ggstatsplot: ggplot2 Based Plots with Statistical Details

Indrajeet Patil*

2020-12-09

Contents

*Max Planck Institute for Human Development, patilindrajeet.science@gmail.com

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'

## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter()  masks ggstatsplot::filter(), stats::filter()
## x purrr::is_null() masks ggstatsplot::is_null(), testthat::is_null()
## x dplyr::lag()     masks stats::lag()
## x dplyr::matches() masks tidyr::matches(), ggstatsplot::matches(), testthat::matches()
```

> "What is to be sought in designs for the display of information is the clear portrayal of complexity. Not the complication of the simple; rather ... the revelation of the complex."
> - Edward R. Tufte

1 Introduction: Raison d'être

ggstatsplot is an extension of ggplot2 package for creating graphics with details from statistical tests included in the plots themselves and targeted primarily at behavioral sciences community to provide a one-line code to produce information-rich plots. In a typical exploratory data analysis workflow, data visualization and statistical modeling are two different phases: visualization informs modeling, and modeling in its turn can suggest a different visualization method, and so on and so forth. The central idea of `ggstatsplot` is simple: combine these two phases into one in the form of graphics with statistical details, which makes data exploration simpler and faster.

1.1 Need for informative visualizations

1.2 Need for better statistical reporting

But why would combining statistical analysis with data visualization be helpful? We list few reasons below-

- A recent survey (Nuijten, Hartgerink, van Assen, Epskamp, & Wicherts, 2016) revealed that one in eight papers in major psychology journals contained a grossly inconsistent p-value that may have affected the statistical conclusion. `ggstatsplot` helps avoid such reporting errors: Since the plot and the statistical analysis are yoked together, the chances of making an error in reporting the results are minimized. One need not write the results manually or copy-paste them from a different statistics software program (like SPSS, SAS, and so on).

2 `ggstatsplot` at a glance

2.1 Summary of types of plots included

It produces a limited kinds of ready-made plots for the supported analyses:

Function	Plot	Description
`ggbetweenstats`	**violin plots**	for comparisons *between* groups/conditions
`ggwithinstats`	**violin plots**	for comparisons *within* groups/conditions
`gghistostats`	**histograms**	for distribution about numeric variable
`ggdotplotstats`	**dot plots/charts**	for distribution about labeled numeric variable
`ggpiestats`	**pie charts**	for categorical data
`ggbarstats`	**bar charts**	for categorical data
`ggscatterstats`	**scatterplots**	for correlations between two variables
`ggcorrmat`	**correlation matrices**	for correlations between multiple variables
`ggcoefstats`	**dot-and-whisker plots**	for regression models

In addition to these basic plots, `ggstatsplot` also provides **`grouped_`** versions (see below) that makes it easy to repeat the same analysis for any grouping variable.

2.2 Summary of types of statistical analyses

Most functions share a `type` (of test) argument that is helpful to specify the type of statistical analysis:

- `"parametric"` (for **parametric** tests)
- `"nonparametric"` (for **non-parametric** tests)
- `"robust"` (for **robust** tests)
- `"bayes"` (for **Bayes Factor** tests)

The table below summarizes all the different types of analyses currently supported in this package-

Functions	Description	Parametric	Non-parametric	Robust	Bayes Factor
`ggbetweenstats`	Between group/condition comparisons	Yes	Yes	Yes	Yes
`ggwithinstats`	Within group/condition comparisons	Yes	Yes	Yes	Yes
`gghistostats`, `ggdotplotstats`	Distribution of a numeric variable	Yes	Yes	Yes	Yes
`ggcorrmat`	Correlation matrix	Yes	Yes	Yes	Yes
`ggscatterstats`	Correlation between two variables	Yes	Yes	Yes	Yes
`ggpiestats`, `ggbarstats`	Association between categorical variables	Yes	`NA`	`NA`	Yes
`ggpiestats`, `ggbarstats`	Equal proportions for categorical variable levels	Yes	`NA`	`NA`	Yes
`ggcoefstats`	Regression model coefficients	Yes	Yes	Yes	Yes
`ggcoefstats`	Random-effects meta-analysis	Yes	`NA`	Yes	Yes

In the following sections, we will discuss at depth justification for why the plots have been designed in certain ways and what principles were followed to report statistical details on the plots.

3 Graphic design principles

3.1 Graphical perception

Graphical perception involves visual decoding of the encoded information in graphs. `ggstatsplot` incorporates the paradigm proposed in ((Cleveland, 1985), Chapter 4) to facilitate making visual judgments about quantitative information effortless and almost instantaneous. Based on experiments, Cleveland proposes that there are ten elementary graphical-perception tasks that we perform to visually decode quantitative information in graphs (organized from most to least accurate; (Cleveland, 1985), p.254)-

- Position along a common scale
- Position along identical, non-aligned scales
- Length
- Angle (Slope)
- Area
- Volume
- Color hue

So the key principle of Cleveland's paradigm for data display is-

> "We should encode data on a graph so that the visual decoding involves [graphical-perception] tasks as high in the ordering as possible."

For example, decoding the data point values in `ggbetweenstats` requires position judgments along a common scale (Figure 1):

```
# for reproducibility
set.seed(123)

# plot
ggstatsplot::ggbetweenstats(
  data = dplyr::filter(
    .data = ggstatsplot::movies_long,
    genre %in% c("Action", "Action Comedy", "Action Drama", "Comedy")
  ),
  x = genre,
  y = rating,
  title = "IMDB rating by film genre",
  xlab = "Genre",
  ylab = "IMDB rating (average)",
  ggtheme = hrbrthemes::theme_ipsum_tw(),
  ggstatsplot.layer = FALSE,
  outlier.tagging = TRUE,
  outlier.label = title
)
```

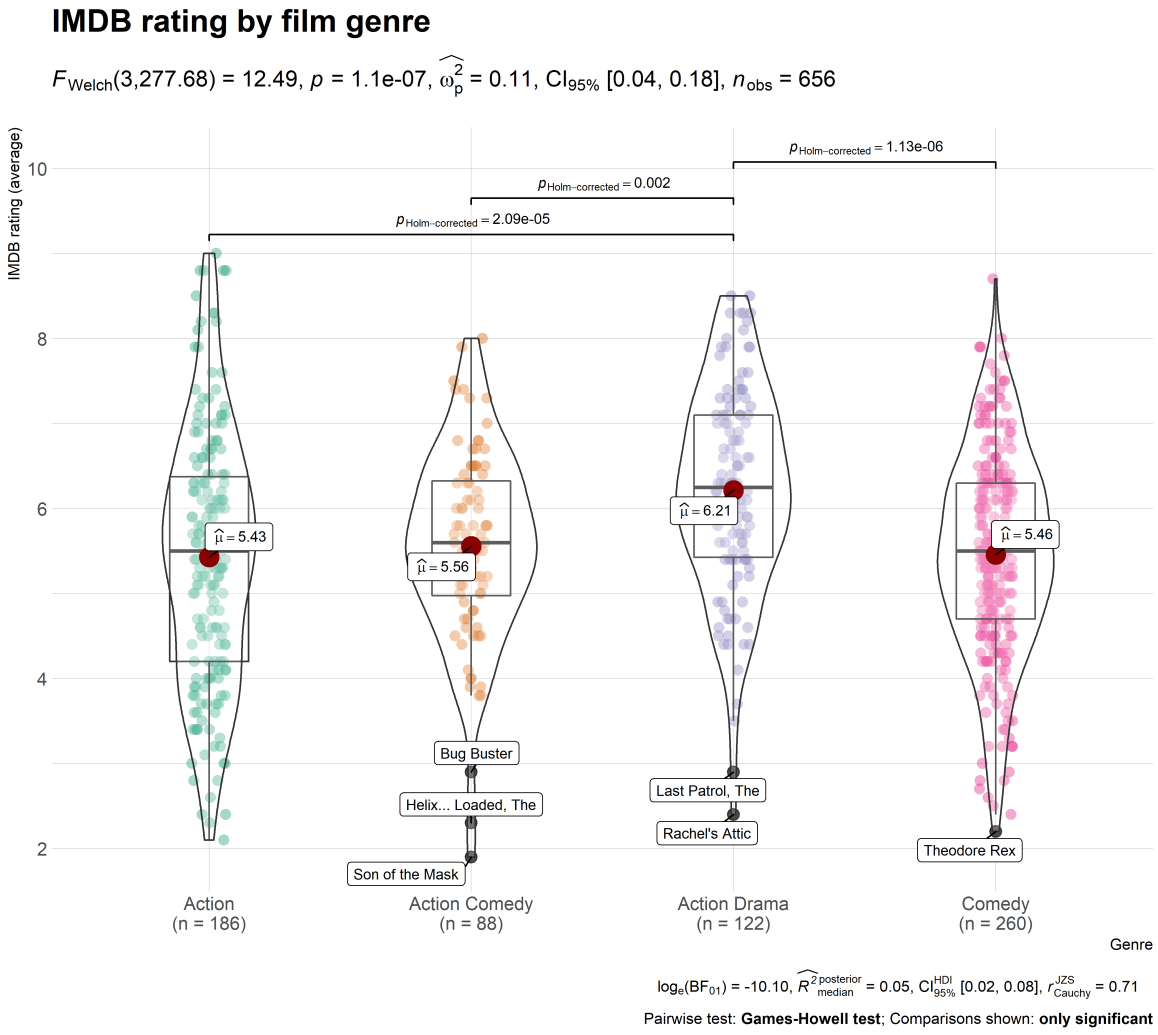

Figure 1: Note that assessing differences in mean values between groups has been made easier with the help of *position* of data points along a common scale (the Y-axis) and labels.

There are few instances where `ggstatsplot` diverges from recommendations made in Cleveland's paradigm:

- For the categorical/nominal data, `ggstatsplot` uses pie charts (see Figure 2) which rely on *angle* judgments, which are less accurate (as compared to bar graphs, e.g., which require *position* judgments). This shortcoming is assuaged to some degree by using plenty of labels that describe percentages for all slices. This makes angle judgment unnecessary and pre-vacates any concerns about inaccurate judgments about percentages. Additionally, it also provides alternative function to `ggpiestats` for working with categorical variables: `ggbarstats`.

```
# for reproducibility
set.seed(123)

# plot
```

```
ggstatsplot::ggpiestats(
  data = ggstatsplot::movies_long,
  x = genre,
  y = mpaa,
  title = "Distribution of MPAA ratings by film genre",
  legend.title = "layout",
  caption = substitute(paste(
    italic("MPAA"), ": Motion Picture Association of America"
  )),
  palette = "Paired"
)
```

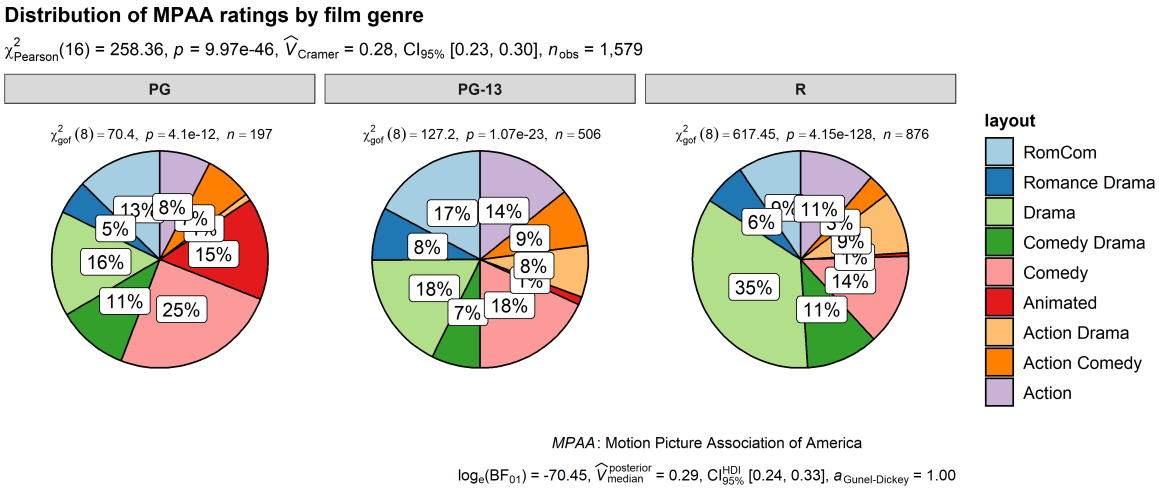

Figure 2: Pie charts don't follow Cleveland's paradigm to data display because they rely on less accurate angle judgments. 'ggstatsplot' sidesteps this issue by always labelling percentages for pie slices, which makes angle judgments unnecessary.

- Cleveland's paradigm also emphasizes that *superposition* of data is better than *juxtaposition* ((Cleveland, 1985), p.201) because this allows for a more incisive comparison of the values from different parts of the dataset. This recommendation is violated in all `grouped_` variants of the function (see Figure 3). Note that the range for Y-axes are no longer the same across juxtaposed subplots and so visually comparing the data becomes difficult. On the other hand, in the superposed plot, all data have the same range and coloring different parts makes the visual discrimination of different components of the data, and their comparison, easier. But the goal of `grouped_` variants of functions is to not only show different aspects of the data but also to run statistical tests and showing detailed results for all aspects of the data in a superposed plot is difficult. Therefore, this is a compromise `ggstatsplot` is comfortable with, at least to produce plots for quick exploration of different aspects of the data.

```
# for reproducibility
set.seed(123)
library(ggplot2)

# creating a smaller dataframe
df <- dplyr::filter(ggstatsplot::movies_long, genre %in% c("Comedy", "Drama"))

# plot
```

```
ggstatsplot::combine_plots(
  # plot 1: superposition
  ggplot(data = df, mapping = ggplot2::aes(x = length, y = rating, color = genre)) +
    geom_jitter(size = 3, alpha = 0.5) +
    geom_smooth(method = "lm") +
    labs(title = "superposition (recommended in Cleveland's paradigm)") +
    ggstatsplot::theme_ggstatsplot(),
  # plot 2: juxtaposition
  ggstatsplot::grouped_ggscatterstats(
    data = df,
    x = length,
    y = rating,
    grouping.var = genre,
    marginal = FALSE,
    title.prefix = "Genre",
    title.text = "juxtaposition (`ggstatsplot` implementation in `grouped_` functions)",
    title.size = 12
  ),
  # combine for comparison
  title.text = "Two ways to compare different aspects of data",
  nrow = 2,
  labels = c("(a)", "(b)")
)
```

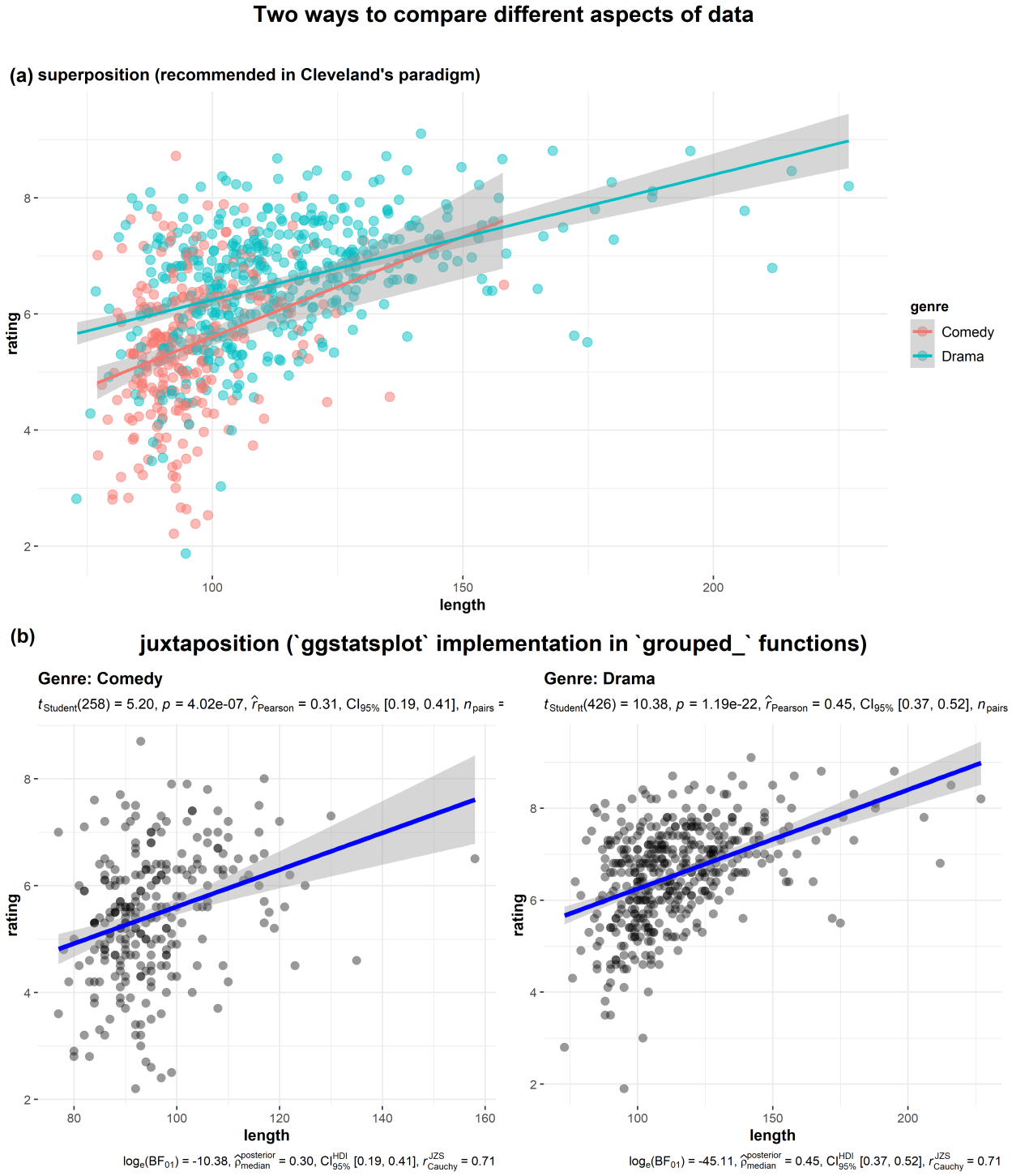

Figure 3: Comparing different aspects of data is much more accurate in (*a*) a *superposed* plot, which is recommended in Cleveland's paradigm, than in (*b*) a *juxtaposed* plot, which is how it is implemented in 'ggstatsplot' package. This is because displaying detailed results from statistical tests would be difficult in a superposed plot.

The `grouped_` plots follow the *Shrink Principle* ((Tufte, 2001), p.166-7) for high-information graphics, which dictates that the data density and the size of the data matrix can be maximized to exploit maximum resolution of the available data-display technology. Given the large maximum resolution afforded by most computer monitors today, saving `grouped_` plots with appropriate resolution ensures no loss in legibility with reduced graphics area.

3.2 Graphical excellence

Graphical excellence consists of communicating complex ideas with clarity and in a way that the viewer understands the greatest number of ideas in a short amount of time all the while not quoting the data out of context. The package follows the principles for *graphical integrity* (Tufte, 2001):

- The physical representation of numbers is proportional to the numerical quantities they represent (e.g., Figure 1 and Figure 2 show how means (in `ggbetweenstats`) or percentages (`ggpiestats`) are proportional to the vertical distance or the area, respectively).
- All important events in the data have clear, detailed, and thorough labeling (e.g., Figure 1 plot shows how `ggbetweenstats` labels means, sample size information, outliers, and pairwise comparisons; same can be appreciated for `ggpiestats` in Figure 2 and `gghistostats` in Figure 4). Note that data labels in the data region are designed in a way that they don't interfere with our ability to assess the overall pattern of the data ((Cleveland, 1985); p.44-45). This is achieved by using `ggrepel` package to place labels in a way that reduces their visual prominence.
- None of the plots have *design* variation (e.g., abrupt change in scales) over the surface of a same graphic because this can lead to a false impression about variation in *data*.
- The number of information-carrying dimensions never exceed the number of dimensions in the data (e.g., using area to show one-dimensional data).
- All plots are designed to have no **chartjunk** (like moiré vibrations, fake perspective, dark grid lines, etc.) ((Tufte, 2001), Chapter 5).

There are some instances where `ggstatsplot` graphs don't follow principles of clean graphics, as formulated in the Tufte theory of data graphics ((Tufte, 2001), Chapter 4). The theory has four key principles:

1. Above all else show the data.
2. Maximize the data-ink ratio.
3. Erase non-data-ink.
4. Erase redundant data-ink, within reason.

In particular, default plots in `ggstatsplot` can sometimes violate one of the principles from 2-4. According to these principles, every bit of ink should have reason for its inclusion in the graphic and should convey some new information to the viewer. If not, such ink should be removed. One instance of this is bilateral symmetry of data measures. For example, in Figure 1, we can see that both the box and violin plots are mirrored, which consumes twice the space in the graphic without adding any new information. But this redundancy is tolerated for the sake of beauty that such symmetrical shapes can bring to the graphic. Even Tufte admits that efficiency is but one consideration in the design of statistical graphics ((Tufte, 2001), p. 137). Additionally, these principles were formulated in an era in which computer graphics had yet to revolutionize the ease with which graphics could be produced and thus some of the concerns about minimizing data-ink for easier production of graphics are not as relevant as they were.

3.3 Statistical variation

One of the important functions of a plot is to show the variation in the data, which comes in two forms:

- **Measurement noise**: In `ggstatsplot`, the actual variation in measurements is shown by plotting a combination of (jittered) raw data points with a boxplot laid on top (Figure 1) or a histogram (Figure 4). None of the plots, where empirical distribution of the data is concerned, show the sample standard deviation because they are poor at conveying information about limits of the sample and presence of outliers ((Cleveland, 1985), p.220).

```
# for reproducibility
set.seed(123)

# plot
ggstatsplot::gghistostats(
  data = morley,
  x = Speed,
  test.value = 792,
  test.value.line = TRUE,
  xlab = "Speed of light (km/sec, with 299000 subtracted)",
  title = "Distribution of measured Speed of light",
  caption = "Note: Data collected across 5 experiments (20 measurements each)"
)
```

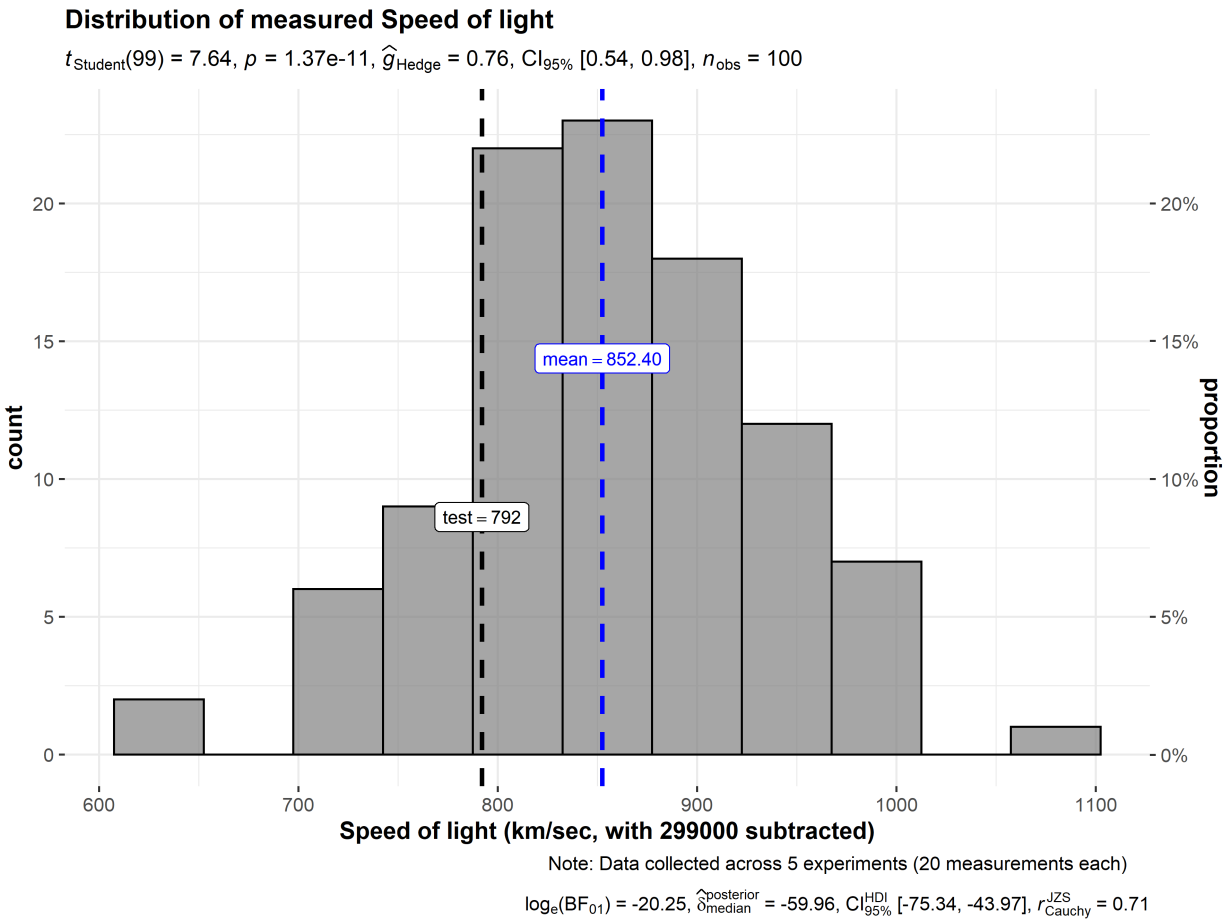

Figure 4: Distribution of a variable shown using 'gghistostats'.

- **Sample-to-sample statistic variation**: Although, traditionally, this variation has been shown using the standard error of the mean (SEM) of the statistic, `ggstatsplot` plots instead use 95% confidence intervals (e.g., Figure 5). This is because the interval formed by error bars correspond to a 68% confidence interval, which is not a particularly interesting interval ((Cleveland, 1985), p.222-225).

```
# for reproducibility
set.seed(123)

# creating model object
mod <- lme4::lmer(
  formula = total.fruits ~ nutrient + rack + (nutrient | gen),
  data = lme4::Arabidopsis
)

# plot
ggstatsplot::ggcoefstats(x = mod)
```

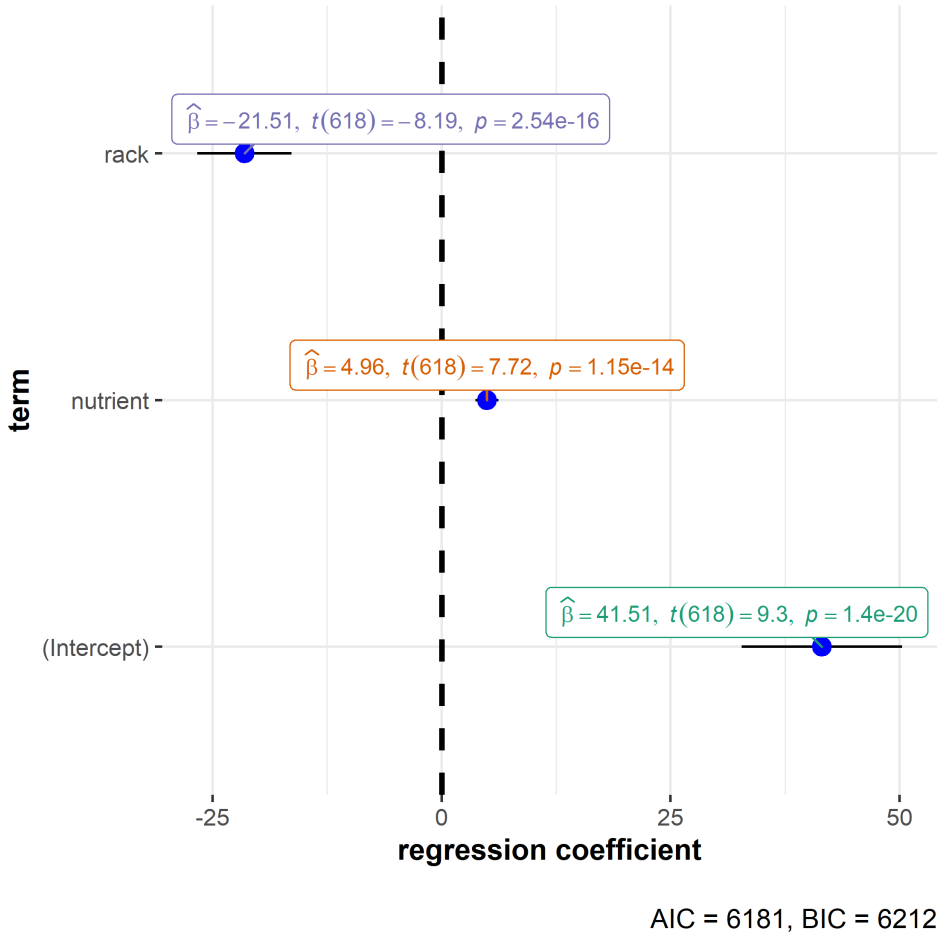

Figure 5: Sample-to-sample variation in regression estimates is displayed using confidence intervals in 'ggcoefstats'.

4 Statistical analysis

4.1 Data requirements

As an extension of `ggplot2`, `ggstatsplot` has the same expectations about the structure of the data. More specifically,

- The data should be organized following the principles of *tidy data*, which specify how statistical structure of a data frame (variables and observations) should be mapped to physical structure (columns and rows). More specifically, tidy data means all variables have their own columns and each row corresponds to a unique observation ((Wickham, 2014)).
- All `ggstatsplot` functions remove `NA`s from variables of interest (similar to `ggplot2`; (Wickham, 2016), p.207) in the data and display total sample size (n, either observations for between-subjects or pairs for within-subjects designs) in the subtitle to inform the user/reader about the number of observations included for both the statistical analysis and the visualization. But, when sample sizes differ *across* tests in the same function, `ggstatsplot` makes an effort to inform the user of this aspect. For example, `ggcorrmat` features several correlation test pairs and, depending on variables in a given pair, the sample sizes may vary (Figure 6).

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

# creating a new dataset without any NAs in variables of interest
msleep_no_na <-
  dplyr::filter(
    .data = ggplot2::msleep,
    !is.na(sleep_rem), !is.na(awake), !is.na(brainwt), !is.na(bodywt)
  )

# variable names vector
var_names <- c("REM sleep", "time awake", "brain weight", "body weight")

# combining two plots using helper function in `ggstatsplot`
ggstatsplot::combine_plots(
  plotlist = purrr::pmap(
    .l = list(data = list(msleep_no_na, ggplot2::msleep)),
    .f = ggstatsplot::ggcorrmat,
    cor.vars = c(sleep_rem, awake:bodywt),
    cor.vars.names = var_names,
    colors = c("#B2182B", "white", "#4D4D4D"),
    title = "Correlalogram for mammals sleep dataset",
    subtitle = "sleep units: hours; weight units: kilograms"
  ),
  labels = c("(a)", "(b)"),
  nrow = 1
)
```

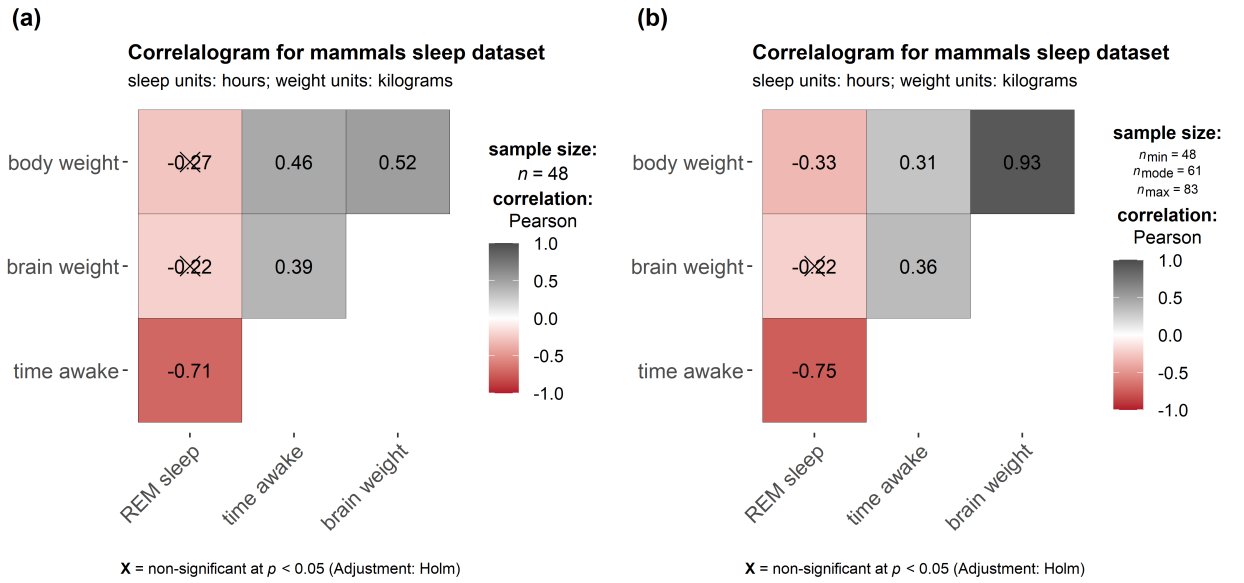

Figure 6: 'ggstatsplot' functions remove 'NA's from variables of interest and display total sample size n, but they can give more nuanced information about sample sizes when n differs across tests. For example, 'ggcorrmat' will display (a) only one total sample size once when no 'NA's present, but (b) will instead show minimum, median, and maximum sample sizes across all correlation tests when 'NA's are present across correlation variables.

4.2 Statistical reporting

The default setting in `ggstatsplot` is to produce plots with statistical details included. Most often than not, these results are displayed as a `subtitle` in the plot. Great care has been taken into which details are included in statistical reporting and why. For example, the template below is used to show results from Yuen's test for trimmed means (robust t-test):

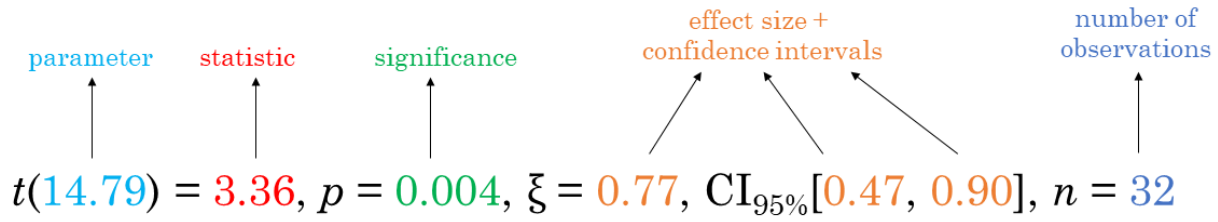

Figure 7: Template for reporting statistical details

APA guidelines (Association, 2009) are followed by default while reporting statistical details:

- Percentages are displayed with no decimal places (Figure 2).
- Correlations, t-tests, and χ^2-tests are reported with the degrees of freedom in parentheses and the significance level (Figure 6, Figure 3, Figure 4).

- ANOVAs are reported with two degrees of freedom and the significance level (Figure 1).
- Regression results are presented with the unstandardized or standardized estimate (beta), whichever was specified by the user, along with the statistic (depending on the model, this can be a t, F, or z statistic) and the corresponding significance level (Figure 5).
- With the exception of p-values, most statistics are rounded to two decimal places by default.

4.3 Statistical tests:

Here is a summary table of all the statistical tests currently supported across various functions:

Functions	Type	Test	Effect size	95% CI available?
`ggbetweenstats` (2 groups)	Parametric	Student's and Welch's t-test	Cohen's d, Hedge's g	✓
`ggbetweenstats` (>2 groups)	Parametric	Fisher's and Welch's one-way ANOVA	$\eta^2, \eta_p^2, \omega^2, \omega_p^2$	✓
`ggbetweenstats` (2 groups)	Non-parametric	Mann-Whitney U-test	r	✓
`ggbetweenstats` (>2 groups)	Non-parametric	Kruskal-Wallis Rank Sum Test	ϵ^2	✓
`ggbetweenstats` (2 groups)	Robust	Yuen's test for trimmed means	ξ	✓
`ggbetweenstats` (>2 groups)	Robust	Heteroscedastic one-way ANOVA for trimmed means	ξ	✓
`ggwithinstats` (2 groups)	Parametric	Student's t-test	Cohen's d, Hedge's g	✓
`ggwithinstats` (>2 groups)	Parametric	Fisher's one-way repeated measures ANOVA	η_p^2, ω^2	✓
`ggwithinstats` (2 groups)	Non-parametric	Wilcoxon signed-rank test	r	✓
`ggwithinstats` (>2 groups)	Non-parametric	Friedman rank sum test	$W_{Kendall}$	✓
`ggwithinstats` (2 groups)	Robust	Yuen's test on trimmed means for dependent samples	δ_R	✓
`ggwithinstats` (>2 groups)	Robust	Heteroscedastic one-way repeated measures ANOVA for trimmed means	×	×

Functions	Type	Test	Effect size	95% CI available?
`ggpiestats` and `ggbarstats` (unpaired)	Parametric	Pearson's χ^2 test	Cramér's V	✓
`ggpiestats` and `ggbarstats` (paired)	Parametric	McNemar's test	Cohen's g	✓
`ggpiestats`	Parametric	One-sample proportion test	Cramér's V	✓
`ggscatterstats` and `ggcorrmat`	Parametric	Pearson's r	r	✓
`ggscatterstats` and `ggcorrmat`	Non-parametric	Spearman's ρ	ρ	✓
`ggscatterstatsand` `ggcorrmat`	Robust	Percentage bend correlation	r	✓
`gghistostats` and `ggdotplotstats`	Parametric	One-sample t-test	Cohen's d, Hedge's g	✓
`gghistostats`	Non-parametric	One-sample Wilcoxon signed rank test	r	✓
`gghistostats` and `ggdotplotstats`	Robust	One-sample percentile bootstrap	robust estimator	✓
`ggcoefstats`	Parametric	Regression models	β	✓

4.4 Dealing with null results:

All functions therefore by default return Bayes Factor in favor of the null hypothesis by default. If the null hypothesis can't be rejected with the null hypothesis significance testing (NHST) approach, the Bayesian approach can help index evidence in favor of the null hypothesis (i.e., BF_01). By default, natural logarithms are shown because Bayes Factor values can sometimes be pretty large. Having values on logarithmic scale also makes it easy to compare evidence in favor alternative (BF_10) versus null (BF_01) hypotheses (since $log\_e(BF\_01) = -log\_e(BF\_01)$).

4.5 Avoiding the "p-value error":

The p-value indexes the probability that the researchers have falsely rejected a true null hypothesis (Type I error, i.e.) and can rarely be *exactly* 0. And yet over 97,000 manuscripts on Google Scholar report the p-value to be `p =0.000`, putatively due to relying on default computer software outputs (Lilienfeld et al., 2015). All p-values displayed in `ggstatsplot` plots avoid this mistake. Anything less than `p < 0.001` is displayed as such (e.g, Figure 1). The package deems it unimportant how infinitesimally small the p-values are and, instead, puts emphasis on the effect size magnitudes and their 95% CIs.

5 Acknowledgments

The authors would like to thank all people who have contributed in some way to the formation of this package. The authors would also like to acknowledge the help and support provided by the larger `#rstats` community on Twitter and StackOverflow for the development of this package. We also appreciate helpful discussions with Fiery Cushman.

6 Appendix

6.1 Appendix A: Documentation

There are three main documents one can rely on to learn how to use `ggstatsplot`:

- **Presentation**: The quickest (and the most fun) way to get an overview of the philosophy behind this package and the offered functionality is to go through the following slides: https://indrajeetpatil.github.io/ggstatsplot_slides/slides/ggstatsplot_presentation.html#1
- **Manual**:
  The `CRAN` reference manual provides detailed documentation about arguments for each function and examples: https://cran.r-project.org/web/packages/ggstatsplot/ggstatsplot.pdf
- **Vignettes**:
  Vignettes contain probably the most detailed exposition. Every single function in `ggstatsplot` has an associated vignette which describes in depth how to use the function and modify the defaults to customize the plot to your liking. All these vignettes can be accessed from the package website: https://indrajeetpatil.github.io/ggstatsplot/articles/

6.2 Appendix B: Suggestions

If you find any bugs or have any suggestions/remarks, please file an issue on `GitHub` repository for this package: https://github.com/IndrajeetPatil/ggstatsplot/issues

6.3 Appendix C: Session information

For reproducibility purposes, the details about the session information in which this document was rendered, see- https://indrajeetpatil.github.io/ggstatsplot/articles/web_only/session_info.html

References

Association, A. P. (2009). *Publication Manual of the American Psychological Association, 6th Edition.* Washington, DC: American Psychological Association.

Cleveland, W. S. (1985). *The Elements of Graphing Data* (1st edition). Monterey, Cal: Wadsworth, Inc.

Lilienfeld, S. O., Sauvign'e, K. C., Lynn, S. J., Cautin, R. L., Latzman, R. D., & Waldman, I. D. (2015). Fifty psychological and psychiatric terms to avoid: A list of inaccurate, misleading, misused, ambiguous, and logically confused words and phrases. *Frontiers in Psychology*, *6*. https://doi.org/10.3389/fpsyg.2015.01100

Nuijten, M. B., Hartgerink, C. H. J., van Assen, M. A. L. M., Epskamp, S., & Wicherts, J. M. (2016). The prevalence of statistical reporting errors in psychology (19852013). *Behavior Research Methods*, *48*(4), 1205–1226. https://doi.org/10.3758/s13428-015-0664-2

Tufte, E. R. (2001). *The Visual Display of Quantitative Information* (2nd edition edition). Cheshire, Conn: Graphics Press.

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(1), 1–23. https://doi.org/10.18637/jss.v059.i10

Wickham, H. (2016). *Ggplot2: Elegant Graphics for Data Analysis* (2nd ed. 2016 edition). New York, NY: Springer.