Exploratory Data Analysis Project

Video Games Sales Dataset

Luis Otero

Description of Video Games Sales Dataset

Link to the Dataset.

For my project, I selected a dataset that contains a list of video games with sales greater than 100,000 copies. It is titled "Video_Games_Sales" and it consists of an enormous dataset categorizing video game sales by video game titles, the platforms on which they are played, the video game publisher, the genre, and the year of release. The dataset contains five columns pertaining to sales revenue the game has produced in millions of dollars, they include "NA_Sales" (sales in North America), "EU_Sales" (sales in Europe), "JP_Sales" (sales in Japan), "Other_Sales" (Sales in the rest of the world), and Global_Sales (total world-wide sales). The variables "Critic_Rating" (aggregate score compiled by Metacritic staff), "Critic_Count" (the number of critics assigning a critic score to the game), "User_Score" (score by Metacritic's subscribers), and User_Count (aggregate score by users) are attributes based on the number of critics and subscribers who gave a score to the game on the website Metacritic. The last two attributes are the video game developer, and the content ratings given by the Entertainment Software Rating Board – making a total of 16 attributes.

A breakdown of the attributes follows:

Attribute	Type of Variable
Name	Categorical
Platform	Categorical
Year_of_Release	Categorical
Genre	Categorical
Publisher	Categorical
NA_Sales	Numerical
EU_Sales	Numerical
JP_Sales	Numerical
Other_Sales	Numerical
Global_Sales	Numerical
Critic_Score	Numerical
Critic_Count	Numerical
User_Score	Numerical
User_Count	Numerical
Developer	Categorical
Rating	Categorical

The dataset consists of a total of 16,719 instances of video game titles, broken down according to the attributes listed above. The dataset does contain several missing values. Specifically, Year_of_Release, Critic_Score, Critic_Count, User_Score, User_Count, and Rating attributes are missing on some of the video game entries. However, I fixed that. I dropped the NAs and missing values in the Rating and Year_of_Release variables, and then replaced the missing values in the User_Score and User_Count columns with the averages of the column.

My main analytics question: What Factors Determine Higher Video Game Sales? To answer this question, we utilized R to perform an exploratory data analysis and test different statistical models on the dataset. I chose R over Python since R is a programming language that is used for statistical analysis while Python provides a more general approach to data science. Furthermore, R offers a vast variety of tools for specialized analytical work and to communicate results.

What Factors Drives Video Game Sales?

Loading Dataset

```
setwd('/Users/luiscarlosotero/Documents/2019-2020/Data_Science')

Video_Games_DF <- read.csv(here::here("Dataset", "Video_Games_Sales.csv"),
                           stringsAsFactors = FALSE)
```

Names of Variables and Dimensions

```
#Names of Attributes
names(Video_Games_DF)
```

```
##  [1] "Name"            "Platform"        "Year_of_Release"
##  [4] "Genre"           "Publisher"       "NA_Sales"
##  [7] "EU_Sales"        "JP_Sales"        "Other_Sales"
## [10] "Global_Sales"    "Critic_Score"    "Critic_Count"
## [13] "User_Score"      "User_Count"      "Developer"
## [16] "Rating"
```

```
#Dimensions of Dataset
dim(Video_Games_DF)
```

```
## [1] 16719    16
```

The dataset contains 16719 rows and 16 columns

Cleaning Dataset/Data Transformations

```
#Dropping NAs from Critic_Score
Video_Games_DF <- Video_Games_DF %>% drop_na(Critic_Score)

#Dropping Missing Values from Rating and Year_of_Release
Video_Games_DF <- Video_Games_DF %>% filter(Rating != "",
                                           Year_of_Release != "N/A")

#Filling in NAs with the Column Averages
list_na <- colnames(Video_Games_DF)[ apply(Video_Games_DF, 2, anyNA) ]

average_missing <- apply(Video_Games_DF[,colnames(Video_Games_DF)
                                        %in% list_na],
      2,
      mean,
      na.rm =  TRUE)

Video_Games_DF <- Video_Games_DF %>%
   mutate(User_Score  = ifelse(is.na(User_Score), average_missing[1],
                               User_Score) %>% round(1),
```

```
          User_Count = ifelse(is.na(User_Count), average_missing[2],
                              User_Count) %>% round(0))

#Changing Variable
Video_Games_DF <- Video_Games_DF %>% mutate(User_Score = User_Score * 10)

#Adding Variable "Total_Count"
Video_Games_DF <- Video_Games_DF %>% mutate(Total_Count =
                                              Critic_Count + User_Count)

#Adding Column "Metascores"
Video_Games_DF <- Video_Games_DF %>%
  mutate(Metascores = case_when(Critic_Score >= 90 ~ "Universal Acclaim",
                                Critic_Score >= 75 ~ "Generally Favorable",
                                Critic_Score >= 50 ~ "Mixed or Average",
                                Critic_Score >= 20 ~ "Generally Unfavorable",
                                TRUE ~ "Just Plain Awful")) %>%
  mutate(Metascores = factor(Metascores,
                                    levels = c("Universal Acclaim",
                                               "Generally Favorable",
                                               "Mixed or Average",
                                               "Generally Unfavorable",
                                               "Just Plain Awful")))

#Creating Binary Variable (Sold Millions)
Video_Games_DF <- Video_Games_DF %>% mutate(Millionaire =
                                             ifelse(Global_Sales > 1.0, 1, 0))

#Erasing One Outlier
Video_Games_DF <- Video_Games_DF %>% filter(NA_Sales < 40.0)
```

This block of code displays the functions used to drop the NAs and missing values from Critic_Score, Rating, and Year_of_Release. After dropping all those missing values, the dataset went from 16719 observations to 7902 observations. However, the User_Count and User_Score variables still had missing values; in order to deal with those, the averages of the two columns were calculated and used to fill in the NAs. I then added columns to the dataset including a column that takes the sum of Critic_Count and User_Count. The last column added was "Metascores," which classifies a video game as either "Universal Acclaim," "Generally Favorable," "Mixed or Average," "Generally Favorable," or "Just Plain Awful," depending on the critic score of the game. A column called "Millionaire" was created to see if the game made at least a million dollars in global sales revenue. In the end, the dataset contains 19 columns.

Splitting Dataset into Training and Test Sets

```
num_rows <- nrow(Video_Games_DF)

set.seed(1861)
train_idx <- sample(1:num_rows, floor(0.8*nrow(Video_Games_DF)))
Video_Games_Train <- Video_Games_DF %>% slice(train_idx)
Video_Games_Test <- Video_Games_DF %>% slice(-train_idx)
```

Dataset Summary Statistics

```
#Summary Stats
summary(Video_Games_Train)
```

```
##      Name             Platform         Year_of_Release
##  Length:6320        Length:6320        Length:6320
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##     Genre             Publisher            NA_Sales          EU_Sales
##  Length:6320        Length:6320        Min.   : 0.0000   Min.   : 0.000
##  Class :character   Class :character   1st Qu.: 0.0500   1st Qu.: 0.010
##  Mode  :character   Mode  :character   Median : 0.1300   Median : 0.050
##                                        Mean   : 0.3502   Mean   : 0.204
##                                        3rd Qu.: 0.3500   3rd Qu.: 0.180
##                                        Max.   :15.6800   Max.   :12.760
##     JP_Sales         Other_Sales        Global_Sales      Critic_Score
##  Min.   :0.00000   Min.   : 0.00000   Min.   : 0.0100   Min.   :13.0
##  1st Qu.:0.00000   1st Qu.: 0.01000   1st Qu.: 0.1000   1st Qu.:60.0
##  Median :0.00000   Median : 0.02000   Median : 0.2500   Median :71.0
##  Mean   :0.05532   Mean   : 0.07059   Mean   : 0.6804   Mean   :68.9
##  3rd Qu.:0.01000   3rd Qu.: 0.06000   3rd Qu.: 0.6600   3rd Qu.:79.0
##  Max.   :6.50000   Max.   :10.57000   Max.   :35.5200   Max.   :98.0
##   Critic_Count      User_Score        User_Count        Developer
##  Min.   :  3.00   Min.   : 5.00   Min.   :    4.0   Length:6320
##  1st Qu.: 12.00   1st Qu.:67.00   1st Qu.:   13.0   Class :character
##  Median : 22.00   Median :72.00   Median :   37.0   Mode  :character
##  Mean   : 26.46   Mean   :71.78   Mean   :  175.2
##  3rd Qu.: 37.00   3rd Qu.:81.00   3rd Qu.:  175.0
```

```
##  Max.   :107.00   Max.   :96.00   Max.   :10665.0
##     Rating           Total_Count                      Metascores
##  Length:6320        Min.   :    8.0   Universal Acclaim    : 256
##  Class :character   1st Qu.:   32.0   Generally Favorable  :2207
##  Mode  :character   Median :   67.0   Mixed or Average     :3220
##                     Mean   :  201.6   Generally Unfavorable: 629
##                     3rd Qu.:  182.0   Just Plain Awful     :   8
##                     Max.   :10697.0
##   Millionaire
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.1669
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

```
#First 6 Rows in Training Dataset
head(Video_Games_Train)
```

```
##                                 Name Platform Year_of_Release   Genre
## 1        Dynasty Warriors 8: Empires      PSV            2015  Action
## 2 Max Payne 2: The Fall of Max Payne       XB            2003 Shooter
## 3                      Mystic Heroes      PS2            2002  Action
## 4                   Enter the Matrix       XB            2003  Action
## 5                    Resident Evil 6     X360            2012 Shooter
## 6                      Tetris Worlds      PS2            2002  Puzzle
##              Publisher NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales
## 1           Tecmo Koei     0.00     0.00     0.02        0.00         0.02
## 2 Take-Two Interactive     0.47     0.15     0.00        0.02         0.64
## 3           Tecmo Koei     0.03     0.02     0.00        0.01         0.06
## 4                Atari     0.72     0.43     0.01        0.04         1.20
## 5               Capcom     1.12     0.60     0.07        0.16         1.95
## 6                  THQ     1.11     0.71     0.00        0.27         2.08
##   Critic_Score Critic_Count User_Score User_Count            Developer
## 1           70           15         74          9          Omega Force
## 2           84           34         84         49 Remedy Entertainment
## 3           67            7         86         14                 Koei
## 4           65           33         71         75  Shiny Entertainment
## 5           67           71         50       1407               Capcom
## 6           44            7         62         11 Blue Planet Software
##   Rating Total_Count           Metascores Millionaire
## 1      T          24      Mixed or Average           0
## 2      M          83   Generally Favorable           0
## 3      T          21      Mixed or Average           0
```

```
## 4      T         108      Mixed or Average           1
## 5      M        1478      Mixed or Average           1
## 6      E          18 Generally Unfavorable           1
```

```
#Averages and Variances in Global Sales by Genre
Video_Games_Train %>% group_by(Genre) %>%
  summarize(Avg_Global_Sales_Genre = mean(Global_Sales, na.rm = TRUE),
            Var_Global_Sales_Genre = var(Global_Sales, na.rm = TRUE),
            Count = n())
```

```
## # A tibble: 12 x 4
##    Genre        Avg_Global_Sales_Genre Var_Global_Sales_Genre Count
##    <chr>                         <dbl>                  <dbl> <int>
##  1 Action                        0.682                  2.20   1440
##  2 Adventure                     0.325                  0.390   242
##  3 Fighting                      0.612                  1.20    321
##  4 Misc                          0.798                  2.83    397
##  5 Platform                      0.718                  3.36    402
##  6 Puzzle                        0.497                  1.72    191
##  7 Racing                        0.670                  3.83    564
##  8 Role-Playing                  0.691                  1.44    590
##  9 Shooter                       0.924                  3.48    727
## 10 Simulation                    0.653                  1.56    283
## 11 Sports                        0.689                  2.78    931
## 12 Strategy                      0.254                  0.204   232
```

The top three genres that have the highest average sales are: shooter, misc., and platform. The top three genres three genres that have the highest variance are racing, shooter, and platform. The type of video game that has the most count is Action.

```
#Averages and Variances in Global Sales by Platform
Video_Games_Train %>% group_by(Platform) %>%
  summarize(Avg_Global_Sales_Platform = mean(Global_Sales, na.rm = TRUE),
            Var_Global_Sales_Platform = var(Global_Sales, na.rm = TRUE),
            Count = n())
```

```
## # A tibble: 17 x 4
##    Platform Avg_Global_Sales_Platform Var_Global_Sales_Platform Count
##    <chr>                        <dbl>                     <dbl> <int>
##  1 3DS                          0.724                     2.85    132
##  2 DC                           0.325                     0.132    14
##  3 DS                           0.626                     3.60    567
##  4 GBA                          0.404                     0.510   353
```

```
##  5 GC                    0.380                   0.382    353
##  6 PC                    0.304                   0.572    517
##  7 PS                    1.20                    3.68     158
##  8 PS2                   0.782                   2.05    1024
##  9 PS3                   0.946                   2.82     620
## 10 PS4                   0.925                   2.36     191
## 11 PSP                   0.435                   0.548    378
## 12 PSV                   0.259                   0.0830    93
## 13 Wii                   1.04                    9.37     429
## 14 WiiU                  0.656                   0.898     73
## 15 X360                  0.939                   3.02     728
## 16 XB                    0.328                   0.350    566
## 17 XOne                  0.877                   1.24     124
```

The top three platforms with the greatest average global sales are Playstation, Wii, and Playstation 3. The top three groups with the greatest variances are Wii, Playstation, and DS. The video game console with the most instances is Playstation 2.

```
#Averages and Variances in Global Sales by Rating
Video_Games_Train %>% group_by(Rating) %>%
  summarize(Avg_Global_Sales_Rating = mean(Global_Sales, na.rm = TRUE),
            Var_Global_Sales_Rating = var(Global_Sales, na.rm = TRUE),
            Count = n())
```

```
## # A tibble: 7 x 4
##   Rating Avg_Global_Sales_Rating Var_Global_Sales_Rating Count
##   <chr>                    <dbl>                   <dbl> <int>
## 1 AO                       1.95                   NA         1
## 2 E                        0.701                   3.35   2198
## 3 E10+                     0.515                   0.798   866
## 4 K-A                      1.92                   NA         1
## 5 M                        1.01                    4.37   1144
## 6 RP                       0.03                   NA         1
## 7 T                        0.548                   0.928  2109
```

The group of games that have an ESRB rating "M" has the highest average and variance. The ESRB rating that appears the most is E.

```
#Publishers That Have the Highest Sales
Video_Games_Train %>% group_by(Publisher) %>%
  summarize(Avg_Global_Sales_Publisher = mean(Global_Sales, na.rm = TRUE),
            Number_Games = n()) %>%
  arrange(desc(Avg_Global_Sales_Publisher)) %>%
  slice(1:20)
```

```
## # A tibble: 20 x 3
##    Publisher                          Avg_Global_Sales_Publish~ Number_Games
##    <chr>                                                  <dbl>        <int>
##  1 SquareSoft                                              2.84            8
##  2 GT Interactive                                          2.83            3
##  3 Red Orb                                                 2.43            1
##  4 Nintendo                                                2.40          245
##  5 Hello Games                                             1.7             1
##  6 Valve                                                   1.7             1
##  7 Bethesda Softworks                                      1.48           39
##  8 Microsoft Game Studios                                  1.42          114
##  9 Sony Oznline Entertainment                              1.28            1
## 10 Take-Two Interactive                                    1.26          229
## 11 RTL                                                     1.24            1
## 12 Black Label Games                                       1.2             1
## 13 RedOctane                                               1.19            3
## 14 Sony Computer Entertainment Euro~                       1.13            8
## 15 Virgin Interactive                                      1.13           17
## 16 Russel                                                  1.12            1
## 17 Slightly Mad Studios                                    1.09            1
## 18 LucasArts                                               1.05           63
## 19 Sony Computer Entertainment                             1.03          277
## 20 Activision                                              1.01          452
```

Out of all the publishers in the dataset, Nintendo has an average of $2.39 million in global sales. Activision has the highest number of games in the training set.

```
#Developers That Have the Highest Sales
Video_Games_Train %>% group_by(Developer) %>%
  summarize(Avg_Global_Sales_Developer = mean(Global_Sales, na.rm = TRUE),
            Number_Games = n()) %>%
  arrange(desc(Avg_Global_Sales_Developer)) %>% slice(1:20)
```

```
## # A tibble: 20 x 3
##    Developer                                Avg_Global_Sales_Dev~ Number_Games
##    <chr>                                                    <dbl>        <int>
##  1 DMA Design                                               13.1             1
##  2 Retro Studios, Entertainment Analysi~                    12.7             1
##  3 Infinity Ward, Sledgehammer Games                         9.92            3
##  4 Rockstar North                                            9.85           12
##  5 Polyphony Digital                                         9.66            4
##  6 Bungie Software                                           7.46            2
##  7 Game Freak                                                6.41            1
##  8 Bungie                                                    5.90            2
```

```
##  9 Nintendo                                5.60          55
## 10 Bungie Software, Bungie                 5.58           4
## 11 Infinity Ward                           4.99          13
## 12 Neversoft Entertainment, BudCat         4.98           1
## 13 343 Industries                          4.96           4
## 14 Naughty Dog, SCE/WWS                    4.92           1
## 15 Bethesda Game Studios                   4.78           7
## 16 Sledgehammer Games                      4.45           3
## 17 Naughty Dog                             4.38           5
## 18 Nintendo EAD Tokyo                      4.35           2
## 19 SCE/WWS, Media Molecule                 4.18           2
## 20 SCEA, Zindagi Games                     3.84           1
```

Out of the first 20 developers in the dataset, Nintendo has the highest number of games with an average of $5.6 million in sales.

```
#Developers That Have the Highest Sales
Video_Games_Train %>% group_by(Metascores) %>%
  summarize(Avg_Global_Sales_Metascore = mean(Global_Sales, na.rm = TRUE),
            Number_Games = n()) %>%
  arrange(desc(Avg_Global_Sales_Metascore)) %>% slice(1:20)
```

```
## # A tibble: 5 x 3
##   Metascores            Avg_Global_Sales_Metascore Number_Games
##   <fct>                                      <dbl>        <int>
## 1 Universal Acclaim                          2.84           256
## 2 Generally Favorable                        0.980         2207
## 3 Mixed or Average                           0.391         3220
## 4 Generally Unfavorable                      0.238          629
## 5 Just Plain Awful                           0.095            8
```

Games that have been deemed acclaimed by critics have generated an average of $2.85 million in global sales.

Correlation Matrix

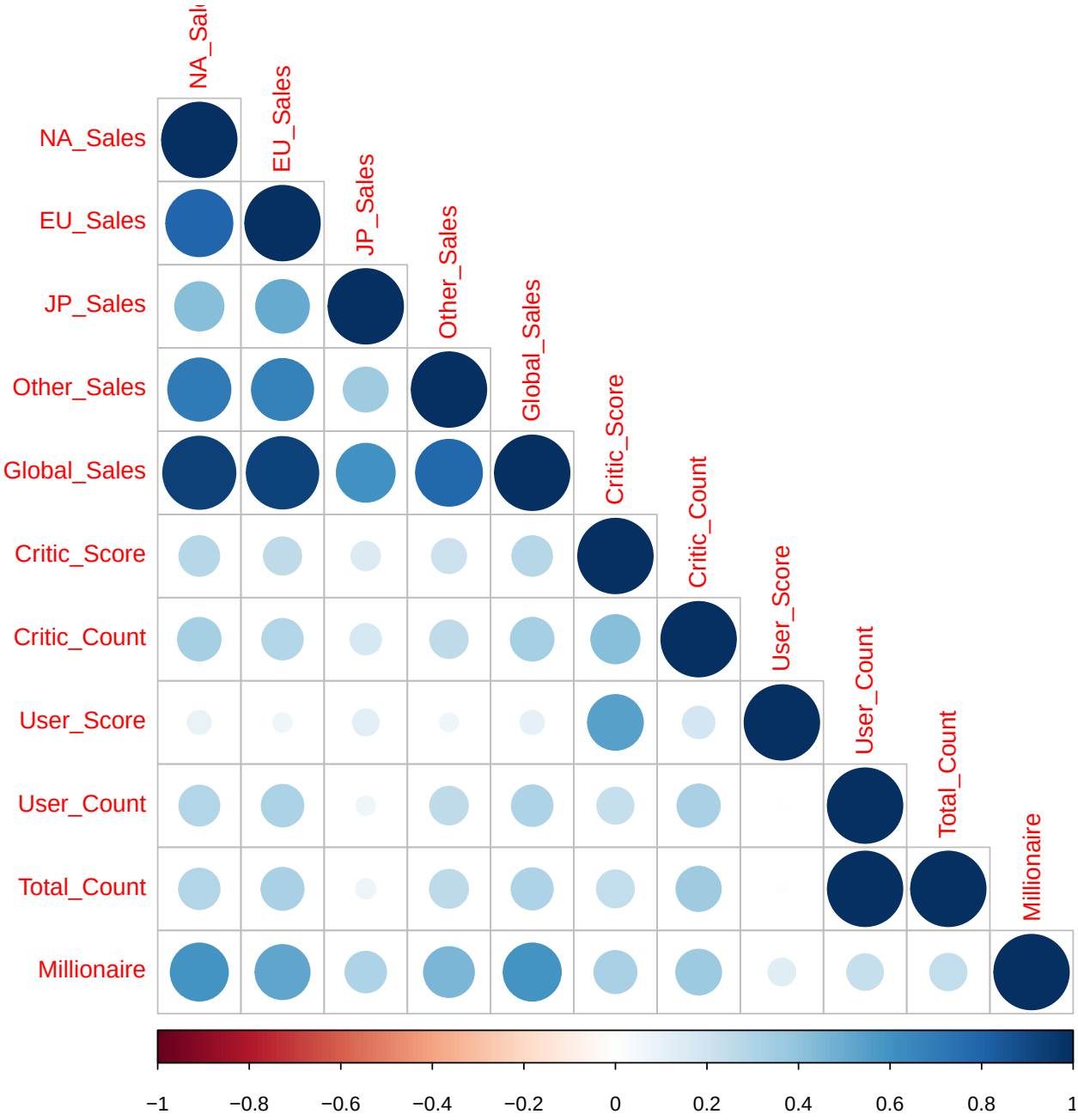
```
Video_Games_Sales <- Video_Games_Train

cormat <- cor(Video_Games_Sales %>% select_if(is.numeric))

print(cormat[, "Global_Sales"])
```

```
##     NA_Sales     EU_Sales     JP_Sales  Other_Sales Global_Sales
##    0.9393315    0.9218306    0.6076356    0.7841502    1.0000000
## Critic_Score Critic_Count   User_Score   User_Count  Total_Count
##    0.2869109    0.3328579    0.1020144    0.3012404    0.3089953
##  Millionaire
##    0.5910451
```

```
library('corrplot')
```

```
## corrplot 0.84 loaded
```

```
corrplot(cormat, method = "circle", type = "lower")
```



A correlation matrix is printed for all the numeric variables within the dataset. These values specifically display their correlations with Global_Sales. For example, Critic_Score has a correlation coefficient of approx. 0.287 with Global_Sales, this indicates a positive relationship since the coefficient is greater than 0. This signifies that as Critic_Score increases, Global_Sales also increases; however, the coefficient is closer to 0, indicating a weaker relationship.

Plots

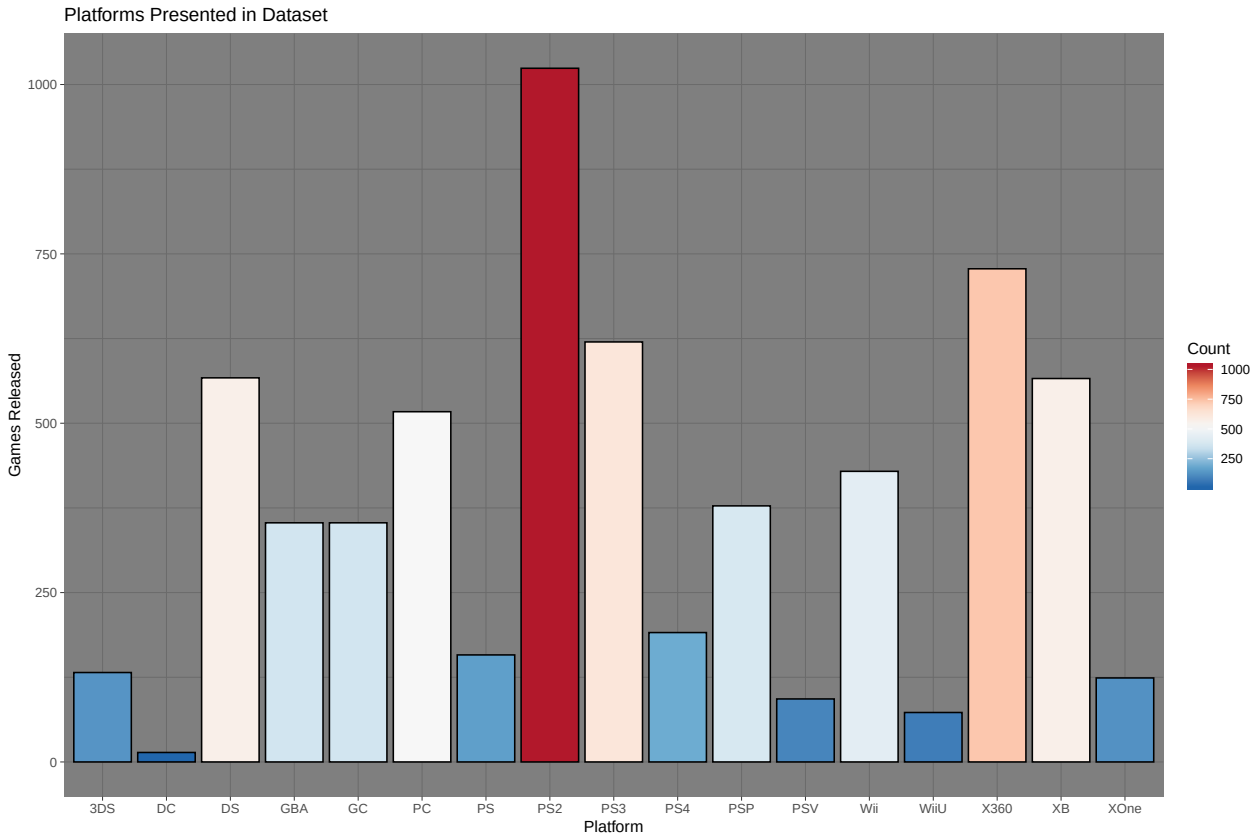
Platforms Presented in Dataset and Number of Times Presented

```
library(ggplot2)
library(ggthemes)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
by_platform <- Video_Games_Train %>% group_by(Platform) %>%
  summarise(Count = n())

ggplot(by_platform, aes(x = Platform, y = Count, fill = Count)) +
  geom_bar(color = "black", stat = "identity") +
  labs(x = "Platform", y = "Games Released",
       title = "Platforms Presented in Dataset") +
  scale_fill_distiller(palette = "RdBu") + theme_dark()
```

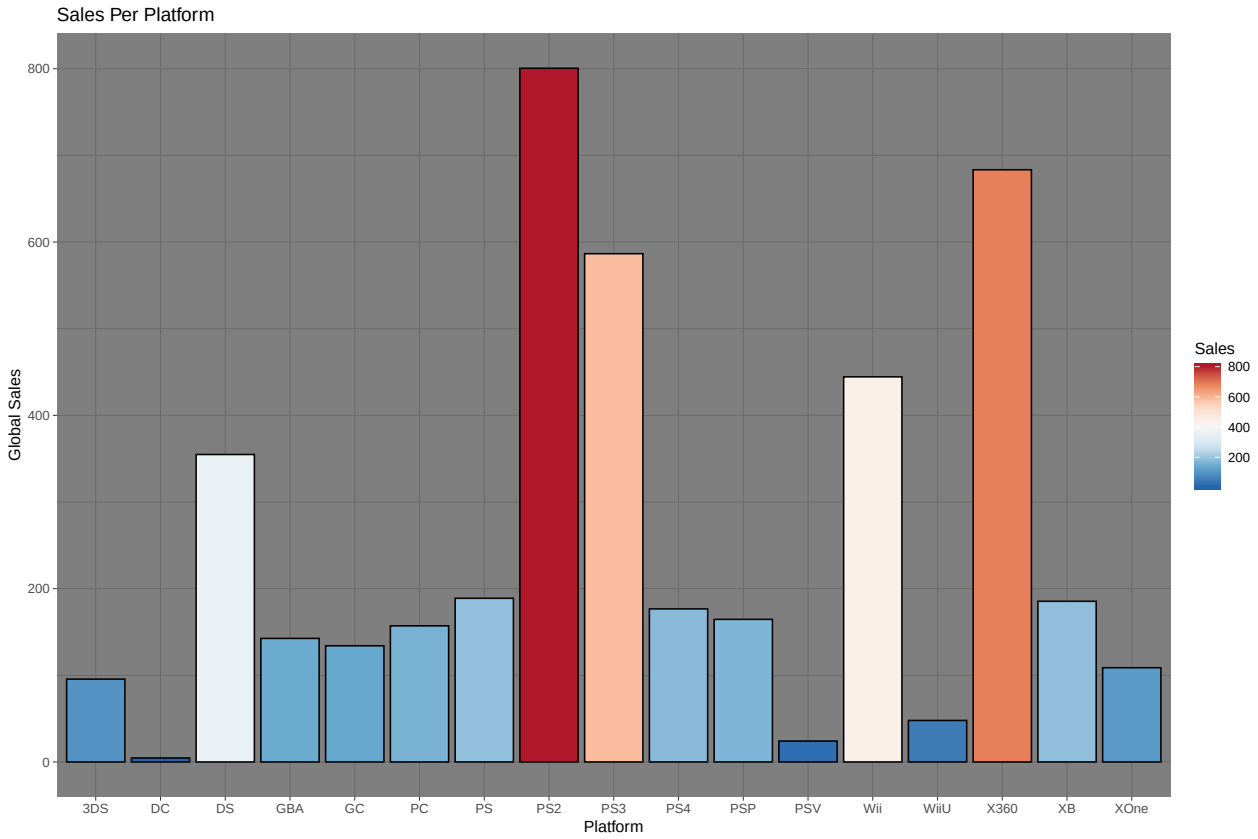


Total Sales Per Platform

```
sales_by_platform <- Video_Games_Train %>% group_by(Platform) %>%
summarise(GlobalSales = sum(Global_Sales))

sales_by_platform <- melt(sales_by_platform)
```

```
## Using Platform as id variables
```

```
colnames(sales_by_platform) = c("Platform", "SalesType", "Sales")

ggplot(sales_by_platform, aes(x = Platform, y = Sales, fill = Sales)) +
  geom_bar(color = "black", stat = "identity") +
  labs(x = "Platform", y = "Global Sales",
       title = "Sales Per Platform") +
  scale_fill_distiller(palette = "RdBu") + theme_dark()
```



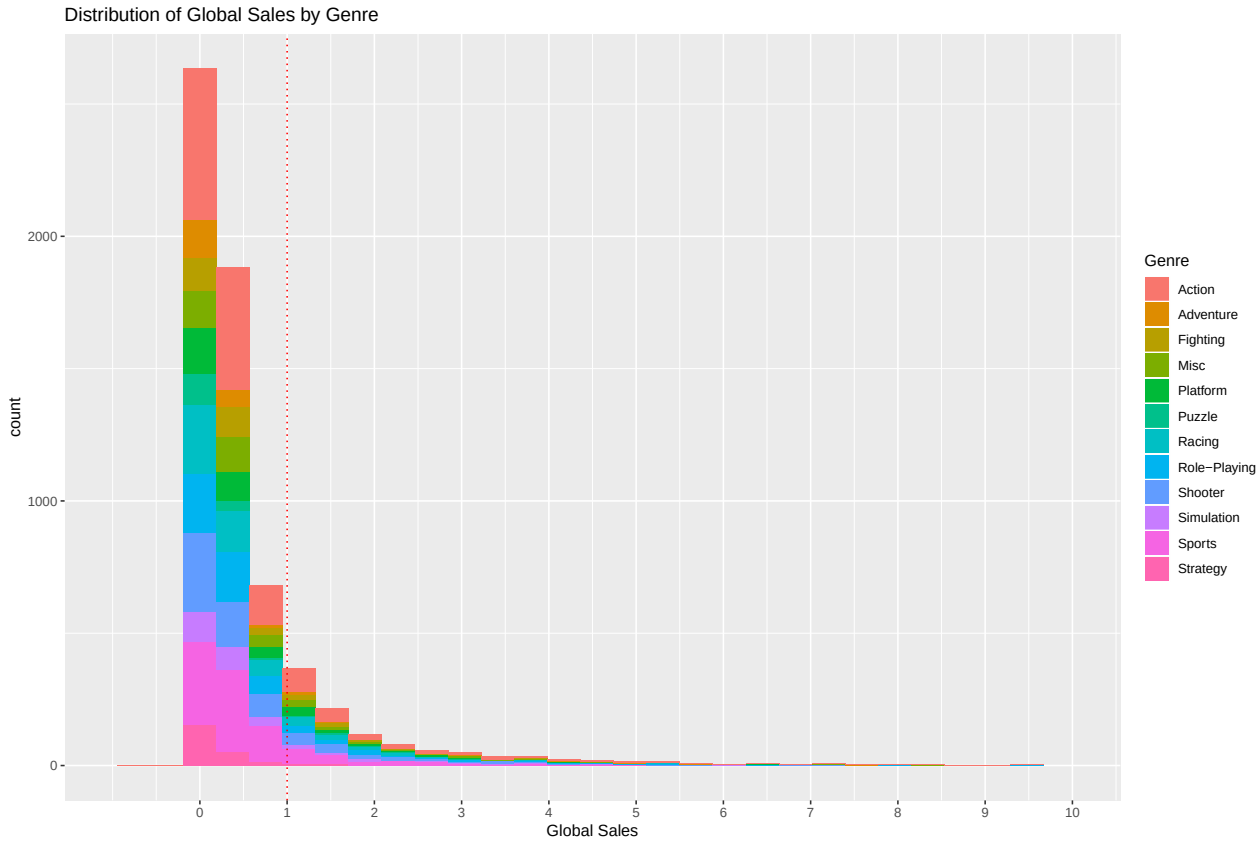
Histogram of Global Sales

```
ggplot(Video_Games_Train, aes(x = Global_Sales)) +
  geom_histogram(aes(fill = Genre)) +
  scale_x_continuous(limits = c(-1, 10), breaks = 0:10) +
  geom_vline(xintercept = 1, linetype = "dotted", color = "red") +
  labs(title = "Distribution of Global Sales by Genre",
       x = "Global Sales")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 30 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 24 rows containing missing values (geom_bar).
```

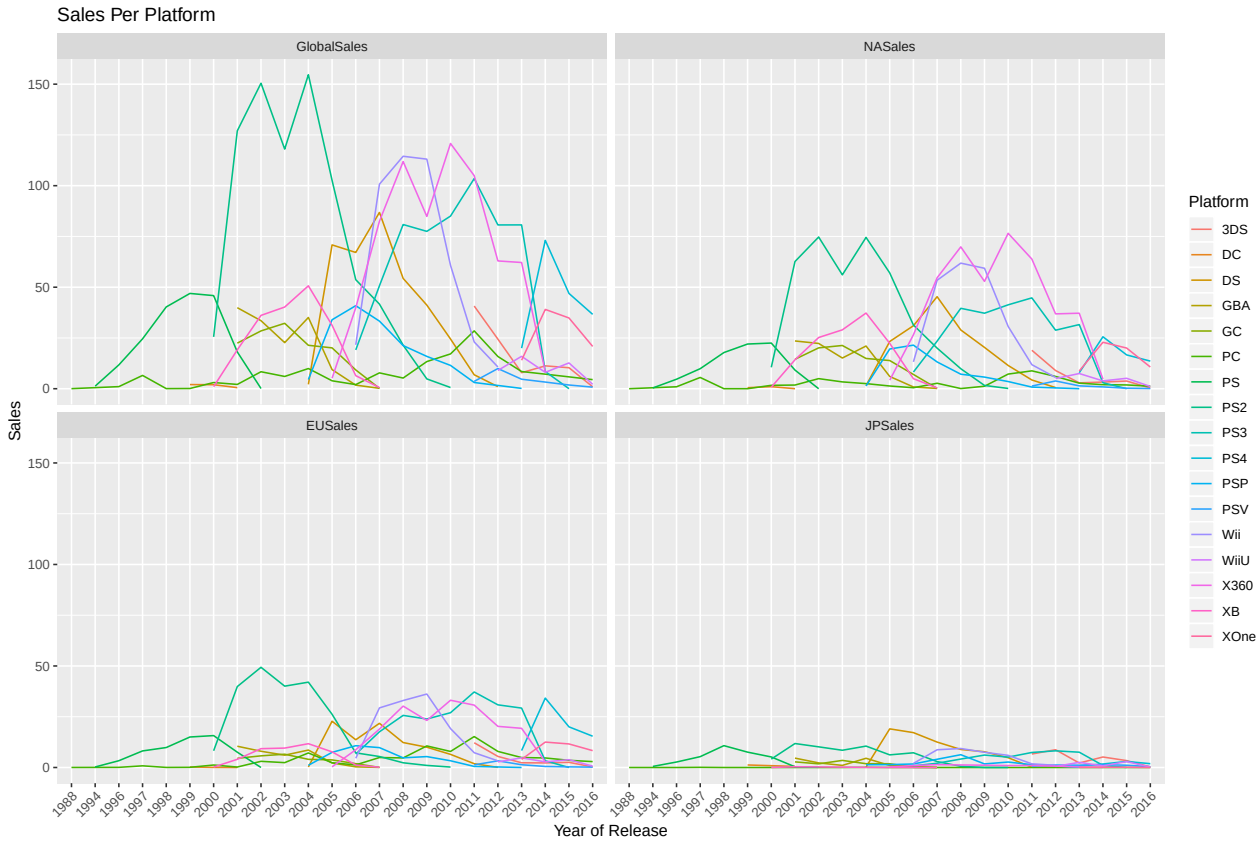


Sales By Year of Platforms

```
sales_by_platform <- Video_Games_Train %>%
group_by(Platform, Year_of_Release) %>%
summarise(GlobalSales = sum(Global_Sales), NASales = sum(NA_Sales),
          EUSales = sum(EU_Sales), JPSales = sum(JP_Sales))

sales_by_platform <- melt(sales_by_platform)
```

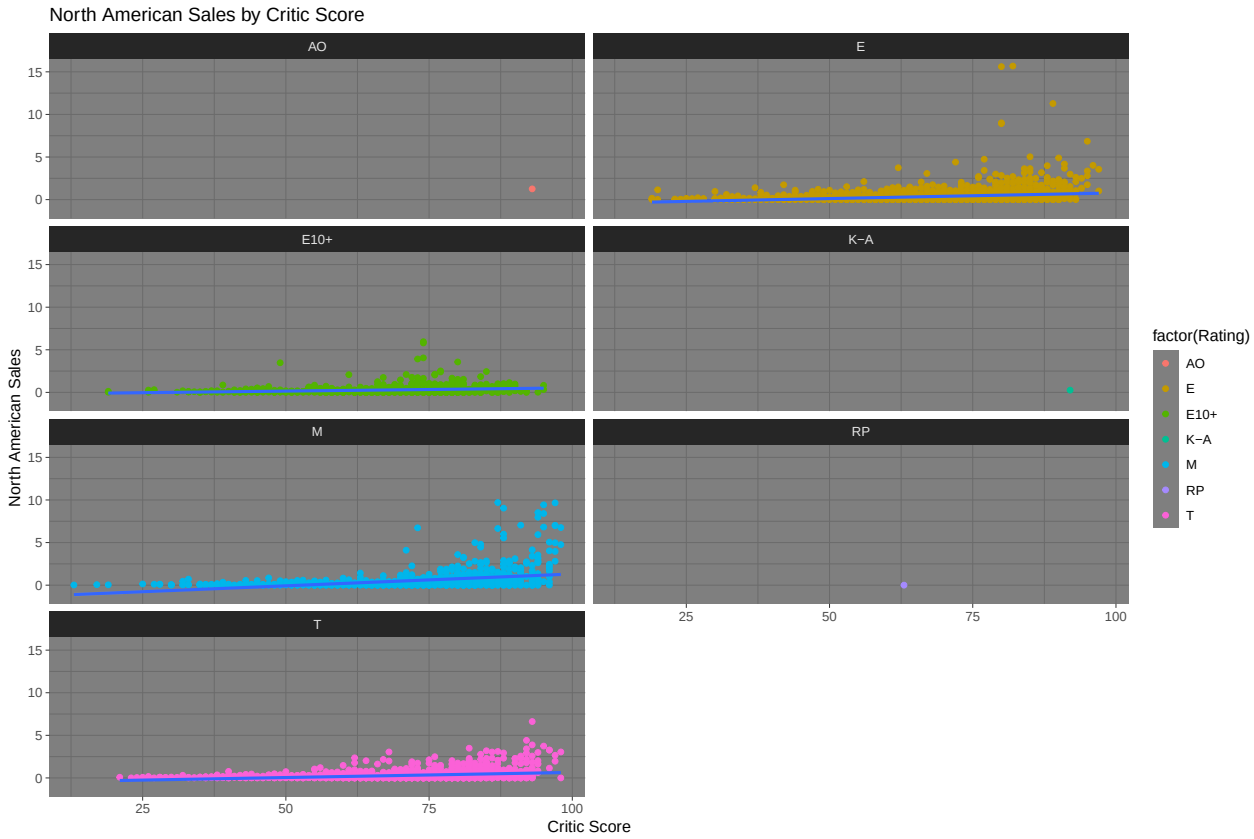
```
## Using Platform, Year_of_Release as id variables
```

```
colnames(sales_by_platform) = c("Platform", "Year", "SalesType", "Sales")
sales_by_platform$Year = as.factor(sales_by_platform$Year)

ggplot(sales_by_platform, aes(x = Year, y = Sales,
                              color = Platform, group = Platform)) +
  geom_line() + labs(x = "Year of Release", y = "Sales",
                     title = "Sales Per Platform") +
  facet_wrap(~SalesType) +  theme_gray() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Scatter Plots of North American Sales by Critic Score

```
ggplot(Video_Games_Train, aes(x = Critic_Score, y = NA_Sales)) +
  geom_point(aes(color = factor(Rating))) +
labs(x = "Critic Score",
       y = "North American Sales",
title = "North American Sales by Critic Score") +
  geom_smooth(method = "lm") + facet_wrap(~ Rating, nrow = 4) +
  theme_dark()
```



Global Sales by Genre

```
library(ggridges)
```

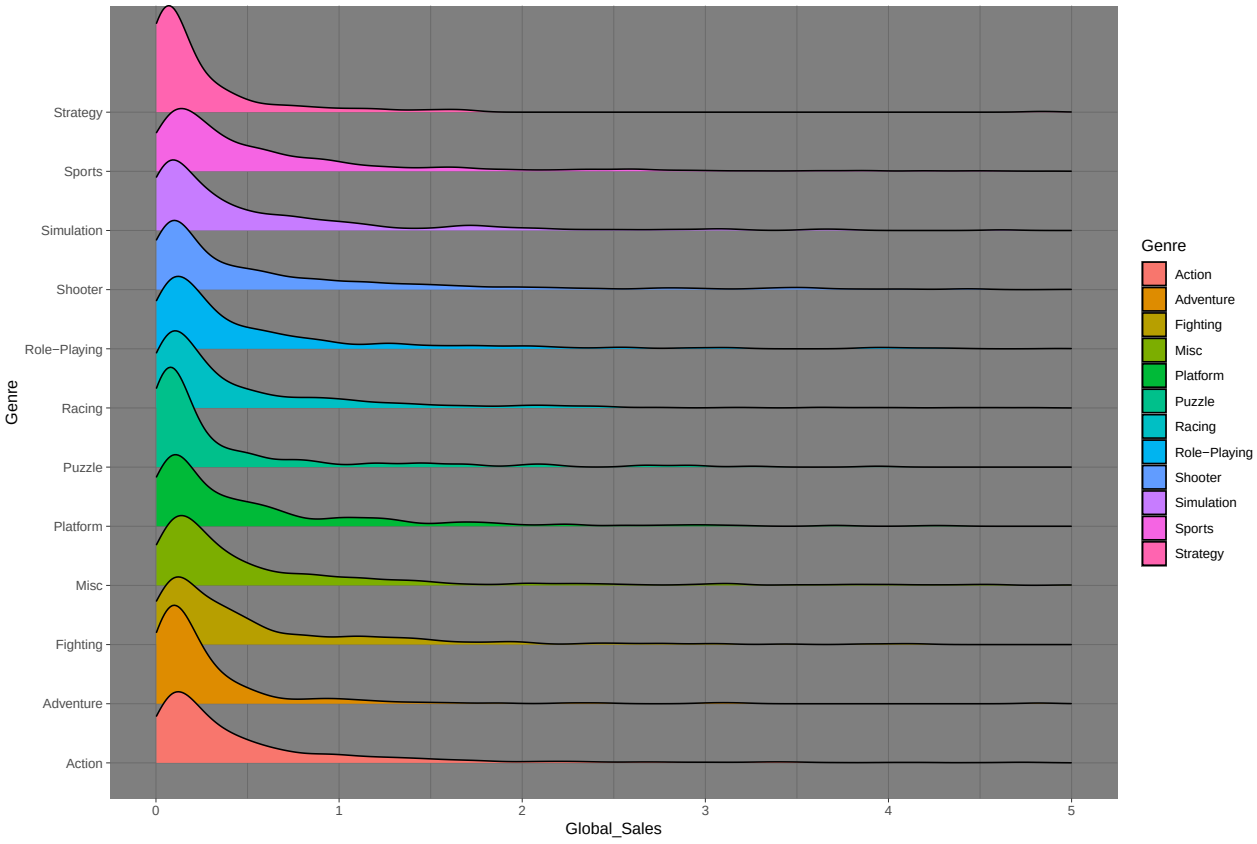
```
##
## Attaching package: 'ggridges'
```

```
## The following object is masked from 'package:ggplot2':
##
##     scale_discrete_manual
```

```
ggplot(Video_Games_Train, aes(x = Global_Sales, y = Genre, fill = Genre)) +
  geom_density_ridges() +
  scale_x_continuous(limits = c(0, 5)) +
  labs() + theme_dark()
```

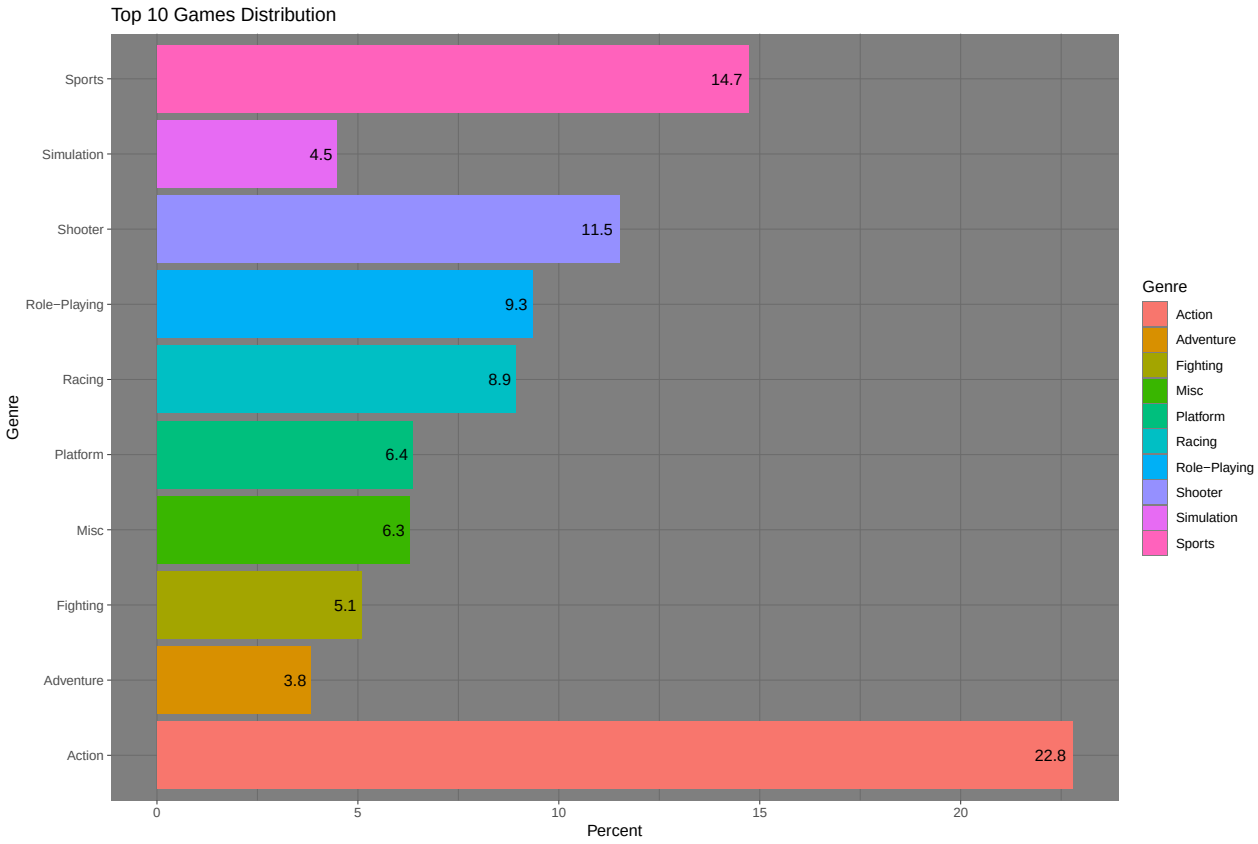
```
## Picking joint bandwidth of 0.0965
```

```
## Warning: Removed 110 rows containing non-finite values
## (stat_density_ridges).
```



Distribution of Top 10 Games

```
top10 <- head(names((sort(table(Video_Games_Train$Genre),
                          decreasing = TRUE))), 10)

Video_Games_Train %>%
    filter(Genre %in% top10) %>%
    group_by(Genre) %>%
    summarize(percentage = n()/nrow(Video_Games_Train)) %>%
    ggplot(aes(x = Genre, y = percentage * 100, fill = Genre)) +
    geom_col() +
    coord_flip() +
    labs(x = "Genre", y = "Percent",
         title = "Top 10 Games Distribution") +
    theme_dark() +
    geom_text(aes(label = round(percentage*100, digits = 1)),
              hjust = 1.2)
```



Statistical Models

Linear Model

```
mod1 <- lm(NA_Sales ~ Platform + Critic_Score + User_Score,
           data = Video_Games_Train)
summary(mod1)
```

```
##
## Call:
## lm(formula = NA_Sales ~ Platform + Critic_Score + User_Score,
##     data = Video_Games_Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8840 -0.3038 -0.1168  0.0976 14.7988
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.7164534  0.0791863  -9.048  < 2e-16 ***
```

```
## PlatformDC   -0.5102167  0.1967634  -2.593  0.00953 **
## PlatformDS    0.0708385  0.0675609   1.049  0.29444
## PlatformGBA  -0.0097434  0.0713890  -0.136  0.89144
## PlatformGC   -0.0463227  0.0713913  -0.649  0.51645
## PlatformPC   -0.3610681  0.0684327  -5.276 1.36e-07 ***
## PlatformPS    0.2033140  0.0825019   2.464  0.01375 *
## PlatformPS2   0.1050970  0.0647816   1.622  0.10478
## PlatformPS3   0.0601966  0.0670162   0.898  0.36909
## PlatformPS4  -0.0731419  0.0792349  -0.923  0.35599
## PlatformPSP  -0.0877507  0.0706479  -1.242  0.21425
## PlatformPSV  -0.2476497  0.0946277  -2.617  0.00889 **
## PlatformWii   0.3586419  0.0696150   5.152 2.66e-07 ***
## PlatformWiiU -0.0519338  0.1018899  -0.510  0.61028
## PlatformX360  0.2581813  0.0660955   3.906 9.47e-05 ***
## PlatformXB   -0.0817716  0.0675995  -1.210  0.22646
## PlatformXOne  0.0497944  0.0876668   0.568  0.57006
## Critic_Score  0.0205179  0.0007942  25.835  < 2e-16 ***
## User_Score   -0.0053434  0.0008272  -6.460 1.13e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6983 on 6301 degrees of freedom
## Multiple R-squared:  0.143,  Adjusted R-squared:  0.1405
## F-statistic:  58.4 on 18 and 6301 DF,  p-value: < 2.2e-16
```

The first model estimated is a linear regression model, in which NA_Sales is the dependent variable and the independent variables are the different platforms, ratings, user score, and critic_score. The greatest coefficient in the summary is PlatformWii, which is 0.3586419. This means that – holding fixed all other variables in the model – video games released on the Wii console will make $358,641.90 more in Global Sales than a game released on the 3DS. The asterisks next to the coefficients signify which are more significant to a game's global sales, in this case, they are: PlatformDC, PlatformPC, PlatformPSV, PlatformWii, PlatformXBOX360, Critic_Score, and User_Score.

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
#Training set predictions
preds_lm_train <- predict(mod1, Video_Games_Train)

#Test set predictions
preds_lm_test <- predict(mod1, newdata = Video_Games_Test)

#Train R2 and RMSE
R2(preds_lm_train, Video_Games_Train$Global_Sales)
```

```
## [1] 0.124508
```

```r
RMSE(preds_lm_train, Video_Games_Train$Global_Sales)
```

```
## [1] 1.510174
```

```r
#Test R2 and RMSE
R2(preds_lm_test, Video_Games_Test$Global_Sales)
```

```
## [1] 0.1113793
```

```r
RMSE(preds_lm_test, Video_Games_Test$Global_Sales)
```

```
## [1] 1.718064
```

Both the training set and test have very low R2 and high RMSE values. This signifies that the model is not that dependable.

Logistic Model

```r
logit_fit <- glm(Millionaire ~ Critic_Score + User_Score,
                 data = Video_Games_Train,
                 family = binomial)

summary(logit_fit)
```

```
##
## Call:
## glm(formula = Millionaire ~ Critic_Score + User_Score, family = binomial,
##     data = Video_Games_Train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4068  -0.6466  -0.4163  -0.2001   3.4735
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.695470   0.311131 -24.734  < 2e-16 ***
## Critic_Score  0.094342   0.004066  23.203  < 2e-16 ***
## User_Score   -0.011653   0.003410  -3.417 0.000633 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5700.5  on 6319  degrees of freedom
## Residual deviance: 4878.7  on 6317  degrees of freedom
## AIC: 4884.7
##
## Number of Fisher Scoring iterations: 5
```

```
exp(logit_fit$coefficients)
```

```
##  (Intercept) Critic_Score   User_Score
## 0.0004548829 1.0989354764 0.9884151274
```

The second model I estimated was a logistic regression model, in which the Millionaire variable was predicted against since it is a classification problem. In this model, we tested whether Critic_Score and User_Score has anything to do with the probabilities of a game having more than a million dollars in global sales. The exponentiated coefficient on Critic_Score indicates that for every one unit increase in a critic's score on a game, there will be a 9.89% chance that the game will make more than a million dollars in sales. The exponentiated coefficient on User_Score indicates that for every one unit increase in a gamer's score, there will be a 0.01% chance that the game will not make more than a million in sales.

```
logit_fit2 <- glm(Millionaire ~ Genre,
                  data = Video_Games_Train,
                  family = binomial)

summary(logit_fit2)
```

```
##
```

```
## Call:
## glm(formula = Millionaire ~ Genre, family = binomial, data = Video_Games_Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7150  -0.6095  -0.6039  -0.5182   2.4007
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.609438   0.070711 -22.761  < 2e-16 ***
## GenreAdventure    -1.038508   0.268187  -3.872 0.000108 ***
## GenreFighting      0.097727   0.161380   0.606 0.544802
## GenreMisc          0.119154   0.147598   0.807 0.419500
## GenrePlatform      0.169418   0.145134   1.167 0.243081
## GenrePuzzle       -0.330502   0.229466  -1.440 0.149780
## GenreRacing       -0.051960   0.134985  -0.385 0.700288
## GenreRole-Playing  0.020203   0.130540   0.155 0.877009
## GenreShooter       0.376025   0.113463   3.314 0.000919 ***
## GenreSimulation   -0.004246   0.174682  -0.024 0.980606
## GenreSports       -0.016851   0.113231  -0.149 0.881698
## GenreStrategy     -1.214684   0.294085  -4.130 3.62e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5700.5  on 6319  degrees of freedom
## Residual deviance: 5629.8  on 6308  degrees of freedom
## AIC: 5653.8
##
## Number of Fisher Scoring iterations: 5
```

```
exp(logit_fit2$coefficients)
```

```
##       (Intercept)    GenreAdventure     GenreFighting         GenreMisc
##         0.2000000         0.3539823         1.1026616         1.1265432
##     GenrePlatform       GenrePuzzle       GenreRacing GenreRole-Playing
##         1.1846154         0.7185629         0.9493671         1.0204082
##      GenreShooter   GenreSimulation       GenreSports     GenreStrategy
##         1.4564831         0.9957627         0.9832905         0.2968037
```

I tested another logistic model with the same estimated variable but using Genre instead. According to the model, if a game is classified in the shooter genre, then it is 45.65% more likely to make more than a million in global sales.

```r
logit_fit3 <- glm(Millionaire ~ Platform,
                  data = Video_Games_Train,
                  family = binomial)

summary(logit_fit3)
```

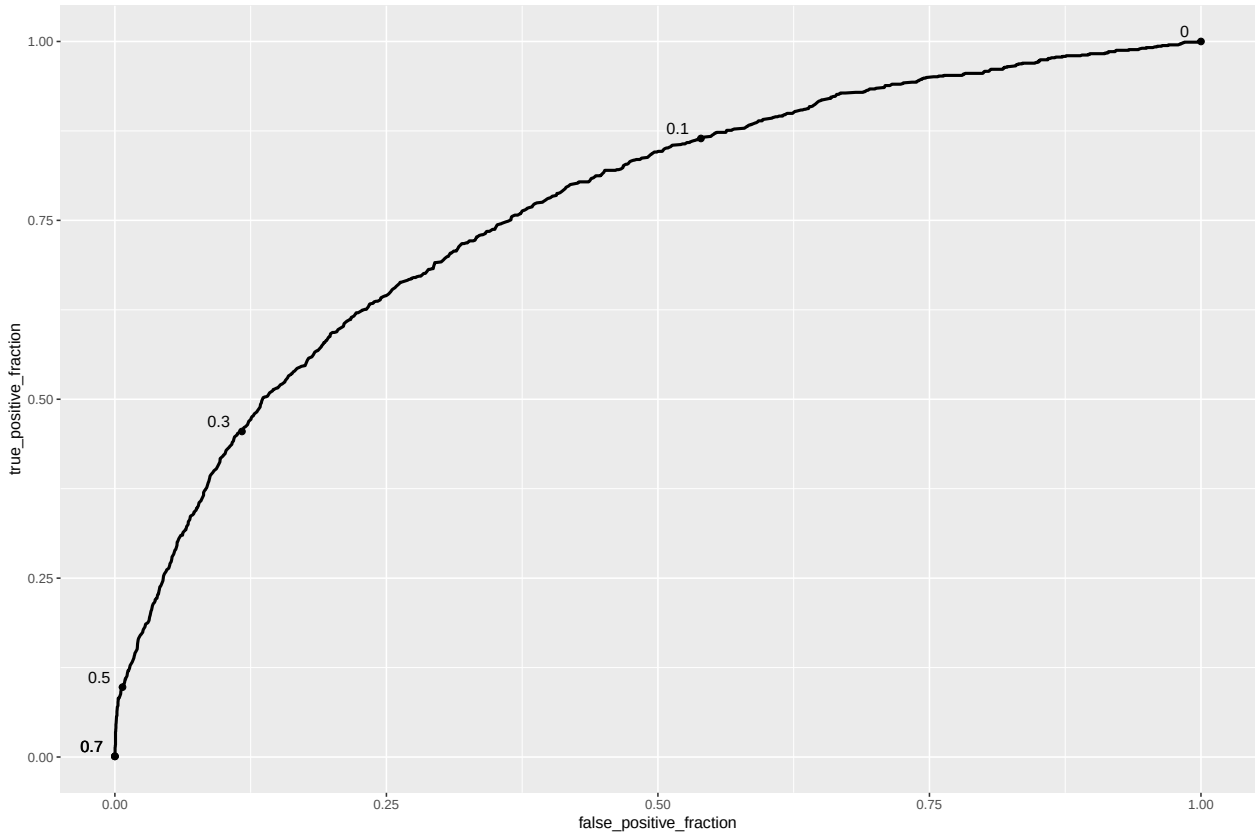
```
##
## Call:
## glm(formula = Millionaire ~ Platform, family = binomial, data = Video_Games_Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8691  -0.6829  -0.5327  -0.3779   2.5085
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.782949   0.247954  -7.191 6.45e-13 ***
## PlatformDC   -0.008811   0.803003  -0.011  0.99125
## PlatformDS   -0.098042   0.277213  -0.354  0.72359
## PlatformGBA  -0.488865   0.308073  -1.587  0.11255
## PlatformGC   -0.423754   0.305282  -1.388  0.16511
## PlatformPC   -0.779919   0.300984  -2.591  0.00956 **
## PlatformPS    0.767028   0.306450   2.503  0.01232 *
## PlatformPS2   0.445973   0.259632   1.718  0.08585 .
## PlatformPS3   0.701468   0.264595   2.651  0.00802 **
## PlatformPS4   0.773132   0.297038   2.603  0.00925 **
## PlatformPSP  -0.408411   0.301228  -1.356  0.17516
## PlatformPSV  -1.319393   0.568008  -2.323  0.02019 *
## PlatformWii   0.428403   0.275277   1.556  0.11965
## PlatformWiiU  0.344469   0.387116   0.890  0.37356
## PlatformX360  0.586698   0.263035   2.230  0.02571 *
## PlatformXB   -0.820690   0.298361  -2.751  0.00595 **
## PlatformXOne  1.003859   0.314463   3.192  0.00141 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5700.5  on 6319  degrees of freedom
## Residual deviance: 5444.2  on 6303  degrees of freedom
## AIC: 5478.2
##
## Number of Fisher Scoring iterations: 5
```

```
exp(logit_fit3$coefficients)
```

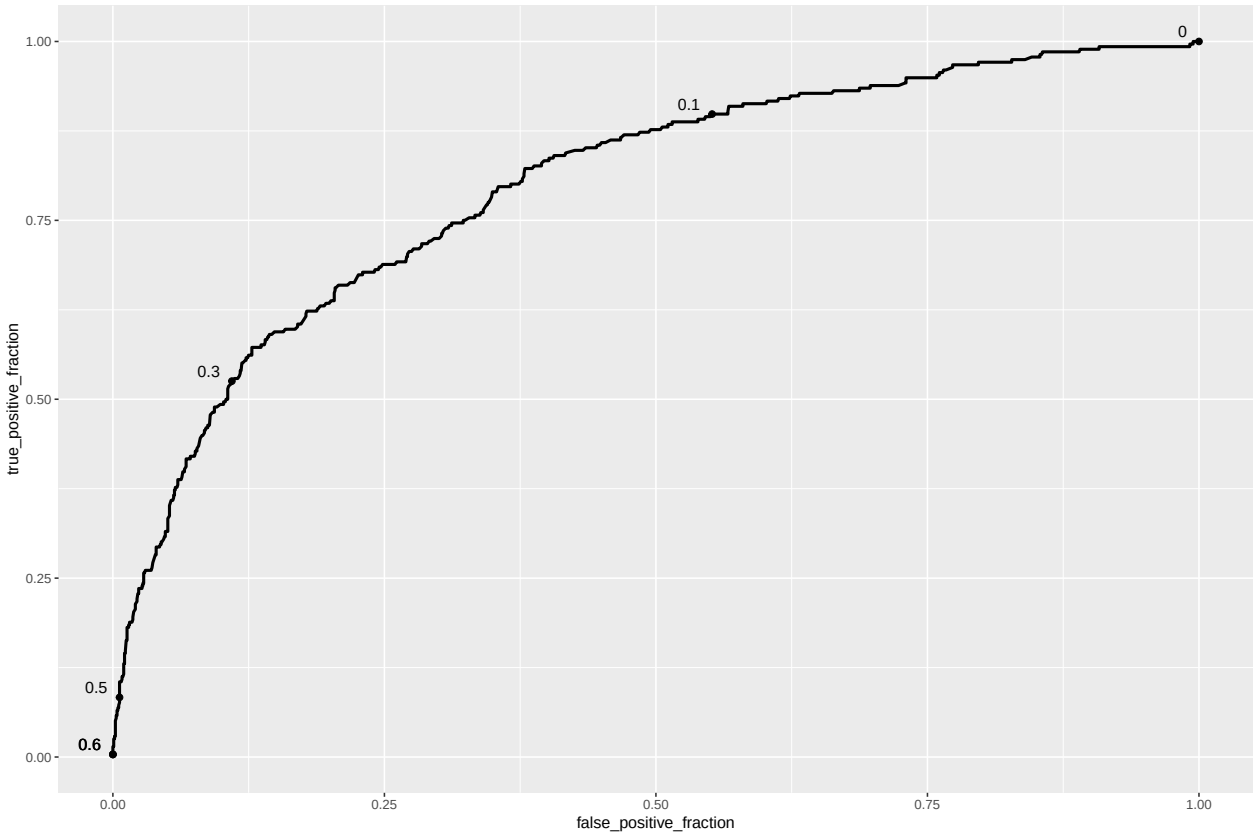
```
##  (Intercept)   PlatformDC   PlatformDS  PlatformGBA   PlatformGC
##    0.1681416    0.9912281    0.9066110    0.6133224    0.6545846
##   PlatformPC   PlatformPS  PlatformPS2  PlatformPS3  PlatformPS4
##    0.4584430    2.1533575    1.5620092    2.0167102    2.1665414
##  PlatformPSP  PlatformPSV  PlatformWii PlatformWiiU PlatformX360
##    0.6647059    0.2672975    1.5348048    1.4112400    1.7980416
##   PlatformXB PlatformXOne
##    0.4401278    2.7287926
```

I tested another logistic model with the same estimated variable but using platforms instead. According to the model, if a game is released on the XBOX One, then it is 173% more likely to make more than a million in global sales.

```
preds_train_logit <- data.frame(
  scores_train = predict(logit_fit, type = "response"),
  Millionaire = Video_Games_Train$Millionaire)

preds_test_logit <- data.frame(
  scores_test = predict(logit_fit,
                        newdata = Video_Games_Test,
                        type = "response"),
  Millionaire = Video_Games_Test$Millionaire)

library(plotROC)
ggplot(preds_train_logit,
                  aes(m = scores_train, d = Millionaire)) +
  geom_roc(cutoffs.at = c(.99, .9, .7, .5, .3, .1, 0))
```




```
ggplot(preds_test_logit,
                aes(m = scores_test, d = Millionaire)) +
                geom_roc(cutoffs.at = c(.99, .9, .7, .5, .3, .1, 0))
```




```
preds_train_logit %<>% mutate(class_pred05 = ifelse(scores_train > 0.5, 1, 0))

preds_test_logit %<>% mutate(class_pred05 = ifelse(scores_test > 0.5, 1, 0))

tab_train <- table(preds_train_logit$Millionaire, preds_train_logit$class_pred05)
tab_test <- table(preds_test_logit$Millionaire, preds_test_logit$class_pred05)

diagnostics <- function(tab) {
  TP <- tab[2,2]
  TN <- tab[1,1]
  FP <- tab[1,2]
  FN <- tab[2,1]
  Ntrue <- tab[1,1] + tab[1,2]
  Ptrue <- tab[2,1] + tab[2,2]
  Accuracy <- (TP + TN)/(Ntrue + Ptrue)
  sens <- TP/Ptrue
  spec <- TN/Ntrue
  FPR <- FP/Ntrue
  cat(sprintf("Accuracy: %s\n", round(Accuracy, 3)))
  cat(sprintf("TP: %s\n", round(TP, 3)))
  cat(sprintf("TN: %s\n", round(TN, 3)))
  cat(sprintf("Sensitivity: %s\n", round(sens, 3)))
```

```
  cat(sprintf("Specificity: %s\n", round(spec, 3)))
  cat(sprintf("False Pos Rate: %s\n",round(FPR, 3)))
}

#Confusion Training Set
diagnostics(tab_train)
```

```
## Accuracy: 0.844
## TP: 103
## TN: 5228
## Sensitivity: 0.098
## Specificity: 0.993
## False Pos Rate: 0.007
```

```
#Confusion Test Set
diagnostics(tab_test)
```

```
## Accuracy: 0.835
## TP: 23
## TN: 1297
## Sensitivity: 0.083
## Specificity: 0.994
## False Pos Rate: 0.006
```

```
print(tab_train)
```

```
##
##        0    1
##   0 5228   37
##   1  952  103
```

```
print(tab_test)
```

```
##
##        0    1
##   0 1297    8
##   1  253   23
```

Here I generated predictions for the logistic model to test how good this model is, and used them to produce ROC plots for the test and training sets. The ROC plots show that this model is not that accurate as both of the curves are closer to the 45-degree diagonal of the ROC space. The ROC plots suggest we should lower out cutoff probabilities. If we do that, then the line in the ROC plot will get closer to the top left of the space and we can get a more accurate model. If we use a cutoff probability of 0.5, then the accuracies of the class predictions for the training set and test set will be 84.4% and 83.5%, respectively.

Random Forest

```
library('randomForest')
```

```
## randomForest 4.6-14
```

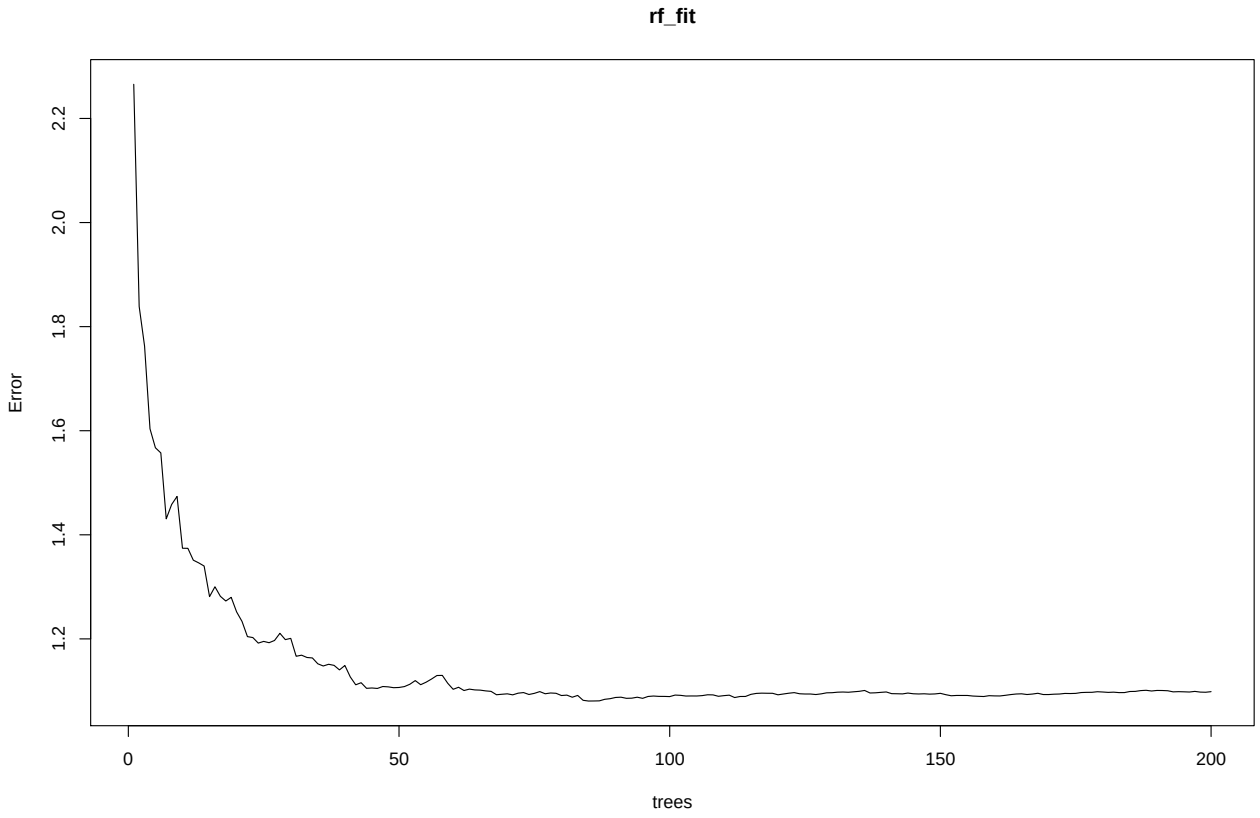
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

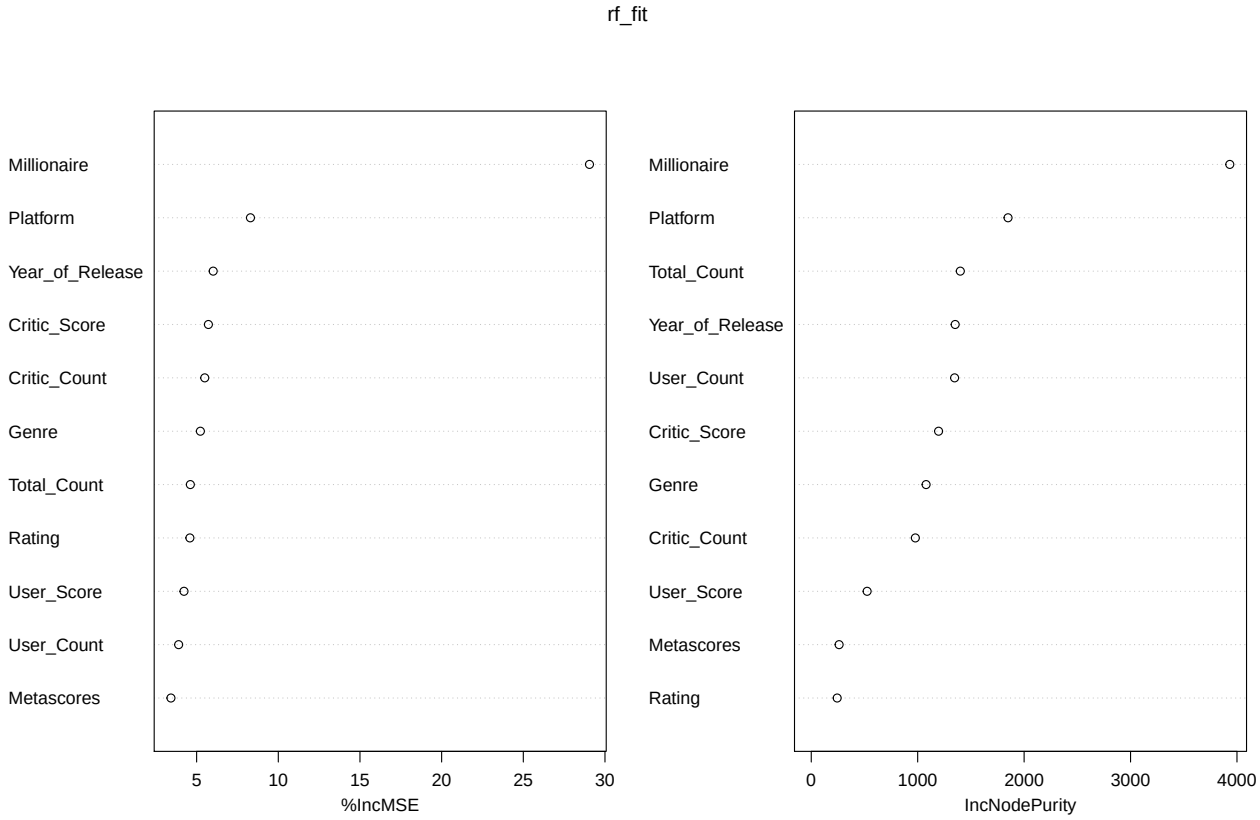
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
VG_rf <- Video_Games_Train %>% mutate(Name = as.factor(Name),
                                      Platform = as.factor(Platform),
                                      Year_of_Release = as.factor(Year_of_Release),
                                      Genre = as.factor(Genre),
                                      Publisher = as.factor(Publisher),
                                      Developer = as.factor(Developer),
                                      Rating = as.factor(Rating)) %>%
  select(-Developer, -Publisher, -Name, -NA_Sales,
         -JP_Sales, -Other_Sales, -EU_Sales)

rf_fit <- randomForest(Global_Sales ~ .,
                       data = VG_rf,
                       type = classification,
                       mtry = 4,
                       ntree = 200,
                       importance = TRUE,
                       localImp = TRUE)

plot(rf_fit)
```




```r
varImpPlot(rf_fit)
```

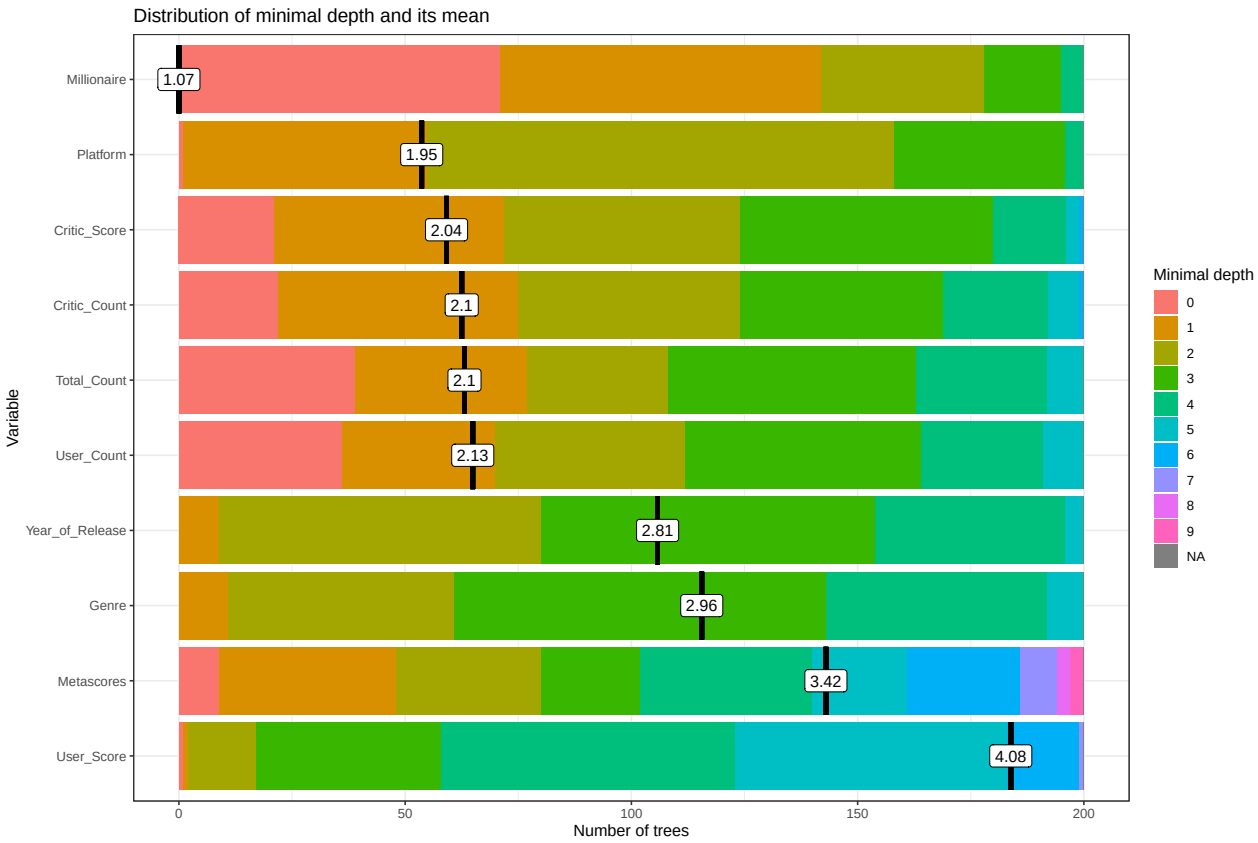


My last model is a random forest model, in which I used a total number of 200 decision trees used most of the variables to predict against Global_Sales. The top 5 most important variables in the random forest object are: Millionaire, Platform, Year_of_Release, Critic_Score, and Critic_Count

```
library(randomForestExplainer)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
plot_min_depth_distribution(rf_fit)
```



The plot above shows the distribution of minimal depth among the trees of the forest model. The scale of the X axis goes from zero to the maximum number of trees in which any variable was used for splitting. The numbers that are in the middle of the plot refer to the average depth or length of a tree when the tree splits on that respective variable. So Millionaire has the lowest mean minimal depth and will most usually show up on the first node of a decision tree, making it the most important. The colors show the different distributions of how often the variables are picked at that depth.

Conclusion

After much analysis, the main factors that help predict how much a video game will make in sales are Platform, Genre, Rating and Critic_Score. The plots and models have been consistenly displaying that games released on the Playstation by Sony generate higher sales worldwide, and video games that are of the shooter and action genres are expected to have higher sales as well. Additionally, games that are rated M are expected to generate higher sales.