# Revenue Prediction model

*Garima Sood*

*March 12, 2018*

```
library(gridExtra)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(Metrics)
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
dataPath <- "C:/Users/garim/Documents/Quarter 2/Data Mining/Project/Merged data"
master_data <- read.csv(paste(dataPath,"master_data_with_imputed_budget_and_revenue.csv", sep =
"/"))
summary(master_data)
```

```
##     movie_id      actor_1_gender  actor_2_gender  actor_3_gender
##  Min.   :     2   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.: 26412   1st Qu.:1.000   1st Qu.:0.000   1st Qu.:0.000
##  Median : 60013   Median :2.000   Median :1.000   Median :1.000
##  Mean   :108317   Mean   :1.331   Mean   :1.163   Mean   :1.153
##  3rd Qu.:157171   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :469172   Max.   :2.000   Max.   :2.000   Max.   :2.000
##                   NA's   :2420    NA's   :3752    NA's   :4664
##  actor_4_gender  actor_5_gender            actor_1_name
##  Min.   :0.000   Min.   :0.000                    : 2420
##  1st Qu.:0.000   1st Qu.:0.000   John Wayne       :   94
##  Median :1.000   Median :1.000   Jackie Chan      :   73
##  Mean   :1.111   Mean   :1.102   Nicolas Cage     :   60
##  3rd Qu.:2.000   3rd Qu.:2.000   Robert De Niro   :   56
##  Max.   :2.000   Max.   :2.000   GÃ©rard Depardieu:   52
##  NA's   :5799    NA's   :8279    (Other)          :42783
##         actor_2_name             actor_3_name           actor_4_name
##               : 3752                   : 4664                 : 5799
##  Barbara Hale :   36   Donald Pleasence:   25   Donald Crisp  :   17
##  Oliver Hardy :   29   George Sanders  :   23   Harry Andrews :   17
##  Huntz Hall   :   28   Susan Sarandon  :   23   Ray Liotta    :   16
##  Lou Costello :   27   William R. Moses:   21   Walter Brennan:   16
##  Michael Caine:   26   Ned Beatty      :   20   Alfred Molina :   15
##  (Other)      :41640   (Other)         :40762   (Other)       :39658
##         actor_5_name   director_gender          director_name
##               : 8279   Min.   :0.000                   :  887
##  Keenan Wynn  :   20   1st Qu.:0.000   John Ford       :   66
##  Gene Lockhart:   19   Median :2.000   Michael Curtiz  :   65
##  Steve Buscemi:   17   Mean   :1.198   Werner Herzog   :   54
##  Alan Hale    :   16   3rd Qu.:2.000   Alfred Hitchcock:   53
##  John Hurt    :   16   Max.   :2.000   Georges MÃ©liÃ¨s:   51
##  (Other)      :37171   NA's   :887     (Other)         :44362
##  producer_gender          producer_name   casting_gender
##  Min.   :0.000                   :23504   Min.   :0.00
##  1st Qu.:0.000   Walt Disney     :   72   1st Qu.:0.00
##  Median :2.000   Darryl F. Zanuck:   71   Median :1.00
##  Mean   :1.142   Hal B. Wallis   :   67   Mean   :0.83
##  3rd Qu.:2.000   Brian Grazer    :   58   3rd Qu.:1.00
##  Max.   :2.000   Roger Corman    :   52   Max.   :2.00
##  NA's   :23504   (Other)         :21714   NA's   :37370
##           casting_name                     belongs_to_collection
##                 :37370                                :41038
##  Avy Kaufman    :  178   The Bowery Boys              :   29
##  Deborah Aquila :  125   TotÃ² Collection             :   27
##  Lynn Stalmaster:  122   James Bond Collection        :   26
##  Mary Vernieu   :  105   ZatÃ´ichi: The Blind Swordsman:   26
##  Nancy Nayor    :   92   The Carry On Collection      :   25
##  (Other)        : 7546   (Other)                      : 4367
##         genre_1          genre_2                genre_3
##  Drama      :11984           :17024                 :31534
##  Comedy     : 8829   Drama   : 6321   Thriller       : 2237
##  Action     : 4496   Comedy  : 3269   Romance        : 2046
##  Documentary: 3419   Romance : 2864   Drama          : 1680
```

```
##  Horror     : 2621   Thriller: 2527   Comedy         :  912  
##             : 2448   Action  : 1546   Science Fiction:  877  
##  (Other)    :11741   (Other) :11987   (Other)        : 6252  
##             genre_4                                production_company_1
##                 :41131                                         :11900  
##  Thriller       :  923   Paramount Pictures                    : 1000  
##  Romance        :  500   Metro-Goldwyn-Mayer (MGM)             :  853  
##  Science Fiction:  394   Twentieth Century Fox Film Corporation:  781  
##  Crime          :  301   Warner Bros.                          :  757  
##  Mystery        :  286   Universal Pictures                    :  754  
##  (Other)        : 2003   (Other)                               :29493  
##                 production_company_2               production_company_3
##                           :28507                            :36476     
##  Warner Bros.             :  270     Warner Bros.             :  130     
##  Metro-Goldwyn-Mayer (MGM):  151     Canal+                   :  109     
##  Canal+                   :  124     Metro-Goldwyn-Mayer (MGM):   44     
##  Touchstone Pictures      :   75     Relativity Media         :   42     
##  Universal Pictures       :   71     TF1 Films Production     :   29     
##  (Other)                  :16340     (Other)                  : 8708     
##                production_country_1             production_country_2
##  United States of America:18449                          :38497    
##                          : 6295     United States of America: 2132    
##  United Kingdom          : 3073     France                  :  918    
##  France                  : 2714     United Kingdom          :  660    
##  Canada                  : 1499     Germany                 :  531    
##  Japan                   : 1498     Italy                   :  485    
##  (Other)                 :12010     (Other)                 : 2315    
##                production_country_3 spoken_language_1 spoken_language_2
##                          :43381     English  :26873            :37765    
##  United States of America:  411              : 4060   English  : 1595    
##  France                  :  247     FranÃ§ais: 2436   FranÃ§ais: 1479    
##  Germany                 :  232     Italiano : 1411   Deutsch  :  920    
##  United Kingdom          :  231     æ□¥æ□¬èª□: 1391   EspaÃ±ol :  782    
##  Italy                   :  154     Deutsch  : 1304   Italiano :  617    
##  (Other)                 :  882     (Other)  : 8063   (Other)  : 2380    
##  spoken_language_3   adult          budget         
##           :43089   False:45529   Min.   :        0  
##  Deutsch  :  328   True :    9   1st Qu.:        0  
##  EspaÃ±ol :  308                 Median :        0  
##  FranÃ§ais:  234                 Mean   :  4212154  
##  English  :  232                 3rd Qu.:        0  
##  Italiano :  225                 Max.   :380000000  
##  (Other)  : 1122                 NA's   :1534       
##                                       homepage          imdb_id     
##                                           :37746            :   17  
##  http://www.georgecarlin.com              :   12   tt1180333:    9  
##  http://www.wernerherzog.com/films-by.html:    7   tt0022537:    4  
##  http://breakblade.jp/                    :    6   tt0022879:    4  
##  http://movies.warnerbros.com/pk3/        :    4   tt0046468:    4  
##  http://phantasm.com                      :    4   tt0062229:    4  
##  (Other)                                  : 7759   (Other)  :45496  
##  original_language             original_title 
##  en     :32316     Blackout            :   12  
##  fr     : 2443     Alice in Wonderland:    8  
```

```
##  it     : 1529     Hamlet             :    8
##  ja     : 1356     King Lear          :    8
##  de     : 1083     A Christmas Carol  :    7
##  es     :  993     Cinderella         :    7
##  (Other): 5818     (Other)            :45488
##

                                                                                                ov
erview
##

  :  954
##  No overview found.

  :  133
##  Recovering from a nail gun shot to the head and 13 months of coma, doctor Pekka Valinta star
ts to unravel the mystery of his past, still suffering from total amnesia.

  :    9
##  No Overview

  :    7
##

  :    5
##  King Lear, old and tired, divides his kingdom among his daughters, giving great importance t
o their protestations of love for him. When Cordelia, youngest and most honest, refuses to idly
flatter the old man in return for favor, he banishes her and turns for support to his remaining
daughters. But Goneril and Regan have no love for him and instead plot to take all his power fro
m him. In a parallel, Lear's loyal courtier Gloucester favors his illegitimate son Edmund after
being told lies about his faithful son Edgar. Madness and tragedy befall both ill-starred father
s.:    5
```

```
## (Other)






  :44425
##    popularity                                 poster_path
##  Min.   :  0.0000                                  :  386
##  1st Qu.:  0.3863   /8VSZ9coCzxOCW2wE2Qene1H1fKO.jpg:    9
##  Median :  1.1283   /5D7UBSEgdyONE6Lql6xS7s6OLcW.jpg:    5
##  Mean   :  2.9219   /2kslZXOaW0HmnGuVPCnQlCdXFR9.jpg:    4
##  3rd Qu.:  3.6815   /4J6Ai4C5YRgfRUTlirrJ7QsmJKU.jpg:    4
##  Max.   :547.4883   /5GasjPRAy5rlEyDOH7MeOyxyQGX.jpg:    4
##  NA's   :3          (Other)                         :45126
##      release_date      revenue             runtime
##  2008-01-01:  136   Min.   :0.000e+00   Min.   :   0.00
##  2009-01-01:  121   1st Qu.:0.000e+00   1st Qu.:  85.00
##  2007-01-01:  120   Median :0.000e+00   Median :  95.00
##  2005-01-01:  111   Mean   :1.152e+07   Mean   :  94.13
##  2006-01-01:  101   3rd Qu.:0.000e+00   3rd Qu.: 107.00
##  2002-01-01:   96   Max.   :2.788e+09   Max.   :1256.00
##  (Other)   :44853   NA's   :1537        NA's   :260
##              status
##                 :   84
##  Canceled       :    2
##  In Production  :   20
##  Planned        :   15
##  Post Production:   98
##  Released       :45087
##  Rumored        :  232
##                                                                                       tagline
##                                                                                           :25099
##  Which one is the first to return - memory or the murderer?                               :    9
##  Based on a true story.                                                                   :    7
##  -                                                                                        :    4
##  A love, a hope, a wall.                                                                  :    4
##  Actually produced during the Great Newfoundland Seal Hunt and You see the REAL thing:    4
##  (Other)                                                                                  :20411
##                   title          video        vote_average
##  Blackout            :   13        :    3   Min.   : 0.000
##  Cinderella          :   11   False:45442   1st Qu.: 5.000
##  Alice in Wonderland :    9   True :   93   Median : 6.000
##  Hamlet              :    9                 Mean   : 5.618
##  Beauty and the Beast:    8                 3rd Qu.: 6.800
##  King Lear           :    8                 Max.   :10.000
##  (Other)             :45480                 NA's   :3
##    vote_count
##  Min.   :    0.0
##  1st Qu.:    3.0
##  Median :   10.0
##  Mean   :  109.8
##  3rd Qu.:   34.0
```

```
## Max.   :14075.0
## NA's   :3
```

#### Data Processsing

```
master_data$release_date <- as.Date(master_data$release_date)

#To cut the impact of inflation on movie revenues & budgets, I am excluding data of movies relea
sed before Jan 1985

master_data <- master_data[master_data$release_date > as.Date("01/01/1985","%m/%d/%Y"),]
master_data <- master_data[master_data$budget > 0,]
master_data$actor_1_gender <- as.factor(ifelse(master_data$actor_1_gender==0,NA,ifelse(master_da
ta$actor_1_gender==2,"Male","Female")))
master_data$actor_2_gender <- as.factor(ifelse(master_data$actor_2_gender==0,NA,ifelse(master_da
ta$actor_2_gender==2,"Male","Female")))
master_data$actor_3_gender <- as.factor(ifelse(master_data$actor_3_gender==0,NA,ifelse(master_da
ta$actor_3_gender==2,"Male","Female")))
master_data$actor_4_gender <- as.factor(ifelse(master_data$actor_4_gender==0,NA,ifelse(master_da
ta$actor_4_gender==2,"Male","Female")))
master_data$actor_5_gender <- as.factor(ifelse(master_data$actor_5_gender==0,NA,ifelse(master_da
ta$actor_5_gender==2,"Male","Female")))
master_data$director_gender <- as.factor(ifelse(master_data$director_gender==0,NA,ifelse(master_
data$director_gender==2,"Male","Female")))
master_data$producer_gender <- as.factor(ifelse(master_data$producer_gender==0,NA,ifelse(master_
data$producer_gender==2,"Male","Female")))
master_data$collection <- as.factor(ifelse(nchar(as.character(master_data$belongs_to_collectio
n))>0,"Yes","No"))

master_data$num_prod_comp <-(master_data$production_company_1!="")+(master_data$production_compa
ny_2!="")+
                            (master_data$production_company_3!="")

master_data$num_prod_ctry <-(master_data$production_country_1!="")+(master_data$production_count
ry_2!="")+
                            (master_data$production_country_3!="")

master_data$release_month <- month.abb[month(master_data$release_date)]

master_data <- master_data[ , -which(names(master_data) %in%
              c( "movie_id" ,"actor_1_name","actor_2_name","actor_3_name","actor_4_name","actor_
5_name","director_name","producer_name",
                 "casting_gender","casting_name","belongs_to_collection","genre_2","genre_3","ge
nre_4","production_company_1",
                 "production_company_2","production_company_3" ,"production_country_1", "product
ion_country_2",  "production_country_3" , "spoken_language_1","spoken_language_2", "spoken_langu
age_3" ,"homepage","imdb_id" ,"original_title","overview","poster_path", "status","title","vide
o"))]

require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
p1<- ggplot(master_data, aes(x = actor_1_gender)) + geom_bar()
p2<- ggplot(master_data, aes(x = actor_2_gender)) + geom_bar()
p3<- ggplot(master_data, aes(x = actor_3_gender)) + geom_bar()
p4<- ggplot(master_data, aes(x = actor_4_gender)) + geom_bar()
p5<- ggplot(master_data, aes(x = actor_5_gender)) + geom_bar()
p6<- ggplot(master_data, aes(x = director_gender)) + geom_bar()
p7<- ggplot(master_data, aes(x = producer_gender)) + geom_bar()
p8<- ggplot(master_data, aes(x = budget)) + geom_histogram()
p9<- ggplot(master_data, aes(x = revenue)) + geom_histogram()
p10<-ggplot(master_data, aes(x = popularity)) + geom_histogram()
p11<-ggplot(master_data, aes(x = vote_average)) + geom_histogram()
p12<-ggplot(master_data, aes(x = vote_count)) + geom_histogram()


grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12, nrow = 4, ncol=3)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1508 rows containing non-finite values (stat_bin).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1508 rows containing non-finite values (stat_bin).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

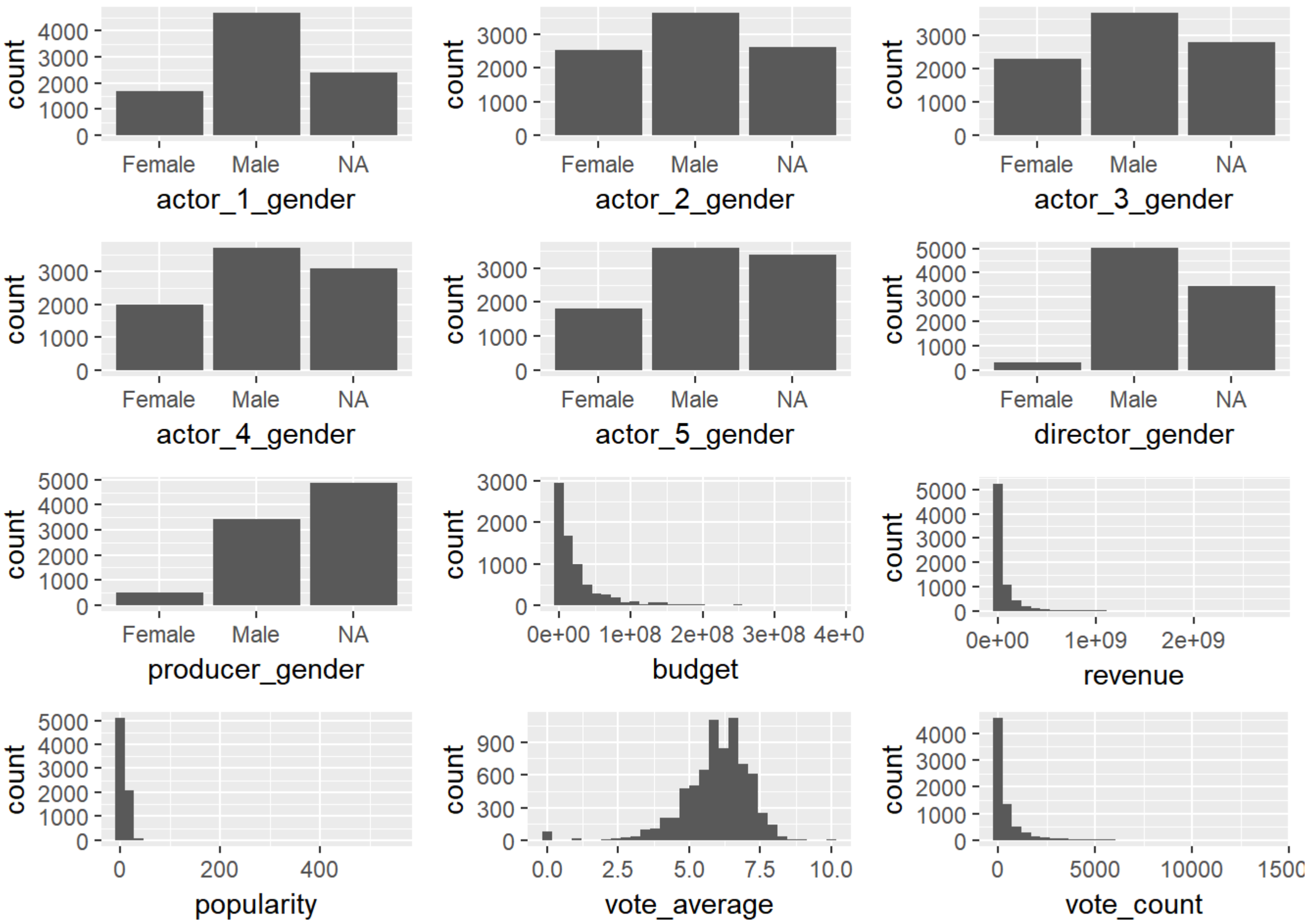
```
## Warning: Removed 1508 rows containing non-finite values (stat_bin).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1508 rows containing non-finite values (stat_bin).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1508 rows containing non-finite values (stat_bin).
```




Plots show that there are a lot of NA values in the different columns. Counting the NA values per column in the data

```
perc_na <- function(x){
  return(sum(is.na(x))/length(x))
}

round(apply(master_data, 2, function(x) perc_na(x)),2)
```

```
##    actor_1_gender    actor_2_gender    actor_3_gender    actor_4_gender
##              0.27              0.30              0.32              0.35
##    actor_5_gender   director_gender   producer_gender           genre_1
##              0.39              0.39              0.55              0.17
##             adult            budget original_language        popularity
##              0.17              0.17              0.17              0.17
##      release_date           revenue           runtime           tagline
##              0.17              0.17              0.17              0.17
##      vote_average        vote_count        collection      num_prod_comp
##              0.17              0.17              0.17              0.17
##      num_prod_ctry     release_month
##              0.17              0.17
```

```
master_data$na_count <- apply(master_data, 1, function(x) sum(is.na(x)))
table(master_data$na_count)
```

```
##
##    0    1    2    3    4    5    6    7    8   22
## 2550 1895 1053  571  386  300  262  244    4 1508
```

Deleting records with missing data in more than 9 columns, and checking the poportion of missing values in the updated data set

```
data <- master_data[master_data$na_count<9,]
dim(data)
```

```
## [1] 7265   23
```

```
round(apply(data, 2, function(x) perc_na(x)),2)
```

```
##    actor_1_gender    actor_2_gender    actor_3_gender    actor_4_gender
##              0.12              0.15              0.18              0.22
##    actor_5_gender   director_gender   producer_gender           genre_1
##              0.26              0.27              0.46              0.00
##             adult            budget original_language        popularity
##              0.00              0.00              0.00              0.00
##      release_date           revenue           runtime           tagline
##              0.00              0.00              0.00              0.00
##      vote_average        vote_count        collection     num_prod_comp
##              0.00              0.00              0.00              0.00
##     num_prod_ctry     release_month          na_count
##              0.00              0.00              0.00
```

After removing thee records, we are left with about 10% missing values in the gender column of lead actors, and 24% missing values in budget. Rest of the columns look good.

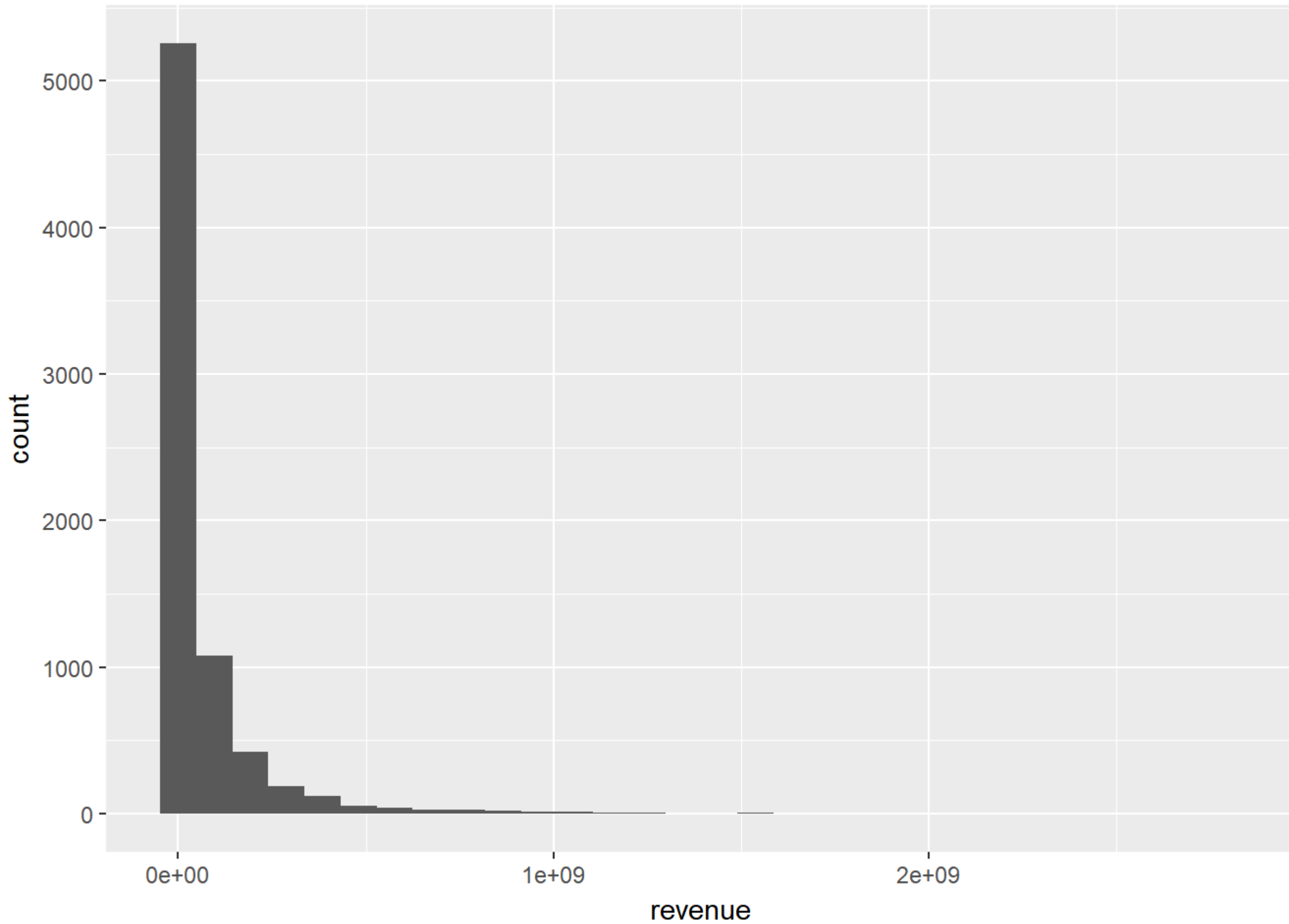
Checking the distribution of our dependent variable (revenue)

```
data$quartile <- ntile(data$revenue, 5)
size = as.matrix(table(data$quartile))
cbind(aggregate(data$revenue, by = list(data$quartile), mean), size)
```

```
##   Group.1           x size
## 1       1         0.0 1453
## 2       2    107560.2 1453
## 3       3   8069558.4 1453
## 4       4  44018587.8 1453
## 5       5 264762403.1 1453
```

```
ggplot(data, aes(x = revenue)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




Missing value estimation:

```r
mean_impute <- function(x){
  a<- (mean(x[!is.na(x)]))
  x <- ifelse(is.na(x), a, x)
  return(x)
}

median_impute <-  function(x){
  a<- (median(x[!is.na(x)]))
  x <- ifelse(is.na(x), a, x)
  return(x)
}

mode_impute <-  function(x){
  ux <- (unique(x))
  a<-ux[which.max(tabulate(match(x[!is.na(x)], ux)))]
  x <- ifelse(is.na(x), a, x)
  return(x)
}

data.imp <-data

#Imputing missing data in gender and runtime columns using mode and median respectively

data.imp$actor_1_gender <- as.factor(mode_impute(data$actor_1_gender))
data.imp$actor_2_gender <- as.factor(mode_impute(data$actor_2_gender))
data.imp$runtime <- median_impute(data.imp$runtime)

perc_blank <- function(x){
  return(sum(x ==""|x==" ")/length(x))
}
round(apply(data.imp, 2, function(x) perc_blank(x)),2)
```

```
##    actor_1_gender    actor_2_gender    actor_3_gender    actor_4_gender
##              0.00              0.00                NA                NA
##    actor_5_gender   director_gender   producer_gender           genre_1
##                NA                NA                NA              0.02
##             adult            budget original_language        popularity
##              0.00              0.00              0.00              0.00
##      release_date           revenue           runtime           tagline
##              0.00              0.00              0.00              0.26
##      vote_average        vote_count        collection     num_prod_comp
##              0.00              0.00              0.00              0.00
##     num_prod_ctry     release_month          na_count          quartile
##              0.00              0.00              0.00              0.00
```

Removing the extreme budget records

```r
sd.budget <- sqrt(var(data.imp$budget))
sd.budget*6
```

```
## [1] 220472844
```

```
length(data.imp$budget[data.imp$budget>2e+08])
```

```
## [1] 32
```

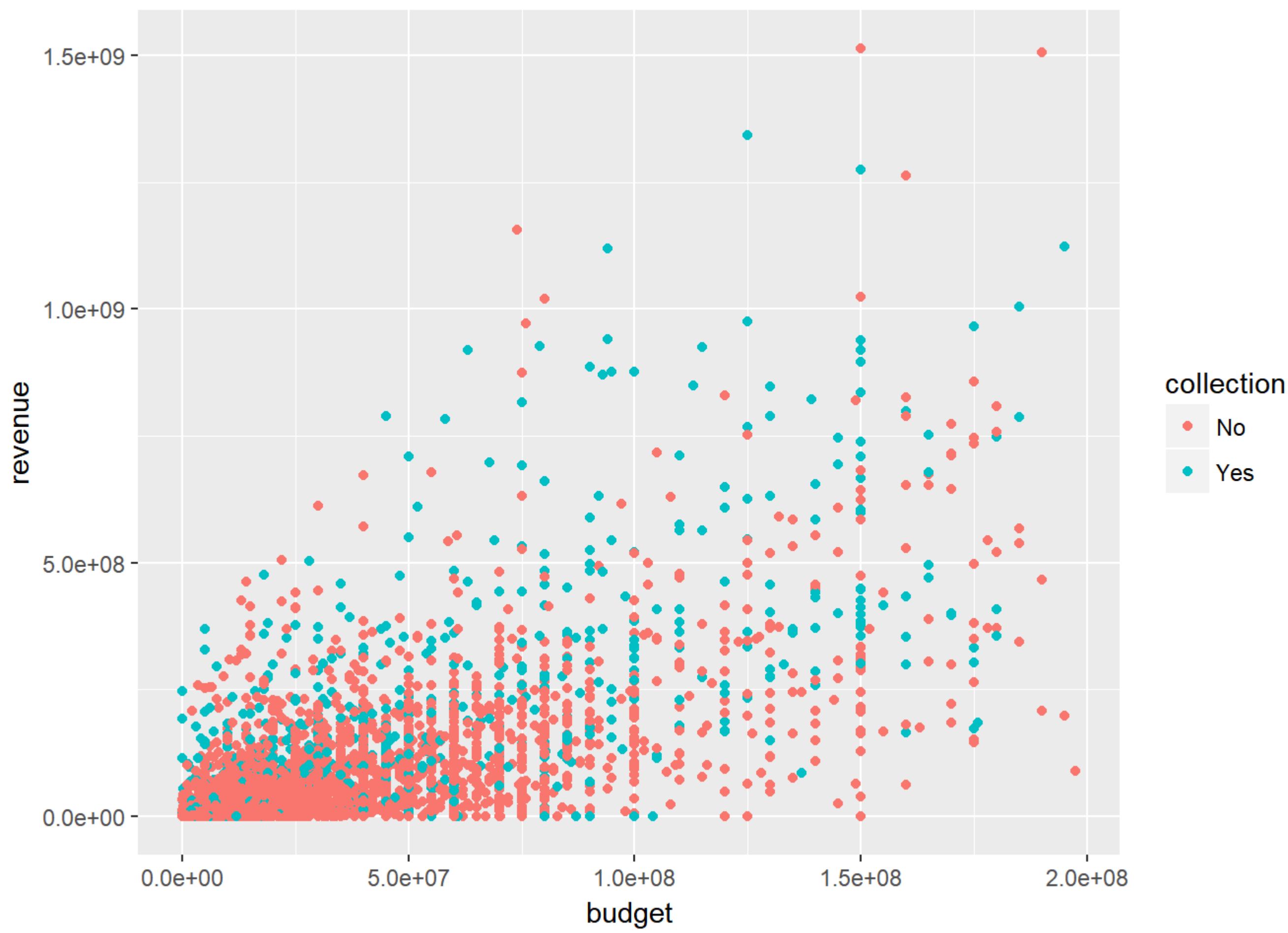
```
data.imp <- data.imp[data.imp$budget<2e+08,]
```

```
data.final <-data.imp[,-which(names(data.imp) %in% c( "actor_2_gender", "actor_3_gender", "actor
_4_gender", "actor_5_gender",  "director_gender", "producer_gender", 'na_count', 'genre_1', 'adu
lt','tagline','original_language'))]
data.final$actor_1_gender <- as.factor(ifelse(data.final$actor_1_gender==2, "Male", "Female"))

#Splitting the data into test & train
c <- round(nrow(data.final)*0.7,0)
s <- sample(1:nrow(data.final), c)

train <- data.final[s,]
test <- data.final[-s,]
```

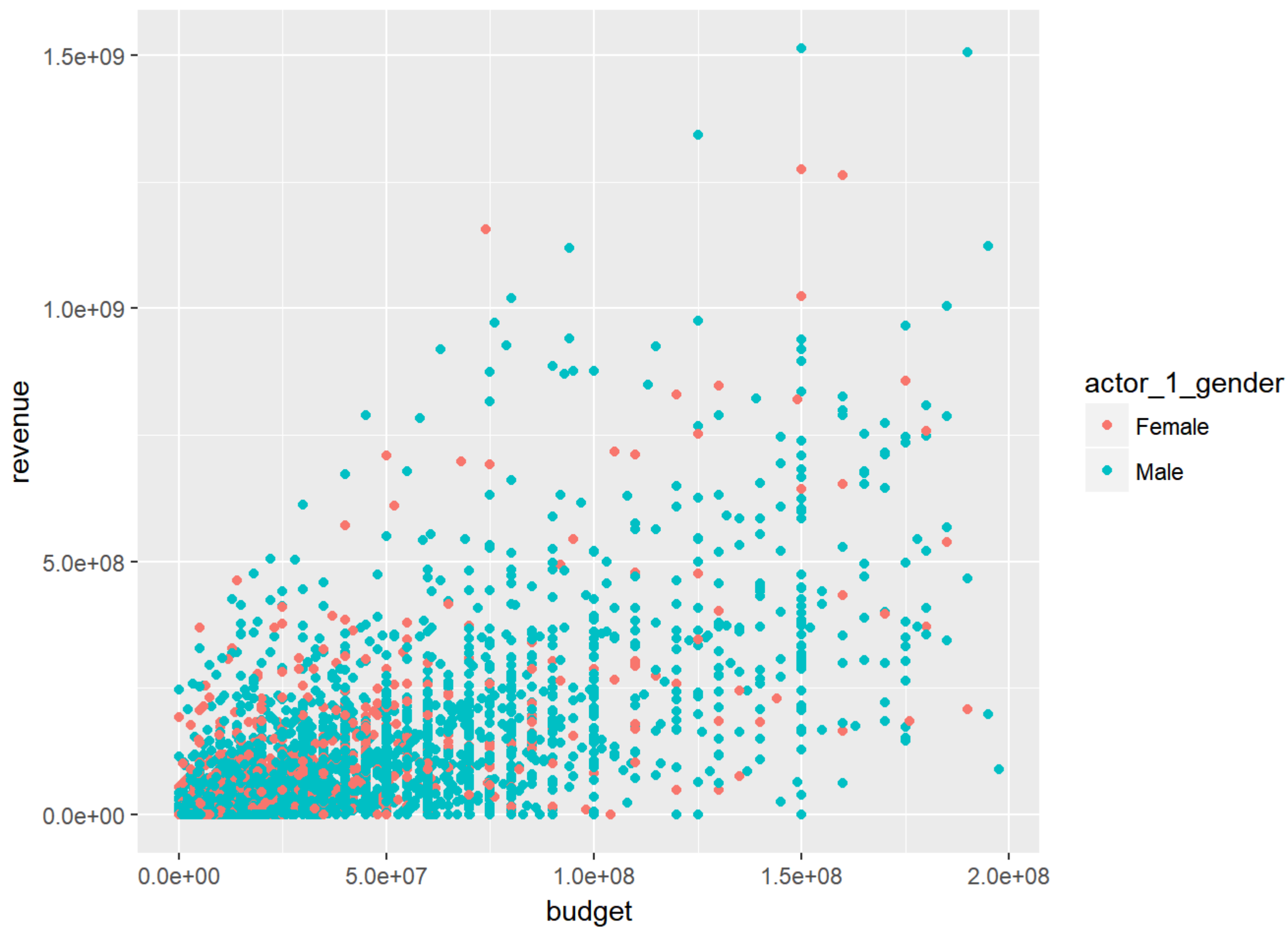
Building a model to predict revenue of the movie before it is released. I will not model the vote count and vote average variables as they are collected after the release of the movie.

Build a multiple linear model for revenue prediction

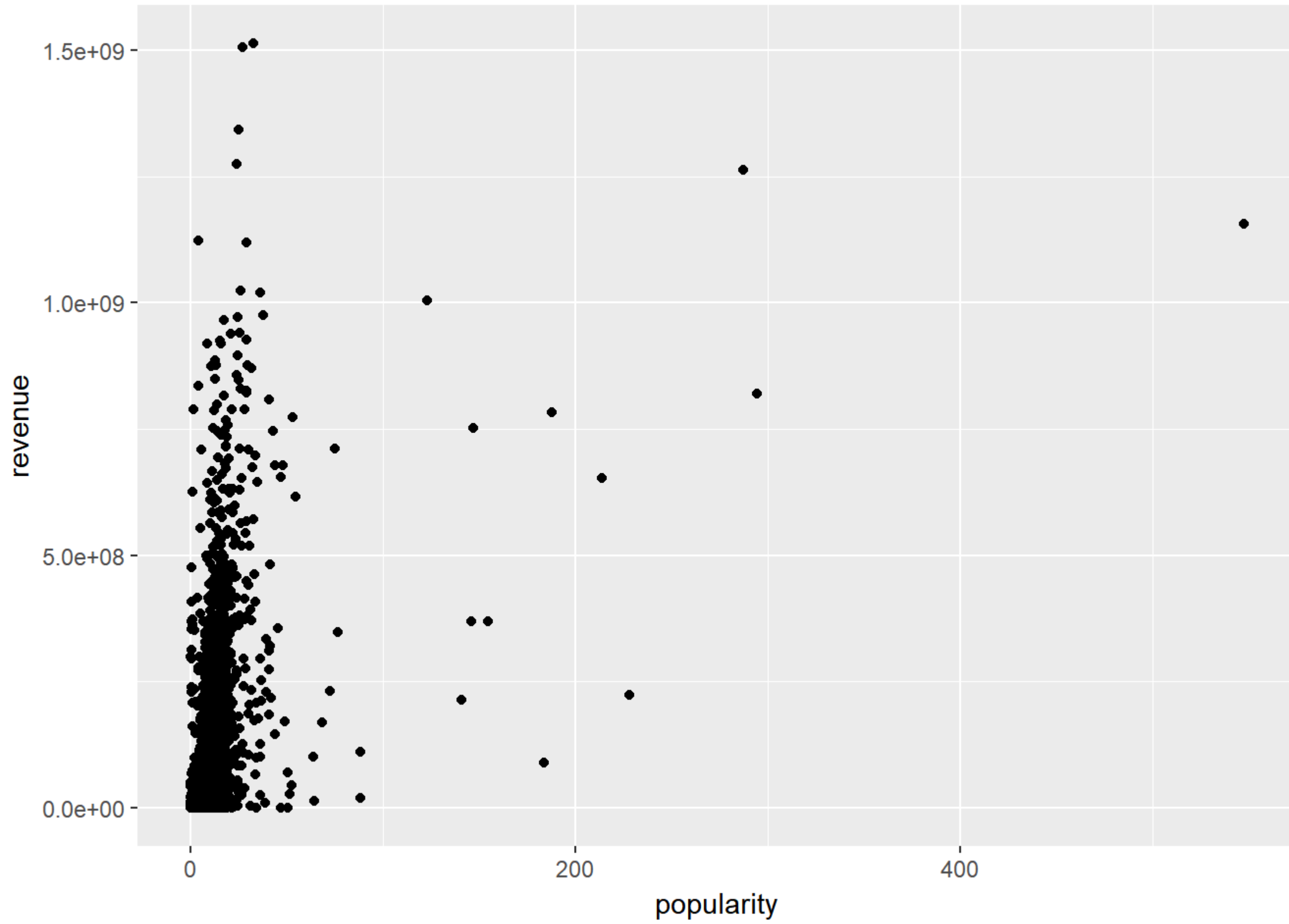
```
ggplot(data.final,aes(budget,revenue,colour = collection)) + geom_point()
```



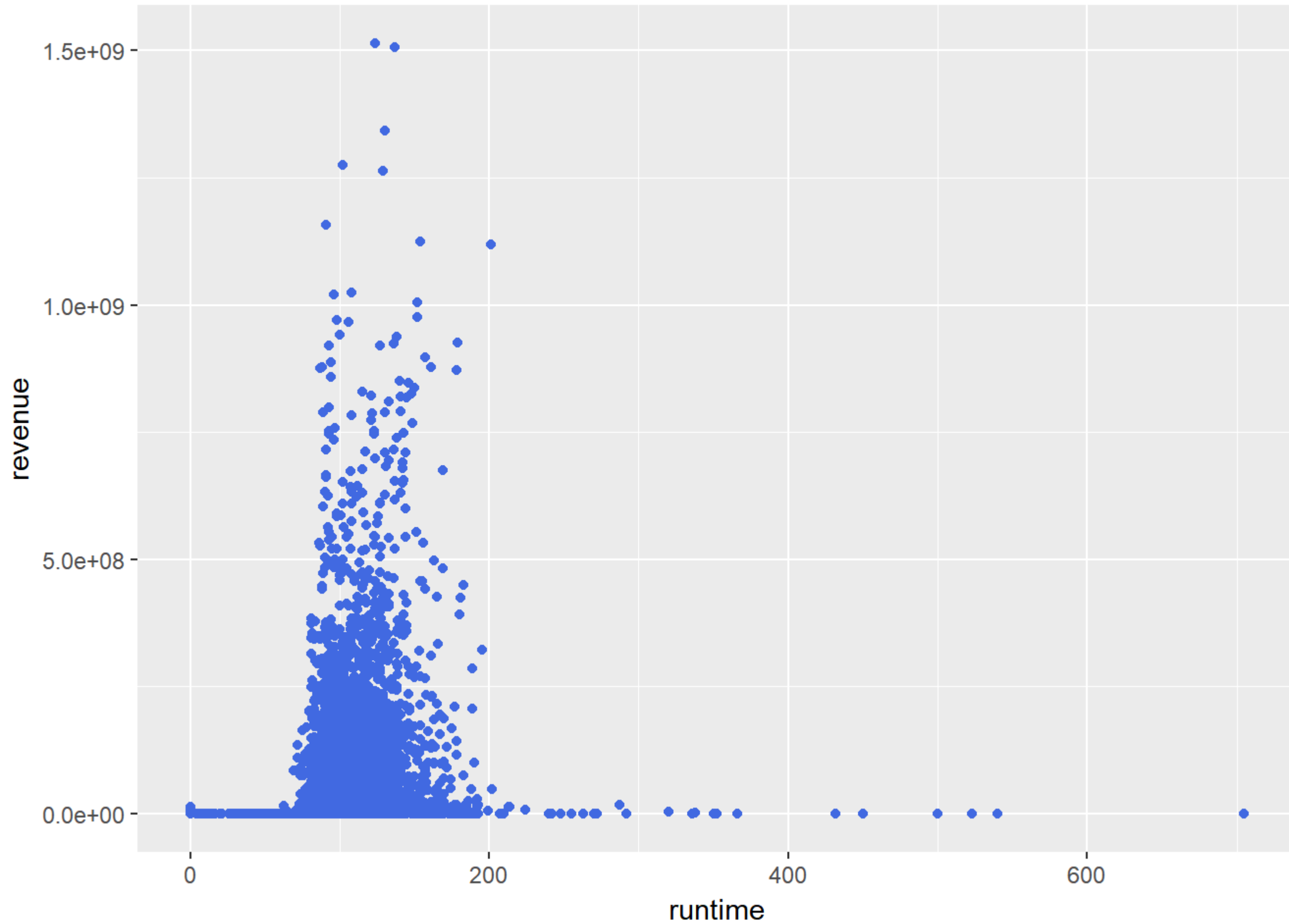

```
ggplot(data.final,aes(budget,revenue,colour = actor_1_gender)) + geom_point()
```



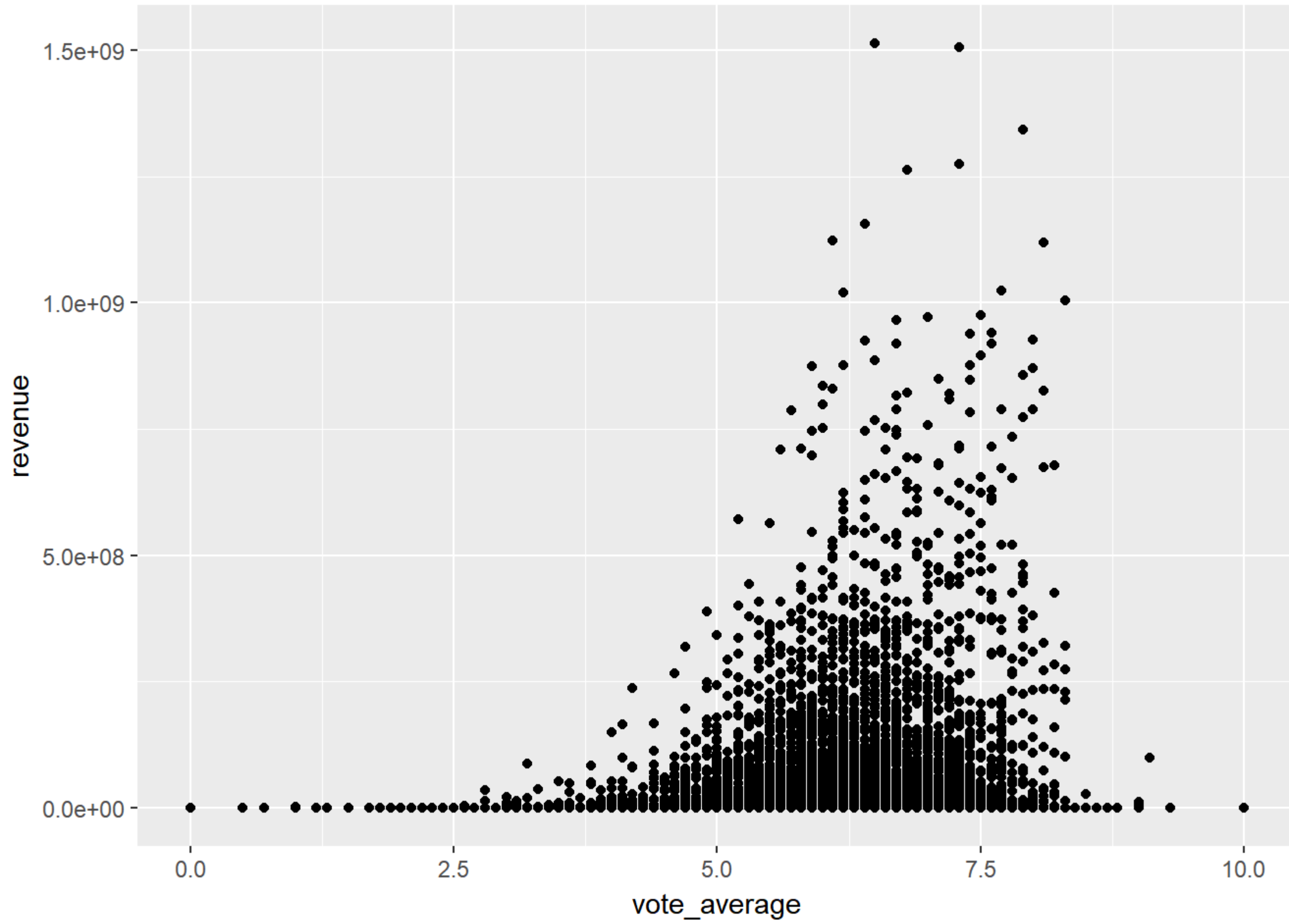

```
ggplot(data.final,aes(popularity,revenue)) + geom_point()
```



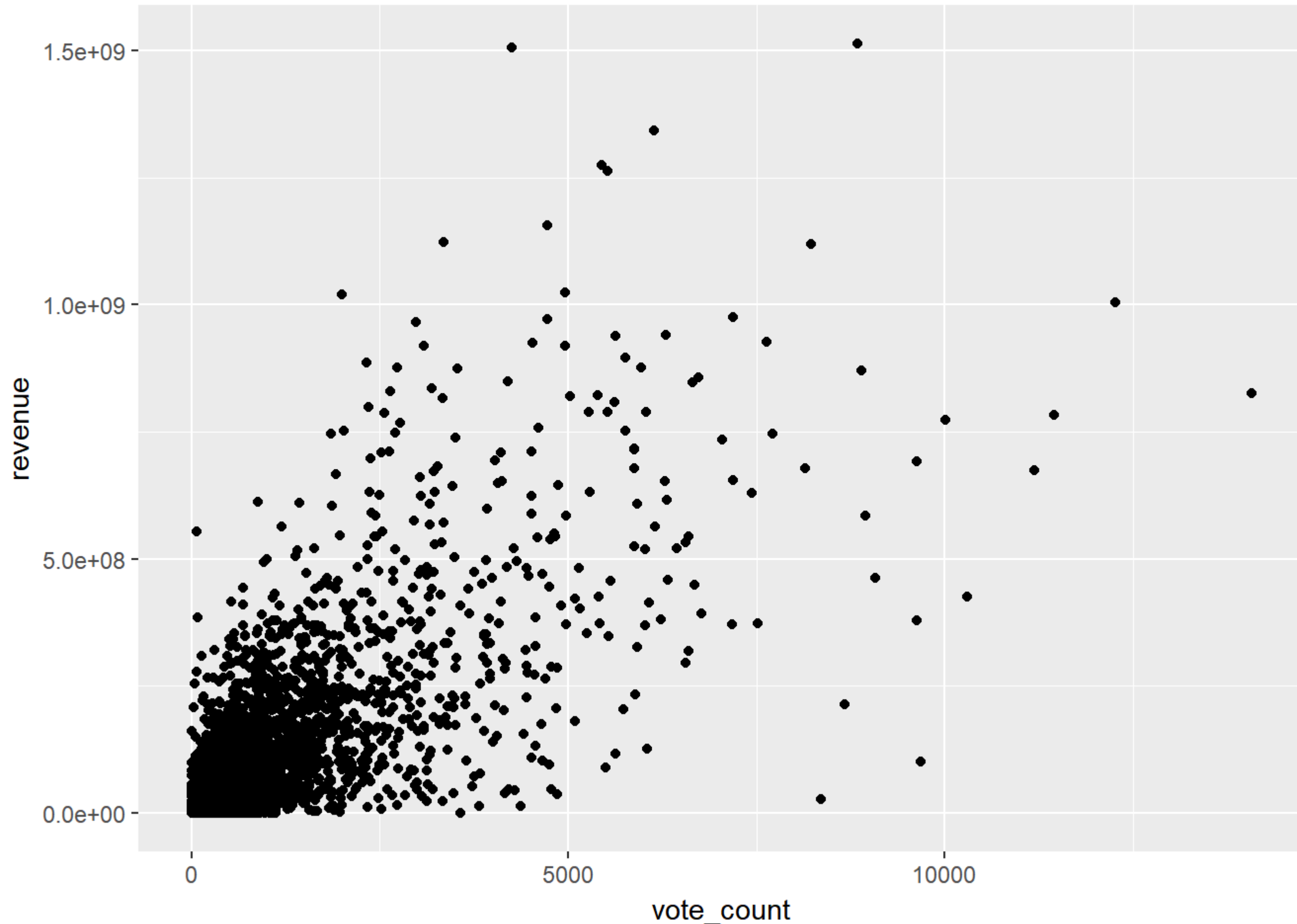

```
ggplot(data.final,aes(runtime,revenue)) + geom_point(color = "royal blue")
```



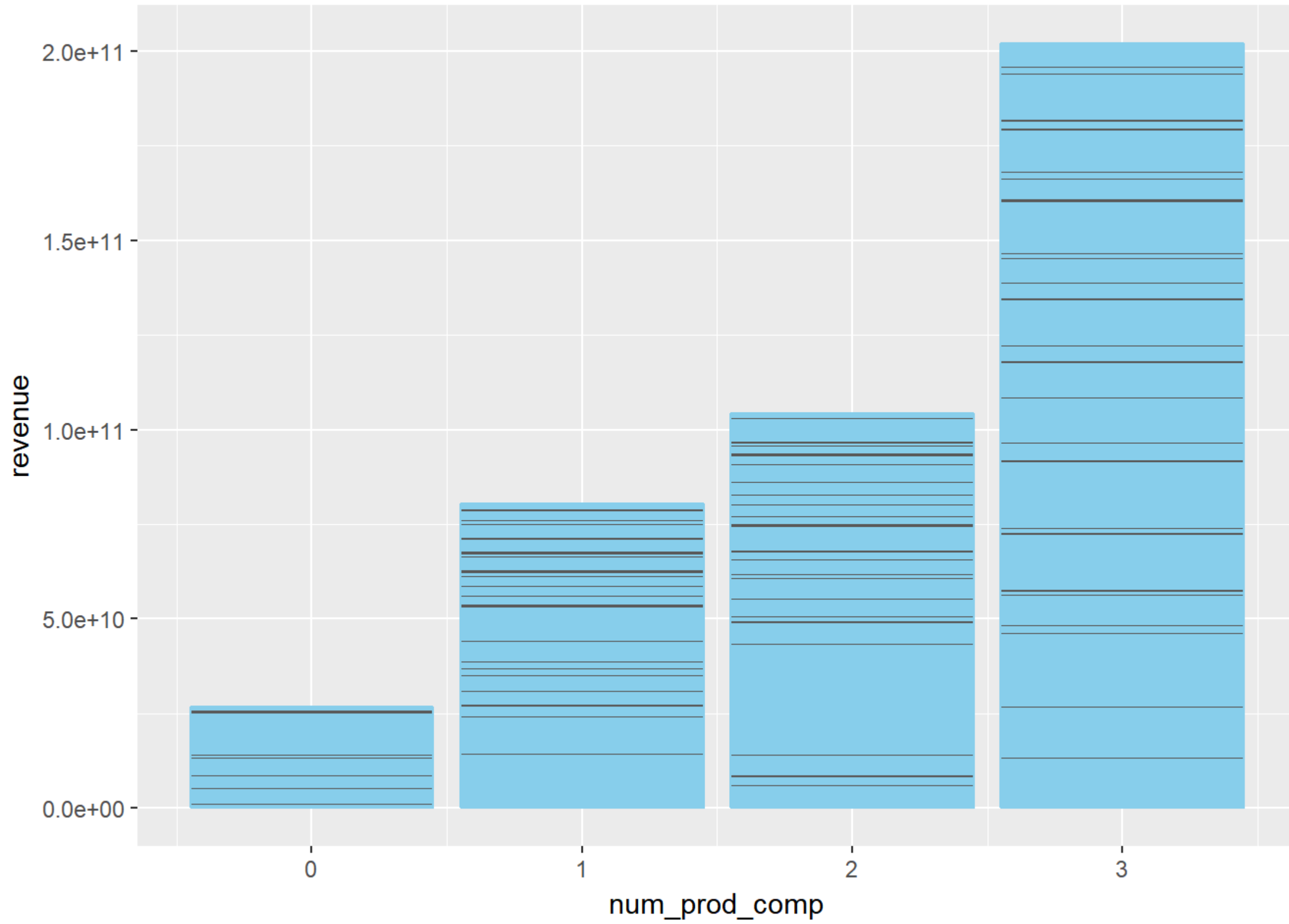

```
ggplot(data.final,aes(vote_average,revenue)) + geom_point()
```



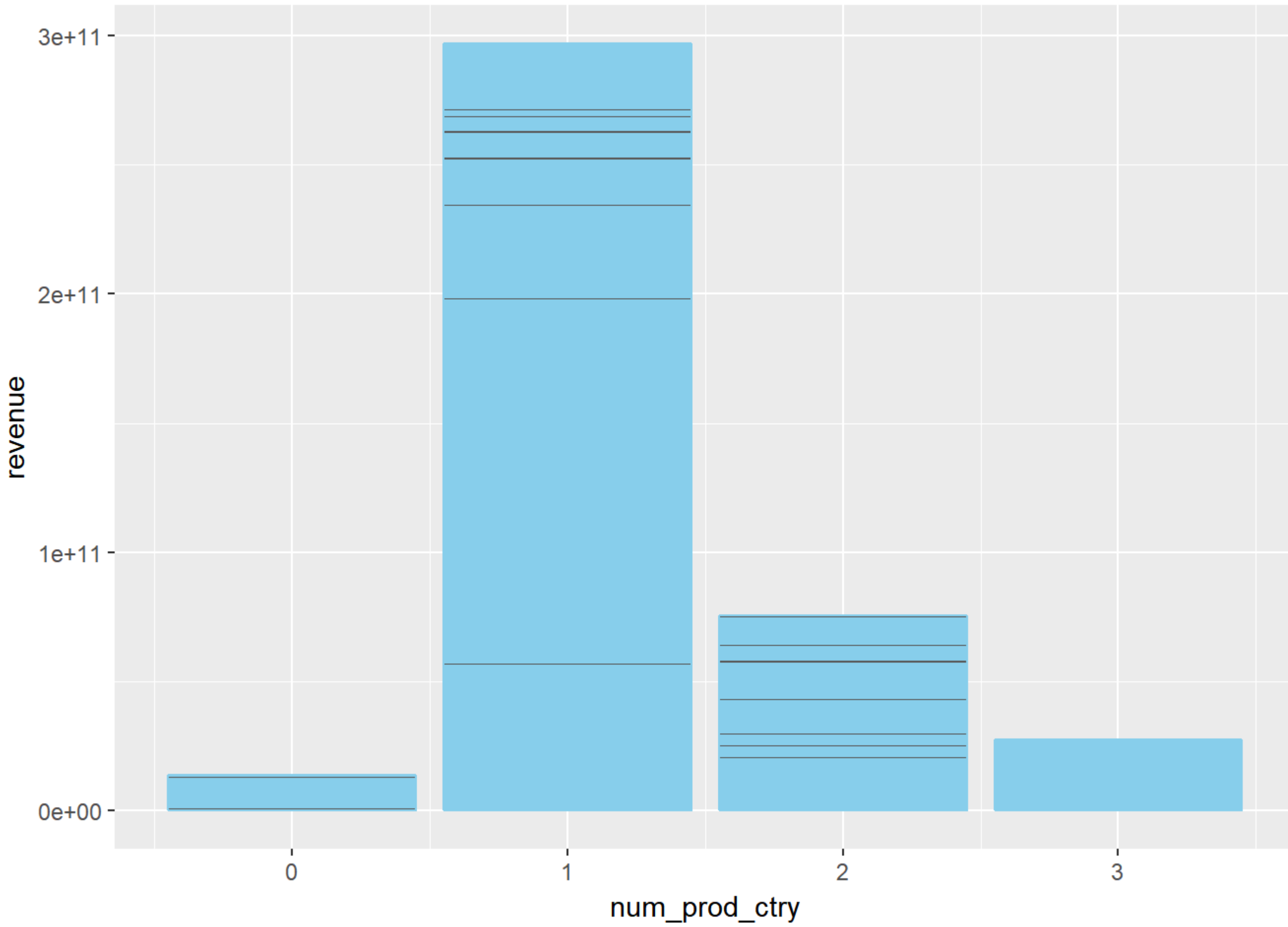

```
ggplot(data.final,aes(vote_count,revenue)) + geom_point()
```



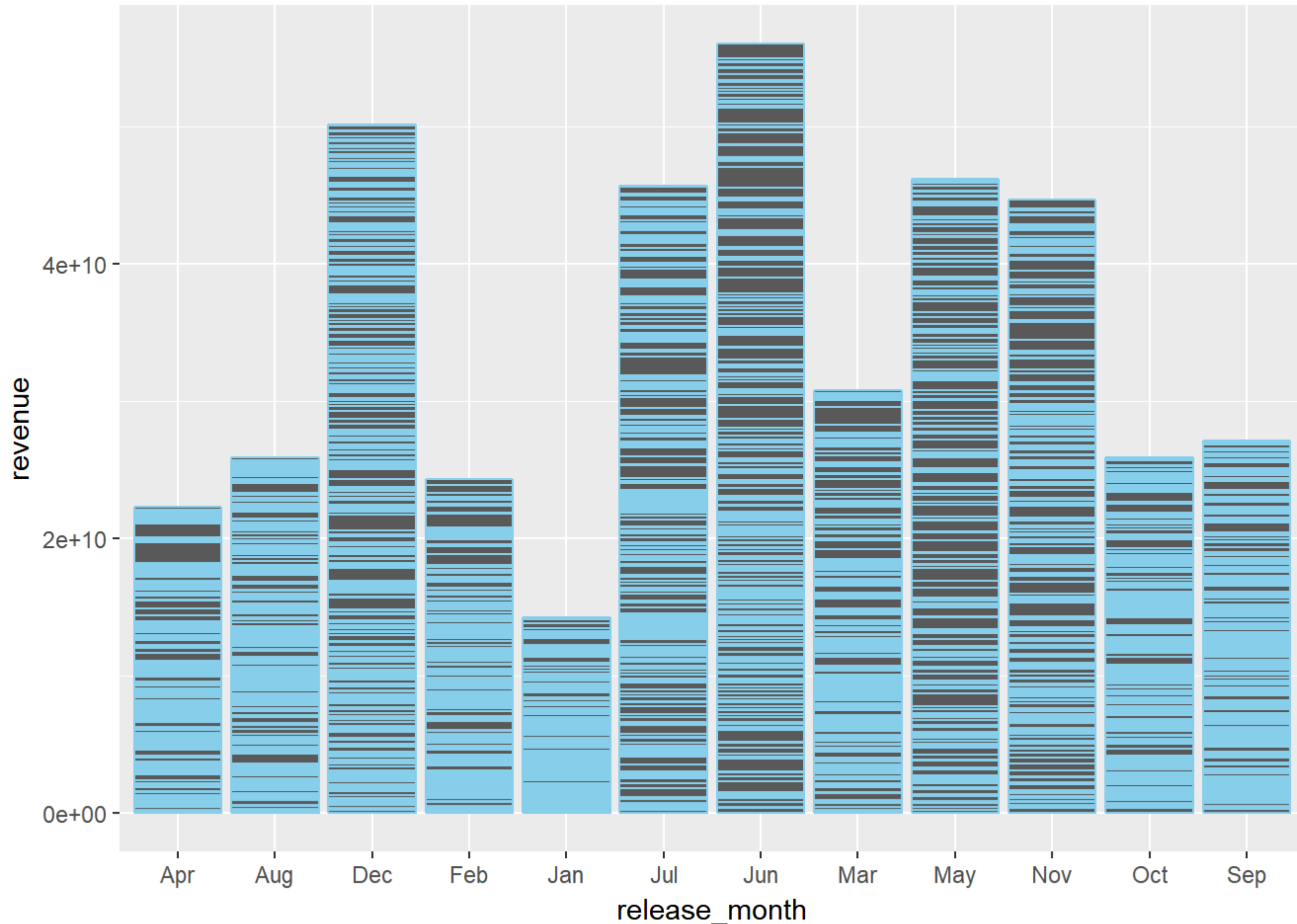

```
ggplot(data.final,aes(num_prod_comp,revenue)) + geom_bar(stat = "identity", color = "sky blue")
```




```
ggplot(data.final,aes(num_prod_ctry ,revenue)) + geom_bar(stat = "identity", color = "sky blue")
```




```
ggplot(data=data.final, aes(x=release_month, y=revenue)) + geom_bar(stat = 'identity', color =
'sky blue')
```




Cleary vote count has an increasing relationship with revenue as that indicates # people who went to watch the movie. But I cannot model vote average and vote count for predicting revenue as they are collected after the movie is released

**Fitting a linear model**

```
revenue_pred <- lm(revenue~ actor_1_gender+ popularity+runtime+collection+num_prod_comp+ budget,
data = train)
summary(revenue_pred)
```

```
##
## Call:
## lm(formula = revenue ~ actor_1_gender + popularity + runtime +
##     collection + num_prod_comp + budget, data = train)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -393935365  -29423923      12089   16745192 1009229798
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -1.775e+07  4.973e+06  -3.568 0.000363 ***
## actor_1_genderMale -3.753e+06  2.673e+06  -1.404 0.160380
## popularity          2.301e+06  9.677e+04  23.781  < 2e-16 ***
## runtime             3.468e+04  4.021e+04   0.862 0.388480
## collectionYes       4.419e+07  3.063e+06  14.426  < 2e-16 ***
## num_prod_comp      -3.420e+06  1.054e+06  -3.245 0.001180 **
## budget              2.410e+00  3.848e-02  62.614  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79960000 on 5040 degrees of freedom
## Multiple R-squared:  0.5926, Adjusted R-squared:  0.5921
## F-statistic:  1222 on 6 and 5040 DF,  p-value: < 2.2e-16
```

```
rmseTest <- rmse((predict(revenue_pred, test)),test$revenue)
rmseTrain <-rmse((predict(revenue_pred, train)),train$revenue)

#Validation
cbind(rmseTrain, rmseTest)
```

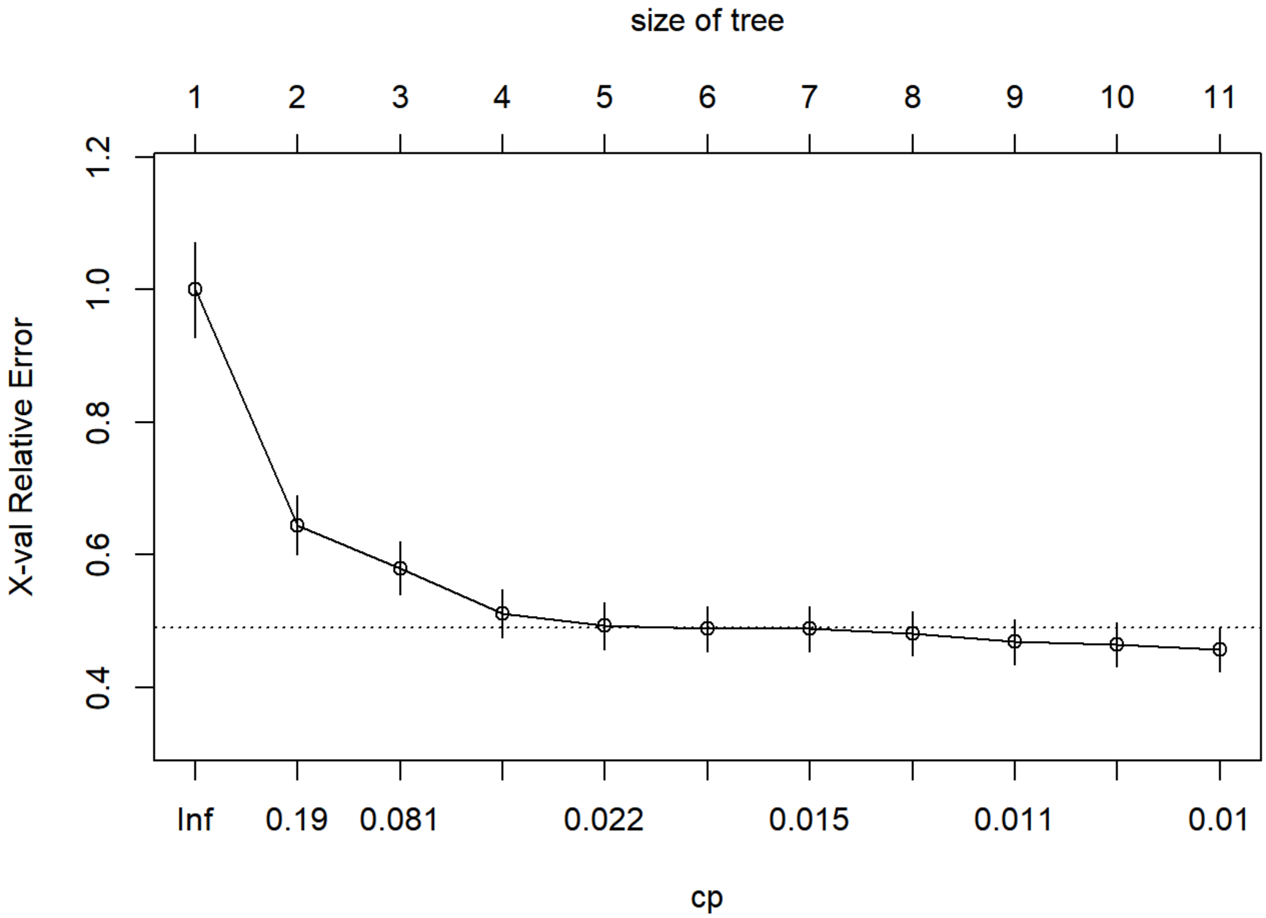
```
##      rmseTrain rmseTest
## [1,]  79909112 82383684
```

The linear model does not give a good fit and the resons could be:

- interaction between variables e.g.: popularity is driven by actors, budget is driven by runtime- number of countries in which the movie is launched
- Scatter plots do not show a perfect linear relationship between the dependent and independent variables

**Fitting a regression tree model** to take into account the interactions between variables

```
revenue_tree <- rpart(revenue~ actor_1_gender+ popularity+runtime+collection+num_prod_comp+num_p
rod_ctry + budget , data = train)

plotcp(revenue_tree)
```

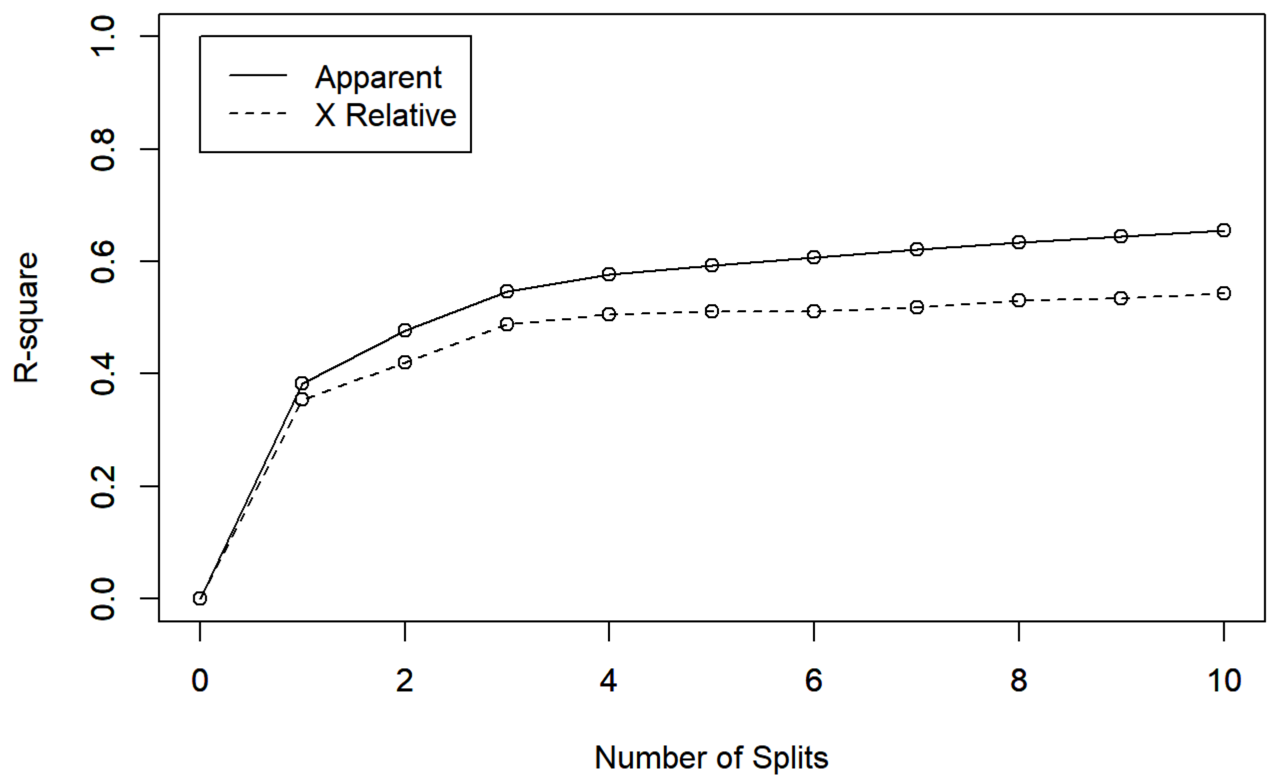


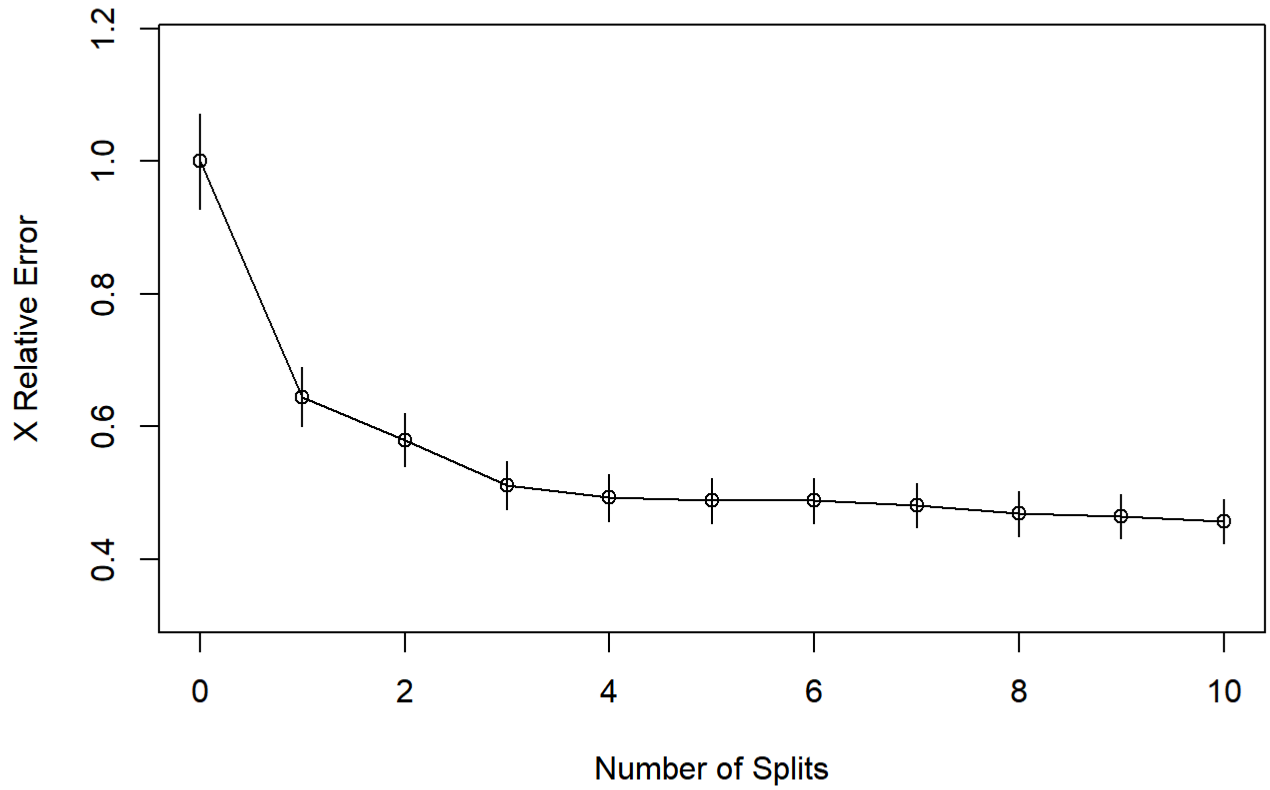

```
x <- printcp(revenue_tree)
```

```
##
## Regression tree:
## rpart(formula = revenue ~ actor_1_gender + popularity + runtime +
##     collection + num_prod_comp + num_prod_ctry + budget, data = train)
##
## Variables actually used in tree construction:
## [1] budget        collection    num_prod_ctry popularity    runtime
##
## Root node error: 7.9107e+19/5047 = 1.5674e+16
##
## n= 5047
##
##          CP nsplit rel error  xerror     xstd
## 1  0.383574      0   1.00000 1.00013 0.071656
## 2  0.093003      1   0.61643 0.64472 0.044729
## 3  0.070956      2   0.52342 0.58030 0.040390
## 4  0.029123      3   0.45247 0.51145 0.036649
## 5  0.016033      4   0.42334 0.49349 0.035326
## 6  0.014639      5   0.40731 0.48846 0.033858
## 7  0.014486      6   0.39267 0.48846 0.033858
## 8  0.011479      7   0.37819 0.48182 0.033231
## 9  0.010907      8   0.36671 0.46912 0.033697
## 10 0.010517      9   0.35580 0.46478 0.033640
## 11 0.010000     10   0.34528 0.45730 0.032872
```

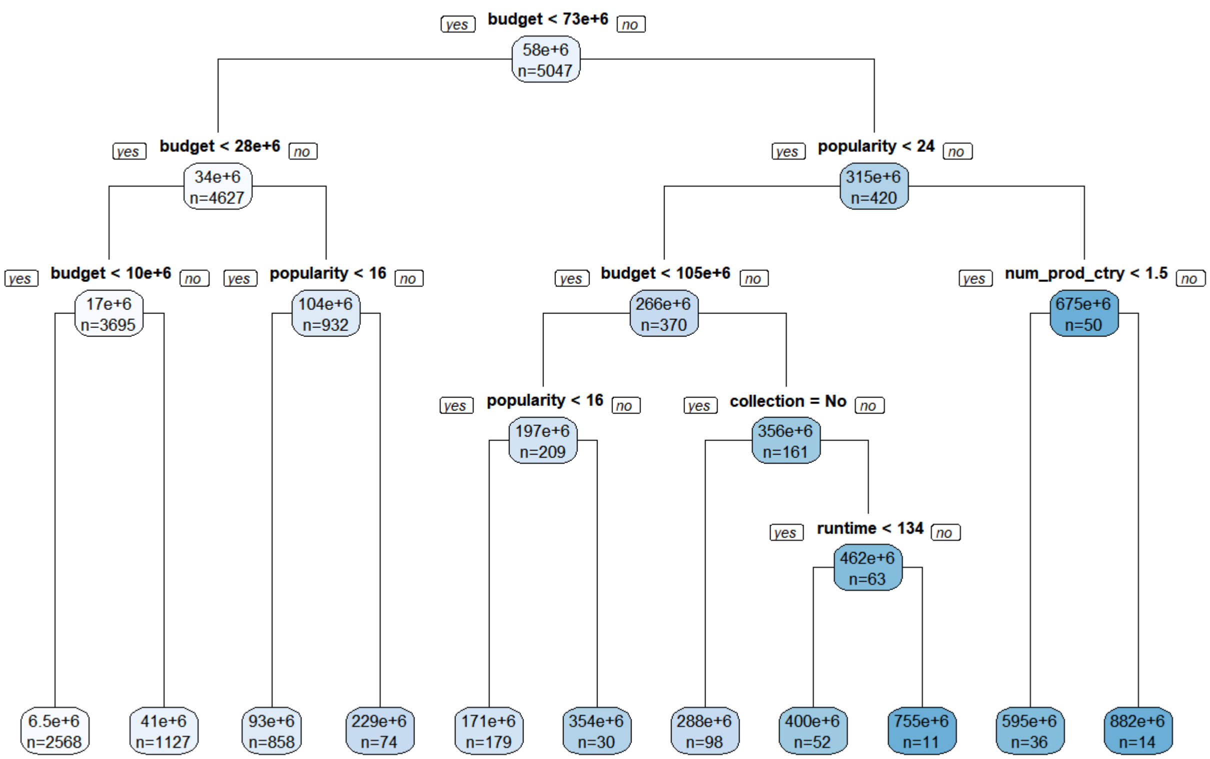
```
rsq.rpart(revenue_tree)
```

```
##
## Regression tree:
## rpart(formula = revenue ~ actor_1_gender + popularity + runtime +
##     collection + num_prod_comp + num_prod_ctry + budget, data = train)
##
## Variables actually used in tree construction:
## [1] budget        collection    num_prod_ctry popularity    runtime
##
## Root node error: 7.9107e+19/5047 = 1.5674e+16
##
## n= 5047
##
##          CP nsplit rel error  xerror     xstd
## 1  0.383574      0   1.00000 1.00013 0.071656
## 2  0.093003      1   0.61643 0.64472 0.044729
## 3  0.070956      2   0.52342 0.58030 0.040390
## 4  0.029123      3   0.45247 0.51145 0.036649
## 5  0.016033      4   0.42334 0.49349 0.035326
## 6  0.014639      5   0.40731 0.48846 0.033858
## 7  0.014486      6   0.39267 0.48846 0.033858
## 8  0.011479      7   0.37819 0.48182 0.033231
## 9  0.010907      8   0.36671 0.46912 0.033697
## 10 0.010517      9   0.35580 0.46478 0.033640
## 11 0.010000     10   0.34528 0.45730 0.032872
```


Apparent
X Relative
R-square
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
Number of Splits

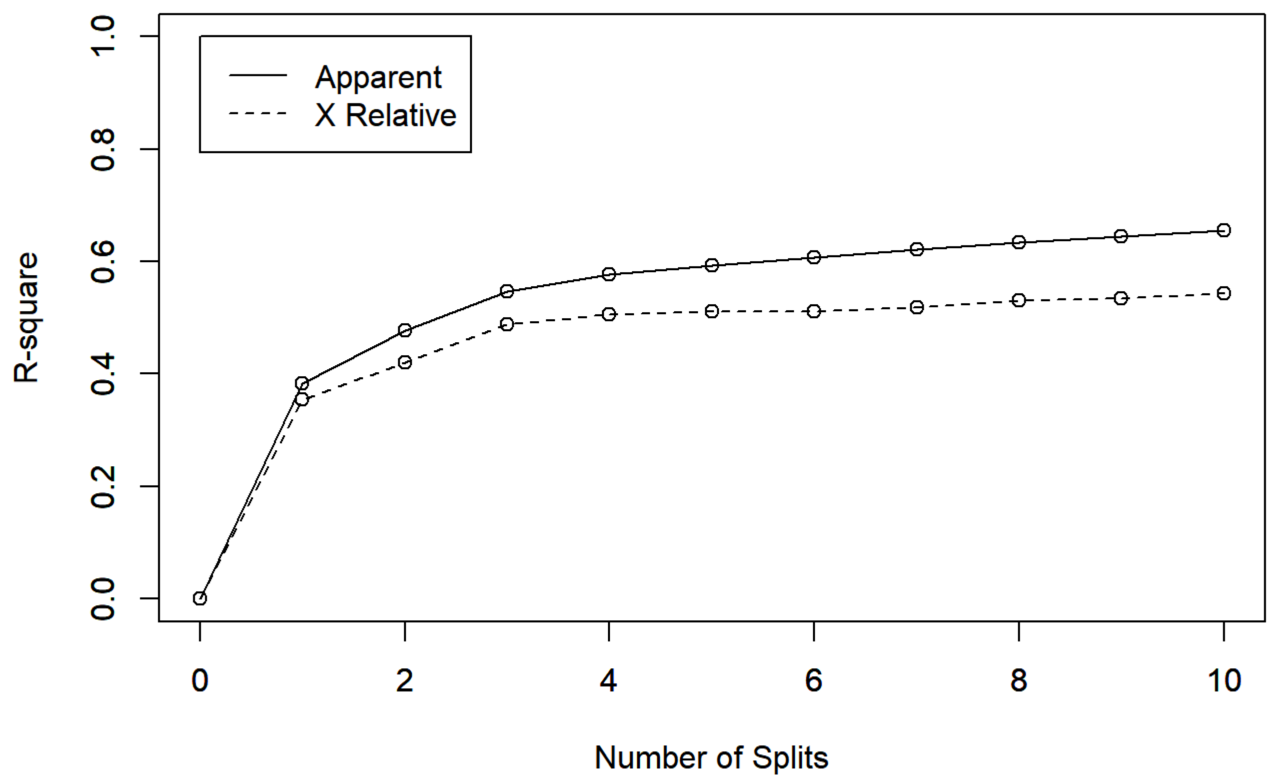

X Relative Error
0.4
0.6
0.8
1.0
1.2
0
2
4
6
8
10
Number of Splits

```
# Retreive optimal cp value based on cross-validated error
opt_index <- which.min(revenue_tree$cptable[, "xerror"])
cp_opt <- revenue_tree$cptable[opt_index, "CP"]

revenue_tree_opt <- prune(tree = revenue_tree,
                          cp = cp_opt)

# Display the pruned tree results
rpart.plot(x = revenue_tree_opt, yesno = 2, type = 1, extra = 1)
```

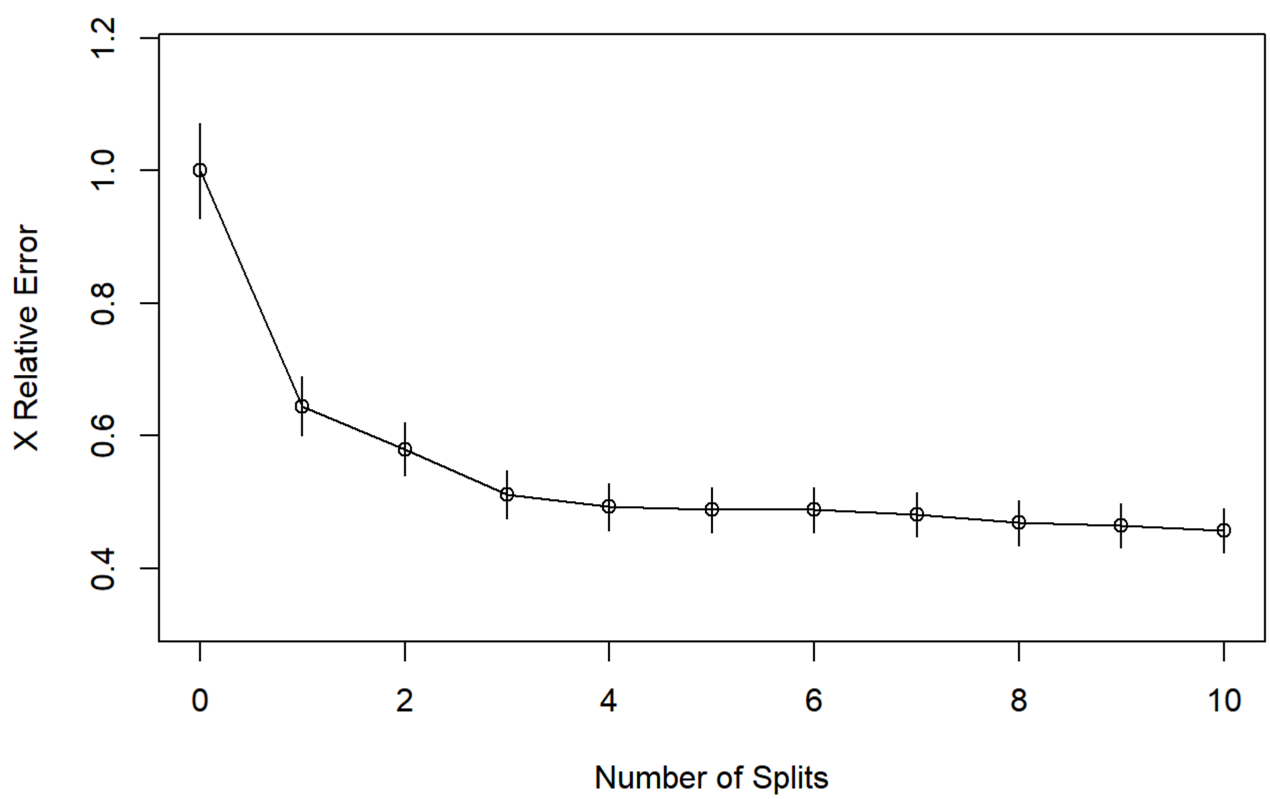



```
rsq.rpart(revenue_tree_opt)
```

```
##
## Regression tree:
## rpart(formula = revenue ~ actor_1_gender + popularity + runtime +
##     collection + num_prod_comp + num_prod_ctry + budget, data = train)
##
## Variables actually used in tree construction:
## [1] budget        collection    num_prod_ctry popularity    runtime
##
## Root node error: 7.9107e+19/5047 = 1.5674e+16
##
## n= 5047
##
##          CP nsplit rel error  xerror     xstd
## 1  0.383574      0   1.00000 1.00013 0.071656
## 2  0.093003      1   0.61643 0.64472 0.044729
## 3  0.070956      2   0.52342 0.58030 0.040390
## 4  0.029123      3   0.45247 0.51145 0.036649
## 5  0.016033      4   0.42334 0.49349 0.035326
## 6  0.014639      5   0.40731 0.48846 0.033858
## 7  0.014486      6   0.39267 0.48846 0.033858
## 8  0.011479      7   0.37819 0.48182 0.033231
## 9  0.010907      8   0.36671 0.46912 0.033697
## 10 0.010517      9   0.35580 0.46478 0.033640
## 11 0.010000     10   0.34528 0.45730 0.032872
```



Apparent
X Relative
R-square
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
Number of Splits

X Relative Error
0.4
0.6
0.8
1.0
1.2
0
2
4
6
8
10
Number of Splits

Looking at the accuracy and confusion matrix from tree model (test vs. train)

```
predictedTest <- predict(revenue_tree_opt, test)
rmseTest <- rmse((test$revenue),predictedTest)
rmseTrain <- rmse((train$revenue),predict(revenue_tree_opt, train))

r_sqTest <- 1-rmseTest^2/var(test$revenue)
r_sqTr <- 1-rmseTrain^2/var(train$revenue)


a<- round(rbind(cbind(rmseTrain, rmseTest), cbind(r_sqTr, r_sqTest)),2)
rownames(a)<- c("RMSE", "R-sq")
colnames(a)<- c("Train", "Test")
a
```

```
##            Train        Test
## RMSE 73566279.81 81099328.11
## R-sq        0.65        0.56
```

The fit of regression tree is similar to the linear regression model. I will fit random forests to see if the accuracy can be improved further by bootstrapping:

```
rfor1 <- randomForest(revenue~ actor_1_gender+ popularity+runtime+collection+num_prod_comp+num_p
rod_ctry + budget, data= train)
```

```
rfor1
```

```
##
## Call:
##  randomForest(formula = revenue ~ actor_1_gender + popularity +      runtime + collection + n
um_prod_comp + num_prod_ctry + budget,      data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 5.067068e+15
##                     % Var explained: 67.67
```

```
rmseTest <- rmse(predict(rfor1, test), test$revenue)
rmseTrain <- rmse(predict(rfor1, train), train$revenue)
Rsq_test <- 1-(rmseTest)^2/var(test$revenue)
R_sqTr <- 1-(rmseTrain)^2/var(train$revenue)

a<- round(rbind(cbind(rmseTrain, rmseTest), cbind(R_sqTr, Rsq_test)),2)
rownames(a)<- c("RMSE", "R-sq")
colnames(a)<- c("Train", "Test")
a
```

```
##             Train        Test
## RMSE 41617858.34 76587561.56
## R-sq        0.89        0.61
```

I see overfitting in this model. Let me set the node size to avoid this:

```
for (i in c(20,25,30,35,40)){

  rfor2 <- randomForest(revenue~ actor_1_gender+ popularity+runtime+collection+num_prod_comp+ bu
dget, data= train, nodesize = i)
  rmseTest <- rmse(predict(rfor2, test), test$revenue)
  rmseTrain <- rmse(predict(rfor2, train), train$revenue)
  Rsq_test <- 1-(rmseTest)^2/var(test$revenue)
  R_sqTr <- 1-(rmseTrain)^2/var(train$revenue)

  a<- round(rbind(cbind(rmseTrain, rmseTest), cbind(R_sqTr, Rsq_test)),2)
  rownames(a)<- c("RMSE", "R-sq")
  colnames(a)<- c("Train", "Test")
  print(i)
  print(a)
}
```

```
## [1] 20
##            Train        Test
## RMSE 55678148.1 75726791.89
## R-sq        0.8        0.62
## [1] 25
##             Train        Test
## RMSE 58042891.53 75725319.85
## R-sq        0.79        0.62
## [1] 30
##             Train        Test
## RMSE 59878822.78 75771324.37
## R-sq        0.77        0.62
## [1] 35
##             Train        Test
## RMSE 61445977.64 75579972.87
## R-sq        0.76        0.62
## [1] 40
##             Train        Test
## RMSE 62836451.57 75594016.01
## R-sq        0.75        0.62
```

I will go with the node size 25. The model fit has clearly improved with random forests compared to the tree model

```
rfor.f <- randomForest(revenue~ actor_1_gender+ popularity+runtime+collection+num_prod_comp+ bud
get, data= train, nodesize = 30)
rfor.f$importance
```

```
##                  IncNodePurity
## actor_1_gender  3.802725e+17
## popularity      1.874705e+19
## runtime         4.798971e+18
## collection      2.967609e+18
## num_prod_comp   1.240090e+18
## budget          3.055979e+19
```

#### Performing cluster wise class regression

```
#clustreg function
clustreg=function(dat,k,tries,sed,niter){

  set.seed(sed)
  dat=as.data.frame(dat)
  rsq=rep(NA,niter)
  res=list()
  rsq.best=0
  for(l in 1:tries) {

    c = sample(1:k,nrow(dat),replace=TRUE)
    yhat=rep(NA,nrow(dat))
    for(i in 1:niter) {
      resid=pred=matrix(0,nrow(dat),k)
      for(j in 1:k){
        pred[,j]=predict(glm(dat[c==j,],family="gaussian"),newdata=dat)
        resid[,j] = (pred[,j]-dat[,1])^2
      }

      c = apply(resid,1,which.min)
      for(m in 1:nrow(dat)) {yhat[m]=pred[m,c[m]]}
      rsq[i] = cor(dat[,1],yhat)^2
      #print(rsq[i])
    }

    if(rsq[niter] > rsq.best) {
      rsq.best=rsq[niter]
      l.best=l
      c.best=c
      yhat.best=yhat
    }
  }

  for(i in k:1) res[[i]]=summary(lm(dat[c.best==i,]))

  return(list(data=dat,nclust=k,tries=tries,seed=sed,rsq.best=rsq.best,number.loops=niter, Best.
try=l.best,cluster=c.best,results=res))
}

clustreg.predict=function(results,newdat){

  yhat=rep(NA,nrow(newdat))
  resid=pred=matrix(0,nrow(newdat),length(table(results$cluster)))

  for(j in 1:length(table(results$cluster))){
    pred[,j]=predict(glm(results$data[results$cluster==j,],family="gaussian"),newdata=newdat)

    resid[,j] = (pred[,j]-newdat[,1])^2
  }

  c = apply(resid,1,which.min)
  for(m in 1:nrow(newdat)) {yhat[m]=pred[m,c[m]]}
  rsq = cor(newdat[,1],yhat)^2
```

```
  return(list(results=results,newdata=newdat,cluster=c,yhat=yhat,rsq=rsq))

}
```

Trying multiple clusters and assessing fits using approximated R-sq and RMSE

```
for (i in 2:5){

  rev_clust<- clustreg(train[,c(5,1,2,3,6,9,10,11)],i,100,881,50)
  ypredTr<- clustreg.predict(rev_clust, train[,c(5,1,2,3,6,9,10,11)])
  ypredTest <- clustreg.predict(rev_clust,test[,c(5,1,2,3,6,9,10,11)])
  yhat_tr<- ypredTr$yhat
  yhat_test<- ypredTest$yhat

  rmseTest <- rmse(yhat_test, test$revenue)
  rmseTrain <- rmse(yhat_tr, train$revenue)
  Rsq_test <- 1-(rmseTest)^2/var(test$revenue)
  R_sqTr <- 1-(rmseTrain)^2/var(train$revenue)

  a<- round(rbind(cbind(rmseTrain, rmseTest), cbind(R_sqTr, Rsq_test)),2)
  rownames(a)<- c("RMSE", "R-sq")
  colnames(a)<- c("Train", "Test")
  print(i)
  x <- table(rev_clust$cluster)
  print(x)
  print(a)

}
```

```
## [1] 2
##
##    1    2
##  493 4554
##            Train        Test
## RMSE 47003363.81 51475109.98
## R-sq        0.86        0.82
## [1] 3
##
##    1    2    3
## 1352 3406  289
##            Train        Test
## RMSE 33288523.42 36322019.25
## R-sq        0.93        0.91
## [1] 4
##
##    1    2    3    4
## 2496  437 1994  120
##            Train        Test
## RMSE 25520717.77 29737408.82
## R-sq        0.96        0.94
## [1] 5
##
##    1    2    3    4    5
##   66 2282 1904  581  214
##            Train        Test
## RMSE 20312198.43 25180686.53
## R-sq        0.97        0.96
```

Cluster wise regression is fitting really well. Based on the size of the clusters, I will choose the 3 cluster solution as clusters get thin beyond that.

```
rev_clust3<- clustreg(train[,c(5,1,2,3,6,9,10,11)],3,100,881,50)
rev_clust3$result
```

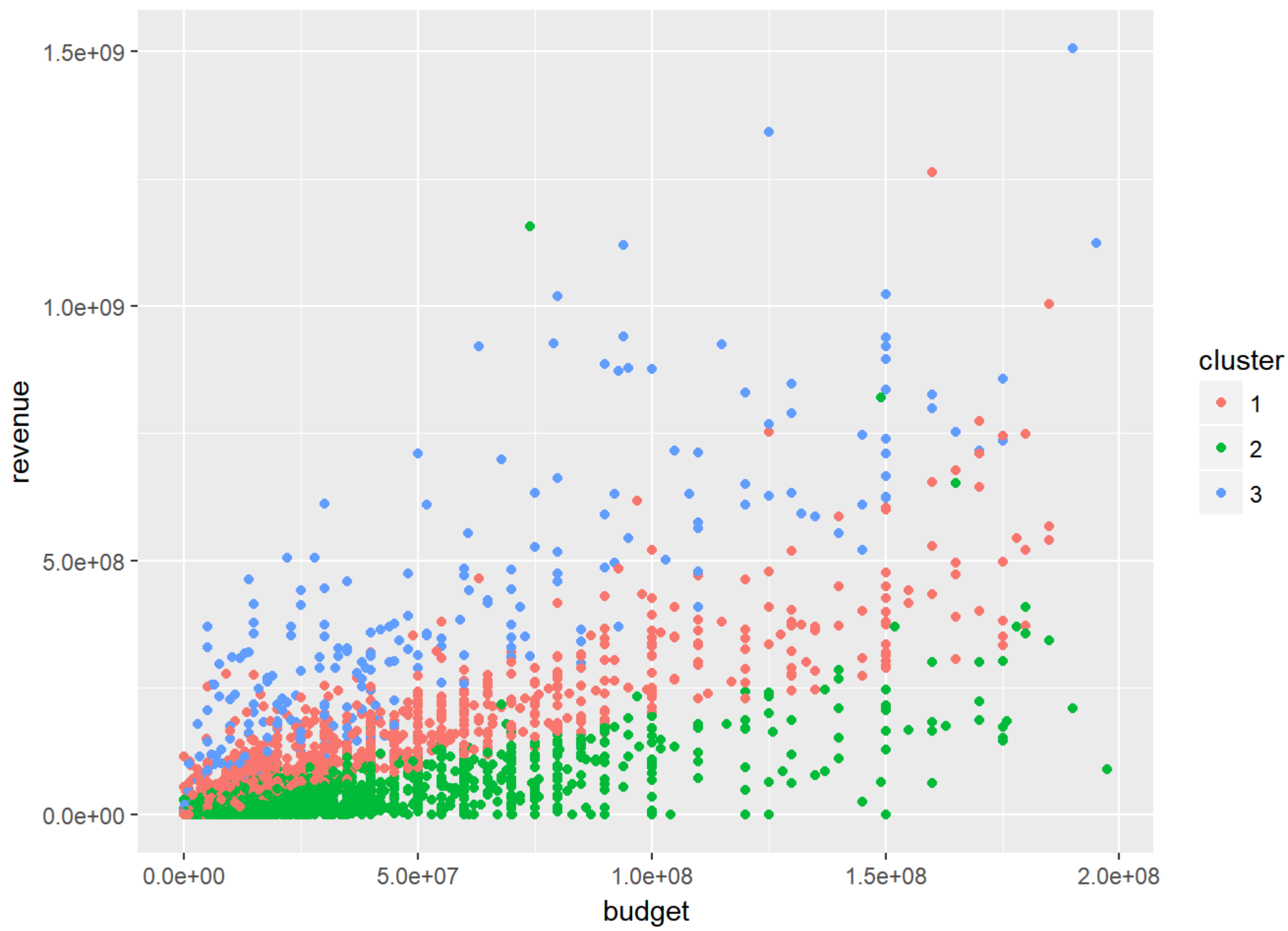
```
## [[1]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -195953616  -13557250   -2453310    8574363  211463384
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -6.636e+07  6.186e+06 -10.727  < 2e-16 ***
## actor_1_genderMale  6.233e+06  2.223e+06   2.804  0.00512 **
## budget              2.202e+00  3.035e-02  72.540  < 2e-16 ***
## popularity          3.742e+06  9.822e+04  38.100  < 2e-16 ***
## runtime             7.593e+05  6.028e+04  12.596  < 2e-16 ***
## collectionYes       4.556e+07  2.798e+06  16.281  < 2e-16 ***
## num_prod_comp      -9.075e+06  1.048e+06  -8.658  < 2e-16 ***
## num_prod_ctry      -1.338e+06  1.570e+06  -0.852  0.39421
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35390000 on 1344 degrees of freedom
## Multiple R-squared:  0.9263, Adjusted R-squared:  0.9259
## F-statistic:  2411 on 7 and 1344 DF,  p-value: < 2.2e-16
##
##
## [[2]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -257013701   -8589500    1256406    8562172  169002814
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -7.632e+06  1.574e+06  -4.850 1.29e-06 ***
## actor_1_genderMale  1.262e+06  8.988e+05   1.404  0.16032
## budget              1.007e+00  1.509e-02  66.699  < 2e-16 ***
## popularity          1.838e+06  2.947e+04  62.382  < 2e-16 ***
## runtime            -1.889e+04  1.172e+04  -1.612  0.10712
## collectionYes       1.077e+07  1.009e+06  10.676  < 2e-16 ***
## num_prod_comp      -1.429e+06  3.779e+05  -3.780  0.00016 ***
## num_prod_ctry      -2.650e+06  6.056e+05  -4.377 1.24e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21600000 on 3398 degrees of freedom
## Multiple R-squared:  0.7786, Adjusted R-squared:  0.7781
## F-statistic:  1707 on 7 and 3398 DF,  p-value: < 2.2e-16
##
```

```
##
## [[3]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -183300228  -44313840   -6877379   12329376  480499914
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -5.472e+07  2.834e+07  -1.931   0.0545 .
## actor_1_genderMale  3.453e+06  1.209e+07   0.286   0.7755
## budget              3.244e+00  1.513e-01  21.448  < 2e-16 ***
## popularity          1.115e+07  9.709e+05  11.479  < 2e-16 ***
## runtime             1.861e+06  2.642e+05   7.042 1.46e-11 ***
## collectionYes       1.006e+08  1.356e+07   7.422 1.38e-12 ***
## num_prod_comp      -5.913e+07  7.430e+06  -7.958 4.32e-14 ***
## num_prod_ctry       1.995e+07  9.189e+06   2.171   0.0308 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 90940000 on 281 degrees of freedom
## Multiple R-squared:  0.9018, Adjusted R-squared:  0.8993
## F-statistic: 368.4 on 7 and 281 DF,  p-value: < 2.2e-16
```

```
train_clus <- as.data.frame(cbind(train, cluster = rev_clust3$cluster))
train_clus$cluster <- as.factor(train_clus$cluster)
cbind(aggregate(train_clus[,c(2,5,3,6,7,8,10)], by = list(train_clus$cluster), mean), size = tab
le(train_clus$cluster))
```

```
##   Group.1   budget   revenue popularity  runtime vote_average vote_count
## 1       1 29919614 100653381   8.103021 103.9238     6.014127   773.6450
## 2       2 19020229  19558555   6.967491 105.7769     5.884234   260.8626
## 3       3 44153409 307385431  10.028977 104.8512     6.380969  1703.5917
##   num_prod_comp size.Var1 size.Freq
## 1      1.998521         1      1352
## 2      1.655608         2      3406
## 3      2.429066         3       289
```

```
ggplot(train_clus,aes(budget,revenue,colour = cluster)) + geom_point()
```




Fitting a classification tree on train data to classify new datasets into clusters for revenue predictions through clusterwise regression results

```
s <- sample(1:nrow(train), round(0.7*nrow(train),0))
train1 <- train_clus[s,]
train2 <- train_clus[-s,]

  classify <- randomForest(cluster~ actor_1_gender+ popularity+runtime+collection+num_prod_comp+
 budget, data= train1, nodesize = 10)
  train1clus <- predict(classify, train1, type = "class")
  train2clus <- predict(classify, train2, type = "class")
  round(prop.table(table(actual = train1$cluster, pred = train1clus),1),2)
```

```
##       pred
## actual    1    2    3
##      1 0.55 0.45 0.00
##      2 0.01 0.99 0.00
##      3 0.22 0.61 0.17
```

```
  round(prop.table(table(actual = train2$cluster, pred = train2clus),1),2)
```

```
##       pred
## actual    1    2    3
##      1 0.35 0.64 0.00
##      2 0.04 0.95 0.00
##      3 0.33 0.61 0.06
```

```
accTrain1 <- sum(train1clus==train1$cluster)/nrow(train1); accTrain1
```

```
## [1] 0.8264931
```

```
accTrain2 <- sum(train2clus==train2$cluster)/nrow(train2); accTrain2
```

```
## [1] 0.7430647
```

The accuracy of classification model for deciding clusters is low for small sized clusters. I would weigh in this factor to decide between random forest and cluster wise regression result. The accuracy of predicting revenue once the clusters are identified is very high but when combined with process of identifying the right clusters, expected accuracy drop close to that of random forests.