Package 'rrda'

May 9, 2026

Title Ridge Redundancy Analysis for High-Dimensional Omics Data

Version 0.2.3

Description Efficient framework for ridge redundancy analysis (rrda), tailored for high-dimensional omics datasets where the number of predictors exceeds the number of samples. The method leverages Singular Value Decomposition (SVD) to avoid direct inversion of the covariance matrix, enhancing scalability and performance. It also introduces a memory-efficient storage strategy for coefficient matrices, enabling practical use in large-scale applications. The package supports cross-validation for selecting regularization parameters and reduced-rank dimensions, making it a robust and flexible tool for multivariate analysis in omics research. Please refer to our article (Yoshioka et al., 2025) for more details.

License GPL (>= 3)

Imports dplyr, furrr, ggplot2, grDevices, MASS, pheatmap, reshape2, RSpectra, scales, stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Hayato Yoshioka [aut] (ORCID: <https://orcid.org/0000-0001-5383-2909>), Julie Aubert [aut, cre] (ORCID: <https://orcid.org/0000-0001-5203-5748>), Tristan Mary-Huard [aut] (ORCID: <https://orcid.org/0000-0002-3839-9067>)

Maintainer Julie Aubert <julie.aubert@inrae.fr>

Repository CRAN

Date/Publication 2025-10-15 12:20:02 UTC

Contents

`Bhat_mat_rlist` *Generate a list of rank-specific Bhat matrices (the coefficient of Ridge Redundancy Analysis for each parameter lambda and nrank).*

Description

Generate a list of rank-specific Bhat matrices (the coefficient of Ridge Redundancy Analysis for each parameter lambda and nrank). In our formula, Bhat is stored as LeftBhatlambda_k (p times r matrix) and RightBhatlambda_k (q times r). Here, n is the number of samples, p is the number of variables of X, q is the number of variables of Y, and r is the specified rank in the Ridge Redundancy Analysis.

Usage

```
Bhat_mat_rlist(Bhat_comp_1L, nrank)
```

Arguments

`Bhat_comp_1L`	A list containing components of Bhat for each lambda value.
`nrank`	A numeric vector indicating the rank(s) of Bhat.

Value

A list of matrices, each representing a rank-specific Bhat matrix.

get_Bhat_comp *Compute the components of the coefficient Bhat using SVD.*

Description

Compute the components of Bhat using SVD. In our formula, Bhat is stored as LeftBhatlambda_k (p times r matrix) and RightBhatlambda_k (q times r). Here, n is the number of samples, p is the number of variables of X, q is the number of variables of Y, and r is the specified rank in the Ridge Redundancy Analysis.

Usage

```
get_Bhat_comp(rsid2_1L, DUtY, nrank, tV)
```

Arguments

rsid2_1L	A numeric vector used for each lambda value.
DUtY	A numeric matrix (n times q).
nrank	A numeric vector indicating the rank(s) of Bhat.
tV	A numeric matrix.

Value

A list containing the left and right components of Bhat (LeftBhatlambda_k , RightBhatlambda_k and singular values (Bd) for GSVD of Bhat).

get_lambda *Estimate an appropriate value for the ridge penalty (lambda).*

Description

Estimate an appropriate value for the ridge penalty (lambda).

Usage

```
get_lambda(
  Y,
  X,
  scale.X = FALSE,
  sample.X = 1000,
  sample.Y = 1000,
  num.lambda = 50
)
```

Arguments

Y	A numeric matrix of response variables.
X	A numeric matrix of explanatory variables.
scale.X	Logical indicating if X is scaled. Default is FALSE.
sample.X	A number of variables sampled from X for the lamdba range estimate. Default is 1000.
sample.Y	A number of variables sampled from Y for the lamdba range estimate. Default is 1000.
num.lambda	A number of lambda generated. Default is 50.

Value

A numeric vector of the range of lambda values.

---

get_rlist *Generate rank-specific matrices by combining the left and right components.*

Description

Generate rank-specific matrices by combining the left and right components of the coefficeint Bhat. In our formula, Bhat is stored as LeftBhatlambda_k (the left component vectors, p times r matrix) and RightBhatlambda_k (the right component vectors, q times r) for each lambda value. Here, n is the number of samples, p is the number of variables of X, q is the number of variables of Y, and r is the specified rank in the Ridge Redundancy Analysis.

Usage

```
get_rlist(LV, RV, nrank)
```

Arguments

LV	A numeric matrix of the left component vectors.
RV	A numeric matrix of the right component vectors.
nrank	A numeric vector indicating the rank(s) of Bhat.

Value

A list of matrices of rank-specific combinations of LV and RV.

MSE_lambda_rank *Compute MSE for different ranks of the coefficient Bhat and lambda.*

Description

For cross-validation, compute MSE for different ranks of the coefficient Bhat and lambda.

Usage

```
MSE_lambda_rank(Bhat_comp_1L, X, Y, nrank)
```

Arguments

Bhat_comp_1L	A list containing components of Bhat.
X	A numeric matrix of explanatory variables.
Y	A numeric matrix of response variables.
nrank	A numeric vector indicating the rank(s) of Bhat.

Value

A numeric vector of MSE values for each rank.

rdasim1 *Generate simulated data for Ridge Redundancy Analysis (RDA).*

Description

The function generates simulated data for Ridge Redundancy Analysis (RDA). It creates two data matrices, X and Y, based on a set of shared latent variables H. The function adds noise to the data and returns a list containing the matrices X, Y, the latent variables H, and the regression coefficients theta.y used for generating Y.

Usage

```
rdasim1(n, p, q, k, s2n = c(5, 5))
```

Arguments

n	The number of samples.
p	The number of variables of X.
q	The number of variables of Y.
k	The number of latent variables.
s2n	The numeric parameters of signal to noise ratio for X and Y, default value is c(1,1).

Value

A list containing matrices X, Y, H, and theta.y.

Examples

```
# Example usage of rdasim1
set.seed(10)
sim_data <- rdasim1(n = 10, p = 5, q = 3, k = 2)
str(sim_data)
```

`rdasim2` *Generate simulated data for Ridge Redundancy Analysis (RDA).*

Description

The rdasim2 function generates simulated data for Ridge Redundancy Analysis (RDA) with adjustable signal-to-noise ratio and covariance structure for X. The data matrix Y is created by a low-rank model, where the rank is set by the product of two matrices A and C corresponding to the number of latent variables (k). The function allows control over the signal-to-noise ratio (s2n) and off-diagonal elements of the covariance matrix for X (xofd). It returns a list containing the matrices X, Y, the regression coefficient matrix B (obtained as the product of A and C), and the error matrix E.

Usage

```
rdasim2(n, p, q, k, s2n = 5, xofd = 0)
```

Arguments

`n`	The number of samples.
`p`	The number of variables of X.
`q`	The number of variables of Y.
`k`	The number of latent variables.
`s2n`	The numeric parameter of signal to noise ratio, default value is 5.
`xofd`	The numeric parameter of the off-diagnal elements of covariance matrix of X, default is 0.

Value

A list containing matrices X, Y, B, E.

Examples

```
# Example usage of rdasim2
set.seed(10)
sim_data2 <- rdasim2(n = 10, p = 5, q = 3, k = 2)
str(sim_data2)
```

`rrda.coef` *Calculate the Bhat matrix from the return of the* `rrda.fit` *function.*

Description

This function calculates the coefficient Bhat (the coefficient of Ridge Redundancy Analysis for each parameter lambda and nrank) as a matrix form by using the Bhat components calculated by the rrda.fit function. This function obtain the matrix form of Bhat as follows

$$\hat{B}(\lambda, r) = FG'$$

Here, the Bhat components F and G are obtained from the `rrda.fit` function as follows

For $i = 1, \ldots, r$, the matrices F and G are defined as:

$$F_{.i} = U^{[i]}_{\hat{B}(\lambda)} D^{[i]}_{\hat{B}(\lambda)}, \quad G_{.i} = V^{[i]}_{\hat{B}(\lambda)}$$

If the input already contains Bhat as matrix form (Bhat_mat), the function selects the preferred matrix from the list of Bhat matrices.

The function can handle different ranks (nrank) and ridge penalty values (lambda) based on the input. If nrank or lambda is NULL, the function will use the values from the Bhat components. Note that if lambda = NULL and B matrix is large (nrow(B)*ncol(B) > 100000), the function is performed for the minimum lambda value only.

Usage

```
rrda.coef(Bhat, nrank = NULL, lambda = NULL)
```

Arguments

`Bhat`	A list of vectors of Bhat components, obtained by the `rrda.fit` function. If Bhat_mat is detected in Input, it selects the preferred matrix from the list of Bhat.
`nrank`	A numeric vector specifying the ranks of Bhat. Default is `NULL`, which sets it to the ranks defined in the Bhat components.
`lambda`	A numeric vector of ridge penalty values. Default is `NULL`, which sets it to the lambda values defined in the Bhat components.

Value

A list containing the Bhat matrix

Examples

```
set.seed(10)
simdata<-rdasim1(n = 100,p = 200,q = 200,k = 5)
X <- simdata$X
Y <- simdata$Y

Bhat <- rrda.fit(Y = Y, X = X, nrank = c(1:10))
Bhat_mat <- rrda.coef(Bhat = Bhat, nrank = 10)
```

`rrda.cv` *Cross-validation for Ridge Redundancy Analysis*

Description

This function performs cross-validation to evaluate the performance of Ridge Redundancy Analysis (RDA) models. It calculates the mean squared error (MSE) for different ranks and ridge penalty values through cross-validation folds. The function also supports centering and scaling of the input matrices.

The range of lambda for the cross-validation is automatically calculated following the method of "glmnet" (Friedman et al., 2010). When we have a matrix of response variables (Y; n times q matrix) and a matrix of explanatory variables (X; n times p matrix), the largest lambda for the validation is obtained as follows

$$\lambda_{\text{max}} = \frac{\max_{j \in \{1,2,\ldots,p\}} \sqrt{\sum_{k=1}^{q} \left(\sum_{i=1}^{n} (x_{ij} \cdot y_{ik})\right)^2}}{N \times 10^{-3}}$$

Then, we define $\lambda_{min} = 10^{-4}\lambda_{max}$, and the sequence λ is generated based on the range.

Also, to reduce the computation, the variable sampling is performed for the large matrix of X and Y (by default, when the number of the variables is over 1000). Alternatively, the range of lambda can be specified manually.

Usage

```
rrda.cv(
  Y,
  X,
  maxrank = NULL,
  lambda = NULL,
  num.lambda = 50,
  nfold = 5,
  folds = NULL,
  sample.X = 1000,
  sample.Y = 1000,
  scale.X = FALSE,
  scale.Y = FALSE,
  center.X = TRUE,
```

```
  center.Y = TRUE,
  verbose = TRUE
)
```

Arguments

Y	A numeric matrix of response variables.
X	A numeric matrix of explanatory variables.
maxrank	A numeric vector specifying the maximum rank of the coefficient Bhat. Default is NULL, which sets it to (min(15, min(dim(X), dim(Y)))).
lambda	A numeric vector of ridge penalty values. Default is NULL, where the lambda values are automatically chosen.
num.lambda	A number of lambda generated (only when the lambda is not given by user). Default is 50.
nfold	The number of folds for cross-validation. Default is 5.
folds	A vector specifying the folds. Default is NULL, which randomly assigns folds.
sample.X	A number of variables sampled from X for the lamdba range estimate. Default is 1000.
sample.Y	A number of variables sampled from Y for the lamdba range estimate. Default is 1000.
scale.X	Logical indicating if X should be scaled. If TRUE, scales X. Default is FALSE.
scale.Y	Logical indicating if Y should be scaled. If TRUE, scales Y. Default is FALSE.
center.X	Logical indicating if X should be centered. If TRUE, scales X. Default is TRUE.
center.Y	Logical indicating if Y should be centered. If TRUE, scales Y. Default is TRUE.
verbose	Logical indicating. If TRUE, the function displays information about the function call. Default is TRUE.

Value

A list containing the cross-validated MSE matrix, lambda values, rank values, and the optimal lambda and rank.

Examples

```
## Not run:
set.seed(10)
simdata<-rdasim1(n = 100,p = 200,q = 200,k = 3)
X <- simdata$X
Y <- simdata$Y
cv_result<- rrda.cv(Y = Y, X = X, maxrank = 5, nfold = 5)
rrda.summary(cv_result = cv_result)

##Complete Example##
```

```
# library(future) # <- if you want to compute in parallel

# plan(multisession) # <- if you want to compute in parallel
# cv_result<- rrda.cv(Y = Y, X = X, maxrank = 5, nfold = 5) # cv
# plan(multisession) # <- To come back to sequential computing

# rrda.summary(cv_result = cv_result) # cv result

p <- rrda.plot(cv_result) # cv result plot
print(p)
h <- rrda.heatmap(cv_result) # cv result heatmao
print(h)

estimated_lambda<-cv_result$opt_min$lambda  # selected parameter
estimated_rank<-cv_result$opt_min$rank # selected parameter

Bhat <- rrda.fit(Y = Y, X = X, nrank = estimated_rank,lambda = estimated_lambda) # fitting
Bhat_mat<-rrda.coef(Bhat)
Yhat_mat <- rrda.predict(Bhat = Bhat, X = X) # prediction
Yhat<-Yhat_mat[[1]][[1]][[1]] # predicted values

cor_Y_Yhat<-diag(cor(Y,Yhat)) # correlation
summary(cor_Y_Yhat)

## End(Not run)
```

`rrda.fit` *Calculate the coefficient Bhat by Ridge Redundancy Analysis.*

Description

This function performs Ridge Redundancy Analysis (RRDA) to obtain the coefficient Bhat, which models the relationship between a matrix of response variables (Y; $n \times q$ matrix) and a matrix of explanatory variables (X; $n \times p$ matrix). Especially, the function is designed to facilitate a high-dimensional computation and storing (for the details, refer to the article Yoshioka et al. 2025).

The Ridge Redundancy Analysis model is represented as:

$$Y = XB + E$$

where:

- Y is the response matrix ($n \times q$),
- X is the predictor matrix ($n \times p$),
- B is the regression coefficient matrix ($p \times q$), and
- E is the error matrix ($n \times q$).

The regularized estimate of B is described as:

$$\hat{B}(\lambda) = (X'X + \lambda P_{X'})^{-} X'Y$$

Additionally, the regularized-rank-restricted estimation of B is represented as:

$$\hat{B}(\lambda, r) = U^{[r]}_{\hat{B}(\lambda)} D^{[r]}_{\hat{B}(\lambda)} V^{[r]'}_{\hat{B}(\lambda)}$$

Here:

- $U^{[r]}_{\hat{B}(\lambda)}$ is a $p \times r$ matrix,
- $D^{[r]}_{\hat{B}(\lambda)}$ is a $r \times r$ diagonal matrix, and
- $V^{[r]}_{\hat{B}(\lambda)}$ is a $q \times r$ matrix.

The user can specify ranks (nrank), ridge penalty values (lambda), and whether to center and scale the X and Y matrices.

The Bhat can be returned as either component vectors or matrices. To store a large size of matrix, the coefficient Bhat is by default stored as LeftBhatlambda_k (F; $p \times r$ matrix) and RightBhatlambda_k (G; $q \times r$). Here, r is the specified rank (nrank) in the Ridge Redundancy Analysis formula.

For $i = 1, \ldots, r$, the matrices F and G are defined as:

$$F_{.i} = U^{[i]}_{\hat{B}(\lambda)} D^{[i]}_{\hat{B}(\lambda)}, \quad G_{.i} = V^{[i]}_{\hat{B}(\lambda)}$$

These definitions allow the decomposition of $\hat{B}(\lambda)$ into rank-specific components, facilitating the storing of the high-dimensional regression coefficients. To reconstruct the matrix form of Bhat, you can use the `rrda.coef` function.

Usage

```
rrda.fit(
  Y,
  X,
  nrank = NULL,
  lambda = 1,
  component = TRUE,
  center.X = TRUE,
  center.Y = TRUE,
  scale.X = FALSE,
  scale.Y = FALSE
)
```

Arguments

`Y`	A numeric matrix of response variables.
`X`	A numeric matrix of explanatory variables.
`nrank`	A numeric vector specifying the ranks of Bhat. Default is `NULL`, which sets it to `(1:min(15, min(dim(X), dim(Y))))`.
`lambda`	A numeric vector of ridge penalty values. Default value is 1.
`component`	Logical indicating if Bhat is returned as vectors or matrices. If `TRUE`, returns Bhat as component vectors. If `FALSE`, returns Bhat as matrices.

center.X	Logical indicating if X should be centered. If TRUE, scales X. Default is TRUE.
center.Y	Logical indicating if Y should be centered. If TRUE, scales Y. Default is TRUE.
scale.X	Logical indicating if X should be scaled. If TRUE, scales X. Default is FALSE.
scale.Y	Logical indicating if Y should be scaled. If TRUE, scales Y. Default is FALSE.

Value

A list containing Bhat components or Bhat matrices (the coefficient of Ridge Redundancy Analysis for each parameter lambda and nrank), ranks, and lambda values.

Examples

```
set.seed(10)
simdata<-rdasim1(n = 100,p = 200,q = 200,k = 5)
X <- simdata$X
Y <- simdata$Y

# Sequential
Bhat <- rrda.fit(Y = Y, X = X, nrank = c(1:10))
names(Bhat)
```

---

rrda.heatmap *Heatmap of the results of cross-validation for Bhat obtained from the* rrda.cv *function.*

Description

This function creates a heatmap to visualize the Mean Squared Error (MSE) results from the cross-validation of the Bhat matrix obtained from the rrda.cv function. The heatmap displays the MSE for different ranks of Bhat and values of the regularization parameter lambda, allowing users to visually assess the best combination of rank and lambda. The function also allows the user to highlight the points corresponding to the minimum MSE and the 1-standard error rule, helping to identify optimal model parameters.

Usage

```
rrda.heatmap(
  cv_result,
  nrank = NULL,
  min_l = NULL,
  max_l = NULL,
  highlight_min = TRUE,
  title = NULL
)
```

Arguments

cv_result	A result list from the function rrda.cv, containing a matrix of MSE values for each rank and lambda, and a vector of lambda values.
nrank	A numeric vector specifying the ranks of Bhat to be plotted. Default is NULL, which plots all ranks.
min_l	Minimum lambda value to be plotted. Default is NULL, which uses the minimum lambda value in cv_result.
max_l	Maximum lambda value to be plotted. Default is NULL, which uses the maximum lambda value in cv_result.
highlight_min	Logical indicating if the marks should be plotted on the best prediction point, and 1se point. Default is TRUE.
title	Title of the figure

Value

A heatmap of MSE cross-validation results.

Examples

```
## Not run:
set.seed(10)
simdata<-rdasim1(n = 100,p = 200,q = 200,k = 3) # data generation
X <- simdata$X
Y <- simdata$Y

cv_result<- rrda.cv(Y = Y, X = X, maxrank = 5, nfold = 5) # cv
rrda.summary(cv_result = cv_result)
rrda.heatmap(cv_result=cv_result)

## End(Not run)
```

---

rrda.plot *Plot the results of cross-validation for Bhat obtained from the* rrda.cv *function.*

Description

This function visualizes the results of cross-validation for the estimated Bhat matrix obtained from the rrda.cv function. It creates a plot of the Mean Squared Error (MSE) for each combination of rank and lambda regularization parameter, allowing for the selection of specific ranks and lambda ranges to be plotted. Error bars representing the standard error of the MSE can be displayed for the best rank.

Usage

```
rrda.plot(
  cv_result,
  nrank = NULL,
  min_l = NULL,
  max_l = NULL,
  show_error_bar = FALSE,
  title = NULL
)
```

Arguments

cv_result	A result list from the function rrda.cv, containing a matrix of MSE values for each rank and lambda, and a vector of lambda values.
nrank	A numeric vector specifying the ranks of Bhat to be plotted. Default is NULL, which plots all ranks.
min_l	Minimum lambda value to be plotted. Default is NULL, which uses the minimum lambda value in cv_result.
max_l	Maximum lambda value to be plotted. Default is NULL, which uses the maximum lambda value in cv_result.
show_error_bar	Logical value indicating if the error bar is shown on the line that gives the best MSE value.
title	Title of the figure

Value

A plot of MSE cross-validation results.

Examples

```
## Not run:
set.seed(10)
simdata<-rdasim1(n = 100,p = 200,q = 200,k = 3) # data generation
X <- simdata$X
Y <- simdata$Y

cv_result<- rrda.cv(Y = Y, X = X, maxrank = 5, nfold = 5) # cv
rrda.summary(cv_result = cv_result)
rrda.plot(cv_result = cv_result)

## End(Not run)
```

`rrda.predict` *Calculate the predicted matrix Yhat using the coefficient Bhat obtained from the* `rrda.fit` *function.*

Description

This function calculates the predicted response matrix (Yhat) using Bhat (the coefficient of Ridge Redundancy Analysis) obtained from the rrda.fit function. The user can specify the ranks and lambda values to be used for the prediction. If not provided, the ranks and lambda values are set based on the Bhat components. The function returns the predicted Yhat matrix, as well as the ranks and lambda values used.

The predicted response matrix is calculated as:

$$\hat{Y}_r = X\left(\sum_{i=1}^{r}(F_{.i}G'_{.i})\right) = \sum_{i=1}^{r}(XF_{.i}G'_{.i}) = \sum_{i=1}^{r}(\tilde{X}_{.i}G'_{.i})$$

Here, $\tilde{X} = XF$, and r is the rank of $\hat{B}(\lambda, r)$. The regularized-rank-restricted estimation of B is obtained from the `rrda.fit` function:

$$\hat{B}(\lambda, r) = FG'$$

Usage

```
rrda.predict(Bhat, X, nrank = NULL, lambda = NULL)
```

Arguments

`Bhat`	A list of vectors of Bhat components, obtained by the `rrda.fit` function.
`X`	A numeric matrix of explanatory variables.
`nrank`	A numeric vector specifying the ranks of Bhat. Default is `NULL`, which sets it to the ranks defined in the Bhat components.
`lambda`	A numeric vector of ridge penalty values. Default is `NULL`, which sets it to the lambda values defined in the Bhat components.

Value

A list containing the predicted Yhat matrix, ranks, and lambda values.

Examples

```
set.seed(10)
simdata<-rdasim1(n = 100,p = 200,q = 200,k = 5)
X <- simdata$X
Y <- simdata$Y
Bhat <- rrda.fit(Y = Y, X = X, nrank = c(1:10))
Yhat_mat <- rrda.predict(Bhat = Bhat, X = X, nrank = 10)
```

`rrda.summary` *Summarize the results of cross-validation for the coefficient Bhat obtained from the* `rrda.cv` *function.*

Description

The function provides a summary of the results from cross-validation for Bhat obtained using the rrda.cv function. It displays the Mean Squared Error (MSE), rank, and lambda values for different cross-validation options.

Usage

```
rrda.summary(cv_result = cv_result)
```

Arguments

`cv_result`	A result list from the function `rrda.cv`, containing a matrix of MSE values for each rank and lambda, and a vector of lambda values.

Value

Prints the estimated Coefficients, rank, lambda, and MSE from cross-validation.

`rrda.top` *Top feature interactions visualization with rank and lambda penalty*

Description

Visualizes the most influential feature interactions (based on the L2 norm) from Ridge Redundancy Analysis (RRDA) as a heatmap.

Let the (rank-r truncated) decomposition of $\hat{B}(\lambda)$ be

$$\hat{B}(\lambda, r) = U_{\hat{B}(\lambda)}\, D_{\hat{B}(\lambda)}\, V'_{\hat{B}(\lambda)}.$$

The following three biplot scalings are defined:

Symmetric scaling (default):

$$\tilde{F} = U_{\hat{B}(\lambda)}\, D^{1/2}_{\hat{B}(\lambda)}, \qquad \tilde{G} = V_{\hat{B}(\lambda)}\, D^{1/2}_{\hat{B}(\lambda)}.$$

X scaling:

$$\tilde{F} = U_{\hat{B}(\lambda)}\, D_{\hat{B}(\lambda)}, \qquad \tilde{G} = V_{\hat{B}(\lambda)}.$$

Y scaling:

$$\tilde{F} = U_{\hat{B}(\lambda)}, \qquad \tilde{G} = V_{\hat{B}(\lambda)}\, D_{\hat{B}(\lambda)}.$$

In all three cases, $\hat{B}(\lambda, r) = \tilde{F}\,\tilde{G}'$.

Variable importance is scored by the row-wise ℓ_2-norms:

$$s_i^{(\tilde{F})} = \|\tilde{F}_{i,\cdot}\|_2, \qquad s_j^{(\tilde{G})} = \|\tilde{G}_{j,\cdot}\|_2.$$

Selecting the top m_x predictors and m_y responses yields the submatrices of the scaled factor matrices (each with r columns).

The reduced coefficient submatrix is then

$$\hat{B}_{\text{sub}}(\lambda, r) = \tilde{F}_{\text{sub}}\,\tilde{G}'_{\text{sub}}.$$

The matrix $\hat{B}_{\text{sub}}(\lambda, r)$ retains the dominant low-rank structure and is visualized as a heatmap (with $m_x = m_y = 20$ by default).

Usage

```
rrda.top(
  Y,
  X,
  nrank = NULL,
  lambda = NULL,
  mx = 20,
  my = 20,
  scaling = c("symmetric", "none", "x", "y"),
  title = TRUE
)
```

Arguments

`Y`	A numeric matrix of response variables.
`X`	A numeric matrix of explanatory variables.
`nrank`	Integer rank r of $\hat{B}$ to visualize. If `NULL` (default), it is set to `min(5, min(dim(X), dim(Y)))`.
`lambda`	A numeric vector of ridge penalty values. If `NULL` (default), it is set to 1.
`mx`	Integer; number of top X-features (predictors) to display. Defaults to `20`.
`my`	Integer; number of top Y-features (responses) to display. Defaults to `20`.
`scaling`	Character string specifying how to apply the singular values from the compositions of $\hat{B}(\lambda)$ when constructing the biplot factors. Options are: `"symmetric"` (default) distributes singular values evenly to both sides (balanced scaling), `"x"` applies them fully to the X (left) side, `"y"` applies them fully to the Y (right) side, and `"none"` removes them (no singular value weighting).
`title`	Figure title. If `TRUE` (default), a formatted title is used. If `FALSE` or `NULL`, no title is drawn. If a single string, it is passed through to the figure title.

Value

A list with elements: `heatmap` (pheatmap object), `B_sub` (mx x my matrix), `top_x`, `top_y`, `b1_sub`, `b2_sub`, `fit`, `scaling`.

Examples

```
set.seed(10)
simdata<-rdasim1(n = 10,p = 50,q = 50,k = 3) # data generation
X <- simdata$X
Y <- simdata$Y
rrda.top(Y=Y,X=X,nrank=5,lambda=1,mx=20,my=20)

## Not run:
### In practice, the parameters nrank and lambda should be selected by CV ###
cv_result<- rrda.cv(Y = Y, X = X, maxrank = 5, nfold = 5) # cv
best_lambda<-cv_result$opt_min$lambda
best_rank<-cv_result$opt_min$rank
rrda.summary(cv_result = cv_result)

rrda.top(Y=Y,X=X,nrank=best_rank,lambda=best_lambda,mx=20,my=20)

## End(Not run)
```

---

sqrt_inv_d2_lambda *Compute the square root of the inverse of (d^2 + lambda).*

Description

Compute the square root of the inverse of (d^2 + lambda).

Usage

```
sqrt_inv_d2_lambda(d, lambda)
```

Arguments

d	A scalar of singular value of X.
lambda	A scalar of the ridge penalty.

Value

A scalar of the square root of the inverse of (d^2 + lambda).

`unbiased_scale` *Scale a matrix using unbiased estimators for the mean and standard deviation.*

Description

Scale a matrix using unbiased estimators for the mean and standard deviation.

Usage

```
unbiased_scale(x)
```

Arguments

`x`	A numeric matrix to be scaled.

Value

A scaled numeric matrix.

`unscale_matrices` *Unscale a matrix based on provided mean and standard deviation values.*

Description

Unscale a matrix based on provided mean and standard deviation values.

Usage

```
unscale_matrices(mat, means, std = NULL)
```

Arguments

`mat`	A numeric matrix to be unscaled.
`means`	A numeric vector of means for each column.
`std`	A numeric vector of standard deviations for each column (optional).

Value

An unscaled numeric matrix.

`unscale_nested_matrices_map` *Apply unscaling to a nested list of matrices using specified mean and standard deviation values.*

Description

Apply unscaling to a nested list of matrices using specified mean and standard deviation values.

Usage

```
unscale_nested_matrices_map(mat, means, std = NULL)
```

Arguments

`mat`	A nested list of matrices.
`means`	A numeric vector of means for each matrix.
`std`	A numeric vector of standard deviations for each matrix (optional).

Value

A nested list of unscaled matrices.

`Yhat_mat_rlist` *Generate a list of rank-specific Yhat matrices.*

Description

Generate a list of rank-specific Yhat matrices.

Usage

```
Yhat_mat_rlist(Bhat_comp_1L, X, nrank)
```

Arguments

`Bhat_comp_1L`	A list containing components of Bhat.
`X`	A numeric matrix of explanatory variables.
`nrank`	A numeric vector indicating the rank(s) of Bhat.

Value

A list of matrices, each representing rank-specific predicted values Yhat.

Index