Package 'infer'

November 15, 2018

Type Package

Title Tidy Statistical Inference

Version 0.4.0

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

License CC0

Encoding UTF-8

LazyData true

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

Depends R (>= 3.1.2)

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

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

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

RoxygenNote 6.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Andrew Bray [aut, cre],
Chester Ismay [aut],
Ben Baumer [aut],
Mine Cetinkaya-Rundel [aut],
Evgeni Chasnovski [ctb],
Ted Laderas [ctb],
Nick Solomon [ctb],
Johanna Hardin [ctb],
Albert Y. Kim [ctb],
Neal Fultz [ctb],
Doug Friedman [ctb],
Richie Cotton [ctb],
Brian Fannin [ctb]

Maintainer Andrew Bray <abray@reed.edu>

Repository CRAN

Date/Publication 2018-11-15 22:10:03 UTC

R topics documented:

`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"))
```

---

chisq_stat *Tidy chi-squared test statistic*

Description

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

Usage

```
chisq_stat(data, formula, ...)
```

Arguments

data	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.
...	Additional arguments for chisq.test().

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(data, formula, ...)
```

Arguments

data	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.
...	Additional arguments for chisq.test().

Examples

```
# chisq test for comparing number of cylinders against automatic/manual
mtcars %>%
  dplyr::mutate(cyl = factor(cyl), am = factor(am)) %>%
  chisq_test(cyl ~ am)
```

deprecated *Deprecated functions*

Description

These functions 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)
```

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.

See Also

get_p_value, get_confidence_interval

`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"))
```

```
# 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)
```

`get_p_value` *Compute p-value*

Description

Simulation-based methods are (currently only) supported.

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"`, or `"both"`.

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()`.

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"))

# Same calculation on 100 permutation replicates
null_distn <- mtcars_df %>%
  specify(mpg ~ am) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 100) %>%
  calculate(stat = "diff in means", order = c("1", "0"))

# What proportion of replicates had a difference
# in means more extreme than in the dataset?
null_distn %>%
  get_p_value(obs_stat = d_hat, direction = "right")
```

`hypothesize` *Declare a null hypothesis*

Description

Declare a null hypothesis

Usage

```
hypothesize(x, null, ...)
```

Arguments

`x`	A data frame that can be coerced into a tibble.
`null`	The null hypothesis. Options include `"independence"` and `"point"`.
`...`	Arguments passed to downstream functions.

Value

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

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")
```

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

Description

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.

Examples

```
# Example usage:
library(infer)
```

`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.

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")
```

---

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.

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

```
viz_plot <- 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")
```

```
viz_plot + shade_confidence_interval(c(-1.5, 1.5))
viz_plot + shade_confidence_interval(c(-1.5, 1.5), fill = NULL)
```

---

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

Description

shade_p_value() plots p-value region(s) on top of the visualize() output. It should be used as {ggplot2} layer function (see examples). shade_pvalue() is its alias.

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", or "both". 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

```
viz_plot <- 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")

viz_plot + shade_p_value(1.5, direction = "right")
viz_plot + shade_p_value(1.5, direction = "both")
viz_plot + shade_p_value(1.5, direction = NULL)
```

`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")
```

t_stat *Tidy t-test statistic*

Description

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

Usage

```
t_stat(data, formula, ...)
```

Arguments

data	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.
...	Pass in arguments to {infer} functions.

t_test *Tidy t-test*

Description

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

Usage

```
t_test(data, formula, order = NULL, alternative = "two_sided",
  mu = 0, conf_int = TRUE, conf_level = 0.95, ...)
```

Arguments

data	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.
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

```
# t test for comparing mpg against automatic/manual
mtcars %>%
  dplyr::mutate(am = factor(am)) %>%
  t_test(mpg ~ am, order = c("1", "0"), alternative = "less")
```

`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")
```

`%>%` *Pipe*

Description

Like {dplyr}, {infer} also uses the pipe function, %>% to turn function composition into a series of imperative statements.

Arguments

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

Index