Package ‘rsprite2’

September 22, 2021

Title Identify Distributions that Match Reported Sample Parameters (SPRITE)

Version 0.1.0

Description The SPRITE algorithm creates possible distributions of discrete responses based on reported sample parameters, such as mean, standard deviation and range (Heathers et al., 2018, <doi:10.7287/peerj.preprints.26968v1>). This package implements it, drawing heavily on the code for Nick Brown's 'rSPRITE' Shiny app <http://shiny.ieis.tue.nl/sprite/>.
In addition, it supports the modeling of distributions based on multi-item (Likert-type) scales and the use of restrictions on the frequency of particular responses.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Suggests ggplot2, testthat (>= 3.0.0), tibble, tidyr, scales

Config/testthat/edition 3

URL https://lukaswallrich.github.io/rsprite2/

BugReports https://github.com/LukasWallrich/rsprite2/issues

Imports checkmate, Rdpack

RdMacros Rdpack

NeedsCompilation no

Author Lukas Wallrich [aut, cre] (<https://orcid.org/0000-0003-2121-5177>)

Maintainer Lukas Wallrich <lukas.wallrich@gmail.com>

Repository CRAN

Date/Publication 2021-09-22 15:00:02 UTC

R topics documented:

`find_possible_distribution` *Find a possible distribution.*

Description

This function aims to find a possible distribution that would give rise to the observed sample parameters. For that, you need to pass a list of parameters, best created with `set_parameters`

Usage

```
find_possible_distribution(parameters, seed = NULL, values_only = FALSE)
```

Arguments

`parameters`	List of parameters, see `set_parameters`
`seed`	An integer to use as the seed for random number generation. Set this in scripts to ensure reproducibility.
`values_only`	Should only values or a more informative list be returned. See details.

Value

Unless `values_only = TRUE`, a list with:

`outcome`	success or failure - character
`distribution`	The distribution that was found (if success) / that had the closest variance (if failure) - numeric
`mean`	The exact mean of the distribution - numeric
`sd`	The SD of the distribution that was found (success) / that came closest (failure) - numeric
`iterations`	The number of iterations required to achieve the specified SD - numeric

If `values_only = TRUE`, then the distribution is returned if one was found, and NULL if it failed.

Examples

```
sprite_parameters <- set_parameters(mean = 2.2, sd = 1.3, n_obs = 20,
                                     min_val = 1, max_val = 5)
find_possible_distribution(sprite_parameters)
```

`find_possible_distributions` *Find several possible distributions.*

Description

This function aims to find several possible distribution that would give rise to the observed sample parameters. For that, you need to pass a list of parameters, created with `set_parameters`

Usage

```
find_possible_distributions(
  parameters,
  n_distributions = 10,
  seed = NULL,
  return_tibble = TRUE,
  return_failures = FALSE
)
```

Arguments

`parameters`	List of parameters, see `set_parameters`
`n_distributions`	The target number of distributions to return.
`seed`	An integer to use as the seed for random number generation. Set this in scripts to ensure reproducibility.
`return_tibble`	Should a tibble, rather than a list, be returned? Requires the `tibble`-package, ignored if that package is not available.
`return_failures`	Should distributions that failed to produce the desired SD be returned? Defaults to false

Value

A tibble or list (depending on the `return_tibble` argument) with:

`outcome`	success or failure - character
`distribution`	The distribution that was found (if success) / that had the closest variance (if failure) - numeric
`mean`	The exact mean of the distribution - numeric
`sd`	The SD of the distribution that was found (success) / that came closest (failure) - numeric
`iterations`	The number of iterations required to achieve the specified SD - numeric - the first time this distribution was found

Examples

```
sprite_parameters <- set_parameters(mean = 2.2, sd = 1.3, n_obs = 20,
                                    min_val = 1, max_val = 5)

find_possible_distributions(sprite_parameters, 5, seed = 1234)
```

GRIM_test — *GRIM test for mean*

Description

This function tests whether a given mean (with a specific precision) can result from a sample of a given size based on integer responses to one or more items. The test is based on Brown & Heathers (2017). If `return_values = TRUE` and if there is more than one precise mean compatible with the given parameters, all possible means are returned. In that case, if the given mean is not consistent, the closest consistent mean is returned with a warning.

Usage

```
GRIM_test(mean, n_obs, m_prec = NULL, n_items = 1, return_values = FALSE)
```

Arguments

`mean`	The mean of the distribution
`n_obs`	The number of observations (sample size)
`m_prec`	The precision of the mean, as number of digits after the decimal point. If not provided, taken based on the significant digits of `mean` - so only needed if reported mean ends in 0
`n_items`	Number of items in scale, if distribution represents scale averages. Defaults to 1, which represents any single-item measure.
`return_values`	Should all means consistent with the given parameters be returned?

Value

Either TRUE/FALSE, or all possible means (if test passes)/closest consistent mean (if test fails)

References

Brown NJ, Heathers JA (2017). "The GRIM test: A simple technique detects numerous anomalies in the reporting of results in psychology." *Social Psychological and Personality Science*, **8**(4), 363–369.

Examples

```
# A sample of 28 integers cannot result in a mean of 5.19. This is shown by
GRIM_test(5.19, 28)

# To find the closest possible mean, set return_values to TRUE
GRIM_test(5.19, 28, return_values = TRUE)
```

`plot_distributions` *Plot distributions*

Description

This plots distributions identified by `find_possible_distributions` using ggplot2. They can be shown as histograms or as cumulative distributions (ECDF) plots. The latter give more information, yet not all audiences are familiar with them.

Usage

```
plot_distributions(
  distributions,
  plot_type = c("auto", "histogram", "ecdf", "density"),
  max_plots = 100,
  show_ids = FALSE,
  facets = NULL
)
```

Arguments

`distributions`	Tibble with a column `distribution` and an identifier (`id`), typically as returned from `find_possible_distributions`.
`plot_type`	Plot multiple histograms, or overlapping cumulative distribution plots, or density plots? "auto" is to plot histograms if up to 9 distributions are passed, or if there are fewer than 10 discrete values, and empirical cumulative distribution plots otherwise
`max_plots`	How many distributions should *at most* be plotted? If more are passed, this number is randomly selected.
`show_ids`	Should ids of the distributions be shown with ecdf and density charts? Defaults to no, since the default ids are not meaningful.
`facets`	Should distributions be shown in one chart or in multiple small charts? Only considered for ecdf and density charts, histograms are always shown in facets

Value

A ggplot2 object that can be styled with functions such as `labs` or `theme_linedraw`

Examples

```
sprite_parameters <- set_parameters(mean = 2.2, sd = 1.3, n_obs = 20,
                                    min_val = 1, max_val = 5)

poss <- find_possible_distributions(sprite_parameters, 5, seed = 1234)

# All distributions in same plot
plot_distributions(poss, plot_type = "ecdf")

# Separate plot for each distribution
plot_distributions(poss, plot_type = "ecdf", facets = TRUE)
```

`set_parameters` *Define parameters for SPRITE algorithm*

Description

The SPRITE algorithm aims to construct possible distributions that conform to observed/reported parameters. This function performs some checks and returns a list of these parameters that can then be passed to the functions that actually generate the distributions (e.g. `find_possible_distribution`)

Usage

```
set_parameters(
  mean,
  sd,
  n_obs,
  min_val,
  max_val,
  m_prec = NULL,
  sd_prec = NULL,
  n_items = 1,
  restrictions_exact = NULL,
  restrictions_minimum = NULL
)
```

Arguments

`mean`	The mean of the distribution
`sd`	The standard deviation of the distribution
`n_obs`	The number of observations (sample size)
`min_val`	The minimum value
`max_val`	The maximum value
`m_prec`	The precision of the mean, as number of digits after the decimal point. If not provided, taken based on the significant digits of `mean` - so only needed if reported mean ends in 0

`sd_prec`	The precision of the standard deviation, again only needed if reported standard deviation ends in 0.
`n_items`	Number of items in scale, if distribution represents scale averages. Defaults to 1, which represents any single-item measure.
`restrictions_exact`	Restrictions on the exact frequency of specific responses, see Details
`restrictions_minimum`	Restrictions on the minimum frequency of specific responses, see Details

Details

Restrictions can be used to define how often a specific value should appear in the sample. They need to be passed as a list in the form `value = frequency`. Thus, to specify that there should be no 3s and five 4s in the distribution, you would pass `restrictions_exact = list("3" = 0,"4" = 5)`. To specify that there should be at least one 1 and one 7, you would pass `restrictions_minimum = list("1" = 1,"7" = 1)`. If you just want to specify that the minimum and maximum values appear at least once (for instance when they are the reported rather than possible range), you can use the shortcut `restrictions_minimum = "range"`. Finally, if you work with multi-item scales that result in decimal responses, round those names to two decimal points, e.g., when `n_items = 3` you could specify `list("1.67" = 0)`.

Value

A named list of parameters, pre-processed for further rsprite2 functions.

Examples

```
set.seed(1234) #To get reproducible results

# Simple case
sprite_parameters <- set_parameters(mean = 2.2, sd = 1.3, n_obs = 20, min_val = 1, max_val = 5)
find_possible_distribution(sprite_parameters)

# With restrictions
sprite_parameters <- set_parameters(mean = 1.95, sd = 1.55, n_obs = 20,
                                    min_val = 1, max_val = 5, n_items = 3,
                                    restrictions_exact = list("3"=0, "3.67" = 2),
                                    restrictions_minimum = "range")
find_possible_distribution(sprite_parameters)
```

Index