The Journal of Open Source Software

statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details

Indrajeet Patil[1]

1 Center for Humans and Machines, Max Planck Institute for Human Development, Berlin, Germany

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

The `statsExpressions` package is designed to facilitate producing dataframes with rich statistical details for the most common types of statistical approaches and tests: parametric, nonparametric, robust, and Bayesian *t*-test, one-way ANOVA, correlation analyses, contingency table analyses, and meta-analyses. The functions are pipe-friendly and provide a consistent syntax to work with tidy data. These dataframes additionally contain expressions with statistical details, and can be used in graphing packages to display these details.

Aim of the Package

The aim of this package is to provide an approachable and intuitive syntax to carry out common statistical tests across diverse statistical approaches.

Comparison to Other Packages

Behind the scenes, `statsExpressions` uses `stats` package for parametric and nonparametric (R Core Team, 2021), `WRS2` package for robust (Mair & Wilcox, 2020), and `BayesFactor` package for Bayesian statistics (Morey & Rouder, 2020). Additionally, random-effects meta-analysis is carried out using `metafor` (parametric) (Viechtbauer, 2010), `metaplus` (robust) (Beath, 2016), and `metaBMA` (Bayesian) (Heck et al., 2019) packages. So one can naturally ask why there needs to be another package that wraps around these packages.

There is a lot of diversity among these packages in terms of their syntax and expected input type that can make it difficult to switch from one statistical approach to another. For example, some functions expect vectors as inputs, while others expect dataframes. Depending on whether it is a repeated measures design or not, different functions might expect data to be in wide or long format. Some functions can internally omit missing values, while other functions error in their presence. Furthermore, if someone wishes to utilize the objects returned by these packages downstream in their workflow, this is not straightforward either because even functions from the same package can return a list, a matrix, an array, a dataframe, etc., depending on the function. So on and so forth.

The result of sustained exposure to such inconsistencies is that data exploration can become a cognitively demanding task and discourage users to explore different statistical approaches. In the long run, this might even solidify into a habit of sticking to the defaults without giving much thought to the alternative approaches (e.g., exploring if Bayesian hypothesis testing is to be preferred over null hypothesis significance testing in context of the problem).

This is where `statsExpressions` comes in: It can be thought of as a unified portal through which most of the functionality in these underlying packages can be accessed, with a little to no cognitive overhead. The package offers just six primary functions that let users choose a statistical approach without changing the syntax. The users are always expected to provide a dataframe in tidy format (Wickham et al., 2019) to functions, all functions work with missing data, and they always return a dataframe that can be further utilized downstream in the pipeline (for a visualization, e.g.).

Table 1: A summary table listing the primary functions in the package and the statistical approaches they support. For a more detailed description of the tests and outputs from these functions, the readers are encouraged to read vignettes on the package website: https://indrajeetpatil.github.io/statsExpressions/articles/.

Function	Parametric	Non-parametric	Robust	Bayesian
`one_sample_test`	✓	✓	✓	✓
`two_sample_test`	✓	✓	✓	✓
`oneway_anova`	✓	✓	✓	✓
`corr_test`	✓	✓	✓	✓
`contingency_table`	✓	✓	-	✓
`meta_analysis`	✓	-	✓	✓

As can be seen, the package significantly simplifies and tidies up the syntax to run these different tests across different statistical frameworks.

Tidy Dataframes from Statistical Analysis

All functions return dataframes containing exhaustive details from inferential statistics, and appropriate effect size/posterior estimates and their confidence/credible intervals. The package internally relies on `easystats` ecosystem of packages to achieve this (Ben-Shachar, Lüdecke, & Makowski, 2020; Lüdecke, Ben-Shachar, Patil, & Makowski, 2020; Lüdecke, Makowski, Waggoner, & Patil, 2020; Lüdecke, Waggoner, & Makowski, 2019; Makowski, Ben-Shachar, & Lüdecke, 2019; Makowski, Ben-Shachar, Patil, & Lüdecke, 2020).

To illustrate the simplicity of this syntax, let's say we want to compare equality of a measure among two independent groups. We can use the `two_sample_test` function here.

If we first run a parametric *t*-test:

```r
# for reproducibility
library(statsExpressions)
set.seed(123)

# Welch's t-test
mtcars %>% two_sample_test(x = am, y = wt, type = "parametric")
#> # A tibble: 1 x 14
#>   term  group mean.group1 mean.group2 statistic df.error    p.value
#>   <chr> <chr>       <dbl>       <dbl>     <dbl>    <dbl>      <dbl>
#> 1 wt    am           3.77        2.41      5.49     29.2 0.00000627
#>   method                  estimate conf.level conf.low conf.high effectsize
#>   <chr>                      <dbl>      <dbl>    <dbl>     <dbl> <chr>
#> 1 Welch Two Sample t-test     1.84       0.95     1.00      2.66 Hedges' g
#>   expression
#>   <list>
#> 1 <language>
```

And then decide to run, instead, a robust *t*-test. The syntax remains the same:

```
# Yuen's t-test
mtcars %>% two_sample_test(x = am, y = wt, type = "robust")
#> # A tibble: 1 x 10
#>   statistic df.error   p.value
#>       <dbl>    <dbl>     <dbl>
#> 1      5.84     13.6 0.0000485
#>   method                                              estimate conf.low
#>   <chr>                                                  <dbl>    <dbl>
#> 1 Yuen's test on trimmed means for independent samples    0.915    0.754
#>   conf.high conf.level effectsize                        expression
#>       <dbl>      <dbl> <chr>                             <list>
#> 1     0.977       0.95 Explanatory measure of effect size <language>
```

These functions also play nicely with popular data manipulation packages. For example, let's say we want to use `dplyr` to repeat the same analysis across *all* levels of a certain grouping variable. Here is how we can do it:

```
# for reproducibility
set.seed(123)
suppressPackageStartupMessages(library(dplyr))

# grouped analysis
# running one-sample proportion test for all levels of `cyl`
mtcars %>%
  group_by(cyl) %>%
  group_modify(~ contingency_table(.x, am), .keep = TRUE) %>%
  ungroup()
#> # A tibble: 3 x 11
#>     cyl statistic    df p.value method
#>   <dbl>     <dbl> <dbl>   <dbl> <chr>
#> 1     4     2.27      1 0.132   Chi-squared test for given probabilities
#> 2     6     0.143     1 0.705   Chi-squared test for given probabilities
#> 3     8     7.14      1 0.00753 Chi-squared test for given probabilities
#>   estimate conf.level conf.low conf.high effectsize        expression
#>      <dbl>      <dbl>    <dbl>     <dbl> <chr>             <list>
#> 1    0.344       0.95    0         0.917 Cramer's V (adj.) <language>
#> 2    0           0.95    0         0     Cramer's V (adj.) <language>
#> 3    0.685       0.95    0.127     1.18  Cramer's V (adj.) <language>
```

Expressions for Plots

In addition to other details contained in the dataframe, there is also a column titled `expression`, which contains expression with statistical details and can be displayed in a plot. Displaying statistical results in the context of a visualization is indeed a philosophy adopted by the `ggstatsplot` package (Patil, 2018), and `statsExpressions` functions as its statistical processing backend.

The example below (Figure 1) shows how one can display results from Welch's one-way ANOVA in a custom plot:

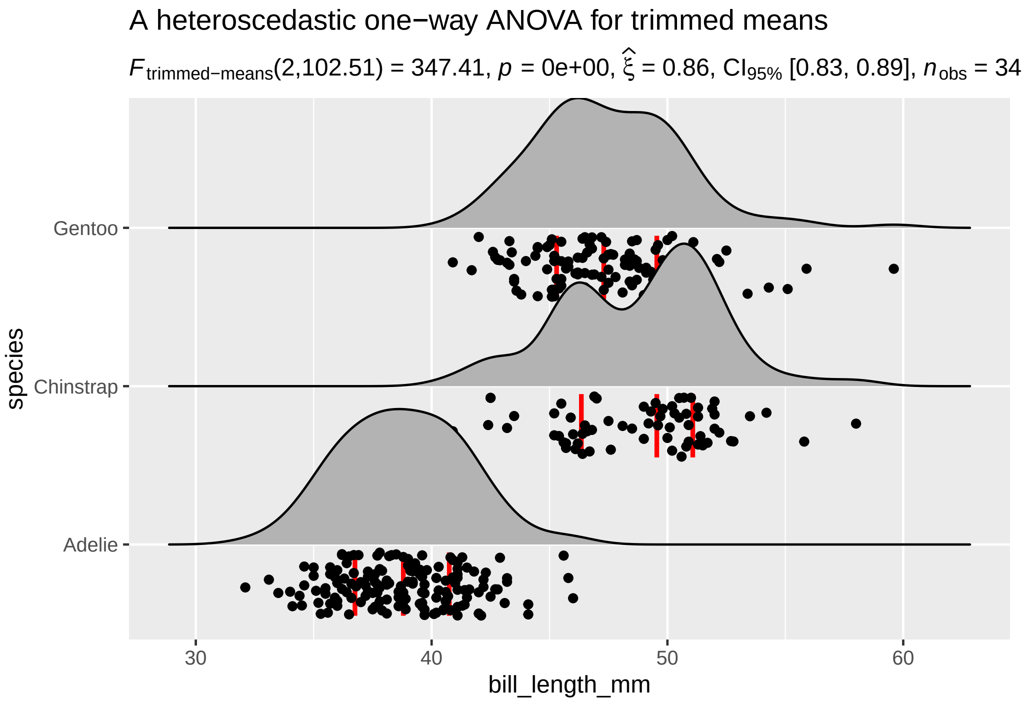

Figure 1: Example illustrating how ‘statsExpressions‘ functions can be used to display results from a statistical test in a plot.

```r
# for reproducibility
set.seed(123)
library(ggplot2)
library(palmerpenguins) # for data
library(ggridges) # for creating a ridgeplot

# creating a dataframe with results and expression
res <- oneway_anova(penguins, species, bill_length_mm, type = "robust")

# create a ridgeplot using `ggridges` package
ggplot(penguins, aes(x = bill_length_mm, y = species)) +
  geom_density_ridges(
    jittered_points = TRUE, quantile_lines = TRUE,
    scale = 0.9, vline_size = 1, vline_color = "red",
    position = position_raincloud(adjust_vlines = TRUE)
  ) + # use 'expression' column to display results in the subtitle
  labs(
    title = "A heteroscedastic one-way ANOVA for trimmed means",
    subtitle = res$expression[[1]]
  )
```

The details contained in these expressions (Figure 2) attempt to follow the gold standard in statistical reporting for both Bayesian (Doorn et al., 2020) and non-Bayesian (Association & others, 1985) framework tests.

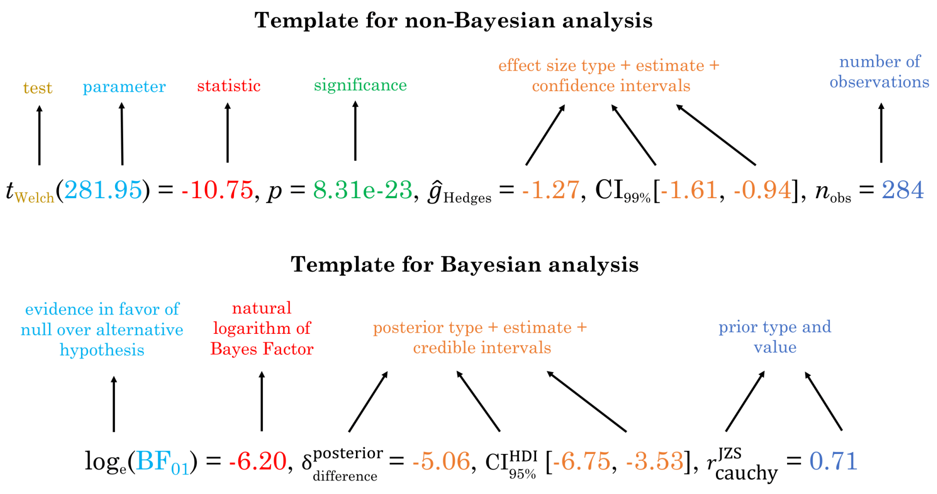
Template for non-Bayesian analysis
test
parameter
statistic
significance
effect size type + estimate + confidence intervals
number of observations
$t_{\text{Welch}}(281.95) = -10.75$, $p = 8.31\text{e-}23$, $\hat{g}_{\text{Hedges}} = -1.27$, $\text{CI}_{99\%}[-1.61, -0.94]$, $n_{\text{obs}} = 284$
Template for Bayesian analysis
evidence in favor of null over alternative hypothesis
natural logarithm of Bayes Factor
posterior type + estimate + credible intervals
prior type and value
$\log_e(\text{BF}_{01}) = -6.20$, $\delta^{\text{posterior}}_{\text{difference}} = -5.06$, $\text{CI}^{\text{HDI}}_{95\%}\,[-6.75, -3.53]$, $r^{\text{JZS}}_{\text{cauchy}} = 0.71$

Figure 2: The templates used in ‘statsExpressions‘ to display statistical details in a plot.

Licensing and Availability

`statsExpressions` is licensed under the GNU General Public License (v3.0), with all source code stored at GitHub, and with a corresponding issue tracker for bug reporting and feature enhancements. In the spirit of honest and open science, we encourage requests/tips for fixes, feature updates, as well as general questions and concerns via direct interaction with contributors and developers, by filing an issue. See the package's *Contribution Guidelines*.

Acknowledgements

I would like to acknowledge the support of Mina Cikara, Fiery Cushman, and Iyad Rahwan during the development of this project. `statsExpressions` relies heavily on the `easystats` ecosystem, a collaborative project created to facilitate the usage of `R` for statistical analyses. Thus, I would like to thank the members of easystats as well as the users.

References

Association, A. P., & others. (1985). *Publication manual of the american psychological association: American psychological association*. American Psychological Association.

Beath, K. J. (2016). metaplus: An R package for the analysis of robust meta-analysis and meta-regression. *R Journal*, *8*(1), 5–16. Retrieved from https://journal.r-project.org/archive/2016-1/beath.pdf

Ben-Shachar, M. S., Lüdecke, D., & Makowski, D. (2020). effectsize: Estimation of effect size indices and standardized parameters. *Journal of Open Source Software*, *5*(56), 2815. doi:10.21105/joss.02815

Doorn, J. van, Bergh, D. van den, Böhm, U., Dablander, F., Derks, K., Draws, T., Etz, A., et al. (2020). The JASP guidelines for conducting and reporting a bayesian analysis. *Psychonomic Bulletin & Review*, 1–14.

Heck, W., D., Gronau, F., Q., Wagenmakers, &, & E.-J. (2019). *metaBMA: Bayesian model averaging for random and fixed effects meta-analysis.* Retrieved from https://CRAN.R-project.org/package=metaBMA

Lüdecke, D., Ben-Shachar, M. S., Patil, I., & Makowski, D. (2020). Parameters: Extracting, computing and exploring the parameters of statistical models using R. *Journal of Open Source Software*, *5*(53), 2445. doi:10.21105/joss.02445

Lüdecke, D., Makowski, D., Waggoner, P., & Patil, I. (2020). Performance: Assessment of regression models performance. *CRAN*. doi:10.5281/zenodo.3952174

Lüdecke, D., Waggoner, P., & Makowski, D. (2019). Insight: A unified interface to access information from model objects in r. *Journal of Open Source Software*, *4*(38), 1412. doi:10.21105/joss.01412

Mair, P., & Wilcox, R. (2020). Robust Statistical Methods in R Using the WRS2 Package. *Behavior Research Methods*, *52*, 464–488.

Makowski, D., Ben-Shachar, M. S., & Lüdecke, D. (2019). bayestestR: Describing effects and their uncertainty, existence and significance within the bayesian framework. *Journal of Open Source Software*, *4*(40), 1541. doi:10.21105/joss.01541

Makowski, D., Ben-Shachar, M. S., Patil, I., & Lüdecke, D. (2020). Methods and algorithms for correlation analysis in r. *Journal of Open Source Software*, *5*(51), 2306. doi:10.21105/joss.02306

Morey, R. D., & Rouder, J. N. (2020). *BayesFactor: Computation of bayes factors for common designs*. Retrieved from https://richarddmorey.github.io/BayesFactor/

Patil, I. (2018). ggstatsplot: 'ggplot2' based plots with statistical details. *CRAN*. doi:10.5281/zenodo.2074621

R Core Team. (2021). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, *36*(3), 1–48. Retrieved from https://www.jstatsoft.org/v36/i03/

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., et al. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. doi:10.21105/joss.01686