Package ‘tadaatoolbox’

June 1, 2020

Type Package

Title Helpers for Data Analysis and Presentation Focused on Undergrad Psychology

Version 0.17.0

Description A teaching project for the display of statistical tests as well as some convenience functions for data cleanup. The primary components are the functions prefixed with 'tadaa_', which are built to work in an interactive environment, but also print markdown tables powered by 'pixiedust' for the creation of 'RMarkdown' reports. This package is no longer actively developed.

License MIT + file LICENSE

URL https://github.com/tadaadata/tadaatoolbox

BugReports https://github.com/tadaadata/tadaatoolbox/issues

Depends R (>= 3.2)

Imports broom, car, DescTools, ggplot2, magrittr, methods, pixiedust, stats, utils, viridis

Suggests covr, cowplot, knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.1.0

NeedsCompilation no

Author Lukas Burk [aut, cre] (<https://orcid.org/0000-0001-7528-3795>), Tobias Anton [aut], Daniel Lüdecke [ctb], Gesa Graf [ctb]

Maintainer Lukas Burk <lukas@tadaa-data.de>

Repository CRAN

Date/Publication 2020-06-01 17:50:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>confint_t</td>
<td>3</td>
</tr>
<tr>
<td>delete_na</td>
<td>3</td>
</tr>
<tr>
<td>effect_size_t</td>
<td>4</td>
</tr>
<tr>
<td>etasq</td>
<td>5</td>
</tr>
<tr>
<td>generate_recodes</td>
<td>6</td>
</tr>
<tr>
<td>interval_labels</td>
<td>6</td>
</tr>
<tr>
<td>inv</td>
<td>7</td>
</tr>
<tr>
<td>mean_ci_sem</td>
<td>8</td>
</tr>
<tr>
<td>mean_ci_t</td>
<td>8</td>
</tr>
<tr>
<td>modus</td>
<td>9</td>
</tr>
<tr>
<td>ngo</td>
<td>10</td>
</tr>
<tr>
<td>nom_c</td>
<td>10</td>
</tr>
<tr>
<td>nom_chisqu</td>
<td>11</td>
</tr>
<tr>
<td>nom_lambda</td>
<td>11</td>
</tr>
<tr>
<td>nom_phi</td>
<td>12</td>
</tr>
<tr>
<td>nom_v</td>
<td>12</td>
</tr>
<tr>
<td>ord_gamma</td>
<td>13</td>
</tr>
<tr>
<td>ord_pairs</td>
<td>14</td>
</tr>
<tr>
<td>ord_somers_d</td>
<td>14</td>
</tr>
<tr>
<td>ord_tau</td>
<td>15</td>
</tr>
<tr>
<td>pval_string</td>
<td>16</td>
</tr>
<tr>
<td>tadaa_aov</td>
<td>16</td>
</tr>
<tr>
<td>tadaa_balance</td>
<td>18</td>
</tr>
<tr>
<td>tadaa_chisq</td>
<td>18</td>
</tr>
<tr>
<td>tadaa_int</td>
<td>20</td>
</tr>
<tr>
<td>tadaa_kruskal</td>
<td>21</td>
</tr>
<tr>
<td>tadaa_levene</td>
<td>22</td>
</tr>
<tr>
<td>tadaa_mean_ci</td>
<td>23</td>
</tr>
<tr>
<td>tadaa_nom</td>
<td>23</td>
</tr>
<tr>
<td>tadaa_one_sample</td>
<td>24</td>
</tr>
<tr>
<td>tadaa_ord</td>
<td>25</td>
</tr>
<tr>
<td>tadaa_pairwise_t</td>
<td>26</td>
</tr>
<tr>
<td>tadaa_pairwise_tukey</td>
<td>28</td>
</tr>
<tr>
<td>tadaa_plot_tukey</td>
<td>29</td>
</tr>
<tr>
<td>tadaa_t.test</td>
<td>30</td>
</tr>
<tr>
<td>tadaa_wilcoxon</td>
<td>31</td>
</tr>
<tr>
<td>tadaa_z.test</td>
<td>32</td>
</tr>
<tr>
<td>theme_readthedown</td>
<td>33</td>
</tr>
<tr>
<td>z</td>
<td>34</td>
</tr>
<tr>
<td>z.test</td>
<td>35</td>
</tr>
</tbody>
</table>

**Index** 37
**Description**

Confidence Intervals

**Usage**

```r
confint_t(x, alpha = 0.05, na.rm = TRUE)
confint_norm(x, alpha = 0.05, na.rm = TRUE)
```

**Arguments**

- `x` A Numeric vector.
- `alpha` Alpha, default is 0.05.
- `na.rm` If TRUE (default), missing values are dropped.

**Value**

numeric of length one (size of CI in one direction).

**Examples**

```r
set.seed(42)
df <- data.frame(x = runif(100), y = sample(c("A", "B"), 100, TRUE))
confint_t(df$x)
set.seed(42)
df <- data.frame(x = runif(100), y = sample(c("A", "B"), 100, TRUE))
confint_norm(df$x)
```

---

**Description**

Delete cases with set amount of missing values

**Usage**

```r
delete_na(df, n = ncol(df) - 1)
```

**Arguments**

- `df` A data.frame.
- `n` Number of NAs allowed, defaults to `ncol(df) - 1`. 
Value

A filtered version of the input data.frame.

Note

Adapted from http://stackoverflow.com/a/30461945/409362.

Examples

```r
set.seed(1445)
dat <- data.frame(
  x = sample(c(1:15, NA, NA), 15),
  y = sample(c(1:15, NA, NA), 15),
  z = sample(c(1:15, NA, NA), 15)
)
dat
# No NAs per row allowed
delete_na(dat, 0)
# One NA per row allowed
delete_na(dat, 1)
```

---

**effect_size_t**

Simple Effect Size Calculation for t-Tests

Description

Calculates Cohen’s d for two sample comparisons.

Usage

```r
effect_size_t(
  data,  # A data.frame.
  response,  # The response variable (dependent).
  group,  # The group variable, usually a factor.
  absolute = FALSE,  # If set to TRUE, the absolute effect size is returned.
  paired = FALSE,  # Whether the effect should be calculated for a paired t-test, default is FALSE.
  na.rm = TRUE  # If TRUE (default), missing values are dropped.
)
```

Arguments

data
response
group
absolute
paired
na.rm
The effect size here is Cohen’s d as calculated by $d = \frac{m_{diff}}{s_p}$, where $m_{diff} = \bar{x}_1 - \bar{x}_2$ and $s_p = \sqrt{\frac{n_1 - 1}{n_1 + n_2 - 2} s_{x_1}^2 + \frac{n_2 - 1}{n_1 + n_2 - 2} s_{x_2}^2}$.

For paired = TRUE, $s_p$ is substituted by $S_D = S_{x_1-x_2}$ via $sd(x1 -x2)$.

Value

numeric of length 1.

Examples

```r
set.seed(42)
df <- data.frame(x = runif(100), group = sample(c("A", "B"), 100, TRUE))
effect_size_t(df, "x", "group")
```

Description

Thin wrapper for DescTools::EtaSq to retrieve $\eta^2$ for a simple use case without having to fit an aov model beforehand. Only use this for a simple one-way design, e.g. only one independent variable.

Usage

`etasq(formula, data)`

Arguments

- `formula` The model formula for `stats::aov`.
- `data` The `data.frame`.

Value

A single numeric value

Examples

```r
etasq(stunzahl ~ jahrgang, ngo)
```
generate_recodes  

Convenience functions for interval recodes

Description
Get recode assignments for even intervals of discrete numeric values compatible with car::recode.

Usage
generate_recodes(from, to, width = 5)

Arguments
from, to A numeric value for the beginning and the end of the interval.
width The width of the interval, e.g. 5 (default) for intervals 0-5.

Value
A character vector of recode assignments compatible with car::recode.

Examples
## Not run:
x <- round(runif(100, 0, 100), 0)
recodes <- generate_recodes(0, 100, 10)

library(car)
recode(x, recodes = recodes)
## End(Not run)

interval_labels  

Convenience functions for interval recodes

Description
Get interval labels for even intervals of discrete numeric values compatible with base::cut.

Usage
interval_labels(from, to, width = 5)

Arguments
from, to A numeric value for the beginning and the end of the interval.
width The width of the interval, e.g. 5 (default) for intervals 0-5.
Value

A character vector of interval labels compatible with `base::cut`.

Examples

```r
# Not run:
set.seed(1)
x <- round(runif(100, 0, 100), 0)
labels <- interval_labels(0, 100, 10)
cut(x, breaks = seq(0, 100, 10), labels = labels)

# End(Not run)
```

---

### Description

Inverting scales

#### Usage

```r
inv(x, min, max)
```

#### Arguments

- **x**: A vector of numeric data.
- **min**: The minimum value of the scale.
- **max**: The maximum value of the scale.

#### Examples

```r
# Assuming you have a Likert-scale from 1 to 9
x <- c(4, 5, 2, 3, 7, 8, 3)
inv(x, 1, 9)
```
mean_ci_sem

*Standard Error of the Mean with CI*

**Description**

Standard Error of the Mean with CI

**Usage**

```r
mean_ci_sem(x, conf.level = 0.95)
```

**Arguments**

- `x` a numeric vector or R object which is coercible to one
- `conf.level` the confidence level (alpha) of the Interval

**Value**

a `data.frame` with the mean, SEM and its Confidence Interval

**Examples**

```r
set.seed(42)
iq <- rnorm(100, 100, 15)
mean_ci_sem(iq)
```

-----------

mean_ci_t

*Get mean and CI for a numeric vector*

**Description**

Suitable for use within `ggplot2:: stat_summary`

**Usage**

```r
mean_ci_t(x, alpha = 0.05, na.rm = TRUE)
```

**Arguments**

- `x` A Numeric vector.
- `alpha` Alpha, default is 0.05.
- `na.rm` If TRUE (default), missing values are dropped.
modus

Value
A data frame with y (mean), ymin and ymax values.

Examples

```r
set.seed(42)
df <- data.frame(x = runif(100), y = sample(c("A", "B"), 100, TRUE))
mean_ci_t(df$x)
```

---

modus  Modus

Description
Calculate the mode of a numeric vector. German name kept to avoid confusion.

Usage

```
modus(x, as_character = TRUE, reduce = TRUE)
```

Arguments

- **x**: A vector with numeric data.
- **as_character**: Always return a character. TRUE by default.
- **reduce**: Since mode can be of length > 1, this option pastes the result into a single character value.

Value
A vector of length 1 of type numeric or character, depending on input.

Examples

```r
## Not run:
x <- c(1, 2, 6, 2, 1, 5, 8, 4, 3, 2, 2, 2)
modus(x)

# Or for nominal data
x <- structure(c(2L, 1L, 2L, 2L, 2L, 1L), .Label = c("Ja", "Nein"), class = "factor")
modus(x)
```

## End(Not run)
The ngo dataset

Description
Sample data for teaching purposes used by Kähler (2008)

Usage
ngo

Format
A data.frame containing numerical and factor data.

Note
ryoureadyd::d.ngo is the source of this data, but with some recoding done.

References

Contingency Coefficient C

Description
Very simple wrapper for DescTools::ContCoef.

Usage
nom_c(x, y = NULL)

Arguments
x Dependent variable. Alternatively a table.
y Independent variable

Value
numeric value

Examples
nom_c(ngo$abschalt, ngo$geschl)
**nom_chisqu**

---

**Simple Chi^2**

**Description**

This is a very simple wrapper for `stats::chisq.test`.

**Usage**

```
nom_chisqu(x, y = NULL, correct = FALSE)
```

**Arguments**

- `x`: Dependent variable. Alternatively a `table`.
- `y`: Independent variable
- `correct`: Apply correction, passed to `chisq.test`.

**Value**

A numeric value

**Note**

The warning message in case of low samples size and possibly incorrect approximation is suppressed silently.

**Examples**

```
nom_chisqu(ngo$abschalt, ngo$geschl)
```

---

**nom_lambda**

---

**Lambda**

**Description**

Very simple wrapper for `DescTools::Lambda`.

**Usage**

```
nom_lambda(x, y = NULL, symmetric = FALSE, reverse = FALSE)
```

**Arguments**

- `x`: Dependent variable. Alternatively a `table`.
- `y`: Independent variable
- `symmetric`: If `TRUE`, symmetric lambda is returned. Default is `FALSE`.
- `reverse`: If `TRUE`, row and column variable are switched.
**nom_v**

**Value**
numeric value

**Examples**
nom_lambda(ngo$abschalt, ngo$geschl)

---

**nom_phi**

*Phi coefficient*

**Description**
Very simple wrapper for DescTools::Phi.

**Usage**
nom_phi(x, y = NULL)

**Arguments**

- **x**
  Dependent variable. Alternatively a table.

- **y**
  Independent variable

**Value**
numeric value

**Examples**
nom_phi(ngo$abschalt, ngo$geschl)

---

**nom_v**

*Cramer’s V*

**Description**
Very simple wrapper for DescTools::CramerV.

**Usage**
nom_v(x, y = NULL)

**Arguments**

- **x**
  Dependent variable. Alternatively a table.

- **y**
  Independent variable
ord_gamma

Value

numeric value

Examples

nom_v(ngo$abschalt, ngo$geschl)

df <- data.frame(
  rating = round(runif(50, 1, 5)),
  group = sample(c("A", "B", "C"), 50, TRUE)
)
tbl <- table(df)
ord_gamma(tbl)

ord gamma | Gamma

Description

Simple wrapper for DescTools::GoodmanKruskalGamma.

Usage

ord.gamma(x, y = NULL)

Arguments

x | A table or dependent numeric variable.
y | Empty or independent grouping variable

Value

numeric of length 1.

Examples

df <- data.frame(
  rating = round(runif(50, 1, 5)),
  group = sample(c("A", "B", "C"), 50, TRUE)
)
tbl <- table(df)
ord.gamma(tbl)
ord_pairs

Retrieve all type of pairs for ordinal statistics

Description
Retrieve all type of pairs for ordinal statistics

Usage
ord_pairs(x, y = NULL)

Arguments
x Dependent variable. Alternatively a table.

y Independent variable

Value
A 1x5 data.frame with numeric values.

Source
Internals for this function are copied from this gist by GitHub user Marc Schwartz.

Examples
ord_pairs(ngo$leistung, ngo$begabung)

ord_somers_d

Somers’ D

Description
Very simple wrapper for DescTools::SomersDelta.

Usage
ord_somers_d(x, y = NULL, symmetric = FALSE, reverse = FALSE)

Arguments
x Dependent variable. Alternatively a table.

y Independent variable

symmetric If TRUE, symmetric D is returned. Default is FALSE.

reverse If TRUE, row and column variable are switched. Default is FALSE, meaning the row variable is considered dependant.
**ord_tau**

**Value**

numeric value

**Examples**

```
ord_somers_d(ngo$abschalt, ngo$geschl)
```

---

**Description**

A wrapper for the appropriate functions from *DescTools* to calculate Tau A, B and C.

**Usage**

```
ord_tau(x, y = NULL, tau = "b", reverse = FALSE)
```

**Arguments**

- **x**: Dependent variable. Alternatively a table.
- **y**: Independent variable
- **tau**: Which of the Taus to return. Default is “b”.
- **reverse**: If TRUE, row and column variable are switched.

**Value**

numeric value

**Examples**

```
ord_tau(ngo$urteil, ngo$leistung, tau = "a")
```
pval_string  

Easy p-value formatting

Description

Easy p-value formatting

Usage

pval_string(pv)

Arguments

pv  
A p-value in numeric form.

Value

A formatted character representation of the input value.

Note

Simplified version of pixiedust::pvalString which considers < 0.05.

Examples

```r
pv <- c(.9, .2, .049, .009, .000003)
names(pv) <- pval_string(pv)
print(pv)
```

tadaa_aov  

Tadaa, ANOVA!

Description

Performs one-, two-way or factorial ANOVA with adjustable sums of squares method and optionally displays effect sizes ((partial) \( \eta^2 \) and Cohen’s f).

Usage

```r
tadaa_aov(
  formula,
  data = NULL,
  show_effect_size = TRUE,
  factorize = TRUE,
  type = 3,
  check_contrasts = TRUE,
  print = c("df", "console", "html", "markdown")
)
```

Arguments

- **formula**: Formula for model, passed to `aov`.
- **data**: Data for model.
- **show_effect_size**: If TRUE (default), effect sizes partial eta^2 and Cohen's f are appended as columns.
- **factorize**: If TRUE (default), non-factor independent variables will automatically converted via `as.factor`, so beware of your inputs.
- **type**: Which type of SS to use. Default is 3, can also be 1 or 2.
- **check_contrasts**: Only applies to type = 3. If TRUE (default), the contrasts of each non-ordered factor are set to "contr.sum".
- **print**: Print method, default df: A regular data.frame. Otherwise passed to `pixiedust::sprinkle_print_method` for fancyness.

Details

- If a specified independent variable is not properly encoded as a factor, it is automatically converted if factorize = TRUE to ensure valid results.
- If type = 3 and check_contrasts = TRUE, the "contrasts" of each non-ordered factor will be checked and set to contr.sum to ensure the function yields usable results. It is highly recommended to only use check_contrasts = FALSE for debugging or educational purposes, or if you know what you're doing and using your own contrast matrix.

Value

A data.frame by default, otherwise dust object, depending on print.

See Also

Other Tadaa-functions: `tadaa_chisq()`, `tadaa_kruskal()`, `tadaa_levene()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_tukey()`, `tadaa_pairwise_t()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

Examples

```r
tadaa_aov(stunzahl ~ jahrgang, data = ngo)
tadaa_aov(stunzahl ~ jahrgang * geschl, data = ngo)

# Other types of sums and print options
## Not run:
tadaa_aov(stunzahl ~ jahrgang * geschl, data = ngo, type = 1, print = "console")
tadaa_aov(stunzahl ~ jahrgang * geschl, data = ngo, type = 3, print = "console")
tadaa_aov(stunzahl ~ jahrgang * geschl, data = ngo, type = 3, check_contrasts = FALSE, print = "console")
## End(Not run)
```
### tadaa_balance

**Grouping design balance**

**Description**

Easily generate heatmaps to show how well balanced groups are designed, e.g. for ANOVA.

**Usage**

`tadaa_balance(data, group1, group2, palette = "D", annotate = TRUE)`

**Arguments**

- `data` A `data.frame`
- `group1` The grouping variable on the x-axis
- `group2` The grouping variable on the y-axis
- `palette` The `viridis::viridis` color palette to use; c("A", "B", "C", "D"), defaults to "D"
- `annotate` Should the n of each group be displayed in each cell of the heatmap?

**Value**

A `ggplot2` object

**See Also**

Other Tadaa-plot functions: `tadaa_int()`, `tadaa_mean_ci()`, `tadaa_plot_tukey()`

**Examples**

`tadaa_balance(ngo, jahrgang, geschl)`

### tadaa_chisq

**Tadaa, Chi-Square Test!**

**Description**

A comfortable wrapper of `stats::chisq.test` with pretty output and effect sizes depending on the size of the contingency table: Phi coefficient and Odds Ratios in case of a 2x2 table, Cramer’s V otherwise. The result is either returned as a `broom::tidy` data.frame or prettified using various `pixiedust::sprinkle` shenanigans.
Usage

```r
tadaa_chisq(
  data,
  x,
  y,
  correct = TRUE,
  print = c("df", "console", "html", "markdown")
)
```

Arguments

- **data**: A `data.frame`.
- **x**: A vector of categorial data (factor or character).
- **y**: Another vector of categorial data (also factor or character).
- **correct**: Apply Yate’s continuity correction for 2x2 tables, passed to `stats::chisq.test`. Defaults to `TRUE`.
- **print**: Print method, default `df`: A regular `data.frame`. Otherwise passed to `pixiedust::sprinkle_print_method` for fancyness.

Value

A `data.frame` by default, otherwise dust object, depending on `print`.

Note

The warning message in case of low samples size and possibly incorrect approximation is suppressed silently.

See Also

Other Tadaa-functions: `tadaa_aov()`, `tadaa_kruskal()`, `tadaa_levene()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_tukey()`, `tadaa_pairwise_t()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

Examples

```r
tadaa_chisq(ngo, abschalt, geschl)
tadaa_chisq(ngo, abschalt, jahrgang)
```
Description

Easily generate interaction plots of two nominal grouping variables and a numeric response variable.

Usage

```r
tadaa_int(
  data,
  response,
  group1,
  group2,
  grid = FALSE,
  brewer_palette = "Set1",
  labels = c("A", "B"),
  show_n = FALSE,
  print = TRUE
)
```

Arguments

- `data`: A `data.frame`.
- `response`: Response variable.
- `group1`: First grouping variable.
- `group2`: Second grouping variable.
- `grid`: If `TRUE`, the resulting graphs will be arranged in a grid via `cowplot::plot_grid`.
- `brewer_palette`: The name of the `RColorBrewer` palette to use, defaults to `Set1`.
- `labels`: Labels used for the plots when printed in a grid (`grid = TRUE`), defaults to `c("A", "B")`.
- `show_n`: If `TRUE`, displays `N` in plot subtitle.
- `print`: Default is `TRUE`, set `FALSE` to suppress automatic printing. Useful if you intend to further modify the output plots.

Value

Invisible: A list with two `ggplot2` objects named `p1` and `p2`. If `print = TRUE`: Printed: The one or two `ggplot2` objects, depending on `grid`.

See Also

Other Tadaa-plot functions: `tadaa_balance()`, `tadaa_mean_ci()`, `tadaa_plot_tukey()`
**Examples**

```r
tadaa_int(ngo, stunzahl, jahrgang, geschl)
```

# As grid, if cowplot is installed
```r
tadaa_int(ngo, stunzahl, jahrgang, geschl, grid = TRUE)
```

**Description**

Tadaa, Kruskal-Wallis!

**Usage**

```r
tadaa_kruskal(
  formula,
  data = NULL,
  print = c("df", "console", "html", "markdown")
)
```

**Arguments**

- **formula**: Formula for model, passed to `kruskal.test`.
- **data**: Data for model.
- **print**: Print method, per default a regular `data.frame`. Otherwise passed to `pixiedust::sprinkle_print_method` for fancyness.

**Value**

A `data.frame` by default, otherwise dust object, depending on `print`.

**See Also**

Other Tadaa-functions: `tadaa_aov()`, `tadaa_chisq()`, `tadaa_levene()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_tukey()`, `tadaa_pairwise_t()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

**Examples**

```r
tadaa_kruskal(stunzahl ~ jahrgang, data = ngo)
```
Levene’s Test for Homoskedasticity

**Description**
A thin wrapper around `car::leveneTest` with some formatting done.

**Usage**
```r
tadaa_levene(
  data,  # Data for the test
  formula,  # Formula specifying groups, passed to `leveneTest`
  center = "median",  # Method to use, either median (default for robustness) or mean.
  print = c("df", "console", "html", "markdown")  # Print method, default df: A regular data.frame. Otherwise passed to `pixiedust::sprinkle_print_method` for fancyness.
)
```

**Arguments**
- `data` Data for the test
- `formula` Formula specifying groups, passed to `leveneTest`
- `center` Method to use, either median (default for robustness) or mean.
- `print` Print method, default df: A regular data.frame. Otherwise passed to `pixiedust::sprinkle_print_method` for fancyness.

**Value**
A data.frame by default, otherwise dust object, depending on `print`.

**Note**
The case of `center = "median"` is technically called Brown–Forsythe test, so if that’s selected the header for non-df returns will reflect that.

**See Also**
Other Tadaa-functions: `tadaa_aov()`, `tadaa_chisq()`, `tadaa_kruskal()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_tukey()`, `tadaa_pairwise_t()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

**Examples**
```r
tadaa_levene(ngo, deutsch ~ jahrgang, print = "console")
tadaa_levene(ngo, deutsch ~ jahrgang * geschl, print = "console")
```
### tadaa_mean_ci

**Plot Means with Errorbars**

**Description**

Plot Means with Errorbars

**Usage**

`tadaa_mean_ci(data, response, group, brewer_palette = "Set1")`

**Arguments**

- `data` A data.frame
- `response` Response variable, numeric.
- `group` Grouping variable, ideally a factor.
- `brewer_palette` Optional: The name of the RColorBrewer palette to use, defaults to Set1. Use `NULL` for no brewer palette.

**Value**

A ggplot2 object.

**See Also**

Other Tadaa-plot functions: `tadaa_balance()`, `tadaa_int()`, `tadaa_plot_tukey()`

**Examples**

`tadaa_mean_ci(ngo, deutsch, jahrgang, brewer_palette = "Set1")`

### tadaa_nom

**Get all the nominal stats**

**Description**

Get all the nominal stats

**Usage**

`tadaa_nom(x, y = NULL, round = 2, print = "console")`
tadaa_one_sample

Arguments

x  Dependent variable. Alternatively a table.
y  Independent variable
round  How many digits should be rounded. Default is 2.
print  Print method. Passed to `pixiedust::sprinkle_print_method` as of now.

Value

A dust object, depending on print.

See Also

Other Tadaa-functions: `tadaa_aov()`, `tadaa_chisq()`, `tadaa_kruskal()`, `tadaa_levene()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_tukey()`, `tadaa_pairwise_t()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

Examples

```r
tadaa_nom(ngo$abschalt, ngo$geschl)
```

---

tadaa_one_sample  Tadaa, one-sample tests!

Description

If `sigma` is omitted, the function will just perform a one-sample `stats::t.test`, but if `sigma` is provided, a z-test is performed. It basically works the same way, except that we pretend we know the population `sigma` and use the normal distribution for comparison.

Usage

```r
tadaa_one_sample(
  data = NULL,
  x,
  mu,
  sigma = NULL,
  direction = "two.sided",
  na.rm = FALSE,
  conf.level = 0.95,
  print = c("df", "console", "html", "markdown")
)
```
tadaa_ord

Arguments

data  A data.frame (optional).
x      A numeric vector or bare column name of data.
mu    The true mean (µ) to test for.
sigma Population sigma. If supplied, a z-test is performed, otherwise a one-sample
         stats::t.test is performed.
direction Test direction, like alternative in t.test.
na.rm  Whether to drop NA values. Default is FALSE.
conf.level Confidence level used for power and CI, default is 0.95.
print  Print method, default df: A regular data.frame. Otherwise passed to pix-
         iedust::sprinkle_print_method for fancyness.

Value

A data.frame by default, otherwise dust object, depending on print.

See Also

Other Tadaa-functions: tadaa_aov(), tadaa_chisq(), tadaa_kruskal(), tadaa_levene(), tadaa_nom(),
                      tadaa_ord(), tadaa_pairwise_tukey(), tadaa_pairwise_t(), tadaa_t.test(), tadaa_wilcoxon()

Examples

set.seed(42)
df <- data.frame(x = rnorm(n = 20, mean = 100, sd = 1))

tadaa_one_sample(df, x, mu = 101, sigma = 1)

# No data.frame, just a vector
tadaa_one_sample(x = rnorm(20), mu = 0)

---

tadaa_ord  Get all the ordinal stats

Description

Collects all ord_ statistics in neat output.

Usage

tadaa_ord(x, y = NULL, round = 2, print = "console")
Arguments

- `x`: Dependent variable. Alternatively a table.
- `y`: Independent variable
- `round`: Number of digits should be rounded. Default is 2.
- `print`: Print method. Passed to `pixiedust::sprinkle_print_method` as of now.

Value

A dust object, depending on `print`.

See Also

Other Tadaa-functions: `tadaa_aov()`, `tadaa_chisq()`, `tadaa_kruskal()`, `tadaa_levene()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_pairwise_tukey()`, `tadaa_pairwise_t()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

Examples

```r
  tadaa_ord(ngo$leistung, ngo$begabung)
```

---

### tadaa_pairwise_t

Extended Pairwise t-Tests

**Description**

This is an extension of `stats::pairwise.t.test` that's meant to deal with interactions out of the box, while also performing pairwise tests for the primary terms. The output of the function is modeled after `stats::TukeyHSD`, unfortunately without confidence intervals or test statistic though.

**Usage**

```r
  tadaa_pairwise_t(
    data, response, group1, group2 = NULL, p.adjust = "bonf", paired = FALSE, pool.sd = !paired, alternative = "two.sided", print = "df"
  )
```
Arguments

- **data**: A `data.frame` containing the variables.
- **response**: The response variable, i.e. the dependent numeric vector.
- **group1**: The grouping variables, typically a factor.
- **group2**: (Optional) second grouping variable.
- **p.adjust**: The p-adjustment method, see `stats::p.adjust.methods`, passed to `stats::pairwise.t.test`. Additionally, `sidak` is supported as a method, which is not the case with `stats::p.adjust`, as is `sidakSD` for the Sidak step-down procedure.
- **paired**: Defaults to `FALSE`, also passed to `stats::pairwise.t.test`.
- **pool.sd**: Defaults to the inverse of `paired`, passed to `stats::pairwise.t.test`.
- **alternative**: Defaults to `two.sided`, also passed to `stats::pairwise.t.test`.
- **print**: Print method, defaults to `df` for `data.frame` output, otherwise passed to `pixiedust::sprinkle_print_method`.

Value

A `data.frame` with columns `term`, `comparison` and `adj.p.value`.

Note

The adjustment method is applied within each term, meaning that the number of pairwise t-tests counted for the adjustment is only equal to the number of rows per term of the output. The additional Sidak adjustment method uses the following method: `p_adj <- 1 - pbinom(q = 0, size = length(p_values), prob = p_values)` And is sometimes preferred over Bonferroni. The Sidak-like (1987) step-down procedure (`sidakSD`) is an improvement over the Holm’s (1979) step-down procedure.

References

- [https://stats.stackexchange.com/questions/20825/sidak-or-bonferroni](https://stats.stackexchange.com/questions/20825/sidak-or-bonferroni)
- [https://rdrr.io/rforge/mutoss/man/SidakSD.html](https://rdrr.io/rforge/mutoss/man/SidakSD.html)

See Also

- `tadaa_pairwise_tukey()`

Other Tadaa-functions: `tadaa_aov()`, `tadaa_chisq()`, `tadaa_kruskal()`, `tadaa_levene()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_tukey()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

Examples

```r
tadaa_pairwise_t(ngo, deutsch, jahrgang, geschl, p.adjust = "none", print = "console")
tadaa_pairwise_t(ngo, deutsch, jahrgang, geschl, p.adjust = "bonf", print = "console")
tadaa_pairwise_t(ngo, deutsch, jahrgang, geschl, p.adjust = "sidak", print = "console")
```
**tadaa_pairwise_tukey**  
*Tukey HSD pairwise comparisons*

**Description**

This function is merely a thin wrapper around `stats::TukeyHSD` with tidying done by `broom::tidy` and optional formatting via `pixiedust::sprinkle`. Its input is not an `aov` model like in the original function, but instead the `aov` model is fit internally based on the arguments given. This is meant to enable a consistent usage between the `tadaa_pairwise`-functions.

**Usage**

```r
tadaa_pairwise_tukey(data, response, group1, group2 = NULL, print = "df", ...)
```

**Arguments**

- `data`  
  A `data.frame` containing the variables.
- `response`  
  The response variable, i.e. the dependent numeric vector.
- `group1`  
  The grouping variables, typically a factor.
- `group2`  
  (Optional) second grouping variable.
- `print`  
  Print method, defaults to `df` for `data.frame` output, otherwise passed to `pixiedust::sprinkle_print_method`.
- `...`  
  Further arguments passed to `stats::TukeyHSD`

**Value**

A `data.frame` or `pixiedust::dust` object depending on `print`.

**See Also**

- `tadaa_pairwise_t`

Other Tadaa-functions: `tadaa_aov()`, `tadaa_chisq()`, `tadaa_kruskal()`, `tadaa_levene()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_t()`, `tadaa_t.test()`, `tadaa_wilcoxon()`

**Examples**

```r
tadaa_pairwise_tukey(data = ngo, deutsch, jahrgang, geschl)
tadaa_pairwise_tukey(data = ngo, deutsch, jahrgang, print = "console")
```
**tadaa_plot_tukey**  
*Plot TukeyHSD Results as Errorbars*

**Description**

This is a simple plotting template that takes the broom::tidy’d output of stats::TukeyHSD or alternatively the print = "df" output of tadaa_pairwise_tukey and plots it nicely with error bars.

**Usage**

```r
tadaa_plot_tukey(data, brewer_palette = "Set1")
```

**Arguments**

- `data`: The broom::tidy’d output of stats::TukeyHSD.
- `brewer_palette`: Optional: The name of the RColorBrewer palette to use, defaults to Set1. Use NULL for no brewer palette.

**Value**

A ggplot2 object.

**Note**

The alpha of the error bars is set to 0.25 if the contrast is not significant, and 1 otherwise. That’s neat.

**See Also**

Other Tadaa-plot functions: tadaa_balance(), tadaa_int(), tadaa_mean_ci()

**Examples**

```r
tests <- tadaa_pairwise_tukey(data = ngo, deutsch, jahrgang, geschl, print = "df")
tadaa_plot_tukey(tests)
```
tadaa_t.test  

**Description**

An extension for `stats::t.test` with added boni and tidy and/or pretty output. The result is either returned as a `broom::tidy` data.frame or prettified using various `pixiedust::sprinkle` shenanigans.

**Usage**

```r
tadaa_t.test(
  data,  # A data.frame.
  response,  # The response variable (dependent).
  group,  # The group variable, usually a factor.
  direction = "two.sided",  # Test direction, like alternative in `t.test`.
  paired = FALSE,  # If TRUE, a paired test is performed, defaults to FALSE.
  var.equal = FALSE,  # var.equal
  conf.level = 0.95,  # Confidence level used for power and CI, default is 0.95.
  print = c("df", "console", "html", "markdown")
)
```

**Arguments**

- `data`: A data.frame.
- `response`: The response variable (dependent).
- `group`: The group variable, usually a factor.
- `direction`: Test direction, like alternative in `t.test`.
- `paired`: If TRUE, a paired test is performed, defaults to FALSE.
- `var.equal`: If set, passed to `stats::t.test` to decide whether to use a Welch-correction. Default is FALSE to automatically use a Welch-test, which is in general the safest option.
- `conf.level`: Confidence level used for power and CI, default is 0.95.
- `print`: Print method, default df: A regular data.frame. Otherwise passed to `pixiedust::sprinkle_print_method` for fancyness.

**Value**

A data.frame by default, otherwise dust object, depending on `print`.

**See Also**

- Other Tadaa-functions: `tadaa_aov()`, `tadaa_chisq()`, `tadaa_kruskal()`, `tadaa_levene()`, `tadaa_nom()`, `tadaa_one_sample()`, `tadaa_ord()`, `tadaa_pairwise_tukey()`, `tadaa_pairwise_t()`, `tadaa_wilcoxon()`
tadaa_wilcoxon

Examples

```r
set.seed(42)
df <- data.frame(x = runif(100), y = sample(c("A", "B"), 100, TRUE))
tadaa_t.test(df, x, y)

df <- data.frame(x = runif(100), y = c(rep("A", 50), rep("B", 50)))
tadaa_t.test(df, x, y, paired = TRUE)

tadaa_t.test(ngo, deutsch, geschl, print = "console")
```

description

Tadaa, Wilcoxon!

Usage

```r
tadaa_wilcoxon(
  data,
  response,
  group,
  direction = "two.sided",
  paired = FALSE,
  print = c("df", "console", "html", "markdown"),
  ...
)
```

Arguments

data A data.frame.
response The response variable (dependent).
group The group variable, usually a factor.
direction Test direction, like alternative in t.test.
paired If TRUE, a paired test is performed, defaults to FALSE.
print Print method, default df: A regular data.frame. Otherwise passed to pixiedust::sprinkle_print_method for fancyness.
... Further arguments passed to stats::wilcox.test, e.g. correct = FALSE.

Value

A data.frame by default, otherwise dust object, depending on print.
See Also

Other Tadaa-functions: tadaa_aov(), tadaa_chisq(), tadaa_kruskal(), tadaa_levene(), tadaa_nom(), tadaa_one_sample(), tadaa_ord(), tadaa_pairwise_tukey(), tadaa_pairwise_t(), tadaa_t.test()

Examples

set.seed(42)
df <- data.frame(x = runif(100), y = sample(c("A", "B"), 100, TRUE))
tadaa_wilcoxon(df, x, y)

df <- data.frame(x = runif(100), y = c(rep("A", 50), rep("B", 50)))
tadaa_wilcoxon(df, x, y, paired = TRUE)

tadaa_z.test

Tadaa, z-test! No seriously.

Description

This is a wrapper around z.test, which in itself is a weird thing to use, but why not.

Usage

tadaa_z.test(
  data,
  x,
  y,
  sigma_x,
  sigma_y,
  direction = "two.sided",
  paired = FALSE,
  conf.level = 0.95,
  print = c("df", "console", "html", "markdown")
)

Arguments

data A data.frame containing variables.
x, y A bare name of a numeric variable in data.
sigma_x, sigma_y Numeric. Known variances of x and y.
direction Test direction, like alternative in t.test.
paired If TRUE, a paired test is performed, defaults to FALSE.
conf.level Confidence level used for power and CI, default is 0.95.
print Print method, default df: A regular data.frame. Otherwise passed to pixiedust::sprinkle_print_method for fancyness.
theme_readthedown

Value

A pixiedust::dust object or data.frame.

Examples

```r
set.seed(192)
df <- data.frame(
  lefties = rnorm(10, mean = 5, sd = 2),
  righties = rnorm(10, mean = 5.5, sd = 2.5)
)
tadaa_z.test(data = df, x = lefties, y = righties, sigma_x = 2, sigma_y = 2.5, print = "console")
```

---

theme_readthedown

ggplot2 theme to fit the readthedown Rmd format

Description

A ggplot theme to fit rmdformats::readthedown in terms of background color and dark grid lines.

Usage

```r
theme_readthedown(
  base_size = 12,
  base_family = "",
  bg = "#fcfcfc",
  axis_emph = "xy",
  ...
)
```

```r
theme_tadaa(
  base_size = 12,
  base_family = "",
  bg = "#fcfcfc",
  axis_emph = "xy",
  ...
)
```

Arguments

- **base_size**: Base text size, defaults to 12.
- **base_family**: Base text family. Use "Roboto Slab" to match the readthedown headers, or "Lato" for the body style.
- **bg**: Background color, defaults to rmdformats::readthedown’s background, #fcfcfc
- **axis_emph**: Which axis to emphasize visually (black lines). One of "x", "y", "xy", NULL.
- **...**: Other arguments passed to ggplot2::theme()
A ggplot2 theme

Examples

```r
## Not run:
library(ggplot2)
p <- ggplot(ngo, aes(x = stunzahl)) +
   geom_bar()

p + theme_readthedown()
p + theme_readthedown(base_family = "Lato")
p + theme_readthedown(base_family = "Roboto Condensed", axis_emph = "x")

## End(Not run)
```

---

**z**  
Convert numeric vector to z-values

---

**Description**

A trivial scaling function. You might as well use `base::scale`, which allows arbitrary centers and scales, but returns a matrix by default.

**Usage**

```r
z(x)
```

**Arguments**

- `x` A numeric vector.

**Value**

A vector of z-values of the same length as `x`.

**Examples**

```r
x <- rnorm(500, mean = 10, sd = 5)
z_vals <- z(x)
round(c(mean = mean(z_vals), sd = sd(z_vals)), 2)
```
Description
Since the "standard" z-test is not available in R as in most real-world scenarios you're only ever going to use a t-test, this function fills that gap for teaching purposes. The function is basically a carbon-copy of stats::t.test, but with user-supplied variances for x and y and p-value and related calculations use a standard normal distribution.

Usage
z.test(
  x,
  y = NULL,
  alternative = c("two.sided", "less", "greater"),
  mu = 0,
  sigma_x,
  sigma_y = NULL,
  paired = FALSE,
  conf.level = 0.95
)

Arguments
  x  A (non-empty) numeric vector of data values
  y  An optional (non-empty) numeric vector of data values. If omitted, a one-sample test is conducted.
  alternative A character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
  mu  A number indicating the true value of the mean (or difference in means if you are performing a two sample test).
  sigma_x, sigma_y The assumed known variance of x and y. Must be numeric.
  paired  A logical indicating whether you want a paired t-test.
  conf.level  Confidence level of the interval.

Value
An object of class htest, see stats::t.test

Source
stats::t.test
Examples

```r
x <- rnorm(10, 5, 1)
y <- 1:10 + rnorm(10, 3, 1.5)

# Two sample
z.test(x, y, sigma_x = 1, sigma_y = 1.5)

# One sample
z.test(x, sigma_x = 1, mu = 5)
```
Index

*Topic dataset  
  ngo, 10

base::cut, 6, 7
base::scale, 34
broom::tidy, 18, 28–30

car::leveneTest, 22

car::recode, 6

confint_norm (confint_t), 3
confint_t, 3

cowplot::plot_grid, 20

delete_na, 3

DescTools, 15
DescTools::ContCoef, 10

DescTools::CramerV, 12

DescTools::EtaSq, 5

DescTools::GoodmanKruskalGamma, 13

DescTools::Lambda, 11

DescTools::Phi, 12

DescTools::SomersDelta, 14

effect_size_t, 4

etasq, 5

generate_recodes, 6

ggplot2, 29
ggplot2::stat_summary, 8
ggplot2::theme(), 33

interval_labels, 6

inv, 7

mean_ci_sem, 8

mean_ci_t, 8

modus, 9

ngo, 10

nom_c, 10

nom_chisqu, 11

nom_lambda, 11

nom_phi, 12

nom_v, 12

ord_gamma, 13

ord_pairs, 14

ord_somers_d, 14

ord_tau, 15

pixiedust::dust, 28, 33

pixiedust::pvalString, 16

pixiedust::sprinkle, 18, 28, 30

pixiedust::sprinkle_print_method, 17, 19, 21, 22, 24–28, 30–32

pval_string, 16

stats::aov, 5

stats::chisq.test, 11, 18, 19

stats::p.adjust, 27

stats::p.adjust.methods, 27

stats::pairwise.t.test, 26, 27

stats::t.test, 24, 25, 30, 35

stats::TukeyHSD, 26, 28, 29

stats::wilcox.test, 31

t.test, 25, 30–32

tadaa_aov, 16, 19, 21, 22, 24–28, 30, 32

tadaa_balance, 18, 20, 23, 29

tadaa_chisq, 17, 18, 21, 22, 24–28, 30, 32

tadaa_int, 18, 20, 23, 29

tadaa_kruskal, 17, 19, 21, 22, 24–28, 30, 32

tadaa_levene, 17, 19, 21, 22, 24–28, 30, 32

tadaa_mean_ci, 18, 20, 23, 29

tadaa_nom, 17, 19, 21, 22, 23, 25–28, 30, 32

tadaa_one_sample, 17, 19, 21, 22, 24, 26–28, 30, 32

tadaa_ord, 17, 19, 21, 22, 24, 25, 27, 28, 30, 32

tadaa_pairwise_t, 17, 19, 21, 22, 24–26, 26, 28, 30, 32
tadaa_pairwise_t(), 28

tadaa_pairwise_tukey, 17, 19, 21, 22, 24–27, 28, 29, 30, 32

tadaa_pairwise_tukey(), 27

tadaa_plot_tukey, 18, 20, 23, 29

tadaa_t.test, 17, 19, 21, 22, 24–28, 30, 32

tadaa_wilcoxon, 17, 19, 21, 22, 24–28, 30, 31

tadaa_z.test, 32

theme_readthedown, 33

theme_tadaa (theme_readthedown), 33

viridis::viridis, 18

z, 34

z.test, 32, 35