

Package ‘arsenal’

January 16, 2019

Title An Arsenal of 'R' Functions for Large-Scale Statistical Summaries

Version 2.0.0

Date 2019-01-15

Description An Arsenal of 'R' functions for large-scale statistical summaries, which are streamlined to work within the latest reporting tools in 'R' and 'RStudio' and which use formulas and versatile summary statistics for summary tables and models. The primary functions include `tableby()`, a Table-1-like summary of multiple variable types 'by' the levels of one or more categorical variables; `paired()`, a Table-1-like summary of multiple variable types paired across two time points; `modelsum()`, which performs simple model fits on one or more endpoints for many variables (univariate or adjusted for covariates); `freqlist()`, a powerful frequency table across many categorical variables; `compare.data.frame()`, the S3 method for comparing data.frames; and `write2()`, a function to output tables to a document.

Suggests broom, stringr, magrittr, knitr, rmarkdown, xtable, pander, survival ($\geq 2.41-3$), coin, pROC, MASS, gam, rpart, yaml

Depends R ($\geq 3.2.0$), stats ($\geq 3.2.0$)

Imports testthat, utils ($\geq 3.2.0$)

URL <https://github.com/eheinzen/arsenal>,
<https://cran.r-project.org/package=arsenal>

BugReports <https://github.com/eheinzen/arsenal/issues>

VignetteBuilder knitr

License GPL (≥ 2)

RoxygenNote 6.1.1

LazyData true

Encoding UTF-8

NeedsCompilation no

Author Ethan Heinzen [aut, cre],
Jason Sinnwell [aut],

Elizabeth Atkinson [aut],
 Tina Gunderson [aut],
 Gregory Dougherty [aut],
 Patrick Votruba [ctb],
 Ryan Lennon [ctb],
 Andrew Hanson [ctb],
 Krista Goergen [ctb],
 Emily Lundt [ctb],
 Brendan Broderick [ctb],
 Maddie McCullough [art]

Maintainer Ethan Heinzen <heinzen.ethan@mayo.edu>

Repository CRAN

Date/Publication 2019-01-16 05:30:02 UTC

R topics documented:

arsenal	3
arsenal_table	4
as.data.frame.freqlist	5
as.data.frame.modelsum	6
as.data.frame.tableby	7
compare.data.frame	8
comparison.control	9
comparison.tolerances	10
diffs	12
formulize	13
freq.control	14
freqlist	15
freqlist.internal	16
internal.functions	17
keep.labels	17
labels	18
mdy.Date	19
mockstudy	21
modelsum	22
modelsum.control	23
modelsum.family	24
modelsum.internal	25
NA.operations	26
padjust	26
paired	27
paired.control	29
paired.internal	30
summary.compare	31
summary.freqlist	32
summary.modelsum	33
summary.tableby	34

tableby	36
tableby.control	38
tableby.internal	41
tableby.stats	42
tableby.stats.internal	44
write2	45
write2.internal	47
write2specific	48
yaml	50
%nin%	51

Index	52
--------------	-----------

arsenal *An Arsenal of 'R' Functions for Large-Scale Statistical Summaries*

Description

An Arsenal of 'R' functions for large-scale statistical summaries, which are streamlined to work within the latest reporting tools in 'R' and 'RStudio' and which use formulas and versatile summary statistics for summary tables and models.

Details

The package download, NEWS, and README are available on CRAN: <https://cran.r-project.org/package=arsenal>

Functions

Below are listed some of the most widely used functions available in arsenal:

tableby: Summary statistics of a set of independent variables by a categorical variable.

paired: Summary statistics of a set of independent variables paired across two timepoints.

modelsum: Fit models over each of a set of independent variables with a response variable.

freqlist: Approximate the output from SAS's PROC FREQ procedure when using the /list option of the TABLE statement.

compare.data.frame: Compare two data.frames and report any differences between them, much like SAS's PROC COMPARE procedure.

write2word, **write2html**, **write2pdf**: Functions to output tables to a single Word, HTML, or PDF document.

write2: Functions to output tables to a single document. (Also the S3 backbone behind the write2* functions.)

keep.labels: Keep the 'label' attribute on an R object when subsetting.

formulize: A shortcut to generate one-, two-, or many-sided formulas.

mdy.Date and **Date.mdy**: Convert numeric dates for month, day, and year to Date object, and vice versa.

`is.Date`: Test if an object is a date.
`%nin%`: Test for "not in".
`allNA` and `includeNA`: some useful functions for dealing with NAs.

Data

`mockstudy`: Mock study data for examples.

Examples

```
library(arsenal)
```

arsenal_table	arsenal <i>tables with common structure</i>
---------------	---

Description

arsenal tables with common structure

Usage

```
has_strata(x)

## S3 method for class 'arsenal_table'
x[i, j, ...]

## S3 method for class 'arsenal_table'
labels(object, ...)

## S3 replacement method for class 'arsenal_table'
labels(x) <- value

## S3 method for class 'arsenal_table'
print(x, ...)

## S3 method for class 'arsenal_table'
merge(x, y, all = FALSE, all.x = all,
      all.y = all, ...)

## S3 method for class 'freqlist'
merge(x, y, all = TRUE, ...)

## S3 method for class 'summary.arsenal_table'
print(x, ..., format = if
      (!is.null(x$text) && x$text %in% c("html", "latex")) x$text else
      "markdown", escape = x$text %nin% c("html", "latex"), width = NULL,
      min.split = NULL)
```

Arguments

x, y, object	An object of class "arsenal_table"
i, j	A vector to index x with: either names of variables, a numeric vector, or a logical vector of appropriate length. i indexes the x-variables, and j indexes the by-variables.
...	Other arguments (only used in print.summary.arsenal_table)
value	A list of new labels.
all, all.x, all.y	Logicals, denoting which terms to keep if not all are in common.
format	Passed to kable : the format for the table. The default here is "markdown". To use the default in kable, pass NULL. If x\$text specifies LaTeX or HTML formatting, that format is used in the table.
escape	Passed to kable : should special characters be escaped when printed?
width, min.split	Passed to smart.split for formatting of the "term" column.

See Also

[merge, labels](#)

as.data.frame.freqlist

as.data.frame.freqlist

Description

Convert [freqlist](#) object to a data.frame.

Usage

```
## S3 method for class 'freqlist'
as.data.frame(x, ..., labelTranslations = NULL,
              list.ok = FALSE)
```

Arguments

x	An object of class "freqlist".
...	Arguments to pass to freq.control
labelTranslations	A named list (or vector) where the name is the label in the output to be replaced in the pretty rendering by the character string value for the named element of the list, e.g., list(age = "Age(Years)", meansd = "Mean(SD)").
list.ok	If the object has multiple by-variables, is it okay to return a list of data.frames instead of a single data.frame? If FALSE but there are multiple by-variables, a warning is issued.

Value

A data.frame corresponding to the freqlist object.

as.data.frame.modelsum

as.data.frame.modelsum

Description

Coerce a `modelsum` object to a `data.frame`.

Usage

```
## S3 method for class 'modelsum'  
as.data.frame(x, ..., labelTranslations = NULL,  
             list.ok = FALSE)
```

Arguments

<code>x</code>	A <code>modelsum</code> object.
<code>...</code>	Arguments to pass to <code>modelsum.control</code> .
<code>labelTranslations</code>	A named list (or vector) where the name is the label in the output to be replaced in the pretty rendering by the character string value for the named element of the list, e.g., <code>list(age = "Age(Years)", meansd = "Mean(SD)")</code> .
<code>list.ok</code>	If the object has multiple by-variables, is it okay to return a list of <code>data.frames</code> instead of a single <code>data.frame</code> ? If <code>FALSE</code> but there are multiple by-variables, a warning is issued.

Value

A `data.frame`.

Author(s)

Ethan Heinzen, based on code originally by Greg Dougherty

See Also

[modelsum](#), [summary.modelsum](#)

as.data.frame.tableby *as.data.frame.tableby*

Description

Coerce a [tableby](#) object to a `data.frame`.

Usage

```
## S3 method for class 'tableby'  
as.data.frame(x, ..., labelTranslations = NULL,  
             list.ok = FALSE)
```

Arguments

<code>x</code>	A tableby object.
<code>...</code>	Arguments to pass to tableby.control .
<code>labelTranslations</code>	A named list (or vector) where the name is the label in the output to be replaced in the pretty rendering by the character string value for the named element of the list, e.g., <code>list(age = "Age(Years)", meansd = "Mean(SD)")</code> .
<code>list.ok</code>	If the object has multiple by-variables, is it okay to return a list of <code>data.frames</code> instead of a single <code>data.frame</code> ? If <code>FALSE</code> but there are multiple by-variables, a warning is issued.

Value

A `data.frame`.

Author(s)

Ethan Heinzen, based on code originally by Greg Dougherty

See Also

[tableby](#), [tableby](#)

compare.data.frame *Compare two data.frames and report differences*

Description

Compare two data.frames and report any differences between them, much like SAS's PROC COMPARE procedure.

Usage

```
## S3 method for class 'data.frame'
compare(x, y, by = NULL, by.x = by, by.y = by,
        control = NULL, ...)
```

```
## S3 method for class 'compare.data.frame'
print(x, ...)
```

Arguments

x, y	A data.frame to compare
by, by.x, by.y	Which variables are IDs to merge the two data.frames? If set to "row.names", merging will occur over the row.names. If set to NULL (default), merging will occur row-by-row.
control	A list of control parameters from comparison.control .
...	Other arguments, passed to comparison.control when appropriate.

Author(s)

Ethan Heinzen, adapted from code from Andrew Hanson

See Also

[summary.compare.data.frame](#), [n.diff](#), [n.diff.obs](#)

Examples

```
df1 <- data.frame(id = paste0("person", 1:3), a = c("a", "b", "c"),
                 b = c(1, 3, 4), c = c("f", "e", "d"),
                 row.names = paste0("rn", 1:3), stringsAsFactors = FALSE)
df2 <- data.frame(id = paste0("person", 3:1), a = c("c", "b", "a"),
                 b = c(1, 3, 4), d = paste0("rn", 1:3),
                 row.names = paste0("rn", c(1,3,2)), stringsAsFactors = FALSE)
summary(compare(df1, df2))
summary(compare(df1, df2, by = "id"))
summary(compare(df1, df2, by = "row.names"))
```

comparison.control *Control settings for 'compare' function*

Description

Control tolerance definitions for the `compare.data.frame` function.

Usage

```
comparison.control(tol.logical = "none", tol.num = c("absolute",
  "percent", "pct"), tol.num.val = sqrt(.Machine$double.eps),
  int.as.num = FALSE, tol.char = c("none", "trim", "case", "both"),
  tol.factor = c("none", "levels", "labels"), factor.as.char = FALSE,
  tol.date = "absolute", tol.date.val = 0, tol.other = "none",
  tol.vars = "none", ...)
```

Arguments

<code>tol.logical</code> , <code>tol.num</code> , <code>tol.char</code> , <code>tol.factor</code> , <code>tol.date</code> , <code>tol.other</code>	A function or one of the shortcut character strings, denoting the tolerance function to use for a given data type. See "details", below.
<code>tol.num.val</code>	Numeric; maximum value of differences allowed in numerics (fed to the function given in <code>tol.num</code>).
<code>int.as.num</code>	Logical; should integers be coerced to numeric before comparison? Default FALSE.
<code>factor.as.char</code>	Logical; should factors be coerced to character before comparison? Default FALSE.
<code>tol.date.val</code>	Numeric; maximum value of differences allowed in dates (fed to the function given in <code>tol.date</code>).
<code>tol.vars</code>	Either "none" (the default), denoting that variable names are to be matched as-is, or a character vector denoting equivalence classes for characters in the variable names. See "details", below.
...	Other arguments (not in use at this time).

Details

The following character strings are accepted:

- `tol.logical = "none"`: compare logicals exactly as they are.
- `tol.num = "absolute"`: compare absolute differences in numerics.
- `tol.num = "percent"`, `tol.num = "pct"` compare percent differences in numerics.
- `tol.char = "none"`: compare character strings exactly as they are.
- `tol.char = "trim"`: left-justify and trim all trailing white space.
- `tol.char = "case"`: allow differences in upper/lower case.

- `tol.char = "both"`: combine "trim" and "case".
- `tol.factor = "none"`: match both character labels and numeric levels.
- `tol.factor = "levels"`: match only the numeric levels.
- `tol.factor = "labels"`: match only the labels.
- `tol.date = "absolute"`: compare absolute differences in dates.
- `tol.other = "none"`: expect objects of other classes to be exactly identical.

`tol.vars`: If not set to "none" (the default), the `tol.vars` argument is a character vector denoting equivalence classes for the characters in the variable names. A single character in this vector means to replace that character with ".". All other strings in this vector are split by character and replaced by the first character in the string.

E.g., a character vector `c(".", "aA", " ")` would denote that the dot and underscore are equivalent (to be translated to a dot), that "a" and "A" are equivalent (to be translated to "a"), and that spaces should be removed.

The special character string "case" in this vector is the same as specifying `paste0(letters, LETTERS)`.

Value

A list containing the necessary parameters for the `compare.data.frame` function.

Author(s)

Ethan Heinzen

See Also

[compare.data.frame](#), [comparison.tolerances](#)

Examples

```
cnt1 <- comparison.control(
  tol.num = "pct",      # calculate percent differences
  tol.vars = c("case", # ignore case
               ".",    # set all underscores to dots.
               "e")    # remove all letter e's
)
```

`comparison.tolerances` *Comparison tolerances*

Description

Internal functions defining tolerances for the `comparison.control` function. To create your own tolerance definitions, see the vignette.

Usage

```
tol.NA(x, y, idx)

tol.num.absolute(x, y, tol)

tol.num.percent(x, y, tol)

tol.num.pct(x, y, tol)

tol.factor.none(x, y)

tol.factor.levels(x, y)

tol.factor.labels(x, y)

tol.char.both(x, y)

tol.char.case(x, y)

tol.char.trim(x, y)

tol.char.none(x, y)

tol.date.absolute(x, y, tol)

tol.logical.none(x, y)

tol.other.none(x, y)
```

Arguments

<code>x, y</code>	vectors of the appropriate lengths and types.
<code>idx</code>	A logical vector of appropriate length.
<code>tol</code>	A numeric tolerance

Details

`tol.NA` takes as differences between two vectors any elements which are NA in one but not the other, or which are non-NA in both and TRUE in `idx`. It is useful for handling NAs in custom tolerance functions.

Value

A logical vector of length equal to that of `x` and `y`, where TRUE denotes a difference between `x` and `y`, and FALSE denotes no difference between `x` and `y`.

Author(s)

Ethan Heinzen

See Also

[comparison.control](#), [compare.data.frame](#)

diffs

Extract differences

Description

Extract differences, number of differences, or number of not-shared observations from a compare object.

Usage

```
n.diff.obs(object, ...)

## S3 method for class 'compare.data.frame'
n.diff.obs(object, ...)

## S3 method for class 'summary.compare.data.frame'
n.diff.obs(object, ...)

n.diffs(object, ...)

## S3 method for class 'compare.data.frame'
n.diffs(object, ...)

## S3 method for class 'summary.compare.data.frame'
n.diffs(object, ...)

diffs(object, ...)

## S3 method for class 'compare.data.frame'
diffs(object, vars = NULL, ...,
      by.var = FALSE)

## S3 method for class 'summary.compare.data.frame'
diffs(object, vars = NULL, ...,
      by.var = FALSE)
```

Arguments

object	An object of class <code>compare.data.frame</code> or <code>summary.compare.data.frame</code> .
...	Other arguments (not in use at this time).
vars	A character vector of variable names to subset the results to.
by.var	Logical: should the number of differences by variable be reported, or should all differences be reported (the default).

Author(s)

Ethan Heinzen

See Also[compare.data.frame.summary.compare.data.frame](#)

`formulize`*formulize*

Description

A shortcut to generate one-, two-, or many-sided formulas from vectors of variable names.

Usage

```
formulize(y = "", x = "", ..., data = NULL)
```

Arguments

`y`, `x`, ... Character vectors, names, or calls to be collapsed (by "+") and put left-to-right in the formula. If `data` is supplied, these can also be numeric, denoting which column name to use. See examples.

`data` An R object with non-null column names.

Author(s)

Ethan Heinzen

See Also[reformulate](#)**Examples**

```
## two-sided formula
formulize("y", c("x1", "x2", "x3"))

## one-sided formula
formulize(x = c("x1", "x2", "x3"))

## multi-sided formula
formulize("y", c("x1", "x2", "x3"), c("z1", "z2"), "w1")

## can use numerics for column names
data(mockstudy)
formulize(y = 1, x = 2:4, data = mockstudy)
```

```
## mix and match
formulize(1, c("x1", "x2", "x3"), data = mockstudy)

## get an interaction
formulize("y", c("x1*x2", "x3"))

## no intercept
formulize("y", "x1 - 1")
formulize("y", c("x1", "x2", "-1"))

## use in an lm
form <- formulize(2, 3:4, data = mockstudy)
summary(lm(form, data = mockstudy))

## using non-syntactic names or calls (like reformulate example)
formulize(as.name("+-"), c("`P/E`", "`% Growth`"))

f <- Surv(ft, case) ~ a + b
formulize(f[[2]], f[[3]])
```

freq.control

Control settings for freqlist function

Description

Control test and summary settings for the [freqlist](#) function.

Usage

```
freq.control(sparse = FALSE, single = FALSE, dupLabels = FALSE,
  digits.count = 0L, digits.pct = 2L, ..., digits = NULL)
```

Arguments

sparse	a logical value indicating whether to keep rows with counts of zero. The default is FALSE (drop zero-count rows).
single	logical, indicating whether to collapse results created using a strata variable into a single table for printing
dupLabels	logical: should labels which are the same as the row above be printed? The default (FALSE) more closely approximates PROC FREQ output from SAS, where a label carried down from the row above is left blank.
digits.count	Number of decimal places for count values.
digits.pct	Number of decimal places for percents.
...	additional arguments.
digits	A deprecated argument

Value

A list with settings to be used within the `freqlist` function.

Author(s)

Ethan Heinzen

See Also

[freqlist](#), [summary.freqlist](#)

freqlist	<i>freqlist</i>
----------	-----------------

Description

Approximate the output from SAS's PROC FREQ procedure when using the `/list` option of the TABLE statement.

Usage

```
freqlist(object, ...)
```

```
## S3 method for class 'table'
freqlist(object, na.options = c("include", "showexclude",
  "remove"), strata = NULL, labelTranslations = NULL, control = NULL,
  ...)
```

```
## S3 method for class 'formula'
freqlist(formula, data, subset, na.action,
  strata = NULL, labelTranslations = NULL, control = NULL, addNA,
  exclude, drop.unused.levels, ...)
```

Arguments

object	An R object, usually of class "table" or class "xtabs"
...	additional arguments. In the formula method, these are passed to the table method. These are also passed to freq.control
na.options	a character string indicating how to handling missing values: "include" (include values with NAs in counts and percentages), "showexclude" (show NAs but exclude from cumulative counts and all percentages), "remove" (remove values with NAs); default is "include".
strata	(formerly <code>groupBy</code>) an optional character string specifying a variable(s) to use for grouping when calculating cumulative counts and percentages. summary.freqlist will also separate by grouping variable for printing. Note that this is different from <code>modelsum</code> and <code>tableby</code> , which take bare column names (and only one, at that!)

labelTranslations

an optional character string (or list) of labels to use for variable levels when summarizing. Names will be matched appropriately.

control

control parameters to handle optional settings within freqlist. See [freq.control](#)

formula, data, subset, na.action, addNA, exclude, drop.unused.levels

Arguments passed to [xtabs](#). Note that addNA= only works in R >= 3.4.0.

Value

An object of class c("freqlist", "arsenal_table")

Author(s)

Tina Gunderson, with revisions by Ethan Heinzen

See Also

[arsenal_table](#), [summary.freqlist](#), [freq.control](#), [table](#), [xtabs](#), [kable](#)

Examples

```
# load mockstudy data
data(mockstudy)
tab.ex <- table(mockstudy[c("arm", "sex", "mdquality.s")], useNA = "ifany")
noby <- freqlist(tab.ex, na.options = "include")
summary(noby)
withby <- freqlist(tab.ex, strata = c("arm","sex"), na.options = "showexclude")
summary(withby)
```

freqlist.internal

Helper functions for freqlist

Description

A set of helper functions for [freqlist](#).

Usage

```
is.freqlist(x)
```

```
is.summary.freqlist(x)
```

Arguments

x A freqlist object.

See Also

[merge.freqlist](#), [arsenal_table](#)

internal.functions *Split a string into pieces intelligently*

Description

Split a string into pieces intelligently

Usage

```
smart.split(string, width = Inf, min.split = -Inf)
```

```
replace2(x, list, values)
```

Arguments

string	A character vector
width	Either Inf or NULL to specify no splitting, or a positive integer giving the largest allowed string length.
min.split	Either -Inf or NULL to specify no lower bound on the string length, or a positive integer giving the minimum string length.
x	vector
list	an index vector
values	replacement values

Value

For `smart.split`, a list of the same length as `string`, with each element being the "intelligently" split string.

For `replace2`, a vector with the proper values replaced.

See Also

[replace](#)

keep.labels *Keep Labels*

Description

Keep the 'label' attribute on an R object when subsetting.

Usage

```
keep.labels(x, ...)

## S3 method for class 'data.frame'
keep.labels(x, ...)

## Default S3 method:
keep.labels(x, ...)

## S3 method for class 'keep_labels'
x[...]
```

Arguments

x An R object
 ... Other arguments (not in use at this time).

Value

A copy of x with a "keep labels" class appended on.

Author(s)

Ethan Heinzen

See Also

[labels](#)

labels

Labels

Description

Assign and extract the 'label' attribute on an R object. `set_labels` is the same as `labels(x) <- value` but returns x for use in a pipe chain. `set_attr` is the same as `attr(x, which) <- value` but returns x for use in a pipe chain.

Usage

```
## S3 method for class 'data.frame'
labels(object, ...)

## S3 method for class 'keep_labels'
labels(object, ...)

labels(x) <- value
```

```

## S3 replacement method for class 'keep_labels'
labels(x) <- value

## Default S3 replacement method:
labels(x) <- value

## S3 replacement method for class 'data.frame'
labels(x) <- value

set_labels(x, value)

set_attr(x, which, value)

```

Arguments

...	Other arguments (not in use at this time).
x, object	An R object.
value	A vector or list containing labels to assign. Labels are assigned based on names, if available; otherwise, they're assigned in order. Can pass NULL to remove all labels.
which	See attr<-

Details

The [data.frame](#) methods put labels on and extract labels from the *columns* of object.

Value

The labels of object, or object with new labels.

Author(s)

Ethan Heinzen

See Also

[keep.labels](#)

mdy.Date

Convert numeric dates to Date object, and vice versa

Description

Convert numeric dates for month, day, and year to Date object, and vice versa.

Usage

```
mdy.Date(month, day, year, yearcut = 120)
```

```
Date.mdy(date)
```

```
is.Date(x)
```

Arguments

month	integer, month (1-12).
day	integer, day of the month (1-31, depending on the month).
year	integer, either 2- or 4-digit year. If two-digit number, will add 1900 onto it, depending on range.
yearcut	cutoff for method to know if to convert to 4-digit year.
date	A date value.
x	An object.

Details

Test if an object is a date.

More work may need to be done with yearcut and 2-digit years. Best to give a full 4-digit year.

Value

mdy.Date returns a Date object, and Date.mdy returns a list with integer values for month, day, and year. is.Date returns a single logical value.

See Also

[Date](#), [DateTimeClasses](#)

Examples

```
mdy.Date(9, 2, 2013)
```

```
tmp <- mdy.Date(9, 2, 2013)  
Date.mdy(tmp)
```

```
is.Date(tmp)
```

`mockstudy`*Mock study data for examples*

Description

Mock clinical study data for examples to test data manipulation and statistical functions. The function `muck_up_mockstudy()` is used in examples for [compare.data.frame](#).

Usage

```
mockstudy
```

```
muck_up_mockstudy()
```

Format

A data frame with 1499 observations on the following 15 variables:

`case` a numeric identifier-patient ID

`age` age in years

`arm` treatment arm divided into 3 groups, character string

`sex` a factor with levels Male Female

`race` self-reported race/ethnicity, character string

`fu.time` survival or censoring time in years

`fu.stat` censoring status; 1=censor, 2=death

`ps` integer, ECOG performance score

`hgb` numeric, hemoglobin count

`bmi` numeric, body mass index, kg/m²

`alk.phos` numeric, alkaline phosphatase

`ast` numeric, aspartate transaminase

`mdquality.s` integer, LASA QOL 0=Clinically Deficient, 1=Not Clinically Deficient

`age.ord` an ordered factor split of age, with levels 10-19 < 20-29 < 30-39 < 40-49 < 50-59 < 60-69 < 70-79 < 80-89

Examples

```
data(mockstudy)
```

```
str(mockstudy)
```

modelsum	<i>Fit models over each of a set of independent variables with a response variable</i>
----------	--

Description

Fit and summarize models for each independent (x) variable with a response variable (y), with options to adjust by variables for each model.

Usage

```
modelsum(formula, family = "gaussian", data, adjust = NULL,
  na.action = NULL, subset = NULL, weights = NULL, strata,
  control = NULL, ...)
```

Arguments

formula	an object of class formula ; a symbolic description of the variables to be modeled. See "Details" for more information.
family	similar mechanism to glm , where the model to be fit is driven by the family. Options include: binomial, gaussian, survival, poisson, negbin, and ordinal. These can be passed as a string, as a function, or as a list resulting from a call to one of the functions. See modelsum.family for details on survival and ordinal families.
data	an optional <code>data.frame</code> , list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>modelsum</code> is called.
adjust	an object of class formula , listing variables to adjust by in all models. Specify as a one-sided formula, like: <code>~Age+ Sex</code> .
na.action	a function which indicates what should happen when the data contain NAs. The default (NULL) is to use the defaults of lm , glm , or coxph , depending on the family specifications.
subset	an optional vector specifying a subset of observations (rows of data) to be used in the results. If <code>strata</code> is missing, this works as vector of logicals or an index; otherwise, it should be a logical vector.
weights	an optional vector specifying the weights to apply to each data observation (rows of data)
strata	a vector of strata to separate model summaries by an additional group.
control	control parameters to handle optional settings within <code>modelsum</code> . Arguments for <code>modelsum.control</code> can be passed to <code>modelsum</code> via the <code>...</code> argument, but if a control object and <code>...</code> arguments are both supplied, the latter are used. See modelsum.control for other details.
...	additional arguments to be passed to internal <code>modelsum</code> functions.

Value

An object with class `c("modelsum", "arsenal_table")`

Author(s)

Jason Sinnwell, Patrick Votruba, Beth Atkinson, Gregory Dougherty, and Ethan Heinzen, adapted from SAS Macro of the same name

See Also

[arsenal_table](#), [modelsum.control](#), [summary.modelsum](#), [formulize](#)

Examples

```
data(mockstudy)

tab1 <- modelsum(bmi ~ sex + age, data = mockstudy)
summary(tab1, text = TRUE)

tab2 <- modelsum(alk.phos ~ arm + ps + hgb, adjust = ~ age + sex,
                family = "gaussian", data = mockstudy)
summary(tab2, text = TRUE)

summary(tab2, show.intercept = FALSE, text = TRUE)

tab2.df <- as.data.frame(tab2)

tab2.df[1:5,]
```

<code>modelsum.control</code>	<i>Control settings for modelsum function</i>
-------------------------------	---

Description

Control test and summary settings for `modelsum` function.

Usage

```
modelsum.control(digits = 3L, digits.ratio = 3L, digits.p = 3L,
  format.p = TRUE, show.adjust = TRUE, show.intercept = TRUE,
  conf.level = 0.95, ordinal.stats = c("OR", "CI.lower.OR",
  "CI.upper.OR", "p.value", "Nmiss"), binomial.stats = c("OR",
  "CI.lower.OR", "CI.upper.OR", "p.value", "concordance", "Nmiss"),
  gaussian.stats = c("estimate", "std.error", "p.value", "adj.r.squared",
  "Nmiss"), poisson.stats = c("RR", "CI.lower.RR", "CI.upper.RR",
  "p.value", "Nmiss"), negbin.stats = c("RR", "CI.lower.RR",
  "CI.upper.RR", "p.value", "Nmiss"), survival.stats = c("HR",
  "CI.lower.HR", "CI.upper.HR", "p.value", "concordance", "Nmiss"),
  stat.labels = list(), ...)
```

Arguments

<code>digits</code>	Numeric, denoting the number of digits after the decimal point for beta coefficients and standard errors.
<code>digits.ratio</code>	Numeric, denoting the number of digits after the decimal point for ratios, e.g. OR, RR, HR.
<code>digits.p</code>	Numeric, denoting the number of digits for p-values. See "Details", below.
<code>format.p</code>	Logical, denoting whether to format p-values. See "Details", below.
<code>show.adjust</code>	Logical, denoting whether to show adjustment terms.
<code>show.intercept</code>	Logical, denoting whether to show intercept terms.
<code>conf.level</code>	Numeric, giving the confidence level.
<code>ordinal.stats</code> , <code>binomial.stats</code> , <code>survival.stats</code> , <code>gaussian.stats</code> , <code>poisson.stats</code> , <code>negbin.stats</code>	Character vectors denoting which stats to show for the various model types.
<code>stat.labels</code>	A named list of labels for all the stats used above.
<code>...</code>	Other arguments (not in use at this time).

Details

If `format.p` is FALSE, `digits.p` denotes the number of significant digits shown. The p-values will be in exponential notation if necessary. If `format.p` is TRUE, `digits.p` will determine the number of digits after the decimal point to show. If the p-value is less than the resulting number of places, it will be formatted to show so.

Value

A list with settings to be used within the `modelsum` function.

See Also

[modelsum](#), [summary.modelsum](#)

<code>modelsum.family</code>	<i>Family functions for modelsum</i>
------------------------------	--------------------------------------

Description

A set of family functions for [modelsum](#).

Usage

```
survival()

ordinal(method = c("logistic", "probit", "loglog", "cloglog", "cauchit"))

negbin(link = c("log", "identity", "sqrt"))
```

Arguments

method See MASS::polr.
link See MASS::glm.nb.

Value

A list, in particular with element family.

See Also

[family](#), [coxph](#), [polr](#)

modelsum.internal *Helper functions for modelsum*

Description

A set of helper functions for [modelsum](#).

Usage

```
is.modelsum(x)  
  
is.summary.modelsum(x)  
  
na.modelsum(object, ...)
```

Arguments

x A modelsum object.
object A data.frame resulting from evaluating a modelsum formula.
... Other arguments, or a vector of indices for extracting.

Value

na.modelsum returns a subsetted version of object (with attributes).

See Also

[arsenal_table](#)

NA.operations

Some functions to handle NAs

Description

allNA tests if all elements are NA, and includeNA sets the NAs in a character vector or factor to an explicit label.

Usage

```
allNA(x)

includeNA(x, label, ...)

## S3 method for class 'factor'
includeNA(x, label = "(Missing)", first = FALSE, ...)

## Default S3 method:
includeNA(x, label = "(Missing)", ...)
```

Arguments

x	An object
label	A character string denoting the label to set NAs to.
...	Other arguments (not in use at this time).
first	Logical; should the new label be the first level?

Author(s)

Ethan Heinzen

See Also

[is.na](#), [anyNA](#)

padjust

Adjust P-values for Multiple Comparisons

Description

Adjust P-values for Multiple Comparisons

Usage

```

padjust(p, method, n, ...)

## Default S3 method:
padjust(p, method, n, ...)

## S3 method for class 'tableby'
padjust(p, method, n,
  suffix = " (adjusted for multiple comparisons)", ...)

## S3 method for class 'summary.tableby'
padjust(p, method, n,
  suffix = " (adjusted for multiple comparisons)", ...)

```

Arguments

p	An object.
method	correction method. Can be abbreviated.
n	number of comparisons, must be at least length(p); only set this (to non-default) when you know what you are doing!
...	Other arguments.
suffix	A suffix to add to the footnotes indicating that the tests were adjusted.

See Also

[p.adjust](#), [modpval.tableby](#), [tests.tableby](#)

paired	<i>Summary Statistics of a Set of Independent Variables Paired Across Two Timepoints</i>
--------	--

Description

Summarize one or more variables (x) by a paired time variable (y). Variables on the right side of the formula, i.e. independent variables, are summarized by the two time points on the left of the formula. Optionally, an appropriate test is performed to test the distribution of the independent variables across the time points.

Usage

```

paired(formula, data, id, na.action, subset = NULL, strata,
  control = NULL, ...)

```

Arguments

formula	an object of class formula of the form <code>time ~ var1 + ...</code> . See "Details" for more information.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the function is called.
id	The vector giving IDs to match up data for the same subject across two time-points.
na.action	a function which indicates what should happen when the data contain NAs. The default is <code>na.paired("in.both")</code> . See na.paired for more details
subset	an optional vector specifying a subset of observations (rows of data) to be used in the results. Works as vector of logicals or an index.
strata	a vector of strata to separate summaries by an additional group.
control	control parameters to handle optional settings within <code>paired</code> . Two aspects of <code>paired</code> are controlled with these: test options of RHS variables and x variable summaries. Arguments for <code>paired.control</code> can be passed to <code>paired</code> via the <code>...</code> argument, but if a control object and <code>...</code> arguments are both supplied, the latter are used. See paired.control for more details.
...	additional arguments to be passed to internal <code>paired</code> functions or paired.control .

Details

Do note that this function piggybacks off of [tableby](#) quite heavily, so there is no `summary.paired` function (for instance).

These tests are accepted:

- `paired.t`: a paired [t-test](#).
- `mcnemar`: [McNemar's test](#).
- `signed.rank`: a [signed rank test](#).
- `sign.test`: a sign test.
- `notest`: no test is performed.

Value

An object with class `c("paired", "tableby", "arsenal_table")`

Author(s)

Jason Sinnwell, Beth Atkinson, Ryan Lennon, and Ethan Heinzen

See Also

[arsenal_table](#), [paired.control](#), [tableby](#), [formulize](#)

paired.control	<i>Control settings for paired function</i>
----------------	---

Description

Control test and summary settings for the [paired](#) function.

Usage

```
paired.control(test = TRUE, diff = TRUE, test.pname = NULL,
  numeric.test = "paired.t", cat.test = "mcnemar",
  ordered.test = "signed.rank", date.test = "paired.t",
  numeric.stats = c("Nmiss", "meansd", "range"), cat.stats = c("Nmiss",
  "countpct"), ordered.stats = c("Nmiss", "countpct"),
  date.stats = c("Nmiss", "median", "range"), stats.labels = list(Nmiss
  = "N-Miss", Nmiss2 = "N-Miss", meansd = "Mean (SD)", medianq1q3 =
  "Median (Q1, Q3)", q1q3 = "Q1, Q3", range = "Range", countpct =
  "Count (Pct)"), digits = 3L, digits.count = 0L, digits.p = 3L,
  format.p = TRUE, conf.level = 0.95, mcnemar.correct = TRUE,
  signed.rank.exact = NULL, signed.rank.correct = TRUE, ...)
```

Arguments

test	logical, telling paired whether to perform tests of x variables across time points.
diff	logical, telling paired whether to calculate a column of differences between time points.
test.pname	character string denoting the p-value column name in summary.tableby . Modifiable also with modpval.tableby .
numeric.test	name of test for numeric RHS variables in paired: paired.t, signed.rank, sign.test
cat.test	name of test for categorical variables: mcnemar
ordered.test	name of test for ordered variables: signed.rank, sign.test
date.test	name of test to perform for date variables: paired.t, signed.rank, sign.test
numeric.stats	summary statistics to include for numeric RHS variables within the levels of the group LHS variable. Options are N, Nmiss, Nmiss2, mean, meansd, median, q1q3, medianq1q3, iqr, range, medianrange, or other R built-in or user-written functions.
cat.stats	summary statistics to include for categorical RHS variables within the levels of the group LHS variable. Options are N, Nmiss, Nmiss2, count, countpct, countrowpct, countcellpct, or other R built-in or user-written functions.
ordered.stats	summary statistics to include for categorical RHS variables within the levels of the group LHS variable. Options are N, Nmiss, count, countpct, or other R built-in or user-written functions.
date.stats	stats functions to perform for Date variables: Nmiss, median, range, medianrange, q1q3, medianq1q3, or other R built-in or user-written functions.

<code>stats.labels</code>	A named list of labels for all the statistics function names, where the function name is the named element in the list and the value that goes with it is a string containing the formal name that will be printed in all printed renderings of the output, e.g., <code>list(countpct="Count (Pct)")</code> .
<code>digits</code>	Number of decimal places for numeric values.
<code>digits.count</code>	Number of decimal places for count values.
<code>digits.p</code>	Number of decimal places for p-values.
<code>format.p</code>	Logical, denoting whether to format p-values. See "Details", below.
<code>conf.level</code>	Numeric, denoting what confidence level to use for confidence intervals. (See, e.g., binomCI)
<code>mcnemar.correct</code> , <code>signed.rank.exact</code> , <code>signed.rank.correct</code>	Options for statistical tests. See wilcox.test and mcnemar.test for details.
<code>...</code>	additional arguments.

Details

All tests can be turned off by setting `test` to `FALSE`. Otherwise, tests are set to default settings in this list, or set explicitly in the formula of `tableby`.

If `format.p` is `FALSE`, `digits.p` denotes the number of significant digits shown. The p-values will be in exponential notation if necessary. If `format.p` is `TRUE`, `digits.p` will determine the number of digits after the decimal point to show. If the p-value is less than the resulting number of places, it will be formatted to show so.

Value

A list with settings to be used within the `paired` function.

Author(s)

Ethan Heinzen

See Also

[paired](#), [tableby](#), [tableby.control](#), [summary.tableby](#)

<code>paired.internal</code>	<i>Helper functions for <code>paired</code></i>
------------------------------	---

Description

A set of helper functions for [paired](#).

Usage

```
na.paired(missings = c("in.both", "fill", "asis"))
```

Arguments

missings A character string denoting which action to take. See "Details", below.
 ... Other arguments, or a vector of indices for extracting.

Details

All methods subset out any NA time points or IDs. "in.both" (the default) subsets the data.frame to individuals who appear at both time points. "fill" adds explicit missings for the people missing second time points. "asis" does nothing to add or remove missings.

Value

na.paired returns a function used to subset data.frames in [paired](#).

See Also

[tableby.internal](#)

summary.compare	<i>The summary method for a compare.data.frame object</i>
-----------------	---

Description

Print a more detailed output of the [compare.data.frame](#) object.

Usage

```
## S3 method for class 'compare.data.frame'
summary(object, ..., show.attrs = FALSE,
         max.print.vars = NA, max.print.obs = NA, max.print.diff = 10,
         max.print.attrs = NA)
```

```
## S3 method for class 'summary.compare.data.frame'
print(x, ..., format = "pandoc")
```

Arguments

object An object of class "compare.data.frame", as made by the [compare.data.frame](#) S3 method.
 ... Other arguments. In print, these are passed to [kable](#).
 show.attrs Logical, denoting whether to show the actual attributes which are different. For (e.g.) factors with lots of levels, this can make the tables quite wide, so this feature is FALSE by default.
 max.print.vars, max.print.obs, max.print.diff, max.print.attrs
 Integers denoting the maximum number of differences to report for each of the three tables. Passing NA will print all differences.

x An object returned by the `summary.compare.data.frame` function.

format Passed to `kable`: the format for the table. The default here is "pandoc". To use the default in `kable`, pass `NULL`.

Value

An object of class "summary.compare.data.frame" is returned.

See Also

`compare.data.frame`

`summary.freqlist` *summary.freqlist*

Description

Summarize the `freqlist` object.

Usage

```
## S3 method for class 'freqlist'
summary(object, ..., labelTranslations = NULL,
         title = NULL)
```

```
## S3 method for class 'summary.freqlist'
as.data.frame(x, ..., list.ok = FALSE)
```

Arguments

`object` an object of class `freqlist`

`...` For `summary.freqlist`, these are passed to `as.data.frame.freqlist` (and hence to `freq.control`). For the print method, these are additional arguments passed to the `kable` function.

`labelTranslations` A named list (or vector) where the name is the label in the output to be replaced in the pretty rendering by the character string value for the named element of the list, e.g., `list(age = "Age(Years)", meansd = "Mean(SD)")`.

`title` Title for the table, defaulting to `NULL` (no title)

`x` An object of class `summary.freqlist`.

`list.ok` If the object has multiple by-variables, is it okay to return a list of `data.frames` instead of a single `data.frame`? If `FALSE` but there are multiple by-variables, a warning is issued.

Value

An object of class "summary.freqlist" (invisibly for the print method).

Author(s)

Tina Gunderson, with major revisions by Ethan Heinzen

See Also

[freqlist](#), [table](#), [xtabs](#), [kable](#)

Examples

```
# load mockstudy data
data(mockstudy)
tab.ex <- table(mockstudy[c("arm", "sex", "mdquality.s")], useNA = "ifany")
noby <- freqlist(tab.ex, na.options = "include")
summary(noby)
withby <- freqlist(tab.ex, strata = c("arm", "sex"), na.options = "showexclude")
summary(withby)
summary(withby, dupLabels = TRUE)
```

summary.modelsum	<i>Summarize a modelsum object.</i>
------------------	-------------------------------------

Description

Format the information in object as a table using Pandoc coding or plain text, and cat it to stdout.

Usage

```
## S3 method for class 'modelsum'
summary(object, ..., labelTranslations = NULL,
        text = FALSE, title = NULL, term.name = "")

## S3 method for class 'summary.modelsum'
as.data.frame(x, ..., text = x$text,
             term.name = x$term.name, width = NULL, min.split = NULL,
             list.ok = FALSE)
```

Arguments

object A [modelsum](#) object.

... For `summary.modelsum`, other arguments passed to `as.data.frame.modelsum`. For `as.data.frame.summary.modelsum`, "width" and "min.split" are passed to `smart.split`. For printing the summary object, these are passed to both `as.data.frame.summary.modelsum` and [kable](#).

labelTranslations A named list (or vector) where the name is the label in the output to be replaced in the pretty rendering by the character string value for the named element of the list, e.g., `list(age = "Age(Years)", meansd = "Mean(SD)")`.

text	An argument denoting how to print the summary to the screen. Default is FALSE (show markdown output). TRUE and NULL output a text-only version, with the latter avoiding all formatting. "html" uses the HTML tag instead of the markdown formatting, and "latex" uses the LaTeX command \textbf.
title	Title for the table, defaulting to NULL (no title)
term.name	A character vector denoting the column name for the "terms" column. It should be the same length as the number of tables or less (it will get recycled if needed). The special value TRUE will use the y-variable's label for each table.
x	An object of class "summary.modelsum".
width	Passed to smart.split for formatting of the "term" column.
min.split	Passed to smart.split for formatting of the "term" column.
list.ok	If the object has multiple by-variables, is it okay to return a list of data.frames instead of a single data.frame? If FALSE but there are multiple by-variables, a warning is issued.

Value

An object of class "summary.modelsum"

Author(s)

Ethan Heinzen, based on code originally by Greg Dougherty

See Also

[modelsum](#), [as.data.frame.modelsum](#)

summary.tableby	<i>The summary method for a tableby object</i>
-----------------	--

Description

The summary method for a [tableby](#) object, which is a pretty rendering of a [tableby](#) object into a publication-quality results table in R Markdown, and can render well in text-only.

Usage

```
## S3 method for class 'tableby'
summary(object, ..., labelTranslations = NULL,
  text = FALSE, title = NULL, pfootnote = FALSE, term.name = "")

## S3 method for class 'summary.tableby'
as.data.frame(x, ..., text = x$text,
  pfootnote = x$pfootnote, term.name = x$term.name, width = NULL,
  min.split = NULL, list.ok = FALSE)
```

Arguments

object	An object of class "tableby", made by the tableby function.
...	For <code>summary.tableby</code> , other arguments passed to as.data.frame.tableby . For printing the summary object, these are passed to both <code>as.data.frame.summary.tableby</code> and kable .
labelTranslations	A named list (or vector) where the name is the label in the output to be replaced in the pretty rendering by the character string value for the named element of the list, e.g., <code>list(age = "Age(Years)", meansd = "Mean(SD)")</code> .
text	An argument denoting how to print the summary to the screen. Default is FALSE (show markdown output). TRUE and NULL output a text-only version, with the latter avoiding all formatting. "html" uses the HTML tag <code></code> instead of the markdown formatting, and "latex" uses the LaTeX command <code>\textbf</code> .
title	Title for the table, defaulting to NULL (no title)
pfootnote	Logical, denoting whether to put footnotes describing the tests used to generate the p-values.
term.name	A character vector denoting the column name for the "terms" column. It should be the same length as the number of tables or less (it will get recycled if needed). The special value TRUE will use the y-variable's label for each table.
x	An object of class "summary.tableby".
width	Passed to smart.split for formatting of the "term" column.
min.split	Passed to smart.split for formatting of the "term" column.
list.ok	If the object has multiple by-variables, is it okay to return a list of data.frames instead of a single data.frame? If FALSE but there are multiple by-variables, a warning is issued.

Value

An object of class `summary.tableby`

Author(s)

Ethan Heinzen, based on code by Gregory Dougherty, Jason Sinnwell, Beth Atkinson, adapted from SAS Macros written by Paul Novotny and Ryan Lennon

See Also

[tableby.control](#), [tableby](#)

Examples

```
set.seed(100)
## make 3+ categories for response
nsubj <- 90
mdat <- data.frame(Response=sample(c(1,2,3),nsubj, replace=TRUE),
```

```

Sex=sample(c("Male", "Female"), nsubj,replace=TRUE),
Age=round(rnorm(nsubj,mean=40, sd=5)),
HtIn=round(rnorm(nsubj,mean=65,sd=5)))

## allow default summaries on RHS variables
out <- tableby(Response ~ Sex + Age + HtIn, data=mdat)
summary(out, text=TRUE)
labels(out)
labels(out) <- c(Age="Age (years)", HtIn="Height (inches)")
summary(out, stats.labels=c(meansd="Mean-SD", q1q3 = "Q1-Q3"), text=TRUE)

```

tableby	<i>Summary Statistics of a Set of Independent Variables by a Categorical Variable</i>
---------	---

Description

Summarize one or more variables (x) by a categorical variable (y). Variables on the right side of the formula, i.e. independent variables, are summarized by the levels of a categorical variable on the left of the formula. Optionally, an appropriate test is performed to test the distribution of the independent variables across the levels of the categorical variable.

Usage

```
tableby(formula, data, na.action, subset = NULL, weights = NULL,
        strata, control = NULL, ...)
```

Arguments

formula	an object of class <code>formula</code> ; a symbolic description of the variables to be summarized by the group, or categorical variable, of interest. See "Details" for more information. To only view overall summary statistics, a one-sided formula can be used.
data	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the function is called.
na.action	a function which indicates what should happen when the data contain NAs. The default is <code>na.tableby(TRUE)</code> if there is a by-variable, and <code>na.tableby(FALSE)</code> if there is not. This schema thus includes observations with NAs in x variables, but removes those with NA in the categorical group variable and strata (if used).
subset	an optional vector specifying a subset of observations (rows of data) to be used in the results. Works as vector of logicals or an index.
weights	a vector of weights. Using weights will disable statistical tests.
strata	a vector of strata to separate summaries by an additional group.

control	control parameters to handle optional settings within tableby. Two aspects of tableby are controlled with these: test options of RHS variables across levels of the categorical grouping variable, and x variable summaries within the grouping variable. Arguments for tableby.control can be passed to tableby via the ... argument, but if a control object and ... arguments are both supplied, the latter are used. See tableby.control for more details.
...	additional arguments to be passed to internal tableby functions or tableby.control .

Details

The group variable (if any) is categorical, which could be an integer, character, factor, or ordered factor. tableby makes a simple summary of the counts within the k-levels of the independent variables on the right side of the formula. Note that unused levels are dropped.

The data argument allows data.frames with label attributes for the columns, and those labels will be used in the summary methods for the tableby class.

The independent variables are a mixture of types: categorical (discrete), numeric (continuous), and time to event (survival). These variables are split by the levels of the group variable (if any), then summarized within those levels, specific to the variable type. A statistical test is performed to compare the distribution of the independent variables across the levels of the grouping variable.

The tests differ by the independent variable type, but can be specified explicitly in the formula statement or in the control function. These tests are accepted:

- anova: analysis of variance test; the default test for continuous variables. When LHS variable has two levels, equivalent to two-sample t-test.
- kwt: Kruskal-Wallis Rank Test, optional test for continuous variables. When LHS variable has two levels, equivalent to Wilcoxon test.
- chisq: chi-square goodness of fit test for equal counts of a categorical variable across categories; the default for categorical or factor variables
- fe: Fisher's exact test for categorical variables
- trend: trend test for equal distribution of an ordered variable across a categorical variable; the default for ordered factor variables
- logrank: log-rank, the default for time-to-event variables
- notest: no test is performed.

To perform a mixture of asymptotic and rank-based tests on two different continuous variables, an example formula is: `formula = group ~ anova(age) + kwt(height)`. The test settings in `tableby.control` apply to all independent variables of a given type.

The summary statistics reported for each independent variable within the group variable can be set in [tableby.control](#).

Finally, multiple by-variables can be set using `list()`. See the examples for more details.

Value

An object with class `c("tableby", "arsenal_table")`

Author(s)

Jason Sinnwell, Beth Atkinson, Gregory Dougherty, and Ethan Heinzen, adapted from SAS Macros written by Paul Novotny and Ryan Lennon

See Also

[arsenal_table](#), [anova](#), [chisq.test](#), [tableby.control](#), [summary.tableby](#), [formulize](#)

Examples

```
data(mockstudy)
tab1 <- tableby(arm ~ sex + age, data=mockstudy)
summary(tab1, text=TRUE)

mylabels <- list(sex = "SEX", age = "Age, yrs")
summary(tab1, labelTranslations = mylabels, text=TRUE)

tab3 <- tableby(arm ~ sex + age, data=mockstudy, test=FALSE, total=FALSE,
               numeric.stats=c("median", "q1q3"), numeric.test="kwt")
summary(tab3, text=TRUE)

# multiple LHS
summary(tableby(list(arm, sex) ~ age, data = mockstudy, strata = ps), text = TRUE)

tab.test <- tableby(arm ~ kwt(age) + anova(bmi) + kwt(ast), data=mockstudy)
tests(tab.test)
```

Control settings for tableby function

Description

Control test and summary settings for the [tableby](#) function.

Usage

```
tableby.control(test = TRUE, total = TRUE, test.pname = NULL,
               cat.simplify = FALSE, numeric.simplify = FALSE,
               numeric.test = "anova", cat.test = "chisq", ordered.test = "trend",
               surv.test = "logrank", date.test = "kwt",
               numeric.stats = c("Nmiss", "meansd", "range"), cat.stats = c("Nmiss",
               "countpct"), ordered.stats = c("Nmiss", "countpct"),
               surv.stats = c("Nevents", "medSurv"), date.stats = c("Nmiss",
               "median", "range"), stats.labels = list(Nmiss = "N-Miss", Nmiss2 =
               "N-Miss", meansd = "Mean (SD)", medianrange = "Median (Range)", median =
               "Median", medianq1q3 = "Median (Q1, Q3)", q1q3 = "Q1, Q3", iqr = "IQR",
               range = "Range", countpct = "Count (Pct)", Nevents = "Events", medSurv =
```

```
"Median Survival", medTime = "Median Follow-Up"), digits = 3L,
digits.count = 0L, digits.pct = 1L, digits.p = 3L,
format.p = TRUE, conf.level = 0.95, chisq.correct = FALSE,
simulate.p.value = FALSE, B = 2000, ...)
```

Arguments

test	logical, telling tableby whether to perform tests of x variables across levels of the group variable.
total	logical, telling tableby whether to calculate a column of totals across group variable.
test.pname	character string denoting the p-value column name in <code>summary.tableby</code> . Modifiable also with <code>modpval.tableby</code> .
cat.simplify	logical, tell tableby whether to remove the first level of the categorical variable if binary. If TRUE, only the summary stats of the second level are reported. NOTE: this only simplifies to one line if <code>cat.stats</code> is only one statistic, such as <code>countpct</code> . In particular, if <code>cat.stats</code> also includes <code>Nmiss</code> and there are missings, then <code>cat.simplify</code> is ignored.
numeric.simplify	logical, tell tableby whether to condense numeric output to a single line. NOTE: this only simplifies to one line if <code>numeric.stats</code> is only one statistic, such as <code>meansd</code> . In particular, if <code>numeric.stats</code> also includes <code>Nmiss</code> and there are missings, then <code>numeric.simplify</code> is ignored.
numeric.test	name of test for numeric RHS variables in tableby: <code>anova</code> , <code>kwt</code> (Kruskal-Wallis). If no LHS variable exists, then a mean is required for a univariate test.
cat.test	name of test for categorical variables: <code>chisq</code> , <code>fe</code> (Fisher's Exact)
ordered.test	name of test for ordered variables: <code>trend</code>
surv.test	name of test for survival variables: <code>logrank</code>
date.test	name of test for date variables: <code>kwt</code>
numeric.stats	summary statistics to include for numeric RHS variables within the levels of the group LHS variable. Options are <code>N</code> , <code>Nmiss</code> , <code>Nmiss2</code> , <code>mean</code> , <code>meansd</code> , <code>median</code> , <code>q1q3</code> , <code>medianq1q3</code> , <code>iqr</code> , <code>range</code> , <code>medianrange</code> , or other R built-in or user-written functions.
cat.stats	summary statistics to include for categorical RHS variables within the levels of the group LHS variable. Options are <code>N</code> , <code>Nmiss</code> , <code>Nmiss2</code> , <code>count</code> , <code>countpct</code> , <code>countrowpct</code> , <code>countcellpct</code> , or other R built-in or user-written functions.
ordered.stats	summary statistics to include for categorical RHS variables within the levels of the group LHS variable. Options are <code>N</code> , <code>Nmiss</code> , <code>count</code> , <code>countpct</code> , or other R built-in or user-written functions.
surv.stats	summary statistics to include for time-to-event (survival) RHS variables within the levels of the group LHS variable. Options are <code>Nevents</code> , <code>medsurv</code> , <code>NeventsSurv</code> , <code>NriskSurv</code> , <code>medTime</code> , <code>rangeTime</code> .
date.stats	stats functions to perform for Date variables: <code>Nmiss</code> , <code>median</code> , <code>range</code> , <code>medianrange</code> , <code>q1q3</code> , <code>medianq1q3</code> , or other R built-in or user-written functions.


```
## allow default summaries in RHS variables, and pass control args to
## main function, to be picked up with ... when calling tableby.control
outResp <- tableby(Response ~ Sex + Age + HtIn, data=mdat, total=FALSE, test=TRUE)
outCtl <- tableby(Response ~ Sex + Age + HtIn, data=mdat,
                  control=tableby.control(total=TRUE, cat.simplify=TRUE,
                  cat.stats=c("Nmiss", "countpct"), digits=1))
summary(outResp, text=TRUE)
summary(outCtl, text=TRUE)
```

tableby.internal *Helper functions for tableby*

Description

A set of helper functions for [tableby](#).

Usage

```
is.tableby(x)

is.summary.tableby(x)

modpval.tableby(x, pdata, use.pname = FALSE)

tests(x)

## S3 method for class 'tableby'
tests(x)

na.tableby(lhs = TRUE)

## S3 method for class 'tableby'
xtfrm(x)

## S3 method for class 'tableby'
sort(x, ...)

## S3 method for class 'tableby'
Ops(e1, e2)

## S3 method for class 'tableby'
head(x, n = 6L, ...)

## S3 method for class 'tableby'
tail(x, n = 6L, ...)
```

Arguments

<code>x</code>	A tableby object.
<code>pdata</code>	A named data.frame where the first column is the by-variable names, the (optional) second is the strata value, the next is the x variable names, the next is p-values (or some test stat), and the (optional) next column is the method name.
<code>use.pname</code>	Logical, denoting whether the column name in <code>pdata</code> corresponding to the p-values should be used in the output of the object.
<code>lhs</code>	Logical, denoting whether to remove NAs from the first column of the data.frame (the "left-hand side")
<code>...</code>	Other arguments, or a vector of indices for extracting.
<code>e1, e2</code>	tableby objects, or numbers to compare them to.
<code>n</code>	A single integer. See head or tail for more details

Details

Logical comparisons are implemented for `Ops.tableby`.

Value

`na.tableby` returns a subsetting version of object (with attributes). `Ops.tableby` returns a logical vector. `xtfrm.tableby` returns the p-values (which are ordered by [order](#) to [sort](#)).

See Also

[arsenal_table](#), [sort](#), [head](#), [tail](#)

tableby.stats	<i>tableby Summary Statistics Functions</i>
---------------	---

Description

A collection of functions that will report summary statistics. To create a custom function, consider using a function with all three arguments and `...`. See the [tableby](#) vignette for an example.

Usage

```
meansd(x, na.rm = TRUE, weights = rep(1, length(x)), ...)
```

```
medianrange(x, na.rm = TRUE, weights = rep(1, length(x)), ...)
```

```
median(x, na.rm = TRUE, weights = rep(1, length(x)), ...)
```

```
range(x, na.rm = TRUE, ...)
```

```
Nevents(x, na.rm = TRUE, weights = rep(1, nrow(x)), ...)
```

```

medSurv(x, na.rm = TRUE, weights = rep(1, nrow(x)), ...)

NeventsSurv(x, na.rm = TRUE, weights = rep(1, nrow(x)), times = 1:5,
  ...)

NriskSurv(x, na.rm = TRUE, weights = rep(1, nrow(x)), times = 1:5,
  ...)

medTime(x, na.rm = TRUE, weights = rep(1, nrow(x)), ...)

q1q3(x, na.rm = TRUE, weights = rep(1, length(x)), ...)

medianq1q3(x, na.rm = TRUE, weights = rep(1, length(x)), ...)

iqr(x, na.rm = TRUE, weights = rep(1, length(x)), ...)

Nmiss(x, na.rm = TRUE, weights = rep(1, length(x)), ...)

Nmiss2(x, na.rm = TRUE, weights = rep(1, length(x)), ...)

N(x, na.rm = TRUE, weights = rep(1, length(x)), ...)

count(x, levels = NULL, na.rm = TRUE, weights = rep(1, length(x)),
  ...)

countpct(x, levels = NULL, na.rm = TRUE, weights = rep(1, length(x)),
  ...)

countrowpct(x, levels = NULL, by, by.levels = sort(unique(by)),
  na.rm = TRUE, weights = rep(1, length(x)), ...)

countcellpct(x, levels = NULL, by, by.levels = sort(unique(by)),
  na.rm = TRUE, weights = rep(1, length(x)), ...)

binomCI(x, levels = NULL, na.rm = TRUE, weights = rep(1, length(x)),
  conf.level = 0.95, ...)

rowbinomCI(x, levels = NULL, by, by.levels = sort(unique(by)),
  na.rm = TRUE, weights = rep(1, length(x)), conf.level = 0.95, ...)

```

Arguments

<code>x</code>	Usually a vector.
<code>na.rm</code>	Should NAs be removed?
<code>weights</code>	A vector of weights.
<code>...</code>	Other arguments.
<code>times</code>	A vector of times to use for survival summaries.

levels	A vector of levels that character xs should have.
by	a vector of the by-values.
by.levels	a vector of the levels of by.
conf.level	Numeric, denoting what confidence level to use for confidence intervals.

Details

Not all these functions are exported, in order to avoid conflicting NAMESPACES.

Value

Usually a vector of the appropriate numbers.

See Also

[includeNA](#), [tableby.control](#)

tableby.stats.internal

Internal tableby functions

Description

A collection of functions that may help users create custom functions that are formatted correctly.

Usage

```
as.tbstat(x, oldClass = NULL, sep = NULL, parens = NULL,
          sep2 = NULL, pct = NULL)
```

```
as.countpct(x, ...)
```

```
as.tbstat_multirow(x)
```

Arguments

x	Usually a vector.
oldClass	class(es) to add to the resulting object.
sep	The separator between x[1] and the rest of the vector.
parens	A length-2 vector denoting parentheses to use around x[2] and x[3].
sep2	The separator between x[2] and x[3].
pct	The symbol to use after percents.
...	arguments to pass to as.tbstat.

Details

as.tbstat defines a tableby statistic with its appropriate formatting.

as.countpct adds another class to as.tbstat to use different "digits" arguments. See [tableby.control](#).

as.tbstat_multirow marks an object (usually a list) for multiple-row printing.

 write2

write2

Description

Functions to output tables to a single document. (Also the S3 backbone behind the write2* functions.)

Usage

```
write2(object, file, ..., output_format)

## S3 method for class 'arsenal_table'
write2(object, file, ..., output_format = NULL)

## S3 method for class 'summary.arsenal_table'
write2(object, file, ...,
       output_format = NULL)

## S3 method for class 'compare.data.frame'
write2(object, file, ...,
       output_format = NULL)

## S3 method for class 'summary.compare.data.frame'
write2(object, file, ...,
       output_format = NULL)

## S3 method for class 'verbatim'
write2(object, file, ..., output_format = NULL)

## S3 method for class 'yaml'
write2(object, file, ..., output_format = NULL)

## S3 method for class 'code.chunk'
write2(object, file, ..., output_format = NULL)

## S3 method for class 'knitr_kable'
write2(object, file, ..., output_format = NULL)

## S3 method for class 'xtable'
write2(object, file, ..., output_format = NULL)
```

```
## S3 method for class 'character'
write2(object, file, ..., output_format = NULL)

## S3 method for class 'list'
write2(object, file, ..., append. = FALSE,
       render. = TRUE, keep.rmd = !render., output_format = NULL)

## Default S3 method:
write2(object, file, FUN = NULL, ...,
       append. = FALSE, render. = TRUE, keep.rmd = !render.,
       output_format = NULL)
```

Arguments

object	An object.
file	A single character string denoting the filename for the output document.
...	Additional arguments to be passed to FUN, rmarkdown::render, etc. One popular option is to use quiet = TRUE to suppress the command line output.
output_format	One of the following: <ol style="list-style-type: none"> 1. An output format object, e.g. rmarkdown::html_document(...). 2. A character string denoting such a format function, e.g. "html_document". In this case, the "..." are NOT passed. 3. The format function itself, e.g. rmarkdown::html_document. In this case, the "..." arguments are passed. 4. One of "html", "pdf", and "word", shortcuts implemented here. In this case, the "..." arguments are passed. 5. NULL, in which the output is HTML by default. See rmarkdown::render for details.
append.	Logical, denoting whether (if a temporary .Rmd file of the same name already exists) to append on. Used mostly for write2.list.
render.	Logical, denoting whether to render the temporary .Rmd file. Used mostly for write2.list.
keep.rmd	Logical, denoting whether to keep the intermediate .Rmd file. Used mostly for write2.list.
FUN	The summary-like or print-like function to use to generate the markdown content. Can be passed as a function or a character string. It's expected that FUN(object, ...) looks "good" when put directly in a .Rmd file.

Details

write2 is an S3 method. The default prints the object (using [print](#)) inside a section surrounded by three back ticks. See [verbatim](#) for details.

There are methods implemented for [tableby](#), [modelsum](#), and [freqlist](#), all of which use the summary function. There are also methods compatible with [kable](#), [xtable](#), and [pander_return](#).

Another option is to coerce an object using `verbatim()` to print out the results monospaced (as if they were in the terminal). To output multiple tables into a document, simply make a list of them and call the same function as before. Finally, to output code chunks to be evaluated, use `code.chunk`.

For more information, see `vignette("write2")`.

Value

object is returned invisibly, and file is written.

Author(s)

Ethan Heinzen, adapted from code from Krista Goergen

See Also

[write2word](#), [write2pdf](#), [write2html](#), [render](#), [word_document](#), [html_document](#), [pdf_document](#), [rtf_document](#), [md_document](#), [odt_document](#)

Examples

```
## Not run:
data(mockstudy)
# tableby example
tab1 <- tableby(arm ~ sex + age, data=mockstudy)
write2(tab1, tempfile(fileext = ".rtf"),
  toc = TRUE, # passed to rmarkdown::rtf_document, though in this case it's not practical
  quiet = TRUE, # passed to rmarkdown::render
  title = "My cool new title", # passed to summary.tableby
  output_format = rmarkdown::rtf_document)

## End(Not run)

write2html(list(
  "# Header 1", # a header
  code.chunk(a <- 1, b <- 2, a + b), # a code chunk
  verbatim("hi there") # verbatim output
),
  tempfile(fileext = ".html"),
  quiet = TRUE)
```

Description

Helper functions for `write2`.

Usage

```
## S3 method for class 'verbatim'
print(x, ...)

verbatim(x)

code.chunk(..., chunk.opts = "r")

## S3 method for class 'code.chunk'
print(x, ...)
```

Arguments

x	An R object to coerce to class "verbatim" or such an object to print.
...	For <code>print.verbatim</code> , other arguments passed to <code>print</code> . For <code>code.chunk</code> , either expressions or single character strings to paste into the code chunk. Not used for <code>print.code.chunk</code> .
chunk.opts	A single character string giving the code chunk options. Make sure to specify the engine!

Details

The "verbatim" class is to tell `write2` to print the object inside a section surrounded by three back ticks. The results will look like it would in the terminal (monospaced).

`code.chunk()` is to write explicit code chunks in the `.Rmd` file; it captures the call and writes it to the file, to execute upon knitting.

write2specific

write2word, write2html, write2pdf

Description

Functions to output tables to a single Word, HTML, or PDF document.

Usage

```
write2word(object, file, ...)

write2pdf(object, file, ...)

write2html(object, file, ...)
```

Arguments

object	An object.
file	A single character string denoting the filename for the output document.
...	Additional arguments to be passed to FUN, rmarkdown::render, etc. One popular option is to use quiet = TRUE to suppress the command line output.

Details

To generate the appropriate file type, the write2* functions use one of rmarkdown::word_document, rmarkdown::html_document, and rmarkdown::pdf_document to get the job done. "... " arguments are passed to these functions, too.

Value

object is returned invisibly, and file is written.

Author(s)

Ethan Heinzen, adapted from code from Krista Goergen

See Also

[write2](#)

Examples

```
## Not run:
data(mockstudy)
# tableby example
tab1 <- tableby(arm ~ sex + age, data=mockstudy)
write2html(tab1, "~/trash.html")

# freqlist example
tab.ex <- table(mockstudy[, c("arm", "sex", "mdquality.s")], useNA = "ifany")
noby <- freqlist(tab.ex, na.options = "include")
write2pdf(noby, "~/trash2.pdf")

# A more complicated example
write2word(tab1, "~/trash.doc",
  keep.md = TRUE,
  reference_docx = mystyles.docx, # passed to rmarkdown::word_document
  quiet = TRUE, # passed to rmarkdown::render
  title = "My cool new title") # passed to summary.tableby

## End(Not run)
```

`yaml`*Include a YAML header in write2*

Description

Include a YAML header in `write2`

Usage

```
yaml(...)  
  
## S3 method for class 'yaml'  
print(x, ...)  
  
## S3 method for class 'yaml'  
c(..., recursive = FALSE)  
  
is.yaml(x)
```

Arguments

<code>...</code>	For <code>yaml()</code> , arguments to be bundled into a list and passed to <code>as.yaml</code> . For <code>print.yaml()</code> , extra arguments. For <code>c.yaml()</code> , "yaml" objects to be concatenated.
<code>x</code>	An object of class "yaml".
<code>recursive</code>	Not in use at this time.

Value

A text string of class "yaml".

Author(s)

Ethan Heinzen, adapted from an idea by Brendan Broderick

See Also

[as.yaml](#), [write2](#)

Examples

```
x <- yaml(title = "My cool title", author = "Ethan P Heinzen")  
x  
y <- yaml("header-includes" = list("\\usepackage[labelformat=empty]{caption}"))  
y  
c(x, y)
```

%nin%	<i>Not in</i>
-------	---------------

Description

The not-in operator for R.

Usage

x %nin% table

Arguments

x vector or NULL: the values to be matched. [Long vectors](#) are supported.
table vector or NULL: the values to be matched against. [Long vectors](#) are not supported.

Value

The negation of [%in%](#).

Author(s)

Raymond Moore

See Also

[%in%](#)

Examples

```
1 %nin% 2:10  
c("a", "b") %nin% c("a", "c", "d")
```

Index

*Topic **datasets**

- mockstudy, [21](#)
- `[.arsenal_table (arsenal_table)`, [4](#)
- `[.keep_labels (keep_labels)`, [17](#)
- `%in%`, [51](#)
- `%nin%`, [4](#), [51](#), [51](#)

- `allNA`, [4](#)
- `allNA (NA.operations)`, [26](#)
- `anova`, [38](#), [40](#)
- `anyNA`, [26](#)
- `arsenal`, [3](#)
- `arsenal-package (arsenal)`, [3](#)
- `arsenal_table`, [4](#), [16](#), [23](#), [25](#), [28](#), [38](#), [42](#)
- `as.countpct (tableby.stats.internal)`, [44](#)
- `as.data.frame`, [22](#), [28](#), [36](#)
- `as.data.frame.freqlist`, [5](#), [32](#)
- `as.data.frame.modelsum`, [6](#), [33](#), [34](#)
- `as.data.frame.summary.freqlist (summary.freqlist)`, [32](#)
- `as.data.frame.summary.modelsum (summary.modelsum)`, [33](#)
- `as.data.frame.summary.tableby (summary.tableby)`, [34](#)
- `as.data.frame.tableby`, [7](#), [35](#)
- `as.tbstat (tableby.stats.internal)`, [44](#)
- `as.tbstat_multirow (tableby.stats.internal)`, [44](#)
- `as.yaml`, [50](#)

- `binomCI`, [30](#), [40](#)
- `binomCI (tableby.stats)`, [42](#)

- `c.yaml (yaml)`, [50](#)
- `chisq.test`, [38](#), [40](#)
- `code.chunk`, [47](#)
- `code.chunk (write2.internal)`, [47](#)
- `compare.data.frame`, [3](#), [8](#), [9](#), [10](#), [12](#), [13](#), [21](#), [31](#)
- `comparison.control`, [8](#), [9](#), [10](#), [12](#)
- `comparison.tolerances`, [10](#), [10](#)
- `count (tableby.stats)`, [42](#)
- `countcellpct (tableby.stats)`, [42](#)
- `countpct (tableby.stats)`, [42](#)
- `countrowpct (tableby.stats)`, [42](#)
- `coxph`, [22](#), [25](#)

- `data.frame`, [19](#)
- `Date`, [20](#)
- `Date.mdy`, [3](#)
- `Date.mdy (mdy.Date)`, [19](#)
- `DateTimeClasses`, [20](#)
- `diffs`, [12](#)

- `family`, [25](#)
- `formula`, [22](#), [28](#), [36](#)
- `formulize`, [3](#), [13](#), [23](#), [28](#), [38](#)
- `freq.control`, [5](#), [14](#), [15](#), [16](#), [32](#)
- `freqlist`, [3](#), [5](#), [14](#), [15](#), [15](#), [16](#), [32](#), [33](#), [46](#)
- `freqlist.internal`, [16](#)

- `glm`, [22](#)
- `glm.nb`, [25](#)

- `has_strata (arsenal_table)`, [4](#)
- `head`, [42](#)
- `head.tableby (tableby.internal)`, [41](#)
- `html_document`, [46](#), [47](#)

- `includeNA`, [4](#), [44](#)
- `includeNA (NA.operations)`, [26](#)
- `internal.functions`, [17](#)
- `iqr (tableby.stats)`, [42](#)
- `is.Date`, [4](#)
- `is.Date (mdy.Date)`, [19](#)
- `is.freqlist (freqlist.internal)`, [16](#)
- `is.modelsum (modelsum.internal)`, [25](#)
- `is.na`, [26](#)
- `is.summary.freqlist (freqlist.internal)`, [16](#)

- is.summary.modelsum
 (modelsum.internal), 25
- is.summary.tableby (tableby.internal),
 41
- is.tableby (tableby.internal), 41
- is.yaml (yaml), 50

- kable, 5, 16, 31–33, 35, 46
- keep.labels, 3, 17, 19

- labels, 5, 18, 18
- labels.arsenal_table (arsenal_table), 4
- labels<- (labels), 18
- labels<- .arsenal_table (arsenal_table),
 4
- lm, 22
- Long vectors, 51

- McNemar's test, 28
- mcnemar.test, 30
- md_document, 47
- mdy.Date, 3, 19
- meansd (tableby.stats), 42
- median (tableby.stats), 42
- medianq1q3 (tableby.stats), 42
- medianrange (tableby.stats), 42
- medSurv (tableby.stats), 42
- medTime (tableby.stats), 42
- merge, 5
- merge.arsenal_table (arsenal_table), 4
- merge.freqlist, 16
- merge.freqlist (arsenal_table), 4
- mockstudy, 4, 21
- modelsum, 3, 6, 22, 23–25, 33, 34, 46
- modelsum.control, 6, 22, 23, 23
- modelsum.family, 22, 24
- modelsum.internal, 25
- modpval.tableby, 27, 29, 39
- modpval.tableby (tableby.internal), 41
- muck_up_mockstudy (mockstudy), 21

- N (tableby.stats), 42
- n.diff.obs, 8
- n.diff.obs (diffs), 12
- n.diffs, 8
- n.diffs (diffs), 12
- na.modelsum (modelsum.internal), 25
- NA.operations, 26
- na.paired, 28
- na.paired (paired.internal), 30
- na.tableby (tableby.internal), 41
- negbin (modelsum.family), 24
- Nevents (tableby.stats), 42
- NeventsSurv (tableby.stats), 42
- nin (%nin%), 51
- Nmiss (tableby.stats), 42
- Nmiss2 (tableby.stats), 42
- NriskSurv (tableby.stats), 42

- odt_document, 47
- Ops.tableby (tableby.internal), 41
- order, 42
- ordinal (modelsum.family), 24

- p.adjust, 27
- padjust, 26
- paired, 3, 27, 29–31
- paired.control, 28, 29
- paired.internal, 30
- pander_return, 46
- pdf_document, 47
- polr, 25
- print, 46
- print.arsenal_table (arsenal_table), 4
- print.code.chunk (write2.internal), 47
- print.compare.data.frame
 (compare.data.frame), 8
- print.summary.arsenal_table
 (arsenal_table), 4
- print.summary.compare.data.frame
 (summary.compare), 31
- print.verbatim (write2.internal), 47
- print.yaml (yaml), 50

- q1q3 (tableby.stats), 42

- range (tableby.stats), 42
- reformulate, 13
- render, 46, 47
- replace, 17
- replace2 (internal.functions), 17
- rowbinomCI (tableby.stats), 42
- rtf_document, 47

- set_attr (labels), 18
- set_labels (labels), 18
- signed rank test, 28
- smart.split, 5, 33–35

- smart.split (internal.functions), 17
- sort, 42
- sort.tableby (tableby.internal), 41
- summary.compare, 31
- summary.compare.data.frame, 8, 13
- summary.freqlist, 15, 16, 32
- summary.modelsum, 6, 23, 24, 33
- summary.tableby, 29, 30, 34, 38–40
- survival (modelsum.family), 24

- table, 16, 33
- tableby, 3, 7, 28, 30, 34, 35, 36, 38, 40–42, 46
- tableby.control, 7, 30, 35, 37, 38, 38, 44, 45
- tableby.internal, 31, 41
- tableby.stats, 40, 42
- tableby.stats.internal, 44
- tail, 42
- tail.tableby (tableby.internal), 41
- tests (tableby.internal), 41
- tests.tableby, 27
- tol.char.both (comparison.tolerances), 10
- tol.char.case (comparison.tolerances), 10
- tol.char.none (comparison.tolerances), 10
- tol.char.trim (comparison.tolerances), 10
- tol.date.absolute (comparison.tolerances), 10
- tol.factor.labels (comparison.tolerances), 10
- tol.factor.levels (comparison.tolerances), 10
- tol.factor.none (comparison.tolerances), 10
- tol.logical.none (comparison.tolerances), 10
- tol.NA (comparison.tolerances), 10
- tol.num.absolute (comparison.tolerances), 10
- tol.num.pct (comparison.tolerances), 10
- tol.num.percent (comparison.tolerances), 10
- tol.other.none (comparison.tolerances), 10

- verbatim, 46, 47
- verbatim (write2.internal), 47

- wilcox.test, 30
- word_document, 47
- write2, 3, 45, 47–50
- write2.internal, 47
- write2html, 3, 47
- write2html (write2specific), 48
- write2pdf, 3, 47
- write2pdf (write2specific), 48
- write2specific, 48
- write2word, 3, 47
- write2word (write2specific), 48

- xtable, 46
- xtabs, 16, 33
- xtfrm.tableby (tableby.internal), 41

- yaml, 50