Package 'infer'

November 18, 2019

Type Package

Title Tidy Statistical Inference

Version 0.5.1

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 (¿= 1.3.0),
grDevices,
purrr,
lifecycle

Depends R (¿= 3.1.2)

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

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

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

Roxygen list(markdown = TRUE)

RoxygenNote 7.0.0

VignetteBuilder knitr

R topics documented:

`calculate` *Calculate summary statistics*

Description

Maturing

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"),
  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

```
# 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 = 1000, 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 = 1000, type = "permute") %>%
 calculate("diff in means", order = c("degree", "no degree"))

# More in-depth explanation of how to use the infer package
vignette("infer")
```

---

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(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().

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

```
# 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 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

Questioning

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 1000 bootstrap samples
gss %>%
 specify(response = hours) %>%
 hypothesize(null = "point", mu = 40) %>%
 generate(reps = 1000, 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 = 1000, type = "permute")

# More in-depth explanation of how to use the infer package
vignette("infer")
```

---

get_confidence_interval

Compute confidence interval

Description

Stable

Compute a confidence interval around a summary statistic. Only simulation-based methods are (currently only) 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) 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

```
# 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 = 10000, type = "bootstrap") %>%
  # finding the null distribution
  calculate(stat = "mean") %>%
  get_confidence_interval(point_estimate = point_estimate,
                          # at the 95% confidence level
                          level = .95,
                          # using the standard error method
                          type = "se")

# More in-depth explanation of how to use the infer package
vignette("infer")
```

get_p_value *Compute p-value*

Description

Stable

Compute a p-value from a null distribution and observed statistc. 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"`, 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

```
# 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 = 10000, 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
vignette("infer")
```

`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 3,000 entries from the GSS, including demographic markers and some economic variables. Note that this data is included for demonstration only, and will not provide accurate estimates relating to GSS variables unless properly weighted. 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 3000 rows and 11 variables:

year year respondent was surveyed

age age at time of survey, truncated at 89

sex respondent's sex

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

Maturing

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)
```

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
vignette("infer")
```

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.

Details

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>
- Evgeni Chasnovski <evgeni.chasnovski@gmail.com>
- Ben Baumer <ben.baumer@gmail.com>
- Mine Cetinkaya-Rundel <mine@stat.duke.edu>

Other contributors:

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

See Also

Useful links:

- https://github.com/tidymodels/infer
- https://infer.netlify.com/
- 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.

rep_sample_n *Perform repeated sampling*

Description

Questioning

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

Maturing

`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 = 10000, 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
vignette("infer")
```

---

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

Description

Maturing

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", 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

```
# 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 = 10000, 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
vignette("infer")
```

`specify` *Specify response and explanatory variables*

Description

Maturing

`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 `factor`s.

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
vignette("infer")
```

---

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.

---

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

```
# 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

Maturing

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

```
# ...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 = 10000, type = "bootstrap") %>%
  # finding the null distribution
  calculate(stat = "mean")

# 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) %>%
  calculate(stat = "mean") %>%
  dplyr::pull()

# 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)

# More in-depth explanation of how to use the infer package
vignette("infer")
```

`%>%` *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