

# Package ‘remMap’

April 19, 2009

**Version** 0.1-0

**Date** 2008-12-3

**Title** Regularized Multivariate Regression for Identifying Master Predictors

**Author** Jie Peng <jie@wald.ucdavis.edu>, Pei Wang <pwang@fhcrc.org>, Ji Zhu <jizhu@umich.edu>.

**Maintainer** Pei Wang <pwang@fhcrc.org>

**Description** remMap is developed for fitting multivariate response regression models under the high-dimension-low-sample-size setting

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2009-01-17 10:13:50

## R topics documented:

remMap . . . . .	1
remMap.BIC . . . . .	4
remMap.CV . . . . .	7
remMap.df . . . . .	11

<b>Index</b>	<b>14</b>
--------------	-----------

---

remMap	<i>A function to fit REgularized Multivariate regression model using the MAP penalty (remMap).</i>
--------	--

---

### Description

A function to fit regularized multivariate regression model using the MAP penalty.

### Usage

```
remMap(X.m, Y.m, lamL1, lamL2, phi0=NULL, C.m=NULL)
```

### Arguments

X.m	numeric matrix (n by p): columns correspond to predictor variables and rows correspond to samples. Missing values are not allowed.
Y.m	numeric matrix (n by q): columns correspond to response variables and rows correspond to samples. Missing values are not allowed.
lamL1	numeric value: $l_1$ norm penalty parameter.
lamL2	numeric value: $l_2$ norm penalty parameter.
phi0	numeric matrix (p by q): an initial estimate of the coefficient matrix of the multivariate regression model; default(=NULL): univariate estimates are used as initial estimates.
C.m	numeric matrix (p by q): $C_m[i, j] = 0$ means the corresponding coefficient $\beta_{i,j}$ is set to be zero in the model; $C_m[i, j] = 1$ means the corresponding $\beta_{i,j}$ is included in the MAP penalty; $C_m[i, j] = 2$ means the corresponding $\beta_{i,j}$ is not included in the MAP penalty; default(=NULL): $C_m[i, j]$ are all set to be 1.

### Details

remMap uses a computationally efficient approach for performing multivariate regression under the high-dimension-low-sample-size setting (Peng and et. al., 2008).

### Value

A list with two components

phi	the estimated coefficient matrix (p by q) of the regularized multivariate regression model.
rss.v	a vector of length q recording the RSS values of the q regressions.

### Author(s)

Jie Peng, Pei Wang, Ji Zhu

## References

J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, P. Wang, Regularized Multivariate Regression for Identifying Master Predictors with Application to Integrative Genomics Study of Breast Cancer. (<http://arxiv.org/abs/0812.3671>)

## Examples

```
#####
##### Generate an example data set
#####
n=100
p=300
q=300
set.seed(1)

### generate X matrix
rho=0.5; Sig<-matrix(0,p,p)
for(i in 2:p){ for(j in 1: (i-1)){
  Sig[i,j]<-rho^(abs(i-j))
  Sig[j,i]<-Sig[i,j]
}}
diag(Sig)<-1
R<-chol(Sig)
X.m<-matrix(rnorm(n*p),n,p)
X.m<-X.m%*%R

### generate coefficient
coef.m<-matrix(0,p,q)
hub.n=20
hub.index=sample(1:p, hub.n)
for(i in 1:q){
  cur=sample(1:3,1)
  temp=sample(hub.index, cur)
  coef.m[temp,i]<-runif(length(temp), min=2, max=3)
}

### generate responses
E.m<-matrix(rnorm(n*q),n,q)
Y.m<-X.m%*%coef.m+E.m

#####
##### perform analysis
#####

#####
## 1. ## fit model for one pair of (lamL1, lamL2)
#####

try1=remMap(X.m, Y.m, lamL1=100, lamL2=50, phi0=NULL, C.m=NULL)

#####
```

```

## 2. ## Select tuning parameters with BIC:
###  ## computationally easy; but the BIC procedure assumes orthogonality of the design matrix
###  ## thus it tends to select too small models when the actual design matrix (X.m) is far
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
df.m=remMap.df(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
###  The estimated degrees of freedom can be used to select the ranges of tuning parameters.

try2=remMap.BIC(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
pick=which.min(as.vector(t(try2$BIC)))
result=try2$phi[[pick]]
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives

print(paste("lamL1=", round(result$lam1,3), "; lamL2=", round(result$lam2,3), sep="")) ##BIC
print(paste("FP=", FP, "; FN=", FN, sep=""))

#####
## 3. ## Select tuning parameters with v-fold cross-validation;
###  ## computationally demanding;
###  ## but cross-validation assumes less assumptions than BIC and thus is recommended unless
###  ## also cv based on unshrunk estimator (ols.cv) is recommended over cv based on shrunk
###  ## the latter tends to select too large models.
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
try3=remMap.CV(X=X.m, Y=Y.m, lamL1.v, lamL2.v, C.m=NULL, fold=10, seed=1)

##### use CV based on unshrunk estimator (ols.cv)
pick=which.min(as.vector(try3$ols.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try3$l.index[2,pick]
result=remMap(X.m, Y.m, lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL) ##fit the model
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives
print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (unshrunk)
print(paste("FP=", FP, "; FN=", FN, sep=""))

##### use CV based on shrunk estimator (rss.cv); it tends to select very large models
pick=which.min(as.vector(try3$rss.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try3$l.index[2,pick]
result=remMap(X.m, Y.m, lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL)
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives
print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (shrunk)
print(paste("FP=", FP, "; FN=", FN, sep=""))

```

---

remMap.BIC	<i>Fit remMap models for a series of tuning parameters and return the corresponding BIC scores.</i>
------------	---

---

### Description

Fit remMap models for a series of tuning parameters and return the corresponding BIC scores. It is computationally easy. But the BIC procedure assumes orthogonality of the design matrix to estimate the degrees of freedom. Thus it tends to select too small models when the actual design matrix (X.m) is far from orthogonal. In that case, cross validation is recommended (see help(remMap.CV) for more details)

### Usage

```
remMap.BIC(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
```

### Arguments

X.m	numeric matrix (n by p): columns correspond to predictor variables and rows correspond to samples. Missing values are not allowed.
Y.m	numeric matrix (n by q): columns correspond to response variables and rows correspond to samples. Missing values are not allowed.
lamL1.v	numeric vector: a set of $l_1$ norm penalty parameters.
lamL2.v	numeric vector: a set of $l_2$ norm penalty parameters.
C.m	numeric matrix (p by q). $C_m[i, j] = 0$ means the corresponding coefficient beta[i,j] is set to be zero in the model; $C_m[i, j] = 1$ means the corresponding beta[i,j] is included in the MAP penalty; $C_m[i, j] = 2$ means the corresponding beta[i,j] is not included in the MAP penalty; default(=NULL): $C_m[i, j]$ are all set to be 1.

### Details

remMap.BIC is used to perform two-dimensional grid search of the tuning parameters (lamL1.v, lamL2.v) based on the BIC scores. (Peng and et.al., 2008).

### Value

A list with two components	
BIC	a numeric matrix recording the BIC scores of the remMap models. Each element corresponds to one pair of (lamL1, lamL2).
phi	a list recording the fitted remMap coefficients. Each component corresponds to one pair of (lamL1, lamL2) in the grid search.

### Author(s)

Jie Peng, Pei Wang, Ji Zhu

## References

J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, P. Wang, Regularized Multivariate Regression for Identifying Master Predictors with Application to Integrative Genomics Study of Breast Cancer. (<http://arxiv.org/abs/0812.3671>)

## Examples

```
#####
##### Generate an example data set
#####
n=100
p=300
q=300
set.seed(1)

### generate X matrix
rho=0.5; Sig<-matrix(0,p,p)
for(i in 2:p){ for(j in 1: (i-1)){
  Sig[i,j]<-rho^(abs(i-j))
  Sig[j,i]<-Sig[i,j]
}}
diag(Sig)<-1
R<-chol(Sig)
X.m<-matrix(rnorm(n*p),n,p)
X.m<-X.m%*%R

### generate coefficient
coef.m<-matrix(0,p,q)
hub.n=20
hub.index=sample(1:p, hub.n)
for(i in 1:q){
  cur=sample(1:3,1)
  temp=sample(hub.index, cur)
  coef.m[temp,i]<-runif(length(temp), min=2, max=3)
}

### generate responses
E.m<-matrix(rnorm(n*q),n,q)
Y.m<-X.m%*%coef.m+E.m

#####
##### perform analysis
#####

#####
## 1. ## fit model for one pair of (lamL1, lamL2)
#####

try1=remMap(X.m, Y.m, lamL1=100, lamL2=50, phi0=NULL, C.m=NULL)

#####
```

```

## 2. ## Select tuning parameters with BIC:
### ## computationally easy; but the BIC procedure assumes orthogonality of the design matrix
### ## thus it tends to select too small models when the actual design matrix (X.m) is far
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
df.m=remMap.df(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
### The estimated degree freedom can be used to select the ranges of tuning parameters.

try2=remMap.BIC(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
pick=which.min(as.vector(t(try2$BIC)))
result=try2$phi[[pick]]
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives

print(paste("lamL1=", round(result$lam1,3), "; lamL2=", round(result$lam2,3), sep="")) ##BIC
print(paste("FP=", FP, "; FN=", FN, sep=""))

#####
## 3. ## Select tuning parameters with v-fold cross-validation;
### ## computationally demanding;
### ## but cross-validation assumes less assumptions than BIC and thus is recommended unless
### ## also cv based on unshrunk estimator (ols.cv) is recommended over cv based on shrinkage
### ## the latter tends to select too large models.
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
try3=remMap.CV(X=X.m, Y=Y.m, lamL1.v, lamL2.v, C.m=NULL, fold=10, seed=1)

##### use CV based on unshrunk estimator (ols.cv)
pick=which.min(as.vector(try3$ols.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try3$l.index[2,pick]
result=remMap(X.m, Y.m, lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL) ##fit the model
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives
print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (unshrunk)
print(paste("FP=", FP, "; FN=", FN, sep=""))

##### use CV based on shrunk estimator (rss.cv); it tends to select very large models
pick=which.min(as.vector(try3$rss.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try3$l.index[2,pick]
result=remMap(X.m, Y.m, lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL)
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives
print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (shrunk)
print(paste("FP=", FP, "; FN=", FN, sep=""))

```

---

remMap.CV	<i>Fit remMap models for a series of tuning parameters and return the corresponding v-fold cross-validation scores.</i>
-----------	---

---

## Description

Fit remMap models for a series of tuning parameters and return the corresponding v-fold cross-validation scores. Two types of cross-validation scores are computed: cv based on unshrunk estimator (ols.cv); and cv based on shrunk estimator (rss.cv); ols.cv is recommended. rss.cv tends to select very large models and thus is not recommended in general (especially for very sparse models). v-fold CV is computationally demanding. But it assumes less assumptions than BIC and thus ols.cv is recommended over BIC unless computation is a concern.

## Usage

```
remMap.CV(X, Y, lamL1.v, lamL2.v, C.m=NULL, fold=10, seed=1)
```

## Arguments

X	numeric matrix (n by p): columns correspond to predictor variables and rows correspond to samples. Missing values are not allowed.
Y	numeric matrix (n by q): columns correspond to response variables and rows correspond to samples. Missing values are not allowed.
lamL1.v	numeric vector: a set of $l_1$ norm penalty parameters.
lamL2.v	numeric vector: a set of $l_2$ norm penalty parameters.
C.m	numeric matrix (p by q). $C_m[i, j] = 0$ means the corresponding coefficient beta[i,j] is set to be zero in the model; $C_m[i, j] = 1$ means the corresponding beta[i,j] is included in the MAP penalty; $C_m[i, j] = 2$ means the corresponding beta[i,j] is not included in the MAP penalty; default(=NULL): $C_m[i, j]$ are all set to be 1.
fold	positive integer. It specifies the cross validation fold number; default=10.
seed	numeric scalar. It specifies the seed of the random number generator in R for generating cross validation subsets; default=1.

## Details

remMap.CV is used to perform two-dimensional grid search of the tuning parameters (lamL1.v, lamL2.v) based on v-fold cross-validation scores. (Peng and et.al., 2008).

## Value

A list with four components

ols.cv	a numeric matrix recording the cross validation scores based on unshrunk OLS estimators for each pair of (lamL1, lamL2).
--------	--

<code>rss.cv</code>	a numeric matrix recording the cross validation scores based on shrunked remMap estimators for each pair of (lamL1, lamL2).
<code>phi.cv</code>	a list recording the fitted remMap coefficients on cross validation training subsets. Each component corresponds to one cv fold and it is again a list with each component corresponding to the estimated remMap coefficients for one pair of (lamL1, lamL2) on that training subset.
<code>l.index</code>	numeric matrix with two rows: each column is a pair of (lamL1, lamL2) and the kth column corresponds to the kth cv score in <code>as.vector(ols.cv)</code> and <code>as.vector(rss.cv)</code> .

**Author(s)**

Jie Peng, Pei Wang, Ji Zhu

**References**

J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, P. Wang, Regularized Multivariate Regression for Identifying Master Predictors with Application to Integrative Genomics Study of Breast Cancer. (<http://arxiv.org/abs/0812.3671>)

**Examples**

```
#####
##### Generate an example data set
#####
n=100
p=300
q=300
set.seed(1)

### generate X matrix
rho=0.5; Sig<-matrix(0,p,p)
for(i in 2:p){ for(j in 1:(i-1)){
  Sig[i,j]<-rho^(abs(i-j))
  Sig[j,i]<-Sig[i,j]
}}
diag(Sig)<-1
R<-chol(Sig)
X.m<-matrix(rnorm(n*p),n,p)
X.m<-X.m%*%R

### generate coefficient
coef.m<-matrix(0,p,q)
hub.n=20
hub.index=sample(1:p, hub.n)
for(i in 1:q){
  cur=sample(1:3,1)
  temp=sample(hub.index, cur)
  coef.m[temp,i]<-runif(length(temp), min=2, max=3)
}
```

```

### generate responses
E.m<-matrix(rnorm(n*q),n,q)
Y.m<-X.m%*%coef.m+E.m

#####
##### perform analysis
#####

#####
## 1. ## fit model for one pair of (lamL1, lamL2)
#####

try1=remMap(X.m, Y.m,lamL1=100, lamL2=50, phi0=NULL, C.m=NULL)

#####
## 2. ## Select tuning parameters with BIC:
### ## computationally easy; but the BIC procedure assumes orthogonality of the design matrix
### ## thus it tends to select too small models when the actual design matrix (X.m) is far from
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
df.m=remMap.df(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
### The estimated degree freedom can be used to select the ranges of tuning parameters.

try2=remMap.BIC(X.m, Y.m,lamL1.v, lamL2.v, C.m=NULL)
pick=which.min(as.vector(t(try2$BIC)))
result=try2$phi[[pick]]
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives

print(paste("lamL1=", round(result$lam1,3), "; lamL2=", round(result$lam2,3), sep="")) ##BIC
print(paste("FP=", FP, "; FN=", FN, sep=""))

#####
## 3. ## Select tuning parameters with v-fold cross-validation;
### ## computationally demanding;
### ## but cross-validation assumes less assumptions than BIC and thus is recommended unless
## ## also cv based on unshrunk estimator (ols.cv) is recommended over cv based on shrunken estimator
### ## the latter tends to select too large models.
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
try3=remMap.CV(X=X.m, Y=Y.m,lamL1.v, lamL2.v, C.m=NULL, fold=10, seed=1)

##### use CV based on unshrunk estimator (ols.cv)
pick=which.min(as.vector(try3$ols.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try3$l.index[2,pick]
result=remMap(X.m, Y.m,lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL) ##fit the model
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives

```

```

print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (u
print(paste("FP=", FP, "; FN=", FN, sep=""))

##### use CV based on shrunked estimator (rss.cv); it tends to select very large mod
pick=which.min(as.vector(try3$rss.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try3$l.index[2,pick]
result=remMap(X.m, Y.m, lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL)
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives
print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (s
print(paste("FP=", FP, "; FN=", FN, sep=""))

```

---

remMap.df	<i>A function to calculate the degrees of freedom of the remMap estimator.</i>
-----------	--

---

## Description

A function to calculate the degrees of freedom of the remMap estimator. It assumes orthogonality of the design matrix to estimate the degrees of freedom and tends to overestimate the actual degrees of freedom when the design matrix is far from orthogonal.

## Usage

```
remMap.df(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
```

## Arguments

X.m	numeric matrix (n by p): columns correspond to predictor variables and rows correspond to samples. Missing values are not allowed.
Y.m	numeric matrix (n by q): columns correspond to response variables and rows correspond to samples. Missing values are not allowed.
lamL1.v	numeric vector: a set of $l_1$ norm penalty parameters.
lamL2.v	numeric vector: a set of $l_2$ norm penalty parameters.
C.m	numeric matrix (p by q). $C_m[i, j] = 0$ means the corresponding coefficient beta[i,j] is set to be zero in the model; $C_m[i, j] = 1$ means the corresponding beta[i,j] is included in the MAP penalty; $C_m[i, j] = 2$ means the corresponding beta[i,j] is not included in the MAP penalty; default(=NULL): $C_m[i, j]$ are all set to be 1.

## Details

remMap.df calculates the estimated degrees of freedom of the remMap estimator for each pair of tuning parameters (lamL1, lamL2) (Peng and et.al., 2008).

**Value**

A numeric matrix. Each element corresponds to the estimated degrees of freedom of the resulting remMap estimator under a pair of lamL1 (rows) and lamL2 (columns).

**Author(s)**

Jie Peng, Pei Wang, Ji Zhu

**References**

J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, P. Wang, Regularized Multivariate Regression for Identifying Master Predictors with Application to Integrative Genomics Study of Breast Cancer. (<http://arxiv.org/abs/0812.3671>)

**Examples**

```
#####
##### Generate an example data set
#####
n=100
p=300
q=300
set.seed(1)

### generate X matrix
rho=0.5; Sig<-matrix(0,p,p)
for(i in 2:p){ for(j in 1: (i-1)){
  Sig[i,j]<-rho^(abs(i-j))
  Sig[j,i]<-Sig[i,j]
}}
diag(Sig)<-1
R<-chol(Sig)
X.m<-matrix(rnorm(n*p),n,p)
X.m<-X.m%%R

### generate coefficient
coef.m<-matrix(0,p,q)
hub.n=20
hub.index=sample(1:p, hub.n)
for(i in 1:q){
  cur=sample(1:3,1)
  temp=sample(hub.index, cur)
  coef.m[temp,i]<-runif(length(temp), min=2, max=3)
}

### generate responses
E.m<-matrix(rnorm(n*q),n,q)
Y.m<-X.m%%coef.m+E.m

#####
##### perform analysis
```

```
#####

#####
## 1. ## fit model for one pair of (lamL1, lamL2)
#####

try1=remMap(X.m, Y.m, lamL1=100, lamL2=50, phi0=NULL, C.m=NULL)

#####
## 2. ## Select tuning parameters with BIC:
### ## computationally easy; but the BIC procedure assumes orthogonality of the design matrix
### ## thus it tends to select too small models when the actual design matrix (X.m) is far
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
df.m=remMap.df(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
### The estimated degree freedom can be used to select the ranges of tuning parameters.

try2=remMap.BIC(X.m, Y.m, lamL1.v, lamL2.v, C.m=NULL)
pick=which.min(as.vector(t(try2$BIC)))
result=try2$phi[[pick]]
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives

print(paste("lamL1=", round(result$lam1,3), "; lamL2=", round(result$lam2,3), sep="")) ##BIC
print(paste("FP=", FP, "; FN=", FN, sep=""))

#####
## 3. ## Select tuning parameters with v-fold cross-validation;
### ## computationally demanding;
### ## but cross-validation assumes less assumptions than BIC and thus is recommended unless
## ## also cv based on unshrunk estimator (ols.cv) is recommended over cv based on shrunk
### ## the latter tends to select too large models.
#####

lamL1.v=exp(seq(log(51),log(150), length=5))
lamL2.v=seq(0,100, length=5)
try3=remMap.CV(X=X.m, Y=Y.m, lamL1.v, lamL2.v, C.m=NULL, fold=10, seed=1)

##### use CV based on unshrunk estimator (ols.cv)
pick=which.min(as.vector(try3$ols.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try3$l.index[2,pick]
result=remMap(X.m, Y.m, lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL) ##fit the model
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives
print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (unshrunk)
print(paste("FP=", FP, "; FN=", FN, sep=""))

##### use CV based on shrunk estimator (rss.cv); it tends to select very large models
pick=which.min(as.vector(try3$rss.cv))
lamL1.pick=try3$l.index[1,pick] ##find the optimal (LamL1,LamL2) based on the cv score
```

```
lamL2.pick=try3$1.index[2,pick]
result=remMap(X.m, Y.m, lamL1=lamL1.pick, lamL2=lamL2.pick, phi0=NULL, C.m=NULL)
FP=sum(result$phi!=0 & coef.m==0) ## number of false positives
FN=sum(result$phi==0 & coef.m!=0) ## number of false negatives
print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep="")) ##CV (s
print(paste("FP=", FP, "; FN=", FN, sep=""))
```

# Index

## \*Topic **methods**

- remMap, [1](#)
- remMap.BIC, [4](#)
- remMap.CV, [7](#)
- remMap.df, [11](#)

- remMap, [1](#)
- remMap.BIC, [4](#)
- remMap.CV, [7](#)
- remMap.df, [11](#)