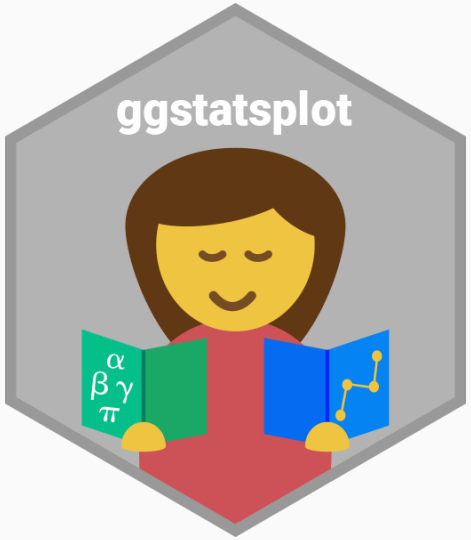
`ggstatsplot`: `ggplot2` Based Plots with Statistical Details

An Introductory Tutorial

Indrajeet Patil (Max Planck Institute, Berlin)
2021-02-08

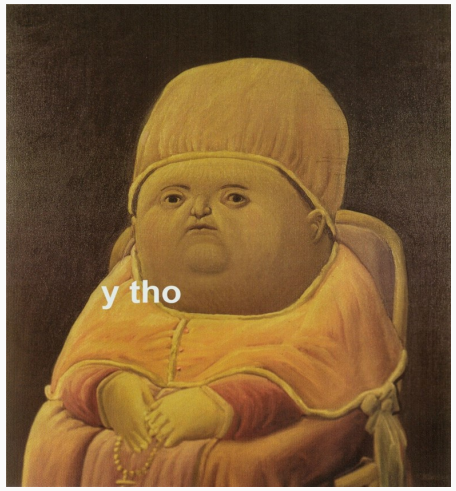


Plan

- Why `ggstatsplot`?
- Primary functions
- Benefits and customizability
- Misconceptions and limitations

Why *ggstatsplot*?

Raison d'être

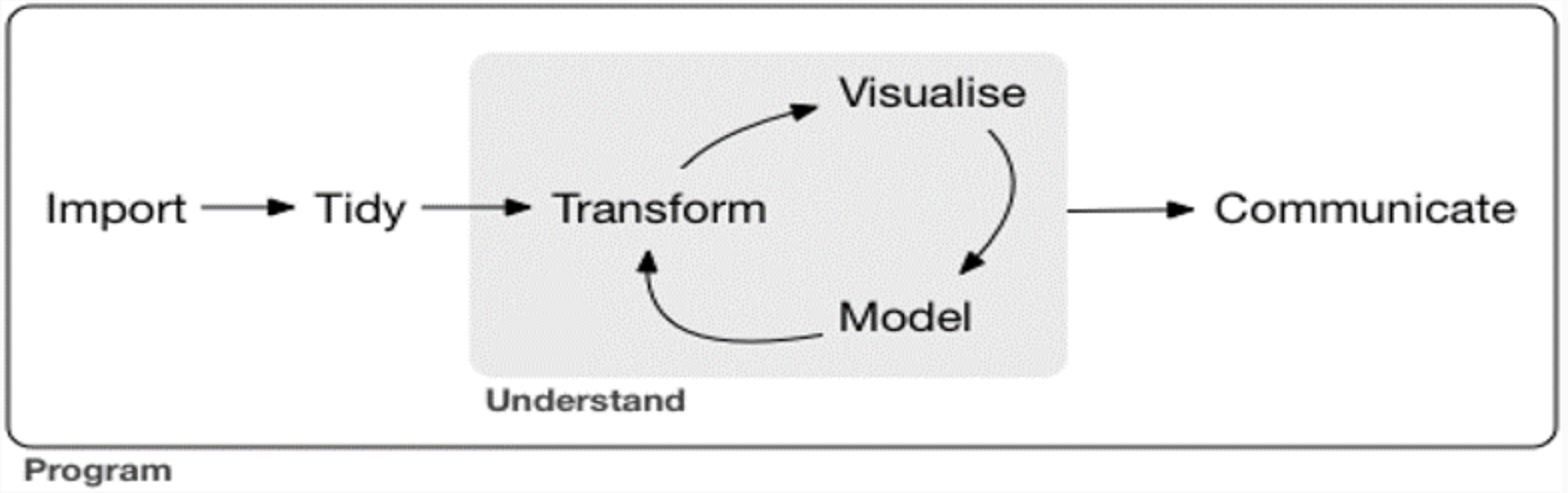


Current count of packages on the Comprehensive R Archive Network (**CRAN**) **> 16,000**

Short answer: `ggstatsplot` provides a collection of *information-rich* plots with *statistical details* and is suitable for scholarly publications and quick (exploratory) data analysis.

https://cran.r-project.org/web/packages/

Simpler data analysis workflow

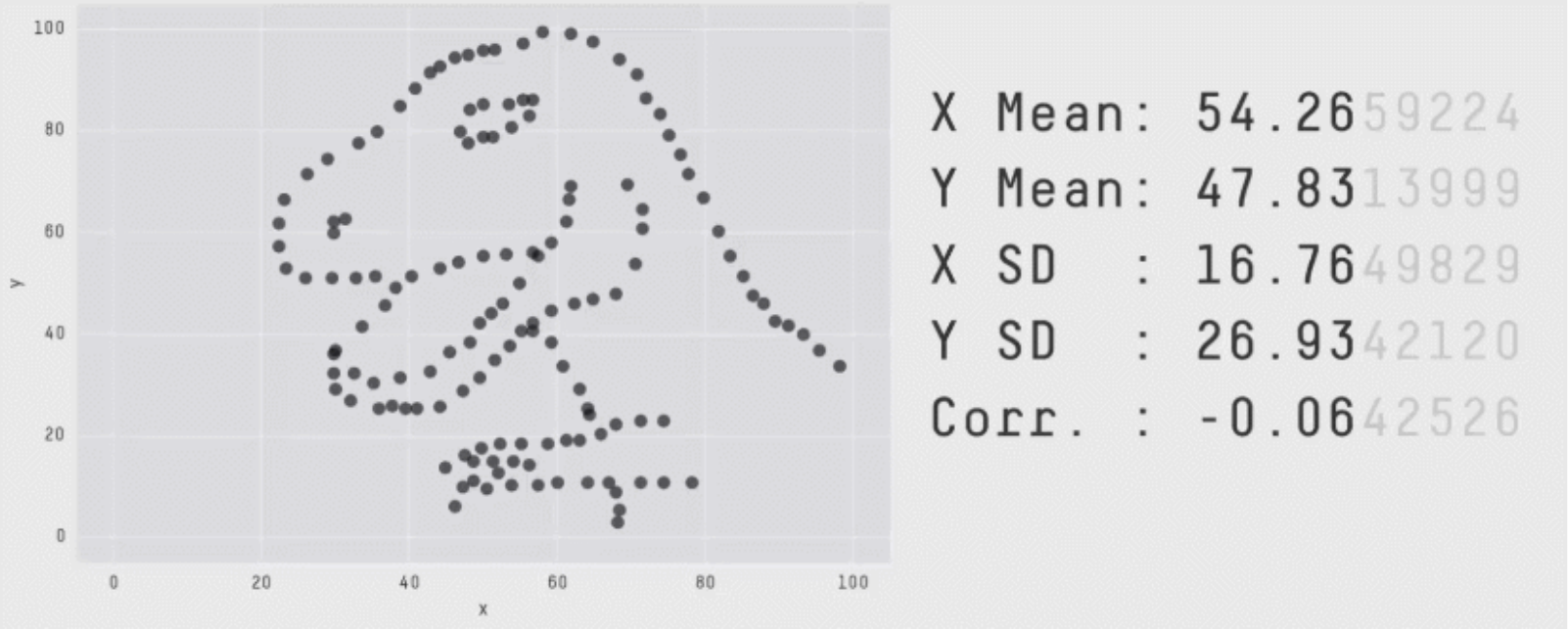


In a typical *exploratory* data analysis workflow, data visualization and statistical modeling are two different phases: visualization informs modeling, and modeling can suggest a different visualization, and so on and so forth.

The central idea of **ggstatsplot** is simple: combine these two phases into one!

(Grolemund & Wickham, *R for Data Science*, 2017)

Information-rich graphic is worth a thousand words



“I plotted my data and what I found surprised me!" - BuzzFeed

(Matejka & Fitzmaurice, *Autodesk Research*, 2017)

Ready-made plot = no customization

The **grammar of graphics** (implemented in `ggplot2`) is a powerful framework (Wilkinson, 2011) to prepare graphics and can help you make infinite number of graphics, each tailored for your specific data visualization problem! But...

Consistent API = No cognitive fatigue

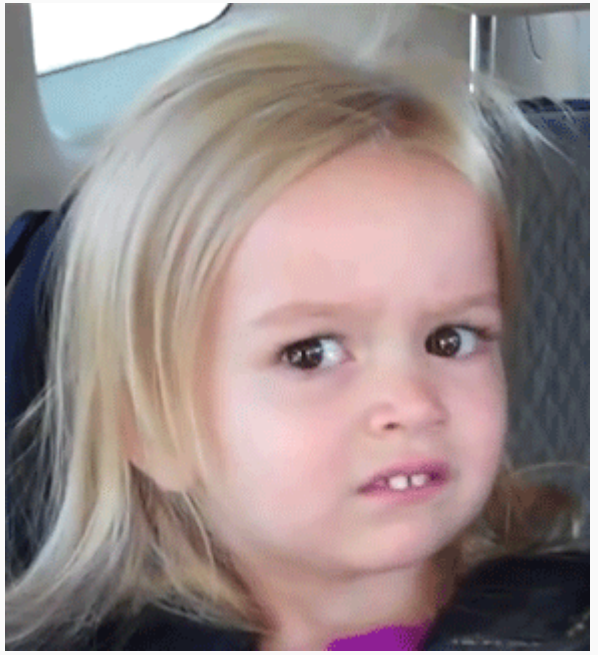
```
stats::lm(formula = wt ~ mpg, data = mtcars)
```

```
stats::cor(x = mtcars$wt, y = mtcars$mpg)
```

```
stats::cor.test(formula = ~ wt + mpg, data = mtcars)
```

All functions in `ggstatsplot` -

- ✅ rely on a **dataframe** (`function(data, x, ...)`)
- ✅ expect **tidy** data



Get Started

Installation

Install the stable version (latest: `0.6.8`) of **ggstatsplot** from CRAN:

```
install.packages("ggstatsplot")
```

You can get the development version of the package from Github:

```
remotes::install_github("IndrajeetPatil/ggstatsplot")
```

Load the needed packages-

```
library(ggstatsplot)
library(ggplot2)
```

Primary functions

Hypothesis about group differences

🔬 *ggbetweenstats*, *ggwithinstats*: multiple groups

🔬 *gghistostats*, *ggdotplotstats*: single group

ggbetweenstats - For between group comparisons

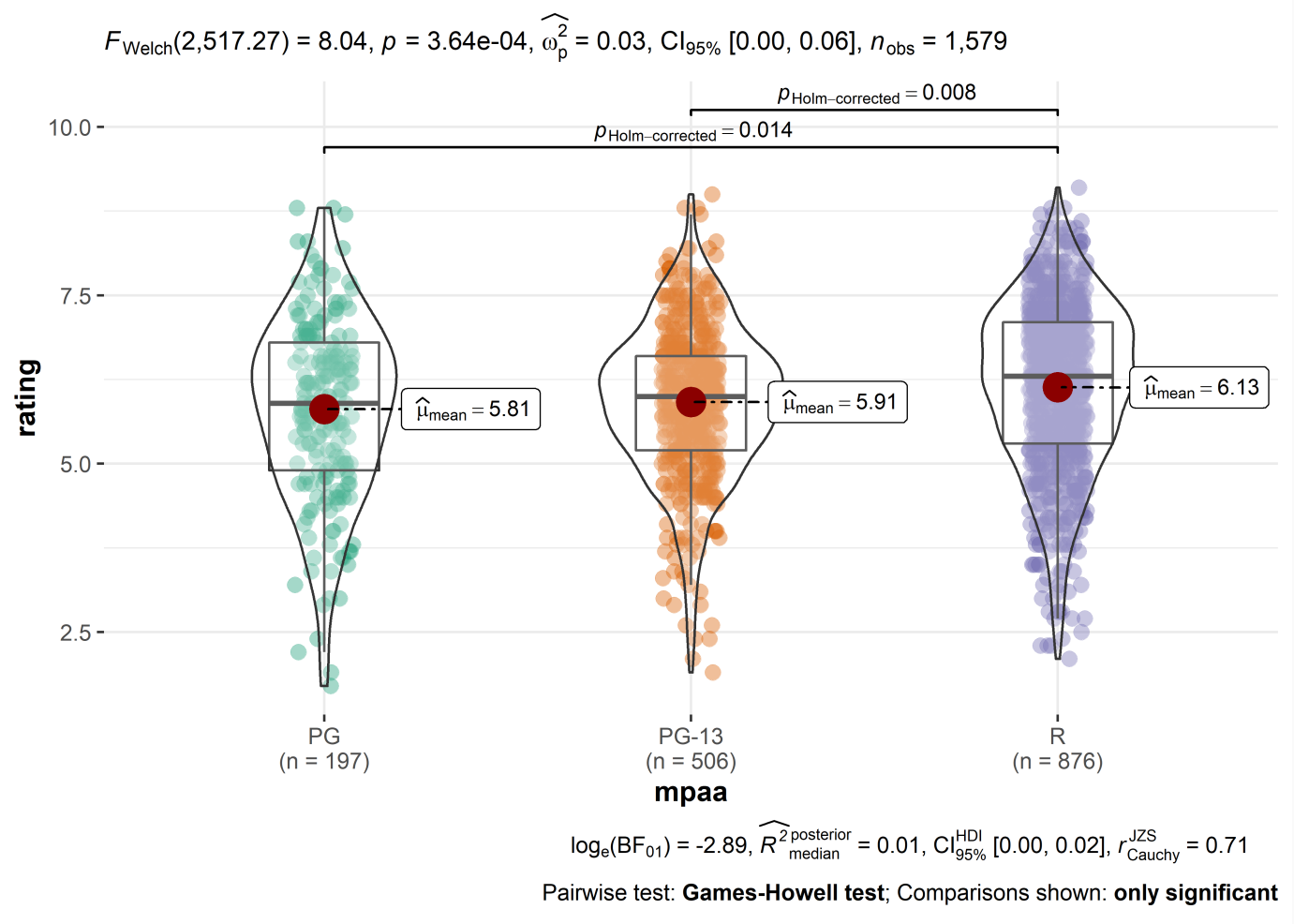
```
ggbetweenstats(
  data = movies_long,
  x = mpaa,
  y = rating
)
```

Function internally decides tests

- *t*-test if **2** groups
- ANOVA if **> 2** groups

Defaults return

- ✅ raw data + distributions
- ✅ descriptive statistics
- ✅ statistic + *p*-value
- ✅ effect size + CIs
- ✅ pairwise comparisons
- ✅ Bayesian hypothesis-testing
- ✅ Bayesian estimation



ggbetweenstats - pairwise comparisons

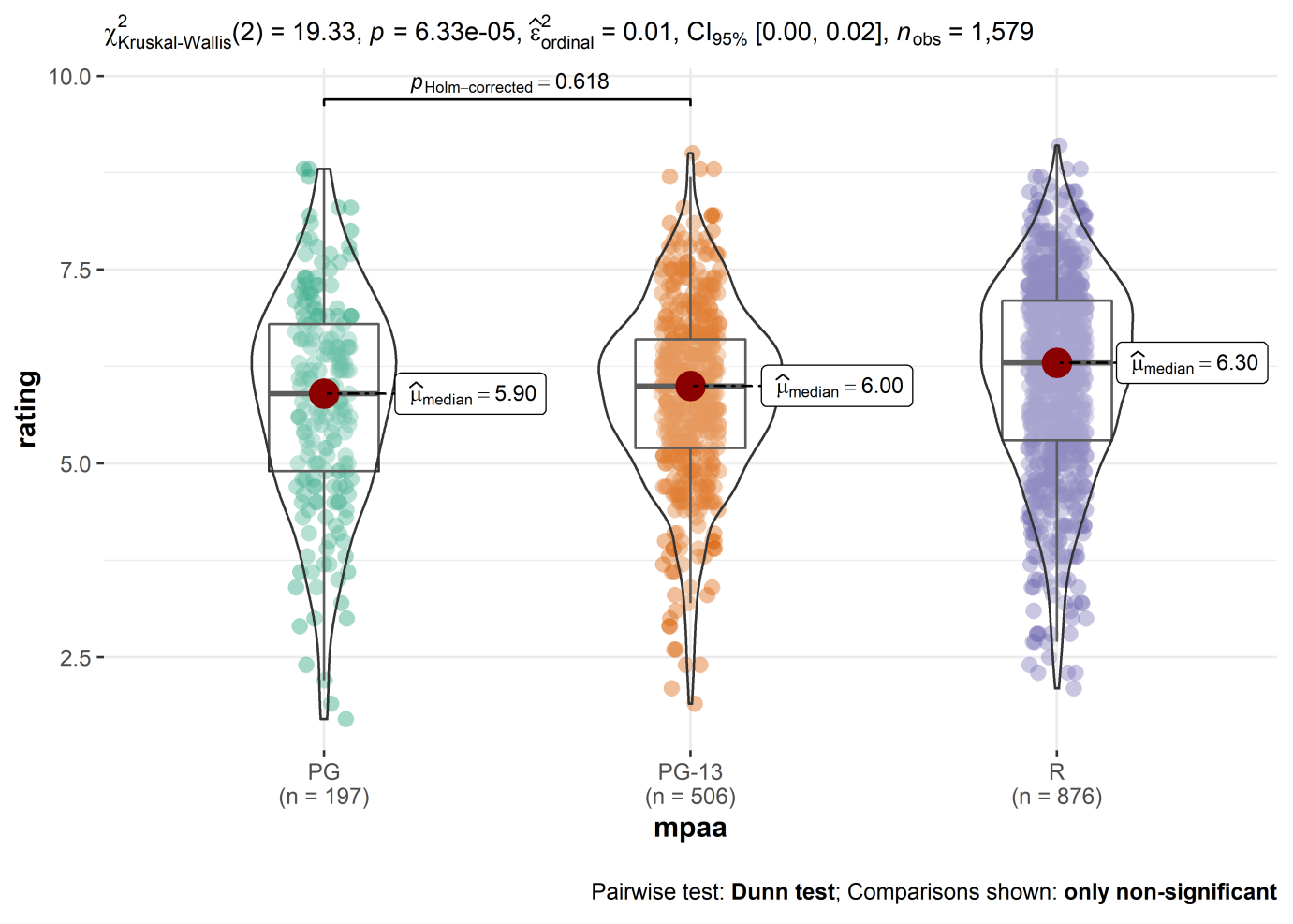
```
ggbetweenstats(
  data = movies_long,
  x = mpaa,
  y = rating,
  type = "np",
  pairwise.display = "ns"
)
```

Changing the `type` of test

- ☑ `"p"` → **parametric** (default)
- ☑ `"np"` → **non-parametric**
- ☑ `"r"` → **robust**
- ☑ `"bf"` → **Bayes Factor**

Changing pairwise comparisons displayed

- `"ns"` → only **non-significant**
- `"s"` → only **significant**
- `"all"` → **all**



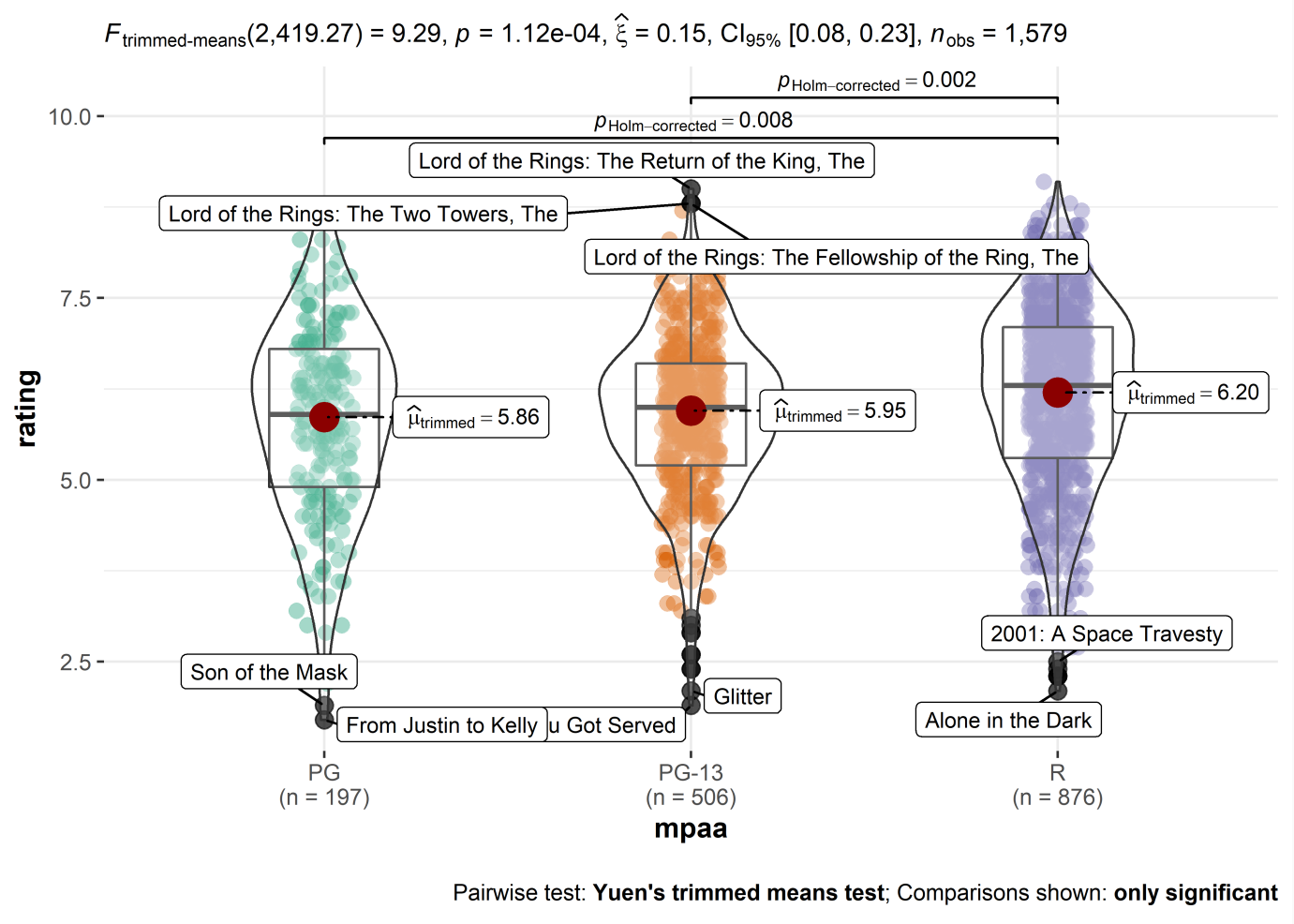
ggbetweenstats - outlier tagging

```
ggbetweenstats(
  data = movies_long,
  x = mpaa,
  y = rating,
  type = "r",
  outlier.tagging = TRUE,
  outlier.label = title
)
```

Tukey's fences method using interquartile range flags outliers.

Centrality measures

- ☑ `"p"` → μ_{mean}
- ☑ `"np"` → μ_{median}
- ☑ `"r"` → $\mu_{trimmed}$
- ☑ `"bf"` → μ_{MAP}



ggwithinstats - repeated measures equivalent

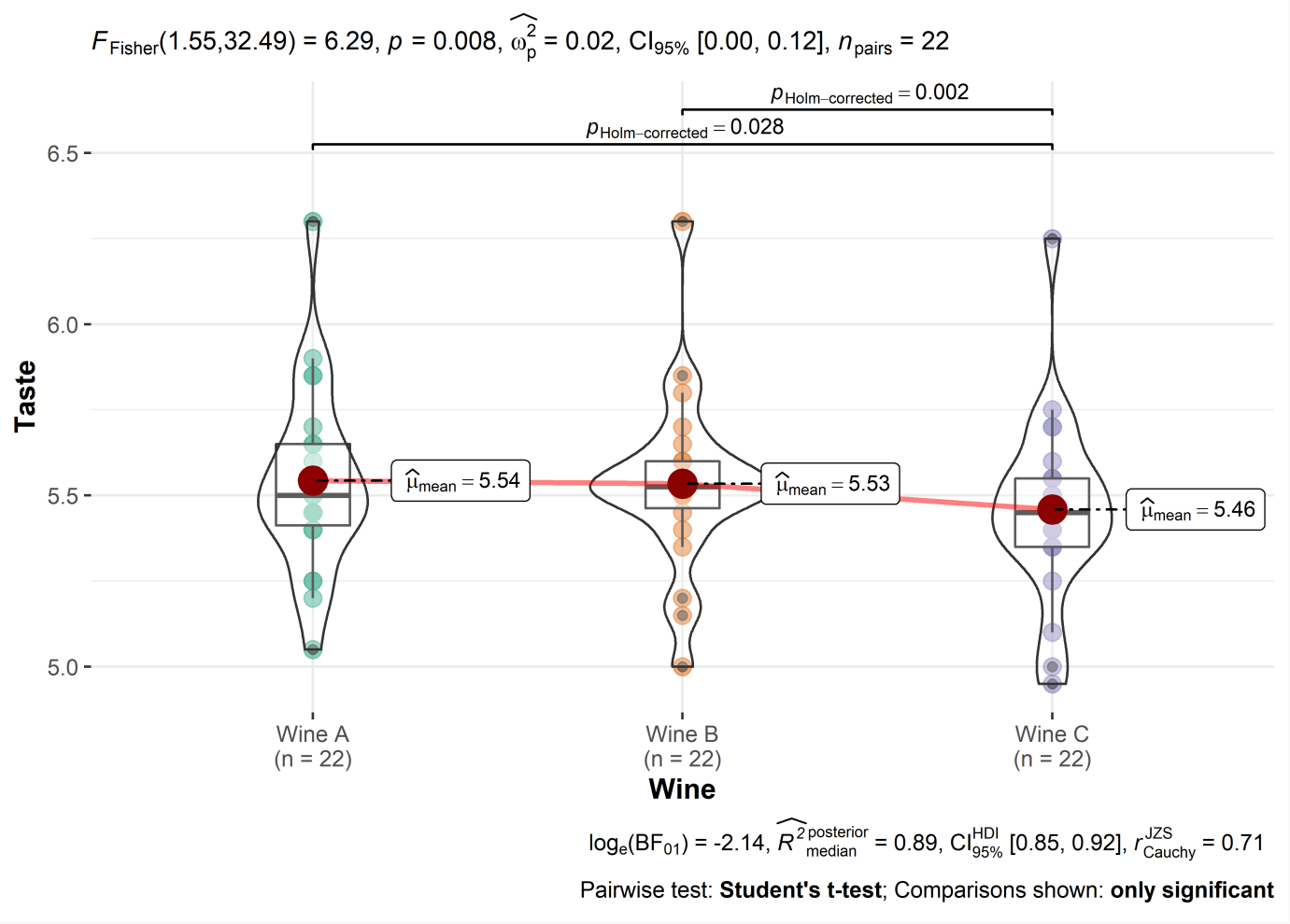
```
ggwithinstats(
  data = WRS2::WineTasting,
  x = Wine,
  y = Taste
)
```

Defaults return

- ✓ raw data + distributions
- ✓ descriptive statistics
- ✓ statistic + *p*-value
- ✓ effect size + CIs
- ✓ pairwise comparisons
- ✓ Bayesian hypothesis-testing
- ✓ Bayesian estimation

Changing the `type` of test

- ✓ `"p"` → **parametric** (default)
- ✓ `"np"` → **non-parametric**
- ✓ `"r"` → **robust**
- ✓ `"bf"` → **Bayes Factor**



gghistostats - Distribution of a numeric variable

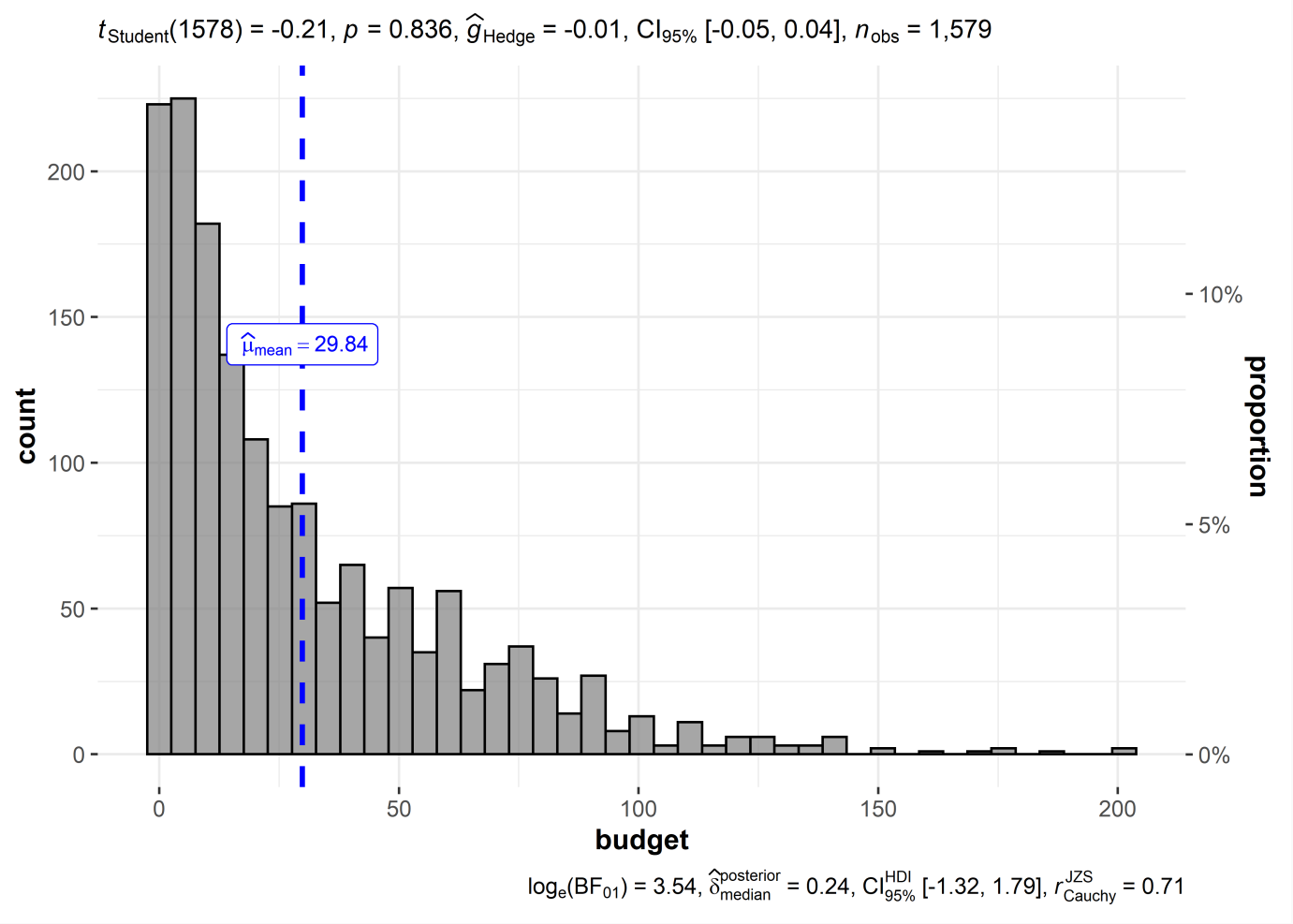
```
gghistostats(
  data = movies_long,
  x = budget,
  test.value = 30
)
```

Defaults return

- ☑ counts + proportion for bins
- ☑ descriptive statistics
- ☑ statistic + *p*-value
- ☑ effect size + CIs
- ☑ Bayesian hypothesis-testing
- ☑ Bayesian estimation

Centrality measures

- ☑ `"p"` → μ_{mean}
- ☑ `"np"` → μ_{median}
- ☑ `"r"` → $\mu_{trimmed}$
- ☑ `"bf"` → μ_{MAP}



ggdotplotstats - Labeled numeric variable

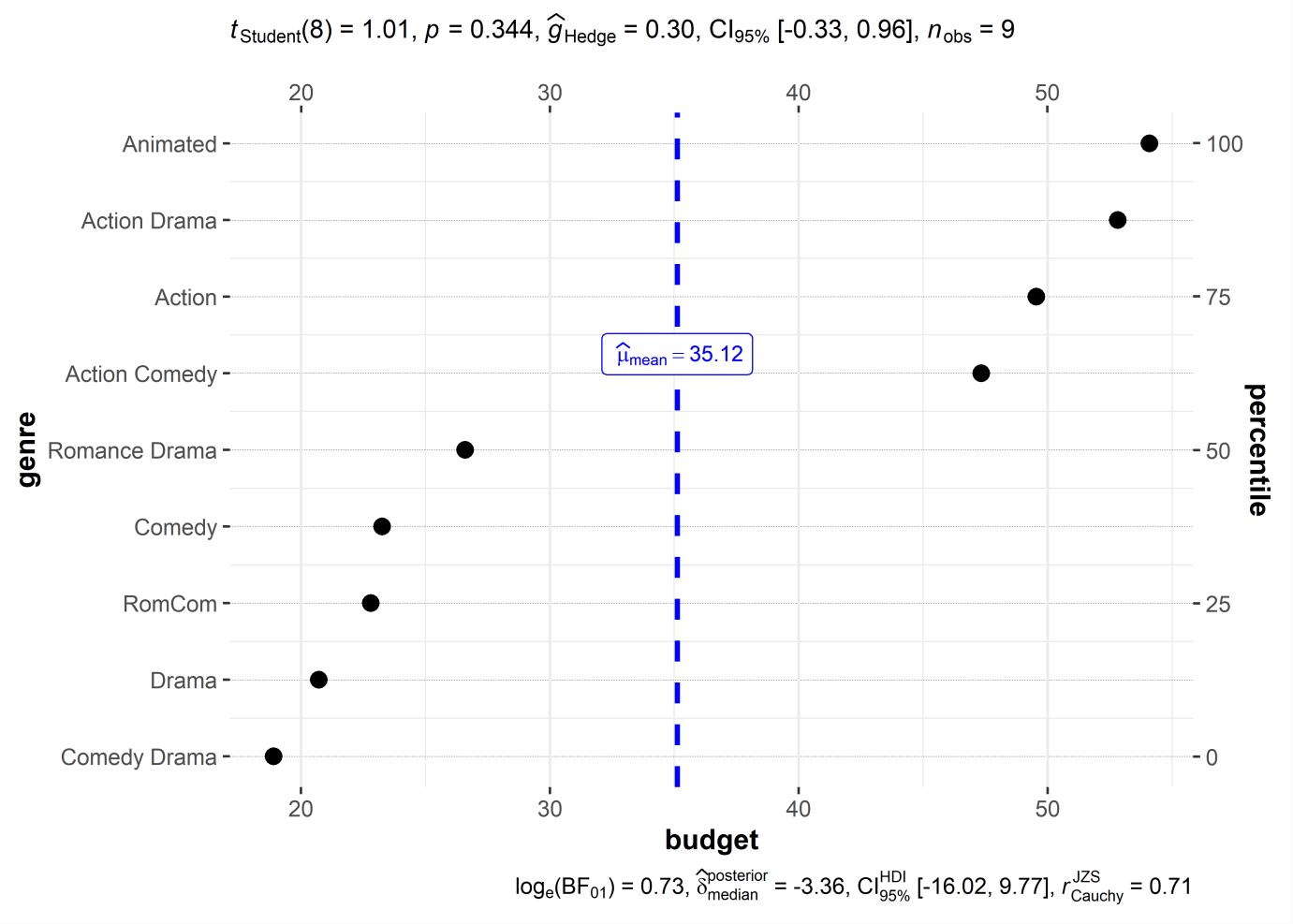
```
ggdotplotstats(
  data = movies_long,
  x = budget,
  y = genre,
  test.value = 30
)
```

Defaults return

- ✓ descriptive statistics
- ✓ statistic + *p*-value
- ✓ effect size + CIs
- ✓ Bayesian hypothesis-testing
- ✓ Bayesian estimation

Centrality measures

- ✓ `"p"` → μ_{mean}
- ✓ `"np"` → μ_{median}
- ✓ `"r"` → $\mu_{trimmed}$
- ✓ `"bf"` → μ_{MAP}



Hypothesis about correlation

- *ggscatterstats*: Two numeric variables
- *ggcorrmat*: Multiple numeric variables

ggscatterstats - Two numeric variables

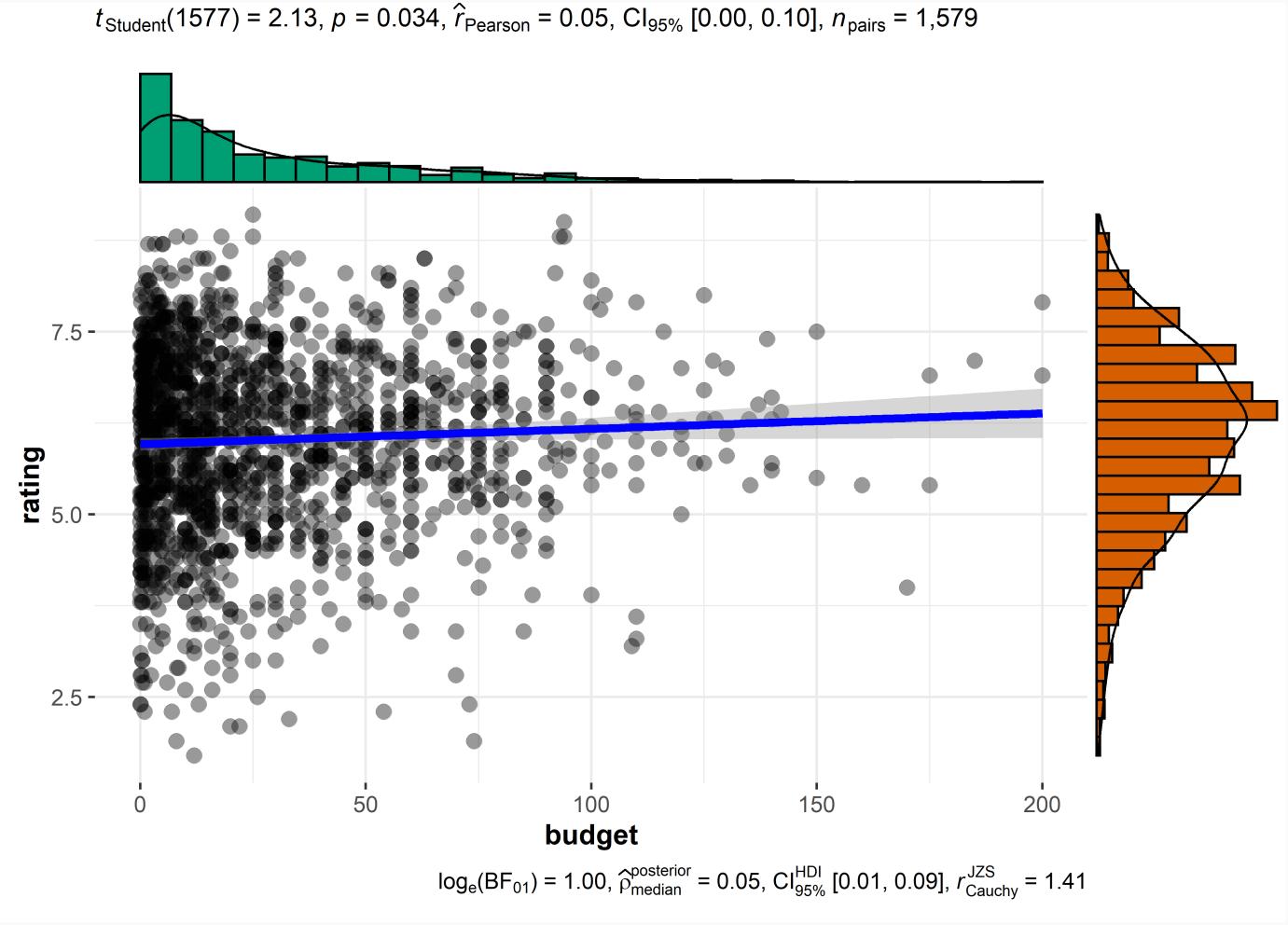
```
ggscatterstats(
  data = movies_long,
  x = budget,
  y = rating
)
```

Defaults return

- ✓ raw data + distributions
- ✓ marginal distributions
- ✓ statistic + *p*-value
- ✓ effect size + CIs
- ✓ Bayesian hypothesis-testing
- ✓ Bayesian estimation

Changing the `type` of test

- ✓ `"p"` → **parametric** (default)
- ✓ `"np"` → **non-parametric**
- ✓ `"r"` → **robust**
- ✓ `"bf"` → **Bayes Factor**

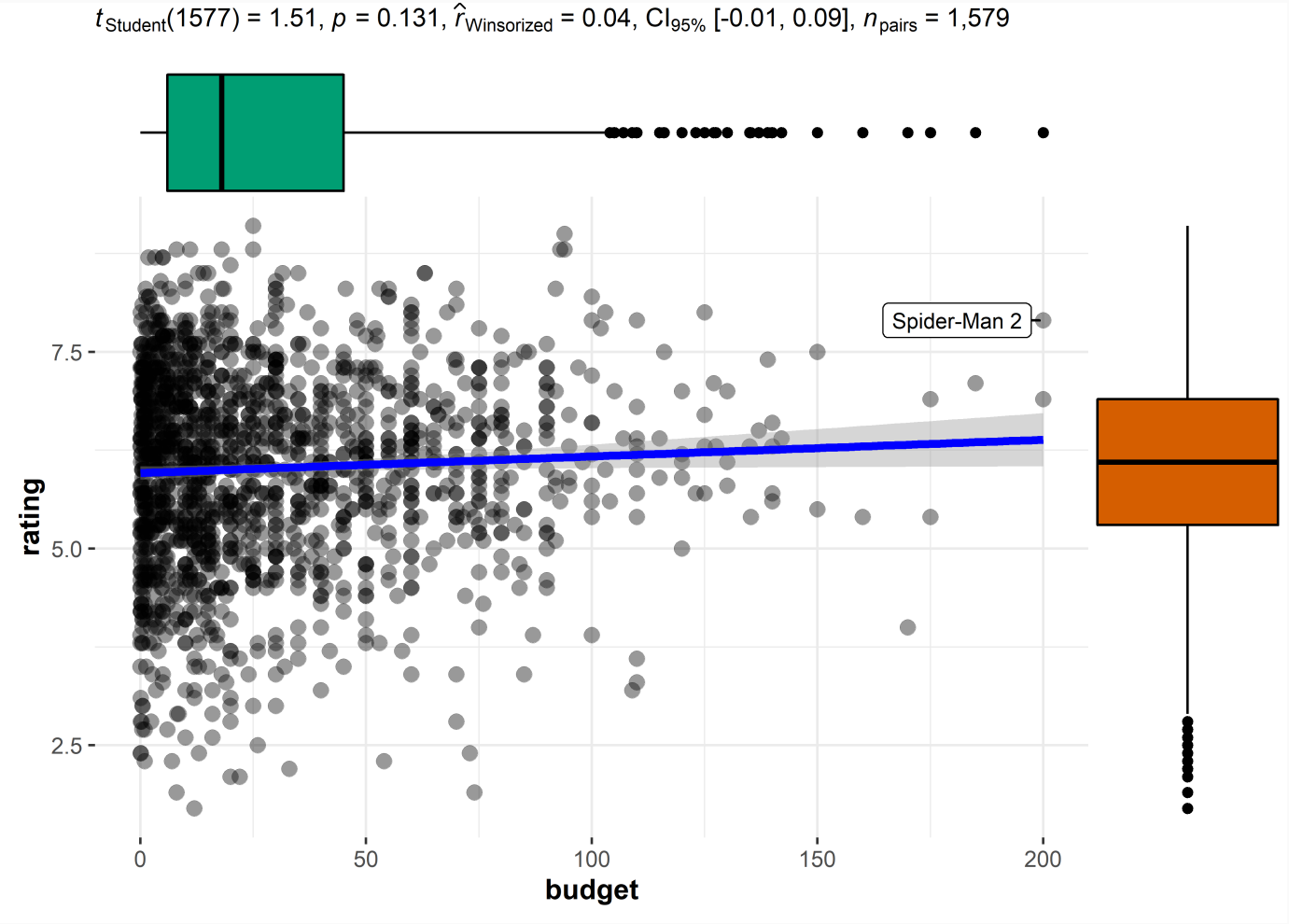


ggscatterstats - conditional point tagging

```
ggscatterstats(
  data = movies_long,
  x = budget,
  y = rating,
  type = "r",
  label.var = title,
  label.expression = budget > 150
  & rating > 7.5,
  marginal.type = "boxplot"
)
```

Changing the marginal type

- **histogram**
- **boxplot**
- **density**
- **violin**
- **densigram**



ggcorrmat - Multiple numeric variables

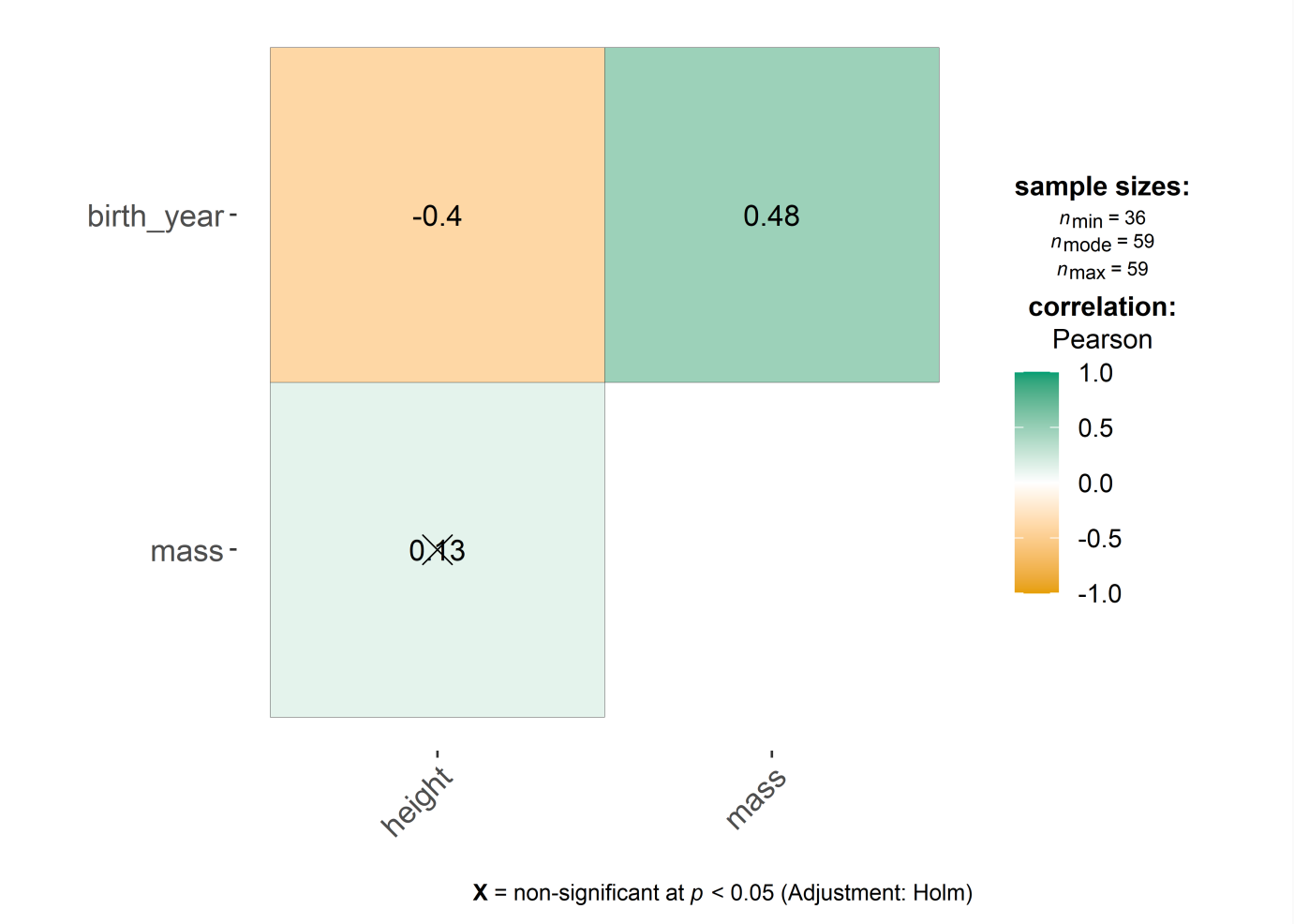
```
ggcorrmat(dplyr::starwars)
```

Defaults return

- ✓ effect size + significance
- ✓ careful handling of `NA`s

Changing the `type` of test

- ✓ `"p"` → **parametric** (default)
- ✓ `"np"` → **non-parametric**
- ✓ `"r"` → **robust**
- ✓ `"bf"` → **Bayes Factor**



ggcorrmat - getting a dataframe

In addition to `output = "plot"`, this function can also be used to get a dataframe:

```
library(ggplot2) # for data

ggcorrmat(
  data = dplyr::select(msleep, sleep_rem, awake, brainwt),
  type = "bf",
  output = "dataframe"
)
```

parameter1	parameter2	estimate	conf.level	conf.low	conf.high	pd	rope.percentage	prior.distribution	prior.location	prior.scale	bayes.factor	method	n.obs
sleep_rem	awake	-0.7336828	0.95	-0.8234833	-0.6313331	1.000	0.000	beta	1.414427	1.414427	3.005547e+09	Bayesian Pearson correlation	61
sleep_rem	brainwt	-0.2078193	0.95	-0.4302893	0.0089833	0.926	0.204	beta	1.414427	1.414427	6.539296e-01	Bayesian Pearson correlation	48
awake	brainwt	0.3424784	0.95	0.1737942	0.5447638	0.997	0.026	beta	1.414427	1.414427	7.292097e+00	Bayesian Pearson correlation	56

Hypothesis of composition of categorical variables

🔭 *ggpiestats*: If you like 🥧

🔭 *ggbarstats*: Otherwise

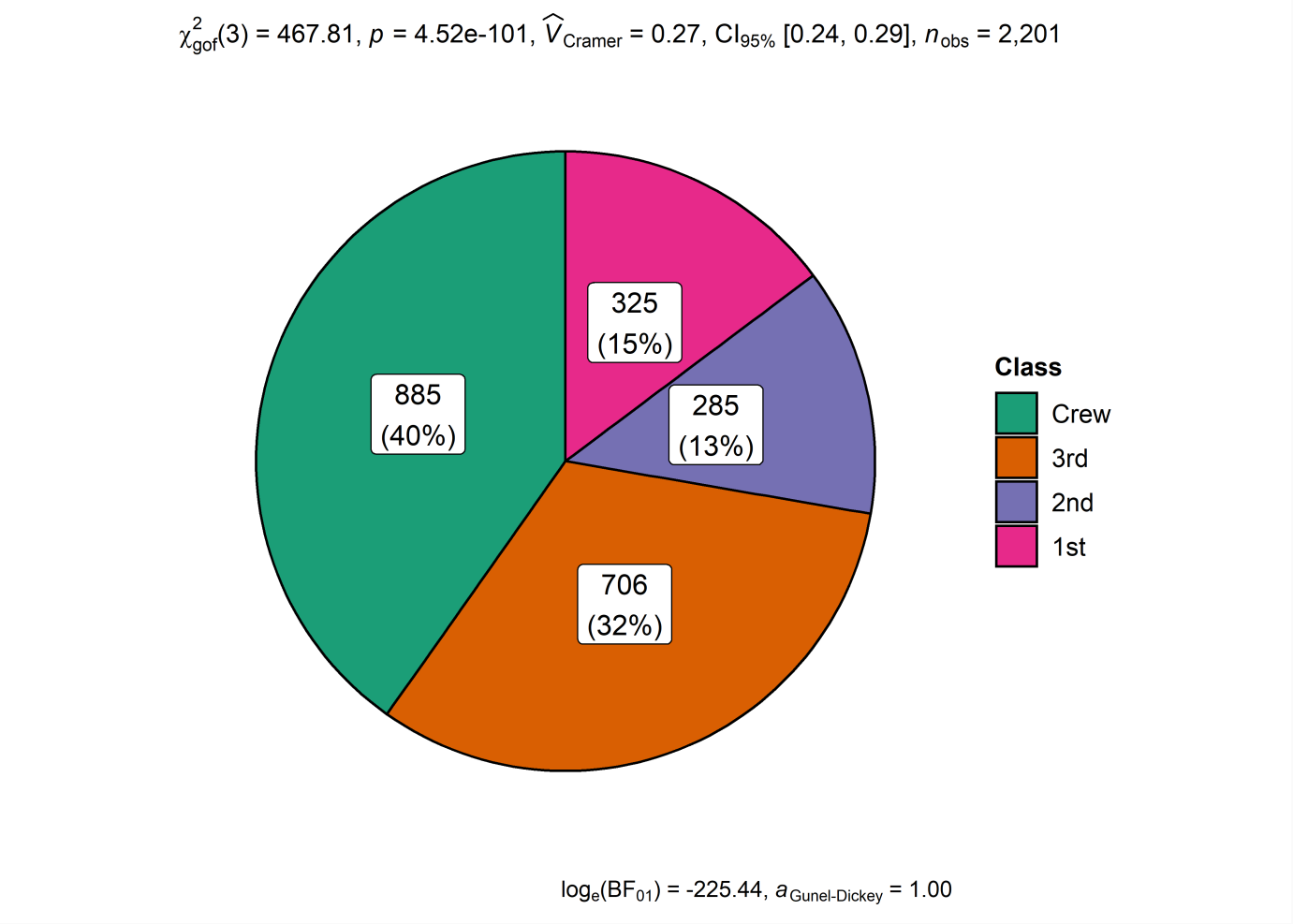
ggpiestats - goodness-of-fit test

```
ggpiestats(
  data = as.data.frame(Titanic),
  x = Class,
  counts = Freq,
  label = "both"
)
```

Note: If the data is in *tabled* format, you can use the `counts` argument.

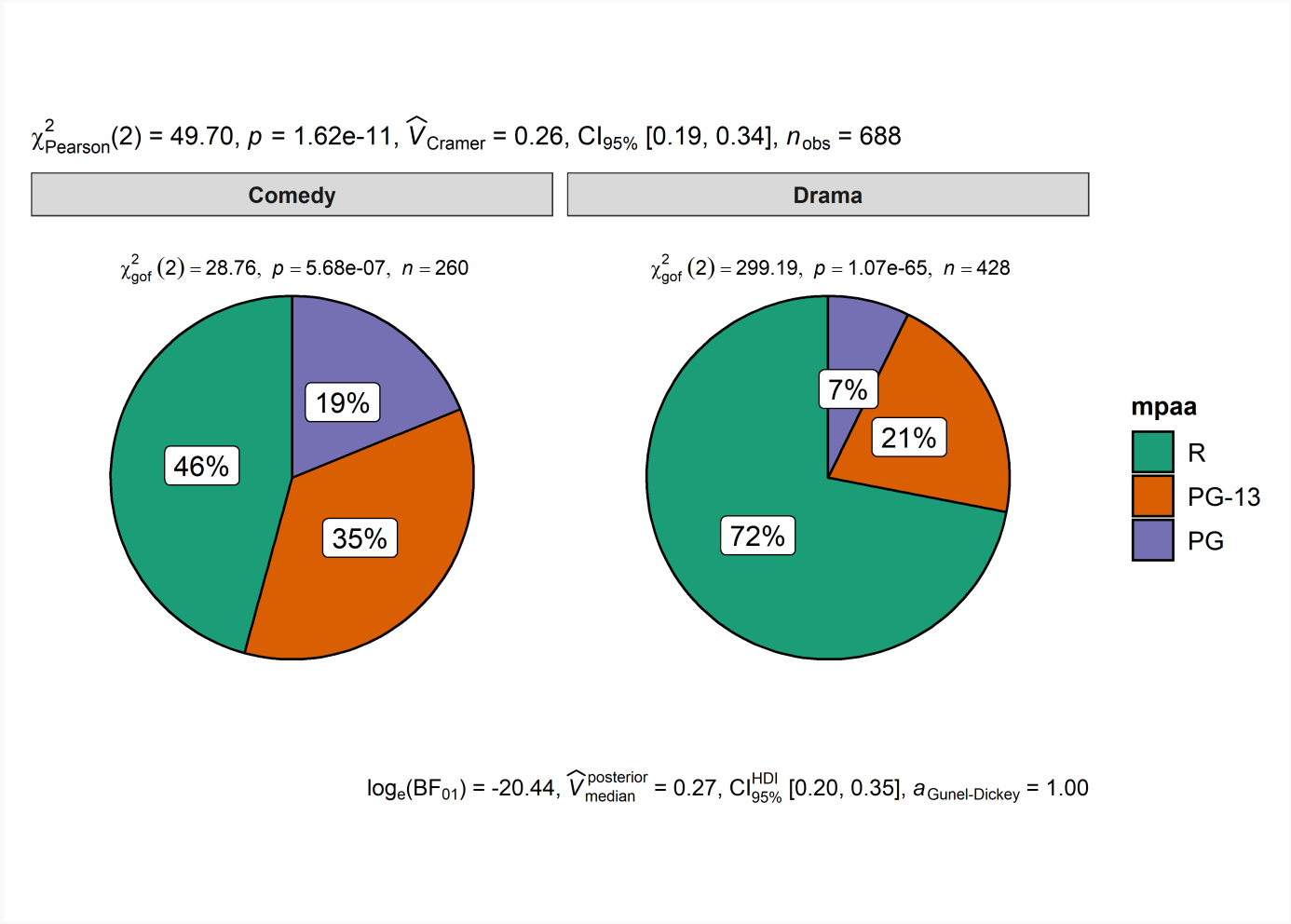
Test by analysis

- `y ≠ NULL` → contingency table
- `y = NULL` → goodness-of-fit



ggpiestats - association between categorical variables

```
# let's use subset of data
ggpiestats(
  data = dplyr::filter(
    .data = movies_long,
    genre %in% c("Drama", "Comedy")
  ),
  x = mpaa,
  y = genre
)
```



Defaults return

- ✅ descriptive statistics
- ✅ statistic + *p*-value
- ✅ effect size + CIs
- ✅ Goodness-of-fit tests
- ✅ Bayesian hypothesis-testing
- ✅ Bayesian estimation

Test by design

- `paired = FALSE` → Pearson's χ^2
- `paired = TRUE` → McNemar's χ^2

ggbarstats - association between categorical variables

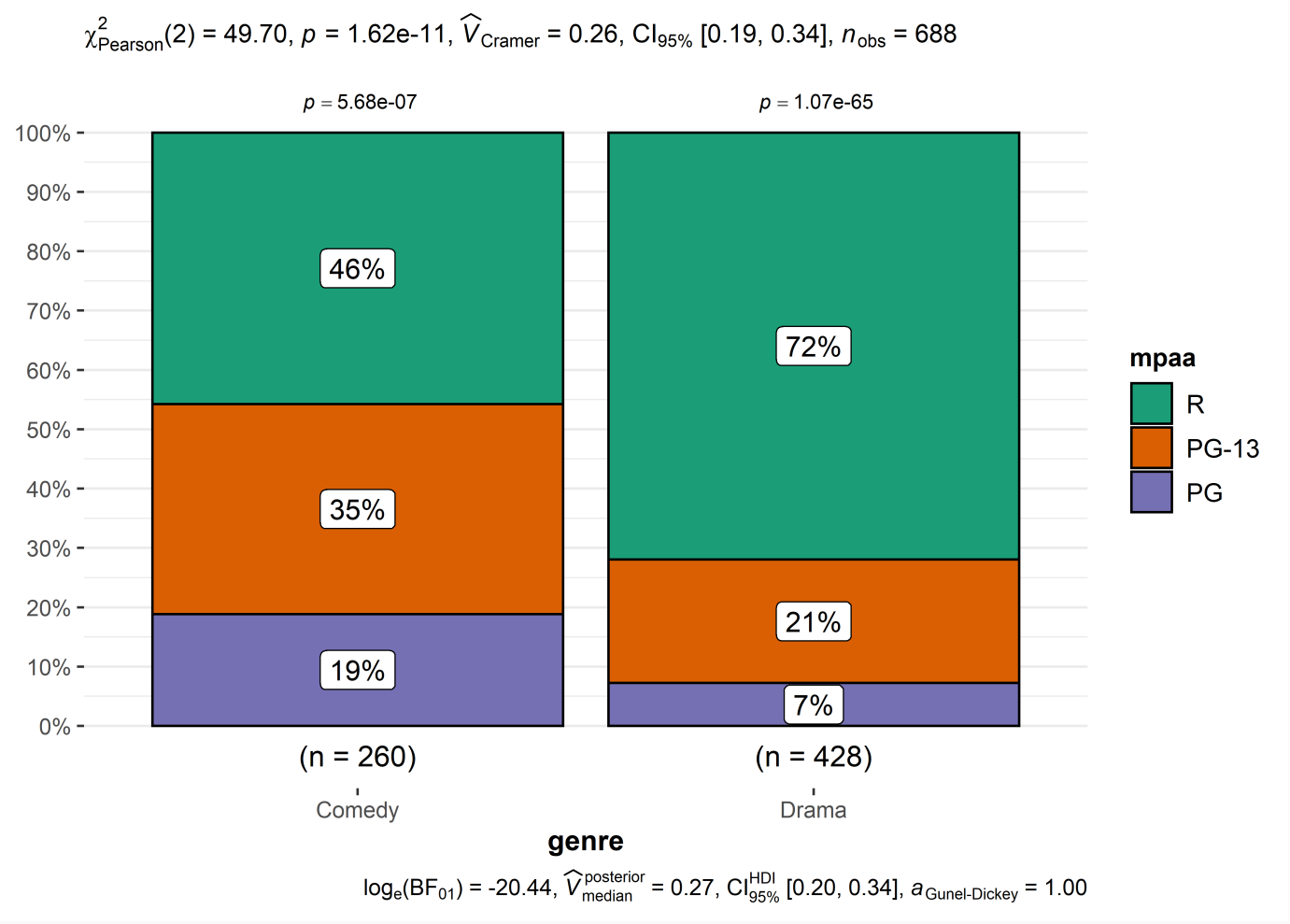
```
ggbarstats(
  data = dplyr::filter(
    .data = movies_long,
    genre %in% c("Drama", "Comedy")
  ),
  x = mpaa,
  y = genre
)
```

Defaults return

- ✅ descriptive statistics
- ✅ statistic + *p*-value
- ✅ effect size + CIs
- ✅ Goodness-of-fit tests
- ✅ Bayesian hypothesis-testing
- ✅ Bayesian estimation

Test by design

- `paired = FALSE` → Pearson's χ^2
- `paired = TRUE` → McNemar's χ^2



Hypothesis about regression cofficients

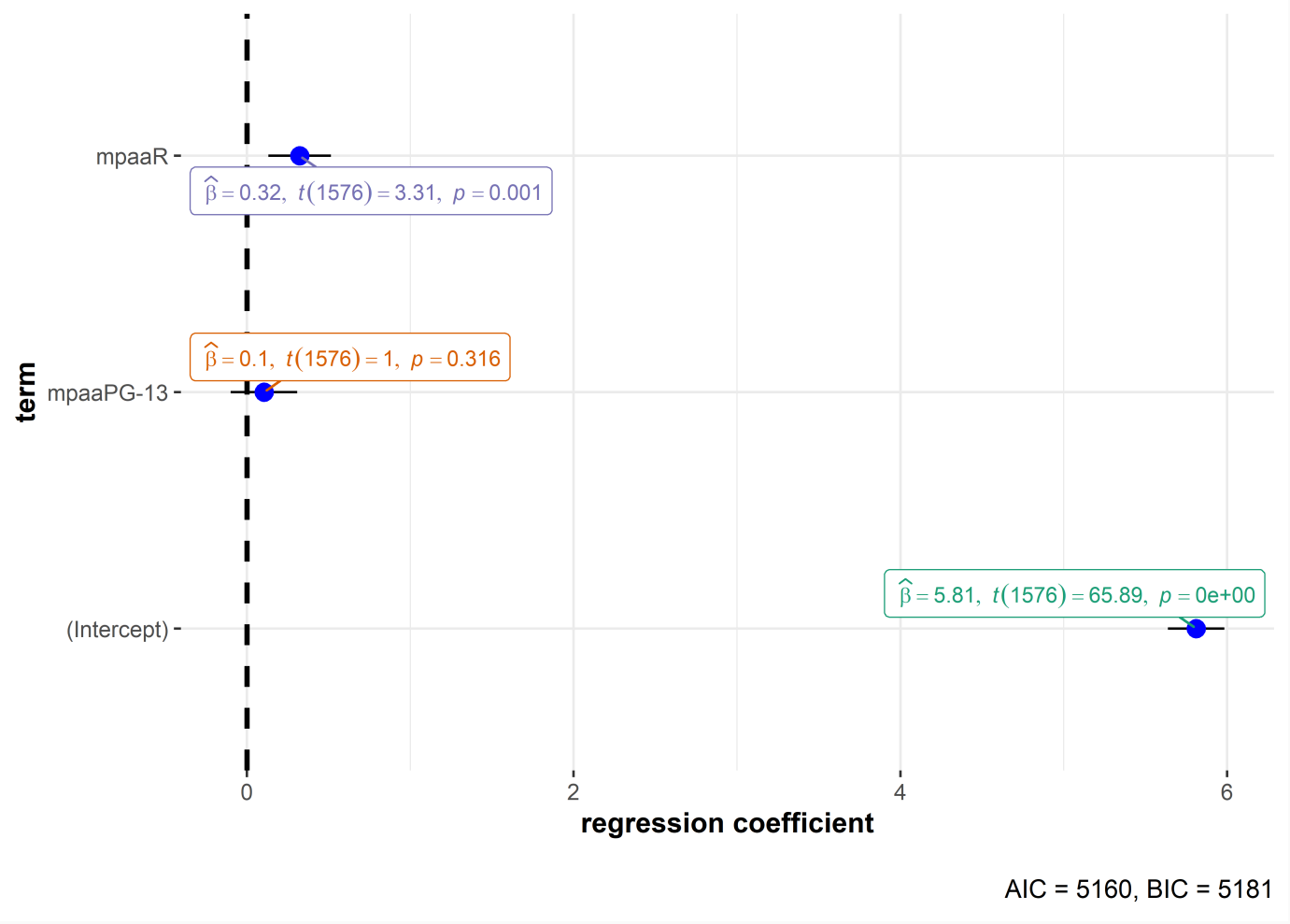
 *ggcoefstats*: Any regression model object

ggcoefstats

```
# model
mod ← stats::lm(
  formula = rating ~ mpaa,
  data = movies_long
)

# plot
ggcoefstats(mod)
```

Defaults return

- ☑ estimate + CIs
- ☑ statistic + *p*-value
- ☑ model summary (AIC + BIC)



ggcoefstats: Supported models

`aareg`, `anova`, `aov`, `aovlist`, `Arima`, `bam`, `bayesx`, `bayesGARCH`, `BBmm`, `BBreg`, `bcplm`, `betamfx`, `betaor`, `BFBayesFactor`, `bglmerMod`, `bife`, `bigglm`, `biglm`, `blavaan`, `bmlm`, `blmerMod`, `bracl`, `brglm`, `brglm2`, `brmsfit`, `brmultinom`, `btergm`, `cch`, `censReg`, `cgam`, `cgamm`, `cglm`, `clm`, `clm2`, `clmm`, `clmm2`, `coeftest`, `complmrob`, `confusionMatrix`, `coxme`, `coxph`, `coxph.penal`, `cpglm`, `cpglmm`, `crch`, `crq`, `DirichReg`, `drc`, `emmGrid`, `epi.2by2`, `ergm`, `feis`, `felm`, `fitdistr`, `fixest`, `flexsurvreg`, `gam`, `Gam`, `gamlss`, `garch`, `geeglm`, `glmc`, `glmerMod`, `glmmTMB`, `gls`, `glht`, `glm`, `glmm`, `glmmadmb`, `glmmPQL`, `glmRob`, `glmrob`, `glmx`, `gmm`, `HLfit`, `hurdle`, `ivFixed`, `ivprobit`, `ivreg`, `iv_robust`, `lavaan`, `lm`, `lm.beta`, `lmerMod`, `lmerModLmerTest`, `lmodel2`, `lmRob`, `lmrob`, `lm_robust`, `logitmfx`, `logitor`, `logitsf`, `LORgee`, `lqm`, `lqmm`, `lrm`, `manova`, `maov`, `margins`, `mcmc`, `mcmc.list`, `MCMCglmm`, `mclogit`, `mice`, `mmclogit`, `mediate`, `metafor`, `merMod`, `merModList`, `metaplus`, `mixor`, `mjoint`, `mle2`, `mlm`, `multinom`, `negbin`, `negbinmfx`, `negbinirr`, `nlmerMod`, `nlrq`, `nlreg`, `nls`, `orcutt`, `orm`, `plm`, `poissonmfx`, `poissonirr`, `polr`, `ridgelm`, `riskRegression`, `rjags`, `rlm`, `rlmerMod`, `robmixglm`, `rq`, `rqss`, `rrvglm`, `scam`, `semLm`, `semLme`, `slm`, `speedglm`, `speedlm`, `stanfit`, `stanreg`, `survreg`, `svyglm`, `svyolr`, `svyglm`, `tobit`, `truncreg`, `vgam`, `vglm`, `wbgee`, `wblm`, `zeroinfl`, etc.

Thanks to `easystats`!

*grouped_* variants of all functions

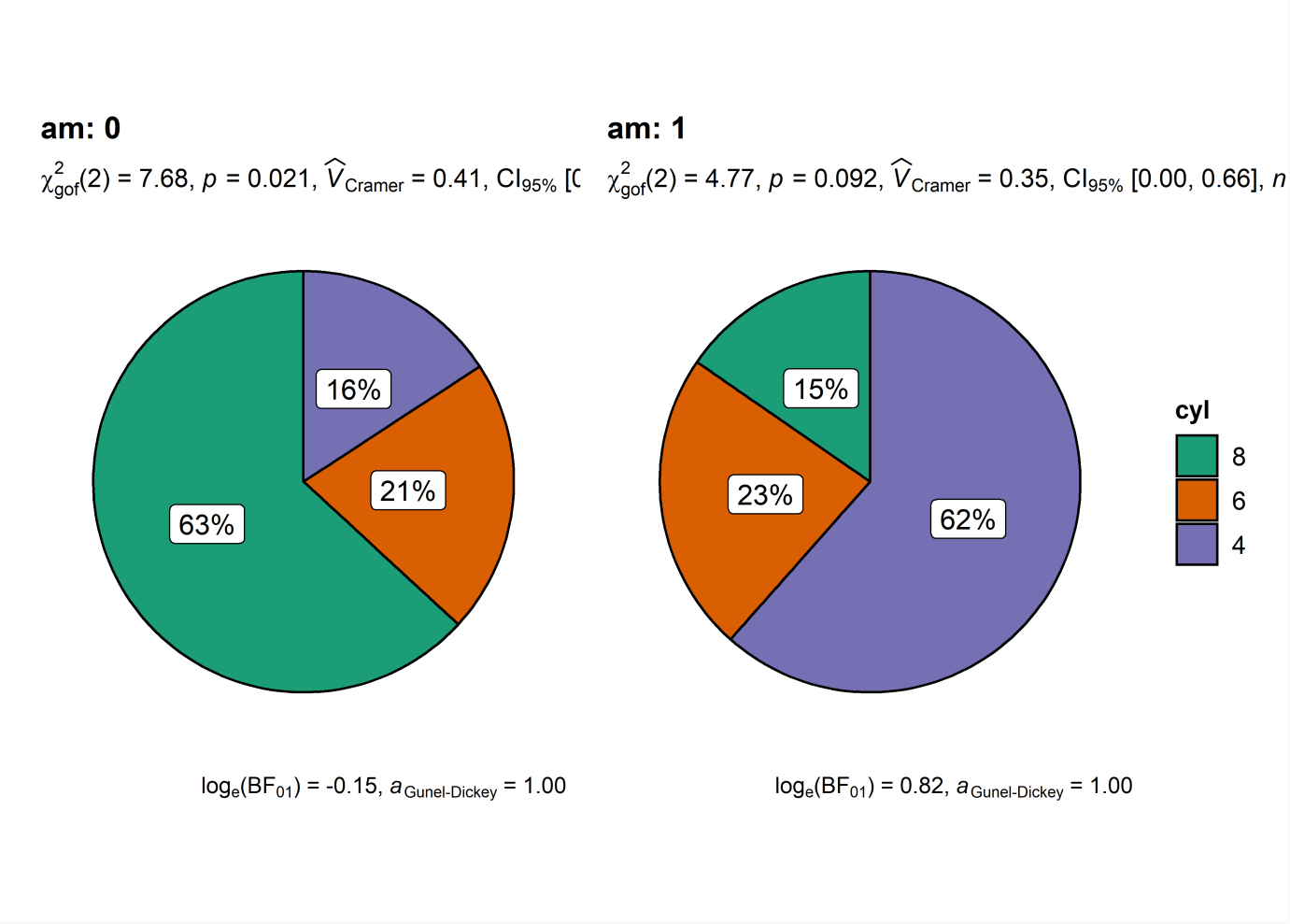
Running the same function for all levels of a single grouping variable

*grouped_* functions

```
grouped_ggpiestats(
  data = mtcars,
  x = cyl,
  grouping.var = am
)
```

Available `grouped_` variants

- `grouped_ggbetweenstats`
- `grouped_ggwithinstats`
- `grouped_gghistostats`
- `grouped_ggdotplotstats`
- `grouped_ggscatterstats`
- `grouped_ggcorrmat`
- `grouped_ggpiestats`
- `grouped_ggbarstats`

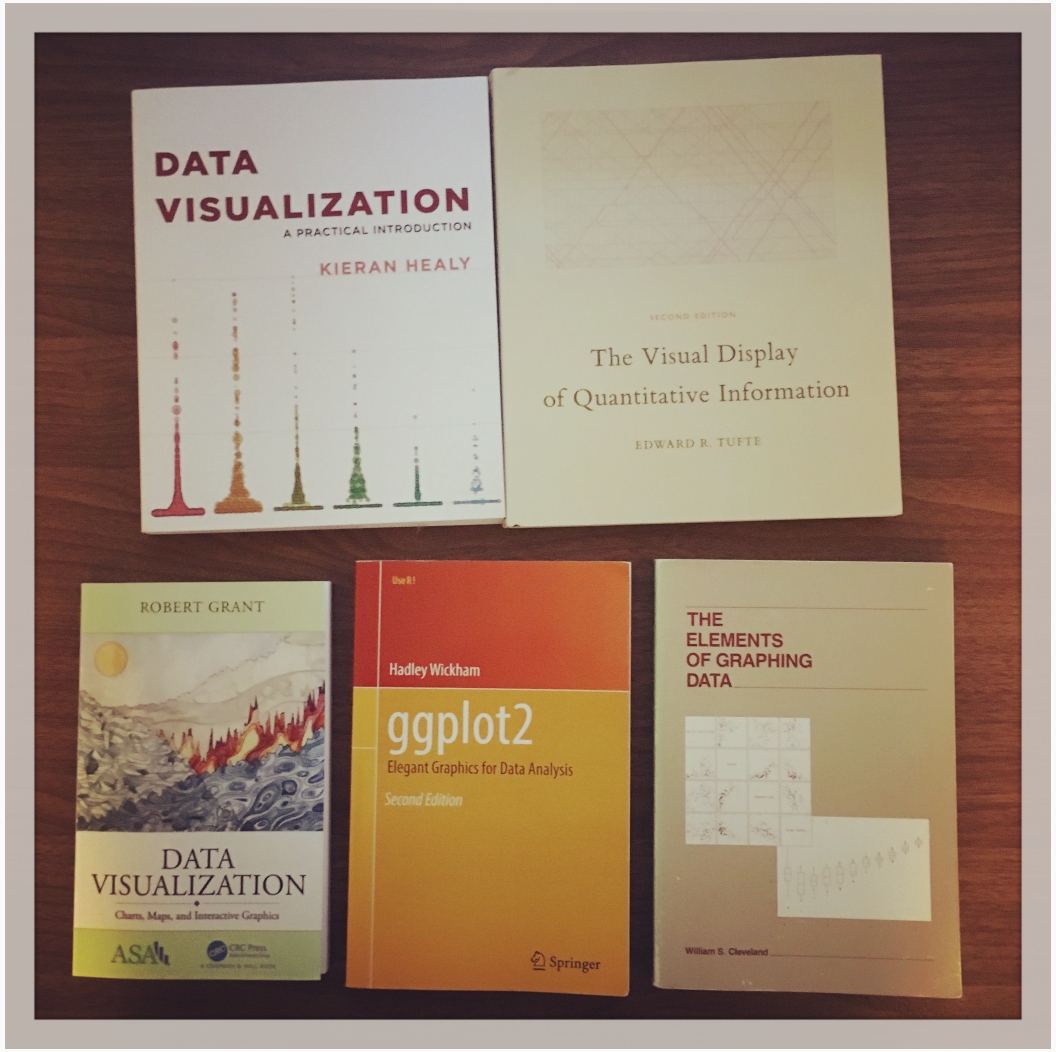


Customizability of *ggstatsplot*

"What if I don't like the default plots?" 🤔

Defaults

The default plots in *ggstatsplot* are **opinionated**, yes, but an attempt has been made to make sure that they follow best practices outlined in the data visualization research.

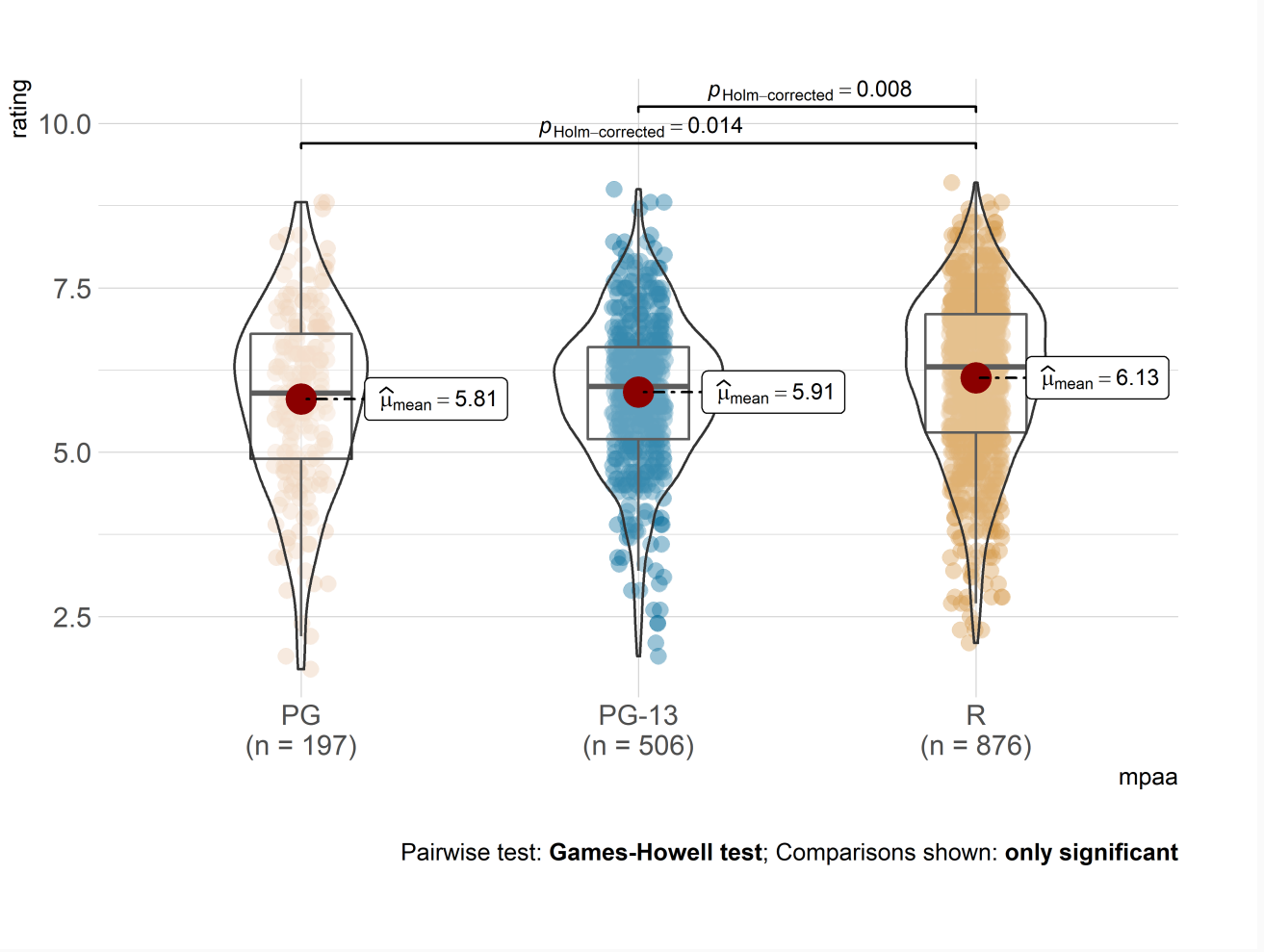


Changing aesthetics (themes + palettes)

```
ggbetweenstats(
  data = movies_long,
  x = mpaa,
  y = rating,
  results.subtitle = FALSE,
  ggtheme = hrbrthemes::theme_ipsum_tw(),
  palette = "Darjeeling2",
  package = "wesanderson"
)
```

Aesthetic preferences are not an excuse to not use `ggstatsplot`!

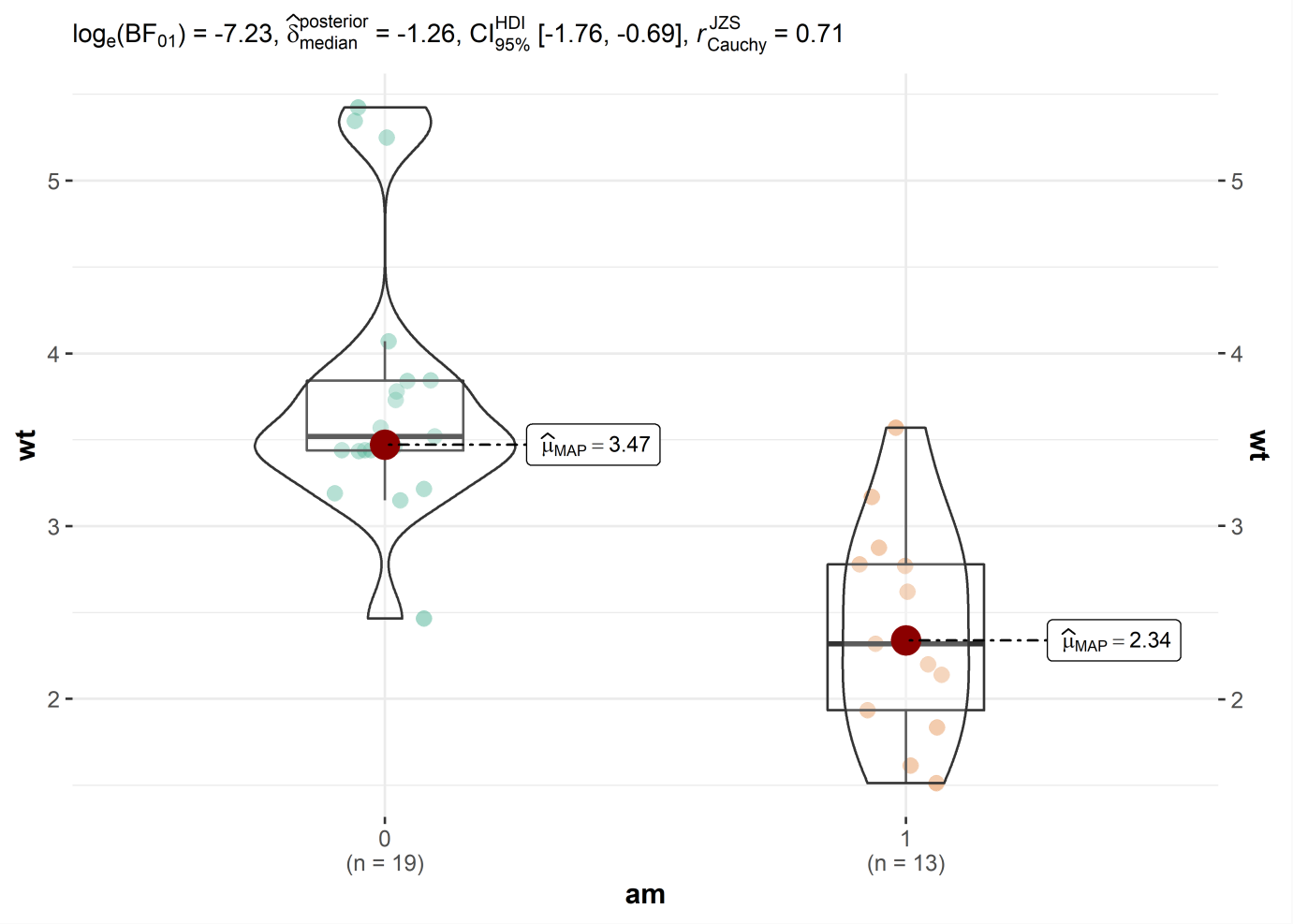
The default palette is **colorblind-friendly**.



Further modification with *ggplot2*

```
ggbetweenstats(
  data = mtcars,
  x = am,
  y = wt,
  type = "bf"
) +
  scale_y_continuous(sec.axis = dup_axis())
```

You can modify `ggstatsplot` plots further using `ggplot2` functions. Yaay!



Extract expressions for custom plots

`ggstatsplot` can also be used to get only the statistical expressions.

```
# using `ggstatsplot` for stats
results ←
  ggstatsplot::ggpiestats(
    data = Titanic_full,
    x = Survived,
    y = Sex,
    output = "subtitle"
  )

# using `ggiraphExtra` for plot
ggiraphExtra::ggSpine(
  data = Titanic_full,
  aes(x = Sex, fill = Survived),
  addlabel = TRUE,
  interactive = FALSE
) + labs(subtitle = results)
```

Why use *ggstatsplot*?

Summary of benefits

Supports different statistical approaches

Functions	Description	Parametric	Non-parametric	Robust	Bayes Factor
`ggbetweenstats`	Between group comparisons	Yes	Yes	Yes	Yes
`ggwithinstats`	Within group 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

Toggling between statistical approaches

Parametric

```
# anova
ggbetweenstats(
  data = mtcars,
  x = cyl,
  y = wt,
  type = "p"
)

# correlation analysis
ggscatterstats(
  data = mtcars,
  x = wt,
  y = mpg,
  type = "p"
)

# t-test
gghistostats(
  data = mtcars,
  x = wt,
  test.value = 2,
  type = "p"
)
```

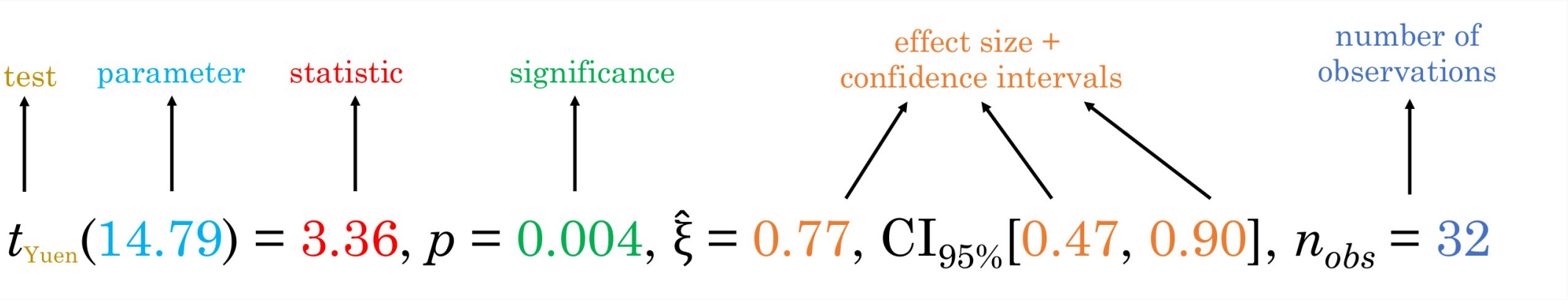
Non-parametric

```
# anova
ggbetweenstats(
  data = mtcars,
  x = cyl,
  y = wt,
  type = "np"
)

# correlation analysis
ggscatterstats(
  data = mtcars,
  x = wt,
  y = mpg,
  type = "np"
)

# t-test
gghistostats(
  data = mtcars,
  x = wt,
  test.value = 2,
  type = "np"
)
```

Best practices in statistical reporting

For all statistical tests reported in the plots, the expression template mostly abides by the APA gold standard for statistical reporting.

For example, here are results from a robust *t*-test:



Statistically informed tests defaults

The default tests follow the best practices. For example,

✅ `ggbetweenstats` and `ggwithinstats` default to Welch's *t*-test and Welch's ANOVA - and not Student's *t*-test and Fisher's ANOVA - based on recent work (Delacre et al., 2017, 2018).

✅ Functions default to reporting unbiased effect size measures (Lakens, 2013).

✅ Whenever multiple tests are carried out, *p*-values are adjusted for them by default.

etc.

Avoiding reporting errors

Since the plot and the statistical analysis are yoked together, the chances of making an error in reporting the results are minimized.

degrees of freedom. One in eight papers contained a grossly inconsistent *p*-value that may have affected the statistical conclusion. In contrast to earlier findings, we found that the average prevalence of inconsistent *p*-values has been stable over the years or has declined. The prevalence of gross inconsistencies was higher in *p*-values reported as significant than in *p*-values reported as nonsignificant. This could indicate a systematic bias in favor of significant results. Possible solutions for the

(Nuijten et al., *Behavior Research Methods*, 2016)

Making sense of null results

Combination of frequentist and Bayesian statistics for each analysis to properly interpret the null results.

Abstract

In the traditional statistical framework, nonsignificant results leave researchers in a state of suspended disbelief. In this study, we examined, empirically, the treatment and evidential impact of nonsignificant results. Our specific goals were twofold: to explore how psychologists interpret and communicate nonsignificant results and to assess how much these results constitute evidence in favor of the null hypothesis. First, we examined all nonsignificant findings mentioned in the abstracts of the 2015 volumes of *Psychonomic Bulletin & Review*, *Journal of Experimental Psychology: General*, and *Psychological Science* ($N = 137$). In 72% of these cases, nonsignificant results were misinterpreted, in that the authors inferred that the effect was absent. Second, a Bayes factor reanalysis revealed that fewer than 5% of the nonsignificant findings provided strong evidence (i.e., $BF_{01} > 10$) in favor of the null hypothesis over the alternative hypothesis. We recommend that researchers expand their statistical tool kit in order to correctly interpret nonsignificant results and to be able to evaluate the evidence for and against the null hypothesis.

(Aczel et al., *AMPPS*, 2018)

Summary of benefits

- No need to use scores of packages for statistical analysis (e.g., one to get stats, one to get effect sizes, another to get Bayes Factors, and yet another to get pairwise comparisons, etc.).
- Minimal amount of code needed for all functions (typically only `data`, `x`, and `y`), which minimizes chances of error and makes for tidy scripts.
- Conveniently toggle between statistical approaches.
- Truly makes your figures worth a thousand words.
- No need to copy-paste results to the text editor (MS-Word, e.g.).
- Disembodied figures stand on their own and are easy to evaluate for the reader.
- More breathing room for theoretical discussion and other text.
- No need to worry about updating figures and statistical details separately.

Misconceptions and limitations

Misconceptions about *ggstatsplot*

This package is...

❌ an alternative to learning `ggplot2`
✅ (The better you know `ggplot2`, the more you can modify the defaults to your liking.)

❌ meant to be used in talks/presentations
✅ (Default plots can be too complicated for effectively communicating results in time-constrained presentation settings, e.g. conference talks.)

❌ the only game in town
✅ (GUI software alternatives: JASP and jamovi).

Limitations of *ggstatsplot*

- Limited kinds of plots available.
- Limited number of statistical tests available.
  This will *always* be the case.
- Although beginner-friendly to use, it expects a non-trivial level of statistical proficiency (but only plots without statistics can also be used).
- Faceting (or small multiples) not implemented.

Overcoming these limitations

Contributions (big or small) welcome!

Ways in which you can contribute

- Read and correct any inconsistencies in the documentation 🕮
- Raise issues about bugs/features 🐛
- Review code 🕵
- Add new functionality 🤖

Acknowledgments

Contributors to *ggstatsplot*

Other developers: Daniel Lüdecke, Dominique Makowski, Mattan S. Ben-Shachar, Patrick Mair

Advisors: Mina Cikara, Fiery Cushman, Iyad Rahwan

The CSS template comes from Garrick Aden-Buie.

Find me at...

`<i class="fab fa-twitter "></i>`{=html} @patilindrajeets

`<i class="fab fa-github "></i>`{=html} @IndrajeetPatil

`<i class="fas fa-link "></i>`{=html}
https://sites.google.com/site/indrajeetspatilmorality/

`<i class="fas fa-paper-plane "></i>`{=html}
patilindrajeet.science@gmail.com

For more information, see

https://indrajeetpatil.github.io/ggstatsplot/