Midterm

Ankita Shankhdhar

Due 30th October 2015

Globally defining the text not to fall off the page.

```
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60))
```

Part 1

Loading the html file

-**Part a** Commented on what each line is doing

```
# reading the the whole page
top250.page <- readLines("http://www.imdb.com/chart/top?ref_=ft_250")
# find where '/title/tt' occurs on that page
top250.codes <- top250.page[grep("/title/tt", top250.page)]
# slit the entries by looking for '/' and only keep the ones
# 3rd entry on the list (which are the codes)
top250.codes <- unlist(strsplit(top250.codes, split = "//?"))[seq(from = 3,
    to = 1999, by = 4)]
# get the codes for each of the movies
top250.codes <- top250.codes[seq(from = 2, to = 500, by = 2)]
# list the top 6 codes
head(top250.codes)
```

```
## [1] "tt0111161" "tt0068646" "tt0071562" "tt0468569" "tt0050083" "tt0108052"
```

-Part b

Function that will read in all the data that we need and only extract information that we want.

```
get.stuff <- function(imdb.code) {
    # reads in the page of a specific moview
    address <- paste("http://www.imdb.com/title/", imdb.code,
        sep = "")
    page.movie <- readLines(address)

    # get title
    title1 <- c(page.movie[grep("<title>", page.movie)])
    title2 <- unlist(strsplit(title1, split = "<title>"))[seq(from = 2,
        to = 200, by = 2)]
    title3 <- strsplit(title2, split = " \\(")[[1]][1]
    title <- gsub(") - IMDb</title>", "", title3)
```

```
# get the year
year1 <- unlist(strsplit(title1, split = "\\("))[seq(from = 2,
    to = 200, by = 2)]
year <- gsub(") - IMDb</title>", "", year1)[1]

# rating of movie
rating1 <- c(page.movie[grep("contentRating", page.movie)[1]])
rating2 <- unlist(strsplit(rating1, split = "content="))[seq(from = 2,
    to = 200, by = 2)]
rating <- unlist(strsplit(rating1, split = ">"))[seq(from = 2,
    to = 200, by = 2)][1]

# user rating
User.Rating1 <- page.movie[grep("Users rated this", page.movie)[1]]
User.Rating2 <- strsplit(User.Rating1, "/")[[1]][1]
User.Rating3 <- as.numeric(strsplit(User.Rating2, "this ")[[1]][2])
User.Rating <- User.Rating3

# get the number of raters
num.rater1 <- unlist(strsplit(User.Rating1, split = "\\("))[seq(from = 2,
    to = 200, by = 2)]
num.rater2 <- strsplit(num.rater1, " votes")[[1]][1]
num.rater <- as.numeric(gsub("\\,", "", num.rater2))

# get the genre
inds <- grep("ref_=tt_stry_gnr", page.movie)
genre1 <- unlist(strsplit(page.movie[inds], "/genre/"))[seq(from = 2,
    to = length(inds) * 2, by = 2)]
genre2 <- unlist(strsplit(genre1, "\\?"))[seq(from = 1, to = length(inds) *
    2, by = 2)]
genre3 <- paste(genre2, collapse = " ")
genre <- gsub(" ", ", ", genre3)

########## BUSINESS ########
budget <- NA
opn <- NA
gross <- NA

# read the business page
address1 <- paste("http://www.imdb.com/title/", imdb.code,
    "/business", sep = "")
business.page <- readLines(address1)

# find where Budget occurs
temp <- grep("Budget", business.page)
# walk through and find the $ signs
if (length(temp) > 0) {
    budget1 <- business.page[temp + 1]
    budget2 <- strsplit(budget1, " \\(")[[1]][1]
    budget3 <- gsub("\\$", "", budget2)
    budget <- as.numeric(gsub("\\,", "", budget3))
}
```

```
    # find where Opening Weekend occurs
    temp2 <- grep("Opening Weekend", business.page)
    # walk through and find the $ signs
    if (length(temp2) > 0) {
        opn1 <- business.page[temp2 + 1]
        opn2 <- strsplit(opn1, " \\(")[[1]][1]
        opn3 <- gsub("\\$", "", opn2)
        opn <- as.numeric(gsub("\\,", "", opn3))
    }
    # find where Gross occurs
    temp3 <- grep("Gross", business.page)
    # walk through and find the $ signs
    if (length(temp3) > 0) {
        gross1 <- business.page[temp3 + 1]
        gross2 <- strsplit(gross1, " \\(")[[1]][1]
        gross3 <- gsub("\\$", "", gross2)
        gross <- as.numeric(gsub("\\,", "", gross3))
    }

    return(list(Title = title, Year = year, Rating = rating,
        UserRating = User.Rating, NumRaters = num.rater, Budget = budget,
        Opening.Night = opn, Gross = gross, Genre = genre))
}
```

This is to test the function on the movie *The Departed*

```
as.vector(get.stuff("tt0407887"))
```

```
## $Title
## [1] "The Departed"
##
## $Year
## [1] "2006"
##
## $Rating
## [1] "R"
##
## $UserRating
## [1] 8.5
##
## $NumRaters
## [1] 791392
##
## $Budget
## [1] 9e+07
##
## $Opening.Night
## [1] 26887467
##
## $Gross
## [1] 132384315
##
## $Genre
```

```
## [1] "Crime, Drama, Thriller"
```

-**Part c** This is to apply to all the Top 250 movie codes and store the values in the proper way.

```
# apply to all the codes using sapply()
data <- sapply(top250.codes, get.stuff)
# turn it into data drame
data <- as.data.frame(t(data))
# sapply() leave each column of data in an odd format (a
# vector of lists!)  to fix that
for (i in 1:dim(data)[2]) data[, i] <- unlist(data[, i])
```

Part 2

-Part a

A function that lists the names and revenues of the **m** movies with the largest grosses.

Evaluate function at **m=5**

```
sort.gross <- function(m) {
    # merge title and the gross amount together
    title.gross <- merge(as.numeric(data$Gross), as.character(data$Title),
        by = c("row.names"))
    # put them all in decreasing order
    gross.sorted <- title.gross[with(title.gross, order(-title.gross$x)),
        ]
    # return them put together
    return(paste(Title = gross.sorted$y[1:m], ":", "$", GrossAmount = gross.sorted$x[1:m]))
}
# find 5 movies that made most money
sort.gross(5)
```

```
## [1] "The Avengers : $ 623357910"
## [2] "The Dark Knight : $ 534858444"
## [3] "Star Wars : $ 460935665"
## [4] "The Dark Knight Rises : $ 448139099"
## [5] "The Lion King : $ 422783777"
```

-Part b

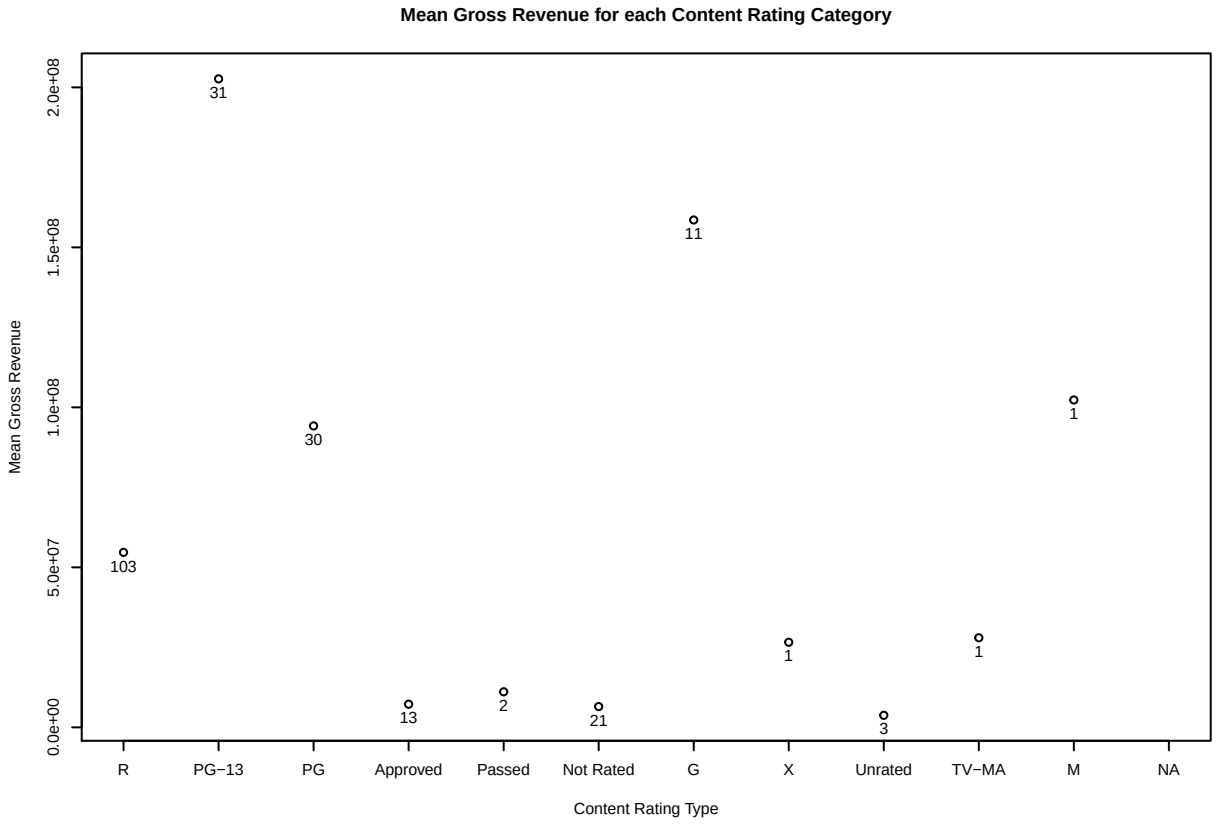
A plot that shows mean gross revenue for each content rating category.

```
# function that gets the mean gross revenue
rating.avg <- function(type) {
    indexr <- data$Gross[which(data$Rating == type)]
    indexr <- indexr[which(!is.na(indexr))]
    avg <- sum(indexr)/length(indexr)
    return(list(average = avg, movNum = length(indexr)))
}

# evaluate the function for different ratings
```

```r
rating <- c("R", "PG-13", "PG", "Approved", "Passed", "Not Rated",
    "G", "X", "Unrated", "TV-MA", "M", "NA")
contentR.avg <- c(rating.avg("R")[[1]], rating.avg("PG-13")[[1]],
    rating.avg("PG")[[1]], rating.avg("Approved")[[1]], rating.avg("Passed")[[1]],
    rating.avg("Not Rated")[[1]], rating.avg("G")[[1]], rating.avg("X")[[1]],
    rating.avg("Unrated")[1], rating.avg("TV-MA")[[1]], rating.avg("M")[[1]],
    rating.avg("NA")[[1]])
# because the values are large I made the text small using
# the par statement
par(cex = 0.5)
plot(unlist(contentR.avg), xaxt = "n", xlab = "Content Rating Type",
    ylab = "Mean Gross Revenue", main = "Mean Gross Revenue for each Content Rating Category")
axis(1, at = 1:12, labels = rating)

contentR.num <- c(rating.avg("R")[[2]], rating.avg("PG-13")[[2]],
    rating.avg("PG")[[2]], rating.avg("Approved")[[2]], rating.avg("Passed")[[2]],
    rating.avg("Not Rated")[[2]], rating.avg("G")[[2]], rating.avg("X")[[2]],
    rating.avg("Unrated")[2], rating.avg("TV-MA")[[2]], rating.avg("M")[[2]],
    rating.avg("NA")[[2]])
text(unlist(contentR.avg), paste(unlist(contentR.num)), pos = 1,
    cex = 1)
```



-Part c

Computes if there is a relationship between user ratings(or number of user ratings) and gross revenue

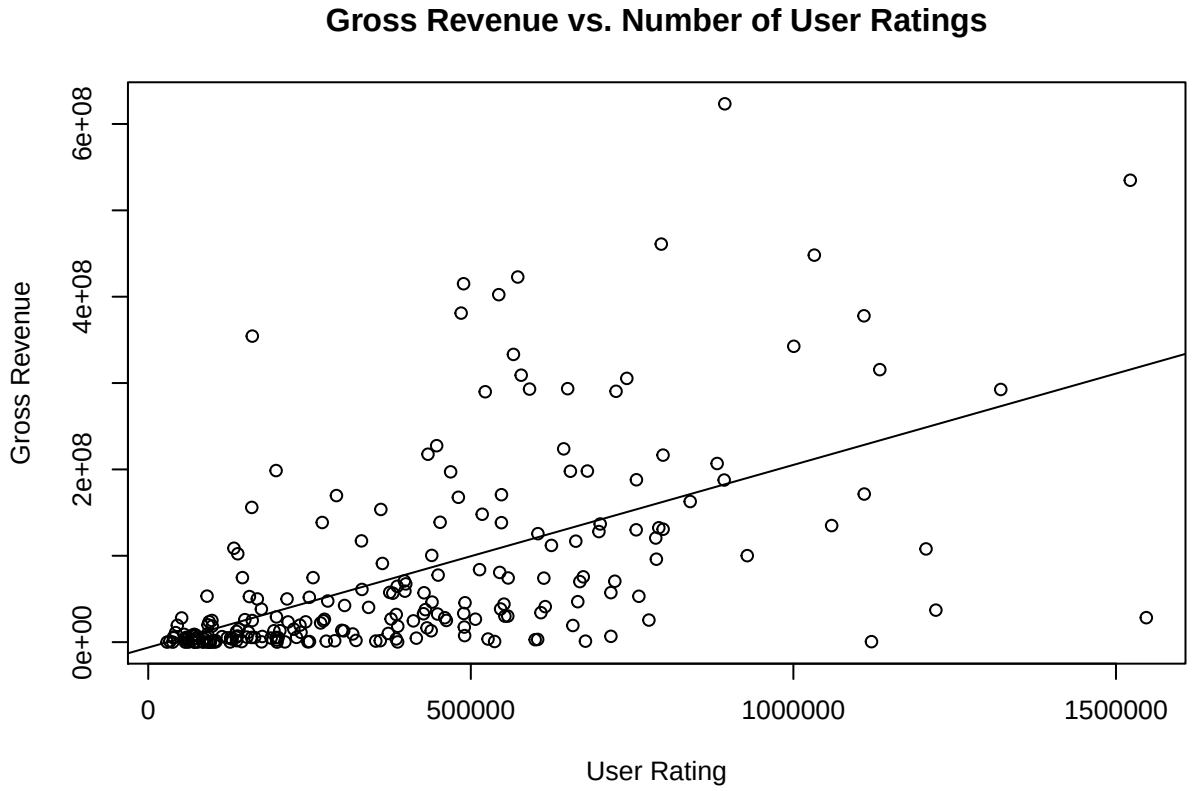
```r
gross <- data$Gross
user <- data$NumRaters[which(!is.na(gross))]
userr <- data$UserRating[which(!is.na(gross))]
gross <- data$Gross[which(!is.na(gross))]
fit <- lm(gross ~ user)
summary(fit)
```

```
##
## Call:
## lm(formula = gross ~ user)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -292578253  -45948890  -14033392    8278405  440617896
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.258e+06  1.044e+07  -0.599    0.549
## user         2.115e+02  2.084e+01  10.150   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 93440000 on 215 degrees of freedom
## Multiple R-squared:  0.324,  Adjusted R-squared:  0.3208
## F-statistic:   103 on 1 and 215 DF,  p-value: < 2.2e-16
```

```r
confint.lm(fit)
```

```
##                      2.5 %       97.5 %
## (Intercept) -2.683595e+07 1.431932e+07
## user         1.704175e+02 2.525547e+02
```

```r
par(cex = 0.8)
plot(user, gross, xlab = "User Rating", ylab = "Gross Revenue",
    main = "Gross Revenue vs. Number of User Ratings")
abline(fit)
```



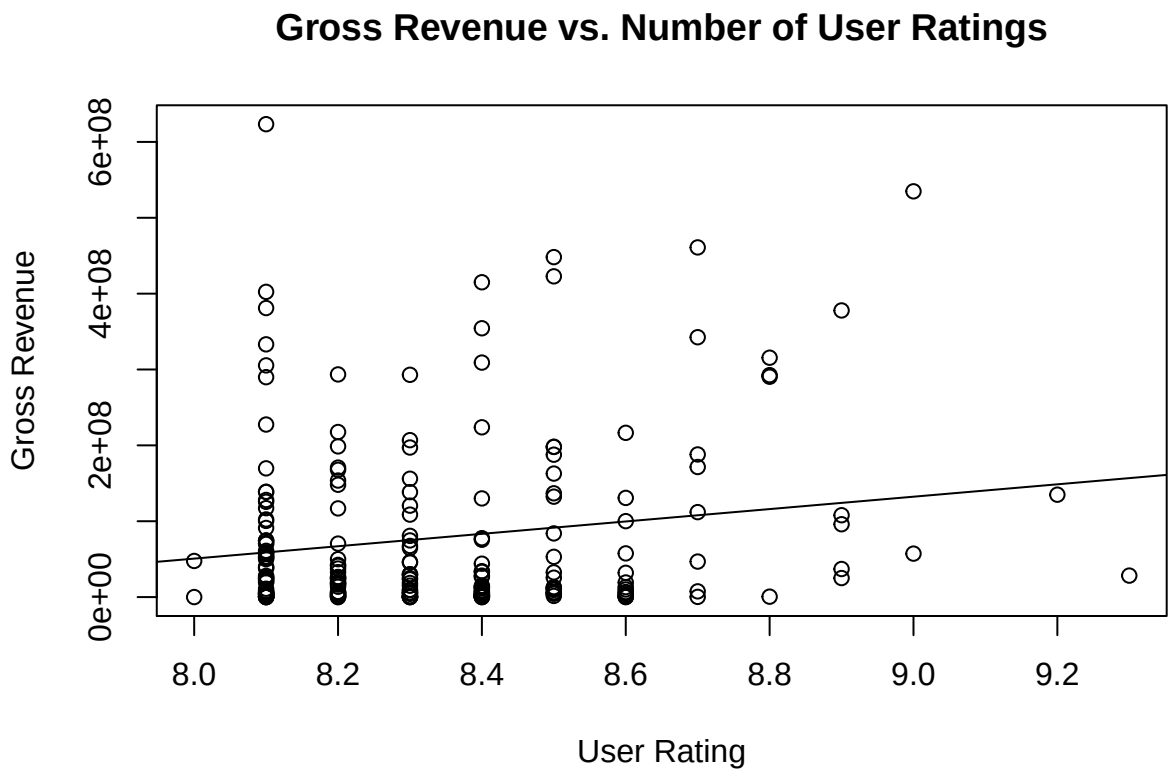

```
par(cex = 1)
fit2 <- lm(gross ~ userr)
summary(fit2)
```

```
##
## Call:
## lm(formula = gross ~ userr)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -128491326  -69197577  -45104470   31054780  564487376
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -602374732  269044222  -2.239   0.0262 *
## userr         81635218   32273010   2.530   0.0121 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.12e+08 on 215 degrees of freedom
## Multiple R-squared:  0.0289, Adjusted R-squared:  0.02438
## F-statistic: 6.398 on 1 and 215 DF,  p-value: 0.01214
```

```
confint.lm(fit2)
```

```
##                    2.5 %    97.5 %
## (Intercept) -1132676799 -72072664
## userr          18023207 145247229
```

```
plot(userr, gross, xlab = "User Rating", ylab = "Gross Revenue",
    main = "Gross Revenue vs. Number of User Ratings")
abline(fit2)
```



The above plot does not include gross revenues (as well as number of user ratings or user rating) that were stated in anything other than US dollars. The fit shows that there is a relationship between the number of user ratings and gross revenue. However, having said that we must notice that the variance below **Number of Raters= 250000** in much less in the data in comparison to the later ones. So we can say there exists a relationship but it is not linear and we could so some transformation and observe a better linear relationship. This statment is validated by the confidence intervals not containing $\beta_1 = 0$ but the r or the correlation coefficient which is small.

-Part d

T-test to compare the mean gross revenue for movies whose genre includes Drama to those whose genre includes Comedy

```
comedy_all <- data$Gross[grep("Comedy", data$Genre)]
drama_all <- data$Gross[grep("Drama", data$Genre)]
t.test(comedy_all, drama_all)
```

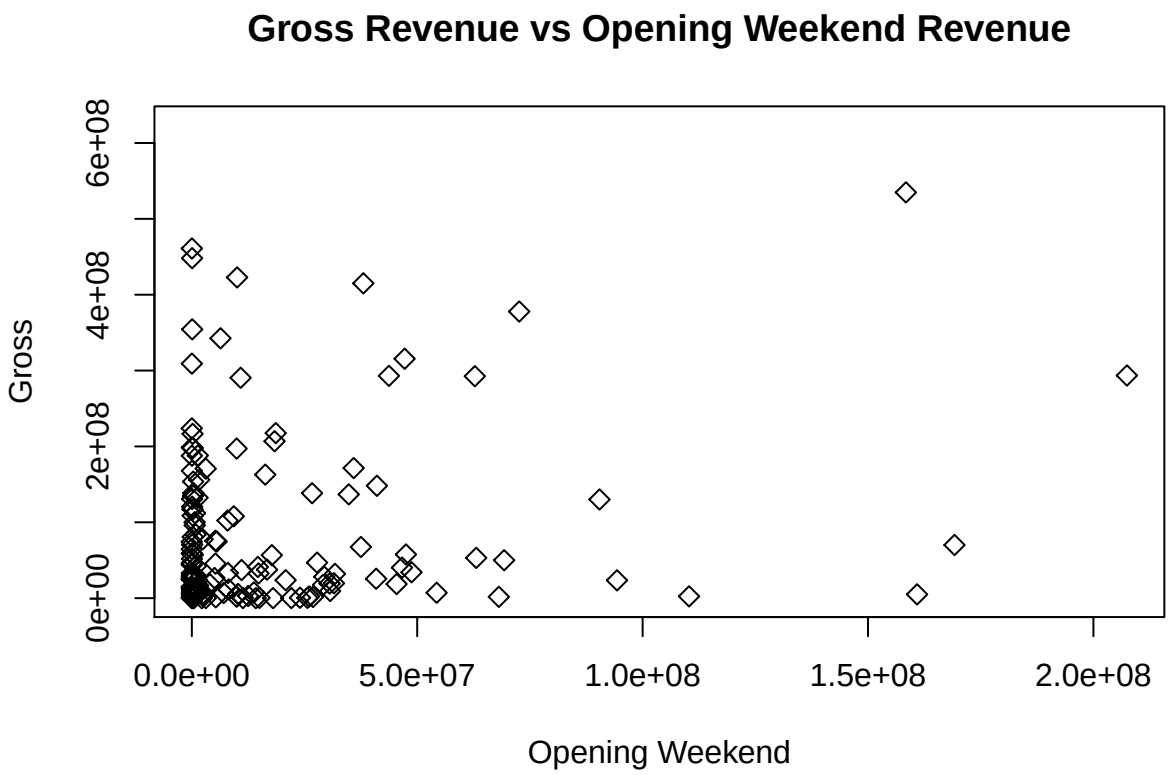
```
##
##  Welch Two Sample t-test
##
## data:  comedy_all and drama_all
## t = 0.5864, df = 44.334, p-value = 0.5606
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -28294526  51520975
```

```
## sample estimates:
## mean of x mean of y
##  67119584  55506359
```

According to the Welch Two Sample t-test, we do not have enough evidence to reject the null hypothesis because our p-value is 0.5754 which is much larger than our alpha 0.05. Hence we fail to reject the null hypothesis and conclude that there is no difference in the mean of the gross revenue of movies of genre Comedy or Drama.

-**Part e** Comparison of gross revenue and opening revenue. (Gross means total revenue, and opening means opening weekend revenue).

```
# calculate the mean of the all the NA gross revenues
mean.gross <- data$Gross[which(!is.na(data$Gross))]
# calculate the mean of all the NA opening weekend revenues
mean.opn <- data$Opening.Night[which(!is.na(data$Opening.Night))]
# make them the same length so I can plot them together
maxl <- max(length(mean.gross), length(mean.opn))
length(mean.opn) <- maxl
plot(mean.opn, mean.gross, main = "Gross Revenue vs Opening Weekend Revenue",
    xlab = "Opening Weekend", ylab = "Gross", pch = 5)
```



We see that there are a lot of movies that made $0 on the opening weekend but still had a huge gross revenue. This is because these movies are foreign movies. Hence they did have some revenue overall in the US but the opening weekend reveue is in something other than USD. Using this analysis we learn that we have to know our data well because if I didn't know that my dataset only contains US dollars then I would not to able to make this inference.

Part 3

Loading the function to get the linked movies

```
linked.movies <- function(imdb.code) {
    # read the movie page
    address <- paste("http://www.imdb.com/title/", imdb.code,
        sep = "")
    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 = "")
    return(recs)
}
```

-Part a

Function to build a matrix with 250 rows and 250 columns where each row and column corresonds to one of the top 250 movies, and labelled with the movie's name.

```
linked.check <- function(imdb.code) {
    # store the top 250 codes
    linked <- c(top250.codes)
    # tranpose them so they become our columns
    linked <- t(linked)
    # set this rows to be the codes as well
    vnew <- linked
    # get the reccomended movies for the code
    linked.codes <- linked.movies(imdb.code)
    temp <- 1
    # walk through the reccomended links
    while (temp <= length(linked.codes)) {
        val <- linked.codes[temp]
        # if they match any codes then keep them
        if (!is.na(match(val, linked))) {
            # set the vector of all codes to be 1 at the matching code
            vnew[which(vnew == val)] <- 1
        }
        temp <- temp + 1
    }
    return(vnew)
}
links <- sapply(top250.codes, linked.check)
links[links != 1] <- 0
links <- as.data.frame(t(links))
colnames(links) <- top250.codes
```

-Part b

Fill in this matrix without a loop! (followed by part (a))

```
linked.checknew <- function(imdb.code) {
    # store the top 250 codes
    linked <- c(top250.codes)
    # tranpose them so they become our columns
    linked <- t(linked)
    # set this rows to be the codes as well
    vnew <- linked
    # get the reccomended movies for the code
    linked.codes <- linked.movies(imdb.code)
    # if the length of the matched codes is not NA
    linked.codes1 <- linked.codes[which(!is.na(match(linked.codes,
        linked)))]
    if (length(linked.codes1) != 0) {
        # set the vector of all codes to be 1 at the matching code
        vnew[which(!is.na(match(vnew, linked.codes1[1:length(linked.codes1)])))] <- 1
    }
    return(c(vnew))
}

links.new <- sapply(top250.codes, linked.checknew)
links.new[links.new != 1] <- 0
links.new <- as.data.frame(t(links.new), stringsAsFactors = FALSE)
colnames(links.new) <- top250.codes

remove <- c("tt0137523", "tt0110912", "tt0172495", "tt0109830",
    "tt0053125")

top250.new <- top250.codes[which(top250.codes != remove)]

links.new2 <- sapply(top250.new, linked.checknew)
links.new2[links.new2 != 1] <- 0
links.new2 <- as.data.frame(t(links.new2), stringsAsFactors = FALSE)
colnames(links.new2) <- top250.codes

remove <- c("tt0137523", "tt0110912", "tt0172495", "tt0109830",
    "tt0053125", "tt0120737", "tt0114369", "tt0120586", "tt1375666",
    "tt0167261", "tt0133093", "tt0043014", "tt1853728", "tt0073486",
    "tt0102926", "tt0033467", "tt0167260", "tt1205489", "tt0034583",
    "tt0120815")

top250.new <- top250.codes[which(top250.codes != remove)]

links.new2 <- sapply(top250.new, linked.checknew)
links.new2[links.new2 != 1] <- 0
links.new2 <- as.data.frame(t(links.new2), stringsAsFactors = FALSE)
colnames(links.new2) <- top250.codes
```

-Part c

Extension on part (b) function to take in an extra input that takes the number of recommendations you want to consider. This input takes values between 1 and 12. If it is 1, the link is only included if the movie is the top recommendation, etc.

```
# function that only looks at the number of reccomended
# movies specifief
linked.checkNum <- function(imdb.code, linkedNum) {
    linked <- c(top250.codes)
    linked <- t(linked)
    vnew <- linked
    linked.codes <- linked.movies(imdb.code)[1:linkedNum]
    linked.codes1 <- linked.codes[which(!is.na(match(linked.codes,
        linked)))]
    if (length(linked.codes1) != 0) {
        vnew[which(!is.na(match(vnew, linked.codes1[1:length(linked.codes1)])))] <- 1
    }
    return(c(vnew))
}

# for the top 2 reccomended
num <- 2
links.num2 <- sapply(top250.codes, linked.checkNum, linkedNum = num)
links.num2[links.num2 != 1] <- 0
links.num2 <- as.data.frame(t(links.num2))
colnames(links.num2) <- top250.codes

# for the top 4 reccomended
num <- 4
links.num4 <- sapply(top250.codes, linked.checkNum, linkedNum = num)
links.num4[links.num4 != 1] <- 0
links.num4 <- as.data.frame(t(links.num4))
colnames(links.num4) <- top250.codes

# for the top 8 reccomended
num <- 8
links.num8 <- sapply(top250.codes, linked.checkNum, linkedNum = num)
links.num8[links.num8 != 1] <- 0
links.num8 <- as.data.frame(t(links.num8))
colnames(links.num8) <- top250.codes
```

-Part d

Movies are linked to the most other movies

```
# convert the output of all recommendations to a matrix
link.matrix <- matrix(as.numeric(unlist(links.new)), nrow = 250,
    ncol = 250)
rownames(link.matrix) <- top250.codes
# sum all the columns of the matrix
linkTotal <- colSums(link.matrix)
# make a new list with the title and the totals
movie.linkNum <- list(Title = as.character(data$Title), NumLink = as.numeric(linkTotal))
movie.linkNum <- merge(as.character(data$Title), as.numeric(linkTotal),
    by = "row.names")
# sort them by decreasing order
movie.linkNum <- movie.linkNum[with(movie.linkNum, order(-movie.linkNum$y)),
    ]
movie.linkNum <- data.frame(Title = movie.linkNum$x, NumOfLinks = movie.linkNum$y)
```

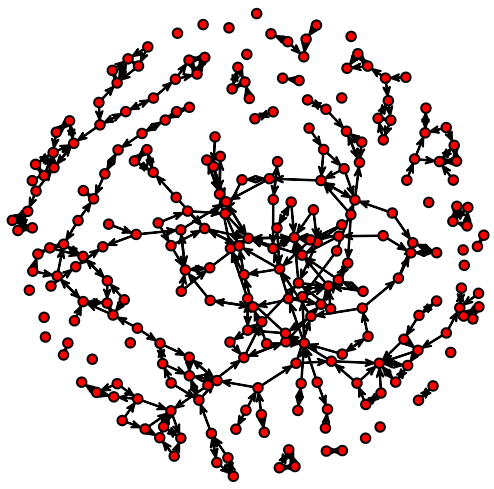
```r
# print the top 6 with most number of matches
head(movie.linkNum)
```

```
##          Title NumOfLinks
## 1   Fight Club         34
## 2 Pulp Fiction         34
## 3 Forrest Gump         32
## 4   The Matrix         30
## 5    Gladiator         30
## 6        Se7en         29
```

-Part e

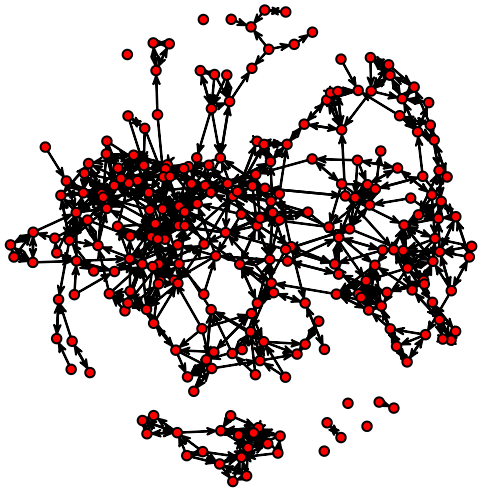
```r
library(statnet)
with.num2 <- network(links.num2)
with.num4 <- network(links.num4)
with.num8 <- network(links.num8)
with.some <- network(links.new2)
with.all <- network(links.new)

plot(with.num2, main = "Network with the top 2 recommended movies")
```



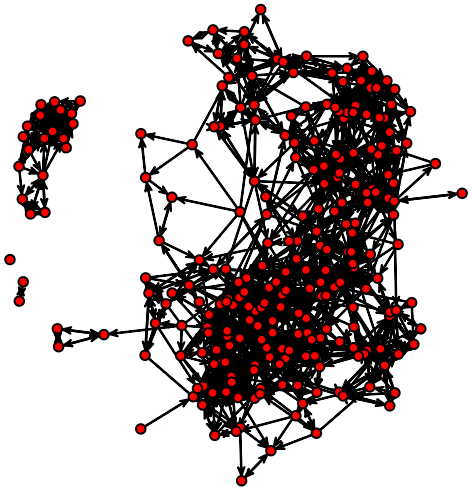

```r
plot(with.num4, main = "Network with the top 4 recommended movies")
```

Network with the top 4 recommended movies



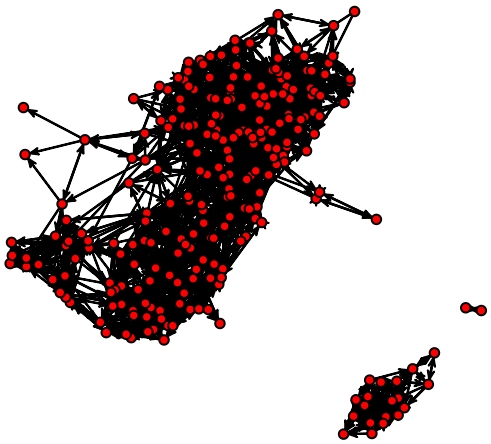
```
plot(with.num8, main = "Network with the top 8 recommended movies")
```

Network with the top 8 recommended movies



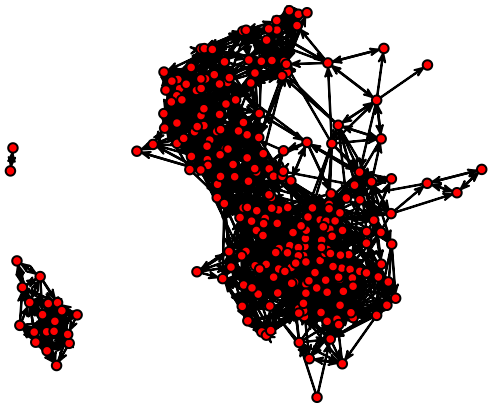
```
plot(with.some, main = "Network after removing top 20 most linked movies")
```

Network after removing top 20 most linked movies



```
plot(with.all, main = "Network with all the recommended movies")
```

Network with all the recommended movies



We see that the first plot is pretty sparse and there are very few connections since we are only looking at the first two. You can see how fast the network becomes dense when you take 4-8 linked movies.