Package 'secure'

October 14, 2022

Type Package

Title Sequential Co-Sparse Factor Regression

Version 0.6

Date 2021-02-22

Author Aditya Mishra [aut, cre], Kun Chen [aut, cre]

Maintainer Aditya Mishra <aditya.mishra@uconn.edu>

Description Sequential factor extraction via co-sparse unit-rank estimation (SeCURE).

Depends R (>= 3.3.1),stats, utils

Imports Rcpp (>= 0.12.9), MASS

License GPL (>= 3.0)

LazyData TRUE

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

RoxygenNote 5.0.1

Repository CRAN

Date/Publication 2021-02-26 10:10:02 UTC

R topics documented:

`CellCycle` *Cell cycle data*

Description

A list of two matrices used in Lee (2002).

Usage

```
data(CellCycle)
```

Format

A list with two components:

X Chromatin immunoprecipitation data, a matrix of 1790 rows and 113 columns.

Y Eukariotic cell cycle data, a matrix of 1790 rows and 18 columns.

Details

Matrix X, the chromatin immunoprecipitation (ChIP) data contain complete binding information of a subset of 1790 genes for a total of 113 transcription factors.

Matrix Y, the Eukariotic cell cycle data were generated using alpha factor arrest method, consisting of RNA levels measured every 7 minutes for 119 minutes with a total of 18 time points covering two cell cycle of 1790 genes.

References

Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., Hannett, N. M., Harbison, C. T., Thompson, C. M., Simon, I. et al. (2002) *Transcriptional regulatory networks in saccharomyces cerevisiae. Science, 298, 799-804*.

Examples

```
# data(CellCycle)
# X <- CellCycle$X;Y <- CellCycle$Y
# n <- nrow(Y); p <- ncol(X); q <-  ncol(Y)
# control <- secure.control(spU=160/p,spV=1)
# fit.cycle <- secure.path(Y, X, nrank = 10, nlambda = 100,
#                   control = control)
```

DLBCL *Chemotherapy data*

Description

A list of two components.

Usage

```
data(DLBCL)
```

Format

A list with two components:

Y Chromatin immunoprecipitation. A matrix of 180 rows and 661 columns.

classIndex Group index of 180 patients where *1,2,3* corresponds to OxPhos, BCR and HR groups respectively.

Details

Matrix Y : Gene expression dataset from the patients with diffuse large-B-cell lymphoma (DLBCL) after chemotherapy. The data has been used for unsupervised analysis i.e. Biclustering. The data consists of expression levels of q = 661 genes from n =180 patients. Among the patients, 42, 51 and 87 of them were classified to OxPhos, BCR and HR groups, respectively. The data thus form an n by q matrix Y whose rows represent the subjects and columns correspond to the genes, and used in Rosenwald (2002).

classIndex: Out of OxPhos (oxidative phosphorylation), BCR(Bcell response) and HR (host response), the index corresponds to the groups in which these 180 subjects belongs as classified by Hoshida (2007).

References

Rosenwald, A., Wright, G., Chan, W. C., Connors, J. M., Campo, E., Fisher, R. I., Gascoyne, R. D., Muller-Hermelink, H. K., Smeland, E. B., Giltnane, J. M. et al. (2002) *The use of molecular profling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. New England Journal of Medicine, 346, 1937-1947*.

Hoshida, Y., Brunet, J.-P., Tamayo, P., Golub, T. R. and Mesirov, J. P. (2007) *Subclass mapping: Identifying common subtypes in independent disease data sets. PLoS ONE, 2, e1195*.

Examples

```
# data(DLBCL)
```

rrr.fit *Fit reduced rank regression*

Description

fit multivariate reduced rank regression for a specified rank.

Usage

```
rrr.fit(Y, X, nrank = nrank)
```

Arguments

Y	a matrix of response (n by q)
X	a matrix of covariate (n by p)
nrank	an integer specifying the desired rank

Value

coef	reduced rank estimate

Examples

```
#require(secure)
Y <- matrix(rnorm(400), 100, 4)
X <- matrix(rnorm(800), 100, 8)
rrr.fit <- rrr.fit(Y, X, nrank = 3)
```

secure.control *Internal control function for secure*

Description

list of parameters for controling secure fitting

Usage

```
secure.control(mu = 1, nu = 1.1, MMerr = 0.001, MMiter = 100,
  outTol = 1e-06, outMaxIter = 200, inMaxIter = 200, inTol = 1e-04,
  lamMaxFac = 1, lamMinFac = 1e-10, gamma0 = 2, elnetAlpha = 0.95,
  spU = 0.25, spV = 0.25)
```

Arguments

mu	penalty parameter used in enforcing orthogonality
nu	penalty parameter used in enforcing orthogonality (incremental rate of mu)
MMerr	tolerence in the majorization maximization(MM) algorithm for computing initial values when missing value occurs
MMiter	maximum number iterations in the MM algorithm
outTol	tolerence of convergence of outer loop in CURE
outMaxIter	maximum number of outer loop iteration in CURE
inMaxIter	maximum number of inner loop iteration in CURE
inTol	tolerence value required for convergence of inner loop in CURE
lamMaxFac	a multiplier of calculated lambda_max
lamMinFac	a multiplier of determing lambda_min as a fraction of lambda_max
gamma0	power parameter in the adaptive weights
elnetAlpha	elastic net penalty parameter
spU	maximum proportion of nonzero elements in each column of U
spV	maximum proportion of nonzero elements in each column of V

Value

a list of controling parameter.

---

secure.path *Sequential Co-Sparse Factor Regression*

Description

Sequential factor extraction via co-sparse unit-rank estimation (SeCURE)

Usage

```
secure.path(Y, X = NULL, nrank = 3, nlambda = 100, U0 = NULL,
  V0 = NULL, D0 = NULL, orthXU = FALSE, orthV = FALSE,
  keepPath = TRUE, control = list(), ic = c("GIC", "BICP", "AIC")[1])
```

Arguments

Y	response matrix
X	covariate matrix; when X = NULL, the fucntion performs unsupervised learning
nrank	an integer specifying the desired rank/number of factors
nlambda	number of lambda values to be used along each path
U0	initial value of U

V0	initial value of V
D0	initial value of D
orthXU	if TRUE, orthogonality of XU is required
orthV	if TRUE, orthogonality of V is required
keepPath	if TRUE, th solution paths of U, V, D are reported
control	a list of internal parameters controlling the model fitting
ic	character specifying which information criterion to use for selecting the tuning parameter: "GIC"(default), "BICP", and "AIC"

Value

C.est	estimated coefficient matrix; based on modified BIC
U	estimated U matrix (factor weights)
D	estimated singular values
V	estimated V matrix (factor loadings)
ortX	if TRUE, X is treated as an orthogonal matrix in the computation
lam	selected lambda values based on the chosen information criterion
lampath	sequences of lambda values used in model fitting. In each sequential unit-rank estimation step, a sequence of length nlambda is first generated between (lamMax*lamMaxFac, lamMax*lamMaxFac*lamMinFac) equally spaced on the log scale, in which lamMax is estimated and the other parameters are specified in secure.control. The model fitting starts from the largest lambda and stops when the maximum proportion of nonzero elements is reached in either u or v, as specified by spU and spV in secure.control.
IC	values of information criteria
Upath	solution path of U
Dpath	solution path of D
Vpath	solution path of D

References

Mishra, A., Dey, D., Chen, K. (2017) *Sequential Co-Sparse Factor Regression, To appear in Journal of Computational and Graphical Statistics (JCGS)*

Examples

```
#require(secure)

# Simulate data from a sparse factor regression model
p <- 100; q <- 100; n <- 200
xrho <- 0.5; nlambda <- 100
nrank <- 3

U <- matrix(0,ncol=nrank ,nrow=p);  V <- matrix(0,ncol=nrank ,nrow=q)
U[,1]<-c(sample(c(1,-1),8,replace=TRUE),rep(0,p-8))
```

```
U[,2]<-c(rep(0,5),sample(c(1,-1),9,replace=TRUE),rep(0,p-14))
U[,3]<-c(rep(0,11),sample(c(1,-1),9,replace=TRUE),rep(0,p-20))
V[,1]<-c(sample(c(1,-1),5,replace=TRUE)*runif(5,0.3,1),rep(0,q-5))
V[,2]<-c(rep(0,5),sample(c(1,-1),5,replace=TRUE)*runif(5,0.3,1),rep(0,q-10))
V[,3]<-c(rep(0,10),sample(c(1,-1),5,replace=TRUE)*runif(5,0.3,1),rep(0,q-15))
U[,1:3]<- apply(U[,1:3],2,function(x)x/sqrt(sum(x^2)))
V[,1:3]<- apply(V[,1:3],2,function(x)x/sqrt(sum(x^2)))
D <- diag(c(20,15,10))
C <- U%*%D%*%t(V)

Xsigma <- xrho^abs(outer(1:p, 1:p,FUN="-"))
sim.sample <- secure.sim(U,D,V,n,snr = 0.25,Xsigma,rho=0.3)
Y <- sim.sample$Y;
X <- sim.sample$X



# Fitting secure. Set maximum rank to be 4.
rank.ini <- 4

# Set largest model to about 25% sparsity
# See secure.control for setting other parameters
control <- secure.control(spU=0.25, spV=0.25)

# Complete data case.
# Fit secure without orthogonality
fit.orthF <- secure.path(Y,X,nrank=rank.ini,nlambda = nlambda,
                         control=control)
# check orthogonality
crossprod(X%*%fit.orthF$U)/n
# check solution
# fit.orthF$U
# fit.orthF$V
# fit.orthF$D

# Fit secure with orthogonality if desired. It takes longer time.
# fit.orthT <- secure.path(Y,X,nrank=rank.ini,nlambda = nlambda,
#                                    orthXU=TRUE,orthV=TRUE,control=control)
# check orthogonality
# crossprod(X%*%fit.orthT$U)/n


# 15% missing case
miss <- 0.15
t.ind <- sample.int(n*q, size = miss*n*q)
y <- as.vector(Y); y[t.ind] <- NA;  Ym <- matrix(y,n,q)

fit.orthF.miss <- secure.path(Ym, X, nrank = rank.ini, nlambda = nlambda,
                              control = control)
# fit.orthT.miss <- secure.path(Ym, X, nrank = rank.ini, nlambda = nlambda,
#                              orthXU=TRUE,orthV=TRUE, control = control)
```

secure.sim *Simulation model*

Description

genertate random samples from a sparse factor regression model

Usage

```
secure.sim(U, D, V, n, snr, Xsigma, rho = 0)
```

Arguments

U	specified value of U
D	specified value of D
V	specified value of V
n	sample size
snr	signal to noise ratio
Xsigma	covariance matrix for generating sample of X
rho	parameter defining correlated error

Value

Y	Generated response matrix
X	Generated predictor matrix

Examples

```
#require(secure)

# Simulate data from a sparse factor regression model
p <- 100; q <- 50; n <- 300
snr <- 0.5; ssigma <- 0.5; nlambda <- 200
nrank <- 3

U <- matrix(0,ncol=nrank ,nrow=p);  V <- matrix(0,ncol=nrank ,nrow=q)
U[,1]<-c(sample(c(1,-1),8,replace=TRUE),rep(0,p-8))
U[,2]<-c(rep(0,5),sample(c(1,-1),9,replace=TRUE),rep(0,p-14))
U[,3]<-c(rep(0,11),sample(c(1,-1),9,replace=TRUE),rep(0,p-20))
V[,1]<-c(sample(c(1,-1),5,replace=TRUE)*runif(5,0.3,1),rep(0,q-5))
V[,2]<-c(rep(0,5),sample(c(1,-1),5,replace=TRUE)*runif(5,0.3,1),rep(0,q-10))
V[,3]<-c(rep(0,10),sample(c(1,-1),5,replace=TRUE)*runif(5,0.3,1),rep(0,q-15))
U[,1:3]<- apply(U[,1:3],2,function(x)x/sqrt(sum(x^2)))
V[,1:3]<- apply(V[,1:3],2,function(x)x/sqrt(sum(x^2)))
D <- diag(c(20,15,10))
C <- U%*%D%*%t(V)
```

```
Xsigma <- ssigma^abs(outer(1:p, 1:p,FUN="-"))
sim.sample <- secure.sim(U,D,V,n,snr,Xsigma)
Y <- sim.sample$Y
X <- sim.sample$X
```

Index