Network Visualization

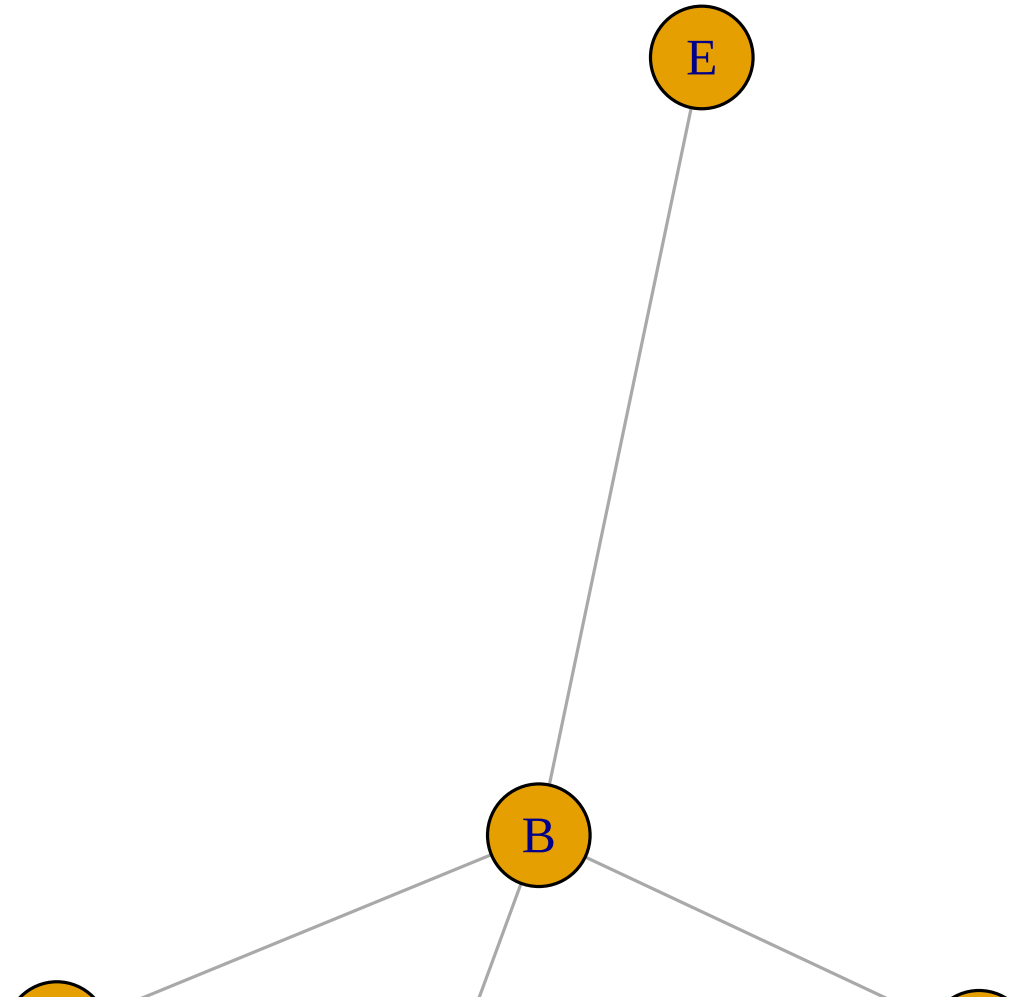
Data Wrangling and Husbandry

03/30/2020

Edges and Vertics

Generally think of graphs as having edges and vertices

- Vertices can be thought of as objects
- Edges can be thought of as relations between objects
  - May be directed or undirected



Social Network Analysis has recently become a subject of considerable interest

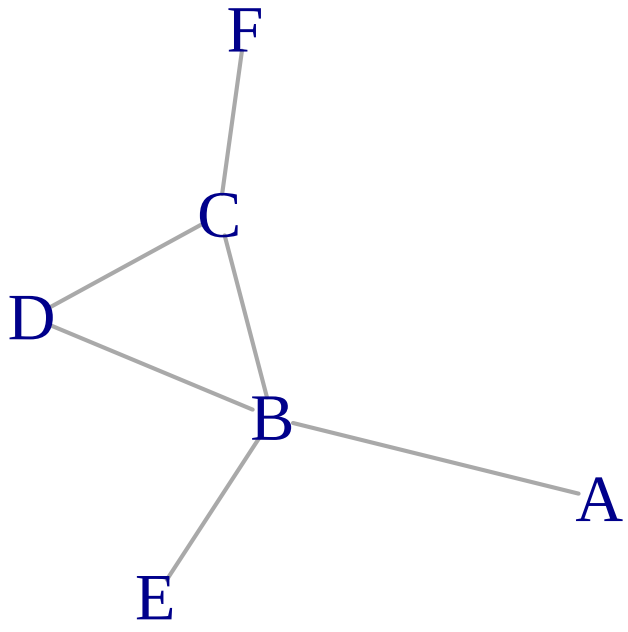
- Social media
- Disease transmission
- Biological networks
- Terrorist networks

There is a nice tutorial on network visualization in R at http://kateto.net/network-visualization by Rutgers's own Katherine Ognyanova. There is also an effort to make a tidy version of network data, called ggnet2

igraph and statnet

- `igraph` is a library of graph visualization functions for Python and R; the R package is simply called `igraph`
- `statnet` is a suite of packages for network analysis (e.g., not just visualization)

Basics



Networks can be represented in multiple ways, including by matrices

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

or as a matrix of edge pairs

```
##   from to
## 1    A  B
## 2    B  C
## 3    B  E
## 4    B  D
## 5    C  D
## 6    C  F
```

igraph representation

igraph had its own representation of graphs. For a small graph, one can import it via graph_from_literal()

```
library(igraph)
(ex_g <- graph_from_literal(
  A -- B,
  B -- C,
  B -- E,
  B -- D,
  D -- C,
  C -- F)
)
```

```
## IGRAPH 87cd3be UN-- 6 6 --
## + attr: name (v/c)
## + edges from 87cd3be (vertex names):
## [1] A--B B--C B--E B--D C--D C--F
```

There are other `graph_from_*()` functions, of which `graph_from_edgelist()` and `graph_from_adjacency_matrix()` are particularly useful.

`as_adjacency_matrix()`, `as_edgelist()` and so on will generate the various data structures.

```
as_adjacency_matrix(ex_g, sparse = FALSE)
```

```
##   A B C E D F
## A 0 1 0 0 0 0
## B 1 0 1 1 1 0
## C 0 1 0 0 1 1
## E 0 1 0 0 0 0
## D 0 1 1 0 0 0
## F 0 0 1 0 0 0
```

Other graph formats

`igraph` allows import and export from a number of common formats, using the `read_graph()` and `write_graph()` functions.

Attributes

Graphs, edges, and vertices all have attributes

```
library(igraphdata)
data(kite)
graph_attr(kite)
```

```
## $name
## [1] "Krackhardt's kite"
##
## $layout
##       [,1] [,2]
##  [1,]    1    4
##  [2,]    1    2
##  [3,]    2    5
##  [4,]    2    3
##  [5,]    2    1
##  [6,]    3    4
##  [7,]    3    2
##  [8,]    4    3
```

```r
vertex_attr_names(kite)
```

```
## [1] "label"     "Firstname" "name"
```

```r
vertex_attr(kite, "Firstname")
```

```
##  [1] "Andre"    "Beverly"  "Carol"    "Diane"    "Ed"
##  [7] "Garth"    "Heather"  "Ike"      "Jane"
```

```r
edge_attr_names(kite)
```

```
## character(0)
```

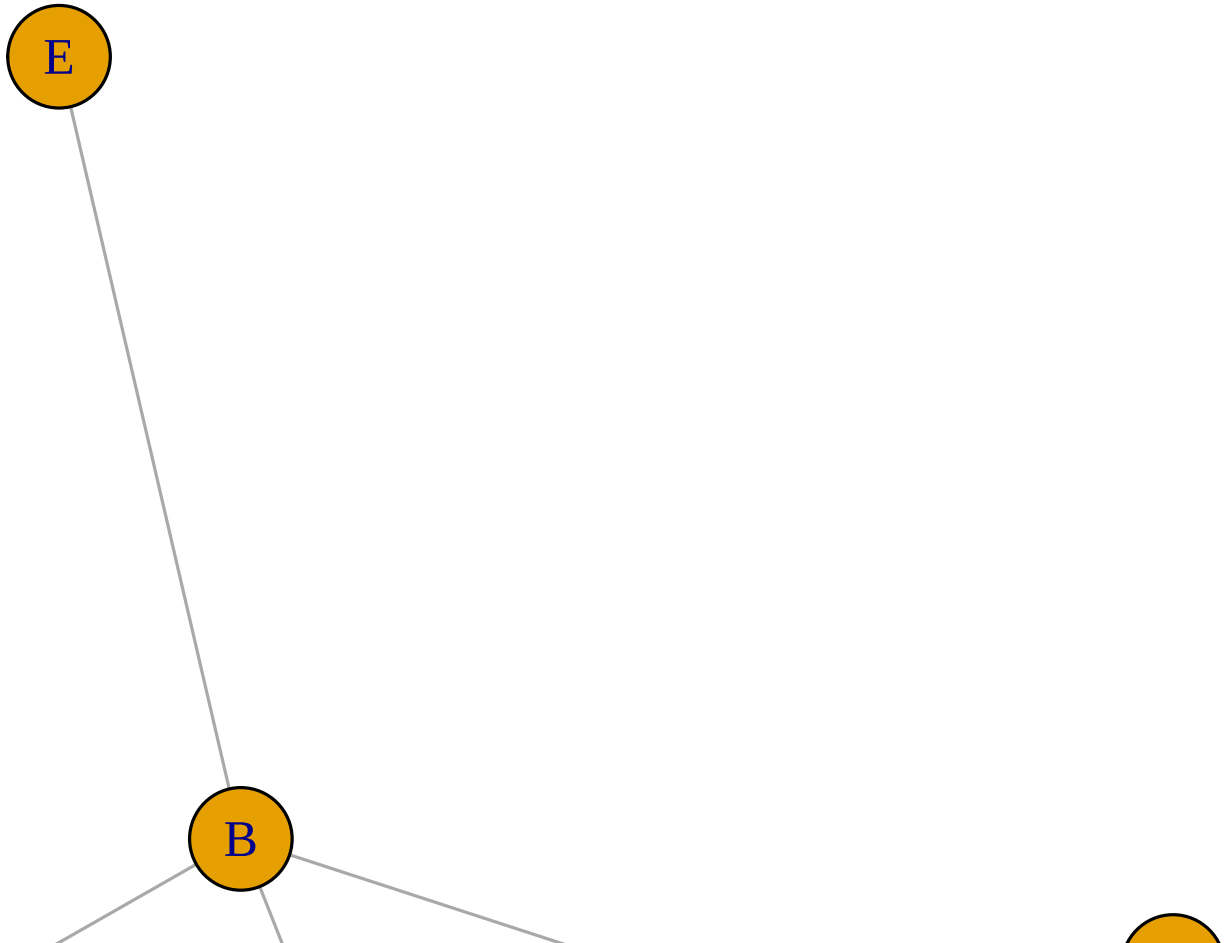
There are various tools to add or delete vertices or edges, to look at neighborhoods, and so on.

Visualization

Since plotting is done using base R and the generic plot function, you will need to look at `?plot.igraph` to get help.

For small graphs, the defaults will work well

```
plot(ex_g)
```



In class exercise

Run the following code and experiment with different plot layouts

```
install.packages("igraphinshiny") # Or install from Tools m
library(igraphinshiny)
plotDemo()
```

An extended example

Inspired by http://varianceexplained.org/r/love-actually-network/

Movie trailer at https://youtu.be/1dYv5u6v55Y

From screenplay to incidence matrix

The screenplay can be found here, but I saved it as a text file.

```
library(stringr)
raw.pnp <- readLines("pride_and_prejudice_screenplay_2005.t

(lines <- data_frame(raw = raw.pnp) %>% filter(raw != ""))
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
## # A tibble: 5,568 x 1
##    raw
##    <chr>
##  1 "PRIDE AND PREJUDICE"
##  2 "                                "
##  3 "                                    Written by"
##  4 "                                  Deborah Moggach"
##  5 "                                "
##  6 "                                "
##  7 "          1. INT. NETHERFIELD - HERTFORDSHIRE - DAY
```

Consider

```
lines[45:52,]
```

```
## # A tibble: 8 x 1
##   raw
##   <chr>
## 1 "           any choice in the matter..."
## 2 "                                "
## 3 "                                "
## 4 "                                "
## 5 "                                "
## 6 "           2."
## 7 "           3 INT. LONGBOURN - CONTINUOUS."
## 8 "           As Elizabeth walks through the hallway, we
```

```
lines <- data_frame(raw = raw.pnp) %>% filter(raw != "") %>
  filter(raw != raw.pnp[3], # blanks
    !str_detect(raw, " {10}[0-9]+\\."), !str_detect(raw, "(
```

```
lines[32:41,]
```

```
## # A tibble: 10 x 1
##    raw
##    <chr>
##  1 "          and father, Mr and Mrs Bennet."
##  2 "                          MRS BENNET"
##  3 "          My dear Mr Bennet, have you heard that"
##  4 "          Netherfield Park is let at last?"
##  5 "          We follow Elizabeth into the house, but st
##  6 "          her parents' conversation."
##  7 "          Do you not want to know who has taken it?'
##  8 "                          MR BENNET"
##  9 "          As you wish to tell me, I doubt I have"
## 10 "          any choice in the matter..."
```

```
lines <- data_frame(raw = raw.pnp) %>%
    filter(raw != "", !str_detect(raw, " {10}[0-9]+\\."),
      raw != raw.pnp[3], !str_detect(raw, "CONT'D")) %>%
    mutate(is_scene = str_detect(raw, " {10}[0-9]+ [A-Z]{2,
           scene = cumsum(is_scene))

with(lines, head(data.frame(line = substr(str_trim(raw), 1,
```

```
##          line scene
## 1 PRIDE AND       0
## 2 Written by      0
## 3 Deborah Mo      0
## 4 1 INT. NET      1
## 5 A vast man      1
## 6 off furnit      1
```

Finally, strip away everything but the full caps lines

```
lines <- data_frame(raw = raw.pnp) %>%
    filter(raw != "", !str_detect(raw, " {10}[0-9]+\\."),
      raw != raw.pnp[3], !str_detect(raw, "CONT'D")) %>%
    mutate(is_scene = str_detect(raw, " {10}[0-9]+ [A-Z]{2
           scene = cumsum(is_scene)) %>%
    filter(!is_scene, scene > 0) %>%
    filter(str_detect(raw, " {25}[A-Z .1-9]{2,}")) %>%
    select(-is_scene)

lines <- rename(lines, character = raw)
lines <- mutate(lines, character = str_trim(character))
```

```
sort(unique(lines$character))
```

```
##  [1] "ACQUAINTANCE\""   "ASKS:"              "BAMBOOZLED)"
##  [5] "BINGLEY)"         "BUTLER"             "CAROLINE BIN
##  [9] "CHILD-"           "COLLINS"            "COURSE -"
## [13] "CUT TO;"          "CUT TO:"            "DANCING PART
## [17] "DARCY -"          "DARCY ELIZABETH"    "DOCTOR"
## [21] "ELISABETH"        "ELIZABETH"          "ELIZABETH'"
## [25] "FOOTMAN"          "GEORGIANA"          "GEORGIANA)"
## [29] "IMPERTINENCE --"  "IS"                 "JANE"
## [33] "LADY CATHERINE"   "LIZZIE"             "LIZZIE_"
## [37] "LYDIA -"          "MAID"               "MARY"
## [41] "MR BENNET"        "MR BENNET'"         "MR BINGLEY"
## [45] "MR DARCY"         "MR GARDINER"        "MR KENNET"
## [49] "MRS BENNET"       "MRS GARDINER"       "MRS HILL"
## [53] "NICELY-"          "P C"                "PARTNER ME_'
## [57] "PLAYING)"         "POSTMASTER"         "RECOVERING)"
## [61] "ROOM:"            "SERVANT"            "SIR WILLIAM"
## [65] "SUBJECT-"         "TITLE:"             "WICKHAM"
```

```
drop_characters <- c("ACQUAINTANCE\"", "ASKS:", "BAMBOOZLE
lines <- lines %>% filter(!(character %in% drop_characters)
sort(unique(lines$character))
```

```
##  [1] "BINGLEY"          "BINGLEY)"         "BUTLER"
##  [5] "CHARLOTTE"        "COLLINS"          "DARCY"
##  [9] "DARCY ELIZABETH"  "ELISABETH"        "ELIZABETH"
## [13] "FITZWILLIAM"      "FOOTMAN"          "GEORGIANA"
## [17] "HER HUSBAND"      "JANE"             "KITTY"
## [21] "LIZZIE"           "LIZZIE_"          "LYDIA"
## [25] "MAID"             "MARY"             "MISS BINGLE
## [29] "MR BENNET'"       "MR BINGLEY"       "MR COLLINS"
## [33] "MR GARDINER"      "MR KENNET"        "MR WICKHAM"
## [37] "MRS GARDINER"     "MRS HILL"         "MRS REYNOLD
## [41] "SERVANT"          "SIR WILLIAM"      "WICKHAM"
```

```r
lines$character <- lines$character %>%
  str_replace_all("[-']", "") %>% str_replace_all("_", "") %>%
  str_replace_all("\\)", "") %>% str_trim()
```

##{.smaller}

```
sort(unique(lines$character))
```

```
##  [1] "BINGLEY"          "BUTLER"           "CAROLINE BIN
##  [5] "COLLINS"          "DARCY"            "DARCY ELIZAB
##  [9] "ELIZABETH"        "FITZWILLIAM"      "FOOTMAN"
## [13] "HER HUSBAND"      "JANE"             "KITTY"
## [17] "LIZZIE"           "LYDIA"            "MAID"
## [21] "MISS BINGLEY"     "MR BENNET"        "MR BINGLEY"
## [25] "MR DARCY"         "MR GARDINER"      "MR KENNET"
## [29] "MRS BENNET"       "MRS GARDINER"     "MRS HILL"
## [33] "POSTMASTER"       "SERVANT"          "SIR WILLIAM"
## [37] "WOMAN"
```

```
lines$character[lines$character == "MR DARCY"] <- "DARCY"
lines$character[lines$character == "MR BINGLEY"] <- "BINGLE
lines$character[lines$character == "ELISABETH"] <- "ELIZABE
lines$character[lines$character == "MISS BINGLEY"] <- "CARO
lines$character[lines$character == "LIZZIE"] <- "ELIZABETH"
lines$character[lines$character == "COLLINS"] <- "MR COLLIN
```

The data is now almost ready to work with

```
by_character_scene <- lines %>% count(scene, character)
by_character_scene
```

```
## # A tibble: 314 x 3
##    scene character      n
##    <int> <chr>      <int>
##  1     2 MR BENNET      1
##  2     2 MRS BENNET     1
##  3     3 JANE           1
##  4     3 KITTY          1
##  5     3 LYDIA          2
##  6     3 MR BENNET      2
##  7     3 MRS BENNET     2
##  8     4 MRS BENNET     1
##  9     5 ELIZABETH      1
## 10     5 JANE           1
## # ... with 304 more rows
```

```
character_scene_matrix <-
   table(unique(lines[, c("character", "scene")]))
head(character_scene_matrix)
```

```
##                     scene
## character            2 3 4 5 6 7 8 11 14 16 18 19 20 21 22
##   BINGLEY            0 0 0 0 1 0 0  0  0  0  0  1  1  1  0
##   BUTLER             0 0 0 0 0 0 0  0  0  0  0  0  0  0  0
##   CAROLINE BINGLEY   0 0 0 0 1 0 0  0  0  0  1  0  0  1  0
##   CHARLOTTE          0 0 0 0 1 0 0  0  0  0  0  0  0  0  0
##   DARCY              0 0 0 0 1 0 0  0  0  0  1  0  0  0  0
##   ELIZABETH          0 0 0 1 1 1 1  0  1  1  1  0  1  0  0
##                     scene
## character           35 36 37 38 39 40 41 42 43 44 45 46 4
##   BINGLEY            0  0  0  0  0  0  0  1  0  0  0  1
##   BUTLER             0  0  0  0  0  0  0  0  0  0  0  0
##   CAROLINE BINGLEY   0  0  0  0  0  0  0  0  0  0  0  1
##   CHARLOTTE          1  0  0  0  0  0  0  0  1  0  0  1
##   DARCY              0  0  0  0  0  0  0  0  0  0  0  1
##   ELIZABETH          1  0  1  0  1  1  1  1  1  1  1  1
```

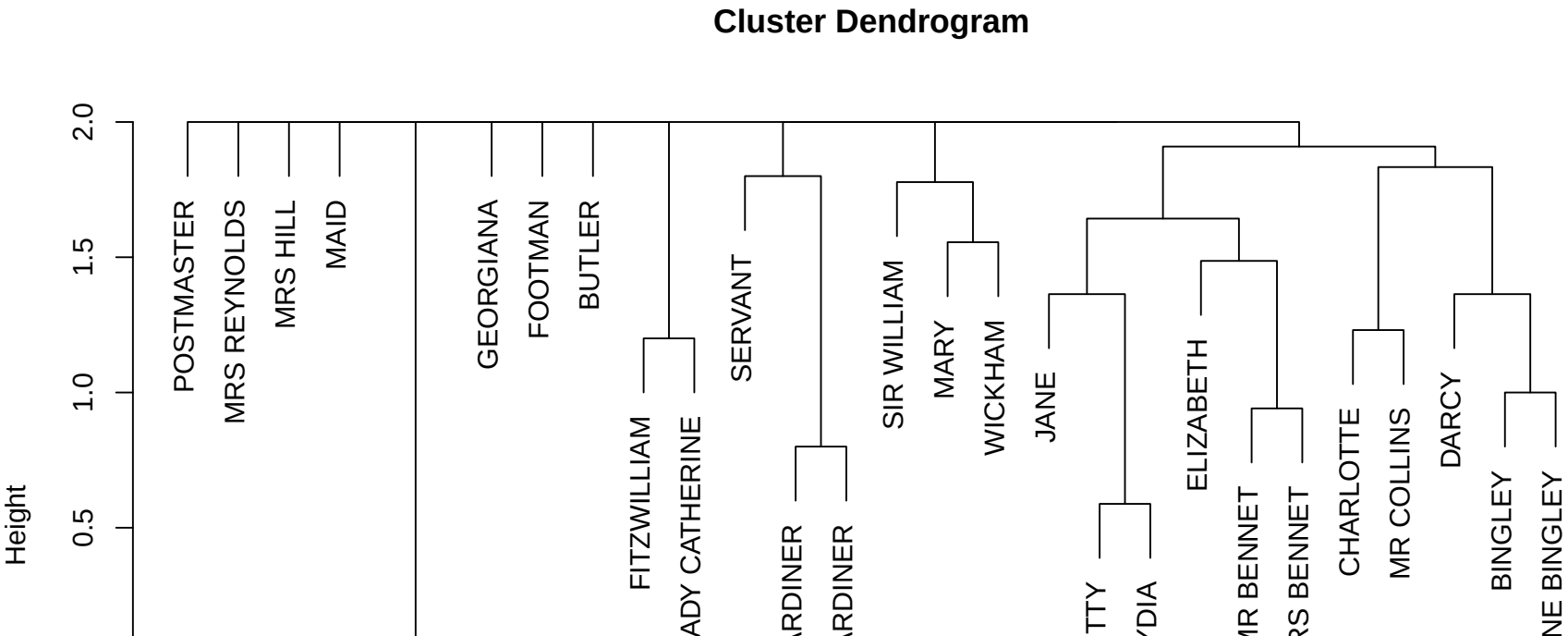
```r
character_adjacency <- character_scene_matrix %*% t(charact
head(character_adjacency)
```

```
##                    character
## character           BINGLEY BUTLER CAROLINE BINGLEY CHARI
##   BINGLEY                12      1                 6
##   BUTLER                  1      2                 1
##   CAROLINE BINGLEY        6      1                 8
##   CHARLOTTE               2      0                 2
##   DARCY                   7      2                 7
##   ELIZABETH               9      1                 7
##                    character
## character           FITZWILLIAM FOOTMAN GEORGIANA HER HUS
##   BINGLEY                     0       0         0
##   BUTLER                      0       0         0
##   CAROLINE BINGLEY            0       1         1
##   CHARLOTTE                   2       0         0
##   DARCY                       2       1         1
##   ELIZABETH                   3       1         1
##                    character
```

Clustering

Not all network visualizations are plots of edges and vertices. For example, we could try clustering.
https://en.wikipedia.org/wiki/Dendrogram

```
character_norm <- character_scene_matrix / rowSums(characte

character_hclust <- hclust(dist(character_norm, method = "m

plot(character_hclust)
```

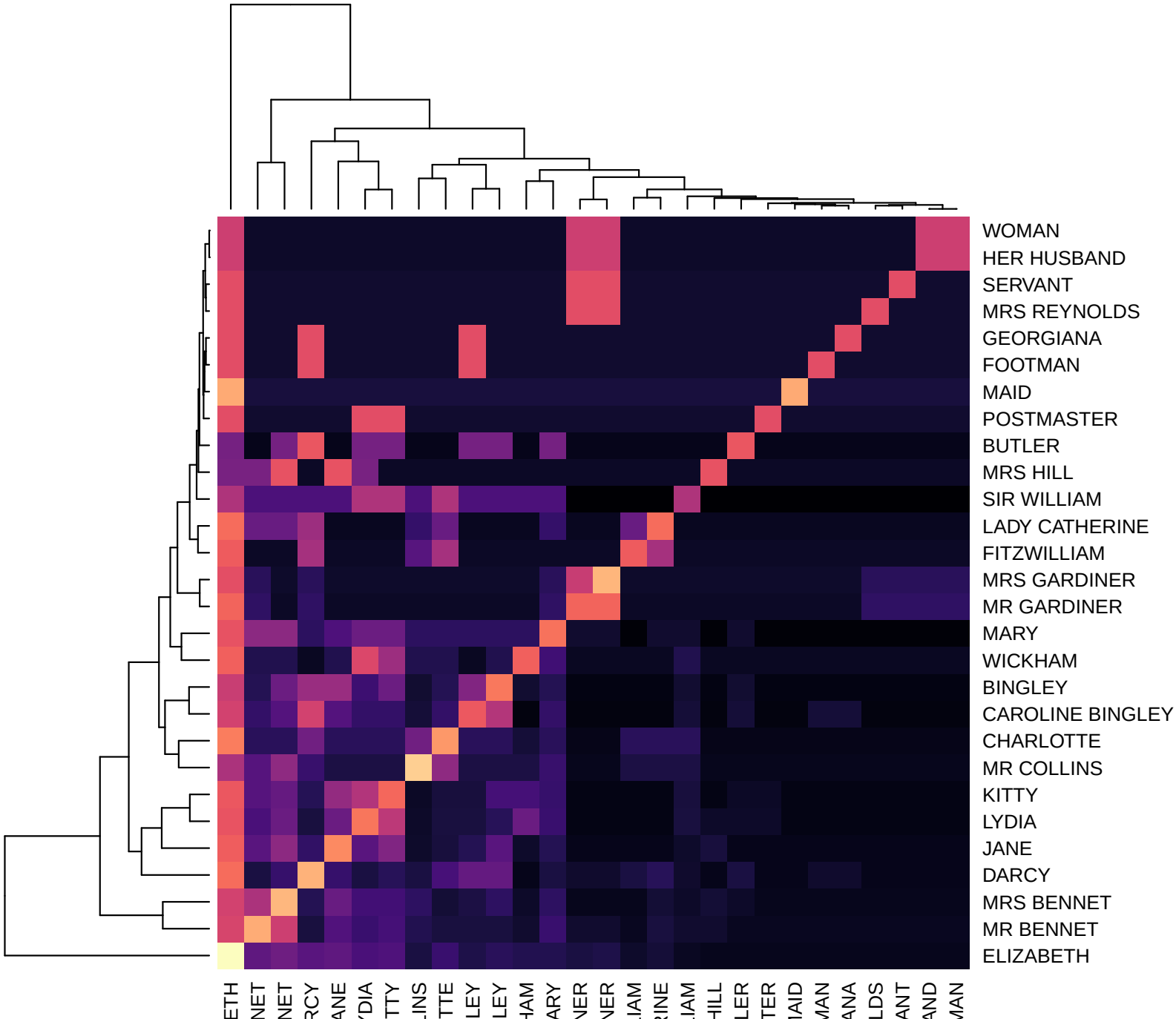


Heatmap

Or a heatmap

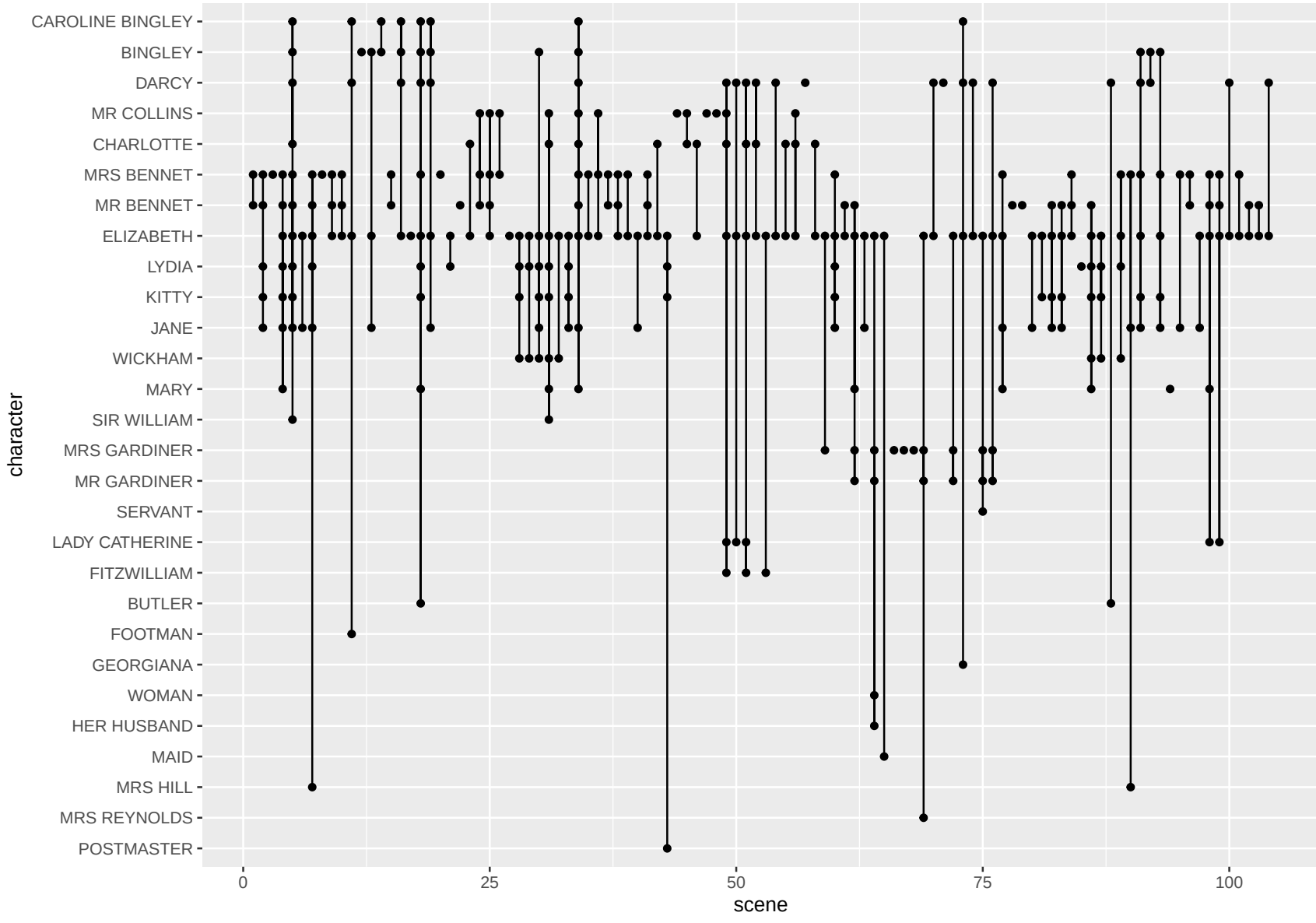
```
library(viridis) # for colors
heatmap(character_adjacency, col = viridis(n = 256, alpha =
```

```
## Loading required package: viridisLite
```



Or maybe a timeline of scenes

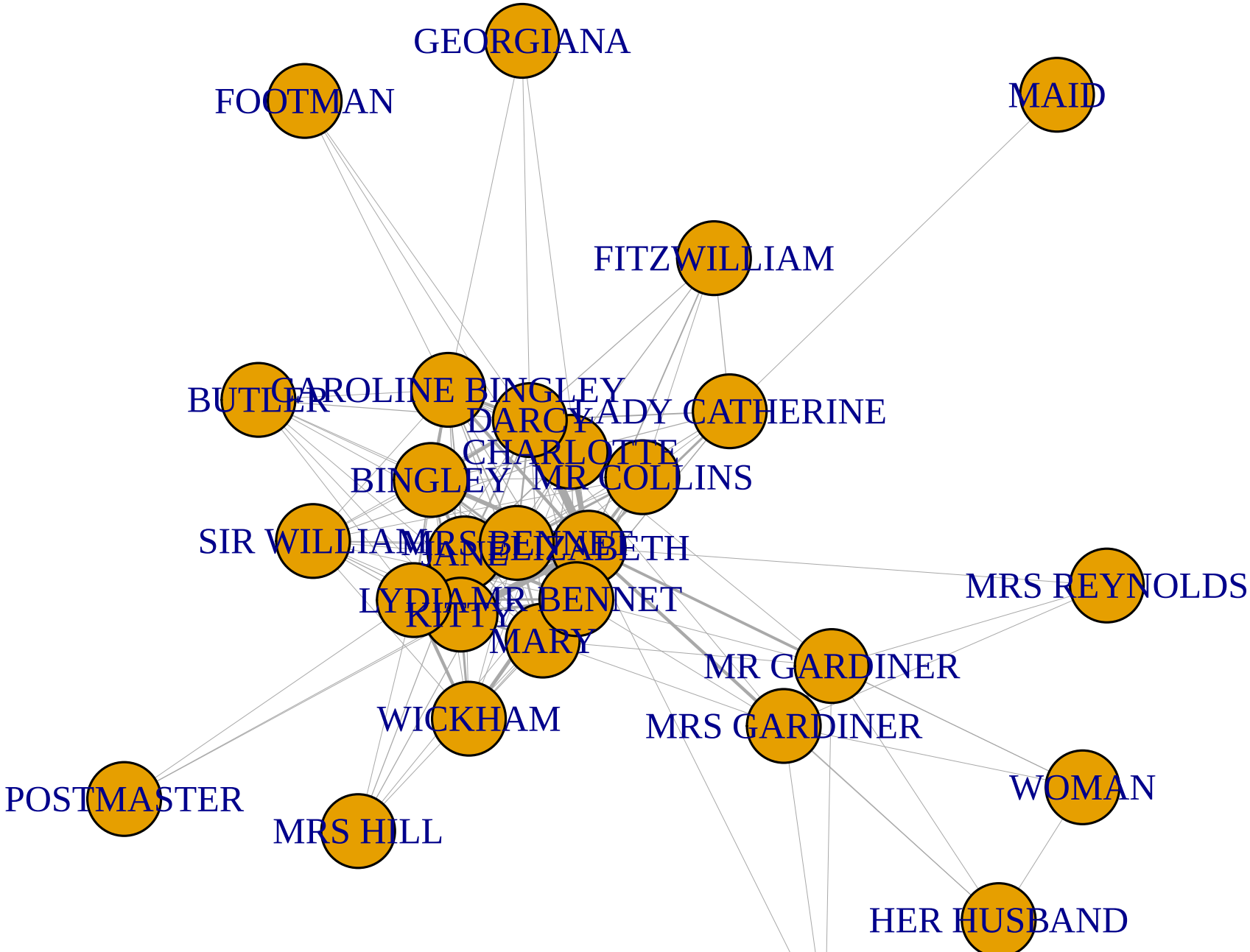
```
character_ordering <- character_hclust$labels[character_hcl

scenes <- by_character_scene %>%
    filter(n() > 1) %>%          # scenes with > 1 character
    ungroup() %>%
    mutate(scene = as.numeric(factor(scene)),
           character = factor(character, levels = characte

ggplot(scenes, aes(scene, character)) +
    geom_point() +
    geom_path(aes(group = scene))
```


CAROLINE BINGLEY
BINGLEY
DARCY
MR COLLINS
CHARLOTTE
MRS BENNET
MR BENNET
ELIZABETH
LYDIA
KITTY
JANE
WICKHAM
MARY
SIR WILLIAM
MRS GARDINER
MR GARDINER
SERVANT
LADY CATHERINE
FITZWILLIAM
BUTLER
FOOTMAN
GEORGIANA
WOMAN
HER HUSBAND
MAID
MRS HILL
MRS REYNOLDS
POSTMASTER
character
0
25
50
75
100
scene

And now some vertices-and-edges

```
character_graph <- graph.adjacency(character_adjacency,
      weighted = TRUE, mode = "undirected", diag = FALSE)
character_graph2 <- delete_vertices(character_graph,
      V(character_graph)[diag(character_adjacency) <= 10])
## You can extract information about vertices with V() and
## edges with E()
```

```
plot(character_graph, edge.width = E(character_graph)$weigh
     layout = layout_with_fr)
```



```
plot(character_graph2, edge.width = E(character_graph2)$we
     layout = layout_with_fr)
```