Package ‘MOSS’

March 25, 2022

Title Multi-Omic Integration via Sparse Singular Value Decomposition

Version 0.2.2

Description High dimensionality, noise and heterogeneity among samples and features challenge the omic integration task. Here we present an omic integration method based on sparse singular value decomposition (SVD) to deal with these limitations, by: a. obtaining the main axes of variation of the combined omics, b. imposing sparsity constraints at both subjects (rows) and features (columns) levels using Elastic Net type of shrinkage, and c. allowing both linear and non-linear projections (via t-Stochastic Neighbor Embedding) of the omic data to detect clusters in very convoluted data (Gonzalez-Reymundez et al, 2022) <doi:10.1093/bioinformatics/btac179>.

License GPL-2

URL https://github.com/agugonrey/MOSS

BugReports https://github.com/agugonrey/MOSS/issues

Imports cluster, dbscan, Rtsne, stats

Suggests annotate, bigparallelr, bigstatsr, future.apply, scatterpie, cValid, ComplexHeatmap, fpc, ggplot2, ggpmisc, ggthemes, gridExtra, irlba, knitr, MASS, rmarkdown, testthat, viridis, spelling, VennDiagram, grid

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.1

Language en-US

NeedsCompilation no

Author Agustin Gonzalez-Reymundez [aut, cre, cph], Alexander Grueneberg [aut], Ana Vazquez [ctb, ths]

Maintainer Agustin Gonzalez-Reymundez <agugonrey@gmail.com>

Repository CRAN

Date/Publication 2022-03-25 15:50:05 UTC
\textbf{R topics documented:}

\begin{itemize}
  \item \texttt{aest.f} \hfill 2
  \item \texttt{cov_adj} \hfill 3
  \item \texttt{metdat} \hfill 4
  \item \texttt{moss} \hfill 4
  \item \texttt{moss_heatmap} \hfill 10
  \item \texttt{moss_select} \hfill 11
  \item \texttt{moss_signatures} \hfill 12
  \item \texttt{moss_venn} \hfill 13
  \item \texttt{pca2tsne} \hfill 14
  \item \texttt{prepro_na} \hfill 15
  \item \texttt{prepro_sub} \hfill 16
  \item \texttt{simulate_data} \hfill 17
  \item \texttt{ssvdEN} \hfill 18
  \item \texttt{ssvdEN_sol_path} \hfill 22
  \item \texttt{ssvdEN_sol_path_par} \hfill 25
  \item \texttt{tsne2clus} \hfill 27
\end{itemize}

Index \hfill 31

\begin{itemize}
  \item \texttt{aest.f} \hfill Assign point color and shape aesthetics.
\end{itemize}

\textbf{Description}

This function is called by moss whenever a plot is produced. It simply assigns colors and shape to points based on input labels.

\textbf{Usage}

\texttt{aest.f(x, n.cat = 2, option = "D")}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} \hfill Character vector with labels, or a numerical vector to be discretized in \texttt{`n.cat`} categories.
  \item \texttt{n.cat} \hfill Number of categories to split vector \texttt{`x`}. Numeric. Ignored if \texttt{`x`} is a character vector.
  \item \texttt{option} \hfill Controls color palette. One of the possible \texttt{`option`} arguments for the \texttt{`viridis`} function.
\end{itemize}

\textbf{Value}

A data.frame with labels as rownames and two columns representing point colors and shape, respectively.
cov_adj

Adjust omic blocks for covariates effects.

Description

This function is called by moss to adjust a series of omic blocks for covariates effects. However, if the covariates object is too big, the user is recommended to call cov_adj ahead of moss.

Usage

cov_adj(data.blocks, covs, n, dim.names = NULL)

Arguments

data.blocks List containing omic blocks of class 'matrix' or 'FBM'. In each block, rows represent subjects and columns features.
covs Covariates which effect we wish to adjust for. Accepts objects of class matrix, data.frame, numeric, or character vectors.
n Number of subjects. Numeric.
dim.names list of vectors with samples names, and features names by omic block. If NULL, a list of artificial names is created. Defaults to NULL.

Value

Returns the covariates-adjusted elements in data.blocks.

Examples

library("MOSS")
sim_data <- simulate_data()
set.seed(43)

# Extracting simulated omic blocks.
sim_blocks <- sim_data$sim_blocks[-4]

# Using fourth block as covariates.
covs <- sim_data$sim_blocks[[4]]

# Adjust omic blocks for covariates effects.
sim_blocks_adj <- cov_adj(sim_blocks, covs, nrow(covs))
**metdat**

*Extracts (and merges) chunks of characters.*

**Description**

Extracts (and merges) chunks of characters.

**Usage**

```r
metdat(x, i, sep = "-", collapse = sep)
```

**Arguments**

- `x`: A character vector.
- `i`: Index specifying which chunks of characters will be extracted (and merged).
- `sep`: Chunks separator character. Defaults to "-".
- `collapse`: New chunks separator character. Default to ‘sep’.

**Value**

A character vector with the extracted (and merged) chunks of characters.

**Examples**

```r
x <- "this is one chunk of characters & this is another one"
metdat(x, 1, " & ")
metdat(x, 2, " & ")
metdat(x, c(1, 2), " & ")
metdat(x, c(1, 2), " & ", " and ")
```

---

**moss**

*Multi-Omic integration via Sparse Singular value decomposition.*

**Description**

This function integrates omic blocks to perform sparse singular value decomposition (SVD), non-linear embedding, and/or cluster analysis. Both supervised and unsupervised methods are supported. In both cases, if multiple omic blocks are used as predictors, they are concatenated and normalized to form an ‘extended’ omic matrix ‘X’ (Gonzalez-Reymundez and Vazquez, 2020). Supervised analysis can be obtained by indicating which omic block defines a multivariate response ‘Y’. Each method within MOSS returns a matrix ‘B’, which form depends on the technique used (e.g. \( B = X \) in pca; \( B = X'Y \), for pls; \( B = (X'X)^{-1}X'Y \), for lrr). A sparse SVD of matrix B is then obtained to summarize the variability among samples and features in terms of latent factors.
moss

Usage

moss(
  data.blocks,
  scale.arg = TRUE,
  norm.arg = TRUE,
  method = "pca",
  resp.block = NULL,
  covs = NULL,
  K.X = 5,
  K.Y = K.X,
  verbose = TRUE,
  nu.parallel = FALSE,
  nu.u = NULL,
  nu.v = NULL,
  alpha.u = 1,
  alpha.v = 1,
  plot = FALSE,
  cluster = FALSE,
  clus.lab = NULL,
  tSNE = FALSE,
  axes.pos = seq_len(K.Y),
  approx.arg = FALSE,
  exact.dg = FALSE,
  use.fbm = FALSE,
  lib.thresh = TRUE
)

Arguments

data.blocks List containing omic blocks of class ‘matrix’ or ‘FBM’. In each block, rows represent subjects and columns features.
scale.arg Should the omic blocks be centered and scaled? Logical. Defaults to TRUE.
norm.arg Should omic blocks be normalized? Logical. Defaults to TRUE.
resp.block What block should be used as response? Integer. Only used when the specified method is supervised.
covs Covariates which effect we wish to adjust for. Accepts matrix, data.frame, numeric, or character vectors.
K.X Number of principal components for predictors. Integer. Defaults to 5.
K.Y Number of responses PC index when method is supervised. Defaults to K.X.
verbose Should we print messages? Logical. Defaults to TRUE.
nu.parallel Tuning degrees of sparsity in parallel. Defaults to FALSE.
uu.u A grid with increasing integers representing degrees of sparsity for left Eigenvectors. Defaults to NULL.
nu.v        Same but for right Eigenvectors. Defaults to NULL.
alpha.u     Elastic Net parameter for left Eigenvectors. Numeric between 0 and 1. Defaults to 1.
alpha.v     Elastic Net parameter for right Eigenvectors. Numeric between 0 and 1. Defaults to 1.
plot        Should results be plotted? Logical. Defaults to FALSE.
cluster     Arguments passed to the function tsne2clus as a list. Defaults to FALSE. If cluster=TRUE, default parameters are used (eps_range=c(0,4), eps_res=100).
clus.lab    A vector of same length than number of subjects with labels used to visualize clusters. Factor. Defaults to NULL. When sparsity is imposed on the left Eigenvectors, the association between non-zero loadings and labels’ groups is shown by a Chi-2 statistics for each pc. When sparsity is not imposed, the association between labels and PC is addressed by a Kruskal-Wallis statistics.
tSNE        Arguments passed to the function pca2tsne as a list. Defaults to FALSE. If tSNE=T, default parameters are used (perp=50,n.samples=1,n.iter=1e3).
axes.pos    PC index used for tSNE. Defaults to 1 : K.Y. Used only when tSNE is different than NULL.
approx.arg  Should we use standard SVD or random approximations? Defaults to FALSE. If TRUE and at least one block is of class 'matrix', irlba is called. If TRUE & is(O,'FBM')==TRUE, big_randomSVD is called.
exact.dg    Should we compute exact degrees of sparsity? Logical. Defaults to FALSE. Only relevant When alpha.s or alpha.f are in the (0,1) interval and exact.dg = TRUE.
use.fbm     Should we treat omic blocks as Filed Backed Matrix (FBM)? Logical. Defaults to FALSE.
lib.thresh  Should we use a liberal or conservative threshold to tune degrees of sparsity? Logical. Defaults to TRUE.

Details

Once 'dense' solutions are found (the result of SVD on a matrix B), the function ssvdEN_sol_path is called to perform sparse SVD (sSVD) on a grid of possible degrees of sparsity (nu), for a possible value of the elastic net parameter (alpha). The sSVD is performed using the algorithm of Shen and Huang (2008), extended to include Elastic Net type of regularization. For one latent factor (rank 1 case), the algorithm finds vectors u and v’ and scalar d that minimize:

\[ \|B - d*uv'\|^2 + \lambda(nu_v)(\alpha_v\|v'\|_1 + (1-\alpha_v)\|v'\|^2) + \lambda(nu_u)(\alpha_u\|u\|_1 + (1-\alpha_u)\|u\|^2) \]

such that \|u\|_1 = 1. The right Eigenvector is obtained from v / \|v\| and the corresponding d from u'Bv. The element \lambda(nu_.) is a monotonically decreasing function of nu_. (the number of desired element different from zero) onto positive real numbers, and alpha_. is any number between zero and one balancing shrinking and variable selection. Selecting degree of sparsity: The function allows to tune the degree of sparsity using an ad-hoc method based on the one presented in Shen & Huang (2008, see reference) and generalized for tuning both nu_v and nu_u. This is done by exploring the proportion of explained variance (PEV) on a grid of possible values. Drastic and/or
steep changes in the PEV trajectory across degrees of sparsity are used for automatic selection (see help for the function ssvdEN_sol_path). By imposing the additional assumption of omic blocks being conditionally independent, each multivariate technique can be extended using a ‘multi-block’ approach, where the contribution of each omic block to the total (co)variance is addressed. When response Y is a character column matrix, with classes or categories by subject, each multivariate technique can be extended to perform linear discriminant analysis.

Value

Returns a list with the results of the sparse SVD. If plot=TRUE, a series of plots is generated as well.

- **B**: The object of the (sparse) SVD. Depending of the method used, B can be a extended matrix of normalized omic blocks, a variance-covariance matrix, or a matrix of regression coefficients. If at least one of the blocks in 'data.blocks' is of class FBM, is(B,'FBM') is TRUE. Otherwise, is(B,'matrix') is TRUE.
- **Q**: Matrix with the SVD projections at the level of subjects.
- **selected_items**: List containing the position, name, and loadings of selected features and subjects by latent dimension. if 'plot=TRUE', a scatterplot is displayed, where the x-axis represents the latent dimensions, the y-axis the total number of features selected in log scale, and each point is a pie chart showing the relative contribution of each omic to the number of features selected. The radio of the pie-chart represents the coefficient of variation among squared loadings (mean squared loadings divided by their standard deviation)
- **dense**: A list containing the results of the dense SVD.
  - u: Matrix with left Eigenvectors.
  - v: Matrix with right Eigenvectors.
  - d: Matrix with singular values.
- **sparse**: A list containing the results of the sparse SVD.
  - u: Matrix with left Eigenvectors.
  - v: Matrix with right Eigenvectors.
  - d: Matrix with singular values.
  - opt_dg_v: Selected degrees of sparsity for right Eigenvectors.
  - opt_dg_u: Selected degrees of sparsity for left Eigenvectors.
- **Graphical displays**: Depending on the values in 'plot', 'tSNE', 'cluster', and 'clus.lab' arguments, the following ggplot objects can be obtained. They contain:
  - **scree_plot**: Plots of Eigenvalues and their first and second order empirical derivatives along PC indexes.
  - **tun_dgSpar_plot**: Plots with the PEV trajectory, as well as its first and second empirical derivatives along the degrees of sparsity path.
  - **PC_plot**: Plot of the first principal components according to axes.pos. By default the first two are plotted.
  - **tSNE_plot**: Plot with the tSNE mapping onto two dimensions.
  - **clus_plot**: The output of function tsne2clus.
  - **subLabels_vs_PCs**: Plot of the Kruskal-Wallis (or Chi-square) statistics of the association test between PC (or selected subjects) and pre-established subjects groups.
  - **clusters_vs_PCs**: Plot of the Kruskal-Wallis (or Chi-square) statistics of the association test between PC (or selected subjects) and detected clusters.
Note

1. The function does not return PEV for EN parameter (alpha_v and/or alpha_u), the user needs to provide a single value for each.
2. When number of PC index > 1, columns of T might not be orthogonal.
3. Although the user is encouraged to perform data projection and cluster separately, MOSS allows to do this automatically. However, both tasks might require further tuning than the provided by default, and computations could become cumbersome.
4. Tuning of degrees of sparsity is done heuristically on training set. In our experience, this results in high specificity, but rather low sensitivity. (i.e. too liberal cutoffs, as compared with extensive cross-validation on testing set).
5. When 'method' is an unsupervised technique, 'K.X' is the number of latent factors returned and used in further analysis. When 'method' is a supervised technique, 'K.X' is used to perform a SVD to facilitate the product of large matrices and inverses.
6. If 'K.X' (or 'K.Y') equal 1, no plots are returned.
7. Although the degree of sparsity maps onto number of features/subjects for Lasso, the user needs to be aware that this conceptual correspondence is lost for full EN (alpha belonging to (0, 1); e.g. the number of features selected with alpha < 1 will be eventually larger than the optimal degree of sparsity). This allows to rapidly increase the number of non-zero elements when tuning the degrees of sparsity. In order to get exact values for the degrees of sparsity at subjects or features levels, the user needs to set the value of 'exact.dg' parameter from 'FALSE' (the default) to 'TRUE'.

References


Examples

```r
# Example: sparse PCA of a list of omic blocks.
library("MOSS")
sim_data <- simulate_data()
set.seed(43)

# Extracting simulated omic blocks.
```
```r

# Extracting subjects and features labels.
lab.sub <- sim_data$labels$lab.sub
lab.feat <- sim_data$labels$lab.feat
out <- moss(sim_blocks[-4],
  method = "pca",
  nu.v = seq(1, 200, by = 100),
  nu.u = seq(1, 100, by = 50),
  alpha.v = 0.5,
  alpha.u = 1
)

library(ggplot2)
library(ggthemes)
library(viridis)
library(cluster)
library(fpc)

set.seed(43)

# Example2: sparse PCA with t-SNE, clustering, and association with # predefined groups of subjects.
out <- moss(sim_blocks[-4], axes.pos=c(1:5),
  method = "pca",
  nu.v = seq(1, 200, by = 10),
  nu.u = seq(1, 100, by = 2),
  alpha.v = 0.5,
  alpha.u = 1,
  tSNE = TRUE,
  cluster = TRUE,
  clus.lab = lab.sub,
  plot = TRUE
)

# This shows clusters obtained with labels from pre-defined groups # of subjects.
out$clus_plot

# This shows the statistical overlap between PCs and the pre-defined # groups of subjects.
out$subLabels_vs_PCs

# This shows the contribution of each omic to the features # selected by PC index.
out$selected_items

# This shows features forming signatures across clusters.
out$feat_signatures

# Example3: Multi-block PCA with sparsity.
out <- moss(sim_blocks[-4], axes.pos=1:5,
  method = "mbpca",
  nu.v = seq(1, 200, by = 10),
  nu.u = seq(1, 100, by = 2),
  alpha.v = 0.5,
  alpha.u = 1,
  tSNE = TRUE,
  cluster = TRUE,
  clus.lab = lab.sub,
  plot = TRUE
)
```

moss_heatmap

Creates a heatmap from the output of MOSS.
**moss_select**

**Description**

Creates a heatmap from the output of MOSS.

**Usage**

```r
moss_heatmap(B, SVD, right.lab, left.lab, axes.pos = NULL, verbose = TRUE)
```

**Arguments**

- `B`: An object of class 'matrix' or 'FBM'.
- `SVD`: List with the results of a sparse or dense SVD.
- `right.lab`: Columns title. Character.
- `left.lab`: Rows title. Character.
- `axes.pos`: What SVD dimensions should be used to plot the heatmap? If NULL, all the SVD dimensions are used. Defaults to NULL.
- `verbose`: Should we print messages? Logical. Defaults to TRUE.

**Value**

Returns a 'ComplexHeatmap' plot representing the cross-product between left and right Eigenvectors.

**moss_select**

*Returns features and subject selected by latent dimension.*

**Description**

This function is meant to be used after moss. Its main purpose is to extract the features and subjects by latent dimension. The selection depends on loadings at each dimension being different from zero.

**Usage**

```r
moss_select(data.blocks, SVD, resp.block = NULL, K = NULL, plot = FALSE)
```

**Arguments**

- `data.blocks`: A list of omic blocks as provided to moss.
- `SVD`: A list with SVD results. The function is meant to work with the results from sparse SVD. However, 'dense' solutions are also accepted.
- `resp.block`: Which omic block was used as response in moss? Integer. Defaults to NULL.
- `K`: How many dimensions should be displayed? Vector. Defaults to the `1 : ncol(SVD$v)`.
- `plot`: Should the results be plotted? Logical. Defaults to FALSE.
moss_signatures

Value
Returns a list containing the position, name, and loadings of selected features and subjects by latent dimension. If 'plot=TRUE', a scatterplot is displayed, where the x-axis represents the latent dimensions, the y-axis the total number of features selected in log scale, and each point is a pie chart showing the relative contribution of each omic to the number of features selected. The ratio of the pie chart represents the coefficient of variation among squared loadings (mean squared loadings divided by their standard deviation).

moss_signatures  Returns signatures of features by groups of subjects

Description
This function is meant to used after moss_select. Its main purpose is to visualize how each selected feature (non-zero loading feature) contributes to each group of subjects by latent dimension.

Usage
moss_signatures(
  data.blocks,
  moss_select.out,
  clus_lab = NULL,
  plot = FALSE,
  feature.labels = NULL,
  th = 1,
  only.candidates = FALSE
)

Arguments
data.blocks  A list of omic blocks as provided to moss.
moss_select.out  The output of moss_select.
clus_lab  A vector of same length than number of subjects with labels used to visualize clusters. Defaults to NULL.
plot  Should the results be plotted? Logical. Defaults to FALSE
feature.labels  List with with features names for each omic. Defaults to NULL.
th  Show the th Default to th=1 (all the features). Numeric.
only.candidates  Should we plot only candidate features? Logical.
Value

Returns a list with 'signatures', and if plot=TRUE, a ggplot object named 'sig_plot'. The element 'signatures' is a data frame with columns corresponding to 'Cluster' (groups of subjects), 'Omic', 'Dim' (PC index or latent dimension), 'Feature_name', 'Feature_pos' (column index of the selected feature within the corresponding omic), 'Loadings' (non-zero loadings from moss), 'Means', 'L1' and 'L2' (mean +/- standard error of the selected feature values within an omic).

Examples

library("MOSS")
# Extracting simulated omic blocks.
sim_data <- simulate_data()
sim_blocks <- sim_data$sim_blocks

# Extracting subjects and features labels.
lab.sub <- sim_data$labels$lab.sub

out <- moss(sim_blocks[-4],
    method = "pca",
    nu.v = 10,
    exact.dg = TRUE,
    plot = TRUE,
    alpha.v = 0.5
)
out2 <- moss_select(data.blocks = sim_blocks[-4],
    SVD = out$sparse,
    plot = TRUE)

# Display signature plots.
out3 <- moss_signatures(data.blocks = sim_blocks[-4],
    clus_lab=lab.sub,
    moss_select.out = out2,
    plot = TRUE)

out3$sig_plot

moss_venn

Useful Venn diagrams to study the overlap between samples row names.

Description

Useful Venn diagrams to study the overlap between samples row names.

Usage

moss_venn(L, a, lty = "blank", fill = NULL, element_names = NULL)
Arguments

- L: List of elements which overlap we wish to check (e.g., row names by omic blocks).
- a: Elements of the list we want to focus on (e.g., a subset of omic blocks). Numerical.
- lty: Line width of circles circumferences.
- fill: Color for each circle. Character vector. Defaults to NULL.
- element_names: Names of each category. Character vector. Defaults to NULL.

Value

A data.frame with labels as rownames and two

---

**pca2tsne**

*Mapping principal components onto a 2D map via tSNE.*

Description

This function is called by moss whenever `moss(tSNE=TRUE)` to project latent factors onto two dimensions via t-stochastic neighbor embedding (tSNE). However, it can be used on any generic data matrix. The function uses the Barnes-Hut tSNE algorithm from Rtsne package, and uses an iterative procedure to select a tSNE map minimizing the projection cost across several random initial conditions. The function is inspired by the iterative procedure discussed in Taskesen et al. 2016 and code originally provided with the publication.

Usage

```r
pca2tsne(Z, perp = 50, n.samples = 1, n.iter = 1000, parallel = FALSE)
```

Arguments

- Z: A matrix with axes of variation (typically PCs) as columns and subjects as rows.
- n.samples: Number of times the algorithm starts from different random initial conditions. Defaults to 1.
- n.iter: Number of iterations for each run of the algorithm.
- parallel: Should random starts be done in parallel? Logical. Default to FALSE. Defaults to 1000.

Value

Returns output of function `Rtsne::Rtsne` from the random initial condition with the smallest 'reconstruction error'.
References

- Krijthe JH. Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation. 2015

Examples

```r
library("MOSS")
sim_blocks <- simulate_data()$sim_blocks

# Example of pca2tsne usage.
Z <- pca2tsne(sim_blocks$`Block 3`,
        perp = 50,
        n.samples = 1,
        n.iter = 1e3)$Y
plot(Z, xlab = "x_tSNE(X)", ylab = "y_tSNE(X)"

# Example of usage within moss.
set.seed(34)
moss(sim_blocks[-4],
    tSNE = list(
        perp = 50,
        n.samples = 1,
        n.iter = 1e3
    ),
    plot = TRUE
)$tSNE_plot
```

prepro_na

*Missing values imputation by the mean of each column.*

Description

This function is called by moss to count the impute missing values by the mean of each column within omic blocks. If any column has more than 20

Usage

```
prepro_na(X)
```
Arguments

X An object of class 'matrix', 'FBM', or 'array'.

Details

Meant for objects of class 'matrix', 'FBM', or 'array'.

Value

Returns input with imputed missing values.

describe prepro_sub

Scale and normalize columns of a matrix.

Description

This function is called by moss to scale and normalize (extended) matrices.

Usage

prepro_sub(X, scale.arg, norm.arg)

Arguments

X An object of class 'matrix', 'FBM', or 'array'.

scale.arg Should we scale columns? Logical.

norm.arg Should we normalize columns? Logical.

Details

Meant for objects of class 'matrix', 'FBM', or 'array'.

Value

A matrix with scaled and/or normalized columns.
simulate_data

Simple simulation of regulatory modules.

**Description**

This a simple simulation to use in MOSS’ examples. The specifics of the simulation are shown in the "Examples" section.

**Usage**

```r
simulate_data(moss_seed = 42)
```

**Arguments**

- `moss_seed` The seed for random number generator. Numeric. Defaults to 42.

**Value**

A list of two elements 'sim_blocks' and 'labels'. First element 'sim_blocks' is a list of three numeric matrices, and one character matrix. Second element 'labels' has two character vectors. The first element 'lab.sub' identifies the groups of 'signal' subjects. The second element 'lab.feat' identifies the groups 'signal' features from background 'noise'.

**Examples**

```r
sim_data <- simulate_data()

# Extracting simulated omic blocks.
sim_blocks <- sim_data$sim_blocks

# Extracting subjects and features labels.
lab.sub <- sim_data$labels$lab.sub
lab.feat <- sim_data$labels$lab.feat

# Check dimensions and objects class.
lapply(sim_blocks, dim)
lapply(sim_blocks, function(x) class(x[, 1]))

# Showing how the data was generated.
set.seed(42)
O1 <- matrix(data = 0, nrow = 5e2, ncol = 1e3)
O2 <- O1
O1[1:20, 1:150] <- 1
O1 <- O1 + rnorm(5e5, mean = 0, sd = 0.5)
O2[71:90, 71:200] <- 1
O2 <- O2 + rnorm(5e5, mean = 0, sd = 0.5)

# Simulating a continuous response blocks.
O3 <- 3 * O1 - 5 * O2 + rnorm(5e5, mean = 0, sd = 0.5)
```
ssvdEN

Sparse Singular Value Decomposition via Elastic Net.

Description

This function performs sparse singular value decomposition (SVD) on a matrix 'x' via Elastic Net types of penalties. For one PC (rank 1 case), the algorithm finds left and right Eigenvectors (u and w, respectively), that minimize: ||x - u w'||_F^2 + lambda_w (alpha_w||w||_1 + (1 - alpha_w)||w||_F^2) + lambda_u (alpha||u||_1 + (1 - alpha_u)||u||_F^2) such that ||u|| = 1. The right Eigen vector is obtained from v = w / ||w|| and the corresponding Eigen value = u^T x v. The penalties lambda_u and lambda_w are mapped from specified desired degree of sparsity (dg.spar.features & dg.spar.subjects).

Usage

ssvdEN(
  O,
  n.PC = 1,
  dg.spar.features = NULL,
  dg.spar.subjects = NULL,
  maxit = 500,
  tol = 0.001,
  scale.arg = TRUE,
  center.arg = TRUE,
  approx.arg = FALSE,
  alpha.f = 1,
  alpha.s = 1,
ssvdEN

```r
svd.0 = NULL,
s.values = TRUE,
ncores = 1,
extact.dg = FALSE
```

**Arguments**

- **O**
  Numeric matrix of `n` subjects (rows) and `p` features (columns). It can be a File-backed Big Matrix.

- **n.PC**
  Number of desired principal axes. Numeric. Defaults to 1.

- **dg.spar.features**
  Degree of sparsity at the features level. Numeric. Defaults to NULL.

- **dg.spar.subjects**
  Degree of sparsity at the subjects level. Numeric. Defaults to NULL.

- **maxit**
  Maximum number of iterations for the sparse SVD algorithm. Numeric. Defaults to 500.

- **tol**
  Convergence tolerance for the sparse SVD algorithm. Numeric. Defaults to 0.001.

- **scale.arg**
  Should `O` be scaled? Logical. Defaults to TRUE.

- **center.arg**
  Should `O` be centered? Logical. Defaults to TRUE.

- **approx.arg**
  Should we use standard SVD or random approximations? Defaults to FALSE. If TRUE & is(O,'matrix') == TRUE, irlba is called. If TRUE & is(O, "FBM") == TRUE, big_randomSVD is called.

- **alpha.f**
  Elastic net mixture parameter at the features level. Measures the compromise between lasso (alpha = 1) and ridge (alpha = 0) types of sparsity. Numeric. Defaults to 1.

- **alpha.s**
  Elastic net mixture parameter at the subjects level. Defaults to alpha.s = 1.

- **svd.0**
  List containing an initial SVD. Defaults to NULL.

- **s.values**
  Should the singular values be calculated? Logical. Defaults to TRUE.

- **ncores**
  Number of cores used by big_randomSVD. Default does not use parallelism. Ignored when class(O)!="FBM".

- **exact.dg**
  Should we compute exact degrees of sparsity? Logical. Defaults to FALSE. Only relevant When alpha.s or alpha.f are in the (0,1) interval and exact.dg = TRUE.

**Details**

The function allows the use of the base svd function for relatively small problems. For larger problems, functions for fast-partial SVD (irlba and big_randomSVD, from irlba and bigstatsr packages) are used.
Value

A list with the results of the (sparse) SVD, containing:

- u: Matrix with left eigenvectors.
- v: Matrix with right eigenvectors.
- d: Matrix with singular values.

Note

When elastic net is used ('alpha.s' or 'alpha.f' in the (0,1) interval), the resulting number of non-zero subjects or features is larger than the 'dg.spar.subjects' or 'dg.spar.features' values. This allows to rapidly increase the number of non-zero elements when tuning the degrees of sparsity with function `ssvdEN_sol_path`. In order to get exact values for the degrees of sparsity at subjects or features levels, the user needs to set the value of 'exact.dg' parameter from 'FALSE' (the default) to 'TRUE'.

References


Examples

```r
library("MOSS")

# Extracting simulated omic blocks.
sim_blocks <- simulate_data()$sim_blocks
X <- sim_blocks$'Block 3'

# Equal to svd solution: exact singular vectors and values.
out <- ssvdEN(X, approx.arg = FALSE)

# Uses irlba to get approximated singular vectors and values.
library(irlba)
out <- ssvdEN(X, approx.arg = TRUE)

# Uses bigstatsr to get approximated singular vectors and values
# of a Filebacked Big Matrix.
library(bigstatsr)
out <- ssvdEN(as_FBM(X), approx.arg = TRUE)

# Sampling a number of subjects and features for a fix sparsity degree.
s.u <- sample(1:nrow(X), 1)
s.v <- sample(1:ncol(X), 1)

# Lasso penalties.
all.equal(sum(ssvdEN(X, dg.spar.features = s.v)$v != 0), s.v)
```

```r
```
```
unique(colSums(ssvdEN(X, dg.spar.features = s.v, n.PC = 5)$v
!= 0)),
s.v
}

all.equal(sum(ssvdEN(X, dg.spar.subjects = s.u)$u != 0), s.u)
all.equal(
  unique(colSums(ssvdEN(X, dg.spar.subjects = s.u, n.PC = 5)$u
  != 0)),
s.u
)

out <- ssvdEN(X, dg.spar.features = s.v, dg.spar.subjects = s.u)
all.equal(sum(out$u != 0), s.u)
all.equal(sum(out$v != 0), s.v)

out <- ssvdEN(X,
  dg.spar.features = s.v, dg.spar.subjects = s.u,
  n.PC = 10
)
all.equal(unique(colSums(out$u != 0)), s.u)
all.equal(unique(colSums(out$v != 0)), s.v)

# Ridge penalties.
all.equal(
  sum(ssvdEN(X, dg.spar.features = s.v, alpha.f = 0)$v != 0),
  ncol(X)
)
all.equal(
  unique(colSums(ssvdEN(X, 
    dg.spar.features = s.v, n.PC = 5, 
    alpha.f = 0)
  )$v != 0)),
  ncol(X)
)

all.equal(
  sum(ssvdEN(X, dg.spar.subjects = s.u, alpha.s = 0)$u != 0),
  nrow(X)
)
all.equal(
  unique(colSums(ssvdEN(X, 
    dg.spar.subjects = s.u, n.PC = 5, 
    alpha.s = 0)
  )$u != 0)),
  nrow(X)
)

out <- ssvdEN(X,
  dg.spar.features = s.v, dg.spar.subjects = s.u,
  alpha.f = 0, alpha.s = 0
)
all.equal(sum(out$u != 0), nrow(X))
```
all.equal(sum(out$v != 0), ncol(X))

out <- ssvdEN(X,
    dg.spar.features = s.v, dg.spar.subjects = s.u,
    n.PC = 10, alpha.f = 0,
    alpha.s = 0
)
all.equal(unique(colSums(out$u != 0)), nrow(X))
all.equal(unique(colSums(out$v != 0)), ncol(X))

# Elastic Net penalties.
sum(ssvdEN(X, dg.spar.features = s.v, alpha.f = 0.5)$v != 0) >= s.v
all(unique(colSums(ssvdEN(X,
    dg.spar.features = s.v, n.PC = 5,
    alpha.f = 0.5
)! = 0)) >= s.v)

sum(ssvdEN(X, dg.spar.subjects = s.u, alpha.s = 0.5)$u != 0) >= s.u
all(unique(colSums(ssvdEN(X,
    dg.spar.subjects = s.u, n.PC = 5,
    alpha.s = 0.5
)! = 0)) >= s.u)

# Elastic Net penalties with exact degrees of sparsity.
sum(ssvdEN(X,
    dg.spar.features = s.v, alpha.f = 0.5,
    exact.dg = TRUE
)! = 0) == s.v
all(unique(colSums(ssvdEN(X,
    dg.spar.features = s.v, n.PC = 5,
    alpha.f = 0.5, exact.dg = TRUE
)! = 0)) == s.v)

sum(ssvdEN(X,
    dg.spar.subjects = s.u, alpha.s = 0.5,
    exact.dg = TRUE
)! = 0) == s.u
all(unique(colSums(ssvdEN(X,
    dg.spar.subjects = s.u, n.PC = 5,
    alpha.s = 0.5, exact.dg = TRUE
)! = 0)) == s.u)

---

**Description**

This function allows to explore values on the solution path of the sparse singular value decomposition (SVD) problem. The goal of this is to tune the degree of sparsity of subjects, features, or both
subjects/features. The function performs a penalized SVD that imposes sparsity/smoothing in both left and right singular vectors. The penalties at both levels are Elastic Net-like, and the trade-off between ridge and Lasso like penalties is controlled by two 'alpha' parameters. The proportion of variance explained is the criteria used to choose the optimal degrees of sparsity.

Usage

```r
ssvdEN_sol_path(
  O,
  center = TRUE,
  scale = TRUE,
  dg.grid.right = seq_len(ncol(O)) - 1,
  dg.grid.left = NULL,
  n.PC = 1,
  svd.0 = NULL,
  alpha.f = 1,
  alpha.s = 1,
  maxit = 500,
  tol = 0.001,
  approx = FALSE,
  plot = FALSE,
  ncores = 1,
  verbose = TRUE,
  lib.thresh = TRUE,
  left.lab = "Subjects",
  right.lab = "Features",
  exact.dg = FALSE
)
```

Arguments

- **O**: Numeric matrix of n subjects (rows) and p features (columns). Only objects supported are 'matrix' and 'FBM'.
- **center**: Should we center? Logical. Defaults to TRUE.
- **scale**: Should we scale? Logical. Defaults to TRUE.
- **dg.grid.right**: Grid with degrees of sparsity at the features level. Numeric. Default is the entire solution path for features (i.e. 1 : (ncol(O) - 1)).
- **dg.grid.left**: Grid with degrees of sparsity at the subjects level. Numeric. Defaults to dg.grid.left = nrow(O).
- **n.PC**: Number of desired principal axes. Numeric. Defaults to 1.
- **svd.0**: Initial SVD (i.e. least squares solution). Defaults to NULL.
- **alpha.f**: Elastic net mixture parameter at the features level. Measures the compromise between lasso (alpha = 1) and ridge (alpha = 0) types of sparsity. Numeric. Defaults to 1.
- **alpha.s**: Elastic net mixture parameter at the subjects level. Defaults to alpha.s = 1.
- **maxit**: Maximum number of iterations. Defaults to 500.
Convergence is determined when \( \|U_j - U_{j-1}\|_F < \text{tol} \), where \( U_j \) is the matrix of estimated left regularized singular vectors at iteration \( j \).

Should we use standard SVD or random approximations? Defaults to FALSE. If TRUE & is(O,'matrix') == TRUE, irlba is called. If TRUE & is(O, "FBM") == TRUE, big_randomSVD is called.

Should we plot the solution path? Logical. Defaults to FALSE.

Number of cores used by big_randomSVD. Default does not use parallelism. Ignored when is(O, "FBM") == TRUE.

Should we print messages? Logical. Defaults to TRUE.

Should we use a liberal or conservative threshold to tune degrees of sparsity? Logical. Defaults to TRUE.

Label for the subjects level. Character. Defaults to ‘subjects’.

Label for the features level. Character. Defaults to ‘features’.

Should we compute exact degrees of sparsity? Logical. Defaults to FALSE. Only relevant When alpha.s or alpha.f are in the (0,1) interval and exact.dg = TRUE.

The function returns the degree of sparsity for which the change in PEV is the steepest (‘liberal’ option), or for which the change in PEV stabilizes (‘conservative’ option). This heuristics relax the need of tuning parameters on a testing set.

For one PC (rank 1 case), the algorithm finds vectors \( u, w \) that minimize: \( \|x - u w\|_F^2 + \lambda_w (\alpha_w\|w\|_1 + (1 - \alpha_w)\|w\|_F^2) + \lambda_u (\alpha\|u\|_1 + (1 - \alpha_u)\|u\|_F^2) \) such that \( \|u\| = 1 \). The right Eigen vector is obtained from \( v = w / \|w\| \) and the corresponding Eigen value = \( u^T x v \). The penalties \( \lambda_u \) and \( \lambda_w \) are mapped from specified desired degrees of sparsity (dg.spar.features & dg.spar.subjects).

A list with the results of the (sparse) SVD and (if argument ‘plot’=TRUE) the corresponding graphical displays.

SVD: a list with the results of the (sparse) SVD, containing:

- \( u \): Matrix with left eigenvectors.
- \( v \): Matrix with right eigenvectors.
- \( d \): Matrix with singular values.
- \( \text{opt.dg.right} \): Selected degrees of sparsity for right eigenvectors.
- \( \text{opt.dg.left} \): Selected degrees of sparsity for left eigenvectors.

plot: A ggplot object.
Note

Although the degree of sparsity maps onto number of features/subjects for Lasso, the user needs to be aware that this conceptual correspondence is lost for full EN (alpha belonging to \((0, 1)\); e.g. the number of features selected with alpha < 1 will be eventually larger than the optimal degree of sparsity). This allows to rapidly increase the number of non-zero elements when tuning the degrees of sparsity. In order to get exact values for the degrees of sparsity at subjects or features levels, the user needs to set the value of 'exact.dg' parameter from 'FALSE' (the default) to 'TRUE'.

References


Examples

```r
library("MOSS")

# Extracting simulated omic blocks.
sim_blocks <- simulate_data()$sim_blocks
X <- sim_blocks$\textnormal{\(\text{Block 3}\)}

# Tuning sparsity degree for features (increments of 20 units).
out <- ssvdEN_sol_path(X, dg.grid.right = seq(1, 1000, by = 20))
```

ssvdEN_sol_path_par

'Solution path' for sparse Singular Value Decomposition via Elastic Net using parallel computing.

Description

This function is a copy of 'ssvdEN_sol_path' meant to be used in combination with the future.apply package to allow for parallel computing of the optimal degrees of sparsity by subjects and/or features.

Usage

```r
ssvdEN_sol_path_par(
  0,
  center = TRUE,
  scale = TRUE,
  dg.grid.right = seq_len(ncol(0)) - 1,
  dg.grid.left = NULL,
  n.PC = 1,
  svd.0 = NULL,
  alpha.f = 1,
)```
alpha.s = 1, 
maxit = 500, 
tol = 0.001, 
approx = FALSE, 
plot = FALSE, 
ncores = 1, 
verbose = TRUE, 
lib.thresh = TRUE, 
left.lab = "Subjects", 
right.lab = "Features", 
extact.dg = FALSE

Arguments

0 Numeric matrix of n subjects (rows) and p features (columns). Only objects supported are 'matrix' and 'FBM'.
center Should we center? Logical. Defaults to TRUE.
scale Should we scale? Logical. Defaults to TRUE.
dg.grid.right Grid with degrees of sparsity at the features level. Numeric. Default is the entire solution path for features (i.e. 1 : (ncol(O) - 1)).
dg.grid.left Grid with degrees of sparsity at the subjects level. Numeric. Defaults to dg.grid.left = nrow(O).
n.PC Number of desired principal axes. Numeric. Defaults to 1.
svd.0 Initial SVD (i.e. least squares solution). Defaults to NULL.
alp...a.f Elastic net mixture parameter at the features level. Measures the compromise between lasso (alpha = 1) and ridge (alpha = 0) types of sparsity. Numeric. Defaults to 1.
alp...a.s Elastic net mixture parameter at the subjects level. Defaults to alpha.s = 1.
maxit Maximum number of iterations. Defaults to 500.
tol Convergence is determined when \|U_{j} - U_{j-1}\|_F < tol, where U_j is the matrix of estimated left regularized singular vectors at iteration j.
approx Should we use standard SVD or random approximations? Defaults to FALSE. If TRUE & is(O, 'matrix') == TRUE, irlba is called. If TRUE & is(O, "FBM") == TRUE, big_randomSVD is called.
plot Should we plot the solution path? Logical. Defaults to FALSE
ncores Number of cores used by big_randomSVD. Default does not use parallelism. Ignored when is(O, "FBM") == TRUE.
verbose Should we print messages? Logical. Defaults to TRUE.
lib.thresh Should we use a liberal or conservative threshold to tune degrees of sparsity? Logical. Defaults to TRUE.
left.lab Label for the subjects level. Character. Defaults to 'subjects'.
right.lab Label for the features level. Character. Defaults to 'features'.
exa...c.dg Should we compute exact degrees of sparsity? Logical. Defaults to FALSE. Only relevant When alpha.s or alpha.f are in the (0,1) interval and exact.dg = TRUE.
Note

Although the degree of sparsity maps onto number of features/subjects for Lasso, the user needs to be aware that this conceptual correspondence is lost for full EN (alpha belonging to (0, 1); e.g. the number of features selected with alpha < 1 will be eventually larger than the optimal degree of sparsity). This allows to rapidly increase the number of non-zero elements when tuning the degrees of sparsity. In order to get exact values for the degrees of sparsity at subjects or features levels, the user needs to set the value of ‘exact.dg’ parameter from ‘FALSE’ (the default) to ‘TRUE’.

Examples

library("MOSS")

# Extracting simulated omic blocks.
sim_blocks <- simulate_data()$sim_blocks
X <- sim_blocks$'Block 3'

# Comparing ssvdEN_sol_path_par and ssvdEN_sol_path.
t1 <- proc.time()
out1 <- ssvdEN_sol_path(X, dg.grid.right = 1:1000, dg.grid.left = 1:500)
t1 <- proc.time() - t1

# Comparing ssvdEN_sol_path_par and ssvdEN_sol_path.
t2 <- proc.time()
out2 <- ssvdEN_sol_path_par(X, dg.grid.right = 1:1000, dg.grid.left = 1:500)
t2 <- proc.time() - t2

---

tsne2clus  
\textit{t-Stochastic Neighbor Embedding to Clusters}

Description

Finds clusters on a 2 dimensional map using Density-based spatial clustering of applications with noise (DBSCAN; Esther et al. 1996).

Usage

tsne2clus(
S.tsne,
ann = NULL,
labels,
aest = NULL,
eps_res = 100,
eps_range = c(0, 4),
min.clus.size = 10,
group.names = "Groups",
xlab = "x: tSNE(X)",
ylab = "y: tSNE(X)"
clus = TRUE
)

Arguments

S.tsne
Outcome of function "pca2tsne"

ann
Subjects’ annotation data. An incidence matrix assigning subjects to classes of biological relevance. Meant to tune cluster assignation via Biological Homogeneity Index (BHI). If ann=NULL, the number of clusters is tuned with the Silhouette index instead of BHI. Defaults to NULL.

labels
Character vector with labels describing subjects. Meant to assign aesthetics to the visual display of clusters.

aest
Data frame containing points shape and color. Defaults to NULL.

eps_res
How many eps values should be explored between the specified range?

eps_range
Vector containing the minimum and maximum eps values to be explored. Defaults to c(0, 4).

min.clus.size
Minimum size for a cluster to appear in the visual display. Defaults to 10

group.names
The title for the legend’s key if ‘aest’ is specified.

xlab
Name of the ‘xlab’. Defaults to "x: tSNE(X)"

ylab
Name of the ‘ylab’. Defaults to "y: tSNE(X)"

clus
Should we do clustering? Defaults to TRUE. If false, only point aesthetics are applied.

Details

The function takes the outcome of pca2tsne (or a list containing any two-columns matrix) and finds clusters via DBSCAN. It extends code from the MEREDITH (Taskesen et al. 2016) and cValid (Datta & Datta, 2018) R packages to tune DBSCAN parameters with Silhouette or Biological Homogeneity indexes.

Value

A list with the results of the DBSCAN clustering and (if argument ‘plot’=TRUE) the corresponding graphical displays.

dbscan.res: a list with the results of the (sparse) SVD, containing:

• cluster: Cluster partition.
  – eps: Optimal eps according to the Silhouette or Biological Homogeneity indexes criteria.
  – SIL: Maximum peak in the trajectory of the Silhouette index.
  – BHI: Maximum peak in the trajectory of the Biological Homogeneity index.

• clusters.plot: A ggplot object with the clusters’ graphical display.
References


Examples

```r
library(MOSS)
library(viridis)
library(cluster)
library(annotate)

# Using the 'iris' data to show cluster definition via BHI criterion.
set.seed(42)
data(iris)
# Scaling columns.
X <- scale(iris[, -5])
# Calling pca2tsne to map the three variables onto a 2-D map.
Z <- pca2tsne(X, perp = 30, n.samples = 1, n.iter = 1000)
# Using 'species' as previous knowledge to identify clusters.
ann <- model.matrix(~ -1 + iris[, 5])
# Getting clusters.
tsne2clus(Z,
  ann = ann,
  labels = iris[, 5],
  aest = aest.f(iris[, 5]),
  group.names = "Species",
  eps_range = c(0, 3)
)

# Example of usage within moss.
set.seed(43)
sim_blocks <- simulate_data()$sim_blocks
out <- moss(sim_blocks[-4],
  tSNE = TRUE,
  cluster = list(eps_range = c(0, 4), eps_res = 100, min_clus_size = 1),
  plot = TRUE)
out$clus_plot
out$clusters_vs_PCs
```
Index

aest.f, 2

cov_adj, 3

metdat, 4
moss, 4
moss_heatmap, 10
moss_select, 11
moss_signatures, 12
moss_venn, 13

pca2tsne, 14
prepro_na, 15
prepro_sub, 16

simulate_data, 17
ssvdEN, 18
ssvdEN_sol_path, 22
ssvdEN_sol_path_par, 25

tsne2clus, 27