Using `igraph` for Visualisations

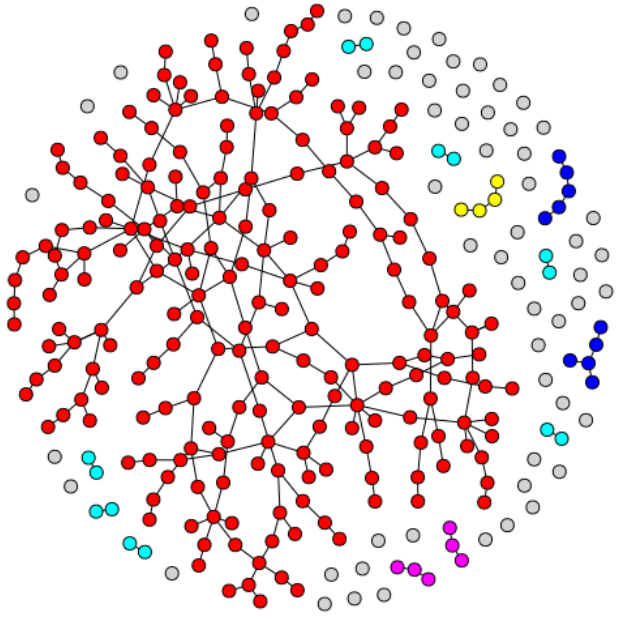
Dr Jamsheed Shorish

The Australian National University
jamsheed.shorish@anu.edu.au

15 July 2013 - Beihang University

Introduction

- `igraph` is a network analysis and visualisation software package, currently for `R` and `Python`.
- It can be found at `igraph.sourceforge.net`.
- For our course, we will be using `igraph` for `R` .



Screenshot from the `igraph` website, depicting an Erdős - Rényi graph and associated colour-coded components, or clusters.

Installing igraph for R

- Installation of igraph for R is very simple–the command is:

```
> install.packages('igraph')
```

- You may need to specify a particular directory if e.g. you don't have privileges to install to the system location for R :

```
> install.packages('igraph', lib='my/
    package/location')
```

- To load the library, use

```
> library('igraph',lib)
```

- or

```
> library('igraph', lib.loc='my/package/
    location')
```

Loading a Dataset

- The first thing to do is to get some data!
- For consistency I'll assume that all data is loaded in `graphml` format.
- This can be exported by the `Python networkx` package.
- To load a network dataset from a file, use:

```
> G = read.graph('network.graphml', format
    ='graphml')
```

- Confirm that your dataset is recognised by `igraph` :

```
> G
IGRAPH D-W- 560 1257 --
\
+ attr: label (v/c), id (v/c), weight (e/n
    ), id (e/c)
```

Layout of a Graph

- To visualise a network well, use the `layout` function of `igraph` to specify the layout prior to plotting.
- Different networks work best with different layouts–this is more art than science.
- As a suggestion, first use `layout.auto` to allow `igraph` to select an appropriate layout.
- For medium-sized networks, a force-directed layout such as `layout.fruchterman-reingold` or `layout.drl` can be used.
- For large connected networks, `layout.lgl` is ideal, or again `layout.drl` can be selected.
- Set a layout with:

```
> layout = layout.auto
```

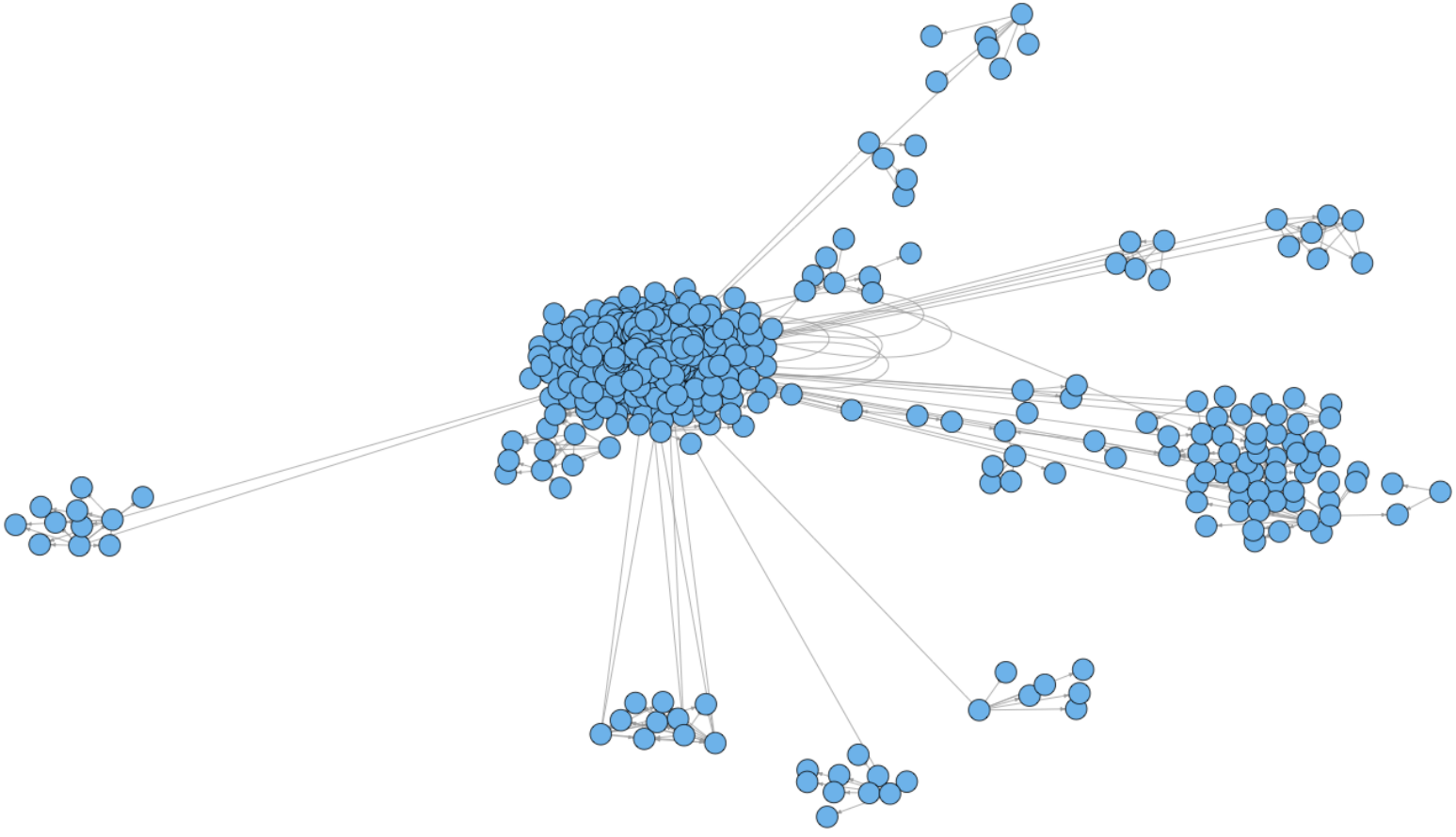
Plotting a Network

- To visualise the network, use the `plot` command.
- This may be *very* slow if you plot to the screen, so one recommendation is to plot to a graphics file instead.
- For example, to plot to a PNG file, use:

```
> png('my_png.png', width = 1600, height
   =900)
> plot(G, layout = layout, vertex.size=3,
   vertex.label=NA, asp=9/16)
> dev.off()
```

Plotting Hashtag Use

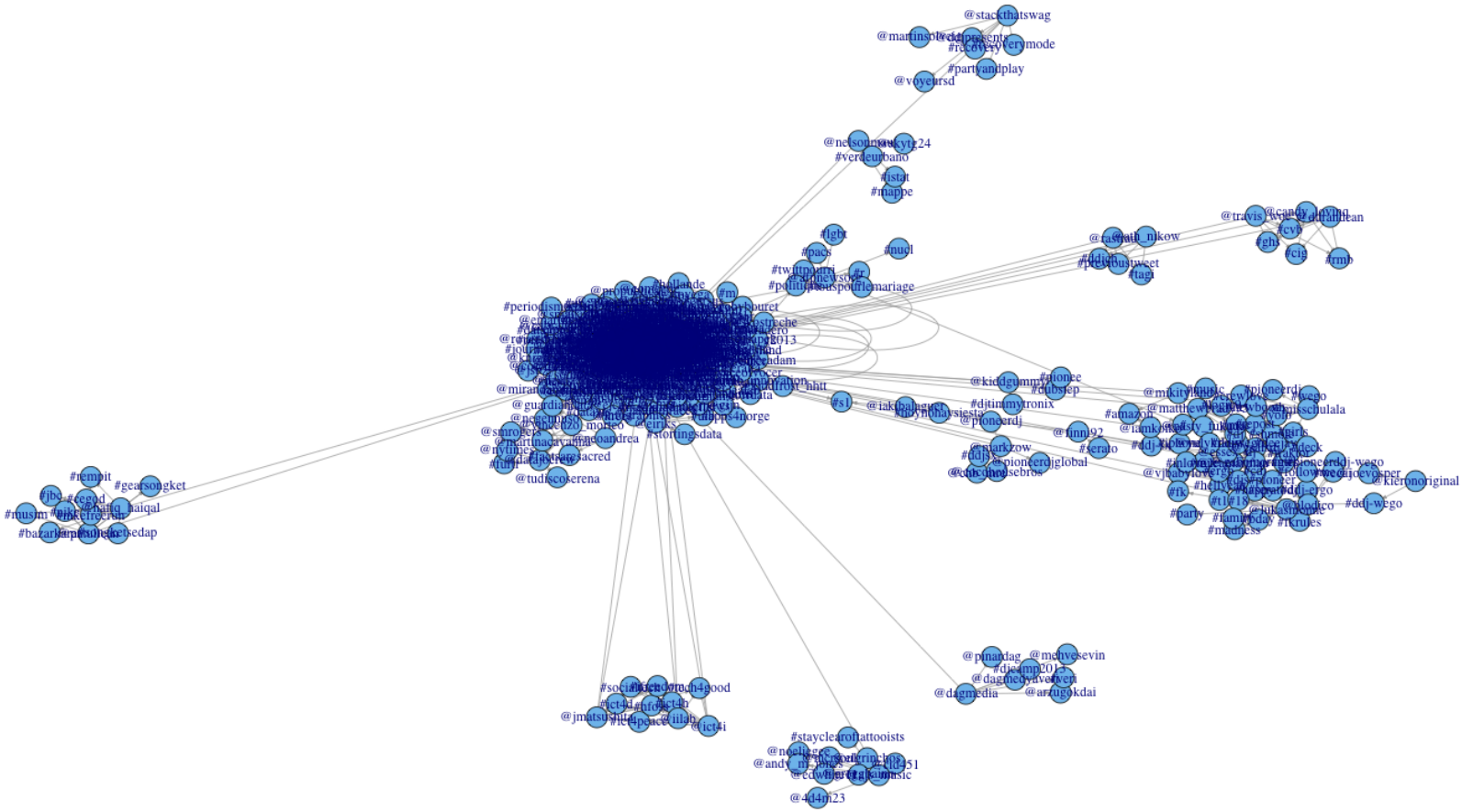
- The following is a plot of the hashtag '#ddj', from SchoolOfData.org, http://schoolofdata.org/2013/04/25/social-network-analysis-for-journalists-using-the-twitter-api.



Using Vertex Labels

- The previous graph had no labels inserted–here is the same figure with labels, using the `V(G)$name` graph attribute:

```
> plot(G, layout = layout, vertex.size=3,
    vertex.label=V(G)$name, asp=9/16)
```



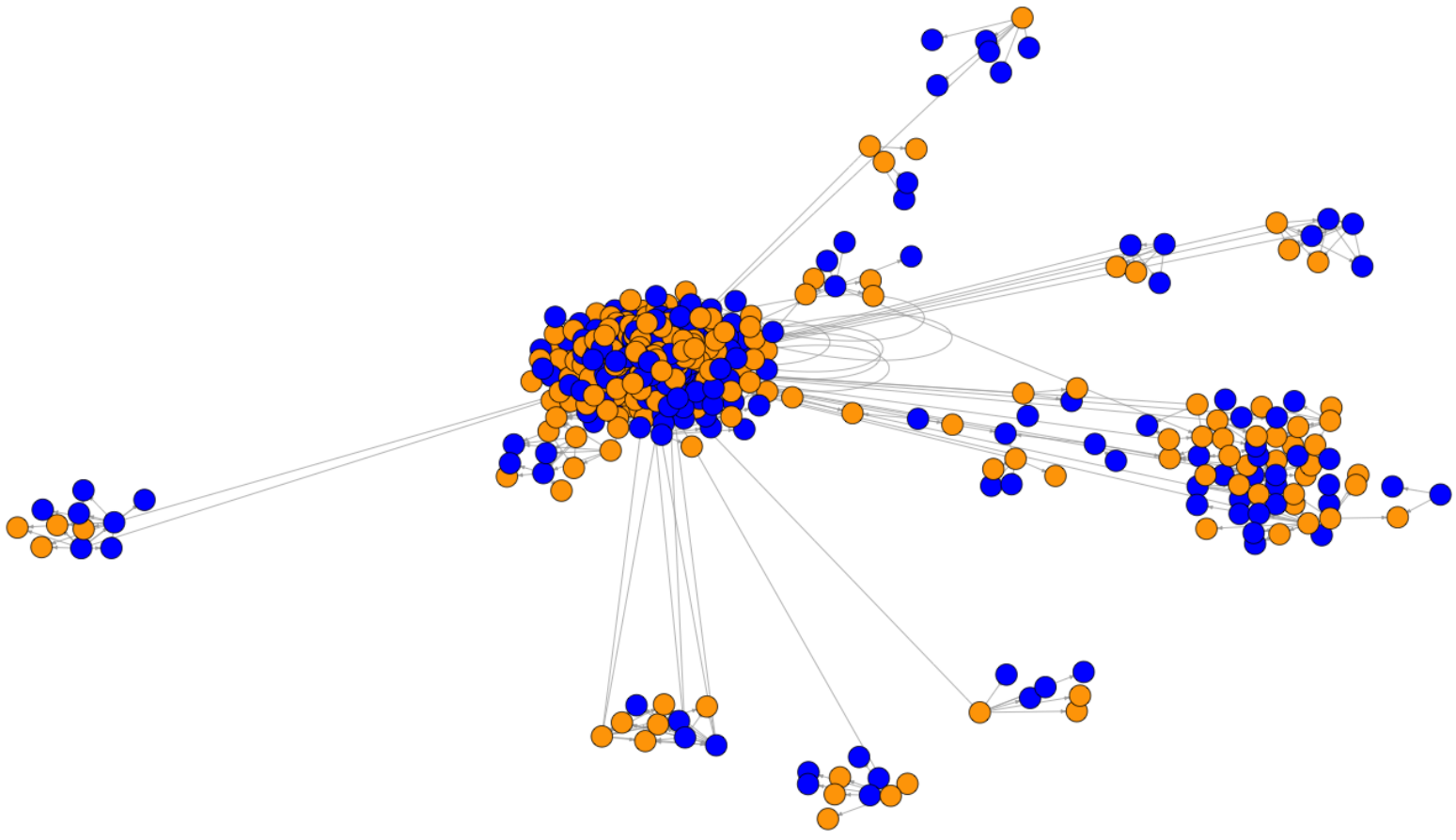
Colouring by Attribute

- Plots can be coloured according to attributes:
- Select colours according to attribute values–for example, if an attribute **emotion** has two values *happy* and *sad*, do:

```
> V(G)$color=ifelse(V(G)$emotion=="happy",
    "blue", "orange")
> plot(G, layout = layout, vertex.size=3,
   vertex.label=NA, asp=9/16)
```

Colouring by Attribute

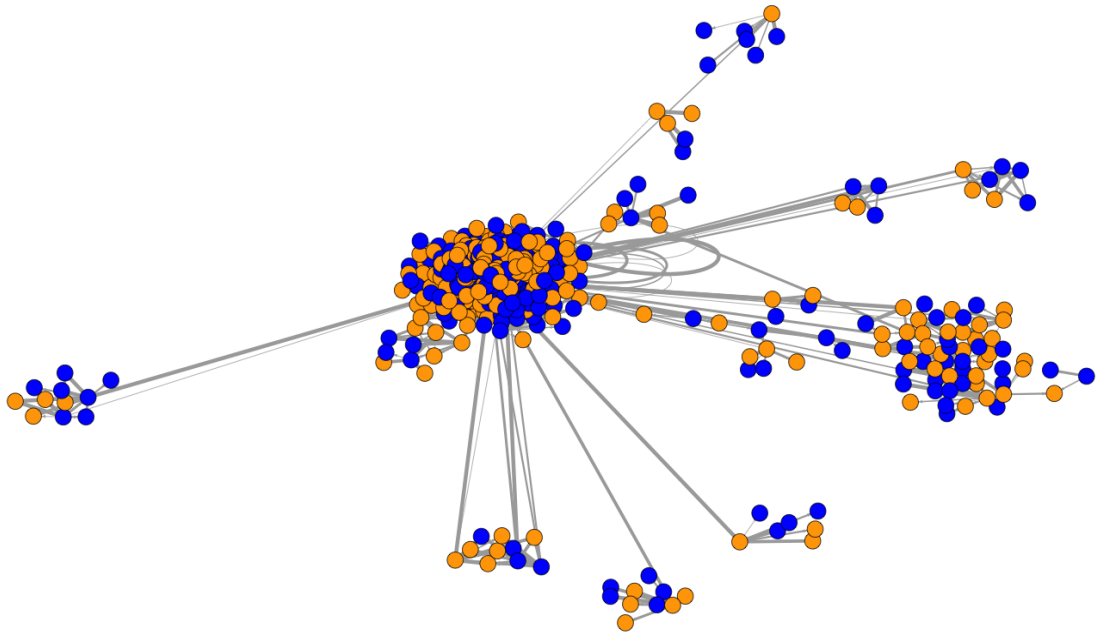
- This will colour happy nodes blue, sad nodes orange, and will suppress vertex labels.



Weighting Edges

- Networks with a `weight` attribute for edges can be plotted with heavier edges for heavier weights:

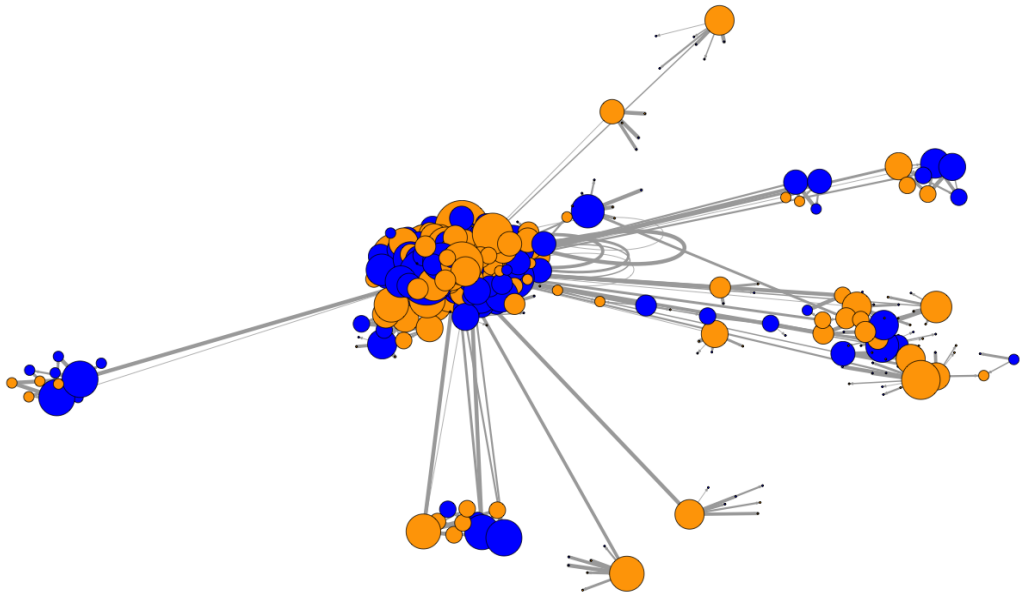
```
> V(G)$color = ifelse(V(G)$emotion == "
    happy", "blue", "orange")
> plot(G, layout = layout, vertex.size=3,
    vertex.label=NA,edge.width=E(G)$weight,
     asp=9/16)
```



Changing Node Size

- The sizes of nodes can also be used to distinguish between different attribute values. For example, we can resize nodes according to degree:

```
> V(G)$size = degree(G)*3
> plot(G, layout = layout, vertex.label=NA
    ,edge.width=E(G)$weight, asp=9/16)
```



More Advanced Plots

- There are many more options to allow for visualisations on the basis of network characteristics.
- For example, if a network has several components, then the components can be coloured differently and visualised:

```
> components = clusters(G)$membership
> colours = sample(rainbow(max(components)
    + 1))
> V(G)$color = colours[components+1]
> plot(G, layout=layout,vertex.label=NA,
   vertex.size=3)
```

Advanced Plots: Clustering

- Here is the result of the previous set of commands for an Erdős Rényi network, similar to the demonstration screenshot from the homepage of `igraph` :