Final Project

Idxian Gonzalez

12/29/2021

Introduction

This is a submission for the EDX Data Science Specialization Capstone project, which requests a Choose Your Own Project approach where we can utilize the tools learned throughout the classwork. For this project, I will be utilizing a previously curated Kaggle data set pertaining to video game sales and their *Critic_Score*. This data set can be downloaded freely at https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings and will also be included in the project submission.

Our goal for this project will be to see if we can predict a games *Critic_Score* by utilizing the variables provided in the data set. We can begin by loading the data in the form of a .csv file into our working directory:

```
vgsales <- read.csv("~/Video_Games_Sales_as_at_22_Dec_2016.csv")
## Initial Data Load, grab file from working directory
str(vgsales) ##Evaluate variable types
```

```
## '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" ...
```

```
head(vgsales) ## View first 10 rows of data set
```

```
##                        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
```

The variable *Year_of_Release* appears to be in a character format, and the *User_Score* variable is both in character format and in a different scale than *Critic_Count*.

```
vgsales$Year_of_Release <- as.numeric(vgsales$Year_of_Release ) ## Coerce Numeric
```

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

```
vgsales$User_Score <- as.numeric(vgsales$User_Score) * 10
```

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

```
## Coerce numeric and multiply by 10 to have User_Score in same format as Critic_Count
```

Besides this, there's not much else to be done to this data set. We can begin our analysis of the contents of these variables.

Methods and Analysis - Exploratory Data Analysis

Our target variable for this analysis is the *Critic_Score*. Lets begin our analysis by evaluating these variables.

```
summary(vgsales$Critic_Score) ## Display summary statistics for target variable
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.00   60.00   71.00   68.97   79.00   98.00    8582
```

With a minimum value of 13 and a maximum of 98, we can observe that the *Critic_Score* variable has wide range of values. We can also observe that 8,582 of the values in the data set are NA. Given that this is the variable we are trying to predict, we will only leave entries in our *vgsales data set* that have a value for this field. We can do so with the following code:

```r
vgsales <- vgsales[complete.cases(vgsales$Critic_Score),]
## Remove all rows with NA values in Critic_Score
sum(is.na(vgsales$Critic_Score))
```

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

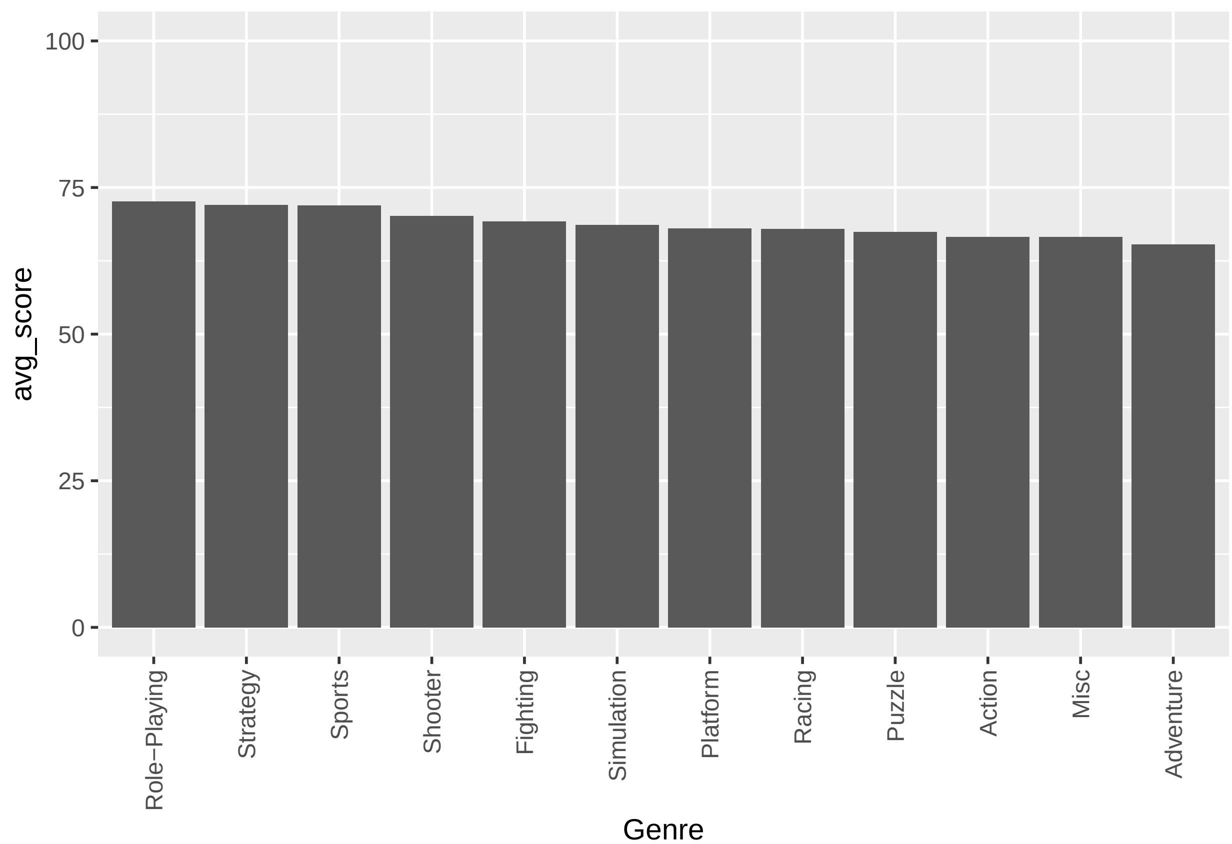
We are then left with 8,137 observations with a valid *Critic_Score*. We can first evaluate the average rating when grouped by *Genre*:

```r
vgsales_genre <- vgsales %>% group_by(Genre) %>% summarize(avg_score = mean(Critic_Score), avg_count = r

head(vgsales_genre[order(-vgsales_genre$avg_score),]) ##Evaluate sorted rows
```

```
## # A tibble: 6 x 3
##   Genre        avg_score avg_count
##   <chr>            <dbl>     <dbl>
## 1 Role-Playing      72.7      32.5
## 2 Strategy          72.1      28.3
## 3 Sports            72.0      21.0
## 4 Shooter           70.2      35.6
## 5 Fighting          69.2      27.9
## 6 Simulation        68.6      21.4
```

```r
vgsales_genre %>% ggplot(aes(x = reorder(Genre, -avg_score) ,y = avg_score)) + geom_col() + scale_y_cont
```

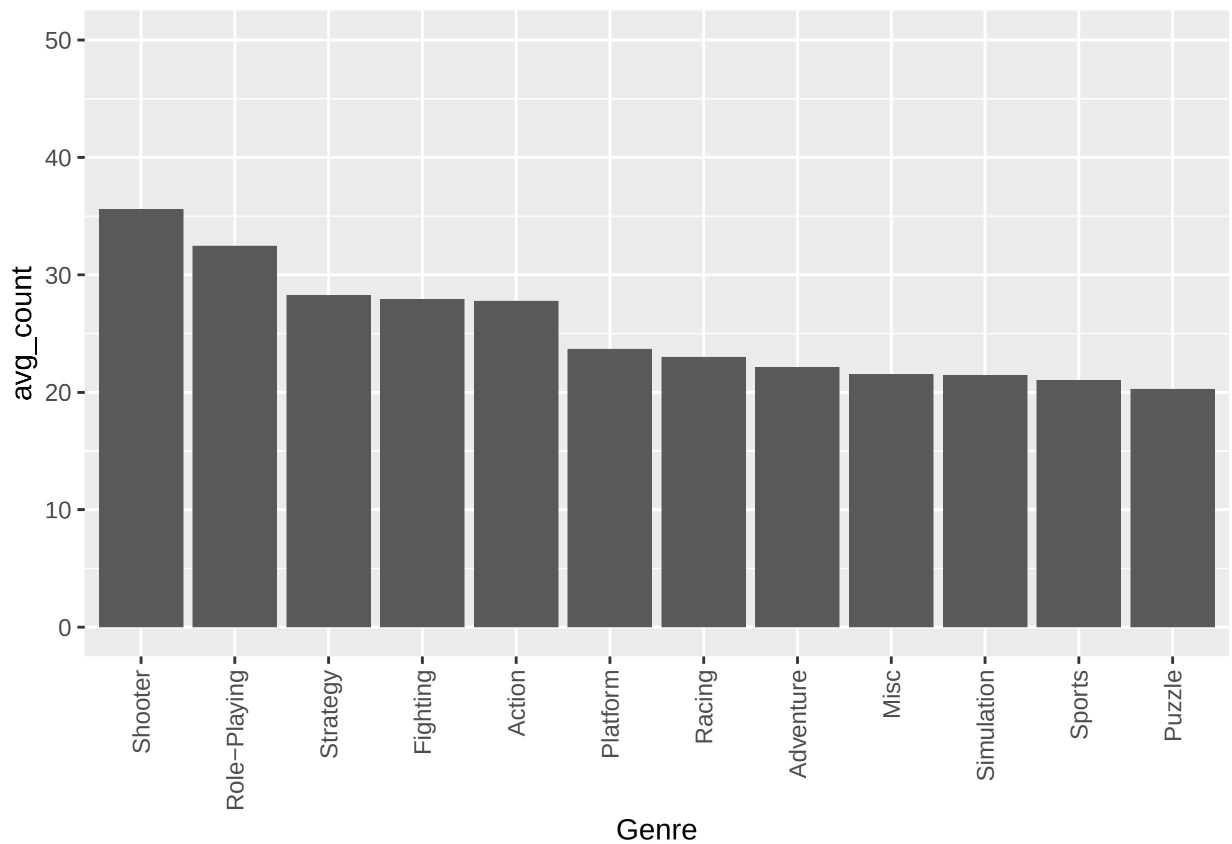


It's interesting to note that sports is third in regards to *Critic_Score*, however it is on average reviewed much less often than other categories. Lets examine this same table, but sorted by average *Critic_count*:

```
head(vgsales_genre[order(-vgsales_genre$avg_count),]) ## View top ordered rows
```

```
## # A tibble: 6 x 3
##   Genre        avg_score avg_count
##   <chr>            <dbl>     <dbl>
## 1 Shooter           70.2      35.6
## 2 Role-Playing      72.7      32.5
## 3 Strategy          72.1      28.3
## 4 Fighting          69.2      27.9
## 5 Action            66.6      27.8
## 6 Platform          68.1      23.7
```

```
vgsales_genre %>% ggplot(aes(x = reorder(Genre, -avg_count) ,y = avg_count)) + geom_col() + scale_y_con
```



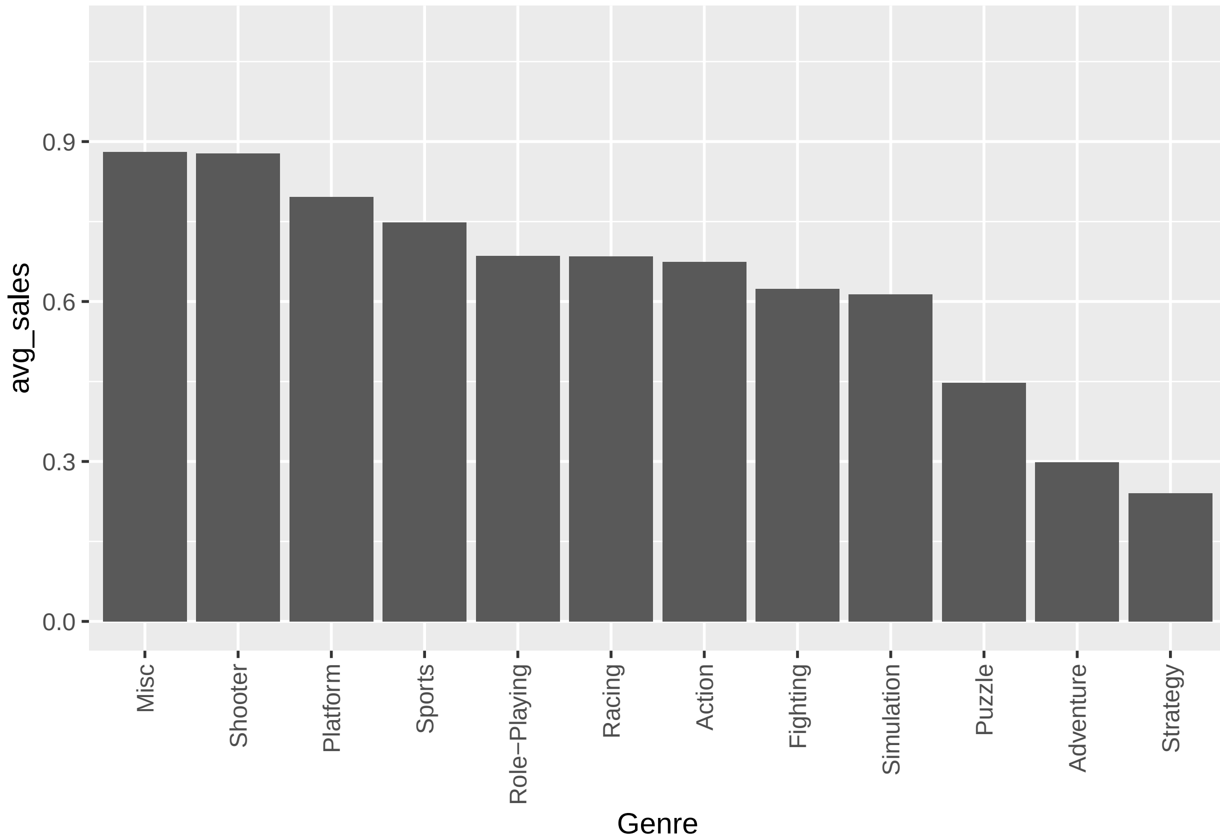
The most reviewed category are Shooter based games, and by a fairly large margin. Is there any particular reason these games are reviewed more often? Lets compare this with worldwide sales by genre:

```
vgsales_worldsales <- vgsales %>% group_by(Genre) %>% summarize(avg_score = mean(Critic_Score), avg_cou
head(vgsales_worldsales[order(-vgsales_worldsales$avg_sales),])
```

```
## # A tibble: 6 x 4
##   Genre        avg_score avg_count avg_sales
```

```
##   <chr>             <dbl>     <dbl>     <dbl>
## 1 Misc               66.6      21.5     0.881
## 2 Shooter            70.2      35.6     0.878
## 3 Platform           68.1      23.7     0.796
## 4 Sports             72.0      21.0     0.749
## 5 Role-Playing       72.7      32.5     0.686
## 6 Racing             68.0      23.0     0.685
```

```r
vgsales_worldsales %>% ggplot(aes(x = reorder(Genre, -avg_sales) ,y = avg_sales)) + geom_col() + scale_
```



Shooters do seems to have more average global sales than sports, however not to a degree that would correspond to a +2 on average game score when compared against the sports genre. We can also note that the category with most average global sales is Misc, although this might just be an effect of pooling various niche Genres into a catch-all Miscellaneous category and as such might have more than one actual genre quantified within it.

Having observed the distribution by Genre over various variables, we will evaluate *Critic_Score*, *Critic_Count* and *Global_Sales* in relation to the Developer variable, which lets us know which company developed the game:

```r
vgsales_developer <- vgsales %>% group_by(Developer) %>% summarize(avg_score = mean(Critic_Score), avg_

head(vgsales_developer[order(-vgsales_developer$avg_score),]) ##Evaluate sorted rows
```

```
## # A tibble: 6 x 4
##   Developer                          avg_score avg_count avg_sales
```

```
##   <chr>                                      <dbl>     <dbl>     <dbl>
## 1 Irrational Games, 2K Marin                  96        66         1.62
## 2 Digital Extremes, 2K Marin                  94        51         1.42
## 3 Kojima Productions, Moby Dick Studio        94        48         2.08
## 4 Bungie Software                             93.7      65.3       5.20
## 5 DMA Design, Rockstar North                  93        20         0.01
## 6 Rockstar North                              92.9      54         8.53
```

By ordering the data by *avg_score*, we can observe the top 6 Developers within this data set. What's interesting here is that while sorting through average score, both *avg_score* and *avg_count* variables are relatively close together, *avg_sales* range from 0.01 to 8.53. This would suggest that the average sales a Developer has generated does not necessarily correlate with *average_score*. We can confirm this by running a correlation analysis between these two variables:

```
cor(vgsales_developer$avg_score,vgsales_developer$avg_sales) ## Calculate Pearson Correlation
```

```
## [1] 0.2824578
```

With a simple correlation coefficient of 0.28, there's not much of a linear relationship between these two variables, which would generally mean any linear models fitted to explain this relationship would find little benefit in considering global sales as a predictor for *Critic_Score*. It would also be beneficial to explore whether the video game platform has an overall effect over critic scores:

```
vgsales_platform <- vgsales %>% group_by(Platform) %>% summarize(avg_score = mean(Critic_Score), avg_co

head(vgsales_platform[order(-vgsales_platform$avg_score),]) ##Evaluate sorted rows
```

```
## # A tibble: 6 x 4
##   Platform avg_score avg_count avg_sales
##   <chr>        <dbl>     <dbl>     <dbl>
## 1 DC            87.4      17.6     0.325
## 2 PC            75.9      27.9     0.275
## 3 XOne          73.3      24.4     0.772
## 4 PS4           72.1      39.0     0.970
## 5 PS            71.5      10.4     1.17
## 6 PSV           70.8      27.1     0.260
```

We can see that the Sega Dreamcast was by far the highest scored, however it's global sales do not go hand-in-hand with this rating. This further cements the idea that score does not necessarily correlate with sales or vice versa.

Next, we will evaluate if average ratings have changed in relation to the release year of each game. We will calculate the average rating by year with the following piece of code:

```
vgsales_year <- vgsales %>% group_by(Year_of_Release) %>% summarize(avg_score = mean(Critic_Score), avg_

head(vgsales_year[order(-vgsales_year$avg_score),]) ##Evaluate sorted rows
```

```
## # A tibble: 6 x 4
##   Year_of_Release avg_score avg_count avg_sales
##             <dbl>     <dbl>     <dbl>     <dbl>
## 1            1996      89.9      11.4      2.54
```

```
## 2             1997      85.3      12.4      2.57
## 3             1992      85        44        0.03
## 4             1998      81.8      12.1      1.76
## 5             1999      75.8      12.8      1.39
## 6             2016      73.2      30.3      0.400
```

We can immediately appreciate that the highest rated years were all in the 90's, with the exception of 2016 who is a fair bit behind them in average rating. Average count of reviews is also low, relative to recent years. This might be because gaming in general was not as mainstream in the 90's as it has been in recent years. If we assume that the average score is a function of overall video game popularity, we can validate this through another correlation analysis:

```
vgsales_year <- vgsales_year[complete.cases(vgsales_year$Year_of_Release),]
cor(vgsales_year$avg_count,as.numeric(vgsales_year$Year_of_Release), use = "complete.obs")
```

```
## [1] 0.1208596
```

```
## Calculate Pearson Correlation
```

A positive pearson coefficient of 0.120 indicates a tenuous yet positive linear relationship between Year of Release and average critic score, which would suggest that video game average scores have risen along with overall video game popularity in recent years.

Methods and Analysis II - Linear Modeling

Having evaluated most of the variable of interest, we can start by fitting a basic linear equation for Critic Score in the *vgsales data set*:

```
lm <- lm(Critic_Score ~ Global_Sales, data = vgsales) #Create the linear regression
summary(lm) #Review the results
```

```
##
## Call:
## lm(formula = Critic_Score ~ Global_Sales, data = vgsales)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -147.099   -8.066    2.049   10.049   29.690
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  67.67001    0.16021  422.38   <2e-16 ***
## Global_Sales  1.88331    0.08246   22.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.51 on 8135 degrees of freedom
## Multiple R-squared:  0.06026,    Adjusted R-squared:  0.06014
## F-statistic: 521.6 on 1 and 8135 DF,  p-value: < 2.2e-16
```

The P-Value for this model is highly significant, suggesting a very strong linear relationship between the independent variable *Global_Sales* and *Critic Score*. However, when we evaluate the adjusted r-squared, we see that approximately only 6% of the variability in critic score is explained by the *global sales* variable. While *global sales* does tend to move forwards and backwards along with average score, by itself it is a poor predictor of our dependent variable. Let's evaluate a different variable:

```r
lm <- lm(Critic_Score ~ Critic_Count, data = vgsales) #Create the linear regression
summary(lm) #Review the results
```

```
##
## Call:
## lm(formula = Critic_Score ~ Critic_Count, data = vgsales)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.917  -7.293   1.333   8.832  32.019
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  60.730840   0.239329  253.75   <2e-16 ***
## Critic_Count  0.312465   0.007368   42.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.61 on 8135 degrees of freedom
## Multiple R-squared:  0.1811, Adjusted R-squared:  0.181
## F-statistic:  1798 on 1 and 8135 DF,  p-value: < 2.2e-16
```

We see that *Critic_Count* is also highly significant, and in this case we can capture around 18% of the variability in *Critic_Score*. This is better than 6%, but still well below what we would like. What would happen if we tried them together?:

```r
lm <- lm(Critic_Score ~ Critic_Count + Global_Sales, data = vgsales) #Create the linear regression
summary(lm) #Review the results
```

```
##
## Call:
## lm(formula = Critic_Score ~ Critic_Count + Global_Sales, data = vgsales)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -80.401  -7.163   1.426   8.688  31.311
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  60.806428   0.237237  256.31   <2e-16 ***
## Critic_Count  0.283907   0.007663   37.05   <2e-16 ***
## Global_Sales  0.982856   0.080058   12.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.5 on 8134 degrees of freedom
## Multiple R-squared:  0.196,  Adjusted R-squared:  0.1958
## F-statistic: 991.2 on 2 and 8134 DF,  p-value: < 2.2e-16
```

Taking both variables into the model helps us explain nearly 20% of the variability in *Critic_Score*. This is helpful, but perhaps not the best way to model this data. Lets calculate how accurate this model is by splitting our data into training/test pairs and using the model to predict the test values:

```
set.seed(123)
training.samples <- vgsales$Critic_Score %>% createDataPartition(p = 0.9, list = FALSE)


train.data  <- vgsales[training.samples, ]
test.data <- vgsales[-training.samples, ]

lm <- lm(Critic_Score ~ Critic_Count + Global_Sales, data = train.data) #Create the linear regression

probabilities <- lm %>% predict(test.data, type = "response")
test.data$Predictions <- probabilities
rmse(test.data$Critic_Score, test.data$Predictions)
```

```
## [1] 12.27108
```

An RMSE of 12.77 would mean that predicted values will be on average nearly 13 points way from the actual value. Maybe utilizing *User_Scores* we can more correctly predict *Critic_Scores*:

```
set.seed(321)
training.samples <- vgsales$Critic_Score %>% createDataPartition(p = 0.9, list = FALSE)

vgsales <- vgsales[complete.cases(vgsales$User_Score),] ## leave only rows with valid user scores
vgsales <- vgsales[complete.cases(vgsales$Year_of_Release),] ## leave only rows with valid user scores

train.data  <- vgsales[training.samples, ]
test.data <- vgsales[-training.samples, ]

lm <- lm(Critic_Score ~ User_Score + Critic_Count + Global_Sales + Genre + Year_of_Release + Platform, d

probabilities <- lm %>% predict(test.data, type = "response") ## Predict Values with model
test.data$Predictions <- probabilities ## load predictions into test.dataframe
rmse(test.data$Critic_Score, test.data$Predictions) ## Verify RMSE for model
```

```
## [1] 9.50552
```

Adding the variables *Year_of_Release*, *Platform* and *User Score*, we are able to get an R-squared of almost 0.52, as well as an RMSE of 9.42. This much deviation from the actual values is to be expected, as we can currently only account for roughly half the variability in the *Critic_Score* variables using linear regression. We can instead attempt to model the probability that a game will be reviewed fairly by splitting the variable into a binary favorable/unfavorable review.

We can assume that games with an 80 or higher in *Critic_Score* are considered favorable, and all other values below it as Unfavorable. Lets create the necessary variables for our analysis:

```
vgsales$favorable <- ifelse(vgsales$Critic_Score >= 80,'Favorable','Not Favorable')
## Create binary favorable/unfavorable variable

vgsales <- vgsales %>% mutate(TopNa = ifelse(NA_Sales > JP_Sales & NA_Sales > EU_Sales,'Top NA', ifelse

str(vgsales)
```

```
## 'data.frame':    6894 obs. of  18 variables:
##  $ Name           : chr  "Wii Sports" "Mario Kart Wii" "Wii Sports Resort" "New Super Mario Bros." .
##  $ Platform       : chr  "Wii" "Wii" "Wii" "DS" ...
##  $ Year_of_Release: num  2006 2008 2009 2006 2006 ...
##  $ Genre          : chr  "Sports" "Racing" "Sports" "Platform" ...
##  $ Publisher      : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ NA_Sales       : num  41.4 15.7 15.6 11.3 14 ...
##  $ EU_Sales       : num  28.96 12.76 10.93 9.14 9.18 ...
##  $ JP_Sales       : num  3.77 3.79 3.28 6.5 2.93 4.7 4.13 3.6 0.24 2.53 ...
##  $ Other_Sales    : num  8.45 3.29 2.95 2.88 2.84 2.24 1.9 2.15 1.69 1.77 ...
##  $ Global_Sales   : num  82.5 35.5 32.8 29.8 28.9 ...
##  $ Critic_Score   : int  76 82 80 89 58 87 91 80 61 80 ...
##  $ Critic_Count   : int  51 73 73 65 41 80 64 63 45 33 ...
##  $ User_Score     : num  80 83 80 85 66 84 86 77 63 74 ...
##  $ User_Count     : int  322 709 192 431 129 594 464 146 106 52 ...
##  $ Developer      : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ Rating         : chr  "E" "E" "E" "E" ...
##  $ favorable      : chr  "Not Favorable" "Favorable" "Favorable" "Favorable" ...
##  $ TopNa          : chr  "Top NA" "Top NA" "Top NA" "Top NA" ...
```

Since our dependent variable is no longer a continuous variable but a binary one, we will be training a K-NN algorithm in order to predict whether a critic review will be favorable or unfavorable:

```
set.seed(321)

vgsales <- vgsales[complete.cases(vgsales), ] ## Remove all NA Values

knsales <- vgsales %>% select(Critic_Score, User_Score,User_Count , Critic_Count , Global_Sales, Genre,

training.samples <- knsales$Critic_Score %>% createDataPartition(p = 0.9, list = FALSE)
## Create partition

train.data  <- knsales[training.samples, ] ## Subset train data
test.data <- knsales[-training.samples, ] ## Subset test Data

train.data.labels <- train.data %>% select(favorable) ## Select train labels
test.data.labels <- test.data %>% select(favorable) ## Select test labels

train.data <- train.data %>% select(User_Score, Critic_Count , Global_Sales, Year_of_Release ,User_Coun
## RM favorable Column

test.data <- test.data %>% select(User_Score, Critic_Count , Global_Sales, Year_of_Release  ,User_Count
## RM favorable Column


test_pred <- knn(train = train.data, test = test.data,cl = train.data.labels$favorable, k= 79)
## Train Model

confusionMatrix(table(test_pred ,test.data.labels$favorable)) ## Evaluate model accuracy
```

```
## Confusion Matrix and Statistics
##
##
```

```
## test_pred       Favorable Not Favorable
##   Favorable            96            51
##   Not Favorable        99           442
##
##                Accuracy : 0.782
##                  95% CI : (0.7492, 0.8123)
##     No Information Rate : 0.7166
##     P-Value [Acc > NIR] : 5.732e-05
##
##                   Kappa : 0.4201
##
##  Mcnemar's Test P-Value : 0.0001243
##
##             Sensitivity : 0.4923
##             Specificity : 0.8966
##          Pos Pred Value : 0.6531
##          Neg Pred Value : 0.8170
##              Prevalence : 0.2834
##          Detection Rate : 0.1395
##    Detection Prevalence : 0.2137
##       Balanced Accuracy : 0.6944
##
##        'Positive' Class : Favorable
##
```

This approach seems to work far better than plotting the relationship linearly, as our K-NN model reaches far higher accuracy in determining whether a game will have a favorable *Critic_Score* by utilizing the variables *User_Score*, *User_Count*, *Critic_Count*, *Global_Sales* and *Year_of_Release*.

By using cross-validation to verify the output of our model, we can compare the output of the model predictions with the test.data results, achieving 78.2% accuracy in predicting whether a video game will be rated favorably or unfavorably by a critic.

Conclusion

Data was extracted, cleaned and wrangled resulting in 8,137 values being utilized as our final data. The largest impact to sample size was removing entries where no *Critic_Score* was logged, which resulted in about half of our data being discarded.

The first model considered was a simple linear model, attempting to predict *Critic_Score* through various independent variables. These efforts resulted in a linear model with an R-squared of 0.52, and an RMSE of 9.42. Were we to attempt predicting *Critic_Score* through this model, answers would be on average 9.42 rating points away from the true value.

The second model considered was a K-NN neighbors approach. The target variable *Critic_Score* was converted into a binary variable, where *Critic_Score* above 80 were considered favorable reviews, and scores below were considered unfavorable.

The data was again split into training/test pairs and the model cross validated with the test data. The K-NN model was able to classify the test.data with 78.2% accuracy by using the variables *User_Score*, *User_Count*, *Year_of_Release* and *Global_Sales* as predictors for the target variable.

Future work and Limitations

Our first attempt at modeling the variable *Critic_score* was through a simple linear regression, this model, however was only able to quantify approximately 52% of the variability in *Critic_score*. It is not surprising then that the average deviation from the actual value, as predicted by the model was 9.42 rating points away from the actual value. The prediction made through this model is too far away from the actual value.

Our second model attempt to predict *Critic_score* not as a continuous variable but a categorical one. We divided the data into favorable and unfavorable critic scores, defined as a game with a critic score over 80 being considered favorable. We utilized a K-NN algorithm which was able to correctly predict whether a game would have a favorable or unfavorable rating with nearly 80% accuracy. Additional models could be explored, such as random forest applications or logistical regression which might serve as better predictors of this variable.