HarvardX: PH125.9x Data Science
Create Your Own Project Submission
Video Game Sales with Ratings Project

Emory Gray

2023-06-02

Introduction

Goal of this project is to create simplistic machine learning models to predict video game global sales using the various variables available in our dataset.

The video game sales data used for this project was obtained from Kaggle.

Data sourced from curated list of datasets from CYO Project Overview page.

Data Loading

```
# Checking required packages

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(ranger)) install.packages("ranger", repos = "http://cran.us.r-project.org")
if(!require(kernlab)) install.packages("kernlab", repos = "http://cran.us.r-project.org")

# Loading necessary packages

library(tidyverse)
library(caret)
library(randomForest)
library(ranger)
library(kernlab)

# Placeholders for PDF generation issues

#library(knitr)
#library(tinytex)

# Data provided from curated list of datasets from CYO Project Overview page
# Curated List of Datasets: https://www.kaggle.com/code/annavictoria/ml-friendly-public-datasets/notebo
# Video Game Sales with Ratings: https://www.kaggle.com/datasets/rush4ratio/video-game-sales-with-ratin
```

```r
# Download dataset from personal github repository

dat <- read.csv("https://raw.githubusercontent.com/emoryg/Video-Game-Sales-with-Ratings/main/video_games
```

Data Preparation

Hindsight, this dataset contains excessive amounts of N/A values caused by the joining of two separate datasets by dataset organizer. To combat this for our purposes of this project, below we will be cleaning up and removing these excessive N/A values so that we have a precise subset. Attempted to add exceptions to the N/A values down further in my Methods Analysis and that was proving extremely difficult to manage.

```r
# Analyzing data structure and generating statistical summary

head(dat)
```

```
##                        Name Platform Year_of_Release        Genre Publisher
## 1                Wii Sports      Wii            2006       Sports  Nintendo
## 2         Super Mario Bros.      NES            1985     Platform  Nintendo
## 3            Mario Kart Wii      Wii            2008       Racing  Nintendo
## 4         Wii Sports Resort      Wii            2009       Sports  Nintendo
## 5 Pokemon Red/Pokemon Blue       GB            1996 Role-Playing  Nintendo
## 6                    Tetris       GB            1989       Puzzle  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
## 1    41.36    28.96     3.77        8.45        82.53           76           51
## 2    29.08     3.58     6.81        0.77        40.24           NA           NA
## 3    15.68    12.76     3.79        3.29        35.52           82           73
## 4    15.61    10.93     3.28        2.95        32.77           80           73
## 5    11.27     8.89    10.22        1.00        31.37           NA           NA
## 6    23.20     2.26     4.22        0.58        30.26           NA           NA
##   User_Score User_Count Developer Rating
## 1          8        322  Nintendo      E
## 2                    NA
## 3        8.3        709  Nintendo      E
## 4          8        192  Nintendo      E
## 5                    NA
## 6                    NA
```

```r
# Data appears clean and variable names are appropriate for this project

str(dat)
```

```
## 'data.frame':    16719 obs. of  16 variables:
##  $ Name           : chr  "Wii Sports" "Super Mario Bros." "Mario Kart Wii" "Wii Sports Resort" ...
##  $ Platform       : chr  "Wii" "NES" "Wii" "Wii" ...
##  $ Year_of_Release: chr  "2006" "1985" "2008" "2009" ...
##  $ Genre          : chr  "Sports" "Platform" "Racing" "Sports" ...
##  $ Publisher      : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ NA_Sales       : num  41.4 29.1 15.7 15.6 11.3 ...
##  $ EU_Sales       : num  28.96 3.58 12.76 10.93 8.89 ...
##  $ JP_Sales       : num  3.77 6.81 3.79 3.28 10.22 ...
##  $ Other_Sales    : num  8.45 0.77 3.29 2.95 1 0.58 2.88 2.84 2.24 0.47 ...
```

```
##  $ Global_Sales   : num  82.5 40.2 35.5 32.8 31.4 ...
##  $ Critic_Score   : int  76 NA 82 80 NA NA 89 58 87 NA ...
##  $ Critic_Count   : int  51 NA 73 73 NA NA 65 41 80 NA ...
##  $ User_Score     : chr  "8" "" "8.3" "8" ...
##  $ User_Count     : int  322 NA 709 192 NA NA 431 129 594 NA ...
##  $ Developer      : chr  "Nintendo" "" "Nintendo" "Nintendo" ...
##  $ Rating         : chr  "E" "" "E" "E" ...
```

```
# Mixture of character, numerical, and integer data types
# rows        16,719
# variables   16
# Dataset appears large enough for our simplistic model goals

# Reviewing statistical summary of dataset

summary(dat)
```

```
##      Name             Platform         Year_of_Release       Genre
##  Length:16719       Length:16719       Length:16719       Length:16719
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   Publisher            NA_Sales          EU_Sales         JP_Sales
##  Length:16719       Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  Class :character   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Mode  :character   Median : 0.0800   Median : 0.020   Median : 0.0000
##                     Mean   : 0.2633   Mean   : 0.145   Mean   : 0.0776
##                     3rd Qu.: 0.2400   3rd Qu.: 0.110   3rd Qu.: 0.0400
##                     Max.   :41.3600   Max.   :28.960   Max.   :10.2200
##
##   Other_Sales        Global_Sales      Critic_Score    Critic_Count
##  Min.   : 0.00000   Min.   : 0.0100   Min.   :13.00   Min.   :  3.00
##  1st Qu.: 0.00000   1st Qu.: 0.0600   1st Qu.:60.00   1st Qu.: 12.00
##  Median : 0.01000   Median : 0.1700   Median :71.00   Median : 21.00
##  Mean   : 0.04733   Mean   : 0.5335   Mean   :68.97   Mean   : 26.36
##  3rd Qu.: 0.03000   3rd Qu.: 0.4700   3rd Qu.:79.00   3rd Qu.: 36.00
##  Max.   :10.57000   Max.   :82.5300   Max.   :98.00   Max.   :113.00
##                                       NA's   :8582    NA's   :8582
##   User_Score         User_Count        Developer            Rating
##  Length:16719       Min.   :    4.0   Length:16719       Length:16719
##  Class :character   1st Qu.:   10.0   Class :character   Class :character
##  Mode  :character   Median :   24.0   Mode  :character   Mode  :character
##                     Mean   :  162.2
##                     3rd Qu.:   81.0
##                     Max.   :10665.0
##                     NA's   :9129
```

```
# Review dataset for N/A values causing issues as previously described in my hindsight comment for this
# We can see that there is a large amount of N/A data

colSums(is.na(dat))
```

```
##            Name        Platform Year_of_Release           Genre       Publisher
##               0               0               0               0               0
##        NA_Sales        EU_Sales        JP_Sales     Other_Sales    Global_Sales
##               0               0               0               0               0
##    Critic_Score    Critic_Count      User_Score      User_Count       Developer
##            8582            8582               0            9129               0
##          Rating
##               0
```

```
# Eliminating records for games that are not complete
# This takes us down to 7,017 rows and still 16 variables

dat <- dat[complete.cases(dat), ]

# Verify N/A's removed
# All columns report zero N/A's now

colSums(is.na(dat))
```

```
##            Name        Platform Year_of_Release           Genre       Publisher
##               0               0               0               0               0
##        NA_Sales        EU_Sales        JP_Sales     Other_Sales    Global_Sales
##               0               0               0               0               0
##    Critic_Score    Critic_Count      User_Score      User_Count       Developer
##               0               0               0               0               0
##          Rating
##               0
```

```
# Note, Year of Release is a character and I want to convert this to an integer for analysis later
# Year of Release includes N/A's which we will need to remove for our methods analysis

unique(dat$Year_of_Release)
```

```
##  [1] "2006" "2008" "2009" "2005" "2007" "2010" "2013" "2004" "2002" "2001"
## [11] "2011" "2012" "2014" "1997" "1999" "2015" "2016" "2003" "1998" "1996"
## [21] "2000" "N/A"  "1994" "1985" "1992" "1988"
```

```
dat <- dat[dat$Year_of_Release != "N/A", ]

dat$Year_of_Release <- as.integer(dat$Year_of_Release)

# Verifying N/A's were removed
# Cleanup of Year of Release takes us down to 6,894 rows and still 16 variables

unique(dat$Year_of_Release)
```

```
##  [1] 2006 2008 2009 2005 2007 2010 2013 2004 2002 2001 2011 2012 2014 1997 1999
## [16] 2015 2016 2003 1998 1996 2000 1994 1985 1992 1988
```

```
# Checking other strings for N/A's and removing when necessary

sum(dat$Developer == "N/A") # 0
```

```
## [1] 0
```

```r
sum(dat$Rating == "N/A") # 0
```

```
## [1] 0
```

```r
sum(dat$Publisher == "N/A") # 1
```

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

```r
# Removing N/A's from Publisher

dat <- dat[dat$Publisher != "N/A", ]

# Verifying N/A's were removed
# We can see that 1 Publisher record was removed
# Taking us down to 6,893 rows and still 16 variables
```

```r
# unique(dat$Publisher)
```

```r
# Reviewing statistical summaries of sales data for noise
# Similar process performed in MovieLens project and turned out to be informative for me so recreating

summary(dat$NA_Sales)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.060   0.150   0.391   0.390  41.360
```

```r
summary(dat$EU_Sales)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0200  0.0600  0.2345  0.2100 28.9600
```

```r
summary(dat$JP_Sales)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.06388 0.01000 6.50000
```

```r
summary(dat$Other_Sales)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00000  0.01000  0.02000  0.08201  0.07000 10.57000
```

```r
summary(dat$Global_Sales)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0100  0.1100  0.2900  0.7716  0.7500 82.5300
```

```
summary(dat$Critic_Score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.00   62.00   72.00   70.26   80.00   98.00
```

```
summary(dat$Critic_Count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00   14.00   24.00   28.84   39.00  113.00
```

```
summary(dat$User_Count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     4.0    11.0    27.0   174.4    89.0 10665.0
```

```
summary(dat$User_Score)
```

```
##    Length     Class      Mode
##      6893 character character
```

```
# User Score is not numeric and I want to covert it to numerical

dat$User_Score <- as.numeric(dat$User_Score)

# Verify User Score was converted to numerical by analyzing statistical summary

summary(dat$User_Score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   6.500   7.500   7.185   8.200   9.600
```

```
# Note, Critic Score and User Score are on on the same scale
# Critic is out of 100 points and User is out of 10 points
# Leaving the scale conversion below commented out unless need for scale symmetry is presented later in

# dat$User_Score <- dat$User_Score * 10

# Final dataset review
# rows        6,893
# variables   16
```

Data Training & Testing

```
# Create training and testing datasets

set.seed(1, sample.kind = "Rounding")
```

```
test_index <- createDataPartition(y = dat$Global_Sales, times = 1, p = 0.5, list = FALSE)
training_set <- dat[-test_index, ]  # 3,446 rows
testing_set <- dat[test_index, ]    # 3,447 rows

# We also want to make sure that we are including the possible values of our variables in the training s

total_data <- rbind(training_set, testing_set)

for (x in 1:length(names(total_data))) {
  levels(training_set[, x]) <- levels(total_data[, x])
}
```

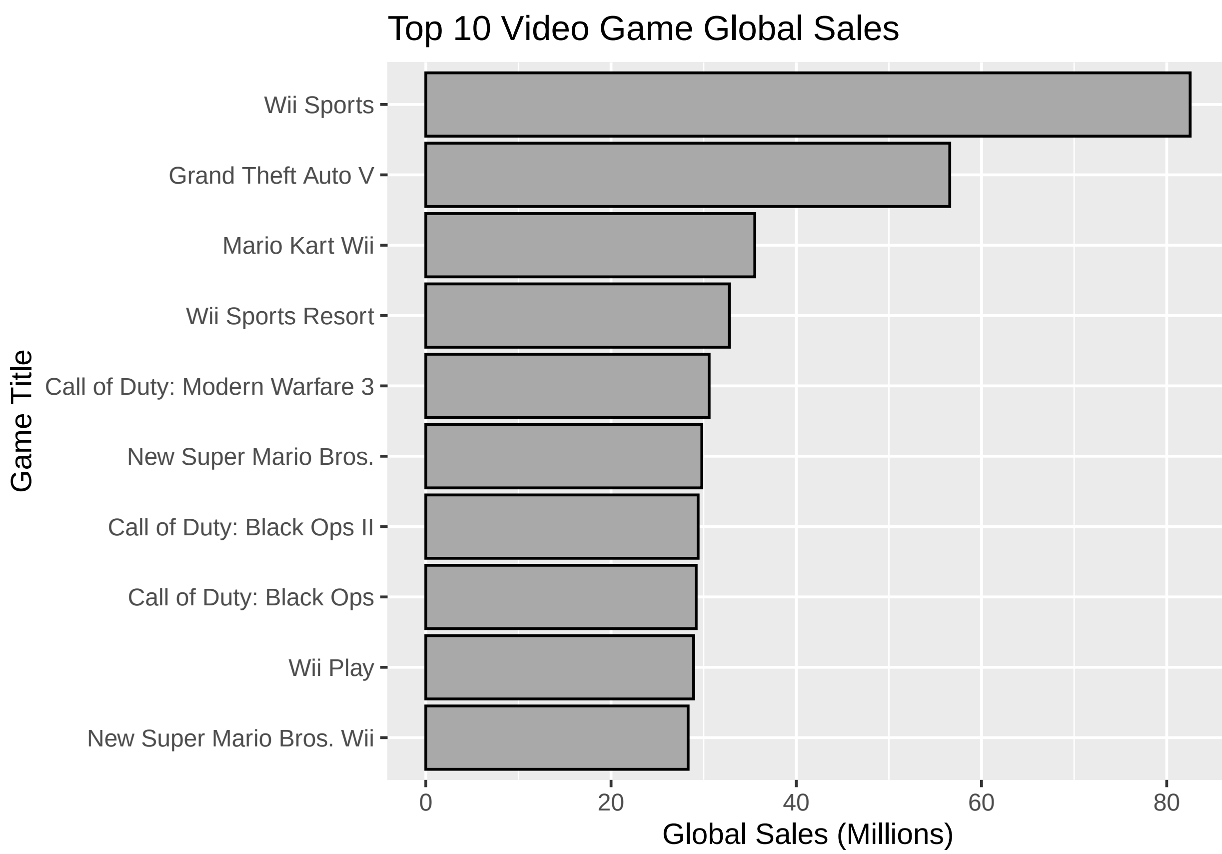
Exploratory Analysis

Review top 10 global sales by video game title (millions):

```
# Note, this is grouping by Name to summarize global sales of titles released across multiple platforms
# Example, Grand Theft Auto V was released on PC, PS3, PS4, Xbox 360, and Xbox One

# Per our dataset Wii Sports is marginally ahead of the other video game titles in our top 10
# Personal note, it is extremely surprising to see Call of Duty: Modern Warfare 3 have higher global sa

dat %>%
  group_by(Name) %>%
  summarize(g_sales = sum(Global_Sales)) %>%
  arrange(-g_sales) %>%
  top_n(10, g_sales) %>%
  ggplot(aes(g_sales, reorder(Name, g_sales))) +
  geom_bar(color = "black", fill = "darkgray", stat = "identity") +
  xlab("Global Sales (Millions)") +
  ylab("Game Title") +
  ggtitle("Top 10 Video Game Global Sales")
```



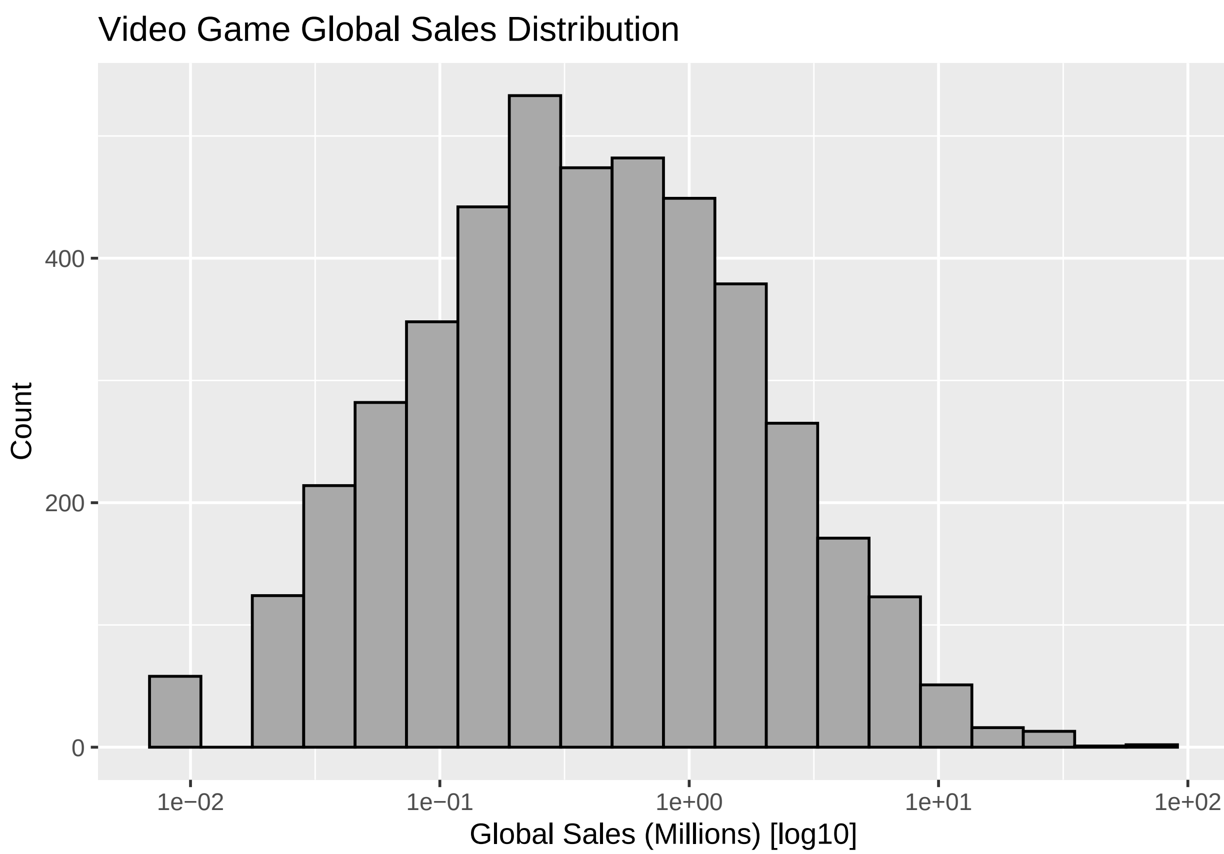

```
dat %>%
  group_by(Name) %>%
  summarize(g_sales = sum(Global_Sales)) %>%
  arrange(-g_sales) %>%
  top_n(10, g_sales)
```

```
## # A tibble: 10 x 2
##    Name                           g_sales
##    <chr>                            <dbl>
##  1 Wii Sports                        82.5
##  2 Grand Theft Auto V                56.6
##  3 Mario Kart Wii                    35.5
##  4 Wii Sports Resort                 32.8
##  5 Call of Duty: Modern Warfare 3    30.6
##  6 New Super Mario Bros.             29.8
##  7 Call of Duty: Black Ops II        29.4
##  8 Call of Duty: Black Ops           29.2
##  9 Wii Play                          28.9
## 10 New Super Mario Bros. Wii         28.3
```

Review global sales distribution:

```r
# Using log10 scale shows us that the peak global sales distribution is marginally below $1M
# The use of the log scale visualizes the understanding that not all video games generate millions of d
# Note, we will use log later in the project for our methods analysis

dat %>%
  group_by(Name) %>%
  summarize(g_sales = sum(Global_Sales)) %>%
  ggplot(aes(g_sales)) +
  geom_histogram(color = "black", fill = "darkgray", bins = 20) +
  scale_x_log10() +
  xlab("Global Sales (Millions) [log10]") +
  ylab("Count") +
  ggtitle("Video Game Global Sales Distribution")
```

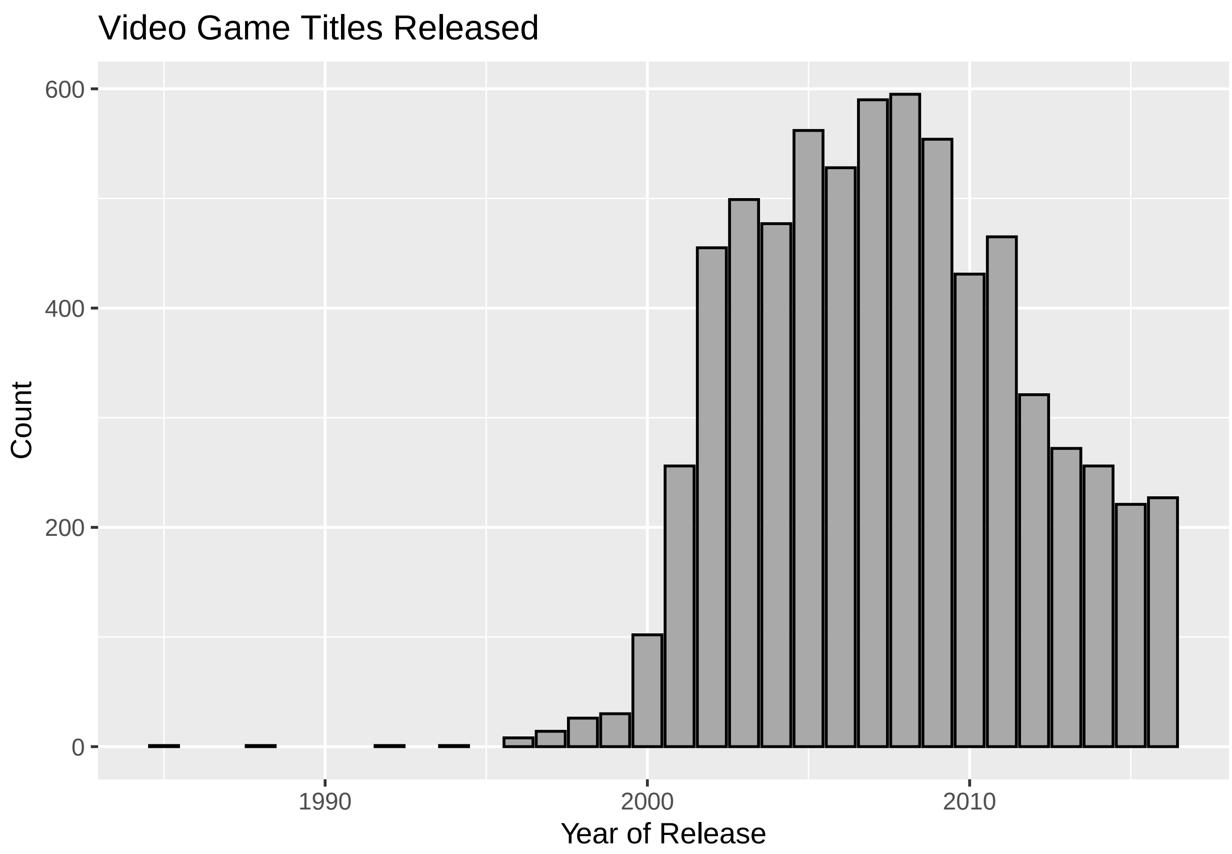


Review of video game titles released by year:

```r
# This shows a clear peak in titles released around 2007 & 2008
# It also shows a steady incline with the coming of the 20th century and video game craze

dat %>%
  group_by(Year_of_Release) %>%
  count() %>%
  ggplot(aes(Year_of_Release, n)) +
```

```
  geom_bar(color = "black", fill = "darkgray", stat = "identity") +
  xlab("Year of Release") +
  ylab("Count") +
  ggtitle("Video Game Titles Released")
```

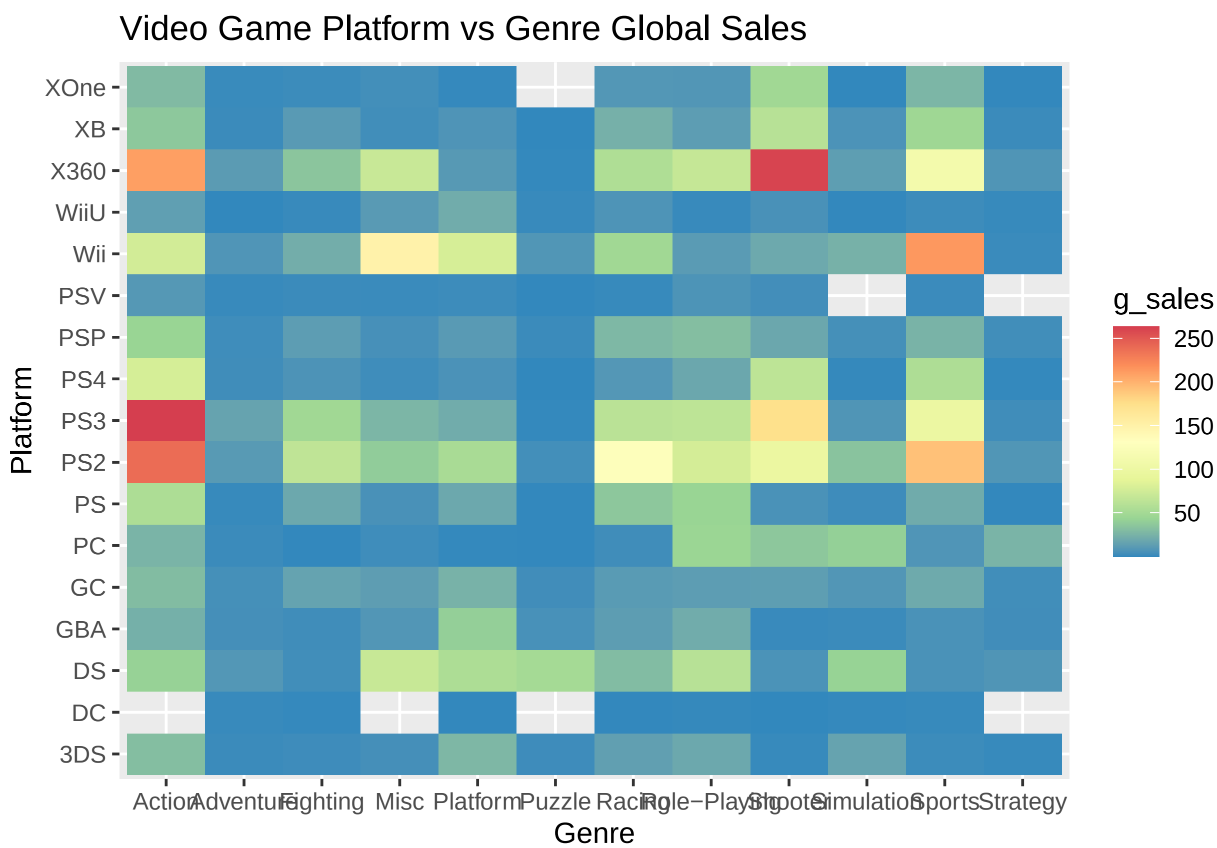


Review of global sales for each platform and genre:

```
# This reveals that PS3 Action games generating the highest global sales with Xbox 360 Shooter games ri

dat %>%
  group_by(Platform, Genre) %>%
  summarize(g_sales = sum(Global_Sales)) %>%
  ggplot(aes(Genre, Platform, fill = g_sales)) +
  geom_raster() +
  scale_fill_distiller(palette = "Spectral") +
  xlab("Genre") +
  ylab("Platform") +
  ggtitle("Video Game Platform vs Genre Global Sales")
```

```
## `summarise()` has grouped output by 'Platform'. You can override using the
## `.groups` argument.
```



Methods Overview

To achieve our goal of modeling video game global sales we will be training our models using various training method models provided through the caret package as well as utilizing previously learned techniques from the linear regression and machine learning courses as part of this HarvardX Data Science course. Citation: Caret Package Document Method specific documentation found under section 7.

Model accuracy will be assessed used Root Mean Square Error or RMSE. RMSE is defined as "the standard deviation of the residuals (prediction errors)". Citation: Statistics How To

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{x}_i - x_i)^2}$$

```
# RMSE function defined

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Methods Analysis

Model 1 (Linear Regression):

This model will predict global sales based on all data points using linear regression and will work as our baseline for this project.

```
# Note, training attempt below on all variables would not fully process so avoiding

# lm_model <- train(log(Global_Sales) ~ ., method = "lm", data = training_set)

# Avoided trying to train this model with the other sales variables and was able to generate lm_model
# Note, this training was successful but produces errors when using predict() function

# Error message:
# Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels) :

#lm_model <- train(log(Global_Sales) ~
#                     Platform +
#                     Year_of_Release +
#                     Genre +
#                     Publisher +
#                     Critic_Score +
#                     Critic_Count +
#                     User_Score +
#                     User_Count +
#                     Developer +
#                     Rating, method = "lm", data = training_set)

# lm_predict <- predict(lm_model, testing_set)

# Going to avoid training with the Publisher and Developer variables from the training set
# Note, this train generated with no problem and predict as well

lm_model <- train(log(Global_Sales) ~
                    Platform +
                    Year_of_Release +
                    Genre +
                    # Publisher +
                    Critic_Score +
                    Critic_Count +
                    User_Score +
                    User_Count +
                    #Developer +
                    Rating, method = "lm", data = training_set)

testing_set$lm_predicted <- predict(lm_model, testing_set)

lm_rmse <- RMSE(log(testing_set$Global_Sales), testing_set$lm_predicted)

lm_rmse # 1.045152
```

```
## [1] 1.045152
```

```r
# Storing results in data frame

results <- tibble(Method = "Model 1 (Linear Regression)", RMSE = lm_rmse)
results %>% knitr::kable()
```

Method	RMSE
Model 1 (Linear Regression)	1.045152

```r
# We knew that this was going to be our baseline model and that since it only uses linear regression wo
# We will attempt to approve upon this RMSE result with additional train methods
```

Model 2 (Random Forest):

This model will use the random forest method to train as it uses randomness to build decision tress or uncorrelated forest of trees that can be used to predict our outcome using cross-validation.

Citation: Random Forest Documentation from cran.r-project.org

```r
rf_fit_control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

# Simplistic attempt of building upon model 1 train proved to be insufficient and results in timeout

#rf_model <- train(log(Global_Sales) ~
#                     Platform +
#                     Year_of_Release +
#                     Genre +
#                     # Publisher +
#                     Critic_Score +
#                     Critic_Count +
#                     User_Score +
#                     User_Count +
#                     #Developer +
#                     Rating, method = "rf", trControl = rf_fit_control, data = training_set)

# We can add additional tuning to our decision trees for train

rf_tuning <- expand.grid(.mtry = c(1:5), .min.node.size = seq(1,5,1), .splitrule = c("xtrees", "varianc
```

```r
# Review of tuning data frame

rf_tuning
```

```r
# Note, if project reviewer is executing this script this will take SEVERAL MINUTES to execute so pleas

rf_model <- train(log(Global_Sales) ~
                    Platform +
                    Year_of_Release +
                    Genre +
                    # Publisher +
                    Critic_Score +
```

```
                     Critic_Count +
                     User_Score +
                     User_Count +
                     #Developer +
                     Rating, method = "ranger", trControl = rf_fit_control, tuneGrid = rf_tuning, data =

testing_set$rf_predicted <- predict(rf_model, testing_set)

rf_rmse <- RMSE(log(testing_set$Global_Sales), testing_set$rf_predicted)

rf_rmse # 0.9550495
```

```
## [1] 0.9550495
```

```
# Storing results in data frame

results <- bind_rows(results, tibble(Method = "Model 2 (Random Forest)", RMSE = rf_rmse))
results %>% knitr::kable()
```

Method	RMSE
Model 1 (Linear Regression)	1.0451519
Model 2 (Random Forest)	0.9550495

```
# Results show only a slight gain compared to the linear regression model
# Was expecting a larger gain but we are sub 1.0000000 so that is a plus
```

Model 3 (SVM Linear):

Taking a step back in my method approach to cover my bases from a linear method perspective. This model will use the SVM Linear method in train.

```
svm_model <- train(log(Global_Sales) ~
                     Platform +
                     Year_of_Release +
                     Genre +
                     # Publisher +
                     Critic_Score +
                     Critic_Count +
                     User_Score +
                     User_Count +
                     #Developer +
                     Rating, method = "svmLinear", data = training_set)

testing_set$svm_predicted <- predict(svm_model, testing_set)

svm_rmse <- RMSE(log(testing_set$Global_Sales), testing_set$svm_predicted)

svm_rmse # 1.0467
```

```
## [1] 1.0467
```

```r
# Storing results in data frame

results <- bind_rows(results, tibble(Method = "Model 3 (SVM Linear)", RMSE = svm_rmse))
results %>% knitr::kable()
```

Method	RMSE
Model 1 (Linear Regression)	1.0451519
Model 2 (Random Forest)	0.9550495
Model 3 (SVM Linear)	1.0466999

```r
# Results show increase in RMSE from random forest model so we went the wrong direction
# RMSE from svm linear is even higher than our baseline linear regression
```

Model 4 (K-Nearest Neighbors):

We are taking a turn in our methods analysis to train and test a non-parametric algorithm. Personal note, never tested or trained this before that I can recall so learning as we go. Referenced Caret Package Documentation on train models.

```r
knn_model <- train(log(Global_Sales) ~
                     Platform +
                     Year_of_Release +
                     Genre +
                     # Publisher +
                     Critic_Score +
                     Critic_Count +
                     User_Score +
                     User_Count +
                     #Developer +
                     Rating, method = "knn", trControl = trainControl(method = "repeatedcv", number = 1

testing_set$knn_predicted <- predict(knn_model, testing_set)

knn_rmse <- RMSE(log(testing_set$Global_Sales), testing_set$knn_predicted)

knn_rmse # 1.262107
```

```
## [1] 1.262107
```

```r
# Storing results in data frame

results <- bind_rows(results, tibble(Method = "Model 4 (K-Nearest Neighbors)", RMSE = knn_rmse))
results %>% knitr::kable()
```

Method	RMSE
Model 1 (Linear Regression)	1.0451519
Model 2 (Random Forest)	0.9550495
Model 3 (SVM Linear)	1.0466999

Method	RMSE
Model 4 (K-Nearest Neighbors)	1.2621069

```
# Results show that again our RMSE went in the wrong direction
# Highest value so far in our method analysis
```

Results Comparison

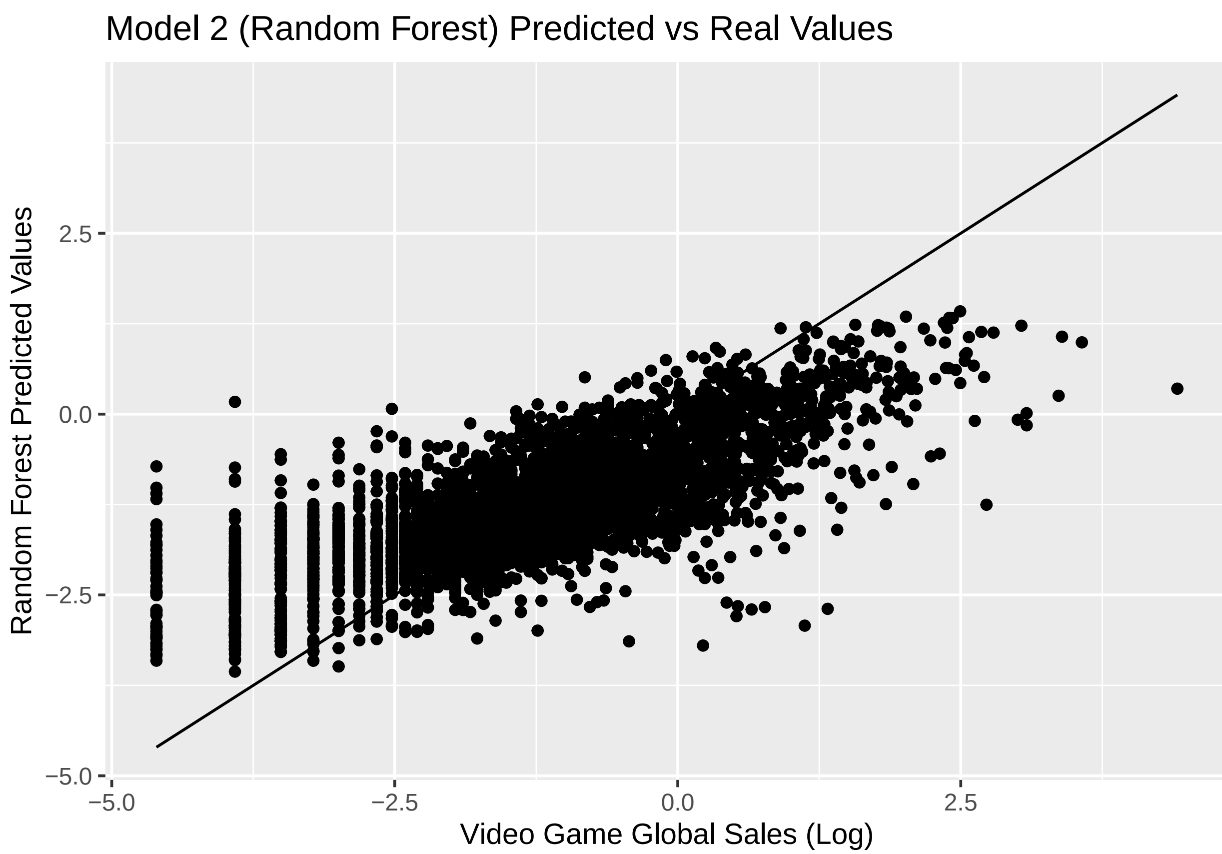
We see that our best performing RMSE was the Random Forest method.

```
results %>% knitr::kable()
```

Method	RMSE
Model 1 (Linear Regression)	1.0451519
Model 2 (Random Forest)	0.9550495
Model 3 (SVM Linear)	1.0466999
Model 4 (K-Nearest Neighbors)	1.2621069

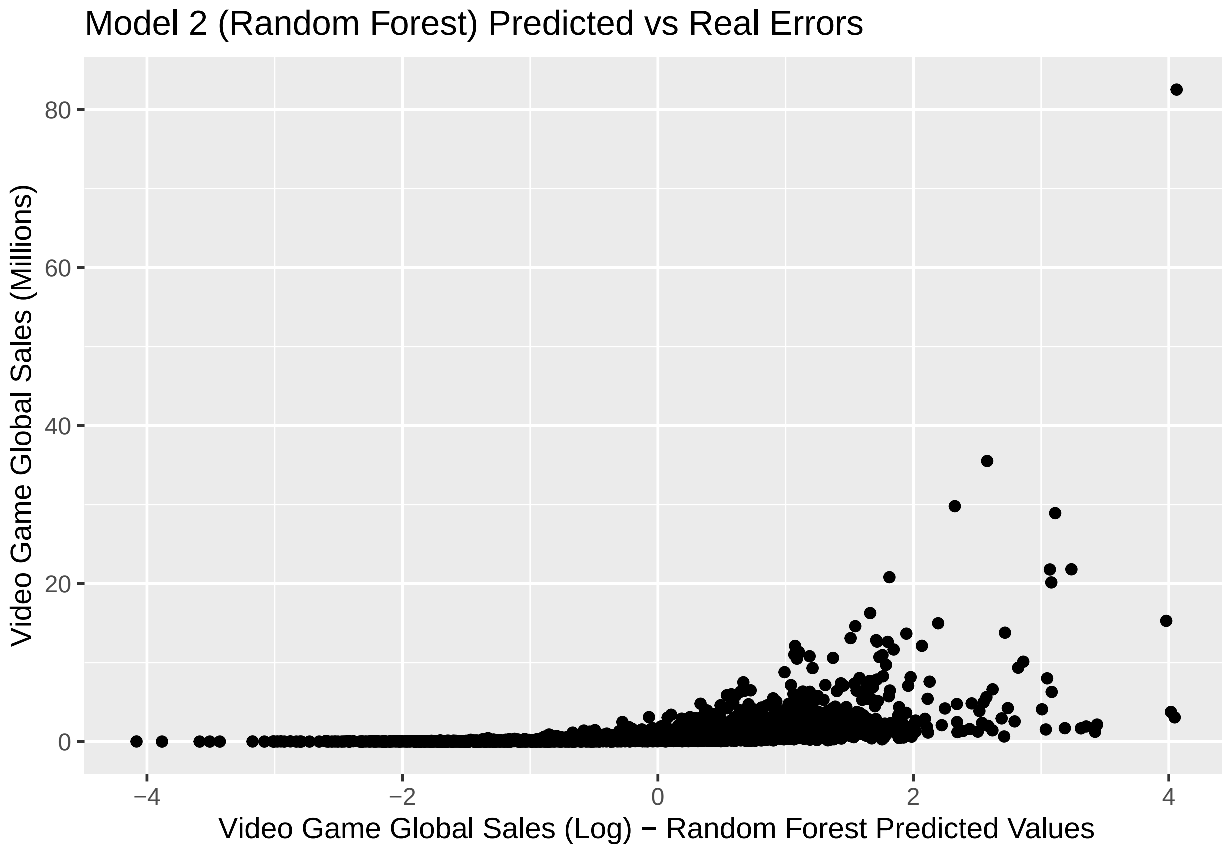
```
# Visualize predicted values vs real values

ggplot(testing_set) +
  geom_point(aes(log(Global_Sales), rf_predicted)) +
  geom_line(aes(log(Global_Sales), log(Global_Sales))) +
  xlab("Video Game Global Sales (Log)") +
  ylab("Random Forest Predicted Values") +
  ggtitle("Model 2 (Random Forest) Predicted vs Real Values")
```




```r
# Visualize errors for potential pattern

ggplot(testing_set) +
  geom_point(aes(log(Global_Sales) - rf_predicted, Global_Sales)) +
  xlab("Video Game Global Sales (Log) - Random Forest Predicted Values") +
  ylab("Video Game Global Sales (Millions)") +
  ggtitle("Model 2 (Random Forest) Predicted vs Real Errors")
```




```
# Results show that the errors are the largest for larger values of video game global sales
# Note, this is a weak point of the model that can be approved upon for future iterations
```

Conclusion

The goal for this project was to collect, manipulate, process, explore, and analyze data on Video Game Global Sales and Ratings from 2016.

Analysis showed that the critic score had a stronger relationship with global sales than the user score. Otherwise said, receiving a higher critic score was more important to global sales than higher user scores. As a video game fan for my entire life this is something that I knew, but it is revolutionary having the skills to analyze this fact with data on my own.

We also learned from our visual analysis that specific genres are much more popular than others. Specifically, we compared genres and platforms in which games were played that resulted in higher global sales. This was insightful even though we are in year 2023 to visualize where the video game sales were as of 2016.

Throughout this project we used multiple algorithms on our sample sizes to aid in predicting the global sales values. We started with a simple baseline method, tested and trained multiple other methods while making improvements to the RMSE and simultaneously taking steps back as well.

Our final results showed that our lowest RMSE was from our Random Forest model.

```
rf_rmse
```

```
## [1] 0.9550495
```

Though we received our lowest RMSE with Random Forest it is clear that this model was not ideal. The errors in some cases were very large and there was a pattern to the errors. Specifically, we visualized that the errors were largest for larger values of video game global sales.

Final thoughts, this model would need to be approved upon in the future in order to achieve lower RMSE and better performance for larger values of video game global sales. With our goals for this project achieved that concludes this project.