Package 'tidystats'

September 21, 2020

Type Package

Title Save Output of Statistical Tests

Version 0.5

Author Willem Sleegers

Maintainer Willem Sleegers <w.sleegers@me.com>

Description Save the output of statistical tests in an organized file that can be shared with others or used to report statistics in scientific papers.

URL https://willemsleegers.github.io/tidystats/

BugReports https://github.com/WillemSleegers/tidystats/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

LazyData true

Depends R (>= 3.2.0)

Imports dplyr, tidyr, purrr, stringr, readr, jsonlite, tibble

Suggests knitr, rmarkdown, testthat, BayesFactor, lme4, lmerTest

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-09-21 16:50:02 UTC

R topics documented:

`add_stats` *Add statistical output to a tidystats list*

Description

`add_stats` is used to add the output of a statistical test to a tidystats list. While adding the output, additional information about the test can be added, including the type of test (primary, secondary, or exploratory), whether the test was preregistered, and additional notes. Please note that not all statistical tests are supported. See 'Details' below for a list of supported statistical tests.

Usage

```
add_stats(
  results,
  output,
  identifier = NULL,
  type = NULL,
  preregistered = NULL,
  notes = NULL
)
```

Arguments

`results`	A tidystats list.
`output`	Output of a statistical test.
`identifier`	A character string identifying the model. Automatically created if not provided.
`type`	A character string specifying the type of analysis: primary, secondary, or exploratory.
`preregistered`	A boolean specifying whether the analysis was preregistered or not.
`notes`	A character string specifying additional information.

Details

Currently supported functions:

`stats`:

- `t.test()`
- `cor.test()`
- `chisq.test()`
- `wilcox.test()`

- fisher.test()
- oneway.test()
- lm()
- glm()
- aov()
- anova()

lme4/lmerTest:

- lmer()

BayesFactor:

- generalTestBF()
- lmBF()
- regressionBF()
- ttestBF()
- anovaBF()
- correlationBF()
- contingencyTableBF()
- proportionBF()
- meta.ttestBF()

tidystats:

- describe_data()
- count_data()

Examples

```
# Load dplyr for access to the piping operator
library(dplyr)

# Conduct statistical tests
# t-test:
sleep_test <- t.test(extra ~ group, data = sleep, paired = TRUE)

# lm:
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)
lm_D9 <- lm(weight ~ group)

# ANOVA:
npk_aov <- aov(yield ~ block + N*P*K, npk)

#' # Create an empty list
```

```
results <- list()

# Add output to the results list
results <- results %>%
  add_stats(sleep_test) %>%
  add_stats(lm_D9, type = "primary", preregistered = TRUE) %>%
  add_stats(npk_aov, notes = "An ANOVA example")
```

`count_data` *Count the number of observations*

Description

`count_data` returns the number and percentage of observations for categorical variables.

Usage

```
count_data(data, ..., na.rm = FALSE)
```

Arguments

`data`	A data frame.
`...`	One or more unquoted (categorical) column names from the data frame, separated by commas.
`na.rm`	Logical. Should missing values (including NaN) be removed?

Details

The data frame can be grouped using **dplyr**'s `group_by` so that the number of observations will be calculated within each group level.

Examples

```
# Load dplyr for access to the %>% operator and group_by()
library(dplyr)

# 1 variable
count_data(quote_source, source)

# 2 variables
count_data(quote_source, source, sex)

# Ignore missing values
count_data(quote_source, source, sex, na.rm = TRUE)

# Use group_by() to get percentages within each group
quote_source %>%
  group_by(source) %>%
```

```
count_data(sex)
```

---

describe_data *Calculate common descriptive statistics*

Description

describe_data returns a set of common descriptive statistics (e.g., n, mean, sd) for numeric variables.

Usage

```
describe_data(data, column, na.rm = TRUE, short = FALSE)
```

Arguments

data	A data frame.
column	An unquoted (numerical) column name from the data frame.
na.rm	Logical. Should missing values (including NaN) be excluded in calculating the descriptives? The default is TRUE.
short	Logical. Should only a subset of descriptives be reported? If set to TRUE, only the N, M, and SD will be returned. The default is FALSE.

Details

The data can be grouped using dplyr::group_by so that descriptives will be calculated for each group level.

When na.rm is set to FALSE, a percentage column will be added to the output that contains the percentage of non-missing data.

Skew and kurtosis are based on the skewness and kurtosis functions of the moments package (Komsta & Novomestky, 2015).

Percentages are calculated based on the total of non-missing observations. When na.rm is set to FALSE, percentages are based on the total of missing and non-missing observations.

Examples

```
# Load the dplyr package for access to the %>% operator and group_by()
library(dplyr)

# Inspect descriptives of the response column from the 'quote_source' data
# frame included in tidystats
describe_data(quote_source, response)

# Repeat the former, now for each level of the source column
quote_source %>%
  group_by(source) %>%
```

```
  describe_data(response)

# Only inspect the total N, mean, and standard deviation
quote_source %>%
  group_by(source) %>%
  describe_data(response, short = TRUE)
```

---

quote_source *A Many Labs replication of Lorge & Curtiss (1936)*

Description

Data of multiple studies from the Many Labs project (Klein et al., 2014) replicating Lorge & Curtiss (1936).

Usage

```
quote_source
```

Format

A data frame with 6343 rows and 15 columns:

ID participant number

source attributed source of the quote: Washington or Bin Laden

response evaluation of the quote on a 9-point Likert scale, with 1 indicating disagreement and 9 indicating agreement

age participant's age

sex participant's sex

citizenship participant's citizenship

race participant's race

major participant's major

native_language participant's native language

referrer location of where the study was conducted

compensation how the participant was compensated for their participation

recruitment how the participant was recruited

separation description of how the study was administered in terms of participant isolation

us_or_international whether the study was conducted in the US or outside of the US (international)

lab_or_online whether the study was conducted in the lab or online

Details

Lorge and Curtiss (1936) examined how a quotation is perceived when it is attributed to a liked or disliked individual. The quotation of interest was: "I hold it that a little rebellion, now and then, is a good thing, and as necessary in the political world as storms are in the physical world." In one condition the quotation was attributed to Thomas Jefferson, a liked individual, and in the other condition it was attributed to Vladimir Lenin, a disliked individual. More agreement was observed when the quotation was attributed to Jefferson than Lenin. In the replication studies, the quotation was: "I have sworn to only live free, even if I find bitter the taste of death." This quotation was attributed to either George Washington, the liked individual, or Osama Bin Laden, the disliked individual.

References

Lorge, I., & Curtiss, C. C. (1936). Prestige, suggestion, and attitudes. The Journal of Social Psychology, 7, 386-402. https://doi.org/10.1080/00224545.1936.9919891

Klein, R.A. et al. (2014) Investigating Variation in Replicability: A "Many Labs" Replication Project. Social Psychology, 45(3), 142-152. https://dx.doi.org/10.1027/1864-9335/a000178

`read_stats` *Read a .json file that was produced with* `write_stats`

Description

`read_stats` can read in a .json file containing the statistical output that was produced with `write_stats`. It returns a list containing the results, with the identifier as the name for each list element.

Usage

```
read_stats(file)
```

Arguments

`file`	Path to the tidy stats data file

Examples

```
results <- read_stats(system.file("results.json", package = "tidystats"))
```

tidy_matrix *Helper functions in tidystats*

Description

Functions used under the hood in the `tidystats` package.

Usage

```
tidy_matrix(m)
```

Arguments

`m`	A matrix.

Functions

- `tidy_matrix`: Function to convert matrix objects to a tidy data frame.

tidy_stats *Tidy the output of a statistics object*

Description

`tidy_stats` is used to convert the output of a statistical object to a list of organized statistics. The `tidy_stats` function is automatically run when `add_stats` is used, so there is generally no need to use this function explicitly. It can be used, however, to peek at how the output of a specific analysis will be organized.

Usage

```
tidy_stats(x)

## S3 method for class 'htest'
tidy_stats(x)

## S3 method for class 'lm'
tidy_stats(x)

## S3 method for class 'glm'
tidy_stats(x)

## S3 method for class 'anova'
tidy_stats(x)

## S3 method for class 'aov'
```

```
tidy_stats(x)

## S3 method for class 'aovlist'
tidy_stats(x)

## S3 method for class 'tidystats_descriptives'
tidy_stats(x)

## S3 method for class 'tidystats_counts'
tidy_stats(x)

## S3 method for class 'lmerMod'
tidy_stats(x)

## S3 method for class 'lmerModLmerTest'
tidy_stats(x)

## S3 method for class 'BFBayesFactor'
tidy_stats(x)

## S3 method for class 'afex_aov'
tidy_stats(x)
```

Arguments

x	The output of a statistical test.

Details

Please note that not all statistical tests are supported. See 'Details' below for a list of supported statistical tests.

Currently supported functions:

stats:

- t.test()
- cor.test()
- chisq.test()
- wilcox.test()
- fisher.test()
- oneway.test()
- lm()
- glm()
- aov()
- anova()

lme4/lmerTest:

- lmer()

BayesFactor:

- generalTestBF()
- lmBF()
- regressionBF()
- ttestBF()
- anovaBF()
- correlationBF()
- contingencyTableBF()
- proportionBF()
- meta.ttestBF()

tidystats:

- describe_data()
- count_data()

Methods (by class)

- htest: tidy_stats method for class 'htest'
- lm: tidy_stats method for class 'lm'
- glm: tidy_stats method for class 'glm'
- anova: tidy_stats method for class 'anova'
- aov: tidy_stats method for class 'aov'
- aovlist: tidy_stats method for class 'aovlist'
- tidystats_descriptives: tidy_stats method for class 'tidystats_descriptives'
- tidystats_counts: tidy_stats method for class 'tidystats_counts'
- lmerMod: tidy_stats method for class 'lmerMod'
- lmerModLmerTest: tidy_stats method for class 'lmerModLmerTest'
- BFBayesFactor: tidy_stats method for class 'BayesFactor'
- afex_aov: tidy_stats method for class 'afex_aov'

Examples

```
# Conduct statistical tests
# t-test:
sleep_test <- t.test(extra ~ group, data = sleep, paired = TRUE)

# lm:
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)
```

```
lm_D9 <- lm(weight ~ group)

# ANOVA:
npk_aov <- aov(yield ~ block + N*P*K, npk)

# Tidy the statistics and store each analysis in a separate variable
list_sleep_test <- tidy_stats(sleep_test)
list_lm_D9 <- tidy_stats(lm_D9)
list_npk_aov <- tidy_stats(npk_aov)

# Now you can inspect each of these variables, e.g.,:
names(list_sleep_test)
str(list_sleep_test)
```

`tidy_stats_to_data_frame` *Convert a tidystats list to a data frame*

Description

`tidy_stats_to_data_frame` converts a tidystats list to a data frame, which can then be used to extract specific statistics using standard subsetting functions (e.g., `dplyr::filter`).

Usage

```
tidy_stats_to_data_frame(x)
```

Arguments

x	A tidystats list.

Examples

```
# Load dplyr for access to the piping operator
library(dplyr)

# t-test:
sleep_test <- t.test(extra ~ group, data = sleep, paired = TRUE)

# lm:
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)
lm_D9 <- lm(weight ~ group)

# ANOVA:
npk_aov <- aov(yield ~ block + N*P*K, npk)
```

```
#' # Create an empty list
results <- list()

# Add output to the results list
results <- results %>%
  add_stats(sleep_test) %>%
  add_stats(lm_D9, type = "primary", preregistered = TRUE) %>%
  add_stats(npk_aov, notes = "An ANOVA example")

# Convert the list to a data frame
results_df <- tidy_stats_to_data_frame(results)

# Select all the p-values
filter(results_df, statistic == "p")
```

`write_stats` *Write a tidystats list to a file*

Description

`write_stats` writes a tidystats list to a .json file.

Usage

```
write_stats(x, path, digits = 6)
```

Arguments

`x`	A tidystats list.
`path`	Path or connection to write to.
`digits`	The number of decimal places to use. The default is 6.

Examples

```
# Load dplyr for access to the piping operator
library(dplyr)

# Conduct statistical tests
# t-test:
sleep_test <- t.test(extra ~ group, data = sleep, paired = TRUE)

# lm:
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)
lm_D9 <- lm(weight ~ group)
```

```
# ANOVA:
npk_aov <- aov(yield ~ block + N*P*K, npk)

#' # Create an empty list
results <- list()

# Add output to the results list
results <- results %>%
  add_stats(sleep_test) %>%
  add_stats(lm_D9, type = "primary", preregistered = TRUE) %>%
  add_stats(npk_aov, notes = "An ANOVA example")

# Save the results
dir <- tempdir()
write_stats(results, file.path(dir, "results.json"))
```

Index