Package ‘PLSbiplot1’

February 19, 2015

Title The Partial Least Squares (PLS) Biplot

Description Principal Component Analysis (PCA) biplots, Covariance monoplots and biplots, Partial Least Squares (PLS) biplots, Partial Least Squares for Generalized Linear Model (PLS-GLM) biplots, Sparse Partial Least Squares (SPLS) biplots and Sparse Partial Least Squares for Generalized Linear Model (SPLS-GLM) biplots.

URL https://www.dropbox.com/sh/wr66u07t1vjm9da/AACg_E4h8MvgOHuCXk69yDIya

Version 0.1

Maintainer Opeoluwa F. Oyedele <OpeoluwaOyedele@gmail.com>

Depends R (>= 3.0.0)

Suggests chemometrics, MASS, mixOmics, mvabund, pls, plsgenomics, robustbase, SensoMineR

License GPL-2

LazyData NA

Author Opeoluwa F. Oyedele [aut, cre], Sugnet Gardner-Lubbe [aut]

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-06 01:34:43

R topics documented:

`cov.biplot` *The covariance biplot*

Description

Takes in a set of predictor variables and a set of response variables and produces a covariance biplot.

Usage

```
cov.biplot(X, Y, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`...`	Other arguments. Currently ignored

Value

The covariance biplot of X and Y

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4",
"I5","S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4",
"I5","S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
cov.biplot(X, Y)

#cocktail data
if(require(SensoMineR))
data(cocktail, package="SensoMineR")
X3 = as.matrix(compo.cocktail, ncol=4)
Y3 = as.matrix(senso.cocktail, ncol=13)
cov.biplot(X3,Y3)
```

`cov.monoplot` *The covariance monoplot*

Description

Takes in only one set of variables (e.g., predictors) and produces a covariance monoplot.

Usage

```
cov.monoplot(X, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`...`	Other arguments. Currently ignored

Value

The covariance monoplot of X

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4",
"I5","S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
cov.monoplot(Y)

#cocktail data
if(require(SensoMineR))
data(cocktail, package="SensoMineR")
Y3 = as.matrix(senso.cocktail, ncol=13)
cov.monoplot(Y3)
```

Mag.Bmat.plot — *Magnitude of the Partial Least Squares Regression (PLSR) coefficients matrix*

Description

Takes in a set of predictor variables and a set of response variables and produces the mean plot of the absolute values of the PLSR coefficients matrix.

Usage

```
Mag.Bmat.plot(X, Y, algorithm = NULL, A, ...)
```

Arguments

X	A (NxP) predictor matrix
Y	A (NxM) response matrix
A	The number of Partial Least Squares (PLS) components
algorithm	Any of the PLS algorithms ("mod.NIPALS", "mod.KernelPLS_R", "mod.KernelPLS_L", "mod.SIMPLS")
...	Other arguments. Currently ignored

Value

The mean plot of the absolute values of the PLSR coefficients matrix

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
Mag.Bmat.plot(X, Y, algorithm=mod.SIMPLS, A=2)

#nutrimouse data
if(require(mixOmics))
data(nutrimouse, package="mixOmics")
X1 = as.matrix(nutrimouse$lipid, ncol=21)
Y1 = as.matrix(nutrimouse$gene, ncol=120)
#VIP
A.final = 9
main2 = mod.VIP(X=X1, Y=Y1, algorithm=mod.SIMPLS, A=A.final, cutoff=0.8)
X.new = X1[,c(main2$X.impor)]  #important X-variables
Mag.Bmat.plot(X=X.new, Y1, algorithm=mod.SIMPLS, A=A.final)
#alternatively
X.scal = scale(X.new, center=TRUE, scale=TRUE)
Y.scal = scale(Y1, center=TRUE, scale=TRUE)
main3 = mod.SIMPLS(X.scal, Y.scal, A.final)
Bmat = main3$X.weights.trans %*% t(main3$Y.loadings)  #PLSR coefficients matrix
dimnames(Bmat) = list(colnames(X.new), colnames(Y1))
Abs.Bmat = abs(Bmat) #absolute values of the coefficients
rowMeans(Abs.Bmat)
```

`mod.KernelPLS_L` *The Kernel algorithm for few(er) variables but large samples by Lindgren et al. (1993)*

Description

Takes in a set of predictor variables and a set of response variables and gives the Partial Least Squares (PLS) parameters.

Usage

```
mod.KernelPLS_L(X, Y, A, ...)
```

Arguments

X	A (NxP) predictor matrix
Y	A (NxM) response matrix

`A` The number of PLS components

`...` Other arguments. Currently ignored

Value

The PLS parameters using the kernel algorithm by Lindgren et al. (1993)

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
mod.KernelPLS_L(X, Y, A=5)
```

mod.KernelPLS_R *The Kernel algorithm for few(er) samples but large variables by Rannar et al. (1994)*

Description

Takes in a set of predictor variables and a set of response variables and gives the Partial Least Squares (PLS) parameters.

Usage

```
mod.KernelPLS_R(X, Y, A, ...)
```

Arguments

`X` A (NxP) predictor matrix

`Y` A (NxM) response matrix

`A` The number of PLS components

`...` Other arguments. Currently ignored

Value

The PLS parameters using the Kernel algorithm by RC$nnar et al. (1994)

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
mod.KernelPLS_R(X, Y, A=5)
```

mod.MMLR *Multivariate Multiple Linear Regression (MMLR)*

Description

Takes in a set of predictor variables and a set of response variables and gives the MMLR parameters.

Usage

```
mod.MMLR(X, Y, ...)
```

Arguments

X	A (NxP) predictor matrix
Y	A (NxM) response matrix
...	Other arguments. Currently ignored

Value

The MMLR parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
mod.MMLR(X, Y)
```

mod.NIPALS *The Nonlinear Iterative PArtial Least Squares (NIPALS) algorithm*

Description

Takes in a set of predictor variables and a set of response variables and gives the Partial Least Squares (PLS) parameters.

Usage

```
mod.NIPALS(X, Y, A, ...)
```

Arguments

X	A (NxP) predictor matrix
Y	A (NxM) response matrix
A	The number of PLS components
...	Other arguments. Currently ignored

Value

The PLS parameters using the NIPALS algorithm

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)  #predictors
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
```

```
Y = as.matrix(oliveoil$sensory, ncol=6)  #responses
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
mod.NIPALS(X, Y, A=5)
```

mod.PCA *Principal Component Analysis (PCA)*

Description

Takes in a samples by variables data matrix and gives the PCA parameters.

Usage

```
mod.PCA(D, r, ...)
```

Arguments

D	A samples by variables data matrix
r	The number of PCA components
...	Other arguments. Currently ignored

Value

The PCA parameters of D

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

mod.PCR *Principal Component Regression (PCR)*

Description

Takes in a set of predictor variables and a set of response variables and gives the PCR parameters.

Usage

```
mod.PCR(X, Y, r, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`r`	The number of PCA components
`...`	Other arguments. Currently ignored

Value

The PCR parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
mod.PCR(X, Y, r=2)
```

`mod.SIMPLS` *The Statistical Inspired Modification to Partial Least Squares (SIMPLS) algorithm*

Description

Takes in a set of predictor variables and a set of response variables and gives the Partial Least Squares (PLS) parameters.

Usage

```
mod.SIMPLS(X, Y, A, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`A`	The number of PLS components
`...`	Other arguments. Currently ignored

Value

The PLS parameters using the SIMPLS algorithm

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
#final number of PLS components
RMSEP = mod.SIMPLS(X, Y, A=5)$RMSEP #RMSEP values
plot(t(RMSEP), type = "b", xlab="Number of components", ylab="RMSEP  values")
A.final = 2 #from the RMSEP plot
#PLS matrices R, P, T, Q, and Y.hat from SIMPLS algorithm
options(digits=3)
mod.SIMPLS(X, Y, A=A.final)
```

`mod.SPLS` *Sparse Partial Least Squares (SPLS) algorithm*

Description

Takes in a set of predictor variables and a set of response variables and gives the SPLS parameters.

Usage

```
mod.SPLS(X, Y, A, lambdaY, lambdaX, eps, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`A`	The number of PLS components
`lambdaY`	A value for the penalty parameters for the soft-thresholding penalization function for Y-weights
`lambdaX`	A value for the penalty parameters for the soft-thresholding penalization function for X-weights
`eps`	Cut off value for convergence step
`...`	Other arguments. Currently ignored

Value

The SPLS parameters of D=[X Y]

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(chemometrics))
data(ash, package=”chemometrics”)
X1 = as.matrix(ash[,10:17], ncol=8)
Y1 = as.matrix(ash$SOT)
colnames(Y1) = paste(”SOT”)
mod.SPLS(X=scale(X1), Y=scale(Y1), A=2, lambdaY=0, lambdaX=10.10, eps=1e-5)
#lambdaX and lambdaY value are determined using function opt.penalty.values
#for more details, see opt.penalty.values help file
```

`mod.VIP` *The Variable Importance in the Projection (VIP) values*

Description

Takes in a set of predictor variables and a set of response variables and gives the VIP values for the predictor variables.

Usage

```
mod.VIP(X, Y, algorithm = NULL, A, cutoff = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`A`	The number of Partial Least Squares (PLS) components
`algorithm`	Any of the PLS algorithms ("mod.SIMPLS","mod.NIPALS", "mod.KernelPLS_R", "mod.KernelPLS_L")
`cutoff`	desired cut off value to use for selecting the important X-variables
`...`	Other arguments. Currently ignored

Value

The VIP value for each of the X-variables

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(chemometrics))
data(cereal, package=”chemometrics”)
X = as.matrix(cbind(cereal$X))
Y = as.matrix(cbind(cereal$Y))
main2 = mod.VIP(X=X, Y=Y, algorithm=mod.SIMPLS, A=2, cutoff=0.8)
main2
X.new = X[,c(main2$X.impor)]  #important X-variables
X.new

#nutrimouse data
if(require(mixOmics))
data(nutrimouse, package=”mixOmics”)
X1 = as.matrix(nutrimouse$lipid, ncol=21)
Y1 = as.matrix(nutrimouse$gene, ncol=120)
main = mod.SIMPLS(X=X1, Y=Y1, A=17) #using the SIMPLS algorithm
#RMSEP
RMSEP = main$RMSEP
plot(t(RMSEP), type = ”b”, xlab=”Number of components”, ylab=”RMSEP  values”)
A.final = 9 #from the RMSEP plot
#Final PLSR
mod.SIMPLS(X=X1, Y=Y1, A=A.final)
#VIP
main2 = mod.VIP(X=X1, Y=Y1, algorithm=mod.SIMPLS, A=A.final, cutoff=0.8)
main2
X.new = X1[,c(main2$X.impor)]  #important X-variables
X.new
```

`opt.penalty.values` *Choosing a value for penalty parameters lambdaX and lambdaY for the Sparse Partial Least Squares (SPLS) and Sparse Partial Least Squares-Generalized Linear Model (SPLS-GLM) analyses*

Description

Gives the value of the penalty parameters (lambdaX,lambdaY) having the minimum RMSEP value.

Usage

```
opt.penalty.values(X, Y, A, algorithm = NULL, eps, from.value.X, to.value.X,
  from.value.Y, to.value.Y, lambdaY.len, lambdaX.len, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix; can also be a vector in the case of the SPLS-GLM
`A`	The number of Partial Least Squares (PLS) components
`algorithm`	Any of the SPLS or SPLS-GLM algorithms ("mod.SPLS", "SPLS.GLM", "SPLS.binomial.GLM")

`eps`	Cut off value for convergence step
`from.value.X`	starting value for lambdaX
`to.value.X`	ending value for lambdaX
`from.value.Y`	starting value for lambdaY
`to.value.Y`	ending value for lambdaY
`lambdaY.len`	length of lambdaY value
`lambdaX.len`	length of lambdaX value
`...`	Other arguments. Currently ignored

Value

the value of the penalty parameters (lambdaX,lambdaY) having the minimum RMSEP value, as well as the RMSEP values obtained when the lambdaX and lambdaY values were paired together

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(chemometrics))
data(ash, package="chemometrics")
X1 = as.matrix(ash[,10:17], ncol=8)
Y1 = as.matrix(ash$SOT)
colnames(Y1) = paste("SOT")
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=500, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
#thus, use lambdaX = 10.10 and lambdaY = 0 for the SPLS analysis of this data

#possum.mat data
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.GLM, eps=1e-3,
from.value.X=1, to.value.X=4, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
#thus, use lambdaY = 0 and lambdaX = 3.3 for the (Poisson) SPLS-GLM analysis of this data

#Pima.tr data
if(require(MASS))
data(Pima.tr, package="MASS")
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
```

```
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.binomial.GLM, eps=1e-3,
from.value.X=0, to.value.X=95, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
#thus, use lambdaY = 0 and lambdaX = 0.96 for the (Binomial) SPLS-GLM analysis of this data
```

PCA.biplot *The Principal Component Analysis (PCA) biplot*

Description

Takes in a samples by variables data matrix and produces a PCA biplot.

Usage

```
PCA.biplot(D, method = NULL, ax.tickvec.D = NULL, ...)
```

Arguments

D	A samples by variables data matrix
method	the mod.PCA algorithm
ax.tickvec.D	tick marker length per axis in the PCA biplot
...	Other arguments. Currently ignored

Value

The PCA biplot of D with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
Dmat = as.matrix(oliveoil)  #(16x11) overall original data matrix
dimnames(Dmat) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK","Yellow",
"Green","Brown","Glossy","Transp","Syrup")))
PCA.biplot(D=Dmat, method=mod.PCA, ax.tickvec.D=c(8,5,5,7,6,4,5,5,8,7,7))
```

PCA.biplot_no.SN *The Principal Component Analysis (PCA) biplot with the labels of the samples points excluded*

Description

Takes in a samples by variables data matrix and produces a PCA biplot, where the labels of the samples points excluded.

Usage

```
PCA.biplot_no.SN(D, method = NULL, ax.tickvec.D = NULL, ...)
```

Arguments

`D`	A samples by variables data matrix
`method`	the mod.PCA algorithm
`ax.tickvec.D`	tick marker length per axis in the PCA biplot
`...`	Other arguments. Currently ignored

Value

The PCA biplot of D with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
Dmat = as.matrix(oliveoil)  #(16x11) overall original data matrix
dimnames(Dmat) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK","Yellow","Green","Brown",
"Glossy","Transp","Syrup")))
PCA.biplot_no.SN(D=Dmat, method=mod.PCA, ax.tickvec.D=c(8,5,5,7,6,4,5,5,8,7,7))

#glass data
if(require(chemometrics))
data(glass, package="chemometrics")
Dmat = matrix(glass,ncol=13)
dimnames(Dmat) = list(1:180, paste(c("Na2O", "MgO", "Al2O3", "SiO2",
"P2O5", "SO3", "Cl", "K2O", "CaO", "MnO", "Fe2O3", "BaO", "PbO")))
PCA.biplot_no.SN(D=Dmat, method=mod.PCA, ax.tickvec.D=rep(5,ncol(Dmat)))
```

`PLS.binomial.GLM` *Partial Least Squares-Generalized Linear Model (PLS-GLM) fitted for Binomial y*

Description

Takes in a set of predictor variables and a set of response variables and gives the PLS-GLM parameters.

Usage

```
PLS.binomial.GLM(X, y, A, PLS_algorithm = NULL, eps = 1e-04, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) Binomial-distributed response vector
`A`	The number of PLS components
`PLS_algorithm`	The mod.NIPALS algorithm
`eps`	Cut off value for convergence step
`...`	Other arguments. Currently ignored

Value

The PLS-GLM parameters of D=[X y]

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

`PLS.biplot` *The Partial Least Squares (PLS) biplot*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot.

Usage

```
PLS.biplot(X, Y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.Y = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`algorithm`	Any of the PLS algorithms ("mod.NIPALS", "mod.KernelPLS_R", "mod.KernelPLS_L", "mod.SIMPLS")
`ax.tickvec.X`	tick marker length for each X-variable axis in the PLS biplot
`ax.tickvec.Y`	tick marker length for each Y-variable axis in the PLS biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of D=[X Y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
#SIMPLS biplot
PLS.biplot(X, Y, algorithm=mod.SIMPLS, ax.tickvec.X=c(8,5,5,5,5), ax.tickvec.Y=c(5,8,5,6,9,8))
#Kernel PLS biplot
PLS.biplot(X, Y, algorithm=mod.KernelPLS_R, ax.tickvec.X=c(3,3,4,5,2), ax.tickvec.Y=c(3,3,5,6,7,6))
```

`PLS.biplot.area` *The Partial Least Squares (PLS) biplot with triangles for estimating the Partial Least Squares Regression (PLSR) coefficients*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot, but with rotated coefficient points.

Usage

```
PLS.biplot.area(X, Y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.Y = NULL, base.tri, bi.value, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`algorithm`	Any of the PLS algorithms ("mod.NIPALS", "mod.KernelPLS_R", "mod.KernelPLS_L", "mod.SIMPLS")
`ax.tickvec.X`	tick marker length for each X-variable axis in the PLS biplot
`ax.tickvec.Y`	tick marker length for each Y-variable axis in the PLS biplot
`base.tri`	The desired Y-variable axis to use as the base for the triangle
`bi.value`	The desired rotated coefficient points (bi) to approximate
`...`	Other arguments. Currently ignored

Value

The PLS biplot of D=[X Y] with rotated coefficient points

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
#with 1 triangle
PLS.biplot.area(X, Y, algorithm=mod.SIMPLS, ax.tickvec.X=c(8,5,5,5,5),
ax.tickvec.Y=c(5,10,5,6,7,10), base.tri=3, bi.value=4)
#with 4 triangles
PLS.biplot.area(X, Y, algorithm=mod.SIMPLS, ax.tickvec.X=c(8,5,5,5,5),
ax.tickvec.Y=c(5,10,5,6,7,10), base.tri=2, bi.value=c(1,2,3,4,5))
```

`PLS.biplot_no.SN` *The Partial Least Squares (PLS) biplot with no sample points names*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot, but with no sample points names.

Usage

```
PLS.biplot_no.SN(X, Y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.Y = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`algorithm`	Any of the PLS algorithms ("mod.NIPALS", "mod.KernelPLS_R", "mod.KernelPLS_L", "mod.SIMPLS")
`ax.tickvec.X`	tick marker length for each X-variable axis in the PLS biplot
`ax.tickvec.Y`	tick marker length for each Y-variable axis in the PLS biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of D=[X Y] with no sample points names

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
PLS.biplot_no.SN(X, Y, algorithm=mod.SIMPLS, ax.tickvec.X=c(8,5,5,5,5),
 ax.tickvec.Y=c(5,8,5,6,9,8))

#cocktail data
if(require(SensoMineR))
data(cocktail, package="SensoMineR")
X3 = as.matrix(compo.cocktail, ncol=4)
Y3 = as.matrix(senso.cocktail, ncol=13)
PLS.biplot_no.SN(X=X3, Y3, algorithm=mod.SIMPLS, ax.tickvec.X=rep(2,ncol(X3)),
 ax.tickvec.Y=rep(3,ncol(Y3)))
```

PLS.biplot_no_labels *The Partial Least Squares (PLS) biplot with the labels of the samples, coefficient points and tick markers excluded*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot, but with no sample points names.

Usage

```
PLS.biplot_no_labels(X, Y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.Y = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`algorithm`	Any of the PLS algorithms ("mod.NIPALS", "mod.KernelPLS_R", "mod.KernelPLS_L", "mod.SIMPLS")
`ax.tickvec.X`	tick marker length for each X-variable axis in the PLS biplot
`ax.tickvec.Y`	tick marker length for each Y-variable axis in the PLS biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of D=[X Y] with no sample points names but with fainted tick markers and labels

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(pls))
data(oliveoil, package="pls")
X = as.matrix(oliveoil$chemical, ncol=5)
dimnames(X) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Acidity","Peroxide","K232","K270","DK")))
Y = as.matrix(oliveoil$sensory, ncol=6)
dimnames(Y) = list(paste(c("G1","G2","G3","G4","G5","I1","I2","I3","I4","I5",
"S1","S2","S3","S4","S5","S6")),
paste(c("Yellow","Green","Brown","Glossy","Transp","Syrup")))
PLS.biplot_no_labels(X, Y, algorithm=mod.SIMPLS, ax.tickvec.X=c(8,5,5,5,5),
ax.tickvec.Y=c(5,8,5,6,9,8))
```

```
#cocktail data
if(require(SensoMineR))
data(cocktail, package="SensoMineR")
X3 = as.matrix(compo.cocktail, ncol=4)
Y3 = as.matrix(senso.cocktail, ncol=13)
PLS.biplot_no_labels(X=X3, Y3, algorithm=mod.SIMPLS, ax.tickvec.X=rep(2,ncol(X3)),
ax.tickvec.Y=rep(3,ncol(Y3)))
```

PLS.GLM *Partial Least Squares-Generalized Linear Model (PLS-GLM) algorithm*

Description

Takes in a set of predictor variables and a set of response variables and gives the PLS-GLM parameters.

Usage

```
PLS.GLM(X, y, A, PLS_algorithm = NULL, eps = 1e-04, ...)
```

Arguments

X	A (NxP) predictor matrix
y	A (Nx1) Poisson-distributed response vector
A	The number of PLS components
PLS_algorithm	The mod.NIPALS algorithm
eps	Cut off value for convergence step
...	Other arguments. Currently ignored

Value

The PLS-GLM parameters of D=[X y]

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

PLS.GLM.biplot *The Partial Least Squares (PLS) biplot for Generalized Linear Model (GLM)*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the (univariate) GLMs.

Usage

```
PLS.GLM.biplot(X, y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.y = NULL, ax.tickvec.b = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) response vector
`algorithm`	The PLS.GLM algorithm
`ax.tickvec.X`	tick marker length for each X-variable axis in the biplot
`ax.tickvec.y`	tick marker length for the y-variable axis in the biplot
`ax.tickvec.b`	(purple) tick marker length for the y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of a GLM of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat #data matrix
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
PLS.GLM.biplot(X, y, algorithm=PLS.GLM, ax.tickvec.X=rep(5,ncol(X)),
 ax.tickvec.y=10, ax.tickvec.b=7)
```

PLS.GLM.biplot_bvec *A zoomed-in display of the coefficient points in the Partial Least Squares (PLS) biplot for Generalized Linear Model (GLM)*

Description

Takes in a set of predictor variables and a set of response variables and produces a zoomed-in display of the coefficient points in the PLS biplot for the (univariate) GLMs.

Usage

```
PLS.GLM.biplot_bvec(X, y, algorithm = NULL, ax.tickvec.b = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) response vector
`algorithm`	The PLS.GLM_SIMPLS algorithm
`ax.tickvec.b`	(purple) tick marker length for the y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

A zoomed-in display of the coefficient points in the PLS biplot of a GLM of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
PLS.GLM.biplot_bvec(X, y, algorithm=PLS.GLM, ax.tickvec.b=10)

#Pima.tr data
if(require(MASS))
data(Pima.tr, package="MASS")
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
PLS.GLM.biplot_bvec(X, y, algorithm=PLS.binomial.GLM,ax.tickvec.b=10)
```

PLS.GLM.biplot_no.SN *The Partial Least Squares (PLS) biplot for Generalized Linear Model (GLM) with the labels of the sample points excluded*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the (univariate) GLMs, with the labels of the sample points excluded.

Usage

```
PLS.GLM.biplot_no.SN(X, y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.y = NULL, ax.tickvec.b = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) response vector
`algorithm`	The PLS.GLM algorithm
`ax.tickvec.X`	tick marker length for each X-variable axis in the biplot
`ax.tickvec.y`	tick marker length for the y-variable axis in the biplot
`ax.tickvec.b`	(purple) tick marker length for the y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of a GLM of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
#Poisson-fitted
PLS.GLM.biplot_no.SN(X, y, algorithm=PLS.GLM, ax.tickvec.X=rep(5,ncol(X)),
 ax.tickvec.y=10, ax.tickvec.b=7)

#Pima.tr data
if(require(MASS))
data(Pima.tr, package="MASS")
```

```
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
PLS.GLM.biplot_no.SN(X, y, algorithm=PLS.binomial.GLM,
ax.tickvec.X=c(3,3,8,7,8,5,2), ax.tickvec.y=3, ax.tickvec.b=3)
```

PLS.GLM.biplot_no_labels

The Partial Least Squares (PLS) biplot for Generalized Linear Model (GLM), with the labels of the samples, coefficient points and tick markers excluded

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the (univariate) GLMs, with the labels of the samples, coefficient points and tick markers excluded.

Usage

```
PLS.GLM.biplot_no_labels(X, y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.y = NULL, ax.tickvec.b = NULL, ...)
```

Arguments

X	A (NxP) predictor matrix
y	A (Nx1) response vector
algorithm	The PLS.GLM algorithm
ax.tickvec.X	tick marker length for each X-variable axis in the biplot
ax.tickvec.y	tick marker length for the y-variable axis in the biplot
ax.tickvec.b	(purple) tick marker length for the y-variable axis in the biplot
...	Other arguments. Currently ignored

Value

The PLS biplot of a GLM of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
PLS.GLM.biplot_no_labels(X, y, algorithm=PLS.GLM, ax.tickvec.X=rep(5,ncol(X)),
ax.tickvec.y=10, ax.tickvec.b=7)

#Pima.tr data
if(require(MASS))
data(Pima.tr, package="MASS")
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
PLS.GLM.biplot_no_labels(X, y, algorithm=PLS.binomial.GLM,
ax.tickvec.X=c(3,3,8,7,8,5,2), ax.tickvec.y=3, ax.tickvec.b=3)
```

PLS.GLM.biplot_SIMPLS *The Partial Least Squares (PLS) biplot for Generalized Linear Model (GLM) fitted using the SIMPLS algorithm*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the (univariate) GLMs.

Usage

```
PLS.GLM.biplot_SIMPLS(X, y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.y = NULL, ax.tickvec.b = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) response vector
`algorithm`	The PLS.GLM_SIMPLS algorithm
`ax.tickvec.X`	tick marker length for each X-variable axis in the biplot
`ax.tickvec.y`	tick marker length for the y-variable axis in the biplot
`ax.tickvec.b`	(purple) tick marker length for the y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of a GLM (fitted using the SIMPLS algorithm) of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
PLS.GLM.biplot_SIMPLS(X, y, algorithm=PLS.GLM,
ax.tickvec.X=rep(5,ncol(X)), ax.tickvec.y=10, ax.tickvec.b=7)
```

PLS.GLM.biplot_SIMPLS_no.SN

The Partial Least Squares (PLS) biplot for Generalized Linear Model (GLM) fitted using the SIMPLS algorithm, with the labels of the sample points excluded

---

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the (univariate) GLMs, with the labels of the sample points excluded.

Usage

```
PLS.GLM.biplot_SIMPLS_no.SN(X, y, algorithm = NULL, ax.tickvec.X = NULL,
  ax.tickvec.y = NULL, ax.tickvec.b = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) response vector
`algorithm`	The PLS.GLM_SIMPLS algorithm
`ax.tickvec.X`	tick marker length for each X-variable axis in the biplot
`ax.tickvec.y`	tick marker length for the y-variable axis in the biplot
`ax.tickvec.b`	(purple) tick marker length for the y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of a GLM (fitted using the SIMPLS algorithm) of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
#Poisson-fitted
PLS.GLM.biplot_SIMPLS_no.SN(X, y, algorithm=PLS.GLM,
ax.tickvec.X=rep(5,ncol(X)), ax.tickvec.y=10, ax.tickvec.b=7)
```

`PLS.GLM_SIMPLS` *Partial Least Squares-Generalized Linear Model (PLS-GLM) fitted using the SIMPLS algorithm*

Description

Takes in a set of predictor variables and a set of response variables and gives the PLS-GLM parameters.

Usage

```
PLS.GLM_SIMPLS(X, y, A, PLS_algorithm = NULL, eps = 1e-04, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) Poisson-distributed response vector
`A`	The number of PLS components
`PLS_algorithm`	The mod.SIMPLS algorithm
`eps`	Cut off value for convergence step
`...`	Other arguments. Currently ignored

Value

The PLS-GLM parameters of D=[X y]

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

SPLS.binomial.GLM *Sparse Partial Least Squares-Generalized Linear Model (SPLS-GLM) algorithm for Binomial y*

Description

Takes in a set of predictor variables and a set of response variables and gives the SPLS-GLM parameters.

Usage

```
SPLS.binomial.GLM(X, y, A, lambdaY, lambdaX, eps = 0.001, ...)
```

Arguments

X	A (NxP) predictor matrix
y	A (Nx1) Binomial-distributed response vector
A	The number of PLS components
lambdaY	A value for the penalty parameters for the soft-thresholding penalization function for Y-weights
lambdaX	A value for the penalty parameters for the soft-thresholding penalization function for X-weights
eps	Cut off value for convergence step
...	Other arguments. Currently ignored

Value

The SPLS-GLM parameters of D=[X y]

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(MASS))
data(Pima.tr, package="MASS")
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
SPLS.binomial.GLM(scale(X), scale(y), A=2, lambdaY=0, lambdaX=0.96, eps=1e-3)
#lambdaX and lambdaY value are determined using function opt.penalty.values
#for more details, see opt.penalty.values help file
```

SPLS.biplot *The Partial Least Squares (PLS) biplot for Sparse Partial Least Squares (SPLS)*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the SPLS.

Usage

```
SPLS.biplot(X, Y, algorithm = NULL, eps, lambdaY = NULL, lambdaX = NULL,
  ax.tickvec.X = NULL, ax.tickvec.Y = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`algorithm`	The SPLS algorithm
`eps`	Cut off value for convergence step
`lambdaY`	A value for the penalty parameters for the soft-thresholding penalization function for Y-weights
`lambdaX`	A value for the penalty parameters for the soft-thresholding penalization function for X-weights
`ax.tickvec.X`	tick marker length for each X-variable axis in the biplot
`ax.tickvec.Y`	tick marker length for the Y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of a SPLS of D=[X Y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
data(toxicity, package="robustbase")
Y1 = as.matrix(cbind(toxicity$toxicity))
dimnames(Y1) = list(paste(1:nrow(Y1)), "toxicity")
X1 = as.matrix(cbind(toxicity[,2:10]))
rownames(X1) = paste(1:nrow(X1))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=10, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
colnames(X.new)
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
colnames(Y.new)
SPLS.biplot(X.new, Y.new, algorithm=mod.SPLS, lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5, ax.tickvec.X=rep(3,ncol(X.new)), ax.tickvec.Y=rep(4,ncol(Y.new)))

#ash data
if(require(chemometrics))
data(ash, package=”chemometrics”)
X1 = as.matrix(ash[,10:17], ncol=8)
Y1 = as.matrix(ash$SOT)
colnames(Y1) = paste(”SOT”)
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=500, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
colnames(X.new)  #P=6
colnames(X1)  #P=8
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
colnames(Y.new)
SPLS.biplot(X.new, Y.new, algorithm=mod.SPLS, lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5, ax.tickvec.X=rep(1,ncol(X.new)), ax.tickvec.Y=rep(5,ncol(Y.new)))
```

SPLS.biplot_Bmat *A zoomed-in display of the coefficient points in the Partial Least Squares (PLS) biplot for Sparse Partial Least Squares (SPLS)*

Description

Takes in a set of predictor variables and a set of response variables and produces a zoomed-in display of the coefficient points in the PLS biplot for SPLS.

Usage

```
SPLS.biplot_Bmat(X, Y, algorithm = NULL, eps, lambdaY = NULL,
  lambdaX = NULL, ax.tickvec.B = NULL, ...)
```

Arguments

`X` A (NxP) predictor matrix

`Y` A (NxM) response matrix

`algorithm` The SPLS algorithm

`eps` Cut off value for convergence step

`lambdaY` A value for the penalty parameters for the soft-thresholding penalization function for Y-weights

`lambdaX` A value for the penalty parameters for the soft-thresholding penalization function for X-weights

`ax.tickvec.B` (purple) tick marker length for the Y-variable axes in the biplot

`...` Other arguments. Currently ignored

Value

A zoomed-in display of the coefficient points in the PLS biplot of a SPLS-GLM of D=[X Y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
data(toxicity, package="robustbase")
Y1 = as.matrix(cbind(toxicity$toxicity))
dimnames(Y1) = list(paste(1:nrow(Y1)), "toxicity")
X1 = as.matrix(cbind(toxicity[,2:10]))
rownames(X1) = paste(1:nrow(X1))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=10, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
SPLS.biplot_Bmat(X.new, Y.new, algorithm=mod.SPLS,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5, ax.tickvec.B=rep(5,ncol(Y.new)))

#ash data
if(require(chemometrics))
data(ash, package="chemometrics")
X1 = as.matrix(ash[,10:17], ncol=8)
Y1 = as.matrix(ash$SOT)
colnames(Y1) = paste("SOT")
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=500, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
colnames(X.new)  #P=6
colnames(X1)  #P=8
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
colnames(Y.new)
SPLS.biplot_Bmat(X.new, Y.new, algorithm=mod.SPLS,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5, ax.tickvec.B=5)
```

SPLS.biplot_no_ax.labels

The Partial Least Squares (PLS) biplot for Sparse Partial Least Squares (SPLS), with the labels of the tick markers excluded

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the SPLS with the labels of the tick markers excluded.

Usage

```
SPLS.biplot_no_ax.labels(X, Y, algorithm = NULL, eps, lambdaY = NULL,
  lambdaX = NULL, ax.tickvec.X = NULL, ax.tickvec.Y = NULL, ...)
```

Arguments

X	A (NxP) predictor matrix
Y	A (NxM) response matrix
algorithm	The SPLS algorithm
eps	Cut off value for convergence step
lambdaY	A value for the penalty parameters for the soft-thresholding penalization function for Y-weights
lambdaX	A value for the penalty parameters for the soft-thresholding penalization function for X-weights
ax.tickvec.X	tick marker length for each X-variable axis in the biplot
ax.tickvec.Y	tick marker length for the Y-variable axis in the biplot
...	Other arguments. Currently ignored

Value

The PLS biplot of a SPLS of D=[X Y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
data(toxicity, package="robustbase")
Y1 = as.matrix(cbind(toxicity$toxicity))
dimnames(Y1) = list(paste(1:nrow(Y1)), "toxicity")
X1 = as.matrix(cbind(toxicity[,2:10]))
rownames(X1) = paste(1:nrow(X1))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=10, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
```

```
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
SPLS.biplot_no_ax.labels(X.new, Y.new,
algorithm=mod.SPLS, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use,
eps=1e-5, ax.tickvec.X=rep(3,ncol(X.new)),
ax.tickvec.Y=rep(4,ncol(Y.new)))

#ash data
if(require(chemometrics))
data(ash, package="chemometrics")
X1 = as.matrix(ash[,10:17], ncol=8)
Y1 = as.matrix(ash$SOT)
colnames(Y1) = paste("SOT")
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=500, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
colnames(X.new)  #P=6
colnames(X1)  #P=8
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
colnames(Y.new)
SPLS.biplot_no_ax.labels(X.new, Y.new,
algorithm=mod.SPLS, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use,
eps=1e-5, ax.tickvec.X=rep(1,ncol(X.new)),
ax.tickvec.Y=rep(5,ncol(Y.new)))
```

`SPLS.biplot_no_labels` *The Partial Least Squares (PLS) biplot for Sparse Partial Least Squares (SPLS), with the labels of the samples, coefficient points and tick markers excluded*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the SPLS with the labels of the samples, coefficient points and tick markers excluded.

Usage

```
SPLS.biplot_no_labels(X, Y, algorithm = NULL, eps, lambdaY = NULL,
  lambdaX = NULL, ax.tickvec.X = NULL, ax.tickvec.Y = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`Y`	A (NxM) response matrix
`algorithm`	The SPLS algorithm
`eps`	Cut off value for convergence step
`lambdaY`	A value for the penalty parameters for the soft-thresholding penalization function for Y-weights
`lambdaX`	A value for the penalty parameters for the soft-thresholding penalization function for X-weights
`ax.tickvec.X`	tick marker length for each X-variable axis in the biplot
`ax.tickvec.Y`	tick marker length for the Y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of a SPLS of D=[X Y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
data(toxicity, package="robustbase")
Y1 = as.matrix(cbind(toxicity$toxicity))
dimnames(Y1) = list(paste(1:nrow(Y1)), "toxicity")
X1 = as.matrix(cbind(toxicity[,2:10]))
rownames(X1) = paste(1:nrow(X1))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=10, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
```

```
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
SPLS.biplot_no_labels(X.new, Y.new,
algorithm=mod.SPLS, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, eps=1e-5,
ax.tickvec.X=rep(3,ncol(X.new)),
ax.tickvec.Y=rep(4,ncol(Y.new)))

#ash data
if(require(chemometrics))
data(ash, package="chemometrics")
X1 = as.matrix(ash[,10:17], ncol=8)
Y1 = as.matrix(ash$SOT)
colnames(Y1) = paste("SOT")
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2 = opt.penalty.values(X=scale(X1), Y=scale(Y1), A=2, algorithm=mod.SPLS, eps=1e-5,
from.value.X=0, to.value.X=500, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS analysis
main3 = mod.SPLS(X=scale(X1), Y=scale(Y1), A=2,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-5)
X.to.use = main3$X.select
Y.to.use = main3$Y.select
X.new = as.matrix(X1[,X.to.use])
colnames(X.new)  #P=6
colnames(X1)  #P=8
Y.new = as.matrix(Y1[,Y.to.use])
colnames(Y.new) = colnames(Y1)
colnames(Y.new)
SPLS.biplot_no_labels(X.new, Y.new,
algorithm=mod.SPLS, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, eps=1e-5,
ax.tickvec.X=rep(1,ncol(X.new)),
ax.tickvec.Y=rep(5,ncol(Y.new)))
```

SPLS.GLM *Sparse Partial Least Squares-Generalized Linear Model (SPLS-GLM) algorithm*

Description

Takes in a set of predictor variables and a set of response variables and gives the SPLS-GLM parameters.

Usage

```
SPLS.GLM(X, y, A, lambdaY, lambdaX, eps = 0.001, ...)
```

Arguments

`X` A (NxP) predictor matrix

`y` A (Nx1) Poisson-distributed response vector

`A` The number of PLS components

`lambdaY` A value for the penalty parameters for the soft-thresholding penalization function for Y-weights

`lambdaX` A value for the penalty parameters for the soft-thresholding penalization function for X-weights

`eps` Cut off value for convergence step

`...` Other arguments. Currently ignored

Value

The SPLS-GLM parameters of D=[X y]

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
SPLS.GLM(scale(X), scale(y), A=2, lambdaY=0, lambdaX=3.3, eps=1e-3)
#lambdaX and lambdaY value are determined using function opt.penalty.values
#for more details, see opt.penalty.values help file
```

`SPLS.GLM.biplot` *The Partial Least Squares (PLS) biplot for Sparse Partial Least Squares-Generalized Linear Model (SPLS-GLM)*

Description

Takes in a set of predictor variables and a set of response variables and produces a PLS biplot for the (univariate) SPLS-GLMs.

Usage

```
SPLS.GLM.biplot(X, y, algorithm = NULL, eps, lambdaY = NULL,
  lambdaX = NULL, ax.tickvec.X = NULL, ax.tickvec.y = NULL,
  ax.tickvec.b = NULL, ...)
```

Arguments

`X` A (NxP) predictor matrix

`y` A (Nx1) response vector

`algorithm` Any of the SPLS-GLM algorithm ("SPLS.GLM", "SPLS.binomial.GLM")

`eps` Cut off value for convergence step

`lambdaY` A value for the penalty parameters for the soft-thresholding penalization function for Y-weights

`lambdaX` A value for the penalty parameters for the soft-thresholding penalization function for X-weights

`ax.tickvec.X` tick marker length for each X-variable axis in the biplot

`ax.tickvec.y` tick marker length for the y-variable axis in the biplot

`ax.tickvec.b` (purple) tick marker length for the y-variable axis in the biplot

`...` Other arguments. Currently ignored

Value

The PLS biplot of a SPLS-GLM of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2B = opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.GLM, eps=1e-3,
from.value.X=1, to.value.X=4, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2B$min.RMSEP.value
lambdaY.to.use = main2B$lambdaY.to.use
lambdaX.to.use = main2B$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS-GLM analysis
main3 = SPLS.GLM(scale(X), scale(y), A=2, lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use, eps=1e-3)
X.to.use = main3$X.select
X.new = as.matrix(X[,names(X.to.use)])
colnames(X.new)
main3$Y.select #note
SPLS.GLM.biplot(X.new, y, algorithm=SPLS.GLM, eps=1e-3, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, ax.tickvec.X=c(10,5,5,5,5,5,5,5,5,5,5,5,5), ax.tickvec.y=8,
ax.tickvec.b=12)

#Pima.tr data
```

```
if(require(MASS))
data(Pima.tr, package="MASS")
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
main2 = opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.binomial.GLM,
eps=1e-3, from.value.X=0, to.value.X=95, from.value.Y=0, to.value.Y=0, lambdaY.len=1,
lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
#SPLS-GLM analysis
main3 = SPLS.binomial.GLM(scale(X), scale(y), A=2, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, eps=1e-3)
X.to.use = main3$X.select
X.new = as.matrix(X[,names(X.to.use)])
colnames(X.new)
SPLS.GLM.biplot(X.new, y, algorithm=SPLS.binomial.GLM, eps=1e-3, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, ax.tickvec.X=c(3,3,3,3,3,3,1), ax.tickvec.y=3, ax.tickvec.b=3)
```

SPLS.GLM.biplot_bvec *A zoomed-in display of the coefficient points in the Partial Least Squares (PLS) biplot for Sparse Partial Least Squares-Generalized Linear Model (SPLS-GLM)*

Description

Takes in a set of predictor variables and a set of response variables and produces a zoomed-in display of the coefficient points in the PLS biplot for the (univariate) SPLS-GLMs.

Usage

```
SPLS.GLM.biplot_bvec(X, y, algorithm = NULL, eps, lambdaY = NULL,
  lambdaX = NULL, ax.tickvec.b = NULL, ...)
```

Arguments

`X` A (NxP) predictor matrix

`y` A (Nx1) response vector

`algorithm` Any of the SPLS-GLM algorithm ("SPLS.GLM", "SPLS.binomial.GLM")

`eps` Cut off value for convergence step

`lambdaY` A value for the penalty parameters for the soft-thresholding penalization function for Y-weights

`lambdaX` A value for the penalty parameters for the soft-thresholding penalization function for X-weights

`ax.tickvec.b` (purple) tick marker length for the y-variable axis in the biplot

`...` Other arguments. Currently ignored

Value

A zoomed-in display of the coefficient points in the PLS biplot of a SPLS-GLM of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2B = opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.GLM, eps=1e-3,
from.value.X=1, to.value.X=4, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2B$min.RMSEP.value
lambdaY.to.use = main2B$lambdaY.to.use
lambdaX.to.use = main2B$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS-GLM analysis
main3 = SPLS.GLM(scale(X), scale(y), A=2, lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use, eps=1e-3)
X.to.use = main3$X.select
X.new = as.matrix(X[,names(X.to.use)])
colnames(X.new)
main3$Y.select #note
SPLS.GLM.biplot_bvec(X.new, y, algorithm=SPLS.GLM, eps=1e-3, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, ax.tickvec.b=30)

#Pima.tr data
if(require(MASS))
data(Pima.tr, package="MASS")
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
main2 = opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.binomial.GLM, eps=1e-3,
from.value.X=0, to.value.X=95, from.value.Y=0, to.value.Y=0, lambdaY.len=1, lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
#SPLS-GLM analysis
main3 = SPLS.binomial.GLM(scale(X), scale(y), A=2, lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-3)
X.to.use = main3$X.select
X.new = as.matrix(X[,names(X.to.use)])
colnames(X.new)
SPLS.GLM.biplot_bvec(X.new, y, algorithm=SPLS.binomial.GLM, eps=1e-3, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, ax.tickvec.b=30)
```

SPLS.GLM.biplot_no.SN *The Partial Least Squares (PLS) biplot for Sparse Partial Least Squares-Generalized Linear Model (SPLS-GLM), with the labels of the sample points excluded*

Description

Takes in a set of predictor variables and a set of response variable and produces a PLS biplot for the SPLS-GLM with the labels of the sample points excluded.

Usage

```
SPLS.GLM.biplot_no.SN(X, y, algorithm = NULL, eps, lambdaY = NULL,
  lambdaX = NULL, ax.tickvec.X = NULL, ax.tickvec.y = NULL,
  ax.tickvec.b = NULL, ...)
```

Arguments

`X`	A (NxP) predictor matrix
`y`	A (Nx1) response vector
`algorithm`	Any of the SPLS-GLM algorithm ("SPLS.GLM", "SPLS.binomial.GLM")
`eps`	Cut off value for convergence step
`lambdaY`	A value for the penalty parameters for the soft-thresholding penalization function for Y-weights
`lambdaX`	A value for the penalty parameters for the soft-thresholding penalization function for X-weights
`ax.tickvec.X`	tick marker length for each X-variable axis in the biplot
`ax.tickvec.y`	tick marker length for the y-variable axis in the biplot
`ax.tickvec.b`	(purple) tick marker length for the y-variable axis in the biplot
`...`	Other arguments. Currently ignored

Value

The PLS biplot of a SPLS-GLM of D=[X y] with some parameters

Author(s)

Opeoluwa F. Oyedele and Sugnet Gardner-Lubbe

Examples

```
if(require(robustbase))
possum.mat
y = as.matrix(possum.mat[,1], ncol=1)
dimnames(y) = list(paste("S", 1:nrow(possum.mat), seq=""), "Diversity")
X = as.matrix(possum.mat[,2:14], ncol=13)
dimnames(X) = list(paste("S", 1:nrow(possum.mat), seq=""), colnames(possum.mat[,2:14]))
#choosing a value for the penalty parameters lambdaY and lambdaX for this data
main2B = opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.GLM,
eps=1e-3, from.value.X=1, to.value.X=4, from.value.Y=0, to.value.Y=0, lambdaY.len=1,
lambdaX.len=100)
min.RMSEP.value = main2B$min.RMSEP.value
lambdaY.to.use = main2B$lambdaY.to.use
lambdaX.to.use = main2B$lambdaX.to.use
list(lambdaY.to.use=lambdaY.to.use, lambdaX.to.use=lambdaX.to.use, min.RMSEP.value=min.RMSEP.value)
#SPLS-GLM analysis
main3 = SPLS.GLM(scale(X), scale(y), A=2, lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
eps=1e-3)
X.to.use = main3$X.select
X.new = as.matrix(X[,names(X.to.use)])
colnames(X.new)
main3$Y.select #note
SPLS.GLM.biplot_no.SN(X.new, y, algorithm=SPLS.GLM, eps=1e-3, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, ax.tickvec.X=c(10,5,5,5,5,5,5,5,5,5,5,5,5), ax.tickvec.y=8,
ax.tickvec.b=12)

#Pima.tr data
if(require(MASS))
data(Pima.tr, package="MASS")
X = as.matrix(cbind(Pima.tr[,1:7]))
dimnames(X) = list(1:nrow(X), colnames(X))
y = as.matrix(as.numeric(Pima.tr$type)-1, ncol=1)
#0=No and 1=Yes
dimnames(y) = list(1:nrow(y), paste("type"))
main2 = opt.penalty.values(X=scale(X), Y=scale(y), A=2, algorithm=SPLS.binomial.GLM,
eps=1e-3, from.value.X=0, to.value.X=95, from.value.Y=0, to.value.Y=0, lambdaY.len=1,
lambdaX.len=100)
min.RMSEP.value = main2$min.RMSEP.value
lambdaY.to.use = main2$lambdaY.to.use
lambdaX.to.use = main2$lambdaX.to.use
#SPLS-GLM analysis
main3 = SPLS.binomial.GLM(scale(X), scale(y), A=2, lambdaY=lambdaY.to.use,
lambdaX=lambdaX.to.use, eps=1e-3)
X.to.use = main3$X.select
X.new = as.matrix(X[,names(X.to.use)])
colnames(X.new)
SPLS.GLM.biplot_no.SN(X.new, y, algorithm=SPLS.binomial.GLM, eps=1e-3,
lambdaY=lambdaY.to.use, lambdaX=lambdaX.to.use,
ax.tickvec.X=c(3,3,3,3,3,3,1), ax.tickvec.y=3, ax.tickvec.b=3)
```

Index