Package ‘tab’
August 2, 2021

Type Package
Title Create Summary Tables for Statistical Reports
Version 5.1.1
Date 2021-08-01
Author Dane R. Van Domelen
Maintainer Dane R. Van Domelen <vandomed@gmail.com>
Description Contains functions for creating various types of summary tables, e.g. comparing characteristics across levels of a categorical variable and summarizing fitted generalized linear models, generalized estimating equations, and Cox proportional hazards models. Functions are available to handle data from simple random samples as well as complex surveys.
License GPL (>= 3)
LazyData true
Encoding UTF-8
Depends dplyr, knitr
Imports kableExtra, MASS, stats, survey (>= 4.1)
RoxygenNote 7.1.1
Suggests gee, rmarkdown, survival
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2021-08-02 04:30:02 UTC

R topics documented:

formatp ......................................................... 2
glm_v ......................................................... 3
tab ......................................................... 4
tabcoxph ..................................................... 5
tabdata ....................................................... 6	tabfreq ....................................................... 7
Description

Formats p-values for tables generated by the functions in the tab package. Handles rounding and presentation of p-values.

Usage

```
formatp(
P, decimals = c(2, 3), cuts = 0.01, lowerbound = 0.001, leading0 = TRUE, avoid1 = FALSE)
```

Arguments

- **p**: Numeric vector of p-values.
- **decimals**: Number of decimal places for p-values. If a vector is provided rather than a single value, number of decimal places will depend on what range the p-value lies in. See cuts input.
- **cuts**: Cut-point(s) to control number of decimal places used for p-values. For example, by default cuts = 0.1 and decimals = c(2, 3). This means that p-values in the range [0.1, 1] will be printed to two decimal places, while p-values in the range [0, 0.1] will be printed to three decimal places.
- **lowerbound**: Controls cut-point at which p-values are no longer printed as their value, but rather <lowerbound. For example, by default lowerbound = 0.001. Under this setting, p-values less than 0.001 are printed as <0.001.
If TRUE, p-values are printed with 0 before decimal place; if FALSE, the leading 0 is omitted.

If TRUE, p-values rounded to 1 are not printed as 1, but as >0.99 (or similarly depending on decimals and cuts).

Value

Character vector.

Examples

# Generate vector of numeric p-values
set.seed(123)
p <- c(runif(n = 5, min = 0, max = 1), 1, 0, 4e-7, 0.009)

# Round to nearest 2 decimals for p in (0.01, 1] and 3 decimals for p < 0.01
pvals <- formatp(p = p)

# Use 2 decimal places, a lower bound of 0.01, and omit the leading 0
pvals <- formatp(p = p, decimals = 2, lowerbound = 0.01, leading0 = FALSE)

 glm_v

Print a GLM Summary Table to the RStudio Viewer

Description

You can call this function as you would glm or pass a previously fitted glm object. Either way, the result is a summary table printed to the Viewer.

Usage

glm_v(...)

Arguments

... Arguments to pass to glm.

Value

kable
Examples

# Fit and view
glm_v(death_1yr ~ Age + Sex + Race, data = tabdata, family = "binomial")

# Fit then view
fit <- glm(death_1yr ~ Age + Sex + Race, data = tabdata, family = "binomial")
glm_v(fit)

# Piping is OMG so cool Hashtag HexStickerz
fit %>% glm_v()

---

Create Summary Tables for Statistical Reports

Description

Contains functions for creating various types of summary tables, e.g. comparing characteristics across levels of a categorical variable and summarizing fitted generalized linear models, generalized estimating equations, and Cox proportional hazards models. Functions are available to handle data from simple random samples as well as complex surveys.

Details

Package: tab
Type: Package
Version: 5.1.1
Date: 2021-08-01
License: GPL-3

See CRAN documentation for full list of functions.

Author(s)

Dane R. Van Domelen
<vandomed@gmail.com>

References

Acknowledgment: This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0940903.
tabcoxph  

Create Summary Table for Fitted Cox Proportional Hazards Model

Description

Creates a table summarizing a GEE fit using the `coxph` function.

Usage

```r
tabcoxph(
  fit,
  columns = c("beta.se", "hr.ci", "p"),
  var.labels = NULL,
  factor.compression = 1,
  sep.char = ",",
  decimals = 2,
  formatp.list = NULL
)
```

Arguments

- **fit**  
  Fitted `coxph` object.

- **columns**  
  Character vector specifying what columns to include. Choices for each element are "events", "beta", "se", "beta.se", "beta.betaci", "betaci", "hr", "hr.hrci", "hrci", "z", and "p".

- **var.labels**  
  Named list specifying labels to use for certain predictors. For example, if `fit` includes a predictor named "race" that you want to label "Race/ethnicity" and a predictor named "age_yrs" that you want to label "Age (years)", use `var.labels = list(race = "Race/ethnicity", age_yrs = "Age (years)"").

- **factor.compression**  
  Integer value from 1 to 5 controlling how much compression is applied to factor predictors (higher value = more compression). If 1, rows are Variable, Level 1 (ref), Level 2, ...; if 2, rows are Variable (ref = Level 1), Level 2, ...; if 3, rows are Level 1 (ref), Level 2, ...; if 4, rows are Level 2 (ref = Level 1), ...; if 5, rows are Level 2, ...

- **sep.char**  
  Character string with separator to place between lower and upper bound of confidence intervals. Typically ",-" or ",,.

- **decimals**  
  Numeric value specifying number of decimal places for numbers other than p-values.

- **formatp.list**  
  List of arguments to pass to `formatp`.

Value

- `kable`.
References


Examples

# Cox PH model with age, sex, race, and treatment
library("survival")
fit <- coxph(
  Surv(time = time, event = delta) ~ Age + Sex + Race + Group,
  data = tabdata
)
tabcoxph(fit)

# Can also use piping
fit %>% tabcoxph()

# Same as previous, but with custom labels for Age and Race and factors displayed in slightly more compressed format
fit %>%
tabcoxph(
  var.labels = list(Age = "Age (years)", Race = "Race/ethnicity"),
  factor.compression = 2
)

# Cox PH model with some higher-order terms
fit <- coxph(
  Surv(time = time, event = delta) ~ poly(Age, 2, raw = TRUE) + Sex + Race + Group + Race*Group,
  data = tabdata
)
fit %>% tabcoxph()

---

tabdata

Sample Dataset for tab Package

Description

Data frame with 15 variables, used to illustrate certain functions.

Source

Simulated data in R
tabfreq  Create Frequency Table

Description

Creates an I-by-J frequency table comparing the distribution of \( y \) across levels of \( x \).

Usage

```
tabfreq(  
  formula = NULL,  
  data = NULL,  
  x = NULL,  
  y = NULL,  
  columns = c("xgroups", "p"),  
  cell = "counts",  
  parenth = "col.percent",  
  sep.char = ",",  
  test = "chi.fisher",  
  xlevels = NULL,  
  yname = NULL,  
  ylevels = NULL,  
  compress.binary = FALSE,  
  yname.row = TRUE,  
  text.label = TRUE,  
  quantiles = NULL,  
  quantile.vals = FALSE,  
  decimals = 1,  
  formatp.list = NULL,  
  n.headings = FALSE,  
  kable = TRUE  
)  
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>formula</code></td>
<td>Formula, e.g. <code>Sex ~ Group</code>.</td>
</tr>
<tr>
<td><code>data</code></td>
<td>Data frame containing variables named in <code>formula</code>.</td>
</tr>
<tr>
<td><code>x</code></td>
<td>Vector indicating group membership for columns of I-by-J table.</td>
</tr>
<tr>
<td><code>y</code></td>
<td>Vector indicating group membership for rows of I-by-J table.</td>
</tr>
<tr>
<td><code>columns</code></td>
<td>Character vector specifying what columns to include. Choices for each element are &quot;n&quot; for total sample size, &quot;overall&quot; for overall distribution of ( y ), &quot;xgroups&quot; for distributions of ( y ) for each ( x ) group, &quot;test&quot; for test statistic, and &quot;p&quot; for ( p )-value.</td>
</tr>
<tr>
<td><code>cell</code></td>
<td>Character string specifying what statistic to display in cells. Choices are &quot;counts&quot;, &quot;tot.percent&quot;, &quot;col.percent&quot;, and &quot;row.percent&quot;.</td>
</tr>
</tbody>
</table>
**parenth**
Character string specifying what statistic to display in parentheses. Choices are "none", "se", "ci", "counts", "tot.percent", "col.percent", and "row.percent".

**sep.char**
Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",".

**test**
Character string specifying which test for association between x and y should be used. Choices are "chi.fisher" for Pearson’s chi-squared test if its assumptions are met, otherwise Fisher’s exact test; "chi"; "fisher"; "z" for z test without continuity correction; and "z.continuity" for z test with continuity correction. The last two only work if both x and y are binary.

**xlevels**
Character vector with labels for the levels of x, used in column headings.

**yname**
Character string with a label for the y variable.

**ylevels**
Character vector with labels for the levels of y. Note that levels of y are listed in the order that they appear when you run `table(y, x)`.

**compress.binary**
Logical value for whether to compress binary y variable to a single row, excluding the first level rather than showing both.

**yname.row**
Logical value for whether to include a row displaying the name of the y variable and indent the factor levels.

**text.label**
Character string with text to put after the y variable name, identifying what cell values and parentheses represent.

**quantiles**
Numeric value. If specified, table compares y across quantiles of x created on the fly.

**quantile.vals**
Logical value for whether labels for x quantiles should show quantile number and corresponding range, e.g. Q1 [0.00, 0.25), rather than just the quantile number.

**decimals**
Numeric value specifying number of decimal places for numbers other than p-values.

**formatp.list**
List of arguments to pass to `formatp`.

**n.headings**
Logical value for whether to display group sample sizes in parentheses in column headings.

**kable**
Logical value for whether to return a `kable`.

---

**Value**
`kable`.

---

**Examples**

```r
# Compare sex distribution by group
(freqtable1 <- tabfreq(Sex ~ Group, data = tabdata))

# Same as previous, but showing male row only and % (SE) rather than n (%)
(freqtable2 <- tabfreq(Sex ~ Group, data = tabdata,
    cell = "col.percent", parenth = "se",
    compress.binary = TRUE))
```
Description

Creates an I-by-J frequency table comparing the distribution of \( y \) across levels of \( x \).

Usage

\[
\text{tabfreq.svy(}
\quad \text{formula,}
\quad \text{design,}
\quad \text{columns = c("xgroups", "p"),}
\quad \text{cell = "col.percent",}
\quad \text{parenth = "se",}
\quad \text{sep.char = ",",}
\quad \text{xlevels = NULL,}
\quad \text{yname = NULL,}
\quad \text{ylevels = NULL,}
\quad \text{compress.binary = FALSE,}
\quad \text{yname.row = TRUE,}
\quad \text{text.label = NULL,}
\quad \text{decimals = 1,}
\quad \text{svychisq.list = NULL,}
\quad \text{formatp.list = NULL,}
\quad \text{n.headings = FALSE,}
\quad \text{N.headings = FALSE,}
\quad \text{kable = TRUE}
\quad \text{)}
\]

Arguments

- **formula**: Formula, e.g. Race ~ Sex.
- **design**: Survey design object from `svydesign`.
- **columns**: Character vector specifying what columns to include. Choices for each element are "n" for total unweighted sample size, "N" for total weighted sample size, "overall" for overall distribution of \( y \), "xgroups" for distributions of \( y \) for each \( x \) group, and "p" for Chi-square p-value.
- **cell**: Character string specifying what statistic to display in cells. Choices are "n", "N", and "col.percent".
- **parenth**: Character string specifying what statistic to display in parentheses. Choices are "none", "n", "N", "col.percent", "se", and "ci".
- **sep.char**: Character string with separator to place between lower and upper bound of confidence intervals. Typically ",-" or ",,\).
- **xlevels**: Character vector with labels for the levels of \( x \), used in column headings.
yname          Character string with a label for the y variable.
ylevels        Character vector with labels for the levels of y. Note that levels of y are listed in the order that they appear when you run `table(y, x)`.
compress.binary Logical value for whether to compress binary y variable to a single row, excluding the first level rather than showing both.
yname.row      Logical value for whether to include a row displaying the name of the y variable.
text.label     Character string with text to put after the y variable name, identifying what cell values and parentheses represent.
decimals       Numeric value specifying number of decimal places for numbers other than p-values.
svychisq.list  List of arguments to pass to `svychisq`.
formatp.list   List of arguments to pass to `formatp`.
n.headings     Logical value for whether to display unweighted sample sizes in parentheses in column headings.
N.headings     Logical value for whether to display weighted sample sizes in parentheses in column headings.
kable          Logical value for whether to return a kable.

Details

Basically `tabmedians` for complex survey data. Relies heavily on the `survey` package.

Value

kable or character matrix.

Examples

```r
# Create survey design object
library("survey")
design <- svydesign(
  data = tabsvydata,
  ids = ~sdmvsu,
  strata = ~sdmvsstra,
  weights = ~wtmec2yr,
  nest = TRUE
)

# Compare race distribution by sex
tabfreq.svy(Race ~ Sex, design)
```
tabgee

Create Summary Table for Fitted Generalized Estimating Equation Model

Description

Creates a table summarizing a GEE fit using the gee function.

Usage

```r
tabgee(
  fit,
  data = NULL,
  columns = NULL,
  robust = TRUE,
  var.labels = NULL,
  factor.compression = 1,
  sep.char = ",",
  decimals = 2,
  formatp.list = NULL
)
```

Arguments

- **fit**: Fitted gee object.
- **data**: Data frame that served as 'data' in function call to gee. Only needs to be specified if one or more of the predictors is a factor and factor.compression is 1, 2, 3, or 4.
- **columns**: Character vector specifying what columns to include. Choices for each element are "beta", "se", "betaci" for 95% CI for Beta, "beta.se" for Beta (SE), "beta.ci" for Beta (95% CI), "or", "orci" for 95% CI for OR, "or.ci" for OR (95% CI), "hr", "hr.ci" for 95% CI for HR, "hr.ci" for HR (95% CI), "z" for z statistic, and "p". If OR's or HR's are requested, the function will trust that exponentiated betas correspond to these quantities.
- **robust**: Logical value for whether to use robust standard errors.
- **var.labels**: Named list specifying labels to use for certain predictors. For example, if fit includes a predictor named "race" that you want to label "Race/ethnicity" and a predictor named "age_yrs" that you want to label "Age (years)", use `var.labels = list(race = "Race/ethnicity", age_yrs = "Age (years)".`
- **factor.compression**: Integer value from 1 to 5 controlling how much compression is applied to factor predictors (higher value = more compression). If 1, rows are Variable, Level 1 (ref), Level 2, ...; if 2, rows are Variable (ref = Level 1), Level 2, ...; if 3, rows are Level 1 (ref), Level 2, ...; if 4, rows are Level 2 (ref = Level 1), ...; if 5, rows are Level 2, ...
sep.char | Character string with separator to place between lower and upper bound of confidence intervals. Typically "," or ",."

decimals | Numeric value specifying number of decimal places for numbers other than p-values.

formatp.list | List of arguments to pass to \texttt{formatp}.

**Value**

\texttt{kable}.

**Examples**

```r
# Load in sample dataset and convert to long format
tabdata2 <- reshape(data = tabdata,
    varying = c("bp.1", "bp.2", "bp.3", "highbp.1",
                "highbp.2", "highbp.3"),
    timevar = "bp.visit", direction = "long")
tabdata2 <- tabdata2[order(tabdata2$id), ]

# Blood pressure at 1, 2, and 3 months vs. age, sex, race, and treatment
library("gee")
fit <- gee(bp ~ Age + Sex + Race + Group, id = id, data = tabdata2,
            corstr = "unstructured")
tabgee(fit)

# Can also use piping
fit %>% tabgee(data = tabdata2)

# Same as previous, but with custom labels for Age and Race and factors
# displayed in slightly more compressed format
fit %>%
    tabgee(
        data = tabdata2,
        var.labels = list(Age = "Age (years)", Race = "Race/ethnicity"),
        factor.compression = 2
    )

# GEE with some higher-order terms
# higher-order terms
fit <- gee(
    highbp ~ poly(Age, 2, raw = TRUE) + Sex + Race + Group + Race*Group,
    id = id,
    data = tabdata2,
    family = "binomial",
    corstr = "unstructured"
)
fit %>% tabgee(data = tabdata2)
```
tabglm

Create Summary Table for Fitted Generalized Linear Model

Description

Creates a table summarizing a GLM fit using \texttt{glm}.

Usage

\begin{verbatim}
tabglm(
  fit,
  columns = NULL,
  xvarlabels = NULL,
  factor.compression = 1,
  sep.char = "\, ",
  decimals = 2,
  formatp.list = NULL
)
\end{verbatim}

Arguments

- \texttt{fit} Fitted \texttt{glm} object.
- \texttt{columns} Character vector specifying what columns to include. Choices for each element are "beta", "se", "betaci" for 95% CI for beta, "beta.se" for Beta (SE), "beta.ci" for Beta (95% CI), "or", "orci" for 95% CI for OR, "or.ci" for OR (95% CI), "hr", "hrci" for 95% CI for HR, "hr.ci" for HR (95% CI), "test" for z/t statistic, and "p". If OR's or HR's are requested, the function will trust that exponentiated betas correspond to these quantities.
- \texttt{xvarlabels} Named list specifying labels to use for certain predictors. For example, if \texttt{fit} includes a predictor named "race" that you want to label "Race/ethnicity" and a predictor named "age_yrs" that you want to label "Age (years)", use \texttt{xvarlabels = list(race = "Race/ethnicity", age_yrs = "Age (years)").
- \texttt{factor.compression} Integer value from 1 to 5 controlling how much compression is applied to factor predictors (higher value = more compression). If 1, rows are Variable (ref = Level 1), Level 2, ...; if 2, rows are Variable (ref = Level 1), Level 2, ...; if 3, rows are Level 1 (ref), Level 2, ...; if 4, rows are Level 2 (ref = Level 1), ...; if 5, rows are Level 2, ...
- \texttt{sep.char} Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",", \texttt{kable}.
- \texttt{decimals} Numeric value specifying number of decimal places for numbers other than p-values.
- \texttt{formatp.list} List of arguments to pass to \texttt{formatp}.
Examples

# Linear regression: BMI vs. age, sex, race, and treatment
fit <- glm(BMI ~ Age + Sex + Race + Group, data = tabdata)
tabglm(fit)

# Can also use piping
fit %>% tabglm()

# Logistic regression: 1-year mortality vs. age, sex, race, and treatment
fit <- glm(
  death_1yr ~ Age + Sex + Race + Group,
  data = tabdata,
  family = binomial
)
fit %>% tabglm()

# Same as previous, but with custom labels for Age and Race and factors
# displayed in slightly more compressed format
fit %>%
tabglm(
  xvarlabels = list(Age = "Age (years)", Race = "Race/ethnicity"),
  factor.compression = 2
)

# Logistic regression model with some higher-order terms
fit <- glm(
  death_1yr ~ poly(Age, 2, raw = TRUE) + Sex + BMI + Sex * BMI,
  data = tabdata,
  family = "binomial"
)
fit %>% tabglm()

---

### tabmeans

Create Table Comparing Group Means

**Description**

Creates a table comparing the mean of y across levels of x.

**Usage**

```r
tabmeans(
  formula = NULL,
```
data = NULL,
        x = NULL,
        y = NULL,
        columns = c("xgroups", "p"),
        parenth = "sd",
        sep.char = ",",
        variance = "unequal",
        xlevels = NULL,
        yname = NULL,
        text.label = NULL,
        quantiles = NULL,
        quantile.vals = FALSE,
        decimals = NULL,
        formatp.list = NULL,
        n.headings = TRUE,
        kable = TRUE
    )

Arguments

formula  Formula, e.g. BMI ~ Group.
data       Data frame containing variables named in formula.
x          Vector of values for the categorical x variable.
y          Vector of values for the continuous y variable.
columns   Character vector specifying what columns to include. Choices for each element are "n" for total sample size, "overall" for overall mean, "xgroups" for x group means, "diff" for difference in x group means (this one and the next two are only available for binary x), "diffci" for 95 x group means, "diff.ci" for difference in group means and 95 confidence interval, "test" for test statistic, and "p" for p-value.
parenth   Character string specifying what statistic to display in parentheses after the means. Choices are "none", "sd", "se", "t.ci", "z.ci", "range", and "minmax".
sep.char  Character string with separator to place between lower and upper bound of confidence intervals. Typically "+" or ",".
variance  Character string specifying which version of the two-sample t-test to use if x has 2 levels. Choices are "equal" for equal variance t-test, "unequal" for unequal variance t-test, and "f" for F test to determine which to use.
xlevels   Character vector with labels for the levels of x, used in column headings.
yname     Character string with a label for the y variable.
text.label Character string with text to put after the y variable name, identifying what cell values and parentheses represent.
quantiles Numeric value. If specified, table compares y across quantiles of x created on the fly.
quantile.vals Logical value for whether labels for x quantiles should show quantile number and corresponding range, e.g. Q1 [0.00, 0.25], rather than just the quantile number.
decimals Numeric value specifying number of decimal places for numbers other than p-values.
formatp.list List of arguments to pass to formatp.
n.headings Logical value for whether to display group sample sizes in parentheses in column headings.
kable Logical value for whether to return a kable.

Details
A t-test is used to compare means if x has two levels, and a one-way analysis of variance is used if x has more than two levels. Observations with missing values for x and/or y are dropped.

Value
kable or character matrix.

Examples
# Compare mean BMI in control vs. treatment group in sample dataset
(meanstable1 <- tabmeans(BMI ~ Group, data = tabdata))

# Compare mean baseline systolic BP across tertiles of BMI
(meanstable2 <- tabmeans(bp.1 ~ BMI, data = tabdata,
quantiles = 3, yname = "Systolic BP"))
n.headings = FALSE,
N.headings = FALSE,
kable = TRUE
)

Arguments

formula  Formula, e.g. BMI ~ Sex.
design  Survey design object from svydesign.
columns  Character vector specifying what columns to include. Choices for each element are "n" for total sample size, "overall" for overall mean, "xgroups" for x group means, "diff" for difference in x group means (this one and the next two are only available for binary x), "diffci" for 95 x group means, "diff.ci" for difference in group means and 95 confidence interval, and "p" for p-value.
parenth  Character string specifying what statistic to display in parentheses after the means. Choices are "none", "sd", "se", "t.ci", "z.ci", "range", and "minmax".
sep.char  Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",".
xlevels  Character vector with labels for the levels of x, used in column headings.
yname  Character string with a label for the y variable.
text.label  Character string with text to put after the y variable name, identifying what cell values and parentheses represent.
decimals  Numeric value specifying number of decimal places for numbers other than p-values.
anova.svyglm.list  List of arguments to pass to anova.svyglm. Only used if x has three or more levels.
formatp.list  List of arguments to pass to formatp.
n.headings  Logical value for whether to display group sample sizes in parentheses in column headings.
N.headings  Logical value for whether to display weighted sample sizes in parentheses in column headings.
kable  Logical value for whether to return a kable.

Details

Basically tabmeans for complex survey data. Relies heavily on the survey package.

Value

kable or character matrix.
Examples

# Create survey design object
library("survey")
design <- svydesign(
data = tabsvydata,
ids = ~sdmvpsu,
strata = ~sdmvstra,
weights = ~wtmec2yr,
nest = TRUE
)

# Compare mean BMI by sex
(meanstable <- tabmeans.svy(BMI ~ Sex, design = design))

---

**tabmedians**  
Create Table Comparing Group Medians

**Description**

Creates a table comparing the median of \( y \) across levels of \( x \).

**Usage**

```r
tabmedians(
  formula = NULL,
  data = NULL,
  x = NULL,
  y = NULL,
  columns = c("xgroups", "p"),
  parenth = "iqr",
  sep.char = ",",
  xlevels = NULL,
  yname = NULL,
  text.label = NULL,
  quantiles = NULL,
  quantile.vals = FALSE,
  decimals = NULL,
  formatp.list = NULL,
  n.headings = TRUE,
  kable = TRUE
)
```

**Arguments**

- `formula`  
  Formula, e.g. `BMI ~ Group`.
- `data`  
  Data frame containing variables named in formula.
### tabmedians

- **x**: Vector of values for the categorical x variable.
- **y**: Vector of values for the continuous y variable.
- **columns**: Character vector specifying what columns to include. Choices for each element are "n" for total sample size, "overall" for overall median, "xgroups" for x group medians, "diff" for difference in x group medians (only available for binary x), "test" for test statistic, and "p" for p-value.
- **parenth**: Character string specifying what values are shown in parentheses after the medians in each cell. Choices are "none", "iqr", "q1q3" for first and third quartiles, "range", "minmax", and "ci" for 95% confidence interval for the medians based on normal approximation to binomial.
- **sep.char**: Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",".
- **xlevels**: Character vector with labels for the levels of x, used in column headings.
- **yname**: Character string with a label for the y variable.
- **text.label**: Character string with text to put after the y variable name, identifying what cell values and parentheses represent.
- **quantiles**: Numeric value. If specified, table compares y across quantiles of x created on the fly.
- **quantile.vals**: Logical value for whether labels for x quantiles should show quantile number and corresponding range, e.g. Q1 [0.00, 0.25), rather than just the quantile number.
- **decimals**: Numeric value specifying number of decimal places for numbers other than p-values.
- **formatp.list**: List of arguments to pass to `formatp`.
- **n.headings**: Logical value for whether to display group sample sizes in parentheses in column headings.
- **kable**: Logical value for whether to return a `kable`.

### Details

If x has 2 levels, a Mann-Whitney U (also known as Wilcoxon rank-sum) test is used to test whether the distribution of y differs in the two groups; if x has more than 2 levels, a Kruskal-Wallis test is used to test whether the distribution of y differs across at least two of the groups. Observations with missing values for x and/or y are dropped.

### Value

- `kable`.

### Examples

```r
# Compare median BMI in control group vs. treatment group in sample dataset
(medtable1 <- tabmedians(BMI ~ Group, data = tabdata))

# Compare median baseline systolic BP across tertiles of BMI
```
(medtable2 <- tabmedians(bp.1 ~ BMI, data = tabdata, 
    quantiles = 3, yname = "Systolic BP"))

**tabmedians.svy**

Create Table Comparing Group Medians (for Complex Survey Data)

**Description**

Creates a table comparing the median of \( y \) across levels of \( x \).

**Usage**

```r
tabmedians.svy(
    formula,
    design,
    columns = c("xgroups", "p"),
    parenth = "iqr",
    sep.char = ",",
    xlevels = NULL,
    yname = NULL,
    text.label = NULL,
    decimals = NULL,
    svyranktest.list = NULL,
    formatp.list = NULL,
    n.headings = FALSE,
    N.headings = FALSE,
    kable = TRUE
)
```

**Arguments**

- `formula`: Formula, e.g. `BMI ~ Sex`.
- `design`: Survey design object from `svydesign`.
- `columns`: Character vector specifying what columns to include. Choices for each element are "n" for total sample size, "overall" for overall median, "xgroups" for \( x \) group medians, "diff" for difference in \( x \) group medians (only available for binary \( x \)), and "p" for p-value.
- `parenth`: Character string specifying what values are shown in parentheses after the medians in each cell. Choices are "none", "iqr", "q1q3" for first and third quartiles, "range", "minmax", and "ci" for 95% confidence interval for the median.
- `sep.char`: Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",".
- `xlevels`: Character vector with labels for the levels of \( x \), used in column headings.
- `yname`: Character string with a label for the \( y \) variable.
text.label Character string with text to put after the y variable name, identifying what cell values and parentheses represent.
decimals Numeric value specifying number of decimal places for numbers other than p-values.
svyranktest.list List of arguments to pass to svyranktest.
formatp.list List of arguments to pass to formatp.
n.headings Logical value for whether to display group sample sizes in parentheses in column headings.
N.headings Logical value for whether to display weighted sample sizes in parentheses in column headings.
kable Logical value for whether to return a kable.

Details

Basically tabmedians for complex survey data. Relies heavily on the survey package.

Value

kable or character matrix.

Examples

# Create survey design object
library("survey")
design <- svydesign(
  data = tabsvydata,
  ids = ~sdmvpsu,
  strata = ~sdmvstra,
  weights = ~wtmec2yr,
  nest = TRUE
)

# Compare median BMI by sex
(medtable1 <- tabmedians.svy(BMI ~ Sex, design = design))
Usage

```r
tabmulti(
  formula = NULL,
  data,
  xvarname = NULL,
  yvarnames = NULL,
  ymeasures = NULL,
  columns = c("xgroups", "p"),
  listwise.deletion = FALSE,
  sep.char = ",",
  xlevels = NULL,
  yvarlabels = NULL,
  ylevels = NULL,
  quantiles = NULL,
  quantile.vals = FALSE,
  decimals = NULL,
  formatp.list = NULL,
  n.headings = FALSE,
  tabmeans.list = NULL,
  tabmedians.list = NULL,
  tabfreq.list = NULL,
  kable = TRUE
)
```

Arguments

- `formula` - Formula, e.g. `Age + Sex + Race + BMI ~ Group`.
- `data` - Data frame containing variables named in `formula`.
- `xvarname` - Character string with name of column variable. Should be one of `names(data)`.
- `yvarnames` - Character vector with names of row variables. Each element should be one of `names(data)`.
- `ymeasures` - Character vector specifying whether each `y` variable should be summarized by mean, median, or frequency. For example, if you want to compare frequencies for the first variable, means for the second, and medians for the third, you would set `ymeasures = c("freq", "mean", "median")`. If unspecified, function compares means for numeric variables and frequencies for factor and character variables.
- `columns` - Character vector specifying what columns to include. Choices for each element are "n" for total sample size, "overall" for overall statistics, "xgroups" for x group statistics, "test" for test statistic, and "p" for p-value.
- `listwise.deletion` - Logical value for whether observations with missing values for any `y` variable should be excluded entirely (as opposed to using all available data for each comparison).
- `sep.char` - Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",".
xlevels Character vector with labels for the levels of x, used in column headings.
yvarlabels Named list specifying labels for certain y variables. For example, if you want variables named "race" and "age_yrs" to print as "Race/ethnicity" and "Age (years)", use \code{yvarlabels = list(race = "Race/ethnicity", age_yrs = "Age (years)").
ylevels Character vector (if only 1 frequency comparison) or list of character vectors with labels for the levels of each categorical y variable.
quantiles Numeric value. If specified, function compares y variables across quantiles of x. For example, if x contains BMI values and yvarnames includes HDL and race, setting quantiles = 3 compares mean BMI and distribution of race across BMI tertiles.
quantile.vals Logical value for whether labels for x quantiles should show quantile number and corresponding range, e.g. Q1 [0.00, 0.25), rather than just the quantile number.
decimals Numeric vector specifying number of decimal places for numbers other than p-values for each y variable. Can be a single value to use for all y variables.
formatp.list List of arguments to pass to \code{formatp}.
n.headings Logical value for whether to display group sample sizes in parentheses in column headings.
tabmeans.list List of arguments to pass to \code{tabmeans}.
tabmedians.list List of arguments to pass to \code{tabmedians}.
tabfreq.list List of arguments to pass to \code{tabfreq}.
kable Logical value for whether to return a \code{kable}.

Value

\code{kable} or character matrix.

Examples

# Compare age, sex, race, and BMI in control vs. treatment group
tabmulti(Age + Sex + Race + BMI ~ Group, data = tabdata)

# Same as previous, but compare medians rather than means for BMI
tabmulti(Age + Sex + Race + BMI ~ Group, data = tabdata,
ymeasures = c("mean", "freq", "freq", "median")
**tabmulti.svy**  
Create Table Comparing Characteristics Across Levels of a Categorical Variable (for Complex Survey Data)

**Description**

Creates a table comparing multiple characteristics (e.g. median age, mean BMI, and race/ethnicity distribution) across levels of `x`.

**Usage**

```r
tabmulti.svy(
  formula = NULL,
  design,
  xvarname = NULL,
  yvarnames = NULL,
  ymeasures = NULL,
  columns = c("xgroups", "p"),
  listwise.deletion = FALSE,
  sep.char = ", ",
  xlevels = NULL,
  yvarlabels = NULL,
  ylevels = NULL,
  decimals = NULL,
  formatp.list = NULL,
  n.headings = FALSE,
  N.headings = FALSE,
  kable = TRUE,
  tabmeans.svy.list = NULL,
  tabmedians.svy.list = NULL,
  tabfreq.svy.list = NULL
)
```

**Arguments**

- **formula**: Formula, e.g. `Age + Race + BMI ~ Sex`.
- **design**: Survey design object from `svydesign`.
- **xvarname**: Character string with name of column variable. Should be one of `names(design$variables)`.
- **yvarnames**: Character vector with names of row variables. Each element should be one of `names(design$variables)`.
- **ymeasures**: Character vector specifying whether each `y` variable should be summarized by mean, median, or frequency. For example, if you want to compare frequencies for the first variable, means for the second, and medians for the third, you would set `ymeasures = c("freq","mean","median")`. If unspecified, function compares means for numeric variables and frequencies for factor and character variables.
columns Character vector specifying what columns to include. Choices for each element are "n" for unweighted sample size, "N" for weighted sample size, "overall" for overall statistics, "xgroups" for x group statistics, and "p" for p-value.

listwise.deletion Logical value for whether observations with missing values for any y variable should be excluded entirely (as opposed to using all available data for each comparison).

sep.char Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",".

xlevels Character vector with labels for the levels of x, used in column headings.

yvarlabels Named list specifying labels for certain y variables. For example, if you want variables named "race" and "age_yrs" to print as "Race/ethnicity" and "Age (years)", use `yvarlabels = list(race = "Race/ethnicity", age_yrs = "Age (years)")`.

ylevels Character vector (if only 1 frequency comparison) or list of character vectors with labels for the levels of each categorical y variable.

decimals Numeric vector specifying number of decimal places for numbers other than p-values for each y variable. Can be a single value to use for all y variables.

formatp.list List of arguments to pass to `formatp`.

n.headings Logical value for whether to display unweighted sample sizes in parentheses in column headings.

N.headings Logical value for whether to display weighted sample sizes in parentheses in column headings.

kable Logical value for whether to return a kable.

tabmeans.svy.list List of arguments to pass to `tabmeans.svy`.

tabmedians.svy.list List of arguments to pass to `tabmedians.svy`.

tabfreq.svy.list List of arguments to pass to `tabfreq.svy`.

Details

Basically `tabmulti` for complex survey data. Relies heavily on the `survey` package.

Value

kable or character matrix.

Examples

# Create survey design object
library("survey")
design <- svydesign(
  data = tabsvydata,
  ids = ~sdmvpsu,
strata = ~sdmvstra,
weights = ~wtmec2yr,
nest = TRUE
)

# Compare age, race, and BMI by sex
tabmulti.svy(Age + Race + BMI ~ Sex, design)

---

**tabreg**  
*Create Regression Table from Betas and Standard Errors*

**Description**

Useful for quickly creating a summary table.

**Usage**

```r
tabreg(
  betas,
  ses = NULL,
  varcov = NULL,
  columns = c("beta.se", "p"),
  sep.char = ", ",
  decimals = NULL,
  formatp.list = NULL,
  labels = NULL
)
```

**Arguments**

- **betas**  
  Numeric vector.
- **ses**  
  Numeric vector.
- **varcov**  
  Numeric matrix.
- **columns**  
  Character vector specifying what columns to include. Choices are "beta", "se","betaci","beta.se","beta.ci","or","orci","or.ci", and "p".
- **sep.char**  
  Character string with separator to place between lower and upper bound of confidence intervals. Typically "-" or ",".
- **decimals**  
  Numeric value specifying number of decimal places for numbers other than p-values.
- **formatp.list**  
  List of arguments to pass to `formatp`.
- **labels**  
  Character vector.

**Value**

`kable`.
# Create summary table for mtcars regression

```r
fit <- lm(mpg ~ wt + hp + drat, data = mtcars)
tabreg(
  betas = fit$coef,
  varcov = vcov(fit),
  labels = c("Intercept", "Weight", "HP", "Rear axle ratio")
)
```

tabsvydata | Sample Survey Dataset for tab Package

## Description

Data frame with with 9 variables, used to illustrate certain functions. Data are derived from the National Health and Nutrition Examination Survey, years 2003-2004, although the variables 'time' and 'event' are simulated (fake).

## Source


## References


toviewer | Output a Table to the RStudio Viewer

## Description

Does some basic formatting and then calls `kable` and `kable_styling` to print table to Viewer.

## Usage

toviewer(x)

## Arguments

x | Character matrix
Index

anova.svyglm, 17
coxph, 5
formatp, 2, 5, 8, 10, 12, 13, 16, 17, 19, 21, 23, 25, 26
gee, 11
glm, 3, 13
glm_v, 3
kable, 5, 8, 10, 12–14, 16, 17, 19, 21, 23, 25–27
kable_styling, 27
svychisq, 10
svydesign, 9, 17, 20, 24
svyranktest, 21

tab, 4
tabcxph, 5	tabdata, 6
tabfreq, 7, 23
tabfreq.svy, 9, 25
tabgee, 11
tabglm, 13
tabmeans, 14, 17, 23
tabmeans.svy, 16, 25
tabmedians, 10, 18, 21, 23
tabmedians.svy, 20, 25
tabmulti, 21, 25
tabmulti.svy, 24	tabreg, 26	tabsydata, 27
toviewer, 27