Package ‘Publish’

December 4, 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 2019.12.04

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

Depends prodlim (&gt;= 1.5.4)

Imports survival (&gt;= 2.38), data.table (&gt;= 1.10.4), lava (&gt;= 1.5.1), multcomp (&gt;= 1.4)

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

License GPL (&gt;= 2)

RoxygenNote 6.1.1

NeedsCompilation no

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

Repository CRAN

Date/Publication 2019-12-04 22:30:10 UTC

R topics documented:

  Publish-package .................................................. 3
  acut .............................................................. 3
  ci.mean .......................................................... 5
  ci.mean.default ................................................ 6
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CiTable                               .................................................. 7</td>
</tr>
<tr>
<td>coxphSeries                           .................................................. 7</td>
</tr>
<tr>
<td>Diabetes                              .................................................. 8</td>
</tr>
<tr>
<td>fixRegressionTable                    .................................................. 10</td>
</tr>
<tr>
<td>followupTable                         .................................................. 11</td>
</tr>
<tr>
<td>formatCI                              .................................................. 12</td>
</tr>
<tr>
<td>glmSeries                             .................................................. 13</td>
</tr>
<tr>
<td>labelUnits                            .................................................. 14</td>
</tr>
<tr>
<td>lazyDateCoding                        .................................................. 15</td>
</tr>
<tr>
<td>lazyFactorCoding                      .................................................. 16</td>
</tr>
<tr>
<td>org                                   .................................................. 17</td>
</tr>
<tr>
<td>parseInteractionTerms                 .................................................. 17</td>
</tr>
<tr>
<td>plot.ci                               .................................................. 19</td>
</tr>
<tr>
<td>plot.regressionTable                  .................................................. 20</td>
</tr>
<tr>
<td>plot.subgroupAnalysis                 .................................................. 22</td>
</tr>
<tr>
<td>plotConfidence                        .................................................. 23</td>
</tr>
<tr>
<td>print.ci                              .................................................. 30</td>
</tr>
<tr>
<td>print.subgroupAnalysis                .................................................. 31</td>
</tr>
<tr>
<td>print.table2x2                        .................................................. 32</td>
</tr>
<tr>
<td>print.univariateTable                 .................................................. 33</td>
</tr>
<tr>
<td>publish                               .................................................. 33</td>
</tr>
<tr>
<td>publish.CauseSpecificCox             .................................................. 35</td>
</tr>
<tr>
<td>publish.ci                            .................................................. 36</td>
</tr>
<tr>
<td>publish.coxph                         .................................................. 37</td>
</tr>
<tr>
<td>publish.glm                           .................................................. 39</td>
</tr>
<tr>
<td>publish.htest                          .................................................. 41</td>
</tr>
<tr>
<td>publish.matrix                        .................................................. 42</td>
</tr>
<tr>
<td>publish.MIresult                      .................................................. 43</td>
</tr>
<tr>
<td>publish.riskRegression                .................................................. 46</td>
</tr>
<tr>
<td>publish.Score                         .................................................. 47</td>
</tr>
<tr>
<td>publish.summary.aov                   .................................................. 48</td>
</tr>
<tr>
<td>publish.survdiff                      .................................................. 49</td>
</tr>
<tr>
<td>regressionTable                       .................................................. 49</td>
</tr>
<tr>
<td>SpaceT                                .................................................. 52</td>
</tr>
<tr>
<td>spaghettiogram                        .................................................. 52</td>
</tr>
<tr>
<td>specialFrame                          .................................................. 53</td>
</tr>
<tr>
<td>splinePlot.lrm                         .................................................. 55</td>
</tr>
<tr>
<td>stripes                               .................................................. 57</td>
</tr>
<tr>
<td>subgroupAnalysis                      .................................................. 58</td>
</tr>
<tr>
<td>summary.ci                            .................................................. 60</td>
</tr>
<tr>
<td>summary.regressionTable               .................................................. 61</td>
</tr>
<tr>
<td>summary.subgroupAnalysis              .................................................. 62</td>
</tr>
<tr>
<td>summary.univariateTable               .................................................. 63</td>
</tr>
<tr>
<td>suitable                              .................................................. 65</td>
</tr>
<tr>
<td>table2x2                              .................................................. 66</td>
</tr>
<tr>
<td>trace                                 .................................................. 67</td>
</tr>
<tr>
<td>traceR                                .................................................. 68</td>
</tr>
</tbody>
</table>
Description

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

acut

*Automatic selection and formatting of breaks in cut*

Description

A version of cut that easily formats the labels and places breaks by default.

Usage

```r
acut(x, n = 5, format = NULL, format.low = NULL, format.high = NULL, dig.lab = 3, right = TRUE, breaks, labels = TRUE, ...)
```

Arguments

- `x`: a numeric vector which is to be converted to a factor by cutting (passed directly to cut).
- `n`: number of bins to create based on the empirical quantiles of x. This will be overruled if breaks is supplied.
- `format`: string used to make labels. `%l` and `%u` identifies the lower and upper value of the breaks respectively. See examples.
- `format.low`: string used specifically on the lowest label.
- `format.high`: string used specifically on the highest label.
- `dig.lab`: integer which is used when labels are not given. It determines the number of digits used in formatting the break numbers. (Passed directly to cut.)
- `right`: logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa (passed directly to cut).
- `breaks`: specify breaks manually as in cut.
- `labels`: logical, indicating whether or not to make labels or simply use ordered numbers. If TRUE, the labels are constructed as described above.
- `...`: further arguments passed to cut.
Details

The formats are supplied by specifying the text around the lower (%l) and upper (%u) value (see examples). If user specified breaks are supplied, the default labels from cut are used. If automatic breaks are used, the default labels are a slight modification at the end point of the default from cut. All this can of course be adjusted manually through the format functionality (see below).

By default, 5 breaks are constructed according to the quantiles with of the input x. The number of breaks can be adjusted, and default specifying breaks (as in cut) can be supplied instead.

Value

same as for cut. A vector of 'factors' is created, unless 'labels=FALSE'.

Author(s)

Anders Munch

Examples

data(Diabetes) # load dataset

## The default uses format similar to cut
chol.groups <- acut(Diabetes$chol)
table(chol.groups)

## The formatting can easily be changed
chol.groups <- acut(Diabetes$chol,format="%l-%u",n=5)
table(chol.groups)

## The default is to automatic place the breaks, so the number of this can easily be changed.
chol.groups <- acut(Diabetes$chol,n=7)
table(chol.groups)

## Manually setting format and breaks
age.groups <- acut(Diabetes$age,format="%l-%u",breaks=seq(0,100,by=10))
table(age.groups)

## Other variations
age.groups <- acut(Diabetes$age,
  format="%l-%u",
  format.low="below %u",
  format.high="above %l",
  breaks=c(0, seq(20,80,by=10), Inf))
table(age.groups)

BMI.groups <- acut(Diabetes$BMI,
  format="BMI between %l and %u",
  format.low="BMI below %u",
  format.high="BMI above %l")
table(BMI.groups)
org(as.data.frame(table(BMI=BMI.groups)))
## Instead of using the quantiles, we can specify equally spaced breaks,
## but still get the same formatting
BMI.grouping <- seq(min(Diabetes$BMI, na.rm = TRUE), max(Diabetes$BMI, na.rm = TRUE), length.out = 6)
BMI.grouping[1] <- -Inf # To get all included
BMI.groups <- acut(Diabetes$BMI,
   breaks = BMI.grouping,
   format = "BMI between %l and %u",
   format.low = "BMI below %u",
   format.high = "BMI above %l")
table(BMI.groups)
org(as.data.frame(table(BMI = BMI.groups)))

---

ci.mean

**ci.mean**

*Compute mean values with confidence intervals*

### Description

Compute mean values with confidence intervals

### Usage

```r
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

```r
## 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

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 associated 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

Examples

```r
data(Diabetes)
```

Description

Expand regression coefficient table

Usage

```r
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**

```r
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**

```r
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

```r
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
**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**

```r
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

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
lazyDateCoding

Description

This function eases the process of generating date variables. All variables in a data.frame which
match a regular expression are included.

Usage

lazyDateCoding(data, format, pattern, varnames, testlength = 10)

Arguments

data Data frame in which to search for date variables.
format passed to as.Date
pattern match date variables
varnames variable names
testlength how many rows of data should be evaluated to guess the format.
lazyFactorCoding

Lazy Factor Coding

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 = "\": ", ...)
```
parseInteractionTerms

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 contrast 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")
pasteInteractionTerms(terms=tt,xlevels=xlev,format.contrast="var(level:ref)"

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

data(Diabetes)
fit <- glm(bp.2s~age*factor(gender)+BMI,data=Diabetes)
pasteInteractionTerms(terms=terms(fit$formula),xlevels=fit$xlevels,
  format.scale="var -- level:ref",units=list("age"='years'))
pasteInteractionTerms(terms=terms(fit$formula),xlevels=fit$xlevels,
  format.scale.unit="var -- level:ref",units=list("age"='years'))
it <- pasteInteractionTerms(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 ele-
moment) 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

```r
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

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

**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**

```r
## 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))
```
plot.subgroupAnalysis

Description

This function operates on a "subgroupAnalysis" object to produce a formatted table and a forest plot.

Usage

## S3 method for class 'subgroupAnalysis'
plot(x, ...)

Arguments

x - a subgroupAnalysis object

... - passed on to plotConfidence

Details

This function produces a formatted table of a subgroupAnalysis object and adds a forest plot. If further details needs attention before plotting is is advisable use adjust the table produced by the summary function and then plotting with the plotConfidence function.

Author(s)

Christian Torp-Pedersen

See Also

subgroupAnalysis, plotConfidence

Examples

#load libraries
library(Publish)
library(survival)
library(data.table)
data(traceR) #get dataframe traceR
setDT(traceR)
traceR[, wmi2 = factor(wallMotionIndex<0.9, levels=c(TRUE,FALSE), labels=c("bad","good")),
      abd2 = factor(abdominalCircumference<95, levels=c(TRUE,FALSE), labels=c("slim","fat"))]
fit_cox <- coxph(Surv(observationTime,dead)~treatment,data=traceR)
# Selected subgroups - univariable analysis
sub_cox <- subgroupAnalysis(fit_cox,traceR,treatment="treatment",subgroup=c("smoking","sex","wmi2","abd2")) # subgroups as character string
plot(sub_cox)
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 arrangement 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

```r
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

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 examples.
Function to plot means and other point estimates with confidence intervals, their values and additional labels. Horizontal margins as determined by $\text{par()\$mar}$ 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., $\text{par(oma=}c(0,4,0,4))$.

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

```r
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","Time","Mean","SD","n")],CiTable[,"Drug"])
labellist

## the data need to be ordered accordingly
CC= data.table::rbindlist(split(CiTable[,c("HazardRatio","lower","upper")],CiTable[,"Drug"]))
plotConfidence(x=CC, 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,
plotConfidence

## 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")],
               xaxis.at=c(0.8,1.1,1.3),
               labels=CiTable[,c("Drug.Time","Dose","Mean","SD","n")],
               xlim=c(0.8,1.3))

## 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),plot.log="x")

## 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))
labels=as.character(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
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)

# combining multiple plots into one
layout(t(matrix(1:5)))
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               labels=CiTable[,c("Mean","n")],
               layout=FALSE)
plotConfidence(x=CiTable[,c("HazardRatio","lower","upper")],
               layout=FALSE)
```

---

**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

```r
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.subgroupAnalysis

*Printing univariate tables*

Description

Print function for subgroupAnalysis

Usage

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

Arguments

- `x` - An object obtained with subgroupAnalysis
- `...` - Passed to summary.subgroupAnalysis

Details

This function is simply calling summary.subgroupAnalysis

Value

The result of summary.subgroupAnalysis(x)

Author(s)

Christian Torp-Pedersen (ctp@heart.dk)

See Also

subgroupAnalysis
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 specifying 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

```r
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**

Tabulating cause-specific hazard ratio from all causes with confidence limits and Wald test p-values.

---

Description

Publish cause-specific Cox models

Usage

```r
## 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

```r
## 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

```r
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

```r
## 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)
**Description**

Tabulate the results of a generalized linear regression analysis.

**Usage**

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

publish.h.test  

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

```r
## S3 method for class 'matrix'
publish(object, title, colnames = TRUE,
        rownames = TRUE, colname = "", 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
- `colname`: 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 arguments 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 environment.
- `latex.table.format`: For style latex only: format of the tabular environment.
- `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

```r
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**

```r
## 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 smcfc based multiple imputations of missing covariates in publishable format

Author(s)

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

Examples

```r
## 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]
d=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",
```
library(smcfcs)
library(mitools)
library(survival)

# surv: Cox regression
library(riskRegression)
set.seed(7)
ds=sampleData(78,outcome="survival")
## generate missing values
ds[X5==1,X6:=NA]
ds[X2==1,X3:=NA]
ds=d[.,(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)
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=d[.,(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)
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=d[.,(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

```r
## 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))

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 subsection 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)
x = Score(list(f1,f2),data=val,formula=Hist(time,event)~1)
publish(x)

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
...

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 \texttt{lm}, \texttt{glm} or \texttt{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 \texttt{lm} and \texttt{glm} objects. 'robust' uses the sandwich form standard error to construct Wald type intervals (see \texttt{lava::estimate.default}). 'simultaneous' calls \texttt{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 \texttt{lava::estimate.default}. If 'simultaneous' call \texttt{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 \texttt{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 \texttt{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".

\texttt{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>
# 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: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)
spaghettiogram

```r
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  

**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 \( Y \sim X + \text{id}(Z) \) 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 \( \text{A} \) of legend is controlled as \( \text{legend.A} \). E.g., when \( \text{legend.cex}=2 \) legend will be called with argument \( \text{cex}=2 \).

- **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

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

---

**specialFrame**  
*Special frame*

**Description**

Extract data and design matrix including specials from call
specialFrame

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>
splinePlot.lrm

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(1,1,1,0,0,1),
    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.
splinePlot.lrm

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 \texttt{rms::lrm}

xvar Name of the variable to show on x-axis

xvalues Sequence of \texttt{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 \texttt{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 \texttt{TRUE} add lines to an existing graph

... Further arguments passed to \texttt{plot}. Only if add is \texttt{FALSE}.

Details

Function which extracts from a logistic regression model fitted with \texttt{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))
**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**

```r
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 used 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**

```r
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))
```
subgroupAnalysis  Subgroup Analysis - Interactions and estimates

Description

The function can examine Cox regression, logistic regression and Poisson regression (Poisson regression for survival analysis) where the effect of one variable is of particular interest. This function systematically checks for effect modification with a list of other variables.

In randomised studies the main regression analysis is often univariate and includes only the exposure of interest. In observational studies the main regression analysis can readily be adjusted for other variables including those which may modify the effect of the variable of interest.

Usage

subgroupAnalysis(object, data, treatment, subgroups, confint.method = "default", factor.reference = "extraline")

Arguments

object - glm, coxph or cph object for which subgroups should be analyzed.
data - Dataset including all relevant variables
treatment - The variable to be examined in subgroups, best coded as 0/1 numeric.
subgroups - A vector of variable names presenting the variables where subgroups should be formed. These variables should all be "factors"
confint.method "default" creates Wald type confidence interval, "robust", creates robust standard errors - see regressionTable function.
factor.reference "extraline" creates an extraline for the reference, "inline" avoids this line.

Details

The function can only handle a bivariate treatment, most conveniently coded as zero or one. The p-value for interaction is obtained with a likelihood ratio test comparing the main regression analysis with the interaction model.

Value

A data.frame with subgroup specifications, number in each subgroup, parameter estimates and p-value for interaction. A forest plot can be obtained with "plotConfidence".

Author(s)

Christian Torp-Pedersen

See Also

coxph, glm, plotConfidence
Examples

```r
# load libraries
library(data.table)
library(Publish)
library(survival)
data(traceR) # get dataframe traceR
data.table::setDT(traceR)
traceR[, 
  wmi2:=factor(wallMotionIndex<0.9, levels=c(TRUE,FALSE),
  labels=c("bad","good")),
  abd2:=factor(abdominalCircumference<95, levels=c(TRUE,FALSE),
  labels=c("slim","fat"))]
traceR[, sex:=as.factor(sex)] # all subgroup variables needs to be factor
traceR[observationTime==0, observationTime:=1] # remove missing covariate values
traceR=na.omit(traceR)

# univariate analysis of smoking in subgroups of age and sex
# Main regression analysis is a simple/univariate Cox regression
fit_cox <- coxph(Surv(observationTime, dead)~ treatment, data=traceR)
sub_cox <- subgroupAnalysis(fit_cox, traceR, treatment="treatment",
  subgroups= c("smoking","sex","wmi2","abd2"))

sub_cox

# to see how the results are obtained consider the variable: smoking
fit_cox_smoke <- coxph(Surv(observationTime, dead)~ treatment*smoking, data=traceR)
# the last three rows of the following output:
publish(fit_cox_smoke)
# are included in the first 3 rows of the result of the sub group analysis:
sub_cox[1:3,]
# the p-value is obtained as:
fit_cox_smoke_add <- coxph(Surv(observationTime, dead)~ treatment + smoking, data=traceR)
anova(fit_cox_smoke_add, fit_cox_smoke, test="Chisq")

# Note that a real subgroup analysis would be to subset the data
fit_cox1a <- coxph(Surv(observationTime, dead)~ treatment, data=traceR[smoking="never"])
fit_cox1b <- coxph(Surv(observationTime, dead)~ treatment, data=traceR[smoking="current"])
fit_cox1c <- coxph(Surv(observationTime, dead)~ treatment, data=traceR[smoking="prior"])

## when the main analysis is already adjusted
fit_cox_adj <- coxph(Surv(observationTime, dead)~ treatment + smoking + sex + wmi2 + abd2, 
data=traceR)
sub_cox_adj <- subgroupAnalysis(fit_cox_adj, traceR, treatment="treatment",
  subgroups= c("smoking","sex","wmi2","abd2")) # subgroups as character string
sub_cox_adj

# When both start and end are in the Surv statement:
traceR[, null:=0]
fit_cox2 <- coxph(Surv(null, observationTime, dead)~ treatment + smoking + sex + wmi2 + abd2, data=traceR)
summary(regressionTable(fit_cox))
sub_cox2 <- subgroupAnalysis(fit_cox2, traceR, treatment="treatment",
  subgroups= c("smoking","sex","wmi2","abd2"))
# Analysis with Poisson - and the unrealistic assumption of constant hazard
```
# and adjusted for age in all subgroups
fit_p <- glm(dead~treatment+age+offset(log(observationTime)),family="poisson",
data=traceR)
sub_pois <- subgroupAnalysis(fit_p,traceR,treatment="treatment",
subgroups=~smoking+sex+wmi2+abd2)

# Analysis with logistic regression - and very wrongly ignoring censoring
fit_log <- glm(dead~treatment+age,family="binomial",data=traceR)
sub_log <- subgroupAnalysis(fit_log,traceR,treatment="treatment",
subgroups=~smoking+sex+wmi2+abd2, factor.reference="inline")

summary.ci

## 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

```r
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**

```r
## 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(regressionTable(fit),handler="sprintf",digits=c(2,2),pValue.stars=TRUE,ci.format="(l,u)")

Description

This function operates on a "subgroupAnalysis" object to produce a formatted table.

Usage

## S3 method for class 'subgroupAnalysis'
summary(object, digits = 3, eps = 0.001,
        subgroup.p = FALSE, keep.digital = FALSE, ...)

Arguments

object - a subgroupAnalysis object
digits - number of digits for risk ratios
esp - lowest value of p to be shown exactly, others will be "<eps"
subgroup.p - present p-values for analyses in subgroups
keep.digital - prevents formatting risk ratio and confidence limits. Useful for cases when further manipulations of rows and columns prior to adding a forest plot is relevant.
... - not currently used
summary.univariateTable

Details

This function produces a formatted or unformatted table of a subgroupAnalysis object. A forest plot can be added with the plot function.

Value

A data.frame with formatted values for subgroups

Author(s)

Christian Torp-Pedersen

See Also

subgroupAnalysis

Examples

#load libraries
library(Publish)
library(survival)
library(data.table)
data(traceR) #get dataframe traceR
setDT(traceR) #get dataframe traceR
traceR[,\'wmi2\':=factor(wallMotionIndex<0.9,levels=c(TRUE,FALSE),
labels=c("bad","good")),

  abd2=factor(abdominalCircumference<95, levels=c(TRUE,FALSE),
labels=c("slim","fat")))
traceR[,sex:=as.factor(sex)] # all subgroup variables needs to be factor
traceR[observationTime==0,observationTime:=1]
# univariate analysis of smoking in subgroups of age and sex
# Basic model from randomised study - but observed for 12 years
fit_cox <- coxph(Surv(observationTime,dead)~treatment,data=traceR)
sub_cox <- subgroupAnalysis(fit_cox,traceR,treatment="treatment",
  subgroup=c("smoking","sex","wmi2","abd2")) # subgroups as character string
summary(sub_cox)

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$n is used.
drop.reference Logical or character (vector). Decide if line with reference level should be sup-
pressed for factors. If TRUE or "all" suppress for all categorical factors. If
'binary' suppress only for binary variables. Can be character vector in which
case reference lines are suppressed for variables that are included in the vector.
pvalue.stars If TRUE use symnum to parse p-values otherwise use format.pval.
pvalue.digits Passed to format.pval.
show.missing Decides if number of missing values are shown in table. Defaults to "ifany",
and can also be set to "always" or "never".
show.pvalues Logical. If set to FALSE the column p-values is removed. If missing the value
object$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)")
```r
e2 <- univariateTable(location~age+AgeGroups+gender+height+weight, 
                     data=Diabetes) 
summary(e2) 
summary(e2, drop.reference=TRUE) 
## same but more flexible 
summary(e2, drop.reference=c("binary")) 
## same but even more flexible 
summary(e2, 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**  
```r 
data(Diabetes) 
sutable(gender~age+location+Q(BMI)+height+weight, data=Diabetes,BMI="Body mass index (kg/m^2)") 
```
Description

2x2 table calculus for teaching

Usage

\[
\text{table2x2}(x, \text{digits} = 1, \text{stats} = \text{c}("\text{table"}, "\text{rd"}, "\text{rr"}, "\text{or"}, "\text{chisq"}, "\text{fisher"}))
\]

Arguments

- \text{x} \quad 2x2 \text{ table}
- \text{digits} \quad \text{rounding digits}
- \text{stats} \quad \text{subset or all of} \ c("\text{table"}, "\text{rd"}, "\text{or"}, "\text{rr"}, "\text{chisq"}, "\text{fisher"}) \text{ 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

\[
\text{table2x2(table("marker"=rbinom(100,1,0.4),"response"=rbinom(100,1,0.1)))}
\]
\[
\text{table2x2(matrix(c(71,18,38,8),ncol=2),stats="table")}
\]
\[
\text{table2x2(matrix(c(71,18,38,8),ncol=2),stats=c("rr","fisher"))}
\]
Description

These data are from screening to the TRACE study, a comparison between the angiotensin converting enzyme inhibitor trandolapril and placebo for 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 (splitLexis 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)
Description

These data are from the TRACE randomised trial, a comparison between the angiotensin converting enzyme inhibitor trandolapril and placebo for large myocardial infarctions. In all, 1749 patients were randomised. The current data are from a 15 year follow-up.

Format

A data frame with 1749 observations on the following variables.

- **weight**  Weight in kilo
- **height** Height in meters
- **abdominalCircumference** in centimeters
- **seCreatinine** in mmol per liter
- **wallMotionIndex** left ventricular function 0-2, 0 worst, 2 normal
- **observationTime** time to death or censor
- **age** age in years
- **sex** 0=female, 1=male
- **smoking** 0=never, 1=prior, 2=current
- **dead** 0=censor, 1=dead
- **treatment** placebo or trandolapril

References

Kober et al 1995 NEJM 333,1670

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

**Description**

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

**Usage**

```r
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**

```r
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

**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), big.mark = ",", 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 differ-
 ent 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
big.mark  For formatting large numbers (i.e., greater than 1,000). "" turn this off.
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.

Author(s)

Thomas A. Gerds

See Also

summary.univariateTable, publish.univariateTable

Examples

data(Diabetes)
library(data.table)
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,
## Use quantiles and rank tests for some variables and mean and standard deviation for others

```r
univariateTable(gender~Q(age)+location+Q(BMI)+height+weight, 
data=Diabetes)
```

## Factor with more than 2 levels

```r
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

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

## change of frequency format

```r
univariateTable(location~gender+age+AgeGroups, 
data=Diabetes, 
column.percent=FALSE, 
freq.format="percent(x) (n=count(x))")
```

## changing Labels

```r
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

```r
Diabetes$frame=factor(Diabetes$frame,levels=c("small","medium","large"))
univariateTable(frame~gender+BMI+age,data=Diabetes)
```

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

## multiple summary formats

```r
## 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:

```r
u1 <- summary(univariateTable(frame~age, 
data=Diabetes, 
Q.format="%mean(x) (range(x))", 
summary.format="median(x) (range(x))")
```
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)

## Large number format (big.mark)
Diabetes$AGE <- 1000*Diabetes$age
u3 <- summary(univariateTable(frame~AGE,
    data=Diabetes,big.mark=""))
Index

*Topic datasets
  CiTable, 7
  Diabetes, 8
  trace, 67
  traceR, 68

*Topic survival
  stripes, 57

acut, 3
anova.coxph, 11

 ci.mean, 5
cia.mean.default, 6
CiTable, 7
coxphSeries, 7
Diabetes, 8

 fixRegressionTable, 10
 followupTable, 11
 formatCI, 12, 36, 60
 glmSeries, 13
 labelUnits, 14
 lazyDateCoding, 15
 lazyFactorCoding, 16
 model.design, 54

 org, 17
 parseInteractionTerms, 17
 plot.ci, 19
 plot.prodlim, 57
 plot.regressionTable, 20
 plot.subgroupAnalysis, 22
 plotConfidence, 23
 print.ci, 30
 print.subgroupAnalysis, 31
 print.summary.regressionTable
  (summary.regressionTable), 61

 print.table2x2, 32
 print.univariateTable, 33
 pubformat, 33
 publish, 34
 Publish-package, 3
 publish.CauseSpecificCox, 35
 publish.ci, 36
 publish.coxph, 37
 publish.glm, 39
 publish.h test, 41
 publish.matrix, 42, 46
 publish.MIresult, 43
 publish.riskRegression, 46
 publish.Score, 47
 publish.summary.aov, 48
 publish.survdiff, 49

 regressionTable, 49

 segments, 25
 SpaceT, 52
 Spaghettogram (spaghettogram), 52
 spaghettogram, 52
 specialFrame, 53
 splinePlot.lrm, 55
 strata, 54
 strip.terms, 54
 stripes, 57
 subgroupAnalysis, 58
 summary.ci, 60
 summary.regressionTable, 61
 summary.subgroupAnalysis, 62
 summary.univariateTable, 63
 sutable, 65

 table2x2, 66
 trace, 67
 traceR, 68

 Units, 69
univariateTable, 69
utable, 69
utable (univariateTable), 69