Package ‘cate’

September 5, 2019

Type Package
Title High Dimensional Factor Analysis and Confounder Adjusted Testing and Estimation
Version 1.1
Date 2019-08-13
Description Provides several methods for factor analysis in high dimension (both n,p >> 1) and methods to adjust for possible confounders in multiple hypothesis testing.
Imports Matrix, MASS, esaBcv, ruv, sva, corpcor, leapp
Suggests knitr, ggplot2, gridExtra
License GPL-2
LazyData TRUE
VignetteBuilder knitr
RoxygenNote 6.1.1
NeedsCompilation no
Author Jingshu Wang [aut]. Qingyuan Zhao [aut, cre]
Maintainer Qingyuan Zhao <qz280@cam.ac.uk>
Repository CRAN
Date/Publication 2019-09-05 04:40:13 UTC

R topics documented:
cate-package ............................................................... 2
adjust.latent ............................................................ 2
cate ................................................................. 3
est.confounder.num ..................................................... 5
fa.em ............................................................... 7
fa.pc ............................................................... 8
factor.analysis .......................................................... 8
gen.sim.data ........................................................... 9
gender.sm ............................................................ 11
wrapper .............................................................. 11
**Description**

Provides several methods for factor analysis in high dimension (both n,p \( \gg 1 \)) and methods to adjust for possible confounders in multiple hypothesis testing.

**See Also**

`factor.analysis`, `cate`

---

**adjust.latent**

Adjust for latent factors, after rotationn

**Usage**

```r
adjust.latent(corr.margin, n, X.cov, Gamma, Sigma, method = c("rr", "nc", "lqs"), psi = psi.huber, nc = NULL, nc.var.correction = TRUE)
```

**Arguments**

- `corr.margin`: marginal correlations, p*d1 matrix
- `n`: sample size
- `X.cov`: estimated second moment of X, d*d matrix
- `Gamma`: estimated confounding effects, p*r matrix
- `Sigma`: diagonal of the estimated noise covariance, p*1 vector
- `method`: adjustment method
- `psi`: derivative of the loss function in robust regression, choices are `psi.huber`, `psi.bisquare`, and `psi.hampel`
- `nc`: position of the negative controls
- `nc.var.correction`: correct asymptotic variance based on our formula
The function essentially runs a regression of \( \text{corr.margin} \sim \text{Gamma} \). The sample size \( n \) is needed to have the right scale.

This function should only be called if you know what you are doing. Most of the time you want to use the main function `cate` to adjust for confounders.

Value

- \texttt{alpha}  estimated alpha, r*d1 matrix
- \texttt{beta}  estimated beta, p*d1 matrix
- \texttt{beta.cov.row}  estimated row covariance of beta, a length p vector
- \texttt{beta.cov.col}  estimated column covariance of beta, a d1*d1 matrix

See Also

- \texttt{cate}

\begin{verbatim}

cate  

\textit{The main function for confounder adjusted testing}
\end{verbatim}

Description

The main function for confounder adjusted testing

Usage

\begin{verbatim}
cate(formula, X.data = NULL, Y, r, fa.method = c("ml", "pc", "esa"),
     adj.method = c("rr", "nc", "lqs", "naive"), psi = psi.huber,
     nc = NULL, nc.var.correction = TRUE, calibrate = TRUE)
cate.fit(X.primary, X.nuis = NULL, Y, r, fa.method = c("ml", "pc",
     "esa"), adj.method = c("rr", "nc", "lqs", "naive"), psi = psi.huber,
     nc = NULL, nc.var.correction = TRUE, calibrate = TRUE)
\end{verbatim}

Arguments

- \texttt{formula}  a formula indicating the known covariates including both primary variables and nuisance variables, which are separated by |. The variables before | are primary variables and the variables after | are nuisance variables. It's OK if there is no nuisance variables, then | is not needed and \texttt{formula} becomes a typical formula with all the covariates considered primary. When there is confusion about where the intercept should be put, \texttt{cate} will include it in \texttt{X.nuis}.
- \texttt{X.data}  the data frame used for \texttt{formula}
- \texttt{Y}  outcome, n*p matrix
r  number of latent factors, can be estimated using the function est.confounder.num
fa.method  factor analysis method
adj.method  adjustment method
psi  derivative of the loss function in robust regression
nc  position of the negative controls, if d0 > 1, this should be a matrix with 2 columns
nc.var.correction  correct asymptotic variance based on our formula
calibrate  if TRUE, use the Median and the Mean Absolute Deviation (MAD) to calibrate the test statistics
X.primary  primary variables, n*d0 matrix or data frame
X.nuis  nuisance covarites, n*d1 matrix

Details

Ideally nc can either be a vector of numbers between 1 and p, if d0 = 1 or the negative controls are the same for every treatment variable, or a 2-column matrix specifying which positions of beta are known to be zero. But this is yet implemented.

Value

a list of objects

alpha  estimated alpha, r*d1 matrix
alpha.p.value  asymptotic p-value for the global chi squared test of alpha, a vector of length d1
beta  estimated beta, p*d1 matrix
beta.cov.row  estimated row covariance of beta, a length p vector
beta.cov.col  estimated column covariance of beta, a d1*d1 matrix
beta.t  asymptotic z statistics for beta
beta.p.value  asymptotic p-values for beta, based on beta.t
Y.tilde  the transformed outcome matrix, an n*p matrix
Gamma  estimated factor loadings, p*r matrix
Z  estimated latent factors
Sigma  estimated noise variance matrix, a length p vector

Functions

- cate.fit: Basic computing function called by cate

References

est.confounder.num

See Also

wrapper for wrapper functions of some existing methods.

Examples

```r
## simulate a dataset with 100 observations, 1000 variables and 5 confounders
data <- gen.sim.data(n = 100, p = 1000, r = 5)
X.data <- data.frame(X1 = data$X1)

## linear regression without any adjustment
output.naive <- cate(~ X1 | 1, X.data, Y = data$Y, r = 0, adj.method = "naive")
## confounder adjusted linear regression
output <- cate(~ X1 | 1, X.data, Y = data$Y, r = 5)
## plot the histograms of unadjusted and adjusted regression statistics
par(mfrow = c(1, 2))
hist(output.naive$beta.t)
hist(output$beta.t)

## simulate a dataset with 100 observations, 1000 variables and 5 confounders
data <- gen.sim.data(n = 100, p = 1000, r = 5)
## linear regression without any adjustment
output.naive <- cate.fit(X.primary = data$X1, X.nuis = NULL, Y = data$Y, r = 0, adj.method = "naive")
## confounder adjusted linear regression
output <- cate.fit(X.primary = data$X1, X.nuis = NULL, Y = data$Y, r = 5)
## plot the histograms of unadjusted and adjusted regression statistics
par(mfrow = c(1, 2))
hist(output.naive$beta.t)
hist(output$beta.t)
```

est.confounder.num  

Estimate the number of confounders

Description

Estimate the number of confounders

Usage

```r
est.confounder.num(formula, X.data = NULL, Y, method = c("bcv", "ed"),
                   rmax = 20, nRepeat = 20, bcv.plot = TRUE, log = "")
est.factor.num(Y, method = c("bcv", "ed"), rmax = 20, nRepeat = 12,
                bcv.plot = TRUE, log = "")
```
Arguments

**formula** a formula indicating the known covariates including both primary variables and nuisance variables, which are seperated by |. The variables before | are primary variables and the variables after | are nuisance variables. It’s OK if there is no nuisance variables, then | is not needed and formula becomes a typical formula with all the covariates considered primary. When there is confusion about where the intercept should be put, cate will include it in X.nuis.

**X.data** the data frame used for formula

**Y** outcome, n*p matrix

**method** method to estimate the number of factors. There are currently two choices, "ed" is the eigenvalue difference method proposed by Onatski (2010) and "bcv" is the bi-cross-validation method proposed by Owen and Wang (2015). "bcv" tends to estimate more weak factors and takes longer time

**rmax** the maximum number of factors to consider. If the estimated number of factors is rmax, then users are encouraged to increase rmax and run again. Default is 20.

**nRepeat** the number of repeats of bi-cross-validation. A larger nRepeat will result in a more accurate estimate of the bcv error, but will need longer time to run.

**bcv.plot** whether to plot the relative bcv error versus the number of estimated ranks. The relative bcv error is the entrywise mean square error devided by the average of the estimated noise variance.

**log** if log = "y", then the y-axis of the bcv plot is in log scale.

Value

if method is "ed", then return the estimated number of confounders/factors. If method is "bcv", then return the a list of objects

- **r** estimated number of confounders/factors
- **errors** the relative bcv errors of length 1 + rmax

Functions

- **est.confounder.num**: Estimate the number of factors

References


Examples

## example for est.confounder.num
```
data <- gen.sim.data(n = 50, p = 50, r = 5)
X.data <- data.frame(X1 = data$X1)
est.confounder.num(~ X1 | 1, X.data, data$Y, method = "ed")
est.confounder.num(~ X1 | 1, X.data, data$Y, method = "bcv")
```

## example for est.factor.num
```
n <- 50
p <- 100
r <- 5
Z <- matrix(rnorm(n * r), n, r)
Gamma <- matrix(rnorm(p * r), p, r)
Y <- Z %*% t(Gamma) + rnorm(n * p)
est.factor.num(Y, method = "ed")
est.factor.num(Y, method = "bcv")
```

---

**fa.em**

*Factor analysis via EM algorithm to maximize likelihood*

### Description

Factor analysis via EM algorithm to maximize likelihood

#### Usage

```
fa.em(Y, r, tol = 1e-06, maxiter = 1000)
```

#### Arguments

- **Y**: data matrix, a n*p matrix
- **r**: number of factors
- **tol**: a tolerance scale of change of log-likelihood for convergence in the EM iterations
- **maxiter**: maximum iterations

#### References


#### See Also

- `factor.analysis` for the main function.
factor.analysis

---

fa.pc  
**Factor analysis via principal components**

Description

Factor analysis via principal components

Usage

fa.pc(Y, r)

Arguments

- **Y**  
data matrix, a n*p matrix
- **r**  
number of factors

See Also

factor.analysis for the main function.

---

factor.analysis  
**Factor analysis**

Description

The main function for factor analysis with potentially high dimensional variables. Here we implement some recent algorithms that is optimized for the high dimensional problem where the number of samples n is less than the number of variables p.

Usage

factor.analysis(Y, r, method = c("ml", "pc", "esa"))

Arguments

- **Y**  
data matrix, a n*p matrix
- **r**  
number of factors
- **method**  
algorithm to be used

Details

The three methods are quasi-maximum likelihood (ml), principal component analysis (pc), and factor analysis using an early stopping criterion (esa).

The ml is iteratively solved the Expectation-Maximization algorithm using the PCA solution as the initial value. See Bai and Li (2012) and for more details. For the esa method, see Owen and Wang (2015) for more details.
Value

- a list of objects
  - **Gamma** estimated factor loadings
  - **Z** estimated latent factors
  - **Sigma** estimated noise variance matrix

References


See Also

- `fa.pc`, `fa.em`, `ESA`

Examples

```r
## a factor model
n <- 100
p <- 1000
r <- 5
Z <- matrix(rnorm(n * r), n, r)
Gamma <- matrix(rnorm(p * r), p, r)
Y <- Z %*% t(Gamma) + rnorm(n * p)

## to check the results, verify the true factors are in the linear span of the estimated factors.
pc.results <- factor.analysis(Y, r = 5, "pc")
sapply(summary(lm(Z ~ pc$results$Z)), function(x) x$r.squared)

ml.results <- factor.analysis(Y, r = 5, "ml")
sapply(summary(lm(Z ~ ml$results$Z)), function(x) x$r.squared)

esa.results <- factor.analysis(Y, r = 5, "esa")
sapply(summary(lm(Z ~ esa$results$Z)), function(x) x$r.squared)
```

---

gen.sim.data

### Generate simulation data set

**Description**

gen.sim.data generates data from the following model $Y = X_0 \ Beta_0^T + X_1 \ Beta_1^T + Z \ Gamma^T + E \ Sigma^1/2$, $Z|X_0, X_1 = X_0 \ Alpha_0^T + X_1 \ Alpha_1^T + D$, $\text{cov}(X_0, X_1) \sim \Sigma_X$.
Usage

gen.sim.data(n, p, r, d0 = 0, d1 = 1, X.dist = c("binary", "normal"),
alpha = matrix(0.5, r, d0 + d1), beta = NULL, beta.strength = 1,
beta.nonzero.frac = 0.05, Gamma = NULL, Gamma.strength = sqrt(p),
Gamma.beta.cor = 0, Sigma = 1, seed = NULL)

Arguments

n  number of observations
p  number of observed variables
r  number of confounders
d0 number of nuisance regression covariates
d1 number of primary regression covariates
X.dist  the distribution of X, either "binary" or "normal"
alpha  association of X and Z, a r*d vector (d = d0 + d1)
beta  treatment effects, a p*d vector
beta.strength  strength of beta
beta.nonzero.frac  if beta is not specified, fraction of nonzeros in beta
Gamma  confounding effects, a p*r matrix
Gamma.strength  strength of Gamma, more precisely the mean of square entries of Gamma * alpha
Gamma.beta.cor  the "correlation" (proportion of variance explained) of beta and Gamma
Sigma  noise variance, a p*p matrix or p*1 vector or a single real number
seed  random seed

Value

a list of objects

X0  matrix of nuisance covariates
X1  matrix of primary covariates
Y  matrix Y
Z  matrix of confounders
alpha  regression coefficients between X and Z
beta  regression coefficients between X and Y
Gamma  coefficients between Z and Y
Sigma  noise variance
beta.nonzero.pos  the nonzero positions in beta
r  number of confounders
Gender study dataset

Description

This genetics dataset is used to demonstrate the usage of `cate` in the vignette. It was originally extracted by Gagnon-Bartsch and Speed (2012) as an example of confounded multiple testing. The data included in this package contains only 500 genes that are sampled from the original 12600 genes, besides keeping all the spike-in controls.

References


Wrapper functions for some previous methods

Description

These functions provide an uniform interface to three existing methods: SVA, RUV, LEAPP. The wrapper functions transform the data into desired forms and call the corresponding functions in the packages `sva`, `ruv`, `leapp`.

Usage

```r
sva.wrapper(formula, X.data = NULL, Y, r, sva.method = c("irw", "two-step"), B = 5)
ruv.wrapper(formula, X.data = NULL, Y, r, nc, lambda = 1, ruv.method = c("RUV2", "RUV4", "RUVinv"))
leapp.wrapper(formula, X.data = NULL, Y, r, search.tuning = F, ipod.method = c("hard", "soft"))
```

Arguments

- `formula`: a formula indicating the known covariates including both primary variables and nuisance variables, which are separated by `|`. The variables before `|` are primary variables and the variables after `|` are nuisance variables. It’s OK if there is no nuisance variables, then `|` is not needed and `formula` becomes a typical formula with all the covariates considered primary. When there is confusion about where the intercept should be put, `cate` will include it in `X.nuis`. 
X.data the data frame used for formula
Y outcome, n*p matrix
r number of latent factors, can be estimated using the function est.confounder.num
sva.method parameter for sva, whether to use an iterative reweighted algorithm (irw) or a two-step algorithm (two-step).
B parameter for sva the number of iterations of the irwsva algorithm
nc parameter for ruv functions: position of the negative controls
lambda parameter for RUVinv
ruv.method either using RUV2, RUV4 or RUVinv functions
search.tuning logical parameter for leapp, whether using BIC to search for tuning parameter of IPOD.
ipod.method parameter for leapp "hard": hard thresholding in the IPOD algorithm; "soft": soft thresholding in the IPOD algorithm

Details
The beta.p.values returned is a length p vector, each for the overall effects of all the primary variables.
Only 1 variable of interest is allowed for leapp.wrapper. The method can be slow.

Value
All functions return beta.p.value which are the p-values after adjustment. For the other returned objects, refer to cate for their meaning.

Examples

```r
## this is the simulation example in Wang et al. (2015).
n <- 100
p <- 1000
r <- 2
set.seed(1)
data <- gen.sim.data(n = n, p = p, r = r,
alpha = rep(1 / sqrt(r), r),
beta.strength = 3 * sqrt(1 + 1) / sqrt(n),
Gamma.strength = c(seq(3, 1, length = r)) * sqrt(p),
Sigma = 1 / rgamma(p, 3, rate = 2),
beta.nonzero.frac = 0.05)
X.data <- data.frame(X1 = data$X1)
sva.results <- sva.wrapper(~ X1 | 1, X.data, data$Y,
r = r, sva.method = "irw")
ruv.results <- ruv.wrapper(~ X1 | 1, X.data, data$Y, r = r,
nc = sample(data$beta.zero.pos, 30), ruv.method = "RUV4")
leapp.results <- leapp.wrapper(~ X1 | 1, X.data, data$Y, r = r)
cate.results <- cate(~ X1 | 1, X.data, data$Y, r = r)

## p-values after adjustment
par(mfrow = c(2, 2))
```

hist(sva.results$beta.p.value)
hist(ruv.results$beta.p.value)
hist(leapp.results$beta.p.value)
hist(cate.results$beta.p.value)

## type I error
mean(sva.results$beta.p.value[data$beta.zero.pos] < 0.05)

## power
mean(sva.results$beta.p.value[data$beta.nonzero.pos] < 0.05)

## false discovery proportion for sva
discoveries.sva <- which(p.adjust(sva.results$beta.p.value, "BH") < 0.2)
fdp.sva <- length(setdiff(discoveries.sva, data$beta.nonzero.pos)) / max(length(discoveries.sva), 1)
fdp.sva
Index

*Topic data
  gender.sm, 11

adjust.latent, 2

cate, 2, 3, 3, 12
cate-package, 2

ESA, 9
est.confounder.num, 5
est.factor.num(est.confounder.num), 5

fa.em, 7, 9
fa.pc, 8, 9
factor.analysis, 2, 7, 8, 8

gen.sim.data, 9
gender.sm, 11

leapp, 11, 12
leapp.wrapper (wrapper), 11

ruv, 11, 12
ruv.wrapper (wrapper), 11
RUV2, 12
RUV4, 12
RUVin, 12

sva, 11, 12
sva.wrapper (wrapper), 11

wrapper, 5, 11