Using the kcorebip package

Version1.0

Javier Garcia-Algarra
jgalgarra@gmail.com

March 13, 2025

Contents

1 Introduction

This package performs the *k-core decomposition* analysis of a bipartite graph and provides two new ways to plot it: *polar* and *ziggurat*. It works for any kind of bipartite network, but as it was developed to study ecological mutualistic communities you will find terminology and examples of that research field throughout this text[GA+18; GA+17].

2 Install kcorebip

To install `kcorebip` the package `devtools` should be available in your computer. Load it with

```
library("devtools")
```

and then issue the following command:

```
install_github("jgalgarra/kcorebip")
```

Warning. A `gcc` compiler is required to install the packge. If you are a developer and you know you do have it, `kcorebip` will install smoothly. Otherwise, you will get a message telling that `Rtools` is not available. In that case follow the instructions of the following page to install `Rtools`:

https://cran.r-project.org/bin/windows/Rtools/

3 Decomposition and definition of k-magnitudes

The *k-decomposition* is an iterative algorithm that prunes links of nodes with degree equal or less than k [Sei83]. The process starts pruning nodes with degree 1 until all the remaining nodes have two or more links. Then it with $k = 2$, and so on. After performing the *k-decomposition*, each species belongs to one of the *k-shells*. The highest value of k, i.e ks_{max}, corresponds to the innermost *core* $ks_{max} \equiv C^{A,B}$. For each *k-shell* there are two subsets, one per guild (A and B), that we call K_j^A, K_j^B where j is the *k-shell* index.

In order to quantify the distance from a node to the innermost shell of the partner guild, we compute the average of the shortest paths to eahc node within that set. We define the k_{radius} of node m of class A as the average distance to the species of C^B:

$$k^A_{radius}(m) = \frac{1}{\mid C^B \mid} \sum_{j \in C^B} dist_{mj} \qquad m \in A \tag{1}$$

where $dist_{mj}$ is the shortest path from species m to each of the j species that belong to C^B. The same definition is valid for species of guild B computing the average distance to C^A instead. The minimum possible radius value is 1 for a node of the maximum shell that is directly linked to each node of the maximum shell set of the opposite guild.

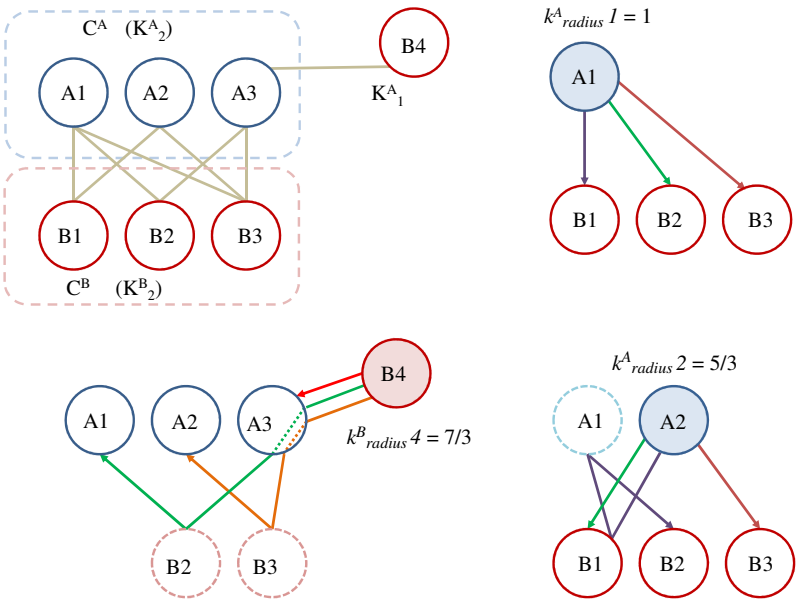


Figure 1: Examples of k_{radius} in a fictional network.

Figure 1 is a toy bipartite network, with only seven nodes. The upper left graph shows network structure. In the upper right figure, species $A1$ belongs to C^A. The distance to each node of C^B is 1, so its $k_{radius1}(1)$ is 1. In the bottom right figure, node $A2$ also belongs to C^A but there is not any direct link with $B2$, so the distance between them is 3 and $k_{radius}(2)$ is $\frac{5}{3}$. In the bottom left figure, node $B4$ is not part of C^B, and as may be expected, the value of its k_{radius} is higher.

A global value can be defined averaging this magnitude across network:

$$\overline{k}_{radius} = \frac{1}{\mid A \cup B \mid} \sum_{l \in A \cup B} k_{radius}(l) \tag{2}$$

For the network in Figure 1, the value is $11/7$.

k_{radius} is a useful measure of network compactness but it does not work as measure of centrality. For instance, its value for an isolated specialist linked to the maximum core is low. *To address this issue,* we define a second *k-magnitude*, called k_{degree}:

$$k^A_{degree}(m) = \sum_j \frac{a_{mj}}{k_{radius}j} \quad m \in A, \forall j \in B \tag{3}$$

where a_{mj} is the element of the interaction matrix that represents the link. So the $k_{degree}m$ is the sum of the inverse of k_{radius} for each node linked to m. A node of the innermost shell will have a high degree, whereas specialists have only one or two links and so a low k_{degree}. In the example of Figure 1, this magnitude is $1 + 3/5 + 3/5 = 11/5$ for node $B3$, while only $3/7$ for node $B4$.

3.1 Input file format

We use the file format of web of life ecological data collection [Bas09]. Data are stored as `.csv` files. Species of guild a are distributed by colums, and those of guild b by rows. Field separator can be a semicolon (by default), a comma or a tabulator. If the file is labelled, first colum contains the labels of guild b nodes, and first row, the labels of guild a. If the interaction matrix is binary, the cell of $species\_a\_m, species\_b\_n$ will be set to 1. If it is weighted, to a positive real number which meaning depends on the type of interaction.

The file naming convention is $M\_XX\_NNN.csv$ where XX is the type, PL for pollinator networks and SD for seed dispersers, and NNN a serial number. Anyway, you are free to call your file in the most convenient way for you.

	A	B	C	D	E
1		Juniperus phoenicea	Osyris quadripartita	Corema album	Phillyrea angustifolia
2	Turdus merula	1	1	1	1
3	Turdus iliacus	1	1	0	0
4	Turdus philomelos	1	1	0	0
5	Turdus torquatus	1	0	0	0
6	Turdus viscivorus	1	0	0	0
7					

Figure 2: Examples of input file, the seed disperser network 029 of web of life site.

3.2 Network analysis

The function `analyze_network` performs the *k-core decomposition* and analysis.

```
analyze_network(namenetwork, directory = "", guild_a = "pl",
                guild_b = "pol", plot_graphs = FALSE, only_NODF = FALSE)
```

Arguments:

- `namenetwork`: name of the interaction matrix file.
- `directory`: folder where the network file is stored.
- `guild_a`: prefix for the guild of nodes stored in rows.

- `guild_b`: prefix for the guild of nodes stored in columns.
- `plot_graphs`: plot kshell histogram and Kamada Kawai plots.
- `only_NODF`: just computes the $NODF$ measurement of nestedness.

The function returns:

- `calc_values`, a list containing the following objects:
  - `graph`: an igraph::graph object. This is the additional information for each node, besides the regular of the object:
    - `V(result_analysis$graph)`: list of node names
    - `V(result_analysis$graph)$kdegree`: list k_{degree}
    - `V(result_analysis$graph)$kradius`: list k_{radius}
    - `V(result_analysis$graph)$krisk`: list k_{riks}, a measurement of vulnerability (See [GA+17])
  - `max_core`: maximum k shell index
  - `nested_values`: a list containing the values provided by the bipartite::nested function, unless `only_NODF` set TRUE.
  - `num_guild_a`: number of nodes of guild a.
  - `num_guild_b`: number of nodes of guild b.
  - `links`: number of network links.
  - `meandist`: network average kradius.
  - `meankdegree`: network average kdegree.
  - `spaths_mat`: matrix with node to node shortest distance paths.
  - `matrix`: interaction matrix with nodes of guild a by columns and guild b by rows.
  - `g_cores`: list with the value of kshell for each node.
  - `modularity_measure`: value of igraph::modularity function.

4 The Ziggurat Plot

Ziggurat is a visualization created from scratch. Species are grouped by their *k-shells*. Each of these groups are plotted as small ziggurats. The maximum *k-shell* is located on the left side, the other ones are arranged following an almond-like distribution.

Species within the maximum shell are ordered by their k_{degree} with the highest one leftmost. Areas do not convey meaning, their height decreases just by convenience of visualization.

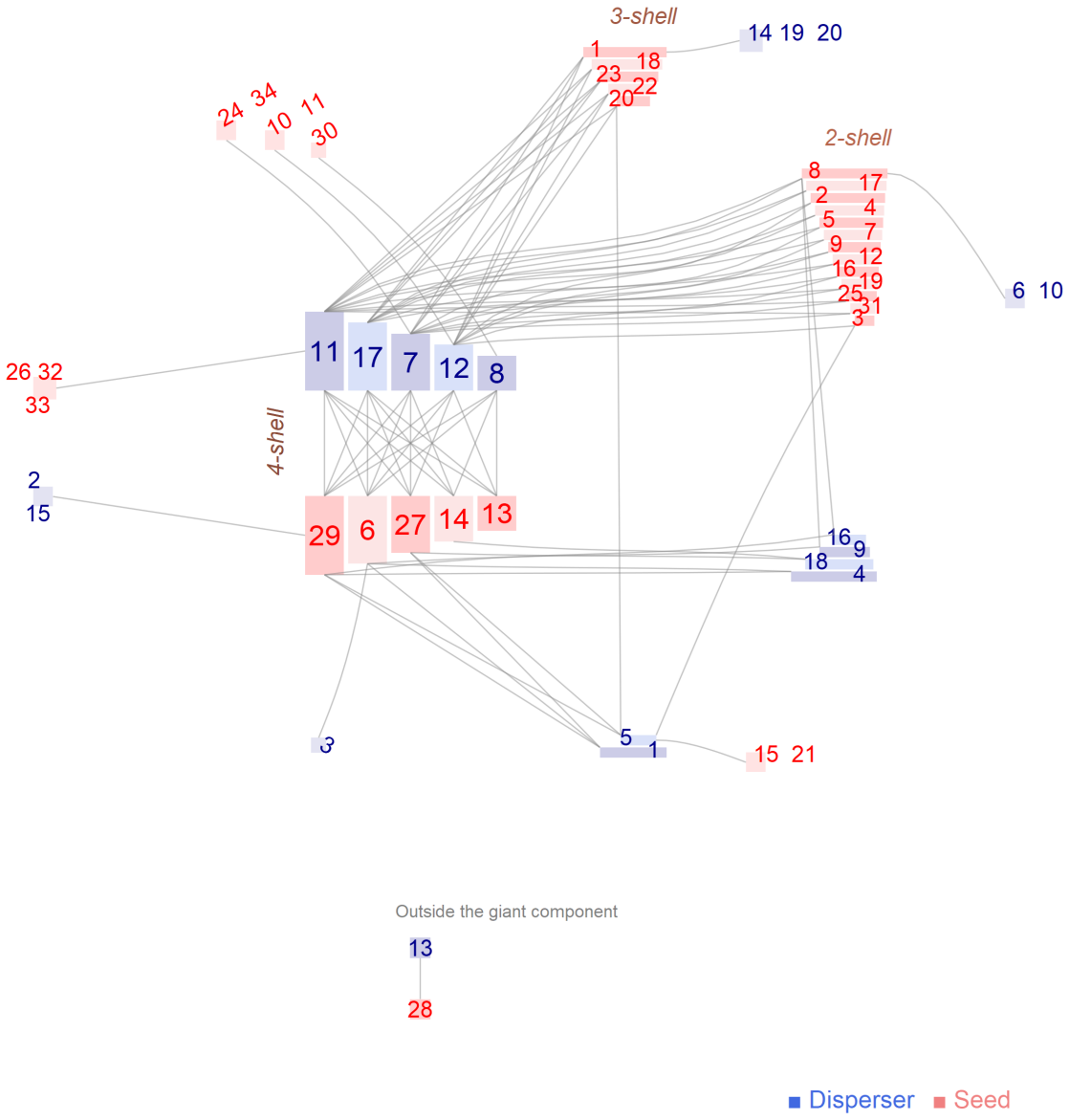


Figure 3: Ziggurat graph of an avian frugivore community in Puerto Rico with 54 species and 95 links [CCG03].

Nodes of 1-*shell* are scattered around the ziggurats. When multiple species of this shell are connected to the same node of any of the ziggurats they are clustered to reduce the number of lines. For instance, plants species $22, 30$ and 33 are specialists linked to the generalist disperser 3. With this organization, we get a clear view of structure and interconnections. The almond shape leaves a wide space in the center of the graph to depict the links and they do not overcross the boxes of the different species.
The function call is:

```
ziggurat_graph <- function(datadir,filename, style='ziggurat',sep=",",
                           speciesinheader=TRUE, paintlinks = TRUE, print_to_file = FALSE,
                           plotsdir ="plotresults/", orderkcoremaxby = "kradius",
                           flip_results = FALSE, aspect_ratio = 1,
                           alpha_level = 0.2, color_guild_a = c("#4169E1","#00008B"),
                           color_guild_b = c("#F08080","#FF0000"),
                           color_link = "slategray3", alpha_link = 0.5, size_link = 0.5,
                           displace_y_b = rep(0,20),
                           displace_y_a = rep(0,20),
                           lsize_kcoremax = 3.0, lsize_zig = 3, lsize_kcore1 = 3.0,
                           lsize_legend = 4, lsize_core_box = 3.0,
                           labels_color = c(),
                           height_box_y_expand = 1, kcore2tail_vertical_separation = 1,
                           kcore1tail_disttocore = c(1,1), innertail_vertical_separation = 1,
                           factor_hop_x = 1, fattailjumphoriz = c(1,1),
                           fattailjumpvert = c(1,1),  coremax_triangle_height_factor = 1,
                           coremax_triangle_width_factor = 1, paint_outsiders = TRUE,
                           displace_outside_component = c(0,0), outsiders_separation_expand = 1,
                           outsiders_legend_expand = 1,
                           specialistskcore2_horizontal_dist_rootleaf_expand = 1,
                           specialistskcore2_vertical_dist_rootleaf_expand = 0,
                           specialists_boxes_separation_count = 1,
                           root_specialist_expand = c(1,1), hide_plot_border = TRUE,
                           rescale_plot_area = c(1,1),
                           kcore1specialists_leafs_vertical_separation = 1,
                           corebox_border_size = 0.2, kcore_species_name_display = c(),
                           kcore_species_name_break = c(), shorten_species_name = 0,
                           exclude_species_number = FALSE, label_strguilda = "",
                           label_strguildb = "", landscape_plot = TRUE,
                           backg_color = "white", show_title = TRUE, show_legend='BOTTOM',
                           use_spline =TRUE, spline_points = 10,
                           file_name_append = "", svg_scale_factor= 10, weighted_links = "none",
                           square_nodes_size_scale = 1, progress=NULL )
```

The configuration of ziggurat plots is quite rich. There are just two mandatory parameters, `datadir` and `filename`, that provide the input directory and input file. The output file is named as the input file plus `_ziggurat.png` and the user may also add the `file_name_append` label.

Graphical parameters provide a powerful toolset to improve visualizations. Figure 3 was created with the default call:

```
ziggurat_graph("../data/","M_SD_004.csv", plotsdir = "grafresults/",
print_to_file = TRUE, label_strguilda = "Disperser", label_strguildb = "Seed")
```

The same graph looks improved tweaking some input parameters.

```
ziggurat_graph("../data/","M_SD_004.csv", height_box_y_expand = 2, label_strguilda = "Plants",
                          label_strguildb = "Dispersers", factor_hop_x=1.5, color_link = "slategray3",
                          alpha_link = 0.7, lsize_kcoremax = 6, lsize_zig = 5,lsize_kcore1 = 5,
                          corebox_border_size=0.5, kcore1tail_disttocore = c(1.2,1),
                          displace_outside_component = c(-0.5,0),
                           lsize_legend = 6,  lsize_core_box = 6, print_to_file = TRUE)
```

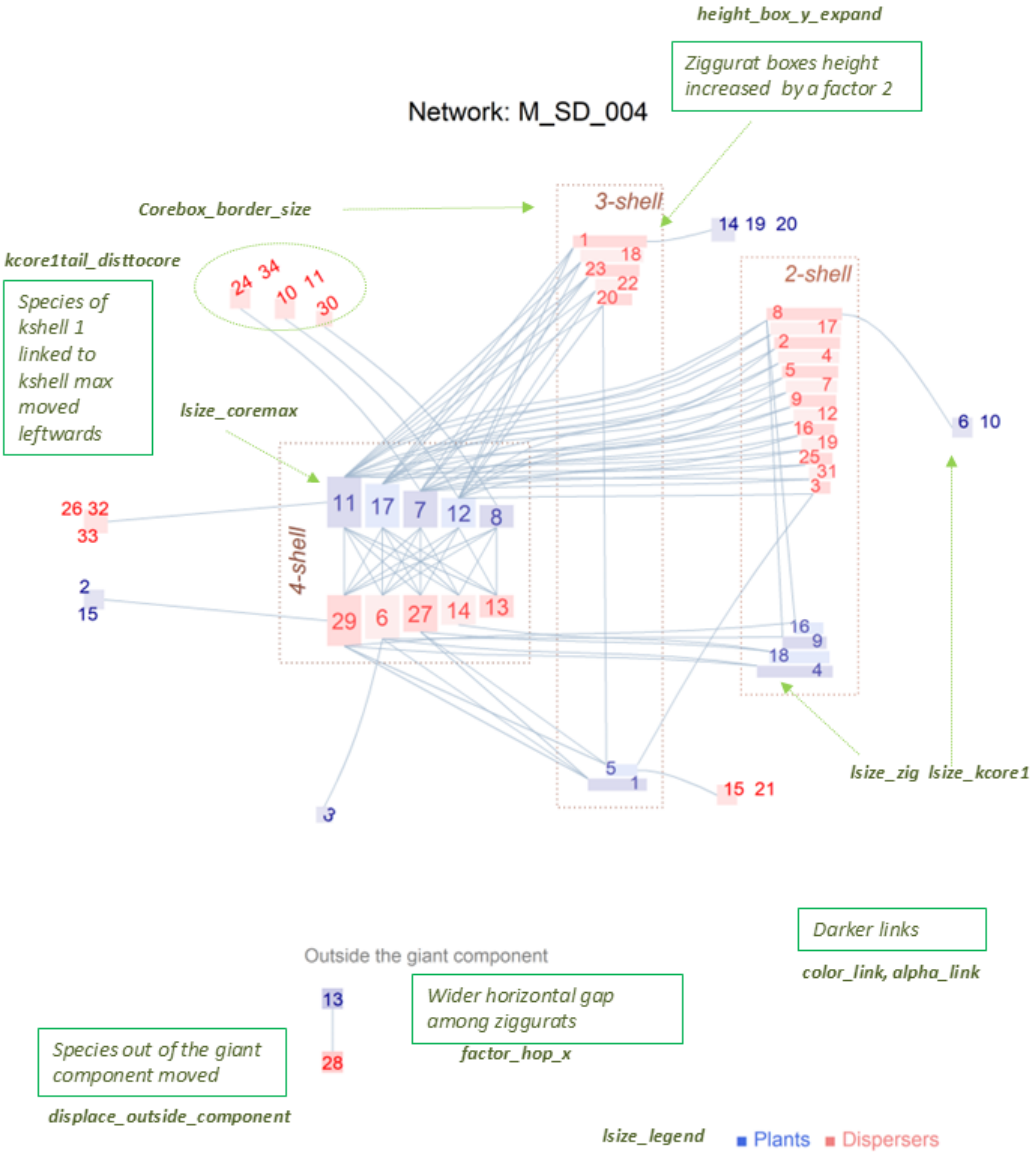


Figure 4: Improved ziggurat graph of an avian frugivore community in Puerto Rico

Plot configuration may becomeer hard when network size grows. We show now the usefulness of another set of input parameters with a mid-sized example. Default function call provides a quite readable ziggurat plot of pollinator network number 12.

```
ziggurat_graph("data/","M_PL_012.csv",  plotsdir = "grafresults/", print_to_file = TRUE)
```

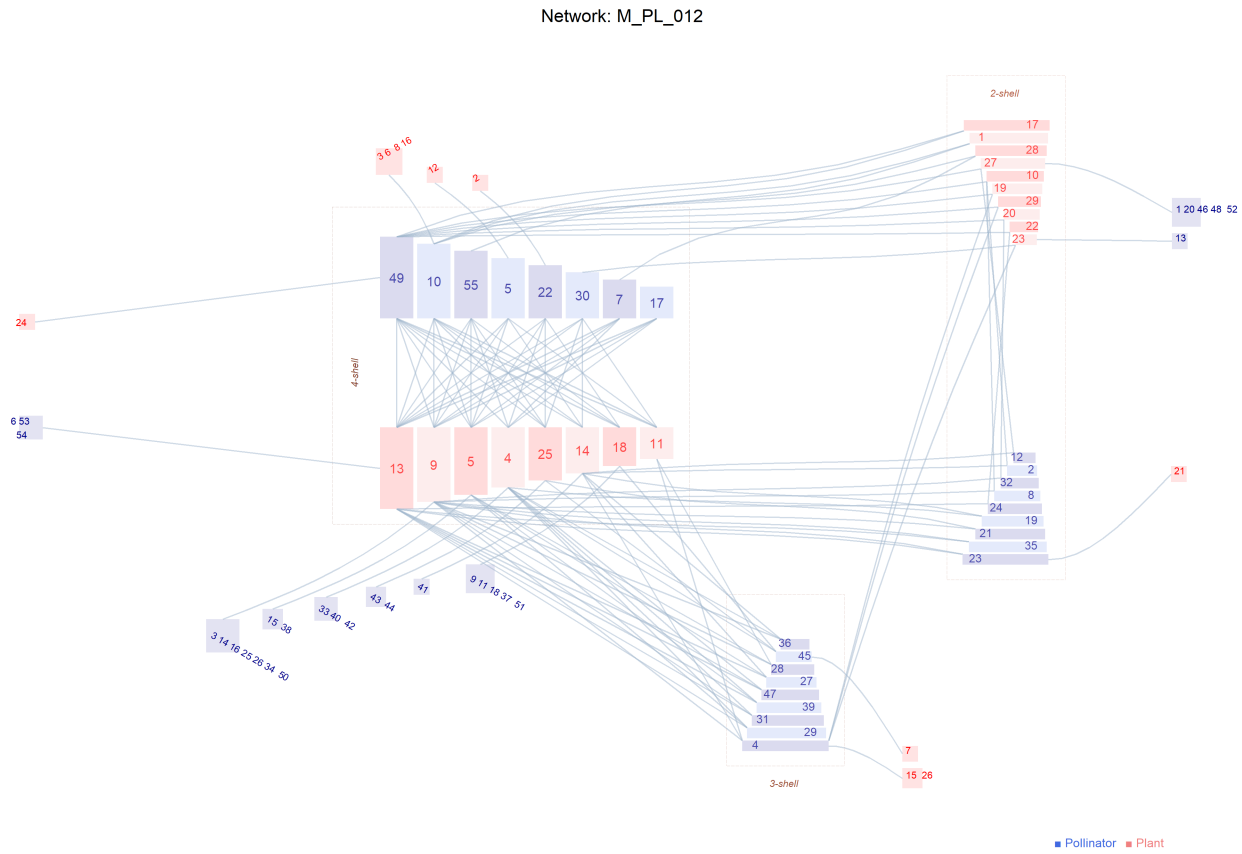
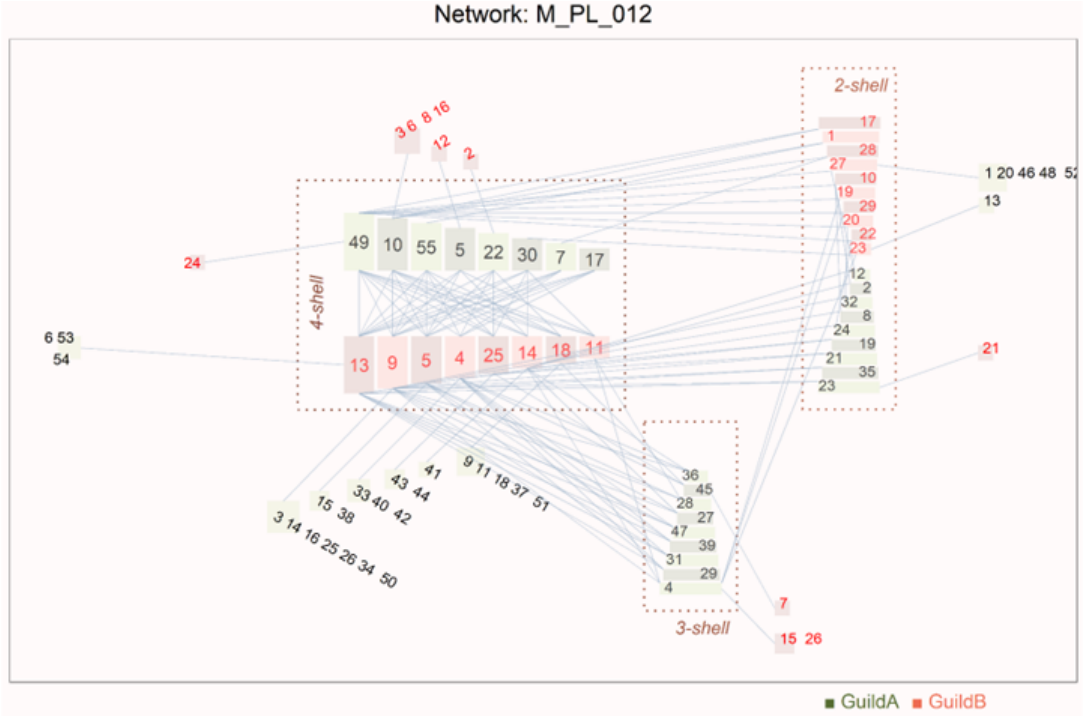


Figure 5: Ziggurat graph of a plant - pollinator network in Garajonay, La Palma (Spain). Olesen, unpublished.

This plot is nearly ready for publishing, some minor improvements would be nice, for instance increasing label sizes. However we are going to modify several input values to show how the picture changes.

```
ziggurat_graph("../data/","M_PL_012.csv", aspect_ratio = 1, height_box_y_expand = 2,
                        factor_hop_x=1.3, plotsdir="grafresults", color_link = "slategray3",
                        alpha_link = 0.5, color_guild_a=c("darkolivegreen","darkolivegreen3"),
                        color_guild_b=c("coral2","coral4"), labels_color= c("black","red"),
                        backg_color = "snow", lsize_legend = 7, lsize_core_box = 6,
                        corebox_border_size=1, innertail_vertical_separation = 2,
                        lsize_kcoremax = 6, lsize_zig = 5,lsize_kcore1 = 5,
                        displace_y_a=c(0,0.5,0,0), displace_y_b=c(0,-0.2,0.1,0),
                        rescale_plot_area=c(1.2,1), coremax_triangle_width_factor = 1.15,
                        coremax_triangle_height_factor = 1.1, fattailjumpvert = c(1.2,1.2),
                        fattailjumphoriz = c(1.25,0.8), print_to_file = TRUE,
                        hide_plot_border = FALSE, use_spline = FALSE, file_name_append = "improved")
```

We don not explain again the meaning of those that we have explained in the previous example. The role of some of the new ones arestraightforwardt. We have changed the filling colors of ziggurats,



providing the pairs `color_guild_a` and `color_guild_b`, and also the species label colors. A dime background is added as well. This trick shows the plotting area, that was horizontally increased by a 20% with `rescale_plot_area=c(1.2,1)`. This change only affects the plotting area, not the the figure. The aspect ratio may be modified to *flatten* the plot is is lesser than 1 or to *stretch* it if bigger. The default value is 1, it is not mandatory to include.

If you do not use splines, as in this example, links appear as straight lines. If you use splines you could also tell the function how many points they should have.

Fat tails are the two sets of *1-shell* species eventually linked to highest k_{degree} generalists, those that are located leftmost in the max *k-shell*.

We saw how `height_box_y_expand` controls the height of outer ziggurat boxes. Height and width of rectangles of the max *k-shell* are modified with `coremax_triangle_height_factor` and `coremax_triangle_width_factor`.

The overall horizontal distance of *1-shell* species linked to max *k-shell* (except fat tails) is increased or decreased with `horiz_kcoremax_tails_expand`.

Vertical separation of ziggurats is changed with vectors `displace_y_a` and `displace_y_b`. In this example, ziggurat of *2-shell* plants is moved upwards by a 50%, *2-shell* pollinators downwards 20% and *3-shell* pollinators upwards 10%.

The parameter `innertail_vertical_separation` resizes the vertical separation of tails connected to inner ziggurats. See plant species 22 and 25.

Next example shows how to manage *specialist tails* and *outsiders*. specialist tails are chains of species of *1-shell*. They are rather unstable so they do not appear frequently. Outsiders are the species not connected to the giant component. Pollinator network 031 has species of both kinds.

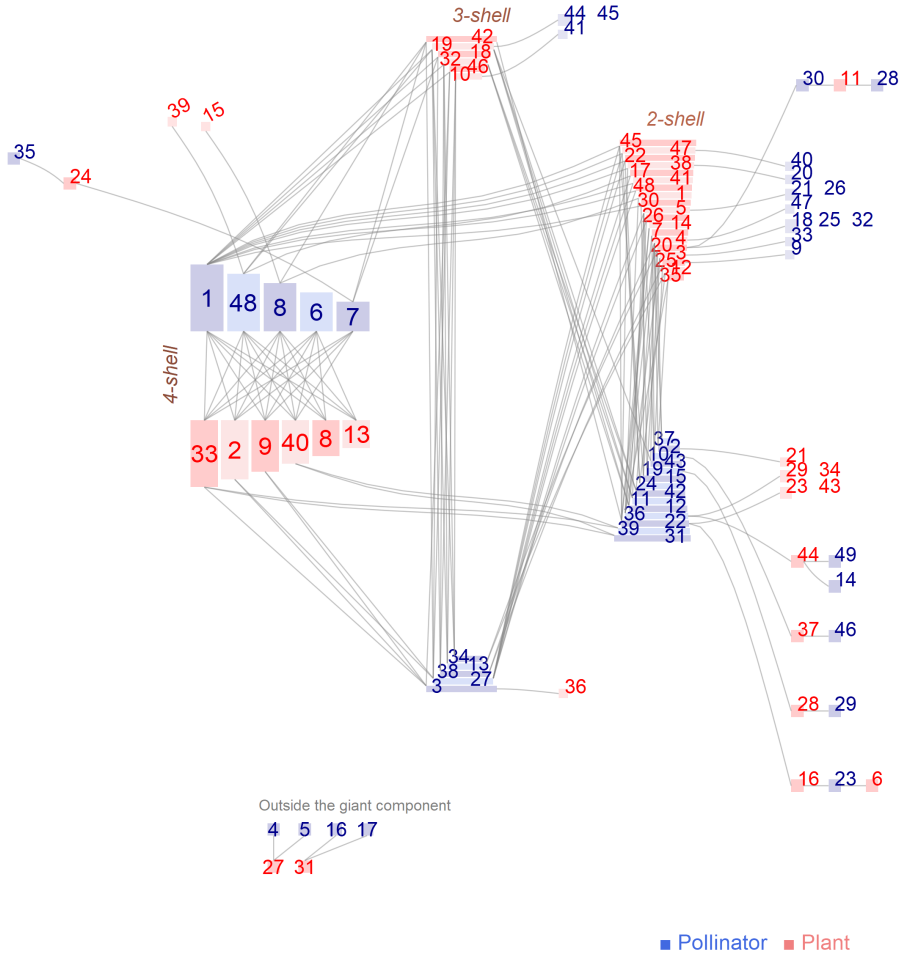


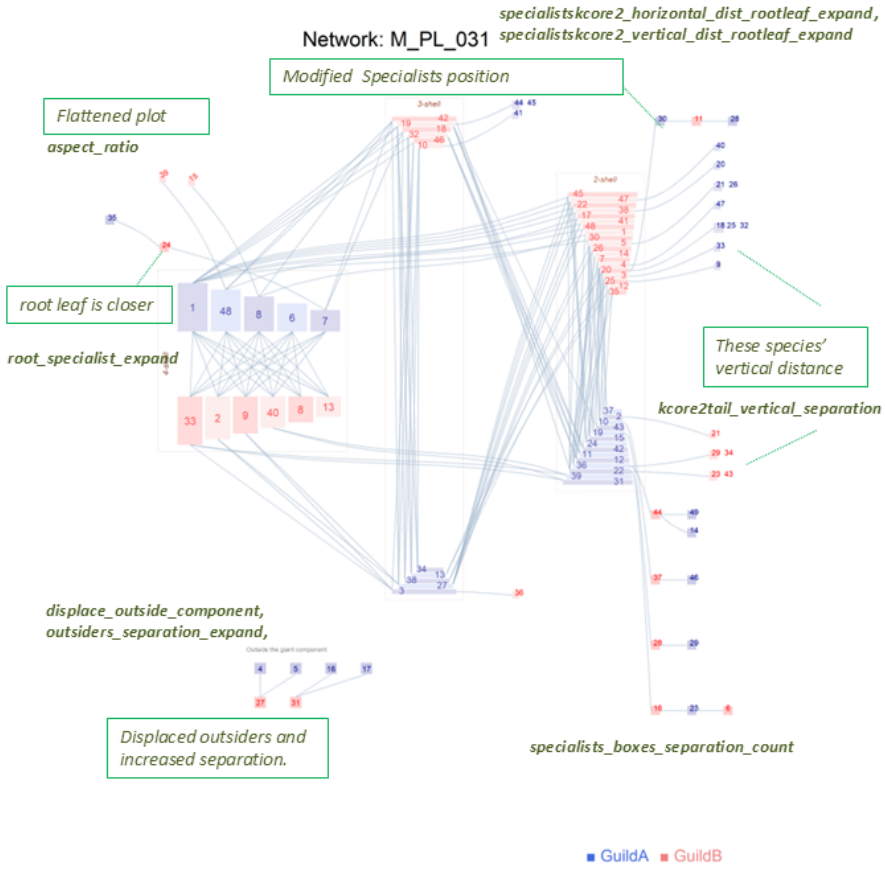
Figure 6: Default ziggurat graph of a plant - pollinator network in Alta Guyana (Venezuela) [Ram89].

Outsiders appear under the main plot. This network has a rich set of specialist chains, linked both to ziggurats of *2-shell* and to the max *shell*. If plant species 28 becomes extinct, it will also drag pollinator 21 and plant 40. This is an uncommon chain of specialists linked among them and very exposed to external perturbations.

The `ziggurat_grap` function offers input fields to manage the appearance and position of these species.

```
ziggurat_graph("../data/","M_PL_031.csv",  plotsdir = "grafresults", print_to_file = TRUE,
                          displace_outside_component = c(0,0.25), kcore2tail_vertical_separation = 3,
                          specialists_boxes_separation_count=3, root_specialist_expand = c(0.5,1),
                          specialistskcore2_horizontal_dist_rootleaf_expand = -0.1,
                          specialistskcore2_vertical_dist_rootleaf_expand = 1.5,
                          outsiders_separation_expand = 1.25, lsize_legend = 6,
                          aspect_ratio= 0.8, file_name_append = "improved")
```

It is possible to display the names of the species inside the ziggurat rectangles. Be careful, because they may make hard to understand the structure, we do not encourage using this feature unless the network is tiny. Species names of *1-shell* cannot be displayed.


specialistskcore2_horizontal_dist_rootleaf_expand,
specialistskcore2_vertical_dist_rootleaf_expand
Network: M_PL_031
Modified Specialists position
Flattened plot
aspect_ratio
root leaf is closer
root_specialist_expand
These species' vertical distance
kcore2tail_vertical_separation
displace_outside_component,
outsiders_separation_expand,
Displaced outsiders and increased separation.
specialists_boxes_separation_count
GuildA
GuildB

```
ziggurat_graph("../data/","M_SD_025.csv",plotsdir="grafresults/", shorten_species_name = 4,
                          aspect_ratio = 0.6,show_legend="TOP", height_box_y_expand = 2,
                          coremax_triangle_width_factor = 1.25,coremax_triangle_height_factor = 2.25,
                          lsize_core_box = 4, lsize_kcoremax = 5, lsize_legend= 7,lsize_kcore1 = 5,
                          lsize_zig = 5, kcore_species_name_display = c(2,3),
                          kcore_species_name_break = c(3), print_to_file = TRUE)
```

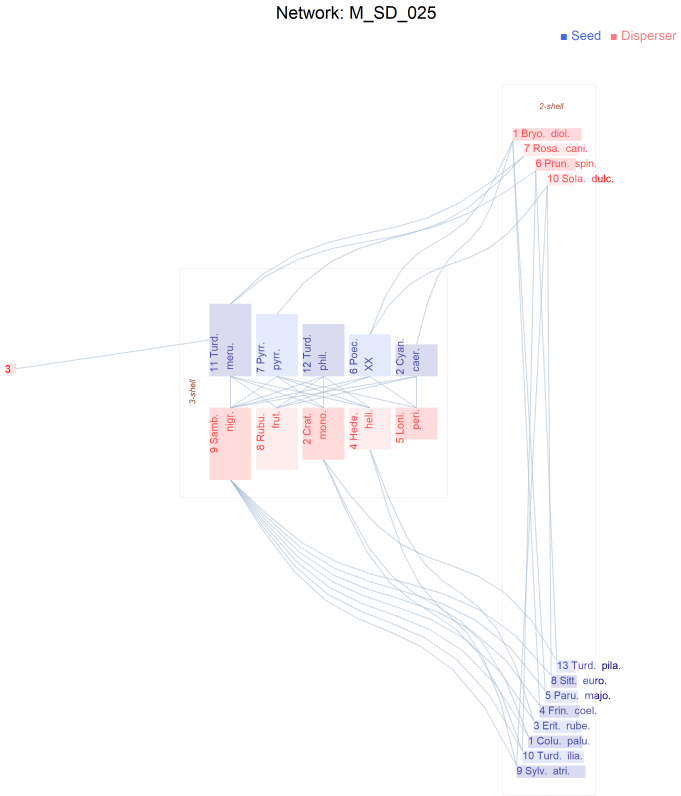


Figure 7: Ziggurat graph of a plant - disperser.

If the network is weighted, links width may appear as a function of the interaction strength, as in figure 8. There are three options: square root, natural logarithm or decimal logarithm of the weight.

```
ziggurat_graph("../data/","RA_HP_001.csv" ,weighted_links = "ln",   plotsdir = "grafresults",
                          label_strguilda = "Parasite" ,  label_strguildb = "Host" ,lsize_kcoremax = 5 ,
                          lsize_zig = 4.5 ,lsize_kcore1 = 4 ,lsize_legend = 6, lsize_core_box = 5,
                           print_to_file = TRUE )
```

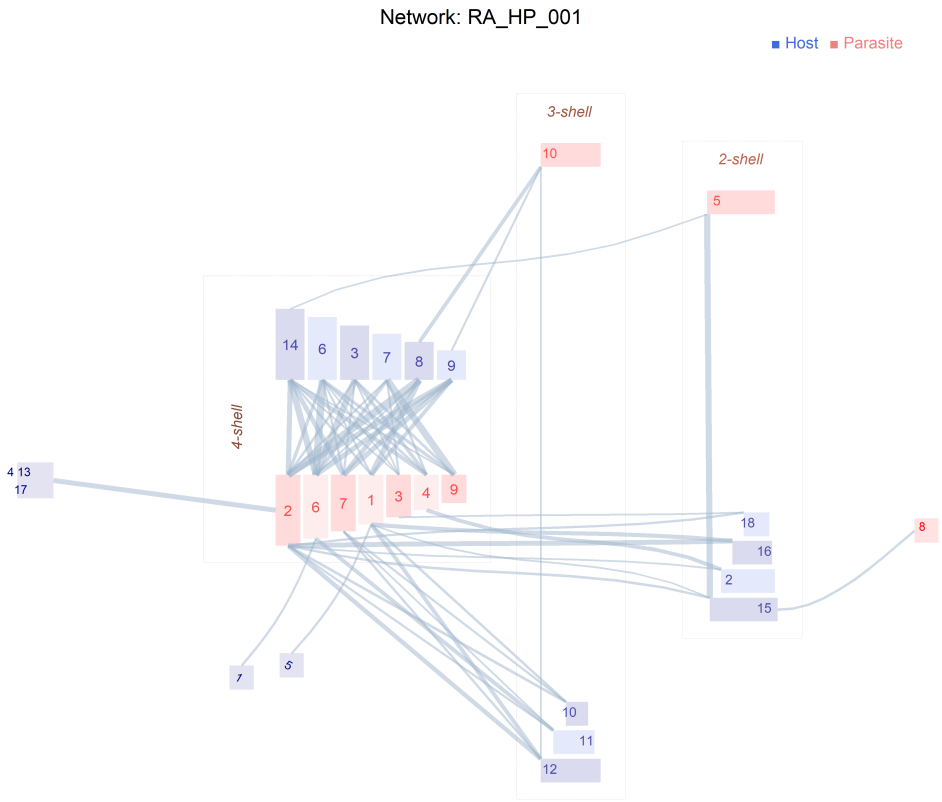


Figure 8: Ziggurat graph of a host - parasite network.

The function returns its own environment called `zgg` where configuration paramenters and results are stored. If you are developing an application, you can retrieve:

- `zgg$plot`: the ziggurat plot.
- `zgg$svg`: the ziggurat plot as an SVG object.
- `zgg$results_analysis`: the internal `analyze_network` call results.

The default `ziggurat_graph` prints the plot in `png` format with a resolution of 300*dpi*, but you can save `zgg$plot` outside the function in any other format and resolution.
Finally, this is the meaning of all the input parameters:

- `datadir`: name of the file of the interaction matrix.
- `filename`: file with the interaction matrix.
- `paintlinks`: if set to FALSE links will be hidden. It is useful to test the plot appearance when the network is huge.
- `print_to_file`: if set to FALSE the plot is displayed in the R session window.
- `plotsdir`: the directory where the plot is stored.
- `flip_results`: displays the graph in portrait configuration.
- `aspect_ratio`: ziggurat plot default aspect ratio.
- `alpha_level`: transparency for ziggurats' filling.
- `color_guild_a`: default filling for nodes of guild_a.
- `color_guild_b`: default filling for nodes of guild_b.
- `color_link default`: link color.
- `alpha_link`: link transparency.

- `size_link`: width of links.
- `displace_y_b`: relative vertical displacement of guild_b inner ziggurats.
- `displace_y_a`: relative vertical displacement of guild_a inner ziggurats.
- `labels_size`: default node label size.
- `lsize_kcoremax`: nodes in *max-kshell* label size.
- `lsize_zig nodes`: in inner ziggurats label size.
- `lsize_kcore1`: labels of nodes in 1-*shell*.
- `lsize_legend`: legend label size.
- `lsize_kcorebox`: default *kshell* boxes label size.
- `labels_color`: default label colors.
- `height_box_y_expand`: expand inner ziggurat rectangles default height by this factor.
- `kcore2tail_vertical_separation`: expand vertical of 1-*shell* species linked to 2-*shell* by this factor.
- `kcore1tail_disttocore`: expand vertical separation of 1-*shell* species from *max-shell* (guild_a, guild,b).
- `innertail_vertical_separation`: expand vertical separation of *kshell* species connected to 2-*shell* $<$ *kshell* $<$ *max-shell*.
- `horiz_kcoremax_tails_expand`: expands horizontal separation of specialist tails connected to *max-shell*.
- `factor_hop_x expand inner`: expands ziggurats horizontal distance.
- `fattailjumphoriz`: displace species linked to leftmost *max-shell* species.
- `fattailjumpvert`: idem for vertical position.
- `coremax_triangle_width_factor`: expand khsell max rectangles width by this factor.
- `coremax_triangle_height_factor`: expand khsell max rectangles height by this factor.
- `paint_outsiders`: paint species disconnected of the giant component.
- `displace_outside_component`: displace outsider species (horizontal, vertical).
- `outsiders_separation_expand`: multiply by this factor outsiders' separation.
- `outsiders_legend_expand`: displace outsiders legend.
- `specialistskcore2_horizontal_dist_rootleaf_expand`: expand horizontal distance of specialist tail root node connected to 2-*shell*.
- `specialistskcore2_vertical_dist_rootleaf_expand`: expand vertical distance of specialist tails connected to 2-*shell*.
- `specialists_boxes_separation_count`: specialist species boxes separation count.
- `root_specialist_expand`: expand root specialist distances of tails connected to *kshell* $\neq 2$.
- `hide_plot_border`: hide border around the plot.
- `rescale_plot_area`: full plot area rescaling (horizontal, vertical).
- `kcore1specialists_leafs_vertical_separation`: expand vertical separation of specialist tails connected to 1-*shell* species.
- `corebox_border_size`: width of *kshell* boxes.
- `kcore_species_name_display`: display species names of shells listed in this vector.
- `kcore_species_name_break`: allow new lines in species names of shells listed in this vector.
- `shorten_species_names`: number of characters of species name to display.
- `label_strguilda`: string labels of guild a.
- `label_strguildb`: string labels of guild b.
- `landscape_plot`: paper landscape configuration.
- `backg_color`: plot background color.

- `show_title`: show plot title.
- `use_spline`: use splines to draw links.
- `spline_points`: number of points for each spline.
- `file_name_append`: a label that the user may append to the plot file name for convenience.
- `svg_scale_factor`: only for interactive apps, do not modify.
- `weighted_links`: weight function to display links, (`"none"`,`"log10"`,`"ln"`,`"sqrt"`)
- `square_nodes_size_scale`: scale the area of nodes in 1–*shell* or outsiders.
- `progress`: only for interactive apps, do not modifiy.

5 Bipartite Plot

The *bipartite* plot is the conventional way to depict a bipartite network. Nodes of both guilds are aligned in two parallel lines (horizontal or vertical is a matter of convenience), orderded by degree. A straight line between two nodes of different guilds means that there is a link and its width may be porportional to the weight of the interaction. The bipartite plot becomes a *hair ball* as the number nodes and links increases. There are some minor mitigation strategies to overcome this problem. This package offers the *legacy* style for bipartite plot, following the traditional layout, but distributing evenly the nodes of the smallest guild. This solution makes the diagram more readable.

```
bipartite_graph("../data/","M_SD_004.csv",style = "legacy", print_to_file = TRUE)
```

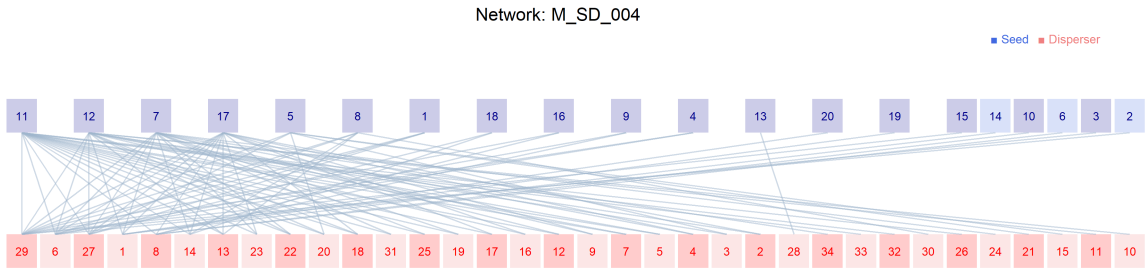


Figure 9: Legacy bipartite graph of an avian frugivore community in Puerto Rico with 54 species and 95 links [CCG03].

There are some details that we can deduce at a glance. The more generalist species are those that lay on the left side of the plot and the pair disperser 13/plant 28 is disconnected of the giant component of the network. We can get further visual information tunning input parameters.

```
bipartite_graph("../data/","M_SD_004.csv" ,paintlinks = TRUE ,print_to_file = TRUE,
                      plotsdir = "plot_results/" ,style = "legacy",guild_gap_increase = 3,
                      lsize_kcoremax = 4.2 ,lsize_legend = 6 ,
                      label_strguilda = "Disperser" ,label_strguildb = "Seed",
                      show_legend = "TOP" ,weighted_links = "sqrt")
```

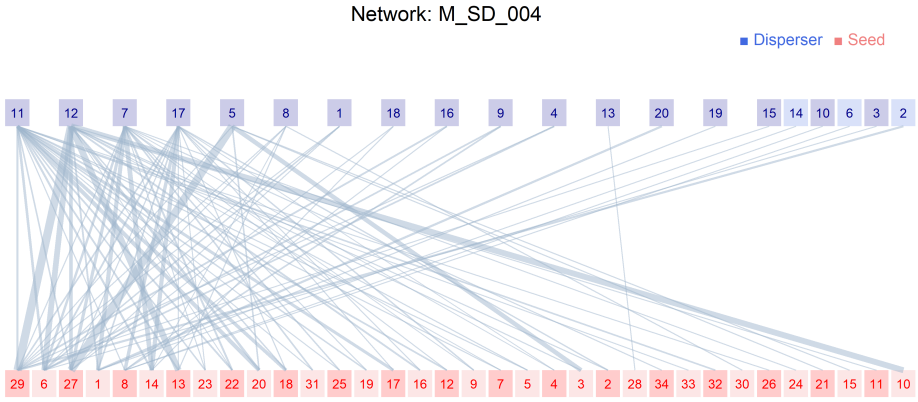


Figure 10: Improved bipartite graph.

The spatial gap between guild lines is now wider. Lilnks width is proportional to the weight and labels size is larger. But the information is harder to discover in this plot than in its ziggurat counterpart (fig. 4). The difference between both kind of graphs is that bipartite is a linear plot, while ziggurat is bidimensional. In addition, nodes in ziggurat are clustered by its `k-shell` number.

This package provides a new style of bipartite plot, named *kcoreorder*, where nodes are ordered and grouped following this criterion. Moreover, nodes of `1-shell` are ploted on the left side to shorten its links length towards the max shell of the network.

```
bipartite_graph("../data/","M_SD_004.csv" ,paintlinks = TRUE ,print_to_file = TRUE,
                        plotsdir = "plot_results/" ,style = "kcoreorder",guild_gap_increase = 2,
                        lsize_kcoremax = 4.2 ,lsize_kcore1 = 3.0 ,lsize_legend = 4 ,
                        label_strguilda = "Disperser" ,label_strguildb = "Seed",
                        show_legend = "TOP")
```

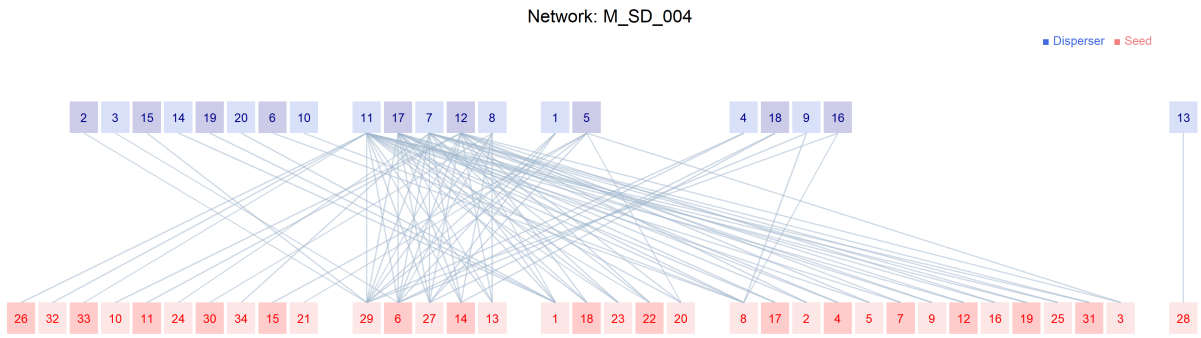


Figure 11: kcoreorder bipartite graph.

Now, it is easy to spot the different `k-shells`, with species of `4-shell` being the most interconnected. On their left, we find `1-shell`, on their right, `3-shell`, `2-shell` and finally, species disconnected of the giant component. The only difference between the two styles is the order of the species, but information is still concentrated in two lines of nodes.

What if we could keep the whole idea of bipartite plot but spreading part of the visual information in two dimensions? That is the idea behind the third style, the *chillopod* plot. The name refers to the resemblance of this plot with a centipede. Nodes of `1-shell` that we call *tails* in the ziggurat plot are mived away from the guild lines, and appear as short appendices. We apply the same grouping idea that in ziggurat's fat tails, nodes that has just one link and are tied to the same species of a higher shell, are plotted together.

```
bipartite_graph("../data/","M_SD_004.csv" ,paintlinks = TRUE ,print_to_file = TRUE,
                        plotsdir = "plot_results/" ,style = "chilopod",lsize_kcoremax = 6,
                        lsize_legend = 6 ,label_strguilda = "Disperser" ,label_strguildb = "Seed",
                        show_legend = "TOP")
```

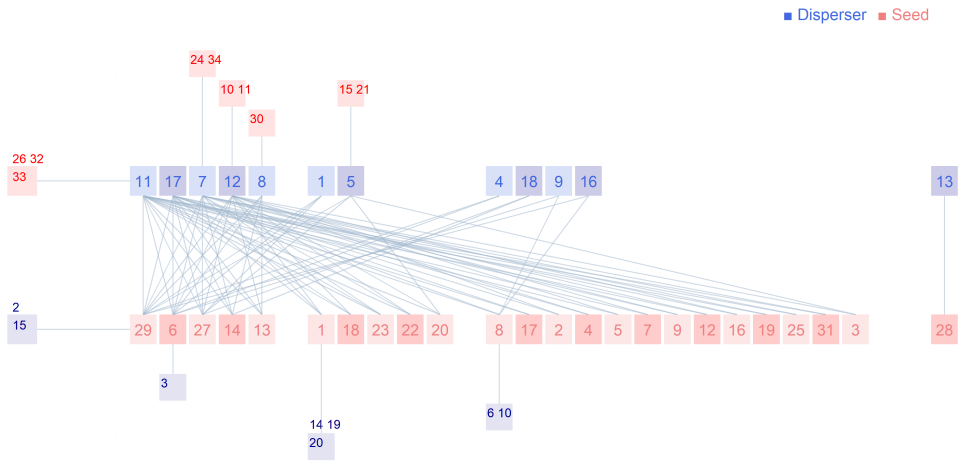


Figure 12: Chilopod bipartite graph.

The default function call is:

```
bipartite_graph <- function(datadir,filename,sep=",",speciesinheader=TRUE,
                            paintlinks = TRUE, print_to_file = FALSE, plotsdir ="plot_results/",
                            orderkcoremaxby = "kradius", style="legacy", guild_gap_increase = 1,
```

```
flip_results = FALSE,  alpha_level = 0.2,
color_guild_a = c("#4169E1","#00008B"),
color_guild_b = c("#F08080","#FF0000"),
color_link = "slategray3", alpha_link = 0.5, size_link = 0.5,
lsize_kcoremax = 3, lsize_legend = 4, lsize_core_box = 3.0,
labels_color = c(),
hide_plot_border = TRUE,
label_strguilda = "",
label_strguildb = "", landscape_plot = TRUE,
backg_color = "white", show_title = TRUE, show_legend = 'TOP',
file_name_append = "", svg_scale_factor= 10, weighted_links = "none",
progress=NULL )
```

This is the meaning of the input parameters:

- `datadir`: name of the file of the interaction matrix.
- `filename`: file with the interaction matrix.
- `sep`: field separator character.
- `speciesinheader`: TRUE if species names are stored as row and column names inside the network data file.
- `print_to_file`: if set to FALSE the plot is displayed in the R session window.
- `plotsdir`: the directory where the plot is stored.
- `orderkcoremaxby`: sets the order of kcoremax to kradius or kdegree
- `style`: bipartite representation style: legacy, kcoreorder, chilopod
- `paintlinks`: if set to FALSE links will be hidden. It is useful to test the plot appearance when the network is huge.
- `flip_results`: displays the graph in portrait configuration.
- `alpha_level`: transparency for ziggurats' filling.
- `color_guild_a`: default filling for nodes of guild_a.
- `color_guild_b`: default filling for nodes of guild_b.
- `color_link default`: link color.
- `alpha_link`: link transparency.
- `size_link`: width of links.
- `lsize_kcoremax`: nodes in main lines label size.
- `lsize_legend`: legend label size.
- `labels_color`: default label colors.
- `hide_plot_border`: hide border around the plot.
- `label_strguilda`: string labels of guild a.
- `label_strguildb`: string labels of guild b.
- `landscape_plot`: paper landscape configuration
- `backg_color`: plot background color.
- `show_title`: show plot title.
- `show_legend`: show plot legend position: "TOP","BOTTOM","NONE".
- `file_name_append`: a label that the user may append to the plot file name for convenience.
- `svg_scale_factor`: only for interactive apps, do not modify.
- `weighted_links`: weight function to display links, (`"none"`,`"log10"`,`"ln"`,`"sqrt"`)
- `progress`: only for interactive apps, do not modify.

The function returns its own environment called `bpp` where configuration paramenters and results are stored. If you are developing an application, you can retrieve among other values:

- `bpp$plot`: the bipartite plot.
- `bpp$svg`: the bipartite plot as an SVG object.
- `bpp$bipartite_argg`: function call parameter values.
- `bpp$g`: the internal graph object.

6 Matrix Plot

The *Matrix Plot* is another traditional way of plotting bipartite networks. It is just the depiction of the interaction matrix, with species of guild A as columns and species of guild B as rows. For binary matrixes, the cell i, j is filled with solid color if there is a link between species i of guild A and species j of guild B. For weighted networks, the interaction strength is usually encoded as a color gradient. Nodes are sorted by their degree in descending order. This kind of plot is simple, and is useful to spot spatial properties as nestedness, but it is pretty hard to discover chains of specialists.

```
matrix_graph("../data/","M_SD_004.csv",orderby = "degree",  label_strguilda = "Disperser" ,
                    label_strguildb = "Seed", print_to_file = TRUE, flip_matrix = TRUE)
```

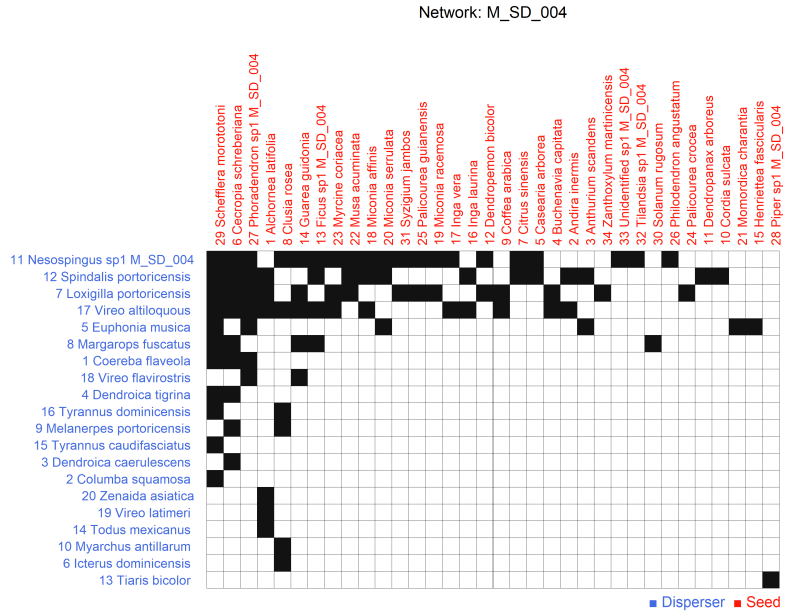


Figure 13: Matrix graph of an avian frugivore community SD_004

The function allows to flip the matrix to show it in landscape layout for convenience, as in this example. Besides the traditional order by degree, *kcorebip* offers order by kradius and kdegree. Species names can be removed, and this feature is of special interest when the network is big.

```
matrix_graph("../data/","M_SD_004.csv",orderby = "kradius",show_species_names = FALSE,
                    label_strguilda = "Disperser" ,label_strguildb = "Seed", print_to_file = TRUE,
                  links_weight = TRUE)
```

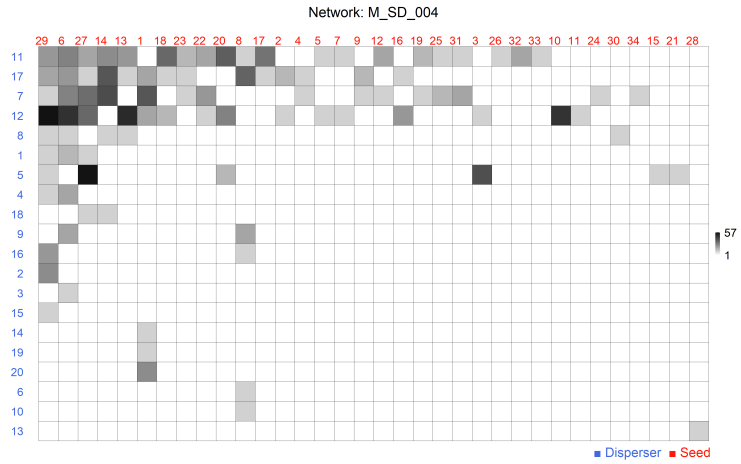


Figure 14: Matrix graph of the same avian frugivore community, with nodes sorted by kradius

The default function call is:

```
matrix_graph <-function(datadir,filename,sep=",",speciesinheader=TRUE,
                        style="matrix", orderby = "kradius",
                        label_strguilda = "Plant",
                        label_strguildb = "Pollinator",
                        label_size = 18,
                        color_guild_a = "#4169E1",
                        color_guild_b = "#FF1808",
                        color_links = 'grey7',  flip_matrix = FALSE,
                        links_weight = FALSE,  show_species_names = TRUE,
                        show_title = TRUE, show_legend = TRUE,
                        print_to_file = FALSE, plotsdir ="plot_results/",
                        plot_size = 10,  ppi = 300,
                        progress = NULL
                        )
```

This is the meaning of the input parameters:

- `datadir`: name of the file of the interaction matrix.
- `filename`: file with the interaction matrix.
- `sep`: field separator character.
- `speciesinheader`: TRUE if species names are stored as row and column names inside the network data file.
- `style`: "matrix", do not change.
  tem `orderby`: sort species of the same guild by degree, kradius or kdegree in descending order.
- `label_strguilda`: string labels of guild a.
- `label_strguildb`: string labels of guild b.
- `label_size`: species label size.
- `color_guild_a`: default filling for nodes of guild_a.
- `color_guild_b`: default filling for nodes of guild_b.
- `flip_matrix`: transpose interaction matrix.
- `color_linkst`: link color.
- `links_weight`: for weighted networks, fill interactions with gradient color
- `show_species_name`: include species names in node labels
- `show_title`: show plot title.
- `show_legend`: show plot legend position: "TOP","BOTTOM","NONE".
- `print_to_file`: if set to FALSE the plot is displayed in the R session window.
- `plotsdir`: the directory where the plot is stored.
- `plot_size`: size of printed plot in inches..
- `ppi`: printer dots per inch.
- `progress`: only for interactive apps, do not modify.

The function returns its own environment called `mat` where configuration paramenters and results are stored. If you are developing an application, you can retrieve among other values:

- `mat$plot`: the matrix plot.
- `mat$mat_argg`: a list with the result of the network analysis. See `analyze_network` for details.
- `mat$result_analysis`: the internal graph object.

7 The Polar Plot

The *Polar Plot* shows nodes (species) centrality and provides an overview of network distribution. It was inspired by the *fingerprint-like* graph, developed by Alvarez-Hamelin *et al.* [AH+05] to plot very large *k-decomposed* networks.

Nodes are depicted with their centers located at k_{radius}. Angles are assigned at random by the visualization algorithm and each guild lies inside one of the half planes. The size of each node is proportional to its k_{degree} and the color represents the k_{shell}. This visualization does not include links. The user may choose adding the histograms of *k-magnitudes*, a handy option because they convey a wealth of structural information.
The function call is:

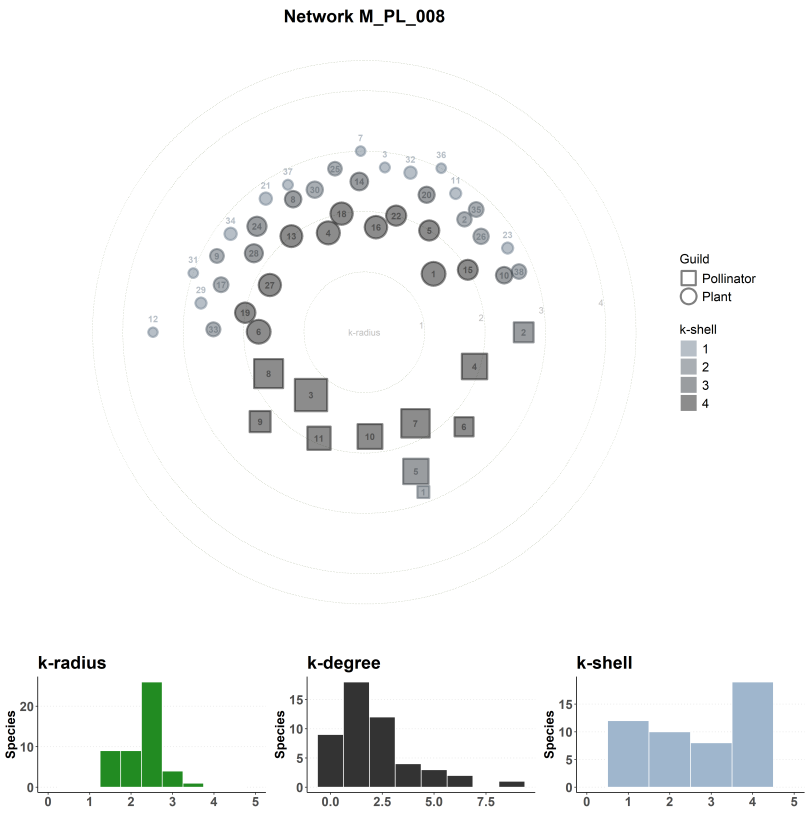
```
polar_graph <- function( red, directorystr = "data/",  plotsdir = "plot_results/polar/",
                         print_to_file = FALSE,  pshowtext = FALSE, show_histograms = TRUE,
                         glabels = c("Plant", "Pollinator"), gshortened = c("pl","pol"),
                         lsize_title = 22, lsize_axis = 16,  lsize_legend = 16, lsize_axis_title = 16,
                         lsize_legend_title = 16, file_name_append = "", print_title = TRUE,
                         progress=NULL, printable_labels = 0, fill_nodes = TRUE,
                         alpha_nodes = 0.5, max_kradius = 0)
```

The parameters `red`, `directorystr` and `plotsdir` are the name of the interaction matrix, the directory where it is stored and the name of the plotting directory. If `print_to_file` is set to `FALSE` the polar plot will be displayed in the current R session. The name of the output file is that of the input file plus `_polar.png`. Paths are relative to the current working path in `R` but you may also specify absolute paths.

File plots have a resolution of 600 dots per inch, and a size of 12 x 12 inches. If the user wants to display the result in the current R session, label sizes may appear different depending on the installed fonts and size of the plotting window.

The following command creates the file `M_PL_008_polar.png` in the directory `graphresults/`

```
pgr <- polar_graph("../data/","M_PL_008.csv",plotsdir="grafresults/", print_to_file = TRUE)
```

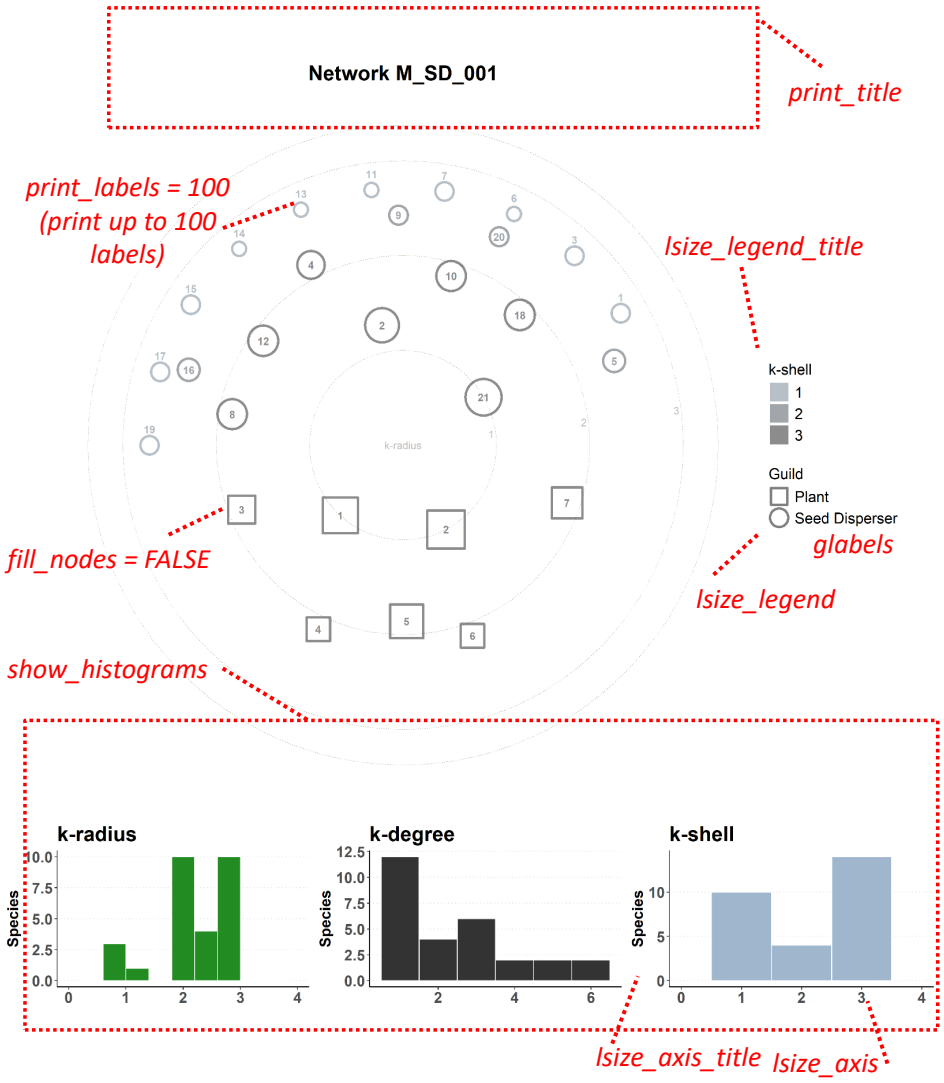


By default, nodes are unlabeled. You may choose to print some of them with the parameter `printable_labels`, but have in mind that the diagram may become messy. Guild labels are also

configurable. The function will set automatically "Plant, Pollinator" and "Plant, Disperser" if the file follows the naming convention of the web of life site; you may choose any other pair of values with the input parameter `glabels`.

Three histrogams are displayed under the main plot, with the distributions of k_{radius}, k_{degree} and k_{shell}.

The configuration of a polar graph is quite simple. The following picture shows how and where the different input parameters modify the plot.




```
polar_graph( "../data/", "M_SD_001.csv",plotsdir="plot_results/",
          print_title = TRUE, print_to_file = TRUE, lsize_axis = 16,
          lsize_legend = 16,lsize_axis_title = 16, lsize_legend_title = 16,
          glabels = c("Seed Disperser", "Plant"), gshortened = c("PL","SD"),
          fill_nodes = FALSE, printable_labels = 100,
          file_name_append = "showpars")
```

References

[AH+05] José Ignacio Alvarez-Hamelin et al. "k-core decomposition: A tool for the visualization of large scale networks". In: *arXiv preprint cs/0504107* (2005).

[Bas09] Jordi Bascompte. "Disentangling the web of life". In: *Science* 325 (2009), pp. 416–419.

[CCG03] Tomás A Carlo, Jaime A Collazo, and Martha J Groom. "Avian fruit preferences across a Puerto Rican forested landscape: pattern consistency and implications for seed removal". In: *Oecologia* 134.1 (2003), pp. 119–131.

[GA+17] Javier Garcia-Algarra et al. "Ranking of critical species to preserve the functionality of mutualistic networks using the k-core decomposition". In: *PeerJ* 5 (2017), e3321. DOI: 10.7717/peerj.3321. URL: http://dx.doi.org/10.7717/peerj.3321.

[GA+18] Javier Garcia-Algarra et al. "A structural approach to disentangle the visualization of bipartite biological networks". In: *Complexity* 2018 (2018), pp. 1–11.

[Ram89] Nelson Ramirez. "Biología de polinización en una comunidad arbustiva tropical de la alta Guayana venezolana". In: *Biotropica* (1989), pp. 319–330.

[Sei83] Stephen B Seidman. "Network structure and minimum degree". In: *Social networks* 5.3 (1983), pp. 269–287.