Box Office CART Exploration

David Miranda, Ian O'Keeffe, Ainsley McCutcheon

```
knitr::opts_chunk$set(echo = TRUE)

# Load any packages, if any, that you use as part of your answers here
# For example:

library(rpart)
library(rpart.plot)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggpubr)
```

The Box Office Collections Dataset

This dataset was obtained from kaggle (https://www.kaggle.com/datasets/anotherbadcode/boxofficecollections)

We'll be using the Box.Office.Collection variable as the outcome for this demonstration.

Other information available in this dataset is:

Numeric Values: Score (Rotten Tomatoes Critic Score), Adjusted.Score (Rotten Tomatoes Adjusted Critic Score), IMDB.Rating (IMDB User Rating), metascore (Metacritic Score), Imdb_genre, time_minute, Votes, Year

Categorical Values: Director, Cast, Consensus

We'll be using Year, Score, IMDB.Rating, Imdb_genre, and time_minute in this example.

```
boxoffice <- read.csv("BoxOfficeCollections.csv", header=TRUE, sep=",")
boxoffice <- drop_na(boxoffice)
str(boxoffice)
```

```
## 'data.frame':    759 obs. of  13 variables:
##  $ Movie               : chr  "Hot Rod" "Game Night" "The First Wives Club" "Scary Movie" ...
##  $ Year                : int  2007 2018 1996 2000 2018 1993 2012 2009 1988 2001 ...
##  $ Score               : int  39 85 49 52 84 72 81 83 97 37 ...
##  $ Adjusted.Score      : num  42.9 99.8 53.2 55 96.9 ...
##  $ Director            : chr  "Akiva Schaffer" "John Francis Daley" "Hugh Wilson" "Keenen Ivory Wayans" ...
```

```
##  $ Cast                 : chr  "Andy Samberg, Jorma Taccone, Bill Hader, Danny R. McBride" "Jason Ba
##  $ Consensus            : chr  "For Rod Kimball (Andy Samberg), performing stunts is a way of life, 
##  $ Box.Office.Collection: num  1.44e+07 1.17e+08 1.81e+08 2.77e+08 9.45e+07 ...
##  $ Imdb_genre           : chr  "Comedy" "Comedy" "Comedy" "Comedy" ...
##  $ IMDB.Rating          : num  6.7 6.9 6.4 6.2 6.2 7 7.1 7.2 7.3 6.5 ...
##  $ metascore            : num  43 66 58 48 69 53 66 68 73 42 ...
##  $ time_minute          : num  88 100 103 88 102 125 112 97 104 97 ...
##  $ Votes                : num  84956 229292 48413 254927 78498 ...
```

Data Fixing

We'll be using rpart to create the regression tree which can handle character or factor variable types when using categorical data. This line of code is not necessary and is just included to demonstrate that the following code will run for either data type. Additionally, the box office collections value is converted to millions of dollars for readability.

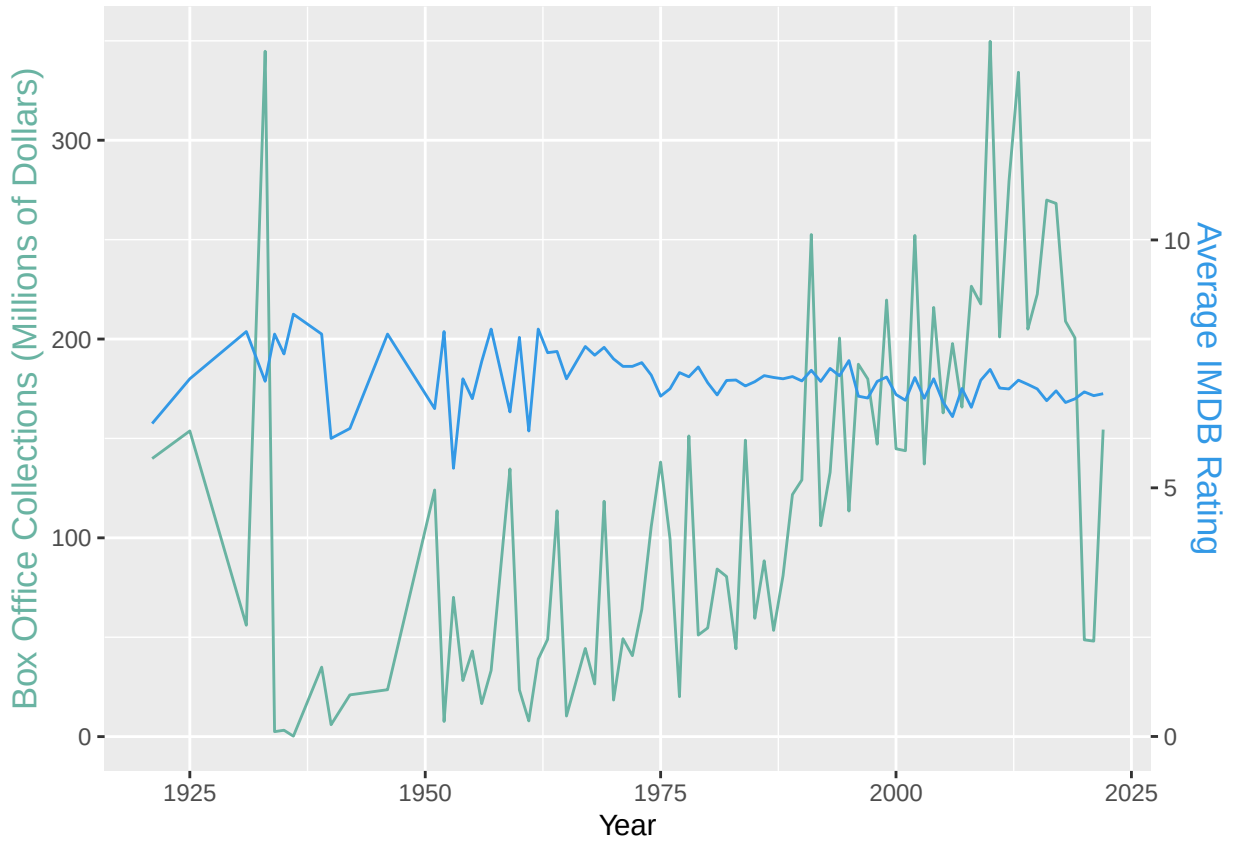
```
boxoffice$Imdb_genre <- as.factor(boxoffice$Imdb_genre)
boxoffice$Box.Office.Collection.Millions <- boxoffice$Box.Office.Collection/1000000
```

Data Exploration

Before making a regression tree to this dataset, let's take a look at the IMDB ratings and the Box Office Collections by year. You can see that while there is an increase in average collections by year, it doesn't visually appear to be linear. There are definitely some outliers (for instance King Kong in 1933), and there are very few movies included for earlier years. It's also noteworthy that average IMDB ratings do not follow the same trend and are fairly consistent.

```
scoreColor <- rgb(0.2, 0.6, 0.9, 1)
boxOfficeColor <- "#69b3a2"

boxoffice.yearly <- boxoffice %>%
  group_by(Year) %>%
  summarize(Sales = mean(Box.Office.Collection.Millions, na.rm=TRUE), Scores = mean(IMDB.Rating, na.rm=
scale = 25
ggplot(boxoffice.yearly, aes(Year)) +
  geom_line(aes(y = Sales), colour=boxOfficeColor) +
  geom_line(aes(y = Scores * scale), colour=scoreColor) +
  scale_y_continuous(
    name = "Box Office Collections (Millions of Dollars)",
    sec.axis = sec_axis(~./scale, name="Average IMDB Rating")
  ) +
  theme(
    axis.title.y = element_text(color = boxOfficeColor, size=13),
    axis.title.y.right = element_text(color = scoreColor, size=13)
  )
```



Making a Regression Tree

To make a regression tree we will be using the rpart package. The rpart() function used below has a very similar structure to the lm/glm functions that we've been using. Note that the method can be excluded and the function will examine the outcome variable to determine if a regression or classification tree should be created. Specifying method = "anova" ensures that a regression tree is being made.

The output of printcp() is displayed below which shows the creation of the tree. In the last table you can find the CP (complexity parameter) which is the r-squared value at each stage. The r-squared value is one of the determining factors for when to stop expanding the tree. Looking at the output you can see that this occurs once the r-squared is less than 0.01 by default. The n-split shows the number of splits (nodes) that have been made. Also included are 3 measure of error: rel error, xerror (cross-validated error), and xstd (cross-validated standard error).

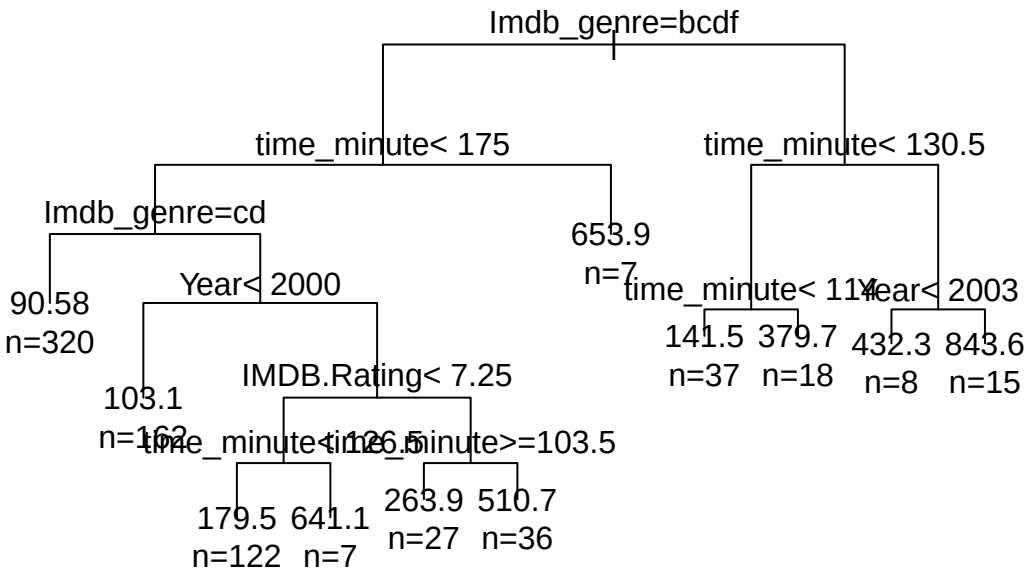
From here you can display the regression tree with plot(). Included are some changes to make the tree slightly more readable, however it is still effectively unreadable.

```
set.seed(39789)
fit <- rpart(Box.Office.Collection.Millions ~ Year + Score + IMDB.Rating + Imdb_genre + time_minute, da
printcp(fit)
```

```
##
## Regression tree:
## rpart(formula = Box.Office.Collection.Millions ~ Year + Score +
##     IMDB.Rating + Imdb_genre + time_minute, data = boxoffice,
##     method = "anova")
##
```

```
## Variables actually used in tree construction:
## [1] IMDB.Rating Imdb_genre  time_minute Year
##
## Root node error: 49660305/759 = 65429
##
## n= 759
##
##         CP nsplit rel error  xerror    xstd
## 1 0.069245      0   1.00000 1.00399 0.16401
## 2 0.040569      2   0.86151 0.94936 0.16243
## 3 0.034215      5   0.73980 0.86357 0.16310
## 4 0.028399      6   0.70559 0.83944 0.16258
## 5 0.018917      7   0.67719 0.84711 0.16319
## 6 0.017778      8   0.65827 0.85318 0.16109
## 7 0.013844      9   0.64049 0.83236 0.16117
## 8 0.010000     10   0.62665 0.84845 0.16229
```

```
par(xpd = NA)
plot(fit)
text(fit, use.n = TRUE)
```



Pruning

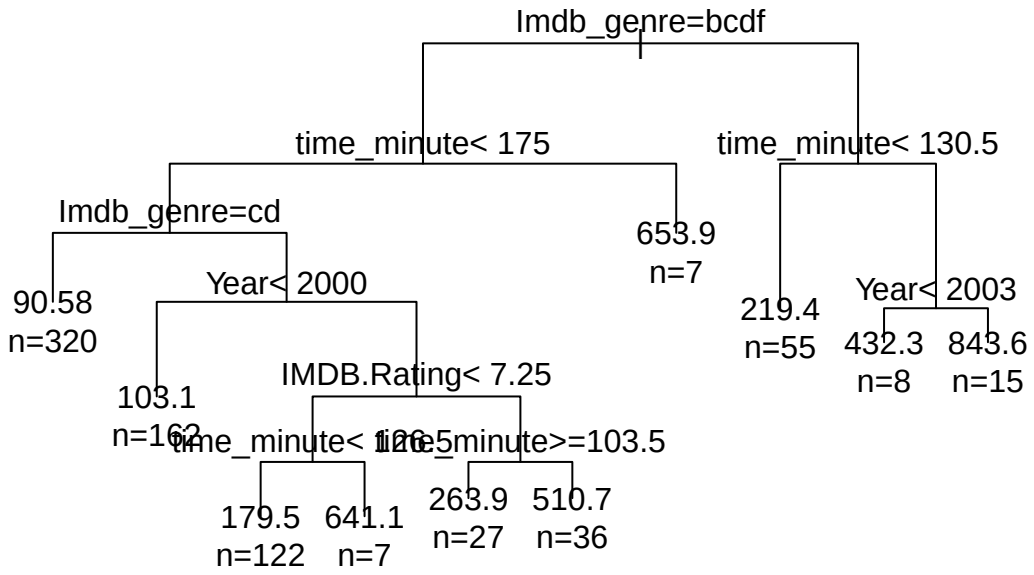
As seen above, our current tree is not particularly usable right now. The last step for making a regression tree is "pruning" the tree back to a less complex one. In this example we've used the "xerror" value that we looked at previously to determine which tree is best. Look back at the printcp() output from above, we can

see that the lowest xerror value was from the tree with 7 nodes which is displayed below.

```
bestcp <- fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"]
tree.pruned <- print(fit, cp = bestcp)
```

```
## n= 759
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 759 49660300.0 171.74100
##    2) Imdb_genre=Comedy,Drama,Horror,Thriller 681 37235550.0 150.02890
##      4) time_minute< 175 674 29407980.0 144.79540
##        8) Imdb_genre=Drama,Horror 320  5290288.0  90.57821 *
##        9) Imdb_genre=Comedy,Thriller 354 22326750.0 193.80530
##         18) Year< 2000.5 162  1891202.0 103.10500 *
##         19) Year>=2000.5 192 17978390.0 270.33360
##           38) IMDB.Rating< 7.25 129  7350706.0 204.59200
##             76) time_minute< 126.5 122  4199032.0 179.54640 *
##             77) time_minute>=126.5 7  1741355.0 641.10240 *
##           39) IMDB.Rating>=7.25 63  8928538.0 404.94720
##             78) time_minute>=103.5 27  2977439.0 263.94440 *
##             79) time_minute< 103.5 36  5011684.0 510.69930 *
##      5) time_minute>=175 7  6031600.0 653.94450 *
##    3) Imdb_genre=Adventure,Sci-Fi 78  9300856.0 361.30400
##      6) time_minute< 130.5 55  2904813.0 219.44520 *
##      7) time_minute>=130.5 23  2642492.0 700.53160
##       14) Year< 2003 8   643087.8 432.25690 *
##       15) Year>=2003 15  1116556.0 843.61150 *
```

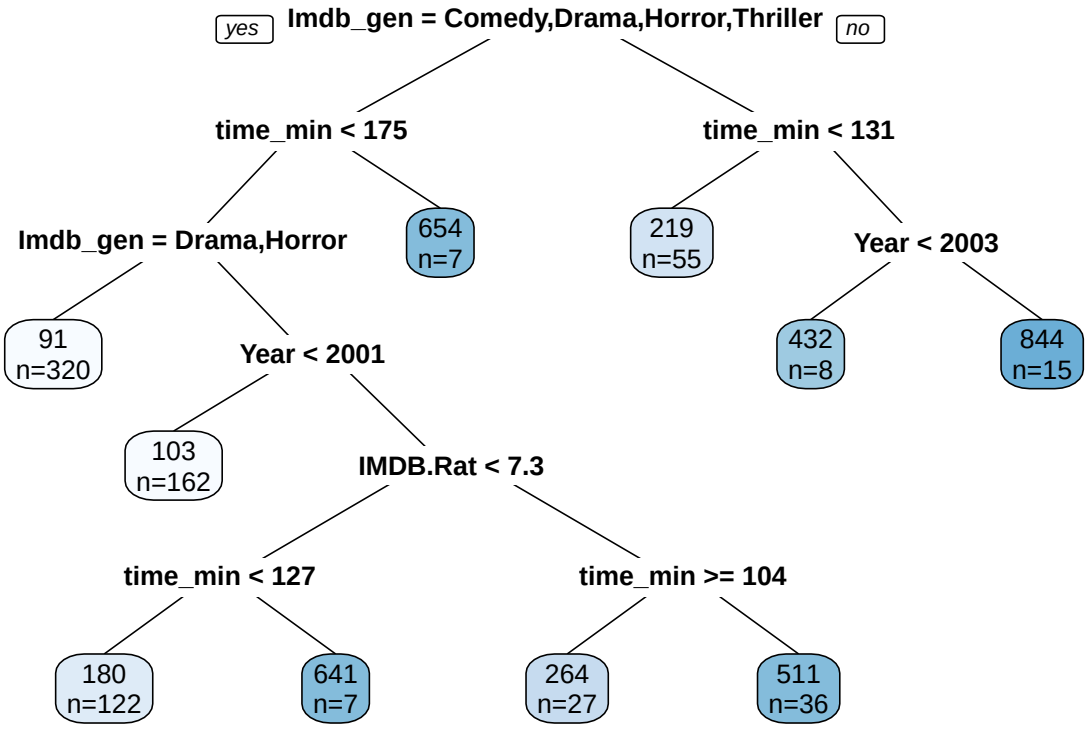
```
par(xpd = NA)
plot(tree.pruned)
text(tree.pruned, use.n = TRUE)
```



Better Plots

So we've pruned back our tree to be a bit more reasonable, but out plot is still not very readable. To improve this we can use the prp() function from the rpart.plot package to easily improve the readability of the plot. Below is a fairly basic set of options to improve the visualization. You can make many tweaks to this plot to better organize you tree. Look at http://www.milbo.org/rpart-plot/prp.pdf to look at all the options for customizing the plot.

```
prp(tree.pruned, faclen = 0, cex = 0.8, extra = 1, box.palette = "auto")
```



Reading the Final Regression Tree

As mentioned in the CART handout, this tree is read from top to bottom. At the top you have the statement that the IMDB genre is comedy, drama, horror or thriller. If this statement is true for a movie, take the left path. If it is false take the right path. Repeat this process going down the tree until you reach leaf which shows and estimate for box office collections. The n listed underneath the estimate for each leaf is the number of observations that met the criteria in training data.

For example, suppose you have a movie from 2008, a length of 180 minutes, an IMDB rating of 8.0, and a genre of Sci-fi. Going through the tree, you would go right at the first node since the movie is not a comedy, drama, horror or thriller. The next node splits based on the length of the movie. Since this movie is longer than 131 minutes we go left again. You would go left again at the next node since the year is after 2003. Finally you reach a leaf with the estimate of $844,000,000.

Conclusion

Based on the observations included in this dataset, you will note that not all the input variables were used in the final regression tree. According to this model the Rotten Tomatoes critic score was not a useful predictor for compared to IMDB user ratings, length of the movie, and genre. Generally speaking, comedy, drama, horror and thriller movies made less, as did movies before the early 2000's. Additionally longer movies had better sales with the specific exception for more recent comedies and thriller that have high user ratings.

It's also worth noting that groups with the lowest predicted box office collections had notably more observations in this dataset when compared to the higher predictions. The nature of this data makes this unsurprising given that movies that make hundreds of millions are rare, but it makes the features of these movies more influential in the model. Therefore I would take the predictions on the high end with a large grain of salt.