Package 'DebiasInfer'

May 7, 2026

Type Package

Title Efficient Inference on High-Dimensional Linear Model with Missing Outcomes

Version 0.2.1

Description A statistically and computationally efficient debiasing method for conducting valid inference on the high-dimensional linear regression function with missing outcomes. The reference paper is Zhang, Giessing, and Chen (2023) <doi:10.48550/arXiv.2309.06429>.

URL https://github.com/zhangyk8/Debias-Infer/

BugReports https://github.com/zhangyk8/Debias-Infer/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports CVXR, caret, stats

Suggests MASS, glmnet

Maintainer Yikun Zhang <yikunzhang@foxmail.com>

NeedsCompilation no

Author Yikun Zhang [aut, cre] (ORCID: <https://orcid.org/0000-0003-3905-6346>), Alexander Giessing [aut] (ORCID: <https://orcid.org/0000-0002-6917-0652>), Yen-Chi Chen [aut] (ORCID: <https://orcid.org/0000-0002-4485-306X>)

Repository CRAN

Date/Publication 2026-03-13 06:20:02 UTC

Contents

DebiasProg *The proposed debiasing (primal) program.*

Description

This function implements our proposed debiasing (primal) program that solves for the weights for correcting the Lasso pilot estimate.

Usage

```
DebiasProg(X, x, Pi, gamma_n = 0.1)
```

Arguments

X	The input design n*d matrix.
x	The current query point, which is a 1*d array.
Pi	An n*n diagonal matrix with (estimated) propensity scores as its diagonal entries.
gamma_n	The regularization parameter "γ/n". (Default: gamma_n=0.1.)

Value

The estimated weights by our debiasing program, which is a n-dim vector.

Author(s)

Yikun Zhang, <yikunzhang@foxmail.com>

References

Zhang, Y., Giessing, A. and Chen, Y.-C. (2023) *Efficient Inference on High-Dimensional Linear Model with Missing Outcomes.* https://arxiv.org/abs/2309.06429.

Fu, A., Narasimhan, B. and Boyd, S. (2017) *CVXR: An R Package for Disciplined Convex Optimization. Journal of Statistical Software 94 (14): 1–34.* doi:10.18637/jss.v094.i14.

Examples

```
require(MASS)
require(glmnet)
d = 600
n = 500
set.seed(123)

Sigma = array(0, dim = c(d,d)) + diag(d)
rho = 0.1
for(i in 1:(d-1)){
  for(j in (i+1):d){
```

```
    if ((j < i+6) | (j > i+d-6)){
      Sigma[i,j] = rho
      Sigma[j,i] = rho
    }
  }
}
sig = 1

## Current query point
x_cur = rep(0, d)
x_cur[c(1, 2, 3, 7, 8)] = c(1, 1/2, 1/4, 1/2, 1/8)
x_cur = array(x_cur, dim = c(1,d))

## True regression coefficient
s_beta = 5
beta_0 = rep(0, d)
beta_0[1:s_beta] = sqrt(5)

## Generate the design matrix and outcomes
X_sim = mvrnorm(n, mu = rep(0, d), Sigma)
eps_err_sim = sig * rnorm(n)
Y_sim = drop(X_sim %*% beta_0) + eps_err_sim

obs_prob = 1 / (1 + exp(-1 + X_sim[, 7] - X_sim[, 8]))
R_sim = rep(1, n)
R_sim[runif(n) >= obs_prob] = 0

## Estimate the propensity scores via the Lasso-type generalized linear model
zeta = 5*sqrt(log(d)/n)/n
lr1 = glmnet(X_sim, R_sim, family = "binomial", alpha = 1, lambda = zeta,
             standardize = TRUE, thresh=1e-6)
prop_score = drop(predict(lr1, newx = X_sim, type = "response"))

## Estimate the debiasing weights
w_obs = DebiasProg(X_sim, x_cur, Pi=diag(prop_score), gamma_n = 0.1)
```

DebiasProgCV *The proposed debiasing (primal) program with cross-validation.*

Description

This function implements our proposed debiasing program that selects the tuning parameter "γ/n" by cross-validation and returns the final debiasing weights.

Usage

```
DebiasProgCV(X, x, prop_score, gamma_lst = NULL, cv_fold = 5, cv_rule = "1se")
```

Arguments

X	The input design n*d matrix.
x	The current query point, which is a 1*d array.
prop_score	An n-dim numeric vector with (estimated) propensity scores as its entries.
gamma_lst	A numeric vector with candidate values for the regularization parameter "γ/n". (Default: gamma_lst=NULL. Then, gamma_lst contains 41 equally spacing value between 0.001 and max(abs(x)).)
cv_fold	The number of folds for cross-validation on the dual program. (Default: cv_fold=5.)
cv_rule	The criteria/rules for selecting the final value of the regularization parameter "γ/n" in the dual program. (Default: cv_rule="1se". The candidate choices include "1se", "minfeas", and "mincv".)

Value

A list that contains three elements.

w_obs	The final estimated weights by our debiasing program.
ll_obs	The final value of the solution to our debiasing dual program.
gamma_n_opt	The final value of the tuning parameter "γ/n" selected by cross-validation.

Author(s)

Yikun Zhang, <yikunzhang@foxmail.com>

References

Zhang, Y., Giessing, A. and Chen, Y.-C. (2023) *Efficient Inference on High-Dimensional Linear Model with Missing Outcomes*. https://arxiv.org/abs/2309.06429.

Examples

```
require(MASS)
require(glmnet)
d = 600
n = 500
set.seed(123)

Sigma = array(0, dim = c(d,d)) + diag(d)
rho = 0.1
for(i in 1:(d-1)){
  for(j in (i+1):d){
    if ((j < i+6) | (j > i+d-6)){
      Sigma[i,j] = rho
      Sigma[j,i] = rho
    }
  }
}
sig = 1
```

```
## Current query point
x_cur = rep(0, d)
x_cur[c(1, 2, 3, 7, 8)] = c(1, 1/2, 1/4, 1/2, 1/8)
x_cur = array(x_cur, dim = c(1,d))

## True regression coefficient
s_beta = 5
beta_0 = rep(0, d)
beta_0[1:s_beta] = sqrt(5)

## Generate the design matrix and outcomes
X_sim = mvrnorm(n, mu = rep(0, d), Sigma)
eps_err_sim = sig * rnorm(n)
Y_sim = drop(X_sim %*% beta_0) + eps_err_sim

obs_prob = 1 / (1 + exp(-1 + X_sim[, 7] - X_sim[, 8]))
R_sim = rep(1, n)
R_sim[runif(n) >= obs_prob] = 0

## Estimate the propensity scores via the Lasso-type generalized linear model
zeta = 5*sqrt(log(d)/n)/n
lr1 = glmnet(X_sim, R_sim, family = "binomial", alpha = 1, lambda = zeta,
             standardize = TRUE, thresh=1e-6)
prop_score = drop(predict(lr1, newx = X_sim, type = "response"))

## Estimate the debiasing weights with the tuning parameter selected by cross-validations.
deb_res = DebiasProgCV(X_sim, x_cur, prop_score, gamma_lst = c(0.1, 0.5, 1),
                       cv_fold = 5, cv_rule = '1se')
```

`DualCD` *Coordinate descent algorithm for solving the dual form of our debiasing program.*

Description

This function implements the coordinate descent algorithm for the debiasing dual program. More details can be found in Appendix A of our paper.

Usage

```
DualCD(
  X,
  x,
  Pi = NULL,
  gamma_n = 0.05,
  ll_init = NULL,
```

```
  eps = 1e-09,
  max_iter = 5000
)
```

Arguments

X	The input design n*d matrix.
x	The current query point, which is a 1*d array.
Pi	An n*n diagonal matrix with (estimated) propensity scores as its diagonal entries.
gamma_n	The regularization parameter "γ/n". (Default: gamma_n=0.05.)
ll_init	The initial value of the dual solution vector. (Default: ll_init=NULL. Then, the vector with all-one entries is used.)
eps	The tolerance value for convergence. (Default: eps=1e-9.)
max_iter	The maximum number of coordinate descent iterations. (Default: max_iter=5000.)

Value

The solution vector to our dual debiasing program.

Author(s)

Yikun Zhang, <yikunzhang@foxmail.com>

References

Zhang, Y., Giessing, A. and Chen, Y.-C. (2023) *Efficient Inference on High-Dimensional Linear Model with Missing Outcomes.* https://arxiv.org/abs/2309.06429.

Examples

```
require(MASS)
require(glmnet)
d = 1000
n = 900

Sigma = array(0, dim = c(d,d)) + diag(d)
rho = 0.1
for(i in 1:(d-1)){
  for(j in (i+1):d){
    if ((j < i+6) | (j > i+d-6)){
      Sigma[i,j] = rho
      Sigma[j,i] = rho
    }
  }
}
sig = 1

## Current query point
```

```
x_cur = rep(0, d)
x_cur[c(1, 2, 3, 7, 8)] = c(1, 1/2, 1/4, 1/2, 1/8)
x_cur = array(x_cur, dim = c(1,d))

## True regression coefficient
s_beta = 5
beta_0 = rep(0, d)
beta_0[1:s_beta] = sqrt(5)

## Generate the design matrix and outcomes
X_sim = mvrnorm(n, mu = rep(0, d), Sigma)
eps_err_sim = sig * rnorm(n)
Y_sim = drop(X_sim %*% beta_0) + eps_err_sim

obs_prob = 1 / (1 + exp(-1 + X_sim[, 7] - X_sim[, 8]))
R_sim = rep(1, n)
R_sim[runif(n) >= obs_prob] = 0

## Estimate the propensity scores via the Lasso-type generalized linear model
zeta = 5*sqrt(log(d)/n)/n
lr1 = glmnet(X_sim, R_sim, family = "binomial", alpha = 1, lambda = zeta,
             standardize = TRUE, thresh=1e-6)
prop_score = drop(predict(lr1, newx = X_sim, type = "response"))

## Solve the debiasing dual program
ll_cur = DualCD(X_sim, x_cur, Pi = diag(prop_score), gamma_n = 0.1, ll_init = NULL,
                eps=1e-9, max_iter = 5000)
```

DualObj *The objective function of the debiasing dual program.*

Description

This function computes the objective function value of the debiasing dual program.

Usage

```
DualObj(X, x, Pi, ll_cur, gamma_n = 0.05)
```

Arguments

X	The input design n*d matrix.
x	The current query point, which is a 1*d array.
Pi	An n*n diagonal matrix with (estimated) propensity scores as its diagonal entries.
ll_cur	The current value of the dual solution vector.
gamma_n	The regularization parameter "γ/n". (Default: gamma_n=0.1.)

Value

The value of the objective function of our dual debiasing program.

Author(s)

Yikun Zhang, <yikunzhang@foxmail.com>

References

Zhang, Y., Giessing, A. and Chen, Y.-C. (2023) *Efficient Inference on High-Dimensional Linear Model with Missing Outcomes.* https://arxiv.org/abs/2309.06429.

Examples

```
require(MASS)
require(glmnet)
d = 1000
n = 900

Sigma = array(0, dim = c(d,d)) + diag(d)
rho = 0.1
for(i in 1:(d-1)){
  for(j in (i+1):d){
    if ((j < i+6) | (j > i+d-6)){
      Sigma[i,j] = rho
      Sigma[j,i] = rho
    }
  }
}
sig = 1

## Current query point
x_cur = rep(0, d)
x_cur[c(1, 2, 3, 7, 8)] = c(1, 1/2, 1/4, 1/2, 1/8)
x_cur = array(x_cur, dim = c(1,d))

## True regression coefficient
s_beta = 5
beta_0 = rep(0, d)
beta_0[1:s_beta] = sqrt(5)

## Generate the design matrix and outcomes
X_sim = mvrnorm(n, mu = rep(0, d), Sigma)
eps_err_sim = sig * rnorm(n)
Y_sim = drop(X_sim %*% beta_0) + eps_err_sim

obs_prob = 1 / (1 + exp(-1 + X_sim[, 7] - X_sim[, 8]))
R_sim = rep(1, n)
R_sim[runif(n) >= obs_prob] = 0

## Estimate the propensity scores via the Lasso-type generalized linear model
zeta = 5*sqrt(log(d)/n)/n
```

```
lr1 = glmnet(X_sim, R_sim, family = "binomial", alpha = 1, lambda = zeta,
             standardize = TRUE, thresh=1e-6)
prop_score = drop(predict(lr1, newx = X_sim, type = "response"))

## Solve the debiasing dual program and estimate the dual objective function value
ll_cur = DualCD(X_sim, x_cur, Pi = diag(prop_score), gamma_n = 0.1, ll_init = NULL,
                eps=1e-9, max_iter = 5000)
dual_val = DualObj(X_sim, x_cur, Pi=diag(prop_score), ll_cur=ll_cur, gamma_n=0.1)
```

SoftThres *The soft-thresholding function*

Description

This function implements the soft-threshold operator $S_\lambda(x) = sign(x) \cdot (x - \lambda)_+$.

Usage

```
SoftThres(theta, lamb)
```

Arguments

theta	The input numeric vector.
lamb	The thresholding parameter.

Value

The resulting vector after soft-thresholding.

Author(s)

Yikun Zhang, <yikunzhang@foxmail.com>

Examples

```
a = c(1,2,4,6)
SoftThres(theta=a, lamb=3)
```

Index