Package ‘transportr’

July 22, 2025

Type Package

Title Transporting Intervention Effects from One Population to Another

Version 0.1.0

Description

Doubly-robust, non-parametric estimators for the transported average treatment effect from Rudolph, Williams, Stuart, and Diaz (2023) <doi:10.48550/arXiv.2304.00117> and the intent-to-treatment average treatment effect from Rudolph and van der Laan (2017) <doi:10.1111/rssb.12213>. Estimators are fit using cross-fitting and nuisance parameters are estimated using the Super Learner algorithm.

License GPL (>= 3)

Encoding UTF-8

Imports checkmate, cli, generics, ife, mlr3superlearner, origami, R6

RoxygenNote 7.3.2

URL `https://github.com/nt-williams/transportr`

BugReports `https://github.com/nt-williams/transportr/issues`

Suggests ranger, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Nicholas Williams [aut, cre, cph] (ORCID:
<`https://orcid.org/0000-0002-1378-4831`>),
Kara Rudolph [aut, cph] (ORCID:
<`https://orcid.org/0000-0002-9417-7960`>)

Maintainer Nicholas Williams <`ntwilliams.personal@gmail.com`>

Repository CRAN

Date/Publication 2025-02-20 18:10:02 UTC

Contents

`transport_ate` *Transported Average Treatment Effect*

Description

Implements two one-step estimators for the transported average treatment effect for a binary or continuous outcome. Nuisance parameters are estimated using the Super Learner algorithm.

Usage

```
transport_ate(
  data,
  trt,
  outcome,
  covar,
  pop,
  obs = NULL,
  id = NULL,
  weights = NULL,
  estimator = c("standard", "collaborative"),
  learners_trt = "glm",
  learners_pop = "glm",
  learners_outcome = "glm",
  learners_heterogeneity = "glm",
  folds = 1,
  control = transport_control()
)
```

Arguments

`data`	`[data.frame]` A `data.frame` in containing all necessary variables for the estimation problem.
`trt`	`[character(1)]` The column name of the treatment variable. This variable must be binary (0/1).
`outcome`	`[character(1)]` The column name of the outcome variable. This variable must be binary (0/1) or numeric.
`covar`	`[character]` An optional vector containing the column names of covariates to be included for adjustment.
`pop`	`[character(1)]` The column name of the population indicator variable. This variable must be binary (0/1) with 0 indicating the target population and 1 the source population.

`obs`	`[character(1)]` An optional column name for a censoring indicator the same. If missingness in the outcome is present, must be provided. This variable must be binary (0/1) with 1 indicating that the outcome is observed.
`id`	`[character(1)]` An optional column name containing cluster level identifiers.
`weights`	`[character(1)]` An optional column name containing sampling weights. Currently not used.
`estimator`	`[character(1)]` The estimator to use. See details for more information.
`learners_trt`	`[character]` A vector of `mlr3superlearner` algorithms for estimation of the propensity score.
`learners_pop`	`[character]` A vector of `mlr3superlearner` algorithms for estimation of the population mechanism.
`learners_outcome`	`[character]` A vector of `mlr3superlearner` algorithms for estimation of the outcome regression.
`learners_heterogeneity`	`[character]` A vector of `mlr3superlearner` algorithms for estimation of collaborative nuisance parameters. Ignored when `estimator = "standard"`.
`folds`	`[integer(1)]` The number of folds to be used for cross-fitting.
`control`	`[list()]` Output of `transport_control()`.

Details

Estimators:

The "collaborative" estimator uses covariate dimension reduction and does not require users to have knowledge about which covariates are effect modifiers and which differ in distribution between the populations. The "standard" estimator assumes all covariates are effect modifiers and differ in distribution between the populations.

Value

An object of class `transported_ate` containing the parameter estimate.

Examples

```
gendata <- function(n, A = NULL) {
  W <- rbinom(n, 1, 0.5)
  V <- rbinom(n, 1, 0.66)
  Z <- rbinom(n, 1, 0.33)
```

```
  if (is.null(A)) A <- rbinom(n, 1, 0.5)

  S <- rbinom(n, 1, 0.4 + 0.5*W - 0.3*Z)

  Yi <- rnorm(n, A + W + A*V + 2.5*A*Z, sqrt((0.1 + 0.8*W)^2))
  Y <- ifelse(S == 1, Yi, NA_real_)

  data.frame(W = W,
             V = V,
             Z = Z,
             S = S,
             A = A,
             Y = Y,
             Yi = Yi)
}

set.seed(123)
n <- 250

tmp <- gendata(n)

transport_ate(data = tmp,
              trt = "A",
              outcome = "Y",
              covar = c("W", "V", "Z"),
              pop = "S",
              estimator = "standard",
              folds = 1)

transport_ate(data = tmp,
              trt = "A",
              outcome = "Y",
              covar = c("W", "V", "Z"),
              pop = "S",
              estimator = "collaborative",
              folds = 1)
```

`transport_control` *Set Estimation Parameters*

Description

Defines default parameters for estimators in the 'transport' package.

Usage

```
transport_control(
  .learners_folds = NULL,
  .bound = 1e+05,
```

```
    .return_full_fits = FALSE,
    .discrete = FALSE,
    .info = FALSE
  )
```

Arguments

`.learners_folds`	`[integer(1)]` The number of cross-validation folds for fitting nuisance parameters.
`.bound`	`[numeric(1)]` Determines that maximum and minimum values binomial predictions will be bounded by. The default is 1e-5, bounding predictions by 1e-5 and 0.9999.
`.return_full_fits`	`[logical(1)]` Return full 'mlr3superlearner' fits? Default is `FALSE`, return only 'mlr3superlearner' weights.
`.discrete`	`[logical(1)]` Use discrete or ensemble super learner? Default is `FALSE`.
`.info`	`[logical(1)]` Print super learner fitting info to the console? Default is `FALSE`.

Value

A list with parameter values.

Examples

```
transport_control(.learners_folds = 10)
```

`transport_ittate` *Transported Intent-to-Treat Average Treatment Effect*

Description

Implements a TMLE for the transported intent-to-treat average treatment effect. Nuisance parameters are estimated using the Super Learner algorithm.

Usage

```
transport_ittate(
  data,
  instrument,
  trt,
  outcome,
  covar,
  pop,
```

```
  obs = NULL,
  id = NULL,
  weights = NULL,
  learners_instrument = "glm",
  learners_trt = "glm",
  learners_pop = "glm",
  learners_outcome = "glm",
  folds = 1,
  control = transport_control()
)
```

Arguments

`data`	[`data.frame`] A `data.frame` in containing all necessary variables for the estimation problem.
`instrument`	[`character(1)`] The column name of the randomization variable. This variable must be binary (0/1).
`trt`	[`character(1)`] The column name of the treatment variable. This variable must be binary (0/1).
`outcome`	[`character(1)`] The column name of the outcome variable. This variable must be binary (0/1) or numeric.
`covar`	[`character`] An optional vector containing the column names of covariates to be included for adjustment.
`pop`	[`character(1)`] The column name of the population indicator variable. This variable must be binary (0/1) with 0 indicating the target population and 1 the source population.
`obs`	[`character(1)`] An optional column name for a censoring indicator the same. If missingness in the outcome is present, must be provided. This variable must be binary (0/1) with 1 indicating that the outcome is observed.
`id`	[`character(1)`] An optional column name containing cluster level identifiers.
`weights`	[`character(1)`] An optional column name containing sampling weights. Currently not used.
`learners_instrument`	[`character`] A vector of `mlr3superlearner` algorithms for estimation of the propensity score of the instrument.
`learners_trt`	[`character`] A vector of `mlr3superlearner` algorithms for estimation of the propensity score of the treatment.
`learners_pop`	[`character`] A vector of `mlr3superlearner` algorithms for estimation of the population mechanism.

`learners_outcome`	`[character]` A vector of `mlr3superlearner` algorithms for estimation of the outcome regression.
`folds`	`[integer(1)]` The number of folds to be used for cross-fitting.
`control`	`[list()]` Output of `transport_control()`.

Value

An object of class `transported_ittate` containing the parameter estimate.

Examples

```
gendata <- function(n, A = NULL, S = NULL) {
  if (is.null(S)) S <- rbinom(n, 1, 0.5)
  W1 <- rbinom(n, 1, 0.4 + (0.2 * S))
  W2 <- rnorm(n, 0.1 * S, 1)
  W3 <- rnorm(n, 1 + (0.2 * S), 1)
  if (is.null(A)) A <- rbinom(n, 1, 0.5)
  Z <- rbinom(n, 1, plogis(-log(1.6) + log(4)*A - log(1.1)*W2 - log(1.3)*W3))
 Yi <- rbinom(n, 1, plogis(log(1.6) + log(1.9)*Z - log(1.3)*W3 - log(1.2)*W1 + log(1.2)*W1*Z))
  Y <- ifelse(S == 1, Yi, NA_real_)

  data.frame(W1 = W1, W2 = W2, W3 = W3,
             S = S,
             A = A,
             Z = Z,
             Y = Y,
             Yi = Yi)
}

set.seed(123)
n <- 1000

tmp <- gendata(n)

if (requireNamespace("ranger", quietly = TRUE)) {
  transport_ittate(data = tmp,
                   trt = "Z",
                   instrument = "A",
                   outcome = "Y",
                   covar = c("W1", "W2", "W3"),
                   pop = "S",
                   folds = 1)
}
```

Index