VGChartz Video Games Sales Dataset
HarvardX: Capstone Project - PH125.9x

Michael Vatt

09 JAN 2020

Contents

Overview

This project stemmed from the HarvardX Data Science Capstone Course PH125.9x. This time, the students were tasked with choosing their own dataset to wrangle, explore, and develop for their Machine Learning Algorithm. Due to my love of video games, I chose the Video Games Sales dataset from Kaggle. This project is outlined as follows: (1) Necessary packages are installed, (2) Dataset is loaded and setup configured, (3) Data Wrangling and familiarization with the dataset, (4) Perform exploratory data analysis, (5) Methods and analysis, (6) Modeling development and training, (7) Results of models and using best model on the validation set, and (8) the Conclusion. Exploratory analysis cannot be underestimated as it is critical to develop a machine learning algorithm that is both informative and meaningful in predicting anything. The final model is then explained and hopefully successful.

- Let's Install all Necessary Packages:

```
#############################################################
# Create Test Set, Validation Set, and Final Prediction
#############################################################

# Note: This Process Could Take a Couple of Minutes for Loading Required Packages

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(doParallel)) install.packages("doParallel", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(ggpubr)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

Introduction

This machine learning algorithm is dependent upon Video Games Sales from the VGChartz Dataset and corresponding ratings from Metacritic. This dataset was motivated by Gregory Smith's web scrape of VGChartz Video Game Sales. This dataset adds variables with another web scrape of Metacritic. Some observations in the dataset are incomplete, thus some data wrangling is necessary. This dataset includes the following variables:

Dataset Variables
Name
Platform
Year_of_Release
Genre
Publisher
NA_Sales
EU_Sales
JP_Sales
Other_Sales
Global_Sales
Critic_Score
Critic_Count
User_Score
User_Count
Developer

Objective

Our desire is to develop an ability to predict Global Sales with accuracy. The term used to give evaluation value to the algorithm is the Root Mean Square Error - also known as RMSE. This measure is a popular method for statistics, estimation, mathematics, and applications. It is used to measure the differences between values predicted by a model and the values observed.

RMSE is a measure of accuracy - without it, the algorithm is impractical and ineffective. The lower an RMSE score, the better! The goal of this project is to develop a model that will produce an RMSE below 0.865 - a lofty goal.

The effect errors have on the RMSE is proportional to the size of the squared error - larger the error, larger the effect. A means to help navigate this problem is built into the formula by reducing the impact of the errors through the square root function.

This project's machine learning algorithm was established through an incremental and iterative development process. Different models were established in layers to compare RMSE results in order to evaluate their prediction qualities. This will enable the best model to be determined and used to predict movie ratings on the dataset.

Dataset

- WARNING: Some Models in this Project Require a Lot of Computing Power. Splitting the Cores Will Aid in Parallel Processing of CPUs
- NOTE: The Support Vector Machine and Random Forest Models Will Take the Longest Time

```
split <- detectCores()
split # To Make Sure it Worked
```

```
## [1] 8
```

```
cl <- makePSOCKcluster(split)
registerDoParallel(cl)
```

Let's Load the Dataset from a CSV File Located in my GitHub Account

```
file <- tempfile()
download.file("https://raw.githubusercontent.com/mjvatt/Create-Your-Own-Capstone-Project/master/Video_Ga
vgs <- fread("Video_Games_Sales_as_at_22_Dec_2016.csv", header = TRUE)
```

Familiarization with the Dataset:

```
dim(vgs) # For Familiarity
```

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

```
str(vgs) # Notice that Some Columns Are Not The Same Class
```

```
## Classes 'data.table' and '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 ...
```

Company	Company	Company	Company	Company	Company	Company
Microsoft	Nintendo	Sony	Sony	Nintendo	Microsoft	Sony

```
##  $ 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" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
class(vgs) # Let's Ensure it is Correct Format
```

```
## [1] "data.table" "data.frame"
```

Now that we are familiar with the dataset, we need to do some data wrangling to clean and prepare the data for ease of use.

```r
vgs$User_Count <- as.numeric(as.character(vgs$User_Count))
vgs$User_Score <- as.numeric(as.character(vgs$User_Score))
```

```
## Warning: NAs introduced by coercion
```

```r
vgs2 <- na.omit(vgs)
dim(vgs2) # To See How it Has Changed
```

```
## [1] 7017   16
```

I noticed a variable that did not exist in the Dataset that I think would be fun to have in there - so let's create it.

```r
# First, Let's Make Some Company Variables

microsoft <- c('X360', 'XB', 'XOne')
nintendo <- c('Wii', 'NES', 'GB', 'DS', 'SNES', 'WiiU', '3DS', 'GBA', 'GC', 'N64')
sony <- c('PS', 'PS2', 'PS3', 'PS4', 'PSP', 'PSV')

# Here We Define the Criteria to Identify What Belongs to Each Company

companies <- function(c) {
  if(c %in% microsoft) {return('Microsoft')}
  else if(c %in% nintendo) {return('Nintendo')}
  else if(c %in% sony) {return('Sony')}
}

# Now We Can Create This New Column

vgs2$companies <- sapply(vgs2$Platform, companies)

# Let's Check it Out to Make Sure it Worked

vgs2$companies[9:15] %>% kable(col.names = "Company")
```

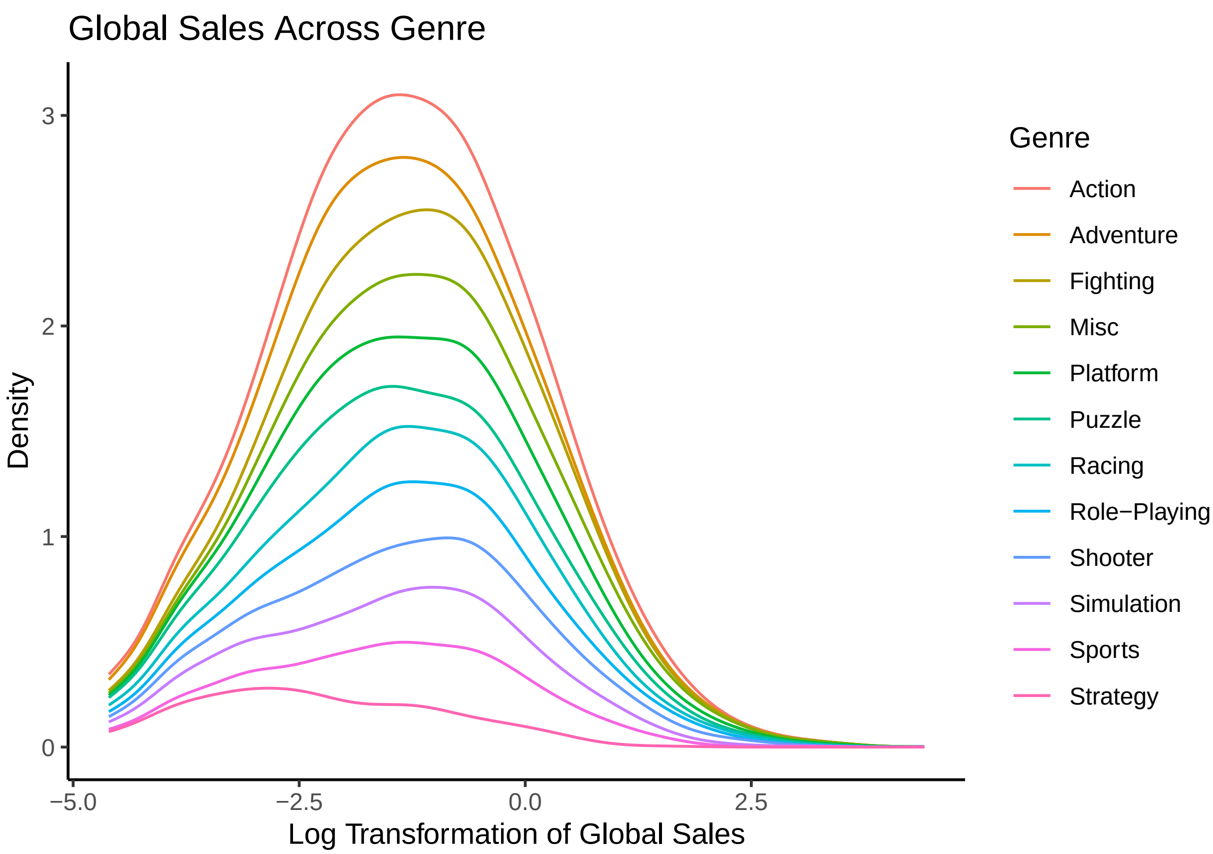
Methods and Analysis

Exploratory Data Analysis

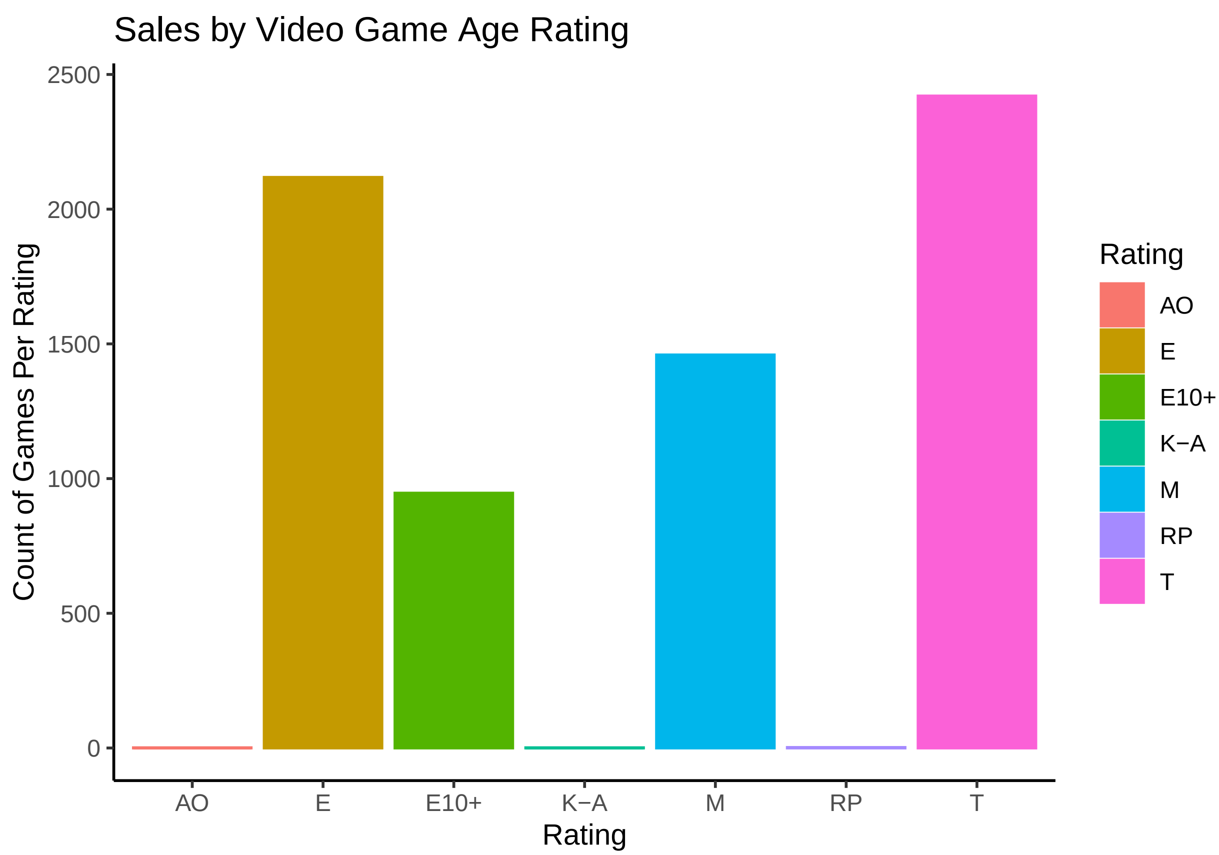
Distribution of Global Sales Across Genres and Ratings

First, let's look at the Distribution of Global Sales Across Genres in the Data. We will also look at Video Games Sales based on their Rating (NOTE: Rating means "Maturity Rating Level" and has no connection or implication on User or Critic Scores given to the video games.)

```
vgs2 %>% ggplot(aes(log(Global_Sales))) + stat_density(aes(color = Genre), geom = "line") +
  labs(x = "Log Transformation of Global Sales", y = "Density") + ggtitle("Global Sales Across Genre") +
  theme_classic()
```




```
vgs2 %>% select(Rating) %>% filter(!Rating == '') %>% count(Rating) %>%
  ggplot(aes(Rating, n, fill = Rating, color = Rating)) +
  geom_bar(stat = "identity") + ylab("Count of Games Per Rating") +
  ggtitle("Sales by Video Game Age Rating") +
  theme_classic() # Notice we Have Three Ratings with No Data (We'll Remove Below)
```

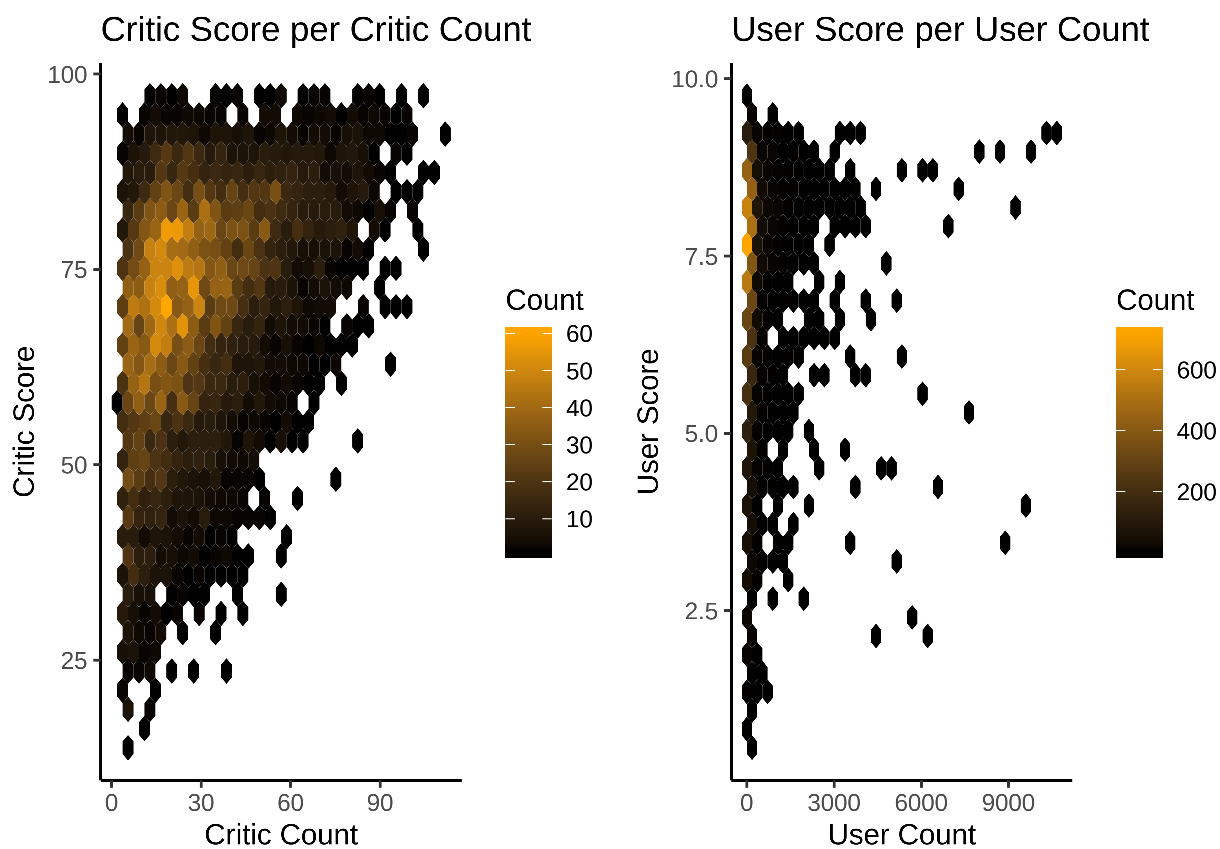


I'd say this chart showed about what would be expected as certain types of games have more mass appeal, across genders, races and languages. Whilst others - for example, sports - while popular in the US, may not have such an appeal for the global masses. The Ratings chart was also not a surprise - but notice that we have three ratings with no data, we'll remove those types as they are not informative.

Relationship Between Critic Score/Critic Count and User Score/User Count

Now, let's take a look at a possible relationship between Critic's data and User's data:

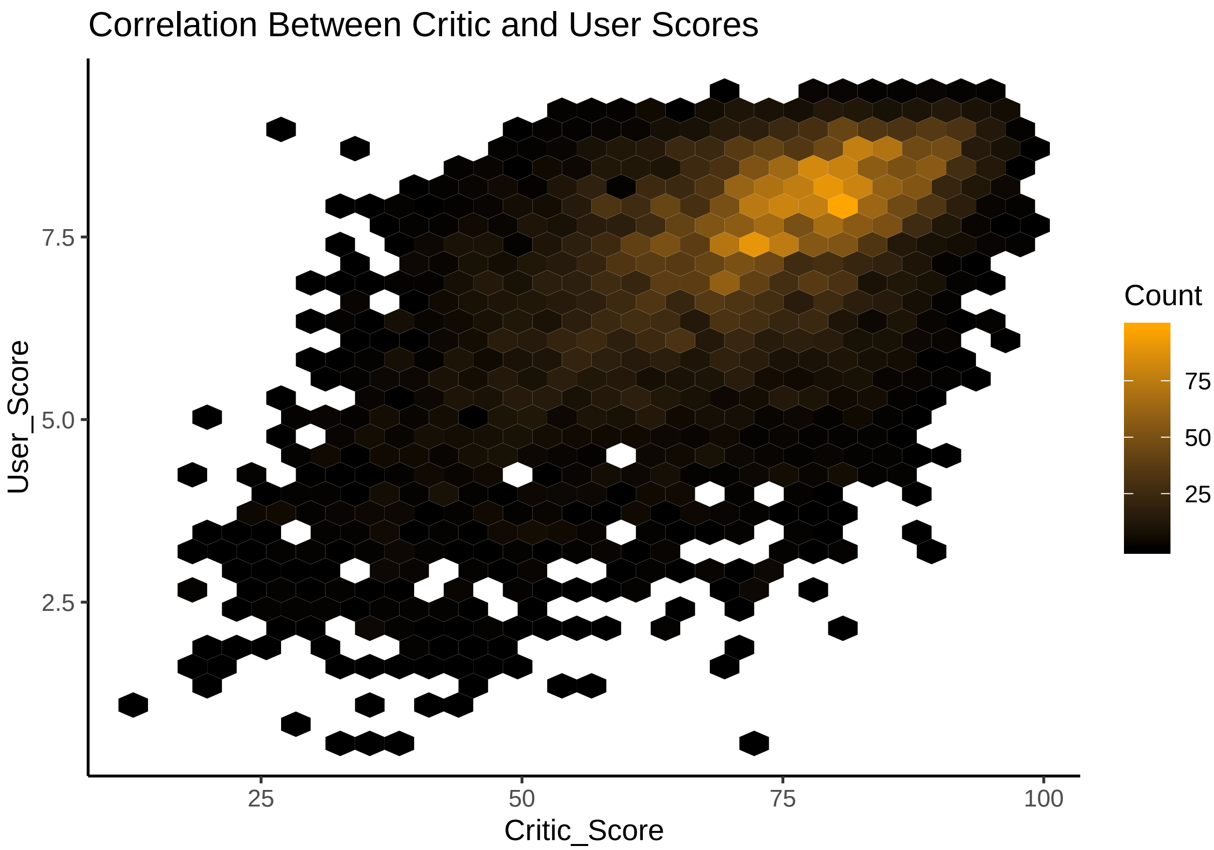
```
plot3 <- vgs2 %>% ggplot(aes(Critic_Count, Critic_Score)) + stat_binhex() +
  scale_fill_gradientn(colors = c("black", "orange"), name = "Count") +
  labs(x = "Critic Count", y = "Critic Score") + ggtitle("Critic Score per Critic Count") +
  theme_classic()

plot4 <- vgs2 %>% ggplot(aes(User_Count, User_Score)) + stat_binhex() +
  scale_fill_gradientn(colors = c("black", "orange"), name = "Count") +
  labs(x = "User Count", y = "User Score") + ggtitle("User Score per User Count") +
  theme_classic()

# For Easy Visual Comparison

grid.arrange(plot3, plot4, nrow=1, ncol=2)
```



Correlation Between Critic and User Scores

Is there a correlation between Critic and User Scores?

```
vgs2 %>% ggplot(aes(Critic_Score, User_Score)) + stat_binhex() +
  scale_fill_gradientn(colors = c("black", "orange"), name = "Count") +
  ggtitle("Correlation Between Critic and User Scores") + theme_classic()
```



It appears that there is some sort of correlation. But the User Scores appear to be more "favorable" or "positive" than critics. Let's actually see what that percentage is for scores:

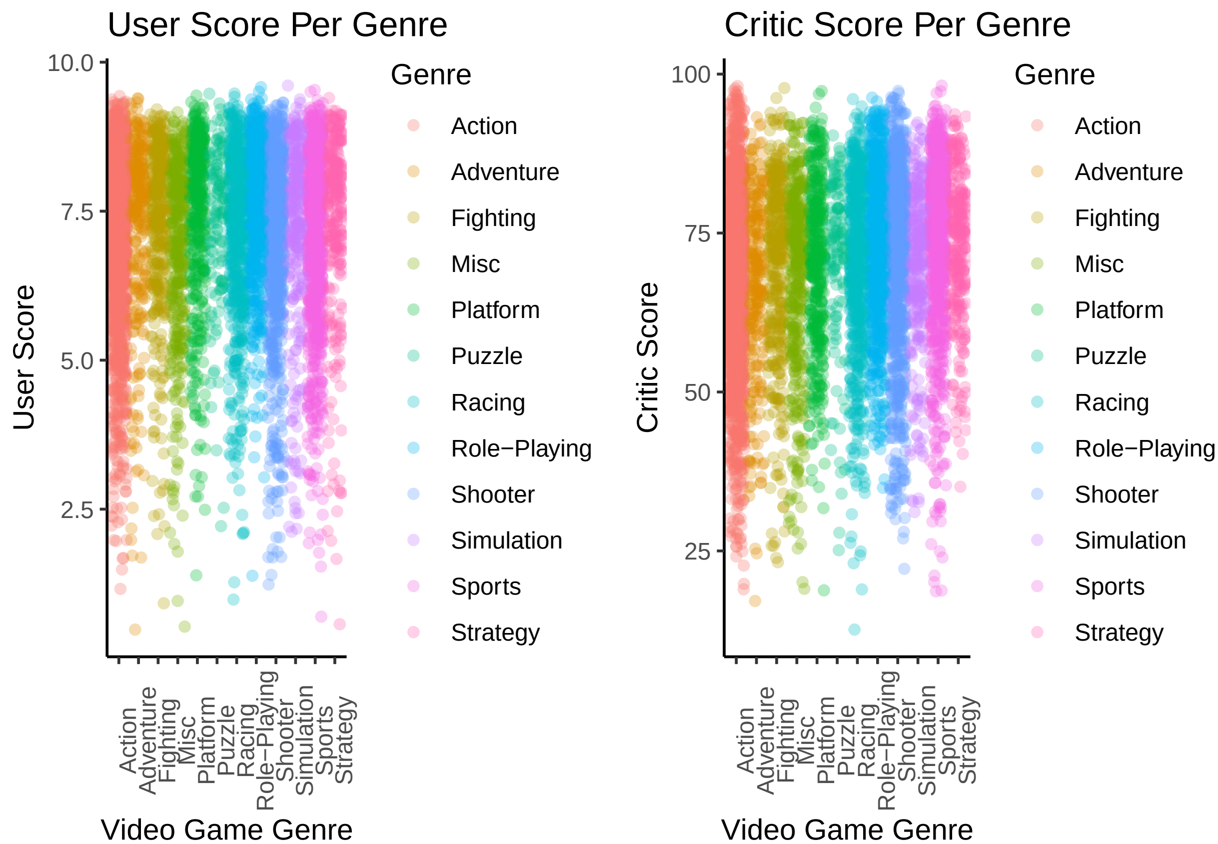
```
cor(vgs2$User_Score, vgs2$Critic_Score, use = "complete.obs") %>% kable()
```

x
0.5808778

Scores Comparison

Now for more detailed scores comparisons:

```
plot9 <- vgs2 %>% select(Genre, User_Score) %>% ggplot(aes(Genre, User_Score, color = Genre)) +
  geom_jitter(alpha = .3) + labs(x = "Video Game Genre", y = "User Score") + ggtitle("User Score Per Ge
  theme_classic() + theme(axis.text.x = element_text(angle = 90))

plot10 <- vgs2 %>% select(Genre, Critic_Score) %>% ggplot(aes(Genre, Critic_Score, color = Genre)) +
  geom_jitter(alpha = .3) + labs(x = "Video Game Genre", y = "Critic Score") +
  ggtitle("Critic Score Per Genre") + theme_classic() + theme(axis.text.x = element_text(angle = 90))

# For Easy Visual Comparison

grid.arrange(plot9, plot10, nrow = 1, ncol = 2)
```

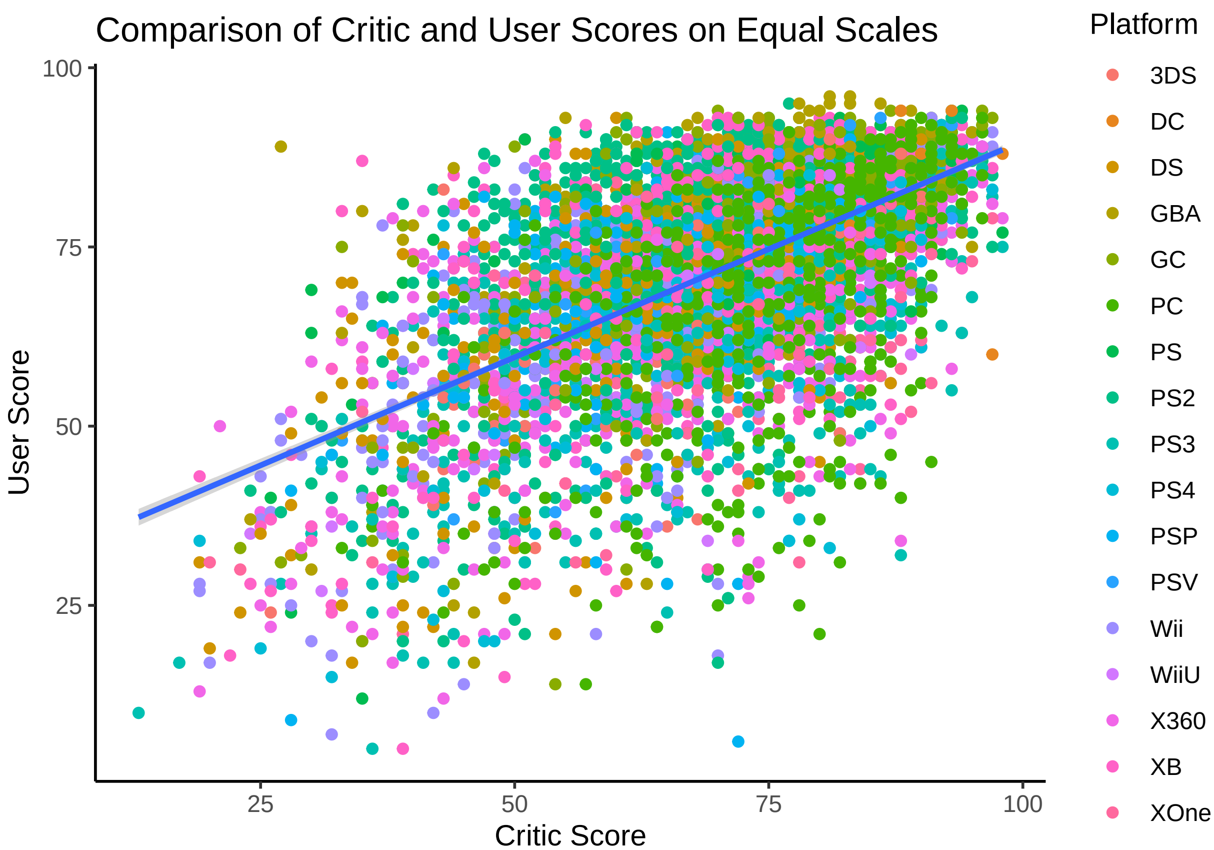


This definitely seems to support the idea that User Scores are generally more favorable or positive towards the games - irrespective of genre - than Critic Scores.

Critic vs. User Scores Scaled

Did you notice that the User Scores and Critic Scores were measured differently? Let's correct that and see if it changes anything.

```
vgs3 <- vgs2

# Let's Do Some More Comparison Between Critic and User Scores
# We Should Scale the User Score to Make it Easier to Compare to Critics (NOTE: Critics had a 0:100 sca

vgs3$User_Score_Scaled <- as.numeric(as.character(vgs3$User_Score)) * 10

# Now, Let's Compare the Critic and User Scores Once Again

vgs3 %>% ggplot(aes(Critic_Score, User_Score_Scaled)) + geom_point(aes(color = Platform)) +
  geom_smooth(method = "lm", size = 1) + labs(x = "Critic Score", y = "User Score") +
  ggtitle("Comparison of Critic and User Scores on Equal Scales") + theme_classic()
```

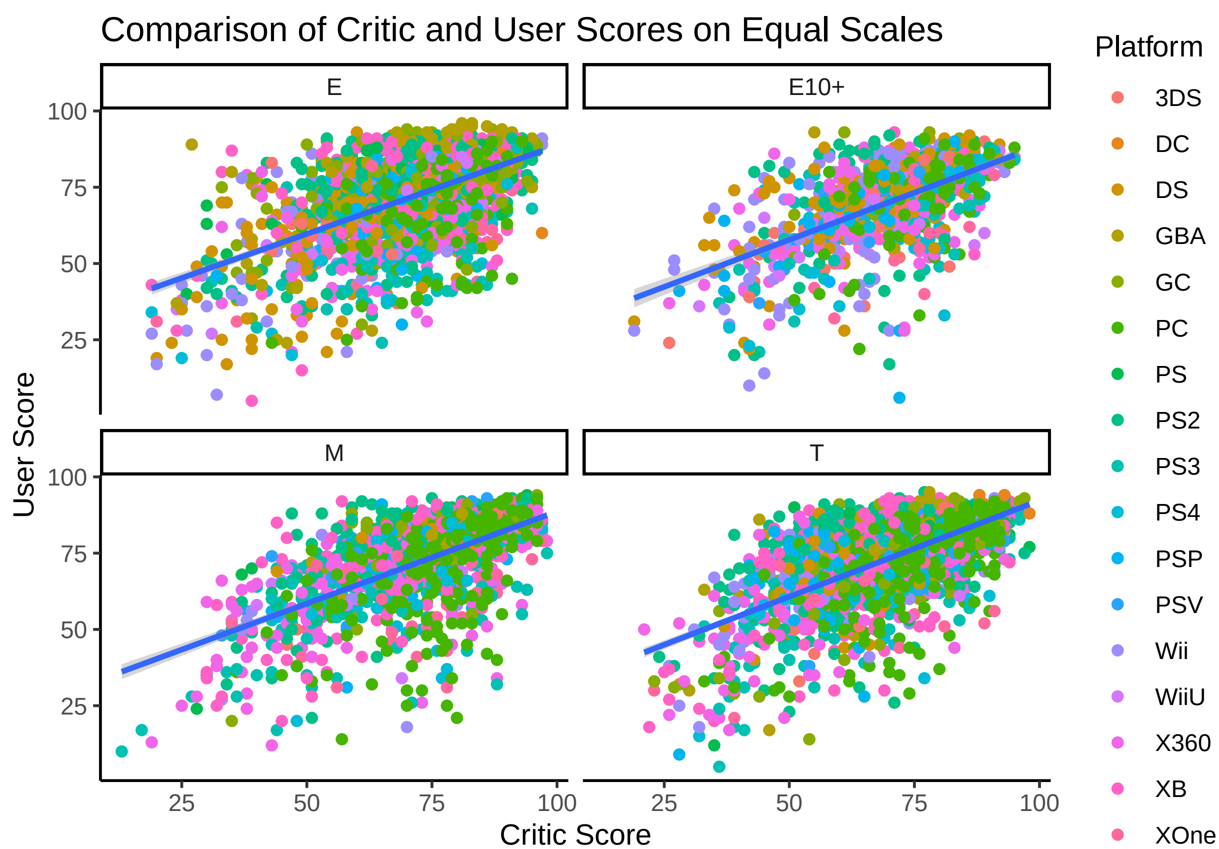


Mathematically, scaling the user scores would explain that, no, it will not change anything as it still represents the same figures despite the change in what the figures display as.

Let's Compare Critic and User Scores Through Ratings and Systems

What about a comparison of Critic and User Scores due to maturity ratings and systems?

```
vgs3 %>% filter(Rating %in% c("E", "E10+", "M", "T")) %>%
  ggplot(aes(Critic_Score, User_Score_Scaled)) + geom_point(aes(color = Platform)) +
  geom_smooth(method = "lm", size = 1) + facet_wrap(~Rating) +
  labs(x = "Critic Score", y = "User Score") +
  ggtitle("Comparison of Critic and User Scores on Equal Scales") + theme_classic()
```

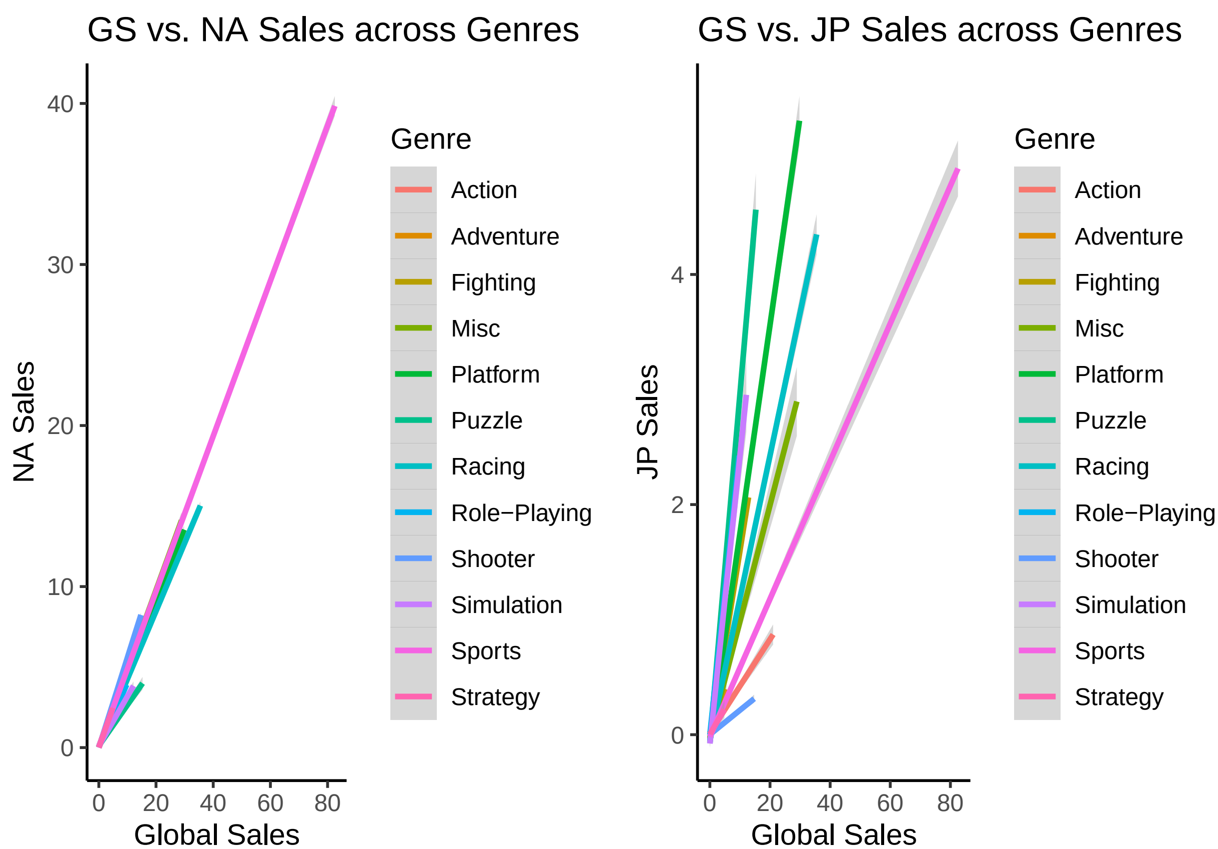


They all seem to follow a well-developed theme.

Global Sales Comparison with Regional Sales

Let's take a peek at some regional sales:

```
plot6 <- vgs2 %>% select(Global_Sales, NA_Sales, Genre) %>%
  ggplot(aes(Global_Sales, NA_Sales, color = Genre)) + stat_smooth(method = "lm") +
  labs(x = "Global Sales", y = "NA Sales") + ggtitle("GS vs. NA Sales across Genres") +
  theme_classic()

plot7 <- vgs2 %>% select(Global_Sales, JP_Sales, Genre) %>%
  ggplot(aes(Global_Sales, JP_Sales, color = Genre)) + stat_smooth(method = "lm") +
  labs(x = "Global Sales", y = "JP Sales") + ggtitle("GS vs. JP Sales across Genres") +
  theme_classic()

# For Easy Visual Comparison

grid.arrange(plot6, plot7, nrow=1, ncol=2)
```



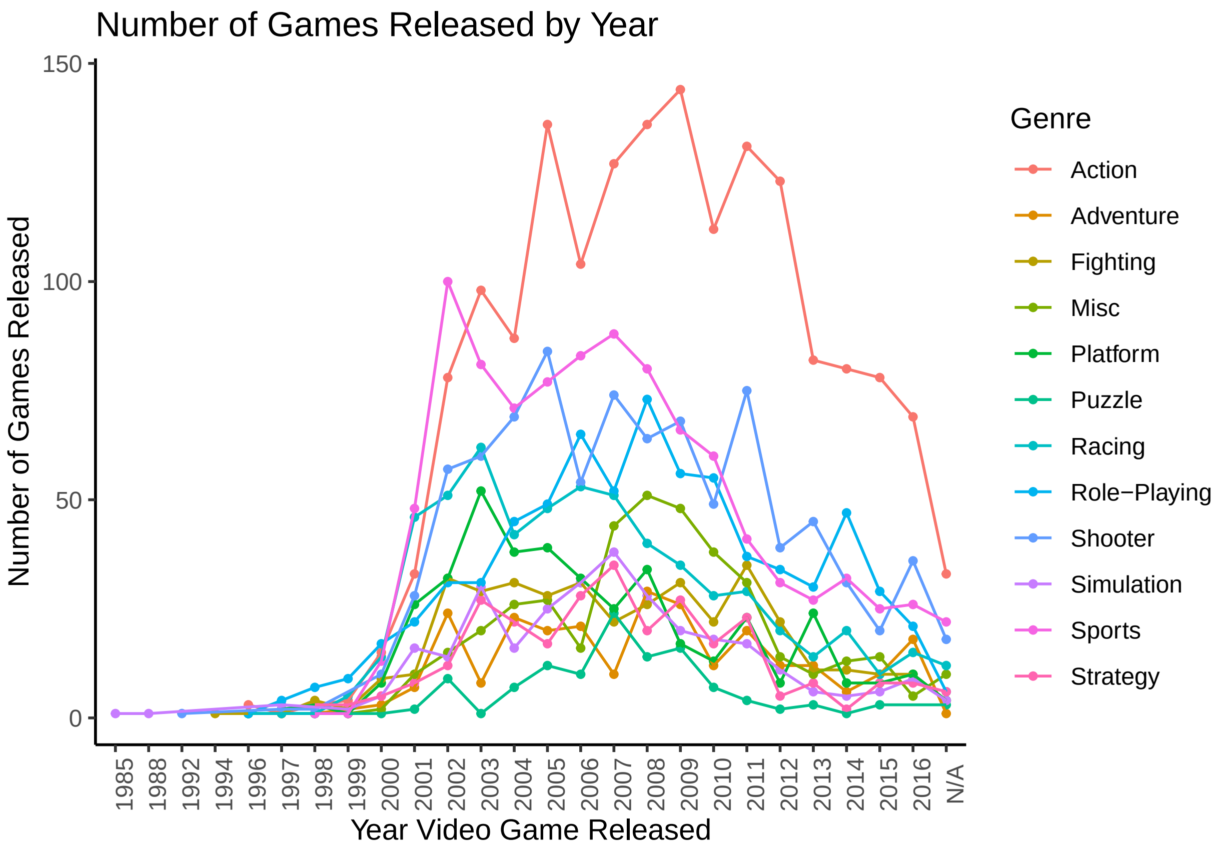
That is interesting but it doesn't appear to be highly enlightening just yet.

Data Analysis

Now to Figure Out How Many Video Games Are Released Each Year

We definitely need to see how the video games industry has behaved in terms of numbers of video games released by year:

```
vgs_per_year <- vgs2 %>% select(Name, Genre, Year_of_Release, Rating) %>%
  group_by(Year_of_Release, Genre) %>% summarize(gamescount = n())

vgs_per_year %>% ggplot(aes(x = Year_of_Release, y = gamescount, group = Genre, color = Genre)) +
  stat_sum(size = 1) + geom_line() + labs(x = "Year Video Game Released", y = "Number of Games Released
  ggtitle("Number of Games Released by Year") + theme_classic() +
  theme(axis.text.x = element_text(angle = 90))
```

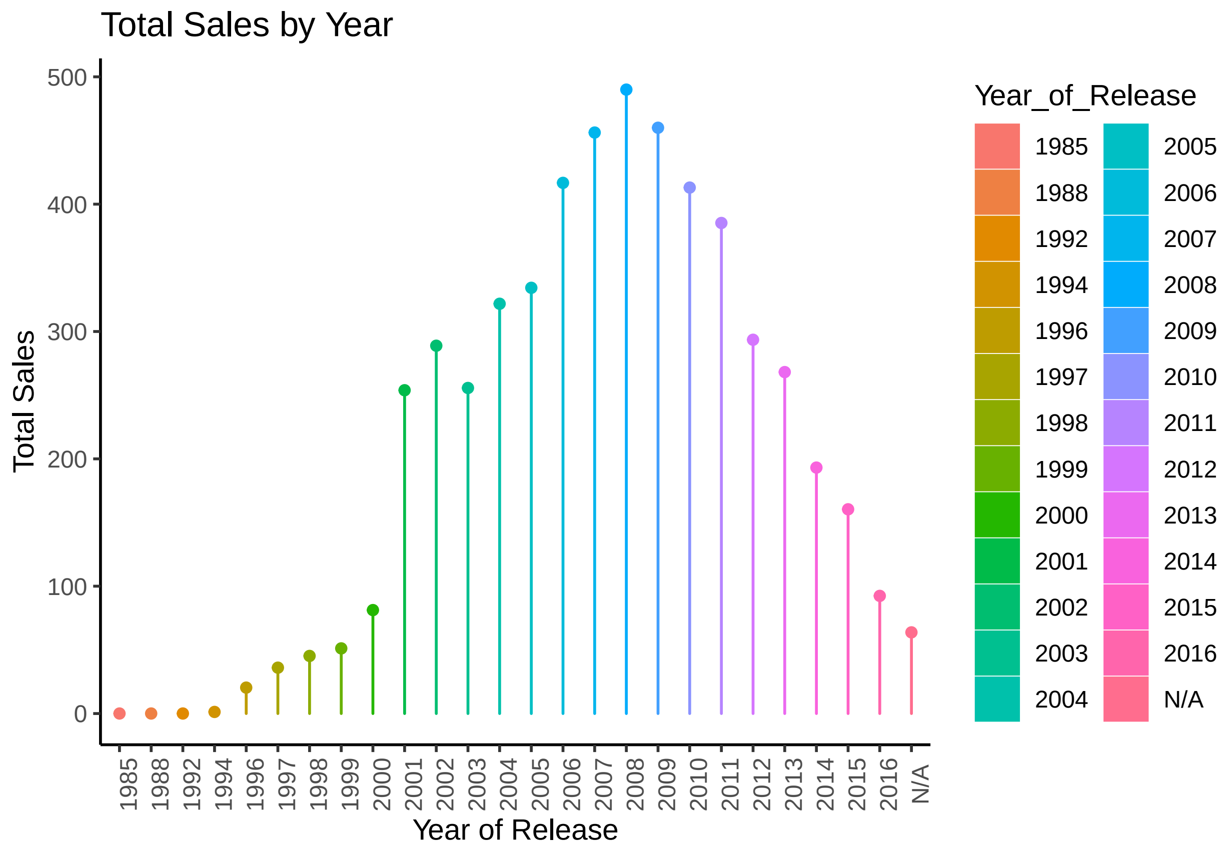


Here we can see the temporal trend of the video game industry the past three decades. Started incredibly slow, a massive expansion, and then a downturn in the third decade.

What are the Annual Video Games Sales?

Annual sales of video games is an important insight into the trend of the video gaming industry. Let's take a look:

```
vgs2 %>% select(Year_of_Release, Global_Sales) %>% group_by(Year_of_Release) %>%
  summarize(GS = sum(Global_Sales)) %>%
  ggplot(aes(Year_of_Release, GS, fill = Year_of_Release, color = Year_of_Release)) +
  geom_area() + geom_point() +
  labs(x = "Year of Release", y = "Total Sales") + ggtitle("Total Sales by Year") +
  theme_classic() + theme(axis.text.x = element_text(angle = 90))
```

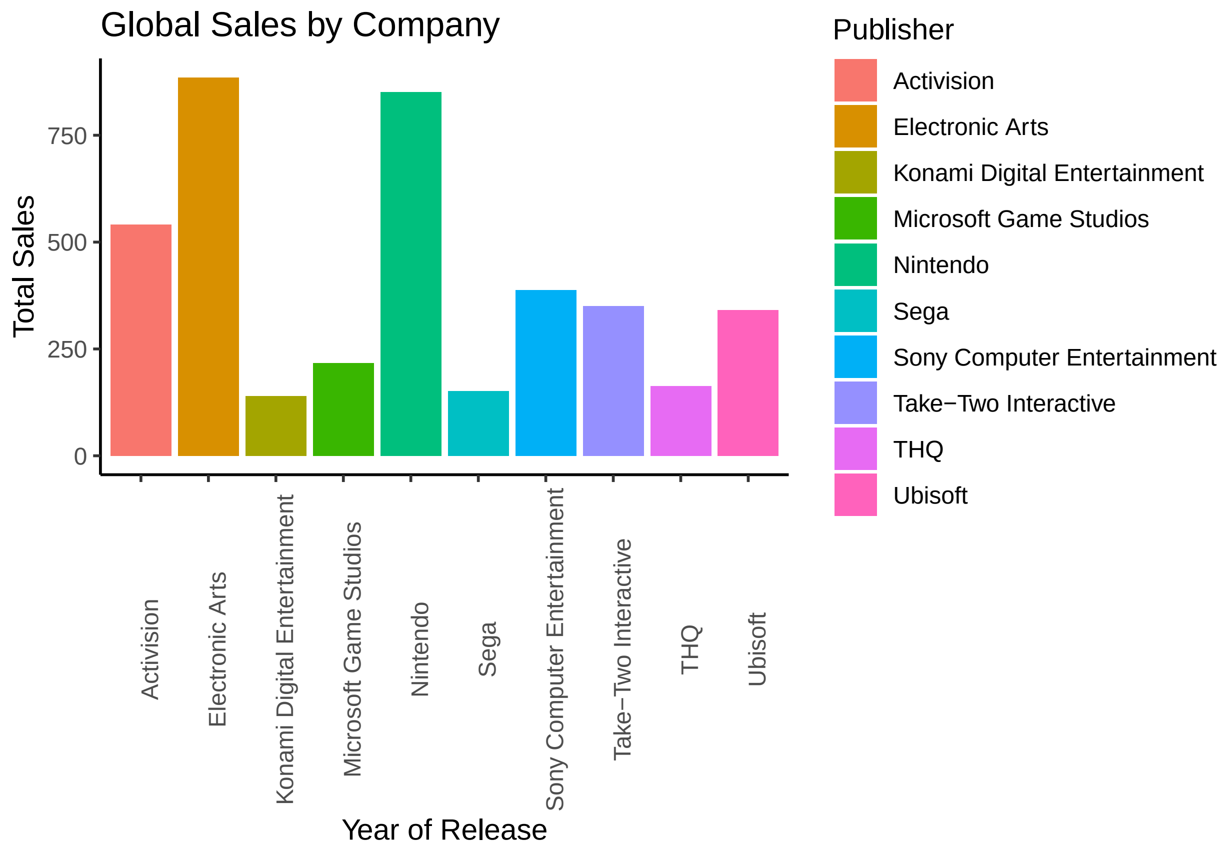


This seems to greatly support the temporal trend of video games released by year.

Top Video Game Companies

Who are the top video game companies by Global Sales?

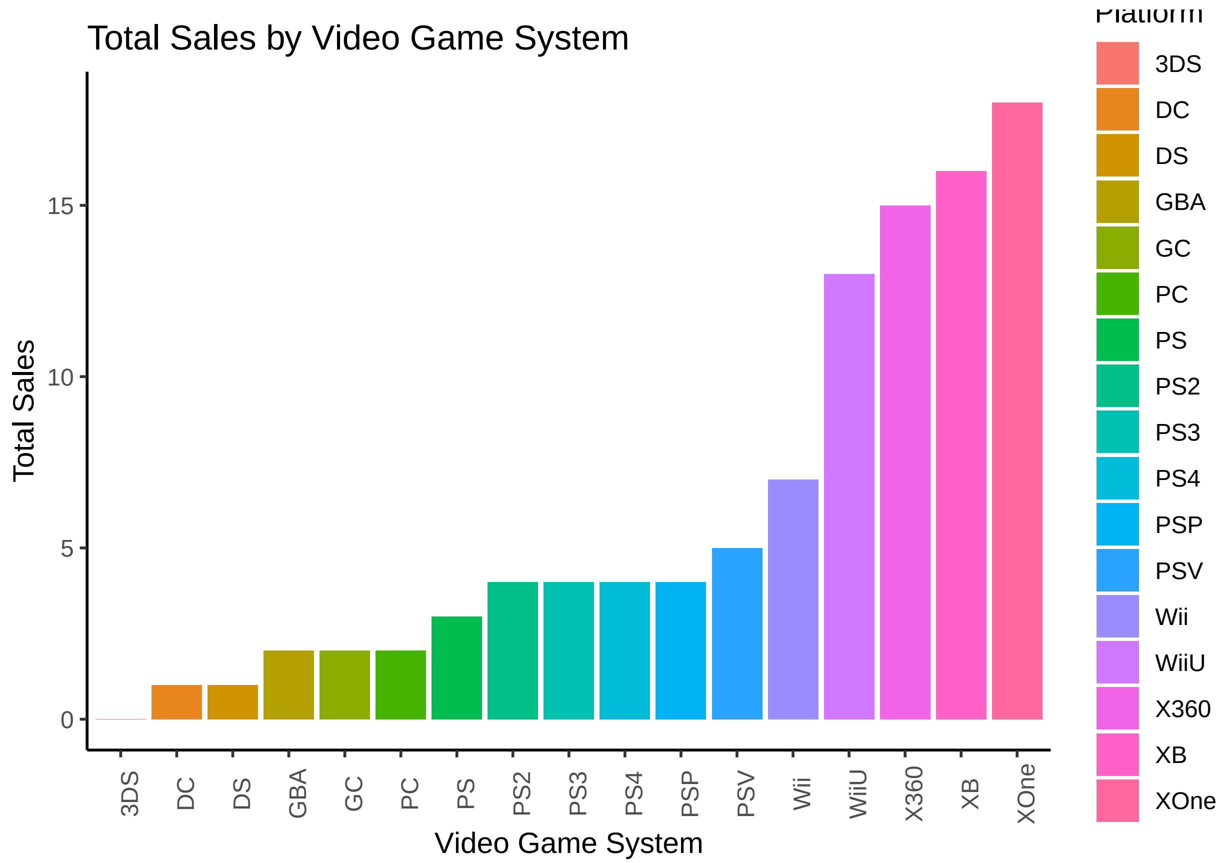
```
vgs2 %>% select(Publisher, Global_Sales) %>% group_by(Publisher) %>% summarize(GS = sum(Global_Sales))
  arrange(desc(GS)) %>% head(10) %>%
  ggplot(aes(Publisher, GS, fill = Publisher)) + geom_bar(stat="identity") +
  labs(x = "Year of Release", y = "Total Sales") + ggtitle("Global Sales by Company") + theme_classic()
  theme(axis.text.x = element_text(angle = 90))
```



Top Video Game Systems

What are the best-selling video game systems?

```
systems <- vgs2 %>% select(Platform, Global_Sales) %>% group_by(Platform) %>%
  summarize(GS = sum(Global_Sales))
systems$market_share <- round(systems$GS / sum(systems$GS) * 100) # Calculating Market Share of Each Sy
systems$market_share <- sort(systems$market_share) # Simply Sorting it for Ease and Clarity

systems %>% ggplot(aes(Platform, market_share, fill = Platform)) + geom_bar(stat = "identity") +
  labs(x = "Video Game System", y = "Total Sales") + ggtitle("Total Sales by Video Game System") +
  theme_classic() + theme(axis.text.x = element_text(angle = 90))
```



Best-Selling Video Games

What about the best-selling video games?

```
vgs2 %>% select(Name, Global_Sales) %>% group_by(Name) %>%
  summarize(GS = sum(Global_Sales)) %>% arrange(desc(GS)) %>%
  head(10) %>% kable(col.names = c("Video Game Name", "Global Sales"))
```

Video Game Name	Global Sales
Wii Sports	82.53
Grand Theft Auto V	56.57
Mario Kart Wii	35.52
Wii Sports Resort	32.77
Call of Duty: Black Ops	30.82
Call of Duty: Modern Warfare 3	30.59
New Super Mario Bros.	29.80
Call of Duty: Black Ops II	29.40
Wii Play	28.92
New Super Mario Bros. Wii	28.32

Top 10 Best-Rated Games by Users

Now, let's view the differences (or similarities) between the top 10 best-rated games by Users and Critics.

```r
vgs2 %>% select(Name, User_Score) %>% group_by(Name) %>% summarize(meanus = mean(User_Score)) %>%
  arrange(desc(meanus)) %>% head(10) %>% kable(caption = "Top 10 Rated Games by Users", col.names = c("
```

Table 4: Top 10 Rated Games by Users

Video Game Name	Avg User Score
Boktai: The Sun is in Your Hand	9.6
Harvest Moon: Friends of Mineral Town	9.6
Golden Sun: The Lost Age	9.5
Karnaaj Rally	9.5
MLB SlugFest Loaded	9.5
Super Puzzle Fighter II	9.5
Wade Hixton's Counter Punch	9.5
Advance Wars 2: Black Hole Rising	9.4
Backyard Baseball	9.4
Castlevania: Symphony of the Night	9.4

Top 10 Best-Rated Games by Critics

Seems to be very little in common for the Top 10 best-rated video games for the two groups.

```r
vgs2 %>% select(Name, Critic_Score) %>% group_by(Name) %>% summarize(meancs = mean(Critic_Score)) %>%
  arrange(desc(meancs)) %>% head(10) %>% kable(caption = "Top 10 Rated Games by Critics", col.names = c
```

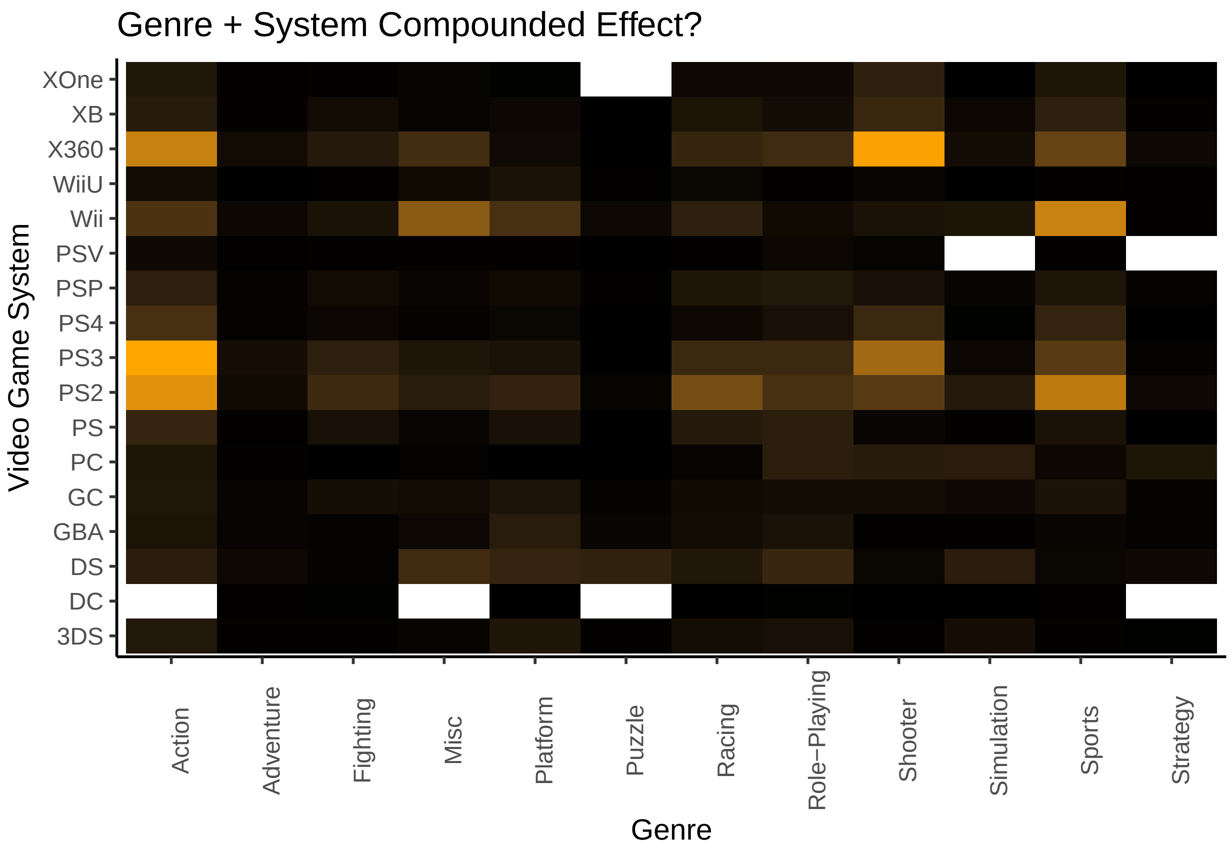
Table 5: Top 10 Rated Games by Critics

Video Game Name	Avg Critic Score
SoulCalibur	98.0
Metroid Prime	97.0
NFL 2K1	97.0
Super Mario Galaxy	97.0
Super Mario Galaxy 2	97.0
Grand Theft Auto V	96.8
Tony Hawk's Pro Skater 2	96.5
Gran Turismo	96.0
Metal Gear Solid 2: Sons of Liberty	96.0
Tekken 3	96.0

Compounded Effect: Genre and System?

What about the possibility of a compounded effect on sales of the variables Genre and System together?

```r
vgs2 %>% select(Platform, Genre, Global_Sales) %>% group_by(Platform, Genre) %>%
  summarize(GS = sum(Global_Sales)) %>% ggplot(aes(Genre, Platform, fill = GS)) +
  geom_tile() + scale_fill_gradientn(colors = c("black", "orange")) +
  labs(x = "Genre", y = "Video Game System") + ggtitle("Genre + System Compounded Effect?") +
  theme_classic() + theme(legend.position = "none", axis.text.x = element_text(angle = 90))
```

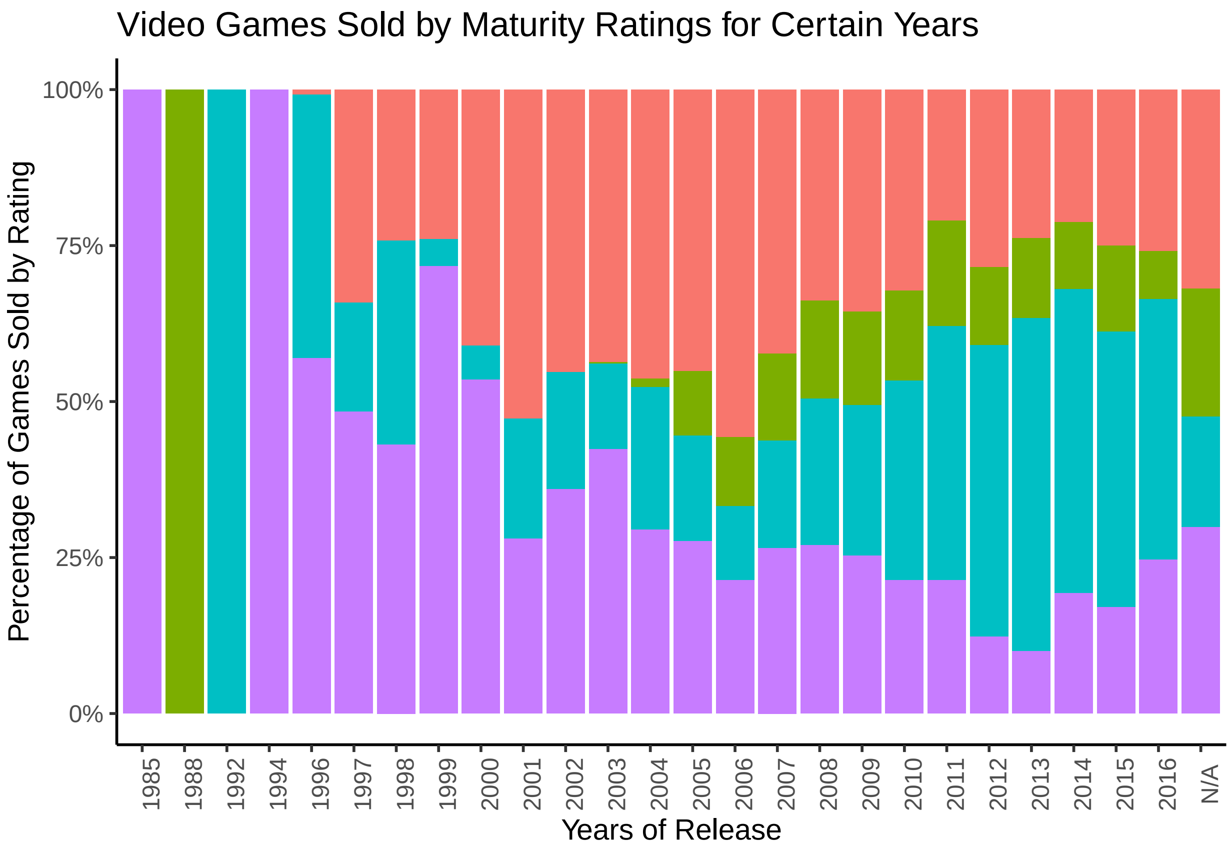


Definitely seems like there's a correlation with hot spots in specific locations.

Popularity Due to Rating?

Maybe it's simply the maturity rating that is driving popularity?

```r
vgs2 %>% select(Year_of_Release, Global_Sales, Rating) %>%
  filter(Rating %in% c("E", "E10+", "M", "T")) %>% group_by(Year_of_Release, Rating) %>%
  summarize(GS = sum(Global_Sales)) %>% arrange(desc(GS)) %>%
  ggplot(aes(Year_of_Release, GS, group = Rating, fill = Rating)) +
  geom_bar(stat = "identity", position = "fill") + scale_y_continuous(labels = percent) +
  labs(x = "Years of Release", y = "Percentage of Games Sold by Rating") +
  ggtitle("Video Games Sold by Maturity Ratings for Certain Years") + theme_classic() +
  theme(legend.position = "none", axis.text.x = element_text(angle = 90))
```

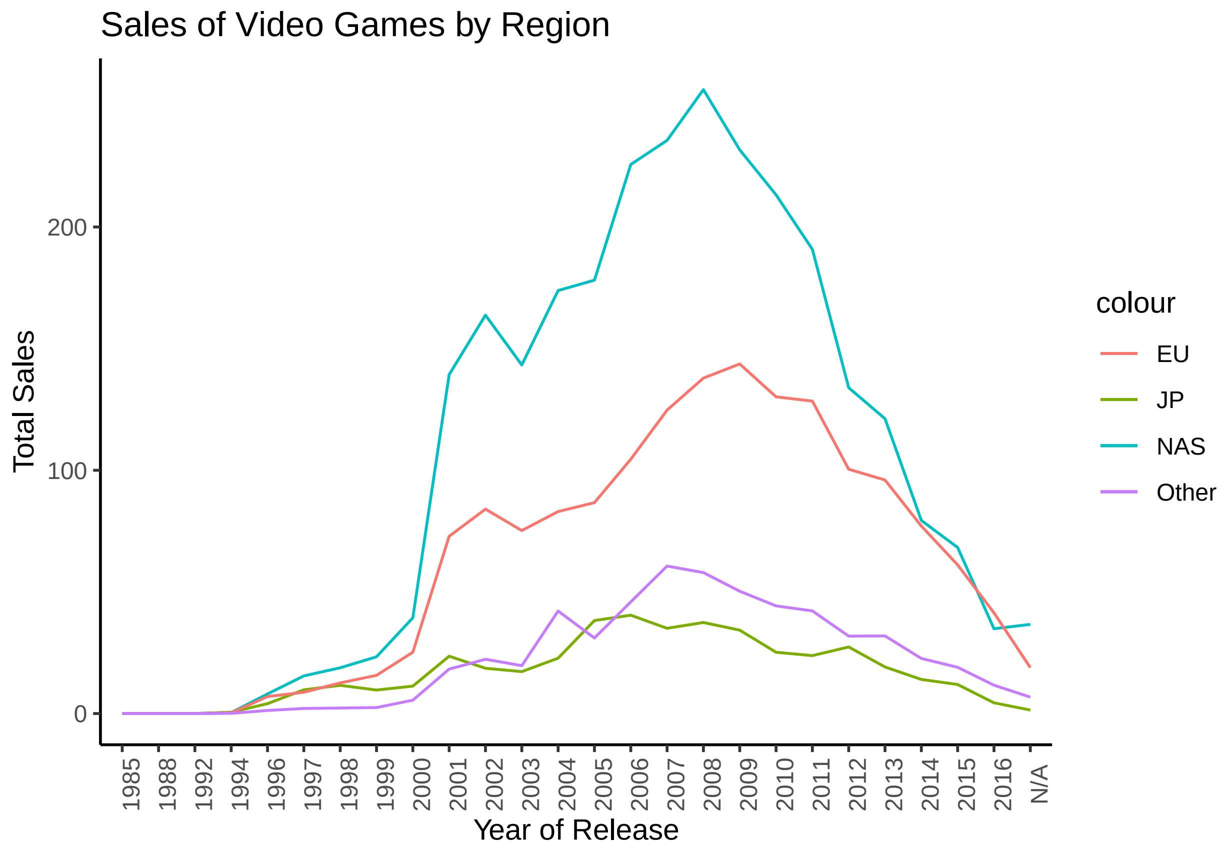


Nope, this chart would suggest that maturity rating has very little impact on popularity or sales.

Overall Sales by Regions

What about sales driven by specific regions? We know North America is a hotbed for video games industry, but what about others?

```
vgs2 %>% select(Year_of_Release, NA_Sales, JP_Sales, EU_Sales, Other_Sales) %>%
  group_by(Year_of_Release) %>%
  summarize(NAS = sum(NA_Sales), JP = sum(JP_Sales), EU = sum(EU_Sales), Other = sum(Other_Sales)) %>%
  ggplot(aes(Year_of_Release, NAS, group = 1, color = "NAS")) +
  geom_line() + geom_line(aes(y = JP, color = "JP")) + geom_line(aes(y = EU, color = "EU")) +
  geom_line(aes(y = Other, color = "Other")) + labs(x = "Year of Release", y = "Total Sales") +
  ggtitle("Sales of Video Games by Region") + theme_classic() + theme(axis.text.x = element_text(angle
```



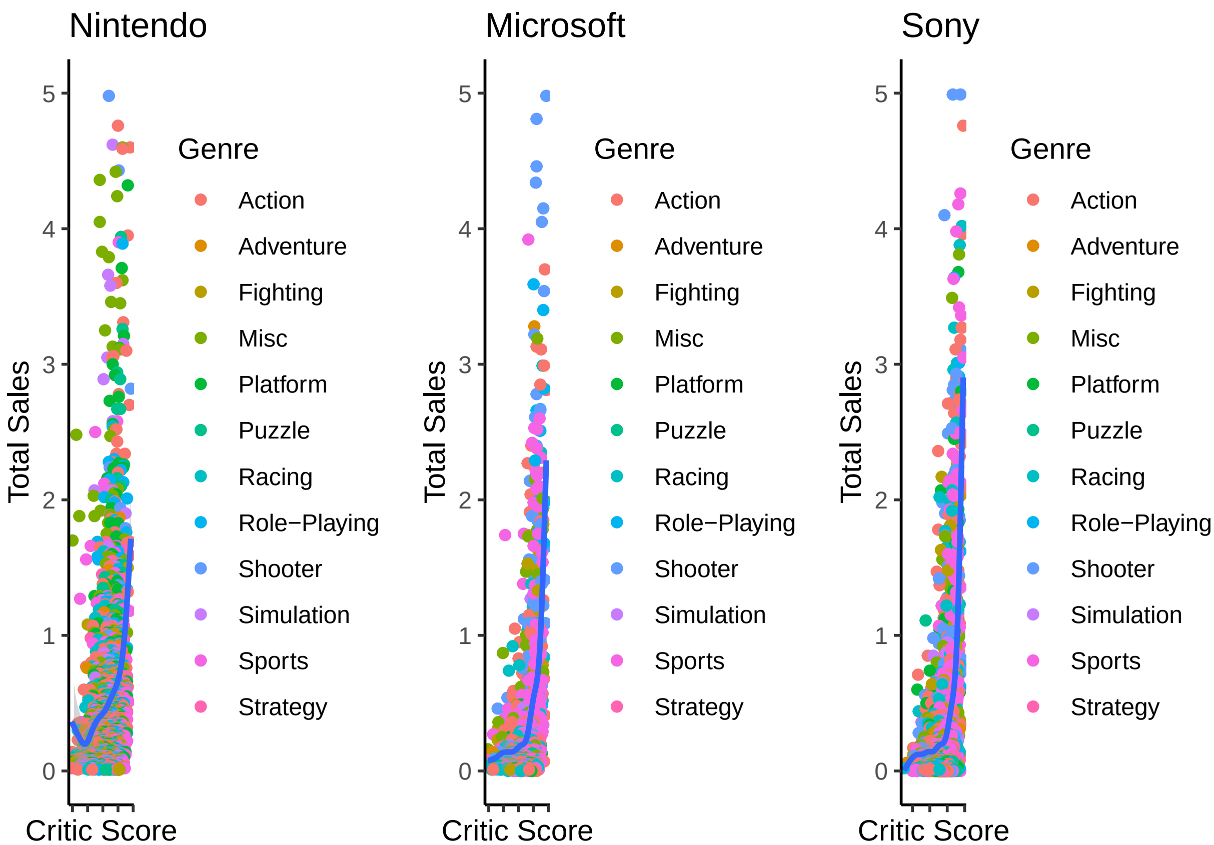
This chart makes absolute sense. It is as predicted for North America; Japan is a major hub of technology and is where Nintendo was founded and continues to develop but is overall a smaller market in comparison; Europe is a larger market and actually has many game development and visual FX studios there; Lastly, the "Other Region" is comprised with the rest of the world and is not surprising. Despite areas like South Korea, these are incredibly small markets in comparison to the others. The strict rules of China, despite being a highly advanced technological society, does not indicate that it has a large video gaming following.

Predicting Sales as a Function of Critic Scores

Since critic scores seem to be less favorable or negative towards video games than user scores, let's take a look at global sales as a function of critic scores. The scores go from negative to positive (left to right):

```
# Let's Look at the Three Major Video Game Companies - Nintendo, Microsoft, and Sony

# Microsoft

microsoft_plot <- vgs3 %>% filter(vgs3$companies == 'Microsoft') %>%
  ggplot(aes(Critic_Score, NA_Sales)) + geom_point(aes(color = Genre)) +
  ylim(0, 5) + geom_smooth() + labs(x = "Critic Score", y = "Total Sales") +
  ggtitle("Microsoft") + theme_classic() + theme(axis.text.x = element_blank())

# Nintendo

nintendo_plot <- vgs3 %>% filter(vgs3$companies == 'Nintendo') %>%
  ggplot(aes(Critic_Score, Global_Sales)) + geom_point(aes(color = Genre)) +
  ylim(0, 5) + geom_smooth() + labs(x = "Critic Score", y = "Total Sales") +
  ggtitle("Nintendo") + theme_classic() + theme(axis.text.x = element_blank())
```

```r
# Sony

sony_plot <- vgs3 %>% filter(vgs3$companies == 'Sony') %>% ggplot(aes(Critic_Score, NA_Sales)) +
  geom_point(aes(color = Genre)) + ylim(0, 5) + geom_smooth() +
  labs(x = "Critic Score", y = "Total Sales") + ggtitle("Sony") + theme_classic() +
  theme(axis.text.x = element_blank())

# Let's Compare those 3 Visually Together

ggarrange(nintendo_plot, microsoft_plot, sony_plot, ncol = 3)
```



As expected, with greater critic scores, a greater level of global sales. However, this is not a linear function of critic scores - it's exponential. This would imply that critic scores are highly important.

Modeling

Model 1: Simple Prediction

As labeled, this will be the simplest possible model to predict Global Sales from the entire population.

```r
set.seed(1)

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings) ^ 2, na.rm = T))
}
```

```r
val_ind <- createDataPartition(vgs3$Global_Sales, times = 1, p = 0.3, list = FALSE)
train_set <- vgs3[-val_ind,]
val_set <- vgs3[val_ind,]

# Simple Prediction Mean

mu <- mean(train_set$Global_Sales)

# What was the Mu?

mu
```

```
## [1] 0.7649898
```

```r
# Simple Prediction

rmse_model1 <- RMSE(train_set$Global_Sales, mu)
rmse_preds <- data.frame(Method = "Model 1: Simple Prediction Model", RMSE = rmse_model1)

# Check Results

rmse_preds %>% kable()
```

Method	RMSE
Model 1: Simple Prediction Model	2.015945

Model 2: Generalized Linear Model

This is a linear model to see if we can do any better.

```r
train_glm <- train(Global_Sales ~ Platform + User_Score + Critic_Score + Critic_Count + Genre + Year_of_

rmse_model2 <- getTrainPerf(train_glm)$TrainRMSE
rmse_preds <- add_row(rmse_preds, Method = "Model 2: Generalized Linear Model", RMSE = rmse_model2)

# Check Results



rmse_preds %>% kable()
```

Model 3: Knn

Now, we move onto the K-Nearest Neighbors algorithm: it is a non-parametric method used to compare across all parameters and considerations.

```r
train_knn <- train(Global_Sales ~ Platform + User_Score + Critic_Score + Critic_Count + Genre + Year_of_

rmse_model3 <- getTrainPerf(train_knn)$TrainRMSE
rmse_preds <- add_row(rmse_preds, Method = "Model 3: K-Nearest Neighbors Model", RMSE = rmse_model3)
```

```r
# Check Results

rmse_preds %>% kable()
```

Method	RMSE
Model 1: Simple Prediction Model	2.015945
Model 2: Generalized Linear Model	1.957060
Model 3: K-Nearest Neighbors Model	1.929684

Model 4: Support Vector Machines

Next is the Support Vector Machines algorithm: it is a discriminative classifier that maximizes the margin between two classes.

```r
train_svm <- train(Global_Sales ~ Platform + User_Score + Critic_Score + Critic_Count + Genre + Year_of_

rmse_model4 <- getTrainPerf(train_svm)$TrainRMSE
rmse_preds <- add_row(rmse_preds, Method = "Model 4: Support Vector Machines Model", RMSE = rmse_model4)

# Check Results

rmse_preds %>% kable()
```

Method	RMSE
Model 1: Simple Prediction Model	2.015945
Model 2: Generalized Linear Model	1.957060
Model 3: K-Nearest Neighbors Model	1.929684
Model 4: Support Vector Machines Model	1.834462

Model 5: Random Forest

Lastly for our training models, is the Random Forest algorithm: it is a method that operates from decision trees and outputs classification of the individual trees that also helps correct against overfitting training sets.

```r
train_rf <- train(Global_Sales ~ Platform + User_Score + Critic_Score + Critic_Count + Genre + Year_of_R

rmse_model5 <- getTrainPerf(train_rf)$TrainRMSE
rmse_preds <- add_row(rmse_preds, Method = "Model 5: Random Forest", RMSE = rmse_model5)

# Check Results

rmse_preds %>% kable()
```

Method	RMSE
Model 1: Simple Prediction Model	2.015945
Model 2: Generalized Linear Model	1.957060
Model 3: K-Nearest Neighbors Model	1.929684
Model 4: Support Vector Machines Model	1.834462
Model 5: Random Forest	1.675478

Variable Importance Per Model

Here, we should look at the variables with the highest importance to see if there are trends and valuable insights:

```r
varImp(train_glm)
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 61)
##
##                       Overall
## Critic_Count        100.000
## Critic_Score         65.298
## PlatformWii          32.272
## User_Score           32.244
## GenrePuzzle          17.208
## GenreMisc            16.718
## GenreStrategy        16.119
## PlatformPC           15.860
## Year_of_Release1997  13.691
## Year_of_Release1996  13.690
## Year_of_Release1998  11.313
## Year_of_Release1999  11.067
## PlatformXB           10.470
## `GenreRole-Playing`   9.783
## GenreAdventure        9.449
## PlatformDC            8.537
## Year_of_Release2001   8.373
## PlatformDS            8.126
## Year_of_Release2000   8.107
## PlatformPSV           7.704
```

```r
varImp(train_knn)
```

```
## loess r-squared variable importance
##
##                     Overall
## Critic_Count    100.00000
## Critic_Score     97.33944
## Platform         19.78415
## User_Score       12.78194
## Year_of_Release   4.36848
## Genre             0.03473
## Rating            0.00000
```

```r
varImp(train_svm)
```

```
## loess r-squared variable importance
##
##                     Overall
## Critic_Count    100.00000
## Critic_Score     97.33944
## Platform         19.78415
## User_Score       12.78194
## Year_of_Release   4.36848
## Genre             0.03473
```

```
## Rating            0.00000
```

```
varImp(train_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 61)
##
##                      Overall
## Critic_Score          100.00
## PlatformPC             91.43
## Year_of_Release2001    65.43
## Critic_Count           64.77
## PlatformPS             61.93
## PlatformPS2            60.72
## PlatformDS             59.32
## RatingE                59.15
## PlatformXB             58.41
## Year_of_Release2007    56.42
## GenreSimulation        55.83
## Year_of_Release1996    55.80
## PlatformGC             51.53
## GenreShooter           50.16
## GenreAdventure         49.57
## RatingM                48.36
## RatingE10+             46.31
## PlatformDC             45.01
## Year_of_Release2008    44.68
## Year_of_Release2002    43.04
```

Results

Our best model was Model 5: Random Forest. Therefore, we will select this algorithm to run our final model against the validation set. Now, let's run the final model to see if we were successful in training our model:

Final Model on Validation Set

```
train_final <- train(Global_Sales ~ Platform + User_Score + Critic_Score + Critic_Count + Genre,
    method = "rf", data = train_set, na.action = na.exclude, ntree = 40, metric="RMSE", trControl = tra

predicted <- predict(train_final, newdata = val_set)
rmse_model6 <- RMSE(val_set$Global_Sales, predicted)
rmse_preds <- add_row(rmse_preds, Method = "Model 6: Final on Validation", RMSE = rmse_model6)
```

We were successful in further reducing our RMSE.

But let's take a look at the variable importance for the validation set:

```
varImp(train_final)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 30)
##
##                  Overall
```

```
## Critic_Score     100.00
## PlatformPS        83.86
## Critic_Count      82.89
## PlatformXOne      58.55
## PlatformPC        58.01
## PlatformPS2       55.98
## GenreStrategy     53.24
## GenreRacing       45.98
## GenreSimulation   45.46
## GenreSports       45.21
## PlatformXB        43.34
## GenreShooter      39.83
## PlatformDS        38.93
## User_Score        38.84
## PlatformGC        34.84
## GenreAdventure    34.19
## PlatformDC        33.76
## PlatformWiiU      30.47
## PlatformWii       28.20
## GenrePlatform     26.77
```

```r
# Check Final Results

rmse_preds %>% kable()
```

Method	RMSE
Model 1: Simple Prediction Model	2.015945
Model 2: Generalized Linear Model	1.957060
Model 3: K-Nearest Neighbors Model	1.929684
Model 4: Support Vector Machines Model	1.834462
Model 5: Random Forest	1.675478
Model 6: Final on Validation	1.330436

It looks like we did better - success!

Conclusion

We were able to get the RMSE down to 1.330436! This shows that the model works pretty well as a machine learning algorithm to predict Global Sales based on VGChartz' Video Games Sales Dataset. The optimal model thus far is based on the Random Forest Model. However, I had to remove two variables in the final RF model due to returned errors - I could not debug in time but figured out that removing the variables did provide at least a good model. Another limitation was hardware - this was difficult to manage even with the parallel processing. My final report took roughly 25 mins to produce with parallel processing, but was successful nonetheless.

Future work would be to further tune the RF model and spend more time finding optimal numbers for ntree, mtry, and variables to use. Seems like Critic_Score and Critic_Count were the heavy favorites for variable importance - it would be interesting if we could develop this further to see if we can refine our prediction algorithm based on those factors.

The final RMSE is still much larger than desired but was successful in improving our algorithm by 34%! For reference, our worst model - the Simple Prediction Model - received an RMSE of 2.015945 It would be beneficial if we could garner more complete data as well through web scraping. One model I wished I had the time to adopt at the end was Matrix Factorization - I think this would be beneficial for future research.

Admin Note

```r
stopCluster(cl) # Stops Parallel Processing
```