# Package 'foodingraph'

October 13, 2022

**Type** Package

**Title** Food Network Inference and Visualization

**Version** 0.1.0

**Maintainer** Victor Gasque <victor.gasque@protonmail.com>

**Description** Displays a weighted undirected food graph from an adjacency matrix.
Can perform confidence-interval bootstrap inference with mutual information
or maximal information coefficient.
Based on my Master 1 internship at the Bordeaux Population Health center.
References : Reshef et al. (2011) <doi:10.1126/science.1205438>,
Meyer et al. (2008) <doi:10.1186/1471-2105-9-461>,
Liu et al. (2016) <doi:10.1371/journal.pone.0158247>.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** dplyr, ggplot2, cowplot, magrittr, stringr, tibble, tidyr,
viridis, igraph, ggraph, minerva, rlang, labeling, grid

**Suggests** knitr, infotheo, minet,

**VignetteBuilder** knitr

**URL** https://github.com/vgasque/foodingraph/

**BugReports** https://github.com/vgasque/foodingraph/issues

**RoxygenNote** 6.1.1

**NeedsCompilation** no

**Author** Victor Gasque [cre, aut],
Boris Hejblum [aut],
Cecilia Samieri [aut]

**Repository** CRAN

**Date/Publication** 2019-10-06 11:30:08 UTC

# R topics documented:



---

`boot_cat_bin` *Bootstrap inference on binary and categorical variables*

---

## Description

For a given dataset, performs a confidence-interval bootstrap of the mutual information or maximal information coefficient (MIC) for each pairwise association.

1. Computes the MI or MIC for each pairwise association.
2. Performs a bootstrap (of `boots` samples), and store each pairwise association
3. Calculate the 1th percentile for each pairwise association from the bootstrap distribution
4. If the percentile is inferior to the threshold of the corresponding pairwise variable type, then the MI or MIC is set to 0.

## Usage

```
boot_cat_bin(obs_data, list_cat_var, list_bin_var, threshold_bin,
  threshold_cat, threshold_bin_cat, method = c("mi", "mic"),
  boots = 5000, show_progress = TRUE)
```

## Arguments

| | |
|---|---|
| `obs_data` | (data.frame or matrix) : a dataset which rows are observations and columns the variables. |
| `list_cat_var` | : list of the categorical variables of the dataset |
| `list_bin_var` | : list of the binary variables of the dataset |
| `threshold_bin` | : the threshold to apply to binary pairwise associations |
| `threshold_cat` | : the threshold to apply to categorical pairwise associations |
| `threshold_bin_cat` | : to apply to a pairwise association between a binary and a categorical variable |

| | |
|---|---|
| method | : the method to use to compute the adjacency matrix ("mi" or "mic"). If "mi", uses mutual information package minet, and Miller-Madow estimator. If "mic", uses maximal information coefficient from minerva package function cstats() |
| boots | : number of bootstraps (default 5000) |
| show_progress | : if TRUE, prints the percentage of completion to keep track of the algorithm's progress. Default is TRUE. Recommended to FALSE for RMarkdown files. |

## Value

The inferred adjacency matrix. All bootstrap 1th percentile values of each pairwise association inferior to their predefined thresholds will be set to 0.

---

`boot_simulated_cat_bin`

*Confidence-interval bootstraps on simulated independent variables*

---

## Description

Create a defined number of simulated independent random variables of a given size according to type : 2 ordinal variables, 2 binary variables, 1 binary and 1 ordinal variable. A number of bootstraps are then performed on the sample to calculate a confidence interval of the bootstrap distribution of the chosen method: mutual information or the maximal information coefficient. The percentile method is used to calculate this interval.

## Usage

```
boot_simulated_cat_bin(type = c("cat", "bin", "bincat"),
  method = c("mic", "mi"), simu = 10, boots = 5000, size = 500,
  percentile = 0.99)
```

## Arguments

| | |
|---|---|
| type | : the type of the simulated variables: cat is for 2 ordinal variables, bin is for 2 binary variables, bincat is for 1 binary and 1 ordinal variable. |
| method | : the method used to calculate the association : mutual information (mi), or the maximal information coefficient (mic). |
| simu | : the number of simulated pairs of variables. For each pair, the confidence-interval bootstrap is calculated from the bootstrap distribution of the MI/MIC of between the two pairs. At the end of the program, the mean of the chosen percentile is given. Default is 10. |
| boots | : the number of bootstraps per simulation. Default is 5000. |
| size | : the size of the sample. Default is 500. |
| percentile | : the percentile kept. Default is 0.99 (the 99th percentile). |

## Value

The mean of the percentile values.

## References

Reshef et al. (2011) <doi:10.1126/science.1205438>

Meyer et al. (2008) <doi:10.1186/1471-2105-9-461>

## Examples

```
boot_simulated_cat_bin("cat", "mic", 2, 500)
```

---

`compare_graphs` *Compare two graphs*

---

## Description

From two graphs generated by `graph_from_matrix` or `graph_from_links_nodes`, displays two graphs with the same legend (edge weights and size and node degrees) to facilitate the visual comparison of the two graphs. NB : if you use node families, make sure they have the same families in the two graphs (this can be done by generating a same palette for both graphs using `family_palette`)

## Usage

```
compare_graphs(graph1, graph2, titles = NULL, position = c("vertical",
  "horizontal"), n_nodes = 5, n_weights = 5,
  edge_width_range = c(0.2, 2), edge_alpha_range = c(0.4, 1),
  node_size_range = c(1, 10), unique_legend = TRUE)
```

## Arguments

| | |
|---|---|
| `graph1` | : the first graph |
| `graph2` | : the second graph |
| `titles` | (optional) : list of 2 : the two title of the graphs. Default are the graph titles from `graph_from_matrix` or `graph_from_links_nodes`. |
| `position` | : should the graphs be displayed vertically (use `vertical`, default) or horizontally (use `horizontal`). |
| `n_nodes` | : the number of nodes to be displayed in the legend. R will do its best to be around this number. |
| `n_weights` | : the number of weights to be displayed in the legend. R will do its best to be around this number. |
| `edge_width_range` | : range of the edges width (default is 0.2 to 2). |

edge_alpha_range
: if edge_alpha is TRUE, the range of the alpha values (between 0 and 1). Default is 0.4 to 1.

node_size_range
: range of the node sizes. (default is 1 to 10)

unique_legend : should there be a unique legend (default is TRUE) BE CAREFUL to have the same family colors if you use families/

---

`family_palette` *Generate a color palette for nodes family*

---

## Description

From a list of food families, create a color for each family.

## Usage

```
family_palette(family)
```

## Arguments

| | |
|---|---|
| `family` | (list) : can be either the family column from the legend table, or just a list of the families. In all cases, the parameter will be converted as a factor and sorted (alphabetically or numerically) Only its unique values are necessary. |

## Details

Very useful when comparing graphs with the same families. It can be used by itself, but this function was created to be the `family_palette` argument when calling `display_graph_from_links_nodes()` The colors will be automatically added to the graph (nodes and legend)

## Value

A list of key and values. - keys are the family names - values are the color

## Examples

```
family_palette(c("Fruits", "Vegetables", "Meats"))
```

foodingraph *foodingraph: a package for food network inference*

## Description

The foodingraph package provide two categories of functions :

1. confidence-interval (CI) bootstrap inference of mutual information (MI) or maximal information coefficient (MIC) adjacency matrices.
2. network visualization in a graph using `igraph` and `ggraph`

## CI bootstrap network inference

The two functions are

1. `boot_cat_bin` : a function to perform the CI bootstrap inference for pairwise associations between ordinal and binary variables. It uses thresholds defined by simulation of independent associations using `boot_simulated_cat_bin`, such that it simulates independent associations between ordinal-ordinal, binary-binary and ordinal-binary pairs of variables. It calculates the CI bootstraps for each pairwise association of the variables' dataset, then compares the 1st percentile of these CI to the corresponding thresholds of independent data.
2. `boot_simulated_cat_bin` : a function to determine the threshold values of MI or MIC of independent pairs of variables (ordinal vs. ordinal, and binary vs binary and ordinal vs. binary). It calculates the CI bootstraps of MI or MIC for these pairs of variables, and return a defined percentile of these CI (e.g. 99th percentile).

## Network visualization

The three main functions are

1. `graph_from_matrix` : create a graph from an adjacency matrix. This function need at least two arguments : **1.** the adjacency matrix, in which the column names and row names are the node names. **2.** the legend, which is a data frame of at least two columns : `name` (the name of the nodes in the adjacency matrix, e.g. CRUDSAL_cat) and `title` (the titles for each name, e.g. raw vegetables)
   Optionally, you can add a column `family` to specify the nodes' families.
2. `graph_from_links_nodes` : create a graph from a list of nodes and links. This function needs two arguments : **1.** the list of nodes and links, which should be the result from `links_nodes_from_mat` (if not, make sure the structure corresponds). **2.** the legend (described above).
3. `compare_graphs` : a function to compare two graphs. It unifies the legends and attributes, so the graphs can be visually comparable.
4. `save_graph` : a function to save the graph in a file at high resolution.

### Utils functions

Other functions include

1. family_palette : to create a color palette to be used in the graph. It is usually done automatically, but can prove useful if comparing multiple graphs, to ensure the family colors remain the same throughout the graphs.
2. links_nodes_from_mat : to extract the links and nodes from an adjacency matrix
3. mic_adj_matrix : using the cstats function from the minerva package, calculate the adjacency MIC matrix.

---

## graph_from_links_nodes

*Display a graph from a list of links and nodes*

---

### Description

Given a list of links and nodes (e.g. from extract_links_nodes func) Uses igraph and ggraph to display the network plots Must have the proper structure OR use extract_links_nodes(), which automatically returns this structure when given an adjacency matrix and its legend (see documentation for this function) network_data should be a list of 2 : edges, nodes For edges (data.frame) : from, to, weight, width, sign (of the weight: neg/pos) For nodes (data.frame) : name, title, family, family_color (optional)

### Usage

```
graph_from_links_nodes(network_data, main_title = "",
  node_type = c("point", "label"), node_label_title = TRUE,
  family_palette = NULL, layout = "nicely", remove_null = TRUE,
  edge_alpha = TRUE, edge_color = c("#6DBDE6", "#FF8C69"),
  edge_width_range = c(0.2, 2), edge_alpha_range = c(0.4, 1),
  node_label_size = 3, legend_label_size = 10, ...)
```

### Arguments

| | |
|---|---|
| network_data | (list of two) : links, nodes with the proper structure |
| main_title | (string, optional) : the title of the network |
| node_type | : point (default) for the graph to display points and the label outside the point, or label to have a node which is the label itself (the text size will then be associated to the node degree) |
| node_label_title | (bool, default F) : should the node labels be the names or title column? (e.g. names : CRUDSAL_cat, title : Raw vegetables) |
| family_palette | (list of key = value) : the keys are the family codes (from family column in the legend), and the values are the corresponding colors. Can be generated using the family_palette func. USEFUL if there is a need to compare multiple graphs of the same families, so the color is consistent. If NULL (default), the palette will be automatically generated using viridis |

`layout` (chr) : the layout to be used to construct the graph

`remove_null` (bool) : should the nodes with 0 connections (degree 0) be removed from the graph. default is TRUE.

`edge_alpha` (bool) : should the edges have a transparent scale? In addition to the width scale.

`edge_color` (list) : list of 2. The first element is the color of the negative edges, the second the positive. Default is `c("#6DBDE6", "#FF8C69")`.

`edge_width_range` : range of the edges width. (default is 0.2 to 2)

`edge_alpha_range` : if `edge_alpha` is TRUE, the range of the alpha values (between 0 and 1). Default is 0.4 to 1.

`node_label_size` : the size of the node labels. Default is 3.

`legend_label_size` : the size of the legend labels. Default is 10.

`...` : other parameters to pass to ggraph `create_layout`

## Value

a list of 3 : `igraph` : the igraph object, `net` the graph, `deg` the degree table.

## References

Csardi et al. (2006) <https://igraph.org>

Perdersen (2019) <https://ggraph.data-imaginist.com>

## See Also

`graph_from_matrix`

## Examples

```
adj_matrix <- cor(iris[,-5])
legend <- data.frame(name = colnames(iris[,-5]),
                     title = colnames(iris[,-5]))
graph_iris <- links_nodes_from_mat(adj_matrix, legend)
graph_from_links_nodes(graph_iris, main_title = "Iris graph")
```

graph_from_matrix *Display a graph from an adjacency matrix*

## Description

Given an adjacency matrix and a legend, displays the graph. This is a shortcut function, rather than using `links_nodes_from_mat()` and `graph_from_links_nodes()`.

## Usage

```
graph_from_matrix(adjacency_matrix, legend, threshold = 0,
  abs_threshold = TRUE, filter_nodes = TRUE, main_title = "",
  node_type = c("point", "label"), node_label_title = TRUE,
  family_palette = NULL, layout = "nicely", remove_null = TRUE,
  edge_alpha = TRUE, edge_color = c("#6DBDE6", "#FF8C69"),
  edge_width_range = c(0.2, 2), edge_alpha_range = c(0.4, 1),
  node_label_size = 3, legend_label_size = 10, ...)
```

## Arguments

`adjacency_matrix`
: a matrix of size n x n, each element being a number explaining the relationship (coefficient, information) between two variables given in the column and row names /!\ As this code is to draw undirected graphs, only the lower triangular part of association matrix is used to extract the information

`legend`
: a data frame of columns in order : 1) name, str : name of the node in the adjacency matrix, e.g. CRUDSAL_cat 2) title, str : name of the node, e.g. Raw vegetables 3) family, factor : (optional) the family the node belongs to, e.g. Vegetables

`threshold`
numeric) : a number defining the minimal threshold. If the weights are less than this threshold, they will be set to 0.

`abs_threshold`
(bool) : should the threshold keep negative values, e.g. if `abs_threshold` is set to `TRUE`, and threshold is set to 0.1, all weights between -0.1 and 0.1 will be set to 0

`filter_nodes`
(bool) : should the variables not in the adjacency matrix be displayed on the graph? Default is TRUE CAREFUL : if set to `TRUE`, be sure to have the same colors in the family legend of the graphs. A fixed palette can be set using `family_palette`. Default is TRUE.

`main_title`
(string, optional) : the title of the network

`node_type`
: `point` (default) for the graph to display points and the label outside the point, or `label` to have a node which is the label itself (the text size will then be associated to the node degree)

`node_label_title`
(bool, default F) : should the node labels be the names or title column? (e.g. names : CRUDSAL_cat, title : Raw vegetables)

`family_palette` (list of key = value) : the keys are the family codes (from family column in the legend), and the values are the corresponding colors. Can be generated using `family_palette`. USEFUL if there is a need to compare multiple graphs of the same families, so the color is consistent. If NULL (default), the palette will be automatically generated using viridis

`layout` (chr) : the layout to be used to construct the graph

`remove_null` (bool) : should the nodes with 0 connections (degree 0) be removed from the graph. Default is TRUE.

`edge_alpha` (bool) : should the edges have a transparent scale? In addition to the width scale.

`edge_color` (list) : list of 2. The first element is the color of the negative edges, the second the positive. Default is `c("#6DBDE6", "#FF8C69")`.

`edge_width_range` : range of the edges width. (default is 0.2 to 2)

`edge_alpha_range` : if `edge_alpha` is TRUE, the range of the alpha values (between 0 and 1). Default is 0.4 to 1.

`node_label_size` : the size of the node labels. Default is 3.

`legend_label_size` : the size of the legend labels. Default is 10.

`...` : other parameters to pass to ggraph 'create_layout'

## Value

a list of 3 : `igraph` : the igraph object, `net` the graph, `deg` the degree table.

## References

Csardi et al. (2006) <https://igraph.org>

Perdersen (2019) <https://ggraph.data-imaginist.com>

## See Also

`graph_from_links_nodes`

## Examples

```
adj_matrix <- cor(iris[,-5])
legend <- data.frame(name = colnames(iris[,-5]),
                     title = colnames(iris[,-5]))
graph_from_matrix(adj_matrix, legend, main_title = "Iris graph")
```

`links_nodes_from_mat` *Extracts links and nodes*

## Description

From an adjacency matrix, extracts two data.frames/tibbles

1. Links. columns : from, to, with, weight
2. Nodes. columns : name, title. name corresponds to the names used in 'from' and 'to'

## Usage

```
links_nodes_from_mat(adjacency_matrix, legend, threshold = 0,
  abs_threshold = TRUE, filter_nodes = TRUE)
```

## Arguments

| | |
|---|---|
| `adjacency_matrix` | : a matrix of size n x n, each element being a number explaining the relationship e.g. coefficient, information between two variables given in the column and row names /!\ As this code is to draw undirected graphs, only the lower triangular part of adjacency matrix is used to extract the information. |
| `legend` | : a data frame of columns in order : 1) name, str : name of the node in the adjacency matrix, e.g. CRUDSAL_cat 2) title, str : name of the node, e.g. Raw vegetables 3) family, factor : (optional) the family the node belongs to, e.g. Vegetables |
| `threshold` | numeric) : a number defining the minimal threshold. If the weights are less than this threshold, they will be set to 0. |
| `abs_threshold` | (bool) : should the threshold keep negative values, e.g. if `abs_threshold` is set to `TRUE`, and threshold is set to 0.1, all weights between -0.1 and 0.1 will be set to 0 |
| `filter_nodes` | (bool) : should the variables not in the adjacency matrix be displayed on the graph? Default is TRUE CAREFUL : if set to `TRUE`, be sure to have the same colors in the family legend of the graphs. A fixed palette can be set using the `family_palette` func. |

## Value

A list of two data frames : links and nodes.

## Examples

```
adj_matrix <- cor(iris[,-5])
legend <- data.frame(name = colnames(iris[,-5]),
                     title = colnames(iris[,-5]))
links_nodes_from_mat(adj_matrix, legend)
```

## mic_adj_matrix *Computes a MIC adjacency matrix*

### Description

For a given dataset, computes the adjacency matrix of maximal information coefficient (MIC) of each pairwise association. NOTE : another approach could have been to give the whole data frame to the `minerva` package func `cstats()`, but it seemed slower in my tests.

### Usage

```
mic_adj_matrix(obs_data)
```

### Arguments

| | |
|---|---|
| `obs_data` | (data.frame or matrix) : a dataset which rows are observations and columns the variables. |

### Value

the adjacency matrix of MIC values for each pairwise association.

### References

Reshef et al. (2011) <doi:10.1126/science.1205438>

### Examples

```
mic_adj_matrix(iris[,-5])
```

## save_graph *Save graph*

### Description

Save the graph generated from `graph_from_matrix` or `graph_from_links_nodes` or `compare_graphs`.

### Usage

```
save_graph(graph, filename = "foodingraph_%03d.png", width = NULL,
  height = NULL, dpi = 300, ...)
```

## Arguments

| | |
|---|---|
| graph | : the graph |
| filename | (optional) : the name of the file and format. Default is "foodingraph_*.png". |
| width | (optional) : width of the image in cm. Default is 25 cm for a single graph or a comparison in a vertical position. For a comparison in an horizontal position, 40cm. |
| height | (optional) : height of the image in cm. Default is 20 cm for a single graph, 25cm for a comparison in an horizontal position. For a comparison in a vertical position, 40cm. |
| dpi | (optional) : the resolution of the image in dpi. Default is 300 |
| ... | : other parameters to pass to the ggsave ggplot2 function |

## Examples

```
adj_matrix <- cor(iris[,-5])
legend <- data.frame(name = colnames(iris[,-5]),
                     title = colnames(iris[,-5]))
graph_iris <- graph_from_matrix(adj_matrix, legend, main_title = "Iris graph")

# Save to a in a temporary file location
save_graph(graph_iris, tempfile(fileext = ".png"))
```

# Index