

# Package ‘fssemR’

February 11, 2022

**Title** Fused Sparse Structural Equation Models to Jointly Infer Gene Regulatory Network

**Version** 0.1.8

**Author** Xin Zhou, Xiaodong Cai

**Maintainer** Xin Zhou <xxz220@miami.edu>

**Description** An optimizer of Fused-Sparse Structural Equation Models, which is the state of the art jointly fused sparse maximum likelihood function for structural equation models proposed by Xin Zhou and Xiaodong Cai (2018 <[doi:10.1101/466623](https://doi.org/10.1101/466623)>).

**License** GPL (>= 3)

**Encoding** UTF-8

**Depends** methods

**Imports** Rcpp, Matrix, stats, igraph, mvtnorm, qtl, stringr, glmnet, MASS, qpdf

**Suggests** plotly, knitr, rmarkdown, network, ggnetwork

**LinkingTo** Rcpp, RcppEigen

**RoxygenNote** 7.1.2

**URL** <https://github.com/Ivis4ml/fssemR>

**NeedsCompilation** yes

**Repository** CRAN

**VignetteBuilder** knitr

**Date/Publication** 2022-02-11 13:00:02 UTC

## R topics documented:

cv.multiFSSEMiPALM . . . . .	2
cv.multiFSSEMiPALM2 . . . . .	4
cv.multiNFSSEMiPALM2 . . . . .	5
cv.multiRegression . . . . .	6
cwiseGradient4FSSEM . . . . .	6

FDR	7
flinvB	7
floneB	8
fssemR	8
implFSSEM	9
initLambdaiPALM	10
initLambdaiPALM2	10
initLambdaiPALM3	11
initRhoiPALM	12
initRhoiPALM2	12
initRhoiPALM3	13
inverseB	14
invoneB	14
logLikFSSEM	15
logLikNFSSEM	15
multiFSSEMiPALM	16
multiFSSEMiPALM2	18
multiNFSSEMiPALM2	20
multiRegression	22
obj.multiRegression	23
opt.multiFSSEMiPALM	24
opt.multiFSSEMiPALM2	25
opt.multiNFSSEMiPALM2	27
pninvB	28
pnoneB	28
proc.centerFSSEM	29
proc.centerFSSEM2	29
randomFSSEMdata	30
randomFSSEMdata2	31
randomFSSEMdata4Cor	32
TPR	33
transx	33
<b>Index</b>	<b>34</b>

---

cv.multiFSSEMiPALM	<i>cv.multiFSSEMiPALM</i>
--------------------	---------------------------

---

## Description

cv.multiFSSEMiPALM

**Usage**

```
cv.multiFSSEMiPALM(  
  Xs,  
  Ys,  
  Bs,  
  Fs,  
  Sk,  
  sigma2,  
  nlambda = 20,  
  nrho = 20,  
  nfold = 5,  
  p,  
  q,  
  wt = TRUE,  
  plot = FALSE  
)
```

**Arguments**

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
nlambda	number of hyper-parameter of lasso term in CV
nrho	number of hyper-parameter of fused-lasso term in CV
nfold	CVfold number. Default 5/10
p	number of genes
q	number of eQTLs
wt	use adaptive lasso or not. Default TRUE.
plot	plot contour of cvmean or not. Default FALSE.

**Value**

list of cross-validation result

---

cv.multiFSSEMiPALM2    *cv.multiFSSEMiPALM2*

---

### Description

cv.multiFSSEMiPALM2

### Usage

```
cv.multiFSSEMiPALM2(
  Xs,
  Ys,
  Bs,
  Fs,
  Sk,
  sigma2,
  nlambda = 20,
  nrho = 20,
  nfold = 5,
  p,
  q,
  wt = TRUE,
  plot = FALSE
)
```

### Arguments

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
nlambda	number of hyper-parameter of lasso term in CV
nrho	number of hyper-parameter of fused-lasso term in CV
nfold	CVfold number. Default 5/10
p	number of genes
q	number of eQTLs
wt	use adaptive lasso or not. Default TRUE.
plot	plot contour of cvmean or not. Default FALSE.

### Value

list of cross-validation result

---

 cv.multiNFSSEMiPALM2 *cv.multiNFSSEMiPALM2*


---

**Description**

cv.multiNFSSEMiPALM2

**Usage**

```
cv.multiNFSSEMiPALM2(
  Xs,
  Ys,
  Bs,
  Fs,
  Sk,
  sigma2,
  nlambda = 20,
  nrho = 20,
  nfold = 5,
  p,
  q,
  wt = TRUE,
  plot = FALSE
)
```

**Arguments**

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
nlambda	number of hyper-parameter of lasso term in CV
nrho	number of hyper-parameter of fused-lasso term in CV
nfold	CVfold number. Default 5/10
p	number of genes
q	number of eQTLs
wt	use adaptive lasso or not. Default TRUE.
plot	plot contour of cvmean or not. Default FALSE.

**Value**

list of cross-validation result for NFSSEM

---

cv.multiRegression      *cv.multiRegression*

---

### Description

cv.multiRegression

### Usage

cv.multiRegression(Xs, Ys, Sk, ngamma = 20, nfold = 5, n, p, k)

### Arguments

Xs	eQTL matrices
Ys	Gene expression matrices
Sk	eQTL index of genes
ngamma	number of hyper-parameter in CV
nfold	CVfold number. Default 5/10
n	number of observations
p	number of genes
k	number of eQTLs

### Value

gamma\_min optimal gamma to minimize cross-validation error

---

cwiseGradient4FSSEM      *cwiseGradient4FSSEM*

---

### Description

function generator function

### Usage

cwiseGradient4FSSEM(n, c, Y, R, Y2norm, sigma2)

### Arguments

n	number of observations
c	cofactor vector
Y	Matrix of gene expression
R	Residual matrix
Y2norm	Column of YtY
sigma2	noise variance

**Value**

function whose argument is column vector  $b_i$

---

FDR

*FDR*

---

**Description**

False discovery rate for network prediction

**Usage**

FDR(X, B, PREC = 0)

**Arguments**

X	list of predicted network matrices
B	list of true network matrices
PREC	precision threshold for FDR test. Default 0.

---

flinvB

*flinvB*

---

**Description**

inversed difference of two B matrices. For adaptive fused lasso penalty

**Usage**

flinvB(Bs)

**Arguments**

Bs	list of network matrices
----	--------------------------

**Value**

inversed difference matrices





```

                                trans = FALSE)
Xs   = data$Data$X
Ys   = data$Data$Y
Sk   = data$Data$Sk

## cross-validation
## cvfitc <- cv.multiFSSEMiPALM(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
##                               sigma2 = fit$sigma2, nlambda = 10, nrho = 10,
##                               nfold = 5, p = Ng, q = Nk, wt = TRUE)

fitm <- opt.multiFSSEMiPALM(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
                             sigma2 = fit$sigma2, nlambda = 10, nrho = 10,
                             p = Ng, q = Nk, wt = TRUE)

fitc0 <- fitm$fit

(TPR(fitc0$Bs[[1]], data$Vars$B[[1]]) + TPR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
(FDR(fitc0$Bs[[1]], data$Vars$B[[1]]) + FDR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
TPR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])
FDR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])

```

---

implFSSEM

*implFSSEM*


---

## Description

implementor function of FSSEM solver

## Usage

```
implFSSEM(data = NULL, method = c("CV", "BIC"))
```

## Arguments

data	Data archive of experiment measurements, includeing eQTL matrices, Gene expression matrices of different conditions, marker of eQTLs and data generation SEM model
method	Use cross-validation (CV) or bayesian-information-criterion(BIC)

## Value

List of TPR and FDR

---

`initLambdaiPALM`      *initLambdaiPALM*

---

**Description**

`initLambdaiPALM`

**Usage**

`initLambdaiPALM(Xs, Ys, Bs, Fs, Sk, sigma2, Wl, Wf, p, k)`

**Arguments**

<code>Xs</code>	eQTL matrices
<code>Ys</code>	Gene expression matrices
<code>Bs</code>	initialized GRN-matrices
<code>Fs</code>	initialized eQTL effect matrices
<code>Sk</code>	eQTL index of genes
<code>sigma2</code>	initialized noise variance
<code>Wl</code>	weight matrices for adaptive lasso terms
<code>Wf</code>	weight matrix for adaptive fused lasso term
<code>p</code>	number of genes
<code>k</code>	number of eQTL

**Value**

`lambda_max`

---

`initLambdaiPALM2`      *initLambdaiPALM2*

---

**Description**

`initLambdaiPALM2`

**Usage**

`initLambdaiPALM2(Xs, Ys, Bs, Fs, Sk, sigma2, Wl, Wf, p, k)`

**Arguments**

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
Wl	weight matrices for adaptive lasso terms
Wf	weight matrix for adaptive fused lasso term
p	number of genes
k	number of eQTL

**Value**

lambda\_max

---

initLambdaiPALM3      *initLambdaiPALM3*

---

**Description**

initLambdaiPALM3

**Usage**

initLambdaiPALM3(Xs, Ys, Bs, Fs, Sk, sigma2, Wl, Wf, p, k)

**Arguments**

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
Wl	weight matrices for adaptive lasso terms
Wf	weight matrix for adaptive fused lasso term
p	number of genes
k	number of eQTL

**Value**

lambda\_max

---

 initRhoiPALM

*initRhoiPALM*


---

**Description**

initRhoiPALM

**Usage**

```
initRhoiPALM(Xs, Ys, Bs, Fs, Sk, sigma2, Wl, Wf, lambda, n, p)
```

**Arguments**

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
Wl	weight matrices for adaptive lasso terms
Wf	weight matrix for adaptive fused lasso term
lambda	lambda w.r.t. rho_max
n	number of observations
p	number of genes

**Value**

rho\_max

---

 initRhoiPALM2

*initRhoiPALM2*


---

**Description**

initRhoiPALM2

**Usage**

```
initRhoiPALM2(Xs, Ys, Bs, Fs, Sk, sigma2, Wl, Wf, lambda, n, p)
```

**Arguments**

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
Wl	weight matrices for adaptive lasso terms
Wf	weight matrix for adaptive fused lasso term
lambda	lambda w.r.t. rho_max
n	number of observations
p	number of genes

**Value**

rho\_max

---

initRhoiPALM3	<i>initRhoiPALM3</i>
---------------	----------------------

---

**Description**

initRhoiPALM3

**Usage**

```
initRhoiPALM3(Xs, Ys, Bs, Fs, Sk, sigma2, Wl, Wf, lambda, n, p)
```

**Arguments**

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
Wl	weight matrices for adaptive lasso terms
Wf	weight matrix for adaptive perturbation group lasso term
lambda	lambda w.r.t. rho_max
n	number of observations
p	number of genes

**Value**

rho\_max

---

*inverseB**inverseB*

---

**Description**

inverse matrices of B network for adaptive FSSEM

**Usage**

*inverseB*(Bs)

**Arguments**

Bs                    list of network matrices

**Value**

list of inversed B matrices

---

*invoneB**invoneB*

---

**Description**

if you do not want to get inversed B matrices, *invoneB* gives you a matrix with constant 1 instead in FSSEM

**Usage**

*invoneB*(Bs)

**Arguments**

Bs                    list of network matrices

**Value**

list of *invoneB* B matrices

---

logLikFSSEM	<i>logLikFSSEM</i>
-------------	--------------------

---

**Description**

logLikFSSEM

**Usage**

logLikFSSEM(Bs, Wl, Wf, lambda, rho, sigma2, Dets, n, p)

**Arguments**

Bs	Network matrices
Wl	Weights for lasso term
Wf	Weights for fused term
lambda	Hyperparameter of lasso term
rho	Hyperparameter of fused lasso term
sigma2	noise variance
Dets	determinants of I-B matrices
n	number of observations
p	number of genes

**Value**

objective value of FSSEM with specified hyper-paramters

---

logLikNFSSEM	<i>logLikNFSSEM</i>
--------------	---------------------

---

**Description**

logLikNFSSEM

**Usage**

logLikNFSSEM(Bs, Wl, Wf, lambda, rho, sigma2, Dets, n, p)

**Arguments**

Bs	Network matrices
wl	Weights for lasso term
wf	Weights for group perturb lasso term
lambda	Hyperparameter of lasso term
rho	Hyperparameter of group fused lasso term
sigma2	noise variance
Dets	determinants of I-B matrices
n	number of observations
p	number of genes

**Value**

objective value of NFSSEM with specified hyper-paramters

---

multiFSSEMiPALM

*multiFSSEMiPALM*

---

**Description**

Implementing FSSELM algorithm for network inference. If Xs is identify for different conditions, multiFSSEMiPALM will be use, otherwise, please use multiFSSEMiPALM2 for general cases

**Usage**

```
multiFSSEMiPALM(
  Xs,
  Ys,
  Bs,
  Fs,
  Sk,
  sigma2,
  lambda,
  rho,
  wl,
  wf,
  p,
  maxit = 100,
  inert = inert_opt("linear"),
  threshold = 1e-06,
  verbose = TRUE,
  sparse = TRUE,
  trans = FALSE,
  B2norm = NULL,
  strict = FALSE
)
```



**Arguments**

<code>Xs</code>	eQTL matrices
<code>Ys</code>	Gene expression matrices
<code>Bs</code>	initialized GRN-matrices
<code>Fs</code>	initialized eQTL effect matrices
<code>Sk</code>	eQTL index of genes
<code>sigma2</code>	initialized noise variance from ridge regression
<code>lambda</code>	Hyperparameter of lasso term in FSSEM
<code>rho</code>	Hyperparameter of fused-lasso term in FSSEM
<code>Wl</code>	weight matrices for adaptive lasso terms
<code>Wf</code>	weight matrix for adaptive fused lasso term
<code>p</code>	number of genes
<code>maxit</code>	maximum iteration number. Default 100
<code>inert</code>	inertial function for iPALM. Default as $k-1/k+2$
<code>threshold</code>	convergence threshold. Default $1e-6$
<code>verbose</code>	Default TRUE
<code>sparse</code>	Sparse Matrix or not
<code>trans</code>	<code>Fs</code> matrix is transposed to $k \times p$ or not. If <code>Fs</code> from ridge regression, <code>trans = TRUE</code> , else, <code>trans = FALSE</code>
<code>B2norm</code>	B2norm matrices generated from ridge regression. Default NULL.
<code>strict</code>	Converge strictly or not. Default False

**Value**

`fit` List of FSSEM model

**Bs** coefficient matrices of gene regulatory networks

**Fs** coefficient matrices of eQTL-gene effect

**mu** Bias vector

**sigma2** estimate of covariance in SEM

**Examples**

```
seed = 1234
N = 100                                # sample size
Ng = 5                                  # gene number
Nk = 5 * 3                              # eQTL number
Ns = 1                                  # sparse ratio
sigma2 = 0.01                            # sigma2
set.seed(seed)
library(fssemR)
data = randomFSSEMdata(n = N, p = Ng, k = Nk, sparse = Ns, df = 0.3, sigma2 = sigma2,
                      u = 5, type = "DG", nhub = 1, dag = TRUE)
```

```

## If we assume that different condition has different genetics perturbations (eQTLs)
## gamma = cv.multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, ngamma = 20, nfold = 5,
##                               N, Ng, Nk)
gamma = 0.6784248    ## optimal gamma computed by cv.multiRegression
fit  = multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, gamma, N, Ng, Nk,
                       trans = FALSE)

Xs   = data$Data$X
Ys   = data$Data$Y
Sk   = data$Data$Sk

cvfitc <- cv.multiFSSEMiPALM(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
                             sigma2 = fit$sigma2, nlambda = 5, nrho = 5,
                             nfold = 5, p = Ng, q = Nk, wt = TRUE)

fitc0 <- multiFSSEMiPALM(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
                         sigma2 = fit$sigma2, lambda = cvfitc$lambda, rho = cvfitc$rho,
                         Wl = inverseB(fit$Bs), Wf = flinvB(fit$Bs),
                         p = Ng, maxit = 100, threshold = 1e-5, sparse = TRUE,
                         verbose = TRUE, trans = TRUE, strict = TRUE)

(TPR(fitc0$Bs[[1]], data$Vars$B[[1]]) + TPR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
(FDR(fitc0$Bs[[1]], data$Vars$B[[1]]) + FDR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
TPR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])
FDR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])

```

---

multiFSSEMiPALM2

*multiFSSEMiPALM2*


---

## Description

Implementing FSSELM algorithm for network inference. If Xs is identify for different conditions, multiFSSEMiPALM will be use, otherwise, please use multiFSSEMiPALM2 for general cases

## Usage

```

multiFSSEMiPALM2(
  Xs,
  Ys,
  Bs,
  Fs,
  Sk,
  sigma2,
  lambda,
  rho,
  Wl,
  Wf,
  p,

```

```

maxit = 100,
inert = inert_opt("linear"),
threshold = 1e-06,
verbose = TRUE,
sparse = TRUE,
trans = FALSE,
B2norm = NULL,
strict = FALSE
)

```

### Arguments

<code>Xs</code>	eQTL matrices
<code>Ys</code>	Gene expression matrices
<code>Bs</code>	initialized GRN-matrices
<code>Fs</code>	initialized eQTL effect matrices
<code>Sk</code>	eQTL index of genes
<code>sigma2</code>	initialized noise variance from ridge regression
<code>lambda</code>	Hyperparameter of lasso term in FSSEM
<code>rho</code>	Hyperparameter of fused-lasso term in FSSEM
<code>Wl</code>	weight matrices for adaptive lasso terms
<code>Wf</code>	weight matrix for adaptive fused lasso term
<code>p</code>	number of genes
<code>maxit</code>	maximum iteration number. Default 100
<code>inert</code>	inertial function for iPALM. Default as $k-1/k+2$
<code>threshold</code>	convergence threshold. Default $1e-6$
<code>verbose</code>	Default TRUE
<code>sparse</code>	Sparse Matrix or not
<code>trans</code>	<code>Fs</code> matrix is transposed to $k \times p$ or not. If <code>Fs</code> from ridge regression, <code>trans = TRUE</code> , else, <code>trans = FALSE</code>
<code>B2norm</code>	<code>B2norm</code> matrices generated from ridge regression. Default NULL.
<code>strict</code>	Converge strictly or not. Default False

### Value

`fit` List of FSSEM model

**Bs** coefficient matrices of gene regulatory networks

**Fs** coefficient matrices of eQTL-gene effect

**mu** Bias vector

**sigma2** estimate of covariance in SEM

**Examples**

```

seed = 1234
N = 100                                # sample size
Ng = 5                                 # gene number
Nk = 5 * 3                             # eQTL number
Ns = 1                                 # sparse ratio
sigma2 = 0.01                          # sigma2
set.seed(seed)
library(fssemR)
data = randomFSSEMdata(n = N, p = Ng, k = Nk, sparse = Ns, df = 0.3, sigma2 = sigma2,
                      u = 5, type = "DG", nhub = 1, dag = TRUE)
## If we assume that different condition has different genetics perturbations (eQTLs)
data$Data$X = list(data$Data$X, data$Data$X)
## gamma = cv.multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, ngamma = 20, nfold = 5,
##                             N, Ng, Nk)
gamma = 0.6784248    ## optimal gamma computed by cv.multiRegression
fit = multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, gamma, N, Ng, Nk,
                     trans = FALSE)

Xs = data$Data$X
Ys = data$Data$Y
Sk = data$Data$Sk

cvfitc <- cv.multiFSSEMiPALM2(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
                             sigma2 = fit$sigma2, nlambda = 5, nrho = 5,
                             nfold = 5, p = Ng, q = Nk, wt = TRUE)

fitc0 <- multiFSSEMiPALM2(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
                         sigma2 = fit$sigma2, lambda = cvfitc$lambda, rho = cvfitc$rho,
                         Wl = inverseB(fit$Bs), Wf = flinvB(fit$Bs),
                         p = Ng, maxit = 100, threshold = 1e-5, sparse = TRUE,
                         verbose = TRUE, trans = TRUE, strict = TRUE)

(TPR(fitc0$Bs[[1]], data$Vars$B[[1]]) + TPR(fitc0$Bs[[2]], data$Vars$B[[2]]) ) / 2
(FDR(fitc0$Bs[[1]], data$Vars$B[[1]]) + FDR(fitc0$Bs[[2]], data$Vars$B[[2]]) ) / 2
TPR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])
FDR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])

```

---

multiNFSSEMiPALM2

*multiNFSSEMiPALM2*


---

**Description**

Implementing NFSSEM algorithm for network inference. If Xs is identify for different conditions, multiNFSSEMiPALM will be use, otherwise, please use multiNFSSEMiPALM2 for general cases

**Usage**

```
multiNFSSEMiPALM2(
```

```

Xs,
Ys,
Bs,
Fs,
Sk,
sigma2,
lambda,
rho,
Wl,
Wf,
p,
maxit = 100,
inert = inert_opt("linear"),
threshold = 1e-06,
verbose = TRUE,
sparse = TRUE,
trans = FALSE,
B2norm = NULL,
strict = FALSE
)

```

### Arguments

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance from ridge regression
lambda	Hyperparameter of lasso term in NFSSEM
rho	Hyperparameter of fused-lasso term in NFSSEM
Wl	weight matrices for adaptive lasso terms
Wf	weight matrix for columnwise l2 norm adaptive group lasso
p	number of genes
maxit	maximum iteration number. Default 100
inert	inertial function for iPALM. Default as $k-1/k+2$
threshold	convergence threshold. Default $1e-6$
verbose	Default TRUE
sparse	Sparse Matrix or not
trans	Fs matrix is transposed to $k \times p$ or not. If Fs from ridge regression, trans = TRUE, else, trans = FALSE
B2norm	B2norm matrices generated from ridge regression. Default NULL.
strict	Converge strictly or not. Default False

**Value**

fit List of NFSSEM model

**Bs** coefficient matrices of gene regulatory networks

**Fs** coefficient matrices of eQTL-gene effect

**mu** Bias vector

**sigma2** estimate of covariance in SEM

---

multiRegression	<i>multiRegression</i>
-----------------	------------------------

---

**Description**

Ridge regression on multiple conditions, initialization of FSSEM algorithm

**Usage**

```
multiRegression(Xs, Ys, Sk, gamma, n, p, k, trans = FALSE)
```

**Arguments**

Xs	eQTL matrices. eQTL matrix can be matrix/list of multiple conditions
Ys	Gene expression matrices
Sk	eQTL index of genes
gamma	Hyperparameter for ridge regression
n	number of observations
p	number of genes
k	number of eQTLs
trans	if rows for sample, trans = TRUE, otherwise, trans = FALSE. Default FALSE

**Value**

fit List of SEM model

**Bs** coefficient matrices of gene regulatory networks

**fs** eQTL's coefficients w.r.t each gene

**Fs** coefficient matrices of eQTL-gene effect

**mu** Bias vector

**sigma2** estimate of covariance in SEM

**Examples**

```

seed = 1234
N = 100                                # sample size
Ng = 5                                 # gene number
Nk = 5 * 3                             # eQTL number
Ns = 1                                 # sparse ratio
sigma2 = 0.01                          # sigma2
set.seed(seed)
data = randomFSSEMdata(n = N, p = Ng, k = Nk, sparse = Ns, df = 0.3, sigma2 = sigma2,
                      u = 5, type = "DG", nhub = 1, dag = TRUE)
## If we assume that different condition has different genetics perturbations (eQTLs)
## data$Data$X = list(data$Data$X, data$Data$X)
gamma = cv.multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, ngamma = 20, nfold = 5,
                          N, Ng, Nk)
fit = multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, gamma, N, Ng, Nk,
                     trans = FALSE)

```

---

obj.multiRegression    *obj.multiRegression*

---

**Description**

obj.multiRegression

**Usage**

obj.multiRegression(Xs, Ys, fit, trans = F)

**Arguments**

Xs	eQTL matrices
Ys	gene expression matrices
fit	regression fit result object
trans	if rows for sample, trans = TRUE, otherwise, trans = FALSE. Default FALSE

**Value**

error squared norm of  $\|(I-B)Y - FX\|_2^2$

---

`opt.multiFSSEMiPALM`    *opt.multiFSSEMiPALM*

---

### Description

optimize multiFSSEMiPALM's parameters by minimize BIC, when feature size is large (> 300), BIC methods will be much faster than Cross-validation

### Usage

```
opt.multiFSSEMiPALM(  
  Xs,  
  Ys,  
  Bs,  
  Fs,  
  Sk,  
  sigma2,  
  nlambda = 20,  
  nrho = 20,  
  p,  
  q,  
  wt = TRUE  
)
```

### Arguments

<code>Xs</code>	eQTL matrices
<code>Ys</code>	Gene expression matrices
<code>Bs</code>	initialized GRN-matrices
<code>Fs</code>	initialized eQTL effect matrices
<code>Sk</code>	eQTL index of genes
<code>sigma2</code>	initialized noise variance
<code>nlambda</code>	number of hyper-parameter of lasso term in CV
<code>nrho</code>	number of hyper-parameter of fused-lasso term in CV
<code>p</code>	number of genes
<code>q</code>	number of eQTLs
<code>wt</code>	use adaptive lasso or not. Default TRUE.

### Value

list of model selection result



**Examples**

```

seed = 1234
N = 100                                # sample size
Ng = 5                                 # gene number
Nk = 5 * 3                             # eQTL number
Ns = 1                                 # sparse ratio
sigma2 = 0.01                          # sigma2
set.seed(seed)
library(fssemR)
data = randomFSSEMData(n = N, p = Ng, k = Nk, sparse = Ns, df = 0.3, sigma2 = sigma2,
                      u = 5, type = "DG", nhub = 1, dag = TRUE)
## If we assume that different condition has different genetics perturbations (eQTLs)
## data$Data$X = list(data$Data$X, data$Data$X)
## gamma = cv.multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, ngamma = 20, nfold = 5,
##                             N, Ng, Nk)
gamma = 0.6784248    ## optimal gamma computed by cv.multiRegression
fit = multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, gamma, N, Ng, Nk,
                     trans = FALSE)

Xs = data$Data$X
Ys = data$Data$Y
Sk = data$Data$Sk

fitm <- opt.multiFSSEMiPALM(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
                          sigma2 = fit$sigma2, nlambdas = 10, nrho = 10,
                          p = Ng, q = Nk, wt = TRUE)

fitc0 <- fitm$fit

(TPR(fitc0$Bs[[1]], data$Vars$B[[1]]) + TPR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
(FDR(fitc0$Bs[[1]], data$Vars$B[[1]]) + FDR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
TPR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])
FDR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])

```

---

opt.multiFSSEMiPALM2    *opt.multiFSSEMiPALM2*

---

**Description**

optimize multiFSSEMiPALM's parameters by minimize BIC, when feature size is large (> 300), BIC methods will be much faster than Cross-validation

**Usage**

```

opt.multiFSSEMiPALM2(
  Xs,
  Ys,
  Bs,
  Fs,
  Sk,

```

```

    sigma2,
    nlambda = 20,
    nrho = 20,
    p,
    q,
    wt = TRUE
  )

```

### Arguments

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
nlambda	number of hyper-parameter of lasso term in CV
nrho	number of hyper-parameter of fused-lasso term in CV
p	number of genes
q	number of eQTLs
wt	use adaptive lasso or not. Default TRUE.

### Value

list of model selection result

### Examples

```

seed = 1234
N = 100 # sample size
Ng = 5 # gene number
Nk = 5 * 3 # eQTL number
Ns = 1 # sparse ratio
sigma2 = 0.01 # sigma2
set.seed(seed)
library(fssemR)
data = randomFSSEMDATA(n = N, p = Ng, k = Nk, sparse = Ns, df = 0.3, sigma2 = sigma2,
  u = 5, type = "DG", nhub = 1, dag = TRUE)
## If we assume that different condition has different genetics perturbations (eQTLs)
data$Data$X = list(data$Data$X, data$Data$X)
## gamma = cv.multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, ngamma = 20, nfold = 5,
## N, Ng, Nk)
gamma = 0.6784248 ## optimal gamma computed by cv.multiRegression
fit = multiRegression(data$Data$X, data$Data$Y, data$Data$Sk, gamma, N, Ng, Nk,
  trans = FALSE)

Xs = data$Data$X
Ys = data$Data$Y
Sk = data$Data$Sk

```

```

fitm <- opt.multiNFSSEMiPALM2(Xs = Xs, Ys = Ys, Bs = fit$Bs, Fs = fit$Fs, Sk = Sk,
                             sigma2 = fit$sigma2, nlambda = 10, nrho = 10,
                             p = Ng, q = Nk, wt = TRUE)

fitc0 <- fitm$fit

(TPR(fitc0$Bs[[1]], data$Vars$B[[1]]) + TPR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
(FDR(fitc0$Bs[[1]], data$Vars$B[[1]]) + FDR(fitc0$Bs[[2]], data$Vars$B[[2]])) / 2
TPR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])
FDR(fitc0$Bs[[1]] - fitc0$Bs[[2]], data$Vars$B[[1]] - data$Vars$B[[2]])

```

---

opt.multiNFSSEMiPALM2 *opt.multiNFSSEMiPALM2*

---

### Description

optimize multiNFSSEMiPALM's parameters by minimize BIC, when feature size is large (> 300), BIC methods will be much faster than Cross-validation

### Usage

```

opt.multiNFSSEMiPALM2(
  Xs,
  Ys,
  Bs,
  Fs,
  Sk,
  sigma2,
  nlambda = 20,
  nrho = 20,
  p,
  q,
  wt = TRUE
)

```

### Arguments

Xs	eQTL matrices
Ys	Gene expression matrices
Bs	initialized GRN-matrices
Fs	initialized eQTL effect matrices
Sk	eQTL index of genes
sigma2	initialized noise variance
nlambda	number of hyper-parameter of lasso term in CV
nrho	number of hyper-parameter of fused-lasso term in CV

p                    number of genes  
 q                    number of eQTLs  
 wt                   use adaptive lasso or not. Default TRUE.

**Value**

list of model selection result

---

pninvB	<i>pninvB</i>
--------	---------------

---

**Description**

inversed column l2 norm for perturbed group lasso penalty

**Usage**

pninvB(Bs)

**Arguments**

Bs                    list of network matrices

**Value**

inversed l2 norm of B2 - B1

---

pnoneB	<i>pnoneB</i>
--------	---------------

---

**Description**

if you do not want adaptive group lasso penalty, pnoneB replace pninvB

**Usage**

pnoneB(Bs)

**Arguments**

Bs                    list of network matrices

**Value**

inversed l2 norm of B2 - B1 with all entries is 1

---

*proc.centerFSSEM*      *proc.centerFSSEM*

---

**Description**

*proc.centerFSSEM*

**Usage**

*proc.centerFSSEM*(Xs, Ys)

**Arguments**

Xs                    eQTL matrices  
Ys                    list of gene expression matrices

**Value**

centered Xs and Ys and mean vectors

---

*proc.centerFSSEM2*      *proc.centerFSSEM2*

---

**Description**

*proc.centerFSSEM2*

**Usage**

*proc.centerFSSEM2*(Xs, Ys)

**Arguments**

Xs                    list of eQTL matrices  
Ys                    list of gene expression matrices

**Value**

centered Xs and Ys and mean vectors

---

randomFSSEMdata	<i>randomFSSEMdata</i>
-----------------	------------------------

---

**Description**

randomFSSEMdata

**Usage**

```
randomFSSEMdata(
  n,
  p,
  k,
  sparse = 0.1,
  df = 0.2,
  sigma2 = 0.01,
  u = 5,
  type = c("DG", "ER"),
  dag = TRUE,
  coef = c(0.2, 0.4),
  nhub = 2
)
```

**Arguments**

n	number of observations
p	number of genes
k	number of eQTLs
sparse	ratio of edges / gene_number
df	ratio of differential edges among two network
sigma2	noise variance of error
u	variance of bias in SEM model.
type	type of generated network, can be selected as DG, ER, Scale-free network
dag	network is directed-acyclic or not. Default TRUE
coef	Range of absolute value of coefficients in simulated network matrices. Default (0.2, 0.4), or (0.5, 1)
nhub	If you select to generate ER network, nhub is the number of pre-defined hub node number. Default 2

**Value**

list of generated data

**Data** List of observed, Xs, Ys, Sk

**Vars** List of model, Bs, Fs, mu, n, p, k

---

randomFSSEMdata2	randomFSSEMdata2
------------------	------------------

---

**Description**

randomFSSEMdata2

**Usage**

```
randomFSSEMdata2(
  n,
  p,
  k,
  sparse = 0.1,
  df = 0.2,
  sigma2 = 0.01,
  u = 5,
  type = c("DG", "ER"),
  dag = TRUE,
  coef = c(0.2, 0.4),
  nhub = 2
)
```

**Arguments**

n	number of observations. Vector for unbalance observations
p	number of genes
k	number of eQTLs
sparse	ratio of edges / gene_number
df	ratio of differential edges among two network
sigma2	noise variance of error
u	variance of bias in SEM model.
type	type of generated network, can be selected as DG, ER, Scale-free network
dag	network is directed-acyclic or not. Default TRUE
coef	Range of absolute value of coefficients in simulated network matrices. Default (0.2, 0.4), or (0.5, 1)
nhub	If you select to generate ER network, nhub is the number of pre-defined hub node number. Default 2

**Value**

list of generated data

**Data** List of observed, Xs, Ys, Sk

**Vars** List of model, Bs, Fs, mu, n, p, k

---

randomFSSEMdata4Cor     *randomFSSEMdata4Cor*

---

## Description

randomFSSEMdata4Cor

## Usage

```
randomFSSEMdata4Cor(
  n,
  p,
  k,
  sparse = 0.1,
  df = 0.2,
  sigma2 = 0.01,
  u = 5,
  type = c("DG", "ER"),
  dag = TRUE,
  coef = c(0.2, 0.4),
  nhub = 2,
  r = 0.5
)
```

## Arguments

n	number of observations. Vector for unbalance observations
p	number of genes
k	number of eQTLs
sparse	ratio of edges / gene_number
df	ratio of differential edges among two network
sigma2	noise variance of error
u	variance of bias in SEM model.
type	type of generated network, can be selected as DG, ER, Scale-free network
dag	network is directed-acyclic or not. Default TRUE
coef	Range of absolute value of coefficients in simulated network matrices. Default (0.2, 0.4), or (0.5, 1)
nhub	If you select to generate ER network, nhub is the number of pre-defined hub node number. Default 2
r	correlation between different observations



**Value**

list of generated data

**Data** List of observed, Xs, Ys, Sk

**Vars** List of model, Bs, Fs, mu, n, p, k

---

TPR

*TPR*

---

**Description**

Power of detection for network prediction

**Usage**

TPR(X, B, PREC = 0)

**Arguments**

X	list of predicted network matrices
B	list of true network matrices
PREC	precision threshold for FDR test. Default 0.

---

transx

*transx*

---

**Description**

transx

**Usage**

transx(data)

**Arguments**

data	Collecting data structure generated by randomFSSEMdata function
------	---

**Value**

transformed list of eQTL matrices

# Index

cv.multiFSSEMiPALM, [2](#)  
cv.multiFSSEMiPALM2, [4](#)  
cv.multiNFSSEMiPALM2, [5](#)  
cv.multiRegression, [6](#)  
cwiseGradient4FSSEM, [6](#)

FDR, [7](#)  
flinvB, [7](#)  
floneB, [8](#)  
fssemR, [8](#)

implFSSEM, [9](#)  
initLambdaiPALM, [10](#)  
initLambdaiPALM2, [10](#)  
initLambdaiPALM3, [11](#)  
initRhoiPALM, [12](#)  
initRhoiPALM2, [12](#)  
initRhoiPALM3, [13](#)  
inverseB, [14](#)  
invoneB, [14](#)

logLikFSSEM, [15](#)  
logLikNFSSEM, [15](#)

multiFSSEMiPALM, [16](#)  
multiFSSEMiPALM2, [18](#)  
multiNFSSEMiPALM2, [20](#)  
multiRegression, [22](#)

obj.multiRegression, [23](#)  
opt.multiFSSEMiPALM, [24](#)  
opt.multiFSSEMiPALM2, [25](#)  
opt.multiNFSSEMiPALM2, [27](#)

package-fssemR (fssemR), [8](#)  
pninvB, [28](#)  
pnoneB, [28](#)  
proc.centerFSSEM, [29](#)  
proc.centerFSSEM2, [29](#)

randomFSSEMdata, [30](#)  
randomFSSEMdata2, [31](#)  
randomFSSEMdata4Cor, [32](#)

TPR, [33](#)  
transx, [33](#)