for sample size of a continuous variable

Usage

```
## S3 method for class 'n.for.survey'
print(x, ...)
```

Arguments

x	object of class 'n.for.survey'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2p'

Examples

```
n.for.survey(p=seq(5,95,5)/100)
```

`print nptrend.test` *Print nptrend.test object*

Description

Print a nptrend.test object

Usage

```
## S3 method for class 'nptrend.test'
print(x, ...)
```

Arguments

`x`	object of class 'nptrend.test'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'nptrend.test'

`print power.for.2means` *Print power.for.2means results*

Description

Print results for power for hypothesis testing of 2 means

Usage

```
## S3 method for class 'power.for.2means'
print(x, ...)
```

Arguments

`x`	object of class 'power.for.2means'
`...`	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2means'

Examples

```
power.for.2means(mu1 = 10, mu2=14,   n1=5,  n2=7,  sd1=3, sd2=3.5)
power.for.2means(mu1 = 10, mu2=7:14, n1=20, n2=25, sd1=3, sd2=3.5) -> a
a
```

---

print power.for.2p *Print power.for.2p results*

Description

Print results for power of hypothesis testing of 2 proportions

Usage

```
## S3 method for class 'power.for.2p'
print(x, ...)
```

Arguments

x	object of class 'power.for.2p'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'n.for.2p'

Examples

```
power.for.2p(p1=.1, p2=.2, n1=10, n2=15)
power.for.2p(p1=seq(1,9,.5)/10, p2=.5, n1=100, n2=120)
```

print statStack *Print statStack object*

Description

Print a statStack object

Usage

```
## S3 method for class 'statStack'
print(x, ...)
```

Arguments

x	object of class 'statStack'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com>»

See Also

'statStack'

print summ *Print 'summ' results*

Description

Print summary of data frame of a variable

Usage

```
## S3 method for class 'summ.default'
print(x, ...)
```

Arguments

x	object of class 'summ'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com>»

See Also

'summ'

print summary of data frame *Print 'summ.data.frame' results*

Description

Print summary of data frame of a variable

Usage

```
## S3 method for class 'summ.data.frame'
print(x, ...)
```

Arguments

x	object of class 'summ.data.frame'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'summ'

print tableStack *Print tableStack object*

Description

Print a tableStack object

Usage

```
## S3 method for class 'tableStack'
print(x, ...)
```

Arguments

x	object of class 'tableStack'
...	further arguments passed to or used by methods.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'tableStack'

Examples

```
data(Attitudes)
tableStack(qa1:qa18, dataFrame=Attitudes) -> a
print(a)
data(Ectopic)
tableStack(hia, gravi, by=outc, dataFrame=Ectopic) -> b
print(b)
```

`pyramid` *Population pyramid*

Description

Create a population pyramid from age and sex

Usage

```
pyramid (age, sex, binwidth = 5, inputTable = NULL, printTable = FALSE,
    percent = c("none", "each", "total"), col.gender = NULL,
    bar.label = "auto", decimal = 1, col = NULL, cex.bar.value = 0.8,
    cex.axis = 1, main = "auto", cex.main = 1.2, ...)
```

Arguments

`age`	a numeric variable for age
`sex`	a variable of two levels for sexes, can be numeric but preferrably factor with labelled levels or characters
`binwidth`	bin width of age for each bar
`inputTable`	a table to read in with two columns of sexes and rows of age groups
`printTable`	whether the output table should be displayed on the console
`percent`	whether the lengths of the bars should be calculated from freqencies (default), percentages of each sex or total percentages
`col.gender`	vector reflecting colours of the two gender
`bar.label`	whether the bars would be labelled with the values
`decimal`	number of decimals displayed in the percent output table
`col`	colour(s) of the bars
`cex.bar.value`	character extension factor of the bar labels
`cex.axis`	character extension factor of the axis
`main`	main title
`cex.main`	character extension factor of main title
`...`	graph options for the bars, e.g. col

Details

'pyramid' draws a horizontal bar graph of age by sex.

The parameters of graph (par) options can be applied to 'font.lab' and those of the bars, e.g. 'col' but not of others.

Other lower level graph commands should be only for adding a 'title'.

'bar.label' when set as "auto", will be TRUE when 'percent="each"' or 'percent="total"'

Value

When the variables age and sex are input arguments, the return object includes age group variable and the output table. The argument 'decimal' controls only decimals of the output displayed on the console but not the returned table.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'barplot', 'levels', 'table'

Examples

```
data(Oswego)
use(Oswego)
pyramid(age, sex)
pyramid(age, sex, bar.label = TRUE)
pyramid(age, sex, printTable=TRUE)
pyramid(age, sex, percent = "each", printTable=TRUE)
pyramid(age, sex, percent = "total", printTable=TRUE)
pyramid(age, sex, percent = "total", bar.label = FALSE)
pyramid(age, sex, percent = "total", cex.bar.value = .5)

pyramid(age, sex, col="red")
pyramid(age, sex, col=1:16) # Too colorful!
pyramid(age, sex, col.gender = c("pink","lightblue"))
output <- pyramid(age, sex, binwidth = 10, percent="each", decimal=2)
output
tabpct(output$ageGroup, chocolate)

pyramid(inputTable=VADeaths[,1:2], font.lab=4)
pyramid(inputTable=VADeaths[,1:2], font.lab=4, main=NULL)
title("Death rates per 100 in rural Virginia in 1940")
```

---

recode *Recode variable(s)*

Description

Change value(s) of variable(s) in the default data frame

Usage

```
recode(vars, ...)

## Default S3 method:
recode(vars, old.value, new.value, dataFrame = .data, ...)

## S3 method for class 'is.na'
recode(vars, new.value = 0, dataFrame = .data, ...)
```

Arguments

vars	a variable or variables with the same recoding scheme
old.value	original values or a condition
new.value	new values for all variables listed
dataFrame	a data frame
...	further arguments passed to or used by other methods.

Details

'recode' is very useful for recoding missing values but can also be used for other purposes.

'vars' can be a single variable or a list of variables in the format of list(var1, var2, var3) or c(var1, var2, var3), which will be recoded simultaneously under the same scheme.

Both 'old.value' and 'new.value' can be vectors of equal length. The elements of old.value and new.value will be matched by corresponding orders. However, 'new.value' can have a single element into which all the old values are recoded.

The argument 'old.value' can be also be a condition for recoding the 'vars' into the single new.value regardless of the old value.

Note that changing the value label of a variable's levels can be done with 'levels(var)[levels(var)=="old name"] <- "new name"'. However, Epicalc 'recode' is more efficient in changing several factors using the same scheme. See example.

All the 'recode'd vars are automatically 'pack'ed into the default data frame which is synchronize with the one in the search path.

'recode.is.na' is used to recode any missing value of one or more variable to a common 'new.value', which is zero by default.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'replace', 'lookup'

Examples

```
age       <- c( 37,  99,  24,  33,  31,  30,  26,  25)
systolic  <- c(120, 120, 110, 120, 130, 120, 888, 999)
diastolic <- c( 80,  80,  70,  80,  70, 999,  70,  60)
sick      <- c(  1,   2,   2,   1,   2,   2,   2,   NA)
treated   <- c(  2,   1,   2,   2,   1,   2,   2,   1)
yesno     <- c("Y", "N")
sick      <- factor(sick, labels=yesno)
treated   <- factor(treated, labels=yesno)
.data     <- data.frame(age, systolic, diastolic, sick, treated)
use(.data)
pack()
# 'pack() integate all variables into .data
# to avoid confusion with free vectors.


# The above lines generate a hypothetical data frame.
```

```
# In reality, one just exploits 'use("datafile")', if the "datafile" exists.
.data
summ()
recode(age, old.value=99, new.value=NA)
summ()
#recode(vars=c(systolic, diastolic), 999, NA) # The value 888 is not recoded.
summ()
recode(systolic, systolic > 250, NA)
summ()
table(sick, treated)
recode(vars=c(sick, treated), old.value="Y", new.value="yes")
table(sick, treated)

# Recode both sick and treated to "N" if sick status is missing.
recode(vars=c(sick,treated), is.na(sick), new.value="N")
table(sick, treated)

# Recode more than one old values
data(VCT)
use(VCT)
des()
table(A16); table(A17); table(A18)
recode(vars=A16:A18,  c("willing","willing if have money"), "willing")
table(A16); table(A17); table(A18)
# Recode two last categories to missing
recode(A16:A18, c("not relevant","not answer"), NA)
table(A16); table(A17); table(A18)

# Use 'recode.is.na' to recode NA to "missing data"
recode.is.na(vars=A16:A18, "missing data")
table(A16); table(A17); table(A18)
recode(vars=A4:A5, 999, NA)
summ()
# recode back from NA to 0
recode.is.na(vars=A4:A5) # Note that new value is 0 by default

# Swaping
data(Hakimi)
use(Hakimi)
des()
summ()
table(treatment)
recode(treatment, c(1,2), c(2,1))
table(treatment)
```

Rename *Rename variable(s) in the default data frame*

Description

Rename a variable or change a pattern of variable names.

Usage

```
rename(x1, x2, dataFrame = .data, ...)

## Default S3 method:
rename(x1, x2, dataFrame = .data, ...)

## S3 method for class 'var'
rename(x1, x2, dataFrame = .data, ...)

## S3 method for class 'pattern'
rename(x1, x2, dataFrame = .data, printNote=TRUE, ...)

ren(x1, x2, dataFrame = .data, ...)
```

Arguments

x1	a variable or a pattern among the names of the variables inside .data.
x2	new name or new pattern of the variable(s).

FUNCTION	'x1'	'x2'
'rename.var'	old variable	new variable
'rename.pattern'	old pattern	new pattern

dataFrame	a data frame, the variable(s) of which will be renamed
printNote	whether the table of old names and new names of the variables(s) should be printed out.
...	further arguments passed to or used by other methods.

Details

'rename.var' renames variable 'x1' to 'x2'. Both arguments may have the quotes omitted.

'rename.pattern' changes substring 'x1' in any names of variables inside .data to 'x2'. With 'printNote=TRUE', a table with columns of old and new variables will be displayed.

'rename.var' is called if 'x1' perfectly matches with a variable name. 'rename.pattern' is called if the pattern 'x1' is found as a substring among the variable names. Otherwise, an error will occur.

Finally, 'ren' is the abbreviated form of 'rename' without any suffix

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'recode' and 'label.var'

Examples

```
data(Oswego)
use(Oswego)
des()
rename.var("ill", "sick")
des()
# Note change of the 4th variable name

rename(timesupper, time.of.supper)
# Note that '.var' and the quotes '"' can be omitted.
# But not 'rename(timesupper, "time of supper")'. Why?

# Even shorter with 'ren'
ren(sex, gender)
des()

rename.pattern("ll", "LL")
des()
rename("onset", "onset_")
# '.pattern' can be omitted but not the quotes.
des()
```

`Risk.display` *Tables for multivariate odds ratio, incidence density etc*

Description

Display of various epidemiological modelling results in a medically understandable format

Usage

```
logistic.display(logistic.model, alpha = 0.05, crude = TRUE,
    crude.p.value = FALSE, decimal = 2, simplified = FALSE,html=FALSE,aic=FALSE)
clogistic.display(clogit.model, alpha = 0.05, crude=TRUE,
    crude.p.value=FALSE, decimal = 2, simplified = FALSE)
cox.display (cox.model, alpha = 0.05, crude=TRUE, crude.p.value=FALSE,
    decimal = 2, simplified = FALSE,html=FALSE,aic=FALSE)
regress.display(regress.model, alpha = 0.05, crude = FALSE,
    crude.p.value = FALSE, decimal = 2, simplified = FALSE,html=FALSE,aic=FALSE)
idr.display(idr.model, alpha = 0.05, crude = TRUE, crude.p.value = FALSE,
    decimal = 2, simplified = FALSE)
mlogit.display(multinom.model, decimal = 2, alpha = 0.05)
ordinal.or.display(ordinal.model, decimal = 3, alpha = 0.05)
tableGlm (model, modified.coeff.array, decimal)
## S3 method for class 'display'
print(x, ...)
```

Arguments

`logistic.model`	a model from a logistic regression
`clogit.model`	a model from a conditional logistic regression
`regress.model`	a model from a linear regression

cox.model	a model from a cox regression
idr.model	a model from a Poisson regression or a negative binomial regression
multinom.model	a model from a multinomial or polytomous regression
ordinal.model	a model from an ordinal logistic regression
alpha	significance level
crude	whether crude results and their confidence intervals should also be displayed
crude.p.value	whether crude P values should also be displayed if and only if 'crude=TRUE'
decimal	number of decimal places displayed
simplified	whether the display should be simplified
model	model passed from logistic.display or regress.display to tableGlm
modified.coeff.array	array of model coefficients sent to the function 'tableGlm' to produce the final output
x	object obtained from these 'display' functions
aic	whether aic value should also be displayed
html	results as html format
...	further arguments passed to or used by methods

Details

R provides several epidemiological modelling techniques. The functions above display these results in a format easier for medical people to understand.

The function 'tableGlm' is not for general use. It is called by other display functions to receive the 'modified.coeff.array' and produce the output table.

The argument 'simplified' has a default value of 'FALSE'. It works best if the 'data' argument has been supplied during creation of the model. Under this condition, the output has three parts. Part 1 (the first line) indicates the type of the regression and the outcome. For logistic regression, if the outcome is a factor then the referent level is shown. Part 2 shows the main output table where each independent variable coefficient is displayed. If the independent variable is continuous (class numeric) then name of the variable is shown (or the descriptive label if it exists). If the variable is a factor then the name of the level is shown with the referent level omitted. In this case, the name of the referent level and the statistic testing for group effects are displayed. An F-test is used when the model is of class 'lm'. If any of the indepedent variables has a missing value, the F-test results will be missing. A Likelihood Ratio test is performed when the model is of class 'glm' with 'family = binomial' or 'family = poisson' specified and for models of class 'coxph' and 'clogit'. These tests are carried out with the records available in the model, not necessary all records in the full 'data' argument. The number of records in the model is displayed in the part 3 of the output. When 'simplified=TRUE', the first and the last parts are omitted from the display.

The result is an object of class 'display' and 'list'. Their apparence on the R console is controlled by 'print.display'. The 'table' attribute of these 'display' objects are ready to write (using 'write.csv') to a .csv file which can then be copied to a manuscript document. This approach can substantially reduce both the time and errors produced due to conventional manual copying.

Value

'logistic.display', 'regress.display', 'clogit.display' and 'cox.display', each produces an output table. See 'details'.

Note

Before using these 'display' functions, please note the following limitations.

1) Users **should** define the 'data' argument of the model.

2) The names of the independent variables **must** be a subset of the names of the variables in the 'data' argument. Sometimes, one of more variables are omitted by the model due to collinearity. In such a case, users have to specify 'simplified=TRUE' in order to get the display function to work.

3) Under the following conditions, 'simplified' will be forced to TRUE and 'crude' forced to FALSE.

3.1) The names of the independent variables contain a function such as 'factor()' or any '\$' sign.

3.2) The levels of the factor variables contain any ':' sign.

3.3) There are more than one interaction terms in the model

3.4) The 'data' argument is missing in the conditional logistic regression and Cox regression model

4) For any other problems with these display results, users are advised to run 'summary(model)' or 'summary(model)$coefficients' to check the consistency between variable names in the model and those in the coefficients. The number in the latter may be fewer than that in the former due to collinearity. In this case, it is advised to specify 'simplified=TRUE' to turn off the attempt to tidy up the rownames of the output from 'summary(model)$coeffients'. The output when 'simplified=TRUE' is more reliable but less understandable.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'glm', 'confint'

Examples

```
model0 <- glm(case ~ induced + spontaneous, family=binomial, data=infert)
summary(model0)
logistic.display(model0)

data(ANCdata)
glm1 <- glm(death ~ anc + clinic, family=binomial, data=ANCdata)
logistic.display(glm1)
logistic.display(glm1, simplified=TRUE)

library(MASS) # necessary for negative binomial regression
data(DHF99); use(DHF99)
model.poisson <- glm(containers ~ education + viltype,
    family=poisson, data=.data)
model.nb <- glm.nb(containers ~ education + viltype,
    data=.data)
idr.display(model.poisson)  -> poiss
print(poiss) # or print.display(poiss) or poiss
idr.display(model.nb)  -> nb
print(nb)
nb # same result
# write.csv(nb$table, file="tablenb.csv")
getwd()
## You may go to this directory (folder) and have a look
```

```
    ## at the file using a spreadsheet programme.

    data(VC1to6)
    use(VC1to6)
    fsmoke <- factor(smoking)
    levels(fsmoke) <- list("no"=0, "yes"=1)
    pack()
    clr1 <- clogit(case ~ alcohol + fsmoke + strata(matset), data=.data)
    clogistic.display(clr1)

    data(BP)
    use(BP)
    age <- as.numeric(as.Date("2000-01-01") - birthdate)/365.25
    agegr <- pyramid(age,sex, bin=20)$ageGroup
    hypertension <- sbp >= 140 | dbp >=90
    pack()
    model1 <- glm(hypertension ~ sex + agegr + saltadd, family=binomial, data=.data)
    logistic.display(model1) -> table3
    attributes(table3)
    table3
    table3$table
    #write.csv(table3$table, file="table3.csv") # Note $table
    ## Have a look at this file in Excel, or similar spreadsheet program

    file.remove(file="table3.csv")
    model2 <- glm(hypertension ~ sex * age + sex * saltadd, family=binomial, data=.data)
    logistic.display(model2)
    # More than 1 interaction term so 'simplified turned to TRUE

reg1 <- lm(sbp ~ sex + agegr + saltadd, data=.data)
regress.display(reg1)
    # Note that 20 subjects had missing information on 'saltadd'
    # A model with 'saltadd' dropped will have 20 records more than 'reg1'
    # thus give missing value of F test P value.

    # remove NA from dataset
    bpp <- .data[complete.cases(.data),]
reg.bpp <- lm(sbp ~ sex + agegr + saltadd, data=bpp)
regress.display(reg.bpp)

data(Compaq)
    cox1 <- coxph(Surv(year, status) ~ hospital + stage * ses, data=Compaq)
    cox.display(cox1, crude.p.value=TRUE)

    # Ordinal logistic regression
    library(nnet)
    options(contrasts = c("contr.treatment", "contr.poly"))
    house.plr <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
    house.plr
    ordinal.or.display(house.plr)

    # Polytomous or multinomial logistic regression
    house.multinom <- multinom(Sat ~ Infl + Type + Cont, weights = Freq,
            data = housing)
    summary(house.multinom)
    mlogit.display(house.multinom, alpha=.01) # with 99 percent confidence limits.
```

ROC *ROC curve*

Description

Receiver Operating Characteristic curve of a logistic regression model and a diagnostic table

Usage

```
lroc(logistic.model, graph = TRUE, add = FALSE, title = FALSE,
    line.col = "red", auc.coords = NULL, grid = TRUE, grid.col = "blue",
cutoff=FALSE,snsp=FALSE, ...)
roc.from.table(table, graph = TRUE, add = FALSE, title = FALSE,
line.col = "red", auc.coords = NULL, grid = TRUE, grid.col = "blue", ...)
```

Arguments

logistic.model	A model from logistic regression
table	A cross tabulation of the levels of a test (rows) vs a gold standard positive and negative (columns)
graph	Draw ROC curve
add	Whether the line is drawn on the existing ROC curve
title	If true, the model will be displayed as main title
line.col	Color of the line
auc.coords	Coordinates for label of 'auc' (area under curve)
grid	Whether the grid should be drawn
grid.col	Grid colour, if drawn
cutoff	Display cut off point
snsp	Display sensitivity and specificity
...	Additional graphic parameters

Details

'lroc' graphs the ROC curve of a logistic regression model. If 'table=TRUE', the diagnostic table based on the regression will be printed out.

'roc.from.table' computes the change of sensitivity and specificity of each cut point and uses these for drawing the ROC curve.

In both cases, the area under the curve is computed.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'glm'

Examples

```
# Single ROC curve from logistic regression
# Note that 'induced' and 'spontaneous' are both originally continuous variables
model1 <- glm(case ~ induced + spontaneous, data=infert, family=binomial)
logistic.display(model1)
# Having two spontaneous abortions is quite close to being infertile!
# This is actually not a causal relationship

lroc(model1, title=TRUE, auc.coords=c(.5,.1))
# For PowerPoint presentation, the graphic elements should be enhanced as followed
lroc(model1, title=TRUE, cex.main=2, cex.lab=1.5, col.lab="blue", cex.axis=1.3, lwd=3)
lroc1 <- lroc(model1) # The main title and auc text have disappeared
model2 <- glm(case ~ spontaneous, data=infert, family=binomial)
logistic.display(model2)
lroc2 <- lroc(model2, add=TRUE, line.col="brown", lty=2)
legend("bottomright",legend=c(lroc1$model.description, lroc2$model.description),
        lty=1:2, col=c("red","brown"), bg="white")
title(main="Comparison of two logistic regression models")
lrtest(model1, model2)
# Number of induced abortions is associated with increased risk for infertility

# Various form of logistic regression
# Case by case data
data(ANCdata)
use(ANCdata)
glm1 <- glm(death ~ anc + clinic, binomial, data=.data)
lroc(glm1)

# Frequency format
data(ANCtable)
ANCtable
use(ANCtable)
death <- factor (death)
levels (death) <- c("no","yes")
anc <- factor (anc)
levels (anc) <- c("old","new")
clinic <- factor (clinic)
levels (clinic) <- c("A","B")
pack()
glm2 <- glm(death ~ anc + clinic, binomial, weights=Freq, data=.data)
lroc(glm2)

# yes/no format
.data$condition <- c(1,1,2,2,3,3,4,4)
data2 <- reshape (.data, timevar="death", v.name="Freq", idvar="condition", direction="wide")
data2
glm3 <- glm(cbind(Freq.yes, Freq.no) ~ anc + clinic, family=binomial, data=data2)
lroc(glm3)


# ROC from a diagnostic table
table1 <- as.table(cbind(c(1,27,56,15,1),c(0,0,10,69,21)))
colnames(table1) <- c("Non-diseased", "Diseased")
rownames(table1) <- c("15-29","30-44","45-59","60-89","90+")
table1
roc.from.table(table1)
```

```
roc.from.table(table1, title=TRUE, auc.coords=c(.4,.1), cex=1.2)

# Application of the returned list
roc1 <- roc.from.table(table1, graph=FALSE)
cut.points <- rownames(roc1$diagnostic.table)
text(x=roc1$diagnostic.table[,1], y=roc1$diagnostic.table[,2],
labels=cut.points, cex=1.2, col="brown")
```

`sampsize` *Sample size calculation*

Description

Sample size calculations for epidemiological studies

Usage

```
n.for.survey (p, delta = "auto", popsize = NULL, deff = 1, alpha = 0.05)
n.for.2means (mu1, mu2, sd1, sd2, ratio = 1, alpha = 0.05, power = 0.8)
n.for.cluster.2means (mu1, mu2, sd1, sd2, alpha = 0.05, power = 0.8,
      ratio = 1, mean.cluster.size = 10, previous.mean.cluster.size = NULL,
      previous.sd.cluster.size = NULL, max.cluster.size = NULL,
      min.cluster.size = NULL, icc = 0.1)
n.for.2p (p1, p2, alpha = 0.05, power = 0.8, ratio = 1)
n.for.cluster.2p (p1, p2, alpha = 0.05, power = 0.8, ratio = 1,
      mean.cluster.size = 10, previous.mean.cluster.size = NULL,
      previous.sd.cluster.size = NULL, max.cluster.size = NULL,
      min.cluster.size = NULL, icc = 0.1)
n.for.equi.2p(p, sig.diff, alpha=.05, power=.8)
n.for.noninferior.2p (p, sig.inferior, alpha = 0.05, power = 0.8)
n.for.lqas (p0, q = 0, N = 10000, alpha = 0.05, exact = FALSE)
```

Arguments

`p`	estimated probability
`delta`	difference between the estimated prevalence and one side of the 95 percent confidence limit (precision)
`popsize`	size of the finite population
`deff`	design effect for cluster sampling
`alpha`	significance level
`mu1`, `mu2`	estimated means of the two populations
`sd1`, `sd2`	estimated standard deviations of the two populations
`ratio`	n2/n1
`mean.cluster.size`	mean of the cluster size planned in the current study
`previous.mean.cluster.size`, `previous.sd.cluster.size`	mean and sd of cluster size from a previous study
`max.cluster.size`, `min.cluster.size`	maximum and minimum of cluster size in the current study

icc	intraclass correlation coefficient
p1, p2	estimated probabilities of the two populations
power	power of the study
sig.diff	level of difference consider as being clinically significant
sig.inferior	level of reduction of effectiveness as being clinically significant
p0	critical proportion beyond which the lot will be rejected
q	critical number of faulty pieces found in the sample, beyond which the lot will be rejected
N	lot size
exact	whether the exact probability is to be computed

Details

'n.for.survey' is used to compute the sample size required to conduct a survey.

When 'delta="auto"', delta will change according to the value of p. If 0.3 <= p <= 0.7, delta = 0.1. If 0.1 <= p < .3, or 0.7< p <=0.9, then delta=.05. Finally, if p < 0.1, then delta = p/2. If 0.9 < p, then delta = (1-p)/2.

When cluster sampling is employed, the design effect (deff) has to be taken into account.

'n.for.2means' is used to compute the sample size needed for testing the hypothesis that the difference of two population means is zero.

'n.for.cluster.2means' and 'n.for.cluster.2p' are for cluster (usually randomized) controlled trial.

'n.for.2p' is used to the compute the sample size needed for testing the hypothesis that the difference of two population proportions is zero.

'n.for.equi.2p' is used for equivalent trial with equal probability of success or fail being p for both groups. 'sig.diff' is a difference in probability considered as being clinically significant. If both sides of limits of 95 percent CI of the difference are within +sig.diff or -sig.diff, there would be neither evidence of inferiority nor of superiority of any arm.

'n.for.noninferior.2p' is similar to 'n.for.equi.2p' except if the lower limit of 95 percent CI of the difference is higher than the sig.inferior level, the hypothesis of inferiority would be rejected.

For a case control study, p1 and p2 are the proportions of exposure among cases and controls.

For a cohort study, p1 and p2 are proportions of positive outcome among the exposed and non-exposed groups.

'ratio' in a case control study is controls:case. In cohort and cross-sectional studies, it is non-exposed:exposed.

LQAS stands for Lot Quality Assurance Sampling. The sample size n is determined to test whether the lot of a product has a defective proportion exceeding a critical proportion, p0. Out of the sample tested, if the number of defective specimens is greater than q, the lot is considered not acceptable. This concept can be applied to quality assurance processes in health care.

When any parameter is a vector of length > 5, a table of sample size by the varying values of parameters is displayed.

Value

a list.

'n.for.survey' returns an object of class "n.for.survey"

'n.for.2p' returns an object of class "n.for.2p"

'n.for.2means' returns an object of class "n.for.2means"

'n.for.lqas' returns an object of class "n.for.lqas"

Each type of returned values consists of vectors of various parameters in the formula and the required sample size(s).

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

References

Eldridge SM, Ashby D, Kerry S. 2006 Sample size for cluster randomized trials: effect of coefficient of variation of cluster size and analysis method. *Int J Epidemiol* **35(5)**: 1292-300.

See Also

'power.for.2means', 'power.for.2p'

Examples

```
# In a standard survey to determine the coverage of immunization needed using
# a cluster sampling technique on a population of approximately 500000, and
# an estimated prevalence of 70 percent, design effect is assumed to be 2.

n.for.survey( p = .8, delta = .1, popsize = 500000, deff =2) # 123 needed

# To see the effect of prevalence on delta and sample size
n.for.survey( p = c(.5, .6, .7, .8, .9, .95, .99))

# Testing the efficacy of measles vaccine in a case control study .
# The coverage in the non-diseased population is estimated at 80 percent.
# That in the diseased is 60 percent.

n.for.2p(p1=.8, p2=.6) # n1=n2=91 needed

# A randomized controlled trial testing cure rate of a disease of
# 90 percent by new drugs  and 80 percent by the old one.

n.for.2p(p1=.9, p2=.8) # 219 subjects needed in each arm.

# To see the effect of p1 on sample size
n.for.2p(p1=seq(1,9,.5)/10, p2=.5) # A table output

# The same randomized trial to check whether the new treatment is 5 percent
# different from the standard treatment assuming both arms has a common
# cure rate of 85 percent would be

n.for.equi.2p(p=.85, sig.diff=0.05)  # 801 each.

# If inferior arm is not allow to be lower than -0.05 (5 percent less effective)
n.for.noninferior.2p(p=.85, sig.inferior=0.05)  # 631 each.

# A cluster randomized controlled trial to test whether training of village
# volunteers would result in reduction of prevalence of a disease from 50 percent
# in control villages to 30 percent in the study village with a cluster size
# varies from 250 to 500 eligible subjects per village (mean of 350) and the
```

```
# intraclass correlation is assumed to be 0.15

n.for.cluster.2p(p1=.5, p2=.3, mean.cluster.size = 350, max.cluster.size = 500,
min.cluster.size = 250, icc = 0.15)



# A quality assurance to check whether the coding of ICD-10 is faulty
# by no more than 2 percent.The minimum sample is required.
# Thus any faulty coding in the sample is not acceptable.

n.for.lqas(p0 = .02, q=0, exact = TRUE) # 148 non-faulty checks is required
# to support the assurance process.

n.for.lqas(p0 = (1:10)/100, q=0, exact = FALSE)
```

`setTitle` *Setting the displayed language of Epicalc graph title*

Description

Setting locale and internationalizing Epicalc graph title

Usage

```
setTitle(locale)
```

Arguments

`locale`	A string denoting international language of choice

Details

On calling 'library(epicalc)', '.locale()' has an inital value of FALSE, ie. the titles of Epicalc's automatic graphs are displayed in the English language. 'setTitle' has two effects. It selects the locale and resets the hidden object '.locale()' to TRUE. The command internationalizes the title of automatic graphs created by Epicalc according to 'locale' given in the function's argument.

If '.locale()' is TRUE, then the automatic graphs produced by Epicalc commands, such as 'summ(var)' or 'tab1(var)' or 'tabpct(var1,var2)', will lookup a language conversion table for the graph title and the title will be changed accordingly.

Internationalization of the title can be disabled by typing '.locale(FALSE)'. This has no effect of locale as a whole unless it is reset to English by issuing the command 'setTitle("English")'.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'Sys.setlocale', 'Sys.getlocale' and 'titleString'

Examples

```
use(iris)
summ(Sepal.Length, by=Species)
setTitle("English")
dotplot(Sepal.Length, by=Species)
setTitle("Malay")
dotplot(Sepal.Length, by=Species)
setTitle("Spanish")
dotplot(Sepal.Length, by=Species)
detach(.data)
rm(.data)
```

---

shapiro.qqnorm *Qqnorm plots with Shapiro-Wilk's test*

Description

Quantile-normal plots with Shapiro-Wilk's test result integrated

Usage

```
shapiro.qqnorm (x, ...)
```

Arguments

x	A numeric vector
...	Graphical parameters passed to 'par'

Details

To test a variable 'x' against the normal distribution, a qqnorm plot is integrated with the Shapiro-Wilk test to enhance interpretation.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'shapiro.test', 'qqnorm', 'par'

Examples

```
x <- rnorm(10)
a <- LETTERS[1:10]
shapiro.qqnorm(x, pch=a, col="red")
qqline(x, lty=2, col="black")
```

Sleepiness *Dataset on sleepiness in a workshop*

Description

Sleepiness among participants in a workshop

Usage

```
data(Sleep3)
```

Format

A data frame with 15 observations on the following 8 variables.

`id` a numeric vector

`gender` a factor with levels `male` `female`

`dbirth` a Date vector for birth date

`sleepy` a numeric vector for any experience of sleepiness in the class: `0=no` `1=yes`

`lecture` a numeric vector for ever felt sleepy during a lecture: `0=no` `1=yes`

`grwork` a numeric vector for ever felt sleepy during a group work: `0=no` `1=yes`

`kg` a numeric vector

`cm` a numeric vector

Examples

```
data(Sleep3)
use(Sleep3)
des()
```

Sort data frame by variable(s) *Sort data frame by variable(s)*

Description

Sort the whole dataset by one or more variables

Usage

```
sortBy(..., dataFrame = .data, inclusive = TRUE)
```

Arguments

`...`	index variable(s) used for sorting
`dataFrame`	Destination data frame where all variables of the same length are sorted
`inclusive`	whether vectors outside the default data frame should also be sorted

Details

The whole dataset can be sorted by an index variable(s) inside the (...).

If 'inclusive = TRUE', variables outside the data frame with same length will also be sorted.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'sort', 'order'

Examples

```
sbp <- c(120, 100, 110, 120, 140, 120,  NA,  NA)
dbp <- c( 80,  80,  70,  80,  70,  NA,  70,  60)
age <- c(37, 32, 24, 33, 31, 30, 26, 25)
data1 <- data.frame(sbp, dbp, age)
use(data1)
age2 <- age^2
sortBy(age, inclusive = FALSE)
age2 # unsorted
use(data1)
age2 <- age^2
sortBy(age, inclusive = TRUE)
age2 # sorted

des()
.data
sortBy(age, decreasing=TRUE)
.data

## Note that the argument of 'sortBy' must not be concatenated vectors
data(Familydata)
use(Familydata)
.data
sortBy(money, sex) # correct
.data
use(Familydata) # Read in the dataset afresh
sortBy(c(money, sex)) # errors.
.data
```

---

statStack *Statistics of a continuous variable stratified by factors*

Description

Compares the difference in means or medians of the levels of a factor or list of factors

Usage

```
statStack (cont.var, by, dataFrame=.data, iqr="auto", var.labels = TRUE, decimal = 1,
    assumption.p.value = .01)
```

Arguments

cont.var	a single continuous variable in the data frame
by	a factor, or list of factors, each of length <code>nrow(dataFrame)</code>
iqr	to display median and inter-quartile range instead of mean and sd.
var.labels	use descriptions of the 'by' variables if available
dataFrame	source data frame of the variables
decimal	number of digits displayed after decimal point
assumption.p.value	level of Bartlett's test P value to judge whether the comparison and the test should be parametric

Details

This function computes means/medians of a continuous variable in each level of the specified factor(s) and performs an appropriate statistical test.

The classes of the variable to compute statistics must be either 'integer' or 'numeric' why all 'by' variables must be 'factor'.

Like in 'tableStack', the argument 'iqr' has a default value being "auto". Non-parametric comparison and test will be automatically chosen if Bartlette's test P value is below the 'assumption.p.value'.Like in 'tableStack', the default value for the 'iqr' argument is "auto", which means non-parametric comparison and test will be automatically chosen if the P-value from Bartlett's test is below the value of the 'assumption.p.value' argument (0.01).

The user can force the function to perform a parametric test by setting 'iqr=NULL' and to perform a non-parametric test by setting 'iqr' to the name or index of the continuous variable.

By default, 'var.labels=TRUE' in order to give nice output.

Value

an object of class 'statStack' and 'table'

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'tableStack'

Examples

```
statStack(Price, by=c(DriveTrain, Origin), dataFrame=Cars93)
statStack(Price, by=c(DriveTrain, Origin), dataFrame=Cars93, iqr=NULL)

Cars93$log10.Price <- log10(Cars93$Price)# added as the 28th variable
statStack(log10.Price, by=c(DriveTrain, Origin), dataFrame=Cars93)
statStack(log10.Price, by=c(DriveTrain, Origin), dataFrame=Cars93, iqr=28)
rm(Cars93)

data(Compaq)
statStack(year, by=c(hospital, stage:ses), dataFrame=Compaq)
# Note that var.labels 'Age group' is displayed instead of var. name 'agegr'
```

summ *Summary with graph*

Description

Summary of data frame in a convenient table. Summary of a variable with statistics and graph

Usage

```
summ(x, ...)
## Default S3 method:
summ(x=, by=NULL, graph = TRUE, box = FALSE, pch = 18,
    ylab = "auto", main = "auto", cex.X.axis = 1, cex.Y.axis = 1,
    dot.col = "auto", ...)
## S3 method for class 'factor'
summ(x, by=NULL, graph=TRUE, ...)
## S3 method for class 'logical'
summ(x, by=NULL, graph=TRUE, ...)
## S3 method for class 'data.frame'
summ(x=.data, ...)
```

Arguments

x	'x' can be a data frame or a vector
by	a stratification variable, valid only when x is a vector
graph	automatic plot (sorted dot chart) if 'x' is a vector
box	add a boxplot to the graph (by=NULL)
pch	plot characters
ylab	annotation on Y axis
main	main title of the graph
cex.X.axis	character extension scale of X axis
cex.Y.axis	character extension scale of Y axis
dot.col	colour(s) of plot character(s)
...	additional graph parameters

Details

For data frames, 'summ' gives basic statistics of each variable in the data frame. The other arguments are ignored.

For single vectors, a sorted dot chart is also provided, if graph=TRUE (default).

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'summary', 'use', 'des'

Examples

```
data(Oswego)
.data <- Oswego
summ(.data)
with(.data, summ(age))
with(.data, summ(age, box=TRUE))
with(.data, summ(age, dot.col="brown"))
with(.data, summ(age, by=sex))
# Changing dot colours
with(.data, summ(age, by=sex, dot.col = c("blue","orange")))
# Enlarging main title and other elements
with(.data, summ(age, by=sex, cex.main=1.5, cex.X.axis=1.5, cex.Y.axis=1.7))

# Free vector
summ(rnorm(1000))
summ((1:100)^2, by=rep(1:2, 50))
summ((1:100)^2, by=rep(c("Odd","Even"), 50), main="Quadratic distribution by odd and even numbers")
```

---

tab1 *One-way tabulation*

Description

One-way tabulation with automatic bar chart and optional indicator variables generation

Usage

```
tab1(x0, decimal = 1, sort.group = FALSE,
    cum.percent = !any(is.na(x0)), graph = TRUE,
    missing = TRUE, bar.values = "frequency",
    horiz = FALSE, cex = 1, cex.names = 1, main = "auto", xlab = "auto",
    ylab = "auto", col = "auto", gen.ind.vars = FALSE, ...)

## S3 method for class 'tab1'
print(x, ...)
```

Arguments

x0	a variable
decimal	number of decimals for the percentages in the table
sort.group	pattern for sorting categories in the table and in the chart. Default is no sorting. Others are "decreasing" and "increasing"
cum.percent	presence of cumulative percentage in the output table. Default is TRUE for a variable without any missing values.
graph	whether a graph should be shown
missing	include the missing values category or <NA> in the graphic display
bar.values	include the value of "frequency"", "percentage"" or "none"" at the end of each bar
horiz	set the bar chart to horizontal orientation

cex	parameter for extension of characters or relative size of the bar.values
cex.names	character extension or relative scale of the name labels for the bars
main	main title of the graph
xlab	label of X axis
ylab	label of Y axis
col	colours of the bar
gen.ind.vars	whether the indicator variables will be generated
x	object of class 'tab1' obtained from saving 'tab1' results
...	further arguments passed to or used by other methods

Details

'tab1' is an advanced one-way tabulation providing a nice frequency table as well as a bar chart. The description of the variable is also used in the main title of the graph.

The bar chart is vertical unless the number of categories is more than six **and** any of the labels of the levels consists of more than 8 characters **or** 'horiz' is set to TRUE.

For table has less than categories, the automatic colour is "grey". Otherwise, the graph will be colourful. The argument, 'col' can be overwritten by the user.

The argument 'gen.ind.vars' is effective only if x0 is factor.

Value

Output table

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'tabpct', 'label.var', 'table', 'barplot', 'model.matrix'

Examples

```
tab1(state.division)
tab1(state.division, bar.values ="percent")
tab1(state.division, sort.group ="decreasing")
tab1(state.division, sort.group ="increasing")
tab1(state.division, col=c("chocolate","brown1","brown4"),
  main="Number of states in each zone")
# For presentation, several 'cex' parameters should increase
tab1(state.division, col=c("chocolate","brown1","brown4"),
  main="Number of states in each zone",
  cex.main=1.7, cex.name=1.2, cex.axis=1.3, cex.lab=1.3)


data(Oswego)
use(Oswego)
tab1(ill) # Note the column of cumulative percentages in the table.
tab1(ill, cum.percent=FALSE)
tab1(chocolate)
# Due to missing values, cumulative percentages are now automatically turned off.
```

```
tab1(chocolate, cum.percent=TRUE)
# Slightly too many columns in text!
tab1(chocolate, missing=FALSE, bar.values="percent")
agegr <- cut(age, breaks=c(0,10,20,30,40,50,60,70,80))
tab1(agegr)
tab1(agegr, col="grey") # graphic output from older versions of 'tab1'
tab1(agegr, col=c("red","yellow","blue")) # Colours recycled
tab1(agegr, horiz=TRUE)

# Keeping output table
dev.off()
tab1(agegr, graph = FALSE) -> a
print(a)
a # same results
attributes(a)
a$output.table
class(a$output.table) # "matrix"
# 'a$output.table' is ready for exporting to a .csv file by
# write.csv(a$output.table, file="table1.csv")
# "table1.csv" is now readable by a spreadsheet program


# Generating indicator variables
data(Compaq)
use(Compaq)
des()
# Illustration of indicator variables
head(cbind( ses, model.matrix(~ses -1)))
tab1(ses, gen=TRUE) # indicator variables of 'ses' have been generated
ls() # Four new names starting with 'ses'
pack()
des()
```

`tableStack` *Tabulation of variables in a stack form*

Description

Tabulation of variables with the same possible range of distribution and stack into a new table with or without other descriptive statistics or to breakdown distribution of more than one row variables against a column variable

Usage

```
tableStack (vars, dataFrame =.data, minlevel = "auto", maxlevel = "auto", count = TRUE,
    na.rm =FALSE, means = TRUE, medians = FALSE, sds = TRUE, decimal = 1,
    total = TRUE, var.labels = TRUE, var.labels.trunc =150, reverse = FALSE,
    vars.to.reverse = NULL, by = NULL, vars.to.factor = NULL, iqr = "auto",
    prevalence = FALSE, percent = c("column", "row", "none"), frequency=TRUE,
   test = TRUE, name.test = TRUE, total.column = FALSE, simulate.p.value = FALSE,
    sample.size=TRUE, assumption.p.value = .01)
```

Arguments

vars	a vector of variables in the data frame
dataFrame	source data frame of the variables
minlevel	possible minimum value of items specified by user
maxlevel	possible maximum value of items specified by user
count	whether number of valid records for each item will be displayed
na.rm	whether missing value would be removed during calculation mean score of each person
means	whether means of all selected items will be displayed
medians	whether medians of all selected items will be displayed
sds	whether standard deviations of all selected items will be displayed
decimal	number of decimals displayed in the statistics
total	display of means and standard deviations of total and average scores
var.labels	presence of descriptions of variables on the last column of output
var.labels.trunc	number of characters used for variable description
reverse	whether item(s) negatively correlated with other majority will be reversed
vars.to.reverse	variable(s) to reverse
by	a variable for column breakdown. If a single character (with quotes) is given, only the 'total column' will be displayed
vars.to.factor	variable(s) to be converted to factor for tabulaton
iqr	variable(s) to display median and inter-quartile range
prevalence	for logical variable, whether prevalence of the dichotomous row variable in each column subgroup will be displayed
percent	type of percentage displayed when the variable is categorical. Default is column
frequency	whether to display frequency in the cells when the variable is categorical
test	whether statistical test(s) will be computed
name.test	display name of the test and relevant degrees of freedom
total.column	whether to add 'total column' to the output or not
simulate.p.value	simulate P value for Fisher's exact test
sample.size	whether to display non-missing sample size of each column
assumption.p.value	level of Bartlett's test P value to judge whether the comparison and the test should be parametric

Details

This function simultaneously explores several variables with a fixed integer rating scale. For non-factor variables, the default values for tabulation are the minimum and the maximum of all variables but can be specified by the user.

When 'by' is omitted, all variables must be of the same class, and must be 'integer', 'factor' or 'logical.

Unlike function 'alpha', the argument 'reverse' has a default value of FALSE. This argument is ignored if 'vars.to.reverse' is specified.

Options for 'reverse', 'vars.to.reverse' and statistics of 'means', 'medians', 'sds' and 'total' are available only if the items are not factor. To obtain statistics of factor items, users need to use 'unclassDataframe' to convert them into integer.

When the 'by' argument is given, 'reverse' and 'vars.to.reverse' do not apply. Instead, columns of the 'by' variable will be formed. A table will be created against each selected variable. If the variable is a factor or coerced to factor with 'vars.to.factor', cross-tabulation will result with percents as specified, ie. "column", "row", or "none" (FALSE). For a dichotomous row variable, if set to 'TRUE', the prevalence of row variable in the form of a fraction is displayed in each subgroup column. For objects of class 'numeric' or 'integer', means with standard deviations will be displayed. For variables with residuals that are not normally distributed or where the variance of subgroups are significantly not normally distributed (using a significance level of 0.01), medians and inter-quartile ranges will be presented if the argument 'iqr' is set to "auto" (by default). Users may specify a subset of the selected variables (from the 'vars' argument) to be presented in such a form. Otherwise, the argument could be set as any other character string such as "none", to insist to present means and standard deviations.

When 'test = TRUE' (default), Pearson's chi-squared test (or a two-sided Fisher's exact test, if the sample size is small) will be carried out for a categorical variable or a factor. Parametric or non-parametric comparison and test will be carried out for a object of class 'numeric' or 'integer' (See 'iqr' and 'assumption.p.value' below). If the sample size of the numeric variable is too small in any group, the test is omitted and the problem reported.

For Fisher's exact test, the default method employs 'simulate.p.value = FALSE'. See further explanation in 'fisher.test' procedure. If the dataset is extraordinarily large, the option may be manually set to TRUE.

When 'by' is specified as a single character object (such as 'by="none"'), there will be no column breakdown and all tests will be omitted. Only the total column is displayed. Only the 'total' column is shown.

If this 'total column' is to accompany the 'by' breakdown, the argument 'total.column=TRUE' should be specified. The 'sample.size' is TRUE by default. The total number of records for each group is displayed in the first row of the output. However, the variable in each row may have some missing records, the information on which is not reported by tableStack.

By default, Epicalc sets 'var.labels=TRUE' in order to give nice output. However, 'var.labels=FALSE' can sometimes be more useful during data exploration. Variable numbers as well as variable names are displayed instead of variable labels. Names and numbers of abnormally distributed variables, especially factors with too many levels, can be easily identified for further relevelling or recoding.

The argument 'iqr' has a default value being "auto". Non-parametric comparison and test will be automatically chosen if Bartlett's test P value is below the 'assumption.p.value'.

The test can be forced to parametric by setting 'iqr=NULL' and to non-parametric by if iqr is set to the variable number of cont.var (See examples.).

Value

an object of class 'tableStack' and 'list' when by=NULL

`results`	an object of class 'noquote' which is used for print out
`items.reversed`	name(s) of variable(s) reversed
`total.score`	a vector from 'rowSums' of the columns of variables specified in 'vars'
`mean.score`	a vector from 'rowMeans' of the columns of variables specified in 'vars'

`mean.of.total.scores`
: mean of total scores

`sd.of.total.scores`
: standard deviation of total scores

`mean.of.average.scores`
: mean of mean scores

`sd.of.average.scores`
: standard deviation of mean scores

When 'by' is specified, an object of class 'tableStack' and 'table is returned.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'table', 'tab1', 'summ', 'alpha', 'unclassDataframe'

Examples

```
data(Oswego)
tableStack(bakedham:fruitsalad, dataFrame=Oswego)
.data <- Oswego
des(.data)
attach(.data)
tableStack(bakedham:fruitsalad, .data) # Default data frame is .data
tableStack(bakedham:fruitsalad, .data, by= ill)
tableStack(bakedham:fruitsalad, .data, by= ill, prevalence=TRUE)
tableStack(bakedham:fruitsalad, .data, by= ill, percent=FALSE)
tableStack(bakedham:fruitsalad, .data, by= ill, percent=FALSE, name.test=FALSE)
detach(.data)


data(Cars93, package="MASS")
.data <- Cars93
des(.data)
tableStack(vars=4:25, .data, by=Origin)
tableStack(vars=4:25, .data, by="none")
tableStack(vars=4:25, .data, by=Origin, total.column=TRUE)


data(Attitudes)
.data <- Attitudes
attach(.data)
tableStack(qa1:qa18, .data)  # May need full screen of Rconsole
tableStack(qa1:qa18, .data, var.labels.trunc=35)
                     # Fits in with default R console screen
tableStack(qa1:qa18, .data, reverse=TRUE) -> a
a
## Components of 'a' have appropriate items reversed
a$mean.score -> mean.score
a$total.score -> total.score
.data$mean.score <- mean.score
.data$total.score <- total.score
rm(total.score, mean.score)
detach(.data)
```

```
attach(.data)
tableStack(c(qa1,qa13:qa18,mean.score,total.score), .data, by=sex, test=FALSE)
tableStack(c(qa15, qa17, mean.score:total.score), .data, by=sex, iqr=c(qa17,total.score))
tableStack(c(qa15, qa17, mean.score:total.score), .data, by=dep, iqr=c(qa17,total.score))
## 'vars' can be mixture of different classes of variables
.data$highscore <- mean.score > 4
tableStack(mean.score:highscore, .data, by=sex, iqr=total.score)
detach(.data)
rm(list=ls())

data(Ectopic)
.data <- Ectopic
des(.data)
tableStack(vars=3:4, .data, by=outc)
tableStack(vars=3:4, .data, by=outc, percent="none")
tableStack(vars=3:4, .data, by=outc, prevalence = TRUE)
tableStack(vars=3:4, .data, by=outc, name.test = FALSE)

## Variable in numeric or factor
data(Outbreak)
.data <- Outbreak
des(.data)
# Comparison of exposure to food items between the two gender
tableStack(vars=5:8, .data, by=sex) # as continuous varaibles
tableStack(vars=5:8, .data, by=sex, vars.to.factor = 5:8) # as factors
```

`tabpct` *Two-way tabulation with mosaic plot*

Description

Two-way tabulation with automatic mosaic plot

Usage

```
tabpct(row, column, decimal = 1, percent = "both",
    graph = TRUE, las = 0, main = "auto", xlab = "auto",
    ylab = "auto", col = "auto", ...)
```

Arguments

`row`, `column`	variables
`decimal`	number of decimals for the percentage in the table
`percent`	orientation of the percentage in the table. Can be "both", "col", or "row"
`graph`	automatic graphing
`las`	orientation of group labelling
`main`	main title
`xlab`	X axis label
`ylab`	Y axis label
`col`	colours of the bars

... additional arguments for 'table'

0: always parallel to axis

1: always horizontal,

2: always perpendicular to the axis,

3: always vertical.

Details

'tabpct' gives column and row percent cross-tabulation as well as mosaic plot.

The width of the bar in the plot denotes the relative proportion of the row variable.

Inside each bar, the relative proportion denotes the distribution of column variables within each row variable.

Note that 'row' and 'col' arguments of this function are for the table, not the mosaic plot and the default value for the 'percent' orientation is "both".

Due to limitation of 'mosaicplot', certain graphic parameters such as 'cex.main', 'cex.lab' are not acceptable. The parameter 'main', 'xlab' and 'ylab' can be suppressed by making equal to " ". An additional line starting with 'title' can be used to write new main and label titles with 'cex.main' and 'cex.lab' specified.

Value

Tables of row and column percentage

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'tab1', 'table', 'mosaicplot'

Examples

```
data(Oswego)
use(Oswego)
agegr <- cut(age, breaks=c(0,20,40,60,80))
label.var(agegr, "age group")
tabpct(agegr, ill)
tabpct(agegr, ill, cex.axis=1) # enlarge value labels
# To increase the size of the various titles:
tabpct(agegr, ill, cex.axis=1, main="", xlab="", ylab="", col=c("blue","purple"))
title(main="Diseased by Age group", cex.main=1.8,
    xlab="Age (years)",ylab="Diseased", cex.lab=1.5)
```

`tally events` *Tally a date variable by larger time unit*

Description

Tabulate dates breakdown by week, month or year and plot the results

Usage

```
## S3 method for class 'events'
tally(x, by=NULL, breaks=c("day","week","month","year"),
     graph=TRUE, type="l", line.col="auto", legend = TRUE, legend.site="topright",
       legend.bg = "white", ylim="auto", cex=1, addmargins=TRUE, ...)
```

Arguments

`x`	a date variable
`by`	a grouping elements
`breaks`	time unit for aggregation of the events
`graph`	whether the table be plotted
`type`	graph type
`line.col`	line colour
`legend`	wheter the legend will be produced if there are more than one groups
`legend.site`	a single character string indicating location of the legend. See details of ?legend
`legend.bg`	background colour of the legend
`ylim`	Y axis limits
`cex`	character expanding factor for the legend
`addmargins`	whether the margin values of the cross-table will be computed
`...`	additional graphic parameters passed on to other methods

Details

This function produces table of events by day, month or year with zero cells included. It also plots the table.

'by' is a grouping variable, usually a factor.

'breaks' can be "day", "week", "month" and "year".

'type' can be "l", "p", "b", "c", "o", "h", "s" and "S" only when there is only group (by=NULL). Otherwise, graph type will be 'l'.

'line.col' control line colours in the graph and the legend

If 'legend = TRUE" (by default), a legend box is automatically drawn on the "topright" corner of the graph. This character string can be changed to others such as, "topleft", "center", etc (see examples).

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'dotplot'

Examples

```
random.dates <- as.Date("2001/1/1") + round(20*stats::runif(100))
tally.events(random.dates)
tally.events(random.dates, las=2)
tally.events(random.dates, las=2, type = "h", lwd =2, ylim = c(0,20))
random.dates2 <- as.Date("2001/1/1") + round(500*stats::runif(100))
gender100 <- c(rep("F", 50),rep("M", 50))
tally.events(random.dates2, las=2, breaks="week")
tally.events(random.dates2, las=2, breaks="month", type="h")
tally.events(random.dates2, breaks="week", by=gender100)
tally.events(random.dates2, breaks="month", by=gender100, cex=2, line.col=c("blue","brown"), lwd=2)
```

Timing exercise *Dataset on time going to bed, waking up and arrival at a workshop*

Description

This dataset came from an interview survey on the workshop attendants on 2004-12-14.

Note that the date of bed time is 2004-12-13 if the respondent went to bed before midnight.

Usage

```
data(Timing)
```

Format

A data frame with 18 observations on the following 11 variables.

`id` a numeric vector

`gender` a factor with levels `male` `female`

`age` a numeric vector

`marital` a factor with levels `single` `married` `others`

`child` a numeric vector indicating number of children

`bedhr` a numeric vector indicating the hour of going to bed

`bedmin` a numeric vector indicating the minute of going to bed

`wokhr` a numeric vector indicating the hour of waking up

`wokmin` a numeric vector indicating the minute of waking up

`arrhr` a numeric vector indicating the hour of arrival at the workshop

`arrmin` a numeric vector indicating the minute of arrival at the workshop

Examples

```
data(Timing)
use(Timing)
des()
arrival.time <- ISOdatetime(year=2004, month=12, day=14, hour=arrhr,
min=arrmin, sec=0)
summary(arrival.time, by= gender)
```

`titleString` *Replace commonly used words in Epicalc graph title*

Description

Setting vocabularies for Epicalc graph title

Usage

```
titleString (distribution.of = .distribution.of, by = .by,
frequency = .frequency, locale = .locale(),
return.look.up.table=FALSE)
```

Arguments

`distribution.of`	A string denoting "Distribution of"
`by`	That for "by"
`frequency`	That for "Frequency"
`locale`	Logical value to overwrite .locale(). The initial value is FALSE
`return.look.up.table`	Should the look-up table be returned?

Details

The two internationalization commands of Epicalc, 'setTitle' and 'titleString', work together to set the langauge and wording of titles of automatic graphs obtained from certain Epicalc functions.

In general, 'setTitle' is simple and works well if the locale required fits in with the version of the operating system. The three commonly used words in the graph titles: "Distribution of", "by" and "Frequency", which are in English, are initially stored in three respective hidden objects '.distribution.of', '.by' and '.frequency' as well as in the look-up table within the 'titleString' function. When the locale is changed to a language other than English, the look-up table is used and wordings are changed accordingly.

The function 'titleString' is useful when the user wants to change the strings stored in the look-up table. It changes the initial values of '.distribution.of', '.by' and '.frequency', respectively. The argument, 'locale', must be manually set to FALSE by the user to disable the use of the look-up table and to enable the use of the three objects assigned by the command instead.

The two functions suppress each other. Use of 'setTitle' disables the effects of 'titleString', switching .locale() to TRUE and forcing Epicalc to read from the look-up table in 'titleString'. However, 'setTitle' does not overwrite the values assigned by the arguments of 'titleString'.

The key and decisive switch object is .locale(). Once .locale() is set to FALSE, either manually or inside the 'titleString' command, the values of the three hidden objects will be used. Setting .locale() to TRUE, either manually or automatically by the 'setTitle' function, points the graph title to use the look-up table inside 'titleString'.

Typing 'titleString()' without an argument displays the current contents of these three objects. The look-up table is also displayed if the return.look.up.table argument is set to TRUE.

International users who want to add their specific locales and corresponding terminology to the look-up table or to suggest more appropriate terminology can contact the author.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'setTitle'

Examples

```
.data <- iris
attach(.data)
dotplot(Sepal.Length, by=Species)

titleString(distribution.of="", by="grouped by", locale=FALSE)
## The above command is equivalent to the following three lines:
## .distribution.of <- ""
## .by <- "grouped by"
## .locale(FALSE)

dotplot(Sepal.Length, by=Species)
titleString()

setTitle("English")
dotplot(Sepal.Length, by=Species)
titleString(return.look.up.table=TRUE)

.locale(FALSE)
dotplot(Sepal.Length, by=Species)
titleString()

.distribution.of <- "Distribution of"
titleString()

.by <- "by"
titleString()

detach(.data)
rm(.data)
```

Tooth decay *Dataset on tooth decay and mutan streptococci*

Description

Relationship between bacteria and presence of any decayed tooth.

Usage

```
data(Decay)
```

Format

A data frame with 436 observations on the following 2 variables.

`decay` a numeric vector indicating presence of tooth decay

`strep` a numeric vector indicating number of colony-forming-units (CFUs) of *Streptococcus mutan* in the saliva

Source

Teanpaisan, R., Kintarak, S., Chuncharoen, C., Akkayanont, P. 1995 Mutans Streptococci and dental -caries in schoolchildren in Southern Thailand. *Community Dentistry and Oral Epidemiology* **23**: 317-318.

Examples

```
data(Decay)
use(Decay)
des()
```

Unclass factors in a dataframe

Unclass factor(s) in the default data frame

Description

This function unclasses factor(s) in the default data frame (.data).

Usage

```
unclassDataframe (vars, dataFrame = .data)
```

Arguments

`vars`	a vector of variables in the data frame, usually factors, that will be unclassed
`dataFrame`	data frame containing the variables

Details

This function 'unclass'es several variables of class factor to their corresponding integer values. This is useful in further summation of items.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'unclass', 'alpha', 'tableStack'

Examples

```
expect1 <- c(3,4,3,2,5,3,2,5,2,4,4,3,2,4,4,
   1,3,2,4,4,4,3,4,2,4,5,4,4,3,4)
expect2 <- c(3,2,4,3,5,3,4,5,4,4,5,5,3,4,4,
   3,4,2,3,5,3,4,4,2,4,5,4,4,3,5)
found1  <- c(1,3,4,3,4,3,3,2,2,4,5,4,3,4,3,
   1,1,2,3,4,4,1,1,3,4,5,4,1,4,2)
found2  <- c(1,1,2,1,3,1,1,2,2,4,3,3,1,1,3,
   3,1,1,2,1,1,1,1,1,3,5,4,4,1,1)
.data <- data.frame(expect1, expect2, found1, found2)
use(.data)
pack() # clean up
des()
level.lab <- list("Very poor"=1, "Poor"=2,
   "Fair"=3, "Good"=4, "Very good"=5)
for (i in 1:4) {
   .data[,i] <- factor(.data[,i])
   levels(.data[,i]) <- level.lab
}
des() # All variables are now factors
unclassDataframe(vars=c(1,4))
des() # Only variables #1 and #4 are 'unclass'ed
```

---

use *Command to read in and attach data*

Description

Command to read in data from Stata, SPSS, EpiInfo and .csv formats in addition to any R data frame

Usage

```
use(filename, dataFrame = .data, clear = TRUE, spss.missing = TRUE,
           tolower = TRUE, tofactor=TRUE, sheet=1)
```

Arguments

filename	a character object ending with one of the following: .dbf, .dta, .sav, .rec, .csv (file with comma and header); data frames in R requires no quote
dataFrame	destination data frame where the read object is store
clear	equal to 'detachAllData()' before reading in the data set and attaching it to the search path
spss.missing	whether the values planned for missing for the SPSS dataset should be replaced with NA
tolower	whether all the names of the variables should be forced to lower case (only if the original file has one the following extensions: '.dbf', 'rec' and '.sav')
tofactor	logical: should character vectors be converted to factors?
sheet	The name or index of the sheet to read data from.

Details

'use' reads in datasets from Dbase (.dbf), Stata (.dta), SPSS(.sav), EpiInfo(.rec) and Comma separated value (.csv) formats as well as R data frames. The destination data frame is saved in memory, by default as '.data', and automatically attached to the search path. This setting is the basis for other commands of 'epicalc' including 'des', 'summ', 'recode', 'label.var' etc.

The 'use' command overwrites the destination data frame ('.data') with the new one.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'read.table', 'read.dta', 'read.SPSS', etc and 'detachAllData'

Examples

```
# data(BOD)
library(foreign)
write.dta(BOD, file="BOD.dta")
rm(list=ls())
ls()
use("BOD.dta", clear=FALSE)

# The above lines write Stata format from R data frame.
# In reality, one just types 'use("filename.dta")', if the file is available.
des()
file.remove("BOD.dta")

# A better way to read an R dataset for exploration with Epicalc is
 use(BOD, clear=FALSE)
 des()
 summ()
```

Variable manipulation *Variable manipulation*

Description

Label a variable; integrate outside variable(s) into .data.

Usage

```
label.var(var, label, pack = TRUE, dataFrame = .data)
pack(dataFrame = .data)
```

Arguments

var	A variable inside .data or a free vector.
label	A short description of the variable
pack	Remove the original free variable?
dataFrame	Destination data frame where all variables of the same length are labeled or packed into

Details

A data frame imported from Stata or SPSS can have 'variable labels', which is adopted as an attribute (but not used) by R.

Epicalc exploits this attribute by displaying the labels in the output from 'des()' and graphs following 'summ(var)', tab1(var)', 'tabpct(var1, var2)'.

For free vector(s) (variables outside the data frame), 'label.var' appends vector(s) to the data frame specified with the descriptive label attached. For variables already in the data frame, the command simply attaches the label to the variable. The 'var.labels' attribute is updated in the data frame. The argument 'pack', if TRUE, removes the original vector. This is useful to avoid redundancy and confusion.

More than one free vector of the same length can be integrated into the data frame (.data) without labelling, using 'pack()'.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'use','des'

Examples

```
sbp <- c(120, 100, 110, 120, 140, 120,  NA,  NA)
dbp <- c( 80,  80,  70,  80,  70,  NA,  70,  60)
.data <- data.frame(sbp, dbp)
use(.data)
pack()
des()
label.var(sbp, "systolic BP")
label.var(dbp, "diastolic BP")
des()
pp <- sbp - dbp # This is a new free vector in the global environment.
summ(pp) # unlabelled
label.var(pp, "pulse pressure")
des()
summ(pp)

## Silly things to do. Just for demonstration.
pp2 <- pp^2
pp3 <- pp^3
pack()
.data
```

Voluntary counselling and testing

Dataset on attitudes toward VCT

Description

This dataset contains information on the records of 200 women working at a tourist destination community.

Usage

```
data(VCT)
```

Format

A subset of a data frame containing 200 observations and 12 variables with variable descriptions.

Details of the codes can be seen from the results of the function 'codebook()' below.

Examples

```
data(VCT)
use(VCT)
des()
codebook()
```

Xerophthalmia and respiratory infection

Dataset from an Indonesian study on vitamin A deficiency and risk of respiratory infection

---

Description

This dataset was adopted from Diggle et al: Analysis of Longitudinal Data. REFERENCE – Zeger and Karim, JASA (1991)

Note that there are some duplications of id and time combination.

Usage

```
data(Xerop)
```

Format

A data frame containing 1200 observations and 10 variables.

`id` a numeric vector for personal identification number

`respinfect` whether the child had respiratory infection in that visit

`age.month` current age in month

`xerop` whether the child currently had vitamin A deficiency

`sex` gender of the child no detail on the code

`ht.for.age` height for age

`stunted` whether the child has stunted growth

`time` time of scheduled visit

`baseline.age` baseline age

`season` season

Examples

```
data(Xerop)
```

zap *Remove and detach all*

Description

Detach and remove all objects and data frames from the global environment

Usage

```
zap()
```

Details

The R command 'attach()' copies the data frame in the argument into a data frame in the search path (usually the second position) consequently making all the variables in the data frame easy to refer to. However, changing any element of the index data frame has no effect on the one in the search path unless the changed data frame is attached to the search path again. Having too many data frames in the search path subsequently causes confusion, not to mention an increase in memory usage. It is a good practice to detach the index data frame first before manipulating it and then attaching to it again. 'detachAllData()' is a self explanatory command which solves the over-attaching problem.

'zap()' is a combination of 'detachAllData()' and removal of non-function objects in the R workspace.

At the commencement of a new session, 'zap()' can be quite useful to clean the objects left over from previous R sessions and detach from any unwanted data frames.

'zap()' as well as 'rm(list=ls())' do not remove any objects starting with a dot '.', which are meant to be hidden. Therefore the object '.data' is resistant to 'zap()'.

Author(s)

Virasakdi Chongsuvivatwong < <cvirasak@gmail.com»

See Also

'use', 'detach', 'ls', 'rm'

Examples

```
object1 <- 1:5
object2 <- list(a=3, b=5)
function1 <- function(x) {x^3 +1}
attach(CO2)
lsNoFunction()
ls()
search()
detachAllData()
ls()
search()
zap()
ls()
search()
rm(function1)
```

Index