Arc Diagrams with `arcdiagram` and `network`

Gaston Sanchez
www.gastonsanchez.com/arcdiagram

1 Introduction

`arcdiagram` is a package for producing pretty arc diagrams of graphs in R. You can think of `arcdiagram` as a plugin of the package `igraph` (by Gabor Csardi and Tamas Nepusz). In this document we will discuss how to use `arcdiagram` to produce arc diagrams of graphs from the package `network` (by Carter Butts, David Hunter, and Mark Handcock).

2 R package `network`

R has several packages dedicated to graphs and network analysis. There is even a **CRAN Task View** for graphical models that you can check at:
http://cran.r-project.org/web/views/gR.html

`arcdiagram` has been designed *to fit like a glove* for graph edge lists obtained from `igraph`. However, graph edge lists can also be obtained using the package `network`. Let's see how to play with `arcdiagram` and `network`. Warning: I'm assuming that you already checked the introductory documentation of the `arcdiagram` package available at:
http://www.gastonsanchez.com/arcdiagram

Step 1: Load packages

First let's load the packages `arcdiagram` and `network` (I'm assuming you already installed them)

```
# load 'arcdiagram'
library(arcdiagram)

# load 'network'
library(network)
```

Step 2: Toy example

Let's start with a very simple example. We will generate a random graph with 7 nodes. One way to do this is by generating an adjacency matrix first and then create the graph with the function `network()`:

```
# generate a random adjacency matrix
set.seed(95)
toy_matrix = matrix(rbinom(49, 1, 0.25), 7, 7)
diag(toy_matrix) = 0
```

```
# create a graph from the adjacency matrix
toy_graph = network(toy_matrix, directed = FALSE)
```

Step 2: Extract `edgelist`

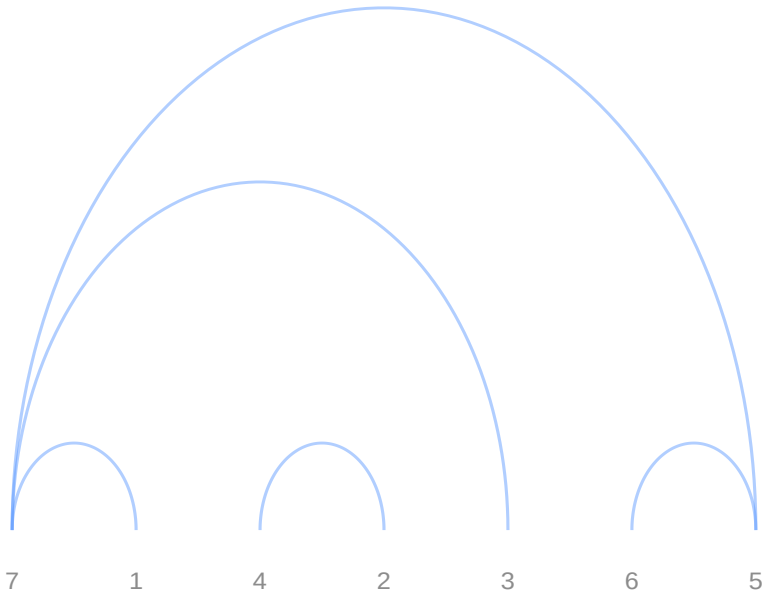
Once we have the network, we need to extract the edge list which will be used as the main argument for `arcplot()`. The way to obtain an edge list from a `"network"` object is with the function `as.matrix()` and the argument `matrix.type = "edgelist"`

```
# edgelist
toy_edges = as.matrix(toy_graph, matrix.type = "edgelist")
```

Step 3: Plot arc diagram

The edge list `toy_edges` is all you need to produce an arc diagram with `arcplot()`:

```
# plot arc diagram
arcplot(toy_edges, las = 1)
```



3 Floretine Weddings Network

Let's see another example using a more interesting data set. We will use the data `"flo"` which is one of the datasets available in `network`. This data set consists of weddings among leading Florentine families (in Italy).

Step 1: Data `flo`

The way to get a graph (i.e. network) from the data `"flo"` is by using the function `network()`:

```
# load data 'flo'
data(flo)

# network
netflo = network(flo)

# what does 'netflo' look like?
netflo

##  Network attributes:
##   vertices = 16
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 40
##     missing edges= 0
##     non-missing edges= 40
##
##  Vertex attribute names:
##     vertex.names
```

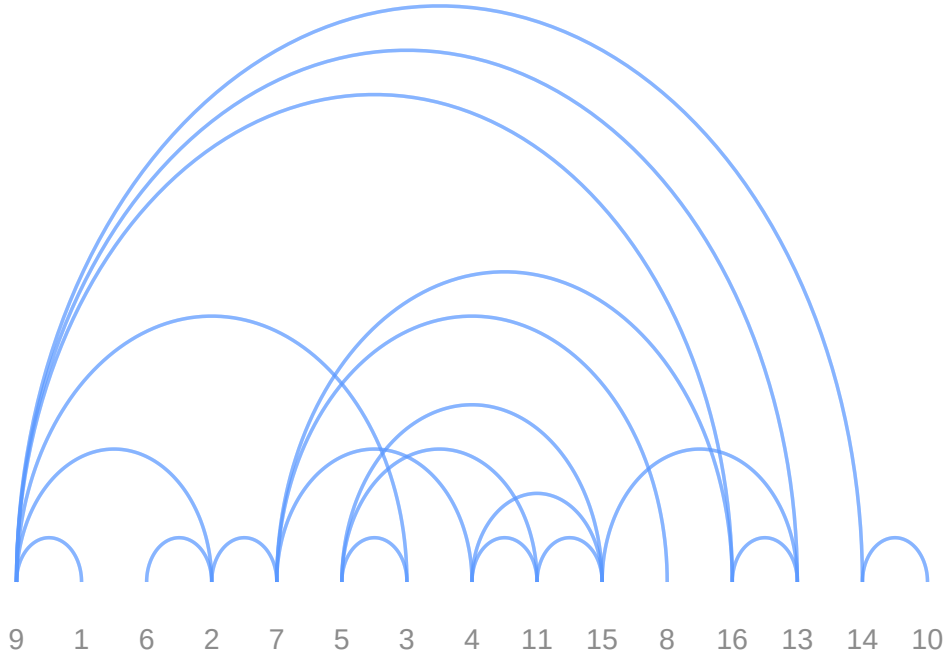
Step 2: Edgelist

Because the main argument for `arcplot()` is an edge list, we need to use `as.matrix()` with its argument `matrix.type = "edgelist"` to get such a list form `netflo`:

```
# edgelist
flo_edges = as.matrix(netflo, matrix.type = "edgelist")
```

Now we can get a first arc diagram:

```
# second plot
arcplot(flo_edges, las = 1)
```



Step 3: Node Labels

If you inspect either the object `netflo` or the edgelist `flo_edges`, you will see that we have the names of the nodes (i.e. the vertices) as an attribute, for instance:

```
# print netflo
netflo

##  Network attributes:
##   vertices = 16
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 40
##     missing edges= 0
##     non-missing edges= 40
##
##  Vertex attribute names:
##     vertex.names
```

As you can see in the last lines of the output, `netflo` contains the vertex attribute names `vertex.names`. One way to get these names is by using the function `get.vertex.attribute()`:

```
# try to get vertex names
get.vertex.attribute(netflo, "vertex.names")

##  [1] "Acciaiuoli"   "Albizzi"      "Barbadori"    "Bischeri"
##  [5] "Castellani"   "Ginori"       "Guadagni"     "Lamberteschi"
##  [9] "Medici"       "Pazzi"        "Peruzzi"      "Pucci"
## [13] "Ridolfi"      "Salviati"     "Strozzi"      "Tornabuoni"
```

If by any chance you find an error message telling you something like this (don't panic):

```
Error in get.vertex.attribute(netflo, "vertex.names") :
  Not a graph object
```

The cause of the problem is a "compatibility" issue between the packages `igraph` and `network` (remember that `arcdiagram` depends on `igraph`). Both packages have the same name for some of their functions —`get.vertex.attribute()` among them—. Actually, since we first load `arcdiagram`, this implies that the homonym functions of `network` are **masked** by those of `igraph`. The solution? Use the double colon operator: look for `help("::")` in your R console to get more information. To get the vertex names from `netflo` we have to specify the **namespace** under which the required `get.vertex.attribute()` function is located:

```
# get vertex names
flo_names = network::get.vertex.attribute(netflo, "vertex.names")

# show me the names
flo_names

##  [1] "Acciaiuoli"   "Albizzi"      "Barbadori"    "Bischeri"
##  [5] "Castellani"   "Ginori"       "Guadagni"     "Lamberteschi"
##  [9] "Medici"       "Pazzi"        "Peruzzi"      "Pucci"
## [13] "Ridolfi"      "Salviati"     "Strozzi"      "Tornabuoni"
```

An alternative way to get the vertex names is by extracting them directly from the edge list `flo_edges` with the function `attributes()`:

```
# another way to get vertex names
attributes(flo_edges)$vnames

##  [1] "Acciaiuoli"   "Albizzi"      "Barbadori"    "Bischeri"
##  [5] "Castellani"   "Ginori"       "Guadagni"     "Lamberteschi"
##  [9] "Medici"       "Pazzi"        "Peruzzi"      "Pucci"
## [13] "Ridolfi"      "Salviati"     "Strozzi"      "Tornabuoni"
```

Of course, this a painless way to get the names, but I wanted to show you the hard way in case you find yourself trapped in that quandary.

Step 4: Arc plot attempt

Having extracted the `vertex.names`, it seems that we are ready to plot an arc diagram with the node labels:

```
# arc plot with node labels
arcplot(flo_edges, labels = flo_names)

## Error:
## Length of 'labels' differs from number of nodes
```

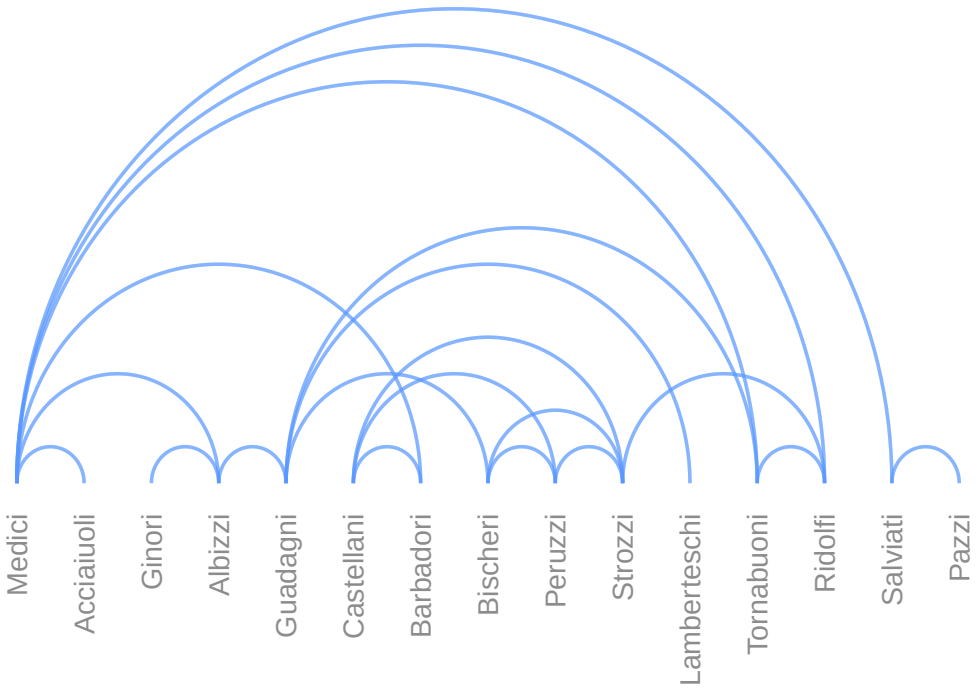
Well, we are not ready yet. The problem is that we have more labels than nodes in the edge list (node 12 does not appear in `flo_edges`). The solution is a bit elaborated: first we need to get the node numbers from the edgelist, and then we have to select their corresponding names:

```
# numeric indices in 'flo_edges'
temp = unique(as.vector(t(flo_edges)))
temp

##  [1]  9  1  6  2  7  5  3  4 11 15  8 16 13 14 10


# node labels
flo_labels = attributes(flo_edges)$vnames[temp]

# check it
flo_labels

##  [1] "Medici"       "Acciaiuoli"   "Ginori"       "Albizzi"
##  [5] "Guadagni"     "Castellani"   "Barbadori"    "Bischeri"
##  [9] "Peruzzi"      "Strozzi"      "Lamberteschi" "Tornabuoni"
## [13] "Ridolfi"      "Salviati"     "Pazzi"
```

Step 5: Final plot

Now we are ready to produce the desired arc diagram:

```
# second plot
arcplot(flo_edges, labels = flo_labels)
```



Some References

- Arc Diagrams in 'Visual Complexity' (by Manuel Lima)
  http://www.visualcomplexity.com/vc/index.cfm?method=Arc%20Diagrams
- Protovis by Mike Bostock
  http://mbostock.github.com/protovis/ex/arc.html
- Arc Diagrams: Visualizing Structure in Strings by Martin Wattenberg
  http://hint.fm/papers/arc-diagrams.pdf
- R-chie: A web server and R package for plotting arc diagrams of RNA secondary structures (by Daniel Lai, Jeff R. Proctor, Jing Yun A. Zhu, and Irmtraud M. Meyer)
  http://www.e-rna.org/r-chie/index.cgi