# The HiveR Package (v 0.1-4)

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

November 17, 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. Many more features remain to be added; see the sections below. There are certainly 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.

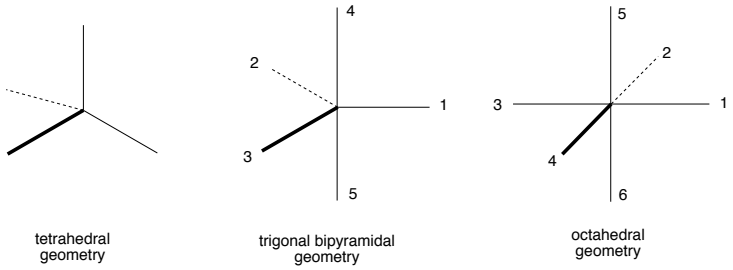
The key innovation in a hive plot, compared to other means of graphically displaying network structure, is in how node information is handled. Nodes are assigned to axes based upon qualitative or quantitative characteristics of the the node, for instance membership in a certain category, and the position 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 in press detailing the features and virtues of hive plots and is a must-read. 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.[1]

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 1

| 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**




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**

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 a step 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.

# 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 2.1.

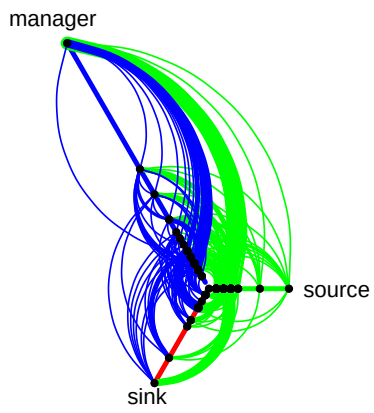



**Figure 2: FuncMap for package lattice**

| *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 |
| $center.hole | | num | Size of center hole |
| - attr | | chr "HivePlotData" | The S3 class designation |

**Table 2: The structure of a HivePlotData object**

| *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 |

**Table 3: Comparison of plotting nodes native, rank and norm**

## 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.*[2] but is based upon data in the RegulonDB.[3] The version here was extended by Krzywinski 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 two 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). The other 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.

## 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), or normalized to run between 0. . . 1, or plotted by rank. Some aspects of the plot that depend upon these options are shown in Table 2.6. 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.

### 2.6.1 2D Mode Hive Plots

Figures 2.6.1 shows a 2 axis hive plot using randomly generated data and the function `plotHive`. Figure 2.6.1 show a hive plot of a random 3 axis network using absolute scaling; Figure 2.6.1 shows the 3 axis example with the nodes

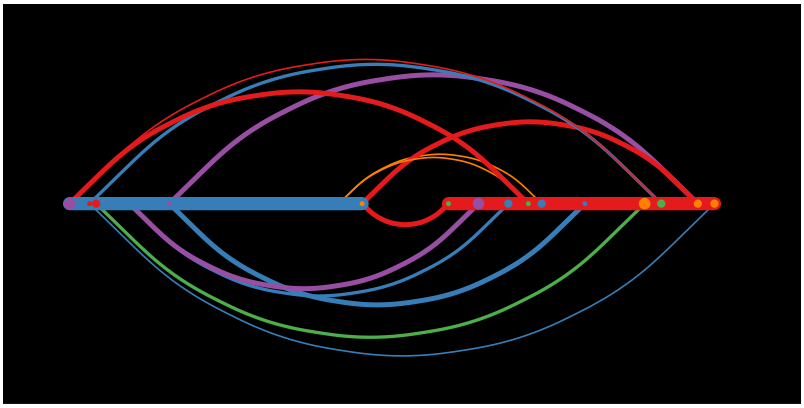


**Figure 3: A randomly generated hive plot with 2 axes (native units)**

displayed by rank and Figure 2.6.1 the same data normed. FIgure 2.6.1 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.

### 2.6.2 3D Mode 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`).

## 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[4] 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. 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).
4. Some ideas about network parameters that might be mapped to hive plot parameters (see Table 1):
   (a) Ecology: see various species descriptors computed by function `specieslevel` in package `bipartite`.

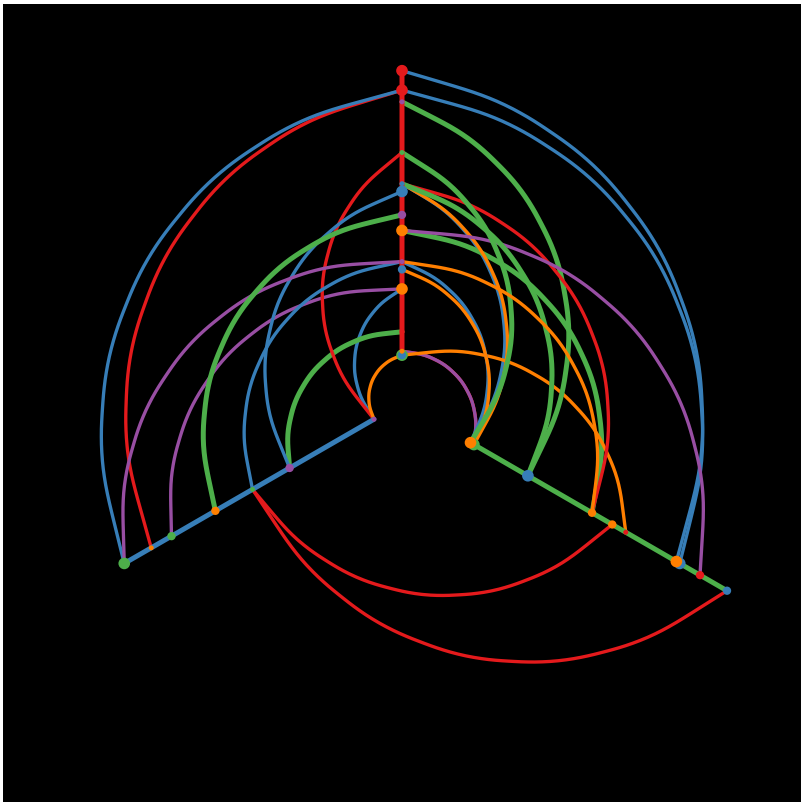


**Figure 4: A randomly generated hive plot with 3 axes (native units)**

(b) Social networks: see the section "Node-level indices" in the article describing package `sna`. Briefly, degree, betweeness and closeness are the key ideas.[5]

# 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 [6]. 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 3 and 3 show two means of plotting `Safari` using package `bipartite`.[2] Figure 3 is a simple table 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 3 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.

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 3 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 3 shows the same data using a different layout

[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`.

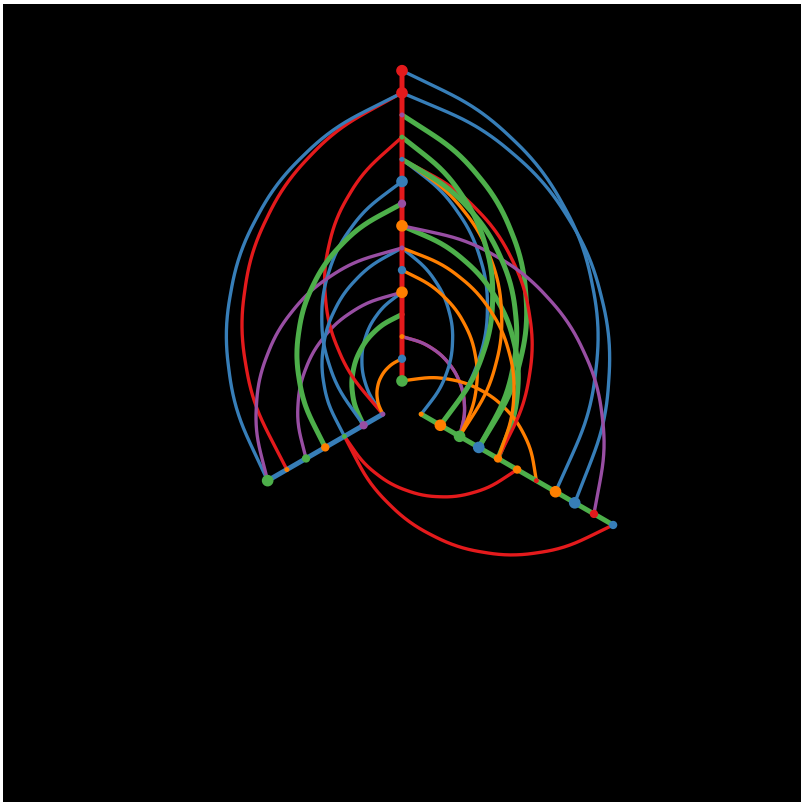


**Figure 5: A randomly generated hive plot with 3 axes (nodes by rank)**

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 3 and 3 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 (plants are on the right). 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 (right) 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.

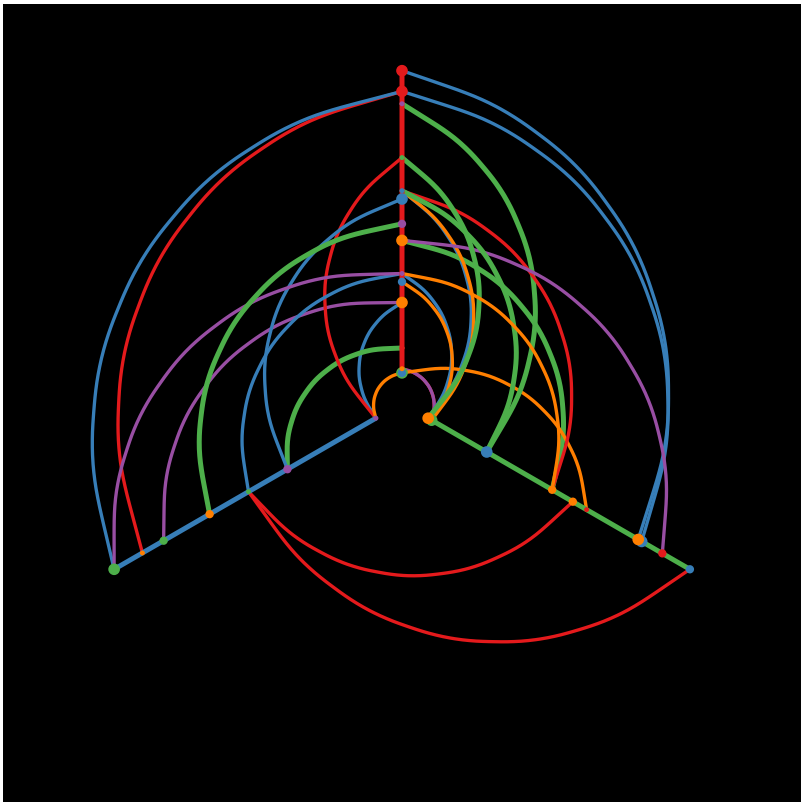


**Figure 6: A randomly generated hive plot with 3 axes (nodes normed)**

## 4 Demonstration of the E. coli Gene Regulatory Network

HiveR includes the *E. coli* gene regulatory network, discussed in Yan *et. al.*[2] and based upon the RegulonDB[3] 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 yellow $\rightarrow$ orange $\rightarrow$ red with red being the most related pairs. Gene pairs that are scattered around the physical genome are colored gray.

| dot.tag | dot.val | hive.tag | hive.val |
|---|---|---|---|
| label | persistent | color | red |
| label | nonpersistent | color | grey |

**Table 4: Contents of NodeInst.csv**

```
> # The call below is complicated by the needs of building this vignette
> EC1 <- dot2HPD(file = system.file("extdata", "E_coli", "ecoli.dot", package = "HiveR"),
+         node.inst = system.file("extdata", "E_coli", "NodeInst.csv", package = "HiveR"),
+         edge.inst = system.file( "extdata", "E_coli", "EdgeInst.csv", package = "HiveR"),
+         desc = "E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010)) ",
+         axis.cols = rep("grey", 3))
> # A more typical version is commented out below.
> #EC1 <- dot2HPD(file = "ecoli.dot,
> #        node.inst = "NodeInst.csv",
```

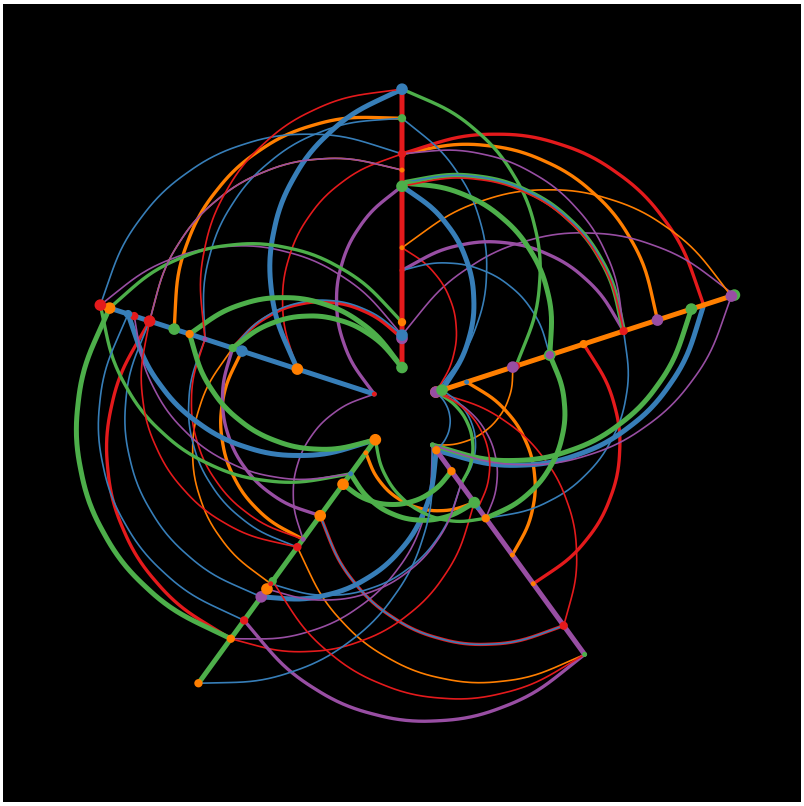


**Figure 7: A randomly generated hive plot with 5 axes (native units)**

| 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**

```
> #         edge.inst = "EdgeInst.csv",
> #         desc = "E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010)) ",
> #         axis.cols = rep("grey", 3))
> sumHPD(EC1)

        E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010))
       This hive plot data set contains 1378 nodes on 1 axes and 2966 edges.
       It is a  2D data set.

> # assign node radius based upon edge degree:
> EC2 <- mineHPD(EC1, option = "rad <- tot.edge.count")
> sumHPD(EC2)

        E coli gene regulatory network (Yan et al PNAS vol 107 pg 9186 (2010))
       This hive plot data set contains 1378 nodes on 1 axes and 2966 edges.
       It is a  2D data set.

> # assign node axis based upon role as source, manager or sink:
> EC3 <- mineHPD(EC2, option = "axis <- source.man.sink")
> sumHPD(EC3)
```

```
------------------------------------------------------------
This is bipartite 1.17.
For latest additions type: ?bipartite.
For citation please type: citation("bipartite").
Have a nice time plotting and analysing two-mode networks.
------------------------------------------------------------
```

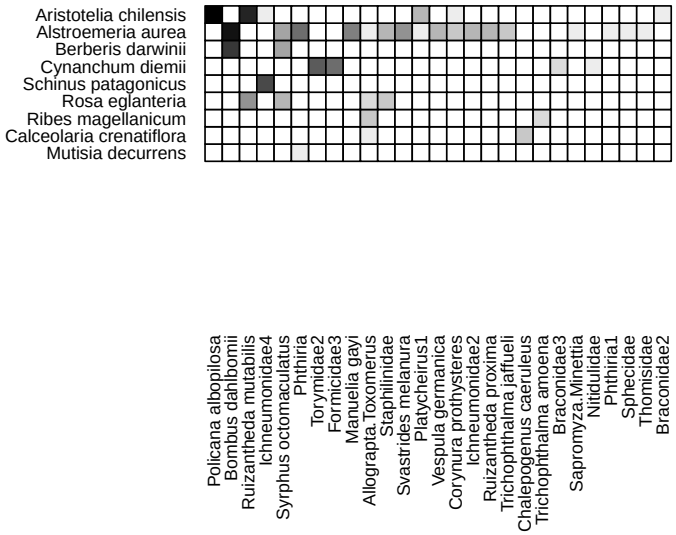



**Figure 8: Safariland data set using visweb**

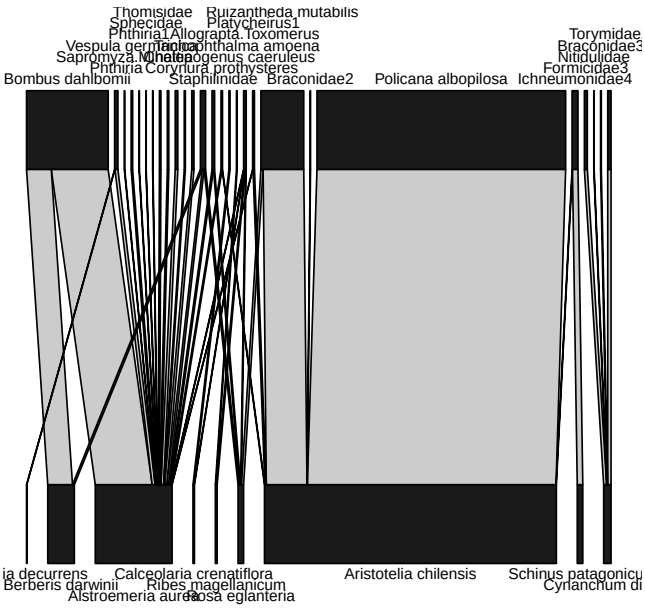



**Figure 9: Safariland data set using plotweb**

```
 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.
```

Figures 4, 4, and 4 shows the hive plot of this network using methods `absolute`, `rank` and `norm` respectively. Each plot takes about 30 seconds to draw. Source only nodes are at the top, managers at lower left, and sinks at lower right. One can clearly see based upon the color of the nodes that "related' genes are rare. Figure 4 shows the same plot but adds the nodes: red nodes are persistant. This figure is plotted with `mode = "rank"` so that each gene gets a unique node (the other two modes 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. One solution would be to write an option for `mineHPD` which makes the persistent (red) nodes plotted last. Another would be to expand the axis length, but that's probably not realistic: there are 1,274 nodes on this axis.

# 5 Comparison to linnet

linnet (for linear networks) is the 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). In `HiveR`, the same notion exists but rather than clone an existing axis, one can simply add a new axis based upon some property of the system. Or, for 2D hive plots, `HiveR` is able to show edges that start and end on the same axis.
2. No segmentation of an axis is currently possible with `HiveR`.
3. As mentioned above, for 2D hive plots `HiveR` is capable of drawing edges that start and end on the same axis. linnet does not do this.

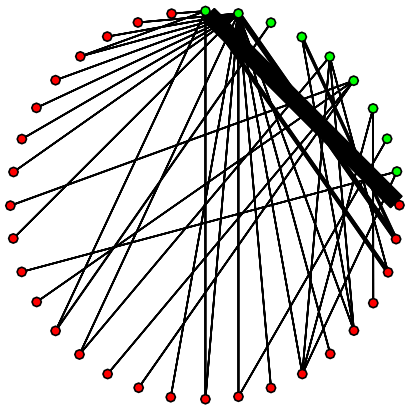


**Figure 10: Safariland data set using gplot (mode = circle)**

# 6 Things to Do

1. Add magnification of an axis. Implement in manipAxis.
2. Add inversion of an axis. Implement in manipAxis.
3. Tasks related to importing real data
   (a) Find a medium sized real data set.
   (b) Experiment with mapping more complex networks such as food webs.
4. Work on speed: for 3D mode hive plots, drawing even a modest number of edges can be prohibitively slow. Testing has shown that the problem is not `rcsr`, so need to look other places when profiling the code.

# 7 Features to Add Long Term

1. Hive Panels: set up a GUI or other display mechanism which can display multiple views of the same hive from different perspectives, or different hives from the same perspective. If making 2D hive plots, this should be pretty trivial using `grid` graphics. Krzywinski calls these *hive panels*.
2. The current 3D spline calculation produces an asymmetric spline. It could be made symmetric.
3. Could add line type as an edge parameter. This might be simple, or not.
4. Some means of optionally labeling axes might be useful.
5. Could add log as a means of scaling values along an axis, though users could readily do this on the fly.
6. Add the ability to subtract 2 hive plots and display the result.

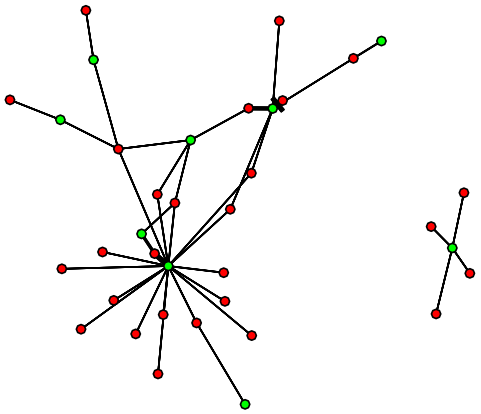


**Figure 11: Safariland data set using gplot (mode = Fruchterman-Reingold)**

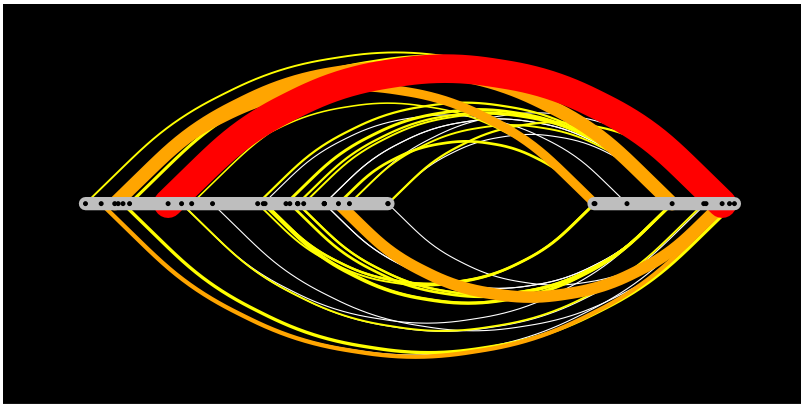


**Figure 12: Safari data set using plotHive**

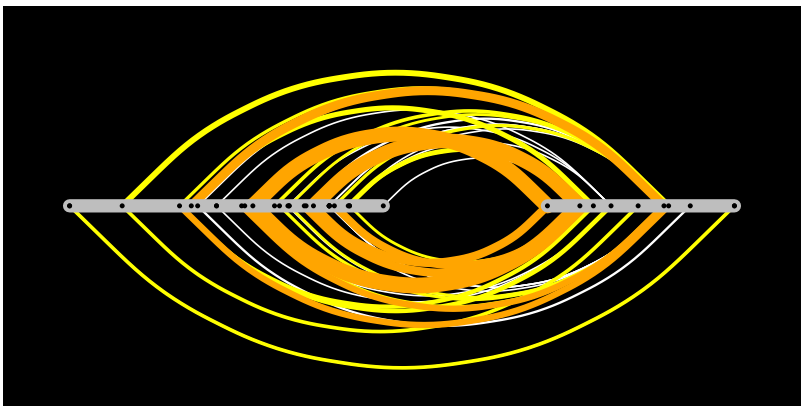


**Figure 13: Arroyo data set using plotHive**

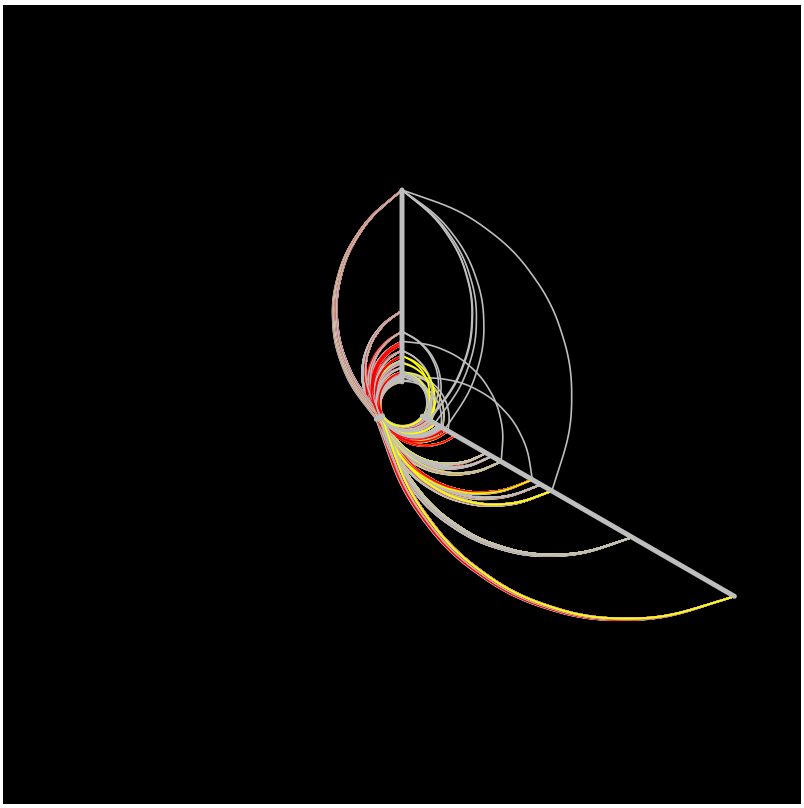


**Figure 14: Hive plot of E. coli gene regulatory network (native node units)**

**Figure 15: Hive plot of E. coli gene regulatory network (nodes ranked)**

7. Set up animations for the 3D mode. Perhaps include the possibility of running two animations of related hives side by side.
8. 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[4] has a formula describing all possible combinations that would be needed.
9. Set up mouse controls in 3D mode.

## References


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

[2] 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.

[3] 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, JAN 2011.

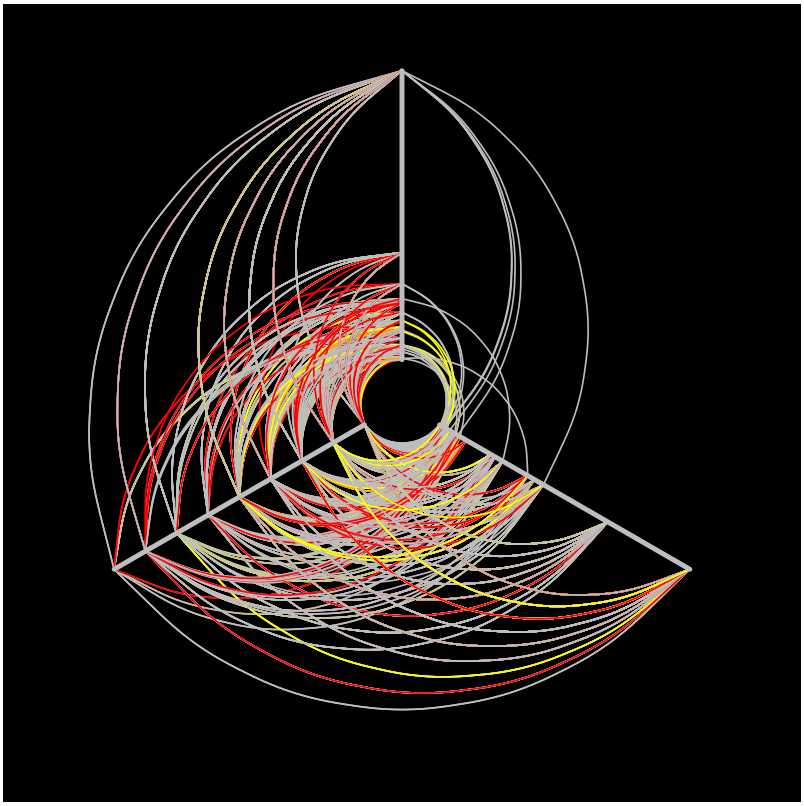


**Figure 16: Hive plot of E. coli gene regulatory network (nodes normed)**


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

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

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

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