Package ‘independenceWeights’

October 13, 2022

Title Estimates Weights for Confounding Control for Continuous-Valued Exposures

Version 0.0.1

Description Estimates weights to make a continuous-valued exposure statistically independent of a vector of pre-treatment covariates using the method proposed in Huling, Greifer, and Chen (2021) <arxiv:2107.07086>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Depends osqp (>= 0.6.0.3)

Imports locfit

Suggests cobalt

NeedsCompilation no

Author Jared Huling [aut, cre] (<https://orcid.org/0000-0003-0670-4845>), Noah Greifer [aut] (<https://orcid.org/0000-0003-3067-7154>)

Maintainer Jared Huling <jaredhuling@gmail.com>

Repository CRAN

Date/Publication 2022-05-10 15:20:02 UTC

R topics documented:

independence_weights .................................................. 2
print.independence_weights ........................................... 4
simulate_confounded_data .............................................. 5
weighted_energy_stats .................................................. 7
weighted_kernel_est .................................................... 8

Index 10
**independence_weights**  

*Construction of distance covariance optimal weights weights*

**Description**

Constructs independence-inducing weights (distance covariance optimal weights) for estimation of causal quantities for continuous-valued treatments.

**Usage**

```r
independence_weights(
  A,
  X,
  lambda = 0,
  decorrelate_moments = FALSE,
  preserve_means = FALSE,
  dimension_adj = TRUE
)
```

**Arguments**

- `A` vector indicating the value of the treatment or exposure variable. Should be a numeric vector.
- `X` matrix of covariates with number of rows equal to the length of `A` and each column is a pre-treatment covariate to be balanced between treatment groups.
- `lambda` tuning parameter for the penalty on the sum of squares of the weights.
- `decorrelate_moments` logical scalar. Whether or not to add constraints that result in exact decorrelation of weighted first order moments of `X` and `A`. Defaults to FALSE.
- `preserve_means` logical scalar. Whether or not to add constraints that result in exact preservation of weighted first order moments of `X` and `A`. Defaults to FALSE.
- `dimension_adj` logical scalar. Whether or not to add adjustment to energy distance terms that account for the dimensionality of `X`. Defaults to TRUE.

**Value**

An object of class "independence_weights" with elements:

- `weights` A vector of length `nrow(X)` containing the estimated sample weights.
- `A` Treatment vector.
- `opt` The optimization object returned by `osqp::solve_osqp()`.
- `objective` The value of the objective function at its optimal value. This is the weighted dependence statistic plus any ridge penalty on the weights.
- `D_unweighted` The value of the weighted dependence distance using all weights = 1 (i.e. unweighted).
**independence_weights**

**D_w**
The value of the weighted dependence distance of Huling, et al. (2021) using the optimal estimated weights. This is the weighted dependence statistic without the ridge penalty on the weights.

**distcov_unweighted**
The unweighted distance covariance term. This is the standard distance covariance of Szekely et al (2007). This term is always equal to D_unweighted.

**distcov_weighted**
The weighted distance covariance term. This term itself does not directly measure weighted dependence but is a critical component of it.

**energy_A**
The weighted energy distance between A and its weighted version

**energy_X**
The weighted energy distance between X and its weighted version

**ess**
The estimated effective sample size of the weights using Kish’s effective sample size formula.

An object of class "independence_weights".

**weights**
the estimated weights, the distance covariance optimal weights (DCOWs)

**A**
the treatment vector

**opt**
the object returned by whatever optimization routine was used

**objective**
the value of the optimized objective function

**distcov_unweighted**
the unweighted distance covariance between treatment and covariates

**distcov_weighted**
the weighted distance covariance between treatment and covariates

**energy_A**
the (energy) distance between the treatment distribution and the weighted treatment distribution. Smaller values mean the marginal distribution of the treatment is preserved after weighting

**energy_X**
the (energy) distance between the covariate distribution and the weighted covariate distribution. Smaller values mean the marginal distribution of the covariates is preserved after weighting

**ess**
the expected sample size after weighting. Kish’s approximation is used

**References**


**See Also**

`print.independence_weights` for printing of fitted energy balancing objects
Examples

```r
simdat <- simulate_confounded_data(seed = 999, nobs = 500)
y <- simdat$data$Y
A <- simdat$data$A
X <- as.matrix(simdat$data[c("Z1", "Z2", "Z3", "Z4", "Z5")])
dcows <- independence_weights(A, X)
print(dcows)

# distribution of response:
quantile(y)

## create grid
trt_vec <- seq(min(simdat$data$A), 50, length.out=500)

## estimate ADRF
adrf_hat <- weighted_kernel_est(A, y, dcows$weights, trt_vec)$est

## estimate naively without weights
adrf_hat_unwtd <- weighted_kernel_est(A, y, rep(1, length(y)), trt_vec)$est

ylims <- range(c(simdat$data$Y, simdat$true_adrf(trt_vec)))
plot(x = simdat$data$A, y = simdat$data$Y, ylim = ylims, xlim = c(0,50))
## true ADRF
lines(x = trt_vec, y = simdat$true_adrf(trt_vec), col = "blue", lwd=2)
## estimated ADRF
lines(x = trt_vec, y = adrf_hat, col = "red", lwd=2)
## naive estimate
lines(x = trt_vec, y = adrf_hat_unwtd, col = "green", lwd=2)
```

print.independence_weights

*Printing results for estimated energy balancing weights*

Description

Prints results for energy balancing weights
Prints weighted energy statistics for given weights

Usage

```r
## S3 method for class 'independence_weights'
print(x, digits = max(getOption("digits") - 3, 3), ...)

## S3 method for class 'weighted_energy_terms'
print(x, digits = max(getOption("digits") - 3, 3), ...)
```
simulate_confounded_data

Arguments

- x: a fitted object from `weighted_energy_stats`
- digits: minimal number of significant digits to print.
- ...: further arguments passed to or from `print.default`.

Value

Nothing returned
Nothing returned

See Also

- `independence_weights` for function which produces energy balancing weights
- `weighted_energy_stats` for function which produces energy balancing weights

Description

Simulates confounded data with continuous treatment based on Vegetabile et al’s simulation

Usage

```r
simulate_confounded_data(
  seed = 1,
  nobs = 1000,
  MX1 = -0.5,
  MX2 = 1,
  MX3 = 0.3,
  A_effect = TRUE
)
```

Arguments

- seed: random seed for reproducibility
- nobs: number of observations
- MX1: the mean of the first covariate. Defaults to -0.5, the value used in the simulations of Vegetabile, et al (2021).
- MX2: the mean of the second and fourth covariates. Defaults to 1, the value used in the simulations of Vegetabile, et al (2021).
- MX3: the probability that the fifth covariate (a binary covariate) is equal to 1. Defaults to 0.3, the value used in the simulations of Vegetabile, et al (2021).
- A_effect: whether (TRUE) or not (FALSE) the treatment has a causal effect on the outcome. If TRUE, the setting used is that of the main text of Vegetabile, et al (2021). If FALSE, the setting is that used in the Appendix of Vegetabile, et al (2021).
**Value**

An list with elements:

- **data**: A simulated dataset with `nobs` rows
- **true_adrf**: A function that inputs values of the treatment `A` and outputs the true ADRF, $E(Y(A))$, of the data-generating mechanism used to generate data.

A list with the following elements:

- **data**: a data.frame with the response (`Y`), treatment (`A`), confounders (`Z1` to `Z5`), and true average dose response function `truth`
- **true_adrf**: a function; true average dose response function
- **original_covariates**: original, untransformed covariates in the simulation setup. Do not use, as it makes the simulation setup significantly easier.

**References**


**Examples**

```r
simdat <- simulate_confounded_data(seed = 999, nobs = 500)
str(simdat$data)
A <- simdat$data$A
y <- simdat$data$Y
trt_vec <- seq(min(simdat$data$A), max(simdat$data$A), length.out=500)
ylims <- range(c(simdat$data$Y, simdat$true_adrf(trt_vec)))
plot(x = simdat$data$A, y = simdat$data$Y, ylim = ylims)
lines(x = trt_vec, y = simdat$true_adrf(trt_vec), col = "blue", lwd=2)

## naive estimate of ADRF without weights
adrf_hat_unwtd <- weighted_kernel_est(A, y, rep(1, length(y)), trt_vec)$est
lines(x = trt_vec, y = adrf_hat_unwtd, col = "green", lwd=2)
```
weighted_energy_stats  Calculation of weighted energy statistics for weighted dependence

Description
Calculates weighted energy statistics used to quantify weighted dependence

Usage
weighted_energy_stats(A, X, weights, dimension_adj = TRUE)

Arguments
- **A**: treatment vector indicating values of the treatment/exposure variable.
- **X**: matrix of covariates with number of rows equal to the length of weights and each column is a covariate.
- **weights**: a vector of sample weights.
- **dimension_adj**: logical scalar. Whether or not to add adjustment to energy distance terms that account for the dimensionality of x. Defaults to TRUE.

Value
A list with the following components:
- **D_w**: The value of the weighted dependence distance of Huling, et al. (2021) using the optimal estimated weights. This is the weighted dependence statistic without the ridge penalty on the weights.
- **distcov_unweighted**: The unweighted distance covariance term. This is the standard distance covariance of Szekely et al (2007). This term is always equal to D_unweighted.
- **distcov_weighted**: The weighted distance covariance term. This term itself does not directly measure weighted dependence but is a critical component of it.
- **energy_A**: The weighted energy distance between A and its weighted version.
- **energy_X**: The weighted energy distance between X and its weighted version.
- **ess**: The estimated effective sample size of the weights using Kish’s effective sample size formula.

An object of class "weighted_energy_terms".

- **D_w**: the value of the DCOW measure.
- **distcov_unweighted**: the unweighted distance covariance between treatment and covariates.
- **distcov_weighted**: the weighted distance covariance between treatment and covariates.
weighted_kernel_est

energy_A the (energy) distance between the treatment distribution and the weighted treatment distribution. Smaller values mean the marginal distribution of the treatment is preserved after weighting.

energy_X the (energy) distance between the covariate distribution and the weighted covariate distribution. Smaller values mean the marginal distribution of the covariates is preserved after weighting.

ess the expected sample size after weighting. Kish’s approximation is used.

References

Examples

```r
simdat <- simulate_confounded_data(seed = 999, nobs = 100)
str(simdat$data)
A <- simdat$data$A
X <- as.matrix(simdat$data[c("Z1", "Z2", "Z3", "Z4", "Z5")])
wts <- runif(length(A))
weighted_energy_stats(A, X, wts)
```

weighted_kernel_est Calculation of weighted nonparametric regression estimate of the dose response function.

Description
Calculates weighted nonparametric regression estimate of the causal average dose response function.

Usage

```r
weighted_kernel_est(A, y, weights, Aseq)
```

Arguments

- **A** vector indicating the value of the treatment or exposure variable. Should be a numeric vector.
- **y** vector of responses
- **weights** a vector of sample weights of length equal to the length of y
- **Aseq** a vector of new points for which to obtain estimates of E(Y(a))
Value

A list with the following elements

- **fit**: A fitted model object from the `lp` function
- **estimated**: A vector of estimates of a causal ADRF at the values of the treatment specified by `Aseq`
Index

independence_weights, 2, 5
lp, 9
print.default, 5
print.independence_weights, 3, 4
print.weighted_energy_terms
  (print.independence_weights), 4
simulate_confounded_data, 5
weighted_energy_stats, 5, 7
weighted_kernel_est, 8