# Data

We will use the same dataset as Project 1: `movies_merged`.

# Objective

Your goal in this project is to build a linear regression model that can predict the `Gross` revenue earned by a movie based on other variables. You may use R packages to fit and evaluate a regression model (no need to implement regression yourself). Please stick to linear regression, however.

# Instructions

You should be familiar with using an RMarkdown Notebook by now. Remember that you have to open it in RStudio, and you can run code chunks by pressing *Cmd+Shift+Enter*.

Please complete the tasks below and submit this R Markdown file (as **pr2.Rmd**) containing all completed code chunks and written responses, as well as a PDF export of it (as **pr2.pdf**) which should include all of that plus output, plots and written responses for each task.

*Note that* ***Setup*** *and* ***Data Preprocessing*** *steps do not carry any points, however, they need to be completed as instructed in order to get meaningful results.*

# Setup

Same as Project 1, load the dataset into memory:

```
load('movies_merged')
```

This creates an object of the same name (`movies_merged`). For convenience, you can copy it to `df` and start using it:

```
df = movies_merged
cat("Dataset has", dim(df)[1], "rows and", dim(df)[2], "columns", end="\n", file="")
```

```
## Dataset has 40789 rows and 39 columns
```

```
colnames(df)
```

```
##  [1] "Title"             "Year"              "Rated"
##  [4] "Released"          "Runtime"           "Genre"
##  [7] "Director"          "Writer"            "Actors"
## [10] "Plot"              "Language"          "Country"
## [13] "Awards"            "Poster"            "Metascore"
## [16] "imdbRating"        "imdbVotes"         "imdbID"
## [19] "Type"              "tomatoMeter"       "tomatoImage"
## [22] "tomatoRating"      "tomatoReviews"     "tomatoFresh"
## [25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"
## [28] "tomatoUserRating"  "tomatoUserReviews" "tomatoURL"
## [31] "DVD"               "BoxOffice"         "Production"
## [34] "Website"           "Response"          "Budget"
## [37] "Domestic_Gross"    "Gross"             "Date"
```

### Load R packages

Load any R packages that you will need to use. You can come back to this chunk, edit it and re-run to load any additional packages later.

```
library(ggplot2)
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-5
```

If you are using any non-standard packages (ones that have not been discussed in class or explicitly allowed for this project), please mention them below. Include any special instructions if they cannot be installed using the regular `install.packages('<pkg name>')` command.

**Non-standard packages used**: None

# Data Preprocessing

Before we start building models, we should clean up the dataset and perform any preprocessing steps that may be necessary. Some of these steps can be copied in from your Project 1 solution. It may be helpful to print the dimensions of the resulting dataframe at each step.

### 1. Remove non-movie rows

```
# TODO: Remove all rows from df that do not correspond to movies
#let's see if any NA in df Type column and what types in the column.
sum(is.na(df$Type))
```

```
## [1] 0
```

```
table(df$Type)
```

```
##
##   game  movie series
##     64  40000    725
```

```
#There is no NA, and in total 40000 rows of movies. Let's remove all non-movie rows from df.
df=df[df$Type=='movie',]
cat("After removal, dataset has", dim(df)[1], "rows", end="\n", file="")
```

```
## After removal, dataset has 40000 rows
```

### 2. Drop rows with missing `Gross` value

Since our goal is to model `Gross` revenue against other variables, rows that have missing `Gross` values are not useful to us.

```
# TODO: Remove rows with missing Gross value
#check how many rows with NA for Gross column
sum(is.na(df$Gross))
```

```
## [1] 35442
```

```r
#remove all rows with NA in Gross
df=df[!is.na(df$Gross),]
#confirm the remaining rows
dim(df)
```

```
## [1] 4558   39
```

### 3. Exclude movies released prior to 2000

Inflation and other global financial factors may affect the revenue earned by movies during certain periods of time. Taking that into account is out of scope for this project, so let's exclude all movies that were released prior to the year 2000 (you may use `Released`, `Date` or `Year` for this purpose).

```r
# TODO: Exclude movies released prior to 2000
#check how many rows with Year 2000 or later
sum(df$Year>=2000)
```

```
## [1] 3332
```

```r
#exclude movies prior to 2000
df=df[df$Year>=2000,]
#confirm the remaining rows
dim(df)
```

```
## [1] 3332   39
```

### 4. Eliminate mismatched rows

*Note: You may compare the `Released` column (string representation of release date) with either `Year` or `Date` (numeric representation of the year) to find mismatches. The goal is to avoid removing more than 10% of the rows.*

```r
# TODO: Remove mismatched rows
#extract year information from Released variable
RLyear=as.numeric(format(df$Released,'%Y'))
#remove rows that df$Year and df$Released have more than 1 year difference
df=df[abs(df$Year-RLyear)<=1,]
#confirm more than 90% rows left after removing mismatches.
dim(df)
```

```
## [1] 3257   39
```

### 5. Drop `Domestic_Gross` column

`Domestic_Gross` is basically the amount of revenue a movie earned within the US. Understandably, it is very highly correlated with `Gross` and is in fact equal to it for movies that were not released globally. Hence, it should be removed for modeling purposes.

```r
# TODO: Exclude the `Domestic_Gross` column
df$Domestic_Gross=NULL
```

## 6. Process `Runtime` column

```
# TODO: Replace df$Runtime with a numeric column containing the runtime in minutes
# first let's see how many time formats in the Runtime column
runtime=df$Runtime
sum(is.na(runtime))
```

```
## [1] 41
```

```
rtFormat=as.factor(gsub("[0-9]+ *","",runtime))
summary(rtFormat)
```

```
##  min  N/A NA's
## 3192   24   41
```

```
#make Runtime to numeric value in mins
df$Runtime=sapply(runtime,function(str) as.numeric(gsub("min", "", str)))
```

Perform any additional preprocessing steps that you find necessary, such as dealing with missing values or highly correlated columns (feel free to add more code chunks, markdown blocks and plots here as necessary).

```
# TODO(optional): Additional preprocessing
df=na.omit(df)
#remove Date since it is highly related to Year
df$Date=NULL
#remove three tomato review related columns since they are highly related to some of #the other tomato
df$tomatoUserMeter=NULL
df$tomatoMeter=NULL
df$tomatoReviews=NULL
```

***Note**: Do NOT convert categorical variables (like `Genre`) into binary columns yet. You will do that later as part of a model improvement task.*

## Final preprocessed dataset

Report the dimensions of the preprocessed dataset you will be using for modeling and evaluation, and print all the final column names. (Again, `Domestic_Gross` should not be in this list!)

```
# TODO: Print the dimensions of the final preprocessed dataset and column names
print(paste("df has",dim(df)[1],"rows and",dim(df)[2],"columns"))
```

```
## [1] "df has 2799 rows and 34 columns"
```

```
colnames(df)
```

```
##  [1] "Title"             "Year"              "Rated"
##  [4] "Released"          "Runtime"           "Genre"
##  [7] "Director"          "Writer"            "Actors"
## [10] "Plot"              "Language"          "Country"
## [13] "Awards"            "Poster"            "Metascore"
## [16] "imdbRating"        "imdbVotes"         "imdbID"
## [19] "Type"              "tomatoImage"       "tomatoRating"
## [22] "tomatoFresh"       "tomatoRotten"      "tomatoConsensus"
## [25] "tomatoUserRating"  "tomatoUserReviews" "tomatoURL"
## [28] "DVD"               "BoxOffice"         "Production"
## [31] "Website"           "Response"          "Budget"
## [34] "Gross"
```

## Evaluation Strategy

In each of the tasks described in the next section, you will build a regression model. In order to compare their performance, use the following evaluation procedure every time:

1. Randomly divide the rows into two sets of sizes 5% and 95%.
2. Use the first set for training and the second for testing.
3. Compute the Root Mean Squared Error (RMSE) on the train and test sets.
4. Repeat the above data partition and model training and evaluation 10 times and average the RMSE results so the results stabilize.
5. Repeat the above steps for different proportions of train and test sizes: 10%-90%, 15%-85%, ..., 95%-5% (total 19 splits including the initial 5%-95%).
6. Generate a graph of the averaged train and test RMSE as a function of the train set size (%).

You can define a helper function that applies this procedure to a given model and reuse it.

## Tasks

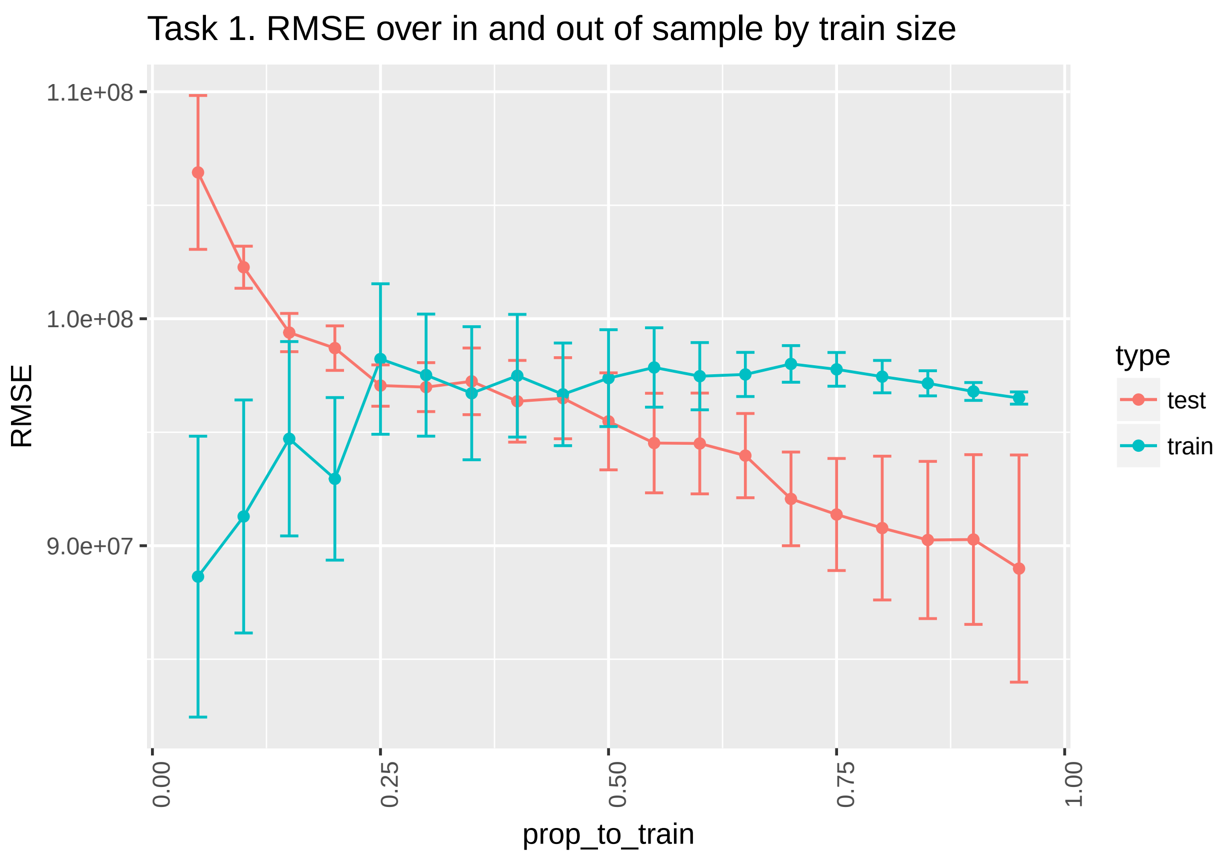
Each of the following tasks is worth 20 points. Remember to build each model as specified, evaluate it using the strategy outlined above, and plot the training and test errors by training set size (%).

### 1. Numeric variables

Use linear regression to predict `Gross` based on all available *numeric* variables.

```
# TODO: Build & evaluate model 1 (numeric variables only)
dfn=df[,c("Gross","Budget","tomatoUserReviews","tomatoUserRating","tomatoRotten","tomatoRating","tomato
N=dim(dfn)[1]
#helper function to iterate  10 times for computing RMSEs given a train size.
get_rmse_list=function(df,prop_to_train,N){
  #lists to record computations
  rmse_list=c()
  prop_to_train_list=c()
  type=c()
  #set random seeds for good comparisons between tasks.
  set.seed(200)
  seeds=sample(10000,10)
  #ten iterations
  for( seed in seeds){
    set.seed(seed)
    #train test partitions
    to_train=sample(N,prop_to_train*N)
    train=df[to_train,]
    test=df[-to_train,]
    #build models using train data
    model=lm(Gross~.,data = train)
    #make prediction  on train data by model
    pred_trainGross=predict(model,train[,-1])
    #compute train error
    rmse_train=sqrt(mean((pred_trainGross-train[,1])^2))
    #make prediction on test data by model
    pred_testGross=predict(model,test[,-1])
    #compute test error
```

```
    rmse_test=sqrt(mean((pred_testGross-test[,1])^2))
    #record the results
    rmse_list=c(rmse_list,rmse_train,rmse_test)
    prop_to_train_list=c(prop_to_train_list,prop_to_train,prop_to_train)
    type=c(type,"train","test")
  }
  data.frame(prop_to_train=prop_to_train_list,type=type,rmse=rmse_list)
}

#start computation from train size 5%
rmse_df=get_rmse_list(dfn,0.05,N)
#iteration from train size 10% to 95%
for (prop_to_train in (2:19)*0.05){
  rmse_df=rbind(rmse_df,get_rmse_list(dfn,prop_to_train,N))
}
#summary rmse to get mean and se
average_rmse=aggregate(rmse_df["rmse"],by=rmse_df[c("type","prop_to_train")],function(X) c(mean=mean(X)
average_rmse=do.call(data.frame,average_rmse)
names(average_rmse)[3:5]=c("rmse","sd","N")
average_rmse$se=average_rmse$sd/sqrt(average_rmse$N)

#plot mean and se of RMSEs for each train size
ggplot(average_rmse,aes(prop_to_train,rmse,group=type,col=type))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin=rmse-se,ymax=rmse+se),width=0.02)+
  ylab("RMSE")+
  theme(axis.text.x = element_text(angle=90, hjust=1))+
  ggtitle("Task 1. RMSE over in and out of sample by train size")
```




**Q**: List all the numeric variables you used.

**A**:

Here are the numeric variables I used in this question.

"Budget","tomatoUserReviews","tomatoUserRating","tomatoRotten", "tomatoRating","tomatoFresh","imdbVotes", "imdbRating", "Runtime","Year".

And use them to make linear regression model for Gross.

So in summary, after task 1, the average train errors are around 96 M, and the average test errors are around 90 M.

```
#print average train rmse
average_rmse$rmse[average_rmse$type=="train"]
```

```
##  [1] 88638211 91288414 94712519 92947185 98225682 97516399 96717979
##  [8] 97489405 96669358 97383489 97853469 97468449 97545471 98009494
## [15] 97770468 97449199 97155643 96795773 96506757
```

```
#print average test rmse
average_rmse$rmse[average_rmse$type=="test"]
```

```
##  [1] 106445643 102269697  99390782  98704912  97056131  96987009  97242389
##  [8]  96363702  96498561  95479511  94523617  94504638  93969973  92063115
## [15]  91375740  90778069  90252732  90273297  88993547
```

## 2. Feature transformations

Try to improve the prediction quality from **Task 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`).

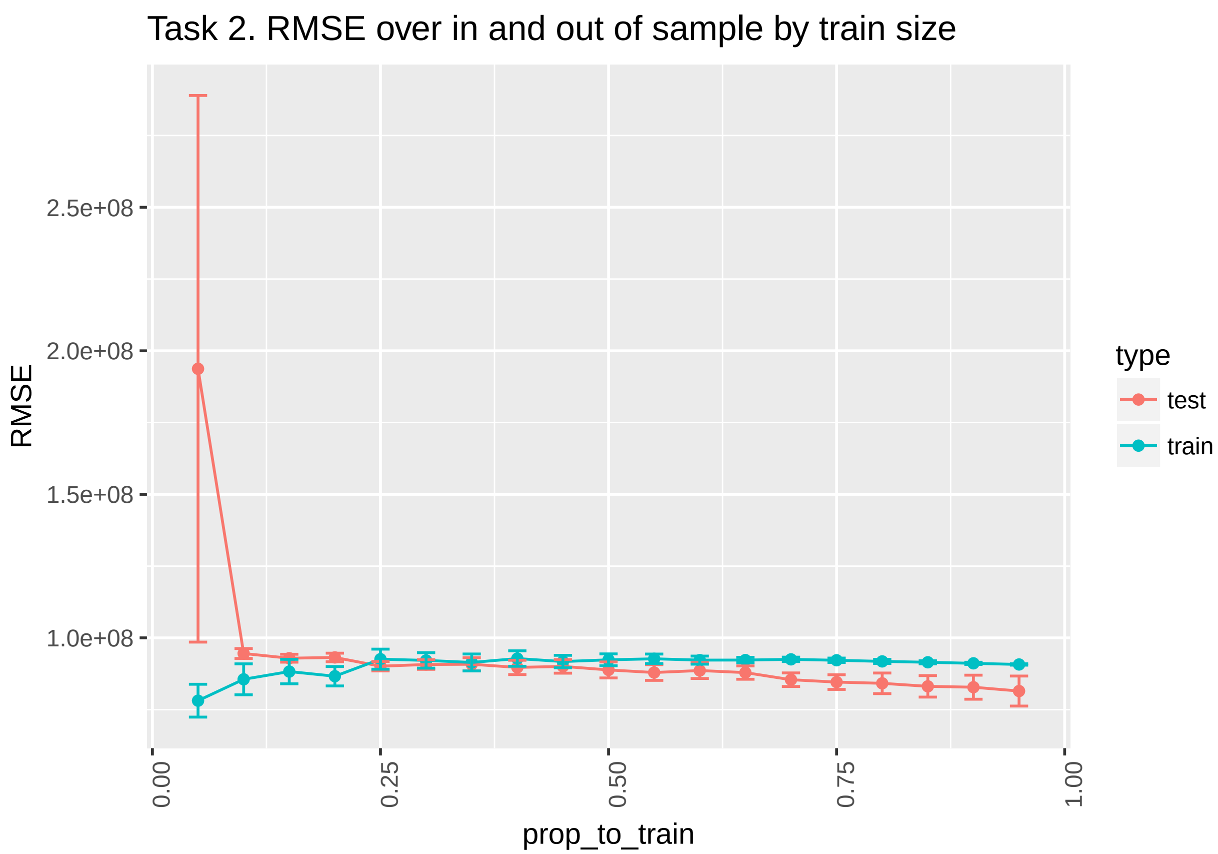
```
#sqrt transformation five numeric variables including Gross
dfn2=sqrt(dfn[c("Gross","Budget","tomatoRotten","tomatoFresh","imdbVotes")])

#binning Runtime
qruntime=q=c(-Inf,quantile(dfn$Runtime,probs = seq(0.1,.9,.1),na.rm = T),Inf)
runtimelist=paste0("Runtime",1:10)
dfn2$Runtime=cut(dfn$Runtime,qruntime,labels =runtimelist )
for(runtime in runtimelist){
  runtimeCol=data.frame(as.numeric(grepl(runtime,dfn2$Runtime)))
  names(runtimeCol)=runtime
  dfn2=cbind(dfn2,runtimeCol)
}


#binning imdbrate
qimdbrate=q=c(-Inf,quantile(dfn$imdbRating,probs = seq(0.1,.9,.1),na.rm = T),Inf)
imdbratelist=paste0("imdbRating",1:10)
dfn2$imdbRating=cut(dfn$imdbRating,qimdbrate,labels =imdbratelist )
for(imdbrate in imdbratelist){
  imdbrateCol=data.frame(as.numeric(grepl(imdbrate,dfn2$imdbRating)))
  names(imdbrateCol)=imdbrate
  dfn2=cbind(dfn2,imdbrateCol)
}


#add the other four numerical variables

dfn2=cbind(dfn2,df[c("tomatoRating","tomatoUserRating","tomatoUserReviews","Year")])
dfn2o=dfn2
#remove factor columns Runtime and imdbRating, and only remain binary columns
dfn2$Runtime=NULL
dfn2$imdbRating=NULL

#let's see if need remove any column
summary(lm(Gross~.,dfn2))
```

```
##
## Call:
## lm(formula = Gross ~ ., data = dfn2)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -11728  -1819   -214   1536  19203
##
## Coefficients: (2 not defined because of singularities)
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -3.026e+04  2.582e+04  -1.172  0.24135
## Budget             1.094e+00  2.552e-02  42.858  < 2e-16 ***
## tomatoRotten      -4.099e+01  4.139e+01  -0.990  0.32206
```

```
## tomatoFresh        2.812e+02  4.807e+01   5.849 5.53e-09 ***
## imdbVotes          1.614e+01  5.977e-01  26.998  < 2e-16 ***
## Runtime1           1.782e+03  2.515e+02   7.085 1.76e-12 ***
## Runtime2           1.469e+03  2.575e+02   5.707 1.27e-08 ***
## Runtime3           8.222e+02  2.519e+02   3.264  0.00111 **
## Runtime4           8.513e+02  2.612e+02   3.260  0.00113 **
## Runtime5           5.518e+02  2.487e+02   2.219  0.02655 *
## Runtime6           2.255e+02  2.494e+02   0.904  0.36594
## Runtime7           5.476e+02  2.525e+02   2.169  0.03017 *
## Runtime8          -1.132e+02  2.459e+02  -0.460  0.64546
## Runtime9                  NA         NA      NA       NA
## Runtime10         -1.537e+03  2.705e+02  -5.683 1.46e-08 ***
## imdbRating1        4.314e+03  3.845e+02  11.219  < 2e-16 ***
## imdbRating2        3.287e+03  3.374e+02   9.740  < 2e-16 ***
## imdbRating3        3.037e+03  3.257e+02   9.325  < 2e-16 ***
## imdbRating4        2.358e+03  2.966e+02   7.948 2.73e-15 ***
## imdbRating5        1.666e+03  2.687e+02   6.199 6.52e-10 ***
## imdbRating6        1.185e+03  2.719e+02   4.359 1.35e-05 ***
## imdbRating7        4.766e+02  2.501e+02   1.906  0.05677 .
## imdbRating8        1.474e+02  2.635e+02   0.559  0.57595
## imdbRating9                NA         NA      NA       NA
## imdbRating10      -6.432e+03  4.570e+02 -14.077  < 2e-16 ***
## tomatoRating      -3.421e+02  1.424e+02  -2.402  0.01638 *
## tomatoUserRating   3.953e+03  2.124e+02  18.611  < 2e-16 ***
## tomatoUserReviews  1.360e-04  1.650e-05   8.242 2.58e-16 ***
## Year               6.406e+00  1.293e+01   0.496  0.62024
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2884 on 2772 degrees of freedom
## Multiple R-squared:  0.8168, Adjusted R-squared:  0.815
## F-statistic: 475.2 on 26 and 2772 DF,  p-value: < 2.2e-16
```

```r
#since one out of all Runtime classes can be calculated by another 9
#same thing applies to imdbRating columns
#we have to remove Runtime9 and imdbRating9 columns
dfn2[,c(14,24)]=NULL

N=dim(dfn2)[1]
#helper function to iterate  10 times for computing RMSEs given a train size.
get_rmse_list2=function(df,prop_to_train,N){
  set.seed(200)
  seeds=sample(10000,10)
  rmse_list=c()
  prop_to_train_list=c()
  type=c()
  for( seed in seeds){
    set.seed(seed)
    #train and test partition
    to_train=sample(N,prop_to_train*N)
    train=df[to_train,]
    test=df[-to_train,]
    #build model
    model=lm(Gross~.,data = train)
```

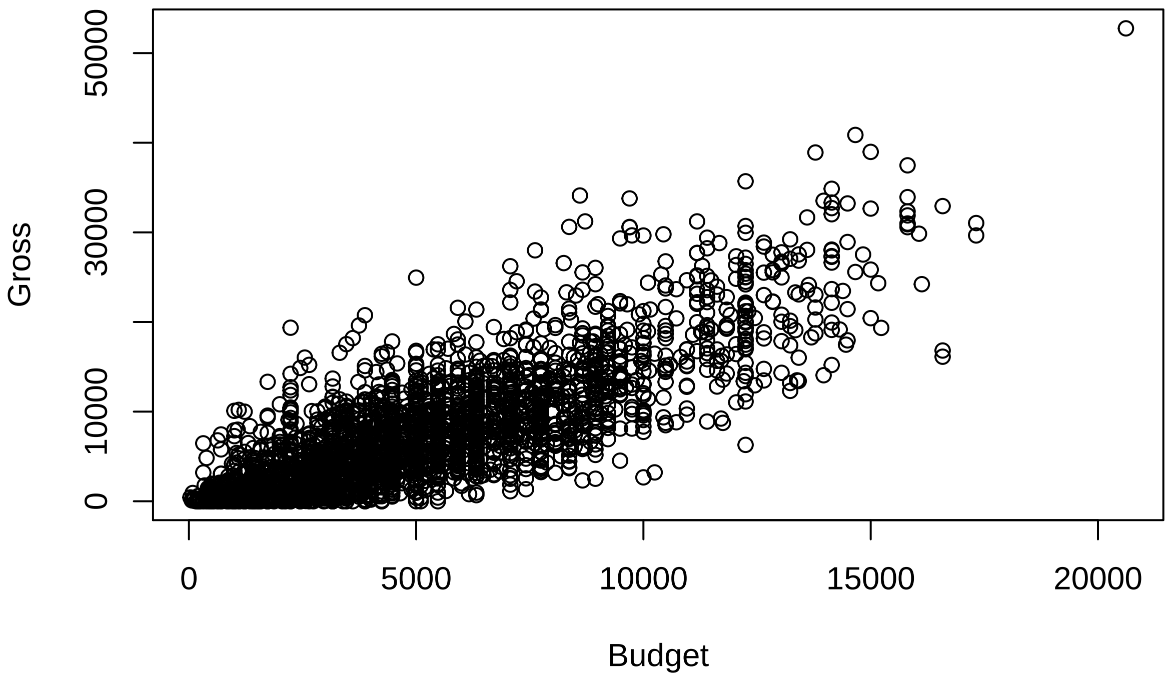
```
    #make predictions and compute RMSEs
    pred_trainGross=(predict(model,train[,-1]))^2
    rmse_train=sqrt(mean((pred_trainGross-(train[,1])^2)^2))

    pred_testGross=(predict(model,test[,-1]))^2
    rmse_test=sqrt(mean((pred_testGross-(test[,1])^2)^2))

    #record results
    rmse_list=c(rmse_list,rmse_train,rmse_test)
    prop_to_train_list=c(prop_to_train_list,prop_to_train,prop_to_train)
    type=c(type,"train","test")
  }
  data.frame(prop_to_train=prop_to_train_list,type=type,rmse=rmse_list)
}
#iterate train size to build models and compute RMSEs
rmse_df2=get_rmse_list2(dfn2,0.05,N)
for (prop_to_train in (2:19)*0.05){
  rmse_df2=rbind(rmse_df2,get_rmse_list2(dfn2,prop_to_train,N))
}
#summary RMSEs to get mean and se
average_rmse2=aggregate(rmse_df2["rmse"],by=rmse_df2[c("type","prop_to_train")],function(X) c(mean=mean
average_rmse2=do.call(data.frame,average_rmse2)
names(average_rmse2)[3:5]=c("rmse","sd","N")
average_rmse2$se=average_rmse2$sd/sqrt(average_rmse2$N)

#plot the results from above summary
ggplot(average_rmse2,aes(prop_to_train,rmse,group=type,col=type))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin=rmse-se,ymax=rmse+se),width=0.02)+
  ylab("RMSE")+
  theme(axis.text.x = element_text(angle=90, hjust=1))+
  ggtitle("Task 2. RMSE over in and out of sample by train size")
```



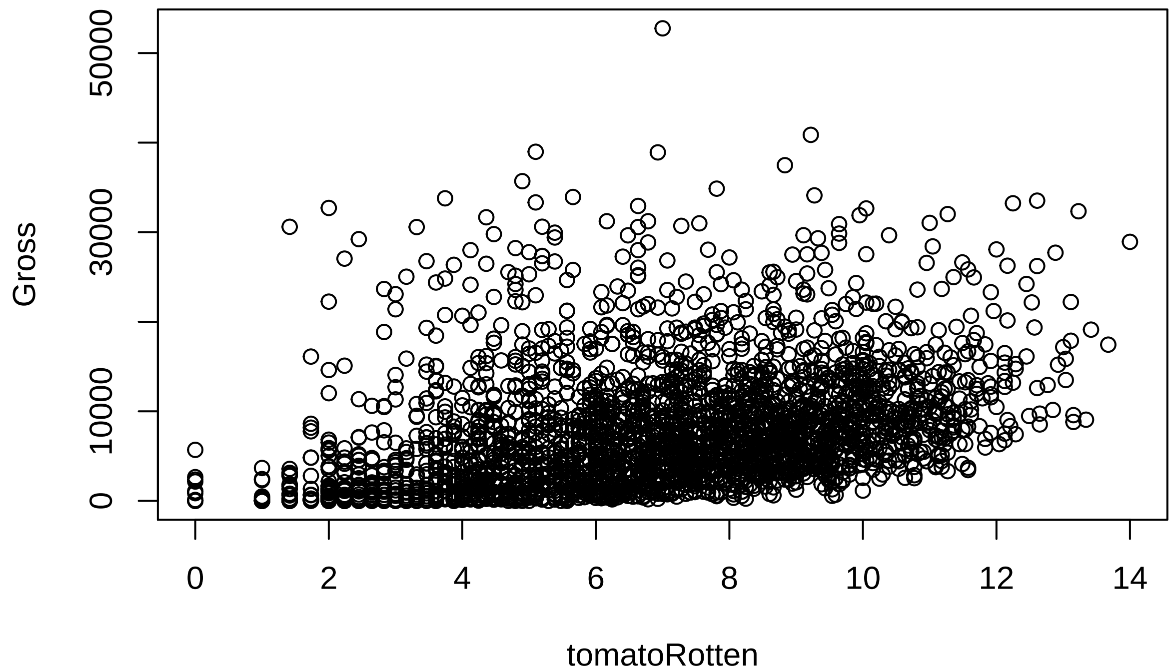

**Q**: Explain which transformations you used and why you chose them.

**A**: In this section, I did sqrt transformation on "Gross","Budget","tomatoRotten","tomatoFresh","imdbVotes". Since after sqrt transformation, the relationship between Gross and the others is much closer to linear. See plots below:

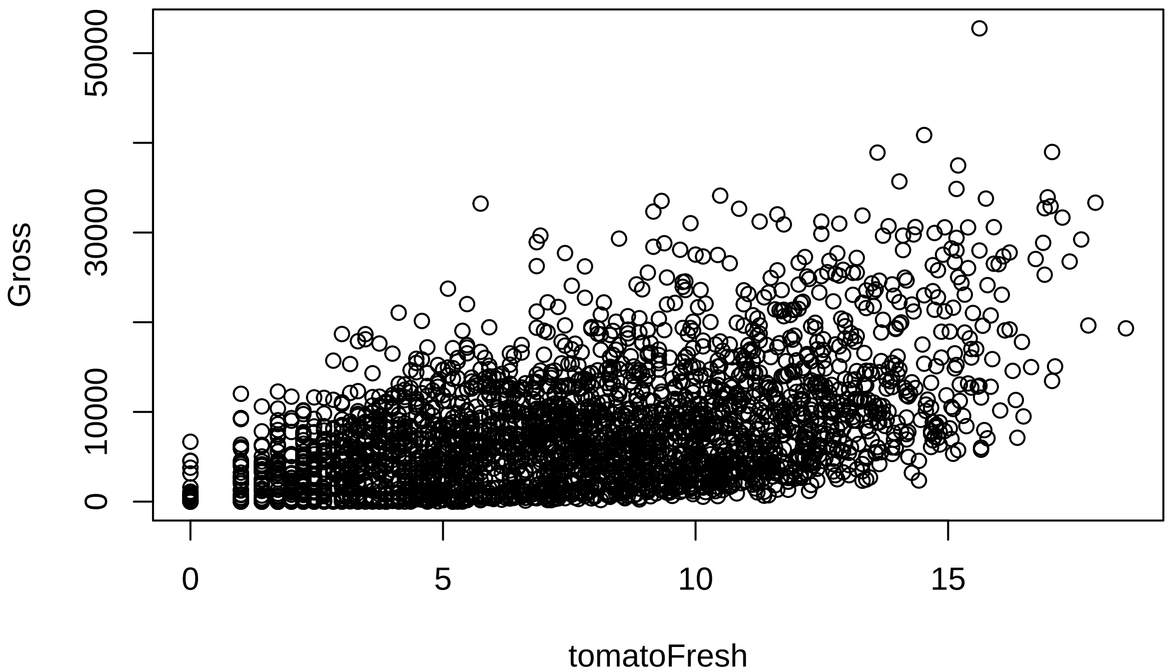
```
plot(Gross~Budget,dfn2)
```



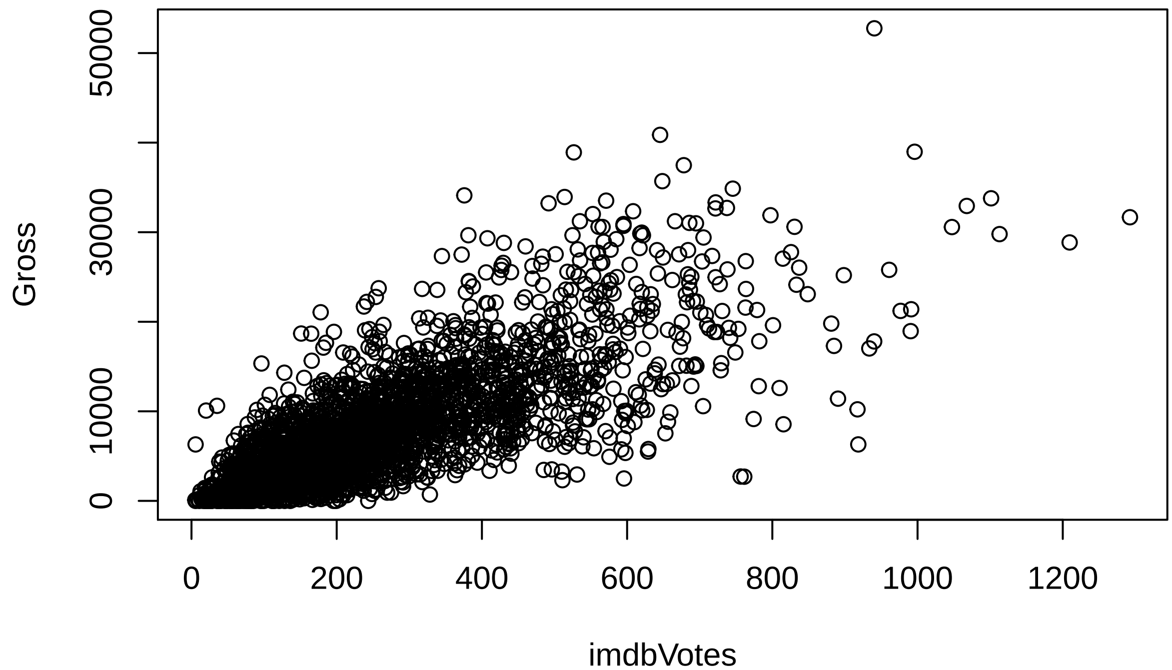

```r
plot(Gross~tomatoRotten,dfn2)
```




```
plot(Gross~tomatoFresh,dfn2)
```




```
plot(Gross~imdbVotes,dfn2)
```




I also binned imdbRating and Runtime to 10 classes for each by combining quantile and cut functions. The reasons for binning is that before binning, we didn't see any strong linear relationships, even after some regular transforms like log or sqrt. After binning, we do see differences of Grosses in different bins. See plots below.

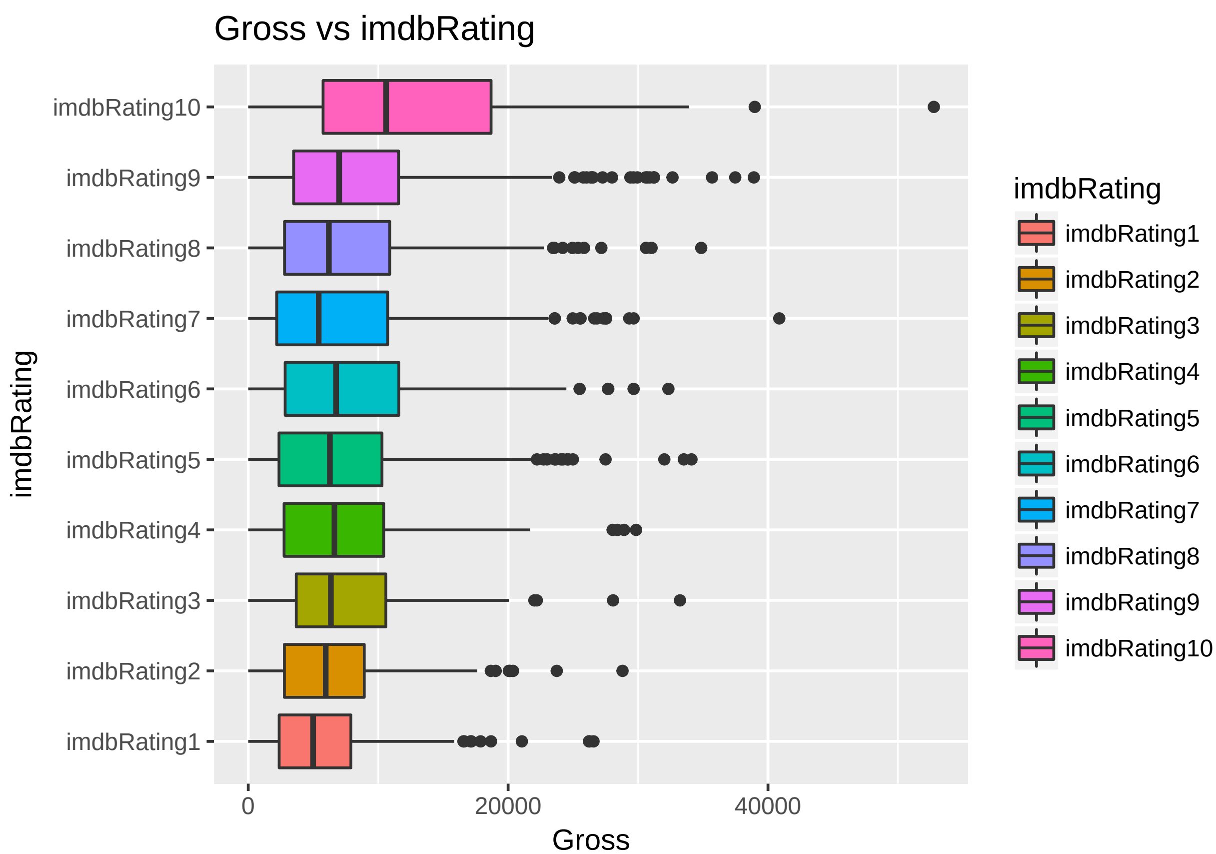
```
#summary dfn2$imdbRating
table(dfn2o$imdbRating)
```

```
##
##  imdbRating1  imdbRating2  imdbRating3  imdbRating4  imdbRating5
##          318          306          246          282          353
##  imdbRating6  imdbRating7  imdbRating8  imdbRating9 imdbRating10
##          247          297          212          300          238
```

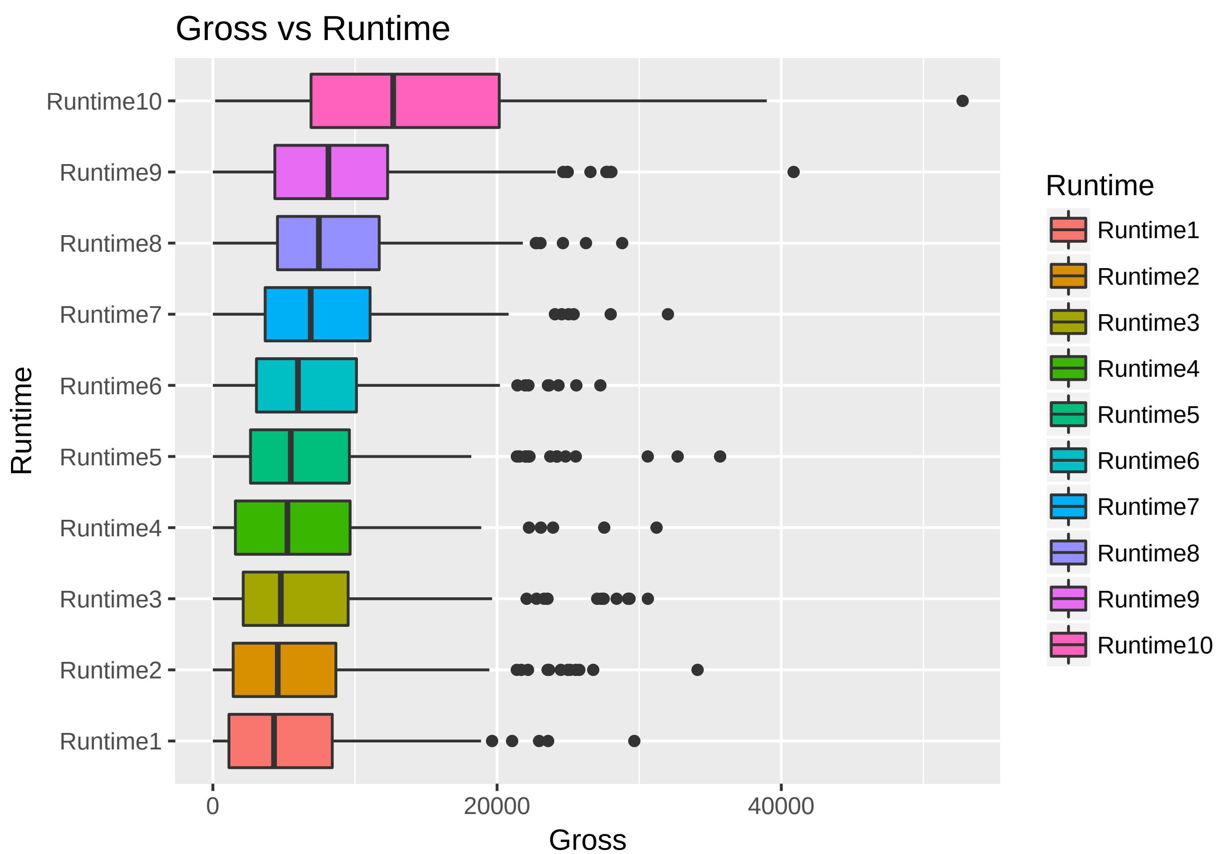
```
#summary dfn2$Runtime
table(dfn2o$Runtime)
```

```
##
##  Runtime1  Runtime2  Runtime3  Runtime4  Runtime5  Runtime6  Runtime7
##       335       268       292       244       286       277       263
##  Runtime8  Runtime9 Runtime10
##       282       278       274
```

```
#plot Gross~imdbRating after binning
ggplot(dfn2o,aes(imdbRating,Gross,fill=imdbRating))+
  geom_boxplot()+coord_flip()+
  ggtitle("Gross vs imdbRating")
```




```r
#plot Gross~Runtime after binning
ggplot(dfn2o,aes(Runtime,Gross,fill=Runtime))+
  geom_boxplot()+coord_flip()+
  ggtitle("Gross vs Runtime")
```




In task 1, the average train errors are around 96 M, and the average test errors are around 89 M. After transformations in task 2, the average train errors dropped to around 91 M, and the average test errors dropped to around 82 M.

```
#print average train rmse
average_rmse2$rmse[average_rmse2$type=="train"]
```

```
##  [1] 78116980 85561091 88260539 86635129 92577031 92146778 91427391
##  [8] 92790757 91750701 92354845 92705877 92245305 92266349 92512511
## [15] 92196328 91799925 91493947 91124395 90732864
```

```
#print average test rmse
average_rmse2$rmse[average_rmse2$type=="test"]
```

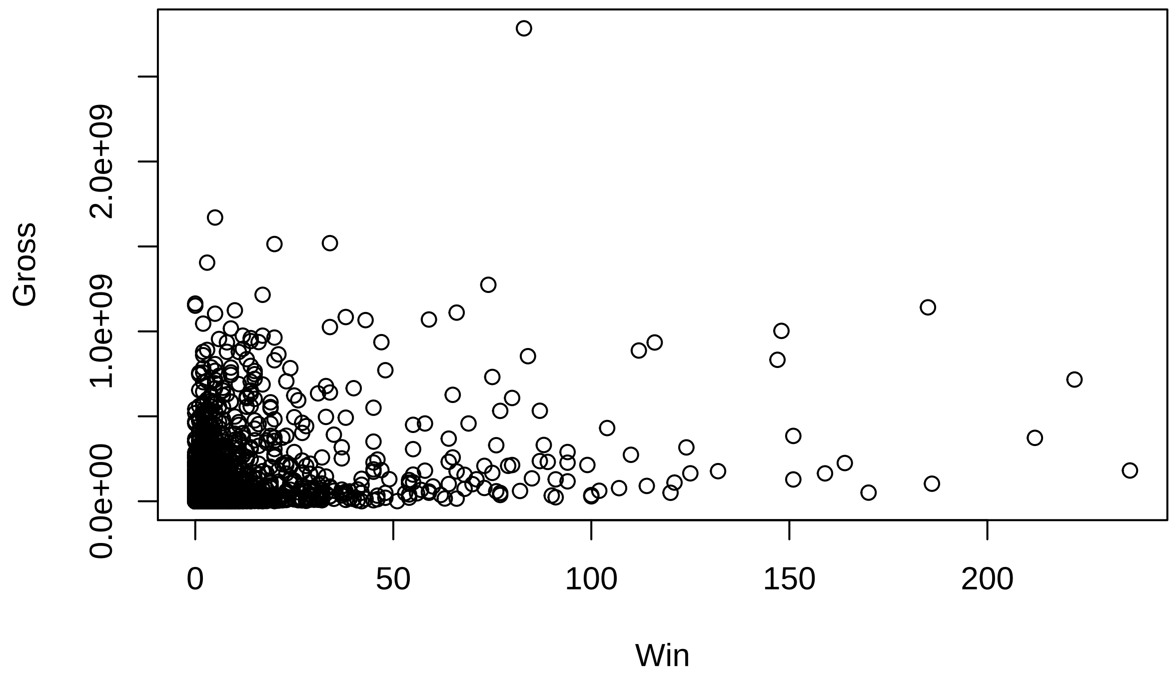
```
##  [1] 193731785  94558065  92910877  93180197  90127381  90718123  90799545
##  [8]  89702152  90057101  88825520  87924617  88615659  87897206  85420851
## [15]  84591403  84163092  83112934  82805450  81471386
```

## 3. Non-numeric variables

Write code that converts genre, actors, directors, and other categorical variables to columns that can be used for regression (e.g. binary columns as you did in Project 1). Also process variables such as awards into more useful columns (again, like you did in Project 1). Now use these converted columns only to build your next model.

```
# TODO: Build & evaluate model 3 (converted non-numeric variables only)
#make genres to binary columns
```

```r
df3=df
genres=trimws(unlist(strsplit(df3$Genre,",")))
genreDict=unique(genres)

for(genre in genreDict){
  genreCol=data.frame(as.numeric(grepl(genre,df3$Genre)))
  names(genreCol)=genre
  df3=cbind(df3,genreCol)
}
df3$Genre=NULL
#make top 10 actors to binary columns
actors=trimws(unlist(strsplit(df3$Actors,",")))
topactorDict=names(head(sort(table(actors),decreasing = T),10))
for(actor in topactorDict){
  actorCol=data.frame(as.numeric(grepl(actor,df3$Actors)))
  names(actorCol)=actor
  df3=cbind(df3,actorCol)
}
df3$Actors=NULL
#make top 10 directors to binary columns
directors=trimws(unlist(strsplit(df3$Director,",")))
topdirectorDict=names(head(sort(table(directors),decreasing = T),10))
for(director in topdirectorDict){
  directCol=data.frame(as.numeric(grepl(director,df3$Director)))
  names(directCol)=director
  df3=cbind(df3,directCol)
}
df3$Director=NULL

#make wins binary columns
convert_wins=function(award){
  num_win=as.numeric(regmatches(award,regexec("(\\d+) *win",award))[[1]][2])
  num_won=as.numeric(regmatches(award,regexec("[Ww]on *(\\d+)",award))[[1]][2])
  sum(num_win,num_won,na.rm = T) #if NA, will set NA to 0
}

df3$Win=sapply(df3$Awards,convert_wins)
plot(Gross~Win,df3)
```




```
#cut wins to five classes

df3$WinRank[df3$Win==0]='NoWin'
df3$WinRank[df3$Win==1]='OneWin'
df3$WinRank[df3$Win==2|df3$Win==3]='TwoOrThreeWins'
df3$WinRank[df3$Win>3 & df3$Win<=10 ]='Fourto10Wins'
df3$WinRank[df3$Win>10]='MoreThan10Wins'

winlist=c('NoWin','OneWin','TwoOrThreeWins','Fourto10Wins','MoreThan10Wins')
for(winrank in winlist){
  winrankCol=data.frame(as.numeric(grepl(winrank,df3$WinRank)))
  names(winrankCol)=winrank
  df3=cbind(df3,winrankCol)
}
df3$WinRank=NULL
df3$Win=NULL
#make nominations binary columns:
convert_nominations=function(award){
  num_nominated=as.numeric(regmatches(award,regexec("Nominated *for *(\\d+)",award))[[1]][2])
  num_nomination=as.numeric(regmatches(award,regexec("(\\d+) *nomi",award))[[1]][2])
  sum(num_nominated,num_nomination,na.rm = T)#if NA, will set NA to 0
}
df3$Nomination=sapply(df3$Awards,convert_nominations)
#cut wins to six classes

df3$NomiRank[df3$Nomination==0]='NoNomi'
```

```r
df3$NomiRank[df3$Nomination==1]='OneNomi'
df3$NomiRank[df3$Nomination==2|df3$Nomination==3]='TwoOrThreeNomi'
df3$NomiRank[df3$Nomination>3 & df3$Nomination<=7 ]='Fourto7Nomi'
df3$NomiRank[df3$Nomination>7 & df3$Nomination<=20]='Eightto20Nomi'
df3$NomiRank[df3$Nomination>20]='MoreThan20Nomi'

nomilist=c('NoNomi','OneNomi','TwoOrThreeNomi','Fourto7Nomi','Eightto20Nomi','MoreThan20Nomi')
for(nomirank in nomilist){
  nomirankCol=data.frame(as.numeric(grepl(nomirank,df3$NomiRank)))
  names(nomirankCol)=nomirank
  df3=cbind(df3,nomirankCol)
}
df3$Nomination=NULL
df3$NomiRank=NULL

df3$Awards=NULL

##make Rated to binary columns
unique(df3$Rated)
```

```
## [1] "R"         "PG"        "PG-13"     "NOT RATED" "UNRATED"   "G"
## [7] "NC-17"     "N/A"       "TV-G"
```

```r
df3$Rated[df3$Rated=="NOT RATED" | df3$Rated=="UNRATED" |df3$Rated=="N/A"] = "UNKNOWN"
ratedlist=unique(df3$Rated)
for(rated in ratedlist){
  ratedCol=data.frame(as.numeric(grepl(rated,df3$Rated)))
  names(ratedCol)=rated
  df3=cbind(df3,ratedCol)
}
df3$Rated=NULL


#Deal with languages

df3$NumLang=sapply(df$Language,function(language) if(is.na(language)) NA else length(strsplit(language,
table(df3$NumLanguage)
```

```
## < table of extent 0 >
```

```r
df3$NumLanguage[df3$NumLang==1]="OneLanguage"
df3$NumLanguage[df3$NumLang==2]="TwoLanguage"
df3$NumLanguage[df3$NumLang==3]="ThreeLanguage"
df3$NumLanguage[df3$NumLang>=4]="MoreThan3Language"

numlanglist=c("OneLanguage","TwoLanguage","ThreeLanguage","MoreThan3Language")

for(numlang in numlanglist){
  numlangCol=data.frame(numl=as.numeric(grepl(numlang,df3$NumLanguage)))
  names(numlangCol)=numlang
  df3=cbind(df3,numlangCol)
}
df3$NumLanguage=NULL
df3$Language=NULL
df3$NumLang=NULL
```

```r
#make top 20 countries to binary columns
countries=trimws(unlist(strsplit(df3$Country,",")))
topcountryDict=names(head(sort(table(countries),decreasing = T),20))
for(country in topcountryDict){
  countryCol=data.frame(as.numeric(grepl(country,df3$Country)))
  names(countryCol)=country
  df3=cbind(df3,countryCol)
}
df3$Country=NULL

#make top 20 productions to binary columns
productions=trimws(unlist(strsplit(df3$Production,",")))
topproductionDict=names(head(sort(table(productions),decreasing = T),20))
for(production in topproductionDict){
  productionCol=data.frame(as.numeric(grepl(production,df3$Production)))
  names(productionCol)=production
  df3=cbind(df3,productionCol)
}
df3$Production=NULL

#use only converted columns to make models to predict Gross
df3_m=df3[,26:ncol(df3)]

#let's if we need to remove any columns
summary(lm(Gross~.,df3_m))
```

```
##
## Call:
## lm(formula = Gross ~ ., data = df3_m)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -453461435  -64896140   -5554020   46178997 2267890265
##
## Coefficients: (5 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               156236592   79974515   1.954 0.050854 .
## Horror                     -9827775   10543926  -0.932 0.351379
## `Sci-Fi`                   41351224   10947168   3.777 0.000162 ***
## Adventure                  89017938    8922456   9.977  < 2e-16 ***
## Comedy                    -11155156    7562081  -1.475 0.140291
## Family                      8813511   13167944   0.669 0.503351
## Crime                      -3804485    7891594  -0.482 0.629779
## Music                      -7637016   14182690  -0.538 0.590294
## Drama                     -45015611    7191282  -6.260 4.47e-10 ***
## Mystery                    -6950403   10021263  -0.694 0.488014
## Thriller                   17385466    8527536   2.039 0.041573 *
## Sport                     -12738237   15482492  -0.823 0.410722
## Fantasy                    52646871   10432233   5.047 4.80e-07 ***
## Romance                     3335055    7837944   0.426 0.670505
## Action                     45851081    8016971   5.719 1.19e-08 ***
## Documentary               -23835278   18396917  -1.296 0.195220
## Biography                 -30562317   11836853  -2.582 0.009876 **
## History                   -16814563   17065223  -0.985 0.324559
```

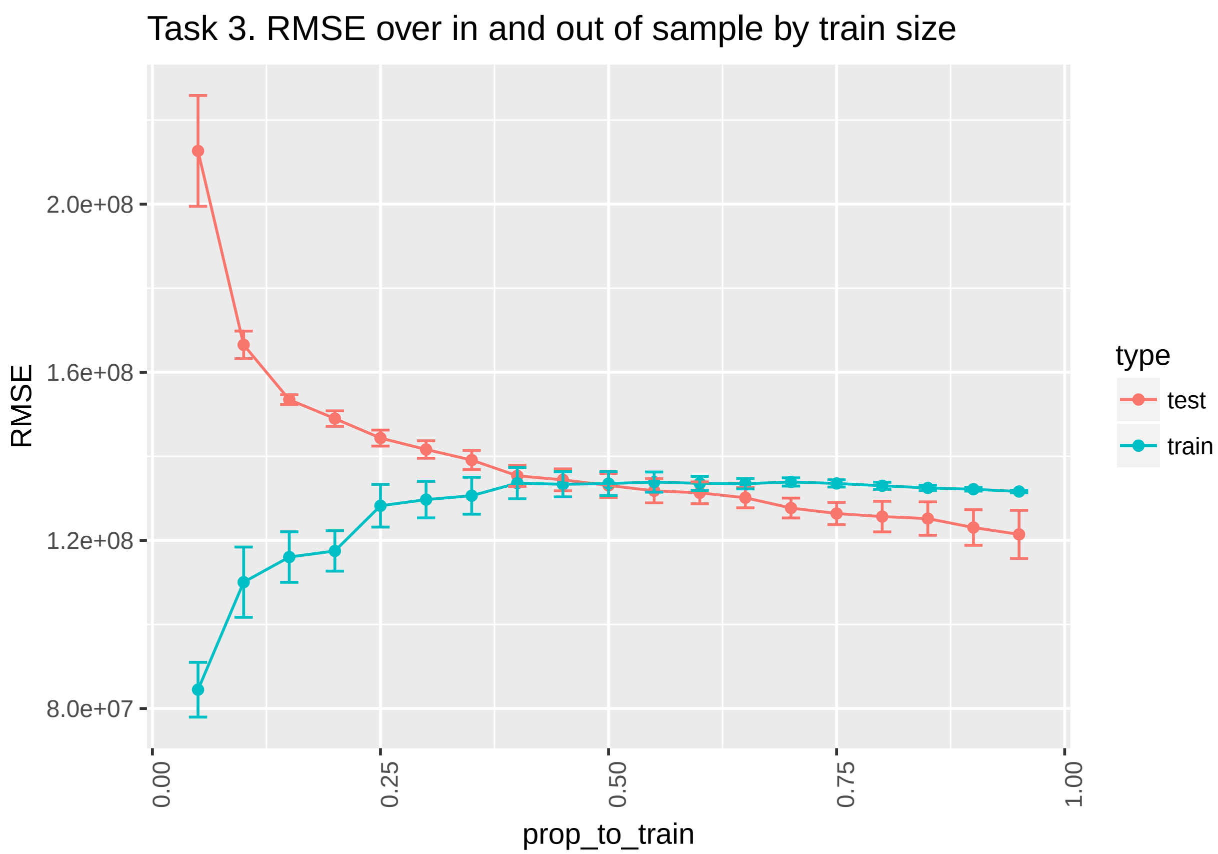
```
## Animation                   71700381   14956597   4.794 1.72e-06 ***
## Musical                    -33889385   31356178  -1.081 0.279888    
## Western                    -50099730   30956068  -1.618 0.105690    
## War                         -7130394   20061931  -0.355 0.722303    
## Short                      125293900   97732214   1.282 0.199948    
## News                       -13341565   96084476  -0.139 0.889577    
## `Robert De Niro`            45969731   25924683   1.773 0.076308 .  
## `Owen Wilson`              -21599192   29697322  -0.727 0.467098    
## `Ben Stiller`               90843711   30378526   2.990 0.002811 ** 
## `Adam Sandler`              33726720   30501761   1.106 0.268942    
## `Mark Wahlberg`             56597768   30116898   1.879 0.060315 .  
## `Samuel L. Jackson`         11456545   29870098   0.384 0.701345    
## `Bruce Willis`               3739332   30542020   0.122 0.902566    
## `Jack Black`                31188602   30523772   1.022 0.306976    
## `Matt Damon`               -19103714   31117711  -0.614 0.539321    
## `Nicolas Cage`                 93446   30425170   0.003 0.997550    
## `Steven Soderbergh`         43256482   36766894   1.177 0.239496    
## `Clint Eastwood`           -16387017   38307505  -0.428 0.668849    
## `Ridley Scott`              85238144   39963938   2.133 0.033025 *  
## `Woody Allen`               -1615162   40155832  -0.040 0.967919    
## `Ethan Coen`               -20369832   43384169  -0.470 0.638734    
## `Joel Coen`                       NA         NA      NA       NA    
## `Shawn Levy`                32705503   44660931   0.732 0.464045    
## `Steven Spielberg`          53260374   43482393   1.225 0.220730    
## `Robert Rodriguez`         -63312870   45593941  -1.389 0.165062    
## `Ron Howard`                49100980   45787190   1.072 0.283648    
## NoWin                      -59413993   13555469  -4.383 1.22e-05 ***
## OneWin                     -42300457   13422438  -3.151 0.001642 ** 
## TwoOrThreeWins             -26044581   12804892  -2.034 0.042054 *  
## Fourto10Wins               -32842501   10998211  -2.986 0.002850 ** 
## MoreThan10Wins                    NA         NA      NA       NA    
## NoNomi                    -155655661   14214829 -10.950  < 2e-16 ***
## OneNomi                   -158837622   14080553 -11.281  < 2e-16 ***
## TwoOrThreeNomi            -146012392   13282604 -10.993  < 2e-16 ***
## Fourto7Nomi               -135675980   12327726 -11.006  < 2e-16 ***
## Eightto20Nomi              -89826440   10910598  -8.233 2.81e-16 ***
## MoreThan20Nomi                    NA         NA      NA       NA    
## R                           35881192   78209082   0.459 0.646425    
## PG                          16765824   20802589   0.806 0.420343    
## `PG-13`                     28997422   11449332   2.533 0.011376 *  
## UNKNOWN                     49101924   79024068   0.621 0.534419    
## G                           33442321   81221051   0.412 0.680559    
## `NC-17`                           NA         NA      NA       NA    
## `TV-G`                    -130508169   97490069  -1.339 0.180787    
## OneLanguage                -36528973   11605898  -3.147 0.001665 ** 
## TwoLanguage                -18265110   12205394  -1.496 0.134646    
## ThreeLanguage              -25041973   14075743  -1.779 0.075338 .  
## MoreThan3Language                 NA         NA      NA       NA    
## USA                         49566471   10169235   4.874 1.16e-06 ***
## UK                           9316039    7688683   1.212 0.225750    
## Germany                    -17452374    8516786  -2.049 0.040542 *  
## Canada                     -14455942    9435155  -1.532 0.125606    
## France                     -11348420   10294126  -1.102 0.270379    
## Australia                  -32127204   15337325  -2.095 0.036290 *  
```

```
## Spain                        -26922800   19875464  -1.355 0.175667
## Japan                         -8737422   20583957  -0.424 0.671252
## Italy                          4426404   22076986   0.200 0.841106
## Ireland                       -9418606   23076804  -0.408 0.683202
## India                        -39488194   24542965  -1.609 0.107746
## China                         48305023   26036071   1.855 0.063661 .
## `Hong Kong`                  -48530774   30173200  -1.608 0.107863
## `New Zealand`                143154708   28878318   4.957 7.60e-07 ***
## `Czech Republic`             -22786834   33423165  -0.682 0.495444
## Switzerland                  -20063805   33530661  -0.598 0.549642
## `South Africa`               -50752926   34484513  -1.472 0.141202
## Mexico                       -40984737   35768116  -1.146 0.251961
## Netherlands                    9600279   36068260   0.266 0.790129
## `United Arab Emirates`       -18902176   35846114  -0.527 0.598019
## `Warner Bros. Pictures`        3168256   21993838   0.144 0.885470
## `Universal Pictures`          50292413   10956195   4.590 4.63e-06 ***
## `20th Century Fox`            52453267   10950544   4.790 1.76e-06 ***
## `Paramount Pictures`          21453201   12712984   1.688 0.091622 .
## `Sony Pictures`               39348614   11731990   3.354 0.000808 ***
## `Sony Pictures Classics`     -71403826   19052066  -3.748 0.000182 ***
## `New Line Cinema`             24388145   15697264   1.554 0.120384
## `Walt Disney Pictures`       177372046   18287863   9.699  < 2e-16 ***
## `Columbia Pictures`           11569954   17678044   0.654 0.512857
## `Miramax Films`              -12862100   18443864  -0.697 0.485635
## `Focus Features`             -43160755   18706356  -2.307 0.021115 *
## `Lionsgate Films`             10429254   27397308   0.381 0.703480
## `Warner Bros.`                57554824   20817763   2.765 0.005736 **
## MGM                           23883336   16743420   1.426 0.153860
## `The Weinstein Company`       -1915647   20917121  -0.092 0.927036
## Lionsgate                     10729507   19988966   0.537 0.591470
## `Magnolia Pictures`           -6686162   22502533  -0.297 0.766391
## `Fox Searchlight`            -42788143   22818735  -1.875 0.060882 .
## `Fox Searchlight Pictures`      768779   31284361   0.025 0.980397
## `Summit Entertainment`        38607878   22726169   1.699 0.089467 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 133500000 on 2698 degrees of freedom
## Multiple R-squared:  0.4975, Adjusted R-squared:  0.4789
## F-statistic: 26.71 on 100 and 2698 DF,  p-value: < 2.2e-16
```

```r
#we have to remove redundant columns
#remove "Joel Coen","NC-17","MoreThan10Wins","MoreThan20Nomi","MoreThan3Language" columns
df3_m=df3_m[,-c(40,49,55,61,66)]
```

```r
N=dim(df3_m)[1]
get_rmse_list3=function(df,prop_to_train,N){#helper funciton in task 3
  rmse_list=c()
  prop_to_train_list=c()
  type=c()
  set.seed(200)
  seeds=sample(10000,10)
  for( seed in seeds){
    set.seed(seed)
    to_train=sample(N,prop_to_train*N)
```

```
    train=df[to_train,]
    test=df[-to_train,]
    #make models
    model=lm(Gross~.,data = train)
    #make predictions and compute RMSEs
    pred_trainGross=predict(model,train[,-1])
    rmse_train=sqrt(mean((pred_trainGross-train[,1])^2))

    pred_testGross=predict(model,test[,-1])
    rmse_test=sqrt(mean((pred_testGross-test[,1])^2))
    #record results
    rmse_list=c(rmse_list,rmse_train,rmse_test)
    prop_to_train_list=c(prop_to_train_list,prop_to_train,prop_to_train)
    type=c(type,"train","test")
  }
  data.frame(prop_to_train=prop_to_train_list,type=type,rmse=rmse_list)
}
rmse_df3=get_rmse_list3(df3_m,0.05,N) #start iterations to compute RMSEs
for (prop_to_train in (2:19)*0.05){
  rmse_df3=rbind(rmse_df3,get_rmse_list3(df3_m,prop_to_train,N))
}
average_rmse3=aggregate(rmse_df3["rmse"],by=rmse_df3[c("type","prop_to_train")],function(X) c(mean=mean
average_rmse3=do.call(data.frame,average_rmse3)
names(average_rmse3)[3:5]=c("rmse","sd","N")
average_rmse3$se=average_rmse3$sd/sqrt(average_rmse3$N)
#plot RMSEs in TASK3
ggplot(average_rmse3,aes(prop_to_train,rmse,group=type,col=type))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin=rmse-se,ymax=rmse+se),width=0.02)+
  ylab("RMSE")+
  theme(axis.text.x = element_text(angle=90, hjust=1))+
  ggtitle("Task 3. RMSE over in and out of sample by train size")
```




**Q**: Explain which categorical variables you used, and how you encoded them into features.

**A**: 1).make genres to binary columns how encoded genres to features: First gain the all unique genres list, iterates the list to see if each row of Genre column has the genre appeared(1 for present, and 0 for not present). So we get 23 new genres columns. Then, the orignial Genre column is removed. See detail codes from lines 380-390 in this rmd file.

2) make top 10 actors to binary columns how encoded top 10 actors to features: First gain all actors appeared in the Actors column in a list. Table the list and get the actor names with top 10 occurance in the summary table. Then iterate top 10 actors to see if each row of Actors column has the actor appeared(1 for present, and 0 for not present). So we get 10 new actors columns. Then, the original Actors column is removed. See detail codes from lines 391-399.

3) make top 10 directors to binary columns how encoded top 10 directors to features: First gain all directors appeared in the Director column in a list. Table the list and get the director names with top 10 occurance in the summary table. Then iterate top 10 directors to see if each row of Director column has the director appeared(1 for present, and 0 for not present). So we get 10 new directors columns. Then, the original Director column is removed.

See detail codes from lines 400-408.

4) extract number of wins and nominations from Awards column and cut them into 5 and 6 classes respectively. how encoded Awards to features: In Awards column, first use regex to find number of wins and nominations for each row, and then bin them into several classes to make two new columns WinRank and NomiRank. See detail summary of two columns below:

```r
table(df3$WinRank)
```

```
## < table of extent 0 >
```

```r
table(df3$NomiRank)
```

```
## < table of extent 0 >
```

Then remove Awards column.

Detail codes can be seen in lines 410-442.

5) make Rated to binary columns how encoded Rated to features: First check how many types of Rated in the column. combine "NOT RATED", "UNRATED","N/A" to a same type "UNKNOWN". Now we have the list of Rated. Iterate the list to see if each row has the type of Rated appeared (1 for present, and 0 for absent). Now we have 7 new Rated columns. Then set original Rated column to NULL.

See detailed codes from line 444 to 453.

6) Deal with languages How code Language to features: First iterate each row of column Language to count how many languages appeared and assign the counts to a new column "NumLanguage". Make rows with more than 3 languages to a same class "MoreThan3". Make the column as factor type. So we have four levels in the column. See table below. Then remove Language column. Detail code can be seen from line 456 to 462.

```r
table(df3$NumLanguage)
```

```
## < table of extent 0 >
```

7) make top 20 countries to binary columns How code top 20 countries to features:

First gain all countries appeared in the Country column in a list. Table the list and get the country names with top 20 occurances in the summary table. Then iterate top 20 countries to see if each row of Country column has the country appeared(1 for present, and 0 for not present). So we get 20 new countries columns. Then, the original Country column is removed.

Detail codes can be seen from line 464 to 472.

8) make top 20 productions to binary columns How code top 20 productions to features:

First gain all productions appeared in the Production column in a list. Table the list and get the production names with top 20 occurances in the summary table. Then iterate top 20 productions to see if each row of Production column has the production appeared(1 for present, and 0 for not present). So we get 20 new productions columns. Then, the original Production column is removed.

Detail codes can be seen from line 474 to 482.

In summary, train RMSEs are around 132M and test RMSEs are around 122M in task 3.

```r
#print average train RMSEs
average_rmse3$rmse[average_rmse3$type=="train"]
```

```
##  [1]  84480165 110058443 116040238 117488259 128234389 129702172 130630356
##  [8] 133624155 133348357 133522990 133879043 133569002 133485150 133900357
## [15] 133549097 132997090 132480588 132166452 131632143
```

```r
#print average test RMSEs
average_rmse3$rmse[average_rmse3$type=="test"]
```

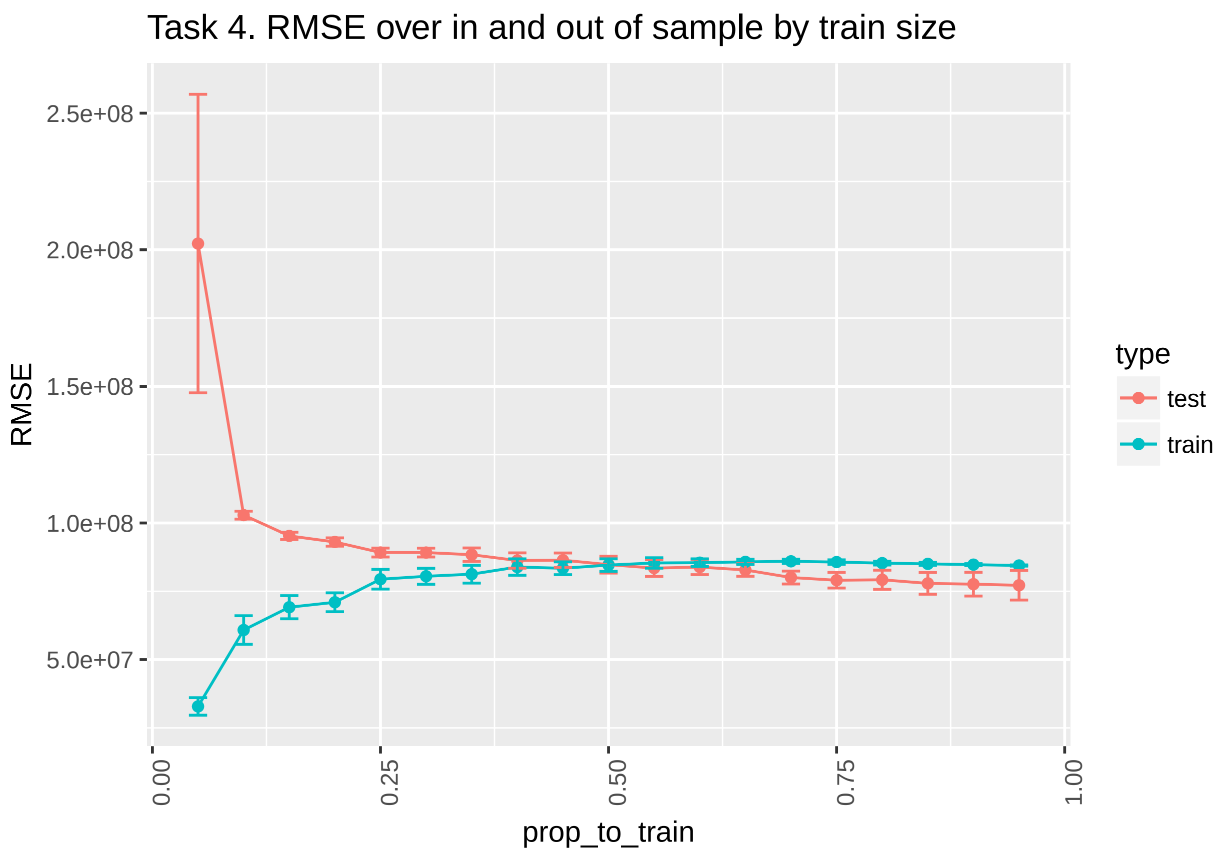
```
##  [1] 212658564 166521649 153483945 148982780 144344417 141626808 139109275
##  [8] 135367698 134402149 133073962 131794820 131319212 130146741 127699150
## [15] 126397545 125661653 125188753 123058394 121428323
```

## 4. Numeric and categorical variables

Try to improve the prediction quality as much as possible by using both numeric and non-numeric variables from **Tasks 2 & 3**.

```
# TODO: Build & evaluate model 4 (numeric & converted non-numeric variables)
#combine numeric variables from task 2 and converted variables from task 3
df4=cbind(dfn2,df3_m[,-1])
N=dim(df4)[1]
#helper function to iterate  10 times for computing RMSEs given a train size in task 4
get_rmse_list4=function(df,prop_to_train,N){
  #lists to record results
  rmse_list=c()
  prop_to_train_list=c()
  type=c()
  #set random seeds for good comparisions between tasks
  set.seed(200)
  seeds=sample(10000,10)
  #10 iterations
  for(seed in seeds){
    set.seed(seed)
    #data partitions
    to_train=sample(N,prop_to_train*N)
    train=df[to_train,]
    test=df[-to_train,]
    #build linear regression model
    model=lm(Gross~.,data = train)
    #make predicitons and compute RMSE
    pred_trainGross=(predict(model,train[,-1]))^2
    rmse_train=sqrt(mean((pred_trainGross-(train[,1])^2)^2))

    pred_testGross=(predict(model,test[,-1]))^2
    rmse_test=sqrt(mean((pred_testGross-(test[,1])^2)^2))
    #record results
    rmse_list=c(rmse_list,rmse_train,rmse_test)
    prop_to_train_list=c(prop_to_train_list,prop_to_train,prop_to_train)
    type=c(type,"train","test")
  }
  data.frame(prop_to_train=prop_to_train_list,type=type,rmse=rmse_list)
}
#iterate train size to compute RMSEs
rmse_df4=get_rmse_list4(df4,0.05,N)
for (prop_to_train in (2:19)*0.05){
  rmse_df4=rbind(rmse_df4,get_rmse_list4(df4,prop_to_train,N))
}
#summary RMSEs results to get mean and se
average_rmse4=aggregate(rmse_df4["rmse"],by=rmse_df4[c("type","prop_to_train")],function(X) c(mean=mean
average_rmse4=do.call(data.frame,average_rmse4)
names(average_rmse4)[3:5]=c("rmse","sd","N")
average_rmse4$se=average_rmse4$sd/sqrt(average_rmse4$N)

#plot RMSE results in task 4
ggplot(average_rmse4,aes(prop_to_train,rmse,group=type,col=type))+
  geom_point()+
```

```
geom_line()+
geom_errorbar(aes(ymin=rmse-se,ymax=rmse+se),width=0.02)+
ylab("RMSE")+
scale_y_continuous()+
theme(axis.text.x = element_text(angle=90, hjust=1))+
ggtitle("Task 4. RMSE over in and out of sample by train size")
```




After Task 4, Train RMSEs drop to around 85 M, and test RMSEs drop to around 77M.

```
#print average train RMSEs
average_rmse4$rmse[average_rmse4$type=="train"]
```

```
##  [1] 32857377 60817084 69164264 70974436 79409552 80490963 81261790
##  [8] 83873678 83417737 84590480 85369380 85469756 85763406 85969770
## [15] 85691401 85301172 85032317 84748063 84437913
```

```
#print average test RMSEs
average_rmse4$rmse[average_rmse4$type=="test"]
```

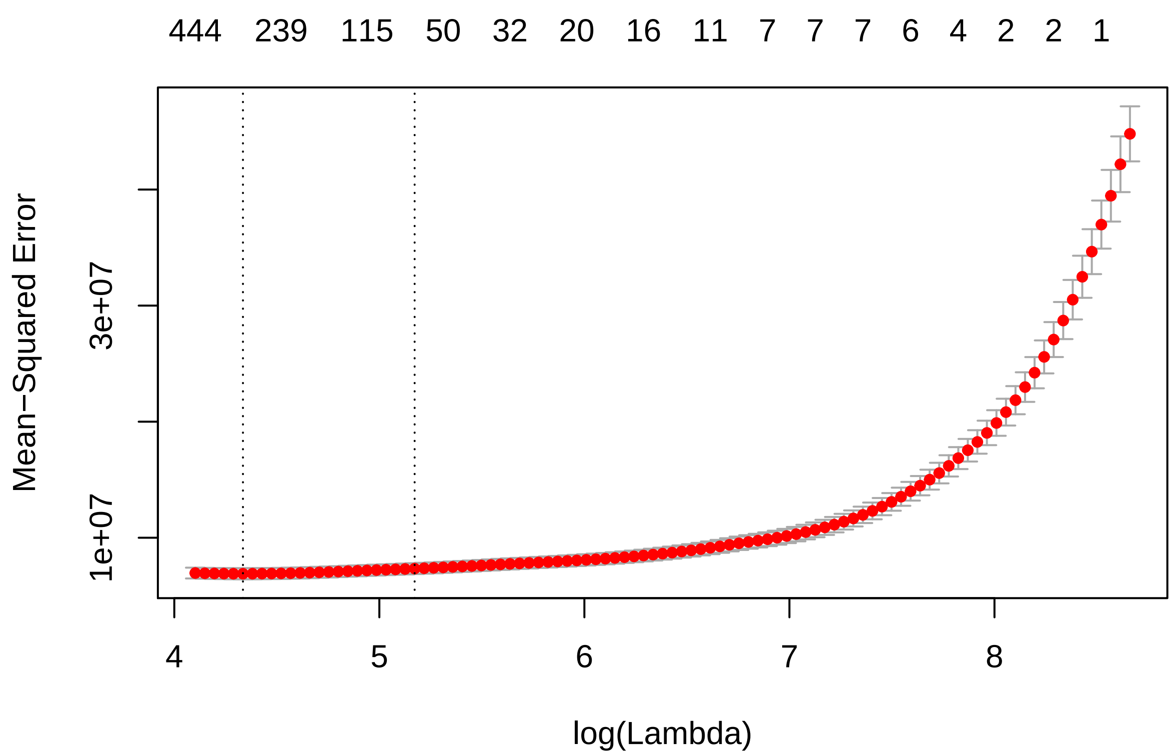
```
##  [1] 202263279 102875791  95278606  93037056  89188505  89176263  88384706
##  [8]  86260294  86379367  84750988  83486120  83858627  82799623  80038342
## [15]  79038740  79210099  77887920  77600911  77196680
```

## 5. Additional features

Now try creating additional features such as interactions (e.g. `is_genre_comedy` x `is_budget_greater_than_3M`) or deeper analysis of complex variables (e.g. text analysis of full-text columns like `Plot`).

```r
# TODO: Build & evaluate model 5 (numeric, non-numeric and additional features)
#label two clusters of tomatoUserReviews for later interation term in linear regression
df5=df4
df5$tomatoUserReviews=sqrt(df4$tomatoUserReviews)

df5$tomatoreviewrank[df5$tomatoUserReviews<=3000]=0
df5$tomatoreviewrank[df5$tomatoUserReviews>3000]=1
```

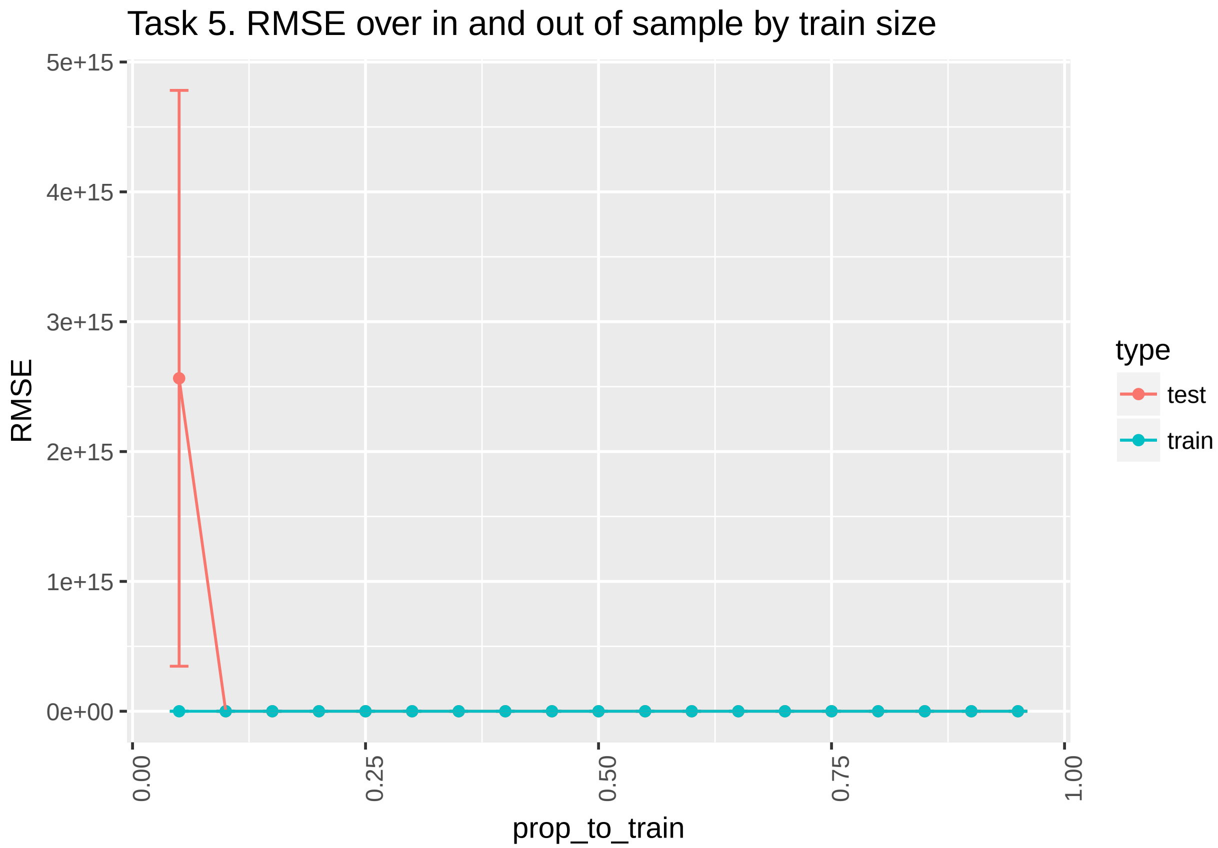
```r
f <- as.formula(Gross~.^2)
x <- model.matrix(f, df5)
cvgnetfit=cv.glmnet(x,df5$Gross,family="gaussian",type.measure = "mse",alpha=1)
plot(cvgnetfit)
```



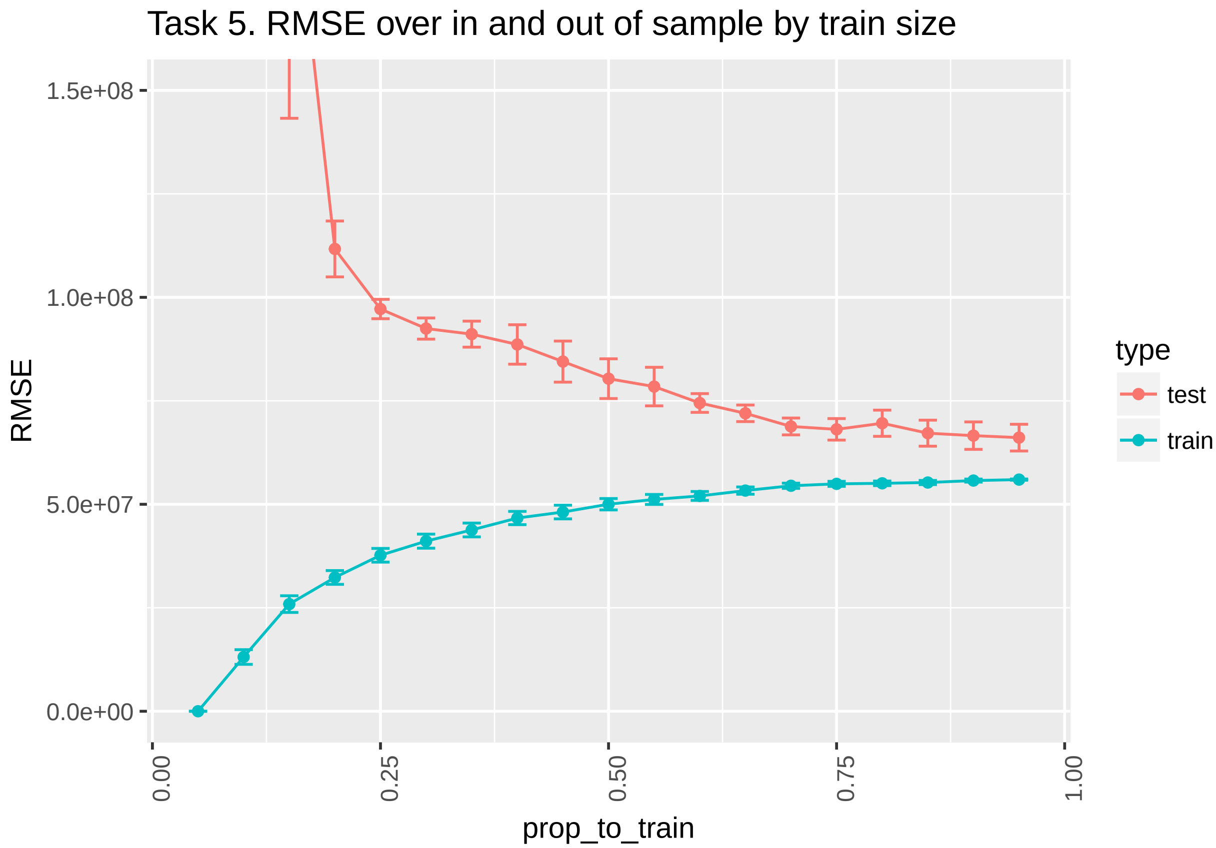

```r
tmp_coeffs <- coef(cvgnetfit, s = "lambda.min")
termname=rownames(tmp_coeffs)
goodterms=termname[which(tmp_coeffs != 0)][-1]
fml=formula(paste0("Gross~",paste0(goodterms,collapse = "+")))
```

```r
#helper function to iterate  10 times for computing RMSEs given a train size in task 5
get_rmse_list5=function(df,prop_to_train,N){
  #lists to record results
  rmse_list=c()
  prop_to_train_list=c()
  type=c()
  #set random seeds for good comparisons between taskes
  set.seed(200)
  seeds=sample(10000,10)
```

```r
  #10 iterations
  for( seed in seeds){
    set.seed(seed)
    #data partition
    to_train=sample(N,prop_to_train*N)
    train=df[to_train,]
    test=df[-to_train,]
    #build linear regression models with interaction terms
    #model=lm(Gross~.+Budget:imdbRating+Budget:tomatoUserRating+Budget:imdbVotes+imdbVotes:imdbRating+t
    model=lm(fml,train)
    #make predicitons and compute RMSEs
    pred_trainGross=(predict(model,train[,-1]))^2
    rmse_train=sqrt(mean((pred_trainGross-(train[,1])^2)^2))

    pred_testGross=(predict(model,test[,-1]))^2
    rmse_test=sqrt(mean((pred_testGross-(test[,1])^2)^2))
    #record results
    rmse_list=c(rmse_list,rmse_train,rmse_test)
    prop_to_train_list=c(prop_to_train_list,prop_to_train,prop_to_train)
    type=c(type,"train","test")
  }
  data.frame(prop_to_train=prop_to_train_list,type=type,rmse=rmse_list)
}
#start iterations on train size to compute RMSEs
rmse_df5=get_rmse_list5(df5,0.05,N)
for (prop_to_train in (2:19)*0.05){
  rmse_df5=rbind(rmse_df5,get_rmse_list5(df5,prop_to_train,N))
}
#summary RMSEs and get mean and se
average_rmse5=aggregate(rmse_df5["rmse"],by=rmse_df5[c("type","prop_to_train")],function(X) c(mean=mean
average_rmse5=do.call(data.frame,average_rmse5)
names(average_rmse5)[3:5]=c("rmse","sd","N")
average_rmse5$se=average_rmse5$sd/sqrt(average_rmse5$N)
#plot RMSEs in task 5
task5plot1=ggplot(average_rmse5,aes(prop_to_train,rmse,group=type,col=type))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin=rmse-se,ymax=rmse+se),width=0.02)+
  ylab("RMSE")+
  theme(axis.text.x = element_text(angle=90, hjust=1))+
  ggtitle("Task 5. RMSE over in and out of sample by train size")
task5plot1
```




```
#zoom in plot of task5
task5plot1+ coord_cartesian(ylim= c(0,150000000))
```




**Q**: Explain what new features you designed and why you chose them.

**A**: I add five interaction terms in the models used in Task 5. Here are the interaction terms: Budget:imdbRating,Budget:tomatoUserRating, Budget:imdbVotes, imdbVotes:imdbRating, tomatoUserReviews:tomatoreviewrank.

First reason, I add them is that when I stepwise add those interaction terms, I see drops in train RMSE by models from full size of data. And also I see the obvious significance effect of the interaction term in the model. Check the full model summary below.

```
s5=summary(lm(fml,df5))
s5
```

```
##
## Call:
## lm(formula = fml, data = df5)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7367.5 -1175.4   -76.2  1026.1 14278.8
##
## Coefficients: (13 not defined because of singularities)
##                                         Estimate Std. Error t value
## (Intercept)                            7.181e+04  3.398e+04   2.113
## imdbVotes                             -5.500e+02  1.233e+02  -4.460
## imdbRating7                           -1.771e+04  6.031e+04  -0.294
## Year                                  -3.761e+01  1.692e+01  -2.222
## `Sci-Fi`                               9.297e+00  2.932e+02   0.032
```

```
## War                                     -8.041e+02  2.476e+03  -0.325
## `TV-G`                                   4.694e+03  2.021e+04   0.232
## UK                                       2.634e+00  1.613e+02   0.016
## Budget:tomatoFresh                       6.395e-03  5.962e-03   1.073
## imdbVotes:Budget                         5.778e-04  1.280e-04   4.513
## Budget:tomatoUserRating                  1.480e-01  1.922e-02   7.702
## Budget:Adventure                        -7.110e-03  3.682e-02  -0.193
## Budget:Comedy                           -1.412e-02  4.042e-02  -0.349
## Budget:Drama                            -5.053e-02  3.645e-02  -1.386
## Budget:Sport                            -1.121e-02  5.090e-02  -0.220
## Budget:Western                          -2.751e-02  8.955e-02  -0.307
## Budget:Short                            -9.905e-01  4.913e+00  -0.202
## Budget:NoWin                            -1.013e-01  2.543e-02  -3.985
## Budget:NoNomi                           -1.165e-01  4.884e-02  -2.386
## Budget:TwoLanguage                       3.050e-02  2.618e-02   1.165
## Budget:`Sony Pictures`                   2.980e-02  2.634e-02   1.131
## Budget:`Magnolia Pictures`              -1.486e-01  2.318e-01  -0.641
## imdbVotes:tomatoRotten                   3.502e-01  1.527e-01   2.293
## tomatoRotten:imdbRating1                 1.459e+02  3.374e+01   4.324
## imdbRating7:tomatoRotten                -3.740e+00  6.392e+01  -0.059
## tomatoRotten:imdbRating10               -1.183e+01  9.304e+01  -0.127
## tomatoUserRating:tomatoRotten            2.453e+01  1.291e+01   1.900
## tomatoRotten:Horror                      4.086e+01  5.037e+01   0.811
## tomatoRotten:Family                     -6.797e+01  5.053e+01  -1.345
## tomatoRotten:Animation                   1.854e+02  6.218e+01   2.981
## Short:tomatoRotten                       1.660e+03  4.309e+03   0.385
## `TV-G`:tomatoRotten                     -6.347e+03  1.082e+04  -0.587
## tomatoFresh:tomatoUserRating             7.047e+00  9.522e+00   0.740
## tomatoFresh:Adventure                    1.828e+01  3.031e+01   0.603
## tomatoFresh:Family                       5.441e+01  4.208e+01   1.293
## tomatoFresh:OneWin                      -1.658e+01  2.183e+01  -0.760
## `TV-G`:tomatoFresh                              NA         NA      NA
## imdbVotes:imdbRating2                    6.452e+00  1.018e+00   6.339
## imdbVotes:imdbRating3                    4.406e+00  9.762e-01   4.514
## imdbVotes:imdbRating4                    3.065e+00  9.140e-01   3.353
## imdbVotes:imdbRating5                    1.744e+00  7.175e-01   2.431
## imdbVotes:imdbRating10                  -4.304e+00  9.991e-01  -4.308
## imdbVotes:Year                           2.783e-01  6.150e-02   4.524
## imdbVotes:Comedy                        -6.076e-03  6.843e-01  -0.009
## imdbVotes:Animation                      3.292e+00  1.280e+00   2.572
## imdbVotes:`Clint Eastwood`               4.056e+00  1.894e+00   2.142
## imdbVotes:PG                             4.861e-01  1.139e+00   0.427
## imdbVotes:G                              2.195e+00  1.316e+00   1.668
## imdbVotes:`TV-G`                                NA         NA      NA
## imdbVotes:USA                            2.734e-01  8.970e-01   0.305
## imdbVotes:`Universal Pictures`           1.523e+00  6.369e-01   2.391
## imdbVotes:`20th Century Fox`             2.901e+00  6.209e-01   4.673
## imdbVotes:`Magnolia Pictures`           -2.157e+00  3.759e+00  -0.574
## imdbVotes:`Fox Searchlight Pictures`     2.418e+00  1.276e+00   1.894
## Runtime1:tomatoUserReviews               5.509e-02  2.112e-01   0.261
## Adventure:Runtime1                       1.413e+02  3.260e+02   0.433
## Family:Runtime1                          3.694e+02  4.261e+02   0.867
## Sport:Runtime1                          -8.132e+02  5.813e+02  -1.399
## Runtime1:Japan                           1.206e+02  1.054e+03   0.114
```

```
## `20th Century Fox`:Runtime1               3.510e+02  3.557e+02   0.987
## Horror:Runtime2                           4.350e+02  3.669e+02   1.186
## Runtime2:Mystery                          5.455e+02  4.992e+02   1.093
## Runtime2:Romance                         -6.655e+02  3.252e+02  -2.046
## Runtime2:Action                           3.997e+02  3.635e+02   1.100
## TwoLanguage:Runtime2                      3.023e+02  3.274e+02   0.923
## Runtime2:`Hong Kong`                      2.089e+03  1.491e+03   1.401
## `Universal Pictures`:Runtime2             2.202e+03  6.271e+02   3.512
## Runtime2:`Walt Disney Pictures`           4.377e+03  2.160e+03   2.026
## Runtime3:Fantasy                         -5.304e+02  4.611e+02  -1.150
## Runtime3:`Ben Stiller`                    4.707e+03  2.637e+03   1.785
## Runtime3:`Matt Damon`                    -6.587e+03  3.070e+03  -2.145
## imdbRating4:Runtime4                      3.208e+02  4.106e+02   0.781
## Runtime4:`New Line Cinema`                1.928e+03  9.185e+02   2.100
## Fantasy:Runtime5                         -4.877e+02  4.406e+02  -1.107
## Action:Runtime6                          -8.878e+02  3.014e+02  -2.945
## Runtime6:R                               -3.169e+02  2.143e+02  -1.478
## imdbRating1:Runtime7                      1.349e+03  4.616e+02   2.923
## Runtime7:TwoOrThreeNomi                  -7.402e+02  3.354e+02  -2.207
## Runtime7:Eightto20Nomi                    2.284e+01  3.168e+02   0.072
## Runtime7:Canada                          -5.576e+02  4.563e+02  -1.222
## Japan:Runtime7                           -4.402e+03  1.633e+03  -2.695
## Runtime7:`Warner Bros. Pictures`          4.300e+02  4.616e+02   0.932
## `Walt Disney Pictures`:Runtime7           3.657e+03  1.045e+03   3.499
## Runtime7:`Lionsgate Films`                2.481e+03  1.225e+03   2.025
## `Sci-Fi`:Runtime8                        -5.599e+02  5.877e+02  -0.953
## Adventure:Runtime8                       -1.220e+03  3.588e+02  -3.400
## Adventure:Runtime10                       9.439e+02  4.567e+02   2.067
## Family:Runtime10                          1.811e+03  9.206e+02   1.967
## Drama:Runtime10                          -2.306e+02  2.237e+02  -1.031
## Fantasy:Runtime10                         8.861e+02  5.094e+02   1.739
## Runtime10:`Steven Soderbergh`             5.819e+03  1.454e+03   4.001
## Runtime10:TwoOrThreeWins                 -5.976e+02  4.026e+02  -1.484
## Eightto20Nomi:Runtime10                  -7.330e+02  3.218e+02  -2.278
## Runtime10:ThreeLanguage                  -7.682e+02  4.119e+02  -1.865
## Japan:Runtime10                           5.195e+03  1.558e+03   3.335
## Comedy:imdbRating1                        2.761e+02  2.233e+02   1.237
## imdbRating1:`Steven Spielberg`            5.163e+03  1.519e+03   3.399
## imdbRating1:`Ron Howard`                 -2.316e+03  1.082e+03  -2.140
## imdbRating1:TwoOrThreeWins                3.541e+02  3.377e+02   1.049
## imdbRating1:G                             1.309e+02  2.249e+02   0.582
## imdbRating1:`New Zealand`                 1.184e+03  9.062e+02   1.306
## imdbRating1:`Walt Disney Pictures`        1.286e+03  6.586e+02   1.953
## imdbRating1:`Fox Searchlight`            -1.015e+01  9.269e+02  -0.011
## imdbRating1:`Summit Entertainment`        1.839e+03  9.742e+02   1.888
## imdbRating2:Fantasy                      -8.899e+02  4.333e+02  -2.054
## imdbRating2:Eightto20Nomi                 3.163e+02  4.057e+02   0.780
## imdbRating3:Musical                      -1.890e+03  1.231e+03  -1.536
## imdbRating3:`Mark Wahlberg`               4.552e+03  2.157e+03   2.111
## imdbRating3:TwoOrThreeWins                6.384e+02  3.842e+02   1.661
## imdbRating3:Fourto7Nomi                   1.537e+02  3.438e+02   0.447
## imdbRating3:China                         3.958e+02  3.399e+03   0.116
## imdbRating3:`Paramount Pictures`          9.838e+02  5.733e+02   1.716
## `Sony Pictures`:imdbRating3               9.788e+02  5.232e+02   1.871
```

```
## imdbRating4:tomatoUserReviews            1.051e-01  3.091e-01   0.340
## imdbRating4:Fourto10Wins                 2.990e+02  4.793e+02   0.624
## imdbRating4:`Paramount Pictures`         1.632e+02  6.303e+02   0.259
## Animation:imdbRating5                    1.805e+03  6.162e+02   2.930
## imdbRating5:Fourto10Wins                 6.592e+02  3.740e+02   1.763
## imdbRating5:OneNomi                     -6.775e+02  3.373e+02  -2.008
## imdbRating5:Eightto20Nomi                2.392e+02  3.452e+02   0.693
## imdbRating5:`Universal Pictures`         3.631e+02  4.260e+02   0.852
## Adventure:imdbRating6                   -2.578e+02  3.771e+02  -0.684
## Action:imdbRating6                       3.796e+01  3.497e+02   0.109
## imdbRating6:Biography                   -9.843e+02  5.712e+02  -1.723
## imdbRating6:`Adam Sandler`               8.117e+03  2.111e+03   3.845
## `Mark Wahlberg`:imdbRating6             -2.347e+03  1.617e+03  -1.451
## `Ron Howard`:imdbRating6                 5.374e+03  1.504e+03   3.574
## OneWin:imdbRating6                      -6.484e+02  3.552e+02  -1.825
## TwoOrThreeNomi:imdbRating6              -3.265e+02  3.416e+02  -0.956
## imdbRating7:Year                         9.055e+00  3.006e+01   0.301
## imdbRating7:Horror                      -1.034e+03  5.051e+02  -2.047
## imdbRating7:NoWin                       -4.745e+02  3.514e+02  -1.350
## imdbRating7:OneWin                      -6.863e+02  3.602e+02  -1.905
## imdbRating7:`PG-13`                     -8.258e+02  2.905e+02  -2.843
## imdbRating7:USA                         -4.766e+02  4.256e+02  -1.120
## Romance:imdbRating8                     -3.090e+02  3.567e+02  -0.866
## OneWin:imdbRating8                      -4.276e+02  3.978e+02  -1.075
## OneNomi:imdbRating8                     -1.314e+03  7.278e+02  -1.805
## Eightto20Nomi:imdbRating8               -5.546e+02  3.001e+02  -1.848
## `PG-13`:imdbRating8                     -3.977e+02  3.307e+02  -1.203
## imdbRating10:tomatoUserReviews          -2.916e-01  2.355e-01  -1.239
## `Sci-Fi`:imdbRating10                   -1.940e+03  5.789e+02  -3.352
## imdbRating10:Family                     -2.972e+03  8.697e+02  -3.417
## imdbRating10:Mystery                    -9.020e+02  5.146e+02  -1.753
## imdbRating10:Thriller                   -1.941e+02  4.379e+02  -0.443
## imdbRating10:`Steven Spielberg`                 NA         NA      NA
## imdbRating10:`Robert Rodriguez`         -4.790e+03  2.373e+03  -2.018
## imdbRating10:`Ron Howard`                       NA         NA      NA
## imdbRating10:OneWin                     -1.538e+03  8.783e+02  -1.751
## imdbRating10:Fourto10Wins               -1.436e+03  3.980e+02  -3.608
## imdbRating10:TwoOrThreeNomi             -3.187e+03  1.261e+03  -2.527
## imdbRating10:`20th Century Fox`          1.728e+03  7.066e+02   2.445
## imdbRating10:Lionsgate                  -3.764e+03  1.089e+03  -3.457
## imdbRating10:`Fox Searchlight`           1.710e+03  1.147e+03   1.491
## NoNomi:tomatoRating                     -1.864e+01  4.431e+01  -0.421
## tomatoUserRating:tomatoUserReviews       3.385e-01  4.670e-02   7.248
## Year:tomatoUserRating                    4.073e-01  9.269e-02   4.394
## War:tomatoUserRating                     7.558e+01  7.041e+02   0.107
## tomatoUserRating:G                       1.314e+02  7.361e+01   1.786
## Adventure:tomatoUserReviews             -4.924e-01  2.051e-01  -2.400
## tomatoUserReviews:Documentary            2.545e+00  1.809e+00   1.407
## Animation:tomatoUserReviews              1.134e+00  3.693e-01   3.071
## tomatoUserReviews:`Adam Sandler`        -6.603e-01  2.899e-01  -2.278
## `TV-G`:tomatoUserReviews                        NA         NA      NA
## TwoLanguage:tomatoUserReviews           -1.017e-01  2.266e-01  -0.449
## tomatoUserReviews:China                 -1.755e-01  1.864e+00  -0.094
## Year:Drama                               1.076e-01  1.514e-01   0.711
```

```
## Comedy:Horror                          -1.087e+03  3.814e+02  -2.851
## Horror:Crime                           -8.751e+02  5.777e+02  -1.515
## Horror:Mystery                          6.934e+02  3.183e+02   2.178
## Horror:Thriller                         3.073e+02  2.834e+02   1.084
## Horror:USA                              2.733e+02  3.970e+02   0.689
## Horror:`Paramount Pictures`             2.665e+03  8.456e+02   3.151
## Horror:`New Line Cinema`                6.195e+02  7.003e+02   0.885
## `Sci-Fi`:Comedy                        -9.368e+02  4.857e+02  -1.929
## `Sci-Fi`:Drama                         -4.232e+02  3.745e+02  -1.130
## `Sci-Fi`:TwoOrThreeNomi                -7.473e+02  5.604e+02  -1.334
## `Sci-Fi`:Fourto7Nomi                   -9.193e+02  4.049e+02  -2.270
## `Sci-Fi`:ThreeLanguage                 -1.813e+03  7.094e+02  -2.555
## `Sci-Fi`:UK                            -4.937e+02  4.336e+02  -1.139
## `Sci-Fi`:China                          1.345e+03  1.642e+03   0.819
## `Sci-Fi`:`Miramax Films`               -4.409e+03  2.144e+03  -2.057
## `Sci-Fi`:`Warner Bros.`                -3.179e+01  5.262e+02  -0.060
## Adventure:Thriller                      5.922e+02  4.919e+02   1.204
## Adventure:`Mark Wahlberg`                      NA         NA      NA
## Adventure:`Robert Rodriguez`            3.995e+03  1.209e+03   3.305
## Adventure:OneLanguage                   2.778e+02  2.112e+02   1.316
## Adventure:`New Zealand`                 2.692e+02  7.362e+02   0.366
## Adventure:Lionsgate                     1.456e+03  5.996e+02   2.429
## Comedy:Family                           5.037e+02  3.644e+02   1.382
## Comedy:Documentary                      2.780e+03  7.821e+02   3.555
## Comedy:Animation                       -1.773e+02  4.290e+02  -0.413
## Comedy:Fourto7Nomi                      2.620e+02  1.808e+02   1.449
## Comedy:Eightto20Nomi                    6.391e+02  2.044e+02   3.126
## Comedy:G                                1.070e+02  1.986e+02   0.539
## Comedy:TwoLanguage                      9.907e+01  1.899e+02   0.522
## Comedy:`Warner Bros. Pictures`          4.895e+02  3.144e+02   1.557
## Comedy:`Focus Features`                -6.221e+02  4.412e+02  -1.410
## Family:Musical                          2.836e+03  2.664e+03   1.064
## Family:TwoOrThreeNomi                   8.852e+02  3.927e+02   2.254
## UK:Family                               1.266e+03  5.126e+02   2.469
## Family:Canada                           2.183e+03  5.759e+02   3.791
## Family:Spain                           -3.337e+03  1.560e+03  -2.140
## Family:`Warner Bros. Pictures`          7.152e+02  5.050e+02   1.416
## TwoOrThreeWins:Crime                   -5.722e+02  2.747e+02  -2.083
## UK:Crime                               -4.996e+02  2.750e+02  -1.817
## Crime:France                           -6.448e+02  3.230e+02  -1.996
## France:Music                            1.600e+03  7.247e+02   2.208
## `Summit Entertainment`:Music            5.708e+03  2.031e+03   2.811
## Drama:Mystery                          -1.528e+02  2.260e+02  -0.676
## Drama:Documentary                       1.523e+03  8.914e+02   1.709
## Drama:`Adam Sandler`                   -2.206e+03  9.440e+02  -2.337
## Drama:TwoOrThreeWins                   -2.816e+02  1.794e+02  -1.569
## Drama:TwoOrThreeNomi                    9.681e+00  1.908e+02   0.051
## Drama:R                                -2.769e+02  1.784e+02  -1.552
## Drama:OneLanguage                      -1.403e+02  1.350e+02  -1.039
## Drama:USA                              -2.155e+01  2.370e+02  -0.091
## Mystery:Fantasy                        -1.495e+03  5.947e+02  -2.514
## Animation:Mystery                      -3.614e+03  2.349e+03  -1.538
## Mystery:`Steven Soderbergh`            -4.602e+03  1.495e+03  -3.078
## Mystery:`Czech Republic`               -2.497e+03  1.495e+03  -1.670
```

```
## PG:Thriller                              2.678e+02  2.272e+02   1.179
## `PG-13`:Thriller                                NA         NA      NA
## Japan:Thriller                           1.967e+03  9.190e+02   2.140
## China:Thriller                           2.204e+03  1.044e+03   2.112
## `Paramount Pictures`:Thriller            8.505e+02  5.130e+02   1.658
## Thriller:MGM                             1.280e+03  7.123e+02   1.796
## `Summit Entertainment`:Thriller         -2.354e+03  7.661e+02  -3.072
## Sport:Lionsgate                         -2.399e+03  2.380e+03  -1.008
## NoWin:Fantasy                            1.841e+02  3.220e+02   0.572
## Fantasy:TwoOrThreeNomi                  -3.813e+02  4.202e+02  -0.907
## Fantasy:Canada                          -8.493e+02  4.321e+02  -1.965
## `Walt Disney Pictures`:Fantasy           5.313e+02  9.167e+02   0.580
## Fantasy:`Summit Entertainment`           4.674e+03  9.661e+02   4.839
## Romance:`Robert De Niro`                 1.721e+03  1.292e+03   1.332
## OneWin:Action                           -1.082e+02  2.519e+02  -0.429
## Action:Fourto10Wins                      5.169e+02  2.879e+02   1.795
## Action:`PG-13`                          -3.040e+02  1.947e+02  -1.562
## UK:Action                               -4.170e+02  2.607e+02  -1.600
## Action:China                             1.552e+03  8.589e+02   1.806
## Action:`Warner Bros. Pictures`          -7.533e+02  3.314e+02  -2.273
## Action:Lionsgate                        -8.260e+02  4.083e+02  -2.023
## OneWin:Biography                        -5.994e+02  4.509e+02  -1.329
## Biography:`The Weinstein Company`        5.939e+02  6.892e+02   0.862
## `The Weinstein Company`:History          5.548e+03  1.596e+03   3.475
## Animation:`Ben Stiller`                  2.654e+03  1.593e+03   1.666
## Animation:`Jack Black`                   2.319e+03  1.104e+03   2.101
## Animation:`Matt Damon`                  -1.651e+03  2.141e+03  -0.771
## Animation:Eightto20Nomi                 -3.790e+01  4.665e+02  -0.081
## Animation:R                             -2.117e+03  1.066e+03  -1.986
## Animation:`PG-13`                       -4.683e+03  8.497e+02  -5.512
## TwoLanguage:Animation                    1.333e+03  5.063e+02   2.634
## Animation:`Universal Pictures`           1.525e+03  8.000e+02   1.906
## Animation:`20th Century Fox`             1.910e+03  5.311e+02   3.596
## Animation:`Summit Entertainment`        -5.523e+03  2.139e+03  -2.582
## Eightto20Nomi:Musical                   -2.954e+03  9.983e+02  -2.959
## TwoLanguage:Musical                      2.101e+03  1.151e+03   1.825
## Musical:Germany                          2.919e+03  1.648e+03   1.771
## Western:Eightto20Nomi                   -1.074e+03  9.344e+02  -1.149
## Western:`Walt Disney Pictures`          -6.263e+03  2.522e+03  -2.483
## War:`Miramax Films`                     -3.498e+03  1.577e+03  -2.218
## `Ben Stiller`:`Robert De Niro`           3.215e+02  1.568e+03   0.205
## `Universal Pictures`:`Robert De Niro`           NA         NA      NA
## `Lionsgate Films`:`Robert De Niro`       3.066e+03  1.440e+03   2.129
## Lionsgate:`Robert De Niro`                      NA         NA      NA
## `Ben Stiller`:TwoOrThreeWins             1.466e+02  9.273e+02   0.158
## `Ben Stiller`:Germany                   -2.435e+03  1.234e+03  -1.974
## `New Zealand`:`Mark Wahlberg`           -2.563e+03  2.730e+03  -0.939
## `Universal Pictures`:`Mark Wahlberg`     3.211e+03  1.137e+03   2.824
## R:`Nicolas Cage`                        -8.607e+02  6.652e+02  -1.294
## `Steven Soderbergh`:Fourto10Wins         4.587e+03  1.618e+03   2.835
## TwoLanguage:`Clint Eastwood`             4.008e+02  1.134e+03   0.353
## MGM:`Ridley Scott`                       6.711e+03  2.256e+03   2.975
## `PG-13`:`Shawn Levy`                    -2.698e+03  1.208e+03  -2.234
## UK:`Shawn Levy`                          2.958e+03  1.591e+03   1.859
```

```
## `Steven Spielberg`:`Paramount Pictures`  3.119e+03  1.655e+03   1.884
## `Robert Rodriguez`:tomatoreviewrank             NA         NA      NA
## NoWin:OneNomi                           -2.037e+02  1.875e+02  -1.086
## NoWin:Eightto20Nomi                      1.762e+02  3.725e+02   0.473
## NoWin:Germany                           -6.006e+02  2.487e+02  -2.415
## NoWin:`Sony Pictures`                    5.432e+02  2.981e+02   1.822
## OneWin:India                            -4.946e+03  2.115e+03  -2.338
## Fourto10Wins:Germany                    -9.954e+02  3.304e+02  -3.013
## Canada:Fourto10Wins                     -1.736e+03  4.017e+02  -4.323
## Fourto10Wins:India                       2.162e+03  1.088e+03   1.987
## Fourto10Wins:`United Arab Emirates`      2.571e+03  1.471e+03   1.748
## `Universal Pictures`:Fourto10Wins        1.422e+03  4.563e+02   3.116
## `Summit Entertainment`:Fourto10Wins     -2.852e+03  1.233e+03  -2.313
## TwoOrThreeNomi:Germany                  -3.084e+02  3.260e+02  -0.946
## TwoOrThreeNomi:Australia                 1.416e+03  6.473e+02   2.188
## TwoOrThreeNomi:Switzerland              -1.062e+03  1.244e+03  -0.853
## TwoOrThreeNomi:tomatoreviewrank         -8.921e+03  2.260e+03  -3.947
## UK:Fourto7Nomi                          -4.216e+02  2.614e+02  -1.613
## Fourto7Nomi:`Paramount Pictures`         1.118e+03  4.718e+02   2.370
## Eightto20Nomi:tomatoreviewrank          -3.414e+03  1.005e+03  -3.396
## R:`Paramount Pictures`                   5.258e+02  3.779e+02   1.391
## R:`Warner Bros.`                         8.593e+02  2.857e+02   3.007
## `PG-13`:Australia                       -8.565e+02  4.462e+02  -1.920
## `Warner Bros. Pictures`:`PG-13`         -5.500e+02  3.066e+02  -1.794
## OneLanguage:Switzerland                 -2.171e+03  8.893e+02  -2.441
## TwoLanguage:`Paramount Pictures`         7.525e+02  4.408e+02   1.707
## ThreeLanguage:`Miramax Films`            8.554e+02  1.119e+03   0.765
## ThreeLanguage:`Warner Bros.`            -1.541e+01  5.021e+02  -0.031
## UK:Italy                                -6.048e+02  4.714e+02  -1.283
## UK:`Paramount Pictures`                 -8.137e+02  5.430e+02  -1.499
## UK:`Walt Disney Pictures`                2.161e+03  9.251e+02   2.336
## `Walt Disney Pictures`:Germany           1.152e+04  2.271e+03   5.072
## Germany:`Columbia Pictures`              2.685e+03  9.383e+02   2.861
## Japan:Canada                            -2.551e+03  1.013e+03  -2.519
## Canada:`Warner Bros. Pictures`          -9.635e+02  6.494e+02  -1.484
## Canada:`Focus Features`                  2.767e+03  1.052e+03   2.630
## `Universal Pictures`:France              8.687e+02  4.335e+02   2.004
## `Miramax Films`:France                   4.199e+03  1.070e+03   3.923
## Japan:`Summit Entertainment`                    NA         NA      NA
## `Hong Kong`:`Warner Bros. Pictures`     -2.665e+03  1.265e+03  -2.107
## `Sony Pictures`:`Hong Kong`              2.224e+03  1.076e+03   2.067
## `Hong Kong`:`Sony Pictures Classics`            NA         NA      NA
## `Hong Kong`:`Summit Entertainment`              NA         NA      NA
## `New Line Cinema`:`New Zealand`          5.626e+03  1.712e+03   3.285
## `Sony Pictures`:`Columbia Pictures`      2.973e+03  2.077e+03   1.431
##                                         Pr(>|t|)
## (Intercept)                             0.034660 *
## imdbVotes                               8.57e-06 ***
## imdbRating7                             0.769104
## Year                                    0.026358 *
## `Sci-Fi`                                0.974708
## War                                     0.745407
## `TV-G`                                  0.816354
## UK                                      0.986972
```

```
## Budget:tomatoFresh                     0.283531
## imdbVotes:Budget                       6.68e-06 ***
## Budget:tomatoUserRating                1.92e-14 ***
## Budget:Adventure                       0.846905
## Budget:Comedy                          0.726786
## Budget:Drama                           0.165790
## Budget:Sport                           0.825742
## Budget:Western                         0.758699
## Budget:Short                           0.840242
## Budget:NoWin                           6.93e-05 ***
## Budget:NoNomi                          0.017118 *
## Budget:TwoLanguage                     0.244247
## Budget:`Sony Pictures`                 0.258144
## Budget:`Magnolia Pictures`             0.521628
## imdbVotes:tomatoRotten                 0.021952 *
## tomatoRotten:imdbRating1               1.59e-05 ***
## imdbRating7:tomatoRotten               0.953348
## tomatoRotten:imdbRating10              0.898796
## tomatoUserRating:tomatoRotten          0.057528 .
## tomatoRotten:Horror                    0.417313
## tomatoRotten:Family                    0.178737
## tomatoRotten:Animation                 0.002898 **
## Short:tomatoRotten                     0.700157
## `TV-G`:tomatoRotten                    0.557337
## tomatoFresh:tomatoUserRating           0.459365
## tomatoFresh:Adventure                  0.546379
## tomatoFresh:Family                     0.196177
## tomatoFresh:OneWin                     0.447437
## `TV-G`:tomatoFresh                           NA
## imdbVotes:imdbRating2                  2.73e-10 ***
## imdbVotes:imdbRating3                  6.66e-06 ***
## imdbVotes:imdbRating4                  0.000810 ***
## imdbVotes:imdbRating5                  0.015129 *
## imdbVotes:imdbRating10                 1.71e-05 ***
## imdbVotes:Year                         6.34e-06 ***
## imdbVotes:Comedy                       0.992916
## imdbVotes:Animation                    0.010158 *
## imdbVotes:`Clint Eastwood`             0.032314 *
## imdbVotes:PG                           0.669626
## imdbVotes:G                            0.095369 .
## imdbVotes:`TV-G`                             NA
## imdbVotes:USA                          0.760545
## imdbVotes:`Universal Pictures`         0.016883 *
## imdbVotes:`20th Century Fox`           3.13e-06 ***
## imdbVotes:`Magnolia Pictures`          0.566148
## imdbVotes:`Fox Searchlight Pictures`   0.058303 .
## Runtime1:tomatoUserReviews             0.794190
## Adventure:Runtime1                     0.664778
## Family:Runtime1                        0.386103
## Sport:Runtime1                         0.161982
## Runtime1:Japan                         0.908866
## `20th Century Fox`:Runtime1            0.323923
## Horror:Runtime2                        0.235866
## Runtime2:Mystery                       0.274621
```

```
## Runtime2:Romance                        0.040831 *
## Runtime2:Action                         0.271654
## TwoLanguage:Runtime2                    0.355910
## Runtime2:`Hong Kong`                    0.161291
## `Universal Pictures`:Runtime2           0.000453 ***
## Runtime2:`Walt Disney Pictures`         0.042873 *
## Runtime3:Fantasy                        0.250062
## Runtime3:`Ben Stiller`                  0.074433 .
## Runtime3:`Matt Damon`                   0.032019 *
## imdbRating4:Runtime4                    0.434609
## Runtime4:`New Line Cinema`              0.035857 *
## Fantasy:Runtime5                        0.268477
## Action:Runtime6                         0.003255 **
## Runtime6:R                              0.139456
## imdbRating1:Runtime7                    0.003500 **
## Runtime7:TwoOrThreeNomi                 0.027408 *
## Runtime7:Eightto20Nomi                  0.942530
## Runtime7:Canada                         0.221827
## Japan:Runtime7                          0.007090 **
## Runtime7:`Warner Bros. Pictures`        0.351662
## `Walt Disney Pictures`:Runtime7         0.000474 ***
## Runtime7:`Lionsgate Films`              0.042992 *
## `Sci-Fi`:Runtime8                       0.340897
## Adventure:Runtime8                      0.000685 ***
## Adventure:Runtime10                     0.038875 *
## Family:Runtime10                        0.049289 *
## Drama:Runtime10                         0.302656
## Fantasy:Runtime10                       0.082078 .
## Runtime10:`Steven Soderbergh`           6.50e-05 ***
## Runtime10:TwoOrThreeWins                0.137807
## Eightto20Nomi:Runtime10                 0.022812 *
## Runtime10:ThreeLanguage                 0.062274 .
## Japan:Runtime10                         0.000865 ***
## Comedy:imdbRating1                      0.216308
## imdbRating1:`Steven Spielberg`          0.000688 ***
## imdbRating1:`Ron Howard`                0.032412 *
## imdbRating1:TwoOrThreeWins              0.294386
## imdbRating1:G                           0.560457
## imdbRating1:`New Zealand`               0.191560
## imdbRating1:`Walt Disney Pictures`      0.050913 .
## imdbRating1:`Fox Searchlight`           0.991266
## imdbRating1:`Summit Entertainment`      0.059115 .
## imdbRating2:Fantasy                     0.040100 *
## imdbRating2:Eightto20Nomi               0.435658
## imdbRating3:Musical                     0.124725
## imdbRating3:`Mark Wahlberg`             0.034913 *
## imdbRating3:TwoOrThreeWins              0.096742 .
## imdbRating3:Fourto7Nomi                 0.654782
## imdbRating3:China                       0.907305
## imdbRating3:`Paramount Pictures`        0.086282 .
## `Sony Pictures`:imdbRating3             0.061502 .
## imdbRating4:tomatoUserReviews           0.733831
## imdbRating4:Fourto10Wins                0.532807
## imdbRating4:`Paramount Pictures`        0.795706
```

```
## Animation:imdbRating5                    0.003424 **
## imdbRating5:Fourto10Wins                 0.078088 .
## imdbRating5:OneNomi                      0.044719 *
## imdbRating5:Eightto20Nomi                0.488442
## imdbRating5:`Universal Pictures`         0.394072
## Adventure:imdbRating6                    0.494344
## Action:imdbRating6                       0.913590
## imdbRating6:Biography                    0.084986 .
## imdbRating6:`Adam Sandler`               0.000124 ***
## `Mark Wahlberg`:imdbRating6              0.146843
## `Ron Howard`:imdbRating6                 0.000358 ***
## OneWin:imdbRating6                       0.068052 .
## TwoOrThreeNomi:imdbRating6               0.339336
## imdbRating7:Year                         0.763271
## imdbRating7:Horror                       0.040733 *
## imdbRating7:NoWin                        0.176984
## imdbRating7:OneWin                       0.056859 .
## imdbRating7:`PG-13`                      0.004510 **
## imdbRating7:USA                          0.262867
## Romance:imdbRating8                      0.386480
## OneWin:imdbRating8                       0.282460
## OneNomi:imdbRating8                      0.071181 .
## Eightto20Nomi:imdbRating8                0.064697 .
## `PG-13`:imdbRating8                      0.229234
## imdbRating10:tomatoUserReviews           0.215625
## `Sci-Fi`:imdbRating10                    0.000815 ***
## imdbRating10:Family                      0.000642 ***
## imdbRating10:Mystery                     0.079782 .
## imdbRating10:Thriller                    0.657526
## imdbRating10:`Steven Spielberg`                NA
## imdbRating10:`Robert Rodriguez`          0.043666 *
## imdbRating10:`Ron Howard`                      NA
## imdbRating10:OneWin                      0.080023 .
## imdbRating10:Fourto10Wins                0.000315 ***
## imdbRating10:TwoOrThreeNomi              0.011561 *
## imdbRating10:`20th Century Fox`          0.014543 *
## imdbRating10:Lionsgate                   0.000555 ***
## imdbRating10:`Fox Searchlight`           0.136009
## NoNomi:tomatoRating                      0.674098
## tomatoUserRating:tomatoUserReviews       5.62e-13 ***
## Year:tomatoUserRating                    1.16e-05 ***
## War:tomatoUserRating                     0.914517
## tomatoUserRating:G                       0.074278 .
## Adventure:tomatoUserReviews              0.016448 *
## tomatoUserReviews:Documentary            0.159474
## Animation:tomatoUserReviews              0.002159 **
## tomatoUserReviews:`Adam Sandler`         0.022837 *
## `TV-G`:tomatoUserReviews                       NA
## TwoLanguage:tomatoUserReviews            0.653478
## tomatoUserReviews:China                  0.925026
## Year:Drama                               0.477311
## Comedy:Horror                            0.004395 **
## Horror:Crime                             0.129985
## Horror:Mystery                           0.029476 *
```

```
## Horror:Thriller                         0.278307
## Horror:USA                              0.491193
## Horror:`Paramount Pictures`             0.001645 **
## Horror:`New Line Cinema`                0.376491
## `Sci-Fi`:Comedy                         0.053885 .
## `Sci-Fi`:Drama                          0.258595
## `Sci-Fi`:TwoOrThreeNomi                 0.182473
## `Sci-Fi`:Fourto7Nomi                    0.023275 *
## `Sci-Fi`:ThreeLanguage                  0.010669 *
## `Sci-Fi`:UK                             0.254953
## `Sci-Fi`:China                          0.412945
## `Sci-Fi`:`Miramax Films`                0.039813 *
## `Sci-Fi`:`Warner Bros.`                 0.951834
## Adventure:Thriller                      0.228748
## Adventure:`Mark Wahlberg`                     NA
## Adventure:`Robert Rodriguez`            0.000964 ***
## Adventure:OneLanguage                   0.188449
## Adventure:`New Zealand`                 0.714683
## Adventure:Lionsgate                     0.015226 *
## Comedy:Family                           0.167001
## Comedy:Documentary                      0.000385 ***
## Comedy:Animation                        0.679505
## Comedy:Fourto7Nomi                      0.147475
## Comedy:Eightto20Nomi                    0.001792 **
## Comedy:G                                0.589961
## Comedy:TwoLanguage                      0.601982
## Comedy:`Warner Bros. Pictures`          0.119647
## Comedy:`Focus Features`                 0.158671
## Family:Musical                          0.287276
## Family:TwoOrThreeNomi                   0.024281 *
## UK:Family                               0.013613 *
## Family:Canada                           0.000154 ***
## Family:Spain                            0.032490 *
## Family:`Warner Bros. Pictures`          0.156871
## TwoOrThreeWins:Crime                    0.037320 *
## UK:Crime                                0.069371 .
## Crime:France                            0.045998 *
## France:Music                            0.027357 *
## `Summit Entertainment`:Music            0.004982 **
## Drama:Mystery                           0.499178
## Drama:Documentary                       0.087625 .
## Drama:`Adam Sandler`                    0.019540 *
## Drama:TwoOrThreeWins                    0.116692
## Drama:TwoOrThreeNomi                    0.959538
## Drama:R                                 0.120772
## Drama:OneLanguage                       0.298991
## Drama:USA                               0.927550
## Mystery:Fantasy                         0.011987 *
## Animation:Mystery                       0.124085
## Mystery:`Steven Soderbergh`             0.002109 **
## Mystery:`Czech Republic`                0.095000 .
## PG:Thriller                             0.238707
## `PG-13`:Thriller                              NA
## Japan:Thriller                          0.032442 *
```

```
## China:Thriller                          0.034798 *
## `Paramount Pictures`:Thriller           0.097461 .
## Thriller:MGM                            0.072550 .
## `Summit Entertainment`:Thriller         0.002146 **
## Sport:Lionsgate                         0.313542
## NoWin:Fantasy                           0.567550
## Fantasy:TwoOrThreeNomi                  0.364292
## Fantasy:Canada                          0.049484 *
## `Walt Disney Pictures`:Fantasy          0.562244
## Fantasy:`Summit Entertainment`          1.39e-06 ***
## Romance:`Robert De Niro`                0.183043
## OneWin:Action                           0.667625
## Action:Fourto10Wins                     0.072708 .
## Action:`PG-13`                          0.118471
## UK:Action                               0.109807
## Action:China                            0.070973 .
## Action:`Warner Bros. Pictures`          0.023092 *
## Action:Lionsgate                        0.043170 *
## OneWin:Biography                        0.183915
## Biography:`The Weinstein Company`       0.388918
## `The Weinstein Company`:History         0.000519 ***
## Animation:`Ben Stiller`                 0.095900 .
## Animation:`Jack Black`                  0.035764 *
## Animation:`Matt Damon`                  0.440585
## Animation:Eightto20Nomi                 0.935255
## Animation:R                             0.047120 *
## Animation:`PG-13`                       3.92e-08 ***
## TwoLanguage:Animation                   0.008499 **
## Animation:`Universal Pictures`          0.056790 .
## Animation:`20th Century Fox`            0.000330 ***
## Animation:`Summit Entertainment`        0.009872 **
## Eightto20Nomi:Musical                   0.003120 **
## TwoLanguage:Musical                     0.068136 .
## Musical:Germany                         0.076684 .
## Western:Eightto20Nomi                   0.250683
## Western:`Walt Disney Pictures`          0.013090 *
## War:`Miramax Films`                     0.026654 *
## `Ben Stiller`:`Robert De Niro`          0.837571
## `Universal Pictures`:`Robert De Niro`         NA
## `Lionsgate Films`:`Robert De Niro`      0.033354 *
## Lionsgate:`Robert De Niro`                    NA
## `Ben Stiller`:TwoOrThreeWins            0.874388
## `Ben Stiller`:Germany                   0.048539 *
## `New Zealand`:`Mark Wahlberg`           0.347800
## `Universal Pictures`:`Mark Wahlberg`    0.004776 **
## R:`Nicolas Cage`                        0.195836
## `Steven Soderbergh`:Fourto10Wins        0.004624 **
## TwoLanguage:`Clint Eastwood`            0.723823
## MGM:`Ridley Scott`                      0.002961 **
## `PG-13`:`Shawn Levy`                    0.025551 *
## UK:`Shawn Levy`                         0.063097 .
## `Steven Spielberg`:`Paramount Pictures` 0.059643 .
## `Robert Rodriguez`:tomatoreviewrank           NA
## NoWin:OneNomi                           0.277399
```

```
## NoWin:Eightto20Nomi                     0.636245
## NoWin:Germany                           0.015818 *
## NoWin:`Sony Pictures`                   0.068522 .
## OneWin:India                            0.019475 *
## Fourto10Wins:Germany                    0.002617 **
## Canada:Fourto10Wins                     1.60e-05 ***
## Fourto10Wins:India                      0.047077 *
## Fourto10Wins:`United Arab Emirates`     0.080609 .
## `Universal Pictures`:Fourto10Wins       0.001857 **
## `Summit Entertainment`:Fourto10Wins     0.020817 *
## TwoOrThreeNomi:Germany                  0.344208
## TwoOrThreeNomi:Australia                0.028783 *
## TwoOrThreeNomi:Switzerland              0.393651
## TwoOrThreeNomi:tomatoreviewrank         8.13e-05 ***
## UK:Fourto7Nomi                          0.106943
## Fourto7Nomi:`Paramount Pictures`        0.017876 *
## Eightto20Nomi:tomatoreviewrank          0.000694 ***
## R:`Paramount Pictures`                  0.164289
## R:`Warner Bros.`                        0.002661 **
## `PG-13`:Australia                       0.055013 .
## `Warner Bros. Pictures`:`PG-13`         0.072994 .
## OneLanguage:Switzerland                 0.014713 *
## TwoLanguage:`Paramount Pictures`        0.087921 .
## ThreeLanguage:`Miramax Films`           0.444605
## ThreeLanguage:`Warner Bros.`            0.975526
## UK:Italy                                0.199603
## UK:`Paramount Pictures`                 0.134110
## UK:`Walt Disney Pictures`               0.019582 *
## `Walt Disney Pictures`:Germany          4.22e-07 ***
## Germany:`Columbia Pictures`             0.004252 **
## Japan:Canada                            0.011821 *
## Canada:`Warner Bros. Pictures`          0.138025
## Canada:`Focus Features`                 0.008584 **
## `Universal Pictures`:France             0.045194 *
## `Miramax Films`:France                  8.96e-05 ***
## Japan:`Summit Entertainment`                  NA
## `Hong Kong`:`Warner Bros. Pictures`     0.035183 *
## `Sony Pictures`:`Hong Kong`             0.038813 *
## `Hong Kong`:`Sony Pictures Classics`          NA
## `Hong Kong`:`Summit Entertainment`            NA
## `New Line Cinema`:`New Zealand`         0.001033 **
## `Sony Pictures`:`Columbia Pictures`     0.152433
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2014 on 2492 degrees of freedom
## Multiple R-squared:  0.9197, Adjusted R-squared:  0.9098
## F-statistic: 93.21 on 306 and 2492 DF,  p-value: < 2.2e-16
```

And also I see increased Adjusted R-squared from model 4 to model 5 based on full data. See below.

```
s4=summary(lm(Gross~.,df4))
#s4$adj.r.squared
print(s4$adj.r.squared)
```

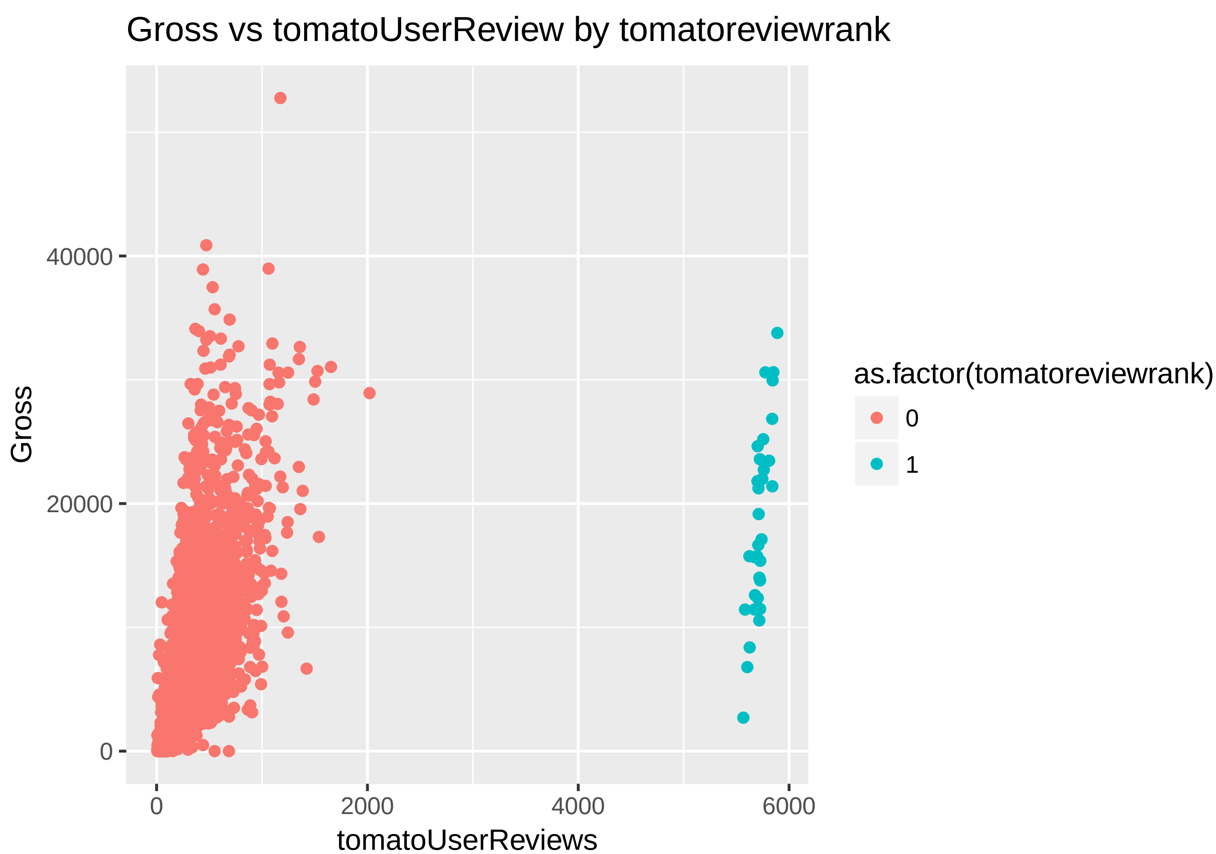
```
## [1] 0.846403
```

```r
#s5$adj.r.squared
print(s5$adj.r.squared)
```

```
## [1] 0.9097837
```

Second, it makes sense to include them in the model. Such as tomatoUserReviews:tomatoreviewrank. See plot below. We know there are two clusters of points, and for each cluster of data, the slope and intercept should be different. After including tomatoreviewrank and tomatoUserReviews:tomatoreviewrank, the intercept and slope for two clusters are adjusted accordingly.

```r
#based on plot below, I included interation term of tomatoUserReviews:tomatoreviewrank
ggplot(data=df5,aes(tomatoUserReviews,Gross,color=as.factor(tomatoreviewrank)))+
  geom_point()+ggtitle("Gross vs tomatoUserReview by tomatoreviewrank")
```




After Task 5, train RMSEs drop to around 56 M and test RMSEs drop to around 66 M.

```r
#print average train RMSEs
average_rmse5$rmse[average_rmse5$type=="train"]
```

```
##  [1] 3.248842e-04 1.310440e+07 2.588711e+07 3.232058e+07 3.768558e+07
##  [6] 4.109709e+07 4.379779e+07 4.668111e+07 4.813049e+07 5.001485e+07
## [11] 5.117392e+07 5.202196e+07 5.331270e+07 5.447757e+07 5.493967e+07
## [16] 5.506902e+07 5.525985e+07 5.573241e+07 5.596594e+07
```

```r
#print average test RMSEs
average_rmse5$rmse[average_rmse5$type=="test"]
```

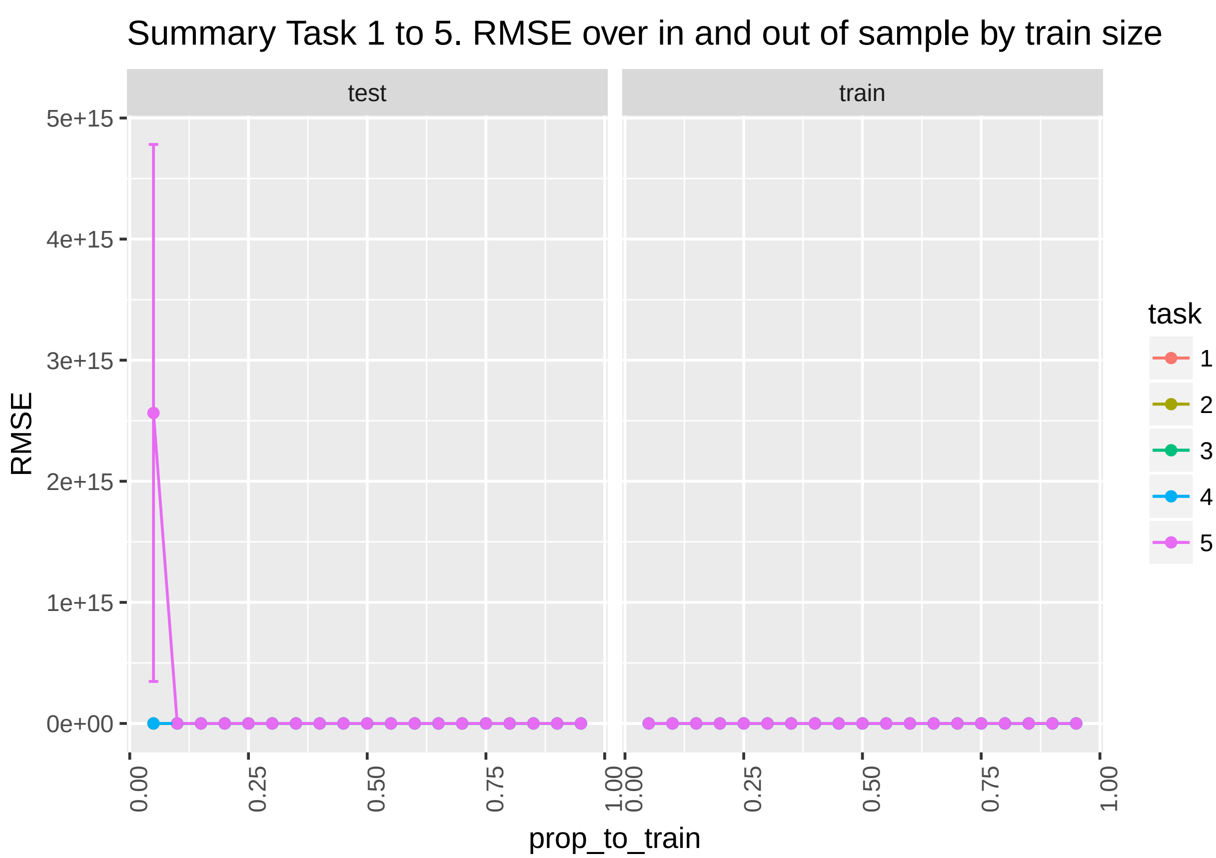
```
##  [1] 2.564238e+15 8.404956e+09 2.099764e+08 1.116889e+08 9.717416e+07
```

```
##  [6] 9.245968e+07 9.110848e+07 8.861296e+07 8.447911e+07 8.035033e+07
## [11] 7.844175e+07 7.447322e+07 7.198069e+07 6.880627e+07 6.811030e+07
## [16] 6.958428e+07 6.718153e+07 6.658831e+07 6.610946e+07
```

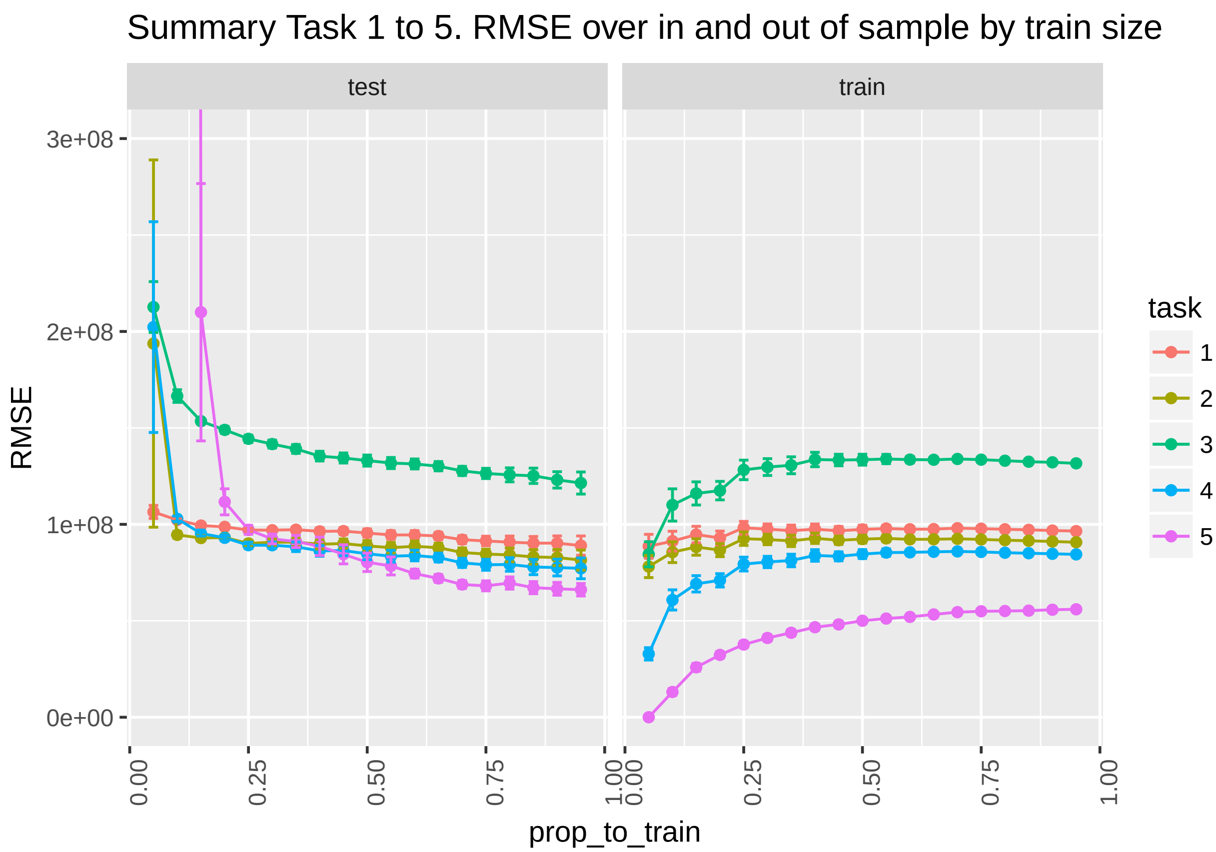
## summary RMSE evaluation results from task 1 to task 5

We see the trend of drop of RMSEs from task 1 to task 2, from task 2/task 3 to task 4, and from task 4 to task 5.

```r
average_rmse$task="1"
average_rmse2$task="2"
average_rmse3$task="3"
average_rmse4$task="4"
average_rmse5$task="5"
average_rmse_all=rbind(average_rmse,average_rmse2,average_rmse3,average_rmse4,average_rmse5)

summaryplot1=ggplot(average_rmse_all,aes(prop_to_train,rmse,group=task,col=task))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin=rmse-se,ymax=rmse+se),width=0.02)+
  ylab("RMSE")+
  theme(axis.text.x = element_text(angle=90, hjust=1))+
  facet_grid(.~type)+
  ggtitle("Summary Task 1 to 5. RMSE over in and out of sample by train size")
summaryplot1
```

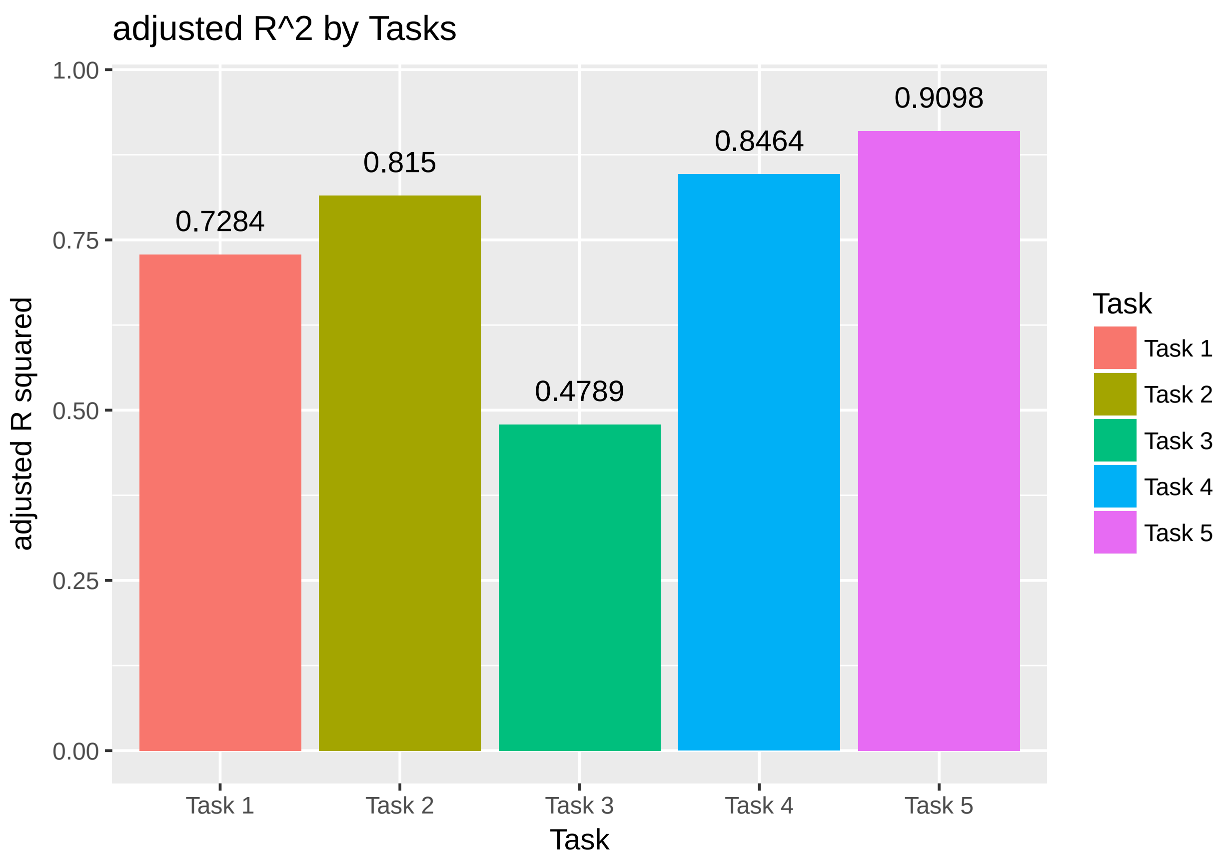


```r
#let's zoom in the summary plot
summaryplot1+coord_cartesian(ylim= c(0,300000000))
```




Here is the summary of the adjusted R squared from task 1 to task 5. We see improvement from task 1 to task 2, from task 2/task 3 to task 4, from task 4 to task 5.

```r
s1=summary(lm(Gross~.,dfn))
s2=summary(lm(Gross~.,dfn2))
s3=summary(lm(Gross~.,df3_m))
R2adj=c(s1$adj.r.squared,s2$adj.r.squared,s3$adj.r.squared,s4$adj.r.squared,s5$adj.r.squared)
Tasks=c("Task 1","Task 2","Task 3","Task 4", "Task 5")
radj_df=data.frame(R2adj=R2adj,Task=Tasks)
ggplot(radj_df,aes(Task,R2adj,fill=Task))+geom_bar(stat="identity")+
  ylab("adjusted R squared")+
  geom_text(aes(label = round(R2adj,digits = 4),y=R2adj+0.05), size = 4)+
  ggtitle("adjusted R^2 by Tasks")
```



adjusted R^2 by Tasks
adjusted R squared
1.00
0.75
0.50
0.25
0.00
0.7284
0.815
0.4789
0.8464
0.9098
Task 1
Task 2
Task 3
Task 4
Task 5
Task
Task
Task 1
Task 2
Task 3
Task 4
Task 5