Package: medoutcon (via r-universe)

November 21, 2024

Title Efficient Natural and Interventional Causal Mediation Analysis

Version 0.2.2

Maintainer Nima Hejazi <nh@nimahejazi.org>

Description Efficient estimators of interventional (in)direct effects in the presence of mediator-outcome confounding affected by exposure. The effects estimated allow for the impact of the exposure on the outcome through a direct path to be disentangled from that through mediators, even in the presence of intermediate confounders that complicate such a relationship. Currently supported are non-parametric efficient one-step and targeted minimum loss estimators based on the formulation of Díaz, Hejazi, Rudolph, and van der Laan (2020) <doi:10.1093/biomet/asaa085>. Support for efficient estimation of the natural (in)direct effects is also provided, appropriate for settings in which intermediate confounders are absent. The package also supports estimation of these effects when the mediators are measured using outcome-dependent two-phase sampling designs (e.g., case-cohort).

Depends R (>= 3.2.0)

Imports stats, data.table, assertthat, tibble, dplyr, zeallot, scales, stringr, origami (>= 1.0.3), glm2, sl3 (>= 1.4.3)

Suggests testthat, knitr, rmarkdown, covr, Rsolnp, nnls, SuperLearner, glmnet, hal9001 (>= 0.4.1), speedglm, xgboost, ranger, arm

Remotes github::tlverse/hal9001, github::tlverse/sl3@devel

License MIT + file LICENSE

URL https://github.com/nhejazi/medoutcon

BugReports https://github.com/nhejazi/medoutcon/issues

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.3.1

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev

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

RemoteUrl https://github.com/nhejazi/medoutcon

RemoteRef HEAD

RemoteSha e2d3fd8a64fec23f24d8cb6270642ffee94f1c9f

Contents

`confint.medoutcon` *Confidence intervals for natural/interventional (in)direct effect estimates*

Description

Compute confidence intervals for objects of class `medoutcon`, which contain estimates produced by `medoutcon`.

Usage

```
## S3 method for class 'medoutcon'
confint(object, parm = seq_len(object$theta), level = 0.95, ...)
```

Arguments

`object`	An object of class `medoutcon`, as produced by invoking `medoutcon`, for which a confidence interval is to be computed.
`parm`	A `numeric` vector indicating indices of `object$est` for which to return confidence intervals.
`level`	A `numeric` indicating the level of the confidence interval to be computed.
`...`	Other arguments. Not currently used.

`fit_moc_mech` *Fit intermediate confounding mechanism with(out) conditioning on mediators*

Description

Fit intermediate confounding mechanism with(out) conditioning on mediators

Usage

```
fit_moc_mech(
  train_data,
  valid_data = NULL,
  contrast,
  learners,
  m_names,
  w_names,
  type = c("q", "r")
)
```

Arguments

`train_data`	A `data.table` containing observed data, with columns in the order specified by the NPSEM (Y, M, R, Z, A, W), with column names set appropriately based on the input data. Such a structure is a convenience utility to passing data around to the various core estimation routines and is automatically generated by `medoutcon`.
`valid_data`	A holdout data set, with columns exactly matching those appearing in the preceding argument `data`, to be used for estimation via cross-fitting. Optional, defaulting to `NULL`.
`contrast`	A `numeric` double indicating the two values of the intervention `A` to be compared. The default value of `c(0, 1)` assumes a binary intervention node `A`.
`learners`	`Stack`, or other learner class (inheriting from `Lrnr_base`), containing a set of learners from **sl3**, to be used in fitting a model for the intermediate confounding mechanism, i.e., q = E[z|a',W] and r = E[z|a',m,w]).
`m_names`	A `character` vector of the names of the columns that correspond to mediators (M). The input for this argument is automatically generated by a call to the wrapper function `medoutcon`.
`w_names`	A `character` vector of the names of the columns that correspond to baseline covariates (W). The input for this argument is automatically generated by `medoutcon`.
`type`	A `character` vector indicating whether to condition on the mediators (M) or not. Specifically, this is an option for specifying one of two types of nuisance regressions: "r" is defined as the component that conditions on the mediators (i.e., r = E[z|a',m,w]) while "q" is defined as the component that does not (i.e., q = E[z|a',w]).

`fit_nuisance_u` *Fit pseudo-outcome regression conditioning on mediator-outcome confounder*

Description

Fit pseudo-outcome regression conditioning on mediator-outcome confounder

Usage

```
fit_nuisance_u(
  train_data,
  valid_data,
  learners,
  b_out,
  q_out,
  r_out,
  g_out,
  h_out,
  w_names
)
```

Arguments

`train_data`	A `data.table` containing observed data, with columns in the order specified by the NPSEM (Y, M, R, Z, A, W), with column names set appropriately based on the input data. Such a structure is a convenience utility to passing data around to the various core estimation routines and is automatically generated by `medoutcon`.
`valid_data`	A holdout data set, with columns exactly matching those appearing in the preceding argument `data`, to be used for estimation via cross-fitting. NOT optional for this nuisance parameter.
`learners`	`Stack`, or other learner class (inheriting from `Lrnr_base`), containing a set of learners from **sl3**, to be used in fitting a model for this nuisance parameter.
`b_out`	Output from the internal function for fitting the outcome regression `fit_out_mech`.
`q_out`	Output from the internal function for fitting the mechanism of the intermediate confounder while conditioning on mediators, i.e., `fit_moc_mech`, setting `type = "q"`.
`r_out`	Output from the internal function for fitting the mechanism of the intermediate confounder without conditioning on mediators, i.e., `fit_moc_mech`, setting `type = "r"`.
`g_out`	Output from the internal function for fitting the treatment mechanism without conditioning on mediators `fit_treat_mech`.
`h_out`	Output from the internal function for fitting the treatment mechanism conditioning on the mediators `fit_treat_mech`.

w_names — A character vector of the names of the columns that correspond to baseline covariates (W). The input for this argument is automatically generated by medoutcon.

---

fit_nuisance_v — *Fit pseudo-outcome regression conditioning on treatment and baseline*

Description

Fit pseudo-outcome regression conditioning on treatment and baseline

Usage

```
fit_nuisance_v(
  train_data,
  valid_data,
  contrast,
  learners,
  b_out,
  q_out,
  m_names,
  w_names
)
```

Arguments

train_data	A data.table containing observed data, with columns in the order specified by the NPSEM (Y, M, R, Z, A, W), with column names set appropriately based on the input data. Such a structure is a convenience utility to passing data around to the various core estimation routines and is automatically generated by medoutcon.
valid_data	A holdout data set, with columns exactly matching those appearing in the preceding argument data, to be used for estimation via cross-fitting. Not optional for this nuisance parameter.
contrast	A numeric double indicating the two values of the intervention A to be compared. The default value of c(0, 1) assumes a binary intervention node A.
learners	Stack, or other learner class (inheriting from Lrnr_base), containing a set of learners from **sl3**, to be used in fitting a model for this nuisance parameter.
b_out	Output from the internal function for fitting the outcome regression fit_out_mech.
q_out	Output from the internal function for fitting the mechanism of the intermediate confounder while conditioning on the mediators, i.e., fit_moc_mech, setting type = "q".
m_names	A character vector of the names of the columns that correspond to mediators (M). The input for this argument is automatically generated by a call to the wrapper function medoutcon.

w_names	A character vector of the names of the columns that correspond to baseline covariates (W). The input for this argument is automatically generated by medoutcon.

---

fit_out_mech *Fit outcome regression*

Description

Fit outcome regression

Usage

```
fit_out_mech(
  train_data,
  valid_data = NULL,
  contrast,
  learners,
  m_names,
  w_names
)
```

Arguments

train_data	A data.table containing the observed data, with columns in the order specified by the NPSEM (Y, M, R, Z, A, W), with column names set based on the input data. Such a structure is a convenience utility to passing data around to the various core estimation routines and is automatically generated medoutcon.
valid_data	A holdout data set, with columns exactly matching those appearing in the preceding argument data, to be used for estimation via cross-fitting. Optional, defaulting to NULL.
contrast	A numeric double indicating the two values of the intervention A to be compared. The default of c(0, 1) assumes a binary intervention node A.
learners	Stack, or other learner class (inheriting from Lrnr_base), containing a set of learners from **sl3**, to be used in fitting the outcome regression, i.e., b(A,Z,M,W).
m_names	A character vector of the names of the columns that correspond to mediators (M). The input for this argument is automatically generated by medoutcon.
w_names	A character vector of the names of the columns that correspond to baseline covariates (W). The input for this argument is automatically generated by medoutcon.

`fit_treat_mech` *Fit propensity scores for treatment contrasts*

Description

Fit propensity scores for treatment contrasts

Usage

```
fit_treat_mech(
  train_data,
  valid_data = NULL,
  contrast,
  learners,
  m_names,
  w_names,
  type = c("g", "h"),
  bounds = c(0.01, 0.99)
)
```

Arguments

`train_data`	A `data.table` containing the observed data; columns are in the order specified by the NPSEM (Y, M, R, Z, A, W), with column names set appropriately based on the data. Such a structure is merely a convenience utility to passing data around to the various core estimation routines and is automatically generated `medoutcon`.
`valid_data`	A holdout data set, with columns exactly matching those appearing in the preceding argument `train_data`, to be used for estimation via cross-fitting. Optional, defaulting to `NULL`.
`contrast`	A `numeric` double indicating the two values of the intervention `A` to be compared. The default value of `c(0, 1)` assumes a binary intervention node `A`.
`learners`	`Stack`, or other learner class (inheriting from `Lrnr_base`), containing a set of learners from **sl3**, to be used in fitting a propensity score models, i.e., g := P(A = 1 | W) and h := P(A = 1 | M, W).
`m_names`	A `character` vector of the names of the columns that correspond to mediators (M). The input for this argument is automatically generated by `medoutcon`.
`w_names`	A `character` vector of the names of the columns that correspond to baseline covariates (W). The input for this argument is automatically generated by `medoutcon`.
`type`	A `character` indicating which of the treatment mechanism variants to estimate. Option `"g"` is the propensity score g(A|W) while option `"h"` is a reparameterized mediator density h(A|M,W).
`bounds`	A `numeric` vector containing two values, the first being the minimum allowable estimated propensity score value and the second being the maximum allowable for estimated propensity score value.

`medoutcon` *Efficient estimation of natural and interventional (in)direct effects*

Description

Efficient estimation of natural and interventional (in)direct effects

Usage

```
medoutcon(
  W,
  A,
  Z,
  M,
  Y,
  R = rep(1, length(Y)),
  obs_weights = rep(1, length(Y)),
  svy_weights = NULL,
  two_phase_weights = rep(1, length(Y)),
  effect = c("direct", "indirect", "pm"),
  contrast = NULL,
  g_learners = sl3::Lrnr_glm_fast$new(),
  h_learners = sl3::Lrnr_glm_fast$new(),
  b_learners = sl3::Lrnr_glm_fast$new(),
  q_learners = sl3::Lrnr_glm_fast$new(),
  r_learners = sl3::Lrnr_glm_fast$new(),
  u_learners = sl3::Lrnr_hal9001$new(),
  v_learners = sl3::Lrnr_hal9001$new(),
  d_learners = sl3::Lrnr_glm_fast$new(),
  estimator = c("tmle", "onestep"),
  estimator_args = list(cv_folds = 5L, max_iter = 5L, tiltmod_tol = 5),
  g_bounds = c(0.01, 0.99)
)
```

Arguments

`W`	A `matrix`, `data.frame`, or similar object corresponding to a set of baseline covariates.
`A`	A `numeric` vector corresponding to a treatment variable. The parameter of interest is defined as a location shift of this quantity.
`Z`	A `numeric` vector corresponding to an intermediate confounder affected by treatment (on the causal pathway between the intervention A, mediators M, and outcome Y, but unaffected itself by the mediators). When set to `NULL`, the natural (in)direct effects are estimated.
`M`	A `numeric` vector, `matrix`, `data.frame`, or similar corresponding to a set of mediators (on the causal pathway between the intervention A and the outcome Y).

`Y` A `numeric` vector corresponding to an outcome variable.

`R` A `logical` vector indicating whether a sampled observation's mediator was measured via a two-phase sampling design. Defaults to a vector of ones, indicating that two-phase sampling was not performed.

`obs_weights` A `numeric` vector of observation-level weights. The default is to give all observations equal weighting.

`svy_weights` A `numeric` vector of observation-level weights that have been computed externally, such as survey sampling weights. Such weights are used in the construction of re-weighted efficient estimators.

`two_phase_weights` A `numeric` vector of known observation-level weights corresponding to the inverse probability of the mediator being measured. Defaults to a vector of ones.

`effect` A `character` indicating whether to compute the direct or the indirect effect as discussed in <https://arxiv.org/abs/1912.09936>. This is ignored when the argument `contrast` is provided. By default, the direct effect is estimated.

`contrast` A `numeric` double indicating the two values of the intervention `A` to be compared. The default value of `NULL` has no effect, as the value of the argument `effect` is instead used to define the contrasts. To override `effect`, provide a `numeric` double vector, giving the values of a' and a*, e.g., `c(0, 1)`.

`g_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit a model for the propensity score.

`h_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit a model for a parameterization of the propensity score that conditions on the mediators.

`b_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit a model for the outcome regression.

`q_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit a model for a nuisance regression of the intermediate confounder, conditioning on the treatment and potential baseline covariates.

`r_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit a model for a nuisance regression of the intermediate confounder, conditioning on the mediators, the treatment, and potential baseline confounders.

`u_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit a pseudo-outcome regression required for in the efficient influence function.

`v_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit a pseudo-outcome regression required for in the efficient influence function.

`d_learners` A `Stack` object, or other learner class (inheriting from `Lrnr_base`), containing instantiated learners from **sl3**; used to fit an initial efficient influence function regression when computing the efficient influence function in a two-phase sampling design.

`estimator` The desired estimator of the direct or indirect effect (or contrast-specific parameter) to be computed. Both an efficient one-step estimator using cross-fitting and a cross-validated targeted minimum loss estimator (TMLE) are available. The default is the TML estimator.

`estimator_args` A `list` of extra arguments to be passed (via `...`) to the function call for the specified estimator. The default is chosen so as to allow the number of folds used to compute the one-step or TML estimators to be easily adjusted. In the case of the TML estimator, the number of update (fluctuation) iterations is limited, and a tolerance is included for the updates introduced by tilting (fluctuation) models.

`g_bounds` A `numeric` vector containing two values, the first being the minimum allowable estimated propensity score value and the second being the maximum allowable for estimated propensity scores. Defaults to `c(0.001, 0.999)`.

Examples

```
# here, we show one-step and TML estimates of the interventional direct
# effect; the indirect effect can be evaluated by a straightforward change
# to the penultimate argument. the natural direct and indirect effects can
# be evaluated by omitting the argument Z (inappropriate in this example).
# create data: covariates W, exposure A, post-exposure-confounder Z,
#              mediator M, outcome Y
n_obs <- 200
w_1 <- rbinom(n_obs, 1, prob = 0.6)
w_2 <- rbinom(n_obs, 1, prob = 0.3)
w <- as.data.frame(cbind(w_1, w_2))
a <- as.numeric(rbinom(n_obs, 1, plogis(rowSums(w) - 2)))
z <- rbinom(n_obs, 1, plogis(rowMeans(-log(2) + w - a) + 0.2))
m_1 <- rbinom(n_obs, 1, plogis(rowSums(log(3) * w + a - z)))
m_2 <- rbinom(n_obs, 1, plogis(rowSums(w - a - z)))
m <- as.data.frame(cbind(m_1, m_2))
y <- rbinom(n_obs, 1, plogis(1 / (rowSums(w) - z + a + rowSums(m))))

# one-step estimate of the interventional direct effect
os_de <- medoutcon(
  W = w, A = a, Z = z, M = m, Y = y,
  effect = "direct",
  estimator = "onestep"
)

# TML estimate of the interventional direct effect
# NOTE: improved variance estimate and de-biasing from targeting procedure
tmle_de <- medoutcon(
  W = w, A = a, Z = z, M = m, Y = y,
  effect = "direct",
  estimator = "tmle"
)
```

print.medoutcon *Print method for natural/interventional (in)direct effect estimate objects*

Description

The `print` method for objects of class `medoutcon`.

Usage

```
## S3 method for class 'medoutcon'
print(x, ...)
```

Arguments

`x`	An object of class `medoutcon`.
`...`	Other options (not currently used).

summary.medoutcon *Summary for natural/interventional (in)direct effect estimate objects*

Description

Print a convenient summary for objects of `S3` class `medoutcon`.

Usage

```
## S3 method for class 'medoutcon'
summary(object, ..., ci_level = 0.95)
```

Arguments

`object`	An object of class `medoutcon`, as produced by invoking `medoutcon`.
`...`	Other arguments. Not currently used.
`ci_level`	A `numeric` indicating the level of the confidence interval to be computed.

Index