geNet package

September 26, 2020

R topics documented:

check_input_geNet *checking input df Function*

Description

function to check the format of the input binary df. It is automatically called by the geNet() function.

Usage

```
check_input_geNet(binary_df)
```

Arguments

binary_df	a binary dataframe to be checked. Mandatory argument.

Value

None

Examples

```
## Not run: check_input_df(binary_df)
```

check_input_visualization *check_input_visualization*

Description

function to check the format of the input binary df. It is automatically called by the geNet() function.

Usage

```
check_input_visualization(data)
```

Arguments

data	output of the geNet() function (list of two ffdf objects)

Value

error message if checking fails

Examples

```
## Not run: check_input_visualization(data)
```

`contract_network` *contract_network*

Description

function to contract the visnetwork dataframe. It is automatically called by the plot_visnetwork() function if contract_net=T.

Usage

```
contract_network(data, seed = 123, show_label = F, show_legend = T)
```

Arguments

`data`	visnetwork dataframe generated by the geNet algorithm. Mandatory argument.
`seed`	set seed of the nodes
`show_label`	should the labels of the nodes be visible? default to False
`show_legend`	should the legend be visible? default to True

Value

list of two objects:

- visnet: object of class "visNetwork" generated from the contracted data.
- data_contracted: a compressed version of the input data object. It is generated by contracting the nodes of each cluster in one vertex.

This dataset reports only the nodes contained in each cluster and the connections between the clusters

`export_geNet_result` *Export geNet results*

Description

function to export the geNet output. It is automatically called by the geNet() function if export_results=T

Usage

```
export_geNet_result(geNet_output, out_directory, format = "csv")
```

Arguments

`geNet_output`	the object generated by the geNet function. It is a list of two ffdf objects. • nodes: ffdf object containing the information about the nodes • edges: ffdf object containing the information about the edges
`out_directory`	the directory to write the output
`format`	the user can export the geNet output in .csv format or in .ffdata format. In both cases, two files are generated. One containing the nodes information (e.g., clustering association), and the other one containing edges information (e.g.,connections weight). Default to "csv".

Value

In case of the csv format, two csv files are generated in the output directory

- nodes_geNet_output.csv: it contains the information about the nodes
- edges_geNet_output.csv: it contains the information about the edges

Examples

```
## Not run: export_geNet_result(geNet_output,out_directory="./",format="csv")
# export output in current working directory "./"

## End(Not run)
```

`filter_id` *function to filter the genes ID*

Description

function to filter the genes ID from data. It is called automatically by the visualize_network() function

Usage

```
filter_id(data, selected_id)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function). Mandatory argument
`selected_id`	select the ids to filter data. Optional argument.

Value

data: data object filtered

`final_score` *function to format the final df of scores*

Description

function to format the final df of scores. The adjustment and filtering steps are executed. It is automatically called by the geNet() function

Usage

```
final_score(
  final_df_phi_score,
  sel_weight = "coeff",
  pval_thr_pos = 0.01,
  pval_thr_neg = 0.1
)
```

Arguments

`final_df_phi_score`	the dataframe of scores generated by the generate_final_df() function. Mandatory argument
`sel_weight`	select the type of weights of the edges. Possible values: logpvalue: choose the negative log-pvalues as weights of the edges coeff: choose the correlation coefficients as weights of the edges Default to "coeff"
`pval_thr_pos`	threhold p-value positive edges. Default to 0.01.
`pval_thr_neg`	threshold p-value negative edges. Default to 0.1.

Value

Object of class "ffdf", containing the connections between the nodes. The connections have been filtered and correctly formatted to be converted in an igraph object

Examples

```
## Not run: final_score(final_df_phi_score,sel_weight="coeff")
```

generate_df_scores *generate_df_scores*

Description

function to generate the weights of the edges given the combinations. It is automatically called by the geNet() function. the correlation coefficient and its pvalue is calculated using the cor.test function from the stat package. See the relative documentation for additional details.

Usage

```
generate_df_scores(all_combs, binary_matrix, signif_test = "cor_test")
```

Arguments

`all_combs`	matrix of combinations
`binary_matrix`	binary matrix of presence/absences genes
`signif_test`	significance test to use. • cor_test: use Pearson correlation test approach (faster) • chisquare: use chi-square test approach (slower) Read vignettes for additional details about the significance tests used.

Value

Object of class "dataframe" containing the connections between the nodes and relative weights

generate_final_df *generate_final_df function to generate the final df of scores based on the weights of edges generated by the generate_df_scores_ff() function. It is called automatically by the geNet() function*

Description

generate_final_df

function to generate the final df of scores based on the weights of edges generated by the generate_df_scores_ff() function. It is called automatically by the geNet() function

Usage

```
generate_final_df(binary_matrix, n_cores = 1, test = "cor_test")
```

Arguments

`binary_matrix`	binary matrix of presence/absences genes
`n_cores`	number of cores to use. Default to 1.
`test`	significance test to use. • cor_test: use Pearson correlation test approach (faster) • chisquare: use chi-square test approach (slower)

Value

Object of class "ffdf" containing the connections between the nodes and relative weights

Examples

```
## Not run: generate_final_df(binary_matrix,progress=F,n_cores=1,test="cor_test")
```

`generate_freqs_genes` *Calculate genes frequency*

Description

function to calculate the percentage of occurrence of every gene across all strains.

Usage

```
generate_freqs_genes(input_binary_df)
```

Arguments

`input_binary_df`	Object of class "dataframe", containing the absence/presence of genes

Value

Object of class "dataframe". The first column reports the ID of the nodes/genes, the second column reports the frequency across all strains

Examples

```
## Not run: generate_freqs_genes(input_binary_df)
```

geNet *Execution geNet algorithm*

Description

function to execute the geNet algorithm on a binary dataframe of genes occurrences (presence/absence data)

Usage

```
geNet(
  input_binary_df,
  clust_method = "infomap",
  type_weight = "coeff",
  cores = 1,
  out_dir,
  export_results = F,
  test_pvalue = "cor_test",
  pval_thr_pos = 0.01,
  pval_thr_neg = 0.1
)
```

Arguments

`input_binary_df`	input binary dataframe containing genes occurrences. Mandatory argument. input_binary_df is an object of class "dataframe". The rownames are the strains names, the columns names are unique genes/nodes IDs.
`clust_method`	clustering method. Default to "infomap".
`type_weight`	possible values: • logpvalue: the weights of the edges will be the negative log adjusted p-values • coeff: the weights of the edges will be the correlation coefficients values Note: This is true only for the positive edges. The negative edges will have 0 weights in both cases. The negative edges doesn't influence the topology of the network or the clustering.
`cores`	number of cores to use for parallel processing
`out_dir`	if export_results=T, select output directory
`export_results`	exporting geNet output in csv files?
`test_pvalue`	significance test to use. • cor_test: use Pearson correlation test approach (faster) • chisquare: use chi-square test approach (slower)
`pval_thr_pos`	threhold p-value positive edges. Default to 0.01.
`pval_thr_neg`	threshold p-value negative edges. Default to 0.1.

Value

list of two objects of class "ffdf".

- nodes: reports info about the nodes and clustering
- edges: reports info about the connections between the nodes This format is designed to be used with the visNetwork package.

Examples

```
## Not run: geNet(input_binary_df,clust_method="infomap",type_weight="coeff",cores=4)
```

`gen_network_obj` *generate igraph network*

Description

function to generate the igraph network object based on the final scores. It is called automatically by the geNet() function.

Usage

```
gen_network_obj(final_df_score)
```

Arguments

`final_df_score`	the dataframe of scores generated by the final_score() function. Mandatory argument.

Value

Object of class "igraph"

`gen_visnetwork_data` *gen_visnetwork_data*

Description

function to get the object to generate the visnetwork. It is automatically called by the geNet() function.

Usage

```
gen_visnetwork_data(igraph_network)
```

Arguments

`igraph_network`	igraph object. Mandatory argument.

Value

list of two objects of class "ffdf".

- nodes: reports info about the nodes and clustering
- edges: reports info about the connections between the nodes This format is designed to be used with the visNetwork package.

`get_colors_based_on_gene_ID`

get_colors_based_on_gene_ID

Description

function to get the colors based on the ID of the nodes

Usage

```
get_colors_based_on_gene_ID(data, vec_gene_ids = "None", col_nodes = "black")
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`vec_gene_ids`	vector of genes IDs to color. Mandatory argument.
`col_nodes`	color of the nodes that match the specified IDs. Default to "black".

Value

Object of class "dataframe", which reports the new colors for each node of the network

Examples

```
## Not run: get_colors_based_on_gene_ID(data,vec_gene_ids="None",col_nodes="black")
```

`get_colors_based_on_size` *get_colors_based_on_size*

Description

function to get the colors based on the size of the nodes

Usage

```
get_colors_based_on_size(
  data,
  size_thr,
  direction = "less",
  col_nodes = "black"
)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`size_thr`	size threshold. Mandatory argument.
`direction`	values greater/less than (or equal) the selected threshold
`col_nodes`	color of the nodes that match the specified condition. Default to "black".

Value

data_new_colors: object of class "dataframe", which reports the new colors for each node of the network

Examples

```
## Not run: get_colors_based_on_size(data,size_thr=10,direction="less",col_nodes="black")
```

`get_colors_based_on_strains` *get new colors based on the strains*

Description

function to predict the colors based on the strains occurrence of genes i.e.,show nodes/genes associated with the selected strains (i.e., that gene is present in that strains)

Usage

```
get_colors_based_on_strains(
  input_binary_df,
  data,
  strains_names,
  col_nodes = "black"
)
```

Arguments

`input_binary_df`	Object of class "dataframe", containing presence/absence of genes. Mandatory argument.
`data`	visnetwork dataframe generated by the geNet algorithm. Mandatory argument.
`strains_names`	the vector containing the names of the strains to visualize. Mandatory argument.
`col_nodes`	Color of the nodes matched with the strains. Default to "black".

Value

Object of class "dataframe", which reports the new colors for each node of the network

Examples

```
## Not run: get_colors_based_on_strains(input_binary_df,data,strains_names=c("strain_x"),col_nodes="black")
```

get_gene_ID_based_on_tooltip

get_gene_ID_based_on_tooltip

Description

function to get the gene IDs based on the annotation (tooltip column)

Usage

```
get_gene_ID_based_on_tooltip(data, vec_tooltip)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`vec_tooltip`	vector of annotation to match with the gene IDs. Mandatory argument.

Value

Object of class "dataframe", which reports the genes ID for the specified annotation keywords.

Examples

```
## Not run: get_gene_ID_based_on_tooltip(data,vec_tooltip=c("protein","secretion"))
# Get the nodes IDs whose annotation reports the word "protein" and "secretion"

## End(Not run)
```

---

get_groups_based_on_annotation
get groups based on the annotation

Description

function to get the groups based on the annotation (tooltip column)

Usage

```
get_groups_based_on_annotation(data, select_annotation)
```

Arguments

data	List of two ffdf objects (format generated by the geNet() function)
select_annotation	vector of annotation words to match with the groups

Value

Object of class "vector", which reports the groups selected

Examples

```
## Not run: get_groups_based_on_tooltip(data,select_tooltip=c("protein","secretion"))
# Get the groups where the annotations words "protein" and "secretion" appear.

## End(Not run)
```

---

get_groups_based_on_clustering
get_groups_based_on_clustering

Description

function to get the groups based on a clustering algorithm Note: the color of the nodes is based on the group (using the function get_new_colors_based_on_groups()) to evaluate the clustering method we use the modularity measure. The modularity of a graph with respect to some division (or vertex types) measures how good the division is, or how separated are the different vertex types from each other.

Usage

```
get_groups_based_on_clustering(igraph_network, method = "infomap")
```

Arguments

`igraph_network`	igraph network generated by the gen_igraph_network() function. Mandatory argument.
`method`	clustering method: • louvain: predict clusters using the Louvain algorithm. The "modularity" is the objective function to maximize. Small clusters may be hidden. It detects only the bigger clusters. • infomap: predict clusters using the Infomap algorithm. The "map equation" is the objective function to minimize. Compared to the Louvain algorithm, it is able to identify smaller clusters (subclusters) • fastgreedy: predicts clusters using the Fastgreedy algorithm • walktrap: predicts clusters using the Walktrap algorithm Default to "infomap" to detect smaller clusters. The Louvain algorithm is faster than the Infomap algorithm in large networks.

Value

Object of class "dataframe", which reports the genes ID associated with the new groups

Examples

```
## Not run: get_groups_based_on_clustering(igraph_network,method="infomap")
```

`get_groups_based_on_gene_ID` *get_groups_based_on_gene_ID*

Description

function to get the groups based on the ID of the nodes

Usage

```
get_groups_based_on_gene_ID(data, vec_gene_ids)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`vec_gene_ids`	vector of genes IDs to which the groups must be matched. Mandatory argument.

Value

Object of class "dataframe", which reports the groups for the specified genes ID.

Examples

```
## Not run: get_groups_based_on_gene_ID(data,vec_gene_ids=c("group_200","group_253"))
# Get the groups for the nodes IDs "group_200" and "group_253"

## End(Not run)
```

`get_group_names_based_on_n_nodes`
get the group names based on the number of nodes

Description

function to get the group names based on the number of nodes in each group.

Usage

```
get_group_names_based_on_n_nodes(data)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)

Value

Object of class "dataframe", which reports the old names associated with the new names

Examples

```
## Not run: get_group_names_based_on_n_nodes(data)
```

`get_group_names_based_on_tooltip`
get group names based on annotation

Description

function to get the group names based on the annotation of the nodes (tooltip column)

Usage

```
get_group_names_based_on_tooltip(data)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)

Value

Object of class "dataframe", which reports the old names associated with the new names

Examples

```
## Not run: get_group_names_based_on_tooltip(data)
```

`get_legend_df` *get_legend_df*

Description

function to get the legend of the groups. It is automatically called by the plot_visnetwork() function.

Usage

```
get_legend_df(data)
```

Arguments

`data`	list of two ffdf objects generated by the geNet algorithm. Mandatory argument.

Value

Object of class "dataframe", which reports the legend information of the nodes

---

`get_mappings` *get_mappings*

Description

function to get the correct mapping to contract the network. It is automatically called by the contract_network() function.

Usage

```
get_mappings(igraph_net, data)
```

Arguments

`igraph_net`	igraph object. Mandatory argument.
`data`	list of two ffdf objects generated by the geNet algorithm. Mandatory argument.

Value

Object of "vector" containing the mapping codes.

`get_new_colors_based_on_groups`
get_new_colors_based_on_groups

Description

function to get the colors based on the groups

Usage

```
get_new_colors_based_on_groups(df_groups)
```

Arguments

`df_groups`	object of class "dataframe", which reports the genes ID associated with the groups. Mandatory argument.

Value

Object of class "dataframe", which reports the gene IDs and the groups associated with the colors

Examples

```
## Not run: get_new_colors_based_on_groups(df_groups)
```

`mod_color_layer` *mod_color_layer*

Description

function to modify the color layer (colors of the node)

Usage

```
mod_color_layer(data, data_new_colors, no_matching_col = "old")
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`data_new_colors`	dataframe that reports the new colors for each node
`no_matching_col`	default color in case of no matching (i.e. gene IDs of the network not reported in data_new_colors) • old: the color of no matching nodes is the old color. • white: the color of no matching nodes is white ("white" is a reserved color) Default to "white".

Value

data: data object modified with the new color layer

Examples

```
## Not run: mod_color_layer(data,data_new_colors,no_matching_col="white")
```

`mod_group_layer` *mod_group_layer*

Description

function to modify the group layer (influence also the structure of the contracted network)

Usage

```
mod_group_layer(data, data_new_groups = NULL)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`data_new_groups`	dataframe that reports the new group for each node

Value

data: data object modified with the new group layer

Examples

```
## Not run: mod_group_layer(data,data_new_groups)
```

`mod_group_names` *mod_group_names*

Description

function to modify the names of the group

Usage

```
mod_group_names(data, df_new_groups_names = NULL)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`df_new_groups_names`	dataframe that reports the new names for each old group name

Value

data: data object modified with the new group names for the current group layer

Examples

```
## Not run: mod_group_names(data,df_new_groups_names)
```

`mod_size_layer` *mod_size_layer*

Description

function to modify the size layer (size of each node)

Usage

```
mod_size_layer(data, data_new_size = NULL)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`data_new_size`	dataframe that reports the size for each node

Value

data: data object modified with the new size layer

Examples

```
## Not run: mod_size_layer(data,data_new_size)
```

`mod_tooltip_visnetwork` *mod_tooltip_visnetwork*

Description

function to modify the tooltip layer of the network

Usage

```
mod_tooltip_visnetwork(data, data_new_tooltip = NULL)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function)
`data_new_tooltip`	dataframe that reports the new tooltip string for each node

Value

data: data object modified with the new tooltip layer

Examples

```
## Not run: mod_tooltip_visnetwork(data,data_new_tooltip)
```

`plot_visnetwork` *plot_visnetwork*

Description

function to plot the visnetwork object based on the data input

Usage

```
plot_visnetwork(
  data,
  show_label = F,
  show_legend = T,
  seed = 123,
  name_net = "None",
  show_negative = F,
  contract_net = T,
  size_opt = "fixed"
)
```

Arguments

`data`	list of two ffdf objects generated by the geNet algorithm. Mandatory argument.
`show_label`	show the labels of the nodes?
`show_legend`	show the legend of groups?
`seed`	set the seed of the current instance
`name_net`	name of the network
`show_negative`	show negative edges?
`contract_net`	Should the network be contracted? Default to True
`size_opt`	• fixed: the size of the node is equal to 8 (size layer values ignored) • size_opt: the size of the node is proportional to the size layer values Default to "fixed". Note: if the there are too many nodes the size of the node is reduced regardless the option specified by the user.

Value

data: data object modified with the new size layer

Examples

```
## Not run:
plot_visnetwork(data,show_label=F,
show_legend=T,seed=123,name_net="None",
show_negative=F,contract_net=T,
size_opt="fixed")

## End(Not run)
```

`visualize_network` *visualize_network*

Description

function to visualize the network based on the connections and different conditions

Usage

```
visualize_network(
  data,
  select_group = NULL,
  select_tooltip = "None",
  select_size = "None",
  clust_method = "None",
  select_ID = NULL
)
```

Arguments

`data`	List of two ffdf objects (format generated by the geNet() function). Mandatory argument
`select_group`	select the groups to filter data. Optional argument.
`select_tooltip`	select the annotation keywords to filter data. Optional argument. Note: the tooltip values are the values that appear when hovering the nodes
`select_size`	select the size of the nodes to filter data. Optional argument.
`clust_method`	select the clustering method. Default to Infomap.
`select_ID`	select the IDs to filter data. Optional argument.

Value

data: data object filtered

Examples

```
## Not run:
data_subsetted<-visualize_network(data,select_group=c("group_1","group_5"))
data_subsetted<-visualize_network(data,select_tooltip=c("protein","secretion"))
data_subsetted<-visualize_network(data,clust_method="louvain")

## End(Not run)
```

Index