Package 'infer'

August 7, 2018

Type Package

Title Tidy Statistical Inference

Version 0.3.1

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

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 6.0.1.9000

VignetteBuilder knitr

NeedsCompilation no

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

Maintainer Andrew Bray <abray@reed.edu>

Repository CRAN

Date/Publication 2018-08-06 22:20:08 UTC

R topics documented:

`calculate` *Calculate summary statistics*

Description

Calculate summary statistics

Usage

```
calculate(x, stat = c("mean", "median", "sd", "prop", "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", "sd", "prop", "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

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

Description

A shortcut wrapper function to get the observed test statistic for a chisq test. Uses stats::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 stats::chisq.test

chisq_test — *A tidier version of chisq.test for goodness of fit tests and tests of independence.*

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)
```

conf_int — *Compute the confidence interval for (currently only) simulation-based methods*

Description

get_confidence_interval() and get_ci() are both aliases of conf_int()

Usage

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

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

get_confidence_interval(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 2 x 1 tibble with values corresponding to lower and upper values in the confidence interval

Examples

```
mtcars_df <- mtcars %>%
    dplyr::mutate(am = factor(am))
d_hat <- mtcars_df %>%
    specify(mpg ~ am) %>%
    calculate(stat = "diff in means", order = c("1", "0"))
bootstrap_distn <- mtcars_df %>%
    specify(mpg ~ am) %>%
    generate(reps = 100) %>%
   calculate(stat = "diff in means", order = c("1", "0"))
bootstrap_distn %>% conf_int(level = 0.9)
bootstrap_distn %>% conf_int(type = "se", point_estimate = d_hat)
```

---

generate — *Generate resamples, permutations, or simulations based on 'specify' and (if needed) 'hypothesize' inputs*

Description

Generate resamples, permutations, or simulations based on 'specify' and (if needed) 'hypothesize' inputs

Usage

```
generate(x, reps = 1, type = attr(x, "type"), ...)
```

Arguments

x	a data frame that can be coerced into a tbl_df
reps	the number of resamples to generate
type	currently either bootstrap, permute, or simulate
...	currently ignored

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")
```

---

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 tbl_df
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

a data frame with attributes set

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

p_value *Compute the p-value for (currently only) simulation-based methods* get_pvalue() *is an alias of* p_value

Description

Compute the p-value for (currently only) simulation-based methods get_pvalue() is an alias of p_value

Usage

```
p_value(x, obs_stat, direction)

get_pvalue(x, obs_stat, direction)
```

Arguments

x	data frame of calculated statistics or containing attributes of theoretical distribution values
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 specify "left", "right", or "both".

Value

a 1x1 data frame with value between 0 and 1

Examples

```
mtcars_df <- mtcars %>%
    dplyr::mutate(am = factor(am))
d_hat <- mtcars_df %>%
    specify(mpg ~ am) %>%
    calculate(stat = "diff in means", order = c("1", "0"))
null_distn <- mtcars_df %>%
    specify(mpg ~ am) %>%
    hypothesize(null = "independence") %>%
    generate(reps = 100) %>%
    calculate(stat = "diff in means", order = c("1", "0"))
null_distn %>%
    p_value(obs_stat = d_hat, direction = "right")
```

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")
```

set_params — *To determine which theoretical distribution to fit (if any)*

Description

To determine which theoretical distribution to fit (if any)

Usage

```
set_params(x)
```

Arguments

x	a data frame that can be coerced into a tibble

specify — *Specify the response and explanatory variables with* specify *also converting character variables chosen to be* factors

Description

Specify the response and explanatory variables with specify also converting 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 *A shortcut wrapper function to get the observed test statistic for a t test*

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 *A tidier version of t.test for two sample tests*

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	#' @param 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 stats::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 the distribution of the simulation-based inferential statistics or the theoretical distribution (or both!)*

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", ...)
```

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
obs_stat_color	a character or hex string specifying the color of the observed statistic as a vertical line on the plot
pvalue_fill	a character or hex string specifying the color to shade the pvalue. In previous versions of the package this was the shade_color argument
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
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.
endpoints_color	a character or hex string specifying the color of the observed statistic as a vertical line on the plot
ci_fill	a character or hex string specifying the color to shade the confidence interval
...	other arguments passed along to ggplot2

Value

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

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