Save Room for Pie: Adding Pie Charts to Network Visualizations in R with statnet and plotrix

Christopher Steven Marcum
National Institutes of Health

David R. Schaefer
Arizona State University

Abstract

In typical network visualization the capacity to convey information about any particular vertex in the graph is limited to around four dimensions. These include manipulation of vertex size, shape, color, and labels. It is often useful, however, to display compositional information about the vertices on the graph that transcend that limit. In this paper, we introduce a method to impose pie charts onto the vertices of social network visualizations. Our code snippet is provided in the text along with two examples: one using fractional data and another using multivariate compositional data. Our goal with this code snippet is to introduce a visual device to convey both relational and compositional information in one figure.

Keywords: network visualization, data visualization, pie charts.

1. Background

The humble pie chart is an important and widely used visual device to display and summarize fractional and compositional data. Typically attributed to the grandfather of statistical visualization, Scottish political economist William Playfair, the pie chart is one of the oldest and most enduring visual statistical tools in use today ([Playfair 1801](#)).

Visualization of graphs (or networks), has an equally storied history dating back to the foundational work on graph theory by [Euler \(1736\[1956\]\)](#). Usually, simple 2-dimensional graphs are depicted by a set of vertices connected by a set of edges, where these features are positioned in Cartesian space based on some rule or algorithm. Often, it is desirable to highlight characteristics of vertices on these graphs by manipulating the features of the vertices to correspond to different variables. However, the number of characteristics one is able to allocate on the visualization using this approach is usually limited to around four (e.g., by manipulating the size, shape, color, and labels of the vertices). With the possible exception of labels, these

visual attributes are limited to conveying singular information about vertices, as opposed to compositional information. Thus, the ability to convey both relational and compositional information in a single graph is currently limited. We've dished up a remedy: superimposing pie charts onto the vertices.

Most simple pie charts are constituted by fractional data where the size and angle of each "slice" of the pie (i.e., each sector of the circle) is proportional to the relative frequency of data falling into a particular class. Formally, each slice of pie in terms of the angles (in degrees) of the whole is defined as:

$$s_i = \frac{k_i}{\sum k} * 360$$

where the angle of the i^{th} slice of pie (s) is equal to the proportion of data falling into class k_i times the number of degrees in a circle (i.e., 360).

Uniquely coloring each slice can highlight the different classes as in the example figure below. However, pie charts may also be useful to convey compositional information, such as the values of a vertex on multiple measures. This is accomplished by sizing each class as a constant, then dividing the pie into equal parts. When used in combination with coloring or shading, this provides a useful display of a set of non-proportional or compositional data, such as multivariate characteristics, in the slices of the pie chart. For example, let a pie chart represent an individual observation (such as a vertex) and each slice represents some independent random variable measured on that vertex. The coloring of each slice can then represent the values of the variables the slice represents.

Figure 1 provides examples of pie charts to display both fractional (on the left) and compositional (on the right) data. This figure was produced using the built-in `pie()` function supplied by the R base **graphics** package and can be reproduced by the following listing:

```
americanpie<-c(47,37,32,27,25,24,24,21,18,16)

aplabels<-c("apple","pumpkin","chocolate creme","cherry","apple crumb",
  "pecan","lemon meringue", "blueberry","key lime","peach")

pie(americanpie,labels=aplabels,xlab="American Pie Preferences",
  col=rainbow(length(aplabels),alpha=.25),
  main="Fractional Data Example")

ptstats<-c(25,25,25,25)
ptlabels<-c("batted right","batted left","threw left","threw right")
ptcols<-c("blue","white","white","blue")

pie(ptstats,labels=ptlabels,col=ptcols,xlab="Pie Traynor's Attributes",
  main="Compositional Data Example")
legend("bottom",legend=c("attribute present","attribute absent"),
  fill=c("blue","white"))
```

The two examples displayed in the figure—American preferences for different pie varieties and the throwing/hitting attributes of baseball's Pie Traynor—illustrate how pie charts may be useful to display information about vertices on network visualizations. In the next section, we define a custom R function to that end.

Figure 1: Examples of Using Pie Charts to Display Different Types of Data. On the left a Pie Chart of the Distribution of American Preferences for Pie Varieties. Data come from the 2008 Schwann's Company Consumer Pie Preference Survey available online at: <http://www.livescience.com/33111-favorite-pie-america.html>. On the right, the throwing and hitting attributes of baseball Hall-of-Famer Pie Traynor.

2. Snippet

We begin with loading the required packages. Specifically, we make use of the **sna** (Butts 2014) and **network** (Butts, Handcock, and Hunter 2014) packages from the **statnet** suite of network analysis software for R. These packages were featured in a special issue of this journal (Handcock, Hunter, Butts, Goodreau, and Morris 2008) and readers are directed there for accessible tutorials. Additionally, we use pie chart functions from the **plotrix** package for R (Lemon 2006). While R supplies a basic pie chart plotting function (`pie()`), as demonstrated above, the `floating.pie()` function from **plotrix** facilitates placement of an arbitrary number of pie charts in a Cartesian coordinate space which is required here.

We define a new convenience function called `add.pie()` that wraps together `gplot()` functionality from the **sna** package and both `floating.pie()` and `pie.labels()` (optionally) from **plotrix**. This new function plots a network with `gplot()`, then uses `floating.pie()` to overlay the network vertices with pie charts. The function definition contains arguments that specify the data sources and additional plotting features, which we discuss in detail below.

```

library(sna)
library(network)
library(plotrix)

add.pie<-function(x,p,sf,coord,cols,r=NULL,pielabel=NULL,...){
  x<-as.sociomatrix(x)
  dp<-dim(p)
  if(any(is.na(p))){
    p[which(is.na(p))]<-0
  }
  if(any(p==0)){
    p.na<-which(p==0,arr.ind=TRUE)
    p[which(p==0)]<-.00001
  }
  p<-prop.table(p,2)
  if(is.null(r)){r<-sf/10}
  gplot(x, vertex.cex=sf, coord=coord, interactive=FALSE,...)
  bisect.angles<-list()

  if(is.matrix(cols)){
    for (i in 1:dp[2]) {
      bisect.angles[[i]]<-floating.pie(coord[i,1],coord[i,2], p[,i],
        edges=500,radius=ifelse(length(r)>1,r[i],r), col=cols[,i])
    }
  }
  else
    for (i in 1:dp[2]) {
      bisect.angles[[i]]<-floating.pie(coord[i,1],coord[i,2], p[,i],
        edges=500,radius= ifelse(length(r)>1,r[i],r), col=cols)
    }
  if(!is.null(pielabel)){
    thelabs<-pielabel$labels
    for (i in 1:dp[2]) {
      if(exists("p.na")){
        if(i%in%p.na[,2]){
          tmp.lab<-thelabs
          tmp.lab[c(p.na[which(p.na[,2]==i),1])]<-" "
          pielabel$labels<-tmp.lab
        }
      }
      pielabel$x<-coord[i,1]
      pielabel$y<-coord[i,2]
      pielabel$angles<-bisect.angles[[i]]
      do.call(pie.labels,pielabel)
      pielabel$labels<-thelabs
    }
  }
}

```

}

The first argument, `x`, is any object accepted by the `dat` argument in `gplot()` such as a binary matrix or a network class object (i.e., from **network**). The second argument, `p` is a $k \times n$ matrix where k is the number of data classes to be represented as sectors on a pie chart and n is the number of vertices in the network (and ordered as they are in x). Each element of `p`, then, represents the filling of each slice of pie for each vertex. Note that `p` can contain missing data and zero-valued cells; these are handled intelligently by setting a trivial value to those cells (0.0001) and any labels associated with those cells are removed per the `pie.labels()` documentation (Lemon 2006).

The third argument, `sf` is a numeric scaling factor (or factors if supplied as a vector) in user-units for the vertices and is both the basis for the default radius (or radii) of the pie charts and passed to the `vertex.cex` argument in `gplot()`. The fourth argument, `coord` is an $n \times 2$ matrix, where the columns supply the x and y coordinates of the centroids of the vertices; this is normally acquired as output from a prior call to `gplot(x)` or possibly to `plot.network(x)`. The fifth argument, `cols` is either a k length character vector of colors for each pie slice (e.g., in the case of fractional data) or a $k \times n$ character matrix (e.g., in the case of multivariate/compositional data). An optional radius for the pies can be supplied by setting `r` to a non-null positive numeric value or vector thereof; the default is $\frac{sf}{10}$ (the relationship, not including vertex borders, between the default vertex size and the units of the default pie radius is about $c \simeq \frac{3}{20}r$). Optional common labels for pie sectors can be supplied as a list via the `pielabels` argument. Finally, additional plotting arguments can be passed to `gplot(...)` via the ellipsis.

Having defined the function for adding pie charts to network visualizations, we now describe three case examples on randomly generated network data. In the first case, we focus on fractional data. The following snippet first sets a random seed for replication, then generates a density-conditioned random graph with ten vertices and twenty edges (`gm`). Next, we call `gplot()`, storing the output (a two column vector of coordinates) for future use to `vc.c`, and assign random data to an object (`p`, which will become the fractional data for our pie charts) and slice colors to an object (`cols`). The call to our convenience function (defined above) `add.pie()` follows. Here, we specify a scaling factor for the vertices of `sf=2` that results in a radius of $\frac{1}{5}$ in user units for the pie charts, which will be circumscribed by a white border as passed to `gplot()` by `vertex.col="white"`. The output is reproduced in the first pane of Figure 2.

```
set.seed(31415)
gm<-rgnm(1, 10, 20)
vc.c<-gplot(gm)
p<-replicate(10,prop.table(rbinom(3,5,.5)))
cols<-c("red","green","blue")
add.pie(x=gm,p=p,cols=cols,sf=2, coord=vc.c,vertex.col="white",jitter=FALSE,
        usearrows=FALSE)
```

It is possible to add labels to the pie charts by passing the appropriate values from the documentation in `help(pie.labels)` as a list to the `pielabels` argument. The code snippet below adds labels to the existing figure, rescales the vertices and radii by specifying `sf=2` and

$r=0.3$, respectively, and drops the white border by specifying `vertex.col="transparent"`: the result is reproduced in the second pane of Figure 3.

```
add.pie(x=gm,p=p,cols=cols,sf=2,r=.3,coord=vc.c,vertex.col="transparent",
        vertex.border=FALSE,jitter=FALSE,pielabel=list(labels=letters[1:3],cex=.5,
        radius=.1,col="white"),usearrows=FALSE)
```

Figure 2: Example of Pie Charts Containing Fractional Data Superimposed on Vertices in a Hypothetical Network Each sector represents the relative fraction of data falling into each class of a categorical variable measured on a vertex (for instance, a country’s ancestral composition).

As we discussed above, another use of pie charts is to display multivariate (non-fractional) data on the vertices. An example of this using `add.pie()` is supplied by the following code snippet. We again begin with a random seed for replication. Next, we create a constant valued 4×10 matrix and store it in `p`, where each cell is set to $\frac{1}{4}$. The rows of this matrix represent equal pie sectors, one for each value of four distinct variables. The variability in each of these hypothetical variables is supplied by each cell’s associated color. These are supplied to the function by the 4×10 matrix of colors in `cols`. We arbitrarily color the vertices here for added visual effect by setting the `vertex.col` argument to `gplot()`. Labeled and unlabeled versions of the resulting plot are reproduced in the first and second panes of Figure 3, respectively. With this example, we also demonstrate different ways to vary the scale of the pie charts,

with or without respect to the vertex scaling factor, by tuning the `sf` and `r` arguments to `add.pie()`, respectively. Manipulating vertex size and background color (seen when the pie chart is smaller than the vertex from `gplot()` it overlays) allows for the simultaneous display of additional vertex attributes.

```
set.seed(92653)
p<-matrix(.25,nrow=4,ncol=10)
cols<-matrix(sample(c("white","gray","red"),40,prob=c(.5,.35,.15),
  replace=TRUE),nrow=4,ncol=10)

add.pie(gm,p=p,cols=cols,sf=sqrt(degree(gm)),r=.15*sqrt(degree(gm)),
  coord=vc.c,vertex.col=cm.colors(5),jitter=FALSE,main="Pane 1",usearrows=FALSE)

add.pie(gm,p=p,cols=cols,sf=sqrt(degree(gm)),coord=vc.c,
  vertex.col=cm.colors(5),jitter=FALSE,
  pielabel=list(labels=c("H","D","C","B"),cex=.5,radius=.1,
  col="black"),main="Pane 2",usearrows=FALSE)
```

3. Discussion

One of the potential uses of superimposing pie charts onto vertices in network visualisation is to display the results of probabilistic latent class assignment or neighborhood membership algorithms on the vertices. Such results can be treated as fractional data because any given connected vertex has a non-zero chance of being assigned to each class. In fact, [Krivitsky and Handcock \(2008\)](#) provide routines to do just that from a fitted latent class exponential random graph model using the **latentnet** package for R. However, their function is specific to the fitted model object class of that package; the custom code snippet that we've introduced here is general to any object that the `gplot()` function in the **sna** package can plot. This previous work was one inspiration for the development of the included code.

Another practical use for the compositional data method comes from networks of health-related communication within families. Here, the method can convey information on the presence of multiple health conditions for each member of the network, as well as the connections between family members (e.g., to show how clustering of heritable diseases may influence family health communication ([Marcum and Koehly 2015](#); [Ersig, Hadley, and Koehly 2011](#))). Existing approaches, such as genograms ([Hardy and Laszloffy 1995](#); [McGoldrick, Gerson, and Shellenberger 1999](#)) and colored eco-genetic relationship maps ([Peters, Kenen, Giusti, Loud, Weissman, and Greene 2004](#)), use consanguine and affine pedigrees (or family trees) to impose relationships (i.e., normally the edges in a network) onto the pedigree vertices. [Ludden, Goergen, and Koehly et al. \(2012\)](#) proposed to use quadrant sectors of a pie chart to visualize four health conditions among family members in a pedigree side-by-side with network data in an Family Health History application. Our method can be helpful in combining aspects of the pedigree with social network information in such contexts.

Finally, it is important to point out that not all data scientists are as enthusiastic about pie charts as we happen to be. For instance, [Cleveland \(1985\)](#) contends (among others) that pie charts are a very bad way to represent data. He writes, "Data that can be shown by pie charts

Figure 3: Example of Pie Charts Containing Compositional Data Superimposed on Vertices in a Hypothetical Network Each quadrant represents that vertex’s value on a single attribute (for instance, a person’s severity of each of four medical conditions).

always can be shown by a dot chart. This means that judgments of position along a common scale can be made instead of the less accurate angle judgements” (pg. 264). In the case of network visualization, however, the alternatives (dot charts, for instance) are not viable. Information overload, however, may still be a concern and we would caution against having too much pie in that regard. We would leave it to individual researchers to determine whether the value of our proposal to leverage the compactness of pie charts on network visualizations outweighs the cost of interpretation. Certainly, with the scripts we’ve supplied here, making this assessment is a piece of cake—err, easy as pie.

References

- Butts CT (2014). *sna: Tools for Social Network Analysis*. R package version 2.3-2, URL <http://CRAN.R-project.org/package=sna>.
- Butts CT, Handcock MS, Hunter DR (2014). *network: Classes for Relational Data*. Irvine,

- CA. R package version 1.11.3, URL <http://statnet.org/>.
- Cleveland WS (1985). *The elements of graphing data*. Wadsworth Advanced Books and Software, Monterey, CA.
- Ersig AL, Hadley DW, Koehly LM (2011). “Understanding patterns of health communication in families at risk for hereditary nonpolypoid colorectal cancer: examining the effect of conclusive versus indeterminate genetic test results.” *Health Communication*, **26**(7), 587–594.
- Euler L (1736[1956]). *The seven bridges of Konigsberg*. Wm. Benton.
- Handcock MS, Hunter DR, Butts CT, Goodreau SM, Morris M (eds.) (2008). *Journal of Statistical Software*, volume 24 of *Special Volume: statnet: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data*. URL <https://www.jstatsoft.org/issue/view/v024>.
- Hardy KV, Laszloffy TA (1995). “The cultural genogram: Key to training culturally competent family therapists.” *Journal of Marital and Family Therapy*, **21**(3), 227–237.
- Krivitsky P, Handcock MS (2008). “Fitting position latent cluster models for social networks with latentnet.” **24**. URL <https://www.jstatsoft.org/issue/view/v024>.
- Lemon J (2006). “Plotrix: a package in the red light district of R.” *R-News*, **6**(4), 8–12.
- Ludden A, Goergen A, Koehly et al L (2012). “Families are an important social context for intervention and lifestyle-focused disease prevention.” In *Translating Genomics through a Social Behavioral Lens: 10th Anniversary of the Social Behavioral Research Branch, National Human Genome Research Institute*. National Institutes of Health, Bethesda, MD. URL <https://www.genome.gov/27555812/translating-genomics-through-a-social-and-behavioral-lens/>.
- Marcum CS, Koehly LM (2015). “Inter-Generational Contact from a Network Perspective.” *Advances in Life Course Research*, **24**(2), 10–20. doi:[doi:10.1016/j.alcr.2015.04.001](https://doi.org/10.1016/j.alcr.2015.04.001).
- McGoldrick M, Gerson R, Shellenberger S (1999). *Genograms: Assessment and Intervention*. Second edition. W.W. Norton and Co., New York.
- Peters JA, Kenen R, Giusti R, Loud J, Weissman N, Greene MH (2004). “Exploratory study of the feasibility and utility of the colored eco-genetic relationship map (CEGRM) in women at high genetic risk of developing breast cancer.” *American Journal of Medical Genetics Part A*, **130**(3), 258–264.
- Playfair W (1801). *The statistical breviary; shewing, on a principle entirely new, the resources of every state and kingdom in Europe; illustrated with stained copper-plate charts, representing the physical powers of each distinct nation with ease and perspicuity*. T. Bensley, J. Wallis [etc., etc.], London.

Affiliation:

Christopher Steven Marcum

National Institutes of Health

Bethesda, Maryland, USA

E-mail: chris.marcum@nih.gov

URL: <http://www.chrismarcum.com>

SocArXiv on Twitter

Temporary Home of SocArXiv Preprints

MMMMMM YYYY, Volume VV, Issue II

[doi:osf.io/0000](https://doi.org/10.0000)

<https://twitter.com/socarxiv>

<https://osf.io/view/socarxiv/>

Submitted: yyyy-mm-dd

Accepted: yyyy-mm-dd