Predicting Box Office Revenue of Movies Released During the COVID-19 Pandemic Lockdowns

Julianna Harwood

February 2022

Introduction

The purpose of this project is to use common machine learning algorithms to predict the box office performance of movies that could not be released in theaters because of the lock downs associated with COVID-19.

The data set consists of 1036 titles released between 2017 and March 2020 with greater than $1M total box office revenue as well as 29 movies released after March 2020 for which I would like to predict their box office revenues. The features are as follows:

- imdb_id (string): Unique identifier of the movie
- title_name (string): Title of movie
- totalbox (int): Total box office revenue of movie ($)
- mpaarating (string): MPA rating of movie
- grade (string): CinemaScore movie grade
- genre (string): Movie genre
- title_launch_date (string): Date of movie opening
- rotten_tomatoes_score (float): Rotten Tomatoes score
- Interest_index (float): Metric to measure interest in movie (unknown source)
- wiki_pre_wk1_pageviews: Total views of the movie's Wikipedia page prior to release

Analysis

Inspect and Clean Data

Read in and summarize data to check for missing data and understand what cleaning needs to be done.

```
df <- read_excel('box_pred_sample_test_data.xlsx')
summary(df)
```

```
##    imdb_id           title_name          totalbox          mpaarating
##  Length:1065        Length:1065        Min.   :  1030502   Length:1065
##  Class :character   Class :character   1st Qu.: 16580391   Class :character
##  Mode  :character   Mode  :character   Median : 39690146   Mode  :character
##                                        Mean   : 75151563
##                                        3rd Qu.: 85482890
##                                        Max.   :858373000
##                                        NA's   :29
##     grade           title_launch_date                genre
##  Length:1065        Min.   :2016-01-04 00:00:00   Length:1065
##  Class :character   1st Qu.:2017-04-21 00:00:00   Class :character
##  Mode  :character   Median :2018-03-23 00:00:00   Mode  :character
##                     Mean   :2018-04-11 22:56:27
##                     3rd Qu.:2019-03-08 00:00:00
```

```
##                       Max.   :2021-07-27 00:00:00
##
##  rotten_tomatoes_score wiki_pre_wk1_pageviews Interest_index
##  Min.   :11.00         Min.   :      15.0      Min.   :     99.6
##  1st Qu.:48.25         1st Qu.:     954.8      1st Qu.:   7816.5
##  Median :67.00         Median :  14802.5      Median :  16955.1
##  Mean   :64.80         Mean   :  47738.3      Mean   :  43284.4
##  3rd Qu.:82.00         3rd Qu.:  57277.2      3rd Qu.:  43334.4
##  Max.   :99.00         Max.   :1156621.0      Max.   :1285447.3
##  NA's   :55            NA's   :585
```

```
sum(is.na(df$grade))
```

```
## [1] 302
```

```
sum(!is.na(df$grade[is.na(df$totalbox)]))
```

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

```
sum(!is.na(df$wiki_pre_wk1_pageviews[is.na(df$totalbox)]))
```

```
## [1] 2
```

Convert categorical data to factors and define *title_launch_date* as a date.

```
df$mpaarating <- as.factor(df$mpaarating)
df$grade <- as.factor(df$grade)
df$genre <- as.factor(df$genre)
df$title_launch_date <- as.Date(df$title_launch_date)
```

From the summary of the dataframe, we can see that 28% of *grade* and 55% *wiki_pre_pageviews* is missing. There is also only 1 post-covid title that has data for *grade* and 2 that have data for *wiki_pre_wk1_pageviews*. There are 44 missing values for *rotten_tomatoes_score*.

To gain more insight from the *title_launch_day* variable, I create variables for release month, day and weekday.

```
# create month, day and weekday variables
df$release_month <- as.numeric(format(df$title_launch_date, "%m"))
months_nums = c(1,2,3,4,5,6,7,8,9,10,11,12)
months_names = c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")
df$release_month <- factor(df$release_month, levels = months_nums, labels = months_names)

df$release_day <- as.numeric(format(df$title_launch_date, "%d"))
df$release_weekday <- as.factor(weekdays(df$title_launch_date))
```

Lastly, I save the *imdb_id* and *title_name* of the post covid movies to be used later.

```
post_covid_ids <- df[is.na(df$totalbox),c("imdb_id","title_name")]
```

Bivariate Analysis

To understand relationships between the data, I perform bivariate analysis on the average *totalbox* for different levels of the factor variables.

```
aggregate(totalbox ~ grade, df, mean)
```

```
##    grade  totalbox
## 1      A 165827890
## 2     A- 108296423
```

```
## 3     A+ 258190255
## 4      B  48437563
## 5     B-  33604006
## 6     B+  69597878
## 7      C  26181532
## 8     C-  14818747
## 9     C+  21877571
## 10     D   8747046
## 11    D-   5775178
## 12    D+  28978276
## 13     F  17800004
```

```
aggregate(totalbox ~ mpaarating, df, mean)
```

```
##   mpaarating  totalbox
## 1          G 152127946
## 2         PG 102164731
## 3      PG-13  88328969
## 4          R  45862474
```

```
aggregate(totalbox ~ genre, df, mean)
```

```
##               genre  totalbox
## 1            Action 136013046
## 2  Action/Adventure  37016922
## 3         Adventure 243550282
## 4         Animation 130668101
## 5            Comedy  41295570
## 6       Documentary  12107972
## 7             Drama  39022035
## 8            Family 150189616
## 9            Horror  57271931
## 10          Musical 118310920
## 11 Mystery Suspense  38545388
## 12  Science Fiction  44627815
```

```
aggregate(totalbox ~ release_month, df, mean)
```

```
##    release_month  totalbox
## 1            Jan  45212807
## 2            Feb  69263279
## 3            Mar  74411715
## 4            Apr  87271966
## 5            May  87725040
## 6            Jun 102479894
## 7            Jul  99466485
## 8            Aug  51265446
## 9            Sep  51580897
## 10           Oct  41188223
## 11           Nov  91764210
## 12           Dec  99143778
```

```
aggregate(totalbox ~ release_weekday, df, mean)
```

```
##   release_weekday  totalbox
## 1          Friday  72931040
## 2          Monday  76324389
```

```
## 3          Sunday  80103209
## 4        Thursday 134189012
## 5         Tuesday  87672007
## 6       Wednesday  95589394
```

Train Models

As a baseline, I will use simple linear regression to predict *totalbox*. The lm package of R does not have a robust way of fitting data with missing values, so I leave *grade*, *rotten_tomatoes_score* and *wiki_pre_wk1_pageviews* out. I choose an 80/20 split of the final dataset to fit and test the model. The primary success metric I will use is Root Mean Squared Error (RMSE), which measures the amount, on average, the predictions deviate from the observed data. Another success metric I will look at is R-Squared, which measures the amount of variance in the response variable explained by the predictors. When tested on the test dataset, the model has an R-Squared of 0.688, meaning that 68.8% of the variance in *totalbox* can be explained from the model. The RMSE of 5.97e+07 means that on average, this model has an error of about $59.7M.

```
# create final dataset
df$title_launch_date <- as.numeric(as.POSIXct(df$title_launch_date))
vars <- c("totalbox","mpaarating","genre","Interest_index",
          "release_month","release_day",
          "title_launch_date","release_weekday")
final_df <- df[,vars]

# split post covid titles for prediction with final model
post_covid_titles <- final_df[is.na(final_df$totalbox),]
final_df <- final_df[!is.na(final_df$totalbox),]

# create test and train datasets
set.seed(1234)
index <- createDataPartition(final_df$totalbox, p=.8, list=FALSE)
train <- final_df[index,]
test <- final_df[-index,]

# fit linear regression model
set.seed(1234)
fit <- lm(totalbox ~ . - release_weekday, data = train, na.action = na.omit)
fit
```

```
##
## Call:
## lm(formula = totalbox ~ . - release_weekday, data = train, na.action = na.omit)
##
## Coefficients:
##          (Intercept)           mpaaratingPG         mpaaratingPG-13
##            5.169e+08             -1.738e+07              -1.848e+07
##          mpaaratingR  genreAction/Adventure          genreAdventure
##           -4.288e+07             -5.312e+07              -2.920e+06
##       genreAnimation            genreComedy        genreDocumentary
##            5.518e+06             -4.115e+07              -6.686e+07
##           genreDrama            genreFamily             genreHorror
##           -5.090e+07              2.506e+07              -3.460e+07
##         genreMusical  genreMystery Suspense    genreScience Fiction
##           -9.741e+07             -3.925e+07              -4.360e+07
##       Interest_index       release_monthFeb        release_monthMar
##            8.275e+02              4.365e+06               1.057e+07
```

```
##      release_monthApr       release_monthMay       release_monthJun
##            -1.027e+07              1.403e+07              3.662e+07
##      release_monthJul       release_monthAug       release_monthSep
##             3.104e+07             -1.091e+05              5.515e+06
##      release_monthOct       release_monthNov       release_monthDec
##             4.701e+06              3.879e+07              3.457e+07
##           release_day      title_launch_date
##            -3.405e+05             -2.830e-01
```

```
pred_lm <- predict(fit, test)
postResample(pred=pred_lm, obs=test$totalbox)
```

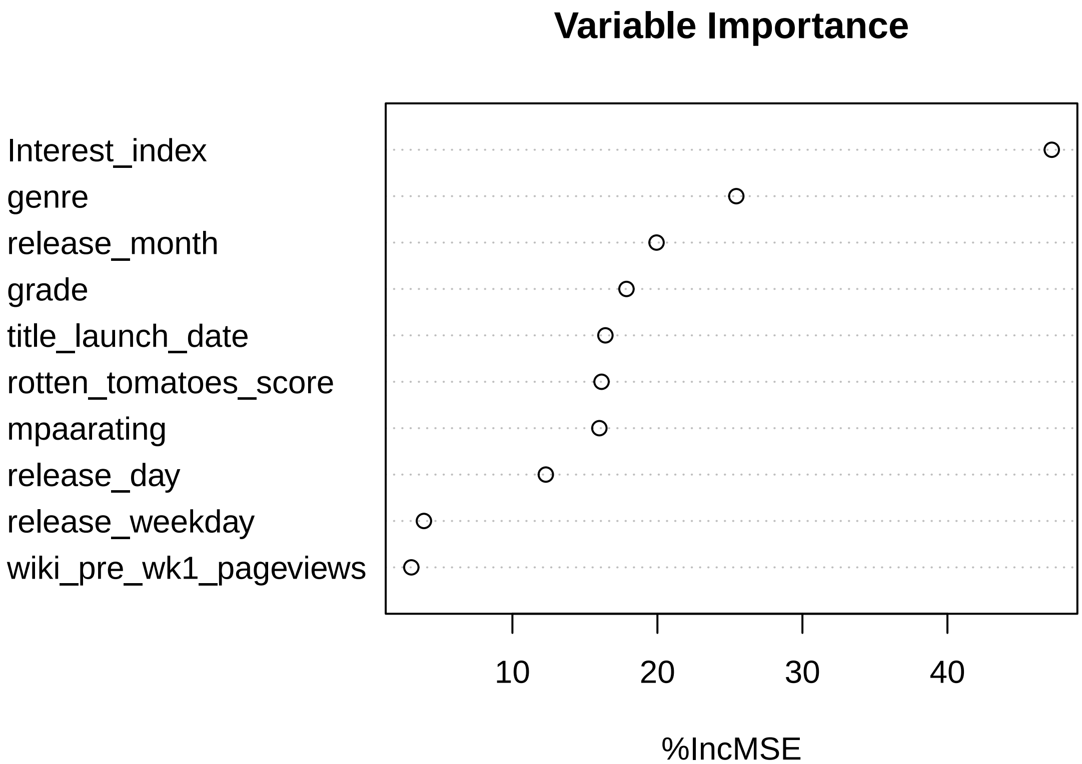
```
##         RMSE     Rsquared          MAE
## 5.974061e+07 6.879007e-01 3.755570e+07
```

To improve on these predictions, I use a Random Forest model. Random Forest models do not have the same assumptions as linear regression models with regards to normality of the predictor variables. They works well "out of the box" for many different types of data. It also has an option to impute missing data while training, so I am able to add in *grade*, *rotten_tomatoes_score* and *wiki_pre_wk1_pageviews* as predictors. I again use an 80/20 split for training and testing data and grow 400 trees.

```
# create final dataset
vars <- c("totalbox","mpaarating","genre","Interest_index",
          "release_month","release_day", "grade", "wiki_pre_wk1_pageviews",
          "rotten_tomatoes_score","title_launch_date","release_weekday")
final_df <- df[,vars]

# split post covid titles for prediction with final model
post_covid_titles <- final_df[is.na(final_df$totalbox),]
final_df <- final_df[!is.na(final_df$totalbox),]

# create test and train datasets
set.seed(1234)
index <- createDataPartition(final_df$totalbox, p=.8, list=FALSE)
train <- final_df[index,]
test <- final_df[-index,]

# fit initial model
set.seed(1234)
fit.forest.initial <- randomForest(totalbox ~ ., data = train,
                              ntree = 400, na.action = na.roughfix, importance = TRUE)
pred.initial <- predict(fit.forest.initial, test)
postResample(pred=pred.initial, obs=test$totalbox)
```

```
##         RMSE     Rsquared          MAE
## 3.947597e+07 9.493500e-01 2.338503e+07
```

```
varImpPlot(fit.forest.initial, type = 1, main = "Variable Importance")
```



The RMSE and Rsquared have improved drastically with the RandomForest. From the Variable Importance plot, *Interest_index* is by far the most important variable. The %IncMSE on the X-axis measures the percent change in the mean squared error of the predictions if that variable had been left out of the model. *wiki_pre_wk1_pageviews* is the least important variable and given that about 50% of it is missing values, I try the model without it, and get a slight decrease in RMSE.

```
# fit model without wiki_pre_wk1_pageviews
set.seed(1234)
fit.forest.initial <- randomForest(totalbox ~ . - wiki_pre_wk1_pageviews, data = train,
                          ntree = 400, na.action = na.roughfix, importance = TRUE)
pred.initial <- predict(fit.forest.initial, test)
postResample(pred=pred.initial, obs=test$totalbox)
```

```
##         RMSE     Rsquared          MAE
## 3.885448e+07 9.484215e-01 2.188866e+07
```

Next, I try a manual grid search across possible hyperparameters mtry and ntree. mtry is the number of variables sampled as candidates at each split and ntree is the number of trees grown for each forest model. I store all results and save the best model.

```
final_df = subset(final_df, select = - c(wiki_pre_wk1_pageviews))
post_covid_titles <- subset(post_covid_titles, select = - wiki_pre_wk1_pageviews)

# create test and train datasets
set.seed(1234)
index <- createDataPartition(final_df$totalbox, p=.8, list=FALSE)
train <- final_df[index,]
test <- final_df[-index,]

# tune random fo
mtrys = c(3,4,5,6,7,8,9)
```

```r
ntrees = c(100, 300, 500,700,900,1100)

tune_results <- data.frame(matrix(ncol = length(ntrees), nrow = length(mtrys)))
colnames(tune_results) <- ntrees
rownames(tune_results) <- mtrys

lowest_RMSE <- 100000000

for (mtry in mtrys)
{
  for (ntree in ntrees)
       {
         set.seed(1234)
         fit.forest <- randomForest(totalbox ~ ., data = train,
                             ntree = ntree, mtry = mtry, na.action = na.roughfix, importance = TRUE)
         pred <- predict(fit.forest, test)
         results <- postResample(pred=pred, obs=test$totalbox)
         tune_results[toString(mtry),toString(ntree)] <- results[["RMSE"]]
         if (results[["RMSE"]] < lowest_RMSE) {
           lowest_RMSE <- results[["RMSE"]]
           final_results <- results
           final_model <- fit.forest
         }
       }
}

tune_results
```

```
##        100      300      500      700      900     1100
## 3 34033792 34507807 34308215 34300363 34182401 33995371
## 4 34048296 31839307 31913169 32228356 32396705 32440715
## 5 30786017 32154464 32799745 32621355 32404281 32409739
## 6 31061917 31824127 32274705 32105901 32297343 32126797
## 7 32750302 32145720 31806920 32119428 32235370 32133905
## 8 31984686 32718936 32737523 32898622 32723159 32790136
## 9 31669429 33242659 32835009 32639757 32691140 32582014
```

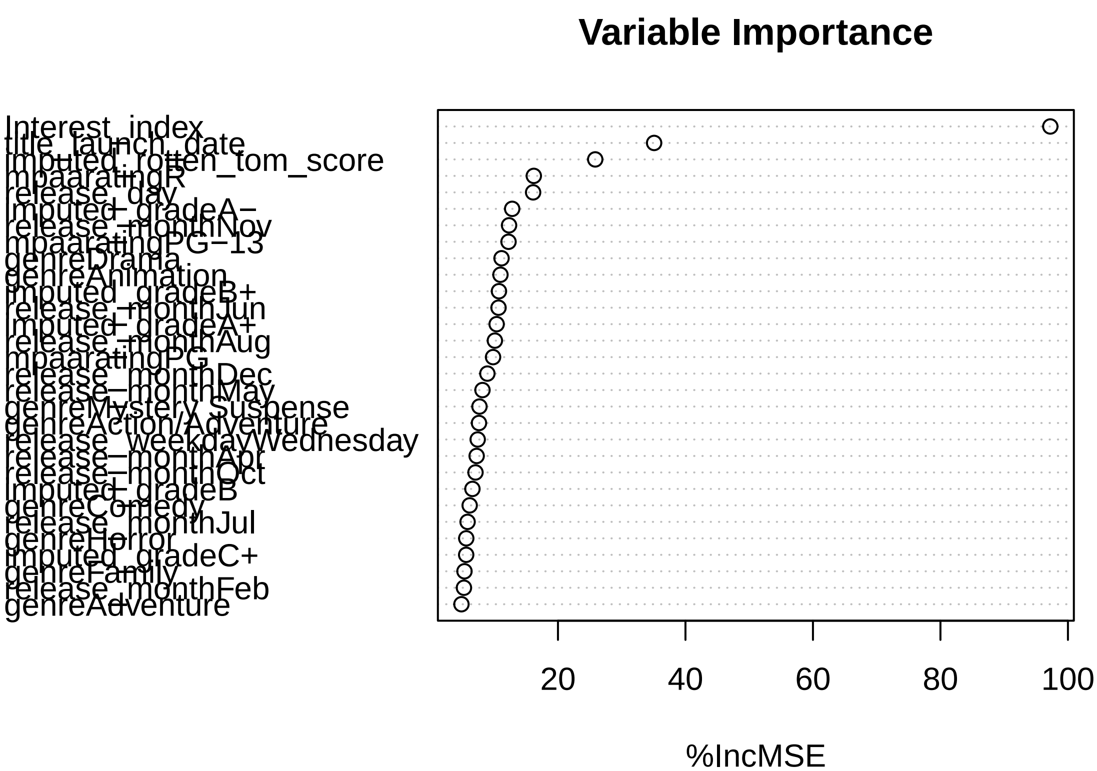
```r
final_results
```

```
##         RMSE     Rsquared          MAE
## 3.078602e+07 9.397916e-01 1.924879e+07
```

As another tuning method, I use the caret package to autotune the RandomForest. I use repeated cross-validation with 10 different values of mtry and 500 trees. As caret does not have the na.roughfix option that the RandomForest package has, I use k nearest neighbors to impute the missing *rotten_tomatoes_score* and *grade* values.

```r
# create final dataset
final_df <- df[,vars]

# impute missing values
imputed_data <- knnImputation(final_df, scale = TRUE)
final_df <- cbind(final_df, imputed_data$rotten_tomatoes_score, imputed_data$grade)
final_df <- rename(final_df, 'imputed_rotten_tom_score' = 'imputed_data$rotten_tomatoes_score')
final_df <- rename(final_df, 'imputed_grade' = 'imputed_data$grade')
final_df = subset(final_df, select = - c(rotten_tomatoes_score,grade,wiki_pre_wk1_pageviews))
```

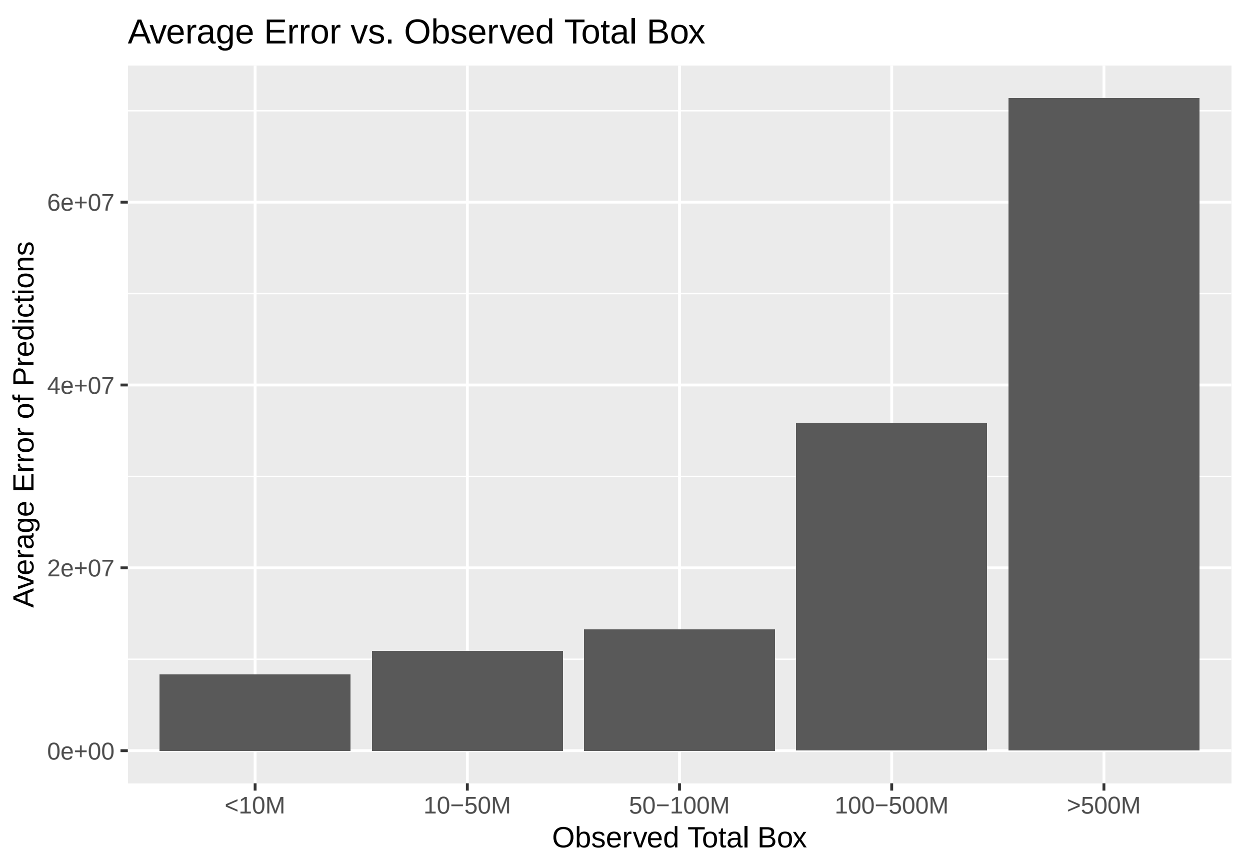
```r
# split post covid titles for prediction with final model
post_covid_titles <- final_df[is.na(final_df$totalbox),]
final_df <- final_df[!is.na(final_df$totalbox),]

# create test and train datasets
set.seed(1234)
index <- createDataPartition(final_df$totalbox, p=.8, list=FALSE)
train <- final_df[index,]
test <- final_df[-index,]

# train and tune model with train data
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
set.seed(1234)
final.forest <- train(totalbox ~ ., data = train,
                      trControl = control,
                      preProcess = c("center","scale"),
                      tuneLength = 10, ntree = 500, metric = "RMSE", method = "rf", importance = TRUE)


# plot variable importance
varImpPlot(final.forest$finalModel, type = 1, main = "Variable Importance")
```




```r
# test on test dataset
tuned_pred <- predict(final.forest, test)
results_caret <- postResample(pred=tuned_pred, obs=test$totalbox)
results_caret
```

```
##         RMSE     Rsquared          MAE
## 2.906332e+07 9.261456e-01 1.713169e+07
```

```
if (results_caret[["RMSE"]] < lowest_RMSE) {
  final_model <- final.forest
}

test$final_pred <- predict(final_model, test)
test$error <- sqrt((test$final_pred - test$totalbox)^2)
final_results <- postResample(pred=test$final_pred, obs=test$totalbox)

test <- test %>% mutate(binned_totalbox =
                     case_when(totalbox <= 10000000 ~ 10,
                               10000000 < totalbox & totalbox <= 50000000 ~ 50,
                               50000000 < totalbox & totalbox <= 100000000 ~ 100,
                               100000000 < totalbox & totalbox <= 500000000 ~ 500,
                               totalbox > 500000000 ~ 900,)
)
test$binned_totalbox <- factor(test$binned_totalbox, levels = c(10,50,100,500,900),
                               labels = c("<10M","10-50M","50-100M","100-500M",">500M"))

ggplot(test, aes(binned_totalbox, error)) +
  geom_bar(stat = "summary",
           fun = "mean") +
  ggtitle("Average Error vs. Observed Total Box") +
  xlab("Observed Total Box") +
  ylab("Average Error of Predictions")
```



Lastly, I use the best model to predict *totalbox* for the post-covid titles.

```r
# make final predictions
post_covid_titles$predicted_totalbox <- predict(final_model, post_covid_titles)
post_covid_titles <- cbind(post_covid_ids,post_covid_titles)
write.csv(post_covid_titles,"predicted_data.csv", row.names = FALSE)
```

Results

As stated above, I chose RandomForest because it does not depend on the distribution or scale of the predictor variables and is a good "out of the box" model for numerical predictions. I tuned it both with a manual grid search and the automatic tuning that caret provides to choose the best hyperparameters for the final model. I considered dropping more variables from the model and briefly tested other numerical prediction approaches, but saw worse results for RMSE and Rsquared.

The best model was the one tuned by caret and has an R-Squared of 0.9261456, meaning that about 93% of the variance in *totalbox* can be explained from the model. The overall RMSE of 2.9063316×10^7 means that on average, this model will have an error of about $29.06M. We can see from the plot of error for different levels of *totalbox* that the error of predictions grows as revenue increases. A possible explanation of this is that because there are more titles that have relatively low revenue, the model is biased towards them and underfit for high revenue titles. Additionally, there are relatively few complete rows and many of the variables are imbalanced, so it's possible that this model might become much more precise if trained on more data.

Conclusion

Knowing which variables lead to higher and lower box office revenue can be leveraged to budget and set expectations for the performance of new projects. From the Variable Importance plots, the *Interest_index* is by far the most important variable across all models. The last Variable Importance plot shows how different levels of factor variables contribute to the predictions. *Imputed_rotten_tomatoes* score is the third most important variable and *Imputed_gradeA-* is the 6th most important variable, indicating that the overall interest and perception of the movie quality are strong indicators of its box office performance. This insight could drive more emphasis on advertising (which might increase the Interest_index), or creating content that's quality is scored well. We can also see that R and PG-13 rated movies, as well as dramatic and animated movies have more influence on *totalbox* than other ratings and genres. When taken together with the findings from the bivariate analysis on mean of *totalbox* by level of the factor variables, we can see that the effect of R and PG-13 movies as well as dramatic movies tend to decrease *totalbox* and animated movies tend to increase *totalbox*. Using this analysis, the box office revenue of a new movie can be compared to that of historical movies that have similar attributes to fairly judge the performance of the new movie.