Package: ri2 (via r-universe)

October 9, 2024

Type Package

Title Randomization Inference for Randomized Experiments

Version 0.4.0

Description Randomization inference procedures for simple and complex randomized designs, including multi-armed trials, as described in Gerber and Green (2012, ISBN: 978-0393979954). Users formally describe their randomization procedure and test statistic. The randomization distribution of the test statistic under some null hypothesis is efficiently simulated.

License MIT + file LICENSE

Encoding UTF-8

Imports generics, ggplot2, pbapply

Depends randomizr (>= 0.16.0), estimatr

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.1.2

VignetteBuilder knitr

Repository https://acoppock.r-universe.dev

RemoteUrl https://github.com/acoppock/ri2

RemoteRef HEAD

RemoteSha 1fe8a4abd68a8eeb7d3228bb5ad31f64321f1856

Contents

`conduct_ri` *Conduct Randomization Inference*

Description

This function makes it easy to conduct three kinds of randomization inference.

Usage

```
conduct_ri(
  formula = NULL,
  model_1 = NULL,
  model_2 = NULL,
  test_function = NULL,
  assignment = "Z",
  outcome = NULL,
  declaration = NULL,
  sharp_hypothesis = 0,
  studentize = FALSE,
  IPW = TRUE,
  IPW_weights = NULL,
  sampling_weights = NULL,
  permutation_matrix = NULL,
  data,
  sims = 1000,
  progress_bar = FALSE,
  p = "two-tailed"
)
```

Arguments

`formula`	an object of class formula, as in `lm`. Use formula when conducting significance tests of an Average Treatment Effect estimate under a sharp null hypothesis. For the difference-in-means estimate, do not include covariates. For the OLS covariate-adjusted estimate, include covariates.
`model_1`	an object of class formula, as in `lm`. Models 1 and 2 must be "nested." model_1 should be the "restricted" model and model_2 should be the "unrestricted" model.
`model_2`	an object of class formula, as in `lm`. Models 1 and 2 must be "nested." model_1 should be the "restricted" model and model_2 should be the "unrestricted" model.
`test_function`	A function that takes data and returns a scalar test statistic.
`assignment`	a character string that indicates which variable is randomly assigned. Defaults to "Z".
`outcome`	a character string that indicates which variable is the outcome variable. Defaults to NULL.
`declaration`	A random assignment declaration, created by `declare_ra`.

`sharp_hypothesis`	either a numeric scalar or a numeric vector of length k - 1, where k is the number of treatment conditions. In a two-arm trial, this number is the *hypothesized* difference between the treated and untreated potential potential outcomes for each unit.. In a multi-arm trial, each number in the vector is the hypothesized difference in potential outcomes between the baseline condition and each successive treatment condition.
`studentize`	logical, defaults to FALSE. Should the test statistic be the t-ratio rather than the estimated ATE? T-ratios will be calculated using HC2 robust standard errors or their clustered equivalent. CLUSTERING NOT YET IMPLEMENTED.
`IPW`	logical, defaults to TRUE. Should inverse probability weights be calculated?
`IPW_weights`	a character string that indicates which variable is the existing inverse probability weights vector. Usually unnecessary, as IPW weights will be incorporated automatically if IPW = TRUE. Defaults to NULL.
`sampling_weights`	a character string that indicates which variable is the sampling weights vector. Optional, defaults to NULL. NOT YET IMPLEMENTED
`permutation_matrix`	An optional matrix of random assignments, typically created by `obtain_permutation_matrix`.
`data`	A data.frame.
`sims`	the number of simulations. Defaults to 1000.
`progress_bar`	logical, defaults to FALSE. Should a progress bar be displayed in the console?
`p`	Should "two-tailed", "upper", or "lower" p-values be reported? Defaults to "two-tailed". For two-tailed p-values, whether or not a simulated value is as large or larger than the observed value is determined with respect to the distance to the sharp null.

Details

1. Conduct hypothesis tests under the sharp null when the test statistic is the difference-in-means or covariate-adjusted average treatment effect estimate. 2. Conduct "ANOVA" style hypothesis tests, where the f-statistic from two nested models is the test statistic. This procedure is especially helpful when testing interaction terms under null of constant effects. 3. Arbitrary (scalar) test statistics

Examples

```
# Data from Gerber and Green Table 2.2


# Randomization Inference for the Average Treatment Effect

table_2.2 <-
    data.frame(d = c(1, 0, 0, 0, 0, 0, 1),
               y = c(15, 15, 20, 20, 10, 15, 30))

## Declare randomization procedure
declaration <- declare_ra(N = 7, m = 2)
```

```
## Conduct Randomization Inference
out <- conduct_ri(y ~ d,
                  declaration = declaration,
                  assignment = "d",
                  sharp_hypothesis = 0,
                  data = table_2.2)

summary(out)
plot(out)
tidy(out)

# Using a custom permutation matrix

permutation_matrix <-
 matrix(c(0, 0, 0, 0, 0, 0, 1,
          0, 0, 0, 0, 0, 1, 0,
          0, 0, 0, 0, 1, 0, 0,
          0, 0, 0, 1, 0, 0, 0,
          0, 0, 1, 0, 0, 0, 0,
          0, 1, 0, 0, 0, 0, 0,
          1, 0, 0, 0, 0, 0, 0),
        ncol = 7)

conduct_ri(y ~d, assignment = "d", data = table_2.2,
                   permutation_matrix = permutation_matrix)




# Randomization Inference for an Interaction


N <- 100
declaration <- randomizr::declare_ra(N = N, m = 50)

Z <- randomizr::conduct_ra(declaration)
X <- rnorm(N)
Y <- .9 * X + .2 * Z + 1 * X * Z + rnorm(N)
dat <- data.frame(Y, X, Z)

ate_obs <- coef(lm(Y ~ Z, data = dat))[2]

out <-
  conduct_ri(
    model_1 = Y ~ Z + X,
    model_2 = Y ~ Z + X + Z * X,
    declaration = declaration,
    assignment = "Z",
    sharp_hypothesis = ate_obs,
    data = dat, sims = 100
  )

plot(out)
```

```
summary(out)

summary(out, p = "two-tailed")
summary(out, p = "upper")
summary(out, p = "lower")

tidy(out)

# Randomization Inference for arbitrary test statistics

## In this example we're conducting a randomization check (in this case, a balance test).

N <- 100
declaration <- randomizr::declare_ra(N = N, m = 50)

Z <- randomizr::conduct_ra(declaration)
X <- rnorm(N)
Y <- .9 * X + .2 * Z + rnorm(N)
dat <- data.frame(Y, X, Z)

balance_fun <- function(data) {
    f_stat <- summary(lm(Z ~ X, data = data))$f[1]
    names(f_stat) <- NULL
    return(f_stat)
}

## confirm function works as expected
balance_fun(dat)

## conduct randomization inference

out <-
  conduct_ri(
    test_function = balance_fun,
    declaration = declaration,
    assignment = "Z",
    sharp_hypothesis = 0,
    data = dat, sims = 100
  )

plot(out)
summary(out)
tidy(out)
```

`ri2` *ri2 package*

Description

Randomization Inference

Index