Network Visualizations in `ggplot2`

Sam Tyner and François Briatte and Heike Hofmann

There are many implementations of static network visualization in R, but none are equipped with the flexibility and functionality of `ggplot2`. `geom_net` was created to fill this gap. Using two data frames to describe vertex and edge information, `geom_net` makes use of the underlying structure of `ggplot2` to visualize networks. This makes it possible to easily facet networks according to covariates or change aesthetics such as shape, size, or color according to additional edge or vertex information.

todo list:

1. change format to comply with journal
2. include installation instruction for package ggnet
3. the biggest change to the previous implementation is that there is no longer a vertices data set. Instead, all of the vertex information has to be included in the edge dataset. The assumption is that any vertex information included relates to the 'from_id' column. If the user does not include ALL vertex information in the dataset (i.e. if some vertices only show up under the 'to_id' column), there will be segment ends without a dot in the graphic. As a measure against that, we suggest to merge edges and vertex information along the 'from_id' column using the parameter setting 'all = TRUE'. This ensures all vertices to be included; if a vertex is a sink, the corresponding 'to_id' value is set to NA. This is accommodated for both in the layout and the drawing of the graph.
4. the parameters have changed names: all of the vertex parameters have the regular names, edge parameters start with 'e'. 'esize' is re-named to 'linewidth'. Some parameters might not be implemented yet, we need to get a good overview of what is missing and what works.
5. we could draw selfies, but at the moment we don't. We need to discuss this.

Contents

1 Introduction **3**

2 Two approaches of network visualizations **5**
2.1 ggnet . . . 5
2.2 geom_net . . . 6
2.2.1 Data Structure . . . 6
2.2.2 Parameters and Aesthetics . . . 6
2.2.3 Vertex Aesthetics . . . 7
2.2.4 Edge Aesthetics . . . 7
2.2.5 Other Arguments . . . 8

3 Examples of Networks **10**
3.1 Blood Donation . . . 10
3.2 Email Network . . . 10
3.3 `ggplot2` Theme Elements . . . 14
3.4 Mad Men Networks . . . 16
3.5 College Football . . . 18
3.6 Les Misérables . . . 19
3.7 Bikesharing in D.C. . . . 21
3.8 Protein Interaction Network in Yeast . . . 23

4 Technical Details **23**
4.1 Creating `stat_net` . . . 23
4.2 `stat_net` Usage . . . 25
4.3 Creating `geom_net` . . . 25
4.4 `geom_net` Usage . . . 26

5 Future Work **27**

1. INTRODUCTION

At its core, a network is simply a set of points connected in pairs by a set of lines (Newman, 2010). Here, we refer to the lines as edges and the points as vertices, although these are also called nodes. These two seemingly simple sets of graphical objects, points and segments, are used to encode a huge variety and quantity of information across many fields of study. For instance, networks of scientific collaboration, a food web of marine animals, and American college football games are all covered in a paper on community detection in networks by Girvan and Newman (2002). Buldyrev et al. (2010) examine node failure in interdependent networks like power grids. Social networks, such as links between actors found on `www.imdb.com`, and neural networks, like the completely mapped neural network of the *C. elegans* worm are also etensively studied (Watts and Strogatz, 1998). Networks vary widely in scope and complexity: the smallest network is simply an edge between two vertices, while one of the most commonly used and most complex networks, the world wide web, has billions of vertices (webpages) and billions of edges (hyperlinks) connecting them. The edges in a network can be directed or undirected: directed edges represent information travelling from one vertex to another, and switching the direction would change the structure of the network. The world wide web is an example of a directed network because one webpage may link to another, but not necessarily the other way around. Undirected edges are simply connections between vertices. In co-authorship networks nodes are authors connected by an edge, if they author an academic publication together. Co-authorship networks are examples of undirected networks because if two people author a paper together, it creates a connection between them that is bidirectional.

A social network is a network that everyone is a part of in one way or another. We do not necessarily refer here to social media like Facebook or LinkedIn, but rather to the connections we form with other people. To demonstrate the functionality of our geometry for plotting networks, we have chosen an example of a social network from the popular television show Mad Men. This network, compiled by Chang (2013), is made up of 52 vertices and 87 edges. Each vertex represents a character on the show, and there is an edge between every two characters who have had a romantic relationship.

Figure 1 shows this network. In the plot, we can see one central character who has many more relationships than any other character. This vertex represents the main character of the show, Don Draper, who is quite the "ladies' man." This example shows just how ubiquitous networks are.

include at least one citation for each one of the examples:

There are many kinds of networks, and networks are extensively studied across many disciplines. Many sociologists study social networks, and many biologists study protein networks. As different as these and the many other disciplines that study networks are, they all need the ability to quickly and effectively visualize networks. quickly and effectively are a bit vague as measures for why we need a geom implementation. We don't want to do any studies measuring time to result or define effectiveness. It might be better to give

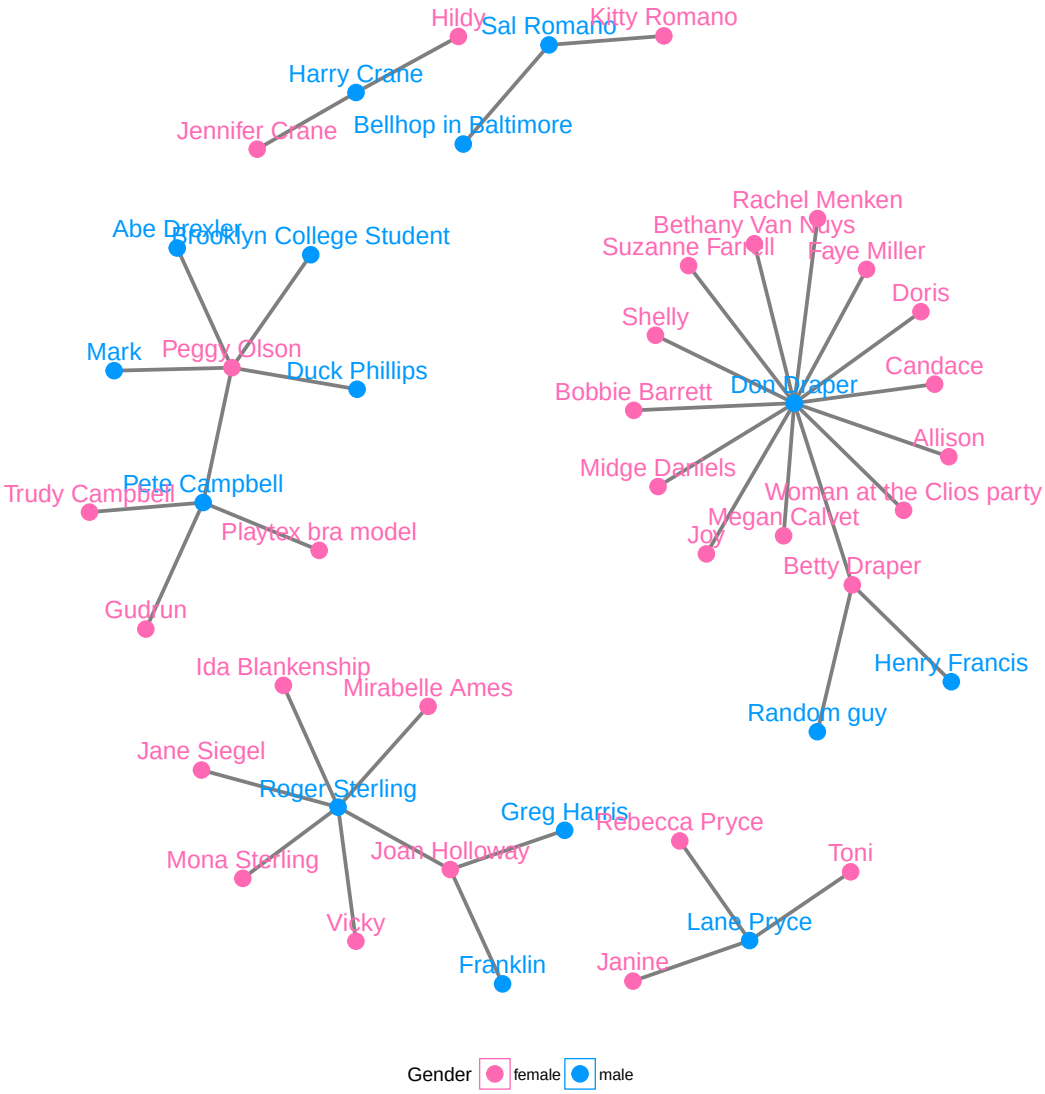


Figure 1: Graph of the characters in the show Mad Men who are linked by a romantic relationship.

some concrete examples of packages doing network layouts and discuss the problematic that there is no standard way of bundling up the results, which makes working with them hard, and even harder to modify output to include additional information for visualizations.

Coloring the vertices or edges in a graph is a quick and easy way to visualize grouping and can help with pattern or cluster detection. The vertices in a network and the edges between them compose the structure of a network, and being able to discover patterns among them visually is a key part of network analysis. Viewing multiple layouts of the same network can also help reveal patterns or clusters that would not be discovered when only viewing one layout or analyzing only an adjacency matrix.

Many `R` packages already exist for network analysis and visualization such as `igraph` by Csardi and Nepusz (2006), `sna` by Butts (2014), and `network` by Butts (2008); Butts et al. (2014) but we have found these packages to have unintuitive or burdensome you're shooting sharp – it might be better to describe that our approach is more customizable and more intuitive to use methods for customizing the colors, sizes, etc of the vertices and edges of the network. For instance, the `igraph` package allows for coloring vertices by groups but the user must assign the colors to each vertex individually as opposed to assigning color by a grouping factor variable.

just moving your words around

We found the current tools to be lacking in this ability, so we chose to fill this gap by adding network plotting capabilities to the popular and widely used `R` package `ggplot2`. Just to give an idea of the popularity and the wide-spread use of `ggplot2`, from January 1, 2015 to March 21, 2015, `ggplot2` was downloaded over 270,000 times, or approximately 3,454 downloads per day. It has also been downloaded in 189 countries at least once, and in 31 of those countries, including China, Israel, and Colombia, it has been downloaded over 1,000 times[1]. This is the user base we are aiming at by making network visualizations a part of `ggplot2`.

There are two main approaches to making use of the ggplot2 framework: (i) implement network visualizations using ggplot2, i.e. providing a wrapper for the user to visualize a network with ggplot2 elements, and (ii) implement networks as an internal layer of ggplot2. We will discuss both of these approaches in this paper.

XXX Provide roadmap for the rest of the paper.

2. TWO APPROACHES OF NETWORK VISUALIZATIONS

The three necessary elements of any network visualization are the vertices, the edges, and the layout.

2.1 ggnet

XXX needs massive changes :)

[1] `ggplot2` usage statistics taken from `http://cran-logs.rstudio.com/`.

The GGally package by Schloerke et al. (2014) contains a very useful function written by François Briatte and Moritz Marbach called ggnet that does allow for fairly straightforward customization of these three necessary graph attributes. The ggnet function, however, requires the graph input to be a network object according to the network package. Our work builds off of the ggnet function and presents an intuitive geom for network visualization within the ggplot2 framework, without the need for objects other than simple data frames. XXXX

2.2 geom_net

2.2.1 Data Structure

Network analysis is usally working with two sources of information: one data set consisting of a description of the 'players', represented as the nodes or vertices in the network, and another data set detailing the relationship between the players, the edge dataset. In order for this geometry to work, these two data sets need to be combined into a single dataset. For this, we are using the convention that all of the node information is merged into the edge data set using the 'from' as the reference column. Generally, there will be some vertices that are sinks in the network, ie. they only show up in the 'to' column. We can easily accommodate for these nodes by adding artificial edges in the data set, that have missing information for the 'to' column. Fortunately, R provides functions that allow for an easy way of producing the required result: both `merge` and `join` can be used. In `merge` the paramter all needs to be set to TRUE, in `join`, the parameter setting has to be `type='full'`, and `all=TRUE`.

The formal requirement of `stat_net` are two columns, called `from_id` and `to_id`. During this routine, columns `x, y` and `xend, yend` are calculated and used as a required input for `geom_net`.

Other variables may also be included for each edge, such as the edge weight or grouping variable.

2.2.2 Parameters and Aesthetics

Parameters currently implemented:

- layout: layout, layout.par, fiteach
- nodes/vertices: colour, size, alpha, shape
- edges: ecolour, ealpha, linewidth, stroke
- arrow: arrowsize, arrowgap (only if `directed = TRUE`)
- labels: label (as parameter and variable), labelcolour

2.2.3 Vertex Aesthetics

In our geometry, we want to create all possible vertex aesthetic to mimic the usual aesthetics in `ggplot2` for points. The `vcolor` aesthetic can be an identity color (e.g. `color = "red"`) to change the color of all vertices, or, in the final version, it will take a factor variable which identifies each vertex as belonging to one of several groups, and it colors the vertices of the graph according to this grouping. In the current version, the `vcolor` aesthetic does not work like a typical color aesthetic in `ggplot2`. It can, however, take HEX color characters and assign them to different levels of a factor variable. We use the `RColorBrewer` package to choose the colors for us in this paper (Neuwirth, 2014). It will also color the vertices along a color gradient according to the different values of some numeric variable such as degree. The vertex color defaults to black.

The `vsize` aesthetic can also be set to an identity value to increase or decrease the size of all vertices in the graph, or it can be set to one of any numerical columns in the vertex data frame. Right now, the numerical values are strict point values: if the size of a vertex is two, then it will be two points in diameter. A point is the size of the default fontsize in the `grid` package. In our final version, the `vsize` will scale to minimum and maximum values as all other size aesthestics in `ggplot2` do. The `vshape` aesthetic is for changing the shape of the vertices from the default circle to other shapes, like square or triangle. It also changes the shape of the vertices according to some grouping variable, which is currently required to consist of integer values from 0 to 25. In the final version, it will be able to change shape based on different levels of a factor variable. Finally, the `valpha` aesthetic will change the vertex transparency to a set value, like 1/10 or 0.5. It operates in exactly the same way as the `alpha` aesthetic in `geom_point`. This aesthetic is very useful for networks with hundreds or thousands of vertices, which can easily crowd the static visualization. Making the vertices more transparent will better show the underlying structure in the network.

Evidently, we created all of these aesthetics to fit in exactly to the `ggplot2` grammar of graphics. We also created the edge aesthetics in a similar fashion.

2.2.4 Edge Aesthetics

Again, our edge aesthetics mimic the familiar `ggplot2` aesthetics, this time for segments. The `ecolor` aesthetic changes the color of all of the edges according to the identity function or can change the color according to HEX colors assigned to levels of a grouping variable. In the final version, the `ecolor` aesthetic will automatically color according to levels of a grouping variable or according to a gradient scale associated with a numeric variable. The `elinetype` aesthetic can also be used with a grouping variable to change the line type of each edge in the graph from solid to one of the different types of lines in `ggplot2`. Currently, the column associated with this aesthetic must be integer valued and take on values from zero to six. In the final version, it will change the edge linetypes according to levels of a grouping variable. The `esize` aesthetic can also be set to the weight

or probability of each edge, or it can be changed for all edges with the identity function. The associated column can take on any numerical values, and currently maps the values directly to points, just like the `vsize` aesthetic. In the final version, this aesthetic will work exactly like the sizing in `ggplot2`, by scaling sizes to the minimum and maximum values. This is one of two ways to visualize the number of edges between two vertices. The other way is through the `ealpha` aesthetic. The `ealpha` aesthetic can change the edge transparency to a set value on the interval $[0, 1]$. If there are multiple edge connections given between two vertices in the edge data frame, the `ealpha` aesthetic can be changed to plot opaque lines between vertices with many mutual connections and varying degrees of transparent lines between vertices with only a few or a single connection. This property is also especially useful for large networks with a lot of connections or with vertices of relatively high degree. Finally, we will add an `elabel` aesthetic to print the name or number assigned to each edge on or next to it. This aesthetic will be very useful for visualizing random graphs or Markov processes where each edge probability is of interest.

2.2.5 Other Arguments

Similar to the other geometries and functions in `ggplot2`, our geometry takes several arguments outside of the aesthetic mapping parameters. One such argument is the `vlabel` argument. The `vlabel` aesthetic is a logical value that, if set to `TRUE` prints the name or number associated with each vertex in the `label` column in order to identify it on the plot. In the final version of this geometry, we will add capabilities to adjust the size and positioning of the labels. For the moment, the color is set to the vertex color, the size is set to twice the vertex size, and all other possible parameters are set to their defaults in `geom_text()`. The `vlabel` argument is most useful for smaller network objects where all vertex names can be printed on their corresponding vertices and still be read clearly. Next, the `directed` argument is a logical value that identifies whether or not the network is directed. When this value is true, arrows are created on the end of the line segment that corresponds to the `to_id` value for each edge. The arrows match the edges in coloring, linetype, and size, and the default length of the arrow is set using the `unit` function in the `grid` package to 0.015 of the normalised parent coordinates of the plot. Finally, the `layout` argument takes a character value corresponding to the possible layouts in the `sna` package that are created by the `gplot.layout.*()` family of functions. The default layout is the Kamada-Kawai layout. This is a force-directed layout for undirected networks (Kamada and Kawai, 1989). There are, however, many other layouts possible.

All layouts that `geom_net` is currently capable of graphing are listed in table 1. The layouts not marked with an asterisk can also take their associated layout parameters. The `layout.par` argument takes a list of named parameters. For instance, if `layout = "random"`, we can also set `layout.par = list(dist = "normal")` to change the distribution of vertex placement from the default uniform to Gaussian.

Layout Name	Description
`"adj"`	a version of `"mds"` which scales the raw adjacency matrix
`"circle"`*	places vertices uniformly in a circle
`"circrand"`	places vertices randomly in a circle
`"eigen"`	places vertices based on the eigenstructure of the adjacency matrix
`"fruchtermanreingold"`	uses a variant of Fruchterman and Reingold's force-directed placement algorithm
`"geodist"`	a version of `"mds"` which scales the matrix of geodesic distances
`"hall"`*	places vertices based on the last two eigenvectors of the Laplacian of the input matrix
`"kamadakawai"`	generates a vertex layout using a version of the Kamada-Kawai force-directed placement algorithm
`"mds"`	places vertices based on a metric multidimensional scaling of a specified distance matrix
`"random"`	places vertices randomly
`"segeo"`	a version of `"mds"` which scales the squared euclidean distances between row-wise geodesic distances
`"seham"`	a version of `"mds"` which scales the Hamming distance between rows/columns of the adjacency matrix
`"target"`	produces a "target diagram" or "bullseye" layout using a force-directed placement algorithm.

Table 1: All possible layouts to pass to `geom_net`. All descriptions are from Butts (2014). Note that the layouts marked with an asterisk (*) are the only layouts that do not take layout parameters.

3. EXAMPLES OF NETWORKS

In this section, we demonstrate the current capabilities of `geom_net` and `ggnet` in a series of side by side examples.

3.1 Blood Donation

In this directed network, there are eight vertices and 27 edges. The vertices represent the eight different blood types in humans that are most important for donation: the ABO blood types A, B, AB, and O, combined with the RhD positive (+) and negative (-) types. The edges are directed: a person whose blood type is that of a *from* vertex can to donate blood to a person whose blood type is that of a corresponding *to* vertex. In the example below, loops are removed because loops exist on every vertex in this example, as blood between two people of matching ABO and RhD type can always be exchanged.

```
ggplot(data = bloodnet, aes(from_id = from, to_id = to)) +
  geom_net(colour="darkred", layout="circle", label=TRUE, size=15,
           directed=TRUE, vjust=0.5, labelcolour="grey80",
           arrowsize=1.5, linewidth=0.75, arrowgap=0.05) + theme_net()
```

```
ggnet2(network(blood$edges[, 1:2]), mode = "circle", size = 15,
       label = TRUE, arrow.size = 7.5, arrow.gap=0.05, vjust = 0.5,
       node.color="darkred", label.color="grey80")
```

This network is shown in figure 2. Here, we have used the aesthetics `colour` and `size` set to identity values to change the size and color of all vertices. We have also used the `layout` and `label` arguments to change the default layout to a circle layout and to print the blood types, respectively. The circle layout places blood types of the same ABO type next to each other and spreads the vertices out far enough to distinguish between the various "in" and "out" types. You can tell clearly from this plot that the O- type is the universal donor: it has an out-degree of seven and an in-degree of zero. Additionally, we can see that the AB+ type is the universal recepient, with an in-degree of seven and an out-degree of zero. Anyone looking at this plot can quickly determine which type(s) of blood they can receive and which type(s) can receive their blood.

3.2 Email Network

This email network comes from the 2014 VAST Challenge (Cook et al., 2014). It is a directed network of emails between company employees with 55 vertices and 9,063 edges. Each vertex is an employee of the company, and each edge is an email sent from one employee to one or more other employees. The arrow of the directed edge points to

(a) geom_net

(b) ggnet2

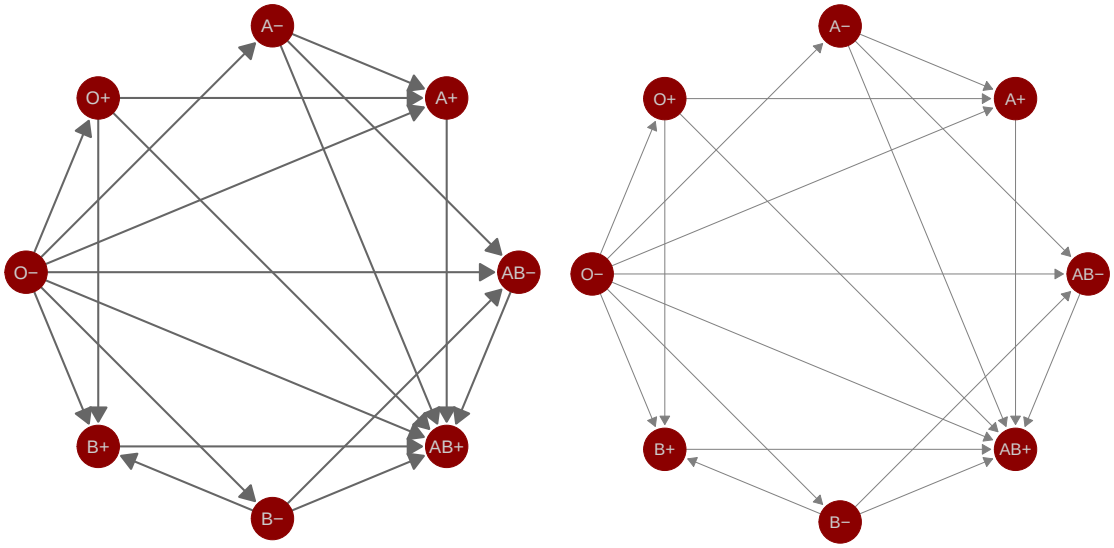


Figure 2: Network of blood donation possibilities in humans by ABO and RhD blood types.

the recipient(s) of the email. The network contains two business weeks of emails across the entire company. In order to better visualize the structure of the communication network between employees, emails that were sent out to all employees are removed in the subsequent examples.

```
ggplot(data = emailnet, aes(from_id = From, to_id = to)) +
  geom_net(aes(colour= CurrentEmploymentType),
           linewidth=0.5, ealpha=0.5, size=4, directed=TRUE) +
  scale_colour_brewer("Employment Type", palette="Set1") + theme_net() +
  theme(legend.position="bottom")
```

```
em.cet = as.character(email$nodes$CurrentEmploymentType)
names(em.cet) = email$nodes$label

# full version
em.net = subset(email$edges, nrecipients < 54)[, c("From", "to") ]
em.net = network(em.net)
em.net %v% "curr_empl_type" = em.cet[ network.vertex.names(em.net) ]

ggnet2(em.net, size = 4, color = "curr_empl_type", palette = "Set1",
       arrow.size = 1.5, edge.alpha = 0.5, color.legend = "Employment Type") +
  theme(legend.position="bottom")
```

This network is plotted in figure 3. There are six distinct clusters in this network which almost perfectly correspond to the six different types of employee in this company: administration, engineering, executive, facilities, information technology, and security. Additionally, the edges between employees in the same cluster are darker than edges between employees in different clusters. This is due to the value of the `ealpha` aesthetic: more emails between two employees lead to darker edges. The value is set to 0.1 in this example, so that ten or more emails between two employees result in a completely opaque edge. This pattern of heavy communication between employees of the same type is fairly unsurprising. To make this visualization more interesting and informative, we facet the network by day: each panel in 4 shows the different email networks associated with each day of the week.

```
ggplot(data = emailnet, aes(from_id = From, to_id = to)) +
  geom_net(aes(colour= CurrentEmploymentType), fiteach=TRUE,
           linewidth=0.5, ealpha=0.1, size=2) +
  scale_colour_brewer(palette="Set1") + theme_net() +
  facet_wrap(~day, nrow = 2, labeller="label_both") +
  theme(legend.position="bottom",
        panel.border = element_rect(fill=NA, colour = "grey60"))
```

(a) geom_net

(b) ggnet2

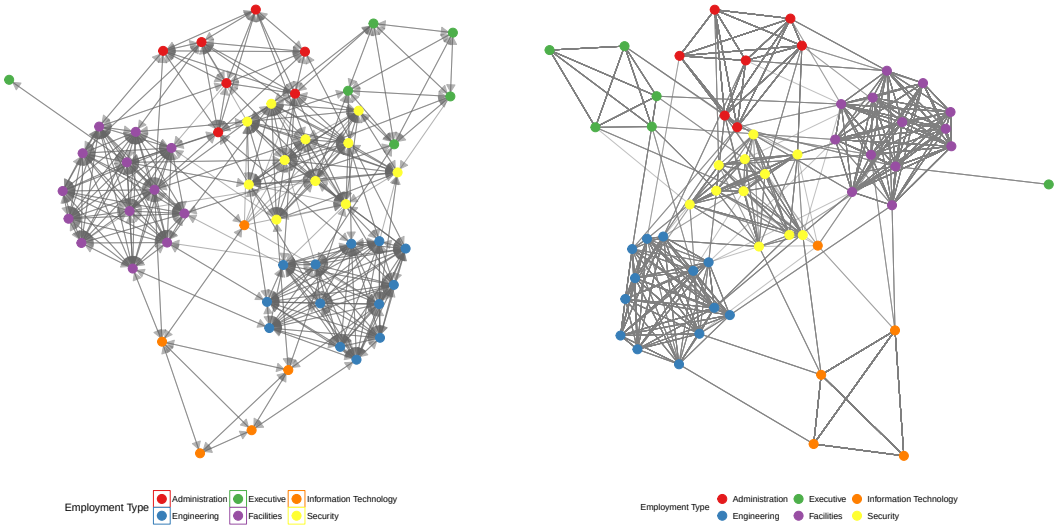


Figure 3: Email network within a company over a two week period.

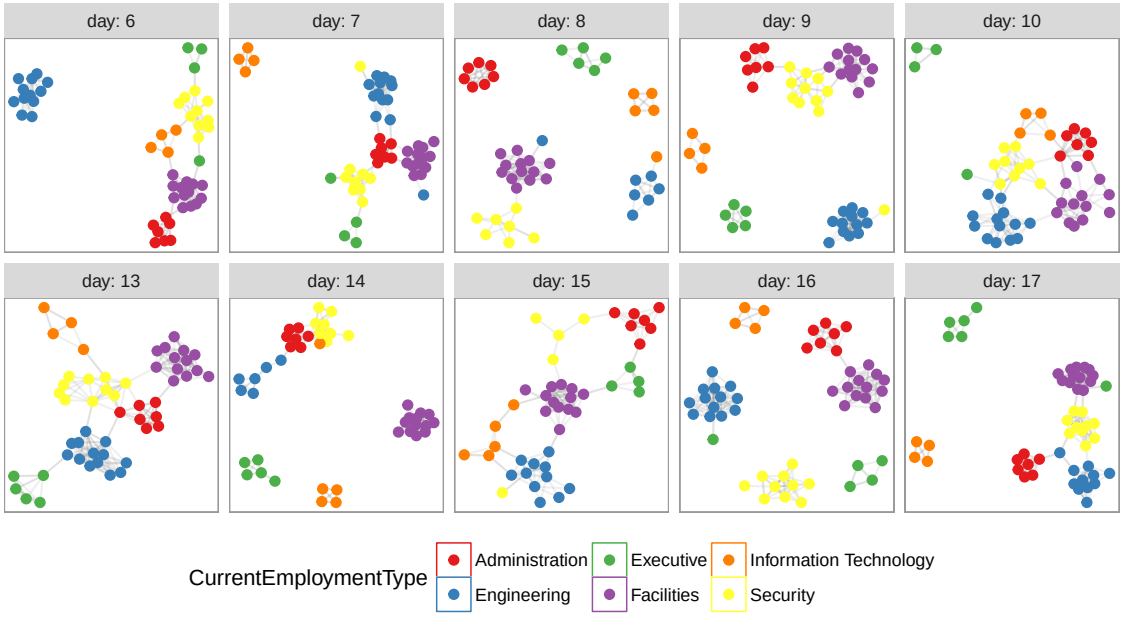
```
em.day = subset(email$edges, nrecipients < 54)[, c("From", "to", "day") ]
em.day = lapply(unique(em.day$day), function(x) subset(em.day, day == x)[, 1:2 ])
em.day = lapply(em.day, network, directed = TRUE)
for (i in 1:length(em.day)) {
  em.day[[ i ]] %v% "curr_empl_type" = em.cet[ network.vertex.names(em.day[[ i ]]) ]
  em.day[[ i ]] %n% "day" = unique(email$edges$day)[ i ]
}

g = list(length(em.day))
for (i in 1:length(em.day)) {
  g[[ i ]] = ggnet2(em.day[[ i ]], size = 2, color = "curr_empl_type",
                    palette = "Set1", arrow.size = 0, arrow.gap=0.01,
                    edge.alpha = 0.1, legend.position = "none") +
    ggtitle(paste("Day", em.day[[ i ]] %n% "day")) +
    theme(panel.border = element_rect(color = "grey50", fill = NA))
}
gridExtra::grid.arrange(grobs = g, nrow = 2)
```

This plot is shown in Figure 4. With the facetting, we can see that there are several days where one or more departments do not communicate with any of the other departments. There are only two days, 13 and 15, without any isolated department communications. Facetting is one of the major benefits of creating a geometry for networks in `ggplot2`. Facetting quickly separates dense networks into separate subnetworks for easy visual comparison and analyses.

3.3 `ggplot2` Theme Elements

This example comes from the `theme()` help page in the `ggplot2` documentation (Wickham, 2009). It is a directed network which shows the structure of the inheritance of theme options in the construction of a `ggplot2` plot. There are 53 vertices and 36 edges in this network. Each vertex represents one possible theme option. There is an arrow from one theme option to another if the element represented by the *to* vertex inherits its values from the *from* vertex. For example, the `axis.ticks.x` option inherits its value from the `axis.ticks` value, which in turn inherits its value from the `line` option. Thus, setting the `line` option to a value such as `element_blank()` sets the entire inheritance tree to `element_blank()`, and no lines appear anywhere on the plot background. Finally, we note that the vertices with no edges were incorporated into the plot by adding their labels to the edges data frame in both the 'from_id' and 'to_id' columns before passing the edges data frame to `ggplot`.

(a) geom_net



(b) ggnet2

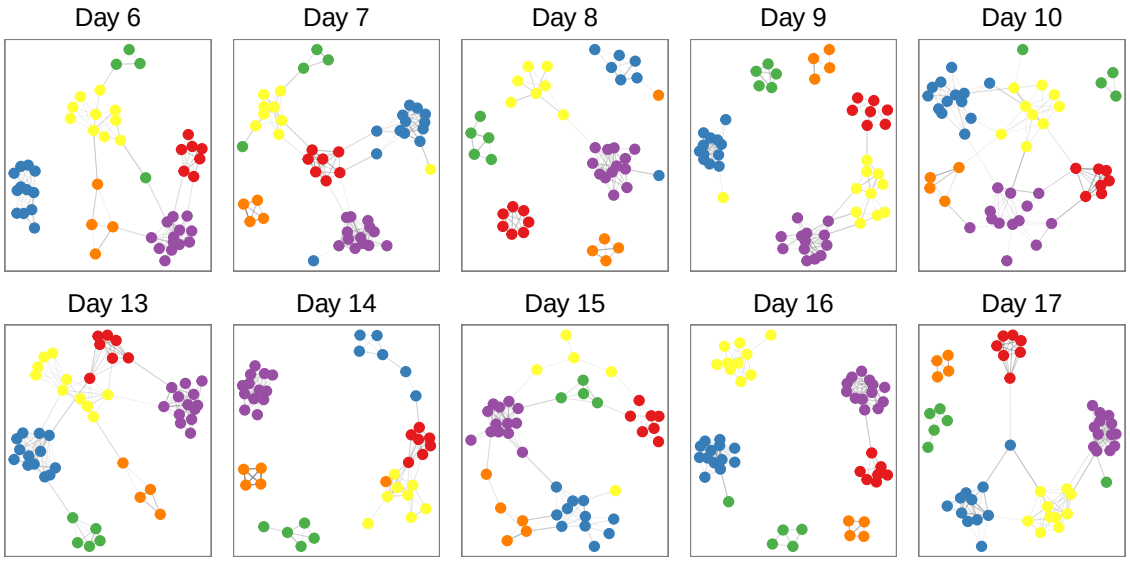


Figure 4: The same email network as in figure 3 facetted by day of the week.

(a) geom_net (b) ggnet2

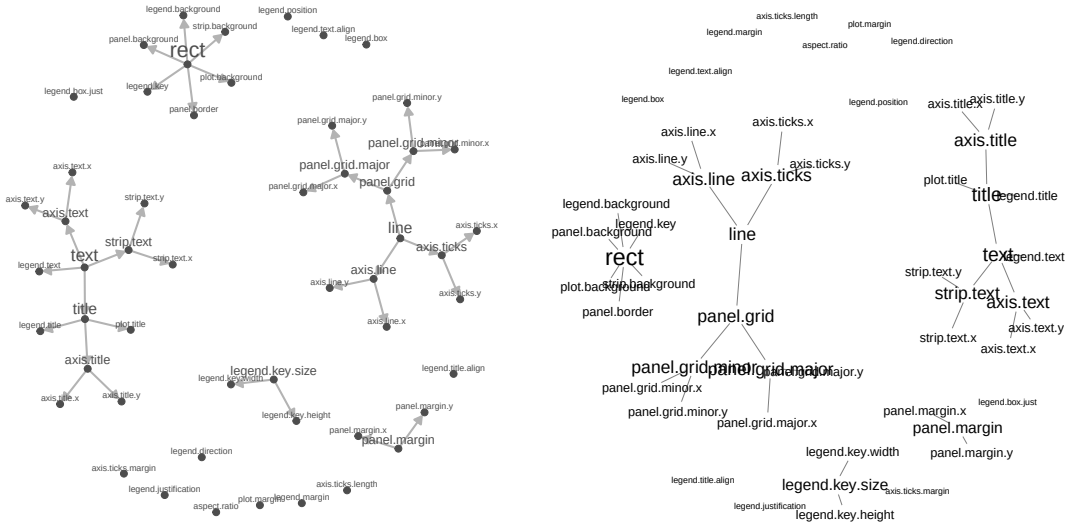


Figure 5: Inheritance structure of `ggplot2` theme elements. This is a recreation of the graph found at `http://docs.ggplot2.org/current/theme.html`.

```
ggplot(data = TEnet, aes(from_id = parent, to_id = child)) +
  geom_net(aes(fontsize=degree), directed = TRUE, label = TRUE, vjust=-.5,
           size=3, ecolour="grey70") +
  theme_net() + xlim(c(-0.05, 1.05))
```

```
te.net = network(theme_elements$edges)
te.net %v% "size" = sqrt(10 * (sna::degree(te.net) + 1))
ggnet2(te.net, label = TRUE, color = "white",
       label.size = "size", layout.exp = 0.15)
```

The inheritance structure is plotted in figure 5. In this plot, it is easy to quickly determine the parent and child vertices. Using this plot made creation of the `theme_net` object used throughout these examples very simple. We just made set each of the major parent elements, `text`, `rect`, and `line` to `element_blank()` and then set the aspect ratio equal to one.

3.4 Mad Men Networks

The following code creates the network example given in the introduction. We changed the vertex size and edge color for all vertices and edges, included vertex labels, and colored

the vertices according to the character's gender.

```
ggplot(data = MMnet, aes(from_id = Name1, to_id = Name2)) +
  geom_net(aes(colour=Gender), size=4, label= TRUE, vjust=-0.6,
           ecolour="grey50") +
  theme_net() +
  scale_colour_manual(values=c( "#FF69B4", "#0099ff")) +
  xlim(c(-0.05, 1.05)) + theme(legend.position="bottom")
```

There is another Mad Men network included in the `gcookbook` package (Chang, 2013). It is a directed network, also of romantic relationships between characters, but it also includes advances made by one character that were rejected by the other. For example, Roger Sterling made advances toward Betty Draper, but Betty refused him, and so there is a directed edge going from Roger to Betty, but not from Betty to Roger. If the advance is reciprocated, like between Sal Romano and the Bellhop, there are two directed edges between the two vertices.

```
ggplot(data = MM2net, aes(from_id = Name1, to_id = Name2)) +
  geom_net(aes(colour=Gender), directed=TRUE, label=TRUE,
           ecolour="grey50", linewidth=0.5, size=2.5, vjust=-.5,
           layout='fruchtermanreingold') +
  scale_colour_manual(values=c( "#FF69B4", "#0099ff")) +
  theme_net() + xlim(c(-.1, 1.1)) + theme(legend.position="bottom")
```

```
rownames(mm.directed$vertices) = mm.directed$vertices$label
mm.directed$vertices$Gender = as.character(mm.directed$vertices$Gender)

mm.dir = network(mm.directed$edges, directed = TRUE)
mm.dir %v% "gender" = mm.directed$vertices[ network.vertex.names(mm.dir), "Gender" ]

# gender color palette
mm.col = c("f" = "#ff69b4", "m" = "#0099ff")

ggnet2(mm.dir, mode = "fruchtermanreingold", size = 3,
       color = mm.col[ mm.dir %v% "gender" ],
       label = TRUE, label.color = mm.col[ mm.dir %v% "gender" ],
       hjust = -0.1, legend.position = "bottom", layout.exp = 0.15,
       arrow.size = 10)
```

This network is shown in figure 6. It is a lot more densely connected than the previous Mad Men example. This network allows us to see much more of the drama in the show.

(a) geom_net (b) ggnet2

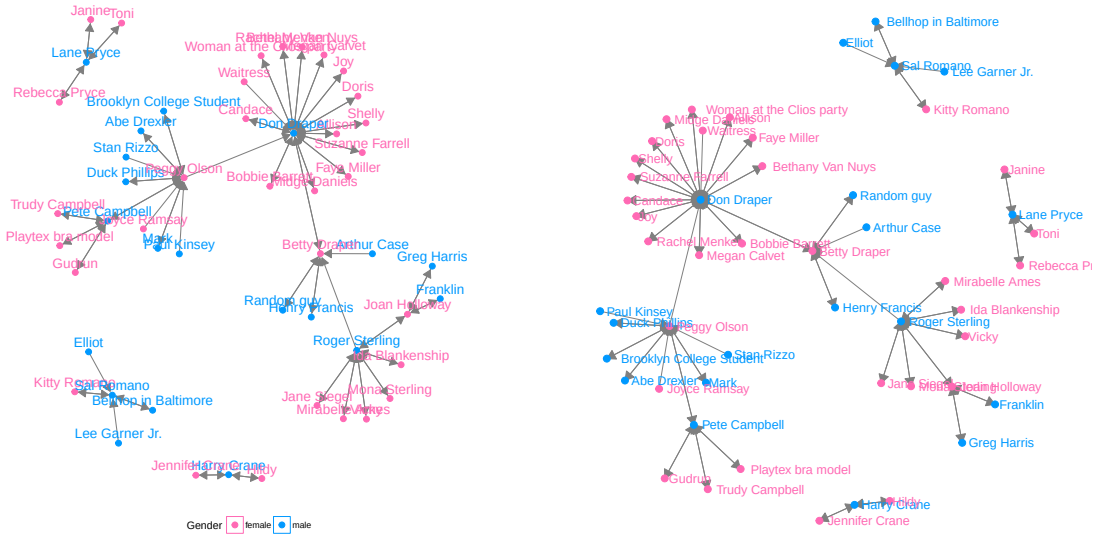


Figure 6: A directed network of relationships in Mad Men.

For instance, we can see that Roger made advances towards Betty, his business partner's wife, which is much more scandalous than what appears in the first network.

3.5 College Football

This next example comes from M.E.J. Newman's network data web page (Girvan and Newman, 2002). It is an undirected network consisting of all regular season college football games played between Division I schools in Fall of 2000. There are 115 vertices and 613 edges: each vertex represents a school, and an edge represents a game played between two schools. There is an additional variable in the vertex data frame corresponding to the conference each team belongs to, and there is an additional variable in the edge data frame that is equal to one if the game occured between teams in the same conference or zero if the game occured between teams in different conferences.

```
sid <- which(ftnet$from %in% c("Navy", "NotreDame", "UtahState",
                               "CentralFlorida", "Connecticut"))
ftnet$schools <- ""
ftnet$schools[sid] <- ftnet$from[sid]

ggplot(data=ftnet, aes(from_id=from, to_id=to)) +
  geom_net(aes(colour=value, linewidth=0.5,
               linetype=factor(same.conf==1), label=schools),
```

```
            size=4,  vjust=-0.5,
            alpha=0.3, layout='fruchtermanreingold') + theme_net() +
  theme(legend.position="bottom") + xlim(c(-0.05, 1.05)) +
  scale_colour_brewer("Conference", palette="Paired")  +
  scale_linetype_manual("Same Conference", values=c(2,1))
```

```
rownames(football$vertices) = football$vertices$label

fb.net = network(football$edges, directed = FALSE)
fb.net %v% "conf" = football$vertices[ network.vertex.names(fb.net), "value" ]

ggnet2(fb.net, mode = "fruchtermanreingold",
       color = "conf", color.legend = "Conference", palette = "Paired")
#       edge.color = c("color", "grey75"))
```

The network of football games is given in figure 7. Here, we have changed the `linetype` aesthetic to correspond to games that occur between teams in the same conference or different conferences. These lines are dotted and solid, respectively. We have also assigned a different color to each conference, and the vertices and their labels are colored according to their conference. This coloring and changing of the linetypes make the structure of the game network easier to view. There is one conference consisting of Navy, Notre Dame, Utah State, Central Florida, and Connecticut, which is spread out, whereas most other conferences' teams are all very close to each other because they play within conference much more than they play out of conference. At the time, these five schools were all independents and did not have a home conference. Without the coloring capability, we would not have been able to pick out that difference as easily.

3.6 Les Misérables

This next network comes from Knuth (1993). It is an undirected network of coappearances of characters in Victor Hugo's *Les Misérables*. There are 77 vertices representing each of the 77 characters in the book. An edge connects two vertices if those two characters appear in the same chapter of the book. There are 254 edges in this network. The edges are also weighted by the number of coappearances. The largest weighting is 31, between the characters Jean Valjean and Cosette. This network is shown in figure 8. This ted daughter.

```
ggplot(data = lesmisnet, aes(from_id = from, to_id = to, linewidth = degree/5 + 0.1 )) +
  geom_net(colour = "grey30",  aes(size=degree),
            ecolour="grey60", vjust=-0.5,
```

(a) geom_net

(b) ggnet2

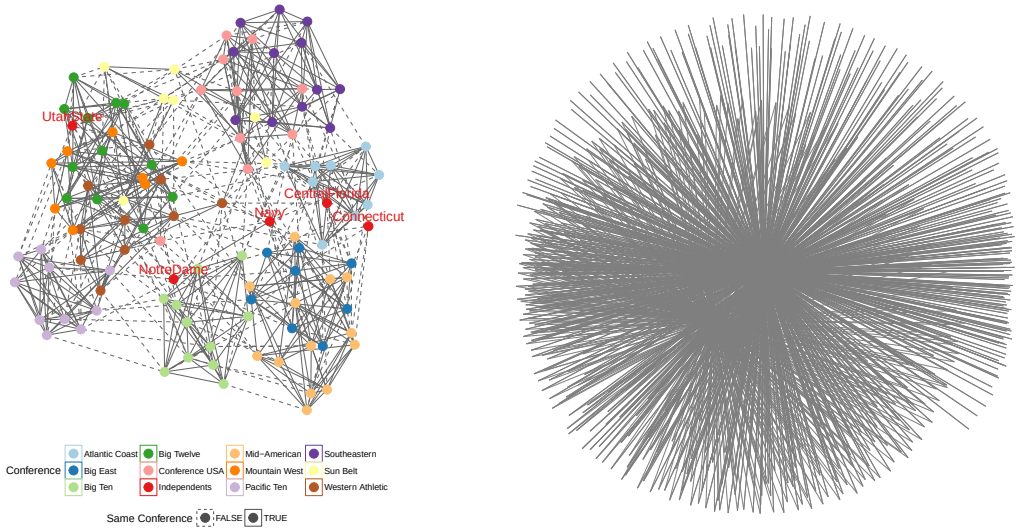


Figure 7: The network of regular season Division I college football games in the season of fall 2000. The vertices and their labels are colored by conference.

(a) geom_net (b) ggnet2

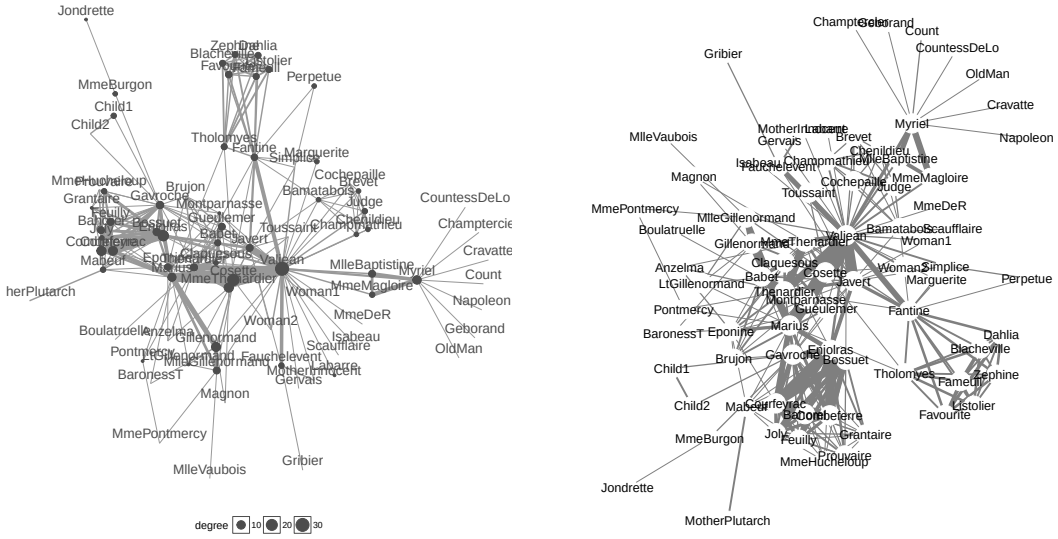


Figure 8: Co-appearance network of characters in Victor Hugo's *Les Misérables*.

```
          layout = 'fruchtermanreingold', label=TRUE) +
  theme_net() +
  theme(legend.position="bottom")
```

```
ggnet2(lesmis$edges[, 1:2 ], mode = "kamadakawai", color = "white", label = TRUE,
       edge.size = lesmis$edges$degree / mean(lesmis$edges$degree),
       layout.exp = 0.25)
```

3.7 Bikesharing in D.C.

The data shows the second quartal trips in 2015 taken with bikes from the bike share company https://secure.capitalbikeshare.com/. While this bikesharing company is located in the heart of Washington D.C. they do have a set of bike stations just outside of Washington in Rockville, MD and north of it. Each station is shown as a dot, lines between stations indicate that at least five trips were taken between these two stations; the wider the line, the more trips have been taken between stations. In order to reflect distance between stations, we use as an additional restriction that the fastest trip was at most ten minutes long.

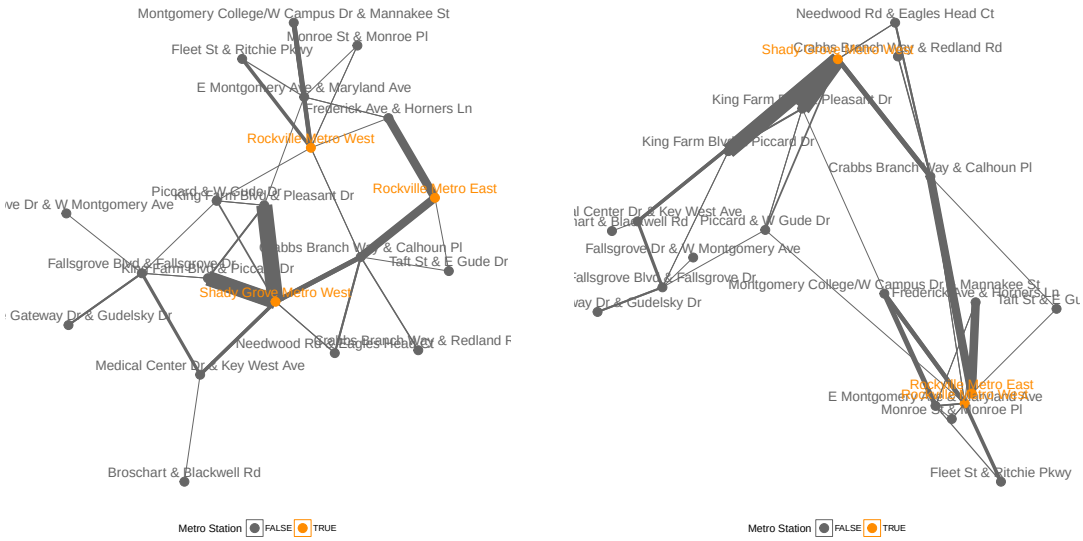


Figure 9: Network of bike trips using a Kamada-Kawai layout (left) and a geographically true representation (right, XXX I would like to include a ggmap here, we need to wait until that package is fixed). Metro stations are shown in orange. In the Kamada-Kawai layout based on the trips taken, metro stations take a much more central position than in the geographically true representation.

```
tripnet$Metro = FALSE
idx <- grep("Metro", tripnet$Start.station)
tripnet$Metro[idx] <- TRUE

ggplot(aes(from_id=Start.station, to_id=End.station), data=tripnet) +
  geom_net(aes(linewidth=n/15, colour=Metro), label=TRUE, vjust=-0.5) +
  theme_net() + xlim(c(-0.1, 1.1)) +
  scale_colour_manual("Metro Station", values=c("grey40", "darkorange")) +
  theme(legend.position="bottom")
# compare to https://secure.capitalbikeshare.com/map/

ggplot(aes(from_id=Start.station, to_id=End.station),  data=tripnet) +
  geom_net(layout=NULL, label=TRUE, vjust=-0.5,
           aes(x=long, y=lat, linewidth=n/15, colour=Metro)) +
  theme_net() +
  scale_colour_manual("Metro Station", values=c("grey40", "darkorange")) +
  theme(legend.position="bottom")
```

3.8 Protein Interaction Network in Yeast

This example of a protein interaction network comes from Jeong et al. (2001). It is the complete protein-protein interaction network in the yeast species *S. cerevisiae*. There are 1,870 proteins that make up the vertices of this network, and there are 2,240 edges between them. These edges represent "direct physical interactions" between any two proteins (Jeong et al., 2001, p. 42). These interactions and their associated proteins are plotted in figure 10. We also demonstrate the layout capabilities by changing the layout to random and setting the distribution to `"uniang"`, which is a "gaussian donut" layout. Indeed, we see a nearly round area in the middle of the graph where the layout parameter has forced there to be no vertices.

```
ggplot(data = protein$edges, aes(from_id = from, to_id = to)) +
  geom_net(alpha=0.25, ealpha = .05, size=2, colour = 'magenta',
           ecolour="grey70", linewidth=0.5,
           layout = 'random', layout.par = list(dist = 'uniang')) +
  theme_net()
```

```
ggnet2(network(yeast$edges[,1:2]), size = 2, color = "magenta",
       mode = "random", layout.par = list(dist = "uniang"),
       edge.alpha = 0.05)
```

4. TECHNICAL DETAILS

In order to construct the `geom` in `ggplot2`, we needed two `R` script files: a 'stat' file and a 'geom' file. The stat file performs all the necessary calculations on the two input data frames, and the geom file performs the drawing of the network.

4.1 Creating `stat_net`

In order to plot the network with only the edge connections and vertex names, we relied on the `sna` and `network` packages (Butts, 2014, 2008). First, we used the edges data frame to construct a network object with the `as.network()` function, then constructed an adjacency matrix from that network using the `as.matrix.network.adjacency()` function. This adjacency matrix is then passed to the chosen `gplot.layout.*()` function from the `sna` package. *When the layout argument is changed in the `geom_net()` function, the corresponding `gplot.layout.*()` function in `sna` is called with a `do.call()` statement.* This layout function produces a matrix of coordinates of the vertices, which we then transform into a data frame containing the coordinates of the edges using the

(a) geom_net

(b) ggnet2

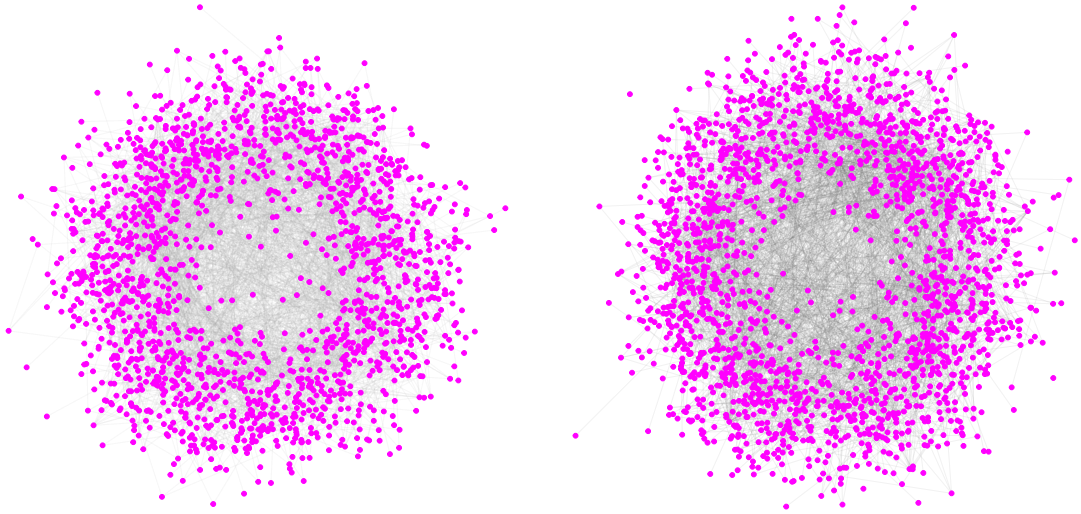


Figure 10: Protein-protein interaction network in *S. cerevisiae*. The layout is random.

as.matrix.network.edgelist() function in network. We are grateful to Moritz Marbach for his gplot() function which we used when creating this process (Marbach, 2011). At the end of the previous routine, we have a data frame with x, y, xend, and yend columns describing the edges and another data frame with x,y columns describing the vertices. These two data frames, along with the many aes() and other argument options get passed on to the actual network geometry for drawing.

4.2 stat_net Usage

The usage of the stat_net arguments is as follows:

- The mapping argument is the aesthetic mapping. Currently, there is no reason to ever change it from NULL.
- The data argument is the edges data frame. This is sent to stat_net through the ggplot() function.
- The vertices argument is the vertex data frame. This is sent to stat_net through the geom_net function.
- The geom argument is the geometric object used to plot the data. It is by default set to net.
- The position argument is set to identity because we wish to plot the network as-is, with no adjustments.
- na.rm

stat_net also accepts the following aesthetic values, none of which are required:

- The layout aesthetic is the layout algorithm to use to place the vertices of the network. The default is the Kamada-Kawai algorithm, and there are other options listed in 1.
- The layout.par aesthetic is the layout arguments to pass to the layout function. It is set to NA by default, in which case the algorithm uses its set defaults in the sna package. This aesthetic takes a list of parameters according to the documentation by Butts (2014).

4.3 Creating geom_net

We constructed geom_net using the drawing capabilities of the geom_point, geom_line, *and* *geom_text* functions that were already in ggplot2. Using these capabilities, we were able to seamlessly incorporate our network geometry into the ggplot2 structure. We took our edges data frame and used that as our input to the GeomSegment draw function to

add a layer of segments to the plot. Then, on top of that, we plot the vertices by passing our vertex data frame to the `GeomPoint` draw function to construct a point layer for the vertices. Finally, we use the `GeomText` draw function to add the vertex labels when the `vlabel` argument is TRUE. This adds a layer of text on top of the other two layers. We added an if/else statement in the `draw` function which created a labeling layer which is `NULL` if `vlabel = FALSE`, which is the default. We then created a new data frame for the label layer. This allows the vertex properties to be propogated through to the labels, including size and color. Finally, we added another if/else statement for the `directed` argument. This adds arrow arguments to the edges data frame which will put arrows pointing toward the "to" vertex in a directed network corresponding to the `to_id` value in the edges data frame.

4.4 `geom_net` Usage

The usage of the `geom_net` arguments is as follows:

- The `mapping` argument is the aesthetic mapping. It is constructed with `aes()` according to the aesthetic values that can be passed to `geom_net`.
- The `data` argument is the edges data frame. There is usually no need to use it in `geom_net()` because it will have already been provided in the `ggplot()` function.
- The `vertices` argument is the vertex data frame. The vertex data frame should be provided to this argument in `geom_net`. This data frame is required to have a column named `label` providing the labels of all vertices in the network.
- The `vlabel` argument is a logical value which plots the label layer on top of the edge and vertex layers when it is set to `TRUE`.
- The `directed` argument is a logical value which adds arrows to the edge layer pointing to the `to_id` vertex when it is set to `TRUE`.
- The `stat` argument is the statistical transformation used on the data for the network layer. It is always set to `net`.

`geom_net` also accepts the following aesthetic values, and the ones marked with an asterisk are required:

- The `from_id`* and `to_id`* aesthetics each take one column which, when combined, provide the edges in a network. If the network is undirected, then their order does not matter. If the network is directed, then the *from* vertex names should all be in the `from_id` column, and their corresponding *to* vertex names should all be in the `to_id` column.
- The `ecolour` aesthetic changes the color of the edges. Its default is `grey20`

- The `esize` aesthetic changes the size (thickness) of the edges. Its default is 1.
- The `elinetype` aesthetic changes the linetype of the edges. Its default is 1.
- The `ealpha` aesthetic changes the transparency of the edges. Its default is 1.
- The `vcolour` aesthetic changes the color of the vertices. Its default is `black`.
- The `vfill` aesthetic changes the ouline color of the vertices. Its default is `black`.
- The `vsize` aesthetic changes the size (diameter) of the vertices. Its default is 2.
- The `vshape` aesthetic changes the shape of the vertices. Its default is 16 (solid circular points).
- The `valpha` aesthetic changes the transparency of the vertices. Its default is 1.

5. FUTURE WORK

As we discussed throughout the paper, there is still some work to be done. The primary change that we need to make is to add the capability of mapping aesthetic values to the plot based on variables directly. Right now, we have the ability to do this indirectly, but we want to add this functionality so that it is easier to change the `ecolor`, `esize`, `elinetype`, `vcolor`, `vfill`, `vsize`, and `vshape` by mapping variables to these aesthetics. We also want to include the legends for each of these aesthetics like we would normally see in `ggplot2`. We also hope to make it an official part of the `ggplot2` world, by either adding it to `ggplot2` itself or by adding it to the `GGally` package. We believe that this is an important contribution to `ggplot2` and to the R user community because it fills in gaps in the network plotting capabilities of R. Ultimately, our goal is to publish this paper in *The R Journal*.

References

Buldyrev, S. V., Parshani, R., Paul, G., Stanley, H. E., and Havlin, S. (2010), "Catastrophic cascade of failures in interdependent networks," *Nature*, 464, 1025–1028.

Butts, C. T. (2008), "network: a Package for Managing Relational Data in R." *Journal of Statistical Software*, 24.

— (2014), *sna: Tools for Social Network Analysis*, R package version 2.3-2.

Butts, C. T., Handcock, M. S., and Hunter, D. R. (2014), *network: Classes for Relational Data*, Irvine, CA, R package version 1.10.2.

Chang, W. (2013), *R graphics cookbook*, Sebastopol, CA: O'Reilly.

Cook, K., Grinstein, G., and Whiting, M. (2014), "Vast Challenge 2014," http://vacommunity.org/VAST+Challenge+2014%3A+Mini-Challenge+1#Download_the_Datasets_Entry_Forms_and_Documentation.

Csardi, G. and Nepusz, T. (2006), "The igraph software package for complex network research," *InterJournal*, Complex Systems, 1695.

Girvan, M. and Newman, M. E. J. (2002), "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. USA*, 99, 7821–7826.

Jeong, H., Barabsi, S. P. M. A.-L., and Oltvai, Z. N. (2001), "Lethality and centrality in protein networks," *Nature*, 411, 41–42.

Kamada, T. and Kawai, S. (1989), "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters*, 31, 7–15.

Knuth, D. (1993), *The Stanford GraphBase : a platform for combinatorial computing*, New York, N.Y. Reading, Mass: ACM Press Addison-Wesley.

Marbach, M. (2011), "Visualizing networks with ggplot2 in R," https://sumtxt.wordpress.com/2011/07/02/visualizing-networks-with-ggplot2-in-r/.

Neuwirth, E. (2014), *RColorBrewer: ColorBrewer Palettes*, R package version 1.1-2.

Newman, M. E. J. (2010), *Networks : an introduction*, Oxford New York: Oxford University Press.

Schloerke, B., Crowley, J., Cook, D., Hofmann, H., Wickham, H., Briatte, F., Marbach, M., and Thoen, E. (2014), *GGally: Extension to ggplot2.*, R package version 0.4.7.

Watts, D. and Strogatz, S. (1998), "Collective dynamics of 'small-world' networks," *Nature*, 393, 440–442.

Wickham, H. (2009), *ggplot2: elegant graphics for data analysis*, Springer New York.