The `coefNA` function is used to create a coefficient table with NA values. It takes a model as input.

**Description**

Make coefficient table with NA

**Usage**

```r
coefNA(model)
```
Arguments

model glm object (gaussian or binomial)

Details

Value

coefficient table with NA

Examples

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

cox2.display

cox2.display: table for coxph.object with model option: TRUE - allow "frailty" or "cluster" model

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 transforemed 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
coxmeTable

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

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,  
pDigits = 3,  
labeldata = NULL,  
minMax = F,  
showpm = T  
)
```

**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.
<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 data argument are used.</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 realign 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>testApprox</code></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><code>argsApprox</code></td>
<td>A named list of arguments passed to the function specified in <code>testApprox</code>. The default is list(correct = TRUE), which turns on the continuity correction for chisq.test, Default: list(correct = TRUE)</td>
</tr>
<tr>
<td><code>testExact</code></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><code>argsExact</code></td>
<td>A named list of arguments passed to the function specified in <code>testExact</code>. 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><code>testNormal</code></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><code>argsNormal</code></td>
<td>A named list of arguments passed to the function specified in <code>testNormal</code>. 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><code>testNonNormal</code></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><code>argsNonNormal</code></td>
<td>A named list of arguments passed to the function specified in <code>testNonNormal</code>. The default is list(NULL), which is just a placeholder., Default: list(NULL)</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>
</tbody>
</table>
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 \([\text{min}, \text{max}]\) instead of \([\text{p}25, \text{p}75]\) for nonnormal variables. The default is FALSE.

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

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)
CreateTableOne2(vars = names(lung), strata = "sex", data = lung)

CreateTableOneJS
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,
  labelldata = NULL,
  psub = T,
  minMax = F,
  showpm = T
)

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)

**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 subgroup 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

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)

descrition extractAIC.coxme: Extract AIC from coxme.object

Argument

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).
**geeExp**

Details

DETAILS

Value

AIC(Integrated, Penalized)

Examples

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

---

**geeExp**

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

Description

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

Usage

```r
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

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)
geeUni

geeUni: The coefficient of univariate gee (internal function)

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")
glmshow.display

glmshow.display: Show summary table of glm object.

Description

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

Usage

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

Arguments

glm.object

decimal
digits, Default: 2

Details

Value
table

See Also

glm

Examples

glmshow.display(glm(mpg ~ wt + qsec, data = mtcars))

LabelepiDisplay

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

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)
labepiDisplay(fit.table, label = TRUE, ref = iris.label)
```

Description

Apply label information to cox2.display object using label data

Usage

```r
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)
```

---

**LabeljsGeeglm**

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

Description

Apply label information to geeglm.display object using label data

Usage

```r
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

LabeljsMetric: Apply label information to jstable metric object using label data

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

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

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

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

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
# EXAMPLE 1

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

```r
opt.data("mtcars")
```

---

**opt.roc**  
**datatable option for ROC result** *(DT package)*

**Description**

DT::datatable option for ROC result

**Usage**

```r
opt.roc(fname)
```

**Arguments**

- `fname`  
  File name to download

**Details**

DETAILS

**Value**

datatable option object

**Examples**

```r
options <- opt.roc("mtcars")
```
**opt.simplesdown**

**datatable option for simple download (DT package)**

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

**Usage**
```
opt.simplesdown(fname)
```

**Arguments**
- `fname`: File name to download

**Details**
DETAILS

**Value**
datatable option object

**Examples**
```
options <- opt.simplesdown("mtcars")
```

---

**opt.tb1**

**datatable option for table 1 (DT 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.tbreg("mtcars")

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: 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
)

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 [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

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

library(survey)
data(nhanes)
nhanes$SDMVPSU <- as.factor(nhanes$SDMVPSU)
nhanesSvy <- svydesign(
  ids = ~SDMVPSU, strata = ~SDMVSTRA, weights = ~WTMEC2YR,
  nest = TRUE, data = nhanes)
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,
  showpm = T
)

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.
svyCreateTableOneJS

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.

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

Details

DETAILS
svyregress.display

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 = ~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: 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

Details

DETAILS

Value

- table
Examples

```r
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**

```r
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

```r
library(survival)
library(dplyr)
lung %>%
mutoe(
  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)
)
```

---

**TableSubgroupGLM**

Sub-group analysis table for GLM.

Description

Sub-group analysis table for GLM.

Usage

```r
TableSubgroupGLM(
  formula,
  var_subgroup = NULL,
  var_cov = NULL,
  data,
  family = "binomial",
  decimal_estimate = 2,
  decimal_percent = 1,
```
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"
- **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`, `map`, `map2`, `glm`, `svyglm`, `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
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

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

```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))
  ) -> 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
)
```

### Description

Multiple sub-group analysis table for GLM.

### Usage

```r
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"
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.

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

AIC, 27

bind, 36, 38

coeffNA, 2  
complete.cases, 13  
confint, 34, 35  
cox2.display, 3  
coxExp, 4  
coxme.display, 4  
coxmeTable, 5  
coxph, 34  
CreateTableOne2, 6  
CreateTableOneJS, 8

eextractAIC.coxme, 11

geeExp, 12  
geeglm.display, 13  
geeUni, 14  
glm, 15, 35  
glmshow.display, 15

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

map, 34–36, 38

map2, 34, 35  
mk.lev, 22  
mk.lev.var, 23

opt.data, 23  
opt.rc, 24  
opt.simplesdown, 25  

opt.tb1, 25  
opt.tbreg, 26

safely, 34, 35  
svycox.display, 27  
svycoxph, 27, 34  
svyCreateTableOne2, 28  
svyCreateTableOneJS, 30  
svyglm, 35  
svyregress.display, 32

TableSubgroupCox, 33  
TableSubgroupGLM, 34  
TableSubgroupMultiCox, 36  
TableSubgroupMultiGLM, 37