Package 'infer'

January 13, 2021

Type Package

Title Tidy Statistical Inference

Version 0.5.4

Description The objective of this package is to perform inference using an expressive statistical grammar that coheres with the tidy design framework.

License CC0

URL https://github.com/tidymodels/infer, https://infer.tidymodels.org/

BugReports https://github.com/tidymodels/infer/issues

Depends R (>= 3.5.0)

Imports dplyr (>= 0.7.0), ggplot2, glue (>= 1.3.0), grDevices, magrittr, methods, purrr, rlang (>= 0.2.0), tibble

Suggests broom, covr, devtools (>= 1.12.0), fs, knitr, nycflights13, rmarkdown, stringr, testthat, tidyr, vdiffr

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Andrew Bray [aut, cre],
Chester Ismay [aut] (<https://orcid.org/0000-0003-2820-2547>),
Evgeni Chasnovski [aut] (<https://orcid.org/0000-0002-1617-4019>),
Ben Baumer [aut] (<https://orcid.org/0000-0002-3279-0516>),
Mine Cetinkaya-Rundel [aut] (<https://orcid.org/0000-0001-6452-2420>),
Simon Couch [ctb],
Ted Laderas [ctb] (<https://orcid.org/0000-0002-6207-7068>),
Nick Solomon [ctb],
Johanna Hardin [ctb],
Albert Y. Kim [ctb] (<https://orcid.org/0000-0001-7824-306X>),
Neal Fultz [ctb],
Doug Friedman [ctb],

Richie Cotton [ctb] (<https://orcid.org/0000-0003-2504-802X>),
Brian Fannin [ctb]

Maintainer Andrew Bray <abray@reed.edu>

Repository CRAN

Date/Publication 2021-01-13 16:50:12 UTC

R topics documented:

`calculate` *Calculate summary statistics*

Description

Calculates summary statistics from outputs of `generate()` or `hypothesize()`.

Learn more in `vignette("infer")`.

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",
    "ratio of props", "odds ratio"),
```

```
  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", "ratio of props", "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.

Missing levels in small samples

In some cases, when bootstrapping with small samples, some generated bootstrap samples will have only one level of the explanatory variable present. For some test statistics, the calculated statistic in these cases will be NaN. The package will omit non-finite values from visualizations (with a warning) and raise an error in p-value calculations.

Examples

```
# calculate a null distribution of hours worked per week under
# the null hypothesis that the mean is 40
gss %>%
  specify(response = hours) %>%
  hypothesize(null = "point", mu = 40) %>%
  generate(reps = 200, type = "bootstrap") %>%
  calculate(stat = "mean")

# calculate a null distribution assuming independence between age
# of respondent and whether they have a college degree
gss %>%
  specify(age ~ college) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 200, type = "permute") %>%
  calculate("diff in means", order = c("degree", "no degree"))

# More in-depth explanation of how to use the infer package
## Not run:
```

```
vignette("infer")

## End(Not run)
```

`chisq_stat` *Tidy chi-squared test statistic*

Description

@description

Usage

```
chisq_stat(x, formula, response = NULL, explanatory = 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.
`...`	Additional arguments for chisq.test().

Details

A shortcut wrapper function to get the observed test statistic for a chisq test. Uses chisq.test(), which applies a continuity correction.

Examples

```
# chi-squared test statistic for test of independence
# of college completion status depending and one's
# self-identified income class
chisq_stat(gss, college ~ finrela)

# chi-squared test statistic for a goodness of fit
# test on whether self-identified income class
# follows a uniform distribution
chisq_stat(gss,
           response = finrela,
           p = c("far below average" = 1/6,
                 "below average" = 1/6,
                 "average" = 1/6,
                 "above average" = 1/6,
                 "far above average" = 1/6,
                 "DK" = 1/6))
```

chisq_test *Tidy chi-squared test*

Description

A tidier version of chisq.test() for goodness of fit tests and tests of independence.

Usage

```
chisq_test(x, formula, response = NULL, explanatory = 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.
...	Additional arguments for chisq.test().

Examples

```
# chi-squared test of independence for college completion
# status depending on one's self-identified income class
chisq_test(gss, college ~ finrela)

# chi-squared goodness of fit test on whether self-identified
# income class follows a uniform distribution
chisq_test(gss,
           response = finrela,
           p = c("far below average" = 1/6,
                 "below average" = 1/6,
                 "average" = 1/6,
                 "above average" = 1/6,
                 "far above average" = 1/6,
                 "DK" = 1/6))
```

deprecated *Deprecated functions and objects*

Description

These functions and objects should no longer be used. They will be removed in a future release of infer.

Usage

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

p_value(x, obs_stat, direction)

GENERATION_TYPES
```

Arguments

x	See the non-deprecated function.
level	See the non-deprecated function.
type	See the non-deprecated function.
point_estimate	See the non-deprecated function.
obs_stat	See the non-deprecated function.
direction	See the non-deprecated function.

Format

An object of class character of length 3.

See Also

get_p_value(), get_confidence_interval(), generate()

---

generate *Generate resamples, permutations, or simulations*

Description

Generation creates a null distribution from specify() and (if needed) hypothesize() inputs.

Learn more in vignette("infer").

Usage

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

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 (see below).
...	Currently ignored.

Value

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

Generation Types

The `type` argument determines the method used to create the null distribution.

- `bootstrap`: A bootstrap sample will be drawn for each replicate, where a sample of size equal to the input sample size is drawn (with replacement) from the input sample data.
- `permute`: For each replicate, each input value will be randomly reassigned (without replacement) to a new output value in the sample.
- `simulate`: A value will be sampled from a theoretical distribution with parameters specified in `hypothesize()` for each replicate. (This option is currently only applicable for testing point estimates.)

Examples

```
# Generate a null distribution by taking 200 bootstrap samples
gss %>%
 specify(response = hours) %>%
 hypothesize(null = "point", mu = 40) %>%
 generate(reps = 200, type = "bootstrap")

# Generate a null distribution for the independence of
# two variables by permuting their values 1000 times
gss %>%
 specify(partyid ~ age) %>%
 hypothesize(null = "independence") %>%
 generate(reps = 200, type = "permute")

# More in-depth explanation of how to use the infer package
## Not run:
vignette("infer")

## End(Not run)
```

---

get_confidence_interval

Compute confidence interval

Description

Compute a confidence interval around a summary statistic. Currently, only simulation-based methods are supported.

Learn more in `vignette("infer")`.

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) and `"bias-corrected"` for bias-corrected interval as other options.
point_estimate	A numeric value or a 1x1 data frame set to `NULL` by default. Needed to be provided if `type` is `"se"` or `"bias-corrected"`.

Details

A null hypothesis is not required to compute a confidence interval, but including `hypothesize()` in a chain leading to `get_confidence_interval()` will not break anything. This can be useful when computing a confidence interval after previously computing a p-value.

Value

A 1 x 2 tibble with 'lower_ci' and 'upper_ci' columns. Values correspond to lower and upper bounds of the confidence interval.

Aliases

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

Examples

```
boot_distr <- gss %>%
  # We're interested in the number of hours worked per week
  specify(response = hours) %>%
  # Generate bootstrap samples
  generate(reps = 1000, type = "bootstrap") %>%
  # Calculate mean of each bootstrap sample
  calculate(stat = "mean")
```

```
boot_distr %>%
  # Calculate the confidence interval around the point estimate
  get_confidence_interval(
    # At the 95% confidence level; percentile method
    level = 0.95
  )

# For type = "se" or type = "bias-corrected" we need a point estimate
sample_mean <- gss %>%
  specify(response = hours) %>%
  calculate(stat = "mean") %>%
  dplyr::pull()

boot_distr %>%
  get_confidence_interval(
    point_estimate = sample_mean,
    # At the 95% confidence level
    level = 0.95,
    # Using the standard error method
    type = "se"
  )

# More in-depth explanation of how to use the infer package
## Not run:
vignette("infer")

## End(Not run)
```

`get_p_value` *Compute p-value*

Description

Compute a p-value from a null distribution and observed statistic. Simulation-based methods are (currently only) supported.

Learn more in `vignette("infer")`.

Usage

```
get_p_value(x, obs_stat, direction)

get_pvalue(x, obs_stat, direction)
```

Arguments

`x`	Data frame of calculated statistics as returned by `generate()`
`obs_stat`	A numeric value or a 1x1 data frame (as extreme or more extreme than this).

direction — A character string. Options are "less", "greater", or "two-sided". Can also use "left", "right", "both", "two_sided", or "two sided".

Value

A 1x1 tibble with value between 0 and 1.

Aliases

get_pvalue() is an alias of get_p_value(). p_value is a deprecated alias of get_p_value().

Zero p-value

Though a true p-value of 0 is impossible, get_p_value() may return 0 in some cases. This is due to the simulation-based nature of the {infer} package; the output of this function is an approximation based on the number of reps chosen in the generate() step. When the observed statistic is very unlikely given the null hypothesis, and only a small number of reps have been generated to form a null distribution, it is possible that the observed statistic will be more extreme than every test statistic generated to form the null distribution, resulting in an approximate p-value of 0. In this case, the true p-value is a small value likely less than 3/reps (based on a poisson approximation).

In the case that a p-value of zero is reported, a warning message will be raised to caution the user against reporting a p-value exactly equal to 0.

Examples

```
# find the point estimate---mean number of hours worked per week
point_estimate <- gss %>%
  specify(response = hours) %>%
  calculate(stat = "mean") %>%
  dplyr::pull()

# starting with the gss dataset
gss %>%
  # ...we're interested in the number of hours worked per week
  specify(response = hours) %>%
  # hypothesizing that the mean is 40
  hypothesize(null = "point", mu = 40) %>%
  # generating data points for a null distribution
  generate(reps = 1000, type = "bootstrap") %>%
  # finding the null distribution
  calculate(stat = "mean") %>%
  get_p_value(obs_stat = point_estimate, direction = "two-sided")

# More in-depth explanation of how to use the infer package
## Not run:
vignette("infer")

## End(Not run)
```

gss *Subset of data from the General Social Survey (GSS).*

Description

The General Social Survey is a high-quality survey which gathers data on American society and opinions, conducted since 1972. This data set is a sample of 500 entries from the GSS, spanning years 1973-2018, including demographic markers and some economic variables. Note that this data is included for demonstration only, and should not be assumed to provide accurate estimates relating to the GSS. However, due to the high quality of the GSS, the unweighted data will approximate the weighted data in some analyses.

Usage

```
gss
```

Format

A tibble with 500 rows and 11 variables:

year year respondent was surveyed

age age at time of survey, truncated at 89

sex respondent's sex (self-identified)

college whether on not respondent has a college degree, including junior/community college

partyid political party affiliation

hompop number of persons in household

hours number of hours worked in week before survey, truncated at 89

income total family income

class subjective socioeconomic class identification

finrela opinion of family income

weight survey weight

Source

https://gss.norc.org

`hypothesize` *Declare a null hypothesis*

Description

Declare a null hypothesis about variables selected in `specify()`.

Learn more in `vignette("infer")`.

Usage

```
hypothesize(x, null, p = NULL, mu = NULL, med = NULL, sigma = NULL)

hypothesise(x, null, p = NULL, mu = NULL, med = NULL, sigma = NULL)
```

Arguments

`x`	A data frame that can be coerced into a tibble.
`null`	The null hypothesis. Options include `"independence"` and `"point"`.
`p`	The true proportion of successes (a number between 0 and 1). To be used with point null hypotheses when the specified response variable is categorical.
`mu`	The true mean (any numerical value). To be used with point null hypotheses when the specified response variable is continuous.
`med`	The true median (any numerical value). To be used with point null hypotheses when the specified response variable is continuous.
`sigma`	The true standard deviation (any numerical value). To be used with point null hypotheses.

Value

A tibble containing the response (and explanatory, if specified) variable data with parameter information stored as well.

Examples

```
# hypothesize independence of two variables
gss %>%
 specify(college ~ partyid, success = "degree") %>%
 hypothesize(null = "independence")

# hypothesize a mean number of hours worked per week of 40
gss %>%
  specify(response = hours) %>%
  hypothesize(null = "point", mu = 40)

# More in-depth explanation of how to use the infer package
## Not run:
```

```
vignette("infer")

## End(Not run)
```

`infer` *infer: a grammar for statistical inference*

Description

Details

The objective of this package is to perform statistical inference using a grammar that illustrates the underlying concepts and a format that coheres with the tidyverse.

For an overview of how to use the core functionality, see `vignette("infer")`

Author(s)

Maintainer: Andrew Bray <abray@reed.edu>

Authors:

- Chester Ismay <chester.ismay@gmail.com> (ORCID)
- Evgeni Chasnovski <evgeni.chasnovski@gmail.com> (ORCID)
- Ben Baumer <ben.baumer@gmail.com> (ORCID)
- Mine Cetinkaya-Rundel <mine@stat.duke.edu> (ORCID)

Other contributors:

- Simon Couch <simonpatrickcouch@gmail.com> [contributor]
- Ted Laderas <tedladeras@gmail.com> (ORCID) [contributor]
- Nick Solomon <nick.solomon@datacamp.com> [contributor]
- Johanna Hardin <Jo.Hardin@pomona.edu> [contributor]
- Albert Y. Kim <albert.ys.kim@gmail.com> (ORCID) [contributor]
- Neal Fultz <nfultz@gmail.com> [contributor]
- Doug Friedman <doug.nhp@gmail.com> [contributor]
- Richie Cotton <richie@datacamp.com> (ORCID) [contributor]
- Brian Fannin <captain@pirategrunt.com> [contributor]

See Also

Useful links:

- https://github.com/tidymodels/infer
- https://infer.tidymodels.org/
- Report bugs at https://github.com/tidymodels/infer/issues

print.infer *Print methods*

Description

Print methods

Usage

```
## S3 method for class 'infer'
print(x, ...)
```

Arguments

x	An object of class `infer`, i.e. output from `specify()` or `hypothesize()`.
...	Arguments passed to methods.

prop_test *Tidy proportion test*

Description

A tidier version of prop.test() for equal or given proportions.

Usage

```
prop_test(
  x,
  formula,
  response = NULL,
  explanatory = NULL,
  p = NULL,
  order = NULL,
  alternative = "two-sided",
  conf_int = TRUE,
  conf_level = 0.95,
  success = NULL,
  correct = NULL,
  z = FALSE,
  ...
)
```

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, where an explanatory variable NULL indicates a test of a single proportion.
response	The variable name in x that will serve as the response. This is alternative to using the formula argument. This is an alternative to the formula interface.
explanatory	The variable name in x that will serve as the explanatory variable. Optional. This is an alternative to the formula interface.
p	A numeric vector giving the hypothesized null proportion of success for each group.
order	A string vector specifying the order in which the proportions should be subtracted, where order = c("first","second") means "first" -"second". Ignored for one-sample tests, and optional for two sample tests.
alternative	Character string giving the direction of the alternative hypothesis. Options are "two-sided" (default), "greater", or "less". Only used when testing the null that a single proportion equals a given value, or that two proportions are equal; ignored otherwise.
conf_int	A logical value for whether to report the confidence interval or not. TRUE by default, ignored if p is specified for a two-sample test. Only used when testing the null that a single proportion equals a given value, or that two proportions are equal; ignored otherwise.
conf_level	A numeric value between 0 and 1. Default value is 0.95. Only used when testing the null that a single proportion equals a given value, or that two proportions are equal; ignored otherwise.
success	The level of response that will be considered a success, as a string. Only used when testing the null that a single proportion equals a given value, or that two proportions are equal; ignored otherwise.
correct	A logical indicating whether Yates' continuity correction should be applied where possible. If z = TRUE, the correct argument will be overwritten as FALSE. Otherwise defaults to correct = TRUE.
z	A logical value for whether to report the statistic as a standard normal deviate or a Pearson's chi-square statistic. z^2 is distributed chi-square with 1 degree of freedom, though note that the user will likely need to turn off Yates' continuity correction by setting correct = FALSE to see this connection.
...	Additional arguments for prop.test().

Examples

```
# two-sample proportion test for difference in proportions of
# college completion by respondent sex
prop_test(gss,
          college ~ sex,
          order = c("female", "male"))

# one-sample proportion test for hypothesized null
```

```
# proportion of college completion of .2
prop_test(gss,
          college ~ NULL,
          p = .2)

# report as a z-statistic rather than chi-square
# and specify the success level of the response
prop_test(gss,
          college ~ NULL,
          success = "degree",
          p = .2,
          z = TRUE)
```

`rep_sample_n` *Perform repeated sampling*

Description

These functions extend the functionality of `dplyr::sample_n()` and `dplyr::slice_sample()` by allowing for repeated sampling of data. This operation is especially helpful while creating sampling distributions—see the examples below!

Usage

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

rep_slice_sample(.data, n = 1, replace = FALSE, weight_by = NULL, reps = 1)
```

Arguments

`tbl, .data`	Data frame of population from which to sample.
`size, n`	Sample size of each sample.
`replace`	Should sampling be with replacement?
`reps`	Number of samples of size n = `size` to take.
`prob, weight_by`	A vector of sampling weights for each of the rows in `tbl`—must have length equal to `nrow(tbl)`.

Details

The `dplyr::sample_n()` function (to which `rep_sample_n()` was originally a supplement) has been superseded by `dplyr::slice_sample()`. `rep_slice_sample()` provides a light wrapper around `rep_sample_n()` that has a more similar interface to `slice_sample()`.

Value

A tibble of size `rep * size` rows corresponding to `reps` samples of size `size` from `tbl`, grouped by `replicate`.

Examples

```
library(dplyr)
library(ggplot2)

# take 1000 samples of size n = 50, without replacement
slices <- gss %>%
  rep_sample_n(size = 50, reps = 1000)

slices

# compute the proportion of respondents with a college
# degree in each replicate
p_hats <- slices %>%
  group_by(replicate) %>%
  summarize(prop_college = mean(college == "degree"))

# plot sampling distribution
ggplot(p_hats, aes(x = prop_college)) +
  geom_density() +
  labs(
    x = "p_hat", y = "Number of samples",
    title = "Sampling distribution of p_hat"
  )

# sampling with probability weights. Note probabilities are automatically
# renormalized to sum to 1
library(tibble)
df <- tibble(
  id = 1:5,
  letter = factor(c("a", "b", "c", "d", "e"))
)
rep_sample_n(df, size = 2, reps = 5, prob = c(.5, .4, .3, .2, .1))
```

---

shade_confidence_interval

Add information about confidence interval

Description

`shade_confidence_interval()` plots confidence interval region on top of the `visualize()` output. It should be used as \ggplot2\ layer function (see examples). `shade_ci()` is its alias.

Learn more in `vignette("infer")`.

Usage

```
shade_confidence_interval(
  endpoints,
  color = "mediumaquamarine",
  fill = "turquoise",
```

```
  ...
)

shade_ci(endpoints, color = "mediumaquamarine", fill = "turquoise", ...)
```

Arguments

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.
color	A character or hex string specifying the color of the end points as a vertical lines on the plot.
fill	A character or hex string specifying the color to shade the confidence interval. If NULL then no shading is actually done.
...	Other arguments passed along to \ggplot2\ functions.

Value

A list of \ggplot2\ objects to be added to the `visualize()` output.

See Also

`shade_p_value()` to add information about p-value region.

Examples

```
# find the point estimate---mean number of hours worked per week
point_estimate <- gss %>%
  specify(response = hours) %>%
  calculate(stat = "mean") %>%
  dplyr::pull()

# ...and a null distribution
null_dist <- gss %>%
  # ...we're interested in the number of hours worked per week
  specify(response = hours) %>%
  # hypothesizing that the mean is 40
  hypothesize(null = "point", mu = 40) %>%
  # generating data points for a null distribution
  generate(reps = 1000, type = "bootstrap") %>%
  # finding the null distribution
  calculate(stat = "mean")

# find a confidence interval around the point estimate
ci <- null_dist %>%
  get_confidence_interval(point_estimate = point_estimate,
                          # at the 95% confidence level
                          level = .95,
                          # using the standard error method
                          type = "se")
```

```
# and plot it!
null_dist %>%
  visualize() +
  shade_confidence_interval(ci)

# or just plot the bounds
null_dist %>%
  visualize() +
  shade_confidence_interval(ci, fill = NULL)

# More in-depth explanation of how to use the infer package
## Not run:
vignette("infer")

## End(Not run)
```

---

shade_p_value *Add information about p-value region(s)*

Description

shade_p_value() plots p-value region(s) (using "area under the curve" approach) on top of the visualize() output. It should be used as \ggplot2\ layer function (see examples). shade_pvalue() is its alias.

Learn more in vignette("infer").

Usage

```
shade_p_value(obs_stat, direction, color = "red2", fill = "pink", ...)

shade_pvalue(obs_stat, direction, color = "red2", fill = "pink", ...)
```

Arguments

obs_stat	A numeric value or 1x1 data frame corresponding to what the observed statistic is.
direction	A string specifying in which direction the shading should occur. Options are "less", "greater", or "two-sided". Can also give "left", "right", "both", "two_sided", or "two sided". If NULL then no shading is actually done.
color	A character or hex string specifying the color of the observed statistic as a vertical line on the plot.
fill	A character or hex string specifying the color to shade the p-value region. If NULL then no shading is actually done.
...	Other arguments passed along to \ggplot2\ functions.

Value

A list of \ggplot2\ objects to be added to the `visualize()` output.

See Also

`shade_confidence_interval()` to add information about confidence interval.

Examples

```
# find the point estimate---mean number of hours worked per week
point_estimate <- gss %>%
  specify(response = hours) %>%
  calculate(stat = "mean") %>%
  dplyr::pull()

# ...and a null distribution
null_dist <- gss %>%
  # ...we're interested in the number of hours worked per week
  specify(response = hours) %>%
  # hypothesizing that the mean is 40
  hypothesize(null = "point", mu = 40) %>%
  # generating data points for a null distribution
  generate(reps = 1000, type = "bootstrap") %>%
  # finding the null distribution
  calculate(stat = "mean")

# shade the p-value of the point estimate
null_dist %>%
  visualize() +
  shade_p_value(obs_stat = point_estimate, direction = "two-sided")

# More in-depth explanation of how to use the infer package
## Not run:
vignette("infer")

## End(Not run)
```

`specify` *Specify response and explanatory variables*

Description

`specify()` is used to specify which columns in the supplied data frame are the relevant response (and, if applicable, explanatory) variables. Note that character variables are converted to `factors`.

Learn more in `vignette("infer")`.

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. Alternatively, a response and explanatory argument can be supplied.
response	The variable name in x that will serve as the response. This is an alternative to using the formula argument.
explanatory	The variable name in x that will serve as the explanatory variable. This is an alternative to using the formula argument.
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

```
# specifying for a point estimate on one variable
gss %>%
   specify(response = age)

# specify a relationship between variables as a formula...
gss %>%
  specify(age ~ partyid)

# ...or with named arguments!
gss %>%
  specify(response = age, explanatory = partyid)

# More in-depth explanation of how to use the infer package
## Not run:
vignette("infer")

## End(Not run)
```

---

t_stat *Tidy t-test statistic*

Description

A shortcut wrapper function to get the observed test statistic for a t test.

Usage

```
t_stat(
  x,
  formula,
  response = NULL,
  explanatory = NULL,
  order = NULL,
  alternative = "two-sided",
  mu = 0,
  conf_int = FALSE,
  conf_level = 0.95,
  ...
)
```

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.
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").
alternative	Character string giving the direction of the alternative hypothesis. Options are "two-sided" (default), "greater", or "less".
mu	A numeric value giving the hypothesized null mean value for a one sample test and the hypothesized difference for a two sample test.
conf_int	A logical value for whether to include the confidence interval or not. TRUE by default.
conf_level	A numeric value between 0 and 1. Default value is 0.95.
...	Pass in arguments to \infer\ functions.

Examples

```
library(tidyr)

# t test statistic for true mean number of hours worked
# per week of 40
gss %>%
   t_stat(response = hours, mu = 40)

# t test statistic for number of hours worked per week
# by college degree status
gss %>%
   tidyr::drop_na(college) %>%
   t_stat(formula = hours ~ college,
```

```
  order = c("degree", "no degree"),
  alternative = "two-sided")
```

`t_test` *Tidy t-test*

Description

A tidier version of t.test() for two sample tests.

Usage

```
t_test(
  x,
  formula,
  response = NULL,
  explanatory = NULL,
  order = NULL,
  alternative = "two-sided",
  mu = 0,
  conf_int = TRUE,
  conf_level = 0.95,
  ...
)
```

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.
`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")`.
`alternative`	Character string giving the direction of the alternative hypothesis. Options are `"two-sided"` (default), `"greater"`, or `"less"`.
`mu`	A numeric value giving the hypothesized null mean value for a one sample test and the hypothesized difference for a two sample test.
`conf_int`	A logical value for whether to include the confidence interval or not. `TRUE` by default.
`conf_level`	A numeric value between 0 and 1. Default value is 0.95.
`...`	For passing in other arguments to t.test().

Examples

```
library(tidyr)

# t test for number of hours worked per week
# by college degree status
gss %>%
   tidyr::drop_na(college) %>%
   t_test(formula = hours ~ college,
      order = c("degree", "no degree"),
      alternative = "two-sided")

# see vignette("infer") for more explanation of the
# intuition behind the infer package, and vignette("t_test")
# for more examples of t-tests using infer
```

`visualize` *Visualize statistical inference*

Description

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

Learn more in `vignette("infer")`.

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

```
# find a null distribution
null_dist <- gss %>%
  # we're interested in the number of hours worked per week
  specify(response = hours) %>%
  # hypothesizing that the mean is 40
  hypothesize(null = "point", mu = 40) %>%
  # generating data points for a null distribution
  generate(reps = 1000, type = "bootstrap") %>%
  # calculating a distribution of t test statistics
  calculate(stat = "t")

# we can easily plot the null distribution by piping into visualize
null_dist %>%
  visualize()

# we can add layers to the plot as in ggplot, as well...
# find the point estimate---mean number of hours worked per week
point_estimate <- gss %>%
  specify(response = hours) %>%
  hypothesize(null = "point", mu = 40) %>%
  calculate(stat = "t")

# find a confidence interval around the point estimate
ci <- null_dist %>%
  get_confidence_interval(point_estimate = point_estimate,
                          # at the 95% confidence level
                          level = .95,
                          # using the standard error method
                          type = "se")

# display a shading of the area beyond the p-value on the plot
null_dist %>%
  visualize() +
  shade_p_value(obs_stat = point_estimate, direction = "two-sided")

null_dist %>%
  visualize() +
  shade_confidence_interval(ci)

# to plot a theoretical null distribution, skip the generate()
# step and supply `method = "theoretical"` to `visualize()`
```

```
null_dist_theoretical <- gss %>%
  specify(response = hours) %>%
  hypothesize(null = "point", mu = 40) %>%
  calculate(stat = "t")

visualize(null_dist_theoretical, method = "theoretical")

# to plot both a theory-based and simulation-based null distribution,
# use the simulation-based null distribution and supply
# `method = "both"` to `visualize()`
visualize(null_dist, method = "both")

# More in-depth explanation of how to use the infer package
## Not run:
vignette("infer")

## End(Not run)
```

`%>%` *Pipe*

Description

Like {dplyr}, {infer} also uses the pipe (`%>%`) function from `magrittr` to turn function composition into a series of iterative statements.

Arguments

`lhs, rhs`	Inference functions and the initial data frame.

Index