Package 'furniture'

May 8, 2026

Type Package

Title Furniture for Quantitative Scientists

Version 1.11.0

Maintainer Tyson S. Barrett <t.barrett88@gmail.com>

Description Contains four main functions (i.e., four pieces of furniture):
table1() which produces a well-formatted table of descriptive statistics common as Table 1 in research articles, tableC() which produces a well-formatted table of correlations, tableF() which provides frequency counts, and washer() which is helpful in cleaning up the data. These furniture-themed functions are designed to simplify common tasks in quantitative analysis. Other data summary and cleaning tools are also available.

Depends R (>= 2.10)

Imports knitr, dplyr (>= 0.8.0), gt, flextable

Suggests magrittr, rmarkdown, testthat, tibble

LazyData true

VignetteBuilder knitr

URL https://tysonbarrett.com/furniture/, https://github.com/TysonStanley/furniture

BugReports https://github.com/TysonStanley/furniture/issues

Encoding UTF-8

License GPL-3

RoxygenNote 7.3.3

NeedsCompilation no

Author Tyson S. Barrett [aut, cre] (ORCID: <https://orcid.org/0000-0002-2137-1391>), Emily Brignone [aut], Daniel J. Laxman [aut]

Repository CRAN

Date/Publication 2026-01-18 06:10:28 UTC

Contents

`furniture` *furniture*

Description

The furniture package offers simple functions (i.e. pieces of furniture) and an operator that are aimed at helping applied researchers explore and communicate their data as well as clean their data in a tidy way. The package follows similar semantics to the "tidyverse" packages. It contains several table functions (`table1()`) being the core one.

Details

- `table1` provides a well-formatted descriptive table often seen as table 1 in academic journals (also a version that simplifies the output is available as `simple_table1`),
- `washer` provides a simple way to clean up data where there are placeholder values, and
- `%xt%` is an operator that takes two factor variables and creates a cross tabulation and tests for significance via a chi-square test.

Table 1 is the main function in furniture. It is useful in both data exploration and data communication. With minimal cleaning, the outputted table can be put into an academic, peer reviewed journal manuscript. As such, it is very useful in exploring your data when you have a stratifying variable. For example, if you are exploring whether the means of several demographic and behavioral characteristics are related to a health condition, the health condition (i.e. "yes" or "no"; "low", "mid", or "high"; or a list of conditions) as the stratifying variable. With little code, you can test for associations and check means or counts by the stratifying variable. See the vignette for more information.

Note: furniture is meant to make life more comfortable and beautiful. In like manner, this package is designed to be "furniture" for quantitative research.

Author(s)

Maintainer: Tyson S. Barrett <t.barrett88@gmail.com> (ORCID)

Authors:

- Emily Brignone
- Daniel J. Laxman

See Also

Useful links:

- https://tysonbarrett.com/furniture/
- https://github.com/TysonStanley/furniture
- Report bugs at https://github.com/TysonStanley/furniture/issues

Examples

```
## Not run:

library(furniture)

## Table 1
data %>%
  table1(var1, var2, var3,
         splitby = ~groupvar,
         test = TRUE)

## Table F
data %>%
  tableF(var1)

## Washer
x = washer(x, 7, 8, 9)
x = washer(x, is.na, value=0)


## End(Not run)
```

---

long *Wide to Long Data Reshaping*

Description

`long()` is a wrapper of `stats::reshape()` that takes the data from a wide format to a long format. It can also handle unbalanced data (where some measures have different number of "time points").

Usage

```
long(
  data,
  ...,
  v.names = NULL,
  id = NULL,
  timevar = NULL,
  times = NULL,
  sep = ""
)
```

Arguments

data	the data.frame containing the wide format data
...	the variables that are time-varying that are to be placed in long format, needs to be in the format c("x1", "x2"), c("z1", "z2"), etc.. If the data is unbalanced (e.g., there are three time points measured for one variable but only two for another), using the placeholder variable miss, helps fix this.
v.names	a vector of the names for the newly created variables (length same as number of vectors in varying)
id	the ID variable in quotes
timevar	the column with the "time" labels
times	the labels of the timevar (default is numeric)
sep	the separating character between the wide format variable names (default is ""); e.g. "x1" and "x2" would create the variable name of "x"; only applicable if v.names

Author(s)

Tyson S. Barrett

See Also

stats::reshape() and sjmisc::to_long()

Examples

```
x1 <- runif(1000)
x2 <- runif(1000)
x3 <- runif(1000)
y1 <- rnorm(1000)
y2 <- rnorm(1000)
z  <- factor(sample(c(0,1), 1000, replace=TRUE))
a  <- factor(sample(c(1,2), 1000, replace=TRUE))
b  <- factor(sample(c(1,2,3,4), 1000, replace=TRUE))
df  <- data.frame(x1, x2, x3, y1, y2, z, a, b)

## "Balanced" Data
```

```
ldf1 <- long(df,
             c("x1", "x2"), c("y1", "y2"),
             v.names = c("x", "y"))

## "Unbalanced" Data
ldf2 = long(df,
            c("x1", "x2", "x3"), c("y1", "y2", "miss"),
            v.names = c("x", "y"))
```

`nhanes_2010` *NHANES 2009-2010*

Description

A dataset containing information on health, healthcare, and demographics of adolescents aged 18 - 30 in the United States from 2009 to 2010. This is a cleaned dataset which is only a subset of the 2009-2010 data release of the National Health and Nutrition Examination Survey (NHANES).

Usage

```
nhanes_2010
```

Format

A data frame with 1417 rows and 24 variables:

id individual ID

gen_health general health indicator with five levels

mod_active minutes of moderate activity

vig_active minutes of vigorous activity

home_meals number of home meals a week

gender gender of the individual (factor with "male" or "female")

age age of the individual in years

marijuana whether the individual has used marijuana

illicit whether the individual has used illicit drugs

rehab whether the individual has been to rehab for their drug usage

asthma whether the individual has asthma

overweight whether the individual is overweight

cancer whether the individual has cancer

low_int rating of whether the individual has low interest in things

down rating of whether the individual has felt down

sleeping rating of whether the individual has had trouble sleeping

low_energy rating of whether the individual has low energy

appetite rating of whether the individual has lost appetite

feel_bad rating of whether the individual has felt bad

no_con rating of whether the individual has felt no confidence

speak_move rating of whether the individual has trouble speaking/moving

dead rating of whether the individual has wished he/she was dead

difficulty rating of whether the individual has felt difficulty from the previous conditions

active minutes of vigorous or moderate activity

Source

https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?BeginYear=2009

`rowmeans` *Get Row Means*

Description

Does what `rowMeans()` does but without having to cbind the variables. Makes it easier to use with the tidyverse

Usage

```
rowmeans(..., na.rm = FALSE)
```

Arguments

`...`	the variables (unquoted) to be included in the row means
`na.rm`	should the missing values be ignored? default is FALSE

Value

the row means

Examples

```
## Not run:

library(furniture)
library(tidyverse)

data <- data.frame(
  x = sample(c(1,2,3,4), 100, replace=TRUE),
  y = rnorm(100),
  z = rnorm(100)
)
```

```
data2 <- data %>%
  mutate(y_z_mean = rowmeans(y, z))
data2 <- data %>%
  mutate(y_z_mean = rowmeans(y, z, na.rm=TRUE))


## End(Not run)
```

`rowmeans.n` *Get Row Means With N Missing Values Per Row*

Description

Does what `furniture::rowmeans()` does while allowing a certain number (`n`) to have missing values.

Usage

```
rowmeans.n(..., n)
```

Arguments

`...`	the variables (unquoted) to be included in the row means
`n`	the number of values without missingness required to get the row mean

Value

the row means

Examples

```
## Not run:

library(furniture)
library(dplyr)

data <- data.frame(
  x = sample(c(1,2,3,4), 100, replace=TRUE),
  y = rnorm(100),
  z = rnorm(100)
)

data2 <- mutate(data, x_y_z_mean = rowmeans.n(x, y, z, n = 2))


## End(Not run)
```

rowsums *Get Row Sums*

Description

Does what rowSums() does but without having to cbind the variables. Makes it easier to use with the tidyverse

Usage

```
rowsums(..., na.rm = FALSE)
```

Arguments

...	the variables to be included in the row sums
na.rm	should the missing values be ignored? default is FALSE

Value

the row sums

Examples

```
## Not run:

library(furniture)
library(tidyverse)

data <- data.frame(
  x = sample(c(1,2,3,4), 100, replace=TRUE),
  y = rnorm(100),
  z = rnorm(100)
)

data2 <- data %>%
  mutate(y_z_sum = rowsums(y, z))
data2 <- data %>%
  mutate(y_z_sum = rowsums(y, z, na.rm=TRUE))


## End(Not run)
```

rowsums.n *Get Row Sums With N Missing Values Per Row*

Description

Does what `furniture::rowsums()` does while allowing a certain number (`n`) to have missing values.

Usage

```
rowsums.n(..., n)
```

Arguments

`...`	the variables (unquoted) to be included in the row means
`n`	the number of values without missingness required to get the row mean

Value

the row sums

Examples

```
## Not run:

library(furniture)
library(dplyr)

data <- data.frame(
  x = sample(c(1,2,3,4), 100, replace=TRUE),
  y = rnorm(100),
  z = rnorm(100)
)

data2 <- mutate(data, x_y_z_mean = rowsums.n(x, y, z, n = 2))


## End(Not run)
```

table1 *Table 1 for Simple and Stratified Descriptive Statistics*

Description

Produces a descriptive table, stratified by an optional categorical variable, providing means/frequencies and standard deviations/percentages. It is well-formatted for easy transition to academic article or report. Can be used within the piping framework [see library(magrittr)].

Usage

```
table1(
  .data,
  ...,
  splitby = NULL,
  FUN = NULL,
  FUN2 = NULL,
  total = FALSE,
  second = NULL,
  row_wise = FALSE,
  test = FALSE,
  param = TRUE,
  header_labels = NULL,
  type = "pvalues",
  output = "text",
  rounding_perc = 1,
  digits = 1,
  var_names = NULL,
  format_number = FALSE,
  NAkeep = NULL,
  na.rm = TRUE,
  booktabs = TRUE,
  caption = NULL,
  align = NULL,
  float = "ht",
  export = NULL,
  label = NULL
)
```

Arguments

.data	the data.frame that is to be summarized
...	variables in the data set that are to be summarized; unquoted names separated by commas (e.g. age, gender, race) or indices. If indices, it needs to be a single vector (e.g. c(1:5, 8, 9:20) instead of 1:5, 8, 9:20). As it is currently, it CANNOT handle both indices and unquoted names simultaneously. Finally, any empty

	rows (where the row is NA for each variable selected) will be removed for an accurate n count.
splitby	the categorical variable to stratify (in formula form splitby = ~gender) or quoted splitby = "gender"; instead, dplyr::group_by(...) can be used within a pipe (this is the default when the data object is a grouped data frame from dplyr::group_by(...)).
FUN	the function to be applied to summarize the numeric data; default is to report the means and standard deviations
FUN2	a secondary function to be applied to summarize the numeric data; default is to report the medians and 25% and 75% quartiles
total	whether a total (not stratified with the splitby or group_by()) should also be reported in the table
second	a vector or list of quoted continuous variables for which the FUN2 should be applied
row_wise	how to calculate percentages for factor variables when splitby != NULL: if FALSE calculates percentages by variable within groups; if TRUE calculates percentages across groups for one level of the factor variable.
test	logical; if set to TRUE then the appropriate bivariate tests of significance are performed if splitby has more than 1 level. A consolidated message is printed listing all continuous variables that do not meet the assumption of Homogeneity of Variance (using Breusch-Pagan Test of Heteroskedasticity), and for those variables the argument ‘var.equal = FALSE‘ is used in the test.
param	logical; if set to TRUE then the appropriate parametric bivariate tests of significance are performed (if ‘test = TRUE‘). For continuous variables, it is a t-test or ANOVA (depending on the number of levels of the group). If set to FALSE, the Kruskal-Wallis Rank Sum Test is performed for the continuous variables. Either way, the chi-square test of independence is performed for categorical variables.
header_labels	a character vector that renames the header labels (e.g., the blank above the variables, the p-value label, and test value label).
type	what is displayed in the table; a string or a vector of strings. Two main sections can be inputted: 1. if test = TRUE, can write "pvalues", "full", or "stars" and 2. can state "simple" and/or "condense". These are discussed in more depth in the details section below.
output	how the table is output; can be "text" or "text2" for regular console output or any of kable()'s options from knitr (e.g., "latex", "markdown", "pandoc"). A new option, 'latex2', although more limited, allows the variable name to show and has an overall better appearance.
rounding_perc	the number of digits after the decimal for percentages; default is 1
digits	the number of significant digits for the numerical variables (if using default functions); default is 1.
var_names	custom variable names to be printed in the table. Variable names can be applied directly in the list of variables. Note: if variables have a "label" attribute (e.g., from Hmisc::label()), these labels will be used automatically unless var_names is explicitly provided.

format_number	default is FALSE; if TRUE, then the numbers are formatted with commas (e.g., 20,000 instead of 20000)
NAkeep	when set to TRUE it also shows how many missing values are in the data for each categorical variable being summarized (deprecated; use na.rm)
na.rm	when set to FALSE it also shows how many missing values are in the data for each categorical variable being summarized
booktabs	when output != "text"; option is passed to knitr::kable
caption	when output != "text"; option is passed to knitr::kable
align	when output != "text"; option is passed to knitr::kable
float	the float applied to the table in Latex when output is latex2, default is "ht".
export	character; when given, it exports the table to a CSV file to folder named "table1" in the working directory with the name of the given string (e.g., "myfile" will save to "myfile.csv")
label	for output == "latex2", this provides a table reference label for latex

Details

In defining type, 1. options are "pvalues" that display the p-values of the tests, "full" which also shows the test statistics, or "stars" which only displays stars to highlight significance with *** < .001 ** .01 * .05; and 2. "simple" then only percentages are shown for categorical variable and "condense" then continuous variables' means and SD's will be on the same line as the variable name and dichotomous variables only show counts and percentages for the reference category.

Value

A table with the number of observations, means/frequencies and standard deviations/percentages is returned. The object is a table1 class object with a print method. Can be printed in LaTex form. For enhanced formatting, use table1_gt() for gt output or table1_flextable() for flextable output.

See Also

table1_gt, table1_flextable

Examples

```
## Fictitious Data ##
library(furniture)
library(dplyr)

x  <- runif(1000)
y  <- rnorm(1000)
z  <- factor(sample(c(0,1), 1000, replace=TRUE))
a  <- factor(sample(c(1,2), 1000, replace=TRUE))
df <- data.frame(x, y, z, a)

## Simple
table1(df, x, y, z, a)
```

```
## Stratified
## all three below are the same
table1(df, x, y, z,
       splitby = ~ a)
table1(df, x, y, z,
       splitby = "a")

## With Piping
df %>%
  table1(x, y, z,
         splitby = ~a)

df %>%
  group_by(a) %>%
  table1(x, y, z)

## Adjust variables within function and assign name
table1(df,
       x2 = ifelse(x > 0, 1, 0), z = z)

## Using variable labels (e.g., with Hmisc)
## Not run:
library(Hmisc)
df2 <- df
label(df2$x) <- "Continuous Variable X"
label(df2$y) <- "Continuous Variable Y"
label(df2$z) <- "Binary Factor Z"
label(df2$a) <- "Binary Factor A"

# Labels are automatically used in the table
table1(df2, x, y, z, splitby = ~a)

## End(Not run)

## Using enhanced formatting with gt or flextable
## Not run:
# With gt
df %>%
  group_by(a) %>%
  table1(x, y, z) %>%
  table1_gt(spanner = "Group A")

# With flextable
df %>%
  group_by(a) %>%
  table1(x, y, z) %>%
  table1_flextable(spanner = "Group A")

## End(Not run)
```

table1_flextable *flextable output for table1*

Description

This takes a table1 object and outputs a ‘flextable‘ version.

Usage

```
table1_flextable(tab, spanner = NULL)
```

Arguments

tab	the table1 object
spanner	the label above the grouping variable (if table1 is grouped) or any label you want to include over the statistics column(s)

Examples

```
library(furniture)
library(dplyr)

data('nhanes_2010')
nhanes_2010 %>%
  group_by(asthma) %>%
  table1(age, marijuana, illicit, rehab, na.rm = FALSE) %>%
  table1_flextable(spanner = "Asthma")
```

table1_gt *gt output for table1*

Description

This takes a table1 object and outputs a ‘gt‘ version.

Usage

```
table1_gt(tab, spanner = NULL)
```

Arguments

tab	the table1 object
spanner	the label above the grouping variable (if table1 is grouped) or any label you want to include over the statistics column(s)

Examples

```
library(furniture)
library(dplyr)

data('nhanes_2010')
nhanes_2010 %>%
  group_by(asthma) %>%
  table1(age, marijuana, illicit, rehab, na.rm = FALSE) %>%
  table1_gt(spanner = "Asthma")
```

`tableC` *Correlation Table*

Description

Correlations printed in a nicely formatted table.

Usage

```
tableC(
  .data,
  ...,
  cor_type = "pearson",
  na.rm = FALSE,
  rounding = 3,
  output = "text",
  booktabs = TRUE,
  caption = NULL,
  align = NULL,
  float = "htb"
)
```

Arguments

`.data`	the data frame containing the variables
`...`	the unquoted variable names to be included in the correlations
`cor_type`	the correlation type; default is "pearson", other option is "spearman"
`na.rm`	logical (default is `FALSE`); if set to `TRUE`, the correlations use the "complete.obs" methods option from `stats::cor()`
`rounding`	the value passed to `round` for the output of both the correlation and p-value; default is 3
`output`	how the table is output; can be "text" for regular console output, "latex2" for specialized latex output, or any of `kable()`'s options from `knitr` (e.g., "latex", "markdown", "pandoc").
`booktabs`	when `output != "text"`; option is passed to `knitr::kable`

caption	when output != "text"; option is passed to knitr::kable
align	when output != "text"; option is passed to knitr::kable
float	when output == "latex2" it controls the floating parameter (h, t, b, H)

See Also

stats::cor

---

tableF *Frequency Table*

Description

Provides in-depth frequency counts and percentages.

Usage

```
tableF(.data, x, n = 20, splitby = NULL)
```

Arguments

.data	the data frame containing the variable
x	the bare variable name (not quoted)
n	the number of values shown int he table
splitby	the stratifying variable

Value

a list of class tableF containing the frequency table(s)

Examples

```
## Not run:

library(furniture)

data <- data.frame(
  x = sample(c(1,2,3,4), 100, replace=TRUE),
  y = rnorm(100)
)

## Basic Use
tableF(data, x)
tableF(data, y)

## Adjust the number of items shown
tableF(data, y, n = 10)
```

```
## Add splitby
tableF(data, x, splitby = y)


## End(Not run)
```

---

tableX *Table X (for Cross-Tabs)*

Description

Provides a pipe-able, clean, flexible version of `table()`.

Usage

```
tableX(.data, x1, x2, type = "count", na.rm = FALSE, format_number = FALSE)
```

Arguments

`.data`	the data frame containing the variables
`x1`	the first bare (not quoted) variable found in .data
`x2`	the second bare (not quoted) variable found in .data
`type`	the summarized output type; can be "count", "cell_perc", "row_perc", or "col_perc"
`na.rm`	logical; whether missing values should be removed
`format_number`	default is FALSE; if TRUE, then the numbers are formatted with commas (e.g., 20,000 instead of 20000)

Examples

```
## Not run:

library(furniture)
library(tidyverse)

data <- data.frame(
  x = sample(c(1,2,3,4), 100, replace=TRUE),
  y = sample(c(0,1), 100, replace=TRUE)
)

tableX(data, x, y)

data %>%
  tableX(x, y)

data %>%
  tableX(x, y, na.rm = TRUE)
```

```
## End(Not run)
```

`to_latex` *From Table 1 to Latex 2*

Description

Internal `table1()` and `tableC()` function for providing output = "latex2"

Usage

```
to_latex(
  tab,
  caption,
  align,
  len,
  splitby,
  float,
  booktabs,
  label,
  total = FALSE,
  cor_type = NULL
)
```

Arguments

`tab`	the table1 object
`caption`	caption character vector
`align`	align character vector
`len`	the number of levels of the grouping factor
`splitby`	the name of the grouping factor
`float`	argument for latex formatting
`booktabs`	add booktabs to latex table
`label`	latex label option
`total`	is there a total column (from Table 1) to be printed?
`cor_type`	optional argument regarding the correlation type (for tableC)

washer *Wash Your Data*

Description

Washes the data by replacing values with either NA's or other values set by the user. Useful for replacing values such as 777's or 999's that represent missing values in survey research. Can also perform many useful functions on factors (e.g., removing a level, replacing a level, etc.)

Usage

```
washer(x, ..., value = NA)
```

Arguments

x	the variable to have values adjusted
...	the values in the variable that are to be replaced by either NA's or the value set by the user. Can be a function (or multiple functions) to specify values to change (e.g., is.nan(), is.na()).
value	(optional) if specified, the values in ... will be replaced by this value (must be a single value)

Value

the original vector (although if the original was a factor, it was changed to a character) with the values changed where indicated.

Examples

```
x = c(1:20, NA, NaN)
washer(x, 9, 10)
washer(x, 9, 10, value=0)
washer(x, 1:10)
washer(x, is.na, is.nan, value=0)
washer(x, is.na, is.nan, 1:3, value=0)
```

wide *Long to Wide Data Reshaping*

Description

`wide()` is a wrapper of `stats::reshape()` that takes the data from a long format to a wide format.

Usage

```
wide(data, v.names, timevar, id = NULL)
```

Arguments

`data`	the data.frame containing the wide format data
`v.names`	the variable names in quotes of the measures to be separated into multiple columns based on the time variable
`timevar`	the variable name in quotes of the time variable
`id`	the ID variable name in quotes

Author(s)

Tyson S. Barrett

See Also

`stats::reshape()`, `tidyr::spread()`

Index