Video Games Sales

Victoria Famakinwa

3/7/2022

INTRODUCTION

This is the second HarvardX PH125.9x Data Science: Capstone course Project I will be working on. The purpose of this project is to create an algorithm using the Video Games Sales dataset which was retrieved from Kaggle. This is to demonstrate the depth of my understanding by working on a "Choose Your Own" dataset. In this project The goal of this report is to try to predict the global sales of a title using the other variables.

For this project the Video Games Sales dataset was retrieved from Kaggle. I'm especially interested in the critic score and the user score to see if they have an impact on game sales. First, we load the packages we'll need for the rest of the analysis:

IMPORTING DATA

The dataset is in a CVS file, and it should be read into Rstudio as a variable as shown below:

```
## Rows: 16719 Columns: 16

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): Name, Platform, Year_of_Release, Genre, Publisher, User_Score, Deve...
## dbl (8): NA_Sales, EU_Sales, JP_Sales, Other_Sales, Global_Sales, Critic_Sco...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

DATA INSPECTION

Here we inspected the data by checking if there are any NA values

```
##            Name        Platform Year_of_Release           Genre       Publisher
##               2               0               0               2               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            6704            9129            6623
##          Rating
##            6769
```

There are far too many missing values, as we can see. This is due to the fact that the data set utilized is a hybrid of two separate sets, with many of the original observations lacking equivalent data from the second data set. We are not interested in games for which we do not have this data because our study is largely focused on the ratings received by each title. As a result, records with missing values are removed:

```
##            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
```

Checked and seen that there are no more NA values. Next we look at the structure of the data set:

```
## tibble [6,947 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Name           : chr [1:6947] "Wii Sports" "Mario Kart Wii" "Wii Sports Resort" "New Super Mario
##  $ Platform       : chr [1:6947] "Wii" "Wii" "Wii" "DS" ...
##  $ Year_of_Release: chr [1:6947] "2006" "2008" "2009" "2006" ...
##  $ Genre          : chr [1:6947] "Sports" "Racing" "Sports" "Platform" ...
##  $ Publisher      : chr [1:6947] "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ NA_Sales       : num [1:6947] 41.4 15.7 15.6 11.3 14 ...
##  $ EU_Sales       : num [1:6947] 28.96 12.76 10.93 9.14 9.18 ...
##  $ JP_Sales       : num [1:6947] 3.77 3.79 3.28 6.5 2.93 4.7 4.13 3.6 0.24 2.53 ...
##  $ Other_Sales    : num [1:6947] 8.45 3.29 2.95 2.88 2.84 2.24 1.9 2.15 1.69 1.77 ...
##  $ Global_Sales   : num [1:6947] 82.5 35.5 32.8 29.8 28.9 ...
##  $ Critic_Score   : num [1:6947] 76 82 80 89 58 87 91 80 61 80 ...
##  $ Critic_Count   : num [1:6947] 51 73 73 65 41 80 64 63 45 33 ...
##  $ User_Score     : chr [1:6947] "8" "8.3" "8" "8.5" ...
##  $ User_Count     : num [1:6947] 322 709 192 431 129 594 464 146 106 52 ...
##  $ Developer      : chr [1:6947] "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ Rating         : chr [1:6947] "E" "E" "E" "E" ...
```

The year of release appears to be non-numerical, despite the fact that it should be. Take a look at the following values for this variable:

```
##  [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"
```

When we checked for NA values, we found a string "N/A" that was overlooked. These, like all other NA values, must be removed:

"NA" values are no longer available. We can now convert the variable to an integer and save it as follows:

Looking to see if other string columns have the same problem. For publisher, developer etc and eliminating it from the column:

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

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

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

Everything appears to be in order. Let's take a look at the sales factors to see if there are any odd outliers:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0600  0.1500  0.3945  0.3900 41.3600
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0200  0.0600  0.2361  0.2100 28.9600
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.06416 0.01000 6.50000
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00000  0.01000  0.02000  0.08268  0.07000 10.57000
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0100  0.1100  0.2900  0.7776  0.7500 82.5300
```

Nothing seems out of the ordinary. Now for the ratings:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.00   62.00   72.00   70.27   80.00   98.00
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00   14.00   25.00   28.93   39.00  113.00
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     4.0    11.0    27.0   174.7    89.0 10665.0
```

```
##    Length     Class      Mode
##      6825 character character
```

The user score is not numeric, as we can see. It must be converted and scaled to match the criticScore variable

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   6.500   7.500   7.186   8.200   9.600
```

We need to analyze the Rating variable and ignore the ratings with insignificant number of ratings

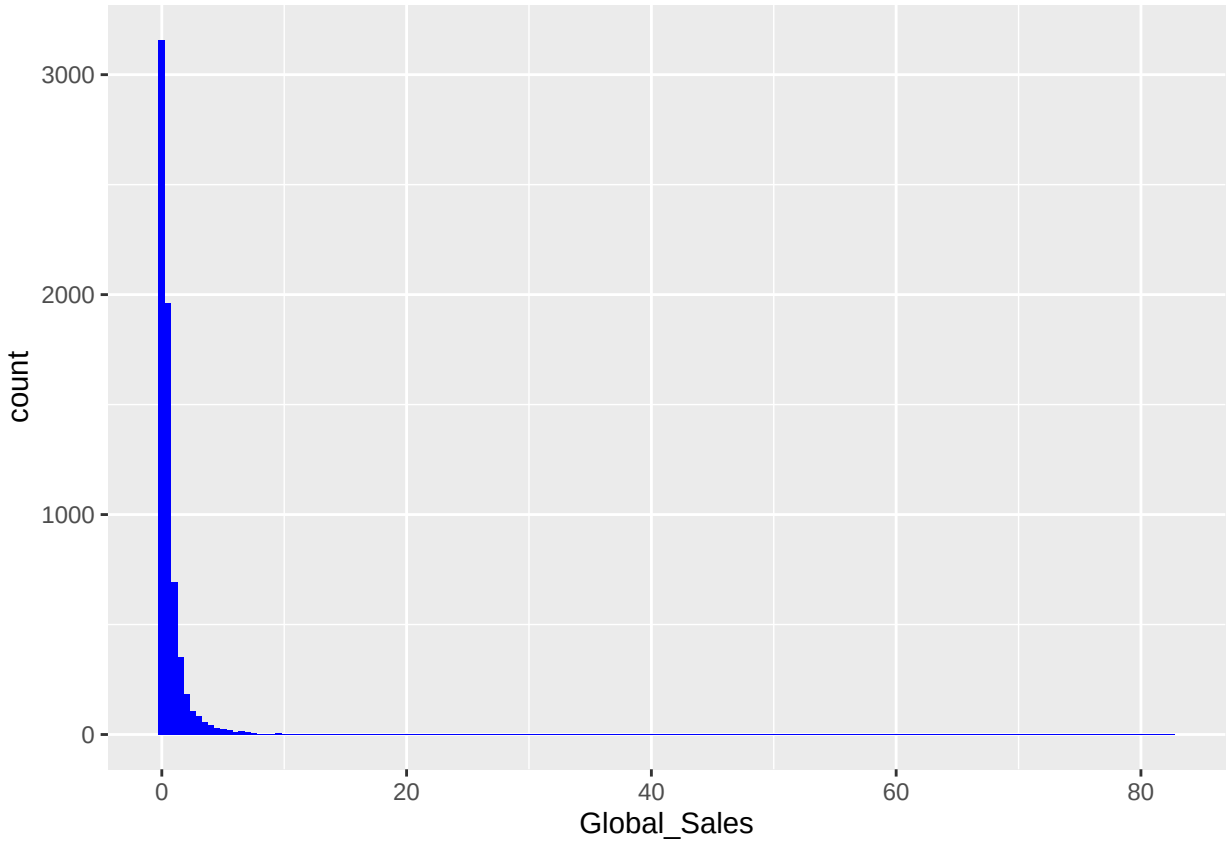
```
## # A tibble: 7 x 2
##   Rating     n
##   <chr>  <int>
## 1 AO         1
## 2 E       2082
## 3 E10+     930
## 4 K-A        1
## 5 M       1433
## 6 RP         1
## 7 T       2377
```

```
## # A tibble: 4 x 2
##   Rating     n
##   <chr>  <int>
## 1 E       2084
## 2 E10+     930
## 3 M       1434
## 4 T       2377
```

DATA VISUALIZATION

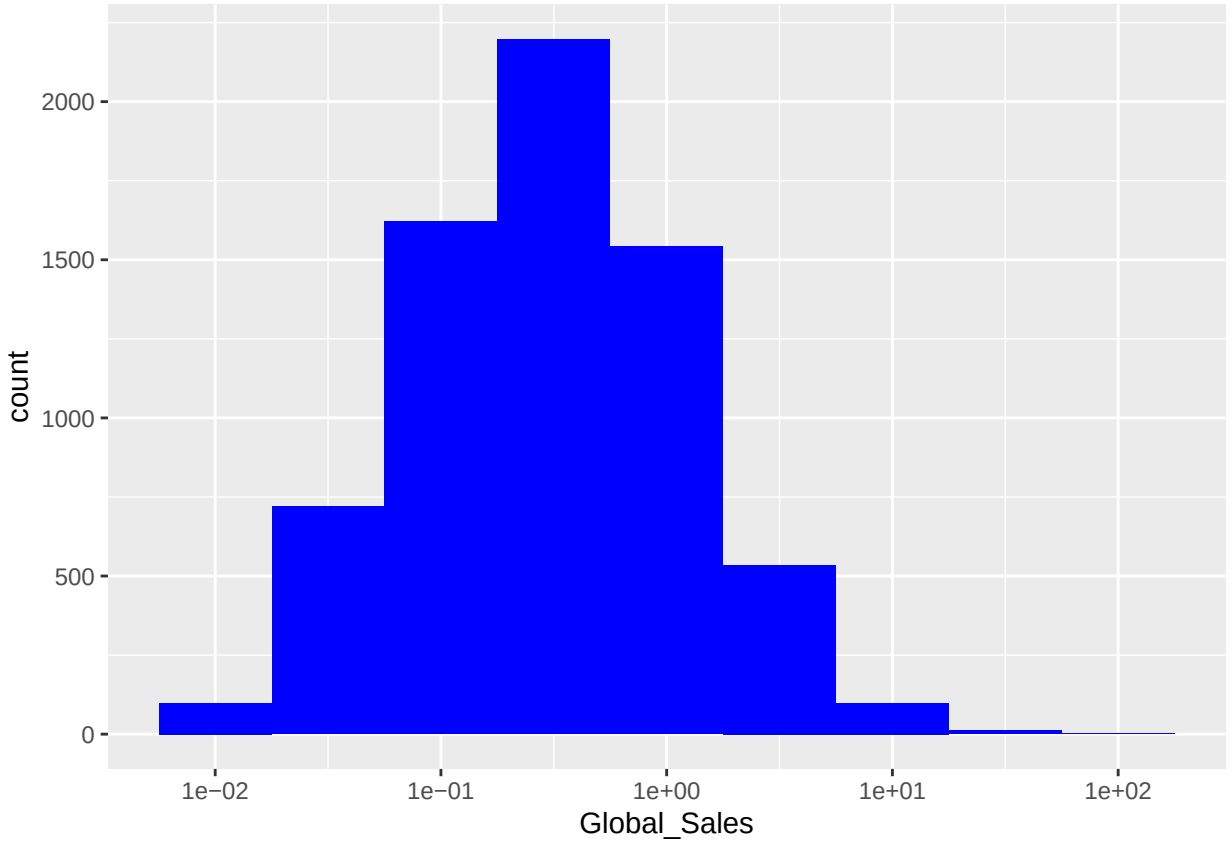
In this part, we create graphs to assist us understand the data. We begin by examining global sales, which is our dependent variable:

```
ggplot(vg_sales) + geom_histogram(aes(Global_Sales), fill = "blue", binwidth = 0.5)
```



The variable is considerably skewed. As a result, it makes sense to log the scale axis:

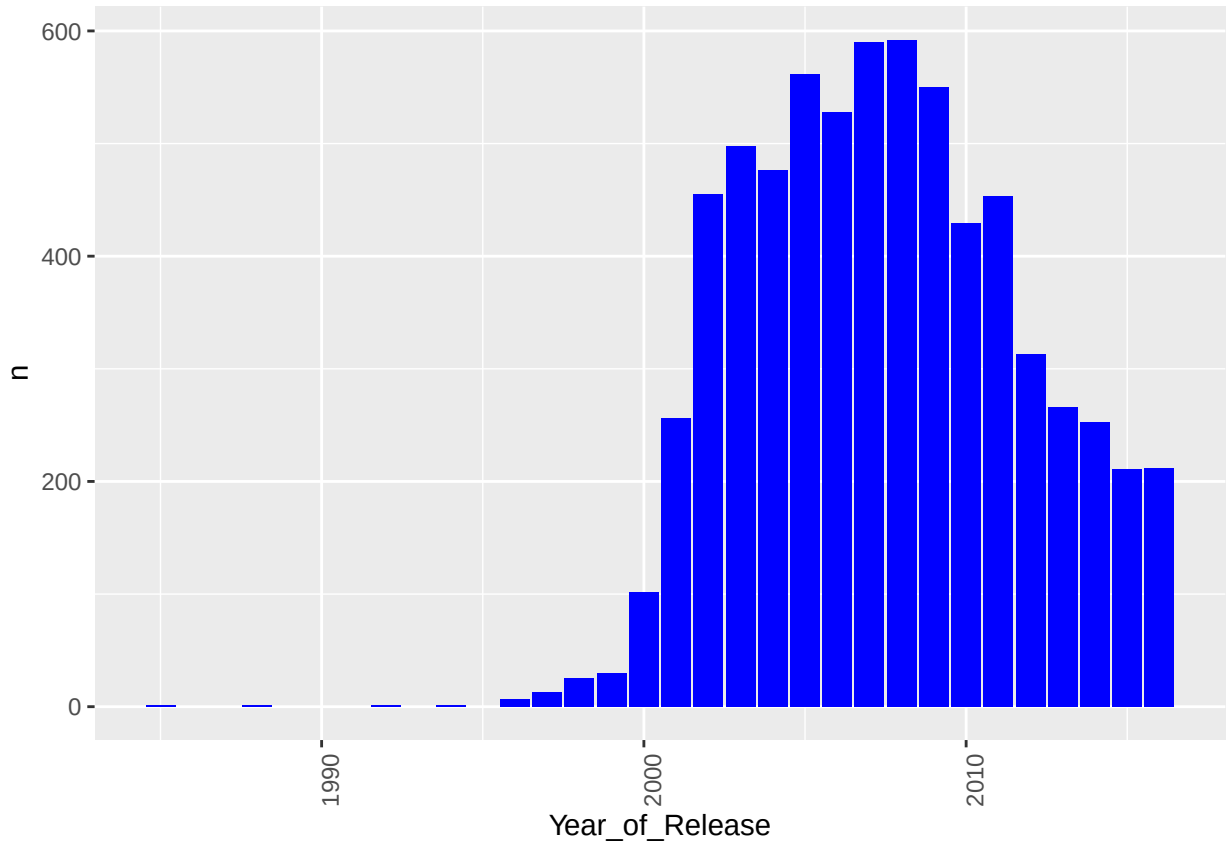
```
ggplot(vg_sales) + geom_histogram(aes(Global_Sales), fill = "blue", binwidth = 0.5) +
scale_x_log10()
```



Based on this, it would be best to take the log of the variable later when we fit the models.

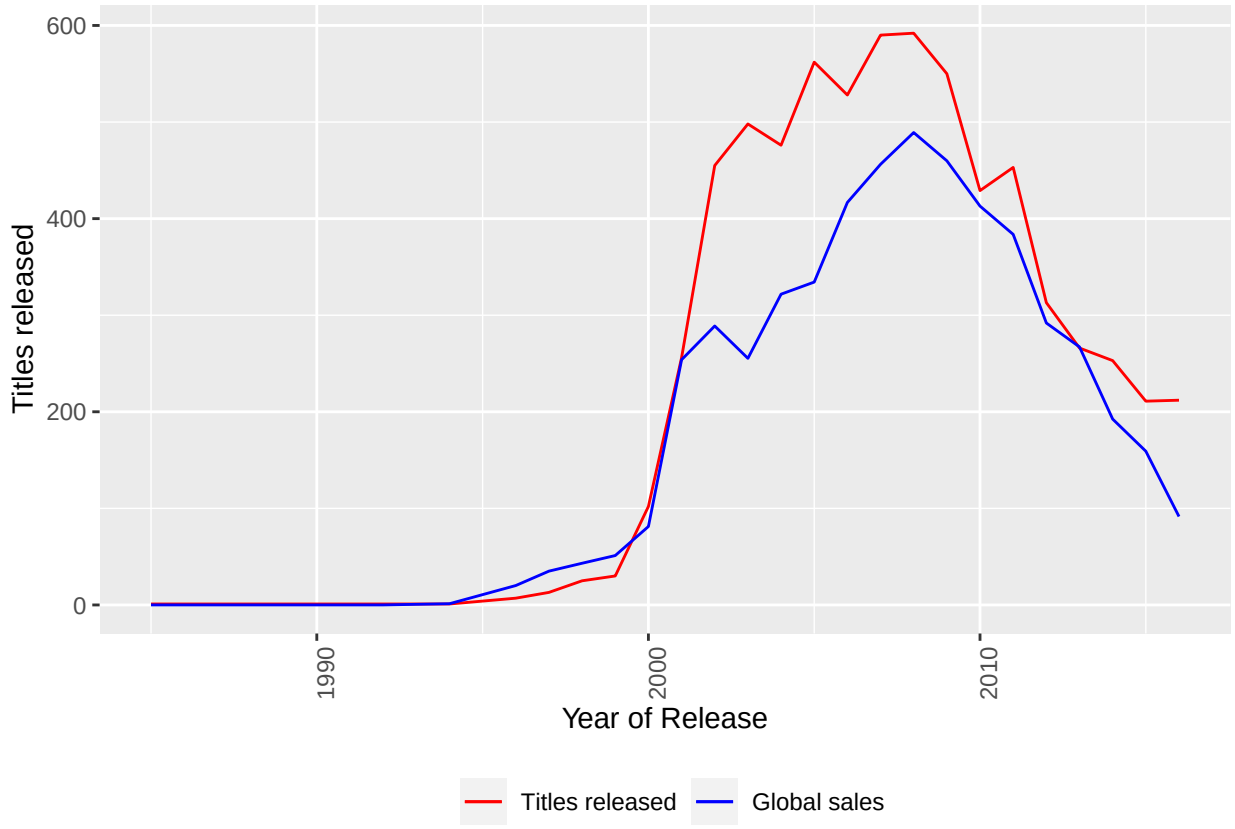
#Exploring the Number of Titles released in each year (Year_of_Release variable)

```
vg_sales %>% group_by(Year_of_Release) %>%
count() %>% ggplot() +
geom_bar(aes(Year_of_Release, n), stat = "identity",
fill = "blue") + theme(axis.text.x = element_text(angle = 90))
```



There is a distinct apex, as we can see. Let's take a look at the year's sales and compare them to the year's release numbers:

```
#comparing sales in each year and the release numbers each year#
color <- c("Titles released" = "red", "Global sales" = "blue")
vg_sales %>% group_by(Year_of_Release) %>%
summarise(sales = sum(Global_Sales), count = n()) %>%
ggplot() + geom_line(aes(Year_of_Release, count, group = 1, color = "Titles released")) +
geom_line(aes(Year_of_Release, sales, group = 1, color = "Global sales")) +
xlab("Year of Release") + ylab("Titles released") +
theme(axis.text.x = element_text(angle = 90), legend.position = "bottom") +
scale_color_manual(values = color) + labs(color = "")
```



We can observe that the more the number of titles, the higher the revenue. The highest sales coincide to the peak in the year histograms, as can be seen. As a result, the majority of the data in the database dates from 2007 to 2009. Looking at the median and mean reveals this:

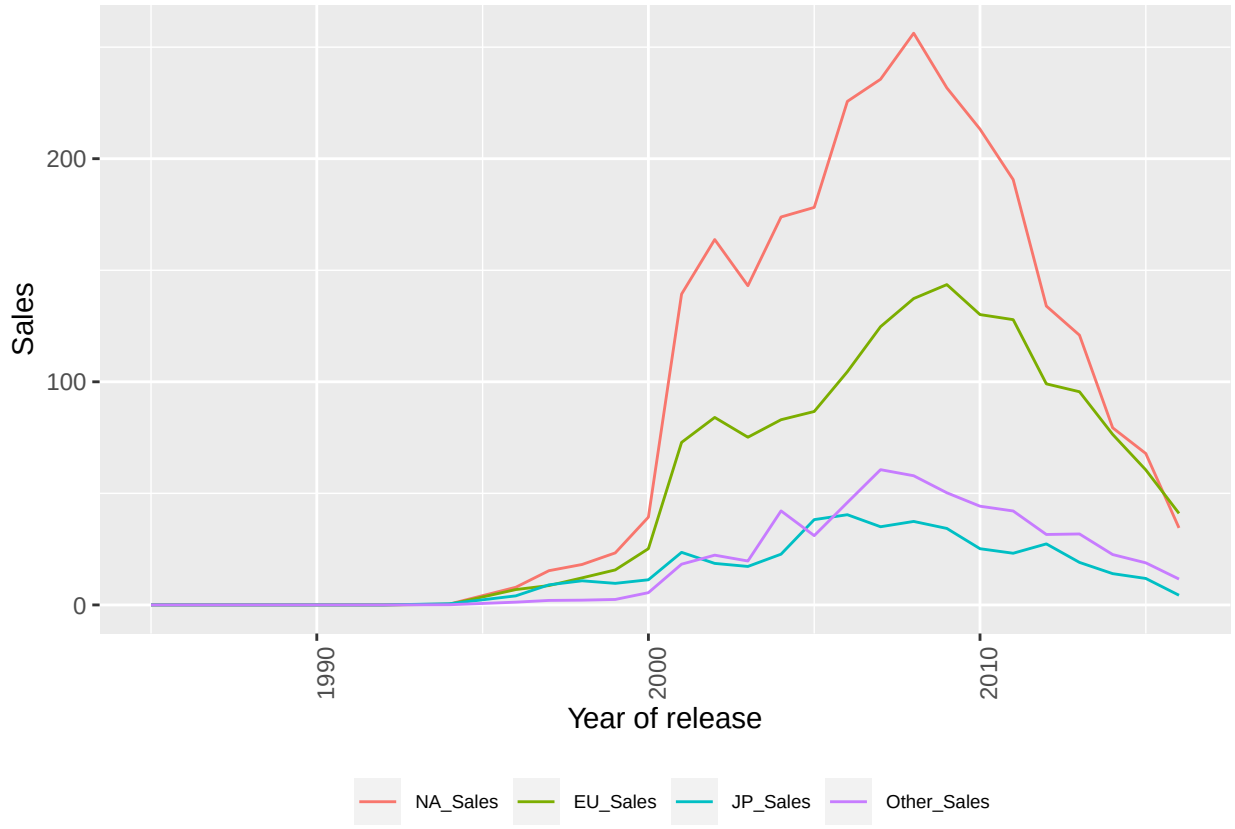
```
summary(vg_sales$Year_of_Release)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1985    2004    2007    2007    2011    2016
```

Now let's look at how sales have changed in different geographical regions:

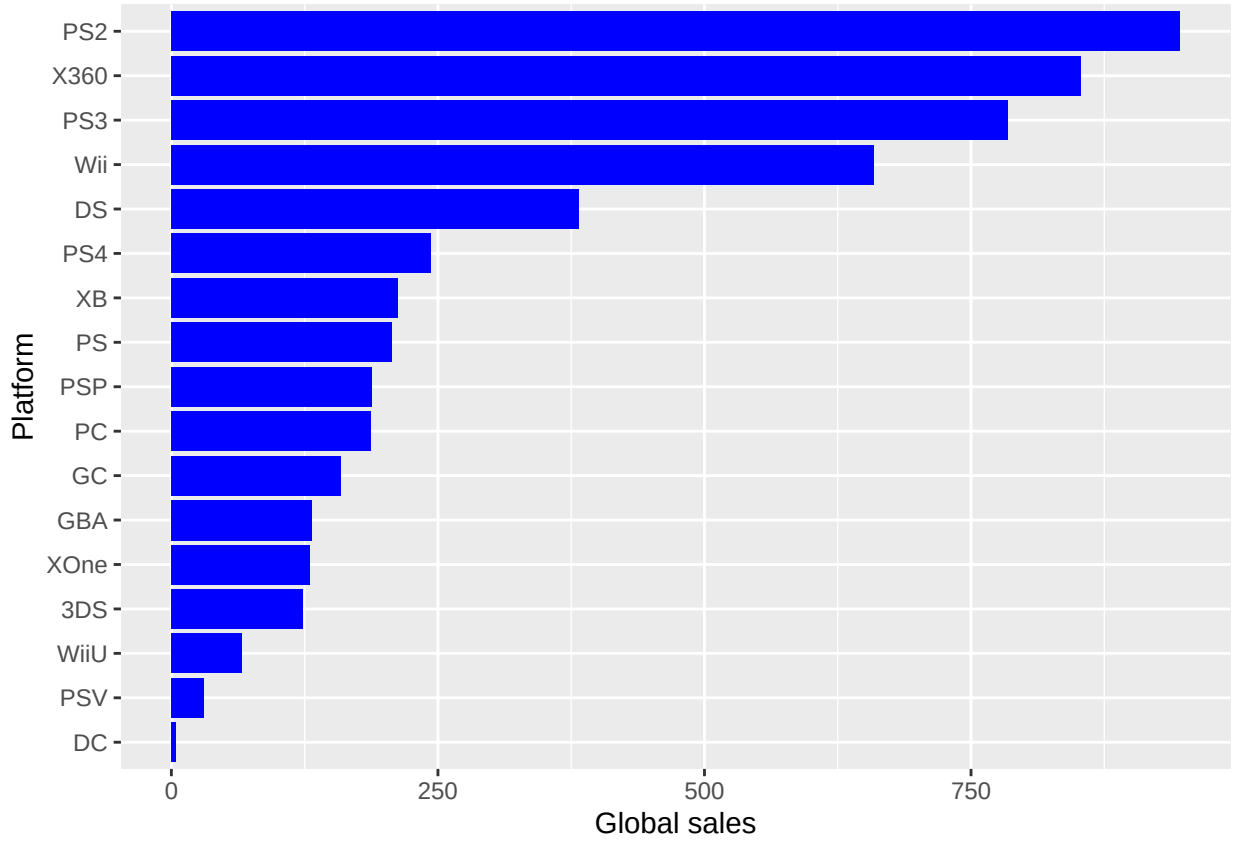
```
vg_sales %>% gather(area, sales, NA_Sales:Other_Sales,
factor_key = TRUE) %>%
group_by(area, Year_of_Release) %>%
summarise(sales = sum(sales)) %>%
ggplot() +
geom_line(aes(Year_of_Release, sales, group = area, color = area)) +
xlab("Year of release") + ylab("Sales") + labs(color = "") +
theme(legend.text = element_text(size = 7),
legend.position = "bottom",
axis.text.x = element_text(angle = 90))
```

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



North America is highest, followed by Europe. Let's take a peek at each platform's sales:

```
#Assessing sales per platform#
vg_sales %>% group_by(Platform) %>%
summarise(sales = sum(Global_Sales)) %>% ggplot() +
geom_bar(aes(reorder(Platform, sales), sales), stat = "identity",
fill = "blue") +
xlab("Platform") + ylab("Global sales") +
coord_flip()
```

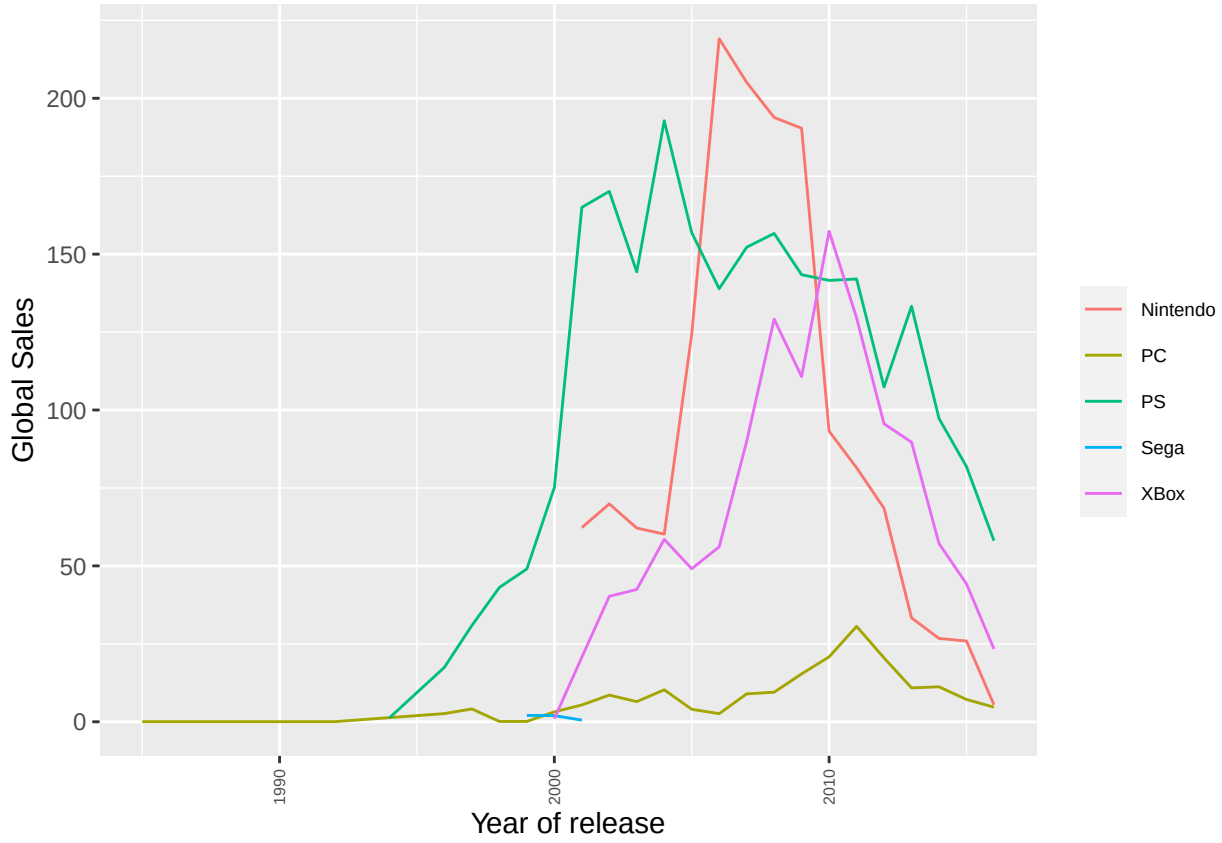


There are far too many platforms available. Create a new platform variable that categorizes these values into four groups: Nintendo, PlayStation, Xbox, PC, and Sega:

Now plot the global sales each year for each:

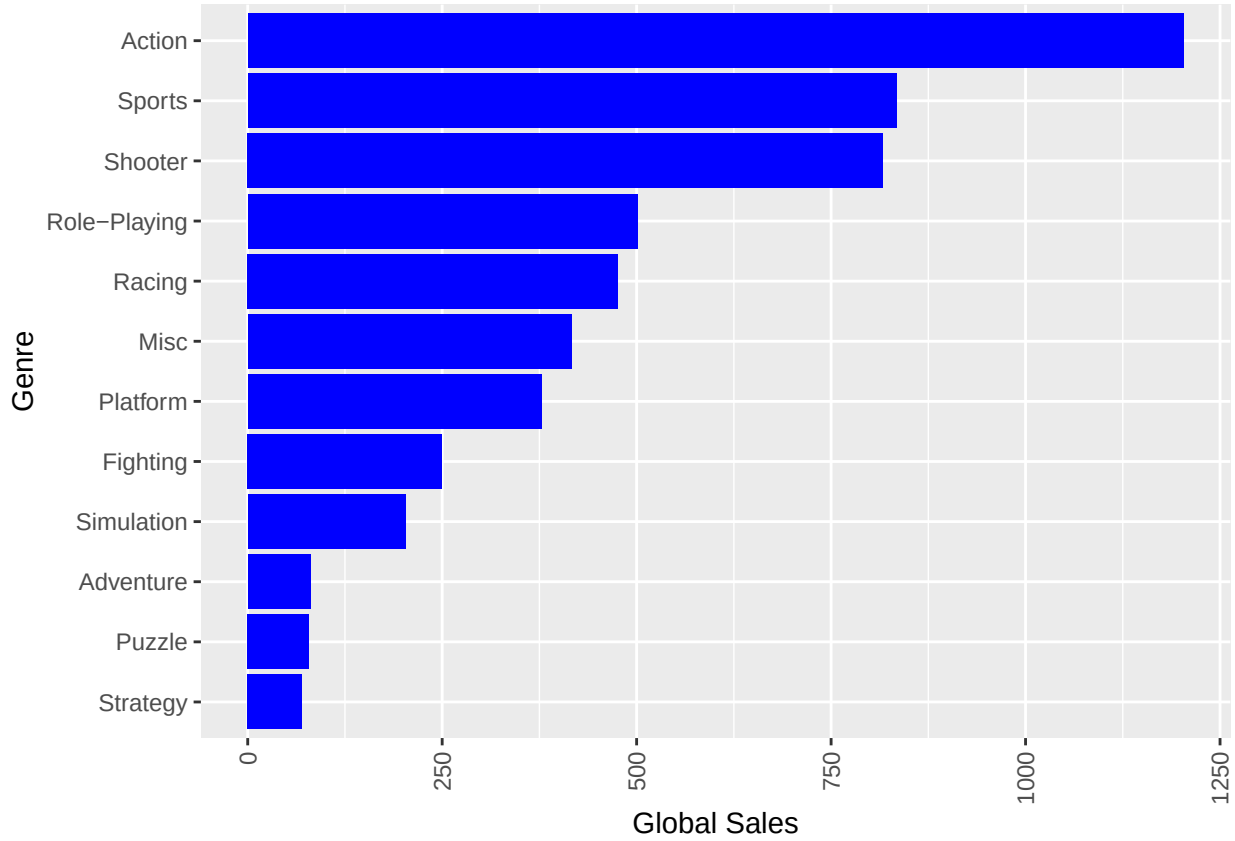
```
vg_sales %>% group_by(platform2, Year_of_Release) %>%
summarise(sales = sum(Global_Sales)) %>%
ggplot() +
geom_line(aes(Year_of_Release, sales, group = platform2, color = platform2)) +
xlab("Year of release") + ylab("Global Sales") + labs(color = "") +
theme(legend.text = element_text(size = 7),
axis.text.x = element_text(angle = 90, hjust = 1,
vjust = 0.5, size = 6))
```

```
## `summarise()` has grouped output by 'platform2'. You can override using the `.groups` argument.
```



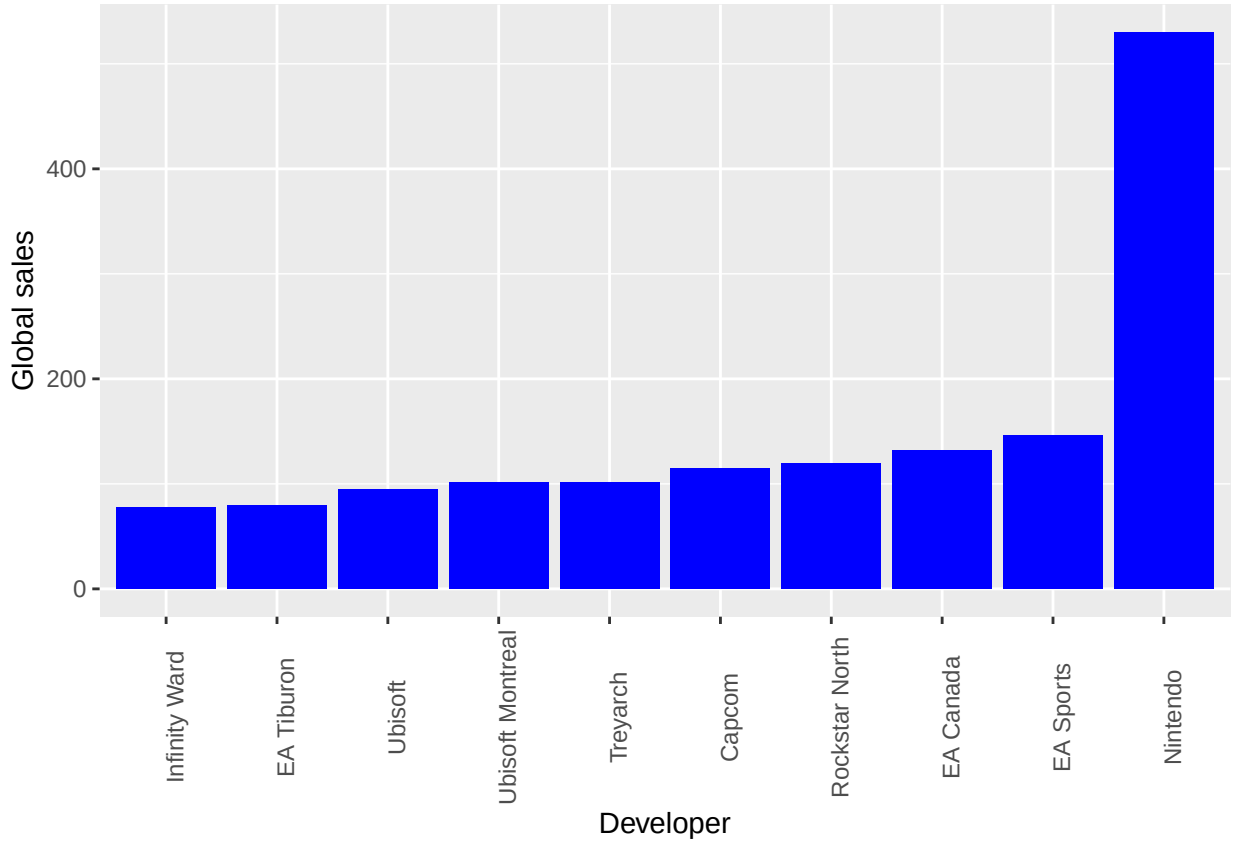
We now look at the sales for each gaming genre:

```
vg_sales %>% group_by(Genre) %>%
summarise(sales = sum(Global_Sales)) %>%
ggplot() +
geom_bar(aes(reorder(Genre, sales), sales), stat = "identity",
fill = "blue") +
ylab("Global Sales") + xlab("Genre") +
theme(axis.text.x = element_text(angle = 90,
hjust = 1, vjust = 0.5)) +
coord_flip()
```



There are major variances, as we can see. Let's take a look at each developer's sales:

```
vg_sales %>% group_by(Developer) %>%
summarise(sales = sum(Global_Sales)) %>%
arrange(desc(sales)) %>% slice(1:10) %>%
ggplot() +
geom_bar(aes(reorder(Developer, sales), sales), stat = "identity", fill = "blue") +
theme(axis.text.x = element_text(angle = 90)) +
xlab("Developer") + ylab("Global sales")
```



Nintendo develops the highest grossing games.

MODELLING

According to the results of the preceding exploratory investigation, there appears to be a link between sales and various characteristics. The number of sales varies significantly based on the developer, user and reviewer ratings, platform, and even release year. Variables for the developer and the publisher would be beneficial. These variables are categorical, yet they have a lot of different values. Because of their brand value, it makes reasonable that games made by top publishers and creators sell better. As a result, we compile a list of the best publishers and developers:

```
publishers_top <- (vg_sales %>% group_by(Publisher) %>%
summarise(sales = sum(Global_Sales)) %>% arrange(desc(sales)) %>%
top_n(10) %>% distinct(Publisher))$Publisher
```

```
## Selecting by sales
```

```
developers_top <- (vg_sales %>% group_by(Developer) %>%
summarise(sales = sum(Global_Sales)) %>% arrange(desc(sales)) %>%
top_n(10) %>% distinct(Developer))$Developer
```

```
## Selecting by sales
```

Now that we know who the top 10 publishers and developers are, we can add additional variables to the data set to indicate whether the game was released by one of them or not:

```
vg_sales <- vg_sales %>%
mutate(publisher_top = ifelse(Publisher %in% publishers_top, TRUE, FALSE),
developer_top = ifelse(Developer %in% developers_top, TRUE, FALSE))
```

The total number of platforms on which the game was released is another information that might be included. Some games are unique to a particular platform and are only available on that platform, while others are available on many platforms. It'd be fascinating to observe if exclusive titles sell better than non-exclusive games, or vice versa:

```
vg_sales <- vg_sales %>% group_by(Name) %>% mutate(num_of_platforms = n()) %>% ungroup(Name)
```

We may begin training the algorithm now that we have the variables that we are interested in. To begin, we must first prepare the training and testing data sets:

```
set.seed(1982, sample.kind = "Rounding")
```

```
## Warning in set.seed(1982, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <- createDataPartition(vg_sales$Global_Sales, p = 0.9, list = FALSE)
train_set <- vg_sales[-test_index, ]
test_set <- vg_sales[test_index, ]
```

We need to make sure that all possible values of the categorical variables are included in the training set.

```
totalData <- rbind(train_set, test_set)
for (f in 1:length(names(totalData))) {
levels(train_set[, f]) <- levels(totalData[, f])
}
```

METRICS

We then define an RMSE function as follows: RMSE measures the difference between actual observations and predictions

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

Linear Regression Model

```
model_lm <- train(log(Global_Sales) ~ Critic_Score +
User_Score + Genre +
Year_of_Release + Platform + Critic_Count +
User_Count + Rating +
publisher_top + developer_top +
num_of_platforms, method = "lm", data = train_set)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

We compute the RMSE using the expected values:

```
test_set$predicted_lm <- predict(model_lm, test_set)
rmse_results <- data.frame(Method = "Linear regression",
RMSE = RMSE(log(test_set$Global_Sales), test_set$predicted_lm))
```

SVM Linear Model

```
model_svm <- train(log(Global_Sales) ~ Critic_Score +
User_Score + Genre +
Year_of_Release + Platform + Critic_Count +
User_Count + Rating +
publisher_top + developer_top +
num_of_platforms, method = "svmLinear",
data = train_set)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.
```

We calculate the predicted values and compute the RMSE:

```
test_set$predicted_svm <- predict(model_svm, test_set)
rmse_results <- rmse_results %>%
add_row(Method = "SVM Linear",
RMSE = RMSE(log(test_set$Global_Sales),
test_set$predicted_svm))
```

Random Forest

Now we'll use cross validation to run a random forest model:

```
cntrl <- trainControl(method = "repeatedcv", number = 10,
repeats = 3)
tunegrid <- expand.grid(.mtry=c(1:5),
.min.node.size = seq(1, 5, 1),
.splitrule = c("extratrees", "variance"))
model_rf <- train(log(Global_Sales) ~ Critic_Score +
User_Score + Genre +
Year_of_Release + Platform + Critic_Count +
User_Count + Rating +
publisher_top + developer_top +
num_of_platforms, data = train_set,
method = "ranger", trControl = cntrl,
tuneGrid = tunegrid)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

We calculate the predicted values and compute the RMSE:

Comparing Models

We can now compare the RMSE values of each model:

```
##               Method     RMSE
## 1 Linear regression 1.028343
## 2        SVM Linear 1.051440
## 3     Random forest 1.000651
```

CONCLUSION

The video game sales data set was examined in this paper. In terms of platform, developer, game rating, and score, the exploratory study indicated differences in sales. The reviewer score has a stronger association with sales than the user score, according to this study. Having a good critic score appears to be far more significant than having a high user score. The research also indicated that some genres are more popular than others, with the action genre standing out. Nintendo and PlayStation were among the highest-grossing consoles. Nintendo was not just a successful gaming platform, but it was also discovered to be a popular developer, with Nintendo-developed titles earning far higher than other games. According to the machine learning methods utilized, the random forest outperformed linear regression and SVM. To get at the best result, cross validation was used. However, the model is not ideal. It was clear that the errors in some cases were not small, and that there was a pattern to these errors. In particular, it was found that the errors are largest for larger values of global sales. This means that future work needs to be done on the model in order to make it perform better for larger values of global sales.