# Package 'infer'

May 15, 2018

**Type** Package

**Title** Tidy Statistical Inference

**Version** 0.2.0

**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** assertive, dplyr (>= 0.7.0), methods, tibble, rlang (>= 0.2.0), ggplot2, magrittr

**Depends** R (>= 3.1.2)

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

**URL** https://github.com/andrewpbray/infer

**BugReports** https://github.com/andrewpbray/infer/issues

**RoxygenNote** 6.0.1

**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]

**Maintainer** Andrew Bray <abray@reed.edu>

**Repository** CRAN

**Date/Publication** 2018-05-15 03:37:47 UTC

# R topics documented:



---

`calculate` *Calculate summary statistics*

---

## Description

Calculate summary statistics

## Usage

```
calculate(x, stat, 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", and "slope". |
| `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. |
| `...` | 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 `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)
```

---

`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 = "bootstrap", ...)
```

## 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

### 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")
```

---

`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 |
| ... | currently ignored |

## 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, alternative = "two_sided", ...)
```

### 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 |
| alternative | character string specifying the direction of the alternative hypothesis. Options are "two_sided" (default), "greater", or "less". |
| ... | currently ignored |

### Examples

```
# t test for comparing mpg against automatic/manual
  mtcars %>%
    dplyr::mutate(am = factor(am)) %>%
    t_test(mpg ~ am, 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 = "#e51010", shade_color = "#efb8b8",
  direction = NULL, ...)
```

## 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 corresponding to what the observed statistic is |
| obs_stat_color | a character or hex string specifying the color of the observed statistic |
| shade_color | a character or hex string specifying the color to shade |
| direction | a string specifying in which direction the shading should occur. Options are "less", "greater", or "two_sided". Can also specify "left", "right", or "both". |
| ... | currently ignored |

## 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