Using the kcorebip package

Version 0.9.1

Javier Garcia-Algarra
jgalgarra@gmail.com

October 9, 2023

Introduction

This package performs the *k-core decomposition* analysis of a bipartite graph and provides two new kinds of plot: *polar* and *ziggurat*. It works for any kind of bipartite network, but we developed it to study ecological mutualistic communities, so we use terminology and examples of that research field [GA+18; GA+17].

Install kcorebip

To install `kcorebip` you need the package `devtools`. Load it with

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

and then issue the following command:

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

Warning. You need a `gcc` compiler in your machine. If you are a developer and you know yo 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/

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 $k = 1$ until all the resting nodes have two or more links. Then it continues 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 calculate the average of the shortest paths to all the nodes in 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 the set C^B. The same definition is valid for species of the guild B computing the average distance to C^A. The minimum possible radius value is 1 for one node of the maximum shell directly linked to each one of the maximum shell set of the opposite guild.

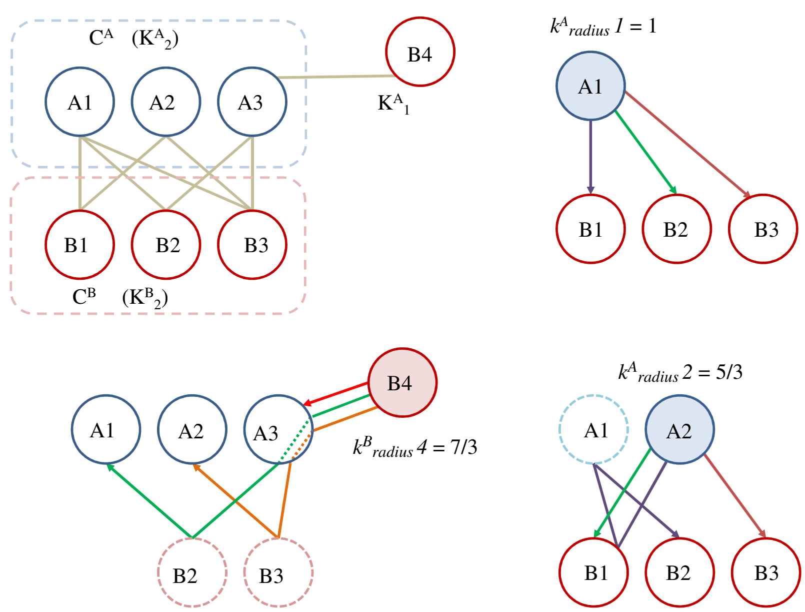


Figure 1: Examples of k_{radius} in a fictional network.

Figure 1 is a very simple fictional 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 of the nodes 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:

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

In our example network of Figure 1, the value is 11/7.

k_{radius} is a useful magnitude to measure network compactness but it not a good measure of centrality. For instance, its value for an isolated specialist linked to the maximum core is low. *To attend this necessity,* we define a second *k-magnitude*, the 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 $B4$.

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. 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 real number different of 0.

The 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 can call your file whatever you want.

Network analysis

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

	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.

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

Arguments:

- `namenetwork`: name of the file of the interaction matrix.
- `directory`: 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 all 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.

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)
```

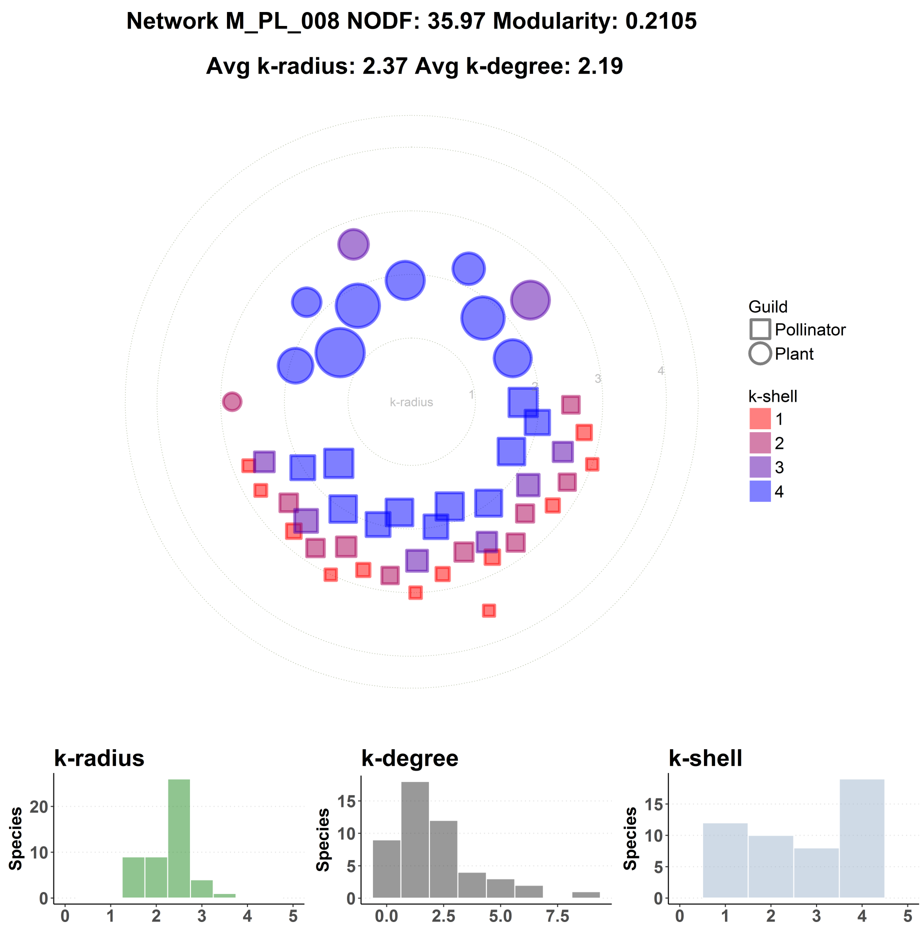
The parameters `red`, `directorystr` and `plotsdir` are the name of the interaction matrix, the directory where it is stored and the name of the plotting directory. If `print_to_file` is set to `FALSE` the polar plot will be displayed in the current R session. The name of the output file is that of the 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("M_PL_008.csv","data/",plotsdir="grafresults/",
                    print_to_file = TRUE))
```

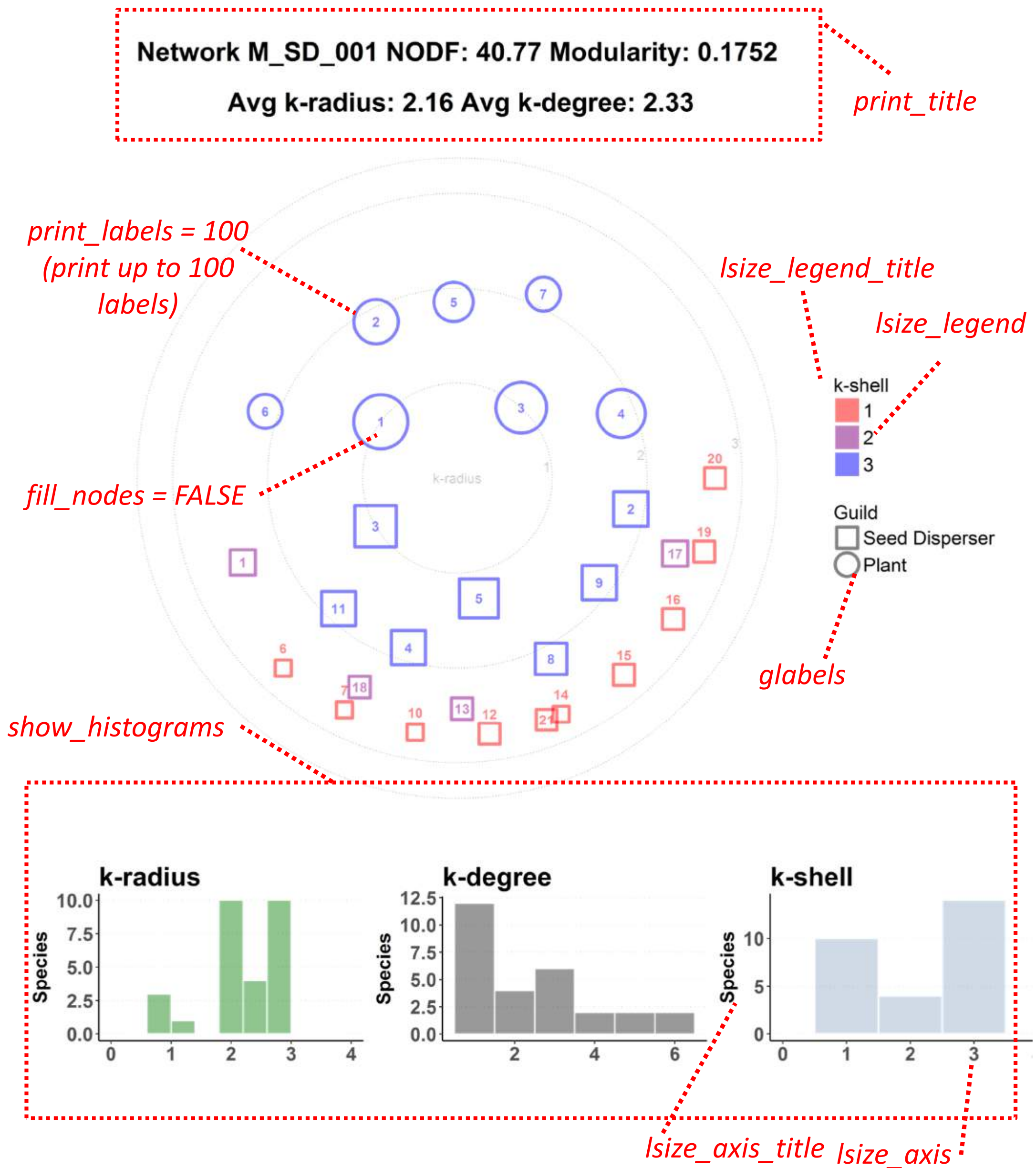
The plot title includes the network average k_{radius} and k_{degree}, a common measurement of *nestedness* called $NODF$ [AN+08] and $Modularity$ following the $QanBiMo$ method [DS14].



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

The Ziggurat Plot

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

Species of 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.

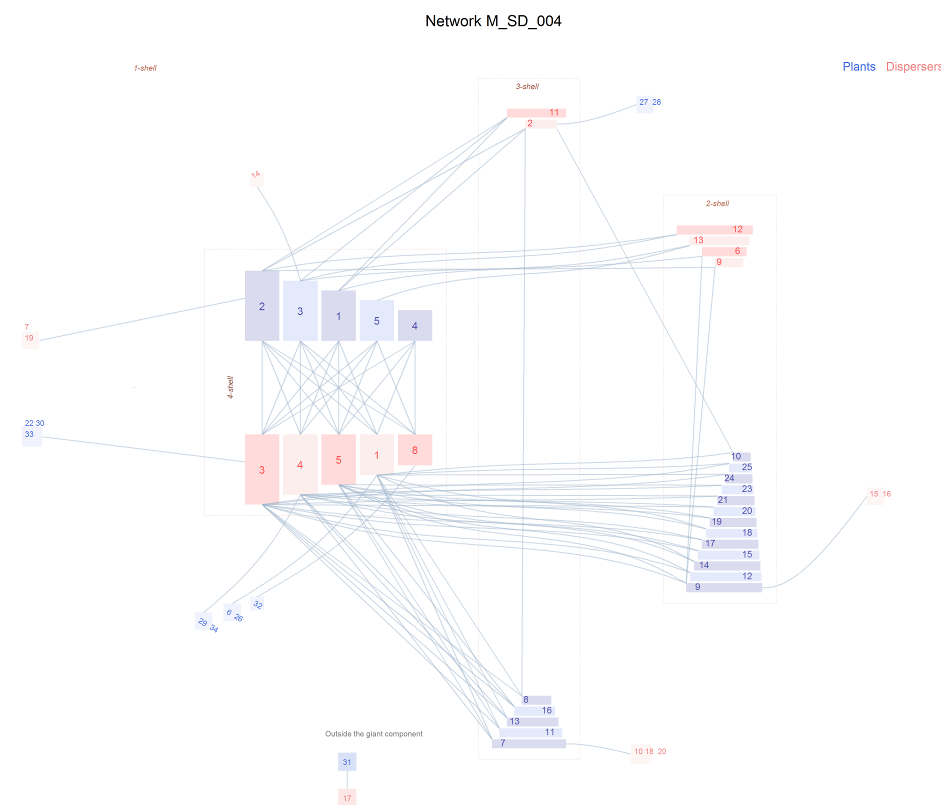


Figure 3: Ziggurat graph of an avian frugivore community in Puerto Rico with 54 species and 95 links [CCG03].

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(datadir, filename, paintlinks = TRUE,
  displaylabelszig = TRUE, print_to_file = FALSE,
  plotsdir = "plot_results/ziggurat/", 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), labels_size = 3.5,
  lsize_kcoremax = 3.5, lsize_zig = 3, lsize_kcore1 = 2.5,
  lsize_legend = 4, lsize_core_box = 2.5, labels_color = c(),
  height_box_y_expand = 1, kcore2tail_vertical_separation = 1,
  kcore1tail_disttocore = c(1, 1), innertail_vertical_separation = 1,
  horiz_kcoremax_tails_expand = 1, factor_hop_x = 1,
  displace_legend = c(0, 0), 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(1, 1), outsiders_separation_expand = 1,
  outsiders_legend_expand = 1,
  weirdskcore2_horizontal_dist_rootleaf_expand = 1,
  weirdskcore2_vertical_dist_rootleaf_expand = 0,
  weirds_boxes_separation_count = 1, root_weird_expand = c(1, 1),
  hide_plot_border = TRUE, rescale_plot_area = c(1, 1),
  kcore1weirds_leafs_vertical_separation = 1, corebox_border_size = 0.2,
  kcore_species_name_display = c(), kcore_species_name_break = c(),
  shorten_species_name = 0, label_strguilda = "", label_strguildb = "",
  landscape_plot = TRUE, backg_color = "white", show_title = TRUE,
  use_spline = TRUE, spline_points = 100, file_name_append = "",
  svg_scale_factor = 10, progress = NULL)
```

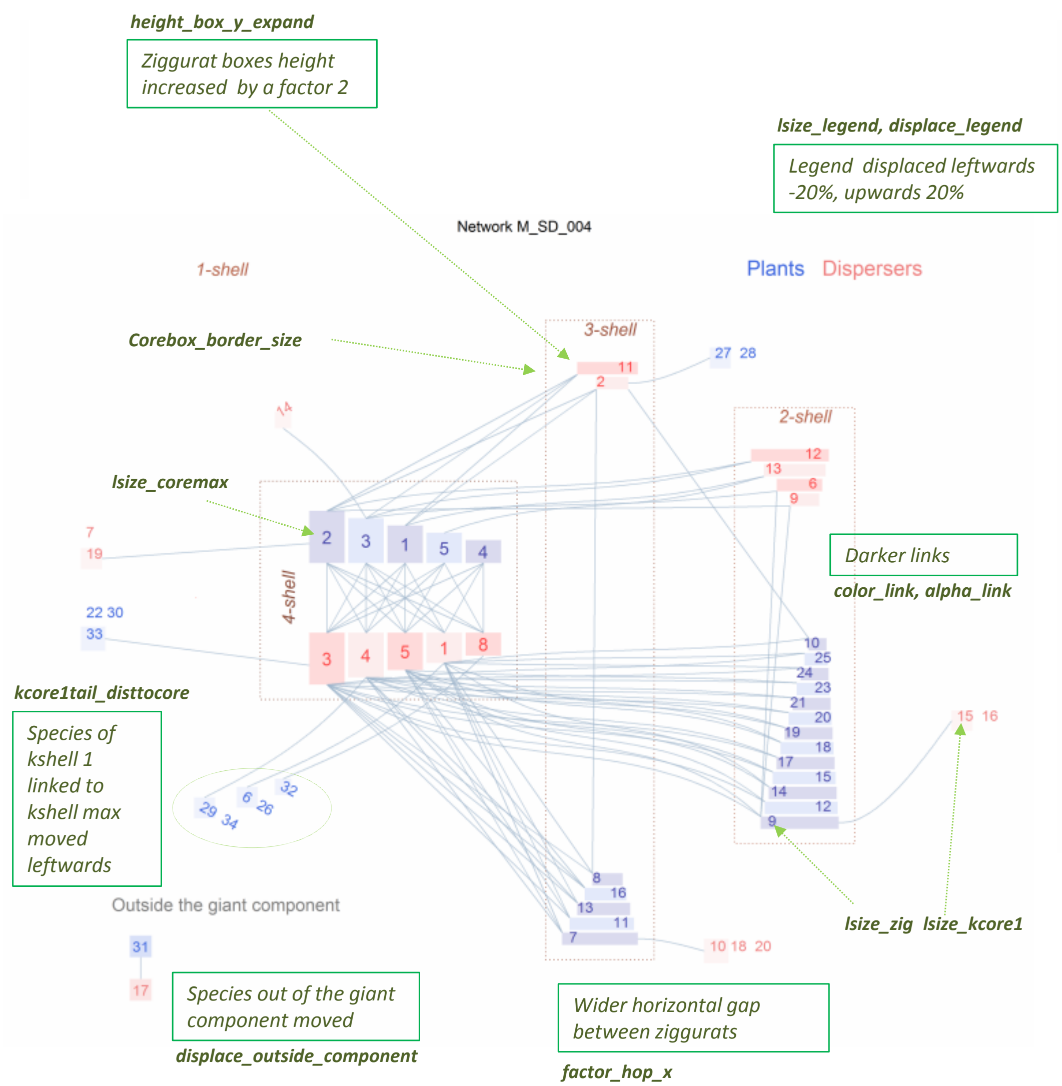
The configuration of ziggurat plots is richer but also more complex than that of polar graphs. Some parameters are equivalent to those of the polar plot, such as `filename`, `datadir` and `plotsdir` that provide the input file, input directory and output directory. The output file is called as the input 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 was created with the default call:

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

The same graph looks improved with some additional input data.

```
ziggurat_graph("data/","M_SD_004.csv", plotsdir = "grafresults/",
       height_box_y_expand = 2, 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(-0.3,1), lsize_legend = 7,
       lsize_core_box = 6, displace_legend = c(-0.2,0.2), print_to_file = TRUE)
```



Plot configuration may become quite complicated when network size grows, but we will show now the usefulness of another set of input parameters with a medium size 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)
```

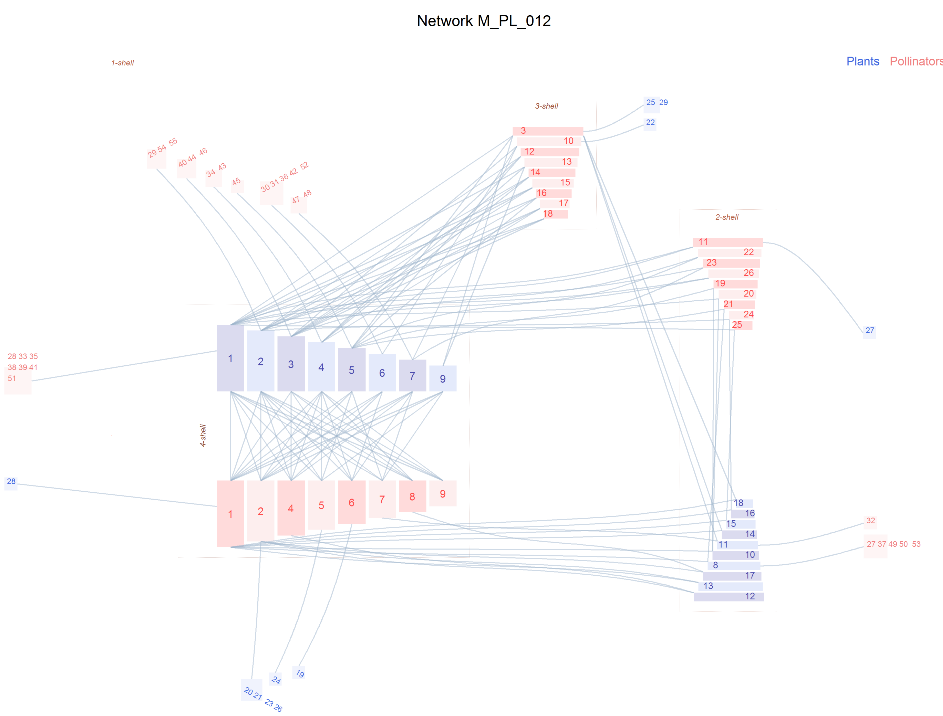
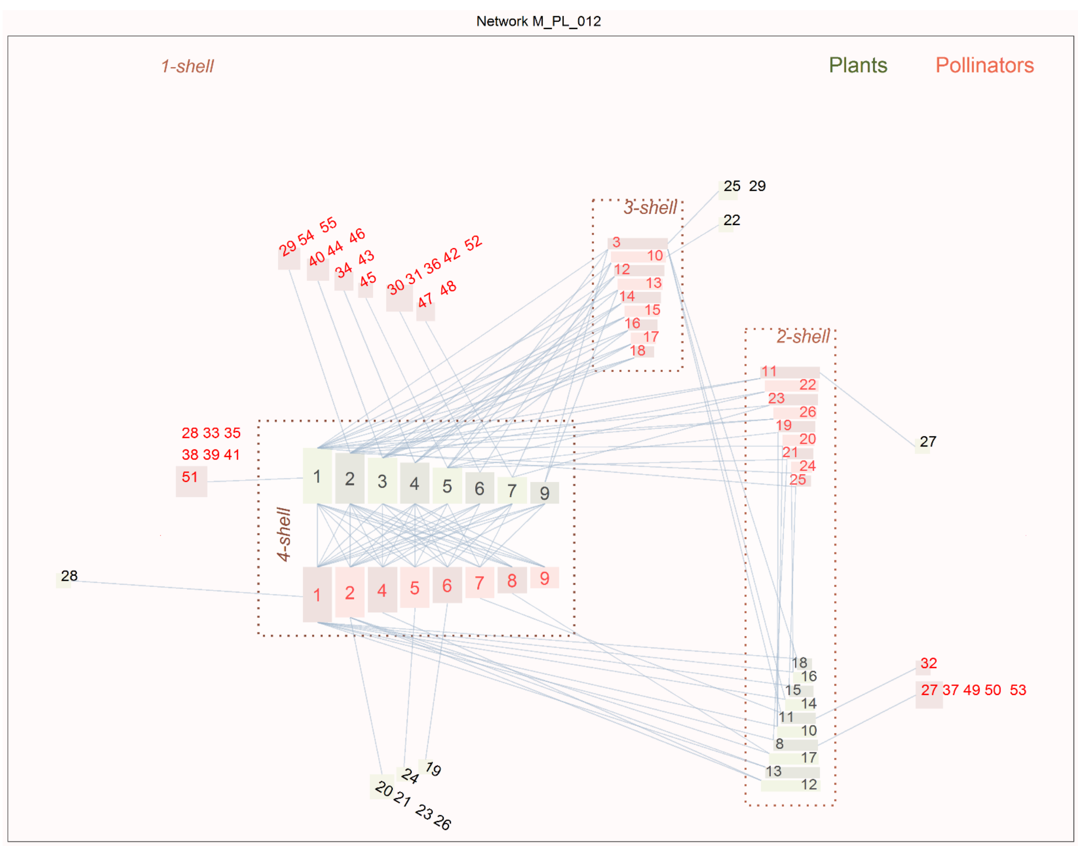


Figure 4: Ziggurat graph of a plant - pollinator network in Garajonay, La Palma (Spain). Olesen, unpublished.

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_legend =c(-0.2,0.2),
                        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 will not repeat the meaning of those that we have explained in the previous example. The role of some of the new ones are quite evident. 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 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, but we have added it to explain its meaning.

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%.

Finally, vertical separation of tails connected to inner ziggurats is increased with `innertail_vertical_separation`. See plant species 22 and 25.

Next example shows how to manage *weird tails* and *outsiders*. Weird 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.

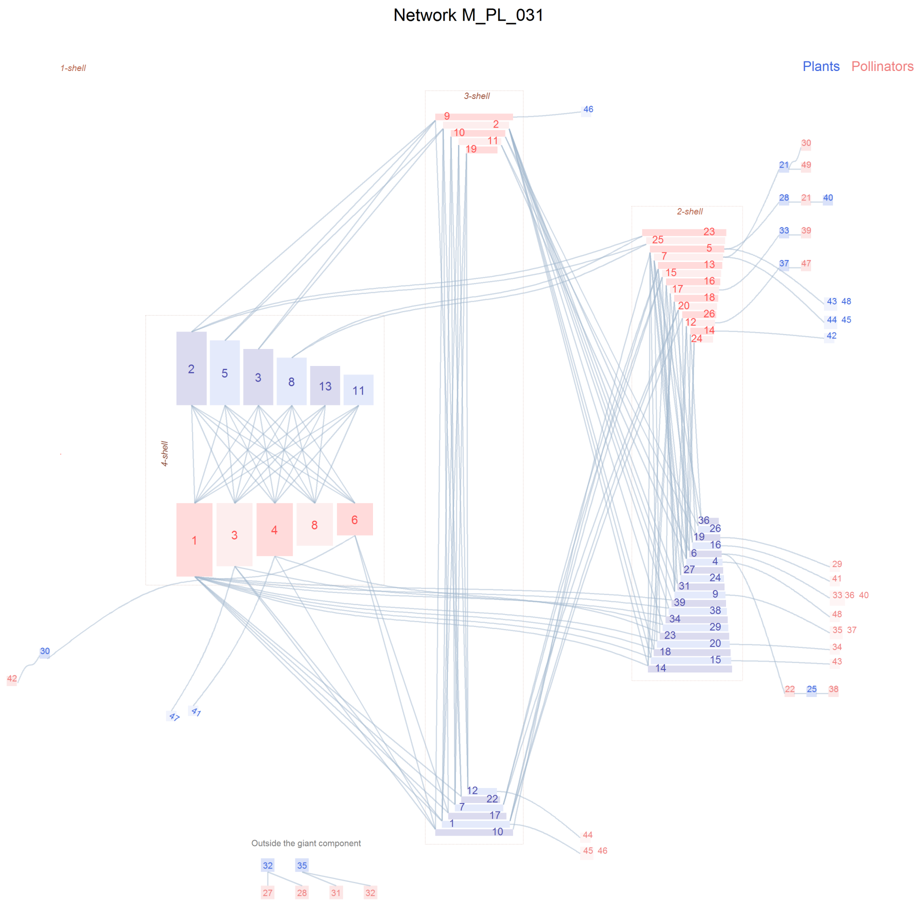
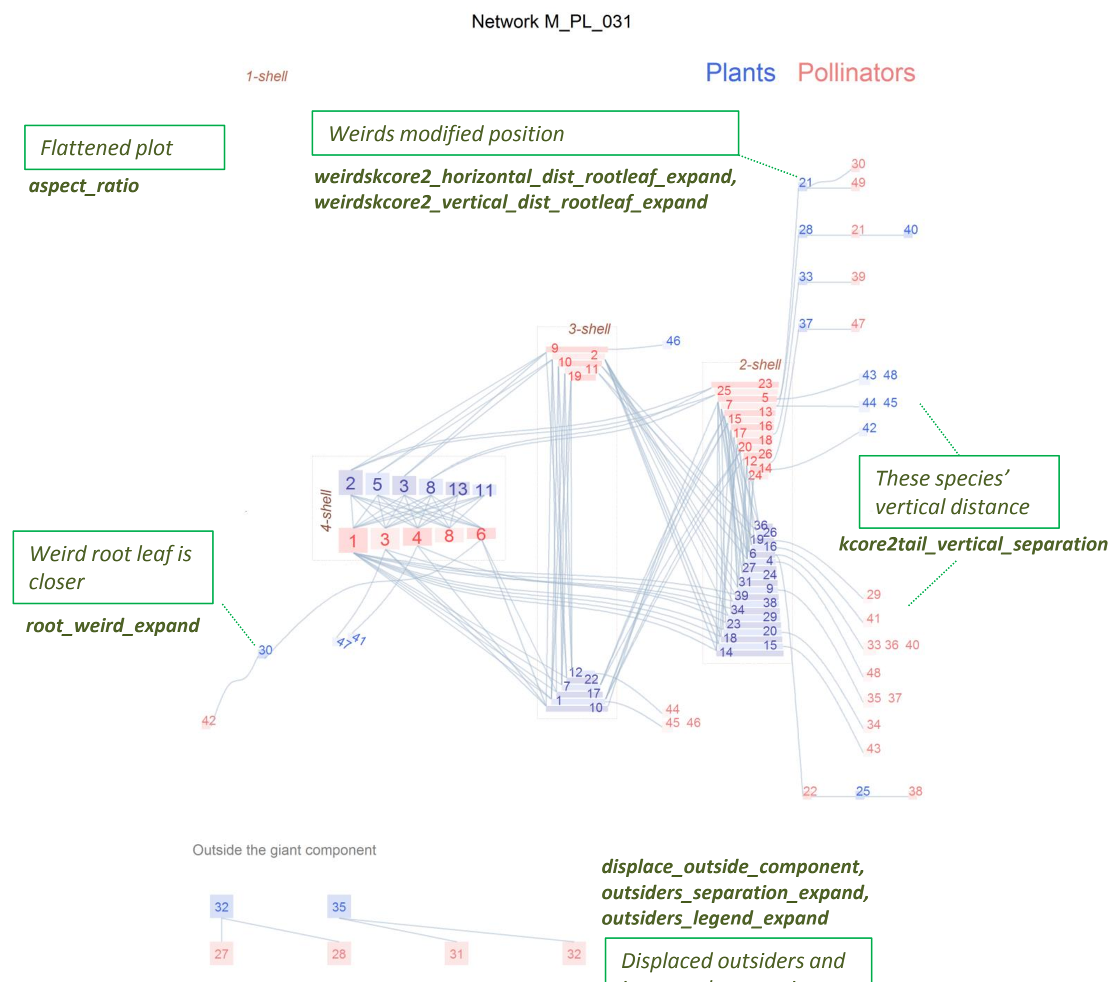


Figure 5: Default ziggurat graph of a plant - pollinator network in Alta Guyana (Venezuela) [Ram89].

Outsiders appear under the main plot. This network has a rich set of weird 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", aspect_ratio =0.50, height_box_y_expand =3,
                 plotsdir = "grafresults",color_link = "slategray3",
                 alpha_link = 0.5, lsize_core_box = 4,
                 displace_legend = c(-0.2,0.2),factor_hop_x = 1.4,
                 lsize_kcoremax = 5,lsize_zig = 3.5,lsize_kcore1 = 3.5,
                 corebox_border_size=0.2, displace_outside_component = c(-0.5,0.8),
                 outsiders_separation_expand = 2,
                 outsiders_legend_expand = 2, kcore2tail_vertical_separation = 3,
                 weirds_boxes_separation_count=3,root_weird_expand = c(0.5,1),
                 weirdskcore2_horizontal_dist_rootleaf_expand = -0.3,
                 weirdskcore2_vertical_dist_rootleaf_expand = 0.5,lsize_legend = 7,
                 print_to_file = TRUE, file_name_append = "improved")
```



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

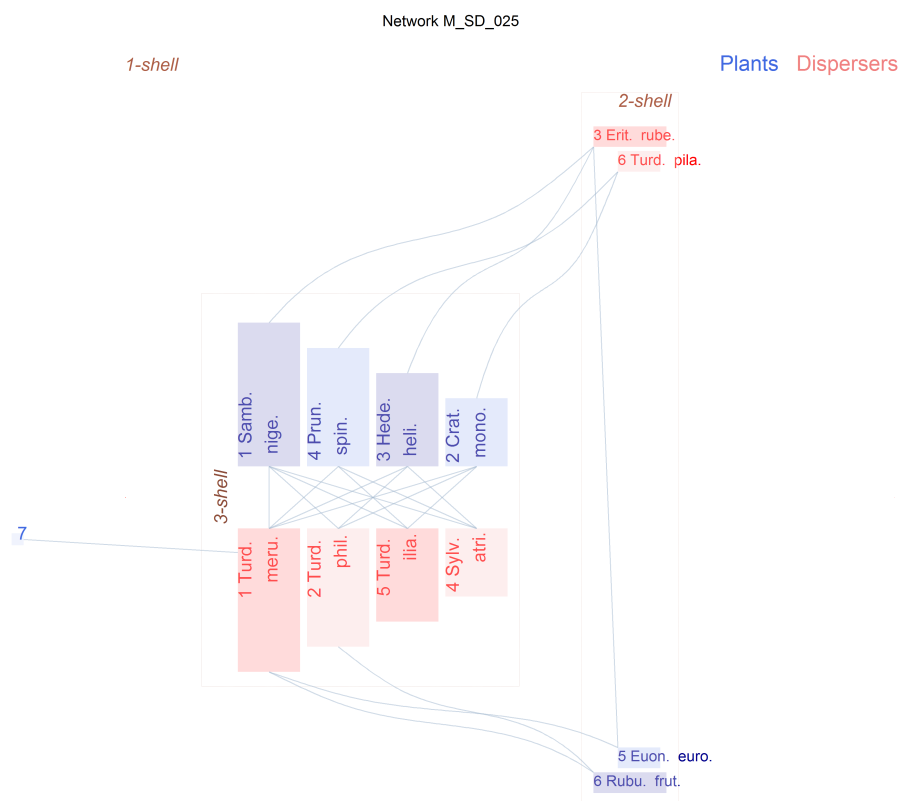


Figure 6: Ziggurat graph of a plant - disperser.

```
ziggurat_graph("./data/","M_SD_025.csv",plotsdir="grafresults/",
               shorten_species_name = 4, displace_legend = c(-0.2,0.2),
               height_box_y_expand = 2, coremax_triangle_width_factor = 1.25,
               coremax_triangle_height_factor = 2.25,lsize_core_box = 6,
               lsize_kcoremax = 6, lsize_legend= 7,lsize_kcore1 = 6,
               lsize_zig = 5, kcore_species_name_display = c(2,3),
               kcore_species_name_break = c(3), print_to_file = TRUE)
```

If the network is weighted, links width may be a function of the interaction strength, as in figure 7. There are two options, the natural or decimal logarithm of the weight.

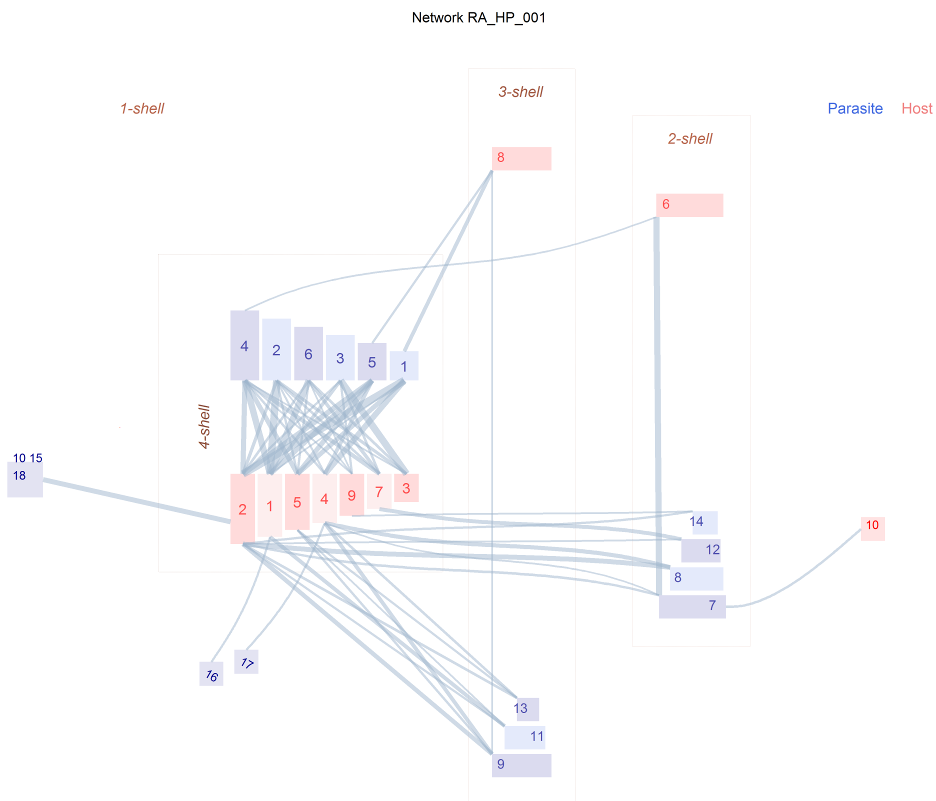


Figure 7: Ziggurat graph of a host - parasite network.

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

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.

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 weird tails connected to *max–shell*.
- `factor_hop_x expand inner`: expands ziggurats horizontal distance.
- `displace_legend modify`: expands legend position by these fractions.
- `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.
- `weirdskcore2_horizontal_dist_rootleaf_expand`: expand horizontal distance of weird tail root node connected to 2–*shell*.
- `weirdskcore2_vertical_dist_rootleaf_expand`: expand vertical distance of weird tails connected to 2–*shell*.
- `weirds_boxes_separation_count`: weird species boxes separation count.
- `root_weird_expand`: expand root weird 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).
- `kcore1weirds_leafs_vertical_separation`: expand vertical separation of weird 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.
- `move_all_SVG_up`: only for interactive apps, do not modifiy.
- `progress`: only for interactive apps, do not modifiy.

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).

[AN+08] Mário Almeida-Neto et al. "A consistent metric for nestedness analysis in ecological systems: reconciling concept and measurement". In: *Oikos* 117.8 (2008), pp. 1227–1239.

[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.

[DS14] Carsten F Dormann and Rouven Strauss. "A method for detecting modules in quantitative bipartite networks". In: *Methods in Ecology and Evolution* 5.1 (2014), pp. 90–98.

[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.