Package ‘timeOmics’

December 31, 2024

Title Time-Course Multi-Omics data integration

Version 1.19.0

Description timeOmics is a generic data-driven framework to integrate multi-Omics longitudinal data measured on the same biological samples and select key temporal features with strong associations within the same sample group.
The main steps of timeOmics are:
1. Plaform and time-specific normalization and filtering steps;
2. Modelling each biological into one time expression profile;
3. Clustering features with the same expression profile over time;
4. Post-hoc validation step.

License GPL-3

Encoding UTF-8

LazyData true

Imports dplyr, tidyr, tibble, purrr, magrittr, ggplot2, stringr, ggrepel, lmtest, plyr, checkmate

biocViews Clustering,FeatureExtraction,TimeCourse,DimensionReduction,Software, Sequencing, Microarray, Metabolomics, Metagenomics, Proteomics, Classification, Regression, ImmunoOncology, GenePrediction, MultipleComparison

Depends mixOmics, R (>= 4.0)

RoxygenNote 7.3.1

VignetteBuilder knitr

Suggests BiocStyle, knitr, rmarkdown, testthat, snow, tidyverse, igraph, gplots

Remotes cran/lmms

BugReports https://github.com/abodein/timeOmics/issues

git_url https://git.bioconductor.org/packages/timeOmics

git_branch devel

git_last_commit 6546b5f

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-30

Author Antoine Bodein [aut, cre],
Olivier Chapleur [aut],
Kim-Anh Le Cao [aut],
Arnaud Droit [aut]

Maintainer Antoine Bodein <antoine.bodein.1@ulaval.ca>

Contents

dmatrix.spearman.dissimilarity 2
getCluster 3
getNcomp 4
getSilhouette 5
getUpDownCluster 6
get_demo_cluster 6
get_demo_silhouette 7
lmms.filter.lines 8
plotLong 9
proportionality 11
remove.low.cv 12
tuneCluster.block.spls 13
tuneCluster.spca 14
tuneCluster.spls 16
unscale 17

Index **19**

`dmatrix.spearman.dissimilarity`
dmatrix.spearman.dissimilarity

Description

Compute the spearman dissimilarity distance.

Usage

```
dmatrix.spearman.dissimilarity(X)
```

Arguments

`X`	A numeric matrix with feature in colnames

Value

Return a dissimilarity matrix of size PxP.

`getCluster` *Get variable cluster from (s)PCA, (s)PLS or block.(s)PLS*

Description

This function returns the cluster associated to each feature from a mixOmics object.

Usage

```
getCluster(X, user.block = NULL, user.cluster = NULL)
```

Arguments

`X`	an object of the class: `pca`, `spca`, `pls`, `spls`, `block.pls` or `block.spls`
`user.block`	a vector to filter the result and return the features of the specified blocks.
`user.cluster`	a vector to filter the result and return only the features of the specified clusters

Details

For each feature, the cluster is assigned according to the maximum contribution on a component and the sign of that contribution.

Value

A data.frame containing the name of feature, its assigned cluster and other information such as selected component, contribution, sign, ...

See Also

`selectVar`

Examples

```
demo <- suppressWarnings(get_demo_cluster())
pca.cluster <- getCluster(demo$pca)
spca.cluster <- getCluster(demo$spca)
pls.cluster <- getCluster(demo$pls)
spls.cluster <- getCluster(demo$spls)
block.pls.cluster <- getCluster(demo$block.pls)
block.spls.cluster <- getCluster(demo$block.spls)
```

getNcomp *Get optimal number of components*

Description

Compute the average silhouette coefficient for a given set of components on a mixOmics result. Foreach given ncomp, the mixOmics method is performed with the sames arguments and the given ‘ncomp‘. Longitudinal clustering is performed and average silhouette coefficient is computed.

Usage

```
getNcomp(object, max.ncomp = NULL, X, Y = NULL, indY = NULL, ...)
```

Arguments

`object`	A mixOmics object of the class ‘pca‘, ‘spca‘, ‘mixo_pls‘, ‘mixo_spls‘, ‘block.pls‘, ‘block.spls‘
`max.ncomp`	integer, maximum number of component to include. If no argument is given, ‘max.ncomp=object$ncomp‘
`X`	a numeric matrix/data.frame or a list of data.frame for `block.pls`
`Y`	(only for `pls`, optional for `block.spls`) a numeric matrix, with the same nrow as `X`
`indY`	(optional and only for `block.pls`, if Y is not provided), an integer which indicates the position of the matrix response in the list X
`...`	Other arguments to be passed to methods (pca, pls, block.pls)

Value

`getNcomp` returns a list with class "ncomp.tune.silhouette" containing the following components:

`ncomp`	a vector containing the tested ncomp
`silhouette`	a vector containing the average silhouette coefficient by ncomp
`dmatrix`	the distance matrix used to compute silhouette coefficient

See Also

`getCluster`, `silhouette`, `pca`, `pls`, `block.pls`

Examples

```
# random input data
demo <- suppressWarnings(get_demo_cluster())

# pca
pca.res <- mixOmics::pca(X=demo$X, ncomp = 5)
res.ncomp <- getNcomp(pca.res, max.ncomp = 4, X = demo$X)
```

```
plot(res.ncomp)

# pls
pls.res <- mixOmics::pls(X=demo$X, Y=demo$Y)
res.ncomp <- getNcomp(pls.res, max.ncomp = 4, X = demo$X, Y=demo$Y)
plot(res.ncomp)

# block.pls
block.pls.res <- suppressWarnings(mixOmics::block.pls(X=list(X=demo$X, Z=demo$Z), Y=demo$Y))
res.ncomp <- suppressWarnings(getNcomp(block.pls.res, max.ncomp = 4,
                                       X=list(X=demo$X, Z=demo$Z), Y=demo$Y))
plot(res.ncomp)
```

`getSilhouette` *Get Silhouette Coefficient from (s)PCA, (s)PLS or block.(s)PLS clusters*

Description

`getSilhouette` is a generic function that compute silhouette coefficient for an object of the type `pca`, `spca`, `pls`, `spls`, `block.pls`, `block.spls`.

Usage

```
getSilhouette(object)
```

Arguments

`object`	a mixOmics object of the class `pca`, `spca`, `pls`, `spls`, `block.pls`, `block.spls`

Details

This method extract the componant contribution depending on the object, perform the clustering step, and compute the silhouette coefficient.

Value

silhouette coefficient

Examples

```
demo <- suppressWarnings(get_demo_cluster())
getSilhouette(object = demo$pca)
getSilhouette(object = demo$spca)
getSilhouette(object = demo$pls)
getSilhouette(object = demo$spls)
getSilhouette(object = demo$block.pls)
getSilhouette(object = demo$block.spls)
```

getUpDownCluster *Up-Down clustering*

Description

Performs a clustering based on the signs of variation between 2 timepoints. Optionally, if the difference between 2 timepoints is lower than a given threshold, the returned difference will be 0.

Usage

```
getUpDownCluster(X, diff_threshold = 0)
```

Arguments

X	a dataframe or list of dataframe with the same number of rows.
diff_threshold	a number (optional, default 0), if the difference between 2 values is lower than the threshold, the returned sign will be 0 (no variation).

Examples

```
demo <- suppressWarnings(get_demo_cluster())
X <- list(X = demo$X, Y = demo$Y, Z = demo$Z)
res <- getUpDownCluster(X)
class(res)
getCluster(res)

X <- demo$X
res <- getUpDownCluster(X)
res <- getUpDownCluster(X, diff_threshold = 15)
res_cluster <- getCluster(res)
```

get_demo_cluster *get_demo_cluster*

Description

Generates random data to be used in examples.

Usage

```
get_demo_cluster()
```

Value

a list containg:

`X`	data.frame
`Y`	data.frame
`Z`	data.frame
`pca`	a mixOmics pca result
`spca`	a mixOmics spca result
`pls`	a mixOmics pls result
`spls`	a mixOmics spls result
`block.pls`	a mixOmics block.pls result
`block.spls`	a mixOmics block.spls result

Examples

```
# Random data could lead to "The SGCCA algorithm did not converge" warning which is not important for a demo
demo <- suppressWarnings(get_demo_cluster())
```

`get_demo_silhouette` *Get data for silhouette demo*

Description

Get data for silhouette demo

Usage

```
get_demo_silhouette()
```

Value

A matrix of expression profile, sample in raws, time in columns.

Examples

```
data <- get_demo_silhouette()
```

`lmms.filter.lines` *Filter Linear Profiles from Linear Mixed Model output*

Description

This function filters linear models with highly heterogeneous variability within residues. From an "lmms" output, 2 parameters are tested:

Usage

```
lmms.filter.lines(
  data,
  lmms.obj,
  time,
  homoskedasticity = TRUE,
  MSE.filter = TRUE,
  homoskedasticity.cutoff = 0.05
)
```

Arguments

`data`	a data.frame used in the `lmms::lmmSpline` command
`lmms.obj`	a `lmmspline` object
`time`	a numeric vector containing the sample time point information.
`homoskedasticity`	a logical whether or not to test for homoscedasticity with the Breusch-Pagan test.
`MSE.filter`	whether or not to test for low dispersion with a cutoff on the MSE.
`homoskedasticity.cutoff`	a numeric scalar between 0 and 1, p-value threshold for B-P test.

Details

* homo-sedasticity of the residues with a Breusch-Pagan test * low dispersion with a cutoff on the MSE (mean squared error)

Value

a list containing the following items

`filtering.summary`	a data.frame with the different tests per features (passed = TRUE, failed = FALSE)
`to.keep`	features which passed all the tests
`filtered`	the filtered data.frame

See Also

bptest

Examples

```
# data and lmms output
data(timeOmics.simdata)
data <- timeOmics.simdata$sim
lmms.output <- timeOmics.simdata$lmms.output
time <- timeOmics.simdata$time

# filter
filter.res <- lmms.filter.lines(data = data, lmms.obj = lmms.output, time = time)
```

---

plotLong *Plot Longitudinal Profiles by Cluster*

Description

This function provides a expression profile representation over time and by cluster.

Usage

```
plotLong(
  object,
  time = NULL,
  plot = TRUE,
  center = TRUE,
  scale = TRUE,
  title = "Time-course Expression",
  X.label = NULL,
  Y.label = NULL,
  legend = FALSE,
  legend.title = NULL,
  legend.block.name = NULL
)
```

Arguments

object	a mixOmics result of class (s)pca, (s)pls, block.(s)pls.
time	(optional) a numeric vector, the same size as `ncol(X)`, to change the time scale.
plot	a logical, if TRUE then a plot is produced. Otherwise, the data.frame on which the plot is based on is returned.
center	a logical value indicating whether the variables should be shifted to be zero centered.

scale	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place.
title	character indicating the title plot.
X.label	x axis titles.
Y.label	y axis titles.
legend	a logical, to display or not the legend.
legend.title	if legend is provided, title of the legend.
legend.block.name	a character vector corresponding to the size of the number of blocks in the mixOmics object.

Value

a data.frame (gathered form) containing the following columns:

time	x axis values
molecule	names of features
value	y axis values
cluster	assigned clusters
block	name of 'blocks'

See Also

getCluster

Examples

```
demo <- suppressWarnings(get_demo_cluster())
X <- demo$X
Y <- demo$Y
Z <- demo$Z

# (s)pca
pca.res <- mixOmics::pca(X, ncomp = 3)
plotLong(pca.res)
spca.res <- mixOmics::spca(X, ncomp =2, keepX = c(15, 10))
plotLong(spca.res)

# (s)pls
pls.res <- mixOmics::pls(X,Y)
plotLong(pls.res)
spls.res <- mixOmics::spls(X,Y, keepX = c(15,10), keepY=c(5,6))
plotLong(spls.res)

# (s)block.spls
block.pls.res <- mixOmics::block.pls(X=list(X=X,Z=Z), Y=Y)
plotLong(block.pls.res)
block.spls.res <- mixOmics::block.spls(X=list(X=X,Z=Z), Y=Y,
```

```
                                keepX = list(X = c(15,10), Z = c(5,6)),
                                keepY = c(3,6))
plotLong(block.spls.res)
```

---

proportionality *Proportionality Distance*

Description

proportionality is a wrapper that compute proportionality distance for a clustering result (pca, spca, pls, spls, block.pls, block.spls). and it performs a u-test to compare the median within a cluster to the median of the entire background set.

Usage

```
proportionality(X)
```

Arguments

X	an object of the class: pca, spca, pls, spls, block.pls or block.spls

Value

Return a list containing the following components:

propr.distance	Square matrix with proportionality distance between pairs of features
propr.distance.w.cluster	distance between pairs with cluster label
pvalue	Wilcoxon U-test p-value comparing the medians within clusters and with the entire background set

References

Lovell, D., Pawlowsky-Glahn, V., Egozcue, J. J., Marguerat, S., Bähler, J. (2015). Proportionality: a valid alternative to correlation for relative data. PLoS Comput. Biol. 11, e1004075. doi: 10.1371/journal.pcbi.1004075

Quinn, T. P., Richardson, M. F., Lovell, D., Crowley, T. M. (2017). propr: an r-package for identifying proportionally abundant features using compositional data analysis. Sci. Rep. 7, 16252. doi: 10.1038/s41598-017-16520-0

Examples

```
demo <- suppressWarnings(get_demo_cluster())

# pca
X <- demo$pca
propr.res <- proportionality(X)
plot(propr.res)

# pls
X <- demo$spls
propr.res <- proportionality(X)
plot(propr.res)

# block.pls
X <- demo$block.spls
propr.res <- proportionality(X)
plot(propr.res)
```

`remove.low.cv` *Remove features with low variation*

Description

`remove.low.cv` that removes variables with low variation. From a matrix/data.frame (samples in rows, features in columns), it computes the coefficient of variation for every features (columns) and return a filtered data.frame with features for which the coefficient of variation is above a given threshold.

Usage

```
remove.low.cv(X, cutoff = 0.5)
```

Arguments

`X`	a matrix/data.frame
`cutoff`	a numeric value

Value

a data.frame/matrix

Examples

```
mat <- matrix(sample(1:3, size = 200, replace = TRUE), ncol=20)
remove.low.cv(mat, 0.4)
```

tuneCluster.block.spls

Feature Selection Optimization for block (s)PLS method

Description

This function identify the number of feautures to keep per component and thus by cluster in `mixOmics::block.spls` by optimizing the silhouette coefficient, which assesses the quality of clustering.

Usage

```
tuneCluster.block.spls(
  X,
  Y = NULL,
  indY = NULL,
  ncomp = 2,
  test.list.keepX = NULL,
  test.keepY = NULL,
  ...
)
```

Arguments

`X`	list of numeric matrix (or data.frame) with features in columns and samples in rows (with samples order matching in all data sets).
`Y`	(optional) numeric matrix (or data.frame) with features in columns and samples in rows (same rows as `X`).
`indY`	integer, to supply if Y is missing, indicates the position of the matrix response in the list `X`.
`ncomp`	integer, number of component to include in the model
`test.list.keepX`	list of integers with the same size as X. Each entry corresponds to the different keepX value to test for each block of `X`.
`test.keepY`	only if Y is provideid. Vector of integer containing the different value of keepY to test for block `Y`.
`...`	other parameters to be included in the spls model (see `mixOmics::block.spls`)

Details

For each component and for each keepX/keepY value, a spls is done from these parameters. Then the clustering is performed and the silhouette coefficient is calculated for this clustering.

We then calculate "slopes" where keepX/keepY are the coordinates and the silhouette is the intensity. A z-score is assigned to each slope. We then identify the most significant slope which indicates a drop in the silhouette coefficient and thus a deterioration of the clustering.

Value

silhouette	silhouette coef. computed for every combinasion of keepX/keepY
ncomp	number of component included in the model
test.keepX	list of tested keepX
test.keepY	list of tested keepY
block	names of blocks
slopes	"slopes" computed from the silhouette coef. for each keepX and keepY, used to determine the best keepX and keepY
choice.keepX	best keepX for each component
choice.keepY	best keepY for each component

See Also

block.spls, getCluster, plotLong

Examples

```
demo <- suppressWarnings(get_demo_cluster())
X <- list(X = demo$X, Z = demo$Z)
Y <- demo$Y
test.list.keepX <- list("X" = c(5,10,15,20), "Z" = c(2,4,6,8))
test.keepY <- c(2:5)

# tuning
tune.block.spls <- tuneCluster.block.spls(X= X, Y= Y,
                                          test.list.keepX= test.list.keepX,
                                          test.keepY= test.keepY,
                                          mode= "canonical")
keepX <- tune.block.spls$choice.keepX
keepY <- tune.block.spls$choice.keepY

# final model
block.spls.res <- mixOmics::block.spls(X= X, Y= Y, keepX = keepX,
                              keepY = keepY, ncomp = 2, mode = "canonical")
# get clusters and plot longitudinal profile by cluster
block.spls.cluster <- getCluster(block.spls.res)
```

---

tuneCluster.spca *Feature Selection Optimization for sPCA method*

Description

This function identify the number of feautures to keep per component and thus by cluster in mixOmics::spca by optimizing the silhouette coefficient, which assesses the quality of clustering.

Usage

```
tuneCluster.spca(X, ncomp = 2, test.keepX = rep(ncol(X), ncomp), ...)
```

Arguments

`X`	numeric matrix (or data.frame) with features in columns and samples in rows
`ncomp`	integer, number of component to include in the model
`test.keepX`	vector of integer containing the different value of keepX to test for block `X`.
`...`	other parameters to be included in the spls model (see `mixOmics::spca`)

Details

For each component and for each keepX value, a spls is done from these parameters. Then the clustering is performed and the silhouette coefficient is calculated for this clustering.

We then calculate "slopes" where keepX are the coordinates and the silhouette is the intensity. A z-score is assigned to each slope. We then identify the most significant slope which indicates a drop in the silhouette coefficient and thus a deterioration of the clustering.

Value

`silhouette`	silhouette coef. computed for every combinasion of keepX/keepY
`ncomp`	number of component included in the model
`test.keepX`	list of tested keepX
`block`	names of blocks
`slopes`	"slopes" computed from the silhouette coef. for each keepX and keepY, used to determine the best keepX and keepY
`choice.keepX`	best `keepX` for each component

Examples

```
demo <- suppressWarnings(get_demo_cluster())
X <- demo$X

# tuning
tune.spca.res <- tuneCluster.spca(X = X, ncomp = 2, test.keepX = c(2:10))
keepX <- tune.spca.res$choice.keepX
plot(tune.spca.res)

# final model
spca.res <- mixOmics::spca(X=X, ncomp = 2, keepX = keepX)
plotLong(spca.res)
```

`tuneCluster.spls` *Feature Selection Optimization for sPLS method*

Description

This function identify the number of feautures to keep per component and thus by cluster in `mixOmics::spls` by optimizing the silhouette coefficient, which assesses the quality of clustering.

Usage

```
tuneCluster.spls(
  X,
  Y,
  ncomp = 2,
  test.keepX = rep(ncol(X), ncomp),
  test.keepY = rep(ncol(Y), ncomp),
  ...
)
```

Arguments

`X`	numeric matrix (or data.frame) with features in columns and samples in rows
`Y`	numeric matrix (or data.frame) with features in columns and samples in rows (same rows as `X`)
`ncomp`	integer, number of component to include in the model
`test.keepX`	vector of integer containing the different value of keepX to test for block `X`.
`test.keepY`	vector of integer containing the different value of keepY to test for block `Y`.
`...`	other parameters to be included in the spls model (see `mixOmics::spls`)

Details

For each component and for each keepX/keepY value, a spls is done from these parameters. Then the clustering is performed and the silhouette coefficient is calculated for this clustering.

We then calculate "slopes" where keepX/keepY are the coordinates and the silhouette is the intensity. A z-score is assigned to each slope. We then identify the most significant slope which indicates a drop in the silhouette coefficient and thus a deterioration of the clustering.

Value

`silhouette`	silhouette coef. computed for every combinasion of keepX/keepY
`ncomp`	number of component included in the model
`test.keepX`	list of tested keepX
`test.keepY`	list of tested keepY
`block`	names of blocks

slopes	"slopes" computed from the silhouette coef. for each keepX and keepY, used to determine the best keepX and keepY
choice.keepX	best keepX for each component
choice.keepY	best keepY for each component

See Also

spls, getCluster, plotLong

Examples

```
demo <- suppressWarnings(get_demo_cluster())
X <- demo$X
Y <- demo$Y

# tuning
tune.spls <- tuneCluster.spls(X, Y, ncomp= 2, test.keepX= c(5,10,15,20), test.keepY= c(2,4,6))
keepX <- tune.spls$choice.keepX
keepY <- tune.spls$choice.keepY

# final model
spls.res <- mixOmics::spls(X, Y, ncomp= 2, keepX= keepX, keepY= keepY)

# get clusters and plot longitudinal profile by cluster
spls.cluster <- getCluster(spls.res)
plotLong(spls.res)
```

---

unscale *Unscales a scaled data.frame*

Description

unscale is a generic function that unscale and/or uncenter the columns of a matrix generated by the scale base function

Usage

```
unscale(x)
```

Arguments

x	A numeric matrix.

Details

unscale uses attributes added by the scale function "scaled:scale" and "scaled:center" and use these scaling factor to retrieve the initial matrix. It first unscales and then uncenters.

Value

Return a matrix, uncenterd and unscaled. Attributes "scaled:center" and "scaled:scale" are removed.

See Also

scale

Examples

```
X <- matrix(1:9, ncol = 3)
X.scale <- scale(X, center = TRUE, scale = TRUE)
X.unscale <- unscale(X.scale)
all(X == X.unscale)
```

Index

block.pls, *4*
block.spls, *14*
bptest, *9*

dmatrix.spearman.dissimilarity, 2

get_demo_cluster, 6
get_demo_silhouette, 7
getCluster, 3, *4*, *10*, *14*, *17*
getNcomp, 4
getSilhouette, 5
getUpDownCluster, 6

lmms.filter.lines, 8

pca, *4*
plotLong, 9, *14*, *17*
pls, *4*
proportionality, 11

remove.low.cv, 12

scale, *18*
selectVar, *3*
silhouette, *4*
spls, *17*

tuneCluster.block.spls, 13
tuneCluster.spca, 14
tuneCluster.spls, 16

unscale, 17