Package ‘infer’

R functions (excerpt for Data Analysis):

calculate   Calculate summary statistics

Description

Calculate summary statistics

Usage

```
calculate(x, stat = c("mean", "median", "sum", "sd", "prop", "count", "diff in
   means", "diff in medians", "diff in props", "Chisq", "F", "slope", "correlation",
   "t", "z"), order = NULL, ...)
```

Arguments

x   The output from generate() for computation-based inference or the output from hypothesize() piped in to here for theory-based inference.

stat   A string giving the type of the statistic to calculate. Current options include "mean", "median", "sum", "sd", "prop", "count", "diff in means", "diff in medians", "diff in props", "Chisq", "F", "t", "z", "slope", and "correlation".

order   A string vector of specifying the order in which the levels of the explanatory variable should be ordered for subtraction, where order = c("first", "second") means ("first" - "second") Needed for inference on difference in means, medians, or proportions and t and z statistics.

...   To pass options like na.rm = TRUE into functions like mean(), sd(), etc.

Value

A tibble containing a stat column of calculated statistics.

Examples

```
# Permutation test for two binary variables mtcars %>%
  dplyr::mutate(am = factor(am), vs = factor(vs)) %>%
  specify(am ~ vs, success = "1") %>% hypothesize(null =
  "independence") %>% generate(reps = 100, type =
  "permute") %>%
  calculate(stat = "diff in props", order = c("1", "0"))
```

generate Generate resamples, permutations, or simulations

Description

Generation is done based on specify() and (if needed) hypothesize() inputs.

Usage

```
generate(x, reps = 1, type = NULL, ...)

GENERATION_TYPES
```

Arguments

x	A data frame that can be coerced into a tibble.
reps	The number of resamples to generate.
type	Currently either bootstrap, permute, or simulate.
...	Currently ignored.

Format

An object of class character of length 3.

Value

A tibble containing rep generated datasets, indicated by the replicate column.

Examples

```
# Permutation test for two binary variables mtcars
%>%
  dplyr::mutate(am = factor(am), vs = factor(vs)) %>%
  specify(am ~ vs, success = "1") %>%
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute"
```

get_confidence_interval Compute confidence interval

Description

Only simulation-based methods are (currently only) supported.

Usage

```
get_confidence_interval(x, level = 0.95, type = "percentile",
  point_estimate = NULL)

get_ci(x, level = 0.95, type = "percentile", point_estimate = NULL)
```

Arguments

x Data frame of calculated statistics or containing attributes of theoretical distribution values. Currently, dependent on statistics being stored in stat column as created in calculate() function.

level A numerical value between 0 and 1 giving the confidence level. Default value is 0.95.

type A string giving which method should be used for creating the confidence interval. The default is "percentile" with "se" corresponding to (multiplier * standard error) as the other option.

point_estimate A numeric value or a 1x1 data frame set to NULL by default. Needed to be provided if type = "se".

Value

A 1 x 2 tibble with values corresponding to lower and upper values in the confidence interval.

Aliases

get_ci() is an alias of get_confidence_interval(). conf_int() is a deprecated alias of get_confidence_interval().

Examples

```
# Prepare the dataset
mtcars_df <- mtcars %>%
  dplyr::mutate(am = factor(am))

# Calculate the difference in means in the dataset d_hat
<- mtcars_df %>%
  specify(mpg ~ am) %>%
  calculate(stat = "diff in means", order = c("1", "0"))
```

```r
# Same calculation on 100 bootstrap replicates
bootstrap_distn <- mtcars_df %>%
  specify(mpg ~ am) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "diff in means", order = c("1", "0"))

# Use level to set the confidence level
bootstrap_distn %>%
  get_confidence_interval(level = 0.9)

# To calculate std error, set the type and point estimate
bootstrap_distn %>%
  get_confidence_interval(type = "se", point_estimate = d_hat)
```

rep_sample_n Perform repeated sampling

Description

Perform repeated sampling of samples of size n. Useful for creating sampling distributions.

Usage

```
rep_sample_n(tbl, size, replace = FALSE, reps = 1, prob = NULL)
```

Arguments

tbl	Data frame of population from which to sample.
size	Sample size of each sample.
replace	Should sampling be with replacement?
reps	Number of samples of size n = size to take.
prob	A vector of probability weights for obtaining the elements of the vector being sampled.

Value

A tibble of size rep times size rows corresponding to rep samples of size n = size from tbl.

Examples

```
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(ggplot2))

# A virtual population of N = 10,010, of which 3091 are hurricanes
population <- dplyr::storms %>%
  select(status)

# Take samples of size n = 50 storms without replacement; do this 1000 times samples
<- population %>%
  rep_sample_n(size = 50, reps = 1000)
samples

# Compute p_hats for all 1000 samples = proportion hurricanes p_hats
<- samples %>%
  group_by(replicate) %>%
  summarize(prop_hurricane = mean(status == "hurricane"))
p_hats

# Plot sampling distribution ggplot(p_hats, aes(x
= prop_hurricane)) +
  geom_density() +
  labs(x = "p_hat", y = "Number of samples",
  title = "Sampling distribution of p_hat from 1000 samples of size 50")
```

specify — Specify the response and explanatory variables

Description

specify() also converts character variables chosen to be factors.

Usage

```
specify(x, formula, response = NULL, explanatory = NULL,
  success = NULL)
```

Arguments

x — A data frame that can be coerced into a tibble.

formula — A formula with the response variable on the left and the explanatory on the right.

response — The variable name in x that will serve as the response. This is alternative to using the formula argument.

explanatory — The variable name in x that will serve as the explanatory variable.

success — The level of response that will be considered a success, as a string. Needed for inference on one proportion, a difference in proportions, and corresponding z stats.

Value

A tibble containing the response (and explanatory, if specified) variable data.

Examples

```
# Permutation test similar to ANOVA
mtcars %>%
  dplyr::mutate(cyl = factor(cyl)) %>%
  specify(mpg ~ cyl) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "F")
```

visualize — Visualize statistical inference

Description

Visualize the distribution of the simulation-based inferential statistics or the theoretical distribution (or both!).

Usage

```
visualize(data, bins = 15, method = "simulation",
  dens_color = "black", obs_stat = NULL, obs_stat_color = "red2",
  pvalue_fill = "pink", direction = NULL, endpoints = NULL,
  endpoints_color = "mediumaquamarine", ci_fill = "turquoise", ...)

visualise(data, bins = 15, method = "simulation",
  dens_color = "black", obs_stat = NULL, obs_stat_color = "red2",
  pvalue_fill = "pink", direction = NULL, endpoints = NULL,
  endpoints_color = "mediumaquamarine", ci_fill = "turquoise", ...)
```

Arguments

data — The output from calculate().

bins — The number of bins in the histogram.

method — A string giving the method to display. Options are "simulation", "theoretical", or "both" with "both" corresponding to "simulation" and "theoretical".

dens_color — A character or hex string specifying the color of the theoretical density curve.

obs_stat — A numeric value or 1x1 data frame corresponding to what the observed statistic is. Deprecated (see Details).

obs_stat_color — A character or hex string specifying the color of the observed statistic as a vertical line on the plot. Deprecated (see Details).

pvalue_fill — A character or hex string specifying the color to shade the p-value. In previous versions of the package this was the shade_color argument. Deprecated (see Details).

direction — A string specifying in which direction the shading should occur. Options are "less", "greater", or "two_sided" for p-value. Can also give "left", "right", or "both" for p-value. For confidence intervals, use "between" and give the endpoint values in endpoints. Deprecated (see Details).

endpoints — A 2 element vector or a 1 x 2 data frame containing the lower and upper values to be plotted. Most useful for visualizing conference intervals. Deprecated (see Details).

endpoints_color — A character or hex string specifying the color of the observed statistic as a vertical line on the plot. Deprecated (see Details).

ci_fill A character or hex string specifying the color to shade the confidence interval. Deprecated (see Details).

... Other arguments passed along to {ggplot2} functions.

Details

In order to make visualization workflow more straightforward and explicit visualize() now only should be used to plot statistics directly. That is why arguments not related to this task are deprecated and will be removed in a future release of {infer}.

To add to plot information related to p-value use shade_p_value(). To add to plot information related to confidence interval use shade_confidence_interval().

Value

A ggplot object showing the simulation-based distribution as a histogram or bar graph. Also used to show the theoretical curves.

See Also

shade_p_value(), shade_confidence_interval().

Examples

```
# Permutations to create a simulation-based null distribution for
# one numerical response and one categorical predictor
# using t statistic
mtcars %>%
  dplyr::mutate(am = factor(am)) %>%
  specify(mpg ~ am) %>% # alt: response = mpg, explanatory = am
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "t", order = c("1", "0")) %>%
  visualize(method = "simulation") #default method

# Theoretical t distribution for
# one numerical response and one categorical predictor
# using t statistic
mtcars %>%
  dplyr::mutate(am = factor(am)) %>%
  specify(mpg ~ am) %>% # alt: response = mpg, explanatory = am
  hypothesize(null = "independence") %>%
  # generate() is not needed since we are not doing simulation
  calculate(stat = "t", order = c("1", "0")) %>% visualize(method =
  "theoretical")
# Overlay theoretical distribution on top of randomized t-statistics mtcars %>%
  dplyr::mutate(am = factor(am)) %>%
  specify(mpg ~ am) %>% # alt: response = mpg, explanatory = am
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "t", order = c("1", "0")) %>%
  visualize(method = "both")
```