Package 'tidystats'

June 28, 2026

Type Package

Title Save Output of Statistical Tests

Version 0.7.1

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

LazyData true

Depends R (>= 4.1.0)

Imports jsonlite, methods, tools

Suggests BayesFactor, brms, knitr, lme4, lmerTest, rmarkdown, effectsize, effsize, Hmisc, afex, emmeans, irr, psych, testthat, lavaan, nlme

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Willem Sleegers [aut, cre]

Repository CRAN

Date/Publication 2026-06-28 21:10:02 UTC

Contents

`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.

Usage

```
add_stats(
  list,
  output,
  identifier = NULL,
  type = NULL,
  preregistered = NULL,
  notes = NULL,
  args = NULL,
  class = NULL
)
```

Arguments

`list`	A tidystats list.
`output`	Output of a statistical test.
`identifier`	A string identifying the model. Automatically created if not provided.
`type`	A string specifying the type of analysis: primary, secondary, or exploratory.
`preregistered`	A boolean specifying whether the analysis was preregistered or not.
`notes`	A string specifying additional information.
`args`	A list of additional arguments to customize which statistics should be extracted. See 'Details' for a list of supported functions and their arguments.
`class`	A string to manually specify the class of the object so that tidystats knows how to extract the statistics. See 'Details' for a list of classes that are supported.

Details

Many functions to perform statistical tests (e.g., `t.test()`, `lm()`) return an object containing the statistics. These objects can be stored in variables and used with `add_stats()` to extract the statistics and add them to a list.

The list can be saved to a file using the `write_stats()` function.

For a list of supported functions, see `vignette("supported-functions", package = "tidystats")`.

Examples

```
# Conduct analyses
sleep_test <- t.test(
  sleep$extra[sleep$group == 1],
  sleep$extra[sleep$group == 2],
  paired = TRUE
)

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)
lm_D9_confint <- confint(lm_D9)

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

# Create an empty list to store the statistics in
statistics <- list()

# Add statistics to the list
statistics <- statistics |>
  add_stats(sleep_test, type = "primary", preregistered = TRUE) |>
  add_stats(lm_D9) |>
  add_stats(lm_D9_confint, class = "confint") |>
  add_stats(npk_aov, notes = "An ANOVA example")
```

---

count_data *Count the number of observations*

Description

`count_data()` returns the number and proportion of observations for categorical variables.

Usage

```
count_data(data, ..., by = NULL, na.rm = FALSE, pct = FALSE)
```

Arguments

`data`	A data frame.
`...`	One or more unquoted (categorical) column names from the data frame, separated by commas.
`by`	An optional character vector of column names to group by.
`na.rm`	A boolean specifying whether missing values (including NaN) should be removed.
`pct`	A boolean indicating whether to calculate percentages instead of proportions. The default is `FALSE`.

Details

Use the by argument to group the data, or alternatively pipe grouped data created with dplyr::group_by().

Examples

```
count_data(quote_source, source)
count_data(quote_source, source, sex)
count_data(quote_source, source, sex, na.rm = TRUE)
count_data(quote_source, source, sex, na.rm = TRUE, pct = TRUE)

# Use the by argument to calculate proportions within a group
count_data(quote_source, sex, by = "source")
```

---

custom_stat *Create a custom statistic*

Description

custom_stat() is used together with the custom_stats() function to add statistics from unsupported functions via add_stats(). See the custom_stats() function for more information.

Usage

```
custom_stat(
  name,
  value,
  symbol = NULL,
  subscript = NULL,
  interval = NULL,
  level = NULL,
  lower = NULL,
  upper = NULL
)
```

Arguments

name	A string specifying the name of the statistic.
value	The numeric value of the statistic.
symbol	A string specifying the symbol of the statistic to use when reporting the statistic.
subscript	A string specifying a subscript to use when reporting the statistic.
interval	A string specifying the type of interval if the statistic is a ranged statistic (e.g., 95% confidence interval)
level	A numeric value between 0 and 1 indicating the level of the interval.
lower	The numeric value of the lower bound of the statistic.
upper	The numeric value of the upper bound of the statistic.

Examples

```
# Example 1: A single mean value
sample <- rnorm(1000, mean = 0, sd = 1)
mean <- mean(sample)

custom_stat(name = "mean", value = mean, symbol = "M")

# Example 2: A mean with a 95% confidence interval
sample <- rnorm(1000, mean = 0, sd = 1)
mean <- mean(sample)
se <- sd(sample) / sqrt(length(sample))
CI <- c(mean - 1.96 * se, mean + 1.96 * se)

custom_stat(
  name = "mean",
  value = mean,
  symbol = "M",
  interval = "CI",
  level = .95,
  lower = CI[1],
  upper = CI[2]
)
```

`custom_stats` *Create a collection of custom statistics*

Description

`custom_stats()` is used to create a collection of statistics from unsupported functions to add to a list via `add_stats()`.

Usage

```
custom_stats(method, statistics)
```

Arguments

`method`	A string specifying the method used to obtain the statistics.
`statistics`	A vector of statistics created with `custom_stat()`.

Details

`custom_stats()` supports adding a single statistic or a group of statistics. Multiple groups of statistics are not (yet) supported.

Examples

```
# Example: BIC Bayes factor (approx.)
# Run the analysis
lm1 <- lm(Fertility ~ ., data = swiss)
lm2 <- update(lm1, . ~ . - Examination)

BF10 <- 1 / exp((BIC(lm2) - BIC(lm1)) / 2)

# Create the custom statistics
BIC_BFs <- custom_stats(
  method = "BIC Bayes factor",
  statistics = c(
    custom_stat(name = "BF", value = BF10, subscript = "10"),
    custom_stat(name = "BF", value = 1 / BF10, subscript = "01")
  )
)

# Create an empty list
statistics <- list()

# Add the custom statistics to the list
statistics <- add_stats(statistics, BIC_BFs)
```

`describe_data` *Calculate common descriptive statistics*

Description

`describe_data()` returns a set of common descriptive statistics (e.g., number of observations, mean, standard deviation) for one or more numeric variables.

Usage

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

Arguments

`data`	A data frame.
`...`	One or more unquoted column names from the data frame.
`by`	An optional character vector of column names to group by.
`na.rm`	A boolean indicating whether missing values (including NaN) should be excluded in calculating the descriptives? The default is TRUE.
`short`	A boolean indicating whether only a subset of descriptives should be reported? If set to `TRUE``, only the N, M, and SD will be returned. The default is FALSE‘.

Details

Use the by argument to group the data, or alternatively pipe grouped data created with `dplyr::group_by()`.

Examples

```
describe_data(quote_source, response)

describe_data(quote_source, response, na.rm = FALSE)

describe_data(quote_source, response, by = "source")

describe_data(quote_source, response, by = "source", 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. doi: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. doi:10.1027/18649335/a000178

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

Description

`read_stats()` can read a .json file containing statistics that was produced using tidystats. It returns a list containing the statistics, with the identifier as the name for each list element.

Usage

```
read_stats(file)
```

Arguments

`file`	A string specifying the path to the tidystats data file.

Examples

```
# A simple example, assuming there is a file called 'statistics.json'
## Not run:
statistics <- read_stats("statistics.json")

## End(Not run)

# A working example
statistics <- read_stats(
  file = system.file("extdata", "statistics.json", package = "tidystats")
)
```

`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., `subset()`).

Usage

```
tidy_stats_to_data_frame(x)
```

Arguments

x	A tidystats list.

Examples

```
# Conduct analyses
sleep_test <- t.test(
  sleep$extra[sleep$group == 1],
  sleep$extra[sleep$group == 2],
  paired = TRUE
)

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)

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

# Create an empty list to store the statistics in
statistics <- list()

# Add statistics
statistics <- statistics |>
  add_stats(sleep_test, type = "primary", preregistered = TRUE) |>
  add_stats(lm_D9) |>
  add_stats(npk_aov, notes = "An ANOVA example")

# Convert the list to a data frame
df <- tidy_stats_to_data_frame(statistics)

# Select all the p-values
subset(df, statistic_name == "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`	A string specifying the path or connection to write to.
`digits`	The number of decimal places to use. The default is 6.

Examples

```
# Conduct a statistical test
sleep_test <- t.test(
  sleep$extra[sleep$group == 1],
  sleep$extra[sleep$group == 2],
  paired = TRUE
)

# Create an empty list
statistics <- list()

# Add statistics to the list
statistics <- add_stats(statistics, sleep_test)

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

Index