Package 'seedCCA'

May 9, 2026

Type Package

Title Seeded Canonical Correlation Analysis

Version 3.1

Date 2022-06-09

Author Jae Keun Yoo, Bo-Young Kim

Maintainer Jae Keun Yoo <peter.yoo@ewha.ac.kr>

Description Functions for dimension reduction through the seeded canonical correlation analysis are provided. A classical canonical correlation analysis (CCA) is one of useful statistical methods in multivariate data analysis, but it is limited in use due to the matrix inversion for large p small n data. To overcome this, a seeded CCA has been proposed in Im, Gang and Yoo (2015) \doi{10.1002/cem.2691}. The seeded CCA is a two-step procedure. The sets of variables are initially reduced by successively projecting cov(X,Y) or cov(Y,X) onto cov(X) and cov(Y), respectively, without loss of information on canonical correlation analysis, following Cook, Li and Chiaromonte (2007) \doi{10.1093/biomet/asm038} and Lee and Yoo (2014) \doi{10.1111/anzs.12057}. Tl
ical correlation is finalized with the initially-reduced two sets of variables.

License GPL (>= 2.0)

Depends R(>= 2.10.0)

Imports CCA, corpcor, stats, graphics

Repository CRAN

LazyData yes

Encoding UTF-8

RoxygenNote 6.0.1

NeedsCompilation no

Date/Publication 2022-06-09 04:40:03 UTC

Contents

`coef.seedCCA` *Coefficients of ordinary and partial least squares through iterative projections*

Description

Returns coefficients of partial least squares through iterative projections. It works only for subclasses "seedols" and seedpls".

Usage

```
## S3 method for class 'seedCCA'
coef(object, u=NULL,...)
```

Arguments

`object`	The name of an object of class "seedCCA"
`u`	numeric, the number of projections. The default is NULL. This option is valid for PLS alone. The option returns the coefficient estimates for u projections. For example, if it is specified at k, then the coefficient estimates with k projections are returned.
`...`	arguments passed to the coef.method

Examples

```
########  data(cookie) ########
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

fit.ols1 <- seedCCA(X[,1:4], Y[,1], type="cca")
```

```
fit.pls1 <- seedCCA(X,Y[,1],type="pls")

coef(fit.ols1)
coef(fit.pls1)
coef(fit.pls1, u=4)
```

`cookie` *cookie dataset*

Description

This data set contains measurements from quantitative NIR spectroscopy. The example studied arises from an experiment done to test the feasibility of NIR spectroscopy to measure the composition of biscuit dough pieces (formed but unbaked biscuits). Two similar sample sets were made up, with the standard recipe varied to provide a large range for each of the four constituents under investigation: fat, sucrose, dry flour, and water. The calculated percentages of these four ingredients represent the 4 responses. There are 40 samples in the calibration or training set (with sample 23 being an outlier) and a further 32 samples in the separate prediction or validation set (with example 21 considered as an outlier).

An NIR reflectance spectrum is available for each dough piece. The spectral data consist of 700 points measured from 1100 to 2498 nanometers (nm) in steps of 2 nm.

Usage

```
data(cookie)
```

Format

A data frame of dimension 72 x 704. The first 700 columns correspond to the NIR reflectance spectrum, the last four columns correspond to the four constituents fat, sucrose, dry flour, and water. The first 40 rows correspond to the calibration data, the last 32 rows correspond to the prediction data.

References

Please cite the following papers if you use this data set.

P.J. Brown, T. Fearn, and M. Vannucci (2001) Bayesian Wavelet Regression on Curves with Applications to a Spectroscopic Calibration Problem. Journal of the American Statistical Association, 96, pp. 398-408.

B.G. Osborne, T. Fearn, A.R. Miller, and S. Douglas (1984) Application of Near-Infrared Reflectance Spectroscopy to Compositional Analysis of Biscuits and Biscuit Dough. Journal of the Science of Food and Agriculture, 35, pp. 99 - 105.

Examples

```
data(cookie) # load data
X<-as.matrix(cookie[,1:700]) # extract NIR spectra
Y<-as.matrix(cookie[,701:704]) # extract constituents
Xtrain<-X[1:40,] # extract training data
Ytrain<-Y[1:40,] # extract training data
Xtest<-X[41:72,] # extract test data
Ytest<-Y[41:72,] # extract test data
```

---

covplot *scree-ploting cov(X, Y)*

Description

Returns a scree-plot of the eigenvalues of cov(first.set, second.set) to select its first d largest eigenvectors.

Usage

```
covplot(X, Y, mind=NULL)
```

Arguments

X	numeric matrix (n * p), X
Y	numeric matrix (n * r), Y
mind	numeric, the number of the eigenvalues to show their cumulative percentages. The default is NULL, and then it is equal to min(p,r)

Value

eigenvalues	the ordiered eigenvalues of cov(X,Y)
cum.percent	the cumulative percentages of the eigenvalues
num.evecs	a vector of the numbers of the eigenvectors which forces the cumulative percentages bigger than 0.6, 0.7, 0.8, 0.9

Examples

```
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])
covplot(X, Y)
covplot(X, Y, mind=4)
```

finalCCA *finalized CCA in seeded CCA*

Description

Returns the results of the finalized step in seeded CCA.

Usage

```
finalCCA(X, Y)
```

Arguments

X	numeric matrix (n * d), the initially-CCAed first set of variables
Y	numeric matrix (n * d), the initially-CCAed second set of variables

Value

cor	canonical correlations in finalized step
xcoef	the estimated canonical coefficient matrix of the initially-CCAed first set of variables
ycoef	the estimated canonical coefficient matrix of the initially-CCAed second set of variables
Xscores	the finalized canonical variates of the first set of variables
Yscores	the finalized canonical variates of the second set of variables

Examples

```
########  data(cookie) ########
data(cookie)
myseq <- seq(141, 651, by=2)
X <- as.matrix(cookie[-c(23,61), myseq])
Y <- as.matrix(cookie[-c(23,61), 701:704])
min.pr <- min( dim(X)[2], dim(Y)[2])
MX0 <- iniCCA(X, Y, u=4, num.d=min.pr)
ini.X <- X %*% MX0
finalCCA(ini.X, Y)

########  data(nutrimouse) ########
data(nutrimouse)
Y<-as.matrix(nutrimouse$lipid)
X<-as.matrix(nutrimouse$gene)
MX0 <- iniCCA(X, Y, u=4, num.d=4)
MY0 <- iniCCA(Y, X, u=5, num.d=4)
ini.X <- X %*% MX0
ini.Y <- Y %*% MY0
finalCCA(ini.X, ini.Y)
```

fitted.seedCCA *Fitted values of ordinary and partial least squares*

Description

Returns fitted values of ordinary and partial least squares through iterative projections. It works only for subclasses "seedols" and "seedpls".

Usage

```
## S3 method for class 'seedCCA'
fitted(object, u=NULL,...)
```

Arguments

object	The name of an object of class "seedCCA"
u	numeric, the number of projections. The default is NULL. This option is valid for PLS alone. The option returns the fitted values for u projections. For example, if it is specified at k, then the fitted values with k projections are returned.
...	arguments passed to the fitted.method

Examples

```
########  data(cookie) ########
########  data(cookie) ########
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

fit.ols1 <- seedCCA(X[,1:4], Y[,1], type="cca")
fit.pls1 <- seedCCA(X, Y[,1], type="pls")
fit.pls2 <- seedCCA(X, Y[,1], type="pls", scale=TRUE)

fitted(fit.ols1)
fitted(fit.pls1)
fitted(fit.pls1, u=4)
fitted(fit.pls2, u=4)
```

iniCCA *Initialized CCA in seeded CCA*

Description

Returns the canonical coefficient matrices from the initialized step in seeded CCA. The initialzied CCA is done only for the first set in its first argument. The num.d must be less than or equal to the dimension of the second set.

Usage

```
iniCCA(X, Y, u, num.d)
```

Arguments

X	numeric matrix (n * p), the first set of variables: this set of variables alone is reduced.
Y	numeric matrix (n * r), the second set of variables
u	numeric, the terminiating index of the projection
num.d	numeric, the first "num.d" eigenvectors of cov(X,Y) to replace cov(X,Y), if min(p,r) relatively bigger than n. The num.d must be less than or equal to min(p,r).

Value

B	the initialized CCAed coefficient matrix projected by the value of u

Examples

```
########  data(cookie) ########
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])
min.pr <- min( dim(X)[2], dim(Y)[2])
MX0 <- iniCCA(X, Y, u=4, num.d=min.pr)
ini.X <- X%*%MX0

########  data(nutrimouse) ########
data(nutrimouse)
Y<-as.matrix(nutrimouse$lipid)
X<-as.matrix(nutrimouse$gene)
MX0 <- iniCCA(X, Y, u=4, num.d=4)
MY0 <- iniCCA(Y, X, u=5, num.d=4)
ini.X <- X %*% MX0
ini.Y <- Y %*% MY0
```

---

nutrimouse *Nutrimouse dataset*

Description

The nutrimouse dataset comes from a nutrition study in the mouse. It was provided by Pascal Martin from the Toxicology and Pharmacology Laboratory (French National Institute for Agronomic Research).

Usage

```
data(nutrimouse)
```

Format

gene: data frame (40 * 120) with numerical variables

lipid: data frame (40 * 21) with numerical variables

diet: factor vector (40)

genotype: factor vector (40)

Details

Two sets of variables were measured on 40 mice:

expressions of 120 genes potentially involved in nutritional problems.

concentrations of 21 hepatic fatty acids: The 40 mice were distributed in a 2-factors experimental design (4 replicates).

Genotype (2-levels factor): wild-type and PPARalpha -/-

Diet (5-levels factor): Oils used for experimental diets preparation were corn and colza oils (50/50) for a reference diet (REF), hydrogenated coconut oil for a saturated fatty acid diet (COC), sunflower oil for an Omega6 fatty acid-rich diet (SUN), linseed oil for an Omega3-rich diet (LIN) and corn/colza/enriched fish oils for the FISH diet (43/43/14).

References

P. Martin, H. Guillou, F. Lasserre, S. D??jean, A. Lan, J-M. Pascussi, M. San Cristobal, P. Legrand, P. Besse, T. Pineau (2007) Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. Hepatology, 45, 767???777

Examples

```
data(nutrimouse)
boxplot(nutrimouse$lipid)
```

---

plot.seedCCA *Plotting class "seedCCA" depending on the value of* type

---

Description

The function is for plotting class "seedCCA". Depending on subclass defined by the value of type, its resulting plot is different.

Usage

```
## S3 method for class 'seedCCA'
plot(x, ref=90, eps=0.01, ...)
```

Arguments

x	The name of an object of class "seedCCA"
ref	numeric, the value for reference line. It must be chosen between 0 and 100. It works only for subclass "finalCCA".
eps	numeric, a value to terminate projections. It must be chosen between 0 and 1. The default is equal to 0.01. It works only for subclass "seedpls".
...	arguments passed to the plot.method

Details

subclass "finalCCA": the function makes a plot for percents of cumulative canonical correlations.

subclass "seedpls": the function returns a proper number of projections and plot of the projection increment against the number of projections. A proper number of projections is indicated with a blue-color vertical bar in the plot. Only for subclass "seedpls", the output is retured. See Value part.

subclass "seedols": No plotting

subclass "selectu": the function makes a plot for increment of iterative projections by the output of subclass "selectu".

Value

proper.u	proper value of the number of projections
nFu	incrments (n*Fu) of the iterative projection.
u	the maximum number of projections from "seedpls" object
eps	a value for terminating the projection. The default value is equal to 0.01.

Examples

```
########  data(cookie) ########
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

fit.cca <- seedCCA(X[,1:4],Y[,1:4],type="cca")
fit.seed1 <- seedCCA(X,Y, type="seed1")
fit.pls1 <- seedCCA(X,Y[,1],type="pls")
fit.selu <- selectu(X,Y, type="seed2")

plot(fit.cca)
plot(fit.seed1, ref=95)
plot(fit.pls1)
plot(fit.pls1, eps=0.00001)
plot(fit.selu)
```

Pm *Projection of a seed matrix on to the column subspace of M with respect to* Sx *inner-product*

Description

The function reuturns a projection of a seed matrix on to the column subspace of M with respect to Sx inner-product.

Usage

```
Pm(M, Sx, seed)
```

Arguments

M	numeric matrix (p * k), a basis matrix of the column space of M
Sx	a inner-product matrix
seed	seed matrix

Examples

```
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

## using cov(X,Y) as a seed matrix
seed <- cov(X,Y)
col.num <- dim(seed)[2]
M <- iniCCA(X, Y, u=4, num.d=col.num)
Sx <- cov(X)
Pm(M, Sx, seed)

## using the first 2 largest eigenvectors of cov(X,Y) as a seed matrix
seed2 <- svd(cov(X,Y))$u[,1:2]
M2 <- iniCCA(X, Y, u=4, num.d=2)
Pm(M, Sx, seed2)
```

print.seedCCA *basic function for printing class "seedCCA"*

Description

The function controls to print class "seedCCA". The function prints the estimated coefficents, if they exist. For subclass "finalCCA", canonical correlations are additionally reported. For subsclass "selectu", increments, suggested number of projections and the values of type and eps are reported.

Usage

```
## S3 method for class 'seedCCA'
print(x,...)
```

Arguments

x	The name of an object of class "seedCCA"
...	arguments passed to the print.method

Examples

```
########  data(cookie) ########
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

fit.seed2 <- seedCCA(X,Y)
fit.seed2
print(fit.seed2)
```

---

seedCCA *Seeded Canonical correlation analysis*

Description

The function seedCCA is mainly for implementing seeded canonical correlation analysis proposed by Im et al. (2015). The function conducts the following four methods, depending on the value of type. The option type has one of c("cca", "seed1", "seed2", "pls").

Usage

```
seedCCA(X,Y,type="seed2",ux=NULL,uy=NULL,u=10,eps=0.01,cut=0.9,d=NULL,AS=TRUE,scale=FALSE)
```

Arguments

X	numeric vector or matrix (n * p), the first set of variables
Y	numeric vector or matrix (n * r), the second set of variables
type	character, a choice of methods among c("cca", "seed1", "seed2", "pls"). The default is "seed2".
ux	numeric, maximum number of projections for X. The default is NULL. If this is not NULL, it surpasses the option u with type="seed1" and p>r. For type="seed2", if ux and uy are not NULL, they surpass u.
uy	numeric, maximum number of projections for Y. The default is NULL. If this is not NULL, it surpasses the option u with type="seed1" and p<r. For type="seed2", if ux and uy are not NULL, they surpass u.

u	numeric, maximum number of projections. The default is 10. This is used for type="seed1", type="seed2" and tyepe="pls".
eps	numeric, the criteria to terminate iterative projections. The default is 0.01. If increment of projections is less than eps, then the iterative projection is terminated.
cut	numeric, between 0 and 1. The default is 0.9. If d is NULL, cut is used for automatic replacements of cov(X,Y) and cov(Y,X) with their eigenvectors, depending on the value of cut. So, if any value of d is given, cut is not effective. cov(X,Y) and cov(Y,X) are replaced with their largest eigenvectors, whose cumulative eigenvalue proportion is bigger than the value of cut. This only works for type="seed2".
d	numeric, the user-selected number of largest eigenvectors of cov(X, Y) and cov(Y, X). The default is NULL. This only works for type="seed2". If any value of d is given, cut does not work.
AS	logical, status of automatic stop of projections. The default is TRUE. If TRUE, the iterative projection is automatically stopped, when the terminaion condition eps is satisfied. IfAS=FALSE, the iterative projections are stopped at the value of u.
scale	logical. scaling predictors to have zero mean and one standard deviation. The default is FALSE. If scale=TRUE, each predictor is scaled with mean 0 and variance 1 for partial least squares. This option works only for type="pls".

Details

Let p and r stand for the numbers of variables in the two sets and n stands for the sample size. The option of type="cca" can work only when max(p,r) < n, and seedCCA conducts standard canonical correlation analysis (Johnson and Wichern, 2007). If type="cca" is given and either p or r is equal to one, ordinary least squares (OLS) is done instead of canonical correlation analysis. If max(p,r) >= n, either type="seed1" or type="seed2" has to be chosen. This is the main purpose of seedCCA. If type="seed1", only one set of variables, saying X with p for convenience, to have more variables than the other, saying Y with r, is initially reduced by the iterative projection approach (Cook et al. 2007). And then, the canonical correlation analysis of the initially-reduced X and the original Y is finalized. If type="seed2", both X and Y are initially reduced. And then, the canonical correlation analysis of the two initially-reduced X and Y are finalzed. If type="pls", partial least squares (PLS) is done. If type="pls" is given, the first set of variables in seedCCA is predictors and the second set is response. This matters The response can be multivariate. Depeding on the value of type, the resulted subclass by seedCCA are different.:

type="cca": subclass "finalCCA" (p >2; r >2; p,r<n)

type="cca": subclass "seedols" (either p or r is equal to 1.)

type="seed1" and type="seed2": subclass "finalCCA" (max(p,r)>n)

type="pls": subclass "seedpls" (p>n and r <n)

So, plot(object) will result in different figure depending on the object.

The order of the values depending on type is follows.:

type="cca": standard CCA (max(p,r)<n, min(p,r)>1) / "finalCCA" subclass

type="cca": ordinary least squares (max(p,r)<n, min(p,r)=1) / "seedols" subclass

type="seed1": seeded CCA with case1 (max(p,r)>n and p>r) / "finalCCA" subclass

type="seed1": seeded CCA with case1 (max(p,r)>n and p<=r) / "finalCCA" subclass

type="seed2": seeded CCA with case2 (max(p,r)>n) / "finalCCA" subclass

type="pls": partial least squares (p>n and r<n) / "seedpls" subclass

Value

type="cca"	Values with selecting type="cca": standard CCA (max(p,r)<n, min(p,r)>1) / "finalCCA" subclass
cor	canonical correlations
xcoef	the estimated canonical coefficients for X
ycoef	the estimated canonical coefficients for Y
Xscores	the estimated canonical variates for X
Yscores	the estimated canonical variates for Y
type="cca"	Values with selecting type="cca": ordinary least squares (max(p,r)<n, min(p,r)=1) / "seedols" subclass
coef	the estimated ordinary least squares coefficients
X	X, the first set
Y	Y, the second set
type="seed1"	Values with selecting type="seed1": seeded CCA with case1 (max(p,r)>n and p>r) / "finalCCA" subclass
cor	canonical correlations
xcoef	the estimated canonical coefficients for X
ycoef	the estimated canonical coefficients for Y
proper.u	a suggested proper number of projections for X
initialMX0	the initialized canonical coefficient matrices of X
newX	initially-reduced X
Y	the original Y
Xscores	the estimated canonical variates for X
Yscores	the estimated canonical variates for Y
type="seed1"	Values with selecting type="seed1": seeded CCA with case1 (max(p,r)>n and p<=r) / "finalCCA" subclass)
cor	canonical correlations
xcoef	the estimated canonical coefficients for X
ycoef	the estimated canonical coefficients for Y
proper.u	a suggested proper number of projections for Y
X	the original X
initialMY0	the initialized canonical coefficient matrices of Y
newY	initially-reduced Y

Xscores	the estimated canonical variates for X
Yscores	the estimated canonical variates for Y
type="seed2"	Values with selecting type="seed2": seeded CCA with case2 (max(p,r)>n) / "finalCCA" subclass
cor	canonical correlations
xcoef	the estimated canonical coefficients for X
ycoef	the estimated canonical coefficients for Y
proper.ux	a suggested proper number of projections for X
proper.uy	a suggested proper number of projections for Y
d	suggested number of eigenvectors of cov(X,Y)
initialMX0	the initialized canonical coefficient matrices of X
initialMY0	the initialized canonical coefficient matrices of Y
newX	initially-reduced X
newY	initially-reduced Y
Xscores	the estimated canonical variates for X
Yscores	the estimated canonical variates for Y
type="pls"	Values with selecting type="pls":: partial least squares (p>n and r<n) / "seed-pls" subclass
coef	the estimated coefficients for each iterative projection upto u
u	the maximum number of projections
X	predictors
Y	response
scale	status of scaling predictors
cases	the number of observations

References

R. D. Cook, B. Li and F. Chiaromonte. Dimension reduction in regression without matrix inversion. Biometrika 2007; 94: 569-584.

Y. Im, H. Gang and JK. Yoo. High-throughput data dimension reduction via seeded canonical correlation analysis, J. Chemometrics 2015; 29: 193-199.

R. A. Johnson and D. W. Wichern. Applied Multivariate Statistical Analysis. Pearson Prentice Hall: New Jersey, USA; 6 edition. 2007; 539-574.

K. Lee and JK. Yoo. Canonical correlation analysis through linear modeling, AUST. NZ. J. STAT. 2014; 56: 59-72.

Examples

```
######  data(cookie) ######
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])
dim(X);dim(Y)

## standard CCA
fit.cca <-seedCCA(X[,1:4], Y, type="cca")  ## standard canonical correlation analysis is done.
plot(fit.cca)

## ordinary least squares
fit.ols1 <-seedCCA(X[,1:4], Y[,1], type="cca")  ## ordinary least squares is done, because r=1.
fit.ols2 <-seedCCA(Y[,1], X[,1:4], type="cca")  ## ordinary least squares is done, because p=1.

## seeded CCA with case 1
fit.seed1 <- seedCCA(X, Y, type="seed1") ## suggested proper value of u is equal to 3.
fit.seed1.ux <- seedCCA(X, Y, ux=6, type="seed1") ## iterative projections done 6 times.
fit.seed1.uy <- seedCCA(Y, X, uy=6, type="seed1", AS=FALSE)  ## projections not done until uy=6.
plot(fit.seed1)

## partial least squares
fit.pls1 <- seedCCA(X, Y[,1], type="pls")
fit.pls.m <- seedCCA(X, Y, type="pls") ## multi-dimensional response
par(mfrow=c(1,2))
plot(fit.pls1); plot(fit.pls.m)


########  data(nutrimouse) ########
data(nutrimouse)
X<-as.matrix(nutrimouse$gene)
Y<-as.matrix(nutrimouse$lipid)
dim(X);dim(Y)

## seeded CCA with case 2
fit.seed2 <- seedCCA(X, Y, type="seed2")  ## d not specified, so cut=0.9 (default) used.
fit.seed2.99 <- seedCCA(X, Y, type="seed2", cut=0.99)  ## cut=0.99 used.
fit.seed2.d3 <- seedCCA(X, Y, type="seed2", d=3)  ## d is specified with 3.

## ux and uy specified, so proper values not suggested.
fit.seed2.uxuy <- seedCCA(X, Y, type="seed2", ux=10, uy=10)
plot(fit.seed2)
```

`seeding` *increments of iterative projections*

Description

Returns increments (nFu) of iterative projections of a seed matrix onto a covariance matrix u times.)

Usage

```
seeding(seed, covx, n,  u=10)
```

Arguments

seed	numeric matrix (p * d), a seed matrix
covx	numeric matrix (p * p), covariance matrix of X
n	numeric, sample sizes
u	numeric, maximum number of projections

Value

nFu	n*Fu values

Examples

```
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])
seed <- cov(X,Y)
covx <- cov(X)
seeding(seed, covx, n=dim(X)[1], u=4)
```

---

seeding.auto.stop *increments of iterative projections with automatic stopping*

Description

Returns increments (nFu) of iterative projections of a seed matrix onto a covariance matrix upto k, which properly chosen by satisfying the terminating condition eps (eps can be selected by users).

Usage

```
seeding.auto.stop(seed, covx, n, u.max=30, eps=0.01)
```

Arguments

seed	numeric matrix (p * d), a seed matrix
covx	numeric matrix (p * p), covariance matrix of X
n	numeric, sample sizes
u.max	numeric, maximum number of projection. The default value is equal to 30.
eps	numeric, a value of a condition for terminating the projection. The default value is equal to 0.01.

Value

nFu	n*Fu values
u	the number of projection properly chosen by satisfying the terminating condition eps

Examples

```
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])
seed <- cov(X,Y)
covx <- cov(X)
seeding.auto.stop(seed, covx, n=dim(X)[1])
seeding.auto.stop(seed, covx, n=dim(X)[1], u.max=20, eps=0.001)
```

---

seedols *Ordinary least squares*

Description

Returns ordinary least squares estimates. And, the function results in subclass "seedols". For this function to work, either X or Y has to be one-dimensional. It is not necessary that X and Y should be predictors and response, respectively. Regardless of the position in the arguments, the one-dimensional and multi-dimensional variables become response and predictors, respectively.

Usage

```
seedols(X, Y)
```

Arguments

X	numeric vector or matrix, a first set of variables
Y	numeric vector or matrix, a second set of variables

Value

coef	the estimated coefficients for each iterative projection upto u
X	X, the first set
Y	Y, the second set

Examples

```
########  data(cookie) ########
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

ols1 <- seedols(X[,1:4],Y[,1])
ols2 <- seedols(Y[,1],X[,1:4])
## ols1 and ols2 are the same results.
```

`seedpls` *Partial least squares through iterative projections*

Description

Returns partial least squares estimates through iterative projections. And, the function results in subclass "seedpls".

Usage

```
seedpls(X, Y, u=5, scale=FALSE)
```

Arguments

`X`	numeric matrix (n * p), a set of predictors
`Y`	numeric vector or matrix (n * r), responses (it can be multi-dimensional)
`u`	numeric, the number of projections. The default is 5.
`scale`	logical, FALSE is default. If TRUE, each predictor is standardized with mean 0 and variance 1

Value

`coef`	the estimated coefficients for each iterative projection upto u
`u`	the maximum number of projections
`X`	Predictors
`Y`	Response
`scale`	status of scaling predictors

Examples

```
########  data(cookie) ########
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

fit.pls1 <- seedpls(X,Y[,1]) ## one-dimensional response
fit.pls2 <- seedpls(X,Y, u=6, scale=TRUE) ## four-dimensional response
```

`selectu` *Function that guides a selection of the terminating index when using seedCCA function*

Description

The usage of `selectu` depends on one of its arguments, `type`. If `tyep="seed1"`, the n*F_u is computed for a higher dimension one of X and Y and a proper number of prjections is reported. For example, suppose that the dimension of X is higher than Y. Then `selectu(X,Y, type="case1")` and `selectu(Y, X, u=5, type="case1")` gives the same results, and it is for X. If `type="seed2"`, n*F_u is computed for X and Y and proper numbers of projections for X and Y are reported. And, For `type="seed2"`, `num.d` must be specified. Its defualt value is 2. The argument `eps` is a terminating condition for stopping projections. The projection is stopped, when the increment is less than the value of `eps`. The argument `auto.stop=TRUE` has the function automatically stopped as soon as the increment is less than the value of `eps`. If not, the increments are computed until the value of `u.max` is reached. The function `selectu` results in subclass "selectu".

Usage

```
selectu(X, Y, type="seed2", u.max=30, auto.stop=TRUE, num.d=2, eps=0.01)
```

Arguments

`X`	numeric matrix (n * p), the first set of variables
`Y`	numeric matrix (n * r), the second set of variables
`type`	character, the default is "seed2". "seed1" is for the first case of Seeded CCA (One set of variable is initially-reduced). "seed2" is for the second case of Seeded CCA (Two sets of variables are initially reduced).
`u.max`	numeric, t he maximum number of u. The default is equal to 30.
`auto.stop`	logical, The default value is TRUE. If TRUE, the iterative projection is automatically stopped, when the terminaion condition eps is satisfied. If FALSE, the iterative projections are stopped at the value of u.max.
`num.d`	numeric, the number of the "num.d" largest eigenvectors of cov(first.set, second.set), if case1=FALSE. The default value is equal to 2. This option works only for type="seed2".
`eps`	numeric, the default value is equal to 0.01. A value for terminating the projection.

Details

The order of the values depending on type is follows:

`type="seed1"`

`type="seed2"`

Value

`type="seed1"`	Values with selecting `type="seed1"`:
`nFu`	incrments (n*Fu) of the iterative projection for initally reduction one set of variable.
`proper.u`	proper value of the number of projections for X
`type`	types of seeded CCA
`eps`	a value for terminating the projection. The default value is equal to 0.01.
`type="seed2"`	Values with selecting `type="seed2"`:
`nFu.x`	incrments (n*Fu) of the iterative projection for initially reducing X.
`nFu.y`	incrments (n*Fu) of the iterative projection for initially reducing Y.
`proper.ux`	proper value of the number of projections for X
`proper.uy`	proper value of the number of projections for Y
`type`	types of seeded CCA
`eps`	a value for terminating the projection. The default value is equal to 0.01.

Examples

```
######  data(cookie) ######
data(cookie)
myseq<-seq(141,651,by=2)
X<-as.matrix(cookie[-c(23,61),myseq])
Y<-as.matrix(cookie[-c(23,61),701:704])

selectu(X, Y, type="seed1")
selectu(X, Y, type="seed1", auto.stop=FALSE)
selectu(X, Y, type="seed2", eps=0.001, num.d=3)
selectu(X, Y, type="seed2", auto.stop=FALSE)



########  data(nutrimouse) ########
data(nutrimouse)
Y<-as.matrix(nutrimouse$lipid)
X<-as.matrix(nutrimouse$gene)
selectu(X, Y, type="seed2", num.d=4)
selectu(X, Y, type="seed2", num.d=4, eps=0.001)
selectu(X, Y, type="seed2", auto.stop=FALSE, num.d=4, eps=0.001)
```

Index