# IMDB_Movie_Links

```r
devtools::install_github("briatte/ggnet")
```

```
## Skipping install of 'ggnet' from a github remote, the SHA1 (da9a7cf2) has not changed since last ins
##   Use `force = TRUE` to force installation
```

```r
library(ggnet)
```

```
##
## Attaching package: 'ggnet'

## The following objects are masked from 'package:GGally':
##
##     ggnet, ggnet2
```

```r
library(network)
library(sna)
```

## Cleaning the Data

```r
imdb = read.csv("imdb.csv", header = T, na.strings=c("","NA"))
awards = read.csv("awards.csv", header = T, na.strings = c("", "NA"))
cpi = read.csv("cpi.csv", header = T)
colnames(awards) = tolower(colnames(awards))

names(cpi) = c("year", "cpi")
colMeans(is.na(imdb))*100
```

```
##                     color             director_name
##                0.37675987                2.06226453
##    num_critic_for_reviews                  duration
##                0.99147333                0.29744200
##   director_facebook_likes    actor_3_facebook_likes
##                2.06226453                0.45607773
##              actor_2_name    actor_1_facebook_likes
##                0.25778307                0.13880627
##                     gross                    genres
##               17.52924846                0.00000000
##              actor_1_name               movie_title
##                0.13880627                0.00000000
##            num_voted_users cast_total_facebook_likes
##                0.00000000                0.00000000
##              actor_3_name      facenumber_in_poster
##                0.45607773                0.25778307
##             plot_keywords           movie_imdb_link
##                3.03390839                0.00000000
##      num_user_for_reviews                  language
##                0.41641880                0.23795360
##                   country            content_rating
##                0.09914733                6.00832838
##                    budget                title_year
##                9.75609756                2.14158239
##    actor_2_facebook_likes                imdb_score
##                0.25778307                0.00000000
##              aspect_ratio      movie_facebook_likes
##                6.52389451                0.00000000
```

```
imdb = na.omit(imdb)
imdb = imdb %>% select(gross, genres, movie_title, country, movie_imdb_link, budget, title_year, imdb_s

names(imdb) = c("gross", "genres", "title", "country","links", "budget", "year", "score", "rating")

#simplify the genres by taking the first entry
imdb$genres = as.character(imdb$genres)
imdb$genres_simple = strsplit(imdb$genres, split = "|", fixed = TRUE)
imdb$genres_simple = as.character(imdb$genres_simple)
imdb$genres_simple = str_extract(imdb$genres, pattern = "^[A-Za-z]{1,20}")
imdb$genres_simple = as.factor(imdb$genres_simple)
imdb$title = gsub(imdb$title, pattern = "?", replacement ="")
imdb$links = as.character(imdb$links)

imdb$genres = as.factor(imdb$genres)
imdb$budget = as.numeric(imdb$budget)
imdb$gross = as.numeric(imdb$gross)
imdb$score = as.numeric(imdb$score)
imdb$rating = as.factor(imdb$rating)

# imdb$year = as.Date(imdb$year,"%Y")
# cpi$year = as.Date(imdb$year, "%Y")

#How many genres are there?
levels(imdb$genres_simple)
```

```
##  [1] "Action"      "Adventure"   "Animation"   "Biography"   "Comedy"
##  [6] "Crime"       "Documentary" "Drama"       "Family"      "Fantasy"
## [11] "Horror"      "Musical"     "Mystery"     "Romance"     "Sci"
## [16] "Thriller"    "Western"
```

```
link_pat = '(.){35}'
imdb$links = str_extract(imdb$links, pattern = link_pat)
imdb = inner_join(imdb, cpi, by = "year")
```

## Convert all dollars to 2016 dollars

```
reference_year_cpi = filter(imdb, year ==2016)$cpi[1]
```

```r
imdb$gross_adj = reference_year_cpi / imdb$cpi

imdb= imdb %>%
  group_by(year) %>%
  mutate(cpi_ratio=  reference_year_cpi/cpi) %>%
  mutate(gross_adj = gross*cpi_ratio) %>%
  mutate(budget_adj = budget*cpi_ratio) %>%
  select(-gross, -budget) %>%
  ungroup() %>%
  filter( country == 'USA')


year_pat = '^[0-9]{4}'

awards$year = str_extract(awards$year, pattern = year_pat) %>%as.numeric()
awards = na.omit(awards)
```

## Network Analysis of Linked Movies

On imdb.com, for each movie, there are 12 recommended movies. For each movie in this imdb data set, I went to imdb.com and pulled down the 12 recommended movie links. That's what this below function does.

```r
get_links <- function(address) {
  # read the movie page
  page <- readLines(address)
  # find the lines with the recommendations and strip the unneeded stuff
  recs <- page[grep("rec_item", page)]
  recs <- unlist(strsplit(recs, "data-tconst="))[seq(from = 2, to = 24, by = 2)]
  # return the codes
  recs <- paste("tt", gsub("[^0-9]", "", recs), sep = "")

  recs = paste("http://www.imdb.com/title/", recs, sep = "")
  return(recs)
}

#Example

get_links(imdb$links[1])
```

```
##  [1] "http://www.imdb.com/title/tt1392170"
##  [2] "http://www.imdb.com/title/tt0120338"
##  [3] "http://www.imdb.com/title/tt1454468"
##  [4] "http://www.imdb.com/title/tt0454876"
##  [5] "http://www.imdb.com/title/tt1010048"
##  [6] "http://www.imdb.com/title/tt3659388"
##  [7] "http://www.imdb.com/title/tt0416449"
##  [8] "http://www.imdb.com/title/tt1951264"
##  [9] "http://www.imdb.com/title/tt0480249"
## [10] "http://www.imdb.com/title/tt0848228"
## [11] "http://www.imdb.com/title/tt0371746"
## [12] "http://www.imdb.com/title/tt0903624"
```

Here I take a smalll sample at first to test the system.

```
mydata = filter(imdb, year > 2015,  country == "USA")
#View(mydata2)
dim(mydata)
```

```
## [1] 45 12
```

This code below creates an adjacency matrix that is used for the network plot. The i,jth entry is 1 if movie i is connected to movie j and zero otherwise.

Here is the upper 1-5th quadrant of one such matrix.

## Batman v Superman: Dawn of Justice Captain America: Civil War

## Batman v Superman: Dawn of Justice 0 1

## Captain America: Civil War 0 0

## Star Trek Beyond 1 1

## The Legend of Tarzan 0 0

## X-Men: Apocalypse 1 1

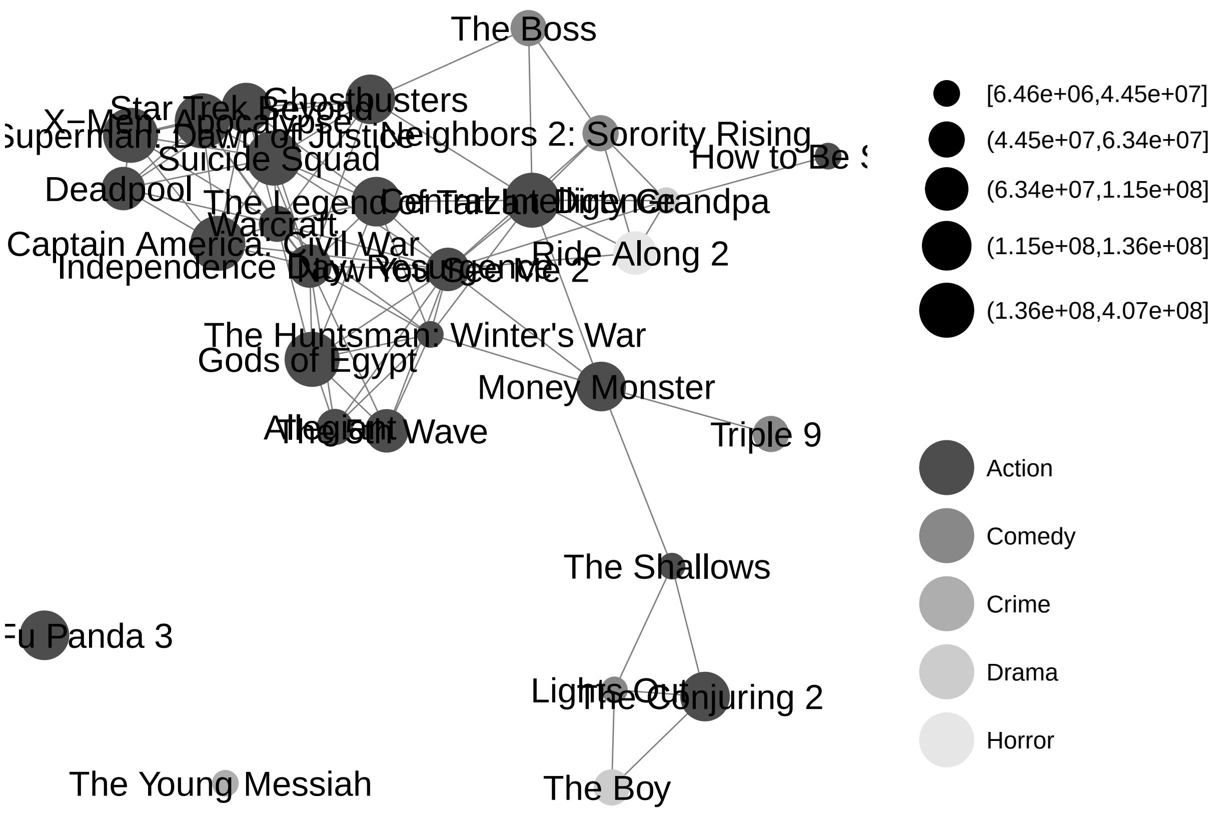
```

make_network = function(cur_data = mydata){

    n = nrow(cur_data)

    #Need an index number for each link to match
    vectorize = function(input_links){
    out = c(rep(0, n))
    for(i in input_links){
          index = match(i, cur_data$links)
          out[index] = 1
      }
    return(out)
    }

  each_links = sapply(cur_data$links, get_links)
  x = ldply(each_links, vectorize)
  links_matrix = as.matrix(x[2:ncol(x)])

  #Remove empty connections

  rownames(links_matrix) = cur_data$title
  colnames(links_matrix) = cur_data$title
  links_matrix2 = links_matrix[,which(!apply(links_matrix,2,FUN = function(x){all(x == 0)}))]
  links_matrix2 = links_matrix2[which(!apply(links_matrix,2,FUN = function(x){all(x == 0)})), ]


  net1 = network( links_matrix2, directed = F, na.omit = T)

  cur_data = filter(cur_data, title %in% colnames(links_matrix2))

  network.vertex.names(net1) = rownames(links_matrix2)

  length = nrow(links_matrix2)

  ggnet2(net1,
       size = cur_data$gross_adj[1:length],
       size.cut = 5, label = T,
       color = factor( cur_data$genres_simple[1:length])
       )
}
```

## Movies in the US released after 2016

You can find interesting patterns between the linked movies over different years.

```
make_network()
```

```
## Loading required package: scales
```

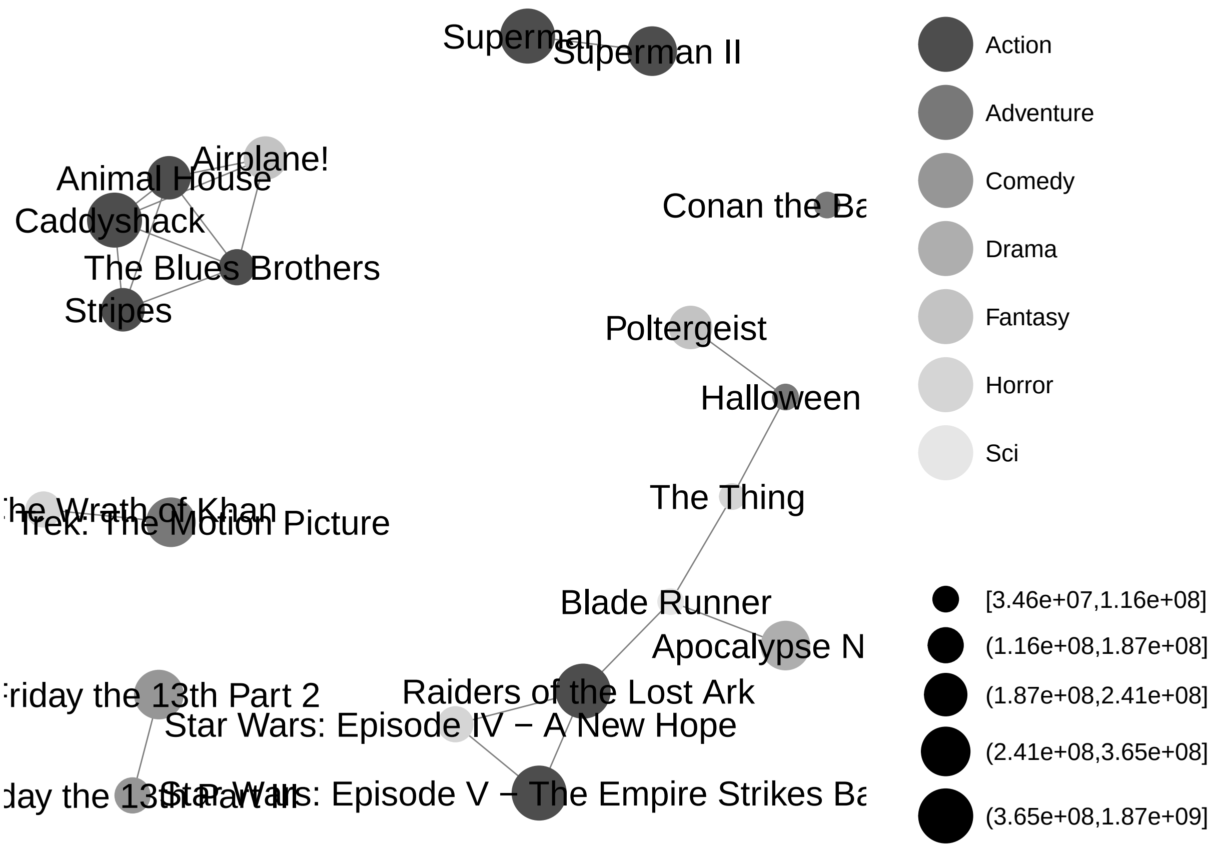


## Movies from 1975 to 1983

```
mydata2 = filter(imdb, year > 1975, year < 1983,  country == "USA")
dim(mydata2)
```

```
## [1] 57 12
```

```
make_network(mydata2)
```



## Movies from Prior to 1975

```
# mydata3 = filter(imdb, year > 1900, year < 1975,  country == "USA")
# dim(mydata3)
#
# make_network(mydata3)
```

## Movies from 2010 - 2014

```
# mydata4 = filter(imdb, year > 2005, year < 2008,  country == "USA")
# dim(mydata4)
#
# make_network(mydata4)
```