Package ‘jstable’

May 7, 2024

Title Create Tables from Different Types of Regression

Version 1.2.5

Date 2024-05-08

Description Create regression tables from generalized linear model(GLM), generalized estimating equation(GEE), generalized linear mixed-effects model(GLMM), Cox proportional hazards model, survey-weighted generalized linear model(svyglm) and survey-weighted Cox model results for publication.

Depends R (>= 3.4.0)

License Apache License 2.0

Encoding UTF-8

RoxygenNote 7.3.1

Imports geepack, lme4, stats, data.table, labelled, tableone, coxme, survival (>= 3.0.0), survey, methods, dplyr, purrr, magrittr, tibble

URL https://github.com/jinseob2kim/jstable

BugReports https://github.com/jinseob2kim/jstable/issues

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

LazyData true

NeedsCompilation no

Author Jinseob Kim [aut, cre] (<https://orcid.org/0000-0002-9403-605X>), Zarathu [cph, fnd], Yoonkyoung Jeon [aut]

Maintainer Jinseob Kim <jinseob2kim@gmail.com>

Repository CRAN

Date/Publication 2024-05-07 16:50:03 UTC
R topics documented:

coefNA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
cox2.display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
coxExp . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
coxme.display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
coxmeTable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
CreateTableOne2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
CreateTableOneJS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
extractAIC.coxme . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
geeExp . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
geeglm.display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
geeUni . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
glmshow.display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
LabelepiDisplay . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
LabeljsCox . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
LabeljsGeeglm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
LabeljsMetric . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
LabeljsMixed . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
LabeljsRanef . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
LabeljsTable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
lmer.display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
lmerExp . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
mk.lev . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
mk.lev.var . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
mort . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
opt.data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
opt.roc . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
opt.simpledown . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
opt.tb1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
opt.tbreg . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
svycox.display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
svyCreateTableOne2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
cvyCreateTableOneJS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
svyregress.display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
TableSubgroupCox . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
TableSubgroupGLM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
TableSubgroupMultiCox . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
TableSubgroupMultiGLM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

Index 41
coefNA  \hspace{1cm} \textit{coefNA}: make coefficient table with NA

\underline{Description}  
Make coefficient table with NA

\underline{Usage}  
\texttt{coefNA(model)}

\underline{Arguments}  
\begin{itemize}
  \item \texttt{model} \hspace{1cm} glm object (gaussian or binomial)
\end{itemize}

\underline{Details}  
DETAILS

\underline{Value}  
coefficient table with NA

\underline{Examples}  
\texttt{coefNA(glm(mpg \sim wt + qsec, data = mtcars))}

\hspace{1cm} \underline{cox2.display}  \hspace{1cm} \textit{cox2.display}: table for coxph.object with model option: TRUE - allow "frailty" or "cluster" model

\underline{Description}  
Table for coxph.object with model option: TRUE - allow "frailty" or "cluster" model

\underline{Usage}  
\texttt{cox2.display(cox.obj.withmodel, dec = 2)}

\underline{Arguments}  
\begin{itemize}
  \item \texttt{cox.obj.withmodel} \hspace{1cm} \texttt{coxph.object with model option: TRUE}
  \item \texttt{dec} \hspace{1cm} \texttt{Decimal point, Default: 2}
\end{itemize}
Details

GEE like - cluster, Mixed effect model like - frailty

Value

Table, cluster/frailty info, metrics, caption

Examples

```r
library(survival)
data(lung)
fit1 <- coxph(Surv(time, status) ~ ph.ecog + age + cluster(inst), data = lung, model = TRUE)
fit2 <- coxph(Surv(time, status) ~ ph.ecog + age + frailty(inst), data = lung, model = TRUE)
cox2.display(fit1)
cox2.display(fit2)
```

Description

Transform the unit of coefficients to "HR"

Usage

```r
coxExp(cox.coef, dec)
```

Arguments

- `cox.coef`: cox model coefficients
- `dec`: Decimal point

Details

DETAILS

Value

The transforemed coefficients

Examples

```r
library(coxme)
fit <- coxme(Surv(time, status) ~ ph.ecog + age + (1 | inst), lung)
jstable:::coxExp(jstable:::coxmeTable(fit))
```
coxme.display

coxme.display: table for coxme.object (coxme package)

Description
Make mixed effect model results from coxme.object (coxme package)

Usage
coxme.display(coxme.obj, dec = 2)

Arguments
coxme.obj coxme.object
dec Decimal point, Default: 2

Details
DETAILS

Value
Fixed effect table, random effect, metrics, caption

Examples
library(coxme)
fit <- coxme(Surv(time, status) ~ ph.ecog + age + (1 | inst), lung)
coxme.display(fit)

coxmeTable
coxmeTable: Summary table of coxme.object(internal function)

Description
Extract fixed effect table in coxme.object

Usage
coxmeTable(mod)

Arguments
mod coxme.object
Details

DETAILS

Value

beta, se, z, p of fixed effects

Examples

library(coxme)
fit <- coxme(Surv(time, status) ~ ph.ecog + age + (1 | inst), lung)
jstable::coxmeTable(fit)

---

CreateTableOne2: Modified CreateTableOne function in tableone package

Description

Combine CreateTableOne & print function in tableone package

Usage

CreateTableOne2(
  data,
  strata,
  vars,
  factorVars,
  includeNA = F,
  test = T,
  testApprox = chisq.test,
  argsApprox = list(correct = TRUE),
  testExact = fisher.test,
  argsExact = list(workspace = 2 * 10^5),
  testNormal = oneway.test,
  argsNormal = list(var.equal = F),
  testNonNormal = kruskal.test,
  argsNonNormal = list(NULL),
  showAllLevels = T,
  printToggle = F,
  quote = F,
  smd = F,
  Labels = F,
  exact = NULL,
  nonnormal = NULL,
  catDigits = 1,
  contDigits = 2,
Arguments

data  A data frame in which these variables exist. All variables (both vars and strata) must be in this data frame.

strata  Stratifying (grouping) variable name(s) given as a character vector. If omitted, the overall results are returned.

vars  Variables to be summarized given as a character vector. Factors are handled as categorical variables, whereas numeric variables are handled as continuous variables. If empty, all variables in the data frame specified in the data argument are used.

factorVars  Numerically coded variables that should be handled as categorical variables given as a character vector. Do not include factors, unless you need to relevel them by removing empty levels. If omitted, only factors are considered categorical variables. The variables specified here must also be specified in the vars argument.

includeNA  If TRUE, NA is handled as a regular factor level rather than missing. NA is shown as the last factor level in the table. Only effective for categorical variables., Default: F

test  If TRUE, as in the default and there are more than two groups, groupwise comparisons are performed, Default: T

testApprox  A function used to perform the large sample approximation based tests. The default is chisq.test. This is not recommended when some of the cell have small counts like fewer than 5, Default: chisq.test

argsApprox  A named list of arguments passed to the function specified in testApprox. The default is list(correct = TRUE), which turns on the continuity correction for chisq.test, Default: list(correct = TRUE)

testExact  A function used to perform the exact tests. The default is fisher.test. If the cells have large numbers, it will fail because of memory limitation. In this situation, the large sample approximation based should suffice., Default: fisher.test

argsExact  A named list of arguments passed to the function specified in testExact. The default is list(workspace = 2 * 10^5), which specifies the memory space allocated for fisher.test, Default: list(workspace = 2 * 10^5)

testNormal  A function used to perform the normal assumption based tests. The default is oneway.test. This is equivalent of the t-test when there are only two groups, Default: oneway.test

argsNormal  A named list of arguments passed to the function specified in testNormal. The default is list(var.equal = TRUE), which makes it the ordinary ANOVA that assumes equal variance across groups., Default: list(var.equal = F)
testNonNormal A function used to perform the nonparametric tests. The default is `kruskal.test` (Kruskal-Wallis Rank Sum Test). This is equivalent of the `wilcox.test` (Man-Whitney U test) when there are only two groups. Default: `kruskal.test`

argsNonNormal A named list of arguments passed to the function specified in testNonNormal. The default is `list(NULL)`, which is just a placeholder. Default: `list(NULL)`

showAllLevels Whether to show all levels. FALSE by default, i.e., for 2-level categorical variables, only the higher level is shown to avoid redundant information. Default: T

printToggle Whether to print the output. If FALSE, no output is created, and a matrix is invisibly returned. Default: F

quote Whether to show everything in quotes. The default is FALSE. If TRUE, everything including the row and column names are quoted so that you can copy it to Excel easily. Default: F

smd If TRUE, as in the default and there are more than two groups, standardized mean differences for all pairwise comparisons are calculated. Default: F

Labels Use Label, Default: F

exact A character vector to specify the variables for which the p-values should be those of exact tests. By default all p-values are from large sample approximation tests (chisq.test). Default: NULL

nonnormal A character vector to specify the variables for which the p-values should be those of nonparametric tests. By default all p-values are from normal assumption-based tests (oneway.test). Default: NULL

catDigits Number of digits to print for proportions. Default: 1

contDigits Number of digits to print for continuous variables. Default: 2

pDigits Number of digits to print for p-values (also used for standardized mean differences). Default: 3

labeldata labeldata to use. Default: NULL

minMax Whether to use [min,max] instead of [p25,p75] for nonnormal variables. The default is FALSE.

showpm Logical, show normal distributed continuous variables as Mean ± SD. Default: T

addOverall (optional, only used if strata are supplied) Adds an overall column to the table. Smd and p-value calculations are performed using only the stratified columns. Default: F

Details

DETAILS

Value

A matrix object containing what you see is also invisibly returned. This can be assigned a name and exported via write.csv.
CreateTableOneJS

Examples

```r
library(survival)
CreateTableOne2(vars = names(lung), strata = "sex", data = lung)
```

CreateTableOneJS: Modified CreateTableOne function in tableone package

Description

Combine CreateTableOne & print function in tableone package

Usage

```r
CreateTableOneJS(
  vars,
  strata = NULL,
  strata2 = NULL,
  data,
  factorVars = NULL,
  includeNA = F,
  test = T,
  testApprox = chisq.test,
  argsApprox = list(correct = TRUE),
  testExact = fisher.test,
  argsExact = list(workspace = 2 * 10^5),
  testNormal = oneway.test,
  argsNormal = list(var.equal = F),
  testNonNormal = kruskal.test,
  argsNonNormal = list(NULL),
  showAllLevels = T,
  printToggle = F,
  quote = F,
  smd = F,
  Labels = F,
  exact = NULL,
  nonnormal = NULL,
  catDigits = 1,
  contDigits = 2,
  pDigits = 3,
  labeldata = NULL,
  psub = T,
  minMax = F,
  showpm = T,
  addOverall = F,
  normalityTest = F
)
```
Arguments

vars
Variables to be summarized given as a character vector. Factors are handled as categorical variables, whereas numeric variables are handled as continuous variables. If empty, all variables in the data frame specified in the data argument are used.

strata
Stratifying grouping variable name(s) given as a character vector. If omitted, the overall results are returned.

strata2
Stratifying 2nd grouping variable name(s) given as a character vector. If omitted, the 1 group results are returned.

data
A data frame in which these variables exist. All variables (both vars and strata) must be in this data frame.

factorVars
Numerically coded variables that should be handled as categorical variables given as a character vector. Do not include factors, unless you need to relevel them by removing empty levels. If omitted, only factors are considered categorical variables. The variables specified here must also be specified in the vars argument.

includeNA
If TRUE, NA is handled as a regular factor level rather than missing. NA is shown as the last factor level in the table. Only effective for categorical variables., Default: F

test
If TRUE, as in the default and there are more than two groups, groupwise comparisons are performed, Default: T

testApprox
A function used to perform the large sample approximation based tests. The default is chisq.test. This is not recommended when some of the cell have small counts like fewer than 5, Default: chisq.test

argsApprox
A named list of arguments passed to the function specified in testApprox. The default is list(correct = TRUE), which turns on the continuity correction for chisq.test, Default: list(correct = TRUE)

testExact
A function used to perform the exact tests. The default is fisher.test. If the cells have large numbers, it will fail because of memory limitation. In this situation, the large sample approximation based should suffice., Default: fisher.test

argsExact
A named list of arguments passed to the function specified in testExact. The default is list(workspace = 2 * 10^5), which specifies the memory space allocated for fisher.test, Default: list(workspace = 2 * 10^5)

testNormal
A function used to perform the normal assumption based tests. The default is oneway.test. This is equivalent of the t-test when there are only two groups, Default: oneway.test

argsNormal
A named list of arguments passed to the function specified in testNormal. The default is list(var.equal = TRUE), which makes it the ordinary ANOVA that assumes equal variance across groups., Default: list(var.equal = F)

testNonNormal
A function used to perform the nonparametric tests. The default is kruskal.test (Kruskal-Wallis Rank Sum Test). This is equivalent of the wilcox.test (Man-Whitney U test) when there are only two groups, Default: kruskal.test

argsNonNormal
A named list of arguments passed to the function specified in testNonNormal. The default is list(NULL), which is just a placeholder., Default: list(NULL)
**CreateTableOneJS**

- **showAllLevels**: Whether to show all levels. FALSE by default, i.e., for 2-level categorical variables, only the higher level is shown to avoid redundant information. Default: T
- **printToggle**: Whether to print the output. If FALSE, no output is created, and a matrix is invisibly returned. Default: F
- **quote**: Whether to show everything in quotes. The default is FALSE. If TRUE, everything including the row and column names are quoted so that you can copy it to Excel easily, Default: F
- **smd**: If TRUE, as in the default and there are more than two groups, standardized mean differences for all pairwise comparisons are calculated, Default: F
- **Labels**: Use Label, Default: F
- **exact**: A character vector to specify the variables for which the p-values should be those of exact tests. By default all p-values are from large sample approximation tests (chisq.test)., Default: NULL
- **nonnormal**: A character vector to specify the variables for which the p-values should be those of nonparametric tests. By default all p-values are from normal assumption-based tests (oneway.test)., Default: NULL
- **catDigits**: Number of digits to print for proportions. Default: 1
- **contDigits**: Number of digits to print for continuous variables. Default 2.
- **pDigits**: Number of digits to print for p-values (also used for standardized mean differences), Default: 3
- **labeldata**: labeldata to use, Default: NULL
- **psub**: show sub-group p-values, Default: F
- **minMax**: Whether to use [min,max] instead of [p25,p75] for nonnormal variables. The default is FALSE.
- **showpm**: Logical, show normal distributed continuous variables as Mean ± SD. Default: T
- **addOverall**: (optional, only used if strata are supplied) Adds an overall column to the table. Smd and p-value calculations are performed using only the stratified columns. Default: F
- **normalityTest**: Logical, perform the Shapiro test for all variables. Default: F

**Details**

DETAILS

**Value**

A matrix object containing what you see is also invisibly returned. This can be assinged a name and exported via write.csv.

**Examples**

```r
library(survival)
CreateTableOneJS(vars = names(lung), strata = "sex", data = lung)
```
extractAIC.coxme: Extract AIC from coxme.object

Description

Extract AIC from coxme.object

Usage

## S3 method for class 'coxme'
extractAIC(fit, scale = NULL, k = 2, ...)

Arguments

fit    coxme.object
scale  NULL
k      numeric specifying the 'weight' of the equivalent degrees of freedom (=: edf) part in the AIC formula.
...    further arguments (currently unused in base R).

Details

DETAILS

Value

AIC(Integrated, Penalized)

Examples

library(coxme)
fit <- coxme(Surv(time, status) ~ ph.ecog + age + (1 | inst), lung)
extractAIC(fit)

geeExp: transform the unit of coefficients (internal function)

Description

Transform the unit of coefficients to "Coeff", "OR" or "RR"

Usage

ggeeExp(gee.coef, family = "binomial", dec)
Arguments

- **gee.coef**: geeUni object.
- **family**: Family: "gaussian", "binomial", "poisson", "quasipoisson", etc..., Default: 'binomial'
- **dec**: Decimal point

Details

DETAILS

Value

The transformed coefficients

Examples

```r
library(geepack)
data(dietox)
dietox$Cu <- as.factor(dietox$Cu)
gee.uni <- geeUni("Weight", c("Time", "Cu"),
    data = dietox, id.vec = dietox$Pig,
    family = "gaussian", cor.type = "exchangeable"
)
gee.exp <- geeExp(gee.uni, "binomial", 2)
```

Description

Make gee results from "geeglm" object

Usage

```r
geeglm.display(geeglm.obj, decimal = 2)
```

Arguments

- **geeglm.obj**: "geeglm" object
- **decimal**: Decimal, Default: 2

Details

DETAILS

Value

List: caption, main table, metrics table
See Also
data.table-package complete.cases

Examples

library(geepack)
data(dietox)
dietox$Cu <- as.factor(dietox$Cu)
gee01 <- geeglm(Weight ~ Time + Cu,
  id = Pig, data = dietox,
  family = gaussian, corstr = "ex"
)
geeglm.display(gee01)

Description

Extract the coefficients of univariate gee using geeglm function (geepack package).

Usage

geeUni(y, x, data, id.vec, family, cor.type = "exchangeable")

Arguments

y Dependant variable
x Independent variable
data Data
id.vec Vector of id (should be ordered)
family Family: "gaussian", "binomial", "poisson", "quasipoisson", etc...
cor.type Correlation structure, Default: 'exchangeable'

Details

DETAILS

Value

coefficient, standard error, p-value
Examples

```
library(geepack)
data(dietox)
dietox$Cu <- as.factor(dietox$Cu)
gee.uni <- geeUni("Weight", "Time",
                   data = dietox, id.vec = dietox$Pig,
                   family = "gaussian", cor.type = "exchangeable"
)
```

Description

Show summary table of glm object (regression, logistic).

Usage

```
glmshow.display(glm.object, decimal = 2)
```

Arguments

- `glm.object` : glm.object
- `decimal` : digits, Default: 2

Details

DETAILS

Value

- table

See Also

- `glm`

Examples

```
glmshow.display(glm(mpg ~ wt + qsec, data = mtcars))
```
LabelepiDisplay: Apply label information to epiDisplay object using label data

Description
Apply label information to epiDisplay.object using label data

Usage
LabelepiDisplay(epiDisplay.obj, label = F, ref)

Arguments
- epiDisplay.obj: epiDisplay.object or glmshow.object
- label: Apply label information, Default: F
- ref: Label data made by mk.lev function

Details

Value
epiDisplay.object with label information

Examples
fit <- glm(Sepal.Length ~ Sepal.Width + Species, data = iris)
fit.table <- glmshow.display(fit)
iris.label <- mk.lev(iris)
LabelepiDisplay(fit.table, label = TRUE, ref = iris.label)

LabeljsCox: Apply label information to cox2.display object using label data

Description
Apply label information to cox2.display object using label data

Usage
LabeljsCox(obj, ref)
LabeljsGeeglm  

Arguments  
  
  obj  
  cox2.display object  
  
  ref  
  Label data made by mk.lev function  
  
Details  
  
  DETAILS  
  
Value  
  
  cox2.display object with label information  
  
Examples  
  
  library(survival)
  fit <- coxph(Surv(time, status) ~ sex + ph.ecog + ph.karno + cluster(inst),
              data = lung, model = TRUE)
  fit.table <- cox2.display(fit)
  lung.label <- mk.lev(lung)
  LabeljsCox(fit.table, ref = lung.label)

LabeljsGeeglm  

LabeljsGeeglm: Apply label information to geeglm.display object using label data  

Description  
  
  Apply label information to geeglm.display object using label data  

Usage  
  
  LabeljsGeeglm(obj, ref)  

Arguments  
  
  obj  
  geeglm.display object  
  
  ref  
  Label data made by mk.lev function  
  
Details  
  
  DETAILS  
  
Value  
  
  geeglm.display object with label information
Examples

```r
library(geepack)
library(jstable)
data(dietox)
dietox$Cu <- as.factor(dietox$Cu)
gee01 <- geeglm(Weight ~ Time + Cu,
    id = Pig, data = dietox,
    family = gaussian, corstr = "ex"
)
g1 <- geeglm.display(gee01)
LabeljsGeeglm(g1, ref = mk.lev(dietox))
```

---

**LabeljsMetric**

Apply label information to jstable metric object using label data

**Description**

Apply label information to metric object of jstable using label data

**Usage**

```r
LabeljsMetric(obj.metric, ref)
```

**Arguments**

- `obj.metric`: metric of `lmer.display`, `coxme.display`
- `ref`: Label data made by `mk.lev` function

**Details**

DETAILS

**Value**

metric of `lmer.display`, `coxme.display` with label information

**Examples**

```r
library(coxme)
fit <- coxme(Surv(time, status) ~ sex + ph.ecog + ph.karno + (1 | inst) + (1 | sex), lung)
fit.table <- coxme.display(fit)
lung.label <- mk.lev(lung)
LabeljsTable(fit.table$table, ref = lung.label)
LabeljsRanef(fit.table$ranef, ref = lung.label)
LabeljsMetric(fit.table$metric, ref = lung.label)
```
**LabeljsMixed**

Apply label information to jstable object using label data

**Description**

Apply label information to object of jstable using label data

**Usage**

LabeljsMixed(obj, ref)

**Arguments**

- **obj**
  - lmer.display, coxme.display
- **ref**
  - Label data made by mk.lev function

**Details**

DETAILS

**Value**

lmer.display, coxme.display with label information

**Examples**

```r
library(coxme)
fit <- coxme(Surv(time, status) ~ sex + ph.ecog + ph.karno + (1 | inst) + (1 | sex), lung)
fit.table <- coxme.display(fit)
lung.label <- mk.lev(lung)
LabeljsMixed(fit.table, ref = lung.label)
```

---

**LabeljsRanef**

Apply label information to jstable random effect object using label data

**Description**

Apply label information to ranef object of jstable using label data

**Usage**

LabeljsRanef(obj.ranef, ref)
Arguments

obj.ranef    ranef of lmer.display, coxme.display, cox2.display
ref          Label data made by mk.lev function

Details

DETAILS

Value

ranef of lmer.display, coxme.display, cox2.display with label information

Examples

library(coxme)
fit <- coxme(Surv(time, status) ~ sex + ph.ecog + ph.karno + (1 | inst) + (1 | sex), lung)
fit.table <- coxme.display(fit)
lung.label <- mk.lev(lung)
LabeljsTable(fit.table$table, ref = lung.label)
LabeljsRanef(fit.table$ranef, ref = lung.label)

LabeljsTable

LabeljsTable: Apply label information to jstable object using label data

Description

Apply label information to table of geeglm.display, lmer.display, coxme.display using label data

Usage

LabeljsTable(obj.table, ref)

Arguments

obj.table    table of geeglm.display, lmer.display, coxme.display
ref          Label data made by mk.lev function

Details

DETAILS

Value

table of geeglm.display, lmer.display, coxme.display with label information
Examples

library(coxme)
fit <- coxme(Surv(time, status) ~ sex + ph.ecog + ph.karno + (1 | inst) + (1 | sex), lung)
fit.table <- coxme.display(fit)
lung.label <- mk.lev(lung)
LabeljsTable(fit.table$table, ref = lung.label)

lmer.display

lmer.display: table for "lmerMod" or "glmerMod" object (lme4 package)

Description

Make mixed effect model results from "lmerMod" or "glmerMod" object (lme4 package)

Usage

lmer.display(lmerMod.obj, dec = 2, ci.ranef = F)

Arguments

lmerMod.obj "lmerMod" or "glmerMod" object
dec Decimal, Default: 2
ci.ranef Show confidence interval of random effects?, Default: F

Details

DETAILS

Value

Table: fixed & random effect

Examples

library(geepack)
data(dietox)
dietox$Cu <- as.factor(dietox$Cu)
l1 <- lme4::lmer(Weight ~ Time + Cu + (1 | Pig) + (1 | Evit), data = dietox)
lmer.display(l1)
**lmerExp**

*lmerExp: transform the unit of coefficients (internal function)*

**Description**

Transform the unit of coefficients to "Coeff", "OR" or "RR".

**Usage**

```r
lmerExp(lmer.coef, family = "binomial", dec)
```

**Arguments**

- `lmer.coef`: lmer coefficients.
- `family`: Family: "gaussian", "binomial", "poisson", "quasipoisson", etc..., Default: 'binomial'
- `dec`: Decimal point

**Details**

DETAILS

**Value**

The transformed coefficients

**Examples**

```r
# EXAMPLE1
```

**mk.lev**

*Export label and level: multiple variable*

**Description**

Export label and level: multiple variable

**Usage**

```r
mk.lev(data)
```

**Arguments**

- `data`: data
**mk.lev.var**

**Details**

DETAILS

**Value**

default label and level data

**Examples**

```r
mk.lev(iris)
```

---

**mk.lev.var**  
*Export label and level: one variable*

**Description**

Export label and level: one variable

**Usage**

```r
mk.lev.var(data, vname)
```

**Arguments**

- `data`  
  data
- `vname`  
  variable to export label and level

**Details**

DETAILS

**Value**

if continuous variable - (label, NA), categorical variable - (label, level)

**Examples**

```r
lapply(names(iris), function(x) {
  jstable::mk.lev.var(iris, x)
})
```
Description

DATASET_DESCRIPTION

Usage

mort

Format

A data frame with 17562 rows and 24 variables:

code  integer COLUMN_DESCRIPTION
name  character COLUMN_DESCRIPTION
yy    integer COLUMN_DESCRIPTION
mm    integer COLUMN_DESCRIPTION
dd    integer COLUMN_DESCRIPTION
date  character COLUMN_DESCRIPTION
nonacc integer COLUMN_DESCRIPTION
cardio integer COLUMN_DESCRIPTION
respir integer COLUMN_DESCRIPTION
influenza integer COLUMN_DESCRIPTION
meanpm10 double COLUMN_DESCRIPTION
meanso2 double COLUMN_DESCRIPTION
meanno2 double COLUMN_DESCRIPTION
meanco double COLUMN_DESCRIPTION
maxco double COLUMN_DESCRIPTION
maxo3 double COLUMN_DESCRIPTION
meantemp double COLUMN_DESCRIPTION
maxtemp double COLUMN_DESCRIPTION
mintemp double COLUMN_DESCRIPTION
meanhumi double COLUMN_DESCRIPTION
meanpress double COLUMN_DESCRIPTION
season integer COLUMN_DESCRIPTION
dow   integer COLUMN_DESCRIPTION
sn    integer COLUMN_DESCRIPTION

Details

DETAILS
**opt.data**

**datatable option for data (DT package)**

**Description**

DT::datatable option for data

**Usage**

`opt.data(fname)`

**Arguments**

- `fname` : File name to download

**Details**

DETAILS

**Value**

datatable option object

**Examples**

`opt.data("mtcars")`

---

**opt.roc**

**datatable option for ROC result (DT package)**

**Description**

DT::datatable option for ROC result

**Usage**

`opt.roc(fname)`

**Arguments**

- `fname` : File name to download

**Details**

DETAILS
Value

datatable option object

Examples

options <- opt.roc("mtcars")

Description

Simple download DT::datatable option - No filter, No page

Usage

opt.simpledown(fname)

Arguments

fname File name to download

Details

DETAILS

Value

datatable option object

Examples

options <- opt.simpledown("mtcars")
Description

DT::datatable option for table 1

Usage

opt.tb1(fname)

Arguments

fname    File name to download

Details

DETAILS

Value

datatable option object

Examples

options <- opt.tb1("mtcars")

Description

DT::datatable option for regression table(DT package)

Usage

opt.tbreg(fname)

Arguments

fname    File name to download

Details

DETAILS
Value
datatable option object

Examples
options <- opt.tbreg("mtcars")

svycox.display

svycoxph.display: table for svycoxph.object in survey package.

Description
Table for complex design cox model.

Usage
svycox.display(svycoxph.obj, decimal = 2)

Arguments

svycoxph.obj svycoxph.object
decimal digit, Default: 2

Details
DETAILS

Value
List including table, metric, caption

See Also

svycoxph AIC

Examples
library(survival)
data(pbc)
pbc$sex <- factor(pbc$sex)
pbc$stage <- factor(pbc$stage)
pbc$randomized <- with(pbc, !is.na(trt) & trt > 0)
biasmodel <- glm(randomized ~ age * edema, data = pbc, family = binomial)
pbc$randprob <- fitted(biasmodel)

if (is.null(pbc$albumin)) pbc$albumin <- pbc$alb # pre2.9.0

dpbc <- survey::svydesign(
svyCreateTableOne2

\[
\begin{align*}
  \text{id} &= \sim 1, \text{prob} = \sim \text{randprob}, \text{strata} = \sim \text{edema}, \\
  \text{data} &= \text{subset(pbc, randomized)} \\
\end{align*}
\]

model <- survey::svycoxph(Surv(time, status > 0) ~ sex + protime + albumin + stage, 
                        design = dpbc
                      )
svycox.display(model)

svyCreateTableOne2

svyCreateTableOne2: Modified svyCreateTableOne function in tableone package

**Description**

Combine svyCreateTableOne & print function in tableone package

**Usage**

svyCreateTableOne2(
  data, 
  strata, 
  vars, 
  factorVars, 
  includeNA = F, 
  test = T, 
  showAllLevels = T, 
  printToggle = F, 
  quote = F, 
  smd = F, 
  nonnormal = NULL, 
  catDigits = 1, 
  contDigits = 2, 
  pDigits = 3, 
  Labels = F, 
  labeldata = NULL, 
  minMax = F, 
  showpm = T, 
  addOverall = F
)

**Arguments**

- **data** A data frame in which these variables exist. All variables (both vars and strata) must be in this data frame.
- **strata** Stratifying (grouping) variable name(s) given as a character vector. If omitted, the overall results are returned.
vars
Variables to be summarized given as a character vector. Factors are handled as categorical variables, whereas numeric variables are handled as continuous variables. If empty, all variables in the data frame specified in the data argument are used.

factorVars
Numerically coded variables that should be handled as categorical variables given as a character vector. Do not include factors, unless you need to relevel them by removing empty levels. If omitted, only factors are considered categorical variables. The variables specified here must also be specified in the vars argument.

includeNA
If TRUE, NA is handled as a regular factor level rather than missing. NA is shown as the last factor level in the table. Only effective for categorical variables., Default: F

test
If TRUE, as in the default and there are more than two groups, groupwise comparisons are performed, Default: T

showAllLevels
Whether to show all levels. FALSE by default, i.e., for 2-level categorical variables, only the higher level is shown to avoid redundant information., Default: T

printToggle
Whether to print the output. If FALSE, no output is created, and a matrix is invisibly returned., Default: F

quote
Whether to show everything in quotes. The default is FALSE. If TRUE, everything including the row and column names are quoted so that you can copy it to Excel easily, Default: F

smd
If TRUE, as in the default and there are more than two groups, standardized mean differences for all pairwise comparisons are calculated, Default: F

nonnormal
A character vector to specify the variables for which the p-values should be those of nonparametric tests. By default all p-values are from normal assumption-based tests (oneway.test), Default: NULL

catDigits
Number of digits to print for proportions., Default: 1

contDigits
Number of digits to print for continuous variables. Default 2.

pDigits
Number of digits to print for p-values (also used for standardized mean differences), Default: 3

Labels
Use Label, Default: F

labeldata
labeldata to use, Default: NULL

minMax
Whether to use [min,max] instead of [p25,p75] for nonnormal variables. The default is FALSE.

showpm
Logical, show normal distributed continuous variables as Mean ± SD. Default: T

addOverall
(optional, only used if strata are supplied) Adds an overall column to the table. Smd and p-value calculations are performed using only the stratified columns. Default: F

Details

DETAILS
Value

A matrix object containing what you see is also invisibly returned. This can be assinged a name and exported via write.csv.

Examples

```r
library(survey)
data(nhanes)
nhanes$SDMVPSU <- as.factor(nhanes$SDMVPSU)
nhanesSvy <- svydesign(
  ids = ~SDMVPSU, strata = ~SDMVSTR, weights = ~WTMEC2YR, 
  nest = TRUE, data = nhanes
)
svyCreateTableOne2( 
  vars = c("HI_CHOL", "race", "agecat", "RIAGENDR"), 
  strata = "RIAGENDR", data = nhanesSvy, 
  factorVars = c("HI_CHOL", "race", "RIAGENDR")
)
```

Description

Combine svyCreateTableOne & print function in tableone package

Usage

```r
svyCreateTableOneJS(
  vars, 
  strata = NULL, 
  strata2 = NULL, 
  data, 
  factorVars = NULL, 
  includeNA = F, 
  test = T, 
  showAllLevels = T, 
  printToggle = F, 
  quote = F, 
  smd = F, 
  Labels = F, 
  nonnormal = NULL, 
  catDigits = 1, 
  contDigits = 2, 
  pDigits = 3, 
  labeldata = NULL, 
  psub = T,
)```
`svyCreateTableOneJS`

```r
minMax = F,
showpm = T,
addOverall = F
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vars</code></td>
<td>Variables to be summarized given as a character vector. Factors are handled as categorical variables, whereas numeric variables are handled as continuous variables. If empty, all variables in the data frame specified in the <code>data</code> argument are used.</td>
</tr>
<tr>
<td><code>strata</code></td>
<td>Stratifying grouping variable name(s) given as a character vector. If omitted, the overall results are returned.</td>
</tr>
<tr>
<td><code>strata2</code></td>
<td>Stratifying 2nd grouping variable name(s) given as a character vector. If omitted, the 1 group results are returned.</td>
</tr>
<tr>
<td><code>data</code></td>
<td>A data frame in which these variables exist. All variables (both <code>vars</code> and <code>strata</code>) must be in this data frame.</td>
</tr>
<tr>
<td><code>factorVars</code></td>
<td>Numerically coded variables that should be handled as categorical variables given as a character vector. Do not include factors, unless you need to relevel them by removing empty levels. If omitted, only factors are considered categorical variables. The variables specified here must also be specified in the <code>vars</code> argument.</td>
</tr>
<tr>
<td><code>includeNA</code></td>
<td>If TRUE, NA is handled as a regular factor level rather than missing. NA is shown as the last factor level in the table. Only effective for categorical variables., Default: F</td>
</tr>
<tr>
<td><code>test</code></td>
<td>If TRUE, as in the default and there are more than two groups, groupwise comparisons are performed, Default: T</td>
</tr>
<tr>
<td><code>showAllLevels</code></td>
<td>Whether to show all levels. FALSE by default, i.e., for 2-level categorical variables, only the higher level is shown to avoid redundant information., Default: T</td>
</tr>
<tr>
<td><code>printToggle</code></td>
<td>Whether to print the output. If FALSE, no output is created, and a matrix is invisibly returned., Default: F</td>
</tr>
<tr>
<td><code>quote</code></td>
<td>Whether to show everything in quotes. The default is FALSE. If TRUE, everything including the row and column names are quoted so that you can copy it to Excel easily, Default: F</td>
</tr>
<tr>
<td><code>smd</code></td>
<td>If TRUE, as in the default and there are more than two groups, standardized mean differences for all pairwise comparisons are calculated, Default: F</td>
</tr>
<tr>
<td><code>Labels</code></td>
<td>Use Label, Default: F</td>
</tr>
<tr>
<td><code>nonnormal</code></td>
<td>A character vector to specify the variables for which the p-values should be those of nonparametric tests. By default all p-values are from normal assumption-based tests (oneway.test), Default: NULL</td>
</tr>
<tr>
<td><code>catDigits</code></td>
<td>Number of digits to print for proportions., Default: 1</td>
</tr>
<tr>
<td><code>contDigits</code></td>
<td>Number of digits to print for continuous variables. Default 2.</td>
</tr>
<tr>
<td><code>pDigits</code></td>
<td>Number of digits to print for p-values (also used for standardized mean differences), Default: 3</td>
</tr>
</tbody>
</table>
svyregress.display

labeldata  
labeldata to use, Default: NULL

psub  
show sub-group p-values, Default: F

minMax  
Whether to use [min, max] instead of [p25, p75] for nonnormal variables. The default is FALSE.

showpm  
Logical, show normal distributed continuous variables as Mean ± SD. Default: T

addOverall  
(optional, only used if strata are supplied) Adds an overall column to the table. Smd and p-value calculations are performed using only the stratified columns. Default: F

Details

DETAILS

Value

A matrix object containing what you see is also invisibly returned. This can be assigned a name and exported via write.csv.

Examples

```r
library(survey)
data(nhanes)
nhanes$SDMVPSU <- as.factor(nhanes$SDMVPSU)
nhanesSvy <- svydesign(
  ids = ~SDMVPSU, strata = ~SDMVSTRA, weights = ~WTMEC2YR,
  nest = TRUE, data = nhanes
)
svyCreateTableOneJS(
  vars = c("HI_CHOL", "race", "agecat", "RIAGENDR"),
  strata = "RIAGENDR", data = nhanesSvy,
  factorVars = c("HI_CHOL", "race", "RIAGENDR")
)
```

svyregress.display  

svyregress.display: table for svyglm.object

Description

table for svyglm.object (survey package).

Usage

```r
svyregress.display(svyglm.obj, decimal = 2)
```

Arguments

svyglm.obj  
svyglm.object

decimal  
digit, Default: 2
TableSubgroupCox

Details

DETAILS

Value

table

Examples

library(survey)
data(api)
apistrat$tt <- c(rep(1, 20), rep(0, nrow(apistrat) - 20))
dstrat <- svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat, fpc = ~fpc)
ds <- svyglm(api00 ~ ell + meals + cname + mobility, design = dstrat)
ds2 <- svyglm(tt ~ ell + meals + cname + mobility, design = dstrat, family = quasibinomial())
svyregress.display(ds)
svyregress.display(ds2)

TableSubgroupCox

TableSubgroupCox: Sub-group analysis table for Cox/svycox model.

Description

Sub-group analysis table for Cox/svycox model.

Usage

TableSubgroupCox(
  formula,
  var_subgroup = NULL,
  var_cov = NULL,
  data,
  time_eventrate = 3 * 365,
  decimal.hr = 2,
  decimal.percent = 1,
  decimal.pvalue = 3,
  cluster = NULL
)

Arguments

formula formula with survival analysis.
var_subgroup 1 sub-group variable for analysis, Default: NULL
var_cov Variables for additional adjust, Default: NULL
data Data or svydesign in survey package.
time_eventrate Time for kaplan-meier based event rate calculation, Default = 365 * 3
TableSubgroupCox

- decimal.hr: Decimal for hazard ratio, Default: 2
- decimal.percent: Decimal for percent, Default: 1
- decimal.pvalue: Decimal for pvalue, Default: 3
- cluster: Cluster variable for coxph, Default: NULL

Details

This result is used to make forestplot.

Value

Sub-group analysis table.

See Also

- safely
- map
- map2
- coxph
- svycoxph
- confint

Examples

```r
library(survival)
library(dplyr)
lung %>%
  mutate(
    status = as.integer(status == 1),
    sex = factor(sex),
    kk = factor(as.integer(pat.karno >= 70))
  ) -> lung
TableSubgroupCox(Surv(time, status) ~ sex, data = lung, time_eventrate = 100)
TableSubgroupCox(Surv(time, status) ~ sex, var_subgroup = "kk", data = lung, time_eventrate = 100)
```

## survey design

```r
library(survey)
data.design <- svydesign(id = ~1, data = lung)
TableSubgroupCox(Surv(time, status) ~ sex, data = data.design, time_eventrate = 100)
TableSubgroupCox(Surv(time, status) ~ sex, var_subgroup = "kk", data = data.design, time_eventrate = 100)
```
TableSubgroupGLM

TableSubgroupGLM: Sub-group analysis table for GLM.

Description

Sub-group analysis table for GLM.

Usage

TableSubgroupGLM(
  formula,
  var_subgroup = NULL,
  var_cov = NULL,
  data,
  family = "binomial",
  decimal.estimate = 2,
  decimal.percent = 1,
  decimal.pvalue = 3
)

Arguments

formula formula with survival analysis.
var_subgroup 1 sub-group variable for analysis, Default: NULL
var_cov Variables for additional adjust, Default: NULL
data Data or svydesign in survey package.
family family, "gaussian" or "binomial" or 'poisson' or 'quasipoisson'
decimal.estimate Decimal for estimate, Default: 2
decimal.percent Decimal for percent, Default: 1
decimal.pvalue Decimal for pvalue, Default: 3

Details

This result is used to make forestplot.

Value

Sub-group analysis table.

See Also

safely, map, map2, glm, svyglm
Examples

```r
library(survival)
library(dplyr)
lung %>%
  mutate(
    status = as.integer(status == 1),
    sex = factor(sex),
    kk = factor(as.integer(pat.karno >= 70))
  ) -> lung
TableSubgroupGLM(status ~ sex, data = lung, family = "binomial")
TableSubgroupGLM(status ~ sex, var_subgroup = "kk", data = lung, family = "binomial")

## survey design
library(survey)
data.design <- svydesign(id = ~1, data = lung)
TableSubgroupGLM(status ~ sex, data = data.design, family = "binomial")
TableSubgroupGLM(status ~ sex, var_subgroup = "kk", data = data.design, family = "binomial")
```

TableSubgroupMultiCox: Multiple sub-group analysis table for Cox/svycox model.

**Description**

Multiple sub-group analysis table for Cox/svycox model.

**Usage**

```r
TableSubgroupMultiCox(
  formula,
  var_subgroups = NULL,
  var_cov = NULL,
  data,
  time_eventrate = 3 * 365,
  decimal.hr = 2,
  decimal.percent = 1,
  decimal.pvalue = 3,
  line = F,
  cluster = NULL
)
```

**Arguments**

- `formula`: formula with survival analysis.
- `var_subgroups`: Multiple sub-group variables for analysis, Default: NULL
- `var_cov`: Variables for additional adjust, Default: NULL
- `data`: Data or svydesign in survey package.
TableSubgroupMultiCox

- `time_eventrate` Time for Kaplan-Meier based event rate calculation, Default = 365 * 3
- `decimal.hr` Decimal for hazard ratio, Default: 2
- `decimal.percent` Decimal for percent, Default: 1
- `decimal.pvalue` Decimal for pvalue, Default: 3
- `line` Include new-line between sub-group variables, Default: F
- `cluster` Cluster variable for coxph, Default: NULL

**Details**

This result is used to make forestplot.

**Value**

Multiple sub-group analysis table.

**See Also**

- `map`
- `bind`

**Examples**

```r
library(survival)
library(dplyr)
lung <-
  %>%
  mutate(
    status = as.integer(status == 1),
    sex = factor(sex),
    kk = factor(as.integer(pat.karno >= 70)),
    kk1 = factor(as.integer(pat.karno >= 60))
  )
TableSubgroupMultiCox(Surv(time, status) ~ sex,
  var_subgroups = c("kk", "kk1"),
  data = lung, time_eventrate = 100, line = TRUE)

## survey design
library(survey)
data.design <- svydesign(id = ~1, data = lung)
TableSubgroupMultiCox(Surv(time, status) ~ sex,
  var_subgroups = c("kk", "kk1"),
  data = data.design, time_eventrate = 100)
```
TableSubgroupMultiGLM

Description

Multiple sub-group analysis table for GLM.

Usage

TableSubgroupMultiGLM(
  formula,
  var_subgroups = NULL,
  var_cov = NULL,
  data,
  family = "binomial",
  decimal.estimate = 2,
  decimal.percent = 1,
  decimal.pvalue = 3,
  line = F
)

Arguments

formula  formula with survival analysis.
var_subgroups Multiple sub-group variables for analysis, Default: NULL
var_cov Variables for additional adjust, Default: NULL
data  Data or svydesign in survey package.
family  family, "gaussian" or "binomial" or 'poisson' or 'quasipoisson'
decimal.estimate  Decimal for estimate, Default: 2
decimal.percent  Decimal for percent, Default: 1
decimal.pvalue  Decimal for pvalue, Default: 3
line Include new-line between sub-group variables, Default: F

Details

This result is used to make forestplot.

Value

Multiple sub-group analysis table.
TableSubgroupMultiGLM

See Also

map, bind

Examples

library(survival)
library(dplyr)
lung %>%
  mutate(
    status = as.integer(status == 1),
    sex = factor(sex),
    kk = factor(as.integer(pat.karno >= 70)),
    kk1 = factor(as.integer(pat.karno >= 60))
  ) -> lung
TableSubgroupMultiGLM(status ~ sex,
  var_subgroups = c("kk", "kk1"),
  data = lung, line = TRUE, family = "binomial"
)

## survey design
library(survey)
data.design <- svydesign(id = ~1, data = lung)
TableSubgroupMultiGLM(status ~ sex,
  var_subgroups = c("kk", "kk1"),
  data = data.design, family = "binomial"
)
Index

* datasets
  mort, 24

AIC, 28

bind, 38, 40

coeffNA, 3
complete.cases, 14
confint, 35
cox2.display, 3
coxExp, 4
coxme.display, 5
coxmeTable, 5
coxph, 35
CreateTableOne2, 6
CreateTableOneJS, 9

ejectAIC.coxme, 12

ggeeExp, 12
ggeglm.display, 13
geeUni, 14
glm, 15, 36
glmshow.display, 15

LabelepiDisplay, 16
LabeljsCox, 16
LabeljsGeeglm, 17
LabeljsMetric, 18
LabeljsMixed, 19
LabeljsRanef, 19
LabeljsTable, 20
lmer.display, 21
lmerExp, 22

map, 35, 36, 38, 40
map2, 35, 36
mk.lev, 22
mk.lev.var, 23
mort, 24

opt.data, 25
opt.roc, 25
opt.simplesdown, 26
opt.tbl, 27
opt.tbreg, 27

safely, 35, 36
svycox.display, 28
svycoxph, 28, 35
svyCreateTableOne2, 29
svyCreateTableOneJS, 31
svyglm, 36
svyregress.display, 33

TableSubgroupCox, 34
TableSubgroupGLM, 36
TableSubgroupMultiCox, 37
TableSubgroupMultiGLM, 39