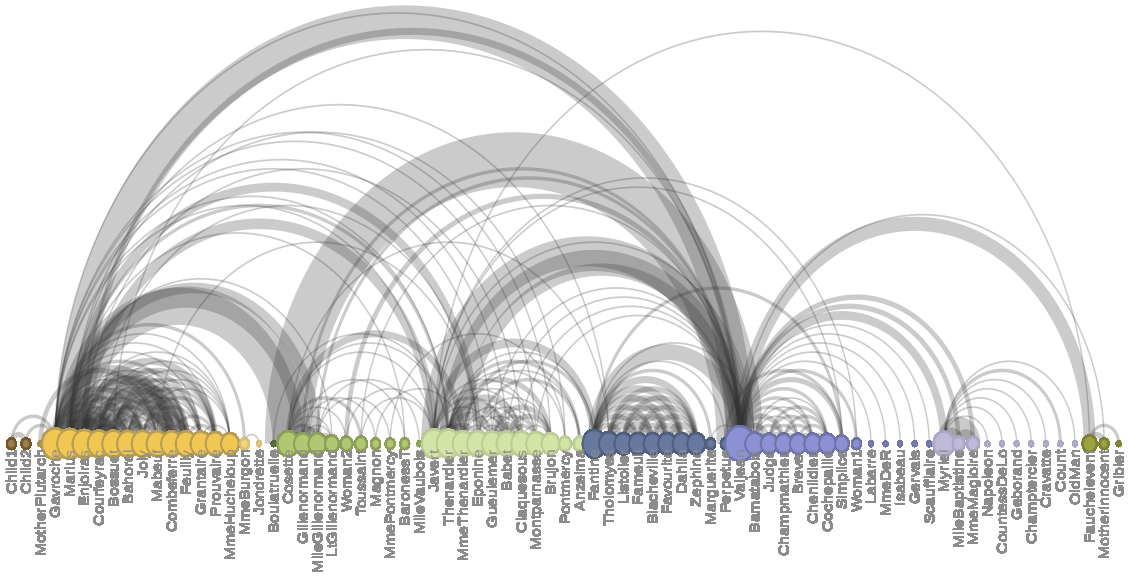
Arc Diagram: *Les Misérables*

Gaston Sanchez
www.gastonsanchez.com



1 Introduction

This document describes the required steps that you'll need to follow to get an arc diagram like the one from *Les Misérables* with the R package `arcdiagram` (a minimalist package designed for plotting pretty arc diagrams).

1.1 Les Misérables

The file for this example is `lesmiserables.gml` which is available at:
https://github.com/gastonstat/arcdiagram
This file is a text file with **GML** format, which is just a type of format for graphs. You can find more information about GML format at:
http://en.wikipedia.org/wiki/Graph_Modelling_Language

Step 1: Read data in R

After downloading the file, you will have to import it in R using the function `read.graph()` with the argument `format = "gml"`. I assume that the file is in your working directory:

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

# read 'gml' file
mis_graph = read.graph("lesmiserables.gml", format="gml")
```

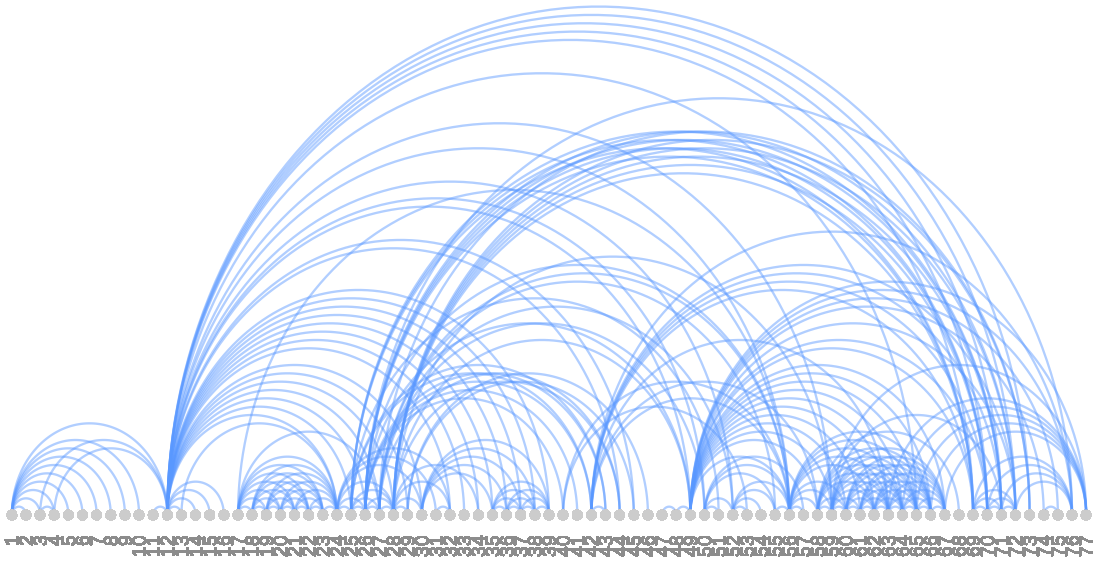
Step 2: Extract edge list

Since we will use the function `arcplot()`, we need an **edgelist**. The good news is that we can use the function `get.edgelist()` to extract it from `mis_graph`:

```r
# get edgelist
edgelist = get.edgelist(mis_graph)
```

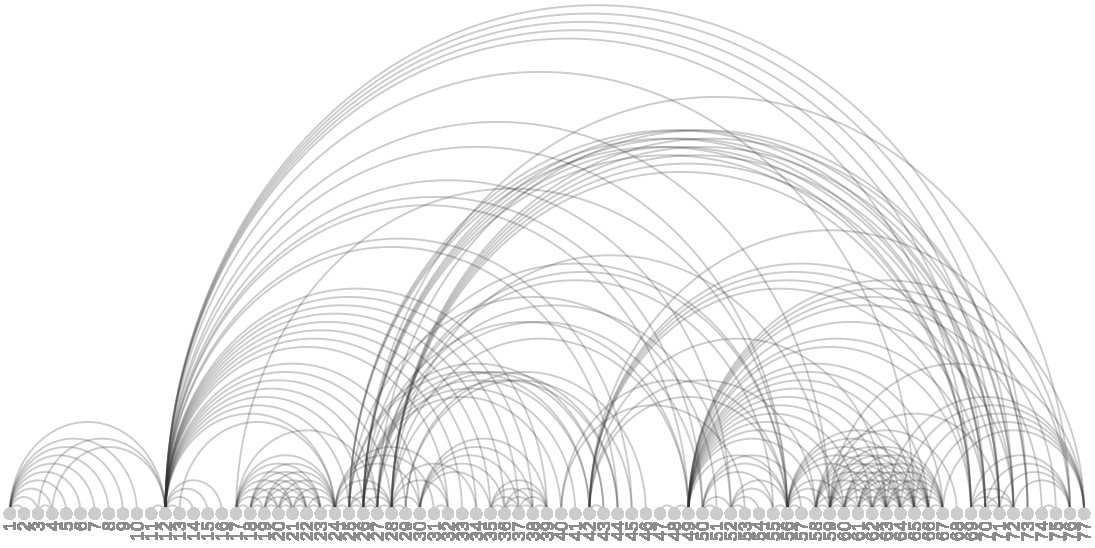
Once we have the `edgelist`, we can try to get a first —very raw— arc diagram with `arcplot()`:

```r
# first plot
arcplot(edgelist)
```



You can see from the previous figure that our first arc diagram has nothing to do with what we are looking for. A better approximation can be obtained if we start tweaking some of the parameters like the symbols of the nodes, the color of the arcs, and their line widths:

```r
# second plot
arcplot(edgelist, cex.labels=0.8,
        show.nodes=TRUE, lwd.nodes = 2, line=-0.5,
        col.arcs = hsv(0, 0, 0.2, 0.25), lwd.arcs = 1.5)
```



Step 3: Information about nodes and edges

Most of the necessary ingredients to create our pretty arc diagram are contained in the graph object `mis_graph`: the fill color of the nodes, the border color of the nodes, the group memberships, the node labels, and the arc widths. If you print `mis_graph` you will see the following output:

```
# what's in mis_graph
mis_graph

## IGRAPH 8814db5 U--- 77 254 --
## + attr: id (v/n), label (v/c), group (v/n), fill (v/c), border
## | (v/c), value (e/n)
## + edges from 8814db5:
##  [1]  1-- 2  1-- 3  1-- 4  3-- 4  1-- 5  1-- 6  1-- 7  1-- 8  1-- 9  1--10
## [11] 11--12  4--12  3--12  1--12 12--13 12--14 12--15 12--16 17--18 17--19
## [21] 18--19 17--20 18--20 19--20 17--21 18--21 19--21 20--21 17--22 18--22
## [31] 19--22 20--22 21--22 17--23 18--23 19--23 20--23 21--23 22--23 17--24
## [41] 18--24 19--24 20--24 21--24 22--24 23--24 13--24 12--24 24--25 12--25
## [51] 25--26 24--26 12--26 25--27 12--27 17--27 26--27 12--28 24--28 26--28
## [61] 25--28 27--28 12--29 28--29 24--30 28--30 12--30 24--31 31--32 12--32
## + ... omitted several edges
```

The first line tells you that `mis_graph` is an undirected graph with 77 nodes and 254 edges (`U--- 77 254 --`). The second and third lines indicate that `mis_graph` has the following attributes (`attr`):

- `id (v/n)`: this is the id of the nodes (numeric)
- `label (v/c)`: this the label of the nodes (character)
- `group (v/n)`: this is the group indicator of nodes (numeric)
- `fill (v/c)`: this is the fill color of the nodes (character)

- `border (v/c)`: this is the border color of the nodes (character)
- `value (e/n)`: this is a value associated to the edges (numeric)

To extract all the data attributes associated with the nodes in the `mis_graph` we have to use the functions `get.vertex.attribute()` and `get.edge.attribute()`:

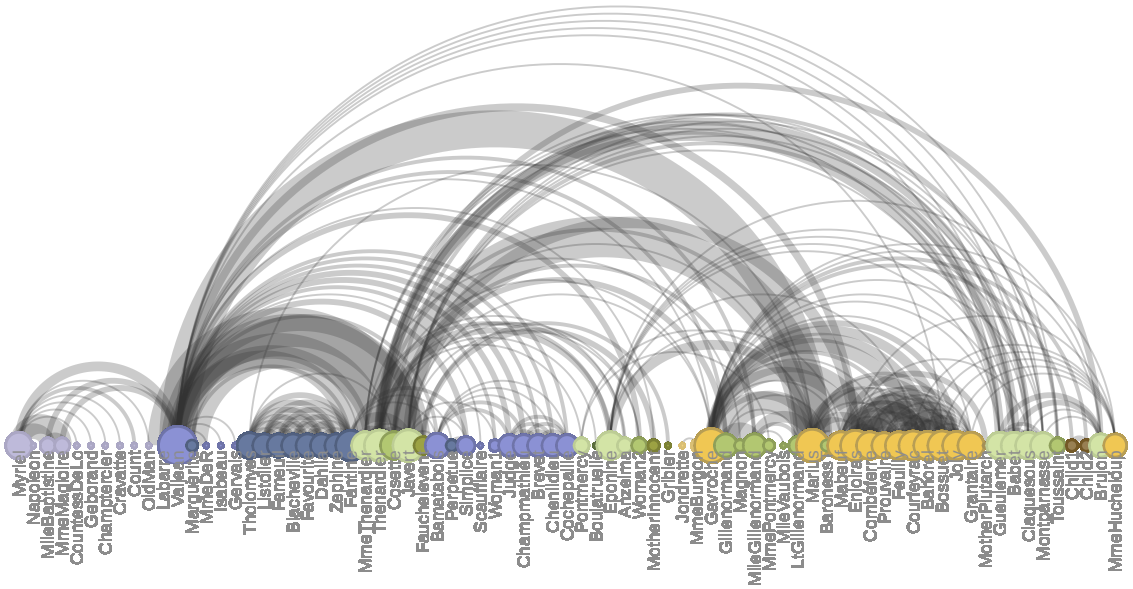
```
# get vertex labels
vlabels = get.vertex.attribute(mis_graph, "label")

# get vertex groups
vgroups = get.vertex.attribute(mis_graph, "group")

# get vertex fill color
vfill = get.vertex.attribute(mis_graph, "fill")

# get vertex border color
vborders = get.vertex.attribute(mis_graph, "border")

# get edges value
values = get.edge.attribute(mis_graph, "value")
```

In addition to the node (i.e. vertices) attributes, we also need to get the degree of the nodes by using the function `degree()`:

```
# get vertex degree
degrees = degree(mis_graph)
```

Ok, let's try a third plot attempt:

```
# third plot
arcplot(edgelist, labels=vlabels, cex.labels=0.8,
        show.nodes=TRUE, col.nodes=vborders, bg.nodes=vfill,
        cex.nodes = log(degrees)+0.5, pch.nodes=21,
        lwd.nodes = 2, line=-0.5,
        col.arcs = hsv(0, 0, 0.2, 0.25), lwd.arcs = 1.5 * values)
```



Step 4: Nodes Ordering

We are very close to our objective but we still need the right ordering for the nodes. One option to get the nodes ordering is by using the package `dplyr` (by Hadley Wickham):

```
# if you haven't installed it
install.packages("dplyr")

# load 'dplyr'
library(dplyr)
```

The idea is to create a data frame with the following variables: `vgroups`, `degrees`, `vlabels`, and a numeric index for the nodes `ind`.

```
# data frame with node attributes
x = data.frame(vgroups, degrees, vlabels, ind=1:vcount(mis_graph))

# take a peek to the data frame
head(x)

##   vgroups degrees        vlabels ind
## 1       1      10         Myriel   1
## 2       1       1       Napoleon   2
## 3       1       3 MlleBaptistine   3
## 4       1       3    MmeMagloire   4
## 5       1       1   CountessDeLo   5
## 6       1       1       Geborand   6
```

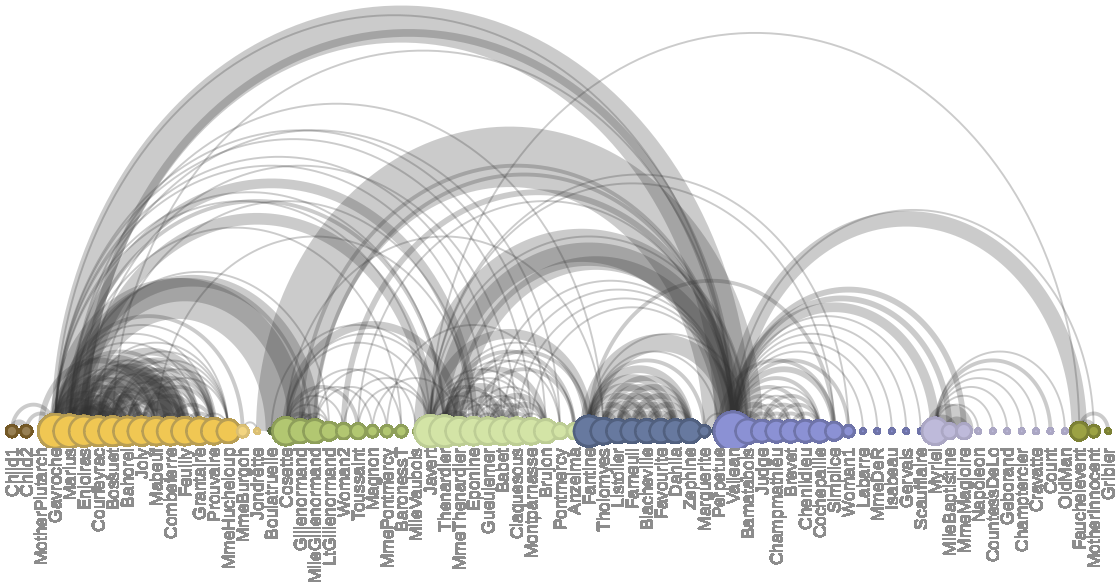
We will arrange the data frame in descending order, first by `vgroups` and then by `degrees`; what we want is the sorted `ind`:

```
# arrange by groups and degree
y = arrange(x, desc(vgroups), desc(degrees))

# what does 'y' look like?
head(y)

##   vgroups degrees        vlabels ind
## 1      10       2         Child1  74
## 2      10       2         Child2  75
## 3       9       1 MotherPlutarch  68
## 4       8      22       Gavroche  49
## 5       8      19         Marius  56
## 6       8      15       Enjolras  59

# get 'ind' ordering
new_ord = y$ind
```

Step 5: Final plot

Now we are ready to produce the desired arc diagram:

```
# plot
arcplot(edgelist, ordering=new_ord, labels=vlabels, cex.labels=0.8,
        show.nodes=TRUE, col.nodes=vborders, bg.nodes=vfill,
        cex.nodes = log(degrees)+0.5, pch.nodes=21,
        lwd.nodes = 2, line=0,
        col.arcs = hsv(0, 0, 0.2, 0.25), lwd.arcs = 1.5 * values)
```



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