Package 'RIIM'

March 12, 2025

Type Package

Title Randomization-Based Inference Under Inexact Matching

Version 2.0.0

Description

Randomization-based inference for average treatment effects in potentially inexactly matched observational studies. It implements the inverse post-matching probability weighting framework proposed by the authors. The post-matching probability calculation follows the approach of Pimentel and Huang (2024) <doi:10.1093/jrsssb/qkae033>. The optimal full matching method is based on Hansen (2004) <doi:10.1198/106186006X137047>. The variance estimator extends the method proposed in Fogarty (2018) <doi:10.1111/rssb.12290> from the perfect randomization settings to the potentially inexact matching case. Comparisons are made with conventional methods, as described in Rosenbaum (2002) <doi:10.1007/978-1-4757-3692-2>, Fogarty (2018) <doi:10.1111/rssb.12290>, and Kang et al. (2016) <doi:10.1214/15-aoas894>.

Imports MASS, xgboost, optmatch

Suggests VGAM, mvtnorm

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Jianan Zhu [aut, cre],
Jeffrey Zhang [aut],
Zijian Guo [aut],
Siyu Heng [aut]

Maintainer Jianan Zhu <jz4698@nyu.edu>

Repository CRAN

Date/Publication 2025-03-12 17:40:05 UTC

Contents

`conditional_p` *The function for calculating the post-matching treatment assignment probabilities*

Description

This function calculates the regularized post-matching treatment assignment probabilities, as described in Zhu, Zhang, Guo, and Heng (2024).

Usage

```
conditional_p(treated.subject.index,matched.control.subject.index,p,alpha=0.1)
```

Arguments

`treated.subject.index`	The index list for treated subjects after matching.
`matched.control.subject.index`	The index list for control subjects after matching.
`p`	The estimated propensity score vector.
`alpha`	Prespecified small number as the regularization threshold. The default is 0.1.

Value

`prob`	The regularized post-matching treatment assignment probabilities vector.

Author(s)

Jianan Zhu (maintainer), Jeffrey Zhang, Zijian Guo and Siyu Heng.

References

Zhu, J., Zhang, J., Guo, Z., and Heng, S. (2024). Randomization-Based Inference for Average Treatment Effect in Inexactly Matched Observational Studies. arXiv:2308.02005.

Examples

```
library(MASS)
library(xgboost)
library(optmatch)

# Generate data
set.seed(1)
d = 5
n = 50
sigma = diag(d)
```

```
# Generate X
X_d = mvtnorm::rmvnorm(n, mean = rep(0,d), sigma = sigma)
X_d[,4] = VGAM::rlaplace(n, location = 0, scale = sqrt(2)/2)
X_d[,5] = VGAM::rlaplace(n, location = 0, scale = sqrt(2)/2)

# Generate Z
C = -2.5
fx = 0.1*(X_d[,1])^3 + 0.3*(X_d[,2]) + 0.2*log((X_d[,3])^2) + 0.1*(X_d[,4]) +
     0.2*X_d[,5] + abs(X_d[,1]*X_d[,2]) + (X_d[,3]*X_d[,4])^2 + 0.5*(X_d[,2]*X_d[,4])^2 +
      rnorm(n,0,1) + C
p = exp(fx)/(1+exp(fx)) # the probability of receiving the treatment
Z = rep(0,length(p))
for(i in seq_along(p)){
  Z[i] = rbinom(1,1,p[i])
}

# Generate Y
Y_0 = 0.2*(X_d[,1])^3 + 0.2*abs(X_d[,2]) + 0.2*X_d[,3]^3 + 0.5*abs(X_d[,4]) +
0.3*X_d[,5] + rnorm(n,0,1)
Y_1 = Y_0 + 1 + 0.3*X_d[,1] + 0.2*X_d[,3]^3
Y = (1-Z)*Y_0 + Z*Y_1

# Smahal function
  smahal=
    function(z,X){
      X<-as.matrix(X)
      n<-dim(X)[1]
      rownames(X)<-1:n
      k<-dim(X)[2]
      m<-sum(z)
      for (j in 1:k) X[,j]<-rank(X[,j])
      cv<-cov(X)
      vuntied<-var(1:n)
      rat<-sqrt(vuntied/diag(cv))
      cv<-diag(rat)%*%cv%*%diag(rat)
      out<-matrix(NA,m,n-m)
      Xc<-X[z==0,]
      Xt<-X[z==1,]
      rownames(out)<-rownames(X)[z==1]
      colnames(out)<-rownames(X)[z==0]
      icov<-MASS::ginv(cv)
      for (i in 1:m) out[i,]<-mahalanobis(Xc,Xt[i,],icov,inverted=TRUE)
      out
    }

  # Matching
  treated.index = which(Z == 1)
  propscore.model = glm(Z ~ X_d, family = 'binomial',x=TRUE,y=TRUE)
  treated = propscore.model$y
  Xmat=propscore.model$x[,-1]
  distmat=smahal(treated,Xmat)
  logit.propscore=predict(propscore.model)
  subject.index=seq(1,length(treated),1)
```

```
  rownames(distmat)=subject.index[treated==1]
  colnames(distmat)=subject.index[treated==0]
  matchvec=fullmatch(distmat,min.controls=0.0001,max.controls=10000)
  treated.subject.index=vector("list",length(treated.index))
  matched.control.subject.index=vector("list",length(treated.index))
  matchedset.index=substr(matchvec,start=3,stop=10)
  matchedset.index.numeric=as.numeric(matchedset.index)
  subjects.match.order=as.numeric(names(matchvec))
  matchedset_index = length(unique(matchedset.index.numeric))

  # total number in each set
  l <- rep(0,length(treated.subject.index))
  for(i in 1:length(treated.subject.index)){
    matched.set.temp=which(matchedset.index.numeric==i)
    matched.set.temp.indices=subjects.match.order[matched.set.temp]
    l[i] <- length(matched.set.temp.indices)
  }

  # the order of matchvec
  for(i in 1:length(treated.index)){
  matched.set.temp=which(matchedset.index.numeric==i)
  matched.set.temp.indices=subjects.match.order[matched.set.temp]
  treated.temp.index=which(matched.set.temp.indices %in% treated.index)
  if(length(treated.temp.index) != 0){
    treated.subject.index[[i]]=matched.set.temp.indices[treated.temp.index]
    matched.control.subject.index[[i]]=matched.set.temp.indices[-treated.temp.index]
  }
}

  # remove null
  if(sum(sapply(treated.subject.index, is.null)) != 0){
   treated.subject.index<- treated.subject.index[-which(sapply(treated.subject.index, is.null))]
    matched.control.subject.index<-matched.control.subject.index[-which(sapply(
    matched.control.subject.index,is.null))]
  }

# Use XGBoost to estimate propensity score
  length_all = length(Z)
  length_X = ncol(X_d)
  df = data.frame(Z,X_d)
  index_model1 = sample(length_all,length_all/2)
  df1 = df[index_model1,]
  df2 = df[-index_model1,]
  prob = rep(0,length_all)
  xgb.model1 = xgboost(data = as.matrix(df1[2:length_X]), label = df1$Z, nrounds = 2,
  objective = "binary:logistic",verbose = 0)
  prob[-index_model1] = predict(xgb.model1, as.matrix(df2[2:length_X]))
  xgb.model2 = xgboost(data = as.matrix(df2[2:length_X]), label = df2$Z, nrounds = 2,
  objective = "binary:logistic",verbose = 0)
  prob[index_model1] = predict(xgb.model2, as.matrix(df1[2:length_X]))

  # calculate the post-matching treatment assignment probabilities
  p = conditional_p(treated.subject.index,matched.control.subject.index,prob,alpha=0.1)
```

IPPW *Randomization-based inference using inverse post-matching probability weighting (IPPW)*

Description

This function implements the inverse post-matching probability weighting (IPPW) method proposed in Zhu, Zhang, Guo and Heng (2024). It can be used for conducting randomization-based inference for the sample average treatment effect in potentially inexactly matched observational studies. By default, the matching design implemented in the package is optimal full matching, and the estimated propensity scores used in our method are obtained by the XGBoost method.

Usage

```
IPPW(Y, Z, X, min.controls = 0.0001,max.controls = 10000, caliper = TRUE,
     calipersd = 0.2, dim = FALSE, gamma = 0.1, alpha = 0.05)
```

Arguments

Y	The observed outcome vector.
Z	The treatment indicator vector.
X	The covariates matrix. Each row is an individual.
min.controls	The minimum ratio of controls to treatments permitted within a matched set. The default is 0.0001.
max.controls	The maximum ratio of controls to treatments permitted within a matched set. The default is 10000.
caliper	Whether adding a propensity score caliper or not. The default is TRUE.
calipersd	The standard deviation of the logit propensity score for caliper. The default is 0.2.
dim	Whether using difference-in-means estimator to estimate the average treatment effect. The default is FALSE.
gamma	The regularization threshold. The default is 0.1.
alpha	The prespecified level alpha for the 1-alpha condifence interval.

Value

estimate	The estimate for the sample average treatment effect using the IPPW estimator.
var	The variance for the sample average treatment effect using the IPPW estimator.
low	The lower bound for the sample average treatment effect using the IPPW estimator.
up	The upper bound for the sample average treatment effect using the IPPW estimator.
CI	The confidence interval for the sample average treatment effect using the IPPW estimator.
balance	The pre- and post-matching covariate balance table.

Author(s)

Jianan Zhu (maintainer), Jeffrey Zhang, Zijian Guo and Siyu Heng.

References

Fogarty, C. B. (2018). On mitigating the analytical limitations of finely stratified experiments. Journal of the Royal Statistical Society Series B: Statistical Methodology, 80(5), 1035-1056.

Hansen, B. B. (2004). Full matching in an observational study of coaching for the SAT. Journal of the American Statistical Association, 99(467), 609-618.

Hansen, B. B. and Klopfer, S. O. (2006). Optimal full matching and related designs via network flows. Journal of Computational and Graphical Statistics, 15(3), 609-627.

Kang, H., Kreuels, B., May, J., and Small, D. S. (2016). Full matching approach to instrumental variables estimation with application to the effect of malaria on stunting. The Annals of Applied Statistics, 10(1), 335-364.

Rosenbaum, P. R. (1991). A characterization of optimal designs for observational studies. Journal of the Royal Statistical Society: Series B (Methodological), 53(3), 597-610.

Zhu, J., Zhang, J., Guo, Z., and Heng, S. (2024). Randomization-Based Inference for Average Treatment Effect in Inexactly Matched Observational Studies. arXiv preprint, arXiv:2308.02005.

Examples

```
library(MASS)
library(xgboost)
library(optmatch)

# Generate data
set.seed(1)
d = 3
n = 30
sigma = diag(d)

# Generate X
X_d = mvtnorm::rmvnorm(n, mean = rep(0,d), sigma = sigma)

# Generate Z
C = -2.5
fx = 0.1*(X_d[,1])^3 + 0.3*(X_d[,2]) + 0.2*log((X_d[,3])^2) +
     abs(X_d[,1]*X_d[,2]) + rnorm(n,0,1) + C
p = exp(fx)/(1+exp(fx)) # the probability of receiving the treatment
Z = rep(0,length(p))
for(i in seq_along(p)){
  Z[i] = rbinom(1,1,p[i])
}

# Generate Y
Y_0 = 0.2*(X_d[,1])^3 + 0.2*abs(X_d[,2]) + 0.5*abs(X_d[,3]) + rnorm(n,0,1)
Y_1 = Y_0 + 1 + 0.3*X_d[,1] + 0.2*X_d[,3]^3
Y = (1-Z)*Y_0 + Z*Y_1
```

```
# The output
est = IPPW(Y,Z,X_d,min.controls = 0.01,max.controls = 100,caliper=FALSE,
calipersd = 0.2,dim=FALSE,gamma=0.1,alpha=0.05)$estimate
est
```

IPPW_IV *The bias-corrected Wald estimator for the complier average treatment effect*

Description

This function implements the bias-corrected Wald estimator for randomization-based estimation and inference for the complier average treatment effect under inexact matching, proposed in Zhu, Zhang, Guo and Heng (2024). By default, the matching design implemented in the package is optimal full matching, and the estimated propensity scores used in our method are obtained by the XGBoost method.

Usage

```
IPPW_IV(Y, Z, X, D, min.controls = 0.0001, max.controls = 10000,
        caliper = TRUE, calipersd = 0.2, classical = FALSE, gamma = 0.1,
        lower.bound, upper.bound, by,alpha)
```

Arguments

Y	The observed outcome vector.
Z	The binary instrument vector.
X	The covariates matrix. Each row is an individual.
D	The binary treatment indicator vector.
min.controls	The minimum ratio of the unencouraged to the encouraged permitted within a matched set. The default is 0.0001.
max.controls	The maximum ratio of the unencouraged to the encouraged permitted within a matched set. The default is 10000.
caliper	Whether adding a propensity score caliper or not. The default is TRUE.
calipersd	The standard deviation of the logit propensity score for caliper. The default is 0.2.
classical	Whether using the classical estimator (proposed by Kang et al. (2016)) to estimate the complier average treatment effect. The default is FALSE.
gamma	The regularization threshold. The default is 0.1.
lower.bound	The starting value of the search region for the point estimate.
upper.bound	The end value of the search region for the point estimate.
by	The increment of the search region for the point estimate.
alpha	The prespecified level alpha for the 1-alpha confidence interval.

Value

estimate	The estimate for the complier average treatment effect using the bias-corrected Wald estimator.
CI	The confidence interval for the complier average treatment effect using the bias-corrected Wald estimator.
value	The corresponding z-scores for the hypothetical values of the complier average treatment effect.
balance	The pre- and post-matching covariate balance table.

Author(s)

Jianan Zhu (maintainer), Jeffrey Zhang, Zijian Guo and Siyu Heng.

References

Fogarty, C. B. (2018). On mitigating the analytical limitations of finely stratified experiments. Journal of the Royal Statistical Society Series B: Statistical Methodology, 80(5), 1035-1056.

Hansen, B. B. (2004). Full matching in an observational study of coaching for the SAT. Journal of the American Statistical Association, 99(467), 609-618.

Hansen, B. B. and Klopfer, S. O. (2006). Optimal full matching and related designs via network flows. Journal of Computational and Graphical Statistics, 15(3), 609-627.

Kang, H., Kreuels, B., May, J., and Small, D. S. (2016). Full matching approach to instrumental variables estimation with application to the effect of malaria on stunting. The Annals of Applied Statistics, 10(1), 335-364.

Rosenbaum, P. R. (1991). A characterization of optimal designs for observational studies. Journal of the Royal Statistical Society: Series B (Methodological), 53(3), 597-610.

Zhu, J., Zhang, J., Guo, Z., and Heng, S. (2024). Randomization-Based Inference for Average Treatment Effect in Inexactly Matched Observational Studies. arXiv preprint, arXiv:2308.02005.

Examples

```
library(MASS)
library(xgboost)
library(optmatch)

# Generate data
set.seed(1)
d = 3
n = 30
sigma = diag(d)

# generate X
X_d = mvtnorm::rmvnorm(n, mean = rep(0,d), sigma = sigma)

# generate Z
C = -2.5
fx = 0.1*(X_d[,1])^3 + 0.3*(X_d[,2]) + 0.2*log((X_d[,3])^2) +
```

```
     abs(X_d[,1]*X_d[,2]) + rnorm(n,0,1) + C
p = exp(fx)/(1+exp(fx)) # the probability of receiving the treatment
Z = rep(0,length(p))
for(i in seq_along(p)){
  Z[i] = rbinom(1,1,p[i])
}

#joint distribution
matrix = matrix(c(1,0.8,0.8,1),2,2)
sigma = mvrnorm(n,c(0,0),matrix)

# generate the treatment effect D:
fx_D0 = 0.1*X_d[,3] + 0.4*sin(X_d[,2]) + 0.4*abs(X_d[,3])- 1 + sigma[,1]
ibu <- rnorm(n,0,1)
D_0 = ifelse(fx_D0>ibu,1,0)

fx_D1 = fx_D0 + 2 + 0.8*X_d[,2]^2
D_1 = ifelse(fx_D1>ibu,1,0)
D = (1-Z)*D_0 + Z*D_1

# generate continuous outcome Y:
Y_0 = 0.4*(X_d[,1])^2 + 0.1*abs(X_d[,2]) + 0.2*cos(X_d[,3]) + sigma[,2]
Y_1 = Y_0 + 1 + 0.1*X_d[,1] + 0.3*X_d[,3]^2
Y = (1-D)*Y_0 + D*Y_1

# The output
est = IPPW_IV(Y,Z,X_d,D,min.controls = 0.01, max.controls = 100,caliper=FALSE,
      calipersd = 0.2, classical = FALSE,gamma = 0.1,lower.bound=0,upper.bound=3,
      by=0.01,alpha=0.05)$estimate
est
```

Index