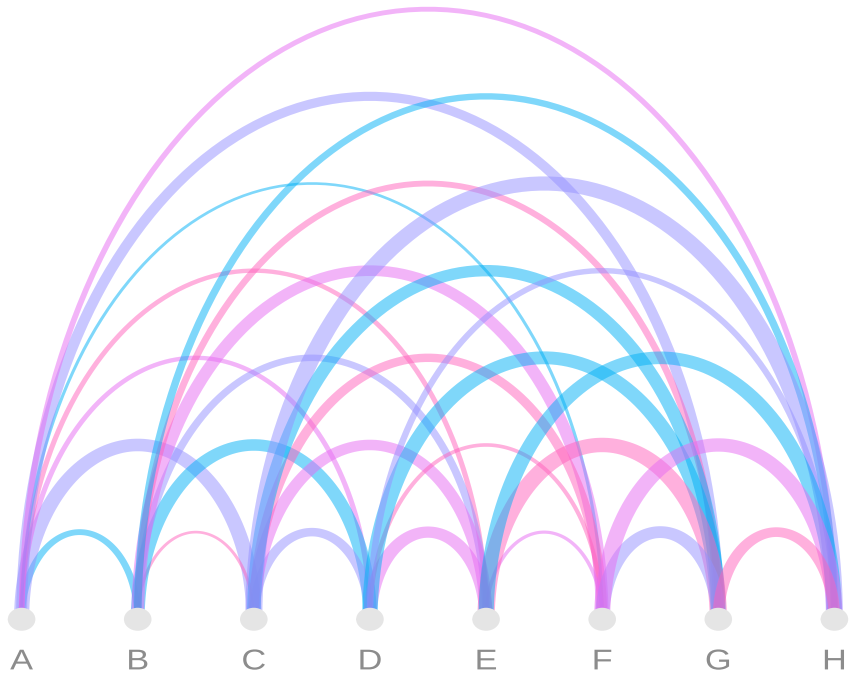
Introduction to the R package `arcdiagram`

Gaston Sanchez
www.gastonsanchez.com/arcdiagram

1 Introduction

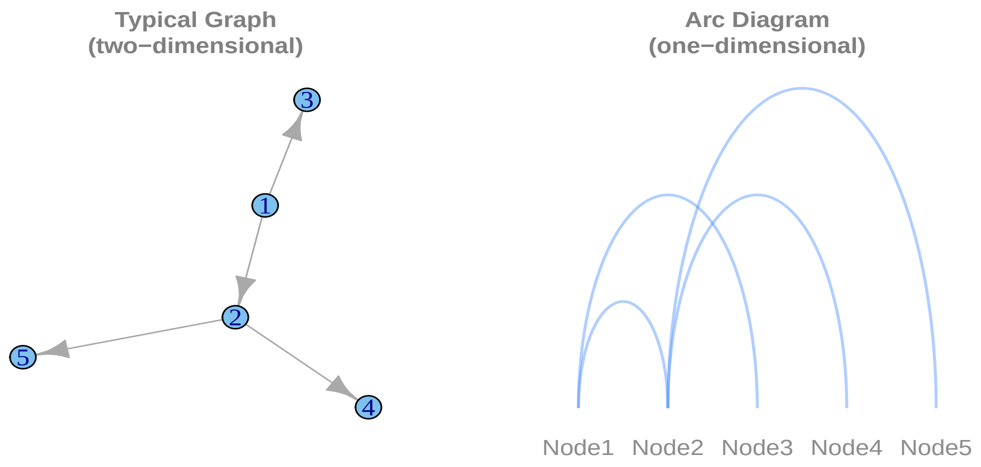
`arcdiagram` is a minimalist R package designed for the solely purpose of helping you plot pretty arc diagrams like the one below:



1.1 Arc Diagrams

So, what is an arc diagram? Briefly, an **arc diagram** is a graphical display to visualize graphs or networks in a *one-dimensional layout*. The main idea is to display nodes along a single axis, while representing the edges or connections between nodes with arcs.

The reason why an arc diagram is said to be a one-dimensional layout is because all the nodes lay on one axis, as opposed to other more common visualizations of graphs where nodes are spreaded in a two-dimensional space (see the following figure):



One of the disadvantages of arc diagrams is that they are sensitive to node ordering; ideally, better visualizations are achieved when related nodes are close to each other, which helps us detect clusters or groups of nodes.

Often, a random node ordering produces poor diagrams in which large arcs have crossovers that difficult visual processing. From the practical point of view, the ordering issue will force you to play with different settings until you find one that allows you to get good results.

Some network specialists acknowledge that arc diagrams might be appropriate for undirected graphs, making them useful for annotations as compared to other two-dimensional network layouts, such as rollup, matrix or table layouts.

2 The R package `arcdiagram`

`arcdiagram` is a minimal package for plotting basic arc diagrams in R. My main motivation for creating `arcdiagram` was because of **Similar Diversity**, a really nice graphic design project by Philipp Steinweber and Andreas Koller (http://similardiversity.net/). Being deeply captivated with such an amazing visualization I decided to do a more modest attempt in R which indirectly took me to the development of `arcdiagram`.

2.1 Installation

`arcdiagram` is not available on CRAN; instead it lives in one of my github repositories:
https://github.com/gastonstat/arcdiagram
This means that you have to use the package `devtools` (by Hadley Wickham) in order to install `arcdiagram` in R:

```
# install 'devtools' if you haven't
install.packages("devtools")

# load devtools
library(devtools)
```

```
# then download 'arcdiagram' using 'install_github'
install_github("arcdiagram", username = "gastonstat")

# load arcdiagram
library(arcdiagram)
```

2.2 Basic Usage

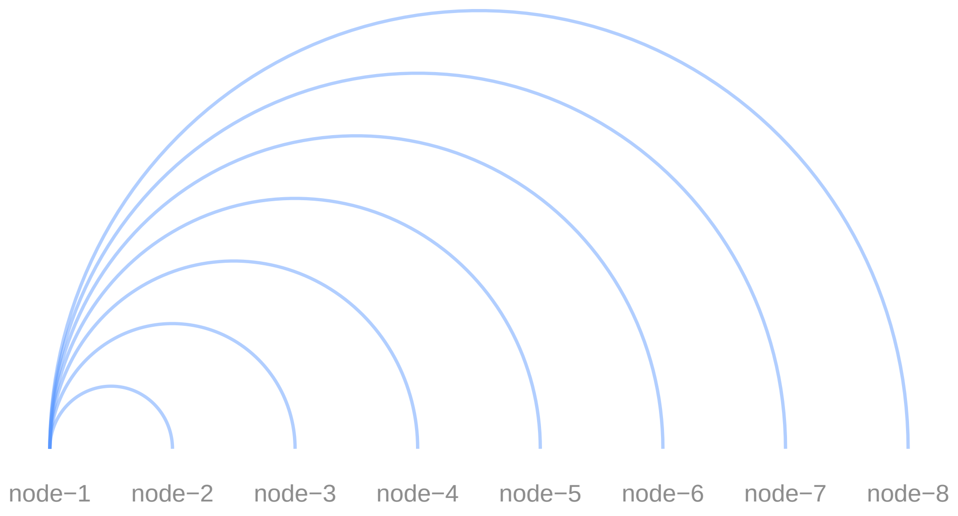
Because the only purpose behind `arcdiagram` is to visualize graphs (i.e. networks), I designed it having in mind the package `igraph` (by Gabor Csardi and Tamas Nepusz). In other words, I developed `arcdiagram` as a *plugin* of `igraph`. The structure of the package is very simple and it consists of one main function: `arcplot()`; and one accesory function: `xynodes()`. The way `arcplot()` works is very simple: it takes an `"edgelist"` object and it plots the edges as arcs. If you are not familiar with the term, an **edge list** is just a two column matrix that gives the list of edges for a graph. Let's see an example:

```
# create a graph with 8 nodes
star_graph = graph.star(8, mode = "out")

# add names to nodes
V(star_graph)$name = paste("node", 1:vcount(star_graph), sep = "-")

# extract edgelist
star_edges = get.edgelist(star_graph)

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



2.2.1 `arcplot()` function

If you check the documentation of the function `arcplot()` you'll see that there's a bunch of parameters to play with. The main parameter is the `edgelist` which is a two-column matrix with the edges of the graph. This is the mandatory ingredient that you have to provide. The rest of parameters are either optional or have default values. However, if you want to spark your creativity with `arcplot()` we need to talk about the elements present in an arc diagram.

Basically we have three elements: 1) the nodes (or vertices), 2) the arcs (or edges), and 3) the node labels. The nodes and the labels are optional (you can choose whether to show them).

Arcs arguments The arcs are plotted by `arcplot()` using the `lines()` function internally. This means that the arc-related arguments are basically the arguments behind `lines()`:

`col.arcs`	color for the arcs
`lwd.arcs`	line width for the arcs (default 1)
`lty`	line type for the arcs
`lend`	the line end style for the arcs
`ljoin`	the line join style for the arcs
`lmitre`	the line mitre limit for the arcs

Node symbols' arguments Node symbols are plotted by `arcplot()` using the `points()` function internally. What this means is that the node symbols arguments are basically the arguments that `points()` uses:

`show.nodes`	logical indicating whether to show node symbols
`pch.nodes`	symbol to use when plotting nodes
`cex.nodes`	expansion of the node symbols
`col.nodes`	color of the node symbols
`bg.nodes`	background (fill) color for the node symbols given by `pch.nodes=21:25`
`lwd.nodes`	line width for drawing node symbols

Node labels' arguments In turn, node labels are plotted by `arcplot()` using the `mtext()` function internally. Node labels' arguments are related to the arguments behind `mtext()`:

`show.labels`	logical indicating whether to show node labels
`labels`	character vector with labels for the nodes
`col.labels`	color of the node labels
`cex.labels`	expansion of node labels
`las`	numeric in 0,1,2,3; the style of axis labels
`font`	font used for node labels
`line`	on which margin line the node labels are displayed
`outer`	use outer margins, if available, to plot node labels
`adj`	adjustment for each string in reading direction
`padj`	adjustment for each string perpendicular to the reading direction

Additional arguments In addition to the previous arguments, there's a handful of additional parameters in `arcplot()` that control the ordering of the nodes and the layout orientation:

`sorted`	logical to indicate if nodes should be sorted
`decreasing`	logical to indicate type of sorting
`ordering`	optional numeric vector providing the ordering of nodes
`horizontal`	logical indicating whether to plot in horizontal orientation

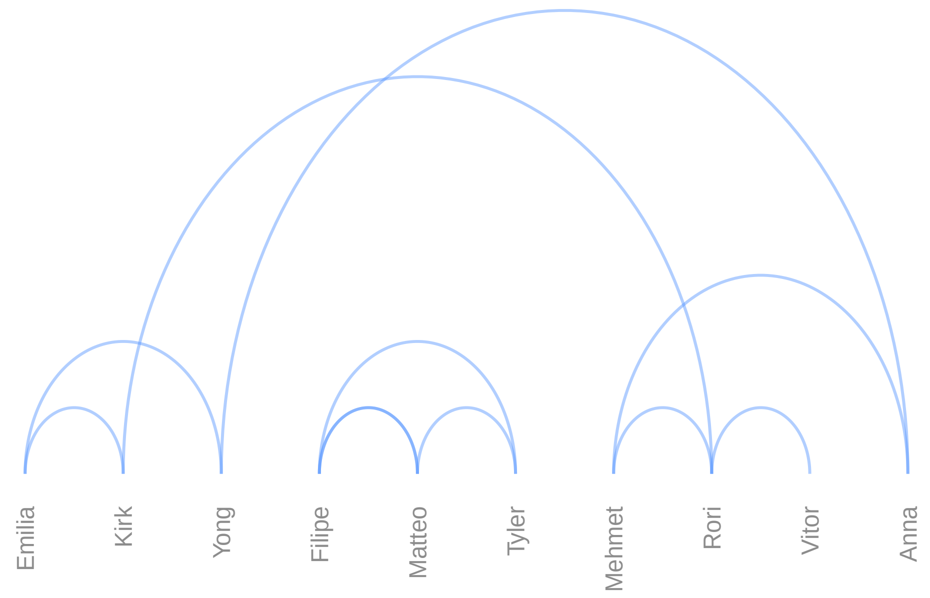
2.3 Examples

Let's see a basic example. We will manually create an `edgelist` representing a graph of projects between coworkers in a scientific laboratory:

```
# create an edgelist
lab = rbind(c("Emilia", "Kirk"), c("Emilia", "Yong"), c("Filipe", "Matteo"),
    c("Filipe", "Tyler"), c("Matteo", "Filipe"), c("Matteo", "Tyler"), c("Mehmet",
        "Rori"), c("Rori", "Kirk"), c("Rori", "Vitor"), c("Anna", "Mehmet"),
    c("Anna", "Yong"))
```

Once we created our `edgelist`, we can create an arc diagram with `arcplot()`:

```
# arc diagram
arcplot(lab)
```



If you want to inspect how the `edgelist` looks like, simply type:

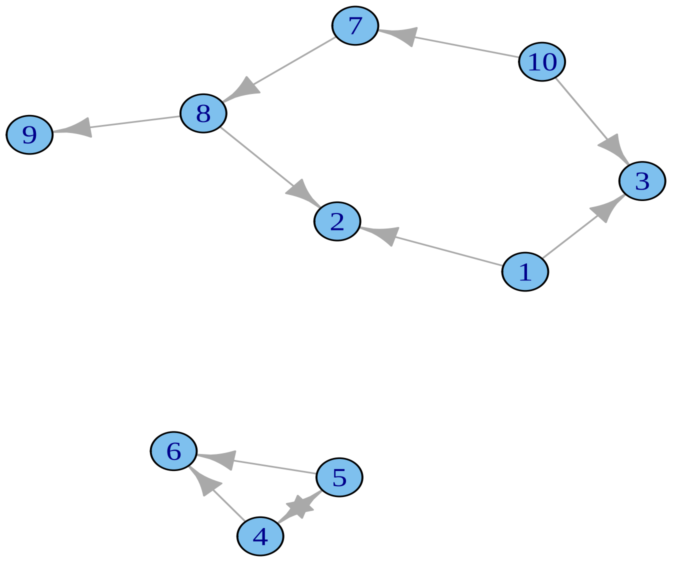
```
# how does it look like
lab

##       [,1]     [,2]
##  [1,] "Emilia" "Kirk"
##  [2,] "Emilia" "Yong"
##  [3,] "Filipe" "Matteo"
##  [4,] "Filipe" "Tyler"
##  [5,] "Matteo" "Filipe"
##  [6,] "Matteo" "Tyler"
##  [7,] "Mehmet" "Rori"
##  [8,] "Rori"   "Kirk"
##  [9,] "Rori"   "Vitor"
## [10,] "Anna"   "Mehmet"
## [11,] "Anna"   "Yong"
```

Creating a graph

If we want to plot our graph using the package `igraph()`, first we need to create a `"graph"` object with the function `graph.edgelist()`, and then we can use the default `plot()` method:

```
# make graph from edgelist
glab = graph.edgelist(lab, directed = TRUE)
# plot graph
plot(glab)
```



Playing with arc diagrams

The first arc diagram that we produced is not bad but we can definitely improve it. For instance, we can take into account nodes' degree —the **degree** of a node is the number of edges connected to it— using the function `degree()`:

```
# degrees
lab_degree = degree(glab)
```

Let's say that we want to put weights on the edges so they reflect some sort of value. We can do this by assigning some random numbers:

```
# assign random weights to edges
set.seed(123)
E(glab)$weight = round(runif(nrow(lab), 0.5, 4))
```

We can also get clusters of the nodes using the function `clusters()`:

```
# get clusters
gclus = clusters(glab)
```

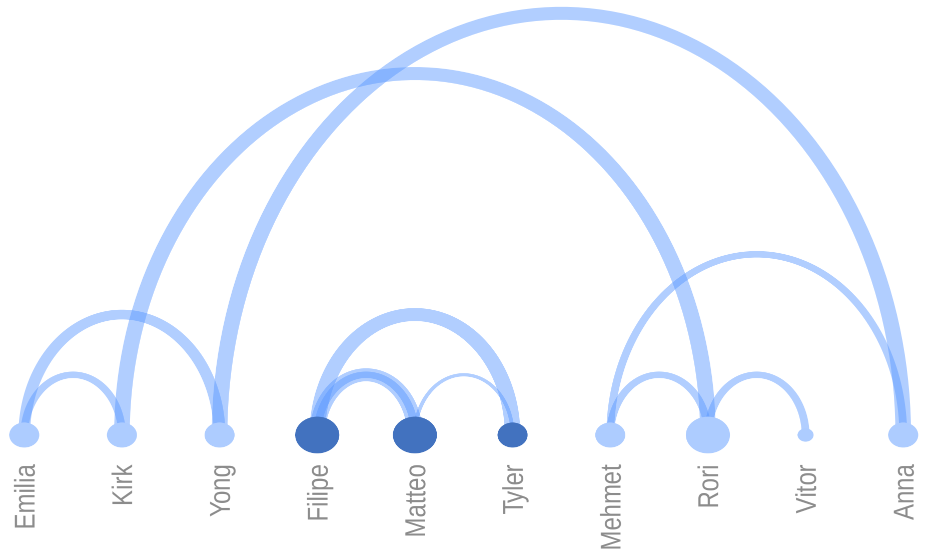
To make our arc diagram more appealing, we need some colors reflecting the cluster memberships:

```
# define two hues of blue
blues = c("#adccff", "#4272bf")

# vector of colors based on cluster membership
cols = blues[gclus$membership]
```

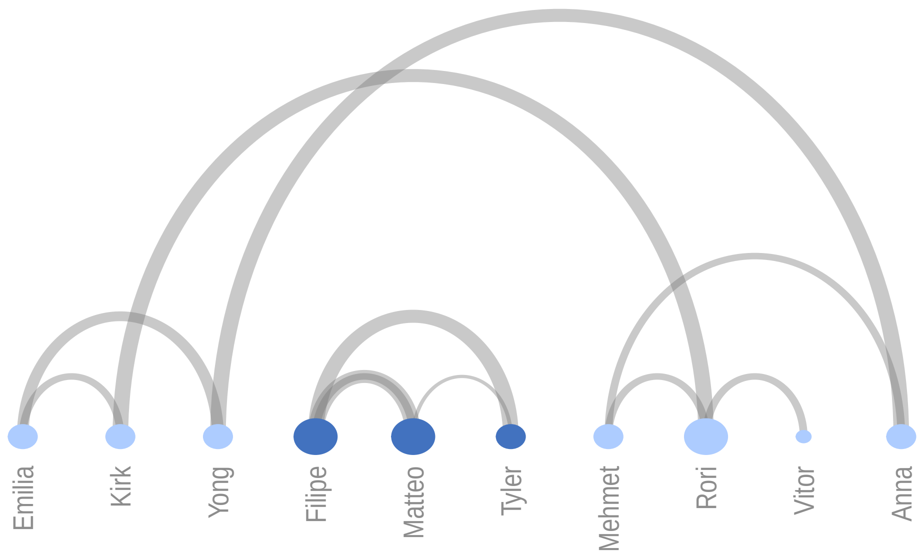
Now that we have all the necessary ingredients, we apply `arcplot()`:

```
# another arc diagram
arcplot(lab, lwd.arcs = 2 * E(glab)$weight, cex.nodes = lab_degree,
    col.nodes = cols, bg.nodes = cols, show.nodes = TRUE)
```



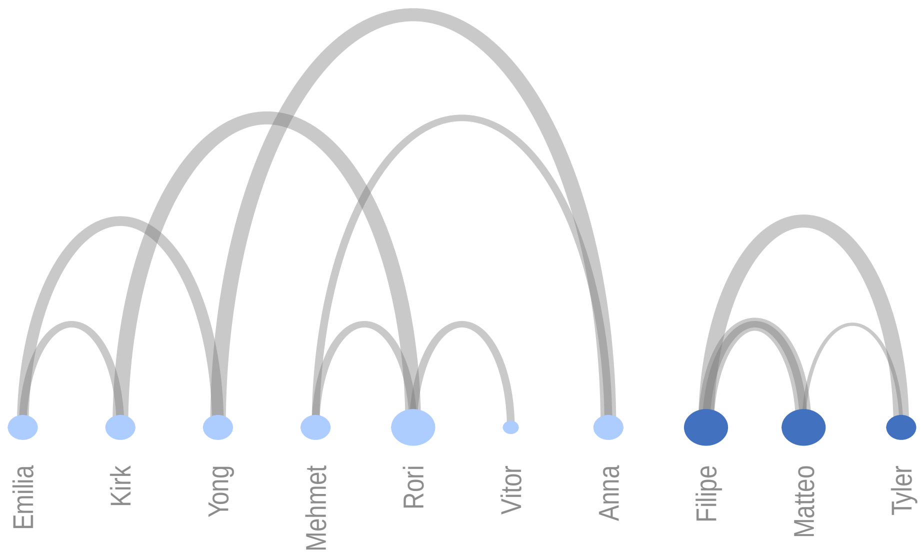
Let's change the colors of the arcs:

```
# another arc diagram
arcplot(lab, lwd.arcs = 2 * E(glab)$weight, col.arcs = "#77777765",
    cex.nodes = lab_degree, col.nodes = cols, bg.nodes = cols,
    show.nodes = TRUE)
```



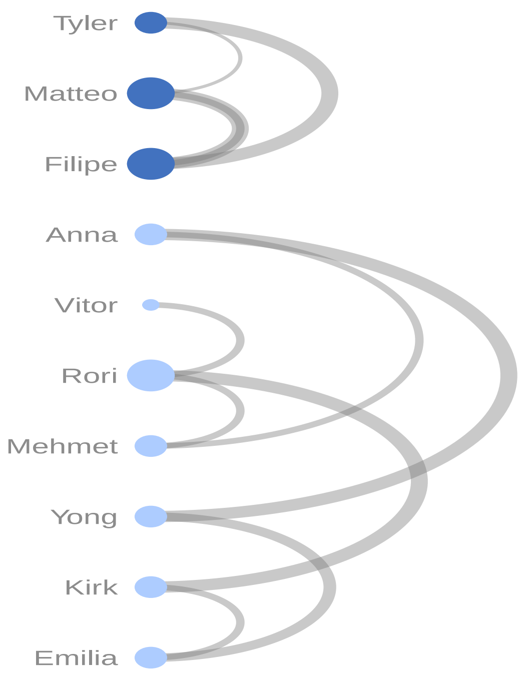
Let's order the nodes according to the clusters:

```
# another arc diagram
arcplot(lab, lwd.arcs = 2 * E(glab)$weight, col.arcs = "#77777765",
    cex.nodes = lab_degree, col.nodes = cols, bg.nodes = cols,
    show.nodes = TRUE, ordering = order(gclus$membership))
```



Let's change the diagram to a vertical orientation

```r
# arc diagram in vertical orientation
arcplot(lab, lwd.arcs = 2 * E(glab)$weight, col.arcs = "#77777765",
    cex.nodes = lab_degree, col.nodes = cols, bg.nodes = cols,
    show.nodes = TRUE, ordering = order(gclus$membership), horizontal = FALSE)
```



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