

## Johnpaul Ogah(jogah3)

```
library("ggplot2")
library("plyr")
library(tm)
library(magrittr)
library(wordcloud)
library(chron)
library(scales)
library(GGally)
library(reshape2)
library(dplyr)
library(stringr)
library(timeDate)
library(lubridate)
library("cowplot")
library("gridExtra")
library("cowplot")

load("movies_merged")
movie_only_data = subset(movies_merged, Type == "movie")
movie_only_data = subset(movie_only_data, is.na(Budget)==FALSE)
movie_only_data = subset(movie_only_data, is.na(Gross)==FALSE)
movie_only_data$profit = movie_only_data$Gross - movie_only_data$Budget
movie_only_data = subset(movie_only_data, Runtime != "N/A")
movie_only_data=subset(movie_only_data, Metascore != "N/A")
movie_only_data$Metascore = sapply(movie_only_data$Metascore, as.numeric)

convert.to.numeric = function(x){
  str = strsplit(x , " ")[[1]]
  len = length(str)
  num = 0;
  if (len == 4){
    num = as.numeric(str[1]) * 60 + as.numeric(str[3])
  }
  if (len == 2){
    if ( str[2] == "h"){
      num = as.numeric(str[1]) * 60
    }
    if( str[2] == "min"){
      num = as.numeric(str[1])
    }
  }
  return(num)
}

extract.numbers = function(x){
  num = str_extract_all(x,"\\((?[0-9]+\\)?)")[[1]]
  num= as.numeric(num)
  num = (sum(num))
  return(num)
}
```

```

calculate_rmse = function(x , y){
  error = x - y
  result = sum(error * error)
  return(result/length(x))
}

ans = data.frame(ldply(movie_only_data[, 'Runtime'], convert.to.numeric))
movie_only_data$Runtime = ans$V1
movie_only_data = na.omit(movie_only_data)
movie_only_data = subset(movie_only_data, Year >= 2000)
ans = ldply(movie_only_data[, "Awards"], extract.numbers)
movie_only_data$Awards = ans$V1
movie_only_data$is_budget_greater_than_3m = movie_only_data$Budget > 3e6

small_movie = movie_only_data[sample(1:dim(movie_only_data)[1], 500),]

```

Use linear regression to predict profit based on all available numeric variables. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

In figure 1.0, the mean square error decreases on the testing data as the training size increases. The best mean square error obtained was 8.088929e+15.

```

train_set = seq(5, 95, by=5)

mse.on.test.data_1 = rep(0, 19)
mse.on.train.data_1 = rep(0, 19)
train_mse = rep(0, 10)
test_mse = rep(0, 10)
n_row = nrow(movie_only_data)
best_mse_on_test = 1.797693e+308
best_train_size = 0
best_mse_on_train = 1.797693e+308
set.seed(50)

for (num in seq(1:19)){
  for(i in seq(1:10)){
    local_var = train_set[num]
    local_var = local_var/100
    random_perm = sample(n_row, n_row)
    first_index = random_perm[1:floor(n_row*local_var)]
    second_index = random_perm[(floor(n_row * local_var)+1):n_row]
    train_data = movie_only_data[first_index,]
    test_data = movie_only_data[second_index,]
    data = train_data
    model = lm(profit~Year+Runtime+Budget+Awards+
               imdbVotes+tomatoReviews+tomatoRotten+
               tomatoUserMeter+tomatoUserReviews+
               tomatoUserRating+tomatoMeter+imdbRating+Metascore, data)
    train_mse[i] = mean(residuals(model)^2)
  }
}

```

```

data = test_data
test_mse[i] = calculate_rmse(test_data$profit, predict(model,data))
}
mse.on.train.data_1[num] = mean(train_mse)
mse.on.test.data_1[num] = mean(test_mse)

if (mean(test_mse) < best_mse_on_test){
  best_mse_on_test = mean(test_mse)
  best_model=model
}

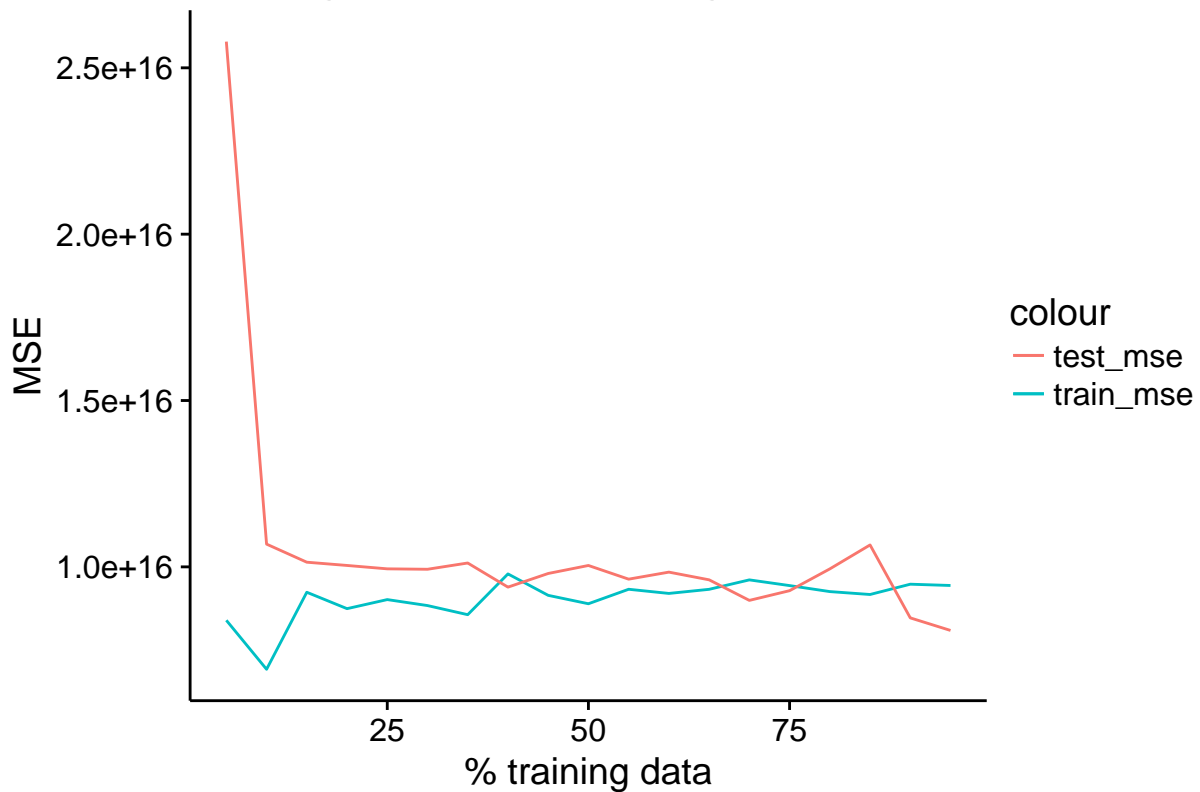
if(mean(train_mse) < best_mse_on_train){
  best_mse_on_train = mean(train_mse)
}
}

result = data.frame(training_data=train_set,
                    mse_train_data=mse.on.train.data_1,
                    mse_test_data=mse.on.test.data_1)

ggplot(result,aes(x=training_data)) +
  geom_line(aes(y=mse_train_data,color="train_mse")) + geom_line(aes(y=mse_test_data,color="test_mse")) +
  ylab("MSE") + xlab("% training data") +
  ggtitle("Fig 1.0 : MSE vs Training data size")

```

**Fig 1.0 : MSE vs Training data size**



```
cat("best MSE on test data:", best_mse_on_test)

## best MSE on test data: 8.088929e+15

cat("\n")

cat("best MSE on train data:", best_mse_on_train)

## best MSE on train data: 6.920891e+15
```

Try to improve the prediction quality in (1) as much as possible by adding feature transformations of the numeric variables. Explore both numeric transformations such as power transforms and non-numeric transformations of the numeric variables like binning (e.g., `is_budget_greater_than_3M`). Explain which transformations you used and why you chose them. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

The code for the question is shown below. A non linear relationship exist between `tomatoReviews` and `profit` as result the particular feature was transformed to account for the non linear relationship. Similarly `imdbRating` was transformed to account for non linear relationship. Also there exist a linear relationship between the log of `tomatoUserReviews` and `profit`. Thus `tomatoUserReviews` was transformed with the log function. Further, I explored the interaction between the various variables with `imdbVotes` and `Budget`. `imdbVotes` and `Budget` are two features that are most correlated with `profit`, hence I explored interaction between the remaining features with the two variables. Figure 2.0 shows the plot of train and test MSE as a function of train set size. The best MSE obtained was  $6.279968e+15$  which is an improvement of  $1.808961e+15$  less than what was obtained in question one. Fig 2.1 graph both test and train MSE for question 1 and question 2 we can see the obvious improvement obtained in question 2 for both the test and train MSE.

```
with(small_movie, cor(Year, profit))

## [1] 0.1157028

with(small_movie, cor(Runtime, profit))

## [1] 0.2689541

with(small_movie, cor(Budget, profit))

## [1] 0.6043316

with(small_movie, cor(Awards, profit))

## [1] 0.2669767

with(small_movie, cor(imdbVotes, profit))

## [1] 0.6583108

with(small_movie, cor(tomatoReviews, profit))

## [1] 0.5030693

with(small_movie, cor(tomatoRotten, profit))

## [1] 0.1241787

with(small_movie, cor(tomatoUserMeter, profit))

## [1] 0.2247348
```

```

with(small_movie,cor(tomatoUserReviews,profit))

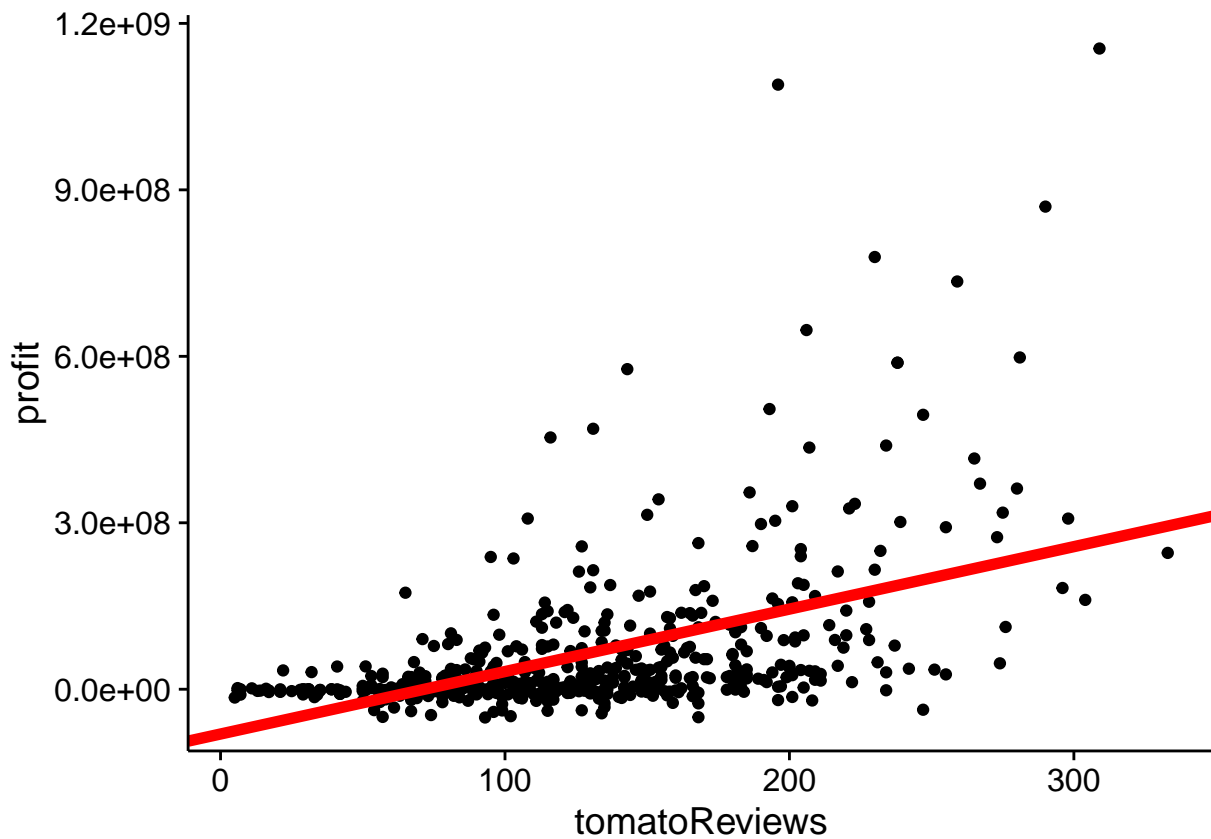
## [1] 0.165001
with(small_movie,cor(tomatoUserRating,profit))

## [1] 0.2795078
with(small_movie,cor(tomatoMeter,profit))

## [1] 0.1834199
with(small_movie,cor(imdbRating,profit))

## [1] 0.2262261
M1 = lm(profit ~tomatoReviews,small_movie)
theta1 = coef(M1)
print(ggplot(small_movie,aes(tomatoReviews,profit)) + geom_point() + geom_abline(intercept = theta1[1],

```



```

summary(M1)$r.squared

## [1] 0.2530787
mean(residuals(M1)^2)

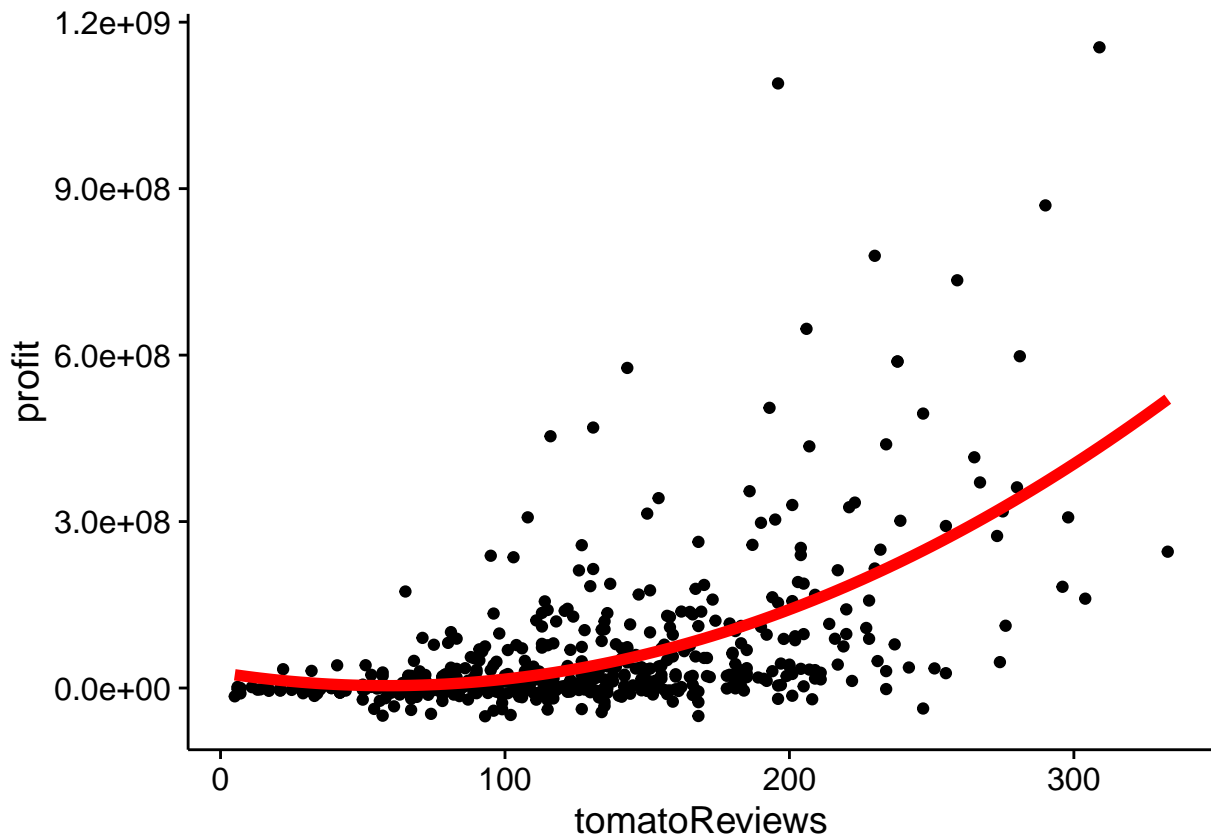
## [1] 1.425114e+16
M2 = lm(profit~tomatoReviews+I(tomatoReviews^2),small_movie)
cX = seq(5,range(small_movie$tomatoReviews)[2],length=500)

```

```

theta2 = coef(M2)
cY = theta2[1] + theta2[2]*cX + theta2[3]*cX^2
print(ggplot(small_movie,aes(tomatoReviews,profit)) + geom_point() + geom_line(aes(x=cX,y=cY), size=2,color="red"))

```



```
summary(M2)$r.squared
```

```
## [1] 0.3176423
```

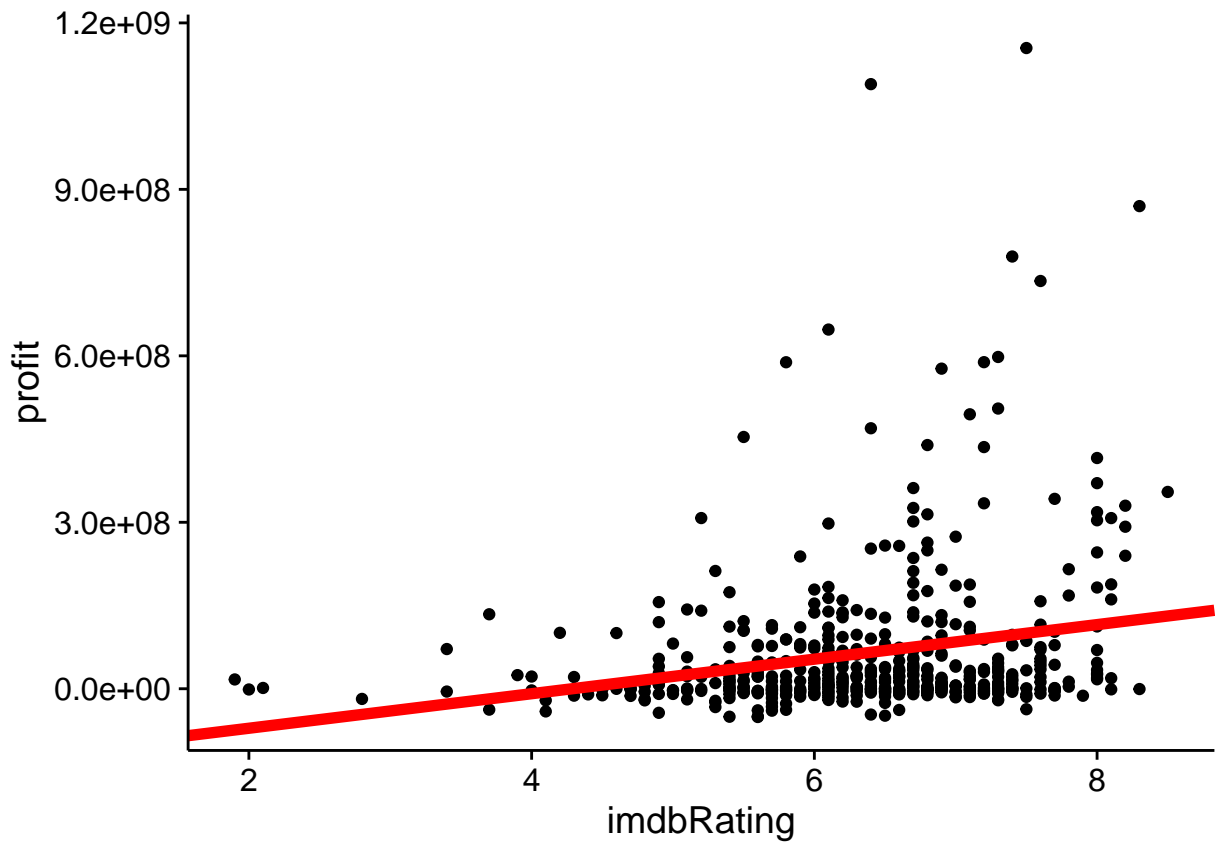
```
mean(residuals(M2)^2)
```

```
## [1] 1.301928e+16
```

```

M3 = lm(profit ~imdbRating,small_movie)
theta3 = coef(M3)
print(ggplot(small_movie,aes(imdbRating,profit)) +
      geom_point() +
      geom_abline(intercept = theta3[1],
                  slope = theta3[2], size=2,color="red"))

```



```
summary(M3)$r.squared
```

```
## [1] 0.05117825
```

```
mean(residuals(M3)^2)
```

```
## [1] 1.810337e+16
```

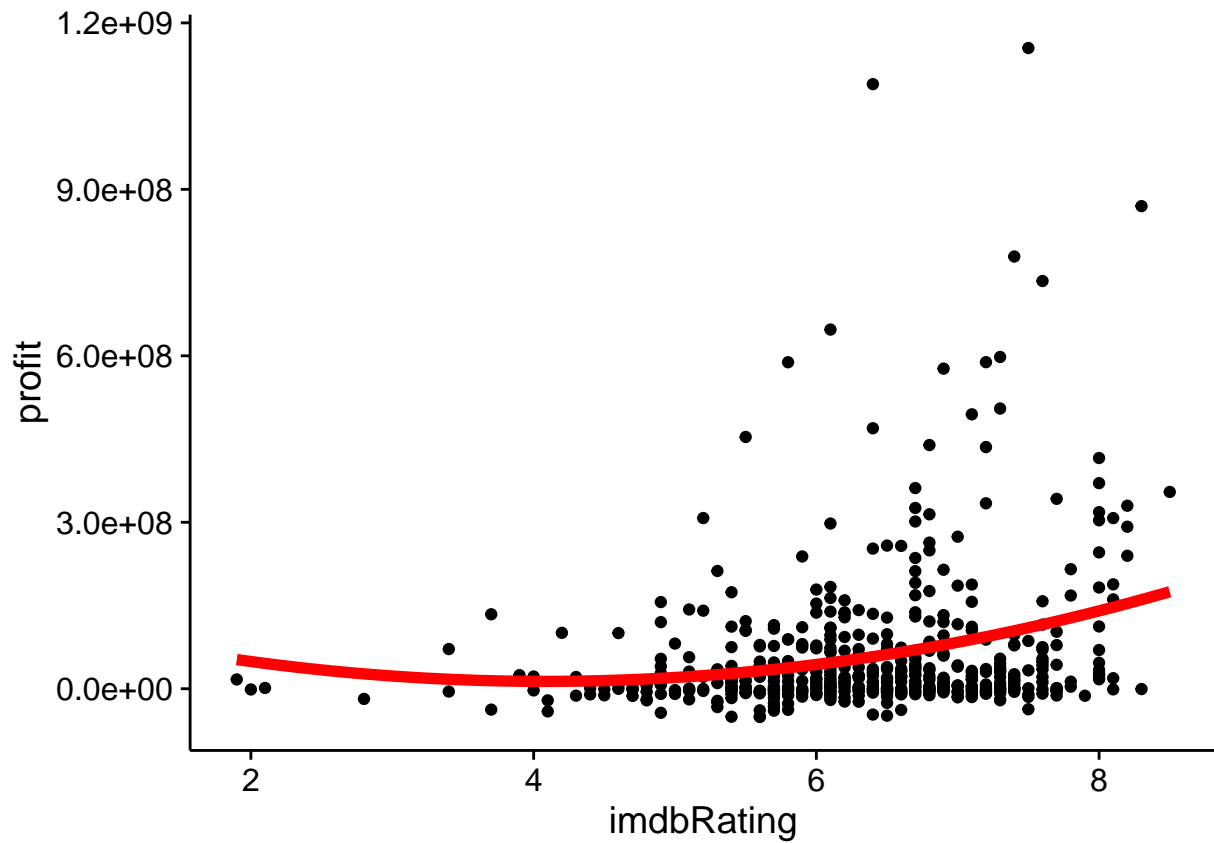
```
M3 = lm(profit~imdbRating+I(imdbRating^2),small_movie)
```

```
cX3 = seq(range(small_movie$imdbRating)[1],
          range(small_movie$imdbRating)[2],length=500)
```

```
theta3 = coef(M3)
```

```
cY3 = theta3[1] + theta3[2]*cX3 + theta3[3]*cX3^2
```

```
print(ggplot(small_movie,aes(imdbRating,profit)) + geom_point() + geom_line(aes(x=cX3,y=cY3), size=2,color="red"))
```



```
summary(M3)$r.squared
```

```
## [1] 0.06216604
```

```
mean(residuals(M3)^2)
```

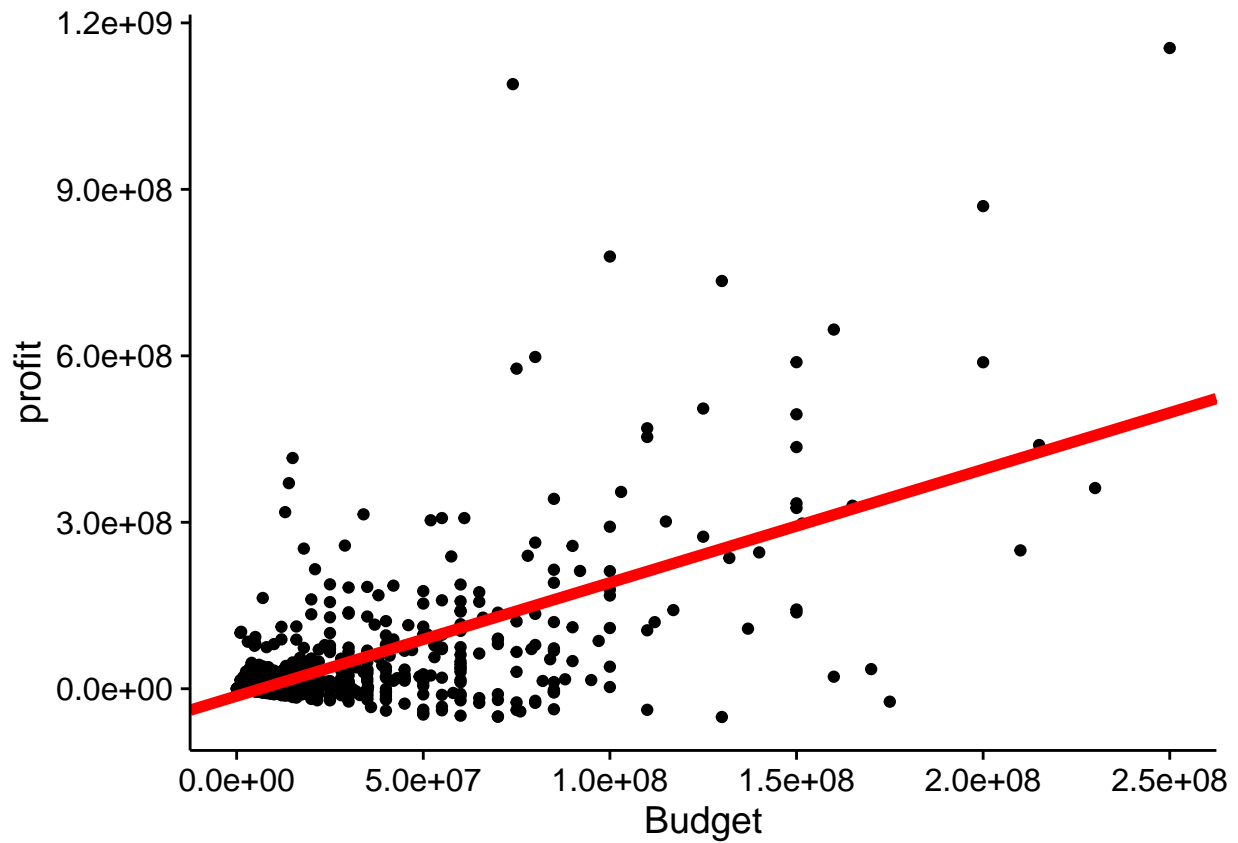
```
## [1] 1.789373e+16
```

```
M4 = lm(profit ~Budget,small_movie)
```

```
theta4 = coef(M4)
```

```
print(ggplot(small_movie,aes(Budget,profit)) + geom_point()
      + geom_abline(intercept = theta4[1], slope = theta4[2], size=2,color="red"))
```





```
summary(M4)$r.squared
```

```
## [1] 0.3652167
```

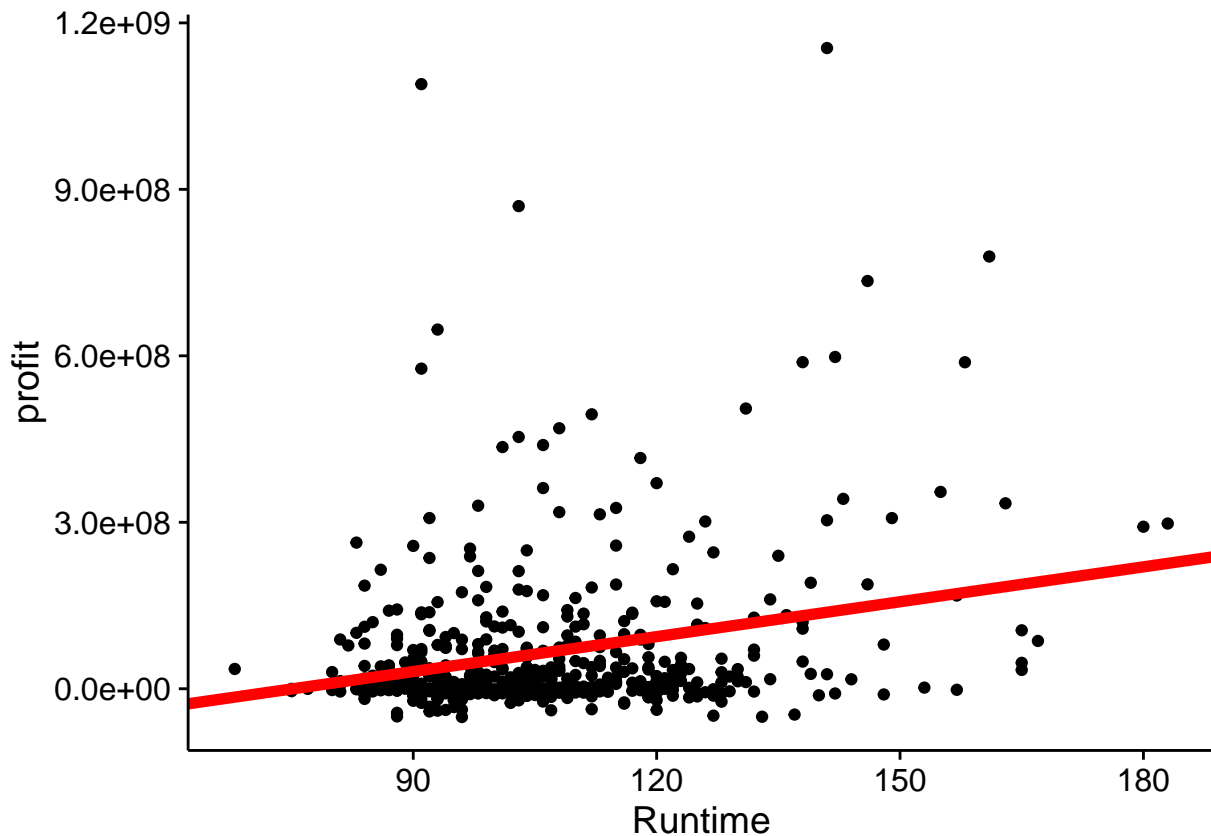
```
mean(residuals(M4)^2)
```

```
## [1] 1.211157e+16
```

```
M5 = lm(profit~Runtime,small_movie)
```

```
theta5 = coef(M5)
```

```
print(ggplot(small_movie,aes(Runtime,profit)) + geom_point() +  
      geom_abline(intercept = theta5[1], slope = theta5[2], size=2,color="red"))
```



```
summary(M5)$r.squared
```

```
## [1] 0.07233629
```

```
mean(residuals(M5)^2)
```

```
## [1] 1.769968e+16
```

```
train_set = seq(5,95, by=5)
```

```
mse.on.test.data_2 = rep(0,19)
```

```
mse.on.train.data_2 = rep(0,19)
```

```
train_mse = rep(0,10)
```

```
test_mse= rep(0,10)
```

```
n_row = nrow(movie_only_data)
```

```
best_mse_on_test = 1.797693e+308
```

```
best_mse_on_train = 1.797693e+308
```

```
best_train_size = 0
```

```
set.seed(50)
```

```
for (num in seq(1:19)){
```

```
  for(i in seq(1:10)){
```

```
    local_var = train_set[num]
```

```
    local_var = local_var/100
```

```
    random_perm = sample(n_row,n_row)
```

```
    first_index = random_perm[1:floor(n_row*local_var)]
```

```

second_index = random_perm[(floor(n_row * local_var)+1):n_row]
train_data = movie_only_data[first_index,]
test_data = movie_only_data[second_index,]
data = train_data
model = lm(profit~Year+Runtime*imdbVotes+Budget+Awards+I(Awards^2) + I(Awards^3)+imdbVotes
           +tomatoReviews*Budget+I(tomatoReviews^2)*Budget+
           tomatoRotten*imdbVotes+tomatoUserMeter*imdbVotes+
           I(log(tomatoUserReviews))*imdbVotes+
           tomatoUserRating*Budget+tomatoMeter*Budget+
           imdbRating*imdbVotes+I(imdbRating^2)*Budget+
           I(imdbRating^3)*Budget+Metascore*imdbVotes,data)
train_mse[i] = mean(residuals(model)^2)
data = test_data
test_mse[i] = calculate_rmse(test_data$profit, predict(model,data))
}
mse.on.train.data_2[num] = mean(train_mse)
mse.on.test.data_2[num] = mean(test_mse)

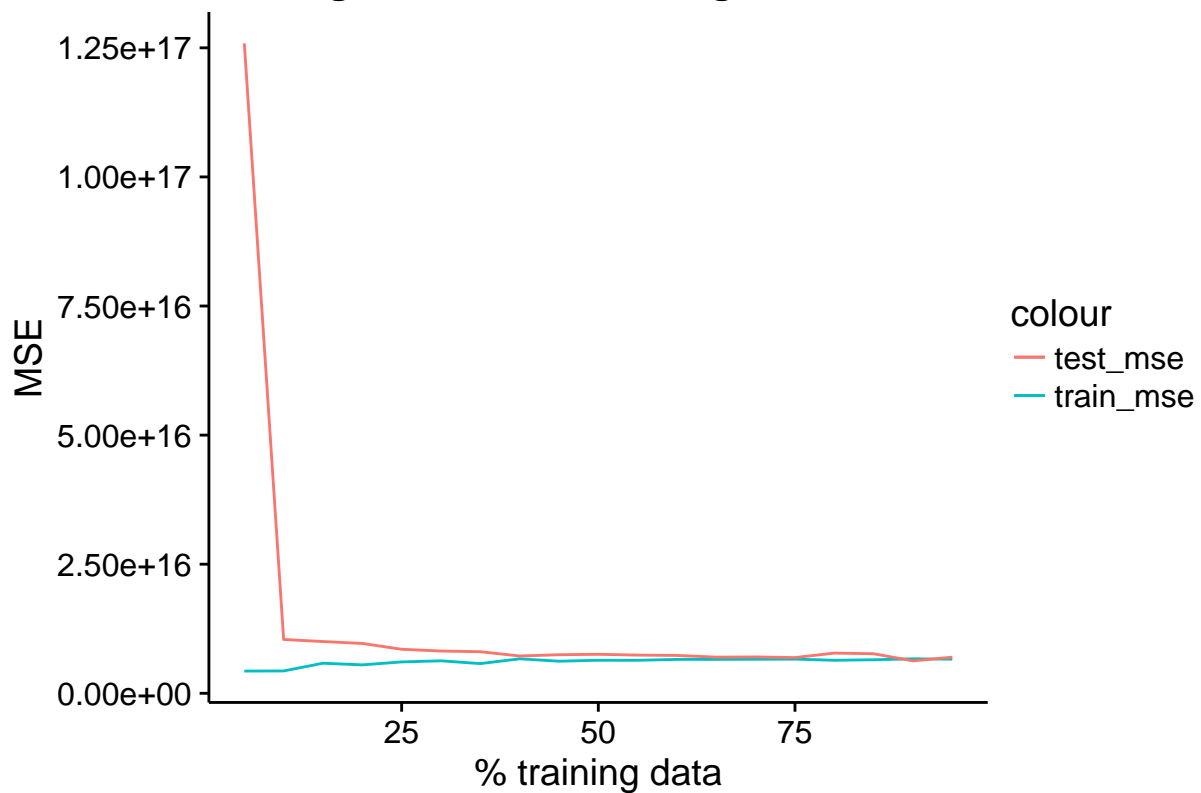
if (mean(test_mse) < best_mse_on_test){
  best_mse_on_test = mean(test_mse)
  best_model=model
}
if(mean(train_mse) < best_mse_on_train){
  best_mse_on_train = mean(train_mse)
}
}

result = data.frame(training_data=train_set,
                    mse_train_data=mse.on.train.data_2,
                    mse_test_data=mse.on.test.data_2)

ggplot(result,aes(x=training_data)) +
  geom_line(aes(y=mse_train_data,color="train_mse")) + geom_line(aes(y=mse_test_data,color="test_mse")) +
  ylab("MSE") +xlab("% training data") +
  ggtitle("Fig 2.0 : MSE vs Training data size")

```

**Fig 2.0 : MSE vs Training data size**



```
cat("best MSE on test data:", best_mse_on_test)
```

```
## best MSE on test data: 6.279968e+15
```

```
cat("\n")
```

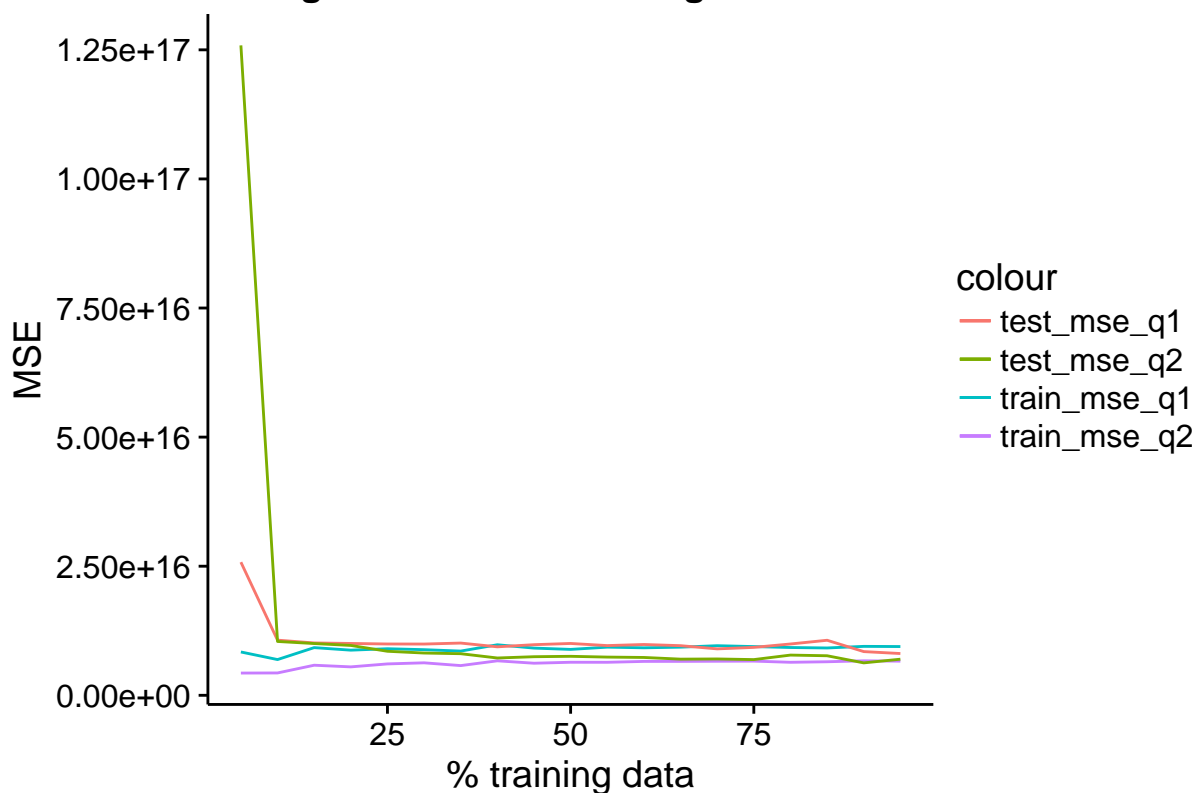
```
cat("best MSE on train data:", best_mse_on_train)
```

```
## best MSE on train data: 4.309105e+15
```

```
result2 = data.frame(training_data=train_set,  
                      mse_train_data_q1=mse.on.train.data_1,  
                      mse_test_data_q1=mse.on.test.data_1,  
                      mse_train_data_q2=mse.on.train.data_2,  
                      mse_test_data_q2=mse.on.test.data_2)
```

```
ggplot(result2,aes(x=training_data)) + geom_line(aes(y=mse_train_data_q1,color="train_mse_q1")) + geom_line(aes(y=mse_test_data_q1,color="test_mse_q1"))
```

**Fig 2.1 : MSE vs Training data size**



Write code that featurizes genre (can use code from Part-I), actors, directors, and other categorical variables. Explain how you encoded the variables into features.

Code for the question were extracted from the code I submitted for project1. For the Genre I transformed the features into an N-dimensional vector of zeros and ones, where one represents the presence of a genre and zero represents the absence of a genre. N is the total number of possible distinct Genre. Genre with high sparsity are dropped. The same techniques used to featurize the genre feature were used for actors, directors, country, production, tomatoImage, language and rated features as shown in the code below.

```
movie_only_data = subset(movie_only_data, Genre != "N/A")
docs = data.frame(movie_only_data$Genre)
ds = DataframeSource(docs)
rm(docs)
docs = VCorpus(ds)
docs = tm_map(docs, content_transformer(tolower))
docs = tm_map(docs, removePunctuation)
docs = tm_map(docs, removeNumbers)
docs = tm_map(docs, stripWhitespace)
dtm = DocumentTermMatrix(docs)
dtm = removeSparseTerms(dtm, 0.925)
newcolumns_1 = data.frame(as.matrix(dtm))
movie_only_data = cbind(movie_only_data, newcolumns_1)
genre_features = names(newcolumns_1)

docs = data.frame(movie_only_data$tomatoImage)
```

```

ds = DataframeSource(docs)
rm(docs)
docs = VCorpus(ds)
docs = tm_map(docs, content_transformer(tolower))
toSpace = content_transformer(function(x,pattern) gsub(pattern," ",x))
docs = tm_map(docs,toSpace, ",")
dtm = DocumentTermMatrix(docs)
newcolumns_4 = data.frame(as.matrix(dtm))
movie_only_data = cbind(movie_only_data,newcolumns_4)
tomato_image_features = names(newcolumns_4)

temp = data.frame(gsub(pattern = " ", replacement = "", x = movie_only_data$Production))
names(temp) = "Production"
temp$Production = sapply(temp$Production, function(x){return(substr(x,1,9))})

docs = data.frame(temp$Production)
ds = DataframeSource(docs)
rm(docs)
docs = VCorpus(ds)
toSpace = content_transformer(function(x,pattern) gsub(pattern," ",x))
docs = tm_map(docs,toSpace, ",")
docs = tm_map(docs,toSpace, "/"")
docs = tm_map(docs, removePunctuation)
docs = tm_map(docs, content_transformer(tolower))
toString = content_transformer(function(x, from ,to) gsub(from , to , x))
docs = tm_map(docs , toString, "sonypicut","sonypictu")

dtm = DocumentTermMatrix(docs)
dtm=removeSparseTerms(dtm,0.95)
newcolumns_5 = data.frame(as.matrix(dtm))
movie_only_data = cbind(movie_only_data,newcolumns_5)
production_features = names(newcolumns_5)

docs = data.frame(movie_only_data$Language)
ds = DataframeSource(docs)
rm(docs)
docs = VCorpus(ds)
docs = tm_map(docs, content_transformer(tolower))
toSpace = content_transformer(function(x,pattern) gsub(pattern," ",x))
docs = tm_map(docs,toSpace, ",")
dtm = DocumentTermMatrix(docs)
dtm = removeSparseTerms(dtm, 0.92)
newcolumns_6 = data.frame(as.matrix(dtm))
movie_only_data = cbind(movie_only_data,newcolumns_6)
language_features = names(newcolumns_6)

mydocs = data.frame(movie_only_data$Country)
ds = DataframeSource(mydocs)

```

```

rm(mydocs)
mydocs = VCorpus(ds)
mydocs = tm_map(mydocs, content_transformer(tolower))
#toSpace = content_transformer(function(x,pattern) gsub(pattern," ",x))
#docs = tm_map(docs,toSpace, ",")
toString = content_transformer(function(x, from ,to) gsub(from , to , x))
mydocs = tm_map(mydocs , toString, "new zealand", "new-zealand")
mydocs = tm_map(mydocs , toString, "burkina faso", "burkina-faso")
mydocs = tm_map(mydocs , toString, "united arab emirates", "united-arab-emirates")
mydocs = tm_map(mydocs , toString, "puerto rico", "puerto-rico")
mydocs = tm_map(mydocs , toString, "costa rica", "costa-rica")
mydocs = tm_map(mydocs , toString, "côte d'ivoire", "côte-d'ivoire")
mydocs = tm_map(mydocs , toString, "dominican republic", "dominican-republic")
mydocs = tm_map(mydocs , toString, "solomon islands", "solomon-islands")
mydocs = tm_map(mydocs , toString, "bosnia and herzegovina", "bosnia-and-herzegovina")
mydocs = tm_map(mydocs , toString, "soviet union", "soviet-union")
mydocs = tm_map(mydocs , toString, "east germany", "east-germany")
mydocs = tm_map(mydocs , toString, "federal republic of yugoslavia", "yugoslavia")
mydocs = tm_map(mydocs , toString, "hong kong", "hong-kong")
mydocs = tm_map(mydocs , toString, "west germany", "west-germany")
mydocs = tm_map(mydocs , toString, "the democratic republic of congo", "congo")
mydocs = tm_map(mydocs , toString, "isle of man", "isle-of-man")
mydocs = tm_map(mydocs , toString, "uk", " united-Kingdom")
mydocs = tm_map(mydocs , toString, "trinidad and tobago", "trinidad-and-tobago")
mydocs = tm_map(mydocs , toString, "turks and caicos islands", "turks-and-caicos-islands")
mydocs = tm_map(mydocs , toString, "serbia and montenegro", "serbia-and-montenegro")
mydocs = tm_map(mydocs , toString, "south africa", "south-africa")
mydocs = tm_map(mydocs , toString, "saudi arabia", "saudi-arabia")
mydocs = tm_map(mydocs , toString, "north korea", "north-korea")
mydocs = tm_map(mydocs , toString, "papua new guinea", "papua-new-guinea")
mydocs = tm_map(mydocs , toString, "czech republic", "czech-republic")
mydocs = tm_map(mydocs , toString, "republic of macedonia", "republic-of-macedonia")
mydocs = tm_map(mydocs, removePunctuation)
dtm = DocumentTermMatrix(mydocs)
dtm = removeSparseTerms(dtm, 0.90)
newcolumns_7 = data.frame(as.matrix(dtm))
movie_only_data = cbind(movie_only_data,newcolumns_7)
country_features = names(newcolumns_7)

temp = data.frame(gsub(pattern = " ", replacement = "", x = movie_only_data$Rated))
names(temp) = "Rated"
temp$Rated = sapply(temp$Rated,function(x){if(x=="R"){return("R-rated")}else if (x=="G"){return("G-Rated")}})

docs = data.frame(temp$Rated)
ds = DataframeSource(docs)
rm(docs)
docs = VCorpus(ds)
docs = tm_map(docs, content_transformer(tolower))
toSpace = content_transformer(function(x,pattern) gsub(pattern," ",x))
docs = tm_map(docs,toSpace, ",")

```

```

dtm = DocumentTermMatrix(docs)
dtm = removeSparseTerms(dtm, 0.92)
newcolumns_8 = data.frame(as.matrix(dtm))
movie_only_data = cbind(movie_only_data,newcolumns_8)
Rated_features = names(newcolumns_8)

hlist =holidayNYSE(1888:2018)
hlist = c(hlist,holidayLONDON(1888:2018))
hlist = c(hlist, holidayTSX(1888:2018))
hlist = c(hlist, holidayZURICH(1888:2018))
temp1 = data.frame(isHoliday(timeDate(movie_only_data$Released), holidays =hlist ))
names(temp1) = "v1"
movie_only_data$Holidays[temp1$v1 == TRUE] = "Yes"
movie_only_data$Holidays[temp1$v1 == FALSE] = "No"

```

Use linear regression to predict profit based on all available non-numeric variables (using the transformations in (3)). Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

The code is shown below. Fig 4.1 shows the test and train MSE , the best MSE on the test data was  $1.657613e+16$  which is lower than the value for question1 and question2 above.

```

features = c(genre_features,tomato_image_features,production_features
            ,language_features,country_features,Rated_features,"Holidays")

mydata = movie_only_data[,c("profit",features)]

train_set = seq(5,95, by=5)

mse.on.test.data = rep(0,19)
mse.on.train.data = rep(0,19)
train_mse = rep(0,10)
test_mse= rep(0,10)
n_row = nrow(movie_only_data)
best_mse_on_test = 1.797693e+308
best_mse_on_train = 1.797693e+308
best_train_size = 0
set.seed(200)

for (num in seq(1:19)){
  for(i in seq(1:10)){
    local_var = train_set[num]
    local_var = local_var/100
    random_perm = sample(n_row,n_row)
    first_index = random_perm[1:floor(n_row*local_var)]
    second_index = random_perm[(floor(n_row * local_var)+1):n_row]
    train_data = mydata[first_index,]
    test_data = mydata[second_index,]

```



```

data = train_data
model = lm(profit~.,data)
train_mse[i] = mean(residuals(model)^2)
data = test_data
test_mse[i] = calculate_rmse(test_data$profit, predict(model,data))
}
mse.on.train.data[num] = mean(train_mse)
mse.on.test.data[num] = mean(test_mse)

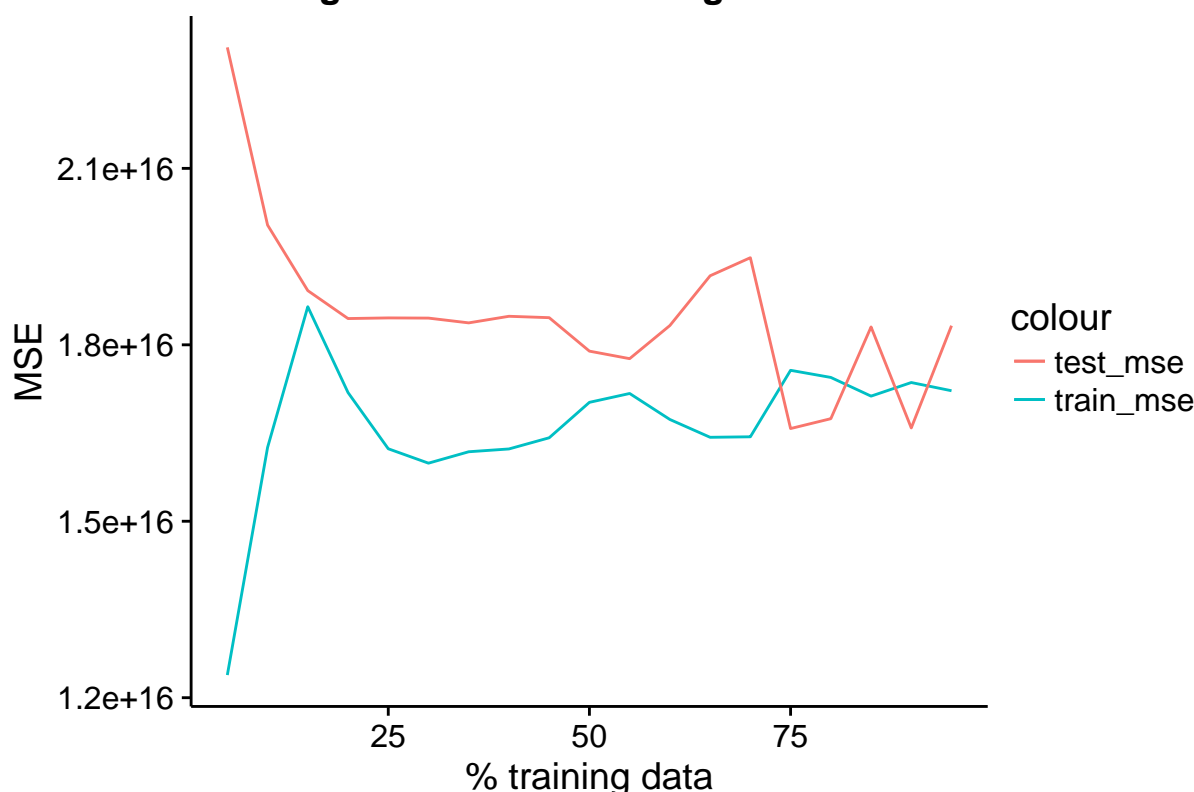
if (mean(test_mse) < best_mse_on_test){
  best_mse_on_test = mean(test_mse)
}
if(mean(train_mse) < best_mse_on_train){
  best_mse_on_train = mean(train_mse)
}
}

result = data.frame(training_data=train_set,
                    mse_train_data=mse.on.train.data,
                    mse_test_data=mse.on.test.data)

ggplot(result,aes(x=training_data)) +
  geom_line(aes(y=mse_train_data,color="train_mse")) + geom_line(aes(y=mse_test_data,color="test_mse")) +
  ylab("MSE") + xlab("% training data") +
  ggtitle("Fig 4.1 : MSE vs Training data size")

```

**Fig 4.1 : MSE vs Training data size**



```
cat("best MSE on test data:", best_mse_on_test)
```

```
## best MSE on test data: 1.657613e+16
```

```
cat("best MSE on train data:", best_mse_on_train)
```

```
## best MSE on train data: 1.2385e+16
```

Try to improve the prediction quality in (1) as much as possible by using both numeric and non-numeric variables as well as creating additional transformed features including interaction features (for example `is_genre_comedy x is_budget_greater_than_3M`). Explain which transformations you used and why you chose them. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

The code for this question is shown below. Fig5.0 shows the test and train MSE, the best MSE was  $5.047046e+15$  which is a significant improvement from question 2. I eliminated removed non-numeric features with very high sparsity. Fig5.1 shows the comparison between Q1, Q2 and Q5 model, overall the Q5 model performed better and has the lowest average MSE on test data.

```
train_set = seq(5,95, by=5)

mse.on.test.data_5 = rep(0,19)
mse.on.train.data_5 = rep(0,19)
train_mse = rep(0,10)
test_mse= rep(0,10)
n_row = nrow(movie_only_data)
best_mse_on_test = 1.797693e+308
```

```

best_mse_on_train = 1.797693e+308
best_train_size = 0
set.seed(201)

for (num in seq(1:19)){
  for(i in seq(1:10)){
    local_var = train_set[num]
    local_var = local_var/100
    random_perm = sample(n_row,n_row)
    first_index = random_perm[1:floor(n_row*local_var)]
    second_index = random_perm[(floor(n_row * local_var)+1):n_row]
    train_data = movie_only_data[first_index,]
    test_data = movie_only_data[second_index,]
    data = train_data
    model = lm(profit~Year+Runtime*imdbVotes+Budget+Awards+I(Awards^2)+I(Awards^3)
               +imdbVotes+tomatoReviews*Budget+
               I(tomatoReviews^2)*Budget+tomatoRotten*imdbVotes+
               tomatoUserMeter*imdbVotes+I(log(tomatoUserReviews))*
               imdbVotes+tomatoUserRating*Budget+tomatoMeter*Budget+
               imdbRating*imdbVotes+I(imdbRating^2)*Budget+
               I(imdbRating^3)*Budget+Metascore*imdbVotes+
               Holidays+action+adventure*imdbVotes+drama+r.rated*imdbVotes+certified+fresh+rotten+pg.13)
    train_mse[i] = mean(residuals(model)^2)
    data = test_data
    test_mse[i] = calculate_rmse(test_data$profit, predict(model,data))
  }
  mse.on.train.data_5[num] = mean(train_mse)
  mse.on.test.data_5[num] = mean(test_mse)

  if (mean(test_mse) < best_mse_on_test){
    best_mse_on_test = mean(test_mse)
    best_model=model
  }

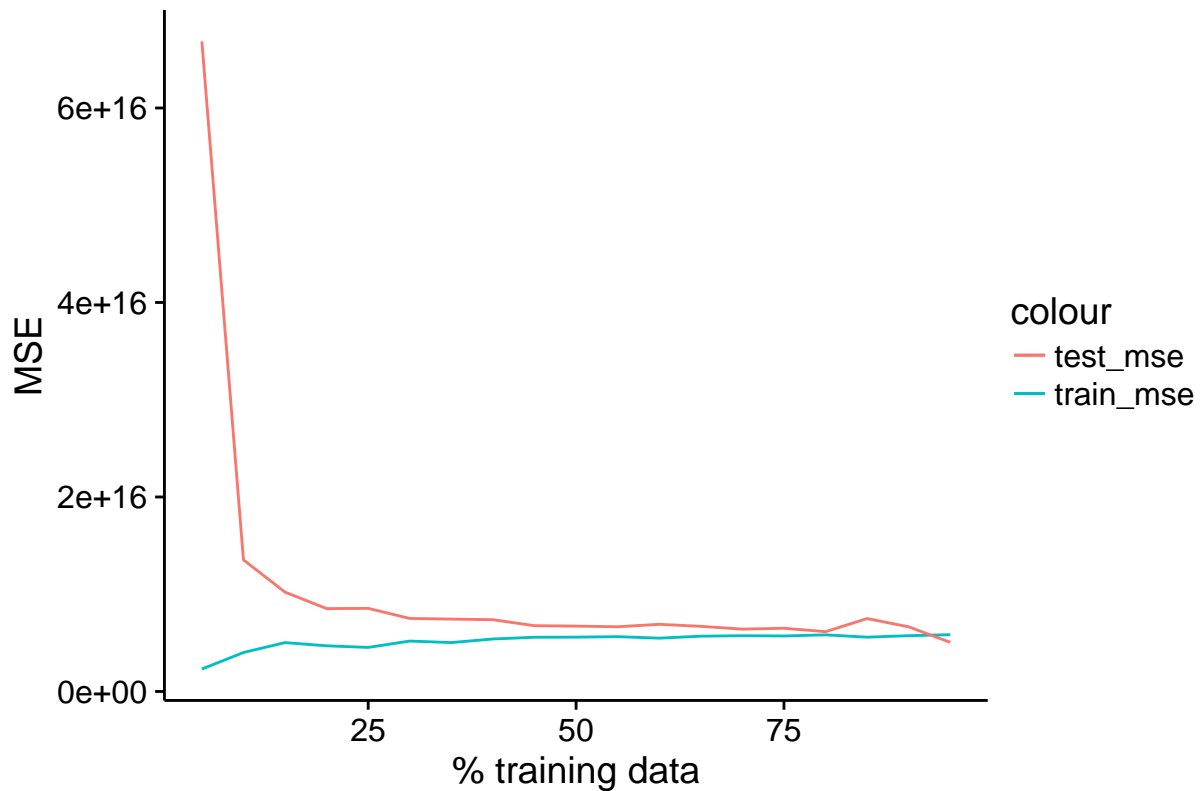
  if(mean(train_mse) < best_mse_on_train){
    best_mse_on_train = mean(train_mse)
  }
}

result = data.frame(training_data=train_set,
                    mse_train_data=mse.on.train.data_5,
                    mse_test_data=mse.on.test.data_5)

ggplot(result,aes(x=training_data)) + geom_line(aes(y=mse_train_data,color="train_mse")) + geom_line(aes(y=mse_test_data,color="test_mse"))

```

**Fig 5.0 : MSE vs Training data size**



```
cat("best MSE on test:", best_mse_on_test)
```

```
## best MSE on test: 5.047046e+15
```

```
cat("best MSE on train:", best_mse_on_train)
```

```
## best MSE on train: 2.303808e+15
```

```
result5 = data.frame(training_data=train_set,
                      mse_train_data_q1=mse.on.train.data_1,
                      mse_test_data_q1=mse.on.test.data_1,
                      mse_train_data_q2=mse.on.train.data_2,
                      mse_test_data_q2=mse.on.test.data_2,
                      mse_train_data_q5=mse.on.train.data_5,
                      mse_test_data_q5=mse.on.test.data_5)
```

```
ggplot(result5,aes(x=training_data)) + geom_line(aes(y=mse_train_data_q1,color="train_mse_q1")) + geom_
  geom_line(aes(y=mse_train_data_q5,color="train_mse_q5")) + geom_line(aes(y=mse_test_data_q5,color="te
```

**Fig 5.1 : MSE vs Training data size**

