Package 'gen3DNet'

May 15, 2024

Title An R Package for Generating 3D Network Models

Version 0.0.0.9000

Author Tina Tang

Maintainer Tina Tang <tang.tina@wustl.edu>

Description Generates 3D network from two correlated objects with shared common factors. Specifically, 'gen3DNet' builds the relationships between samples and shared factors using the non-negative matrix factorization, where three clustering techniques are evaluated to determine the number of functional modules. In addition, it builds the relationships between samples from the two distinct data objects based on a partial least squares regression model.

Imports plsdof, MASS, NbClust, cli, progress

Depends R (>= 3.1.0), NMF

License GPL (>= 2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

R topics documented:

`create_gen3DNet` *Create a general 3D network.*

Description

This function performs calculations to generate a 3D network. It is not meant to be called directly. You should use the gen3DNet function, which sanitizes arguments and reads in data from file paths. See help(gen3DNet) for further documentation.

Usage

```
create_gen3DNet(
  left,
  right,
  nmf_nrun = 10,
  k = c(),
  k_picker = max_ward_kl,
  seed = 123456,
  p_val_threshold = 1e-04,
  out_folder = "gen3DNet",
  verbose = TRUE
)
```

Arguments

`left`	Matrix of left objects
`right`	Matrix of right objects
`nmf_nrun`	Number of iterations to use for NMF
`k`	Number of clusters to use for NMF. This can be specified as a • single number, or • consecutive range. If unspecified, k is picked from 1 to min(num_cols - 1, num_rows - 1)
`k_picker`	Method for picking k. If unspecified, k values are compared using the KL-index of Ward clusterings based on euclidean distance. Possible values: • max_cophenetic • kneedle_silhouette_consensus • kneedle_cophenetic • max_silhouette_consensus • max_cophenetic • max_ward_kl To learn more, use help() for each of these functions.
`seed`	Seed to use for NMF.
`p_val_threshold`	Threshold for significant p-values in PLSR.
`out_folder`	Folder used for outputting results.
`verbose`	Whether to print output (default TRUE).

Examples

```
library("gen3DNet")
histon_path <- system.file("extdata", "histon_data.csv", package="gen3DNet")
phospho_path <- system.file("extdata", "phospho_data.csv", package="gen3DNet")
result <- gen3DNet(
   histon_path,
   phospho_path,
   nmf_nrun = 10,
   p_val_threshold = 0.01,
   # k_picker = max_cophenetic
   # k_picker = kneedle_silhouette_consensus
   # k_picker = kneedle_cophenetic
   # k_picker = max_silhouette_consensus
   # k_picker = max_cophenetic
   # k_picker = max_ward_kl
   k_picker = max_ward_kl
)
```

`gen3DNet` *Create a 3D network model.*

Description

This function creates a 3D network model relating two matrices. The columns of each matrix should represent two sets of objects, with the rows of both matrices corresponding to observations under the same circumstances. For example, the columns might be phosphoproteins and histones, and the rows might correspond to drugs. In this case, the data values would represent the transcription or GCP activation level in cells treated with different drugs.

Usage

```
gen3DNet(
  left,
  right,
  nmf_nrun = 10,
  k = NULL,
  k_picker = max_ward_kl,
  seed = 123456,
  p_val_threshold = 1e-04,
  out_folder = NULL,
  verbose = TRUE
)
```

Arguments

`left`	Matrix of left objects
`right`	Matrix of right objects
`nmf_nrun`	Number of iterations to use for NMF
`k`	Number of clusters to use for NMF. This can be specified as a • single number, or

- consecutive range. If unspecified, k is picked from 1 to min(num_cols - 1, num_rows - 1)

`k_picker` Method for picking k. If unspecified, k values are compared using the KL-index of Ward clusterings based on euclidean distance. Possible values:

- max_cophenetic
- kneedle_silhouette_consensus
- kneedle_cophenetic
- max_silhouette_consensus
- max_cophenetic
- max_ward_kl

To learn more, use help() for each of these functions.

`seed` Seed to use for NMF.

`p_val_threshold`
Threshold for significant p-values in PLSR.

`out_folder` Folder used for outputting results.

`verbose` Whether to print output (default TRUE).

Details

The final 3D network model consists of connections between the rows and both sets of columns, as follows:

- Connections between the rows and the left columns. These are found using NMF (nonnegative matrix factorization). The data can contain negative values, as an antilog transformation is used internally. The number of NMF clusters can be specified, but otherwise it will be chosen automatically based on the value that gives the highest KL-index when using Ward hierarchical clustering. The connection weight between a given left column and row is the maximum basis value for the left column object. See the function generate_nmf_modules for further information.
- Connections between the rows of the right and left columns. These are found using PLSR. The connection strength is the absolute value of the PLSR regression coefficient. See the function generate_plsr for more information.

Examples

```
library("gen3DNet")
histon_path <- system.file("extdata", "histon_data.csv", package="gen3DNet")
phospho_path <- system.file("extdata", "phospho_data.csv", package="gen3DNet")
result <- gen3DNet(
   histon_path,
   phospho_path,
   nmf_nrun = 10,
   p_val_threshold = 0.01,
   # k_picker = max_cophenetic
   # k_picker = kneedle_silhouette_consensus
   # k_picker = kneedle_cophenetic
   # k_picker = max_silhouette_consensus
   # k_picker = max_cophenetic
   # k_picker = max_ward_kl
   k_picker = max_ward_kl
)
```

`generate_nmf_modules` *Generate NMF Modules*

Description

This function uses NMF to generate functional modules representing interactions between the entities in the rows and columns of the dataframe. This has two components: Each entity is assigned to a cluster "Column signatures" are generated for each row object. For example, if rows are histones and columns are drugs, this function will generate "histone signatures" that express Because NMF relies on knowing the number of clusters (k) in advance, this function also determines this automatically. Currently, the default option uses the number of clusters that produces the lowest KL index with Ward clustering. The package depends on NbClust to implement this feature. However, the user can override this option with a custom function.

Usage

```
generate_nmf_modules(
  left_data,
  nmf_nrun,
  k_range,
  k_picker = gen3DNet::max_kl_ward,
  seed,
  verbose = FALSE
)
```

Arguments

`left_data`	This is the data that will be clustered using NMF. Signatures will be generated for the object type that is represented by rows.
`nmf_nrun`	The number of r
`k_range`	Either a) A consecutive range of possible k values, b) a single k value, or c) nothing, meaning that generate_nmf_modules will assume the range is from 2 to the number of rows.
`k_picker`	Any function that takes a dataframe and a consecutive range of potential k values
`seed`	The seed to use with NMF
`verbose`	Whether to print output.

`generate_plsr` *Identify significant PLSR coefficients for pairs of left and right objects.*

Description

This function computes PLSR coefficients and p-values between the rows of the two matrices provided ("left" and "right"). This regression is performed separately for each left item; i.e. in each iteration, a separate PLS regression is performed which explains

Usage

```
generate_plsr(left, right, p_val_threshold = 1e-04, verbose = FALSE)
```

Arguments

left	Matrix of left objects
right	Matrix of right objects
p_val_threshold	Threshold for significant p-values in PLSR.
verbose	Whether to print output.

Details

The maximum number of components for PLSR is the number of columns in the right matrix or the number of rows minus one, whichever is less.

Value

A list containing the following named values: p_list - A list containing left-right pairs with significant values (above p_val_threshold) non_p_list - A list containing left-right pairs with non-significant values comp - the maximum number of PLS components.

---

kneedle *Find the knee/elbow point in a vector using the Kneedle algorithm.*

Description

This function uses the Kneedle algorithm (Satopaa 2011) to find the index of the knee point in the provided vector. If the values are mostly increasing, use sign = 1. If they are mostly decreasing, use sign = -1.

Usage

```
kneedle(values, sign)
```

Arguments

values	The values to find a knee/elbow in.
sign	-1 if the values are mostly decreasing, 1 if the values are mostly decreasing.

Value

The index of the knee/elbow.

kneedle_cophenetic *Pick k based on the knee point in the cophenetic correlation*

Description

This function uses the knee point (Satopaa 2011) of the cophenetic correlation (Lessig 1972), which is implemented in the NMF library (Gaujoux 2010).

Usage

```
kneedle_cophenetic(data, k_range)
```

Arguments

data	The data with an unknown number of clusters.
k_range	The range of possible k values

Value

The chosen value of k.

kneedle_silhouette_consensus *Pick k based on the knee point in the silhouette score*

Description

This function uses the knee point (Satopaa 2011) of the average silhouette width (Lessig 1972), which is implemented in the NMF library (Gaujoux 2010).

Usage

```
kneedle_silhouette_consensus(data, k_range)
```

Arguments

data	The data with an unknown number of clusters.
k_range	The range of possible k values

Value

The chosen value of k.

max_cophenetic *Pick k based on the maximum cophenetic score*

Description

This function uses the cophenetic correlation (Lessig 1972), which is implemented in the NMF library (Gaujoux 2010).

Usage

```
max_cophenetic(data, k_range)
```

Arguments

data	The data with an unknown number of clusters.
k_range	The range of possible k values

Value

The chosen value of k.

max_silhouette_consensus *Pick k based on the maximum silhouette score*

Description

This function uses the average silhouette width (Rousseeuw 1987), as implemented in the NMF library (Gaujoux 2010).

Usage

```
max_silhouette_consensus(data, k_range)
```

Arguments

data	The data with an unknown number of clusters.
k_range	The range of possible k values

Value

The chosen value of k.

max_ward_kl *Pick k based on the maximum KL index of Ward clusterings*

Description

This function uses the NbClust package (Charrad 2014) to estimate the number of clusters present.

Usage

```
max_ward_kl(data, k_range)
```

Arguments

data	The data with an unknown number of clusters.
k_range	The range of possible k values

Details

First, the data is repeatedly clustered with 1 cluster, then 2, and so on, using the Ward hierarchical clustering algorithm (Ward 1963). Afterwards, each set of clusters is scored using the Krzanowski-Lai index (Krzanowski 1988). The number of clusters with the highest index is then chosen.

Value

The chosen value of k.

normalize_nmf *Normalize NMF*

Description

This function applies an antilog transformation to the data, ensuring that all values are positive, and then scales the result from 0 to 1.

Usage

```
normalize_nmf(left_data)
```

Arguments

left_data	The data to be normalized

scale_0_1 *Scale 0 to 1*

Description

This function scales data from 0 to 1.

Usage

```
scale_0_1(x)
```

Arguments

x	the data to be scaled from 0 to 1.

user_pick_k *User picks k*

Description

This is a k_picker callback that simply plots the performance of NMF on every number of clusters in k_range, and asks the user for input.

Usage

```
user_pick_k(nmf_data, k_range)
```

Arguments

nmf_data	The data whose number of clusters is in question
k_range	The numbers of clusters (k values) being considered.

write_all_to_disk *Write a nested list as nested folders, with data.frame objects as csvs.*

Description

Write a nested list as nested folders, with data.frame objects as csvs.

Usage

```
write_all_to_disk(name, object, folder)
```

Arguments

name	The name of the top-level folder to be written
object	The nested list
folder	The parent folder where the top-level folder should be placed.

Examples

```
# For example, if the object is
object = list(
    item1 = list(
        df1 = data.frame(a=c(1,2),b=c(3,4)),
        df2 = data.frame(a=c(1,2),b=c(3,4))
    ),
    item2 = list(
        df3=data.frame(a=c(1,2),b=c(3,4))
    ),
    df4 = data.frame(a=c(1,2),b=c(3,4))
)
# and the call is
write_all_to_disk("toplevelfolder", object, ".")
# The following files and folders would be written:
# toplevelfolder/
#     item1/
#         df1.csv
#         df2.csv
#     item2/
#         df3.csv
#     df4.csv
#
```

Index