The HiveR Package
Version 0.2-1

Bryan A. Hanson

DePauw University
Department of Chemistry & Biochemistry
Greencastle Indiana USA

e-mail: hanson@depauw.edu

github.com/bryanhanson/HiveR
CRAN.R-project.org/package=HiveR

December 12, 2011

This document describes some features of the `HiveR` package including current capabilities and future plans. The current release contains a core set of functions for creating and drawing hive plots. Additional features are contemplated. There may well be bugs and features that can be improved. Your comments are always welcome.

As with any `R` package, details on functions discussed below can be found by typing `?function_name` in the `R` console after installing `HiveR`. A complete list of functions available can be had by typing `?HiveR` and then at the bottom of the page that opens, click on the "index" link.

1 Background, Inspiration and Motivation

`HiveR` was inspired by the concept of hive plots as developed by Martin Krzywinski at the Genome Science Center (www.hiveplot.com). Hive plots are a reaction to "hair ball" style networks in which the layout of the network is arbitrary and hypersensitive to even small changes in the underlying network. Hive plots are particularly useful for the discovery of emergent properties of networks.

The key innovation in a hive plot, compared to other means of graphically displaying network structure, is in how node information is handled. Nodes are assigned to axes based upon qualitative or quantitative characteristics of the the node, for instance membership in a certain category, and the position of the node along the axis is based upon some quantitative characteristic of the node. In a hive plot, edges are handled in a fairly standard way, but may be colored or have a width or weight which encodes an interesting value. In creating a hive plot, one maps network parameters to the hive plot, and thus the process can be readily tuned to meet one's needs. The mappable parameters are listed in Table 1, and the mapping is limited only by one's creativity and the particular knowledge domain. Thus ecologists have their own measures of food webs, social network analysts have various measures describing interconnectedness etc. An essential point is that mapping network parameters in this way results in a reproducible plot which is particularly well-suited for comparing related networks. Comparison of "hair balls" is notoriously fraught with problems.

Krzywinski has an excellent paper detailing the features and virtues of hive plots and is a must-read.[1] He notes the following virtues of hive plots:

- Hive plots are rational in that only the structural properties of the network determine the layout.
- Hive plots are flexible and can be tuned to show interesting features.
- Hive plots are predictable since they arise from rules that map network features to plot features.

mappable hive plot parameters
Axis to which a node is assigned
Radius of a node
Color of a node
Size of a node
Color of an edge
Width or weight of an edge

Table 1: Hive plot features that can be mapped to network parameters

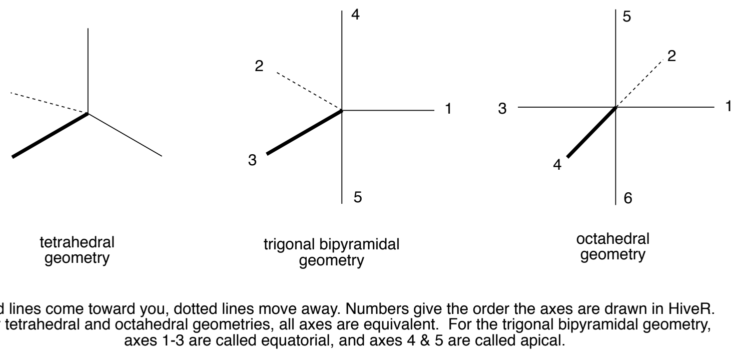
4
2
1
3
5
5
2
3
1
4
6
tetrahedral geometry
trigonal bipyramidal geometry
octahedral geometry
Bold lines come toward you, dotted lines move away. Numbers give the order the axes are drawn in HiveR.
For tetrahedral and octahedral geometries, all axes are equivalent. For the trigonal bipyramidal geometry, axes 1-3 are called equatorial, and axes 4 & 5 are called apical.

Figure 1: Idealized geometries according to VSEPR theory

- Hive plots are robust to changes in the underlying network.
- Hive plots of different networks can be compared.
- Hive plots are transparent and practical.
- Plots of networks are generally complex and require some investment to understand. Complexity scales well in a hive plot and details can be inspected.

For comparison, Suderman and Hallett have published a nice review of a wide range of other programs for visualizing biological networks though it is now slightly out of date.[2]

Inspired by the examples given by Kryzwinski in his materials on the web, I created the `R` package `FuncMap` in December 2010. This single function package maps the function calls made by an `R` package into 3 types: sources, which are functions that make only outgoing calls, sinks, which take only incoming calls, and managers, which do both. Figure 2 shows an example of a plot made by `FuncMap`; this is a true hive plot. In this plot, functions in a package are assigned to an axis by their role, and the radius is determined by the number of calls made or received by a function (which is the number of edges or degree of the node). This is also the basis for the width of the edges. In this plot, calls (edges) originating on the source axis are shown in green, while those originating on the manager axis are in blue. By defintion, the sink axis only receives calls.

`HiveR` takes things quite a bit further. `HiveR` is intended as an implementation of hive plots in `R`, not a port of linnet *per se* (Krzywinski's program that draws hive plots, written in Perl). As such, it does some things differently, and not all features are implemented (and they may or may not be in the future). `HiveR` will draw 2D hive plots with 2-6 axes in a style close to that created by linnet. However, `HiveR` adds value by making 3D, interactive plots possible when there are 4-6 axes. These 3D plots were inspired by the ideas of VSEPR theory in chemistry: the axes of these 3D plots are arranged with tetrahedral, trigonal bipyramidal or octahedral geometries for 4-6 axes respectively (see Figure 1 and wikipedia/VSEPR). Other differences are discussed below.

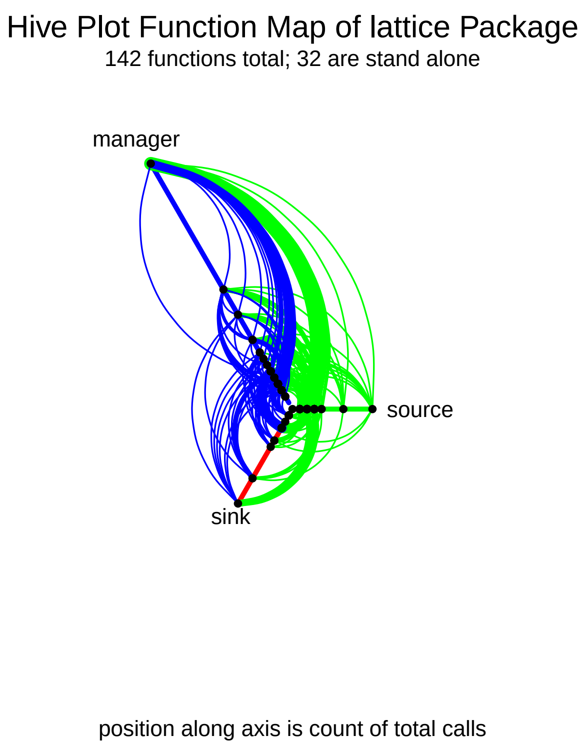

Figure 2: FuncMap for package lattice

2 `HiveR` Features

2.1 Internal Storage

`HiveR` stores the information needed to create a hive plot in a `HivePlotData` object which is an S3 class. As an S3 class, this structure can be easily extended by the user to store additional information (though using that information as part of a hive plot would require more work). Utilities are provided to summarize the contents of these objects and to check their integrity (functions `sumHPD` and `chkHPD` respectively). The structure and content of a `HivePlotData` object is shown in Table 2.

2.2 Generation of Random Network Data Sets

`HiveR` has the ability to generate random network data sets with between 2 and 6 axes, using function `ranHiveData`. These are useful for testing and demonstration purposes and will be used in the examples below. A data set has a type, either 2D or 3D. Type 2D may have 2-6 axes and is plotted in a 2D window using `grid` graphics which are extremely fast. Type 3D applies to 4-6 axes only and these hive plots are drawn in 3D using `rgl` and are interactive. When using `ranHiveData` you can specify which type you desire.

2.3 Built-in Data Sets

`HiveR` contains two related 2D type data sets, `Safari` and `Arroyo`. These plant-pollinator data sets give the number of visits for each plant-pollinator pair. The *E. coli* gene regulatory network is also included as a .dot file. This data is discussed in Yan *et. al.*[3] but is based upon data in the RegulonDB.[4] The version here was extended by Krzywinski

element	*(element)*	*type*	*description*
$nodes		data frame	Data frame of node properties
	$id	int	Node identifier
	$lab	chr	Node label
	$axis	int	Axis to which node is assigned
	$radius	num	Radius (position) of node along the axis
	$size	num	Node size in pixels
	$color	chr	Node color
$edges		data frame	Data frame of edge properties
	$id1	int	Starting node id
	$id2	int	Ending node id
	$weight	num	Width of edge in pixels
	$color	chr	Edge color
$type		chr	Type of hive (2D or 3D)
$desc		chr	Description of data
$axis.cols		chr	Colors for axes
- attr		chr "HivePlotData"	The S3 class designation

Table 2: The structure of a HivePlotData object

and provided in the linnet package. This .dot file can be processed into either a 2D or 3D type hive plot. Each of these data sets are used in the examples below.

2.4 Importing Real Data Sets

The function `dot2HPD` will import files in .dot format and convert them to `HivePlotData` objects (see wikipedia/DOT_language). This is done with the aid of two external files. One contains information about how to map node labels to `HivePlotData` properties. The other contains information about mapping edge properties. This approach gives one a lot of flexibility to process the same graph into various hive plots. This process is demonstrated later for the *E. coli* data set. Currently, only a very small set of the .dot standard is implemented and one should not expect any particular .dot file to process correctly.

2.5 Modifying `HivePlotData` Sets

Function `mineHPD` has several options for extracting information within an existing `HivePlotData` object and converting it to a modifed `HivePlotData` object. Currently, there are three options, but more are easily added. One option assigns the radius of a node based upon the number of edges connected to it (the degree). Another assigns axes based upon whether a given node is a source node, manager node or sink node. This latter option is designed to create hive plots similar to those featured by Krzywinski for the *E. coli* data set, and is demonstrated later. The final option removes any orphaned nodes (these have no edges). In addition, function `manipAxis` can also be used to modify a `HivePlotData` object by scaling or inverting axes.

2.6 Making Hive Plots

In a hive plot, because the position of the node along an axis (the radius) is quantitative, the nodes can be plotted at their absolute value (native units), normalized to run between $0 \ldots 1$, plotted by rank or by a combination of ranking and norming. Some aspects of the plot that depend upon these options are shown in Table 3. These different ways of plotting the same data often look dramatically different, and for a particular data set, some methods of plotting nodes may provide more insight. Functions `plotHive` and `plot3dHive` have an argument `method` which controls node plotting on the fly; function `manipAxis` is used in the background and can be called independently if desired.

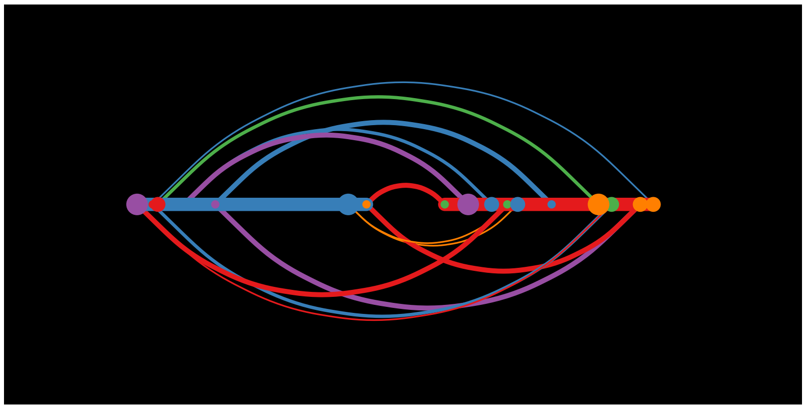

Figure 3: A randomly generated hive plot with 2 axes (native units)

method	*axis length*	*center hole*	*other*
native units (abs)	varies ($\propto$ *no. nodes*)	asymmetric	nodes may overlap
ranked units (rank)	varies ($\propto rank(no.\ nodes)$)	circular	nodes evenly spaced (1, 2, 3 ...) and don't overlap
normed units (norm)	all equal	circular	nodes may overlap
ranked & normed (ranknorm)	all equal	circular	nodes evenly spaced (1, 2, 3 ...) and don't overlap

Table 3: Comparison of methods for plotting node radii

2.6.1 Type 2D Hive Plots

Figures 3 shows a 2 axis hive plot using randomly generated data and the function `plotHive`. Figure 4 shows a hive plot of a random 3 axis network using absolute scaling; Figure 5 shows the 3 axis example with the nodes displayed by rank and Figure 6 the same data normed. FIgure 7 shows a 5 axis example. `plotHive` places axis number 1 at the top (vertical) except in the 2 axis case where it is on the right. Nodes are drawn in these examples, however, drawing nodes is optional and the more nodes there are, the less likely you will want to draw them. As these plots show, depending upon their size and radii, nodes may overlap. The nodes "on top" will be those drawn last (also true of edges). In some cases users may wish to sort the nodes and edges so that certain nodes and edges are drawn last and thus "show". Nodes and edges with various characteristics can also be subsetted and recombined if simple sorting won't do the job. This method is used in some of the examples which follow.

2.6.2 Type 3D Hive Plots

With type 3D and 4 to 6 axes, plots are interactive and cannot be shown here. See the help page for `plot3dHive` for an example you can run when have the package installed (`?plot3dHive`). Note that `plot3dHive` has an argument `LA` which controls whether antialiasing is used when drawing the edges. `LA` defaults to `FALSE` which plots quickly. Further testing and optimization is needed, but `LA = TRUE` should probably be reserved for making final plots, as it is at least 20 times slower.

2.6.3 Performance

`HiveR` draws hive plots very quickly when using either `plotHive` or `plot3dHive`. As of version 0.1-5, the bottlenecks holding `plot3dHive` back have been eliminated. Figure 8 shows the performance of this function on a MacBook Pro running OSX 10.6.8 using 8 Mb RAM and an Intel i7 chip at 2 GHz. As of version 0.1-6, speed improvements have been made to `plotHive` and Figure 9 shows the performance on the same hardware. These benchmarks were determined before byte compiling was turned on and so the performance is likely even better.

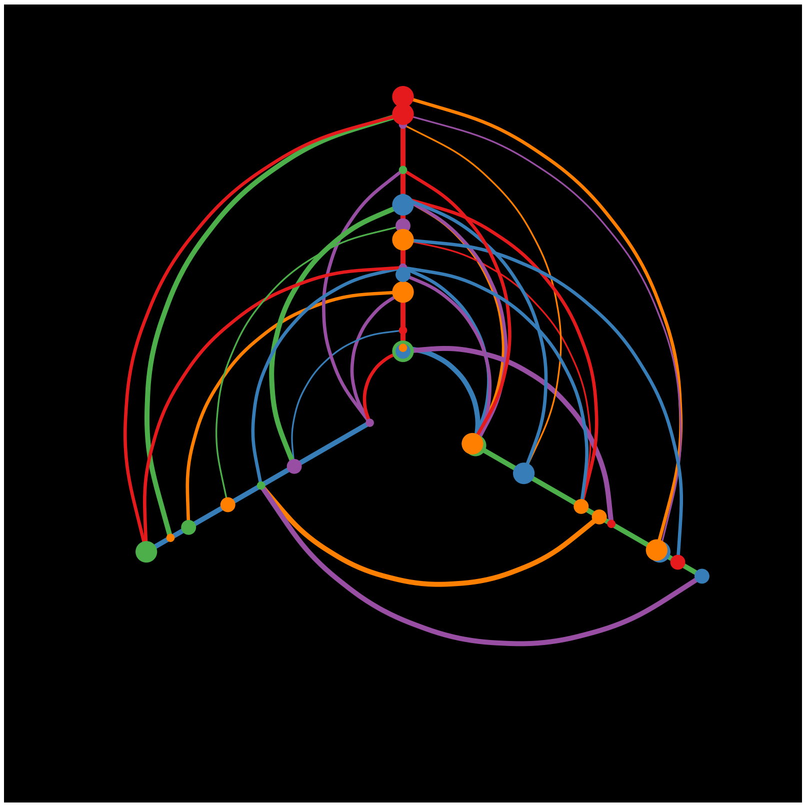

Figure 4: A randomly generated hive plot with 3 axes (native units)

2.7 Some Things to Keep in Mind

1. As currently implemented in `HiveR`, hive plots are agnostic graphs in that they are not necessarily directed or undirected. However, some of the functions actually do draw edges in a way that could readily be converted into a directed graph in the future. For example, `plotHive` draws edges between axes 1 and 2 in a separate step from those starting on 2 and ending on 1. This is so that the correct curvature of the splines is used, but it could be used to encode directionality. Further, some options in `mineHPD` assume that the `HivePlotData` object represents a directed graph, and while `dot2HPD` currently doesn't distinguish between directed and non-directed graphs, it could in the future.

2. linnet creates hive plots that are essentially parallel coordinate plots[5] that have been wrapped into a radial arrangement. `HiveR` plots of type 2D are essentially the same thing. As with any parallel coordinate plot, the order of the axes affects what you see. With 2 or 3 axes this isn't a problem. For 4-6 axes and type 2D, the user has to give some thought as to how to assign the axes. One should assign the axes in a way that avoids edges jumping over or crossing an axis when using type 2D. Edges should be arranged $1 \rightarrow 2$, $2 \rightarrow 3$, $\ldots 5 \rightarrow 6$ but not $1 \rightarrow 4$ for example. Function `sumHPD` with `chk.ax.jump = TRUE` will tell you if any edges cross. For type 3D, one doesn't have to worry about this, but must guard against edges that start and end on the same axis or start and end on colinear axes. `ranHiveData` takes care of these exceptions automatically. By they way, these conditions don't cause errors, but they overdraw the axes and it doesn't look good.

3. On the other hand, `HiveR` plots using type 3D are not a parallel coordinate plots. For 4 axes plotted as a tetrahedron, any pair of axes are intrinsically next to each other and it is not possible for an edge to cross another axis. For 5 and 6 axes, crossings are a potential problem but generally it is possible to connect axes in more combinations than for type 2D. For instance, with 5 axes and type 2D, any one axis is between only 2 other axes, and hence can be connected to at most 2 other axes. But for type 2D and 5 axes, an axis in the apical position can be connected to 3 other axes, and an axis in the equatorial position can be connected to 4 other axes (could use a diagram showing this).

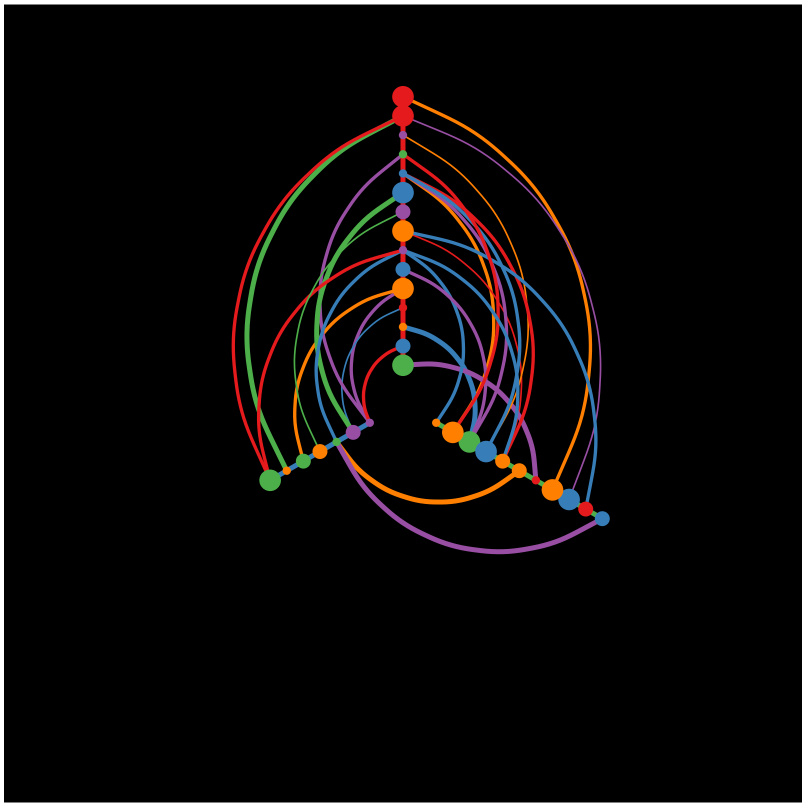

Figure 5: A randomly generated hive plot with 3 axes (nodes by rank)

4. Some ideas for network parameters that might be mapped to node radii (see Table 1):
   (a) Ecology: see various species descriptors computed by function `specieslevel` in package `bipartite`.
   (b) Social networks: see the section "Node-level indices" in the article describing package `sna`.[6] Briefly, degree, betweeness and closeness are the key ideas.
   (c) See Table 1 in the article by Krzywinski.[1]

3 A Simple Example Using a Plant-Pollinator Network

`HiveR` currently contains the built-in data sets, `Safari` and `Arroyo` which provide a useful demonstration of `HiveR`.[1] These are plant-pollinator data sets which were derived from Vasquez and Simberloff, 2003 [7]. These describe two-trophic level systems that consist of almost exactly the same suite of plants and pollinators. `Safari` is based upon observations of an undisturbed area, while `Arroyo` is from a nearby location grazed by cattle. The original data is composed of plant-pollinator pairs and a count of visits for each pair.

Figures 10 and 11 show two means of plotting `Safari` using package `bipartite`.[2] Figure 10 is a simple diagram giving plant-pollinator visits as a gray scale heat map. There are two parameters encoded here: the pairings and the number of visits (arguably, the dimensions of the matrix give the number of species involved as well). Figure 11 displays plants across the bottom and pollinators across the top. The width of the connecting bands in the middle encodes the number of visits for a given plant-pollinator pair. The width of the top or bottom panel for a species is the total number of visits in which that species participates. Thus there are three parameters shown in this figure: the pairings, the total visits for a single species, and visits between a given pair. This second plot makes it pretty clear that four plant-pollinator pairs have by far the most number of visits.

[1]Be warned: I am not an ecologist and these data sets and plots are merely a demonstration of `HiveR`.
[2]Note that we are using the data set `Safariland` from package `bipartite`; `Safari` was derived from `Safariland`.

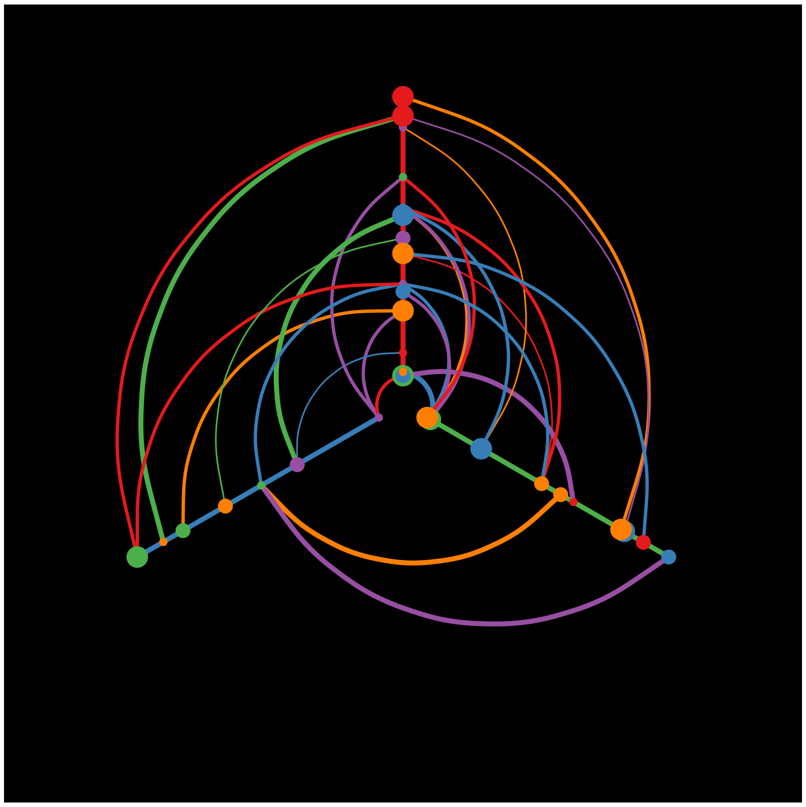
Figure 6: A randomly generated hive plot with 3 axes (nodes normed)

Another approach to presenting this network graphically would be to use function `gplot` in the very powerful social network analysis package `sna`. `gplot` is flexible and has many options. Figure 12 shows one possible display of `Safari` (actually, `Safariland`). In this plot, plant nodes are colored green and insect nodes red. The width of the edges is proportional to the number of visits between a pair of species. Figure 13 shows the same data using a different layout algorithm, one which shows that there are actually two networks present (and which is not apparent from the hive plots below). Edge width here is the same as before, but because high traffic pair nodes are close to each other, the connecting, wide edge looks a bit odd (clearly, one could experiment to improve this detail).

Figures 14 and 15 show `Safari` and `Arroyo` respectively, using `plotHive` (instrinically type 2D since there are only 2 axes in the data set). In these plots, plants are on one axis, and pollinators are on the other. Each organism was assigned a radius on its axis based by calculating d' using function `dfun` in package `bipartite`. d' is an index of specialization; higher values mean the plant or pollinator is more specialized.[3] Edge weights were assigned proportional to the square root of the normalized number of visits of a pollinator to a plant. Thus the width of the edge drawn is an indication of the visitation rate. The transformed number of visits was divided manually into 4 groups and used to assign edge colors ranging from white to red. The redder colors represent greater numbers of visits, and the color-coding is comparable for each figure. Thus both the edge color and the edge weight encode the same information. It would of course be possible to encode an additional variables by changing either edge color or weight, or node size. These plots show a rich amount of information not available from the more standard plots and show that the networks are fundamentally different:

- The degree of specialization with each network is different. A greater number of visits (wider, redder edges) occur between more specialized species (nodes at larger radii) in `Safari` than `Arroyo`.
- There are more plant species in `Arroyo`: the plant axis is longer.
- The huge number of visits encoded in red in `Safari` (the ungrazed site) is missing in `Arroyo`, which was an interesting aspect of the study.

[3]These plots use the absolute value of d' for the node radii.

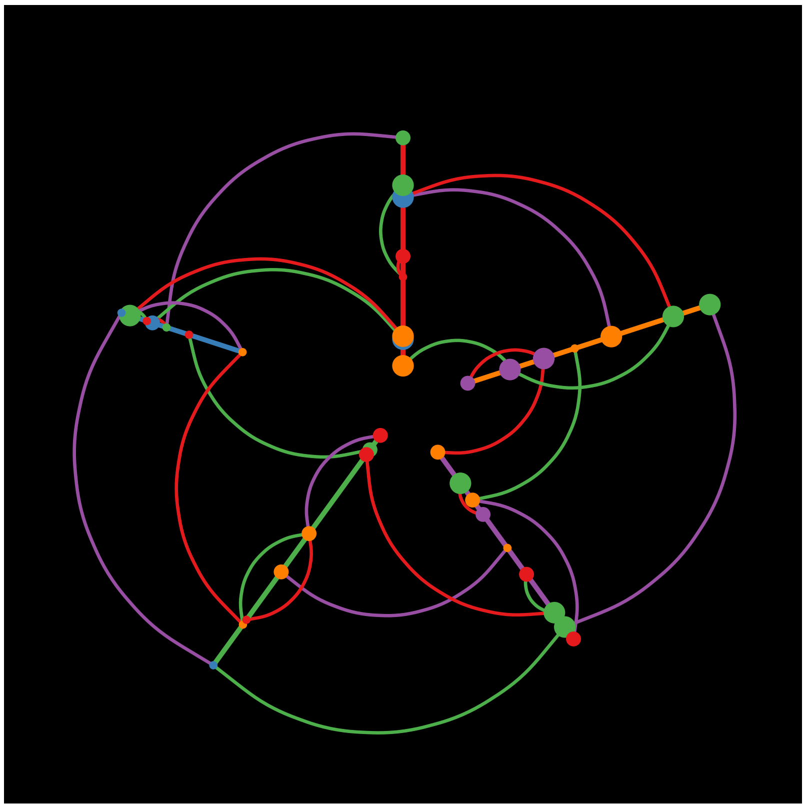

Figure 7: A randomly generated hive plot with 5 axes (native units; edges along the same axis permitted)

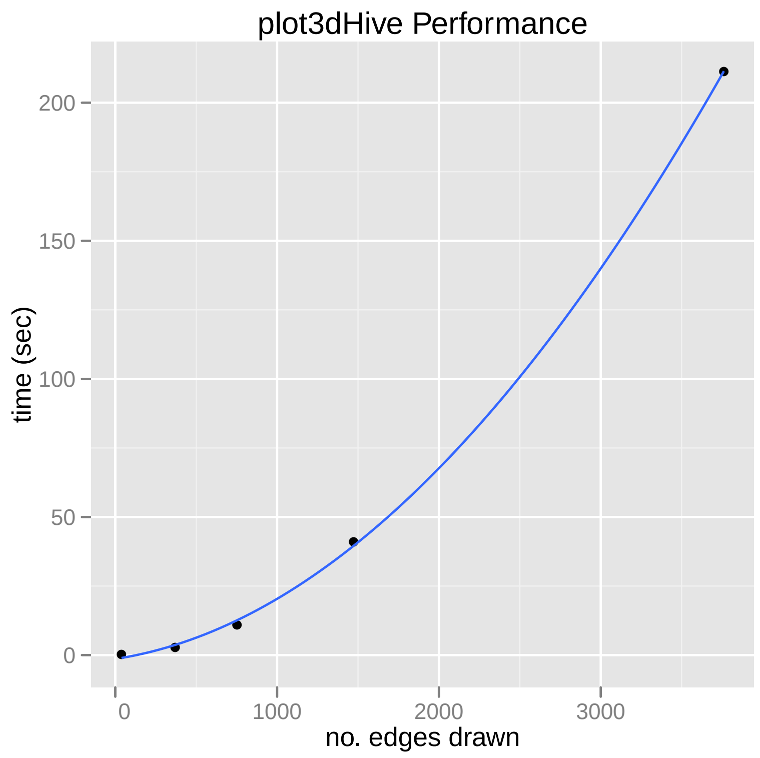

Figure 8: Performance of plot3dHive

4 The E. coli Gene Regulatory Network

HiveR includes the *E. coli* gene regulatory network, discussed in Yan *et. al.*[3] and based upon the RegulonDB[4] and extended by Krzywinski. It is contained in a file called `ecoli.dot` in the `extdata/E_coli` directory. It can be read in with `dot2HPD` and further processed with `mineHPD` as shown below. `dot2HPD` relies on two external .csv files which tell the function how to map node and edge information in the .dot file to the `HivePlotData` object. Tables 4 and 5 show the contents of the files used in this case. If you choose to draw the nodes, persistent nodes will be red and non-persistent nodes grey. The type of edge (1. . . 4) is also encoded by color. Gene pairs (edges) that are closer physically and genetically are colored gray $\rightarrow$ yellow $\rightarrow$ orange $\rightarrow$ red with red being the most related pairs.

dot.tag	dot.val	hive.tag	hive.val
label	persistent	color	red
label	nonpersistent	color	black

Table 4: Contents of NodeInst.csv

First, read in the data set and process it using the two external files (this assumes your working directory is set to the folder with the relevant files).

```
> EC1 <- dot2HPD(file = "ecoli.dot",
+        node.inst = "NodeInst.csv",
+        edge.inst = "EdgeInst.csv",
+        desc = "E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010)) ",
+        axis.cols = rep("grey", 3))
```

Next, assign the node radius based upon the edge degree. Then assign the nodes to axes based upon their role as source, manager or sink. Finally, let's remove any orphaned nodes (nodes that have no edges). Note that if desired, >

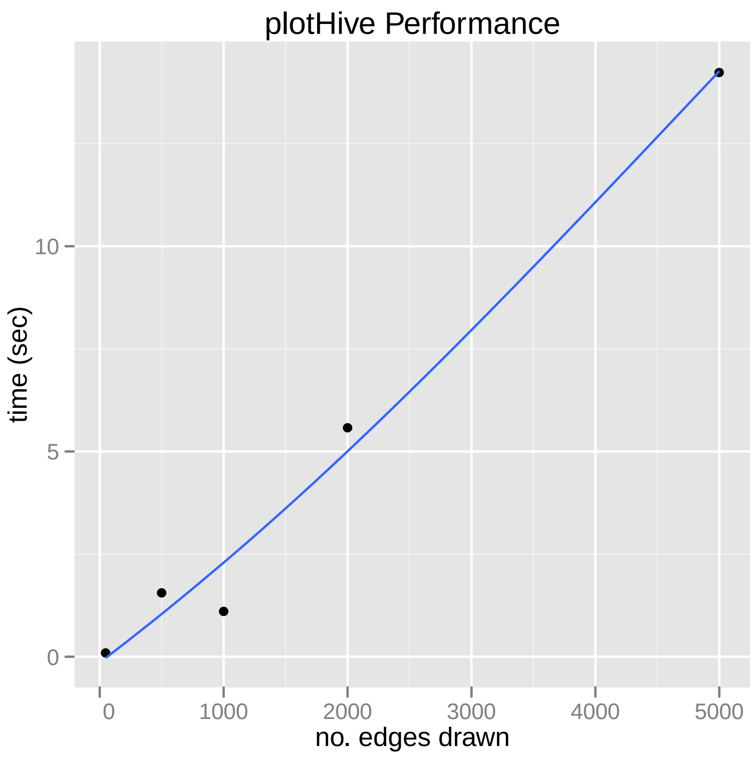

Figure 9: Performance of plotHive

dot.tag	dot.val	hive.tag	hive.val
type	0	color	grey
type	1	color	yellow
type	2	color	orange
type	3	color	red

Table 5: Contents of EdgeInst.csv

`sumHPD(EC3, chk.orphan.node = TRUE)` could be used to preview the list of orphans.

```
> EC2 <- mineHPD(EC1, option = "rad <- tot.edge.count")
> EC3 <- mineHPD(EC2, option = "axis <- source.man.sink")
> EC4 <- mineHPD(EC3, option = "remove orphans")

         No orphaned nodes were found
```

If you try to plot this now (`> plotHive(EC4)`), you encounter an error because two edges start and end on the same node (so they are on the same axis with the same radius). This would result in an edge length of zero, which is not possible (see `?sumHPD` for more details). We can use `sumHPD` to find out where the problem is. It turns out that two nodes are common to both problem edges. To avoid this problem, we'll nudge one node to a different value.

```
> sumHPD(EC4, chk.sm.pt = TRUE)

         E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010))
        This hive plot data set contains 1378 nodes on 3 axes and 2966 edges.
        It is a  2D data set.

                Axis 1 has 64 nodes spanning radii from 1 to 214
```

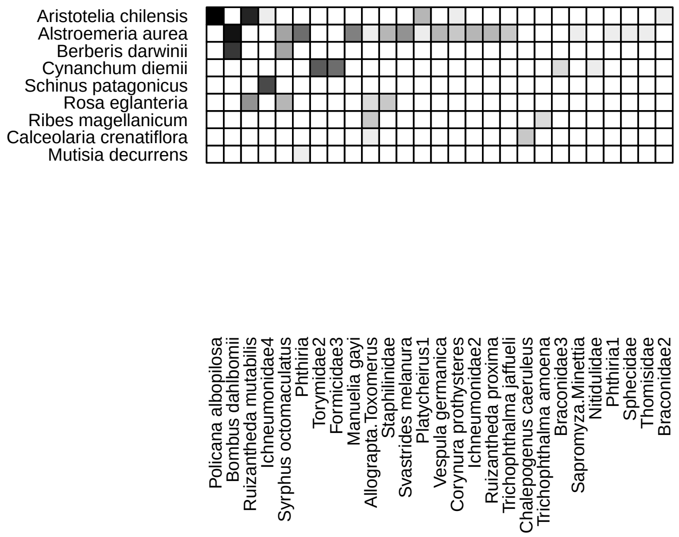

Figure 10: Safariland data set using visweb

```
            Axis 2 has 1243 nodes spanning radii from 1 to 9
            Axis 3 has 71 nodes spanning radii from 2 to 402

        The following edges start and end at the same point and the
        corresponding nodes should be deleted, offset or
        jittered (or the edge deleted) before plotting:

 n1.id n1.ax n1.lab n1.rad n2.id n2.ax n2.lab n2.rad e.wt e.col
  1149     3   srlr      9   547     3   gutm      9    1  grey
   547     3   gutm      9  1149     3   srlr      9    1  grey
```

```
> EC4$nodes$radius[1149] <- 9.5
```

Finally, we'll need to organize the edge list so that the reddest edges are drawn last, which will make the plots a bit easier to interpret (see later for another approach).

```
> edges <- EC4$edges
> gray_edges <- subset(edges, color == "gray")
> yel_edges <- subset(edges, color == "yellow")
> or_edges <- subset(edges, color == "orange")
> red_edges <- subset(edges, color == "red")
> edges <- rbind(gray_edges, yel_edges, or_edges, red_edges)
> EC4$edges <- edges
```

Now we're ready to plot!

Figures 16, 17, and 18 shows the hive plot of this network using methods `absolute`, `rank` and `norm` respectively. Each plot takes about 10 seconds to draw. Figure 19 is the same as Figure 17 but adds the nodes: red nodes are persistent meaning they are common to a group of about 200 bacterial species. When plotting with `method = "rank"` (as here)

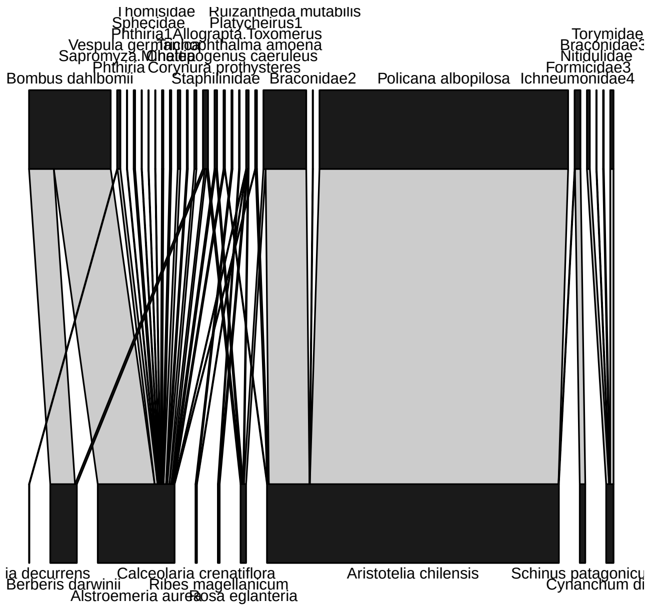

Figure 11: Safariland data set using plotweb

each gene gets a unique node (the other two method overlap nodes if more than one is present, and thus the last node plotted determines the color). With this many nodes, overplotting is a problem, so we shrank the node size and sorted the nodes so that the red nodes were drawn last (a strategy documented in more detail in an upcoming example). Another approach might be to expand the axis length, but that's probably not realistic: there are 1,274 nodes on this axis. Note that the manager axis nodes all appear to be persistent (red).

5 Further Explorations of the E. coli Network

In this section we'll demonstrate some slightly more advanced manipulations of the *E. coli* network data, including how one can make *hive panels* which are useful in comparing multiple hive plots. In some of the manipulations below, data types are coerced away from the definition found in a `HivePlotData` object and must be restored. It might be helpful to study the description of the required structure at `?HPD`.

First, we are going to re-code some of the information in the network. In the original publication, nodes were classified as either persistent or non-persistent. This classification was based upon comparison of the *E. coli* genome to roughly 200 other bacterial genomes. A gene was considered persistent if it was present in these other genomes, otherwise it is non-persistent and unique to *E. coli*. In our processing above, genes (nodes) that are persistent are red, while non-persistent nodes are black. We can use the existing axis assignments, based upon role as source, manager or sink, along with the persistence information, to display the network taking this information into account. In principle, there are six possible combinations: (persistent, non-persistent) x (source, manager, sink). However, it turns out that one of these combinations doesn't exist (persistent sources), so we'll re-code this information into a five axis hive plot.[4] Here's the first step, starting from where we left off above:

```
> EC5 <- EC4
```

[4]Not only am I not an ecologist, I am not a molecular biologist. I have no idea if this analysis is actually worthwhile, I just thought it would be interesting to see these relationships. Plus, it also permits further manipulations to be demonstrated.

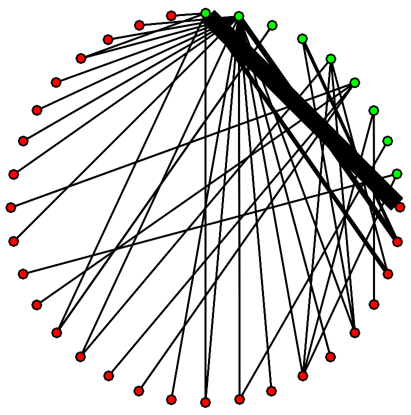

Figure 12: Safariland data set using gplot (mode = circle)

```
> nodes2 <- nodes <- EC5$nodes
> nn <- length(nodes$axis)
> #
> for (n in 1:nn) {
+         if ((nodes$axis[n] == 1) & (nodes$color[n] == "black")) nodes2$axis[n] <- 1
+         if ((nodes$axis[n] == 2) & (nodes$color[n] == "black")) nodes2$axis[n] <- 2
+         if ((nodes$axis[n] == 3) & (nodes$color[n] == "black")) nodes2$axis[n] <- 3
+         if ((nodes$axis[n] == 2) & (nodes$color[n] == "red")) nodes2$axis[n] <- 4
+         if ((nodes$axis[n] == 3) & (nodes$color[n] == "red")) nodes2$axis[n] <- 5
+         }
> #
> # Final assembly & checking...
> #
> nodes2$axis <- as.integer(nodes2$axis)
> EC5$nodes <- nodes2
> EC5$axis.cols <- rep("gray", 5) # we added 2 more axes!
> #
> sumHPD(EC5)
```

```
        E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010))
       This hive plot data set contains 1378 nodes on 5 axes and 687 edges.
       It is a  2D data set.

               Axis 1 has 64 nodes spanning radii from 1 to 214
               Axis 2 has 1172 nodes spanning radii from 1 to 9
               Axis 3 has 70 nodes spanning radii from 2 to 402
               Axis 4 has 71 nodes spanning radii from 1 to 7
```

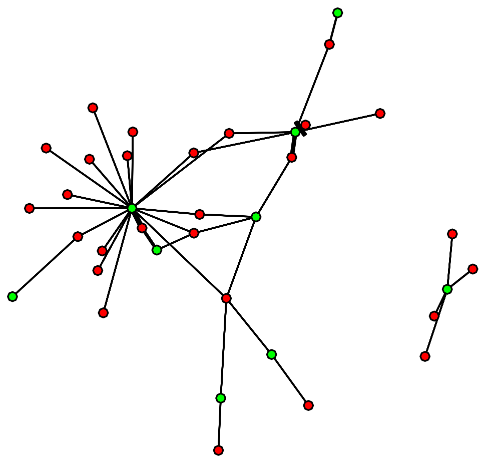

Figure 13: Safariland data set using gplot (mode = Fruchterman-Reingold)

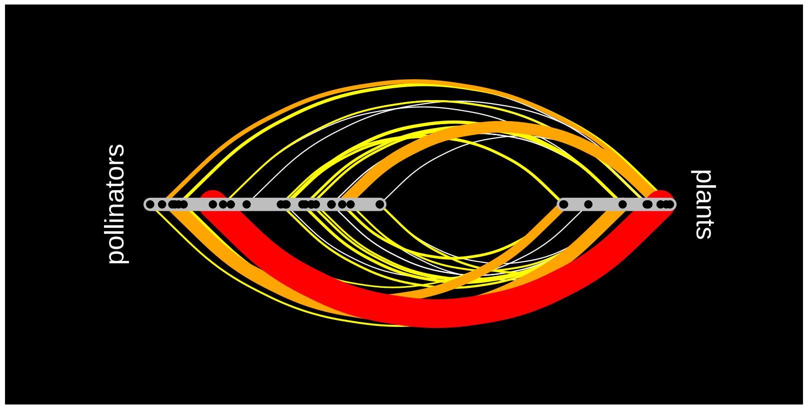

Figure 14: Safari data set using plotHive

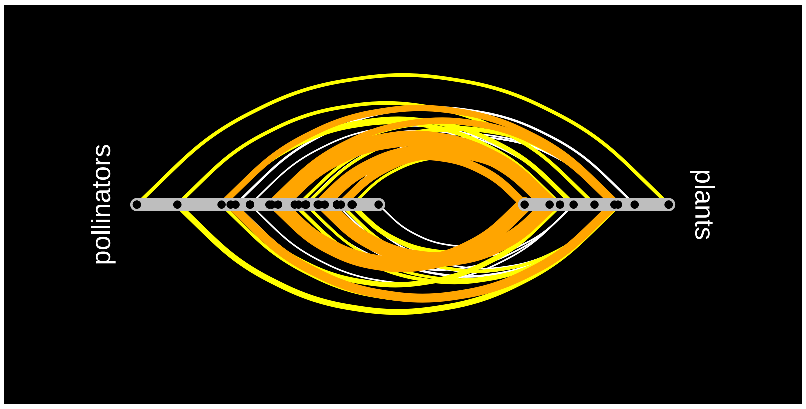

Figure 15: Arroyo data set using plotHive

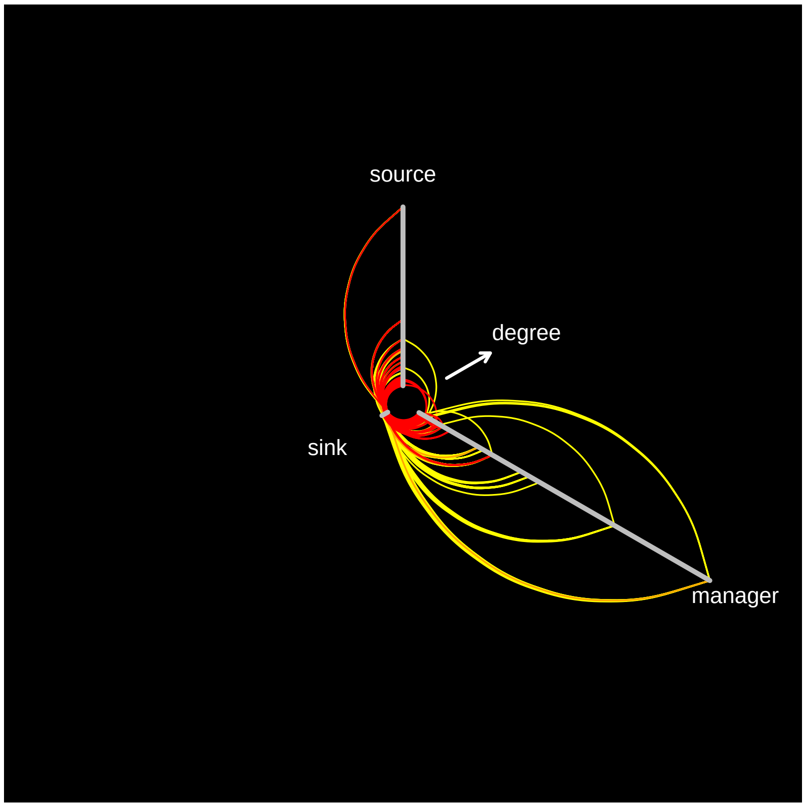

Figure 16: Hive plot of E. coli gene regulatory network (native node units)

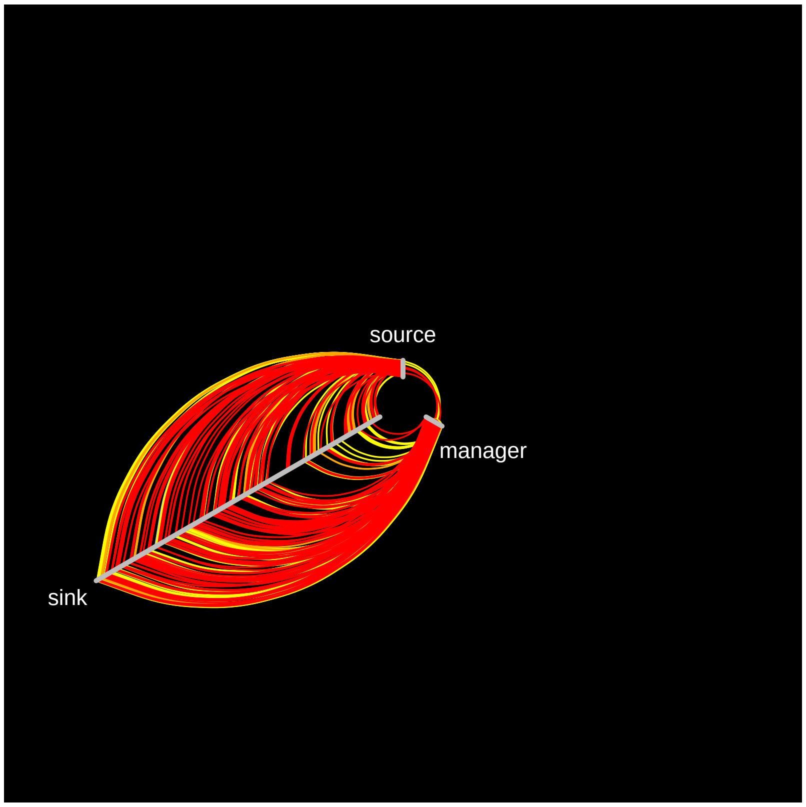

Figure 17: Hive plot of E. coli gene regulatory network (nodes ranked)

```
                Axis 5 has 1 nodes spanning radii from 10 to 10
```

```
> # sumHPD(EC5, chk.all = TRUE) # not run, the output is long
```

With sumHPD, one can use `chk.all = TRUE` which runs some additional checks on the data (see `?sumHPD` for full details). Had we done so in this case, we would find that some edges start and stop on the manager axis; perhaps you noticed this earlier. These are managers that call other managers. Also, somewhat miraculously, there are no edges crossing axes in this particular partitioning of nodes (`chk.all` also looks for this condition). In the basic summary that we did run, axis five has only one node on it. This will plot fine except for the case where one uses `method = "norm"` which will fail because to normalize the node radii, there has to be more than one radius value. To fix this, we'll add in a phantom, invisible node to anchor axis five as follows:

```
> EC6 <- EC5
> tmp <- data.frame(id = 1379, lab = "axis_5_anchor",
+         axis = 5, radius = 1, size = 1, color = "grey")
> EC6$nodes <- rbind(EC6$nodes, tmp)
> #
> # Clean up, re-size nodes, sort nodes so
> # persistent (red) ones are drawn last & check:
> #
> EC6$nodes$axis <- as.integer(EC6$nodes$axis)
> EC6$nodes$id <- as.integer(EC6$nodes$id)
> EC6$nodes$size <- EC6$nodes$size * 0.1
> #
> nodes <- EC6$nodes
> nodes <- sort_df(nodes, vars = "color")
> EC6$nodes <- nodes
> #
```

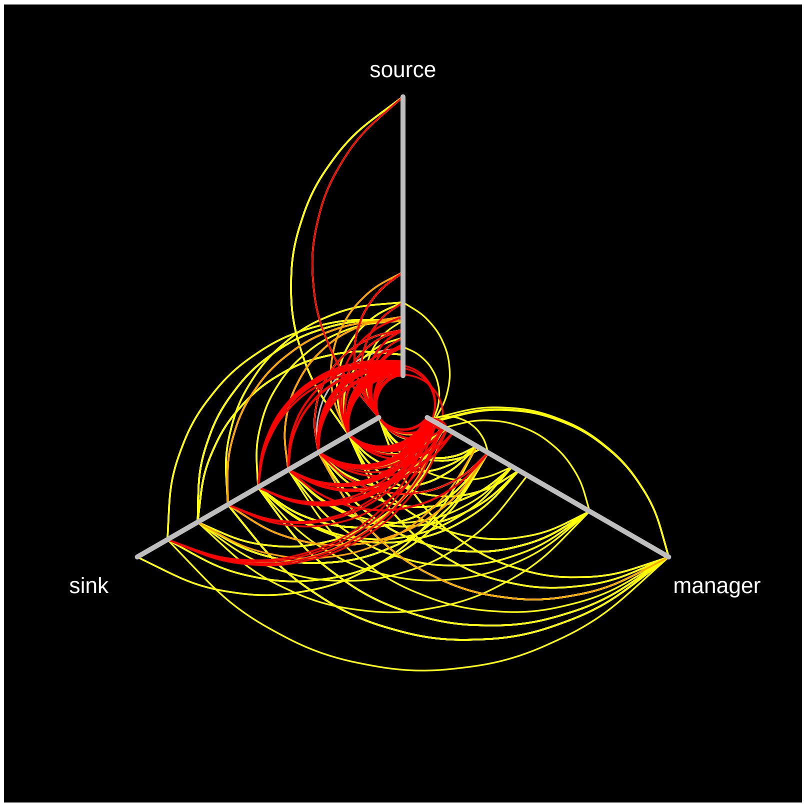

Figure 18: Hive plot of E. coli gene regulatory network (nodes normed)

```
> sumHPD(EC6)

         E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010))
        This hive plot data set contains 1379 nodes on 5 axes and 687 edges.
        It is a  2D data set.

                Axis 1 has 64 nodes spanning radii from 1 to 214
                Axis 2 has 1172 nodes spanning radii from 1 to 9
                Axis 3 has 70 nodes spanning radii from 2 to 402
                Axis 4 has 71 nodes spanning radii from 1 to 7
                Axis 5 has 2 nodes spanning radii from 1 to 10
```

Next, we are going to copy the current version of the network (EC6) and scale the axes of the copy, because the summary above shows that the axis lengths are quite different and the shorter axes will be nearly invisible if we don't scale them up at least a bit.

```
> EC7 <- manipAxis(EC6, method = "scale", action = c(1, 10, 1, 10, 10))
> sumHPD(EC6)

         E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010))
        This hive plot data set contains 1379 nodes on 5 axes and 687 edges.
        It is a  2D data set.

                Axis 1 has 64 nodes spanning radii from 1 to 214
                Axis 2 has 1172 nodes spanning radii from 1 to 9
                Axis 3 has 70 nodes spanning radii from 2 to 402
                Axis 4 has 71 nodes spanning radii from 1 to 7
                Axis 5 has 2 nodes spanning radii from 1 to 10
```

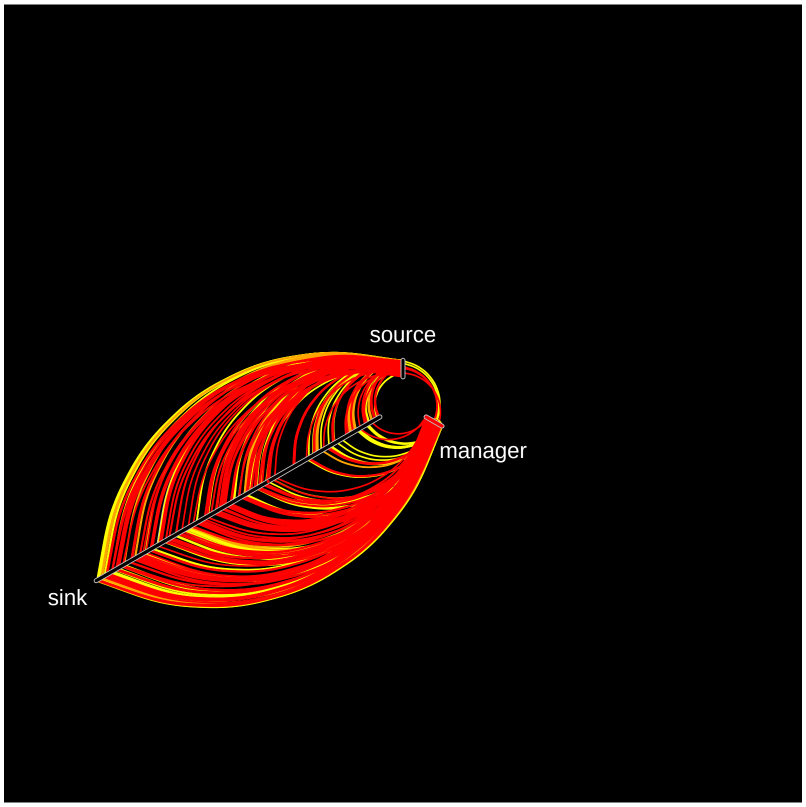

Figure 19: Hive plot of E. coli gene regulatory network (nodes ranked)

Now we'll make a hive panel showing this same network displayed using different methods. The code follows; it uses the `grid` graphics systems and associated viewport concepts to create a 2 x 2 hive panel. The resulting hive panel is Figure 20.

Finally, the lower right hive plot in Figure 20 can serve as a starting point for teasing out even more information. Instead of drawing all the edges in one hive plot, we'll make a hive panel showing each edge category in a different panel. The steps are given below; the resulting panel is Figure 21 (the steps to produce the panel are not shown here, but are the same as before).

```
> EC11 <- EC10 <- EC9 <- EC8 <- EC7
> edges <- EC7$edges
> gray_edges <- subset(edges, color == "gray")
> yel_edges <- subset(edges, color == "yellow")
> or_edges <- subset(edges, color == "orange")
> red_edges <- subset(edges, color == "red")
> EC8$edges <- gray_edges
> EC9$edges <- yel_edges
> EC10$edges <- or_edges
> EC11$edges <- red_edges
```

6 Comparison to linnet

linnet (for linear networks) is the Perl program written by Krzywinski that draws hive plots. Here are some notes about how `HiveR` compares to linnet.

1. To show more information, in linnet one can clone an axis to specifically show connections that would start and end on the same axis (if it isn't cloned). Cloned axes appear a bit on either side of where the original axis would

```
> vplayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
> #
> grid.newpage()
> pushViewport(viewport(layout = grid.layout(2, 2)))
> #
> pushViewport(vplayout(1, 1)) # upper left plot
> plotHive(EC6, ch = 20, np = FALSE)
> popViewport(2)
> #
> pushViewport(vplayout(1, 2)) # upper right plot
> plotHive(EC7, ch = 0.1, method = "norm", np = FALSE,
+         axLabs = c("non-persistent\nsource", "non-persistent\nsink",
+         "non-persistent\nmanager", "persistent\nmanager", "persistent \nsink"),
+         axLab.pos = rep(0.2, 5), axLab.gpar = gpar(fontsize = 10, col = "white"),
+         rot = c(0, 72, 0, 0, -72), anNode.gpar = gpar(fontsize = 10, col = "pink", lwd = 0.5),
+         anNodes = system.file("extdata", "E_coli", "NodeLabels.csv", package = "HiveR"))
> popViewport(2)
> #
> pushViewport(vplayout(2,1)) # lower left plot
> plotHive(EC7, ch = 100, method = "rank", np = FALSE)
> popViewport(2)
> #
> pushViewport(vplayout(2,2)) # lower right plot
> plotHive(EC7, ch = 20, np = FALSE)
```

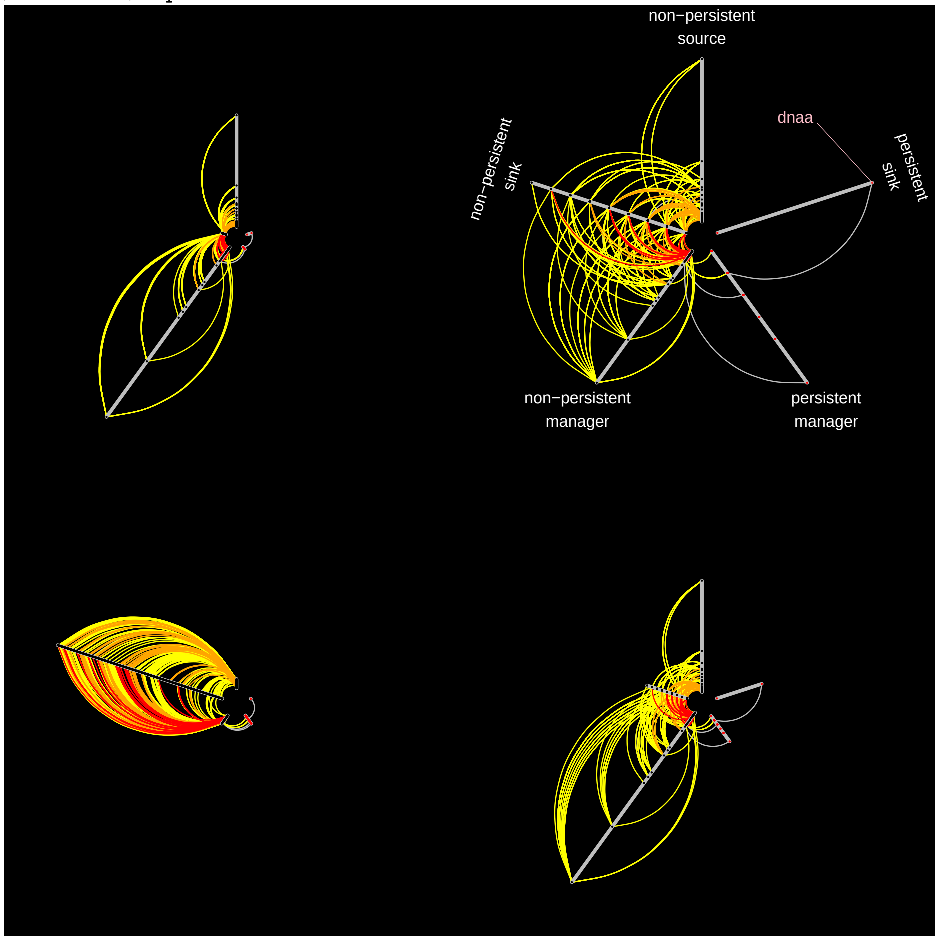

Figure 20: Hive panel showing E. coli regulatory network with different display options

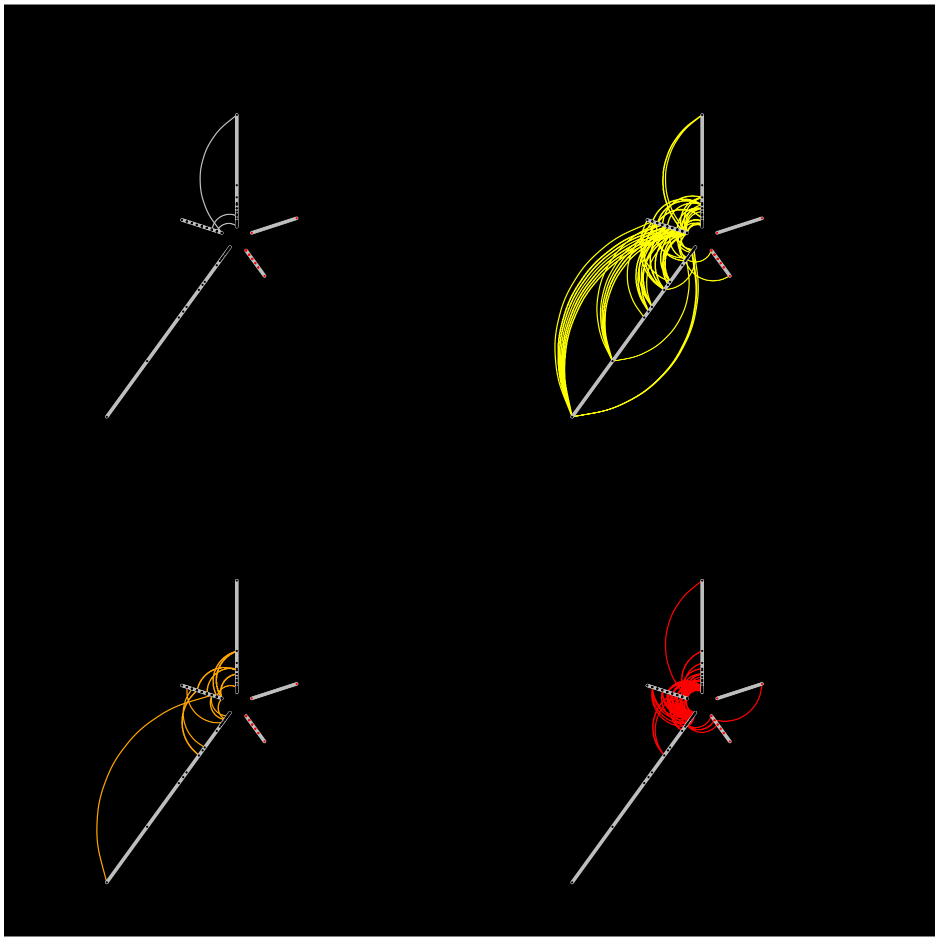

Figure 21: Hive panel showing E. coli regulatory network with edges encoded by genetic distance (Red edges are the closest; each set of edges plotted separately)

have been. In `HiveR`, the same notion can be implemented, but rather than clone an existing axis, one can simply add a new axis based upon some property of the system. Alternatively, for 2D hive plots, `HiveR` is able to show edges that start and end on the same axis (linnet does not do this).

2. No segmentation of an axis is currently possible with `HiveR`.
3. linnet uses bezier curves to create the edges; `HiveR` uses splines with a single control point.

7 Features Planned and Under Consideration

1. Add the ability to subtract 2 hive plots and display the result.
2. Set up animations for the 3D mode. Perhaps include the possibility of running two animations of related hives side by side.
3. Set up a mechanism to automatically permute the axes in 3D mode when nx = 5 or 6 so that the best option can be selected. Might also be worth doing in 2D mode for 4-6 axes, except in this case it's not a question of how you display but how you import the data. Wegman[5] has a formula describing all possible combinations that would be needed.
4. Set up mouse controls in 3D mode.
5. Smallish items
   (a) The current 3D spline calculation produces an asymmetric spline. It could be made symmetric.
   (b) The current splines could be converted to Bezier curves.
   (c) Could add line type as an edge parameter. This might be simple, or not.

8 Acknowledgements

Naturally, I thank Martin Krzywinski for numerous helpful communications. I also appreciate helpful discussions on gene ontology concepts with my colleague Professor Chet Fornari.

References

[1] M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra, "Hive plots – rational approach to visualizing networks," *Briefings in Bioinformatics*, 2011.

[2] M. Suderman and M. Hallett, "Tools for visually exploring biological networks," *Bioinformatics*, vol. 23, pp. 2651–2659, Oct 15 2007.

[3] K.-K. Yan, G. Fang, N. Bhardwaj, R. P. Alexander, and M. Gerstein, "Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, pp. 9186–9191, May 18 2010.

[4] S. Gama-Castro, H. Salgado, M. Peralta-Gil, A. Santos-Zavaleta, L. Muniz-Rascado, H. Solano-Lira, V. Jimenez-Jacinto, V. Weiss, J. S. Garcia-Sotelo, A. Lopez-Fuentes, L. Porron-Sotelo, S. Alquicira-Hernandez, A. Medina-Rivera, I. Martinez-Flores, K. Alquicira-Hernandez, R. Martinez-Adame, C. Bonavides-Martinez, J. Miranda-Rios, A. M. Huerta, A. Mendoza-Vargas, L. Collado-Torres, B. Taboada, L. Vega-Alvarado, M. Olvera, L. Olvera, R. Grande, E. Morett, and J. Collado-Vides, "RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units)," *Nucleic Acid Research*, vol. 39, pp. D98–D105, January 2011.

[5] E. J. Wegman, "Hyperdimensional data-analysis using parallel coordinates," *Journal of the American Statistical Association*, vol. 85, pp. 664–675, Sep 1990.

[6] C. T. Butts, "Social network analysis with sna," *Journal of Statistical Software*, vol. 24, pp. 1–51, 5 2008.

[7] D. P. Vazquez and D. Simberloff, “Changes in interaction biodiversity induced by an introduced ungulate,” *Ecology Letters*, vol. 6, pp. 1077–1083, 2003.