Choose Your Own Project - Video Game Sales with Ratings

Wayne Jamieson

2022-10-28

Introduction

For the Data Science: Capstone 'Choose Your Own' project I have considered a Video Game Sales with Ratings dataset. This dataset was made available via kaggle.com and consists of a web scrape of VGChartz Video Game Sales data extended with critic and user score data from Meta Critic. I have considered a number of predictive approaches using machine learning in relation to expected sales along with critic and user reception.

The approach taken can be summarised as:

- Download and create the game sales dataset
- Analyse the dataset to understand its variables and properties
- Transform the sales data for predictive analysis
- Create train and test sets from game sales dataset
- Conduct analysis of algorithmic and logisitic models to predict game sales
- Perform further transformations to exclude missing score data in a separate dataset
- Consider linear regression models for the relationships between critic & user scores in determining game sales
- Conduct analysis of algorithmic models to predict critical and user reception

Start by installing and loading the required libraries then downloading from a csv in a github repository and creating the game sales dataset.

```
# Install if required and load 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(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(arules)) install.packages("arules", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(randomForest)
library(arules)

# Download and read the vieao game sales csv file
dl <- tempfile()
download.file("https://github.com/w-jamieson/EDX-CYO/raw/main/Video_Games_Sales_as_at_22_Dec_2016.csv",

gamesales <- read.csv(file = dl)
```

Methods & Analysis

We start by examing the game sales dataset created.

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

```
##                      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 data has 16,719 unique games and consists of 16 variables capturing:

- Identification and classification data such as:
  - Name
  - Platform
  - Year of Release
  - Genre
  - Publisher
  - Developer
  - Rating
- Sales figures in millions by region
- Critic scores and number of reviews
- User Scores and number of reviews

It is noted that critic and user scores are not available for all games and user scores are returned as a character variable. These points will need to be adjusted via transformation prior to conducting analysis related to

critic and user scores.

We next look at high level characteristics of the Global Sales data

```
# Number of games
nrow(gamesales)
```

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

```
# Total sales
sum(gamesales$Global_Sales)
```

```
## [1] 8920.3
```

```
# Average sales
mean(gamesales$Global_Sales)
```

```
## [1] 0.5335427
```

```
# Standard deviation of sales
sd(gamesales$Global_Sales)
```

```
## [1] 1.547935
```

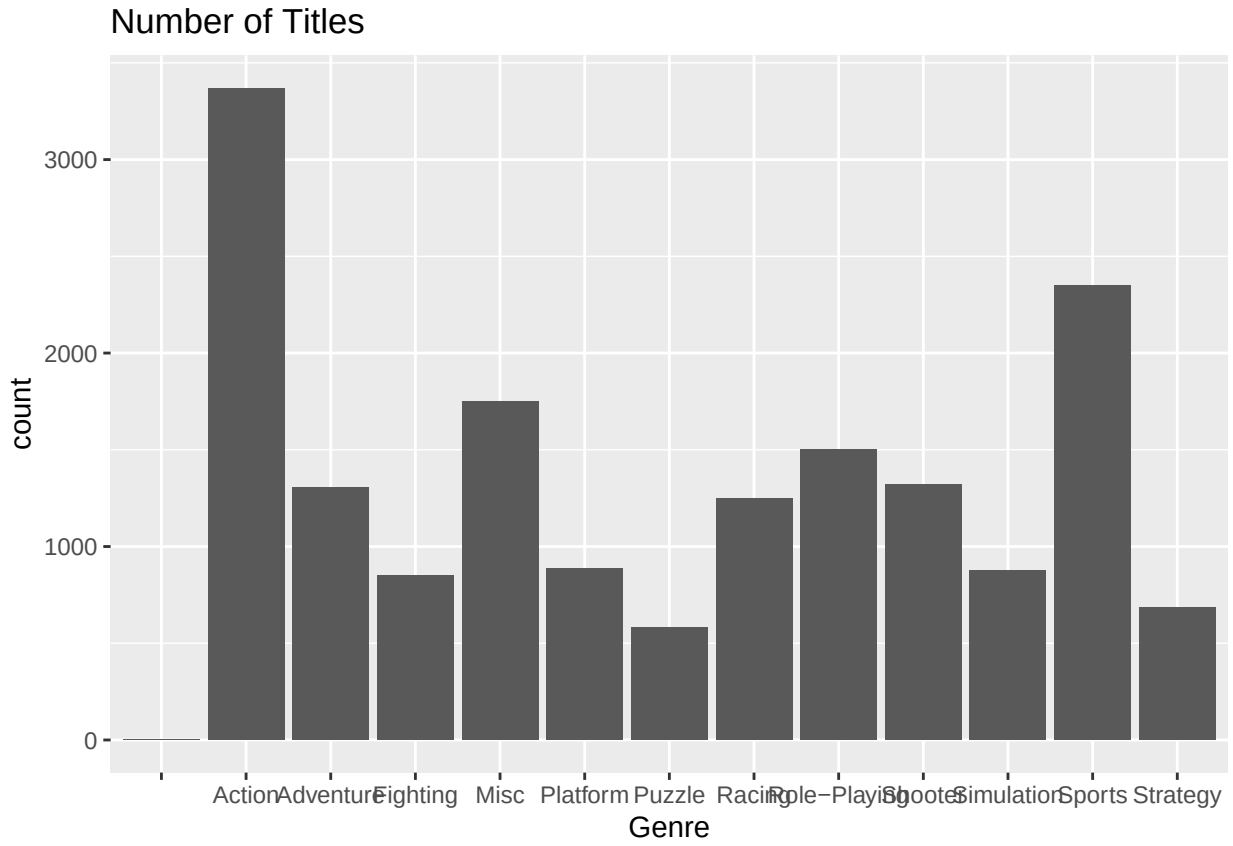
```
# Highest selling game figure
max(gamesales$Global_Sales)
```

```
## [1] 82.53
```

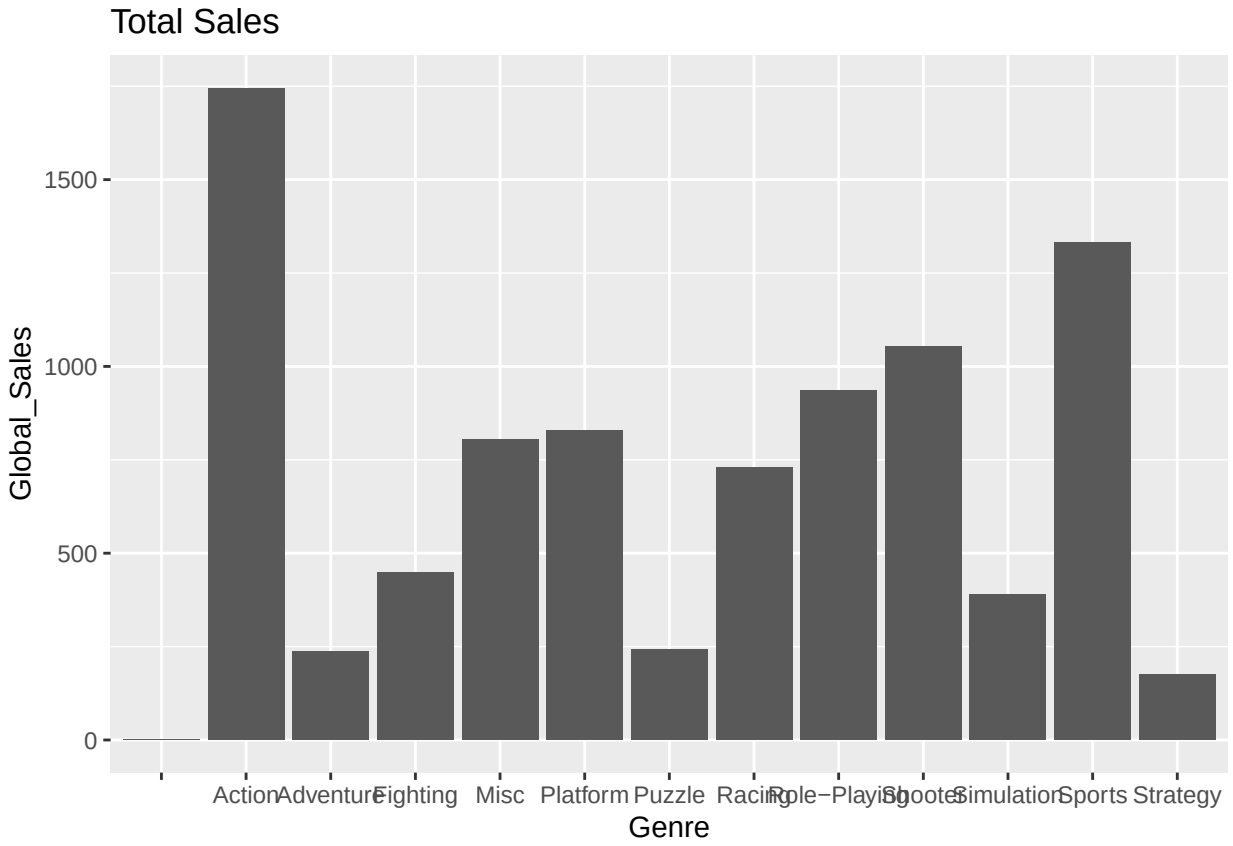
Its noted that a small number of the many games make up the bulk of the sales.

We continue by considering analysis of sales data by various characteristics, firstly by Genre looking at:

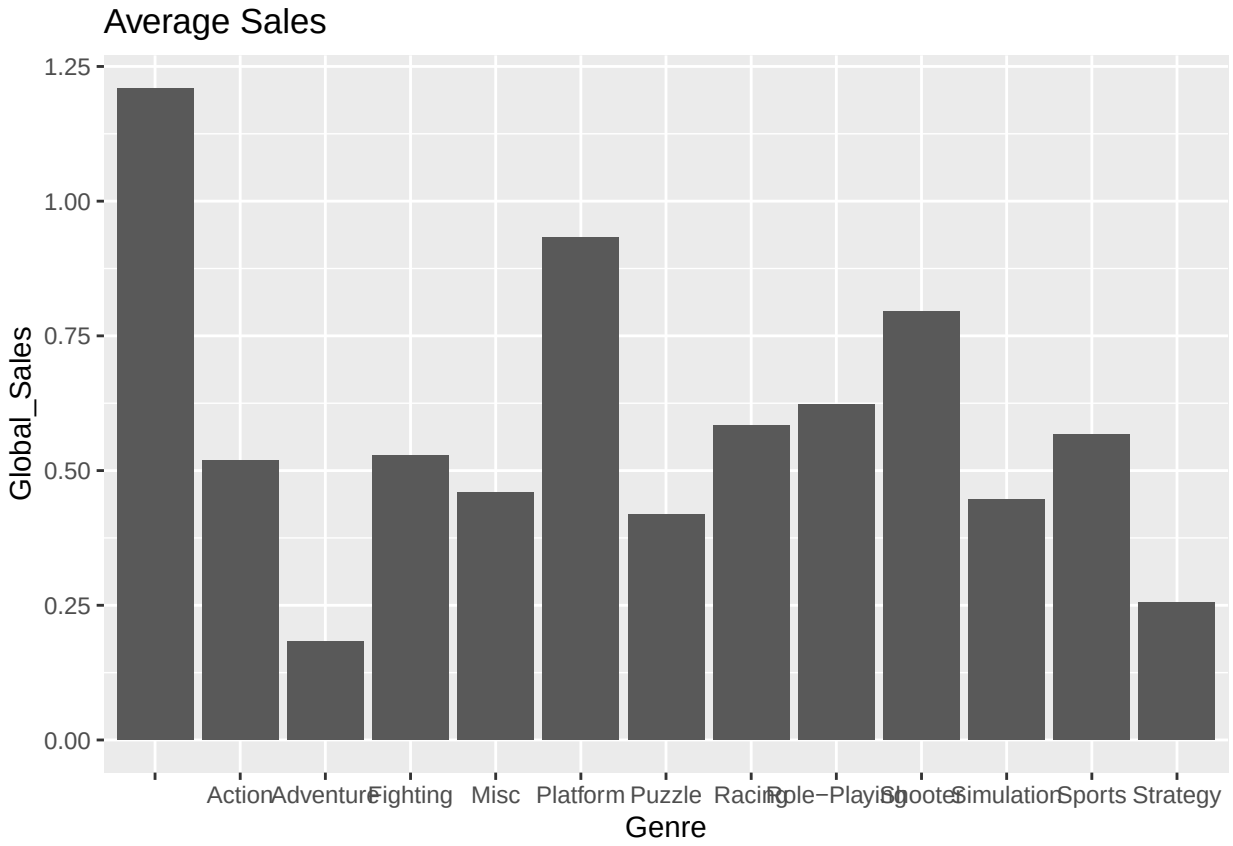
- Number of titles
- Total sales
- Average sales per title
- Standard deviation of sales



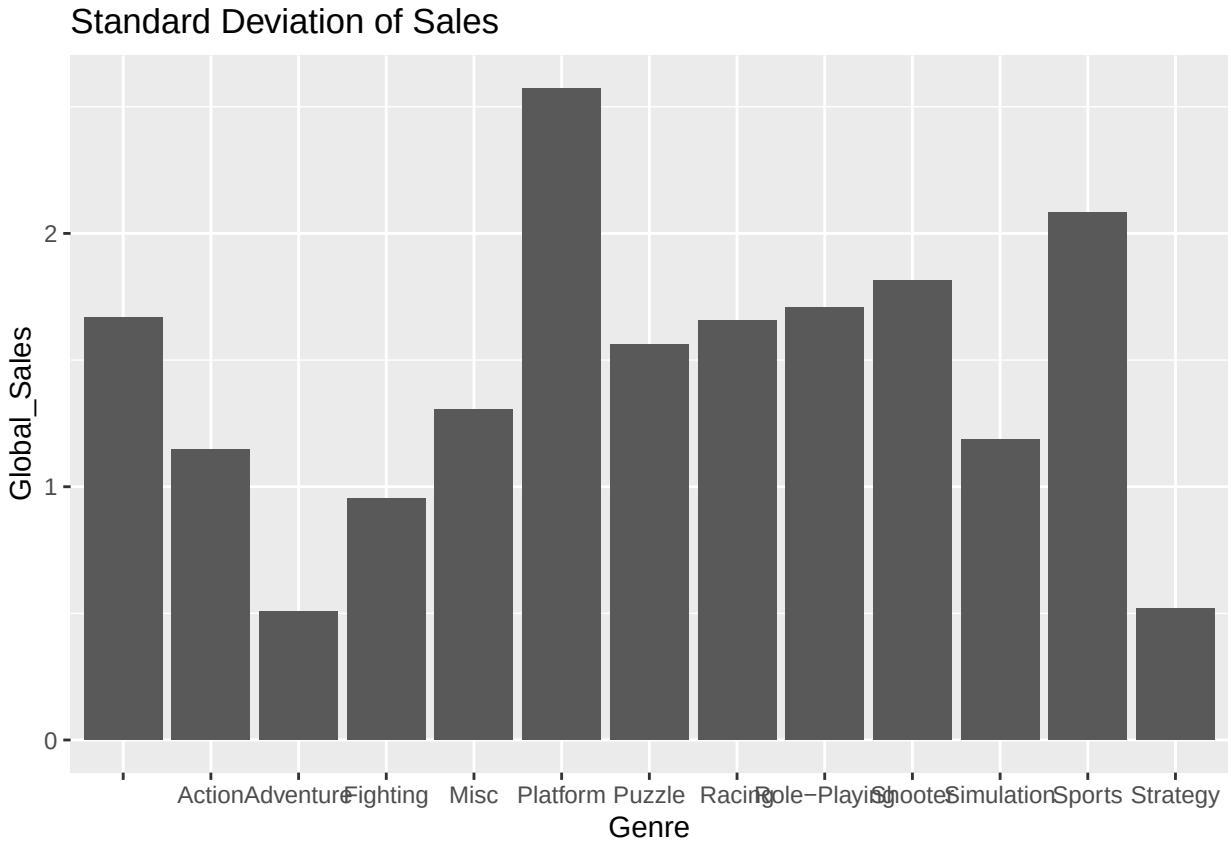

```
##           Genre    n
## 1                  2
## 2        Action 3370
## 3     Adventure 1303
## 4      Fighting  849
## 5          Misc 1750
## 6      Platform  888
## 7        Puzzle  580
## 8        Racing 1249
## 9  Role-Playing 1500
## 10      Shooter 1323
## 11   Simulation  874
## 12       Sports 2348
## 13     Strategy  683
```




```
## # A tibble: 13 x 2
##    Genre          TotalSales
##    <chr>               <dbl>
##  1 ""                   2.42
##  2 "Action"          1745.
##  3 "Adventure"        238.
##  4 "Fighting"         447.
##  5 "Misc"             803.
##  6 "Platform"         828.
##  7 "Puzzle"           243.
##  8 "Racing"           729.
##  9 "Role-Playing"     934.
## 10 "Shooter"         1053.
## 11 "Simulation"       390.
## 12 "Sports"          1332.
## 13 "Strategy"         175.
```




```
## # A tibble: 13 x 2
##    Genre          AvgSales
##    <chr>             <dbl>
##  1 ""                1.21
##  2 "Action"          0.518
##  3 "Adventure"       0.182
##  4 "Fighting"        0.527
##  5 "Misc"            0.459
##  6 "Platform"        0.933
##  7 "Puzzle"          0.419
##  8 "Racing"          0.584
##  9 "Role-Playing"    0.623
## 10 "Shooter"         0.796
## 11 "Simulation"      0.447
## 12 "Sports"          0.567
## 13 "Strategy"        0.255
```

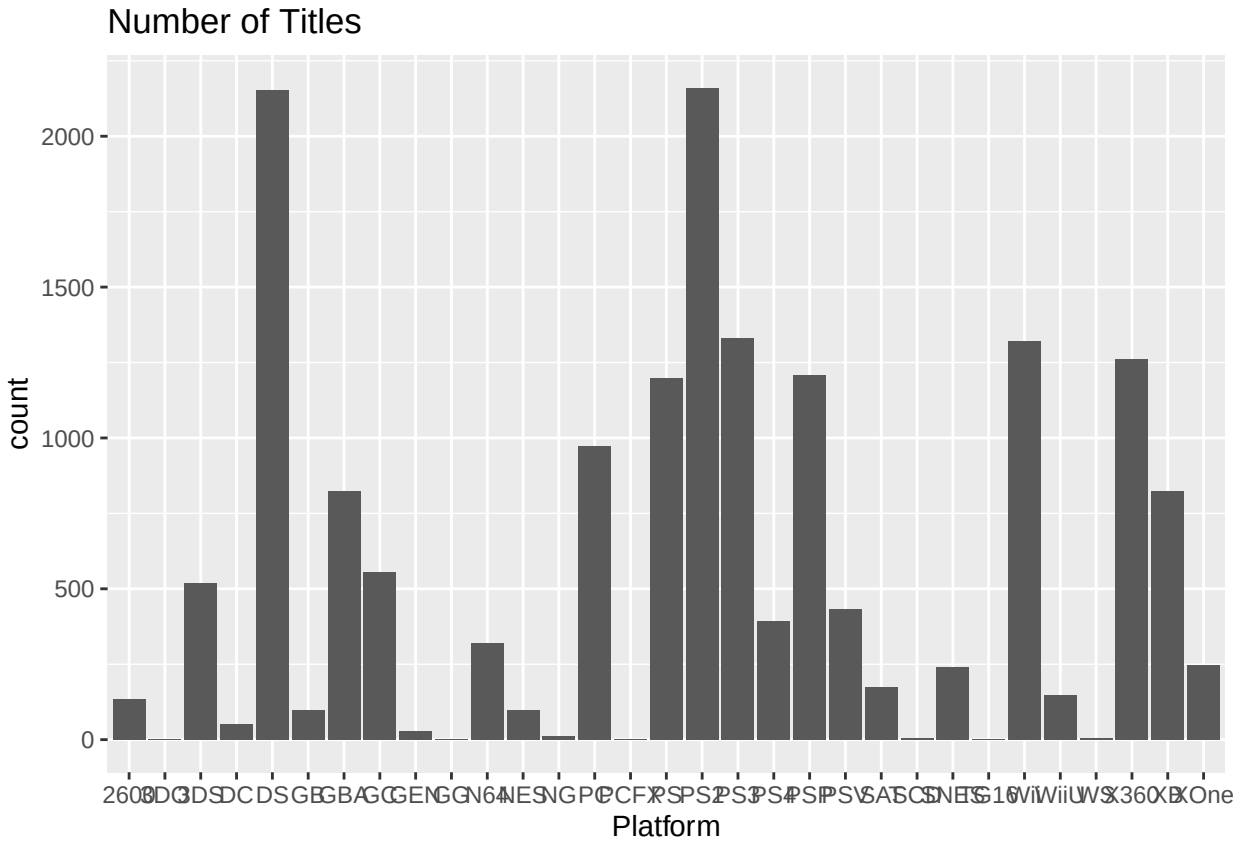



```
## # A tibble: 13 x 2
##    Genre          SdSales
##    <chr>            <dbl>
##  1 ""               1.67
##  2 "Action"         1.15
##  3 "Adventure"      0.508
##  4 "Fighting"       0.952
##  5 "Misc"           1.30
##  6 "Platform"       2.57
##  7 "Puzzle"         1.56
##  8 "Racing"         1.66
##  9 "Role-Playing"   1.71
## 10 "Shooter"        1.82
## 11 "Simulation"     1.19
## 12 "Sports"         2.08
## 13 "Strategy"       0.519
```

```
## # A tibble: 13 x 4
##    Genre          TotalSales AvgSales SdSales
##    <chr>               <dbl>    <dbl>   <dbl>
##  1 "Action"          1745.      0.518   1.15
##  2 "Sports"          1332.      0.567   2.08
##  3 "Shooter"         1053.      0.796   1.82
##  4 "Role-Playing"     934.      0.623   1.71
##  5 "Platform"         828.      0.933   2.57
##  6 "Misc"             803.      0.459   1.30
##  7 "Racing"           729.      0.584   1.66
##  8 "Fighting"         447.      0.527   0.952
```

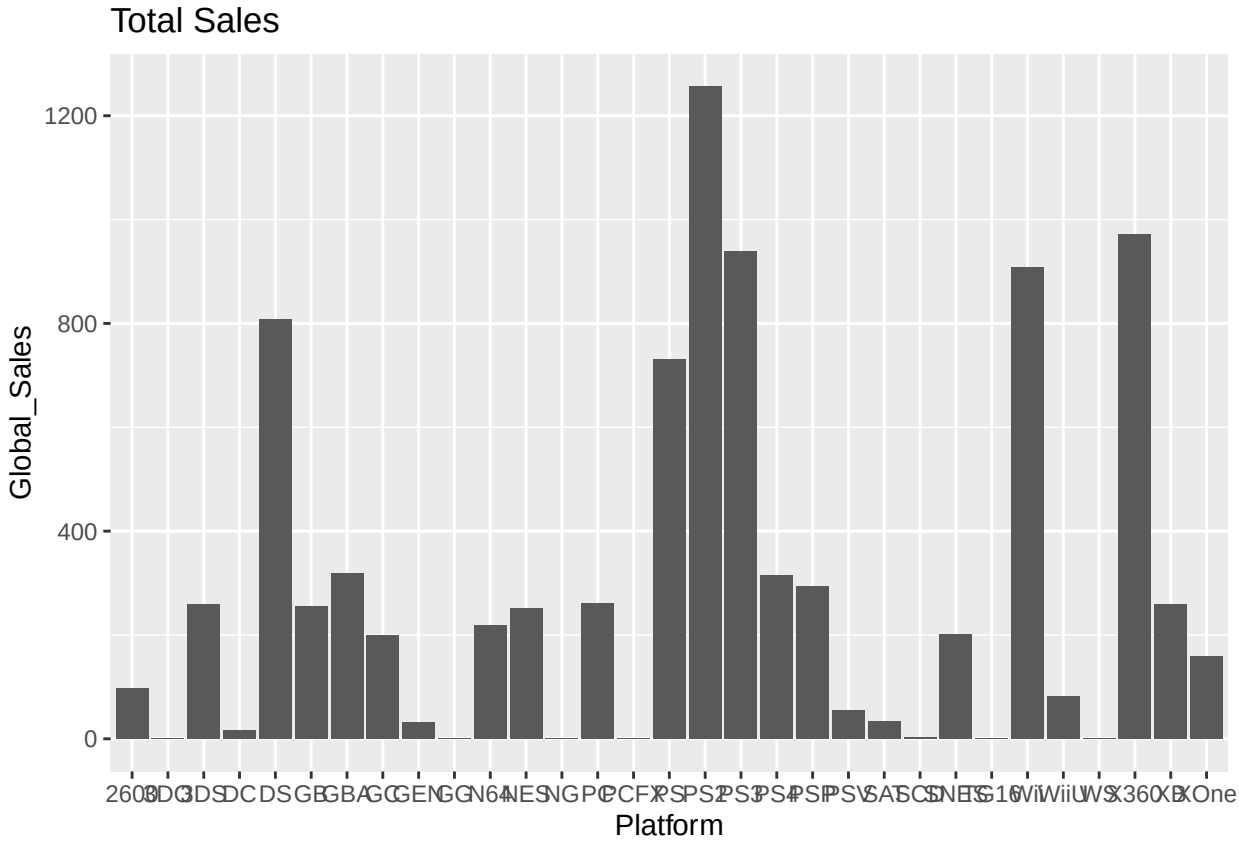
```
##  9 "Simulation"       390.      0.447   1.19
## 10 "Puzzle"           243.      0.419   1.56
## 11 "Adventure"        238.      0.182   0.508
## 12 "Strategy"         175.      0.255   0.519
## 13 ""                   2.42    1.21    1.67
```

Repeating the above analysis by Platform:

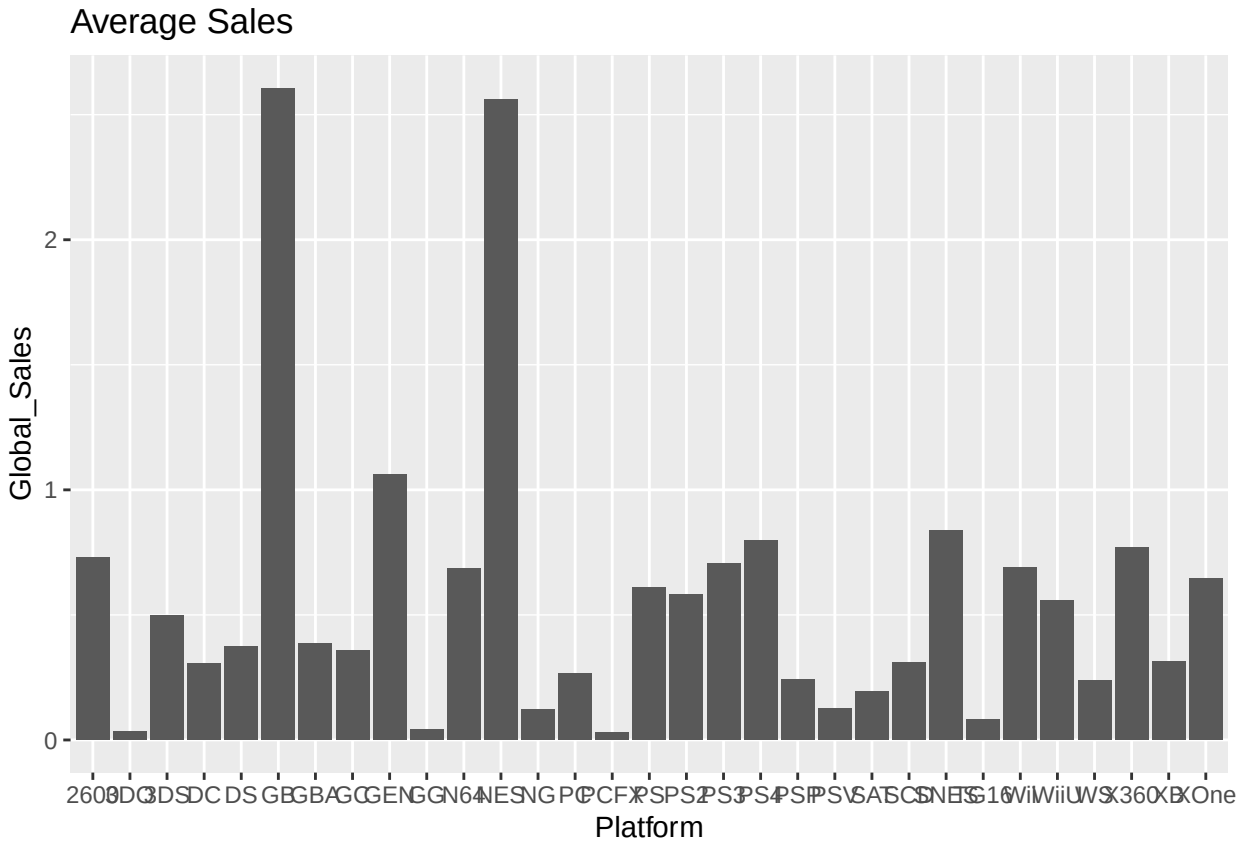



```
##    Platform    n
## 1      2600  133
## 2       3DO    3
## 3       3DS  520
## 4        DC   52
## 5        DS 2152
## 6        GB   98
## 7       GBA  822
## 8        GC  556
## 9       GEN   29
## 10       GG    1
## 11      N64  319
## 12      NES   98
## 13       NG   12
## 14       PC  974
## 15     PCFX    1
## 16       PS 1197
## 17      PS2 2161
## 18      PS3 1331
## 19      PS4  393
```

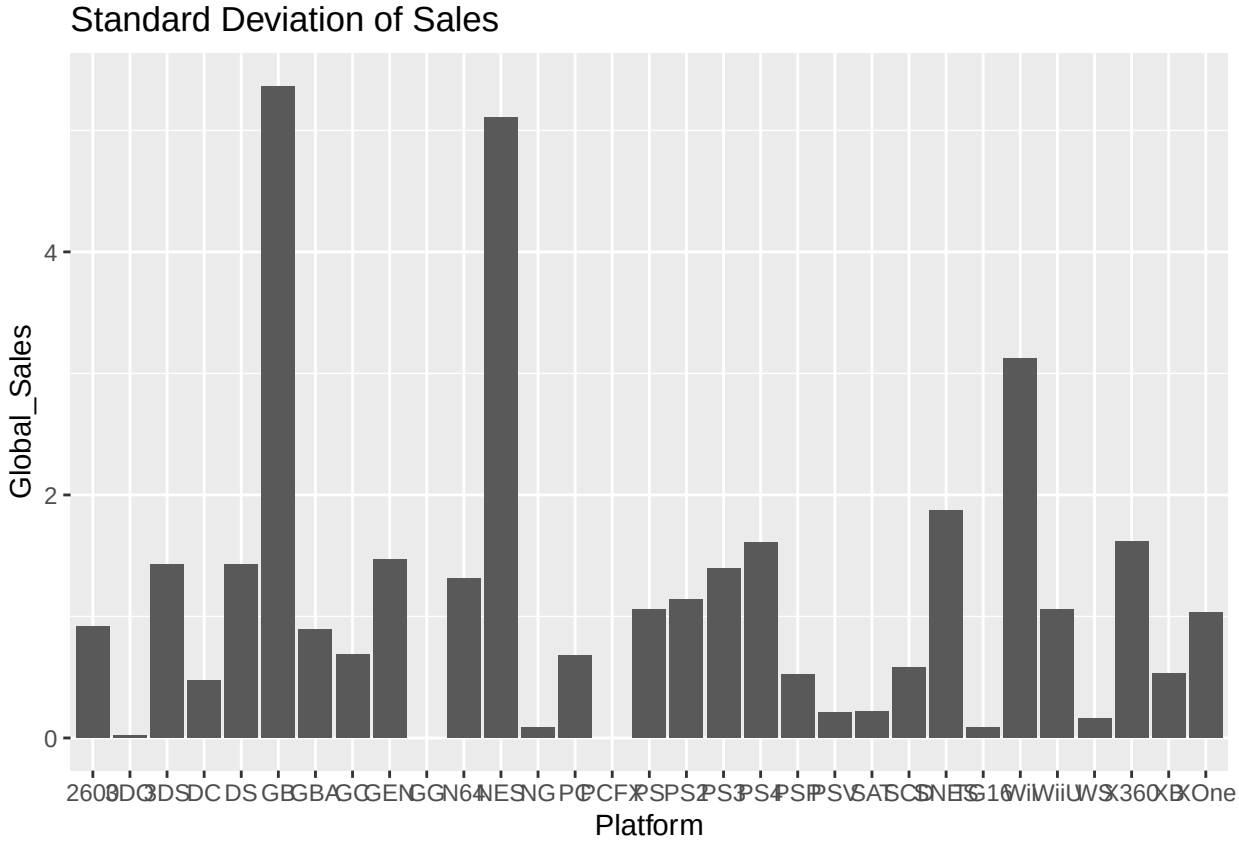
```
## 20      PSP 1209
## 21      PSV  432
## 22      SAT  173
## 23      SCD    6
## 24     SNES  239
## 25     TG16    2
## 26      Wii 1320
## 27     WiiU  147
## 28       WS    6
## 29     X360 1262
## 30       XB  824
## 31     XOne  247
```




```
## # A tibble: 31 x 2
##    Platform TotalSales
##    <chr>         <dbl>
##  1 2600          97.1
##  2 3DO            0.1
##  3 3DS          259.
##  4 DC            16.0
##  5 DS           807.
##  6 GB           255.
##  7 GBA          318.
##  8 GC           199.
##  9 GEN           30.8
## 10 GG             0.04
## # ... with 21 more rows
```




```
## # A tibble: 31 x 2
##    Platform AvgSales
##    <chr>       <dbl>
##  1 2600       0.730
##  2 3DO        0.0333
##  3 3DS        0.498
##  4 DC         0.307
##  5 DS         0.375
##  6 GB         2.61
##  7 GBA        0.387
##  8 GC         0.359
##  9 GEN        1.06
## 10 GG         0.04
## # ... with 21 more rows
```

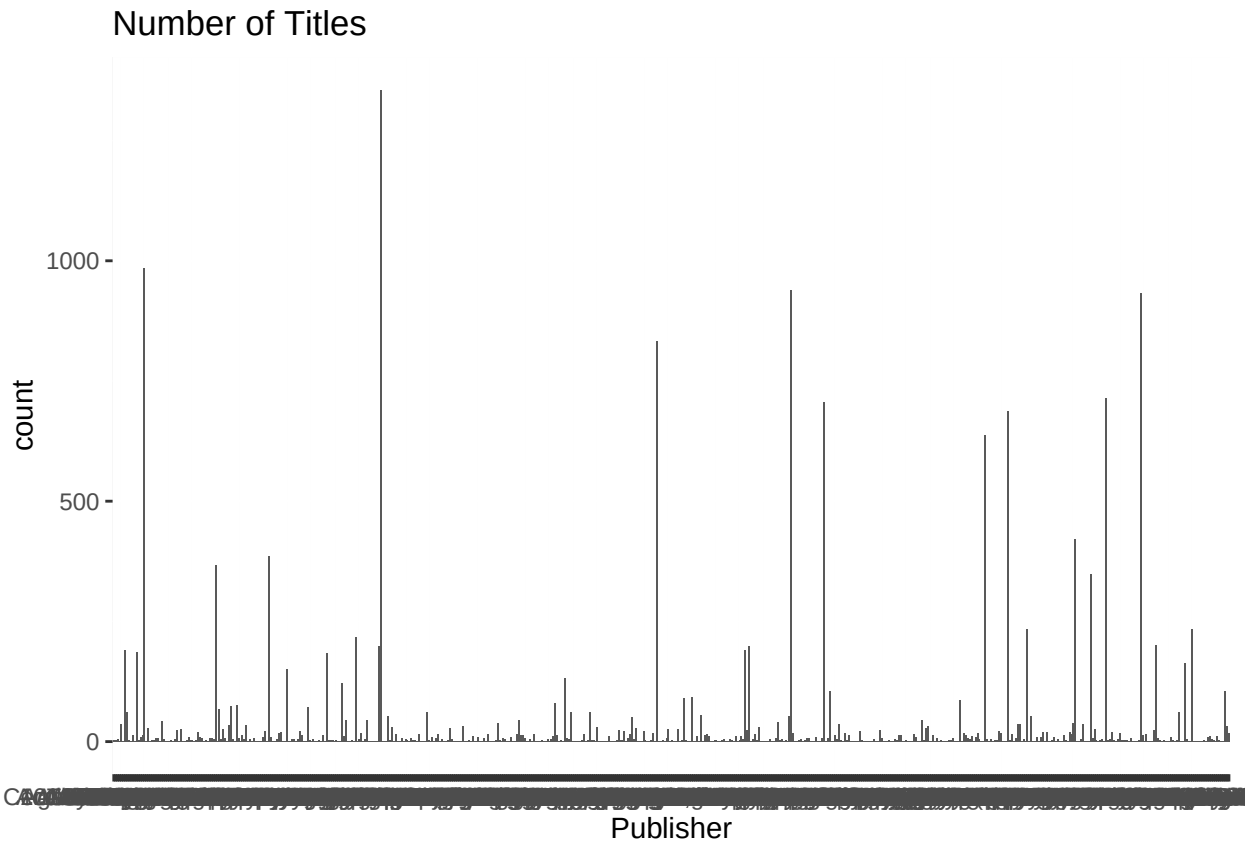



```
## # A tibble: 31 x 2
##    Platform SdSales
##    <chr>      <dbl>
##  1 2600      0.917
##  2 3DO       0.0231
##  3 3DS       1.43
##  4 DC        0.470
##  5 DS        1.43
##  6 GB        5.37
##  7 GBA       0.897
##  8 GC        0.685
##  9 GEN       1.47
## 10 GG       NA
## # ... with 21 more rows
```

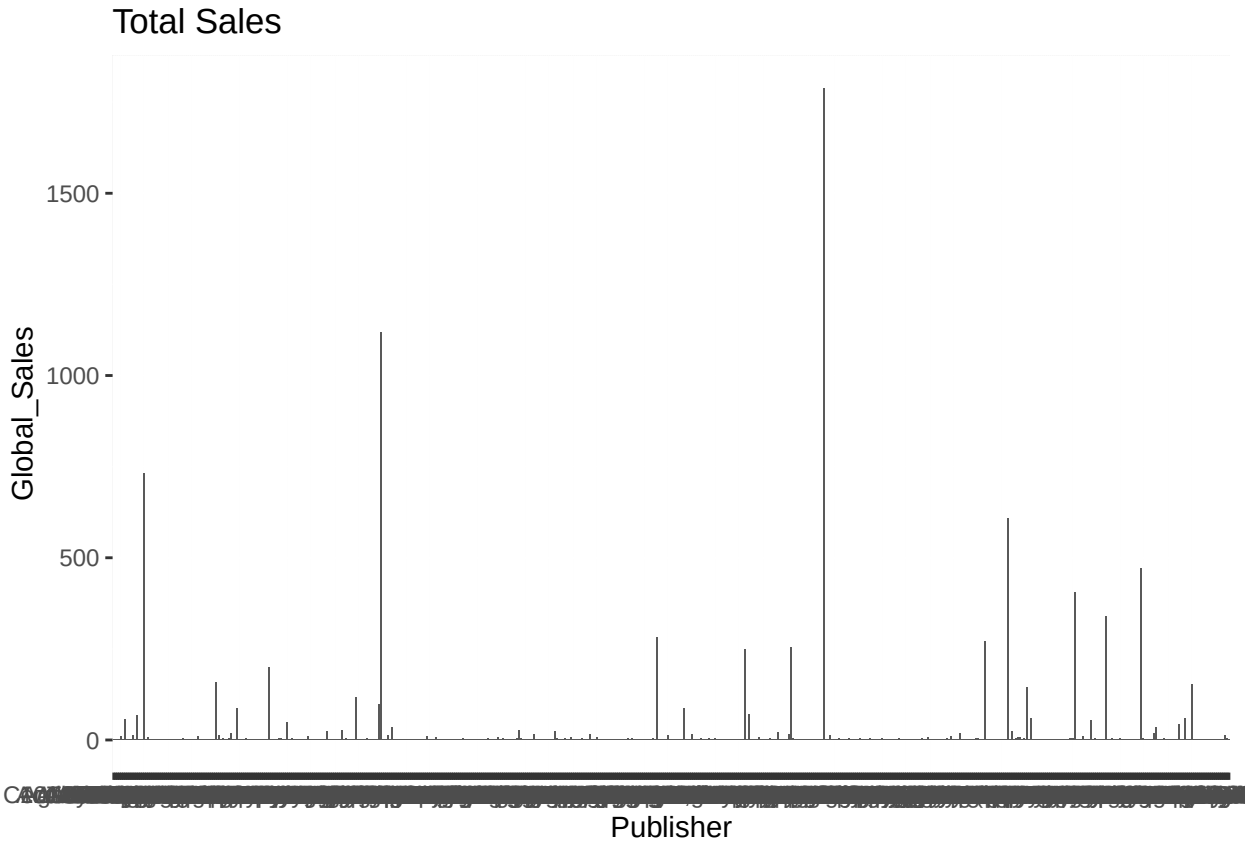
```
## # A tibble: 31 x 4
##    Platform TotalSales AvgSales SdSales
##    <chr>         <dbl>    <dbl>   <dbl>
##  1 PS2           1256.    0.581   1.14
##  2 X360           972.    0.770   1.62
##  3 PS3            939.    0.706   1.39
##  4 Wii            908.    0.688   3.13
##  5 DS             807.    0.375   1.43
##  6 PS             731.    0.610   1.05
##  7 GBA            318.    0.387   0.897
##  8 PS4            314.    0.800   1.61
##  9 PSP            294.    0.243   0.520
## 10 PC             260.    0.267   0.676
```

```
## # ... with 21 more rows
```

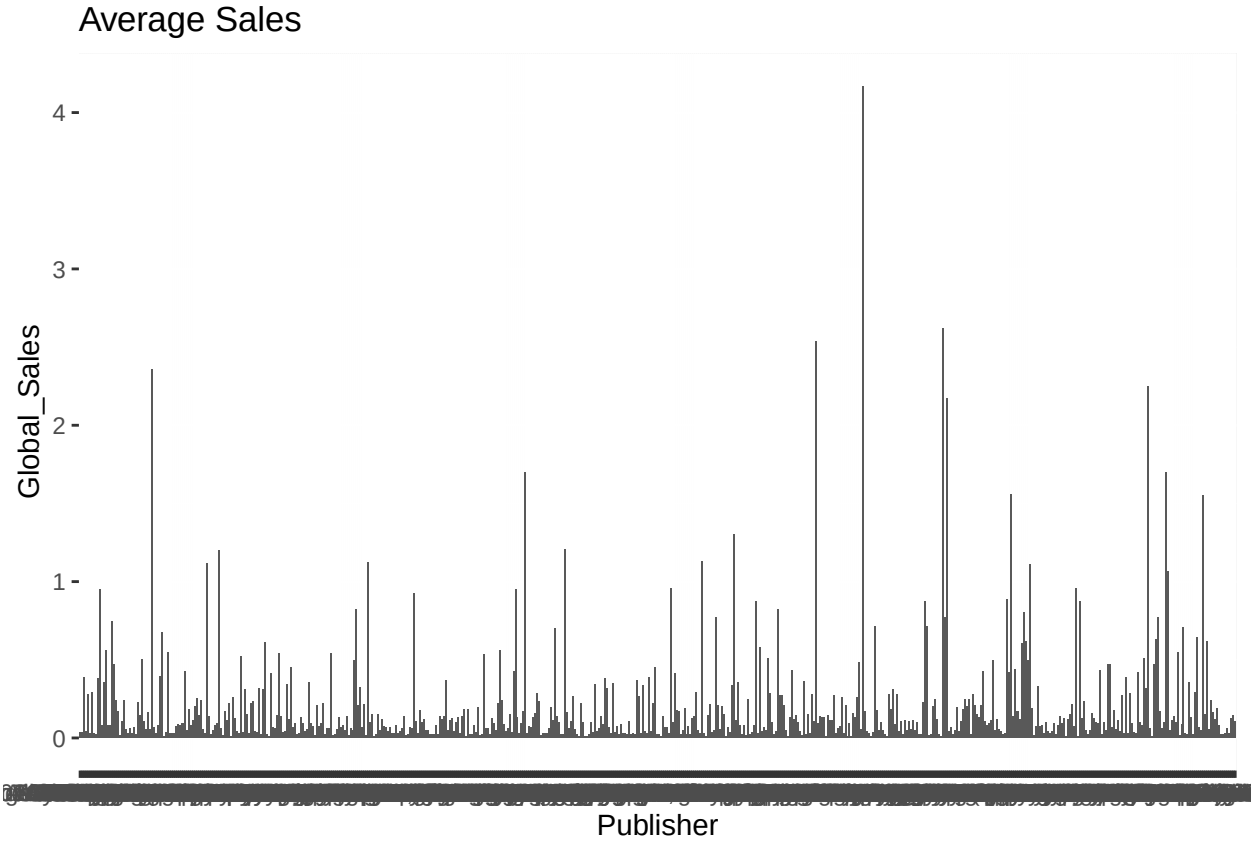
Repeating the analysis by Publisher (tables outputs heavily summarised due to number of publishers):



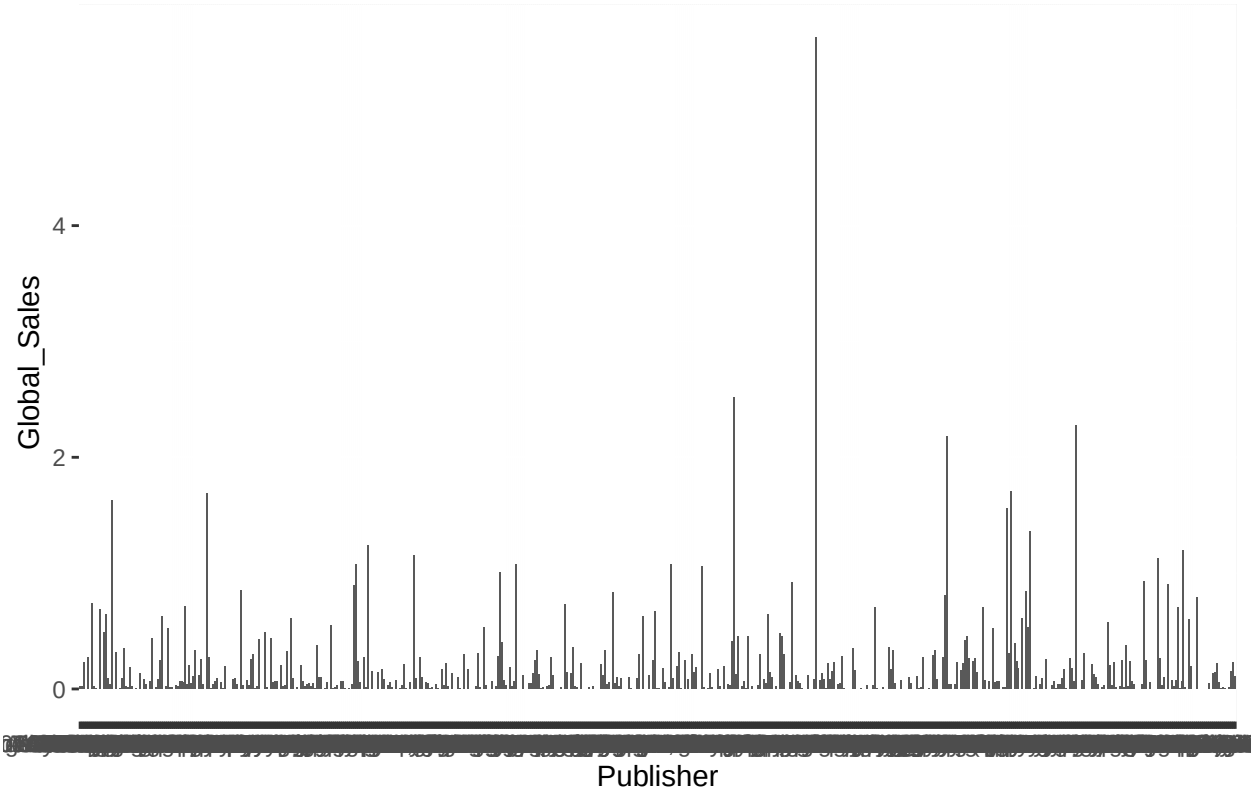

```
##                       Publisher  n
## 1               10TACLE Studios  3
## 2                    1C Company  3
## 3 20th Century Fox Video Games  5
## 4                         2D Boy  1
## 5                            3DO 36
## 6                        49Games  1
```




```
## # A tibble: 582 x 2
##    Publisher                     TotalSales
##    <chr>                              <dbl>
##  1 10TACLE Studios                     0.11
##  2 1C Company                          0.1
##  3 20th Century Fox Video Games        1.94
##  4 2D Boy                              0.04
##  5 3DO                                10.1
##  6 49Games                             0.03
##  7 505 Games                          55.3
##  8 5pb                                 1.66
##  9 7G//AMES                            0.06
## 10 989 Sports                          0.38
## # ... with 572 more rows
```




```
## # A tibble: 582 x 2
##    Publisher                     AvgSales
##    <chr>                            <dbl>
##  1 10TACLE Studios                 0.0367
##  2 1C Company                      0.0333
##  3 20th Century Fox Video Games    0.388
##  4 2D Boy                          0.04
##  5 3DO                             0.281
##  6 49Games                         0.03
##  7 505 Games                       0.290
##  8 5pb                             0.0268
##  9 7G//AMES                        0.02
## 10 989 Sports                      0.38
## # ... with 572 more rows
```

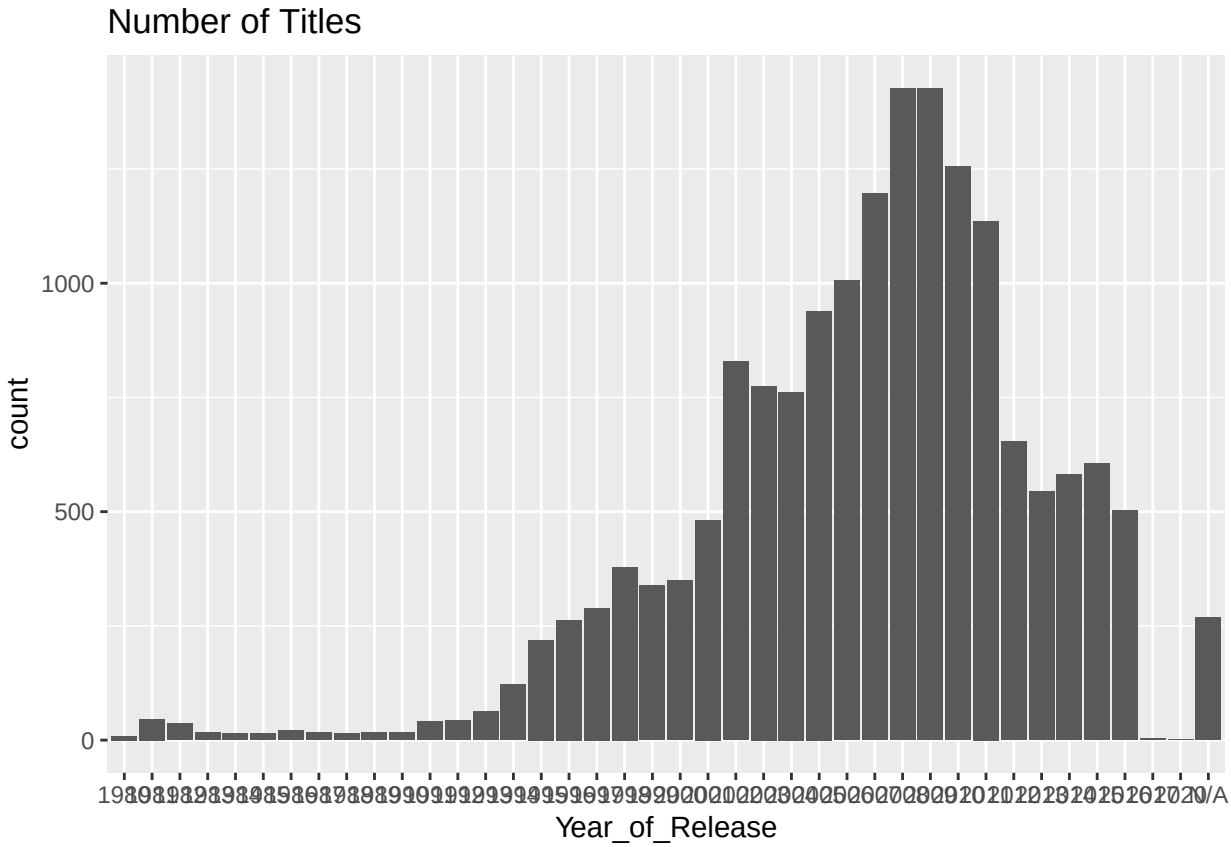



```
## # A tibble: 582 x 2
##    Publisher                     SdSales
##    <chr>                           <dbl>
##  1 10TACLE Studios                0.0208
##  2 1C Company                     0.0208
##  3 20th Century Fox Video Games   0.231
##  4 2D Boy                        NA
##  5 3DO                            0.273
##  6 49Games                       NA
##  7 505 Games                      0.737
##  8 5pb                            0.0192
##  9 7G//AMES                       0
## 10 989 Sports                    NA
## # ... with 572 more rows
```

```
## # A tibble: 582 x 4
##    Publisher                     TotalSales AvgSales SdSales
##    <chr>                              <dbl>    <dbl>   <dbl>
##  1 Nintendo                           1789.    2.53    5.63
##  2 Electronic Arts                    1117.    0.824   1.07
##  3 Activision                          731.    0.742   1.63
##  4 Sony Computer Entertainment         606.    0.883   1.56
##  5 Ubisoft                             472.    0.505   0.927
##  6 Take-Two Interactive                404.    0.957   2.28
##  7 THQ                                 338.    0.473   0.573
##  8 Konami Digital Entertainment        282.    0.339   0.625
##  9 Sega                                270.    0.424   0.707
## 10 Namco Bandai Games                  255.    0.271   0.456
```

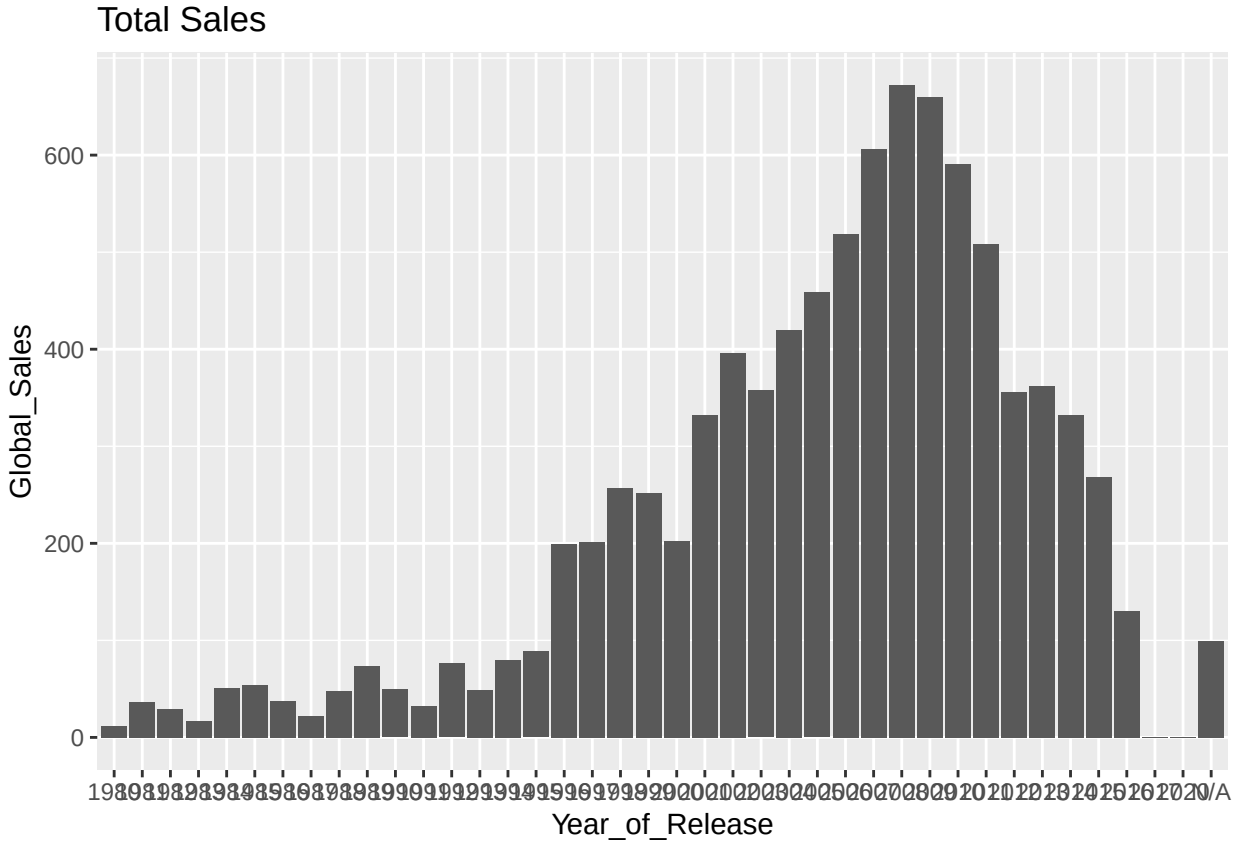
```
## # ... with 572 more rows
```

Finally repeat the analysis by Year:



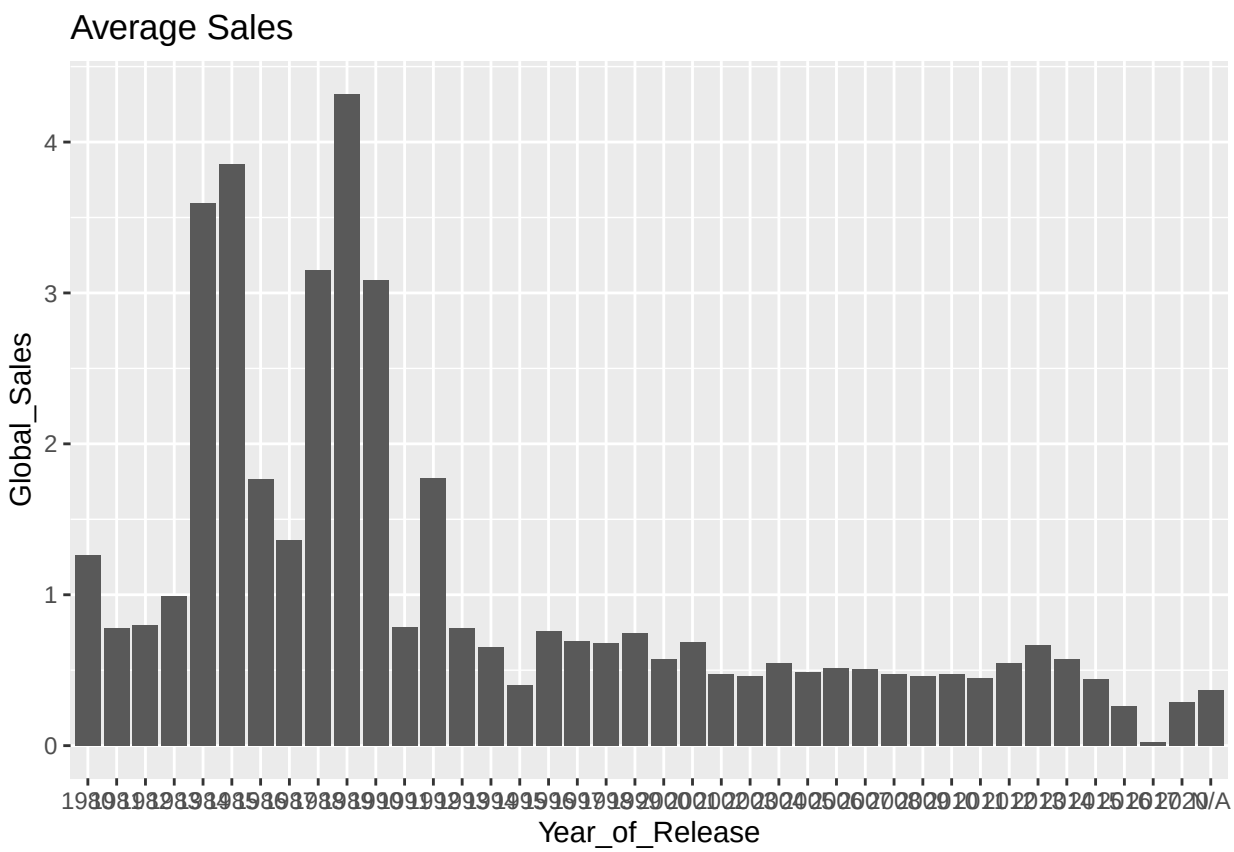

```
##    Year_of_Release    n
## 1             1980    9
## 2             1981   46
## 3             1982   36
## 4             1983   17
## 5             1984   14
## 6             1985   14
## 7             1986   21
## 8             1987   16
## 9             1988   15
## 10            1989   17
## 11            1990   16
## 12            1991   41
## 13            1992   43
## 14            1993   62
## 15            1994  121
## 16            1995  219
## 17            1996  263
## 18            1997  289
## 19            1998  379
## 20            1999  338
## 21            2000  350
## 22            2001  482
## 23            2002  829
```

```
## 24            2003  775
## 25            2004  762
## 26            2005  939
## 27            2006 1006
## 28            2007 1197
## 29            2008 1427
## 30            2009 1426
## 31            2010 1255
## 32            2011 1136
## 33            2012  653
## 34            2013  544
## 35            2014  581
## 36            2015  606
## 37            2016  502
## 38            2017    3
## 39            2020    1
## 40             N/A  269
```

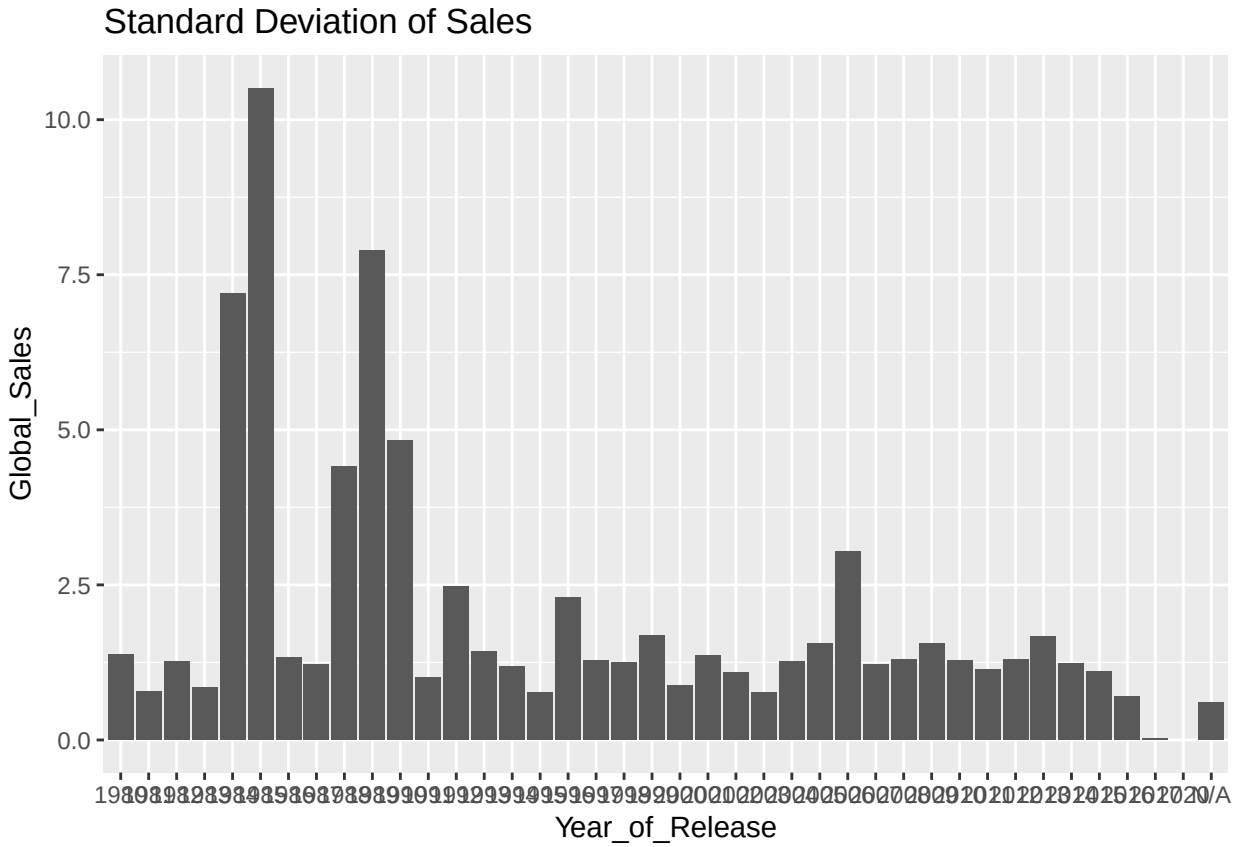



```
## # A tibble: 40 x 2
##    Year_of_Release TotalSales
##    <chr>                <dbl>
##  1 1980                  11.4
##  2 1981                  35.8
##  3 1982                  28.9
##  4 1983                  16.8
##  5 1984                  50.4
##  6 1985                  53.9
```

```
##  7 1986                  37.1
##  8 1987                  21.7
##  9 1988                  47.2
## 10 1989                  73.4
## # ... with 30 more rows
```




```
## # A tibble: 40 x 2
##    Year_of_Release AvgSales
##    <chr>              <dbl>
##  1 1980               1.26
##  2 1981               0.778
##  3 1982               0.802
##  4 1983               0.988
##  5 1984               3.60
##  6 1985               3.85
##  7 1986               1.77
##  8 1987               1.36
##  9 1988               3.15
## 10 1989               4.32
## # ... with 30 more rows
```



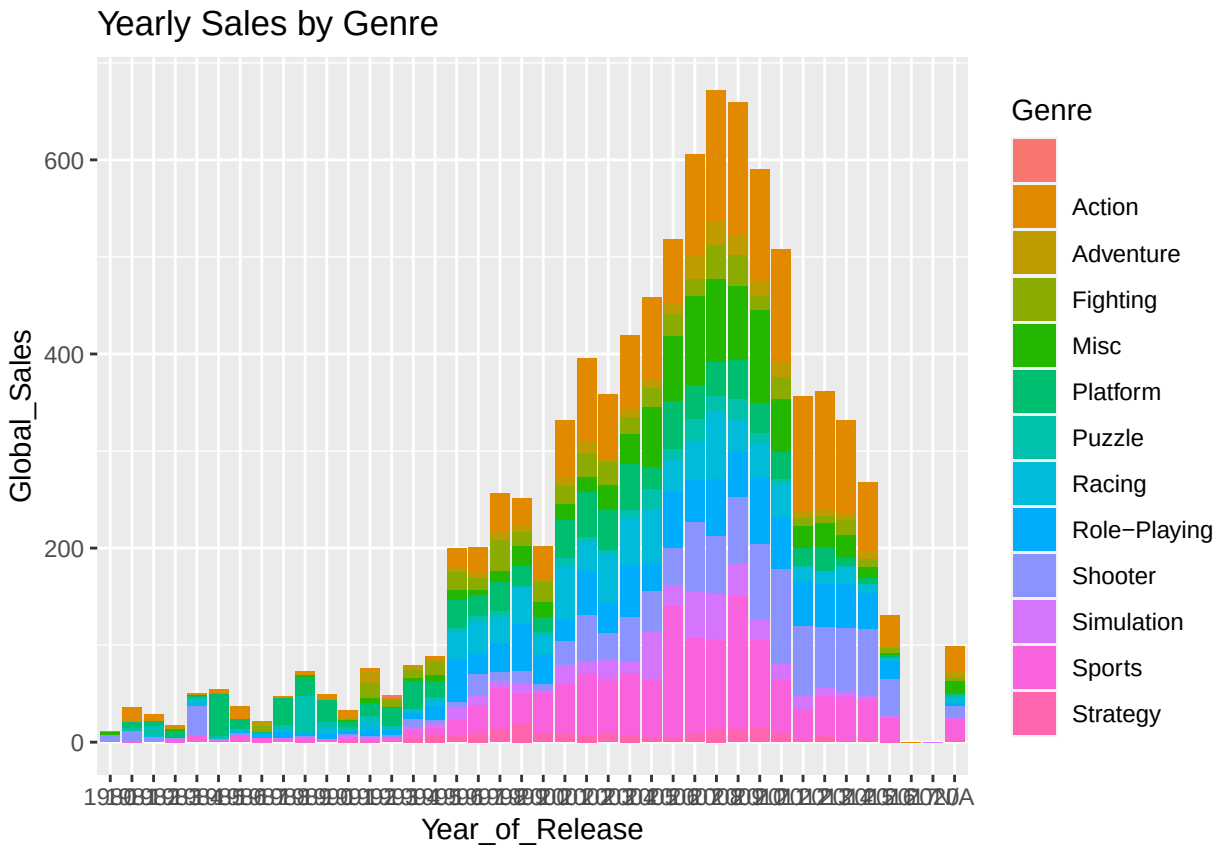

```
## # A tibble: 40 x 2
##    Year_of_Release SdSales
##    <chr>             <dbl>
##  1 1980              1.38
##  2 1981              0.782
##  3 1982              1.26
##  4 1983              0.839
##  5 1984              7.20
##  6 1985             10.5
##  7 1986              1.33
##  8 1987              1.22
##  9 1988              4.41
## 10 1989              7.90
## # ... with 30 more rows
```

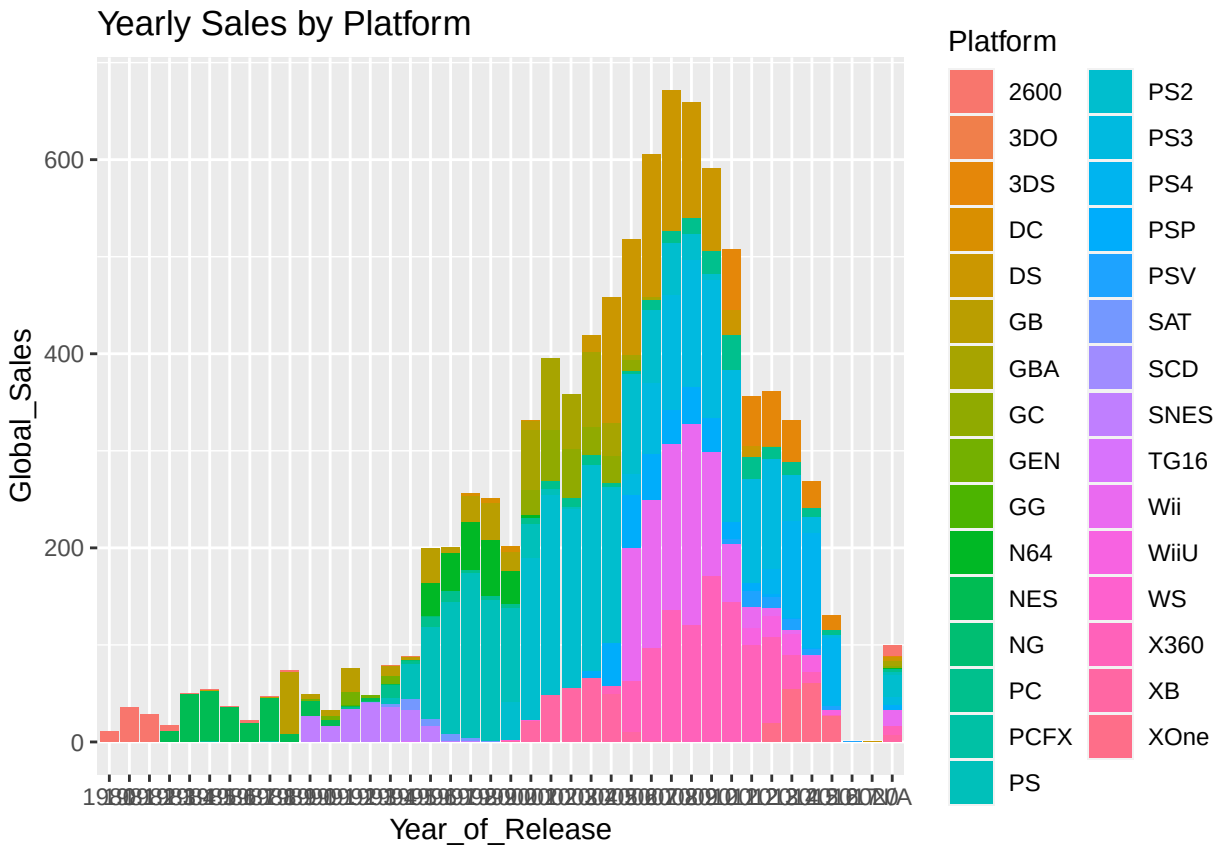
```
## # A tibble: 40 x 4
##    Year_of_Release TotalSales AvgSales SdSales
##    <chr>                <dbl>    <dbl>   <dbl>
##  1 2008                 672.    0.471    1.29
##  2 2009                 659.    0.462    1.55
##  3 2007                 605.    0.506    1.22
##  4 2010                 591.    0.471    1.29
##  5 2006                 518.    0.515    3.04
##  6 2011                 508.    0.447    1.14
##  7 2005                 458.    0.488    1.56
##  8 2004                 419.    0.550    1.26
##  9 2002                 396.    0.477    1.08
```

```
## 10 2013                   361.    0.664    1.66
## # ... with 30 more rows
```

The year with the most total sales is 2008.

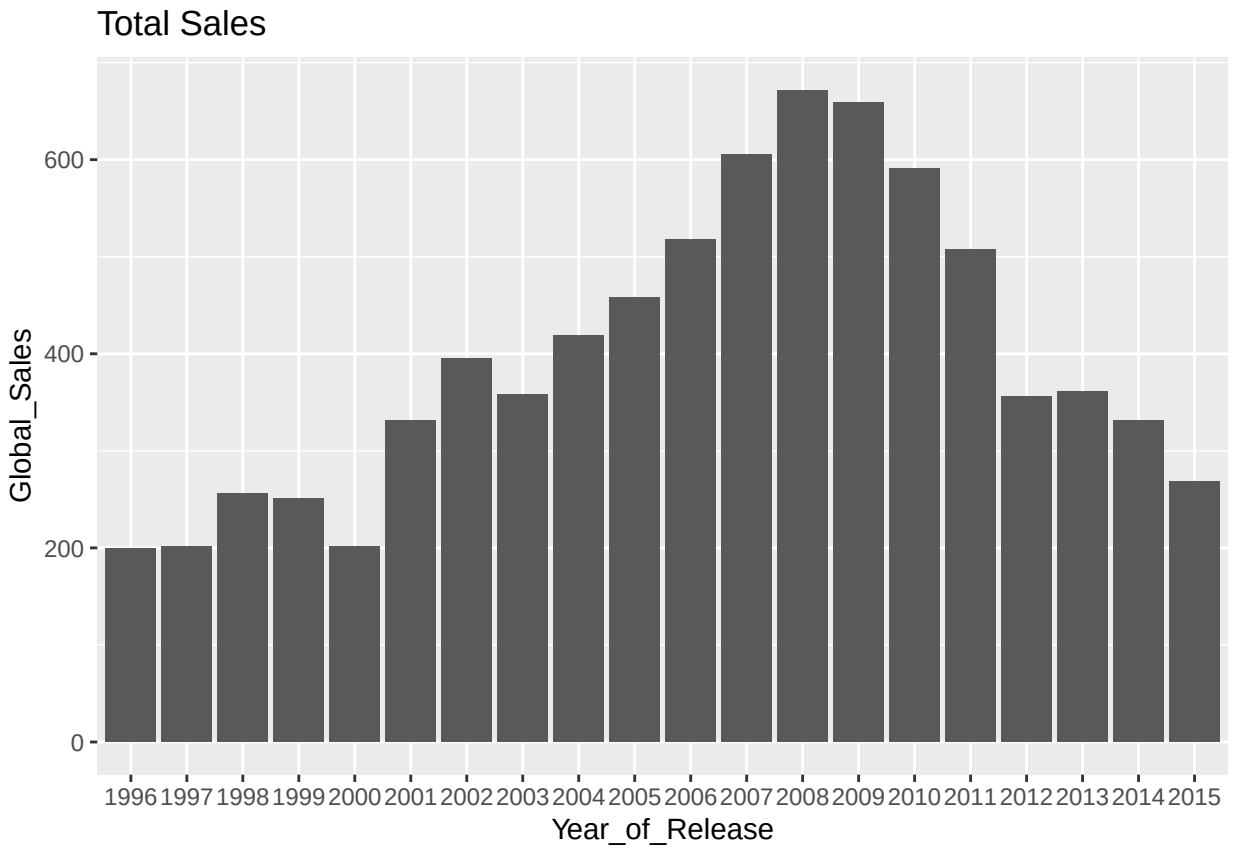
We can also break up the sales per year by genre & platform (there are too many publishers to chart by this characteristic)





From the prior analysis by year it can be seen that sales data prior to 1996 is highly statistically different to future years. Also data appears incomplete post 2016. We now filter the dataset to years between these values.

```
#Sales data prior to 1996 is highly statistically different to future years. Also data appears incomple
gamesales <- gamesales %>%
  filter(Year_of_Release >= 1996) %>%
  filter(Year_of_Release < 2016)
```



Also Wii Sports can be seen as an extreme outlier relative to other sales. We remove this title as it is unlikely to support predictive modelling.

```
# Also Wii Sports can be seen as extreme outlier relative to other sales. Removing this title as it as
gamesales <- gamesales %>%
    filter(Name != "Wii Sports")
```

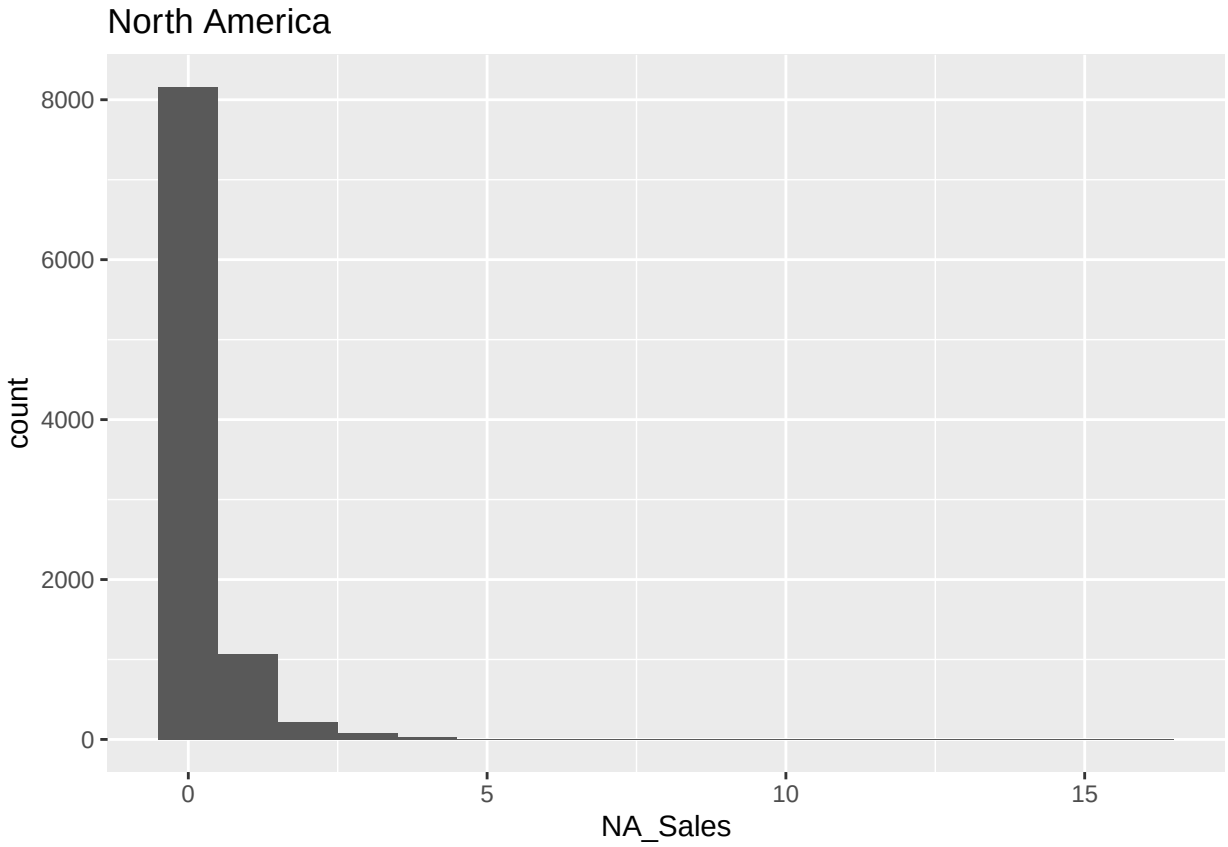
Another observation is that a large number of titles are missing a Developer. Filtering out titles that do not have a developer such that this characteristic can also be used in modelling.

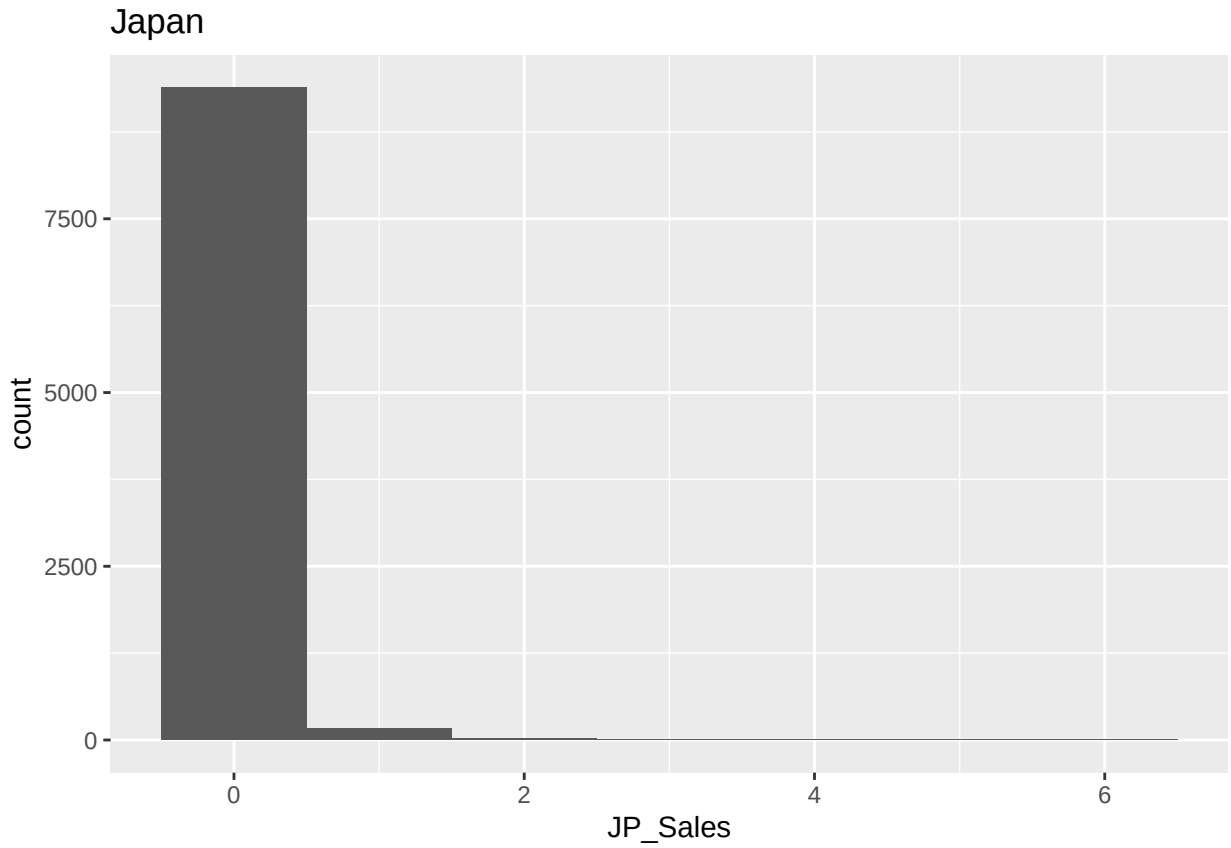
```
# Filter out missing developers
gamesales <- gamesales %>%
  filter(Developer != "")
head(gamesales)
```

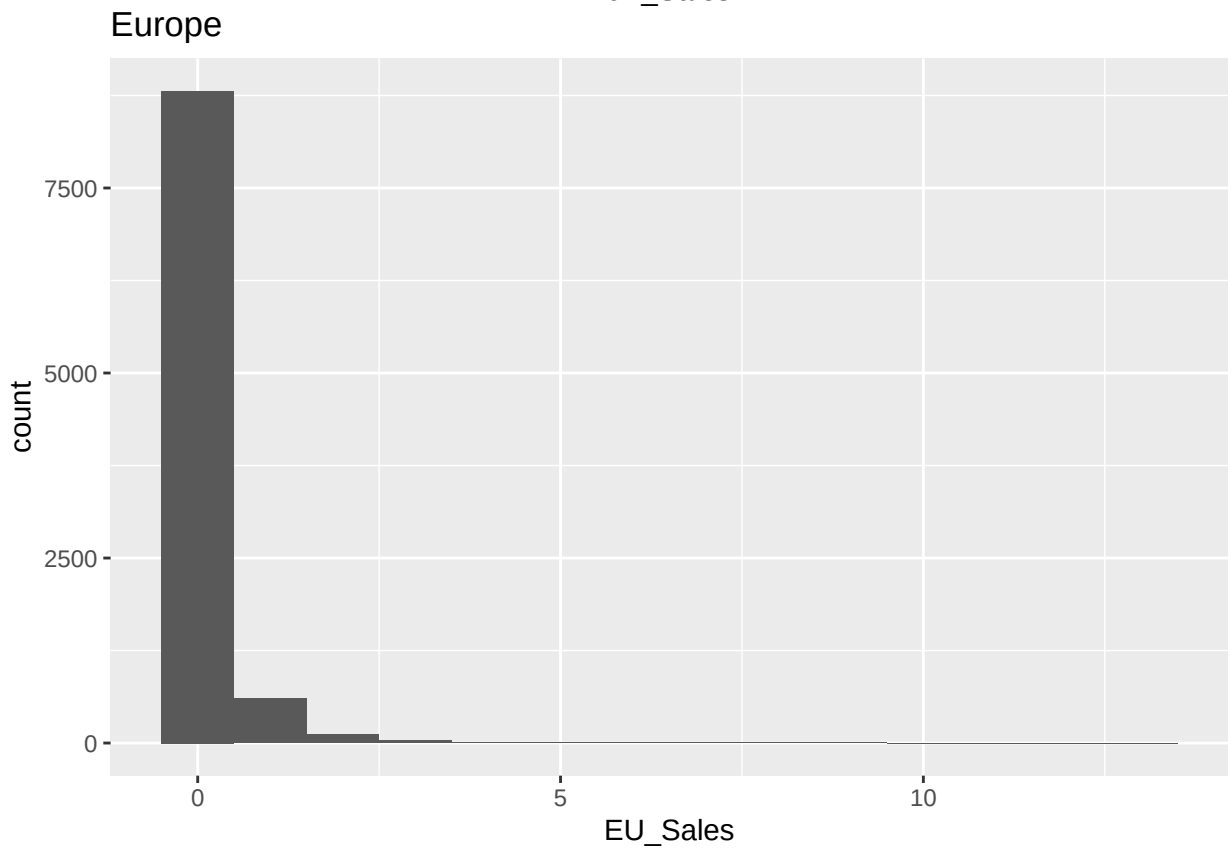
```
##                       Name Platform Year_of_Release    Genre Publisher
## 1            Mario Kart Wii      Wii            2008   Racing  Nintendo
## 2         Wii Sports Resort      Wii            2009   Sports  Nintendo
## 3     New Super Mario Bros.       DS            2006 Platform  Nintendo
## 4                  Wii Play      Wii            2006     Misc  Nintendo
## 5 New Super Mario Bros. Wii      Wii            2009 Platform  Nintendo
## 6             Mario Kart DS       DS            2005   Racing  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
## 1    15.68    12.76     3.79        3.29        35.52           82           73
## 2    15.61    10.93     3.28        2.95        32.77           80           73
## 3    11.28     9.14     6.50        2.88        29.80           89           65
## 4    13.96     9.18     2.93        2.84        28.92           58           41
## 5    14.44     6.94     4.70        2.24        28.32           87           80
## 6     9.71     7.47     4.13        1.90        23.21           91           64
```

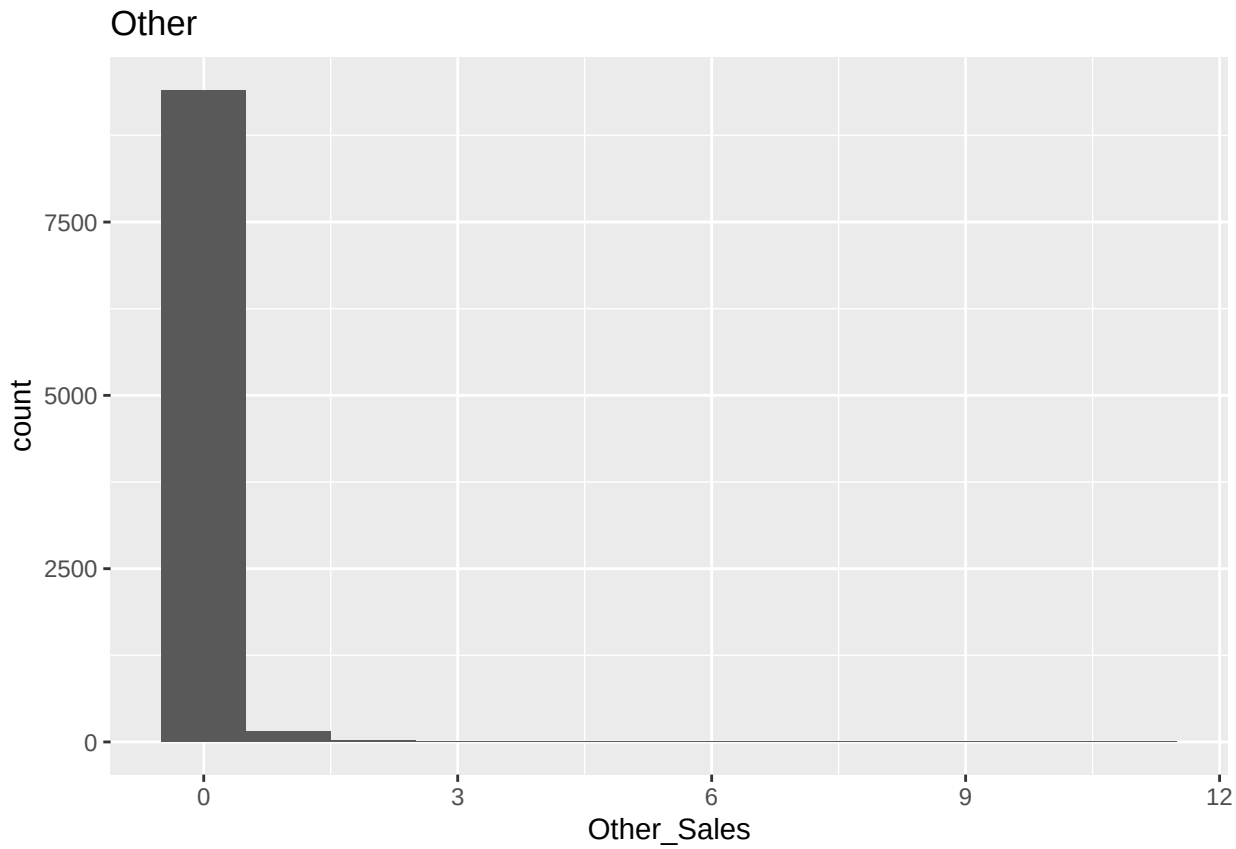
```
##   User_Score User_Count Developer Rating
## 1        8.3        709  Nintendo      E
## 2          8        192  Nintendo      E
## 3        8.5        431  Nintendo      E
## 4        6.6        129  Nintendo      E
## 5        8.4        594  Nintendo      E
## 6        8.6        464  Nintendo      E
```

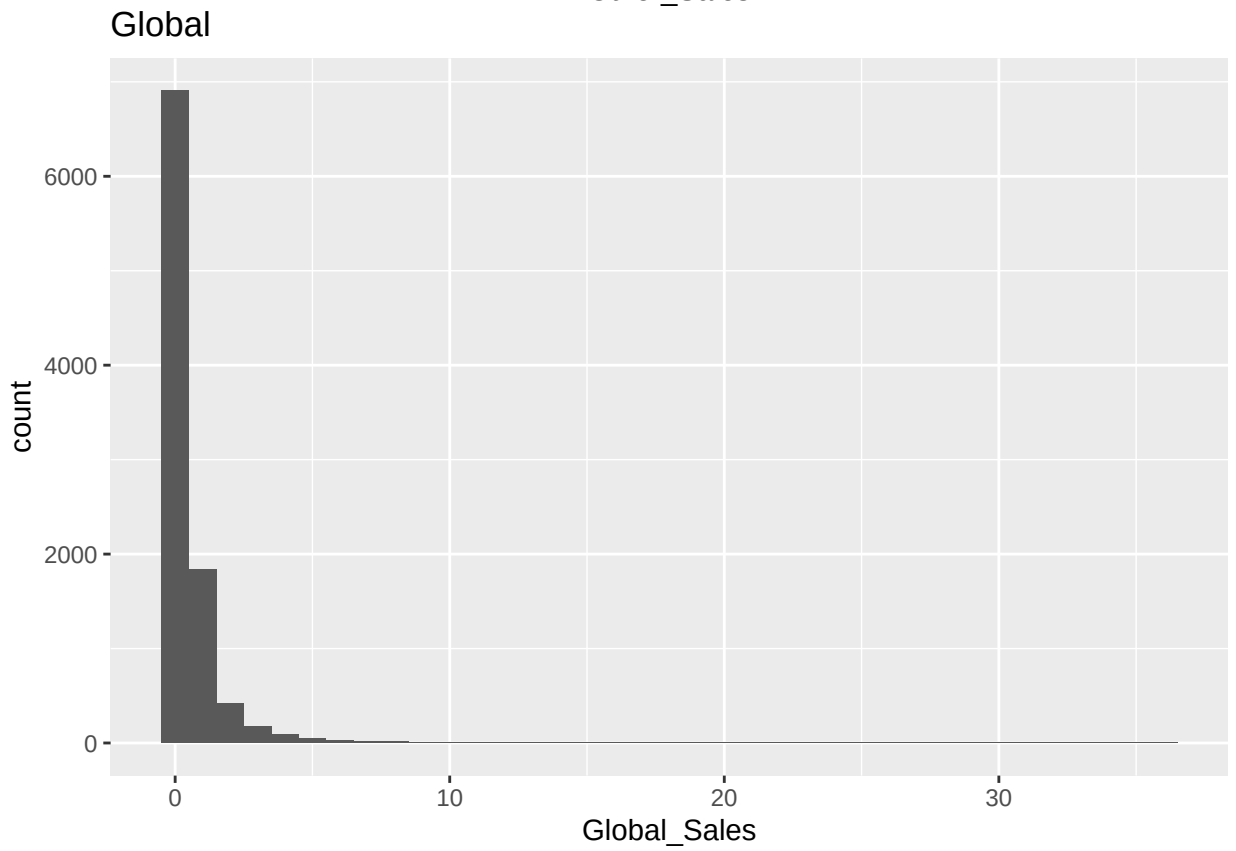
Next we consider the distribution of sales by region and globally.




Japan
7500
5000
2500
0
count
0
2
4
6
JP_Sales


Europe
7500
5000
2500
0
count
0
5
10
EU_Sales

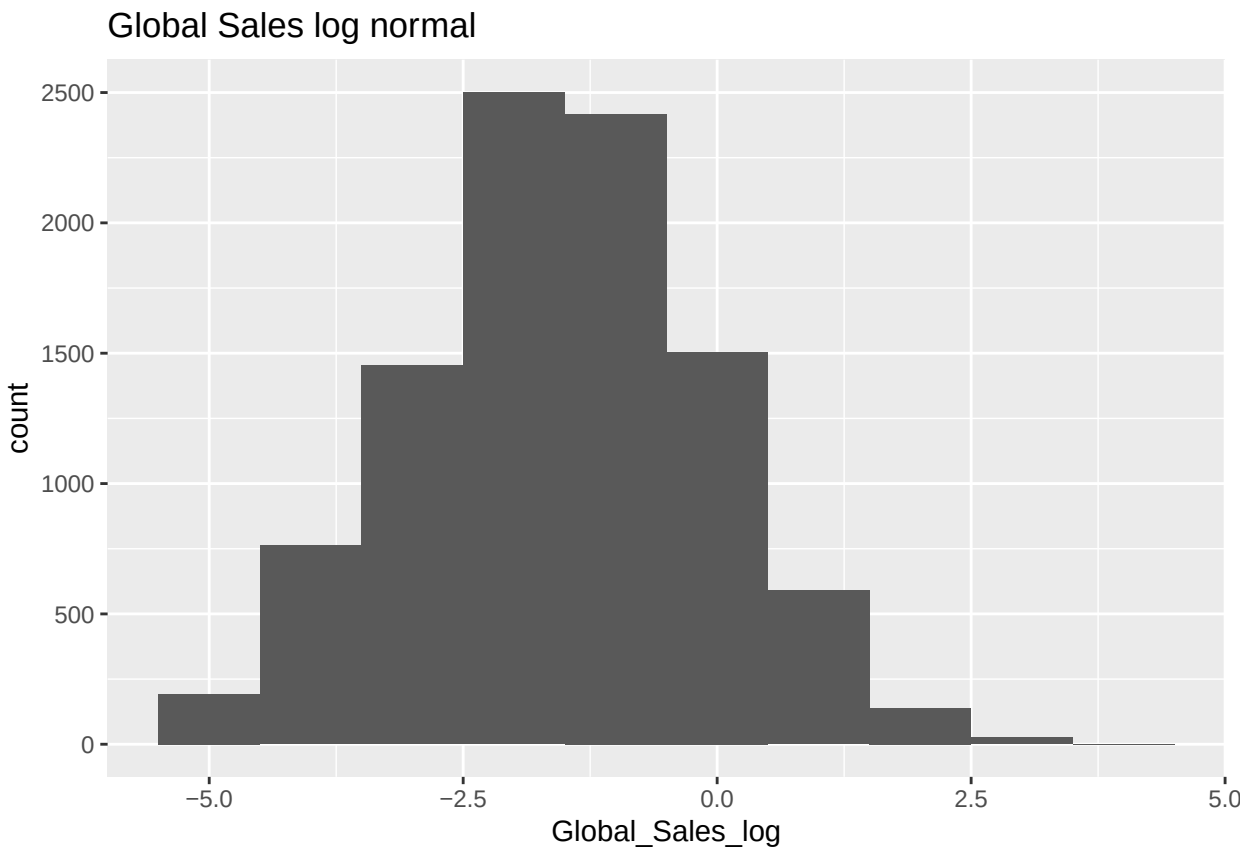




Distribution of sales data does not appear to be normal with a very small proportion of games making up the

bulk of the sales. Adding a variable to the dataset converting Global Sales to log normal for future regression analysis.

```
# Converting sales to log to create normal distribution assumption
gamesales <- gamesales %>%
  mutate(Global_Sales_log = log(Global_Sales))
gamesales %>%
    ggplot(aes(Global_Sales_log)) + geom_histogram(binwidth = 1) + ggtitle("Global Sales log normal")
```



In future analysis we will consider Random Forest as a logistic prediction model. Creating ranged buckets with even observations of global sales to perform this analysis.

```
# Add categorical bins of global for classification prediction models
gamesales <- gamesales %>%
  mutate(Global_Sales_range = discretize(Global_Sales, method = "frequency", breaks = 10))
head(gamesales)
```

```
##                       Name Platform Year_of_Release    Genre Publisher
## 1            Mario Kart Wii      Wii            2008   Racing  Nintendo
## 2         Wii Sports Resort      Wii            2009   Sports  Nintendo
## 3     New Super Mario Bros.       DS            2006 Platform  Nintendo
## 4                  Wii Play      Wii            2006     Misc  Nintendo
## 5 New Super Mario Bros. Wii      Wii            2009 Platform  Nintendo
## 6             Mario Kart DS       DS            2005   Racing  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
## 1    15.68    12.76     3.79        3.29        35.52           82           73
## 2    15.61    10.93     3.28        2.95        32.77           80           73
## 3    11.28     9.14     6.50        2.88        29.80           89           65
## 4    13.96     9.18     2.93        2.84        28.92           58           41
```

```
## 5    14.44     6.94     4.70        2.24        28.32           87           80
## 6     9.71     7.47     4.13        1.90        23.21           91           64
##   User_Score User_Count Developer Rating Global_Sales_log Global_Sales_range
## 1        8.3        709  Nintendo      E         3.570096        [1.36,35.5]
## 2          8        192  Nintendo      E         3.489513        [1.36,35.5]
## 3        8.5        431  Nintendo      E         3.394508        [1.36,35.5]
## 4        6.6        129  Nintendo      E         3.364533        [1.36,35.5]
## 5        8.4        594  Nintendo      E         3.343568        [1.36,35.5]
## 6        8.6        464  Nintendo      E         3.144583        [1.36,35.5]
```

```
table(gamesales$Global_Sales_range)
```

```
##
## [0.01,0.04) [0.04,0.07)  [0.07,0.1)  [0.1,0.15) [0.15,0.21) [0.21,0.31)
##         956         907         796        1095         916        1032
## [0.31,0.46) [0.46,0.72) [0.72,1.36) [1.36,35.5]
##         967         972         982         965
```

We can now create train and tests on the game sales data including removing missing genres, platforms, publishers and developers from both sets.

```
#create train & test sets for data
set.seed(1,sample.kind="Rounding")
test_index <- createDataPartition(y = gamesales$Global_Sales_log, times = 1, p = 0.2, list = FALSE)
train_set <- gamesales[-test_index,]
test_set <- gamesales[test_index,]

test_set <- test_set %>%
  semi_join(train_set, by = "Genre") %>%
  semi_join(train_set, by = "Platform") %>%
  semi_join(train_set, by = "Publisher") %>%
  semi_join(train_set, by = "Developer")
```

The first set of analysis will consider an algorithm to predict expected global sales based on effects including genre, platform, publisher & developer. Results will be assessed by reducing the Root Mean Squared Error (RMSE). Start by creating a function to calculate the RMSE.

```
# Function to calculate RMSE
RMSE <- function(true_sales, predicted_sales){
  sqrt(mean((true_sales - predicted_sales)^2))
}
```

Next we'll consider a basic model assuming the average of all sales as the predicted result.

```
# Model assuming same sames for all games
# Average
mu_hat <- mean(train_set$Global_Sales)
mu_hat
```

```
## [1] 0.6025965
```

method	RMSE
Just the average	1.548368

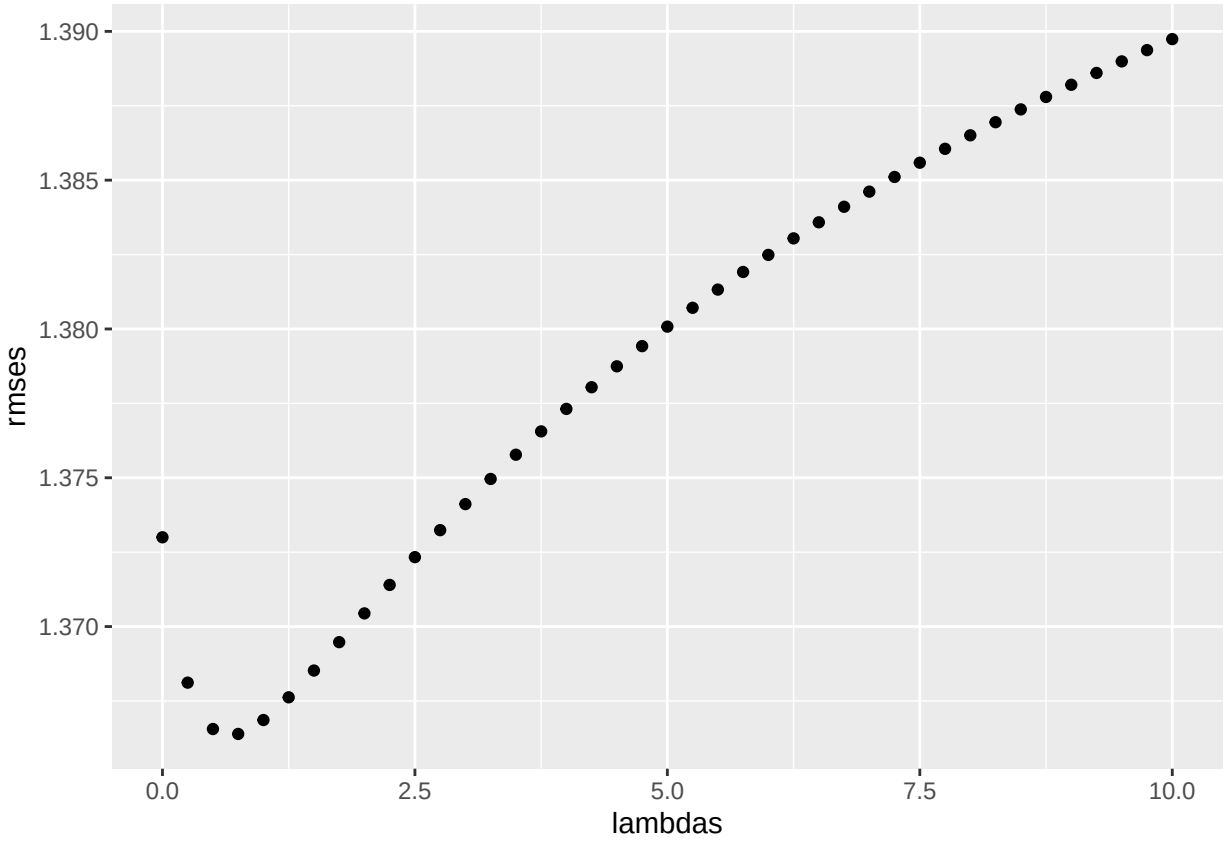
Next we look at whether the RMSE can be improved by adjusting by the average of different characteristic effects starting with Genre.

method	RMSE
Just the average	1.548368
Genre Effect Model	1.541062

Continue by incrementally adding average effects for Platform, Publisher & Developer to model

method	RMSE
Just the average	1.548368
Genre Effect Model	1.541062
Genre + Platform Effects Model	1.520924
Genre + Platform + Publisher Effects Model	1.446446
Genre + Platform + Publisher + Developer Effects Model	1.373000

The RMSE continues to improve with each effect, however, considering of starting average of 0.60 an RMSE of 1.373 does not reflect a highly predictive model. Will also consider regularization impact on low observations by optimising for the penalty terms on all effects.




```
## [1] 0.75
```

method	RMSE
Just the average	1.548368
Genre Effect Model	1.541062
Genre + Platform Effects Model	1.520924
Genre + Platform + Publisher Effects Model	1.446446

method	RMSE
Genre + Platform + Publisher + Developer Effects Model	1.373000
Regularized Genre + Platform + Publisher + Developer Effect Model	1.366391

The optimal lambda is 0.75 however there is no material improvement to the RMSE.

Will now consider random forest as a logistic prediction model of the global sales ranges determined above based on the same predictors as the previous algorithm.

```
##  Accuracy
## 0.1419576
```

The accuracy of this model is very low at 14%. Limiting the predictors to just Genre & Platfrom improves the accuracy slightly.

```
##  Accuracy
## 0.1803091
```

However, at just 18% accuracy this is still very low and it appears these models will not be useful in attempting to predict continuous sales data.

We continue with further analyses utilising critic and user scores. To undertake this analysis we will first cleanse the data by filtering out all games that are missing critic and user scores in a new dataset.

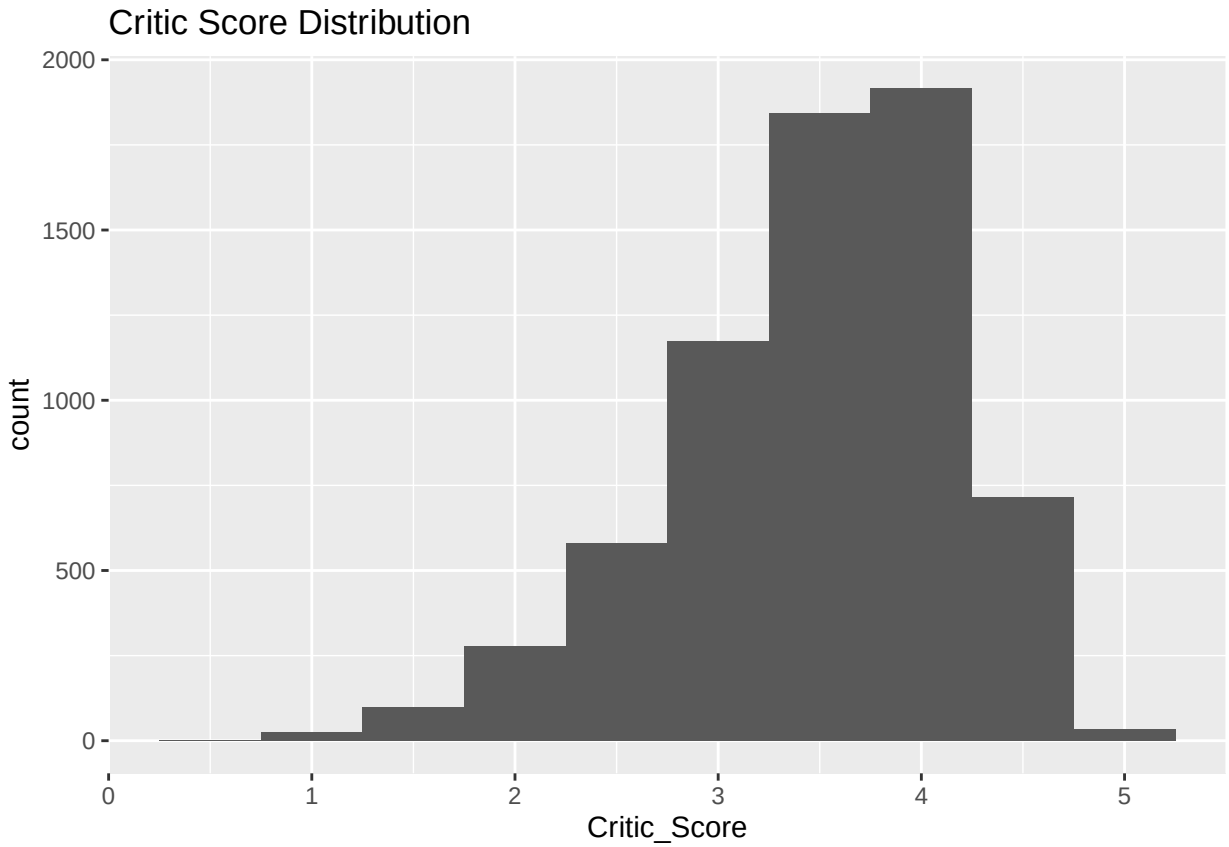
```
##                         Name Platform Year_of_Release    Genre Publisher
## 1            Mario Kart Wii      Wii            2008   Racing  Nintendo
## 2         Wii Sports Resort      Wii            2009   Sports  Nintendo
## 3     New Super Mario Bros.       DS            2006 Platform  Nintendo
## 4                  Wii Play      Wii            2006     Misc  Nintendo
## 5 New Super Mario Bros. Wii      Wii            2009 Platform  Nintendo
## 6             Mario Kart DS       DS            2005   Racing  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
## 1    15.68    12.76     3.79        3.29        35.52           82           73
## 2    15.61    10.93     3.28        2.95        32.77           80           73
## 3    11.28     9.14     6.50        2.88        29.80           89           65
## 4    13.96     9.18     2.93        2.84        28.92           58           41
## 5    14.44     6.94     4.70        2.24        28.32           87           80
## 6     9.71     7.47     4.13        1.90        23.21           91           64
##   User_Score User_Count Developer Rating Global_Sales_log Global_Sales_range
## 1        8.3        709  Nintendo      E         3.570096        [1.36,35.5]
## 2          8        192  Nintendo      E         3.489513        [1.36,35.5]
## 3        8.5        431  Nintendo      E         3.394508        [1.36,35.5]
## 4        6.6        129  Nintendo      E         3.364533        [1.36,35.5]
## 5        8.4        594  Nintendo      E         3.343568        [1.36,35.5]
## 6        8.6        464  Nintendo      E         3.144583        [1.36,35.5]
```

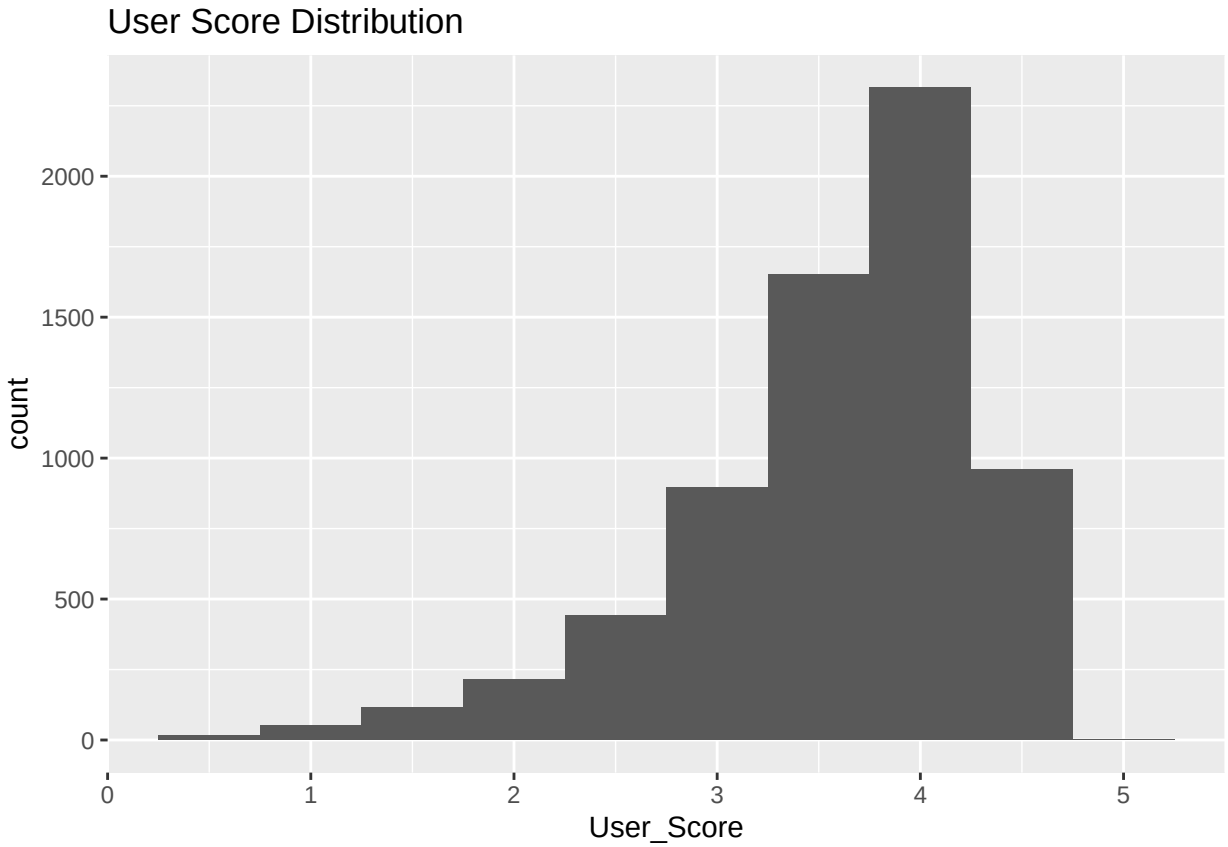
To achieve greater consistency across all the scoring data will convert character user scores to numeric and all scores to a 5 point scale.

```
##                         Name Platform Year_of_Release    Genre Publisher
## 1            Mario Kart Wii      Wii            2008   Racing  Nintendo
## 2         Wii Sports Resort      Wii            2009   Sports  Nintendo
## 3     New Super Mario Bros.       DS            2006 Platform  Nintendo
## 4                  Wii Play      Wii            2006     Misc  Nintendo
## 5 New Super Mario Bros. Wii      Wii            2009 Platform  Nintendo
## 6             Mario Kart DS       DS            2005   Racing  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
```

```
## 1    15.68    12.76     3.79        3.29        35.52        4.10           73
## 2    15.61    10.93     3.28        2.95        32.77        4.00           73
## 3    11.28     9.14     6.50        2.88        29.80        4.45           65
## 4    13.96     9.18     2.93        2.84        28.92        2.90           41
## 5    14.44     6.94     4.70        2.24        28.32        4.35           80
## 6     9.71     7.47     4.13        1.90        23.21        4.55           64
##   User_Score User_Count Developer Rating Global_Sales_log Global_Sales_range
## 1       4.15        709  Nintendo      E         3.570096        [1.36,35.5]
## 2       4.00        192  Nintendo      E         3.489513        [1.36,35.5]
## 3       4.25        431  Nintendo      E         3.394508        [1.36,35.5]
## 4       3.30        129  Nintendo      E         3.364533        [1.36,35.5]
## 5       4.20        594  Nintendo      E         3.343568        [1.36,35.5]
## 6       4.30        464  Nintendo      E         3.144583        [1.36,35.5]
```

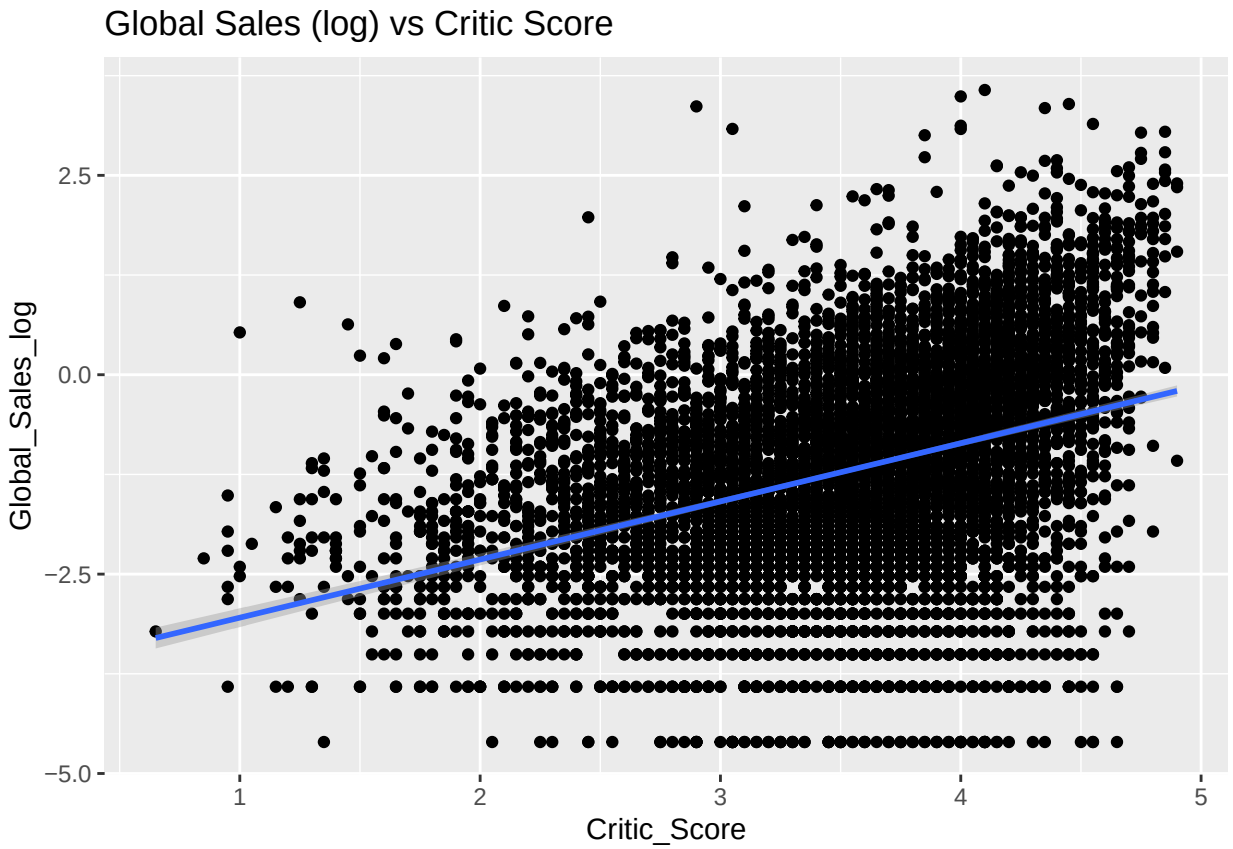
To start this next stage of analysis we will first look at the distribution of critic & user scores.

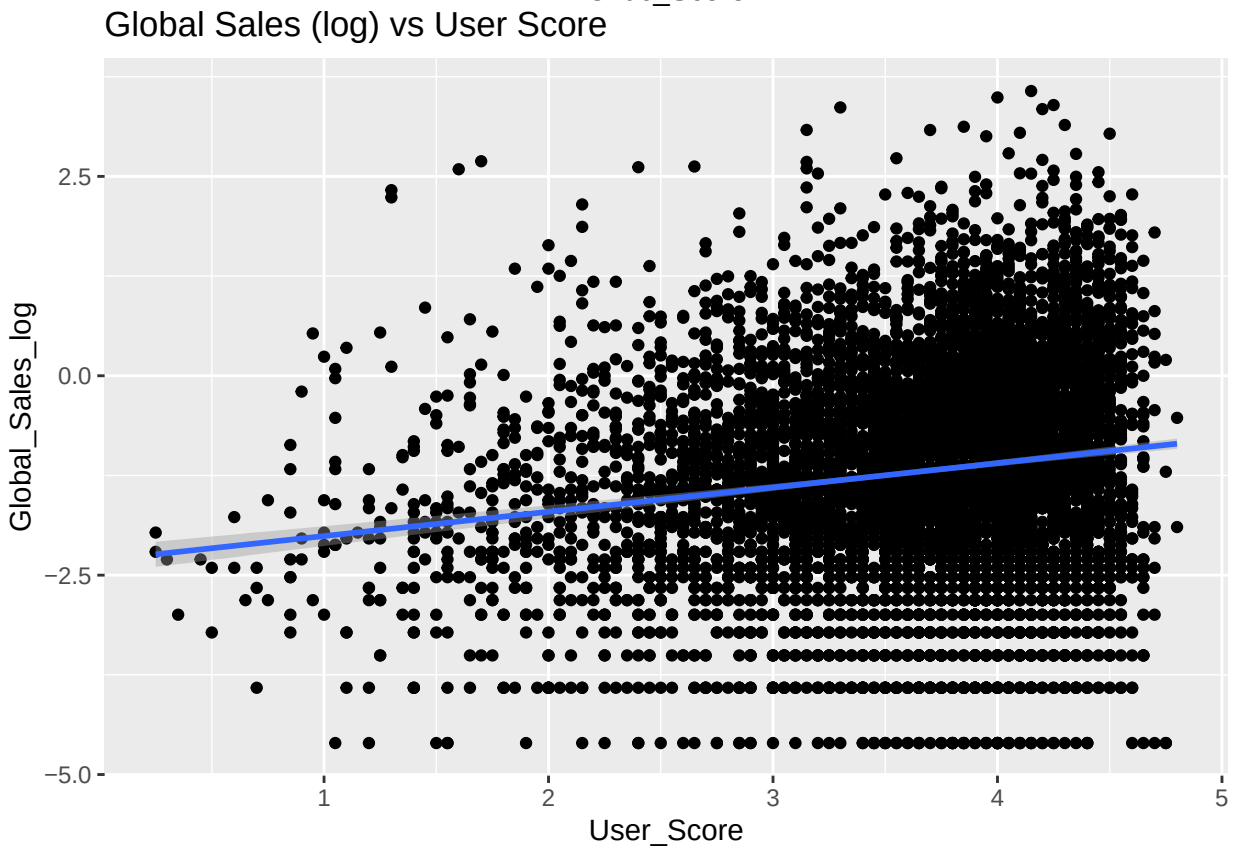


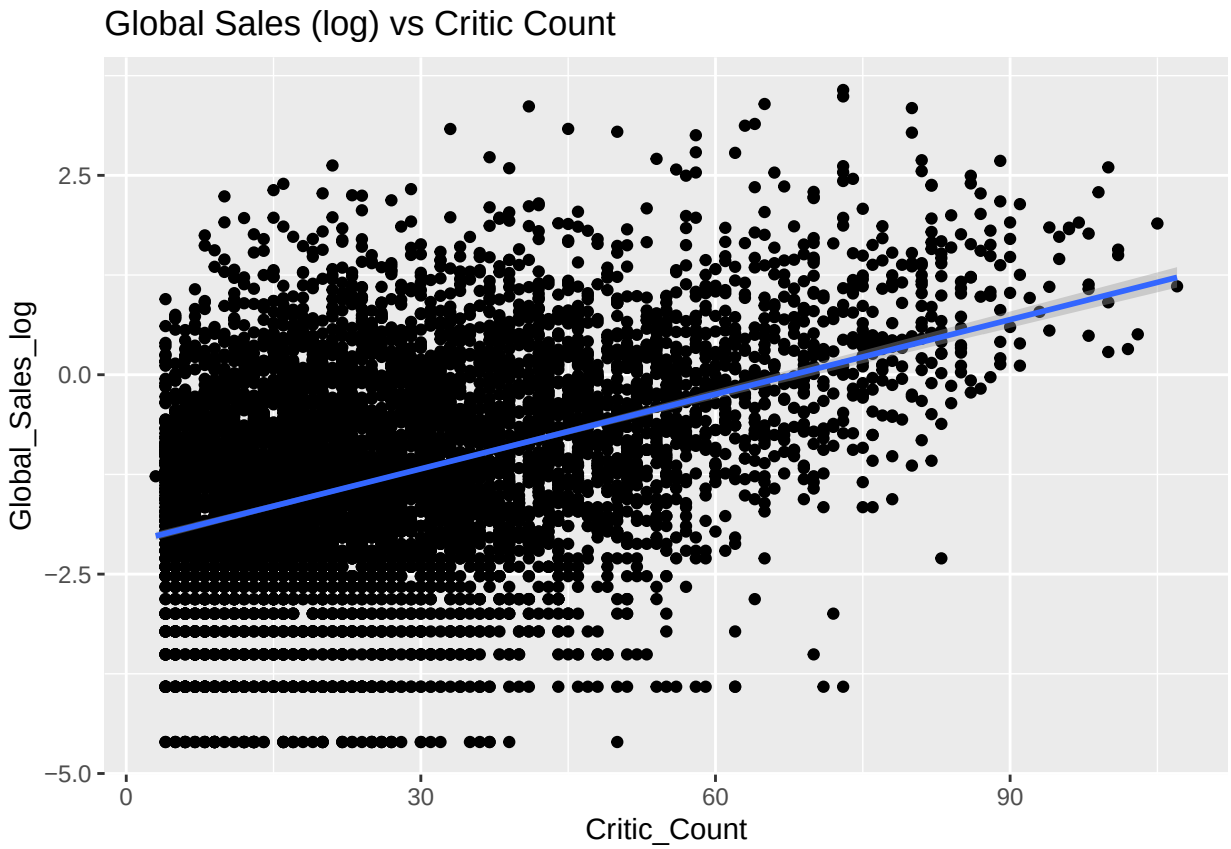


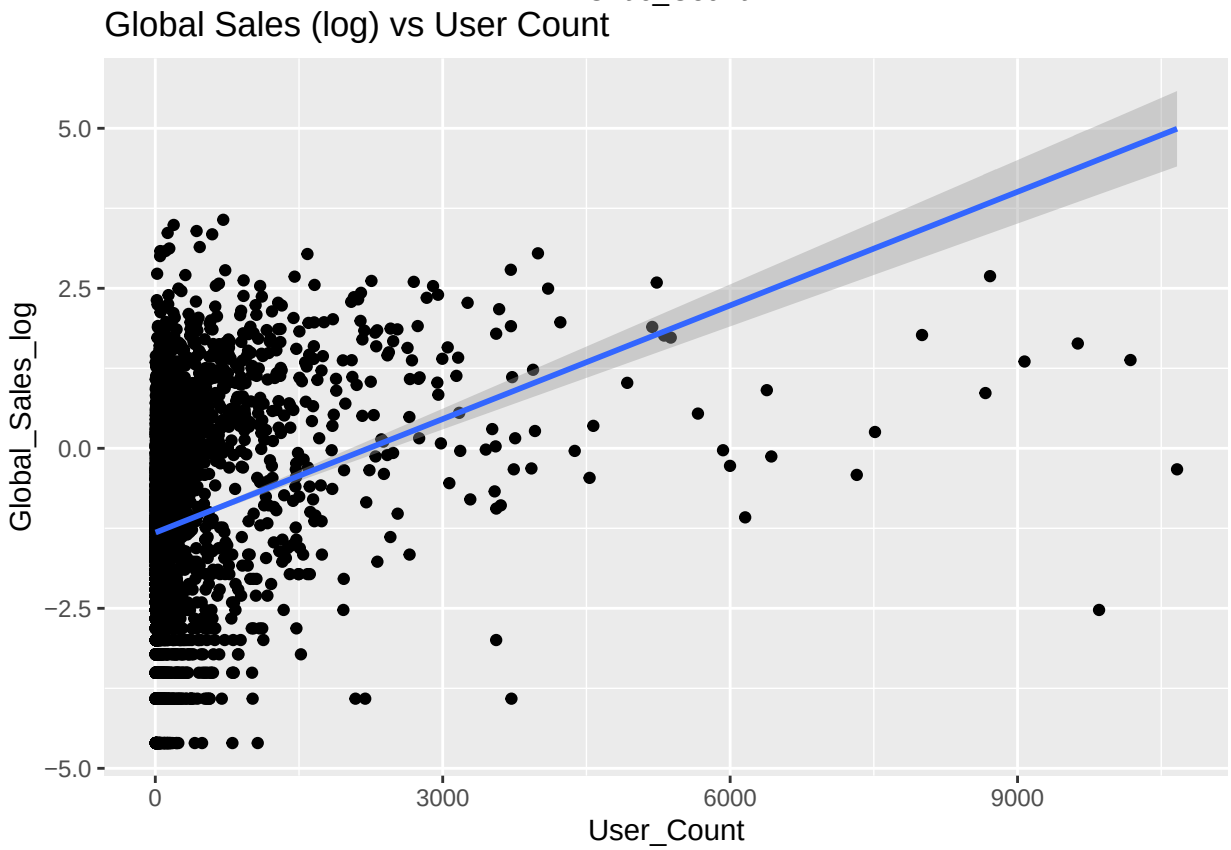
Both critic & user scores appear to be normally distributed.

Next we look at the relationship between Critic/User Scores/Count and Global Sales (log normal)


Global Sales (log) vs Critic Score
Global_Sales_log
2.5
0.0
−2.5
−5.0
1
2
3
4
5
Critic_Score


Global Sales (log) vs User Score
Global_Sales_log
2.5
0.0
−2.5
−5.0
1
2
3
4
5
User_Score





There is a clear linear relationship between scoring metrics and global sales with the relationship least strong

with User Count.

We will now create train and test sets based on the scoring cleansed dataset.

```
#create train & test sets for critic filtered data
set.seed(1,sample.kind="Rounding")
test_index <- createDataPartition(y = gamesales_critic$Global_Sales, times = 1, p = 0.2, list = FALSE)
train_critic <- gamesales_critic[-test_index,]
test_critic <- gamesales_critic[test_index,]

test_critic <- test_critic %>%
  semi_join(train_critic, by = "Genre") %>%
  semi_join(train_critic, by = "Platform") %>%
  semi_join(train_critic, by = "Publisher") %>%
  semi_join(train_critic, by = "Developer")
```

First consider mean squared loss of assuming the average of all sales followed linear regression models on each of the 4 individual scoring metrics.

```
##  (Intercept) Critic_Score
##   -3.7929673    0.7348099
```

```
##  (Intercept) Critic_Count
##  -2.12889007   0.03158828
```

```
## (Intercept)  User_Score
##  -2.3528963   0.3152178
```

```
##   (Intercept)    User_Count
## -1.3158467081  0.0005773653
```

method	SL
Just the average	1.962568
Critic Scores	1.694551
Critic Count	1.642008
User Score	1.919023
User Count	1.805586

Of the individual scoring metrics the best predictor of global sales appears to be the number of critics who reviewed the game.

Next we consider linear models with multiple predictors.

```
##  (Intercept) Critic_Score   User_Score
##   -3.5532965    0.8354372   -0.1646572
```

```
##  (Intercept) Critic_Score Critic_Count
##  -3.56609677   0.46567304   0.02485869
```

```
##   (Intercept)  Critic_Score  Critic_Count    User_Score    User_Count
## -3.3326744172  0.5209258718  0.0234912112 -0.1140081974  0.0001341032
```

method	SL
Just the average	1.962568
Critic Scores	1.694551
Critic Count	1.642008
User Score	1.919023

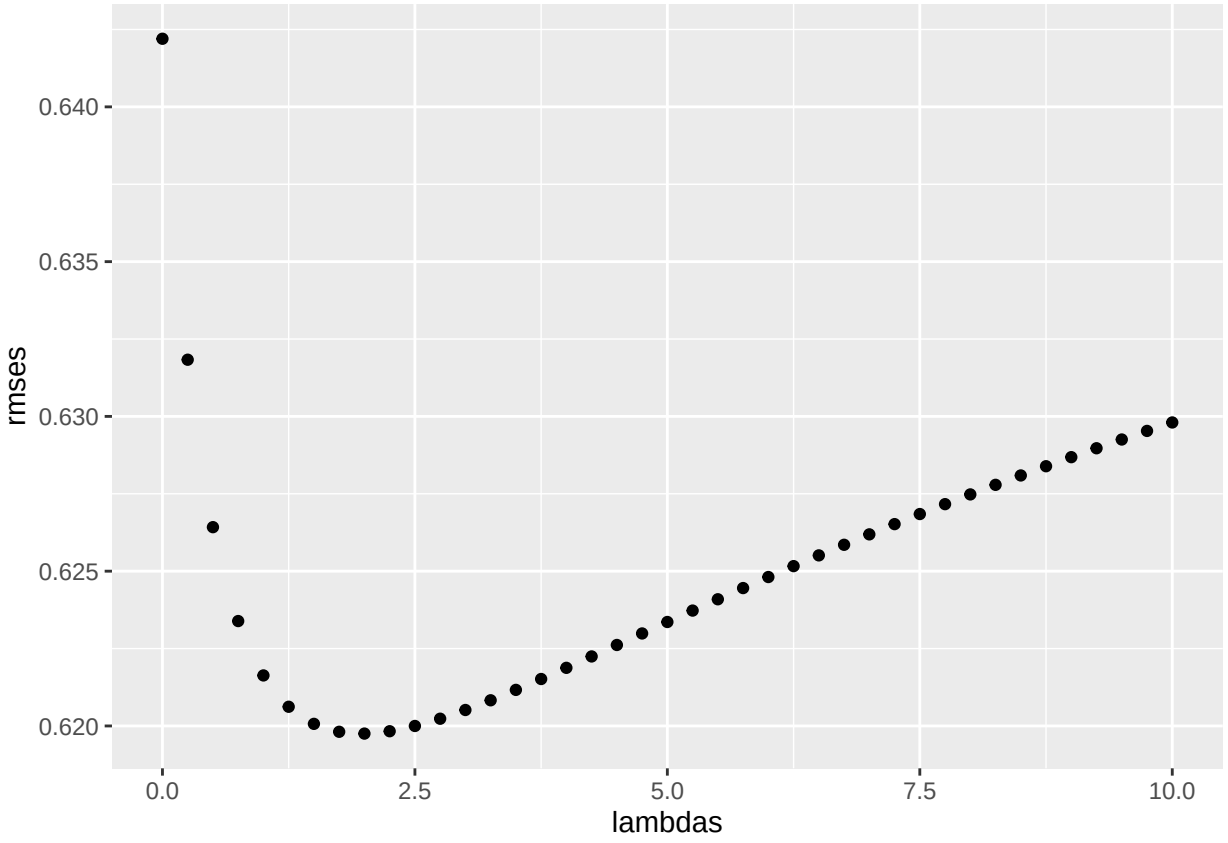
method	SL
User Count	1.805586
Critic + User Score	1.686929
Critic Score + Count	1.540933
Critic Score + User Score + Critic Count + User Count	1.516969

The lowest mean squared loss is produced by the model using all 4 predictors.

We will now return to the previous algorithmic approach using the classification categories genre, platform, publisher and distributor to predictor critic and user reception by lowering the RMSE (rather than predicting global sales).

Firstly we will repeat the modelling to predict User Scores by incrementally adding classification categories.

```
## [1] 3.596
```



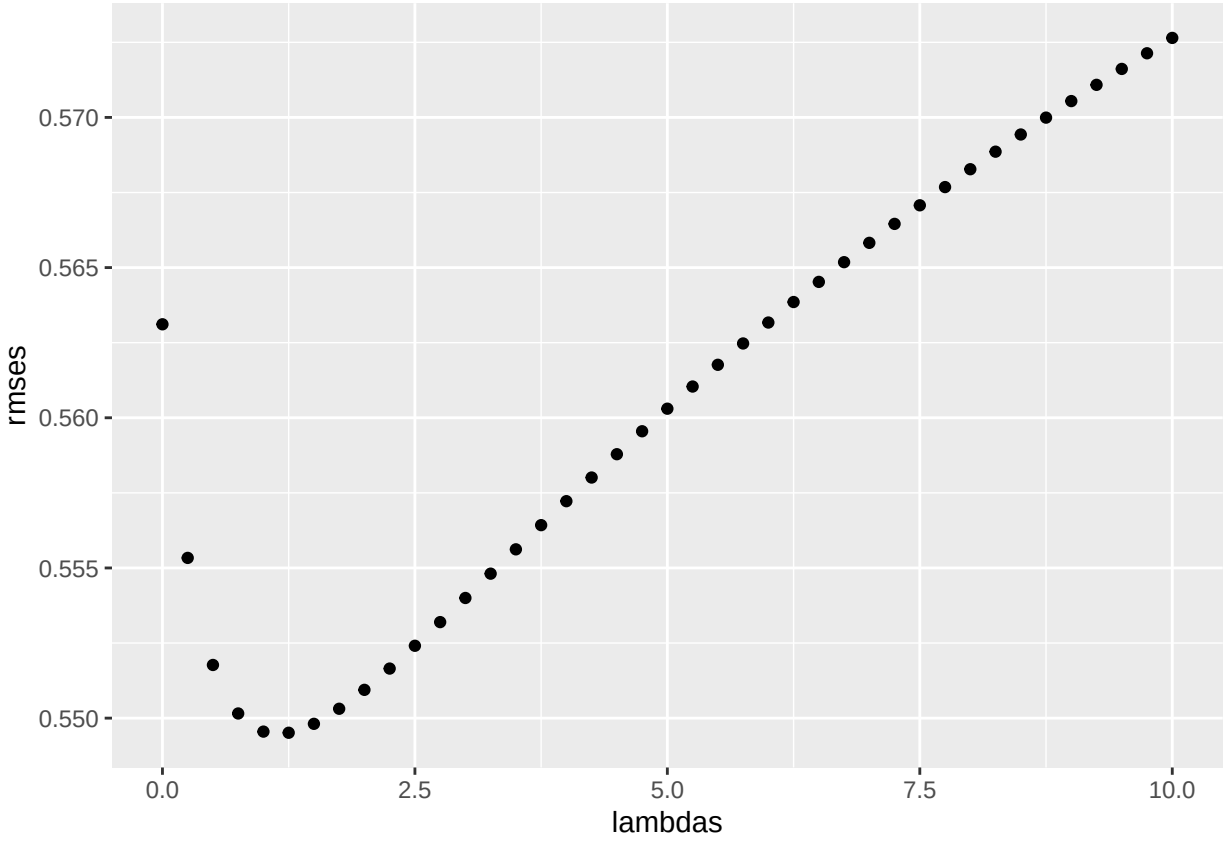

```
## [1] 2
```

method	RMSE
Just the average	0.6947961
Genre Effect Model	0.6877080
Genre + Platform Effects Model	0.6711623
Genre + Platform + Publisher Effects Model	0.6548805
Genre + Platform + Publisher + Developer Effects Model	0.6422001
Regularized Genre + Platform + Publisher + Developer Effect Model	0.6197537

The regularised model using all predictors and an optimal penalty of 2 returns an RMSE of 0.6198 which is a reasonable estimate of accuracy with overall average of 3.596.

Next we will repeat the algorithm modelling again, this time for Critic Scores.

```
## [1] 3.502394
```




```
## [1] 1.25
```

method	RMSE
Just the average	0.6737698
Genre Effect Model	0.6676705
Genre + Platform Effects Model	0.6513971
Genre + Platform + Publisher Effects Model	0.6221835
Genre + Platform + Publisher + Developer Effects Model	0.5631122
Regularized Genre + Platform + Publisher + Developer Effect Model	0.5495135

The regularised model using all predictors and an optimal penalty of 1.25 returns an RMSE of 0.5495 against an overall average of 3.502 demonstrating even further improved accuracy relative to predicting user scores. This final model for predicting critic scores also shows greater improvement over a model only assuming just the average (down from 0.6738).

Results

From the supporting analysis conducted the best predictive model identified is an aglorithmic approach using the classifications genre, platform, publisher and developer with an optimal penalty for low observations of 1.25 in order to predict critical reception (score).

The final RMSE of this model is:

```
## [1] 0.5495135
```

Conclusion

Through the analysis conducted in this report I have considered a number of potential predictive models using video game sales and scoring data. Initially focusing on using classification categories to predict global sales I did not find a suitable model. In particular using Random Forest as a logistic predictive model proved highly inaccurate.

Next I looked at the relationship between critic and user scoring to predict sales. These linear regression models provided a better fit than the previous algorithmic and logistic models to predict global sales.

Finally I reconsidered the algorithmic models but this time to predict user and critical reception. This analysis identified using genre, platform, publisher and developer to predict critic scores to be the most effective model.