# Package 'jstable'

March 30, 2026

**Title** Create Tables from Different Types of Regression

**Version** 1.3.25

**Date** 2026-03-02

**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** 7.3.3

**Imports** geepack, lme4, stats, data.table, labelled, tableone, coxme (>= 2.2-22), survival (>= 3.0.0), survey, methods, dplyr, purrr, magrittr, tibble, rlang, car, lmerTest, nortest

**URL** <https://github.com/jinseob2kim/jstable>,

<https://jinseob2kim.github.io/jstable/>

**BugReports** <https://github.com/jinseob2kim/jstable/issues>

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**LazyData** true

**NeedsCompilation** no

**Author** Jinseob Kim [aut, cre] (ORCID: <https://orcid.org/0000-0002-9403-605X>),
Zarathu [cph, fnd],
Yoonkyoung Jeon [aut],
Jaehun Shon [aut],
Hyojong Myung [aut],
Hyungwoo Jo [aut],
Sungho Choi [aut],
Jaewoong Heo [aut],
Mingu Jee [aut],

Yujeong Yoon [aut],
Minhyuk Kim [aut],
Wonbin Hahn [aut],
Taehong Kim [aut]

**Maintainer** Jinseob Kim <jinseob2kim@gmail.com>

# Contents

---

ChangeSvyTable *ChangeSvyTable: Modify the number n*

---

#### Description

Replace the number of weights taken into account with the number of n in the original data

#### Usage

```
ChangeSvyTable(svy, ori)
```

#### Arguments

| | |
|---|---|
| svy | TableOne object that take weights into account |
| ori | TableOne object in the original data |

#### Details

DETAILS

#### Value

A matrix that replaces the n number of weights with the n number of the original data

#### Examples

```
# example code
library('survey'); library('tableone')
data(nhanes)
nhanes$SDMVPSU <- as.factor(nhanes$SDMVPSU)
nhanesSvy <- svydesign(
  ids = ~SDMVPSU, strata = ~SDMVSTRA, weights = ~WTMEC2YR,
  nest = TRUE, data = nhanes
)

res <- svyCreateTableOne(
  vars = c("HI_CHOL", "race", "agecat", "RIAGENDR"),
  strata = "RIAGENDR", data = nhanesSvy,
  factorVars = c("HI_CHOL", "race", "RIAGENDR")
)

ori <- CreateTableOne(
  vars = c("HI_CHOL", "race", "agecat", "RIAGENDR"),
```

```
  strata = "RIAGENDR", data = nhanes,
  factorVars = c("HI_CHOL", "race", "RIAGENDR")
)
ChangeSvyTable(res, ori)
```

---

| coefNA | *coefNA: make coefficient table with NA* |
|---|---|

---

### Description

Make coefficient table with NA

### Usage

```
coefNA(model)
```

### Arguments

model         glm object (gaussian or binomial)

### Details

DETAILS

### Value

coefficient table with NA

### Examples

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

---

| collapse_counts | *collapse_counts* |
|---|---|

---

### Description

#' When a grouping variable count_by has more than two levels, this utility collapses all of the level-specific columns

### Usage

```
collapse_counts(df, count_by)
```

## Arguments

| | |
|---|---|
| df | A data.frame produced by one of the TableSubgroup functions |
| count_by | A string giving the name of the variable whose per-level count columns should be collapsed. If NULL, df is returned unmodified. |

## Details

This helper is intended for internal use by TableSubgroupCox(), TableSubgroupMultiCox(), and their GLM equivalents. It simplifies reporting when you request counts stratified by a factor with more than two levels.

## Value

A data.frame identical to 'f except that: All Count(count_by=...) columns are removed. A single column Count(by count_by) remains, containing the first non-missing, non-empty value from the original per-level columns.

---

| count_event_by | *count_event_by: funciton to count event, subgroup number inside TableSubgroupCox, TableSubgroupMultiCox* |
|---|---|

---

## Description

Function to count event, subgroup number

## Usage

```
count_event_by(
  formula,
  data,
  count_by_var = NULL,
  var_subgroup = NULL,
  decimal.percent = 1,
  data_original = NULL,
  formula_original = NULL
)
```

## Arguments

| | |
|---|---|
| formula | formula with survival analysis |
| data | same data as in formula |
| count_by_var | variables to count subgroup for |
| var_subgroup | 1 sub-group variable for analysis, |
| decimal.percent | |
| | decimals to show percent of, Default: 1 |

data_original     Original data for competing risk analysis (before finegray transformation), De-
                  fault: NULL

formula_original

                  Original formula for competing risk analysis (before finegray transformation),
                  Default: NULL

## Details

This function is used inside TableSubgroupCox, TableSubgroupMultiCox for calculation. When
data_original and formula_original are provided, counts and percentages are calculated from the
original data instead of the finegray-transformed data, ensuring accurate event counts in competing
risk analysis.

## Value

Table with event, subgroup number

## See Also

[group_by](), [summarise](), [mutate](), [bind_rows](), [arrange]()

## Examples

```
## Not run:
if (interactive()) {

}

## End(Not run)
```

---

count_event_by_glm          *count_event_by_glm: function to count event rates or summary met-*
                            *rics for GLM analysis*

---

## Description

Count event and subgroup summary, supports automatic parsing of fixed-effect syntax '(1|sub-
group)' and multiple families.

## Usage

```
count_event_by_glm(
  formula,
  data,
  count_by_var = NULL,
  var_subgroup = NULL,
  decimal.percent = 1,
  family = "binomial"
)
```

## Arguments

| | |
|---|---|
| `formula` | formula with response (0/1 or count or continuous) and optional fixed-effect '(1\|subgroup)' term. |
| `data` | data.frame or survey.design |
| `count_by_var` | variable name to stratify by (string), default NULL |
| `var_subgroup` | subgroup variable name (string); parsed from formula if not provided, default NULL |
| `decimal.percent` | |
| | decimals for percent or mean/sd, default 1 |
| `family` | family type: "gaussian", "binomial", "poisson", or "quasipoisson" |

## Value

tibble with grouping columns and Metric column (rate or mean sd)

---

| | |
|---|---|
| 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,
  event_msm = NULL,
  pcut.univariate = NULL,
  data_for_univariate = NULL
)
```

## Arguments

| | |
|---|---|
| `cox.obj.withmodel` | |
| | coxph.object with model option: TRUE |
| `dec` | Decimal point, Default: 2 |
| `event_msm` | Character or numeric vector of destination states to keep (multi-state models only) |
| `pcut.univariate` | |
| | pcut.univariate, Default: NULL |
| `data_for_univariate` | |
| | data for univariate model, Default: NULL |

## 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

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

## Arguments

cox.coef      cox model coefficients

dec           Decimal point

## Details

DETAILS

## Value

The transforemed coefficients(95

## Examples

```
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, pcut.univariate = NULL)
```

### Arguments

| | |
|---|---|
| coxme.obj | coxme.object |
| dec | Decimal point, Default: 2 |
| pcut.univariate | |
| | pcut.univariate, Default: NULL |

### 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 | *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,
    addOverall = F,
    pairwise = F,
    pairwise.showtest = 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 (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 stratifed clolumns. Default: F |
| pairwise | (optional, only used if strata are supplied) When there are three or more strata, it displays the p-values for pairwise comparisons. Default: F |
| pairwise.showtest | |
| | (optional, only used if strata are supplied) When using pairwise comparison, it displays the test used to calculate p-values for pairwise comparisons. 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

```
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,
  labeldata = NULL,
  psub = T,
  minMax = F,
  showpm = T,
  addOverall = F,
  normalityTest = F,
  pairwise = F,
  pairwise.showtest = 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) |
| 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 stratifed clolumns. Default: F |
| normalityTest | Logical, perform the Shapiro test for all variables. Default: F |

pairwise                    (optional, only used if strata are supplied) When there are three or more strata, it
                            displays the p-values for pairwise comparisons. Default: F#' @return A matrix
                            object containing what you see is also invisibly returned. This can be assinged
                            a name and exported via write.csv.

pairwise.showtest
                            (optional, only used if strata are supplied) When using pairwise comparison, it
                            displays the test used to calculate p-values for pairwise comparisons. Default: F

## Details

DETAILS

## 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

```
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

```
geeExp(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 transforemed coefficients(95

## Examples

```
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)
```

---

geeglm.display                    *geeglm.display*

---

### Description

Make gee results from "geeglm" object

### Usage

```
geeglm.display(
  geeglm.obj,
  decimal = 2,
  pcut.univariate = NULL,
  data_for_univariate = NULL
)
```

### Arguments

geeglm.obj          "geeglm" object

decimal             Decimal, Default: 2

pcut.univariate

                    pcut.univariate, Default: NULL

data_for_univariate

                    data for univariate model, Default: NULL

### 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, pcut.univariate = NULL)
```

### Arguments

glm.object        glm.object

decimal           digits, Default: 2

pcut.univariate

                  pcut.univariate, Default: NULL

### Details

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

```
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**

```
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 us-*
                        *ing 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**

```
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)
```

## 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

```
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

```
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,
  pcut.univariate = NULL,
  data_for_univariate = NULL
)
```

## Arguments

| | |
|---|---|
| `lmerMod.obj` | "lmerMod" or "glmerMod" object |
| `dec` | Decimal, Default: 2 |
| `ci.ranef` | Show confidence interval of random effects?, Default: F |
| `pcut.univariate` | |
| | pcut.univariate, Default: NULL |
| `data_for_univariate` | |
| | data for univariate model, Default: NULL |

## 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 transforemed coefficients(95

### Examples

```
# EXAMPLE1
```

---

mk.lev                    *Export label and level: multiple variable*

---

### Description

Export label and level: multiple variable

### Usage

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

### Arguments

| | |
|---|---|
| data | data |

**Details**

DETAILS

**Value**

default label and level data

**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)
})
```

---

mort *DATASET_TITLE*

---

## Description

DATASET_DESCRIPTION

## Usage

```
mort
```

## Format

A data frame with 17562 rows and 24 variables:

ccode integer COLUMN_DESCRIPTION
cname 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 *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.simpledown *datable option for simple download(DT package)*

---

## 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")
```

---

opt.tb1 *datable 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.tb1("mtcars")
```

---

opt.tbreg *datable 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          *svycoxph.display: table for svycoxph.object in survey package.*

---

### Description

Table for complex design cox model.

### Usage

```
svycox.display(svycoxph.obj, decimal = 2, pcut.univariate = NULL)
```

### Arguments

svycoxph.obj     svycoxph.object

decimal          digit, Default: 2

pcut.univariate

           pcut.univariate, Default: NULL

### 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)
```

---

| svyCreateTableOne | *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,
  pairwise = F,
  pairwise.showtest = F,
  n_original = 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. |
| 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 stratifed clolumns. Default: F |
|---|---|
| pairwise | (optional, only used if strata are supplied) When there are three or more strata, it displays the p-values for pairwise comparisons. Default: F |
| pairwise.showtest | |
| | (optional, only used if strata are supplied) When using pairwise comparison, it displays the test used to calculate p-values for pairwise comparisons. Default: F |
| n_original | Replace the number of weighted n with the n in the original data. 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(survey)
data(nhanes)
nhanes$SDMVPSU <- as.factor(nhanes$SDMVPSU)
nhanesSvy <- svydesign(
  ids = ~SDMVPSU, strata = ~SDMVSTRA, weights = ~WTMEC2YR,
  nest = TRUE, data = nhanes
)
svyCreateTableOne2(
  vars = c("HI_CHOL", "race", "agecat", "RIAGENDR"),
  strata = "RIAGENDR", data = nhanesSvy,
  factorVars = c("HI_CHOL", "race", "RIAGENDR")
)
```

---

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,
  addOverall = F,
  pairwise = F,
  pairwise.showtest = F,
  n_original = 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       |
| 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                                     |
| 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 stratifed clolumns. Default: F |
| pairwise         | (optional, only used if strata are supplied) When there are three or more strata, it displays the p-values for pairwise comparisons. Default: F |
| pairwise.showtest |                                                                                                                     |
|                  | (optional, only used if strata are supplied) When using pairwise comparison, it displays the test used to calculate p-values for pairwise comparisons. Default: F |
| n_original       | Replace the number of weighted n with the n in the original data. 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(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

```
svyregress.display(svyglm.obj, decimal = 2, pcut.univariate = NULL)
```

## Arguments

svyglm.obj        svyglm.object

decimal           digit, Default: 2

pcut.univariate

                  pcut.univariate, Default: NULL

## 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,
  strata = NULL,
  weights = NULL,
  event = FALSE,
  count_by = NULL,
  labeldata = NULL,
  data_original = NULL,
  formula_original = 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 |
| 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

strata           Strata variable for coxph, Default: NULL

weights         Weights variable for coxph, Default: NULL

event            Show number and rates of event in survival analysis default:F

count_by        Select variables to count by subgroup, Default: NULL

labeldata       Label info, made by 'mk.lev' function, Default: NULL

data_original   Original data for competing risk analysis (before finegray transformation), Default: NULL

formula_original

                 Original formula for competing risk analysis (before finegray transformation), Default: NULL

## 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
)
```

---

TableSubgroupGLM          *TableSubgroupGLM: Sub-group analysis table for GLM and
                           GLMM(lme4 package).*

---

## 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,
  labeldata = NULL,
  count_by = NULL,
  event = FALSE
)
```

## 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 |
| labeldata | Label info, made by 'mk.lev' function, Default: NULL |
| count_by | Variable name to stratify counts by (string). Default: NULL. |
| event | If 'TRUE', show counts/metrics instead of only model estimates. Default: FALSE. |

## Details

This result is used to make forestplot.

## Value

Sub-group analysis table.

## See Also

safely,map,map2 glm svyglm

## Examples

```
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   *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,
  cluster = NULL,
  strata = NULL,
```

```
    weights = NULL,
    event = FALSE,
    count_by = NULL,
    labeldata = NULL,
    data_original = NULL,
    formula_original = 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. |
| 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 |
| strata | Strata variable for coxph, Default: NULL |
| weights | Weights variable for coxph, Default: NULL |
| event | Show number and rates of event in survival analysis default:F |
| count_by | Select variables to count by subgroup, Default: NULL |
| labeldata | Label info, made by 'mk.lev' function, Default: NULL |
| data_original | Original data for competing risk analysis (before finegray transformation), Default: NULL |
| formula_original | |
| | Original formula for competing risk analysis (before finegray transformation), Default: NULL |

## Details

This result is used to make forestplot.

## Value

Multiple sub-group analysis table.

## See Also

[map](map) [bind](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
)
```

---

TableSubgroupMultiGLM *TableSubgroupMultiGLM: Multiple sub-group analysis table for GLM.*

---

## 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,
  labeldata = NULL,
  count_by = NULL,
  event = FALSE
)
```

## 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 |
| `labeldata` | Label info, made by 'mk.lev' function, Default: NULL |
| `count_by` | Variable name to stratify counts by (string). Default: NULL. |
| `event` | If 'TRUE', show counts/metrics instead of only model estimates. Default: FALSE. |

## 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