

# Package ‘pgraph’

October 13, 2016

**Type** Package

**Title** Build Dependency Graphs using Projection

**Version** 0.8

**Date** 2016-09-25

**Author** Jianqing Fan, Yang Feng and Lucy Xia

**Maintainer** Yang Feng <yang.feng@columbia.edu>

**Imports** SAM, energy, glasso, glmnet, splines

**Description** Implements a general framework for creating dependency graphs using projection. Both lasso and sparse additive model projections are implemented. Both Pearson correlation and distance covariance are used to generate the graph.

**License** GPL-2

**LazyData** TRUE

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2016-10-13 20:37:09

## R topics documented:

greg . . . . .	2
pgraph . . . . .	3
projcore . . . . .	4
projcov . . . . .	5
roc . . . . .	6

<b>Index</b>	<b>7</b>
--------------	----------

---

 greg

*Regularized graphical model estimation*


---

### Description

greg calculate the regularized graphical model estimation using lasso, scad and adaptive lasso penalties. It report the results in the form of roc results for each method.

### Usage

```
greg(z, A, eps = 1e-15, rholist = NULL, gamma = 0.5, trace = FALSE)
```

### Arguments

z	n * p dimensional matrix
A	p * p true graph
eps	a tolerance level for thresholding
rholist	a sequence of penalty parameters
gamma	the adaptive lasso penalty parameter
trace	whether to trace to estimation process.

### Value

a list.

roc.lasso	roc results for lasso
roc.scad	roc results for scad
roc.lasso	roc results for adaptive lasso

### See Also

[pgraph](#), [roc](#), [projcov](#)

### Examples

```
set.seed(0)
p = 20;
n = 300;
tmp=runif(p-1,1,3)
s=c(0,cumsum(tmp));
s1=matrix(s,p,p)
cov.mat.true=exp(-abs(s1-t(s1)))
prec.mat.true=solve(cov.mat.true);
a=matrix(rnorm(p*n),n,p)
data.sa=a%%chol(cov.mat.true);
true.graph = outer(1:p,1:p,f<-function(x,y){(abs(x-y)==1)})
greg.fit = greg(data.sa, true.graph)
```

```

auc.lasso = sum(diff(greg.fit$roc.lasso[,1])*greg.fit$roc.lasso[-1,2])
auc.lasso = sum(diff(greg.fit$roc.lasso[,1])*greg.fit$roc.lasso[-1,2])
auc.scad = sum(diff(greg.fit$roc.scad[,1])*greg.fit$roc.scad[-1,2])
auc.lasso
auc.lasso
auc.scad

```

---

pgraph

*Calculate the Conditional Dependency Graph*


---

### Description

pgraph calculate the conditional dependency graph (with/without external factors) via projection using lasso or sparse additive model.

### Usage

```

pgraph(z, f = NULL, method = c("lasso", "sam"), cond = TRUE,
       randSeed = 0, trace = FALSE)

```

### Arguments

z	n * p dimensional matrix
f	n * q factor matrix. Default = 'NULL'.
method	projection method. Default = 'linear'.
cond	whether to create a conditional graph or unconditional graph. Default = TRUE. If cond = FALSE, f must be provided.
randSeed	the random seed for the program. Default = 0.
trace	whether to trace to estimation process.

### Value

a list to be used to calculate the ROC curve.

statmat.pearson	matrix with pearson correlation test
statmat.dcov	matrix with distance covariance test

### See Also

[greg](#), [roc](#), [projcov](#)

**Examples**

```

library(splines)
set.seed(0)
p = 5;
n = 100;
tmp=runif(p-1,1,3)
s=c(0,cumsum(tmp));
s1=matrix(s,p,p)
cov.mat.true=exp(-abs(s1-t(s1)))
prec.mat.true=solve(cov.mat.true);
a=matrix(rnorm(p*n),n,p)
data.sa=a%%chol(cov.mat.true);
true.graph = outer(1:p,1:p,f<-function(x,y){(abs(x-y)==1)})
methodlist = c("lasso","sam")
fit = vector(mode="list", length=2)
info = vector(mode="list", length=2)
auc = NULL
for(i in 1:2){
method = methodlist[i]
fit[[i]] = pgraph(data.sa, method = method)
info[[i]] = roc(fit[[i]]$statmat.pearson, true.graph)
auc[i] = sum(-diff(info[[i]][,1])*info[[i]][-1,2])
  cat(method, ': auc=', auc[i],'\n')
}

```

---

projcore

---

*Calculate the Projected matrix given factors*


---

**Description**

projcore calculate the projected matrix given factors.

**Usage**

```

projcore(x, b, method = c("lasso", "sam"), one.SE = TRUE, refit = TRUE,
  randSeed = 0)

```

**Arguments**

x	first vector
b	factor matrix
method	projection method. Default = 'linear'.
one.SE	whether to use the 1se rule for glmnet. Default = TRUE.
refit	whether to refit the selected model. Default = TRUE.
randSeed	the random seed for the program. Default = 0.

**Value**

eps                    the residual matrix after projection

**See Also**

[greg](#), [roc](#), [pgraph](#)

---

projcov

*Calculate the Projected Covariance of Two Vectors*

---

**Description**

projcov calculate the projected distance covariance of two vectors given common factors.

**Usage**

```
projcov(x, y, b, method = c("lasso", "sam"), one.SE = TRUE, refit = TRUE,
        randSeed = 0)
```

**Arguments**

x                    first vector  
y                    second vector  
b                    factor matrix  
method              projection method. Default = 'linear'.  
one.SE              whether to use the 1se rule for glmnet. Default = TRUE.  
refit                whether to refit the selected model. Default = TRUE.  
randSeed            the random seed for the program. Default = 0.

**Value**

a list.

test.pearson      pearson correlection test statistic  
test.dcov         distance covariance test statistic  
xeps                residual of projection of x on b  
yeps                residual of projection of y on b

**See Also**

[greg](#), [roc](#), [pgraph](#)

**Examples**

```
library(splines)
set.seed(0)
K = 3
n = 100
b = matrix(rnorm(K*n),n,K)
bx = 1:3
by = c(1,2,2)
x = b%%bx+rnorm(n)
y = b%%by+rnorm(n)
fit1 = projcov(x, y, b, method = "lasso")
fit2 = projcov(x, y, b, method = "sam")
```

---

roc

*Compute the Projected Graph*

---

**Description**

roc calculate the fpr and tpr for the roc curve

**Usage**

```
roc(a, a0)
```

**Arguments**

a	p * p estimated graph
a0	p * p true graph

**Value**

a list.	
tpr	tpr sequence
fpr	fpr sequence

# Index

greg, [2](#), [3](#), [5](#)

pgraph, [2](#), [3](#), [5](#)

projcore, [4](#)

projcov, [2](#), [3](#), [5](#)

roc, [2](#), [3](#), [5](#), [6](#)