Project 2: Modeling and Evaluation

(Student GT acount name: xtao41)

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)
install.packages("tidyr",repos="http://cran.us.r-project.org")

## package 'tidyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\xin_t\AppData\Local\Temp\RtmpkrXSMy\downloaded_packages
```

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: Used package "tidyr" for generate frequency table.

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
df<-df[(df$Type=="movie"),]
```

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
df = subset(df, !is.na(df$Gross))
```

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).

```
# TODO: Exclude movies released prior to 2000
df = subset(df, df$Year >= 2000)
```

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.

```
# TODO: Remove mismatched rows
df_match_year<-df
df_match_year$Released<-as.numeric(format(df$Released,'%Y'))
df_match_year<-subset(df_match_year,is.na(df_match_year$Released)|
(df_match_year$Released>=df_match_year$Year))
```

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.

```
# TODO: Exclude the `Domestic_Gross` column
df_col_remove=subset(df_match_year,select=-c(Domestic_Gross))
```

6. Process `Runtime` column

```
# TODO: Replace df$Runtime with a numeric column containing the runtime in
minutes
convert_time<-function(runtime){
if(is.na(runtime)){return(NA)}
if(!grepl("h",runtime)){
n_runtime<-as.numeric(runtime)
}
else{
a=(strsplit(runtime, " h "))[[1]][1]
```

```
b=(strsplit(runtime, " h "))[[1]][2]
a<-as.numeric(a)
b<-as.numeric(b)
n_runtime<-a*60+b
}
return(n_runtime)
}
df_col_remove$Runtime<-gsub("min","", df_col_remove$Runtime)
df_col_remove$Runtime<-sapply(df_col_remove$Runtime,function(x)
convert_time(x))
```

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
#remove repeated columns that gives similar/same information
df_final<-subset(df_col_remove,select = -c(Date,Released))

#remove coumns that has no relationship and meaning regarding with Gross
df_final<-subset(df_final,select = -c(Poster,Plot,Type,tomatoConsensus,
tomatoURL,Website, Response, imdbID, BoxOffice))

#convert Metascore to numeric
df_final$Metascore<- as.numeric(as.character(df_final$Metascore))

df_final=na.omit(df_final)
```

***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 (dim(df_final))

## [1] 2759   27

print (colnames(df_final))

##  [1] "Title"            "Year"             "Rated"
##  [4] "Runtime"          "Genre"            "Director"
##  [7] "Writer"           "Actors"           "Language"
## [10] "Country"          "Awards"           "Metascore"
## [13] "imdbRating"       "imdbVotes"        "tomatoMeter"
## [16] "tomatoImage"      "tomatoRating"     "tomatoReviews"
```

```
## [19] "tomatoFresh"       "tomatoRotten"      "tomatoUserMeter"
## [22] "tomatoUserRating"  "tomatoUserReviews" "DVD"
## [25] "Production"        "Budget"            "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

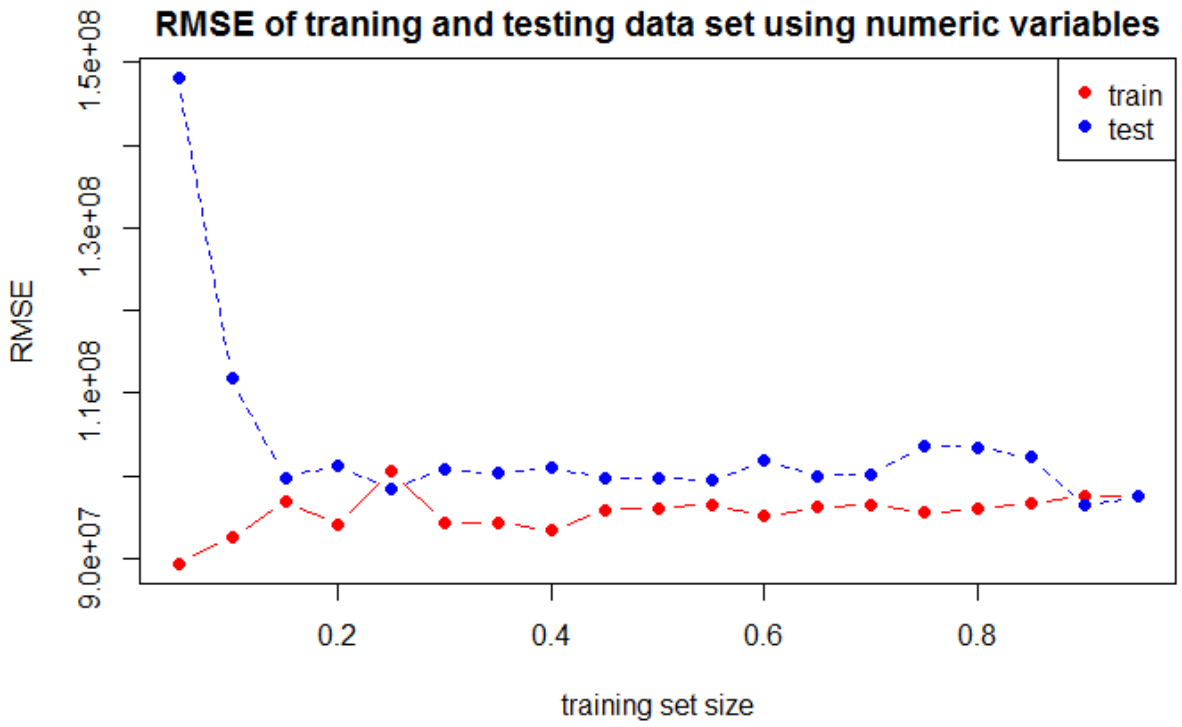
Each of the following tasks is worth 20 points. Remember to build each model as specified, evaluate it using the strategy outlined above, and plot the training and test errors by training set size (%).

1. Numeric variables

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

```
# TODO: Build & evaluate model 1 (numeric variables only)


RMSE=function(formula,df){
  RMSE_train<-vector(length = 19,mode = "numeric")
  RMSE_test<-vector(length = 19,mode = "numeric")
  for(j in 1:19){
    split=j*0.05
    sum_train<-0
    sum_test<-0
    for (i in 1:10){
      #allow random selection for train data
      traindata<-sample(1:nrow(df),round(split*nrow(df)))
      df_train<-df[traindata,]
      df_test<-df[-traindata,]
```

```
      M<-lm(formula = formula, df_train )

      rmse_TR<-(mean(M$residuals^2))^0.5
      rmse_TE<-(mean((df_test$Gross - predict(M, newdata = df_test))^2))^0.5
      sum_train<-sum_train+rmse_TR
      sum_test<-sum_test+rmse_TE
    }
      RMSE_train[j]=sum_train/10
      RMSE_test[j]=sum_test/10
  }
  return(list(RMSE_train, RMSE_test))
}

feature_name=c("Year","Runtime","imdbRating","imdbVotes","tomatoMeter","tomat
oReviews","tomatoFresh","tomatoRotten","tomatoUserMeter","tomatoUserReviews",
"Budget","Metascore","tomatoRating")
formula<- as.formula(paste("Gross ~ ", paste(feature_name, collapse= "+")))
rmse=RMSE(formula,df_final)
RMSE_train=unlist(rmse[1])
RMSE_test=unlist(rmse[2])

size=seq(0.05, 0.95, length.out = 19)
matplot(size,cbind(RMSE_train, RMSE_test), type="b",pch=19, col=c("red",
"blue"), xlab="training set size", ylab="RMSE")
legend("topright", legend=c("train", "test"), pch=19, col=c("red", "blue"))
title ("RMSE of traning and testing data set using numeric variables")
```



Q: List all the numeric variables you used.

A: The numeric variables I used are:
"Year","Runtime","imdbRating","imdbVotes","tomatoMeter","tomatoRating","tomatoReviews","tomatoFresh","tomatoRotten","tomatoUserMeter","tomatoUserReviews","Budget"

The figure above shows the train and test RMSE as a function of train set size for numeric variables, with each point averaged over 10 random runs. It looks like RMSE initially increases for training data and decreases for testing data, indicating a overfitting for small training set size.The reason for training RMSE increases is becauase more training data results a higher variance. On the other hand, larger training set generates a more accurate model and thus test RMSE decreases. as the trainign set size is above 40%, both train and test become stable. The results from this model yields:

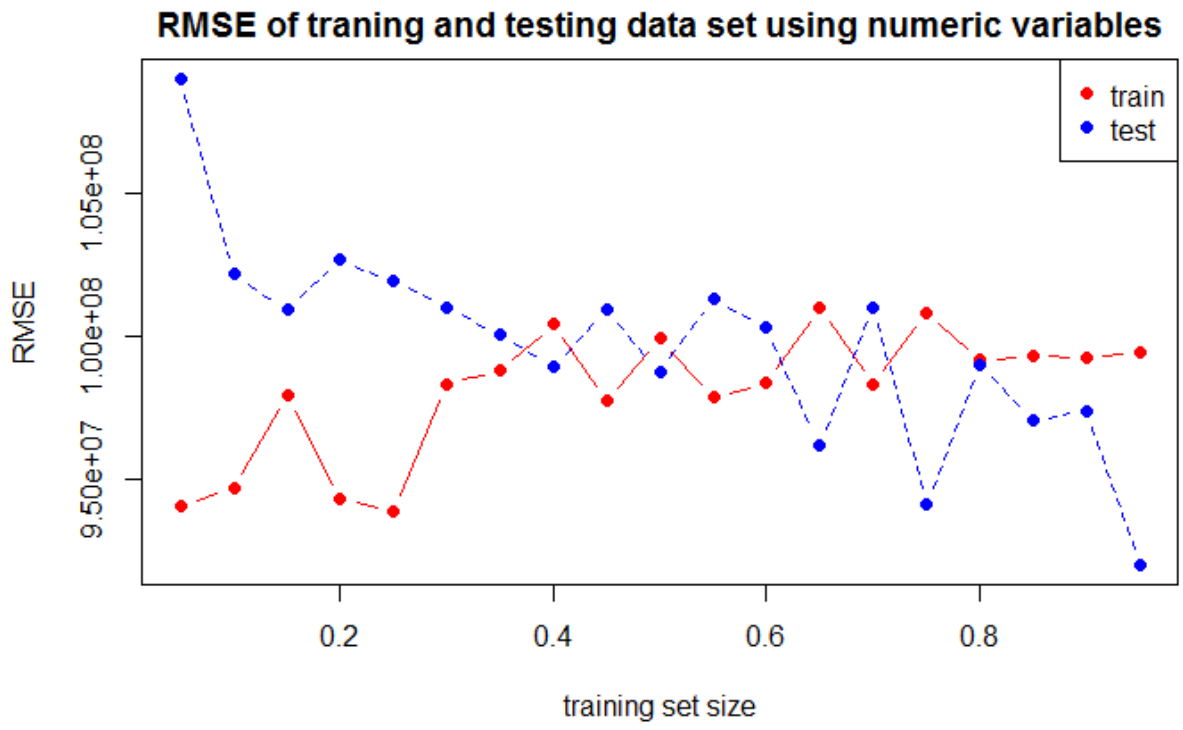
Mean of training RMSE=97649830; Mean of testing RMSE=105134067

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`).

```
# TODO: Build & evaluate model 2 (transformed numeric variables only)
# Remove the variables that has no (very small) correlations with Gross and
only gives noise. Also remove highly correlated cols. This analysis has been
done on project 1. The variables that are removed are: "Year",
"Runtime","tomatoRotten","tomatoUserMeter","tomatoUserReviews","tomatoMeter"

# Use binning for budget 0-10M->binning1; 10-20M->binning2 ; >20M-> binning3.
df_final$Budget_bin<-
as.numeric(cut(df_final$Budget,breaks=c(0,8000000,Inf),labels=0:1))


#Use binning for Year
# df_final$Year_bin<-
cut(df_final$Year,breaks=c(0,2006,2011,Inf),labels=c("1","2","3"))
# df_final$Year_bin<- as.numeric(as.character(df_final$Year_bin))
#
# df_rating_gross <- df_final[, c("tomatoUserMeter", "Gross")]
# library(GGally)
# ggpairs(na.omit(df_rating_gross),upper = list(continuous = wrap("cor",size
= 10)), lower = list(continuous = wrap("smooth", color="red")))

# use power for several parameters
df_trans<-
df_final[c("Title","Gross","Budget","Budget_bin","imdbVotes","tomatoReviews",
"tomatoFresh","imdbRating" ,"tomatoUserMeter","Metascore","tomatoRating")]
feature_name=c("Budget","Budget^2","Budget^3","Budget_bin","imdbVotes^2","tom
```

```
atoReviews","tomatoFresh","imdbRating^2" ,"tomatoUserMeter^2","Metascore")
formula<- as.formula(paste("Gross ~ ", paste(feature_name, collapse= "+")))
rmse=RMSE(formula,df_trans)
RMSE_train=unlist(rmse[1])
RMSE_test=unlist(rmse[2])

size=seq(0.05, 0.95, length.out = 19)
matplot(size,cbind(RMSE_train, RMSE_test), type="b",pch=19, col=c("red",
"blue"), xlab="training set size", ylab="RMSE")
legend("topright", legend=c("train", "test"), pch=19, col=c("red", "blue"))
title ("RMSE of traning and testing data set using numeric variables")
```



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

A: I used two types of transformations, which slighly reduced the RMSEs of testing sets. 1st type is power transforms that applied to "Bugdet", "imdbRating", and "tomatoUserMeter" because all have a non-linear relation with Gross. A transformation to 2rd or 3rd power provides better fits to the linear model.2nd type is binning. The binning of Budget into 2 bins yields smaller RMSE and reduces overfitting, because by wisely choosing binning bucket (through experiment), the gross has established a relation to bins.

In addition, it is found through plot Gross vs. other numeric variables, that not all of the numeric variables have strong correlation with Gross, thus these variables, including "Year","Runtime","tomatoRotten","tomatoUserMeter","tomatoUserReviews","tomatoMeter ", are removed.

The RMSE of this model is overall slightly better than Model 1. The results are:

Mean of training RMSE=97611965; Mean of testing RMSE=99161536

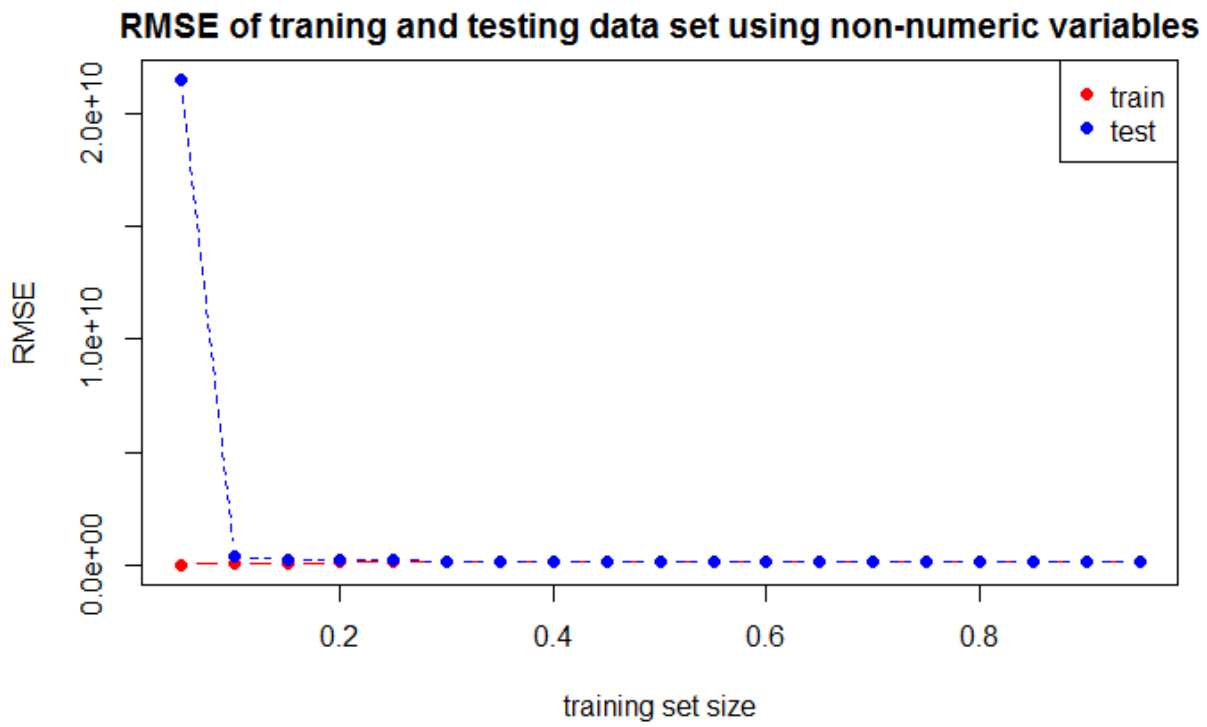
3. Non-numeric variables

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.

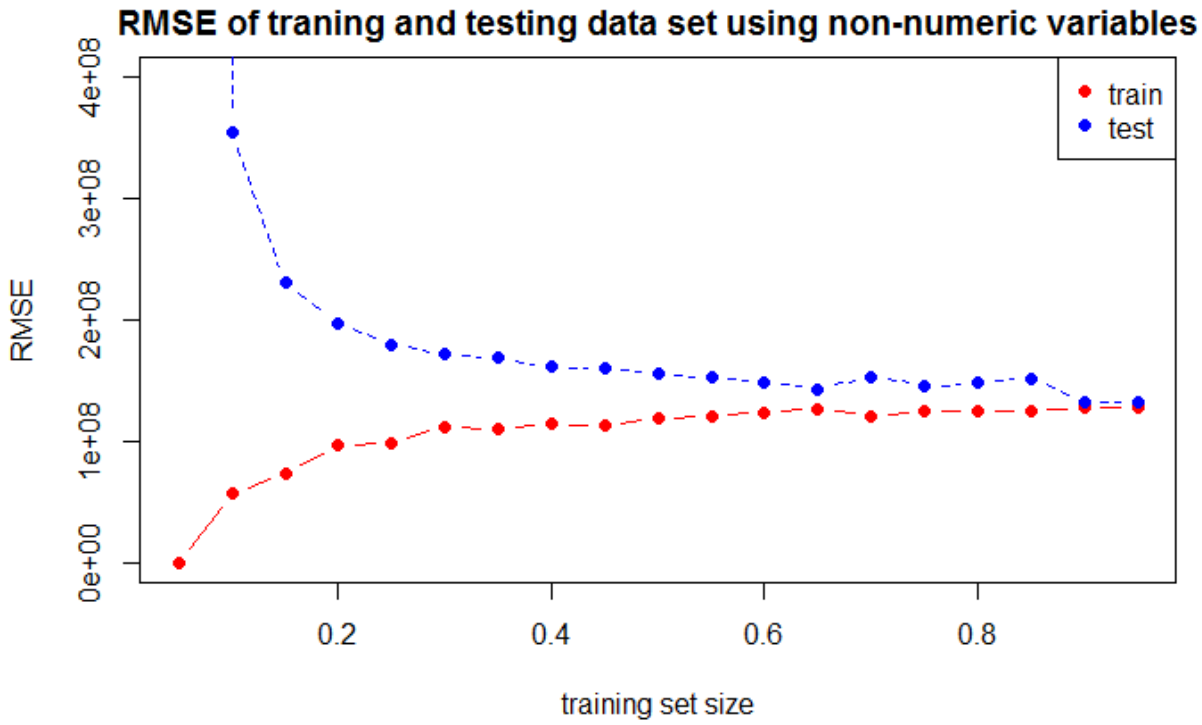
```
# TODO: Build & evaluate model 3 (converted non-numeric variables only)

#Convert Awards to 2 numeric columns: wins and nominations
get_wins<-function(str){
wins=0
if(str=="N/A" | is.na(str)){return(0)}
else {
s=unlist(strsplit(str,split="[\\&\\.]"))
for(i in 1:length(s)){
if (grepl("win",s[i],ignore.case = TRUE) | grepl("won",s[i],ignore.case
= TRUE)){
n=as.numeric(strsplit(s[i],"\\D+")[[1]])
n=n[length(n)]
wins=wins+n
}
}
}
return(wins)
}
get_nomin<-function(str){
nomin=0
if(str=="N/A" | is.na(str)){return(0)}
else {
s=unlist(strsplit(str,split="[\\&\\.]"))
for(i in 1:length(s)){
if (grepl("nomin",s[i],ignore.case = TRUE)){
n=as.numeric(strsplit(s[i],"\\D+")[[1]])
n=n[length(n)]
nomin=nomin+n
}
}
}
return(nomin)
}
df_final$Wins<-sapply(df_final$Awards,get_wins)
df_final$Nominations<-sapply(df_final$Awards,get_nomin)

# #Select the most frequent Catergories with proportions higher than
threshold.
```

```r
library(tidyr)
library(reshape2)
df_non_numeric<-
df_final[c("Title","Gross","Wins","Nominations","Genre","Language", "Country"
, "tomatoImage","Production","Actors","Director","Writer")]
for (col in c("Genre","Language", "Country" ,
"tomatoImage","Production","Actors","Director","Writer")){
  df_temp<-df_non_numeric
  df_temp$new<-strsplit(df_temp[,col],"(\\s)?,(\\s)?")  #split string by ","
  df_temp<-unnest(df_temp)

  temp_table<-table(as.character(df_temp$new),exclude=NA)
  Freq_table<- prop.table(temp_table)  #generate frequency table
  Top50<-sort(Freq_table, decreasing=TRUE)[1:50] #select top 50 from the
frequeny table
  namelist<-names(Top50)
  namelist <- gsub("([A-Za-z]+).*", "\\1", namelist)
  namelist<-tolower(namelist)

  df_non_numeric[,col]<-gsub(","," ", df_non_numeric[,col])
  Col_genre<-c(df_non_numeric[,col])
  library(tm)
  Corpus_genre <- Corpus(VectorSource(Col_genre))
  TDM_genre<-TermDocumentMatrix(Corpus_genre, control = list(minWordLength =
1))
  Gen_cols<- data.frame(t(as.matrix(TDM_genre)))

  Gen_cols<-Gen_cols[,colnames(Gen_cols)%in%namelist]
  df_non_numeric<-cbind(df_non_numeric,Gen_cols)
  df_non_numeric<-df_non_numeric[,-which(names(df_non_numeric) == col)]
}

df_non_numeric2=subset(df_non_numeric,select=-c(Title))

formula<- as.formula("Gross~.")
rmse=RMSE(formula,df_non_numeric2)
RMSE_train=unlist(rmse[1])
RMSE_test=unlist(rmse[2])
print(mean(RMSE_train))

## [1] 105867493

print(mean(RMSE_test))

## [1] 1296273454

size=seq(0.05, 0.95, length.out = 19)
matplot(size,cbind(RMSE_train, RMSE_test), type="b",pch=19, col=c("red",
"blue"), xlab="training set size", ylab="RMSE")
```

```
legend("topright", legend=c("train", "test"), pch=19, col=c("red", "blue"))
title ("RMSE of traning and testing data set using non-numeric variables")
```




```
matplot(size,cbind(RMSE_train, RMSE_test), type="b",pch=19, col=c("red",
"blue"), xlab="training set size",ylim = c(0,4e08), ylab="RMSE")
legend("topright", legend=c("train", "test"), pch=19, col=c("red", "blue"))
title ("RMSE of traning and testing data set using non-numeric variables")
```



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

A: The categorical variables I used are "Wins","Nominations","Genre","Language", "Country" , "tomatoImage","Production","Actors","Director","Writer". The method of encoding is to convert each catergorical varaiables among "Genre","Language", "Country" , "tomatoImage","Production","Actors","Director" and "Writer" into multiple binary vectors with 1s representing the presence and 0s the absence. The method is similar to the Genre column in project 1. By doing so, it will generate thousands of additional columns of author names, actor names, writer names, etc. To reduce the large number of columns and substantial computational load/time, I only choose the top 50 text strings in each category (genre, actor, writer, etc) that most freqently appear in that column. This reduced the number new parsed columns to ~280, yet produces reasonable RMSE of the prediction.

For the Awards column, I use similar method as project, i.e. to parse it to a "Wins" column that calculate total wins and a "Nomination" column that calculate total nominations.This way it converts Awards into numerical variables.

The above graph shows that the trendings of traning RMSE and testing RMSE of non-numerical varaible are similar The results for the non-numerical model and numerical modle, as trainign set increases. Since for testing, the first data point from 5% training set size yields too much overfitting, it maybe more meaningful to exclude it.

Mean of training RMSE=105867493; Mean of testing RMSE =1296273454, excluding the first point (from high overfitting), RMSE=171037956

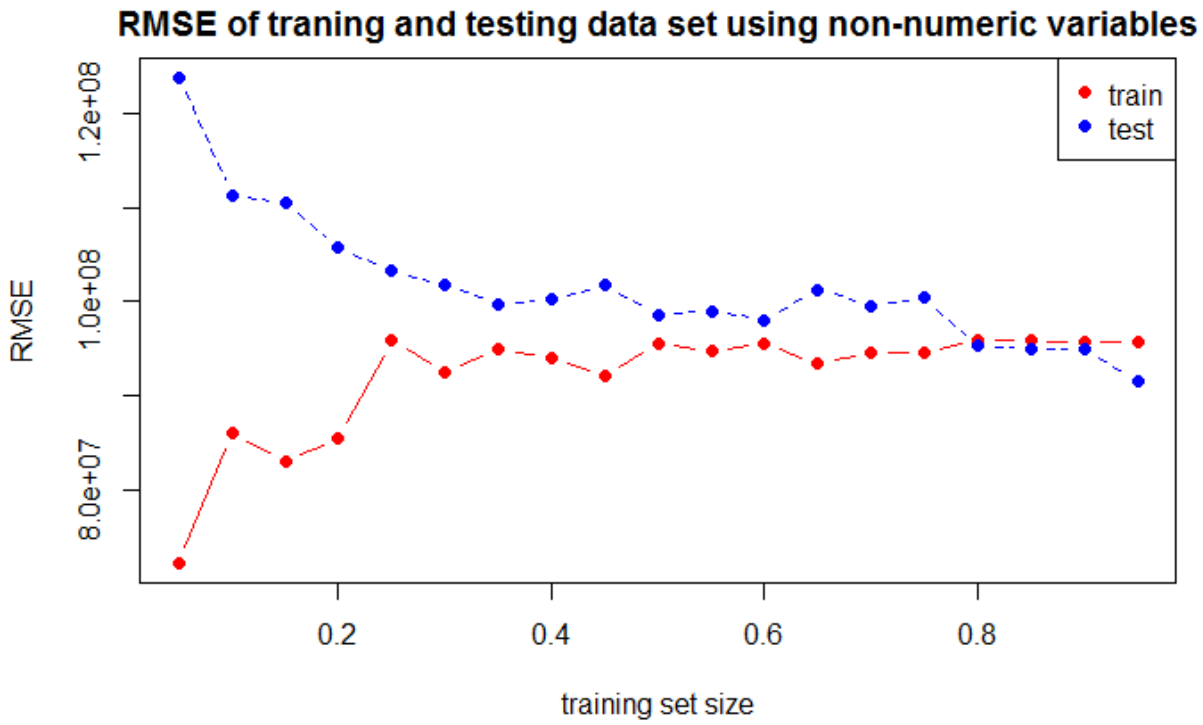
The train RMSE is simiar to the previous (a little bit larger), but the test RMSE is much larger .This maybe due to non-numeric variables have higher variance. Another observation is that non-numeric model plot is more smooth, which maybe because of more predictors (each categorical variable can be decomposed to many many binary sets of sub-variables).

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)
library(tidyr)
df_non_numeric<-df_final[c("Title","Nominations","Wins","Genre",
"Country","Production","Director","Actors")]
for (col in c("Genre", "Country","Production" ,"Director","Actors")){
  df_temp<-df_non_numeric
  df_temp$new<-strsplit(df_temp[,col],"(\\s)?,(\\s)?")  #split string by ","
  df_temp<-unnest(df_temp)

  temp_table<-table(as.character(df_temp$new),exclude=NA)
  Freq_table<- prop.table(temp_table)  #generate frequency table
  Top50<-sort(Freq_table, decreasing=TRUE)[1:50] #select top 50 from the
frequeny table
  namelist<-names(Top50)
  namelist <- gsub("([A-Za-z]+).*", "\\1", namelist)
  namelist<-tolower(namelist)

  df_non_numeric[,col]<-gsub(","," ", df_non_numeric[,col])
  Col_genre<-c(df_non_numeric[,col])
  library(tm)
  Corpus_genre <- Corpus(VectorSource(Col_genre))
  TDM_genre<-TermDocumentMatrix(Corpus_genre, control = list(minWordLength =
1))
  Gen_cols<- data.frame(t(as.matrix(TDM_genre)))

  Gen_cols<-Gen_cols[,colnames(Gen_cols)%in%namelist]
  df_non_numeric<-cbind(df_non_numeric,Gen_cols)
  df_non_numeric<-df_non_numeric[,-which(names(df_non_numeric) == col)]
}

df_all<-merge(df_trans,df_non_numeric,by="Title")
df_all=subset(df_all,select=-c(Title))
# df_rating_gross <- df_all[, c("Nominations.x", "Gross.x")]
# library(GGally)
# ggpairs(na.omit(df_rating_gross),upper = list(continuous = wrap("cor",size
= 10)), lower = list(continuous = wrap("smooth", color="red")))

formula<- as.formula("Gross~.")
```

```
rmse=RMSE(formula,df_all)
RMSE_train=unlist(rmse[1])
RMSE_test=unlist(rmse[2])
print(mean(RMSE_train))

## [1] 79548491

print(mean(RMSE_test))

## [1] 127459019

size=seq(0.05, 0.95, length.out = 19)
matplot(size,cbind(RMSE_train, RMSE_test), type="b",pch=19, col=c("red",
"blue"), xlab="training set size", ylab="RMSE")
legend("topright", legend=c("train", "test"), pch=19, col=c("red", "blue"))
title ("RMSE of traning and testing data set using non-numeric variables")
```



The results for the combined numeric and non-numeric model are as follows.Since for testing, the first data point from 5% training set size yields too much overfitting, it maybe more meaningful to exclude it.

Mean of training RMSE=79548491; Mean of testing RMSE =127459019; Excluding the 5% training set data poing, RMSE= 108941951

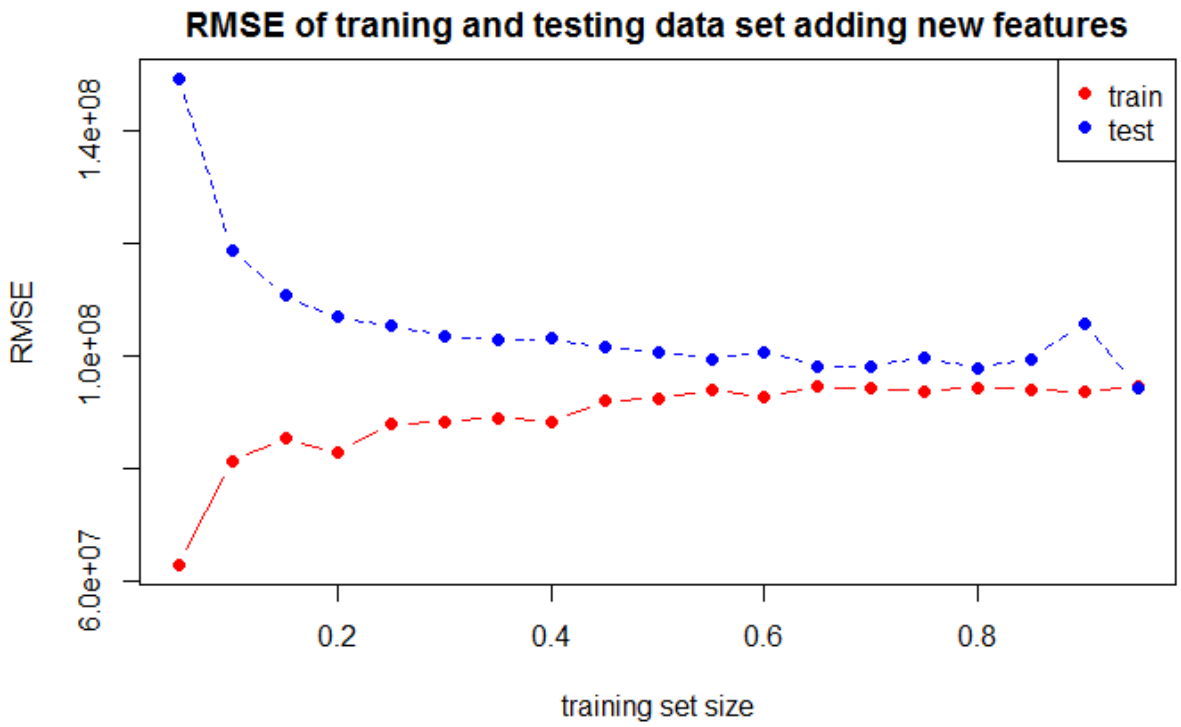
There is considerably improvement especially for training RMSE, by combining both model.Actually because train and test converged as training set size increases, testing RMSE should also be improved if not considering the overfitting for small training set.

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`).

```
# TODO: Build & evaluate model 5 (numeric, non-numeric and additional
features)
df_add<-df_final
for (col in c("tomatoImage")){
  df_add$new<-strsplit(df_add[,col],"(\\s)?,(\\s)?")  #split string by ","
  df_add<-unnest(df_add)

  temp_table<-table(as.character(df_add$new),exclude=NA)
  Freq_table<- prop.table(temp_table)  #generate frequency table
  Top15<-sort(Freq_table, decreasing=TRUE)[1:15] #select top 50 from the
frequeny table
  namelist<-names(Top15)

  df_add=df_add[df_add$new%in%namelist,]
  df_add<-df_add[,-which(names(df_add) == col)]

  library(plyr)
  names(df_add)[names(df_add)=="new"]<-col
}
# See which categorical variable can be used as binning.
#ggplot(data = na.omit(df_add), aes(tomatoImage, Gross)) +
geom_boxplot(outlier.size = NA)

#binning by is_genre_animation....

df_all$Budget_bin<-
as.numeric(cut(df_all$Budget,breaks=c(0,10000000,Inf),labels=0:1))

df_add_features<-df_all

df_add_features$is_genre_animation<-ifelse(df_add_features$animation>0 &
df_add_features$Budget_bin>0,1,0)
df_add_features$is_direct_chris<-ifelse(df_add_features$christopher>0 &
df_add_features$Budget_bin>0,1,0)
df_add_features$is_actor_johnny<-ifelse(df_add_features$johnny>0 &
df_add_features$Budget_bin>0,1,0)
df_add_features$is_country_new<-ifelse(df_add_features$new>0 &
df_add_features$Budget_bin>0,1,0)
df_add_features$is_production_disney<-ifelse(df_add_features$walt >0 &
df_add_features$Budget_bin>0,1,0)

# df_add_features<-subset(df_add_features,select =
c("Gross","Budget","Budget_bin","imdbVotes","tomatoReviews","tomatoFresh","im
```

```
dbRating"
,"tomatoUserMeter","Metascore","is_genre_animation","is_direct_chris","is_act
or_johnny","is_country_new","is_production_disney","Nominations","Wins"))


feature_name=c("Budget","Budget_bin","imdbVotes","tomatoReviews","tomatoFresh
","imdbRating","tomatoUserMeter","Metascore","is_genre_animation","is_direct_
chris","is_actor_johnny","is_country_new","is_production_disney","Nominations
","Wins","horror", "sci", "adventure", "comedy", "family", "crime", "music", "
drama", "mystery", "thriller", "sport", "fantasy", "romance", "action", "biogr
aphy", "documentary", "history", "animation", "musical",  "western", "war",
"news", "canada", "germany", "usa", "new", "denmark",
"france", "italy", "netherlands", "spain", "australia", "morocco", "china", "h
ong", "ireland", "hungary", "india", "japan",
"argentina", "south", "belgium", "finland", "romania", "malta", "luxembourg",
"mexico", "czech", "switzerland", "jordan", "taiwan", "united",
"austria", "norway", "bosnia", "brazil", "egypt", "singapore", "thailand", "sw
eden",  "iceland", "poland",  "bahamas",

"chile", "russia", "bulgaria", "greece", "israel", "lithuania", "isle",
"lions", "sony", "mgm", "fox", "screen", "paramount", "warner", "miramax", "bu
ena", "universal", "ifc", "columbia", "new.1", "anchor", "dimension", "thinkfi
lm", "dreamworks", "touchstone", "summit", "magnolia", "lionsgate",
"focus", "walt", "cbs"  )
formula<- as.formula(paste("Gross ~ ", paste(feature_name, collapse= "+")))
rmse=RMSE(formula,df_add_features)
RMSE_train=unlist(rmse[1])
RMSE_test=unlist(rmse[2])
print(mean(RMSE_train))

## [1] 89281079

print(mean(RMSE_test))

## [1] 105092365

size=seq(0.05, 0.95, length.out = 19)
matplot(size,cbind(RMSE_train, RMSE_test), type="b",pch=19, col=c("red",
"blue"), xlab="training set size", ylab="RMSE")
legend("topright", legend=c("train", "test"), pch=19, col=c("red", "blue"))
title ("RMSE of traning and testing data set adding new features")
```



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

A: The new model has embeded with additional features as follows. a) interactions of is_genre_animation and is_budget_greater_than_10M (choose animiation because it has the highest median of Gross and are more likely has budget larger than 10M) b) interactions of is_director_chris and is_budget_greater_than_10M (choose Chris because it has the highest median of Gross and are more likely has budget larger than 10M) c) interactions of is_actor_johnny and is_budget_greater_than_10M (choose Johnny because it has the highest median of Gross and are more likely has budget larger than 10M) d) interactions of is_country_new and is_budget_greater_than_10M (choose New Zealand because it has the highest median of Gross and are more likely has budget larger than 10M) e) interactions of is_production_disney and is_budget_greater_than_10M (choose Disney because it has the highest median of Gross and are more likely has budget larger than 10M)

The results are:

Mean of training RMSE=89281079; Mean of testing RMSE =105092365; Excluding the 5% training set data point, RMSE= 102629973

This new model has produced relatively small RMSE than both numeric variable and non-numeric variables with less noise (more stable). Therefore it gives a more accurate prediction. Though adding more non-numeric variables could reduce the train RMSE, the test RMSE will be higher (based on Q4). Thus, this model provides relative reasonable balance.