Package ‘jstable’

January 9, 2020

Title Create Tables from Different Types of Regression

Version 0.9.2

Date 2019-12-24

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

LazyData true

RoxygenNote 7.0.1

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

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

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

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

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

Maintainer Jinseob Kim <jinseob2kim@gmail.com>

Repository CRAN

Date/Publication 2020-01-08 23:01:09 UTC
R topics documented:

- coefNA ........................................... 2
- cox2.display .................................... 3
- coxExp ........................................... 4
- coxme.display ................................... 5
- coxmeTable ....................................... 5
- CreateTableOne2 ................................. 6
- CreateTableOneJS ................................. 9
- extractAIC.coxme ................................. 11
- geeExp ........................................... 12
- geeglm.display ................................... 13
- geeUni ........................................... 14
- glmshow.display .................................. 15
- LabelepiDisplay ................................ 15
- LabeljsCox ....................................... 16
- LabeljsGeeglm ................................... 17
- LabeljsMetric .................................... 18
- LabeljsMixed .................................... 18
- LabeljsRanef ..................................... 19
- LabeljsTable ..................................... 20
- lmer.display ..................................... 21
- lmerExp .......................................... 21
- mk.lev ........................................... 22
- mk.lev.var ....................................... 23
- opt.data ......................................... 23
- opt roc ........................................... 24
- opt.simplesdown ................................. 24
- opt.tb1 .......................................... 25
- opt.tbreg ........................................ 25
- svycox.display .................................. 27
- svyCreateTableOne2 ............................. 28
- svyCreateTableOneJS ............................ 30
- svyregress.display .............................. 32
- TableSubgroupCox ............................... 33
- TableSubgroupMultiCox ......................... 34

Index 36

coefNA

coefNA: make coefficient table with NA

Description

Make coefficient table with NA
Usage

coeNA(model)

Arguments

model glm object (gaussian or binomial)

Details

DETAILS

Value

coefficient table with NA

Examples

coeNA(glm(mpg ~ wt + qsec, data = mtcars))

Description

Table for coxph.object with model option: TRUE - allow "frailty" or "cluster" model

Usage

cox2.display(cox.obj.withmodel, dec = 2)

Arguments

cox.obj.withmodel coxph.object with model option: TRUE

dec Decimal point, Default: 2

Details

GEE like - cluster, Mixed effect model like - frailty

Value

Table, cluster/frailty info, metrics, caption
Examples

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)

---

coxExp

**coxExp**: transform the unit of coefficients in cox model (internal function)

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 transformed coefficients (95

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**

```r
coxme.display(coxme.obj, dec = 2)
```

**Arguments**

- `coxme.obj`: coxme.object
- `dec`: Decimal point, Default: 2

**Details**

**Value**

Fixed effect table, random effect, metrics, caption

**Examples**

```r
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**

```r
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

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,
pDigits = 3,
labeldata = NULL,
minMax = 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
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 (Mann-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.

### 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)
CreateTableOne2(vars = names(lung), strata = "sex", data = lung)
```
CreateTableOneJS: Modified CreateTableOne function in tableone package

Description

Combine CreateTableOne & print function in tableone package

Usage

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
)

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.
<table>
<thead>
<tr>
<th><strong>strata</strong></th>
<th>Stratifying grouping variable name(s) given as a character vector. If omitted, the overall results are returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>strata2</strong></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><strong>data</strong></td>
<td>A data frame in which these variables exist. All variables (both vars and strata) must be in this data frame.</td>
</tr>
<tr>
<td><strong>factorVars</strong></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 vars argument.</td>
</tr>
<tr>
<td><strong>includeNA</strong></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><strong>test</strong></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><strong>testApprox</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>argsApprox</strong></td>
<td>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)</td>
</tr>
<tr>
<td><strong>testExact</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>argsExact</strong></td>
<td>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)</td>
</tr>
<tr>
<td><strong>testNormal</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>argsNormal</strong></td>
<td>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)</td>
</tr>
<tr>
<td><strong>testNonNormal</strong></td>
<td>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 (Mann-Whitney U test) when there are only two groups, Default: kruskal.test</td>
</tr>
<tr>
<td><strong>argsNonNormal</strong></td>
<td>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)</td>
</tr>
<tr>
<td><strong>showAllLevels</strong></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><strong>printToggle</strong></td>
<td>Whether to print the output. If FALSE, no output is created, and a matrix is invisibly returned., Default: F</td>
</tr>
</tbody>
</table>
extractAIC.coxme

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

ccontDigits 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.

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

library(survival)
CreateTableOneJS(vars = names(lung), strata = "sex", data = lung)

extractAIC.coxme

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(Integreted, Penalized)

Examples

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

---

geeExp

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

Description

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

Usage

```r
geeExp(gee.coef, family = "binomial", dec)
```

Arguments

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

Details

DETAILS

Value

The transformed coefficients(95
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

```r
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)
```
The coefficient of univariate gee (internal function)

Description

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

Usage

```r
ggeeUni(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

```r
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")
```
glmshow.display

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

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**

DETAILS

**Value**

epiDisplay.object with label information

**Examples**

```r
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**

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)`

**Arguments**

- `obj` cox2.display object
- `ref` Label data made by mk.lev function

**Details**

DETAILS

**Value**

cox2.display object with label information
Examples

```r
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)
```

---

**Description**

Apply label information to geeglm.display object using label data

**Usage**

```r
LabeljsGeeglm(obj, ref)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>geeglm.display object</td>
</tr>
<tr>
<td>ref</td>
<td>Label data made by mk.lev function</td>
</tr>
</tbody>
</table>

**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

Description

Apply label information to metric object of jstable using label data

Usage

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

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

Description

Apply label information to object of jstable using label data

Usage

LabeljsMixed(obj, ref)
**LabeljsRanef**

**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**

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

```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)
```

---

**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**

```r
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**

```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)
```
lmer.display

**Description**

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

**Usage**

```r
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**

```r
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

**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

mk.lev(iris)
mk.lev.var

Export label and level: one variable

Description

Export label and level: one variable

Usage

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

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

opt.data
datable 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

datable 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")
**opt.simplesdown**

datable option for simple download\((DT\text{ package})\)

---

**Description**

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

**Usage**

opt.simplesdown(fname)

**Arguments**

- fname: File name to download

**Details**

DETAILS

**Value**

datable option object

**Examples**

```r
options = opt.simplesdown("mtcars")
```

---

**opt.tb1**

datable option for table 1\((DT\text{ package})\)

---

**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")

---

**Option: opt.tbreg**

datatable option for regression table (DT package)

Description

DT::datatable option for glm, gee(geepack package), lmer/glmer(lme4 package)

Usage

opt.tbreg(fname)

Arguments

fname

File name to download

Details

DETAILS

Value

datatable option object

Examples

options = opt.tbreg("mtcars")
svycox.display

svycox.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(id=~1, prob=~randprob, strata=~edema, data=subset(pbc, randomized))
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  
)

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.
svyCreateTableOne2

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 \[\text{min},\text{max}\] instead of \[\text{p25},\text{p75}\] for nonnormal variables. The default is FALSE.

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)
```
svyCreateTableOneJS

svyCreateTableOneJS: Modified CreateTableOne function in tableone package

Description

Combine svyCreateTableOne & print function in tableone package

Usage

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,  
  minMax = 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

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

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.

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

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)

svyregress.display(svyglm.object)

Usage

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

Arguments

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

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)
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
)

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
decimal.hr  Decimal for hazard ratio, 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, coxph, svycoxph, confint
Examples

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
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)

TableSubgroupMultiCox 

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

Description

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

Usage

TableSubgroupMultiCox(
  formula,
  var_subgroups = NULL,
  var_cov = NULL,
  data,
  time_eventrate = 3 * 365,
  decimal.hr = 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.
- 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

Details
This result is used to make forestplot.

Value
Multiple sub-group analysis table.

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
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)
Index

AIC, 27
bind, 35

coefNA, 2
complete.cases, 13
confInt, 33
cox2.display, 3
coxExp, 4
coxme.display, 5
coxmeTable, 5
coxph, 33
CreateTableOne2, 6
CreateTableOneJS, 9

extractAIC.coxme, 11

geeExp, 12
ggeglm.display, 13
ggeUni, 14
glm, 15
glmshow.display, 15

LabelepiDisplay, 15
LabeljsCox, 16
LabeljsGeeGlm, 17
LabeljsMetric, 18
LabeljsMixed, 18
LabeljsRanef, 19
LabeljsTable, 20
lmer.display, 21
lmerExp, 21

map, 33, 35
map2, 33
mk.lev, 22
mk.lev.var, 23

opt.data, 23
opt.roc, 24
opt.simpledown, 25

opt.tb1, 25
opt.tbreg, 26

safety, 33
svycox.display, 27
svycoxph, 27, 33
svyCreateTableOne2, 28
svyCreateTableOneJS, 30
svyregress.display, 32

TableSubgroupCox, 33
TableSubgroupMultiCox, 34