Package 'Publish'

March 27, 2019

Type Package

Title Format Output of Various Routines in a Suitable Way for Reports and Publication

Description
A bunch of convenience functions that transform the results of some basic statistical analyses into table format nearly ready for publication. This includes descriptive tables, tables of logistic regression and Cox regression results as well as forest plots.

Version 2018.04.17

Maintainer Thomas A. Gerds <tag@biostat.ku.dk>

Depends prodlim (>= 1.5.4)

Imports survival (>= 2.38), data.table (>= 1.10.4), lava (>= 1.5.1), multcomp (>= 1.4)

Suggests lme4 (>= 1.1), mets (>= 1.1.1), pec (>= 2.4.4), riskRegression (>= 1.4.3), testthat, smcfcs (>= 1.3.0), rms (>= 5.1), mitools (>= 2.3), nlme (>= 3.1-131)

License GPL (>= 2)

RoxygenNote 6.0.1

NeedsCompilation no

Author Thomas A. Gerds [aut, cre],
Christian Torp-Pedersen [ctb],
Klaus K Holst [ctb],
Brice Ozenne [aut]

Repository CRAN

Date/Publication 2018-04-17 21:24:49 UTC

R topics documented:

`Publish-package` *Publish package*

Description

This package processes results of descriptive statistcs and regression analysis into final tables and figures of a manuscript

ci.mean *Compute mean values with confidence intervals*

Description

Compute mean values with confidence intervals

Usage

```
ci.mean(x, ...)
```

Arguments

x	object passed to methods
...	passed to methods

Details

Normal approximation

Value

a list with mean values and confidence limits

ci.mean.default *Compute mean values with confidence intervals*

Description

Compute mean values with confidence intervals

Usage

```
## Default S3 method:
ci.mean(x, alpha = 0.05, normal = TRUE, na.rm = TRUE,
  statistic = "arithmetic", ...)
```

Arguments

x	numeric vector
alpha	level of significance
normal	If TRUE use quantile of t-distribution else use normal approximation and quantile of normal approximation. Do you think this is confusing?
na.rm	If TRUE remove missing values from x.
statistic	Decide which mean to compute: either "arithmetic" or "geometric"
...	not used

Details

Normal approximation

Value

a list with mean values and confidence limits

Author(s)

Thomas Gerds

`CiTable` *CiTable data*

Description

These data are used for testing Publish package functionality.

Format

A data frame with 27 observations on the following 9 variables.

Drug

Time

Drug.Time

Dose

Mean

SD

n

HazardRatio

lower

upper

p

Examples

```
data(CiTable)
labellist <- split(CiTable[,c("Dose","Mean","SD","n")],CiTable[,"Drug"])
labellist
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")], labels=labellist)
```

coxphSeries *Run a series of Cox regression models*

Description

Run a series of Cox regression analyses for a list of predictor variables and summarize the results in a table. The Cox models can be adjusted for a fixed set of covariates

This function runs on coxph from the survival package.

Usage

```
coxphSeries(formula, data, vars, ...)
```

Arguments

formula	The fixed part of the regression formula. For univariate analyses this is simply Surv(time,status)~1 where Surv(time,status) is the outcome variable. When the aim is to control the effect of vars in each element of the series by a fixed set of variables it is Surv(time,status)~x1+x2 where again Surv(time,status) is the outcome and x1 and x2 are confounders.
data	A data.frame in which the formula gets evaluated.
vars	A list of variable names, the changing part of the regression formula.
...	passed to publish.coxph

Value

matrix with results

Author(s)

Thomas Alexander Gerds

Examples

```
library(survival)
data(pbc)
## collect hazard ratios from three univariate Cox regression analyses
pbc$edema <- factor(pbc$edema,levels=c("0","0.5","1"),labels=c("0","0.5","1"))
uni.hr <- coxphSeries(Surv(time,status==2)~1,vars=c("edema","bili","protime"),data=pbc)
uni.hr

## control the logistic regression analyses for age and gender
## but collect only information on the variables in `vars'.
controlled.hr <- coxphSeries(Surv(time,status==2)~age+sex,vars=c("edema","bili","protime"),data=pbc)
controlled.hr
```

`Diabetes` *Diabetes data of Dr John Schorling*

Description

These data are courtesy of Dr John Schorling, Department of Medicine, University of Virginia School of Medicine. The data consist of 19 variables on 403 subjects from 1046 subjects who were interviewed in a study to understand the prevalence of obesity, diabetes, and other cardiovascular risk factors in central Virginia for African Americans. According to Dr John Hong, Diabetes Mellitus Type II (adult onset diabetes) is associated most strongly with obesity. The waist/hip ratio may be a predictor in diabetes and heart disease. DM II is also agssociated with hypertension - they may both be part of "Syndrome X". The 403 subjects were the ones who were actually screened for diabetes. Glycosolated hemoglobin > 7.0 is usually taken as a positive diagnosis of diabetes.

Format

A data frame with 205 observations on the following 12 variables.

id subject id

chol Total Cholesterol

stab.glu Stabilized Glucose

hdl High Density Lipoprotein

ratio Cholesterol/HDL Ratio

glyhb Glycosolated Hemoglobin

location a factor with levels (Buckingham,Louisa)

age age (years)

gender male or female

height height (inches)

height.europe height (cm)

weight weight (pounds)

weight.europe weight (kg)

frame a factor with levels (small,medium,large)

bp.1s First Systolic Blood Pressure

bp.1d First Diastolic Blood Pressure

bp.2s Second Diastolic Blood Pressure

bp.2d Second Diastolic Blood Pressure

waist waist in inches

hip hip in inches

time.ppn Postprandial Time when Labs were Drawn in minutes

AgeGroups Categorized age

BMI Categorized BMI

Source

http://192.38.117.59/~tag/Teaching/share/data/Diabetes.html

References

Willems JP, Saunders JT, DE Hunt, JB Schorling: Prevalence of coronary heart disease risk factors among rural blacks: A community-based study. Southern Medical Journal 90:814-820; 1997
Schorling JB, Roach J, Siegel M, Baturka N, Hunt DE, Guterbock TM, Stewart HL: A trial of church-based smoking cessation interventions for rural African Americans. Preventive Medicine 26:92-101; 1997.

Examples

```
data(Diabetes)
```

`fixRegressionTable` *Expand regression coefficient table*

Description

Expand regression coefficient table

Usage

```
fixRegressionTable(x, varnames, reference.value, reference.style = NULL,
  factorlevels, scale = NULL, nmiss, intercept)
```

Arguments

`x`	object resulting from `lm`, `glm` or `coxph`.
`varnames`	Names of variables
`reference.value`	Reference value for reference categories
`reference.style`	Style for showing results for categorical variables. If `"extraline"` show an additional line for the reference category.
`factorlevels`	Levels of the categorical variables.
`scale`	Scale for some or all of the variables
`nmiss`	Number of missing values
`intercept`	Intercept

Details

This function expands results from "regressionTable" with extralines and columns

For factor variables the reference group is shown. For continuous variables the units are shown and for transformed continuous variables also the scale. For all variables the numbers of missing values are added.

Value

a table with regression coefficients

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

`followupTable` *Summary tables for a given followup time point.*

Description

Summarize baseline variables in groups defined by outcome at a given followup time point

Usage

```
followupTable(formula, data, followup.time, compare.groups, ...)
```

Arguments

`formula`	Formula A formula whose left hand side is a `Hist` object. In some special cases it can also be a `Surv` response object. The right hand side is as in `utable`.
`data`	A data.frame in which all the variables of `formula` can be interpreted.
`followup.time`	Time point at which to evaluate outcome status.
`compare.groups`	Method for comparing groups.
`...`	Passed to `utable`. All arguments of `utable` can be controlled in this way except for `compare.groups` which is set to `"Cox"`. See details.

Details

If `compare.groups!=FALSE`, p-values are obtained from stopped Cox regression, i.e., all events are censored at follow-up time. A univariate Cox regression model is fitted to assess the effect of each variable on the right hand side of the formula on the event hazard and shown is the p-value of `anova(fit)`, see `anova.coxph`.

Value

Summary table.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

univariateTable

Examples

```
library(survival)
data(pbc)
pbc$edema <- factor(pbc$edema,levels=c("0","0.5","1"),labels=c("0","0.5","1"))
pbc$sex <- factor(pbc$sex,levels=c("m","f"),labels=c("m","f"))
followupTable(Hist(time,status)~age+edema+sex,data=pbc,followup.time=1000)
```

formatCI *Formatting confidence intervals*

Description

Format confidence intervals

Usage

```
formatCI(x, lower, upper, show.x = FALSE, handler = "sprintf",
  format = "[l;u]", degenerated = "asis", digits = 2, nsmall = digits,
  trim = TRUE, sep = "", reference.pos, reference.label = "", ...)
```

Arguments

x	not used (for compatibility with format)
lower	Numeric vector of lower limits
upper	Numeric vector of upper limits
show.x	Logical. If TRUE show value of x in front of confidence interval.
handler	Function to format numeric values. Default is sprintf, also supported are format and prettyNum
format	Character string in which l will be replaced by the value of the lower limit (argument lower) and u by the value of the upper upper limit. For example, (l,u) yields confidence intervals in round parenthesis in which the upper and lower limits are comma separated. Default is [l;u].
degenerated	String to show when lower==upper. Default is '–'
digits	If handler format or prettyNum used format numeric vectors.
nsmall	If handler format or prettyNum used format numeric vectors.
trim	Used to aligning resulting intervals. Also, if handler format or prettyNum used format numeric vectors .
sep	Field separator
reference.pos	Position of factor reference
reference.label	Label for factor reference
...	passed to handler

Details

The default format for confidence intervals is [lower; upper].

Value

String vector with confidence intervals

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

plot.ci ci.mean

Examples

```
x=ci.mean(rnorm(10))
formatCI(lower=x[3],upper=x[4])
formatCI(lower=c(0.001,-2.8413),upper=c(1,3.0008884))
# change format
formatCI(lower=c(0.001,-2.8413),upper=c(1,3.0008884),format="(l, u)")
# show x
formatCI(x=x$mean,lower=x$lower,upper=x$upper,format="(l, u)",show.x=TRUE)

# if the first lower limit is shorter than the second (missing negative sign),
# then, option trim will make a difference:
formatCI(lower=c(0.001,-2.8413),upper=c(1,3.0008884),format="l--u",trim=FALSE)
formatCI(lower=c(0.001,-2.8413),upper=c(1,3.0008884),format="l--u",trim=TRUE)

# change of handler function
l <- c(-0.0890139,0.0084736,144.898333,0.000000001)
u <- c(0.03911392,0.3784706,3338944.8821221,0.00001)
cbind(format=formatCI(lower=l,upper=u,format="[l;u)",digits=2,nsmall=2,handler="format"),
    prettyNum=formatCI(lower=l,upper=u,format="[l;u)",digits=2,nsmall=2,handler="prettyNum"),
    sprintf=formatCI(lower=l,upper=u,format="[l;u)",digits=2,nsmall=2,handler="sprintf"))
```

---

glmSeries *Run a series of generalized linear regression analyses*

Description

Run a series of generalized linear regression analyses for a list of predictor variables and summarize the results in a table. The regression models can be adjusted for a fixed set of covariates.

Usage

```
glmSeries(formula, data, vars, ...)
```

Arguments

formula	The fixed part of the regression formula. For univariate analyses this is simply y~1 where y is the outcome variable. When the aim is to control the effect of vars in each element of the series by a fixed set of variables it is y~x1+x2 where again y is the outcome and x1 and x2 are confounders.
data	A data.frame in which we evaluate the formula.
vars	A list of variable names, the changing part of the regression formula.
...	passed to glm

Value

Matrix with regression coefficients, one for each element of vars.

Author(s)

Thomas Alexander Gerds

Examples

```
data(Diabetes)
Diabetes$hyper1 <- factor(1*(Diabetes$bp.1s>140))
## collect odds ratios from three univariate logistic regression analyses
uni.odds <- glmSeries(hyper1~1,vars=c("chol","hdl","location"),data=Diabetes,family=binomial)
uni.odds
## control the logistic regression analyses for age and gender
## but collect only information on the variables in `vars'.
controlled.odds <- glmSeries(hyper1~age+gender,
                             vars=c("chol","hdl","location"),
                             data=Diabetes, family=binomial)
controlled.odds
```

---

labelUnits *labelUnits*

Description

Label output tables

Usage

```
labelUnits(x, ...)
```

Arguments

x	A matrix obtained with univariateTable.
...	not used

Details

Modify labels and values of variables in summary tables

Value

The re-labeled matrix

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

univariateTable

Examples

```
data(Diabetes)
tab <- summary(univariateTable(gender~AgeGroups+chol+waist,data=Diabetes))
publish(tab)
ltab <- labelUnits(tab,"chol"="Cholesterol (mg/dL)","<40"="younger than 40")
publish(ltab)

## pass labels immediately to utable
utable(gender~AgeGroups+chol+waist,data=Diabetes,
      "chol"="Cholesterol (mg/dL)","<40"="younger than 40")

## sometimes useful to state explicitly which variables value
## should be re-labelled
utable(gender~AgeGroups+chol+waist,data=Diabetes,
      "chol"="Cholesterol (mg/dL)","AgeGroups.<40"="younger than 40")
```

---

lazyFactorCoding *Efficient coding of factor levels*

Description

This function eases the process of generating factor variables with relevant labels. All variables in a data.frame with less than a user set number of levels result in a line which suggests levels and labels. The result can then be modified for use.

Usage

```
lazyFactorCoding(data, max.levels = 10)
```

Arguments

`data`	Data frame in which to search for categorical variables.
`max.levels`	Treat non-factor variables only if the number of unique values less than max.levels. Defaults to 10.

Details

The code needs to be copy-and-pasted from the R-output buffer into the R-code buffer. This can be customized for the really efficiently working people e.g. in emacs.

Value

R-code one line for each variable.

Author(s)

Thomas Alexander Gerds

Examples

```
data(Diabetes)
lazyFactorCoding(Diabetes)
```

org *Wrapper function for publish with output format org*

Description

Wrapper for `publish(...,org=TRUE)`

Usage

```
org(x, ...)
```

Arguments

`x`	object to format as org
`...`	passed to publish

Value

See publish

Author(s)

Thomas Alexander Gerds

parseInteractionTerms *Parse interaction terms*

Description

Parse interaction terms for regression tables

Usage

```
parseInteractionTerms(terms, xlevels, units, format.factor, format.contrast,
  format.scale, format.scale.unit, sep = ": ", ...)
```

Arguments

`terms`	Terms of a formula
`xlevels`	Factor levels corresponding to the variables in `terms`
`units`	named list with unit labels. names should match variable names in formula.
`format.factor`	For categorical variables. A string which specifies the format for factor labels. The string should contain the keywords `"var"` and `"level"` which will be replaced by the name of the variable and the current level, respectively. Default is `"var(level)"`.
`format.contrast`	For categorical variables. A string which specifies the format for constrast statements. The string should contain the keywords `"var"`, `"level"` and `"ref"` which will be replaced by the name of the variable, the current level and the reference level, respectively.

`format.scale` For continuous variables. For categorical variables. A string which specifies the format for factor labels. The string should contain the keywords `"var"` and `"level"` which will be replaced by the name of the variable and the current level, respectively. Default is `"var(level)"`.

`format.scale.unit` For continuous variables which have a unit. A string which specifies the format for factor labels. The string should contain the keywords `"var"` and `"unit"` which will be replaced by the name of the variable and the unit, respectively. Default is `"var(unit)"`.

`sep` a character string to separate the terms. Default is `": "`.

`...` Not yet used

Details

Prepare a list of contrasts which combines regression coefficients to describe statistical interactions.

Value

List of contrasts which can be passed to `lava::estimate`.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

lava::estimate

Examples

```
tt <- terms(formula(SBP~age+sex*BMI))
xlev <- list(sex=c("male","female"),BMI=c("normal","overweight","obese"))
parseInteractionTerms(terms=tt,xlevels=xlev)
parseInteractionTerms(terms=tt,xlevels=xlev,format.factor="var level")
parseInteractionTerms(terms=tt,xlevels=xlev,format.contrast="var(level:ref)")

tt2 <- terms(formula(SBP~age*factor(sex)+BMI))
xlev2 <- list("factor(sex)"=c("male","female"))
parseInteractionTerms(terms=tt2,xlevels=xlev2)
parseInteractionTerms(terms=tt2,xlevels=xlev2,units=list(age="yrs"))


data(Diabetes)
fit <- glm(bp.2s~age*factor(gender)+BMI,data=Diabetes)
parseInteractionTerms(terms=terms(fit$formula),xlevels=fit$xlevels,
                      format.scale="var -- level:ref",units=list("age"='years'))
parseInteractionTerms(terms=terms(fit$formula),xlevels=fit$xlevels,
                      format.scale.unit="var -- level:ref",units=list("age"='years'))
it <- parseInteractionTerms(terms=terms(fit$formula),xlevels=fit$xlevels)
ivars <- unlist(lapply(it,function(x)attr(x,"variables")))
lava::estimate(fit,function(p)lapply(unlist(it),eval,envir=sys.parent(-1)))
```

plot.ci *Plot confidence intervals*

Description

Function to plot confidence intervals

Usage

```
## S3 method for class 'ci'
plot(x, xlim, xlab = "", labels, ...)
```

Arguments

x	List, data.frame or other object of this form containing point estimates (first element) and the corresponding confidence intervals as elements lower and upper.
xlim	Limit of the x-axis
xlab	Label for the y-axis
labels	labels
...	Used to transport arguments to plotConfidence.

Details

Function to plot means and other point estimates with confidence intervals

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
data(Diabetes)
x=ci.mean(bp.2s~AgeGroups,data=Diabetes)
plot(x,title.labels="Age groups",xratio=c(0.4,0.3))
x=ci.mean(bp.2s/500~AgeGroups+gender,data=Diabetes)
plot(x,xratio=c(0.4,0.2))
plot(x,xratio=c(0.4,0.2),
     labels=split(x$labels[,"AgeGroups"],x$labels[,"gender"]),
     title.labels="Age groups")
## Not run:
plot(x, leftmargin=0, rightmargin=0)
plotConfidence(x, leftmargin=0, rightmargin=0)

data(CiTable)
with(CiTable,plotConfidence(x=list(HazardRatio),
                            lower=lower,
                            upper=upper,
                            labels=CiTable[,2:6],
                            factor.reference.pos=c(1,10,19),
                            format="(u-l)",
                            points.col="blue",
```

```
                                digits=2))

with(CiTable,Publish::plot.ci(x=list(HazardRatio),
                                lower=lower,
                                upper=upper,
                                labels=CiTable[,2:6],
                                factor.reference.pos=c(1,10,19),
                                format="(u-l)",
                                points.col="blue",
                                digits=2,
                                leftmargin=-2,
                                title.labels.cex=1.1,
                                labels.cex=0.8,values.cex=0.8))

## End(Not run)
```

`plot.regressionTable` *Plotting regression coefficients with confidence limits*

Description

Plotting regression coefficients with confidence limits

Usage

```
## S3 method for class 'regressionTable'
plot(x, xlim, xlab, style = 1, ...)
```

Arguments

`x`	regression table obtained with regressionTable
`xlim`	Limits for x-axis
`xlab`	Label for x-axis
`style`	Determines how to arrange variable names and their corresponding units
`...`	passed to plotConfidence

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

regressionTable

Examples

```
## linear regression
data(Diabetes)
f <- glm(bp.1s~AgeGroups+chol+gender+location,data=Diabetes)
rtf <- regressionTable(f,factor.reference = "inline")
plot(rtf,cex=1.3)
```

```
## logistic regression
data(Diabetes)
f <- glm(I(BMI>25)~bp.1s+AgeGroups+chol+gender+location,data=Diabetes,family=binomial)
rtf <- regressionTable(f,factor.reference = "inline")
plot(rtf,cex=1.3)

## Poisson regression
data(trace)
fit <- glm(dead ~ smoking+ sex+ age+Time+offset(log(ObsTime)), family = poisson,data=trace)
rtab <- regressionTable(fit,factor.reference = "inline")
plot(rtab,xlim=c(0.85,1.15),cex=1.8,xaxis.cex=1.5)

## Cox regression
library(survival)
data(pbc)
coxfit <- coxph(Surv(time,status!=0)~age+log(bili)+log(albumin)+factor(edema)+sex,data=pbc)
pubcox <- publish(coxfit)
plot(pubcox,cex=1.5,xratio=c(0.4,0.2))
```

`plotConfidence` *Plot confidence intervals*

Description

Function to plot confidence intervals with their values and additional labels. One anticipated use of this function involves first the generation of a regression object, then arrangement of a result table with "regressionTable", further arrangment of table with with e.g. "fixRegressionTable" and various user defined changes - and then finally table along with forest plot using the current function.

Usage

```
plotConfidence(x, y.at, lower, upper, pch = 16, cex = 1, lwd = 1,
  col = 4, xlim, xlab, labels, title.labels, values, title.values,
  section.pos, section.sep, section.title = NULL, section.title.x,
  section.title.offset, order, leftmargin = 0.025, rightmargin = 0.025,
  stripes, factor.reference.pos, factor.reference.label = "Reference",
  factor.reference.pch = 16, refline = 1, title.line = TRUE, xratio,
  y.offset = 0, y.title.offset, digits = 2, format,
  extremearrows.length = 0.05, extremearrows.angle = 30, add = FALSE,
  layout = TRUE, xaxis = TRUE, ...)
```

Arguments

`x`	Either a vector containing the point estimates or a list whose first element contains the point estimates. Further list elements can contain the confidence intervals and labels. In this case the list needs to have names 'lower' and 'upper' to indicate the values of the lower and the upper limits of the confidence intervals, respectively, and may have an element 'labels' which is a vector or matrix or list with labels.
`y.at`	Optional vector of y-position for the confidence intervals and corresponding values and labels.

`lower` Lower confidence limits. Used if object `x` is a vector and if `x` is a list `lower` overwrites element `x$lower`.

`upper` Upper confidence limits. Used if object `x` is a vector and if `x` is a list `upper` overwrites element `x$upper`.

`pch` Symbol for points.

`cex` Defaults size of all figures and plotting symbol. Single elements are controlled separately. See ....

`lwd` Default width of all lines Single elements are controlled separately. See ....

`col` Default colour of confidence intervals.

`xlim` Plotting limits for the confidence intervals. See also `xratio` on how to control the layout.

`xlab` Label for the x-axis.

`labels` Vector or matrix or list with `labels`. Used if object `x` is a vector and if `x` is a list it overwrites element `x$labels`. To avoid drawing of labels, set `labels=FALSE`.

`title.labels` Main title for the column which shows the `labels`. If `labels` is a matrix or list `title.labels` should be a vector with as many elements as labels has columns or elements.

`values` Either logical or vector, matrix or list with values. If `values=TRUE` values are constructed according to `format` from `lower` and `upper` overwrites constructed values. If `values=FALSE` do not draw values.

`title.values` Main title for the column `values`. If `values` is a matrix or list `title.labels` should be a vector with as many elements as values has columns or elements.

`section.pos` Vector with y-axis posititions for section.titles.

`section.sep` Amount of space between paragraphs (applies only if `labels` is a named list)

`section.title` Intermediate section headings.

`section.title.x`
x-position for section.titles

`section.title.offset`
Y-offset for section.titles

`order` Order of the three columns: labels, confidence limits, values. See examples.

`leftmargin` Percentage of plotting region used for leftmargin. Default is 0.025. See also Details.

`rightmargin` Percentage of plotting region used for rightmargin. Default is 0.025. See also Details.

`stripes` Vector of up to three Logicals. If `TRUE` draw stripes into the background. The first applies to the labels, the second to the graphical presentation of the confidence intervals and the third to the values. Thus, stripes

`factor.reference.pos`
Position at which factors attain reference values.

`factor.reference.label`
Label to use at `factor.reference.pos` instead of values.

`factor.reference.pch`
Plotting symbol to use at `factor.reference.pos`

`refline` Position of a vertical line to indicate the null hypothesis. Default is 1 which would work for odds ratios and hazard ratios.

title.line	Position of a horizontal line to separate the title line from the plot
xratio	One or two values between 0 and 1 which determine how to split the plot window in horizontal x-direction. If there are two columns (labels, CI) or (CI, values) only one value is used and the default is 0.618 (goldener schnitt) which gives the graphical presentation of the confidence intervals 38.2 graph. The remaining 61.8 If there are three columns (labels, CI, values), xratio has two values which default to fractions of 0.7 according to the relative widths of labels and values, thus by default only 0.3 are used for the graphical presentation of the confidence intervals. The remaining 30 confidence intervals. See examles.
y.offset	Either a single value or a vector determining the vertical offset of all rows. If it is a single value all rows are shifted up (or down if negative) by this value. This can be used to add a second set of confidence intervals to an existing graph or to achieve a visual grouping of rows that belong together. See examples.
y.title.offset	Numeric value by which to vertically shift the titles of the labels and values.
digits	Number of digits, passed to pubformat and formatCI.
format	Format for constructing values of confidence intervals. Defaults to '(u;l)' if there are negative lower or upper values and to '(u-l)' otherwise.
extremearrows.length	Length of the arrows in case of confidence intervals that stretch beyond xlim.
extremearrows.angle	Angle of the arrows in case of confidence intervals that stretch beyond xlim.
add	Logical. If TRUE do not draw labels or values and add confidence intervals to existing plot.
layout	Logical. If FALSE do not call layout. This is useful when several plotConfidence results should be combined in one graph and hence layout is called externally.
xaxis	Logical. If FALSE do not draw x-axis.
...	Used to control arguments of the following subroutines: plot: Applies to plotting frame of the graphical presentation of confidence intervals. Use arguments of plot, e.g., plot.main="Odds ratio". points, arrows: Use arguments of points and arrows, respectively. E.g., points.pch=8 and arrows.lwd=2. refline: Use arguments of segments, e.g., refline.lwd=2. See segments. labels, values, title.labels, title.values: Use arguments of text, e.g., labels.col="red" or title.values.cex=1.8. xaxis: Use arguments of axis, e.g., xaxis.at=c(-0.3,0,0.3) xlab: Use arguments of mtext, e.g., xlab.line=2. stripes: Use arguments of stripes. See examples. See examples for usage.

Details

Function to plot means and other point estimates with confidence intervals, their values and additional labels . Horizonal margins as determined by par()$mar are ignored. Instead layout is used to divide the plotting region horizontally into two or three parts plus leftmargin and rightmargin.

When values is FALSE there are only two parts. The default order is labels on the left confidence intervals on the right. When no labels are given or labels is FALSE there are only two parts. The default order is confidence intervals on the left values on the right.

The default order of three parts from left to right is labels, confidence intervals, values. The order can be changed as shown by the examples below. The relative widths of the two or three parts need to be adapted to the actual size of the text of the labels. This depends on the plotting device and the size of the font and figures and thus has to be adjusted manually.

Oma can be used to further control horizontal margins, e.g., par(oma=c(0,4,0,4)).

If confidence limits extend beyond the range determined by xlim, then arrows are drawn at the x-lim borders to indicate that the confidence limits continue.

Value

List of dimensions and coordinates

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
library(Publish)
data(CiTable)

## A first draft version of the plot is obtained as follows
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper","p")],
          labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")])

## if argument labels is a named list the table is subdivided:
labellist <- split(CiTable[,c("Dose","Mean","SD","n")],CiTable[,"Drug"])
labellist
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")], labels=labellist)


## The graph consist of at most three columns:
##
## column 1: labels
## column 2: printed values of the confidence intervals
## column 3: graphical presentation of the confidence intervals
##
## NOTE: column 3 appears always, the user decides if also
##       column 1, 2 should appear
##
## The columns are arranged with the function layout
## and the default order is 1,3,2 such that the graphical
## display of the confidence intervals appears in the middle
##
## the order of appearance of the three columns can be changed as follows
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               order=c(1,3,2))
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               order=c(2,3,1))
## if there are only two columns the order is 1, 2
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               values=FALSE,
               order=c(2,1))
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               values=FALSE,
               order=c(1,2))
```

```
## The relative size of the columns needs to be controlled manually
## by using the argument xratio. If there are only two columns
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xratio=c(0.4,0.15))

## The amount of space on the left and right margin can be controlled
## as follows:
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xratio=c(0.4,0.15),
               leftmargin=0.1,rightmargin=0.00)

## The actual size of the current graphics device determines
## the size of the figures and the space between them.
## The sizes and line widths are increased as follows:
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               xlab="Hazard ratio",
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               points.cex=3,
               cex=2,
               lwd=3,
               xaxis.lwd=1.3,
               xaxis.cex=1.3)
## Note that 'cex' of axis ticks is controlled via 'par' but
## cex of the label via argument 'cex' of 'mtext'.
## The sizes and line widths are decreased as follows:
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               cex=0.8,
               lwd=0.8,
               xaxis.lwd=0.8,
               xaxis.cex=0.8)

## Another good news is that all figures can be controlled separately

## The size of the graphic device can be controlled in the usual way, e.g.:
## Not run:
    pdf("~/tmp/testCI.pdf",width=8,height=8)
    plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
                   labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")])
    dev.off()

## End(Not run)

## More control of the x-axis and confidence intervals that
## stretch outside the x-range end in an arrow.
## the argument xlab.line adjusts the distance of the x-axis
## label from the graph
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               xlab="Hazard ratio",
               xlab.line=1.8,
               xaxis.at=c(0.8,1,1.3),
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xlim=c(0.8,1.3))
```

```
## log-scale
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               xlab="Hazard ratio",
               xlab.line=1.8,
               xaxis.at=c(0.8,1,1.3),
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xlim=c(0.8,1.3),plot.log="x")
## More pronounced arrows
## Coloured xlab expression
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               xlab=expression(HR[1](s)),
               xlab.line=1.8,
               xlab.col="darkred",
               extremearrows.angle=50,
               extremearrows.length=0.1,
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xlim=c(0.8,1.3))

## Controlling the labels and their titles
## and the values and their titles
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xlab="Hazard ratio",
               title.values=expression(bold(HR (CI[95]))),
               title.labels=c("Drug/Time","Dose","Mean","St.dev.","N"),
               factor.reference.pos=c(1,10,19),
               factor.reference.pch=16,
               cex=1.3,
               xaxis.at=c(0.75,1,1.25,1.5,2))

## For factor reference groups, one may want to replace the
## confidence intervals by the word Reference, as in the previous example.
## To change the word 'Reference' we use the argument factor.reference.label:
## To change the plot symbol for the reference lines factor.reference.pch
## To remove the plot symbol in the reference lines use 'NA' as follows:
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xlab="Hazard ratio",
               factor.reference.label="Ref",
               title.values=expression(bold(HR (CI[95]))),
               title.labels=c("Drug/Time","Dose","Mean","St.dev.","N"),
               factor.reference.pos=c(1,10,19),
               factor.reference.pch=NA,
               cex=1.3,
               xaxis.at=c(0.75,1,1.25,1.5,2))


## changing the style of the graphical confidence intervals
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xlab="Hazard ratio",
               factor.reference.pos=c(1,10,19),
               points.pch=15,
               points.col=rainbow(27),
               points.cex=2,
               arrows.col="darkblue",
```

```
               cex=1.3,
               order=c(1,3,2),
               xaxis.at=c(0.75,1,1.25,1.5))

## the values column of the graph can have multiple columns as well
## to illustrate this we create the confidence intervals
## before calling the function and then cbind them
## to the pvalues
HR <- pubformat(CiTable[,6])
CI95 <- formatCI(lower=CiTable[,7],upper=CiTable[,8],format="(l-u)")
pval <- format.pval(CiTable[,9],digits=3,eps=10^{-3})
pval[pval=="NA"] <- ""
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               values=list("HR"=HR,"CI-95"=CI95,"P-value"=pval),
               cex=1.2,
               xratio=c(0.5,0.3))

## Finally, vertical columns can be delimited with background color
## NOTE: this may slow things down and potentially create
##       large figures (many bytes)
col1 <- rep(c(prodlim::dimColor("green",density=22),
              prodlim::dimColor("green")),length.out=9)
col2 <- rep(c(prodlim::dimColor("orange",density=22),
              prodlim::dimColor("orange")),length.out=9)
col3 <- rep(c(prodlim::dimColor("blue",density=22),
              prodlim::dimColor("blue")),length.out=9)
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               stripes=c(1,0,1),
               stripes.col=c(col1,col2,col3))
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               stripes=c(1,1,1),
               stripes.col=c(col1,col2,col3))

threegreens <- c(prodlim::dimColor("green",density=55),
                 prodlim::dimColor("green",density=33),
                 prodlim::dimColor("green",density=22))
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               values=FALSE,
               xlim=c(0.75,1.5),
               stripes=c(1,1,1),
               xratio=c(0.5,0.15),
               stripes.horizontal=c(0,9,18,27)+0.5,
               stripes.col=threegreens)
```

`print.ci` *Print confidence intervals*

Description

Print confidence intervals

Usage

```
## S3 method for class 'ci'
print(x, se = FALSE, print = TRUE, ...)
```

Arguments

x	Object containing point estimates and the corresponding confidence intervals
se	If TRUE add the standard error.
print	Logical: if FALSE do not actually print confidence intervals but just return them invisibly.
...	passed to summary.ci

Details

This format of the confidence intervals is user-manipulable.

Value

A string: the formatted confidence intervals

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

ci plot.ci formatCI summary.ci

Examples

```
library(lava)
m <- lvm(Y~X)
m <- categorical(m,Y~X,K=4)
set.seed(4)
d <- sim(m,24)
ci.mean(Y~X,data=d)
x <- ci.mean(Y~X,data=d)
print(x,format="(l,u)")
```

print.table2x2 *print results of 2x2 contingency table analysis*

Description

print results of 2x2 contingency table analysis

Usage

```
## S3 method for class 'table2x2'
print(x, digits = 1, ...)
```

Arguments

x	object obtained with table2x2
digits	rounding digits
...	not used

Value

invisible x

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

table2x2

Examples

```
table2x2(table("marker"=rbinom(100,1,0.4),"response"=rbinom(100,1,0.1)))
table2x2(matrix(c(71,18,38,8),ncol=2),stats="table")
table2x2(matrix(c(71,18,38,8),ncol=2),stats=c("rr","fisher"))
```

print.univariateTable *Printing univariate tables*

Description

Print function for univariate tables

Usage

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

Arguments

x	An object obtained with univariateTable
...	Passed to summary.univariateTable

Details

This function is simply calling summary.univariateTable

Value

The result of summary.univariateTable(x)

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

univariateTable

---

pubformat *Format numbers for publication*

Description

Format numbers according to a specified handler function. Currently supported are sprintf, format and prettyNum.

Usage

```
pubformat(x, digits = 2, nsmall = digits, handler = "sprintf", ...)
```

Arguments

x	numeric vector
digits	number of digits
nsmall	see handler
handler	String specifiying the name of the function which should perform the formatting. See sprintf, format and prettyNum.
...	Passed to handler function if applicable, i.e., not to sprintf.

Value

Formatted number

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

sprintf, format, prettyNum

Examples

```
pubformat(c(0.000143,12.8,1))
pubformat(c(0.000143,12.8,1),handler="format")
pubformat(c(0.000143,12.8,1),handler="format",trim=TRUE)
pubformat(c(0.000143,12.8,1),handler="prettyNum")
```

---

publish *Publishing tables and figures*

Description

Publish provides summary functions for data and results of statistical analysis in ready-for-publication design

Usage

```
publish(object, ...)
```

Arguments

object	object to be published
...	Passed to method.

Details

Some warnings are currently suppressed.

Value

Tables and figures

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

publish.CauseSpecificCox publish.ci publish.coxph publish.glm publish.riskRegression publish.survdiff

publish.CauseSpecificCox *Tabulizing cause-specific hazard ratio from all causes with confidence limits and Wald test p-values.*

Description

Publish cause-specific Cox models

Usage

```
## S3 method for class 'CauseSpecificCox'
publish(object, cause, confint.method, pvalue.method,
  factor.reference = "extraline", units = NULL, print = TRUE, ...)
```

Arguments

object	Cause-specific hazard model obtained with CSC.
cause	Show a table for this cause. If omitted, list all causes.
confint.method	See regressionTable
pvalue.method	See regressionTable
factor.reference	See regressionTable
units	See regressionTable
print	If TRUE print the table(s).
...	passed on to control formatting of parameters, confidence intervals and p-values. See summary.regressionTable.

Details

The cause-specific hazard ratio's are combined into one table.

Value

Table with cause-specific hazard ratios, confidence limits and p-values.

Author(s)

Thomas Alexander Gerds <tab@biostat.ku.dk>

Examples

```
library(riskRegression)
library(prodlim)
library(pec)
library(survival)
data(Melanoma,package="riskRegression")
fit1 <- CSC(list(Hist(time,status)~sex,Hist(time,status)~invasion+epicel+age),
            data=Melanoma)
publish(fit1)
publish(fit1,pvalue.stars=TRUE)
publish(fit1,factor.reference="inline",units=list("age"="years"))
```

publish.ci *Publish tables with confidence intervals*

Description

Publish tables with confidence intervals

Usage

```
## S3 method for class 'ci'
publish(object, format = "[u;l]", se = FALSE, ...)
```

Arguments

object	Object of class ci containing point estimates and the corresponding confidence intervals
format	A string which indicates the format used for confidence intervals. The string is passed to formatCI with two arguments: the lower and the upper limit. For example '(l;u)' yields confidence intervals with round parenthesis in which the upper and the lower limits are separated by semicolon.
se	If TRUE add standard error.
...	passed to publish

Details

This function calls summary.ci with print=FALSE and then publish

Value

table with confidence intervals

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

summary.ci

Examples

```
data(Diabetes)
publish(ci.mean(chol~location+gender,data=Diabetes),org=TRUE)
```

publish.coxph *Tabulize hazard ratios with confidence intervals and p-values.*

Description

Tabulize the part of the result of a Cox regression analysis which is commonly shown in publications.

Usage

```
## S3 method for class 'coxph'
publish(object, confint.method, pvalue.method, print = TRUE,
  factor.reference = "extraline", units = NULL, probindex = FALSE, ...)
```

Arguments

object	A coxph object.
confint.method	See regressionTable
pvalue.method	See regressionTable
print	If FALSE do not print results.
factor.reference	See regressionTable
units	See regressionTable
probindex	Logical. If TRUE show coefficients on probabilistic index scale instead of hazard ratio scale.
...	passed to summary.regressionTable and also to labelUnits.

Details

Transforms the log hazard ratios to hazard ratios and returns them with confidence limits and p-values. If explanatory variables are log transformed or log2 transformed, a scaling factor is multiplied to both the log-hazard ratio and its standard-error.

Value

Table with hazard ratios, confidence intervals and p-values.

Author(s)

Thomas Alexander Gerds

Examples

```
library(survival)
data(pbc)
pbc$edema <- factor(pbc$edema,
             levels=c("0","0.5","1"), labels=c("0","0.5","1"))
fit = coxph(Surv(time,status!=0)~age+sex+edema+log(bili)+log(albumin),
            data=na.omit(pbc))
publish(fit)
```

```
## forest plot
plot(publish(fit),cex=1.3)

publish(fit,ci.digits=2,pvalue.eps=0.01,pvalue.digits=2,pvalue.stars=TRUE)
publish(fit,ci.digits=2,ci.handler="prettyNum",pvalue.eps=0.01,
        pvalue.digits=2,pvalue.stars=TRUE)
publish(fit, ci.digits=2, ci.handler="sprintf", pvalue.eps=0.01,
        pvalue.digits=2,pvalue.stars=TRUE, ci.trim=FALSE)

fit2 = coxph(Surv(time,status!=0)~age+sex+edema+log(bili,base=2)+log(albumin)+log(protime),
    data=na.omit(pbc))
publish(fit2)

# with cluster variable
fit3 = coxph(Surv(time,status!=0)~age+cluster(sex)+edema+log(bili,base=2)
                                    +log(albumin)+log(protime),
    data=na.omit(pbc))
publish(fit3)

# with strata and cluster variable
fit4 = coxph(Surv(time,status!=0)~age+cluster(sex)+strata(edema)+log(bili,base=2)
                 +log(albumin)+log(protime),
    data=pbc)
publish(fit4)
```

---

publish.glm *Tabulize regression coefficients with confidence intervals and p-values.*

Description

Tabulate the results of a generalized linear regression analysis.

Usage

```
## S3 method for class 'glm'
publish(object, confint.method, pvalue.method, digits = c(2, 4),
  print = TRUE, factor.reference = "extraline",
  intercept = ifelse((is.null(object$family) || object$family$family ==
  "gaussian"), 1L, 0L), units = NULL, ...)
```

Arguments

object	A glm object.
confint.method	See regressionTable.
pvalue.method	See regressionTable.
digits	A vector of two integer values. These determine how to round numbers (first value) and p-values (second value). E.g., c(1,3) would mean 1 digit for all numbers and 3 digits for p-values. The actual rounding is done by summary.regressionTable.
print	If FALSE do not print results.
factor.reference	Style for showing results for categorical. See regressionTable.

intercept	See regressionTable.
units	See regressionTable.
...	passed to summary.regressionTable and also to labelUnits.
reference	Style for showing results for categorical variables. If "extraline" show an additional line for the reference category.

Details

The table shows changes in mean for linear regression and odds ratios for logistic regression (family = binomial).

Value

Table with regression coefficients, confidence intervals and p-values.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

```
data(Diabetes)
## Linear regression
f = glm(bp.2s~frame+gender+age,data=Diabetes)
publish(f)
publish(f,factor.reference="inline")
publish(f,pvalue.stars=TRUE)
publish(f,ci.format="(l,u)")

### interaction
fit = glm(bp.2s~frame+gender*age,data=Diabetes)
summary(fit)
publish(fit)

Fit = glm(bp.2s~frame*gender+age,data=Diabetes)
publish(Fit)

## Logistic regression
Diabetes$hyper1 <- factor(1*(Diabetes$bp.1s>140))
lrfit <- glm(hyper1~frame+gender+age,data=Diabetes,family=binomial)
publish(lrfit)

### interaction
lrfit1 <- glm(hyper1~frame+gender*age,data=Diabetes,family=binomial)
publish(lrfit1)

lrfit2 <- glm(hyper1~frame*gender+age,data=Diabetes,family=binomial)
publish(lrfit2)

## Poisson regression
data(trace)
trace <- Units(trace,list("age"="years"))
fit <- glm(dead ~ smoking+sex+age+Time+offset(log(ObsTime)), family="poisson",data=trace)
rtf <- regressionTable(fit,factor.reference = "inline")
summary(rtf)
```

```
publish(fit)

## gls regression
library(nlme)
library(lava)
m <- lvm(Y ~ X1 + gender + group + Interaction)
distribution(m, ~gender) <- binomial.lvm()
distribution(m, ~group) <- binomial.lvm(size = 2)
constrain(m, Interaction ~ gender + group) <- function(x){x[,1]*x[,2]}
d <- sim(m, 1e2)
d$gender <- factor(d$gender, labels = letters[1:2])
d$group <- factor(d$group)

e.gls <- gls(Y ~ X1 + gender*group, data = d,
             weights = varIdent(form = ~1|group))
publish(e.gls)

## lme
library(nlme)
fm1 <- lme(distance ~ age*Sex,
            random = ~1|Subject,
            data = Orthodont)
res <- publish(fm1)
```

`publish.htest` *Pretty printing of test results.*

Description

Pretty printing of test results.

Usage

```
## S3 method for class 'htest'
publish(object, title, ...)
```

Arguments

`object`	Result of `t.test` or `wilcox.test`
`title`	Decoration also used to name output
`...`	Used to transport arguments `ci.arg` and `pvalue.arg` to subroutines `format.pval` and `formatCI`. See also `prodlim::SmartControl`.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
data(Diabetes)
publish(t.test(bp.2s~gender,data=Diabetes))
publish(wilcox.test(bp.2s~gender,data=Diabetes))
publish(with(Diabetes,t.test(bp.2s,bp.1s,paired=TRUE)))
publish(with(Diabetes,wilcox.test(bp.2s,bp.1s,paired=TRUE)))
```

`publish.matrix` *Publishing a matrix in raw, org, latex, or muse format*

Description

This is the heart of the Publish package

Usage

```
## S3 method for class 'matrix'
publish(object, title, colnames = TRUE, rownames = TRUE,
  col1name = "", digits = 4, sep = " ", endhead, endrow, style,
  inter.lines, latex = FALSE, wiki = FALSE, org = FALSE,
  markdown = FALSE, tabular = TRUE, latex.table.format = NA,
  latex.hline = 1, latex.nodollar = FALSE, ...)
```

Arguments

`object`	Matrix to be published
`title`	Title for table, only in wiki and muse format
`colnames`	If `TRUE` show column names
`rownames`	If `TRUE` show row names
`col1name`	Name for first column
`digits`	Numbers are rounded according to digits
`sep`	Field separator when style is `"none"`
`endhead`	String to be pasted at the end of the first row (header)
`endrow`	String to be pasted at the end of each row
`style`	Table style for export to `"latex"`, `"org"`, `"markdown"`, `"wiki"`, `"none"`. Overwritten by argments below.
`inter.lines`	A named list which contains strings to be placed between the rows of the table. An element with name `"0"` is used to place a line before the first column, elements with name `"r"` are placed between line r and r+1.
`latex`	If `TRUE` use latex table format
`wiki`	If `TRUE` use mediawiki table format
`org`	If `TRUE` use emacs orgmode table format
`markdown`	If `TRUE` use markdown table format
`tabular`	For style `latex` only: if `TRUE` enclose the table in begin/end tabular environement.
`latex.table.format`	For style `latex` only: format of the tabular environement.
`latex.hline`	For style `latex` only: if `TRUE` add hline statements add the end of each line.
`latex.nodollar`	For style `latex` only: if `TRUE` do not enclose numbers in dollars.
`...`	Used to transport arguments. Currently supports `wiki.class`.

Examples

```
x <- matrix(1:12,ncol=3)
publish(x)

# rounding the numeric part of data mixtures
y <- cbind(matrix(letters[1:12],ncol=3),x,matrix(rnorm(12),ncol=3))
publish(y,digits=1)

publish(x,inter.lines=list("1"="text between line 1 and line 2",
                           "3"="text between line 3 and line 4"))
```

`publish.MIresult` *Present logistic regression and Cox regression obtained with mitools::MIcombine based on smcfcs::smcfcs multiple imputation analysis*

Description

Regression tables after multiple imputations

Usage

```
## S3 method for class 'MIresult'
publish(object, confint.method, pvalue.method,
  digits = c(2, 4), print = TRUE, factor.reference = "extraline",
  intercept, units = NULL, fit, data, ...)
```

Arguments

`object`	Object obtained with mitools::MIcombine based on smcfcs::smcfcs multiple imputation analysis
`confint.method`	No options here. Only Wald type confidence intervals.
`pvalue.method`	No options here. Only Wald type tests.
`digits`	Rounding digits for all numbers but the p-values.
`print`	If `FALSE` suppress printing of the results
`factor.reference`	Style for showing results for categorical. See `regressionTable`.
`intercept`	See `regressionTable`.
`units`	See `regressionTable`.
`fit`	One fitted model using the same formula as `object`. This can be the fit to the complete case data or the fit to one of the completed data. It is used to get xlevels, formula and terms. For usage see examples. is used to fit
`data`	Original data set which includes the missing values
`...`	passed to summary.regressionTable, labelUnits and publish.default.

Details

Show results of smcfcs based multiple imputations of missing covariates in publishable format

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
## Not run:
## continuous outcome: linear regression
# lava some data with missing values
library(riskRegression)
set.seed(7)
d=sampleData(78)
## generate missing values
d[X1==1,X6:=NA]
d[X2==1,X3:=NA]
d=d[,.(X8,X4,X3,X6,X7)]
sapply(d,function(x)sum(is.na(x)))

# multiple imputation (should set m to a large value)

library(smcfcs)
library(mitools)
set.seed(17)
f= smcfcs(d,smtype="lm",
           smformula=X8~X4+X3+X6+X7,
           method=c("","","logreg","norm",""),m=3)
ccfit=lm(X8~X4+X3+X6+X7,data=d)
mifit=MIcombine(with(imputationList(f$impDatasets),
                lm(X8~X4+X3+X6+X7)))
publish(mifit,fit=ccfit,data=d)
publish(ccfit)

## binary outcome
# lava some data with missing values
library(riskRegression)
set.seed(7)
db=sampleData(78,outcome="binary")
## generate missing values
db[X1==1,X6:=NA]
db[X2==1,X3:=NA]
db=db[,.(Y,X4,X3,X6,X7)]
sapply(db,function(x)sum(is.na(x)))

# multiple imputation (should set m to a large value)
library(smcfcs)
library(mitools)
set.seed(17)
fb= smcfcs(db,smtype="logistic",
           smformula=Y~X4+X3+X6+X7,
           method=c("","","logreg","norm",""),m=2)
ccfit=glm(Y~X4+X3+X6+X7,family="binomial",data=db)
mifit=MIcombine(with(imputationList(fb$impDatasets),
                glm(Y~X4+X3+X6+X7,family="binomial")))
publish(mifit,fit=ccfit)
publish(ccfit)
```

```
## survival: Cox regression
library(smcfcs)
library(mitools)
library(survival)
# lava some data with missing values
library(riskRegression)
set.seed(7)
ds=sampleData(78,outcome="survival")
## generate missing values
ds[X5==1,X6:=NA]
ds[X2==1,X3:=NA]
ds=ds[,.(time,event,X4,X3,X6,X7)]
sapply(ds,function(x)sum(is.na(x)))

set.seed(17)
fs= smcfcs(ds,smtype="coxph",
           smformula="Surv(time,event)~X4+X3+X6+X7",
           method=c("","","","logreg","norm",""),m=2)
ccfit=coxph(Surv(time,event)~X4+X3+X6+X7,data=ds)
mifit=MIcombine(with(imputationList(fs$impDatasets),
                coxph(Surv(time,event)~X4+X3+X6+X7)))
publish(mifit,fit=ccfit,data=ds)
publish(ccfit)

## competing risks: Cause-specific Cox regression
library(survival)
library(smcfcs)
library(mitools)
# lava some data with missing values
library(riskRegression)
set.seed(7)
dcr=sampleData(78,outcome="competing.risks")
## generate missing values
dcr[X5==1,X6:=NA]
dcr[X2==1,X3:=NA]
dcr=dcr[,.(time,event,X4,X3,X6,X7)]
sapply(dcr,function(x)sum(is.na(x)))

set.seed(17)
fcr= smcfcs(dcr,smtype="compet",
           smformula=c("Surv(time,event==1)~X4+X3+X6+X7",
                       "Surv(time,event==2)~X4+X3+X6+X7"),
           method=c("","","","logreg","norm",""),m=2)
## cause 2
ccfit2=coxph(Surv(time,event==2)~X4+X3+X6+X7,data=dcr)
mifit2=MIcombine(with(imputationList(fcr$impDatasets),
                coxph(Surv(time,event==2)~X4+X3+X6+X7)))
publish(mifit2,fit=ccfit2,data=dcr)
publish(ccfit2)

## End(Not run)
```

---

publish.riskRegression

Publishing results of riskRegression

Description

Preparing a publishable table from riskRegression results

Usage

```
## S3 method for class 'riskRegression'
publish(object, digits = c(2, 4), print = TRUE,
  ...)
```

Arguments

object	object of class riskRegression as obtained with functions ARR and LRR.
digits	Number of digits for regression coefficients
print	If FALSE do not print the results
...	passed to publish.matrix

Value

Table with regression coefficients, confidence intervals and p-values

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

ARR LRR

Examples

```
library(prodlim)
library(riskRegression)
library(lava)
library(survival)
set.seed(20)
d <- SimCompRisk(20)
f <- ARR(Hist(time,event)~X1+X2,data=d,cause=1)
publish(f)
publish(f,digits=c(1,3))
```

---

publish.Score *Publish predictive accuracy results*

Description

Write output of riskRegression::Score in tables

Usage

```
## S3 method for class 'Score'
publish(object, metrics, score = TRUE, contrasts = TRUE,
  level = 3, ...)
```

Arguments

object	Object obtained with riskRegression::Score
metrics	Which metrics to put into tables. Defaults to object$metrics.
score	Logical. If TRUE print the score elements, i.e., metric applied to the risk prediction models.
contrasts	Logical. If TRUE print the contrast elements (if any). These compare risk prediction models according to metrics.
level	Level of subsection headers, i.e., ** for level 2 and *** for level 3 (useful for emacs org-users). Default is plain subsection headers no stars. A negative value will suppress subjection headers.
...	Passed to publish

Details

Collect prediction accuracy results in tables

Value

Results of Score in tabular form

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
library(riskRegression)
library(survival)
learn = sampleData(100)
val= sampleData(100)
f1=CSC(Hist(time,event)~X1+X8,data=learn)
f2=CSC(Hist(time,event)~X1+X5+X6+X8,learn)
xs=Score(list(f1,f2),data=val,formula=Hist(time,event)~1)
publish(xs)
```

---

publish.summary.aov *Format summary table of aov results*

Description

Format summary table of aov results

Usage

```
## S3 method for class 'summary.aov'
publish(object, print = TRUE, handler = "sprintf",
  digits = c(2, 4), nsmall = digits, ...)
```

Arguments

object	glm object
print	Logical. Decide about whether or not to print the results.
handler	see pubformat
digits	see pubformat
nsmall	see pubformat
...	used to transport further arguments

Examples

```
data(Diabetes)
f <- glm(bp.1s~age+chol+gender+location,data=Diabetes)
publish(summary(aov(f)),digits=c(1,2))
```

---

publish.survdiff *Alternative summary of survdiff results*

Description

Alternative summary of survdiff results

Usage

```
## S3 method for class 'survdiff'
publish(object, digits = c(2, 4), print = TRUE, ...)
```

Arguments

object	Object obtained with survival::survdiff.
digits	Vector with digits for rounding numbers: the second for pvalues, the first for all other numbers.
print	If FALSE do not print results.
...	Not (yet) used.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
library(survival)
data(pbc)
sd <- survdiff(Surv(time,status!=0)~sex,data=pbc)
publish(sd)
publish(sd,digits=c(3,2))
```

`regressionTable` *Regression table*

Description

Tabulate the results of a regression analysis.

Usage

```
regressionTable(object, param.method = "coef", confint.method = c("default",
  "profile", "robust", "simultaneous"), pvalue.method = c("default", "robust",
  "simultaneous"), factor.reference = "extraline", intercept = 0L,
  units = NULL, noterms = NULL, probindex = 0L, ...)
```

Arguments

`object`	Fitted regression model obtained with `lm`, `glm` or `coxph`.
`param.method`	Method to obtain model coefficients.
`confint.method`	Method to obtain confidence intervals. Default is `'default'` which leads to Wald type intervals using the model based estimate of standard error. `'profile'` yields profile likelihood confidence intervals, available from library MASS for `lm` and `glm` objects. `'robust'` uses the sandwich form standard error to construct Wald type intervals (see `lava::estimate.default`). `simultaneous` calls `multcomp::glht` to obtain simultaneous confidence intervals.
`pvalue.method`	Method to obtain p-values. If `'default'` show raw p-values. If `'robust'` use p-value corresponding to robust standard error as provided by `lava::estimate.default`. If `'simultaneous'` call `multcomp::glht` to obtain p-values.
`factor.reference`	Style for showing results for categorical variables. If `'extraline'` show an additional line for the reference category. If `'inline'` display as level vs. reference.
`intercept`	Logical. If `FALSE` suppress intercept.
`units`	List of units for continuous variables. See examples.
`noterms`	Position of terms that should be ignored. E.g., for a Cox model with a cluster(id) term, there will be no hazard ratio for variable id.
`probindex`	Logical. If `TRUE` show coefficients on probabilistic index scale instead of hazard ratio scale.
`...`	Not yet used

Details

The basic use of this function is to generate a near publication worthy table from a regression object. As with summary(object) reference levels of factor variables are not included. Expansion of the table with such values can be performed using the "fixRegressionTable" function. Forest plot can be added to the output with "plotRegressionTable".

regressionTable produces an object (list) with the parameters deriveds. The summary function creates a data frame which can be used as a (near) publication ready table.

The table shows changes in mean for linear regression, odds ratios for logistic regression (family = binomial) and hazard ratios for Cox regression.

Value

List of regression blocks

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
# linear regression
data(Diabetes)
f1 <- glm(bp.1s~age+gender+frame+chol,data=Diabetes)
summary(regressionTable(f1))
summary(regressionTable(f1,units=list("chol"="mmol/L","age"="years")))
## with interaction
f2 <- glm(bp.1s~age*gender+frame+chol,data=Diabetes)
summary(regressionTable(f2))
#Add reference values
summary(regressionTable(f2))
f3 <- glm(bp.1s~age+gender*frame+chol,data=Diabetes)
publish(f3)
regressionTable(f3)

# logistic regression
Diabetes$hyp1 <- factor(1*(Diabetes$bp.1s>140))
l1 <- glm(hyp1~age+gender+frame+chol,data=Diabetes,family="binomial")
regressionTable(l1)
publish(l1)
plot(regressionTable(l1))

## with interaction
l2 <- glm(hyp1~age+gender+frame*chol,data=Diabetes,family="binomial")
regressionTable(l2)
l3 <- glm(hyp1~age*gender+frame*chol,data=Diabetes,family="binomial")
regressionTable(l3)

# Cox regression
library(survival)
data(pbc)
pbc$edema <- factor(pbc$edema,levels=c("0","0.5","1"),labels=c("0","0.5","1"))
c1 <- coxph(Surv(time,status!=0)~log(bili)+age+protime+sex+edema,data=pbc)
regressionTable(c1)
# with interaction
c2 <- coxph(Surv(time,status!=0)~log(bili)+age+protime*sex+edema,data=pbc)
regressionTable(c2)
c3 <- coxph(Surv(time,status!=0)~edema*log(bili)+age+protime+sex+edema+edema:sex,data=pbc)
regressionTable(c3)


## gls regression
library(nlme)
library(lava)
m <- lvm(Y ~ X1 + gender + group + Interaction)
distribution(m, ~gender) <- binomial.lvm()
distribution(m, ~group) <- binomial.lvm(size = 2)
constrain(m, Interaction ~ gender + group) <- function(x){x[,1]*x[,2]}
```

```
d <- sim(m, 1e2)
d$gender <- factor(d$gender, labels = letters[1:2])
d$group <- factor(d$group)

e.gls <- gls(Y ~ X1 + gender*group, data = d,
             weights = varIdent(form = ~1|group))
regressionTable(e.gls)
summary(regressionTable(e.gls))
```

SpaceT *A study was made of all 26 astronauts on the first eight space shuttle flights (Bungo et.al., 1985). On a voluntary basis 17 astronauts consumed large quantities of salt and fluid prior to landing as a countermeasure to space deconditioning, while nine did not.*

Description

A study was made of all 26 astronauts on the first eight space shuttle flights (Bungo et.al., 1985). On a voluntary basis 17 astronauts consumed large quantities of salt and fluid prior to landing as a countermeasure to space deconditioning, while nine did not.

Format

A data frame with 52 observations on the following 4 variables:

Status Factor with levels Post (after flight) and Pre (before flight)

HR Supine heart rate(beats per minute)

Treatment Countermeasure salt/fluid (1= yes, 0=no)

ID Person id

References

Altman, Practical statistics for medical research, Page 223, Ex. 9.1. Bungo et.al., 1985

Examples

```
data(SpaceT)
```

---

spaghettiogram *Spaghettiogram*

Description

A spaghettiogram is showing repeated measures (longitudinal data)

Usage

```
spaghettiogram(formula, data, xlim, ylim, xlab = "", ylab = "",
  axes = TRUE, col, lwd, lty, pch, legend = FALSE, add = FALSE,
  background = TRUE, ...)
```

Arguments

formula	A formula which specifies the variables for the spaghettiograms. If Y ~ X + id(Z) then for each value of Z the spaghettiogram is the graph (X,Y) in the subset defined by the value of Z. Data are expected to be in the "long" format. Y is a numeric vector and X is a factor whose levels define the X-axis. Each level of the id-vector corresponds to one line (spaghetti) in the plot.
data	data set in which variables X, Y and Z are defined.
xlim	Limits for x-axis
ylim	Limits for y-axis
xlab	Label for x-axis
ylab	Label for x-axis
axes	Logical indicating if axes should be drawn.
col	Colors for the spaghettiograms
lwd	Widths for the spaghettiograms
lty	Type for the spaghettiograms
pch	Point-type for the spaghettiograms
legend	If TRUE add a legend. Argument A of legend is controlled as legend.A. E.g., when legend.cex=2 legend will be called with argument cex=2.
add	If TRUE add to existing plot device.
background	Control the background color of the graph.
...	used to transport arguments which are passed to the following subroutines: "plot", "lines", "legend", "background", "axis1", "axis2".

Value

List with data of each subject

Examples

```
data(SpaceT)
Spaghettiogram(HR~Status+id(ID),
               data=SpaceT)
```

---

specialFrame *Special frame*

Description

Extract data and design matrix including specials from call

Usage

```
specialFrame(formula, data, unspecials.design = TRUE, specials,
  specials.factor = TRUE, specials.design = FALSE, strip.specials = TRUE,
  strip.arguments = NULL, strip.alias = NULL, strip.unspecials = NULL,
  drop.intercept = TRUE, response = TRUE, na.action = options()$na.action)
```

Arguments

formula	Formula whose left hand side specifies the event history, i.e., either via Surv() or Hist().
data	Data frame in which the formula is interpreted
unspecials.design	Passed as is to model.design.
specials	Character vector of special function names. Usually the body of the special functions is function(x)x but e.g., strata from the survival package does treat the values
specials.factor	Passed as is to model.design.
specials.design	Passed as is to model.design
strip.specials	Passed as specials to strip.terms
strip.arguments	Passed as arguments to strip.terms
strip.alias	Passed as alias.names to strip.terms
strip.unspecials	Passed as unspecials to strip.terms
drop.intercept	Passed as is to model.design
response	If FALSE do not get response data.
na.action	Decide what to do with missing values.

Details

Obtain a list with the data used for event history regression analysis. This function cannot be used directly on the user level but inside a function to prepare data for survival analysis.

Value

A list which contains - the response - the design matrix (see model.design) - one entry for each special (see model.design)

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

model.frame model.design Hist

Examples

```
## Here are some data with an event time and no competing risks
## and two covariates X1 and X2.
## Suppose we want to declare that variable X1 is treated differently
## than variable X2. For example, X1 could be a cluster variable, or
## X1 should have a proportional effect on the outcome.
d <- data.frame(y=1:7,
                X2=c(2.24,3.22,9.59,4.4,3.54,6.81,5.05),
```

```
                X3=c(1,1,1,1,0,0,1),
                X4=c(44.69,37.41,68.54,38.85,35.9,27.02,41.84),
                X1=factor(c("a","b","a","c","c","a","b"),
                    levels=c("c","a","b")))
## define special functions prop and cluster
prop <- function(x)x
cluster <- function(x)x
## We pass a formula and the data
e <- specialFrame(y~prop(X1)+X2+cluster(X3)+X4,
                  data=d,
                  specials=c("prop","cluster"))
## The first element is the response
e$response
## The other elements are the design, i.e., model.matrix for the non-special covariates
e$design
## and a data.frame for the special covariates
e$prop
## The special covariates can be returned as a model.matrix
e2 <- specialFrame(y~prop(X1)+X2+cluster(X3)+X4,
                   data=d,
                   specials=c("prop","cluster"),
                   specials.design=TRUE)
e2$prop
## and the non-special covariates can be returned as a data.frame
e3 <- specialFrame(y~prop(X1)+X2+cluster(X3)+X4,
                   data=d,
                   specials=c("prop","cluster"),
                   specials.design=TRUE,
                   unspecials.design=FALSE)
e3$design
```

`splinePlot.lrm` *Plot predictions of logistic regression*

Description

Plotting the prediction of a logistic regression model with confidence bands against one continuous variable.

Usage

```
splinePlot.lrm(object, xvar, xvalues, xlim = range(xvalues), ylim,
  xlab = xvar, ylab = scale[[1]], col = 1, lwd = 3, confint = TRUE,
  newdata = NULL, scale = c("risk", "odds"), add = FALSE, ...)
```

Arguments

`object`	Logistic regression model fitted with `rms::lrm`
`xvar`	Name of the variable to show on x-axis
`xvalues`	Sequence of `xvar` values
`xlim`	x-axis limits
`ylim`	y-axis limits

xlab	x-axis labels
ylab	y-axis labels
col	color of the line
lwd	line width
confint	Logical. If TRUE show confidence shadows
newdata	How to adjust
scale	Character string that determines the outcome scale (y-axis). Choose between "risk" and "odds".
add	Logical. If TRUE add lines to an existing graph
...	Further arguments passed to plot. Only if add is FALSE.

Details

Function which extracts from a logistic regression model fitted with rms::lrm the predicted risks or odds.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
data(Diabetes)
Diabetes$hypertension=  1*(Diabetes$bp.1s>140)
library(rms)
uu <- datadist(Diabetes)
options(datadist="uu")
fit=lrm(hypertension~rcs(age)+gender+hdl,data=Diabetes)
splinePlot.lrm(fit,xvar="age",xvalues=seq(30,50,1))
```

---

stripes *Background and grid color control.*

Description

Some users like background colors, and it may be helpful to have grid lines to read off e.g. probabilities from a Kaplan-Meier graph. Both things can be controlled with this function. However, it mainly serves plot.prodlim.

Usage

```
stripes(xlim, ylim, col = "white", lwd = 1, gridcol = "gray77",
  fill = "white", horizontal = NULL, vertical = NULL, border = "black",
  xpd = FALSE)
```

Arguments

xlim	Limits for the horizontal x-dimension. Defaults to par("usr")[1:2].
ylim	Limits for the vertical y-dimension.
col	Colors use for the stripes. Can be a vector of colors which are then repeated appropriately.
lwd	Line width
gridcol	Color of grid lines
fill	Color to fill the background rectangle given by par("usr").
horizontal	Numerical values at which to show horizontal grid lines, and at which to change the color of the stripes.
vertical	Numerical values at which to show vertical grid lines.
border	If a fill color is provided, the color of the border around the background.
xpd	From help(par): A logical value or NA. If FALSE, all plotting is clipped to the plot region, if TRUE, all plotting is clipped to the figure region, and if NA, all plotting is clipped to the device region. See also clip.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

```
plot(0,0)
backGround(bg="beige",fg="red",vertical=0,horizontal=0)

plot(0,0)
stripes(col=c("yellow","green"),gridcol="red",xlim=c(-1,1),horizontal=seq(0,1,.1))
stripes(col=c("yellow","green"),gridcol="red",horizontal=seq(0,1,.1))
```

---

summary.ci *Summarize confidence intervals*

Description

Summarize confidence intervals

Usage

```
## S3 method for class 'ci'
summary(object, format = "[u;l]", se = FALSE, print = TRUE,
  ...)
```

Arguments

object	Object of class ci containing point estimates and the corresponding confidence intervals
format	A string which indicates the format used for confidence intervals. The string is passed to formatCI with two arguments: the lower and the upper limit. For example '(l;u)' yields confidence intervals with round parenthesis in which the upper and the lower limits are separated by semicolon.
se	If TRUE add standard error.
print	Logical: if FALSE do not actually print confidence intervals but just return them invisibly.
...	used to control formatting of numbers

Details

This format of the confidence intervals is user-manipulable.

Value

Formatted confidence intervals

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

ci plot.ci format.ci

Examples

```
library(lava)
m <- lvm(Y~X)
m <- categorical(m,Y~X,K=4)
set.seed(4)
d <- sim(m,24)
ci.mean(Y~X,data=d)
x <- summary(ci.mean(Y~X,data=d),digits=2)
x
x <- summary(ci.mean(Y~X,data=d),format="(u,l)",digits=2)
x <- summary(ci.mean(Y~X,data=d),format="(u,l)",digits=1,se=TRUE)
x <- summary(ci.mean(Y~X,data=d),format="(u,l)",digits=1,handler="format")
x <- summary(ci.mean(Y~X,data=d),format="(u,l)",digits=1,handler="prettyNum")
```

---

summary.regressionTable

Formatting regression tables

Description

Preparing regression results for publication

Usage

```
## S3 method for class 'regressionTable'
summary(object, show.missing = "ifany",
  print = TRUE, ...)
```

Arguments

object	object obtained with regressionTable or summary.regressionTable.
show.missing	Decide if number of missing values are shown. Either logical or character. If 'ifany' then number missing values are shown if there are some.
print	If TRUE print results.
...	Used to control formatting of parameter estimates, confidence intervals and p-values. See examples.

Value

List with two elements:

- regressionTable: the formatted regression table (a data.frame)
- rawTable: table with the unformatted values (a data.frame)

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

publish.glm publish.coxph

Examples

```
library(survival)
data(pbc)
pbc$edema <- factor(pbc$edema,levels=c("0","0.5","1"),labels=c("0","0.5","1"))
fit = coxph(Surv(time,status!=0)~age+sex+edema+log(bili)+log(albumin)+log(protime),
            data=pbc)
u=summary(regressionTable(fit))
u$regressionTable
u$rawTable
summary(regressionTable(fit),handler="prettyNum")
summary(regressionTable(fit),handler="format")
summary(regressionTable(fit),handler="sprintf",digits=c(2,2),pValue.stars=TRUE)
```

summary.univariateTable

Preparing univariate tables for publication

Description

Summary function for univariate table

Usage

```
## S3 method for class 'univariateTable'
summary(object, n = "inNames",
  drop.reference = FALSE, pvalue.stars = FALSE, pvalue.digits = 4,
  show.missing = c("ifany", "always", "never"), show.pvalues, show.totals,
  ...)
```

Arguments

object	univariateTable object as obtained with function univariateTable.
n	If not missing, show the number of subjects in each column. If equal to "inNames", show the numbers in parentheses in the column names. If missing the value object$n is used.
drop.reference	Logical or character (vector). Decide if line with reference level should be suppressed for factors. If TRUE or "all" suppress for all categorical factors. If 'binary' suppress only for binary variables. Can be character vector in which case reference lines are suppressed for variables that are included in the vector.
pvalue.stars	If TRUE use symnum to parse p-values otherwise use format.pval.
pvalue.digits	Passed to format.pval.
show.missing	Decides if number of missing values are shown in table. Defaults to "ifany", and can also be set to "always" or "never".
show.pvalues	Logical. If set to FALSE the column p-values is removed. If missing the value object$compare.groups[[1]]==TRUE is used.
show.totals	Logical. If set to FALSE the column Totals is removed. If missing the value object$show.totals is used.
...	passed on to labelUnits. This overwrites labels stored in object$labels

Details

Collects results of univariate table in a matrix.

Value

Summary table

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
data(Diabetes)
u <- univariateTable(gender~age+location+Q(BMI)+height+weight,
                data=Diabetes)
summary(u)
summary(u,n=NULL)
summary(u,pvalue.digits=2,"age"="Age (years)","height"="Body height (cm)")

u2 <- univariateTable(location~age+AgeGroups+gender+height+weight,
                data=Diabetes)
summary(u2)
summary(u2,drop.reference=TRUE)
## same but more flexible
summary(u2,drop.reference=c("binary"))
## same but even more flexible
summary(u2,drop.reference=c("gender"))
```

`sutable` *Fast summary of a univariate table*

Description

First apply univariateTable then call summary.

Usage

```
sutable(...)
```

Arguments

`...`	Unnamed arguments and are passed to `univariateTable` as well as named arguments that match `univariateTable`'s arguments, other arguments are passed to `summary.univariateTable`

Value

Summary table

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

summary.univariateTable univariateTable

Examples

```
data(Diabetes)
sutable(gender~age+location+Q(BMI)+height+weight,data=Diabetes,BMI="Body mass index (kg/m^2)")
```

table2x2 *2x2 table calculus for teaching*

Description

2x2 table calculus for teaching

Usage

```
table2x2(x, digits = 1, stats = c("table", "rd", "rr", "or", "chisq",
  "fisher"))
```

Arguments

x	2x2 table
digits	rounding digits
stats	subset or all of c("table","rd","or","rr","chisq","fisher") where rd= risk difference, rr = risk ratio, or = odds ratio, chisq = chi-square test, fisher= fisher's exact test and table = the 2x2 table

Details

2x2 table calculus for teaching

Value

see example

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
table2x2(table("marker"=rbinom(100,1,0.4),"response"=rbinom(100,1,0.1)))
table2x2(matrix(c(71,18,38,8),ncol=2),stats="table")
table2x2(matrix(c(71,18,38,8),ncol=2),stats=c("rr","fisher"))
```

trace *trace data*

Description

These data are from screening to the TRACE study, a comparison between the angiotensin converting enzyme inhibitor trandolapril and placebo ford large myocardial infarctions. A total of 6676 patients were screened for the study. Survival has been followed for the screened population for 16 years. The current data has been prepared for a poisson regression to examine survival. The data has been "split" in 0.5 year intervals (plitLexis function from Epi package) and then collapsed on all variables (aggregate function).

Format

A data frame with 1832 observations on the following 6 variables.

Time Time after myocardial infarction, in 6 months intervals

smoking Smoking status. A factor with levels (Never, Current, Previous)

sex A factor with levels (Female, Male)

age Age in years at the time of myocardial infarction

ObsTime Cumulative risk time in each split

dead Count of deaths

References

Kober et al 1995 Am. J. Cardiol 76,1-5

Examples

```
data(trace)
Units(trace,list("age"="years"))
fit <- glm(dead ~ smoking+sex+age+Time+offset(log(ObsTime)), family="poisson",data=trace)
rtf <- regressionTable(fit,factor.reference = "inline")
summary(rtf)
publish(fit)
```

`Units` *Add units to data set*

Description

Add variable units to data.frame (or data.table).

Usage

```
Units(object, units)
```

Arguments

`object`	A data.frame or data.table
`units`	Named list of units. Names are variable names. If omitted, show existing units.

Details

If the object has units existing units are replaced by given units.

Value

The object augmented with attribute `"units"`

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
data(Diabetes)
Diabetes <- Units(Diabetes,list(BMI="kg/m^2"))
Units(Diabetes)
Diabetes <- Units(Diabetes,list(bp.1s="mm Hg",bp.2s="mm Hg"))
Units(Diabetes)
```

`univariateTable` *Univariate table*

Description

Categorical variables are summarized using counts and frequencies.

Usage

```
univariateTable(formula, data = parent.frame(),
  summary.format = "mean(x) (sd(x))", Q.format = "median(x) [iqr(x)]",
  freq.format = "count(x) (percent(x))", column.percent = TRUE,
  digits = c(1, 1, 3), short.groupnames, compare.groups = TRUE,
  show.totals = TRUE, n = "inNames", outcome = NULL, na.rm = FALSE, ...)
```

Arguments

`formula`	Formula specifying the grouping variable (strata) on the left hand side (can be omitted) and on the right hand side the variables for which to obtain (descriptive) statistics.
`data`	Data set in which formula is evaluated
`summary.format`	Format for the numeric (non-factor) variables. Default is mean (SD). If different formats are desired, either special Q can be used or the function is called multiple times and the results are rbinded. See examples.
`Q.format`	Format for quantile summary of numerical variables: Default is median (inter quartile range).
`freq.format`	Format for categorical variables. Default is count (percentage).
`column.percent`	Logical, if `TRUE` and the default freq.format is used then column percentages are given instead of row percentages for categorical variables (factors).
`digits`	Number of digits
`short.groupnames`	If `TRUE` group names are abbreviated.
`compare.groups`	Method used to compare groups. If `"logistic"` and there are exactly two groups logistic regression is used instead of t-tests and Wilcoxon rank tests to compare numeric variables across groups.
`show.totals`	If `TRUE` show a column with totals.
`n`	If `TRUE` show the number of subjects as a separate row. If equal to `"inNames"`, show the numbers in parentheses in the column names. If `FALSE` do not show number of subjects.
`outcome`	Outcome data used to calculate p-values when compare groups method is `'logistic'` or `'cox'`.

na.rm	If TRUE remove missing values from categorical variables when calculating p-values.
...	saved as part of the result to be passed on to labelUnits

Details

This function can generate the baseline demographic characteristics that forms table 1 in many publications. It is also useful for generating other tables of univariate statistics.

The result of the function is an object (list) which containe the various data generated. In most applications the summary function should be applied which generates a data.frame with a (nearly) publication ready table. Standard manipulation can be used to modify, add or remove columns/rows and for users not accustomed to R the table generated can be exported to a text file which can be read by other software, e.g., via write.csv(table,file="path/to/results/table.csv")

Continuous variables are summarized by means and standard deviations. Deviations from the above defaults are obtained when the arguments summary.format and freq.format are combined with suitable summary functions.

Value

List with one summary table element for each variable on the right hand side of formula. The summary tables can be combined with rbind. The function summary.univariateTable combines the tables, and shows p-values in custom format. The summary tables

Author(s)

Thomas A. Gerds

See Also

summary.univariateTable, publish.univariateTable

Examples

```
data(Diabetes)
univariateTable(~age,data=Diabetes)
univariateTable(~gender,data=Diabetes)
univariateTable(~age+gender+ height+weight,data=Diabetes)
## same thing but less typing
utable(~age+gender+ height+weight,data=Diabetes)

## summary by location:
univariateTable(location~Q(age)+gender+height+weight,data=Diabetes)
## continuous variables marked with Q() are (by default) summarized
## with median (IQR) and kruskal.test (with two groups equivalent to wilcox.test)
## variables not marked with Q() are (by default) summarized
## with mean (sd) and anova.glm(...,test="Chisq")
## the p-value of anova.glm with only two groups is similar
## but not exactly equal to that of a t.test
## categorical variables are (by default) summarized by count
## (percent) and anova.glm(...,family=binomial,test="Chisq")

## export result to csv
table1 = summary(univariateTable(location~age+gender+height+weight,data=Diabetes),
show.pvalues=FALSE)
```

```
# write.csv(table1,file="~/table1.csv",rownames=FALSE)

## change labels and values
utable(location~age+gender+height+weight,data=Diabetes,
       age="Age (years)",gender="Sex",
       gender.female="Female",
       gender.male="Male",
       height="Body height (inches)",
       weight="Body weight (pounds)")

## Use quantiles and rank tests for some variables and mean and standard deviation for others
univariateTable(gender~Q(age)+location+Q(BMI)+height+weight,
                data=Diabetes)

## Factor with more than 2 levels
Diabetes$AgeGroups <- cut(Diabetes$age,
                          c(19,29,39,49,59,69,92),
                          include.lowest=TRUE)
univariateTable(location~AgeGroups+gender+height+weight,
                data=Diabetes)

## Row percent
univariateTable(location~gender+age+AgeGroups,
                data=Diabetes,
                column.percent=FALSE)

## change of frequency format
univariateTable(location~gender+age+AgeGroups,
                data=Diabetes,
                column.percent=FALSE,
                freq.format="percent(x) (n=count(x))")

## changing Labels
u <- univariateTable(location~gender+AgeGroups+ height + weight,
                     data=Diabetes,
                     column.percent=TRUE,
                     freq.format="count(x) (percent(x))")
summary(u,"AgeGroups"="Age (years)","height"="Height (inches)")

## more than two groups
Diabetes$frame=factor(Diabetes$frame,levels=c("small","medium","large"))
univariateTable(frame~gender+BMI+age,data=Diabetes)

Diabetes$sex=as.numeric(Diabetes$gender)
univariateTable(frame~sex+gender+BMI+age,
                data=Diabetes,freq.format="count(x) (percent(x))")

## multiple summary formats
## suppose we want for some reason mean (range) for age
## and median (range) for BMI.
## method 1:
univariateTable(frame~Q(age)+BMI,
                data=Diabetes,
                Q.format="mean(x) (range(x))",
                summary.format="median(x) (range(x))")
## method 2:
u1 <- summary(univariateTable(frame~age,
```

```
                              data=na.omit(Diabetes),
                              summary.format="mean(x) (range(x))"))
u2 <- summary(univariateTable(frame~BMI,
                              data=na.omit(Diabetes),
                              summary.format="median(x) (range(x))"))
publish(rbind(u1,u2),digits=2)
```

Index