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(skimr)
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 in the dataset be using is:

Numeric Values: Score (Rotten Tomatoes Critic Score) Adjusted.Score (Rotten Tomatoes Adjusted Critic Score) IMDB.Rating Imdb__genre metascore 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 Waya
##  $ 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 do 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.

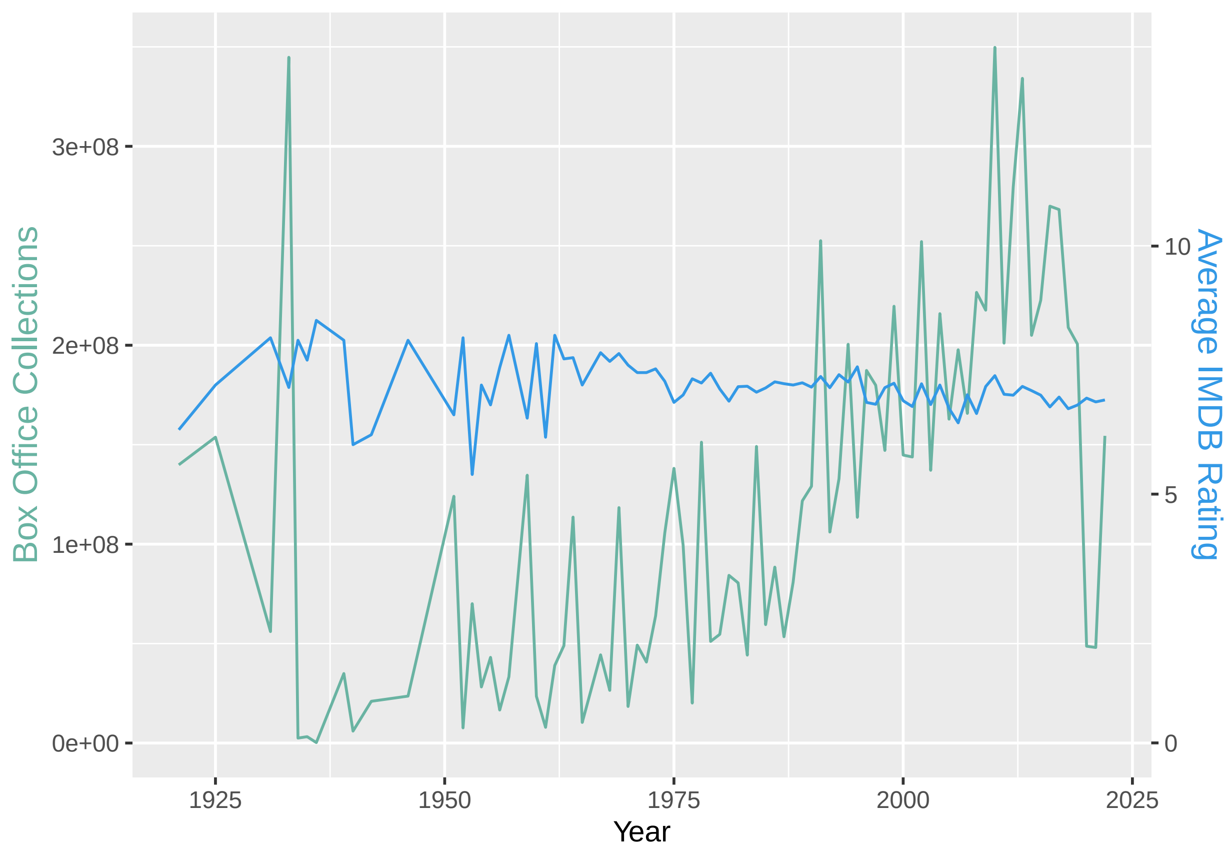
```
boxoffice$Imdb_genre <- as.factor(boxoffice$Imdb_genre)
```

Mean sales

Before we look applying a regression tree to this dataset, lets 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 outlier (for instance King Kong in 1933), and there are very few movies included for older years. It's also noteworthy that average IMDB ratings do 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, na.rm=TRUE), Scores = mean(IMDB.Rating, na.rm=TRUE))
scale = 25000000
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",
    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)
  )
```



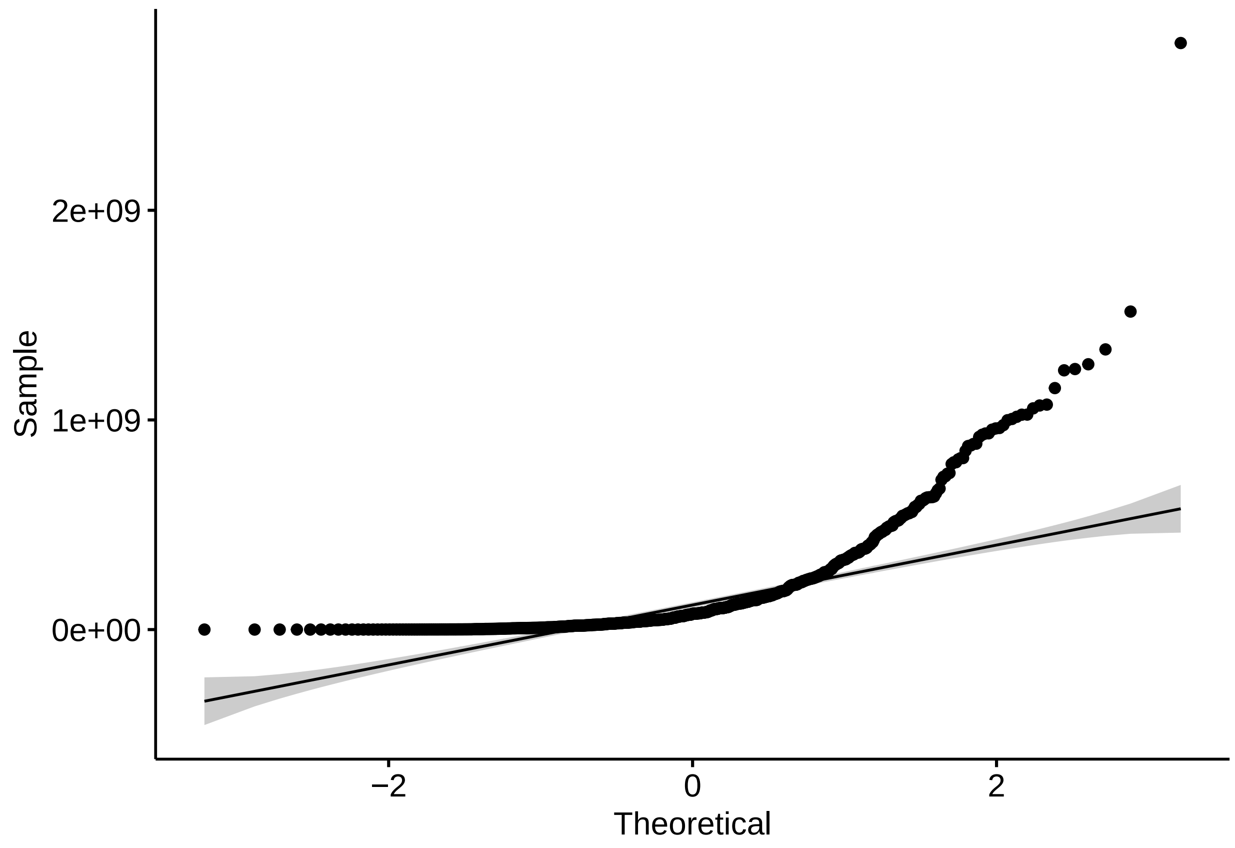
Data Requirements

The only requirement for a Regression Tree is that the outcome variable is numeric and that all the input variables are either numeric or categorical. In fact the the outcome variable that we are looking at is not normally distributed as shown below.

```
shapiro.test(boxoffice$Box.Office.Collection)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  boxoffice$Box.Office.Collection
## W = 0.65552, p-value < 2.2e-16
```

```
ggqqplot(boxoffice$Box.Office.Collection)
```



Forward and Backward Selection

```
#null.boxoffice.model <- lm(Box.Office.Collection ~ 1, data=boxoffice)
#full.boxoffice.model <- lm(Box.Office.Collection ~ Adjusted.Score + Score + IMDB.Rating + metascore + 
#
#null.boxoffice.formula <- as.formula("Box.Office.Collection ~ 1")
#full.boxoffice.formula <- as.formula("Box.Office.Collection ~ Adjusted.Score + Score + IMDB.Rating + m
#
#boxoffice.fwd.selection <- step(null.boxoffice.model, scope=full.boxoffice.formula, direction="forward
#summary(boxoffice.fwd.selection)
#
```

Backwards Model Selection

```
#boxoffice.bwd.selection <- step(full.boxoffice.model, scope=null.boxoffice.formula, direction="backwar
#summary(boxoffice.bwd.selection)
```

Cross Validation

Maybe if we can get it to work.

```
# tc_kfold <- trainControl(method = "cv", number = 5,
#                          savePredictions = TRUE, classProbs = TRUE)
#
# all_x <- boxoffice %>%
#  select(Adjusted.Score, Score, metascore, time_minute, Votes) %>%
#  as.data.frame()
#
# all_y <- boxoffice %>%
#  pull(Box.Office.Collection)
#
# set.seed(20140102)
# m_kfold <- train(x = all_x, y = all_y,
#                 method = "glm", family = "binomial",
#                 trControl = tc_kfold)
#
# m_kfold$results
```

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.

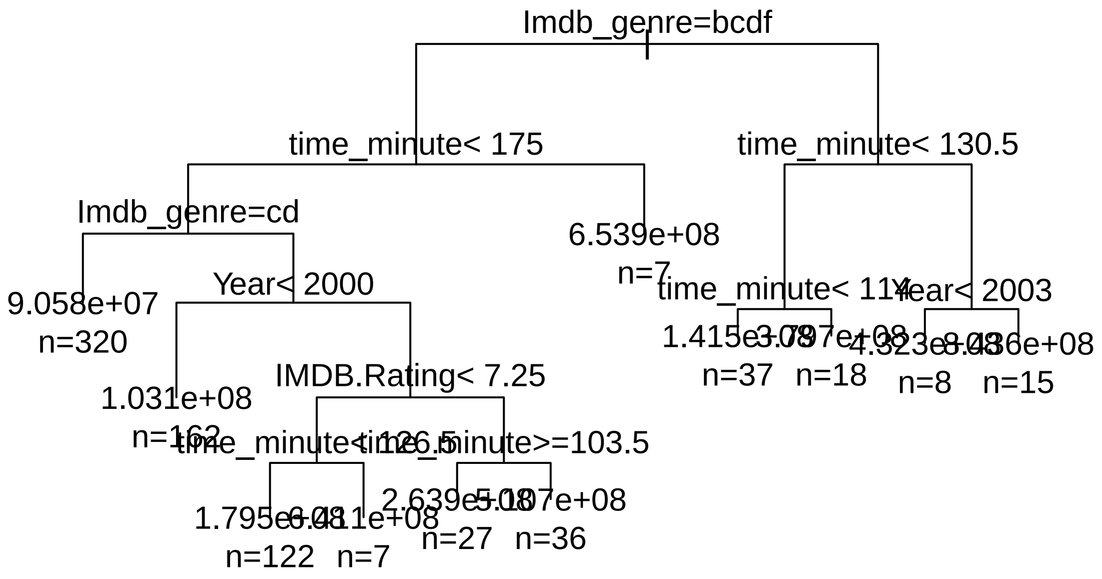
The output of summary() is shown below which shows the creation of the tree. In the last table you can find the CP (complexity parameter) which is the r-squared value at each stage. The r-squared value is one of the determining factors for when to stop and looking at the output you can see that this 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.

```
fit <- rpart(Box.Office.Collection ~ Year + Score + IMDB.Rating + Imdb_genre + time_minute, data = boxo
printcp(fit)
```

```
##
## Regression tree:
## rpart(formula = Box.Office.Collection ~ 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: 4.966e+19/759 = 6.5429e+16
##
## n= 759
##
##          CP nsplit rel error  xerror    xstd
## 1 0.069245      0   1.00000 1.00404 0.16388
## 2 0.040569      2   0.86151 0.91683 0.16235
## 3 0.034215      5   0.73980 0.85437 0.16201
## 4 0.028399      6   0.70559 0.86004 0.16258
## 5 0.018917      7   0.67719 0.85557 0.16241
## 6 0.017778      8   0.65827 0.87510 0.16511
## 7 0.013844      9   0.64049 0.88036 0.16532
## 8 0.010000     10   0.62665 0.87449 0.16531
```

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



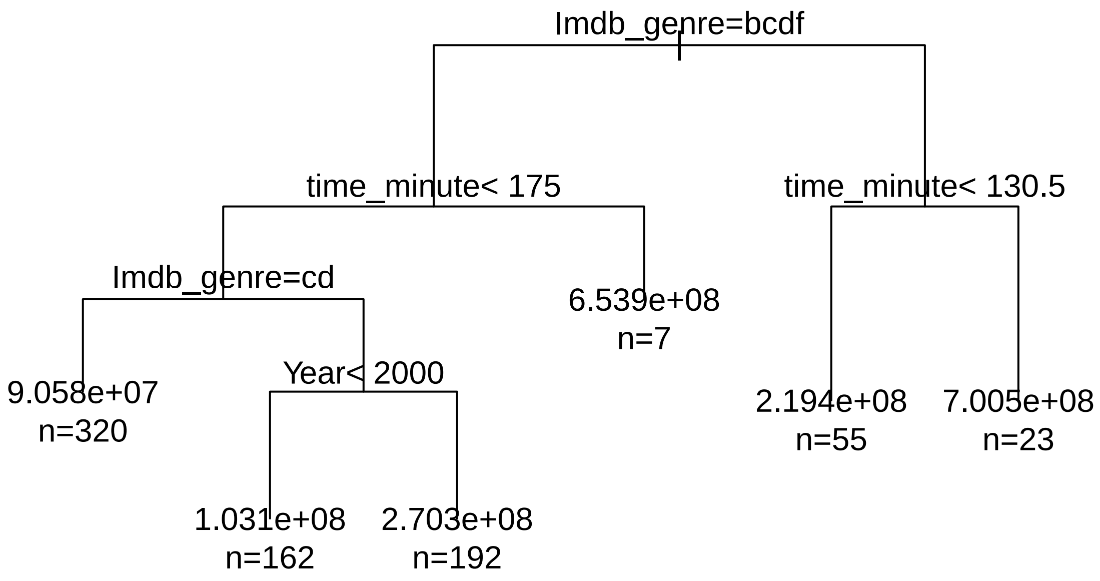
Pruning

As seen above, our current tree is not particularly useable 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 at back at the summary from above, we can see that the lowest xerror value was from the tree with 7 nodes which is diplayed below.

```r
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 4.966030e+19 171741000
##    2) Imdb_genre=Comedy,Drama,Horror,Thriller 681 3.723555e+19 150028900
##      4) time_minute< 175 674 2.940798e+19 144795400
##        8) Imdb_genre=Drama,Horror 320 5.290288e+18  90578210 *
##        9) Imdb_genre=Comedy,Thriller 354 2.232675e+19 193805300
##         18) Year< 2000.5 162 1.891202e+18 103105000 *
##         19) Year>=2000.5 192 1.797839e+19 270333600 *
##      5) time_minute>=175 7 6.031600e+18 653944500 *
##    3) Imdb_genre=Adventure,Sci-Fi 78 9.300856e+18 361304000
##      6) time_minute< 130.5 55 2.904813e+18 219445200 *
##      7) time_minute>=130.5 23 2.642492e+18 700531600 *
```

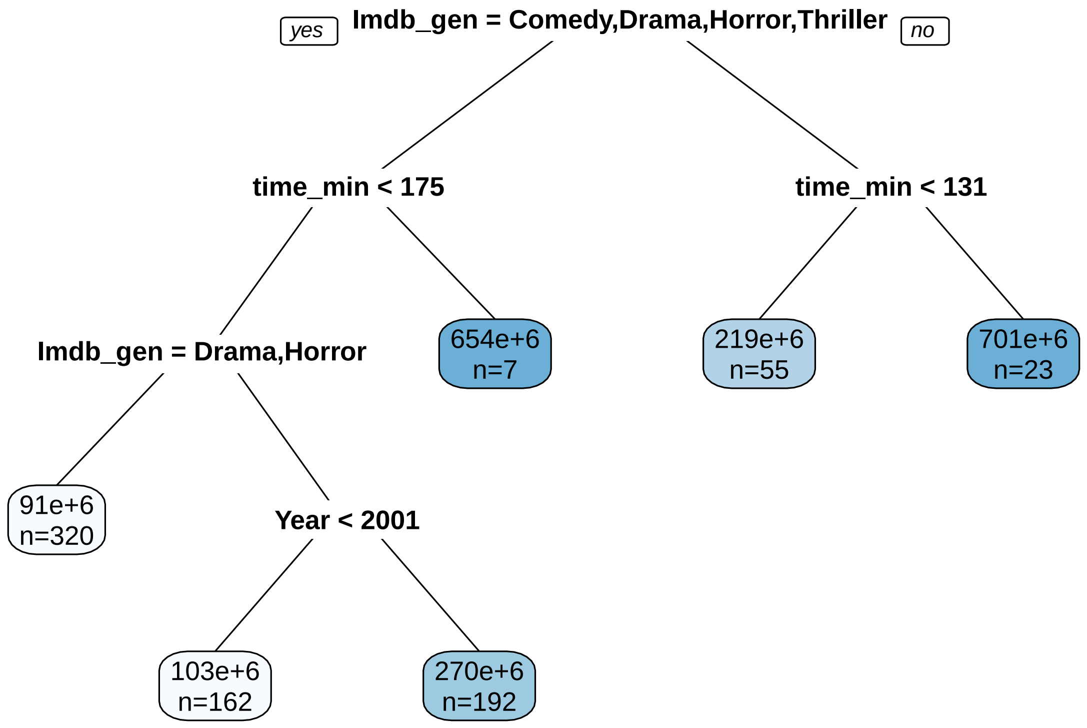
```
par(xpd = NA)
plot(tree.pruned)
text(tree.pruned, use.n = TRUE)
```



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 https://rpubs.com/minma/cart_with_rpart to look at all the options for customizing the plot.

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


yes
Imdb_gen = Comedy,Drama,Horror,Thriller
no
time_min < 175
time_min < 131
Imdb_gen = Drama,Horror
654e+6
n=7
219e+6
n=55
701e+6
n=23
91e+6
n=320
Year < 2001
103e+6
n=162
270e+6
n=192