Kaggle Video Games Sales Project

ongmq

2022-09-06

Section 1: Introduction

This project is part of the final course of the HarvardX Data Science Professional Certificate program. The aim is to create a machine learning algorithm to predict video game sales using the Kaggle Video Games Sales dataset.

- For each video game released, the dataset contains information on the platform (console on which the game is running), genre (game's category), ESRB ratings, and year of release.
- The dataset also contains information on the critic score (aggregate score compiled by Metacritic) and user score (score by Metacritic's subscribers), as well as the publisher and developer (partly responsible for creating the game).
- We will use these variables to predict the global sales of video games, which are measured in million of units.

Section 1.1: Loading R packages

We start by loading all the required R packages, namely the tidyverse, caret, and randomForest packages.

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

library(tidyverse)
library(caret)
library(randomForest)
```

Section 1.2: Importing & cleaning dataset

We then read the Kaggle Video Game Sales dataset which we uploaded on GitHub. The csv file contains a lot of blank, "NA", and "N/A" entries. We will use the na.strings argument to convert all those entries into na.

```
url <- "https://raw.githubusercontent.com/ongmq/harvardx-vg-sales-project/main/Video_Games_Sales_as_at_
vg_sales <- read.csv(url, na.strings = c("", " ", "NA", "N/A"))
```

We will remove all the na entries in the dataset. We now see that our dataset does not contain any na entry.

```
vg_sales <- na.omit(vg_sales)
colSums(is.na(vg_sales))
```

```
##            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
```

We will start to inspect our dataset. After removing all the na entries, we are left with 6825 observations of 16 variables.

```
str(vg_sales)
```

```
## 'data.frame':    6825 obs. of  16 variables:
##  $ Name           : chr  "Wii Sports" "Mario Kart Wii" "Wii Sports Resort" "New Super Mario Bros." .
##  $ Platform       : chr  "Wii" "Wii" "Wii" "DS" ...
##  $ Year_of_Release: int  2006 2008 2009 2006 2006 2009 2005 2007 2010 2009 ...
##  $ 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     : chr  "8" "8.3" "8" "8.5" ...
##  $ 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" ...
##  - attr(*, "na.action")= 'omit' Named int [1:9894] 2 5 6 10 11 13 19 21 22 23 ...
##   ..- attr(*, "names")= chr [1:9894] "2" "5" "6" "10" ...
```

We note that the user score variable is stored as a character variable instead of numeric. We will change its type and make it the same scale as the critic score variable.

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

We now have all the variables in the correct type.

We will continue to inspect and clean up our dataset. We see that there is only 1 entry for some rating categories. We will merge these with the other rating categories.

```
##   Rating    n
## 1     AO    1
## 2      E 2082
## 3   E10+  930
## 4    K-A    1
## 5      M 1433
## 6     RP    1
## 7      T 2377
```

```
# We will merge the "adult only" category with the "mature" category.

vg_sales <- vg_sales %>%
  mutate(Rating = ifelse(Rating == "AO", "M", Rating))

# We will merge the "kids to adults" category with the "everyone" category.

vg_sales <- vg_sales %>%
  mutate(Rating = ifelse(Rating == "K-A", "E", Rating))
```

```
# We will merge the "rating pending" category with the "everyone 10+" category.

vg_sales <- vg_sales %>%
  mutate(Rating = ifelse(Rating == "RP", "E10+", Rating))
```

We now have only 4 unique rating categories.

```
## [1] "E"    "M"    "T"    "E10+"
```

We also notice that the platform variable has many categories, and could be regrouped into 5 main ones.

```
vg_sales <- vg_sales %>%
  mutate(Platform = case_when(
    Platform %in% c("Wii", "WiiU", "DS", "3DS", "GC", "GBA") ~ "Nintendo",
    Platform %in% c("X360", "XB", "XOne") ~ "XBox",
    Platform %in% c("PS", "PS2", "PS3", "PS4", "PSP", "PSV") ~ "PS",
    Platform == "PC" ~ "PC",
    Platform == "DC" ~ "Sega"
  ))
unique(vg_sales$Platform)
```

```
## [1] "Nintendo" "XBox"     "PS"       "PC"       "Sega"
```

Here is a snapshot of our dataset after cleaning up.

```
head(vg_sales)
```

```
##                       Name Platform Year_of_Release    Genre Publisher
## 1               Wii Sports Nintendo            2006   Sports  Nintendo
## 3            Mario Kart Wii Nintendo            2008   Racing  Nintendo
## 4         Wii Sports Resort Nintendo            2009   Sports  Nintendo
## 7     New Super Mario Bros. Nintendo            2006 Platform  Nintendo
## 8                  Wii Play Nintendo            2006     Misc  Nintendo
## 9 New Super Mario Bros. Wii Nintendo            2009 Platform  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
## 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
## 7    11.28     9.14     6.50        2.88        29.80           89           65
## 8    13.96     9.18     2.93        2.84        28.92           58           41
## 9    14.44     6.94     4.70        2.24        28.32           87           80
##   User_Score User_Count Developer Rating
## 1         80        322  Nintendo      E
## 3         83        709  Nintendo      E
## 4         80        192  Nintendo      E
## 7         85        431  Nintendo      E
## 8         66        129  Nintendo      E
## 9         84        594  Nintendo      E
```

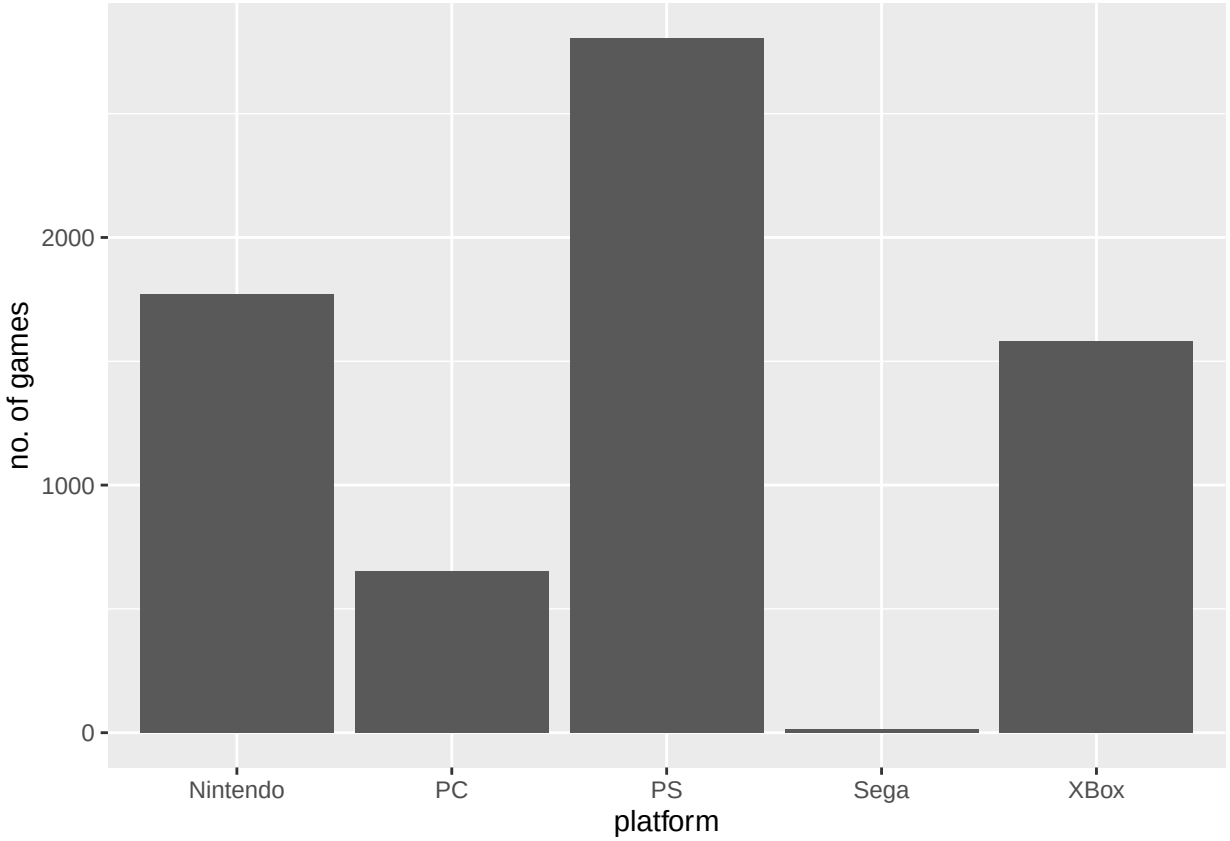
Section 2: Analysis

We will now explore the cleaned Kaggle Video Game Sales dataset.

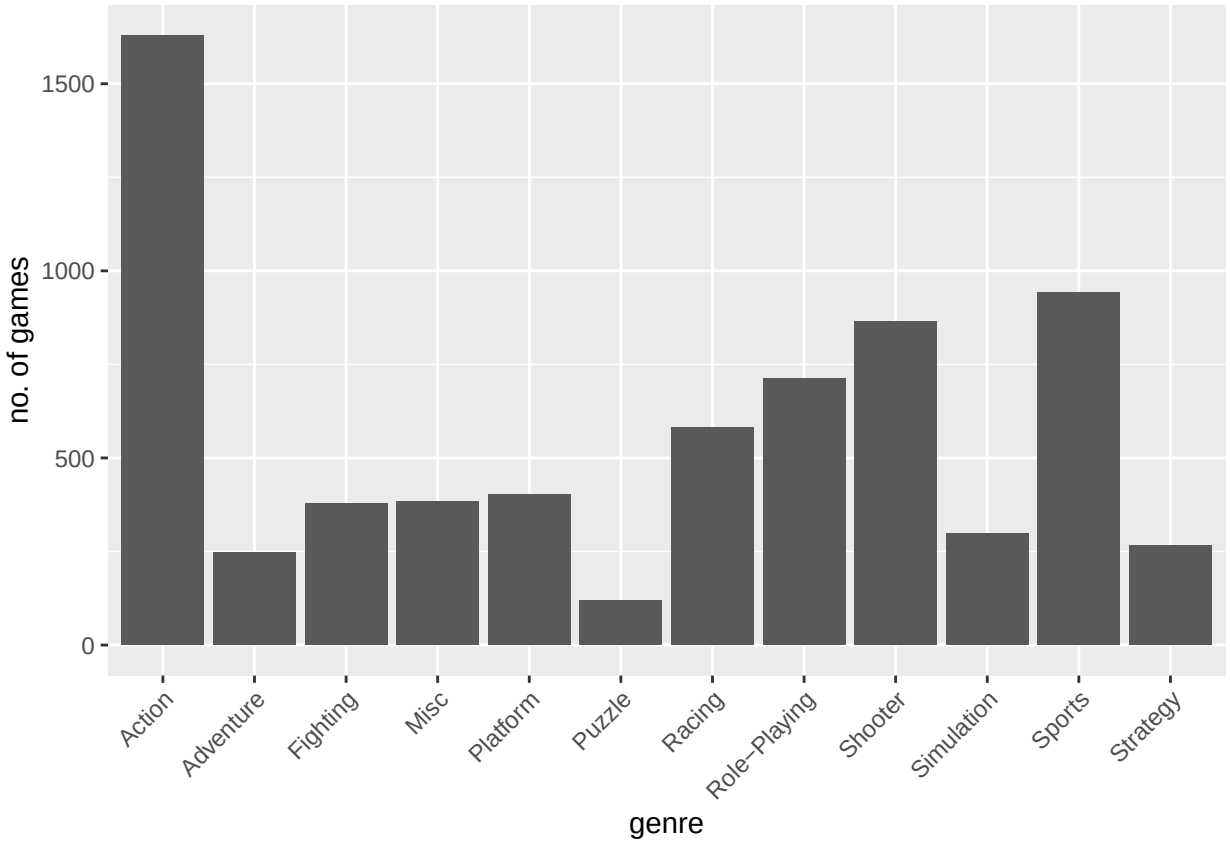
Section 2.1: Distribution of video games

We start by exploring the distribution of video games released across the key variables (i) platform, (ii) genre, (iii) rating, (iv) year of release, (v) critic score, (vi) user score, (vii) publisher, and (viii) developer.

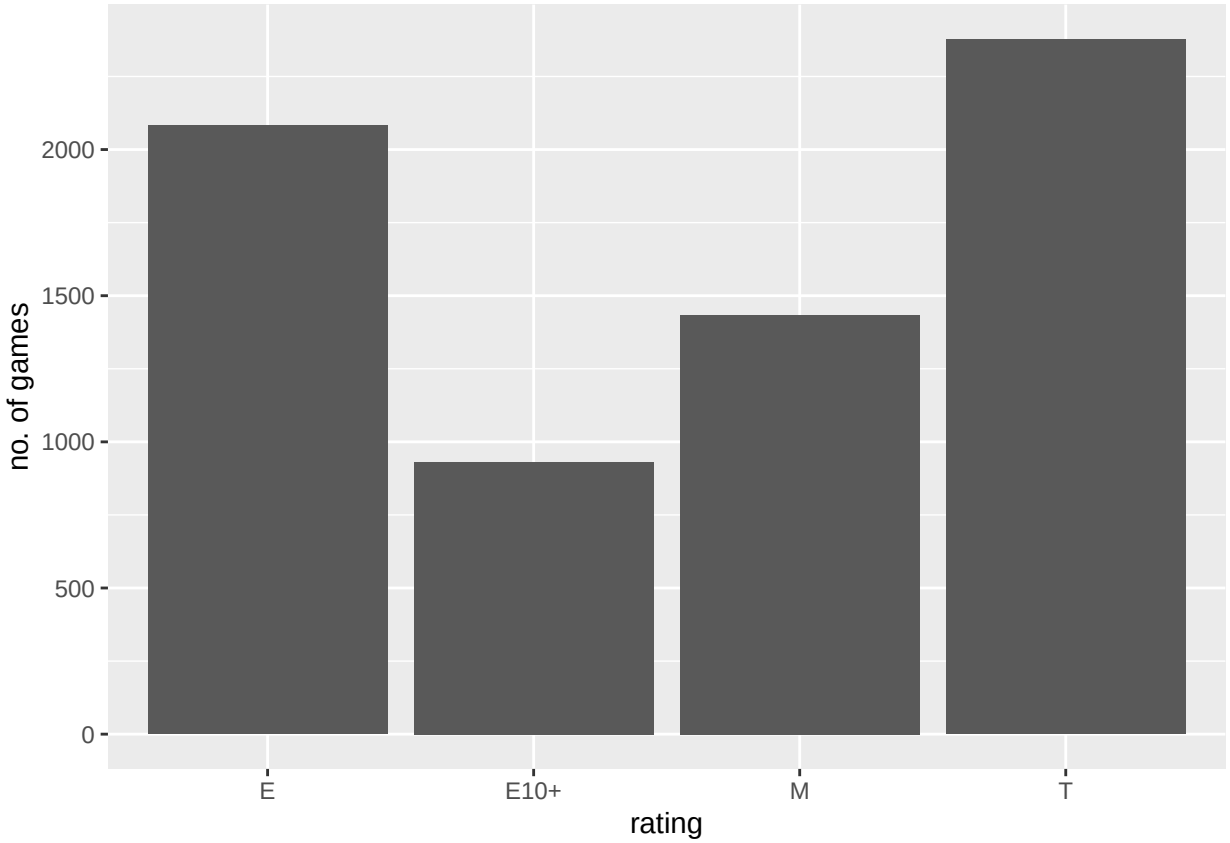
(i) Platform: Most of the games released are PS games, followed by Nintendo, XBox, PC and lastly Sega.



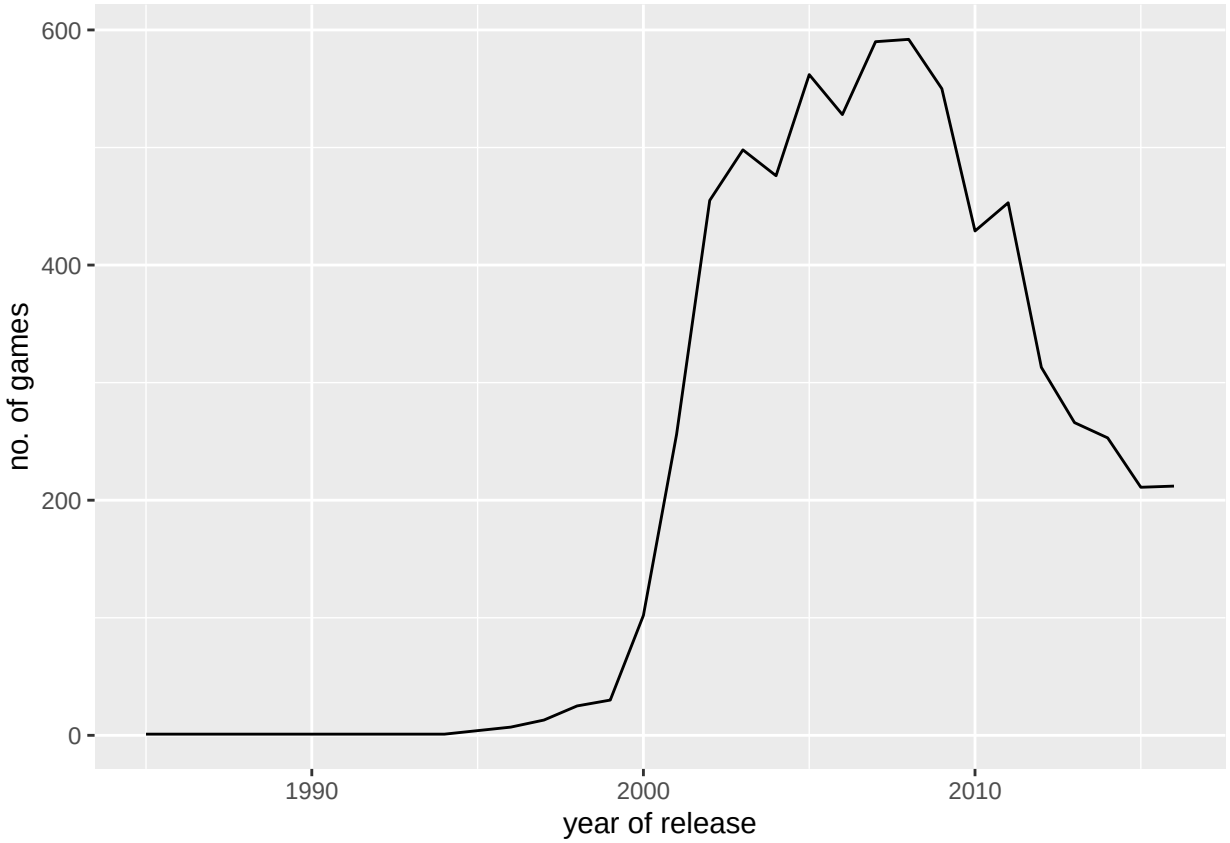
(ii) Genre: The number of games released differ greatly across genres, with action games being the most popular and puzzle games being the least popular.



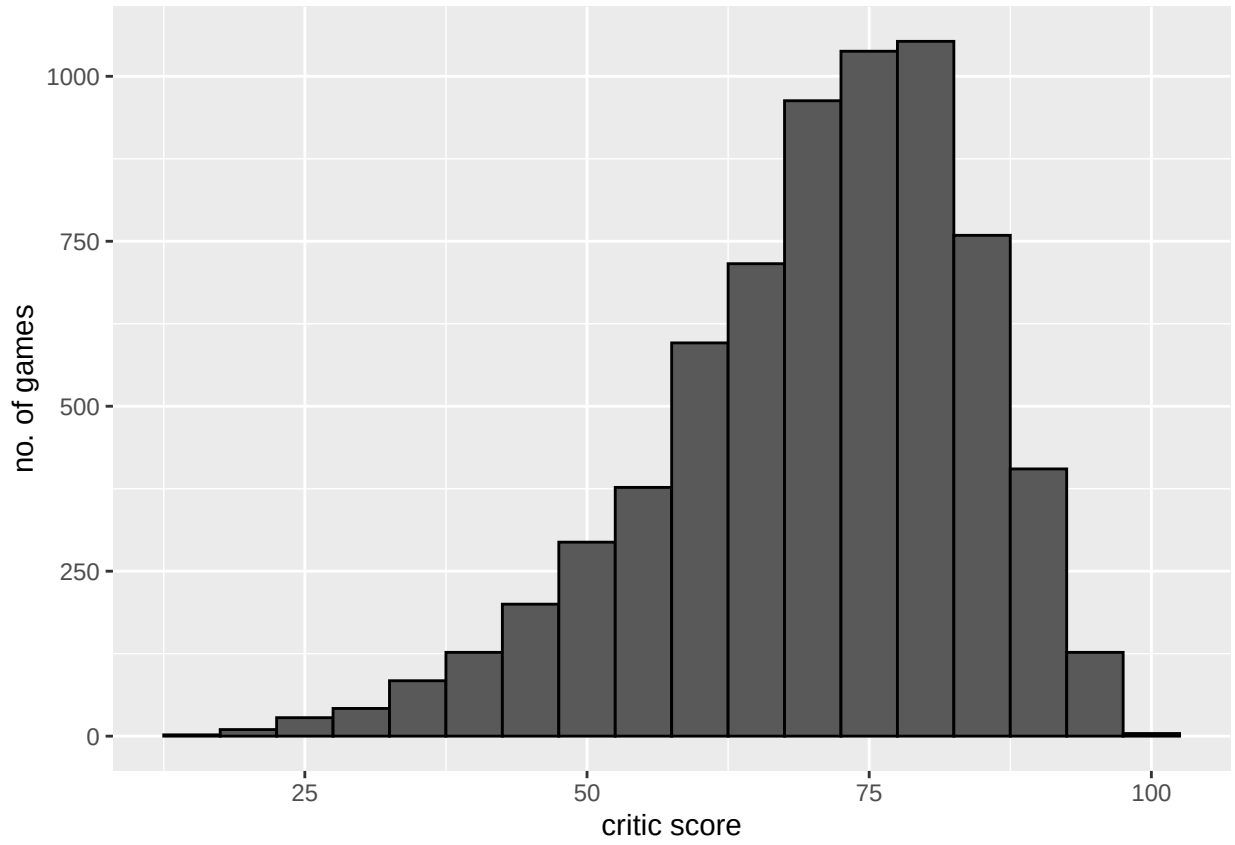
(iii) Rating: Most of the games released are rated "Teen", following by "Everyone", "Mature", and lastly "Everyone 10+".



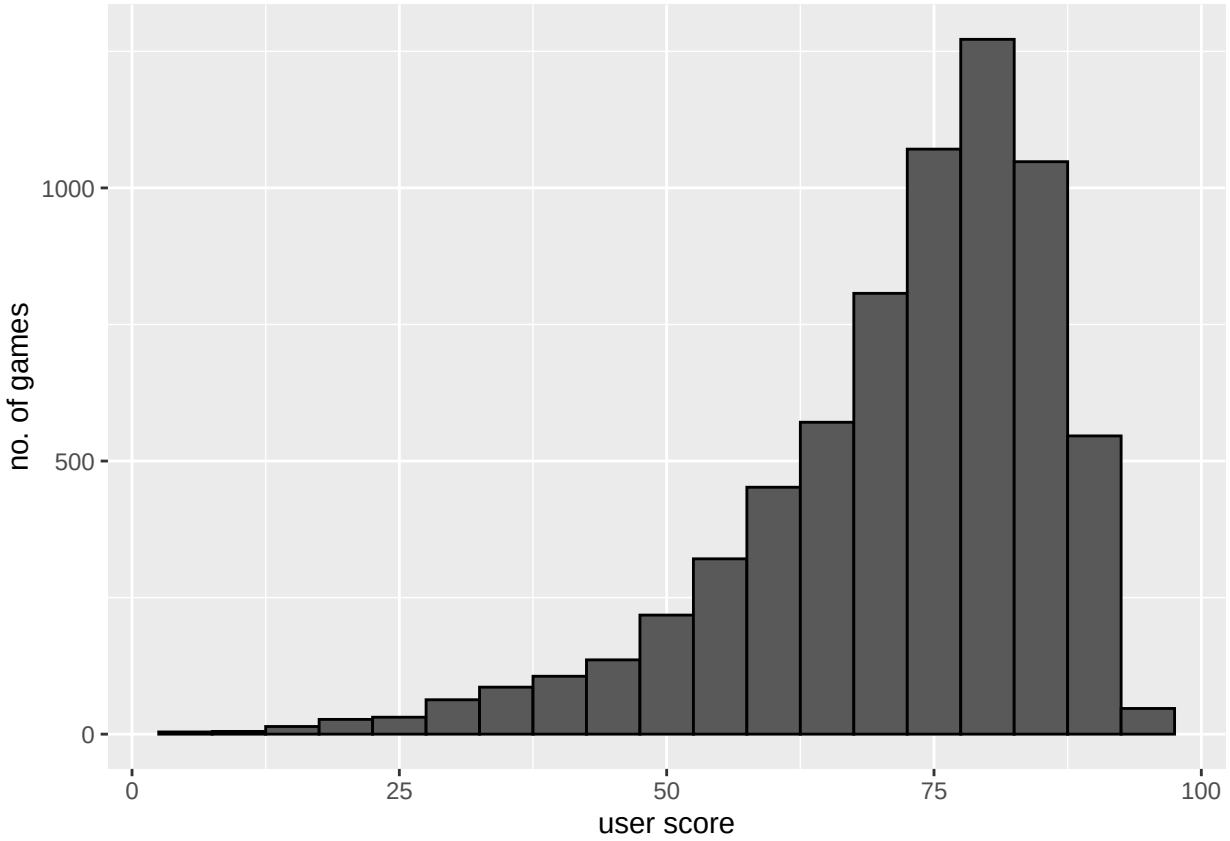
(iv) Year of release: Bulk of the games were released during 2000-2008. Beyond that, there is a steady decline in the number of games released each year.



(v) Critic score & (vi) User score: Both critic score and user score have similar distributions, with user score having a slightly higher mean and median than critic score.

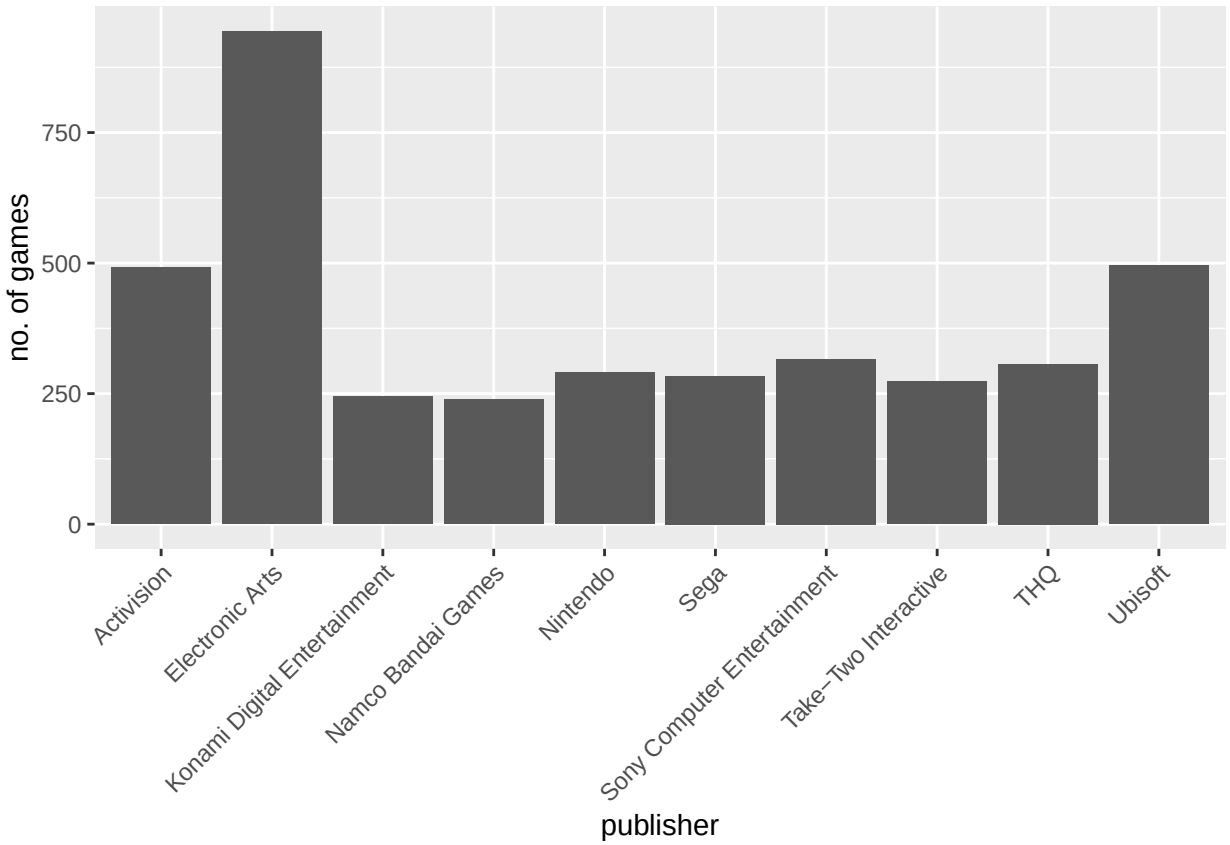


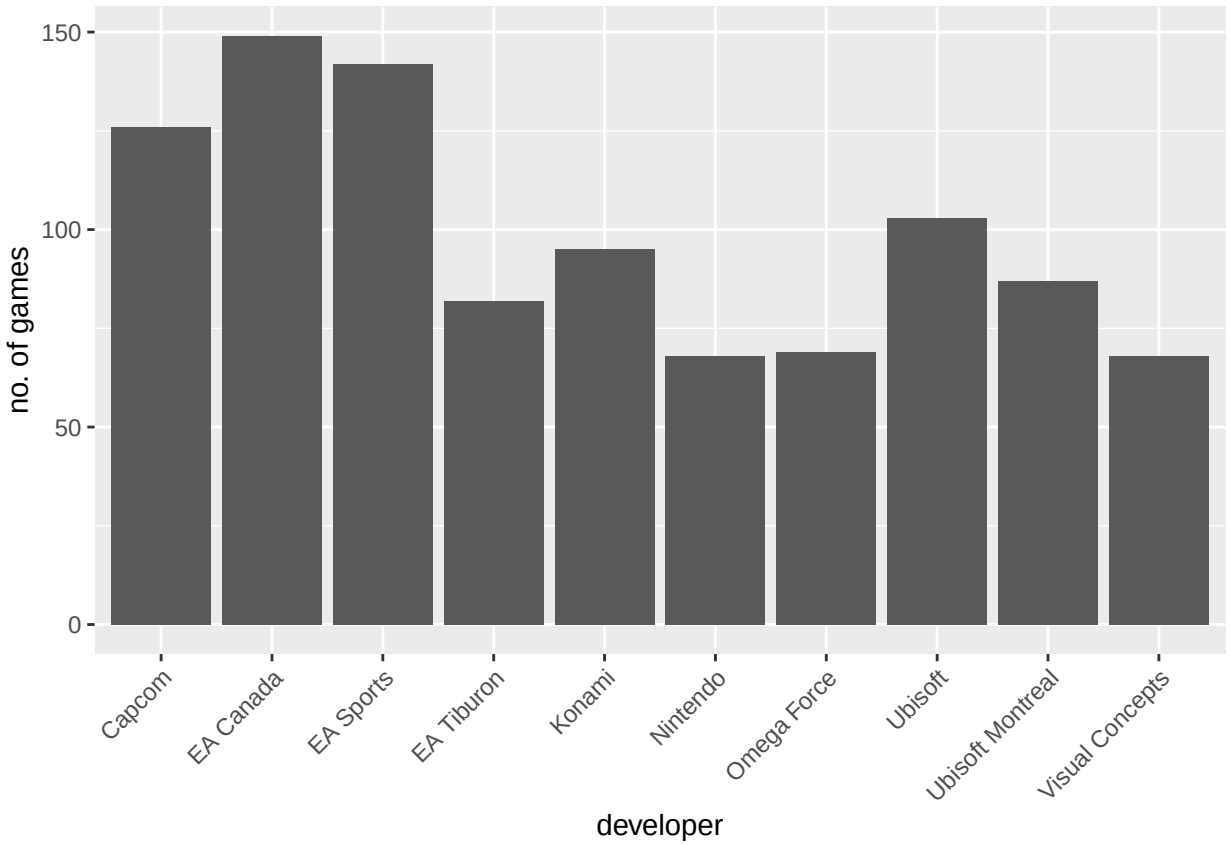

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.00   62.00   72.00   70.27   80.00   98.00
```




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   65.00   75.00   71.86   82.00   96.00
```

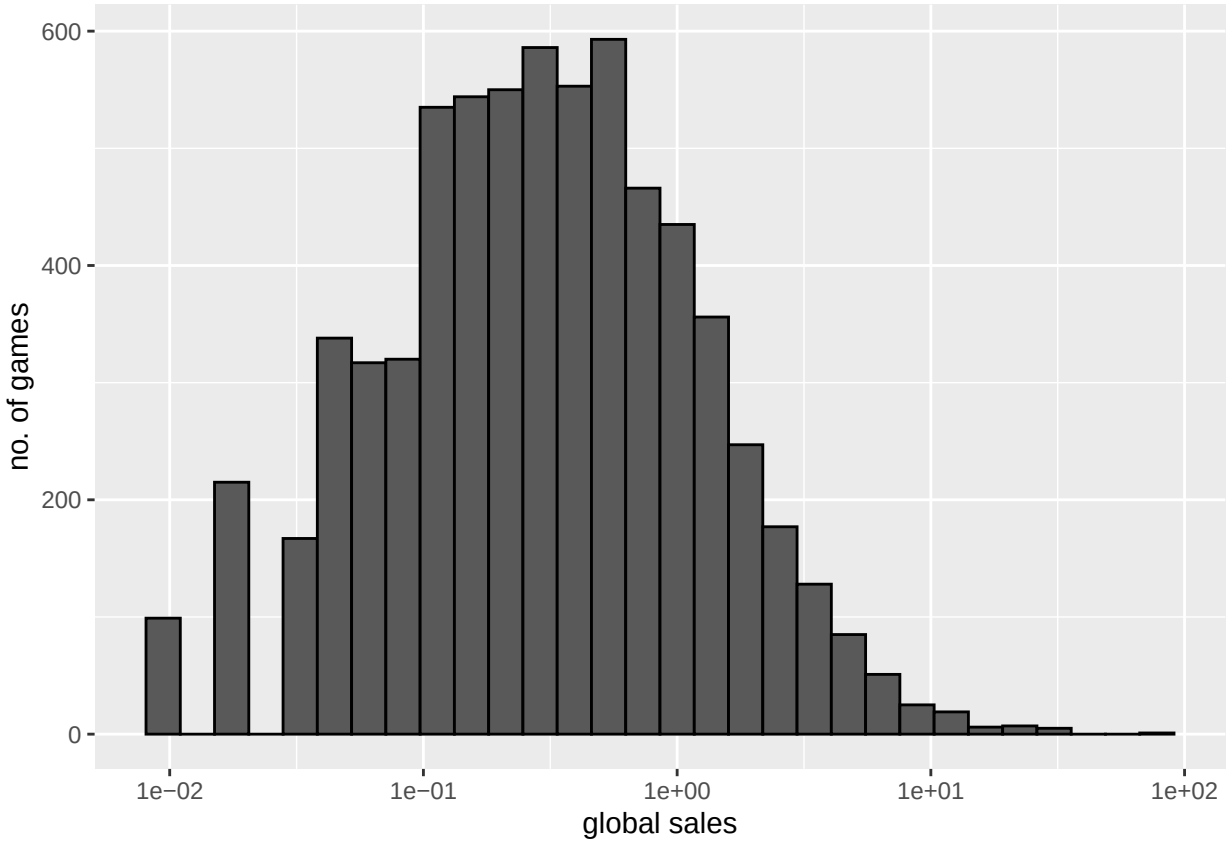
(vii) Publisher & (viii) Developer: There are many different game publishers and developers. Below are the top 10 publishers and developers based on the number of games released.


no. of games
750
500
250
0
Activision
Electronic Arts
Konami Digital Entertainment
Namco Bandai Games
Nintendo
Sega
Sony Computer Entertainment
Take−Two Interactive
THQ
Ubisoft
publisher

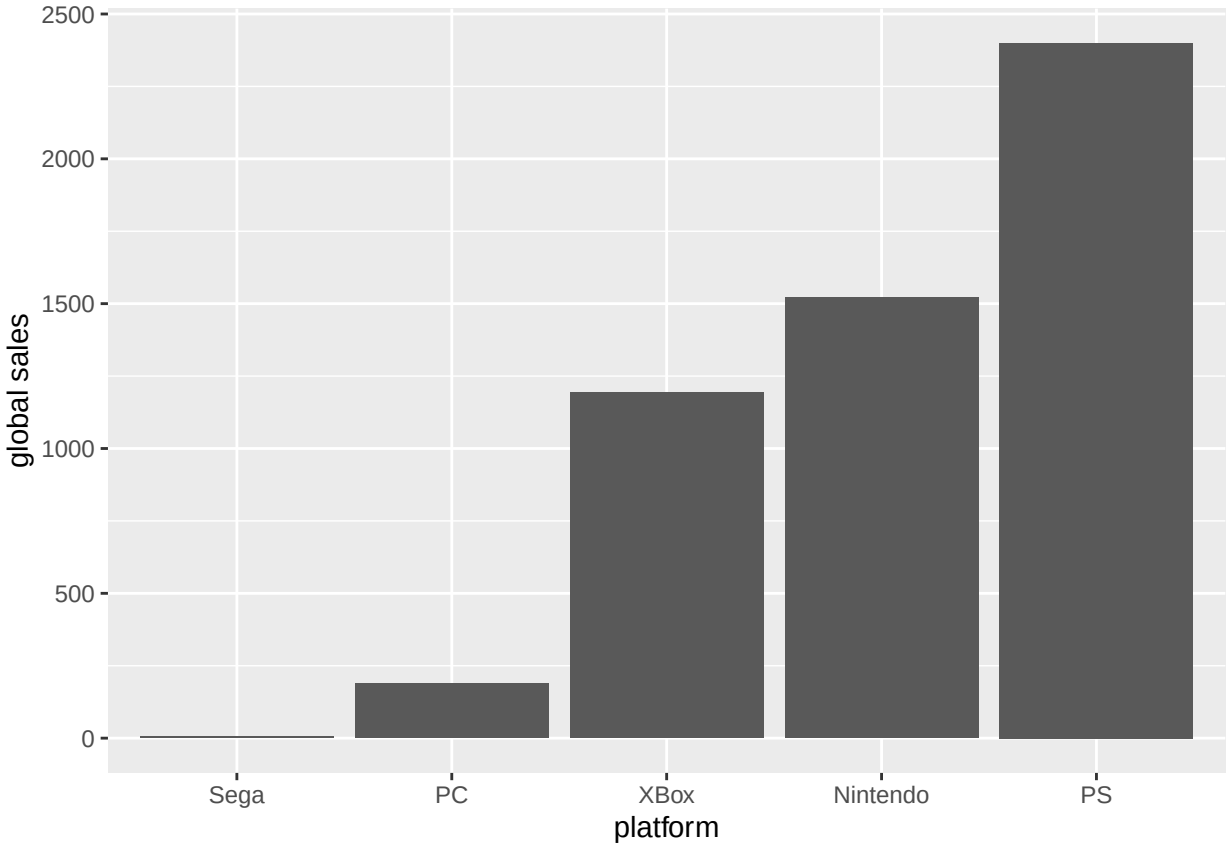


Section 2.2: Distribution of global sales

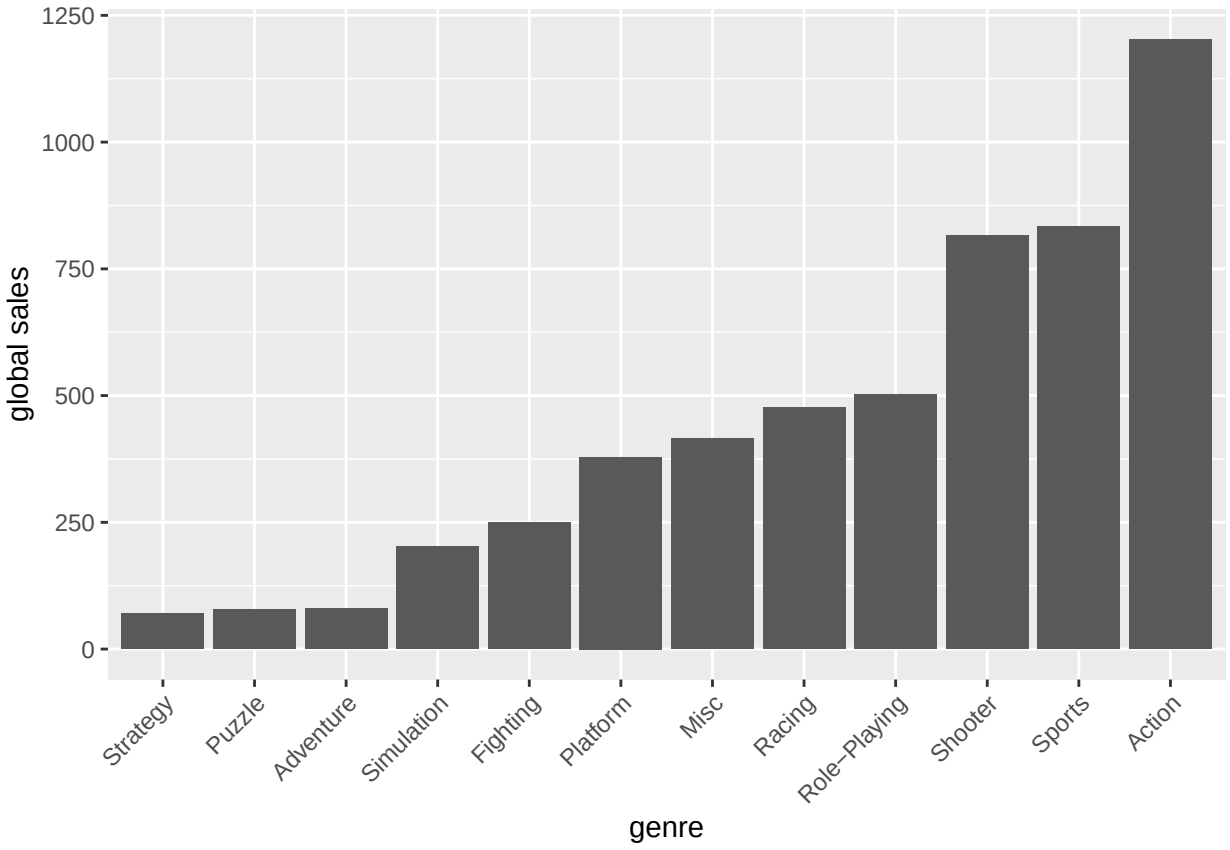
We now zoom in to global sales, which is what we want to predict using our algorithm. We can see that global sales differ significantly across games. (Note: We are using log scale for the x-axis as the variable is highly skewed.)



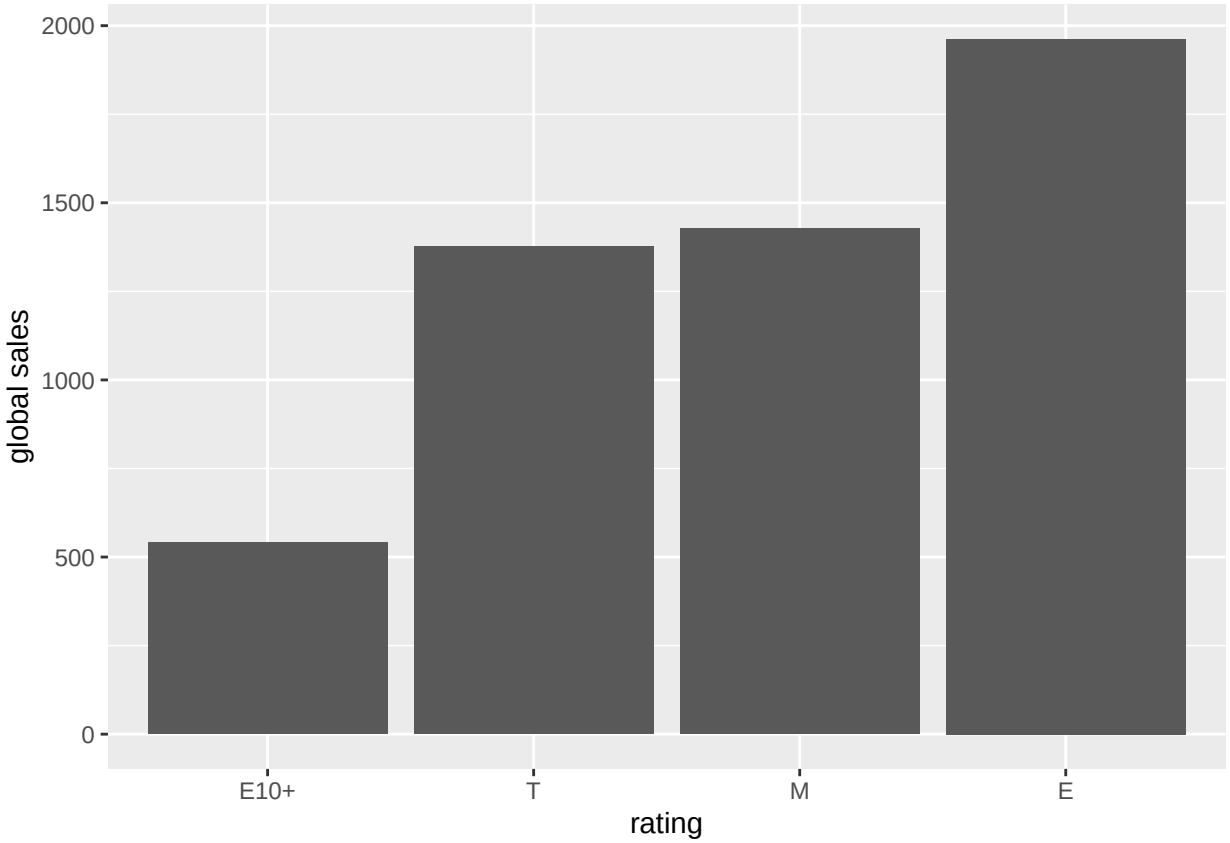
(i) Global sales by platform: Global sales differ significantly across platform, with PS games having the highest sales, followed by Nintendo, Xbox, PC, and lastly Sega.



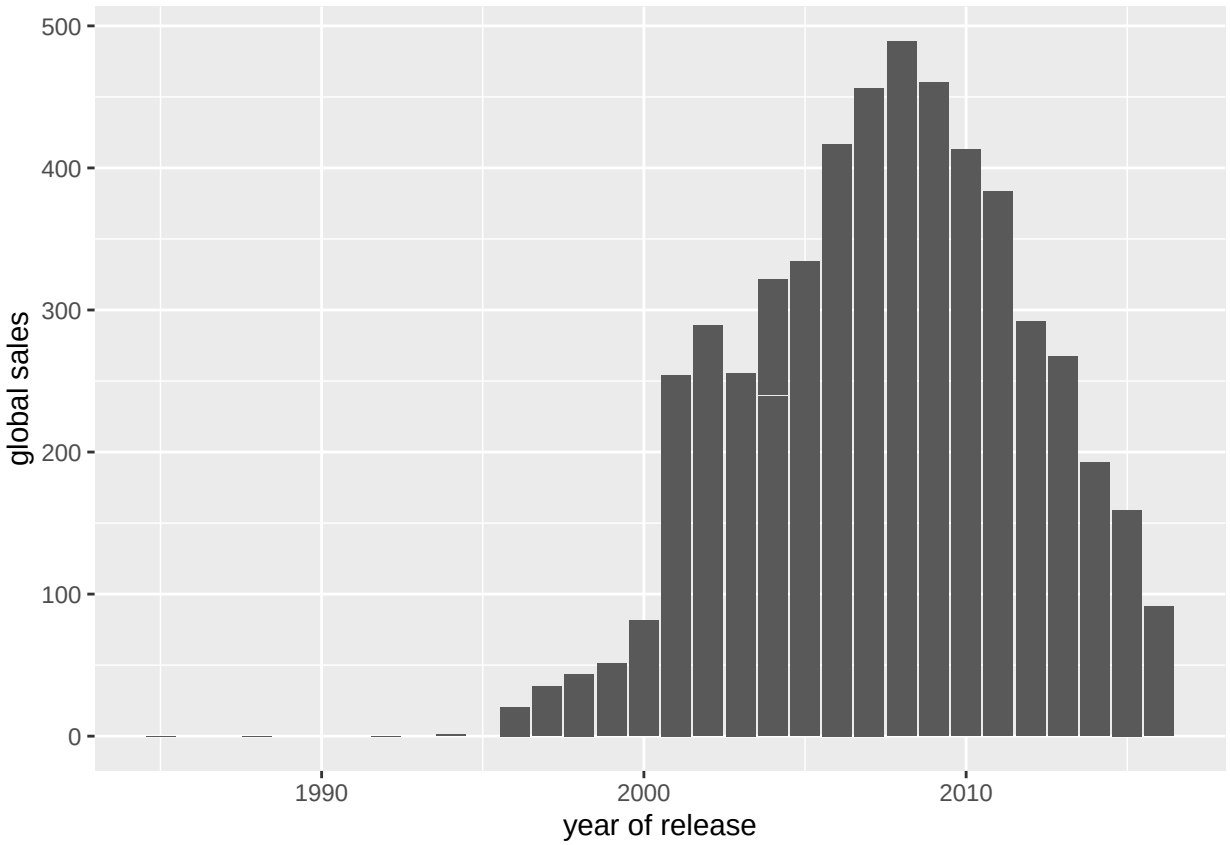
(ii) Global sales by genre: Global sales also vary significantly across genre, with action games having the highest sales, followed by sports etc., and strategy games having the least sales.



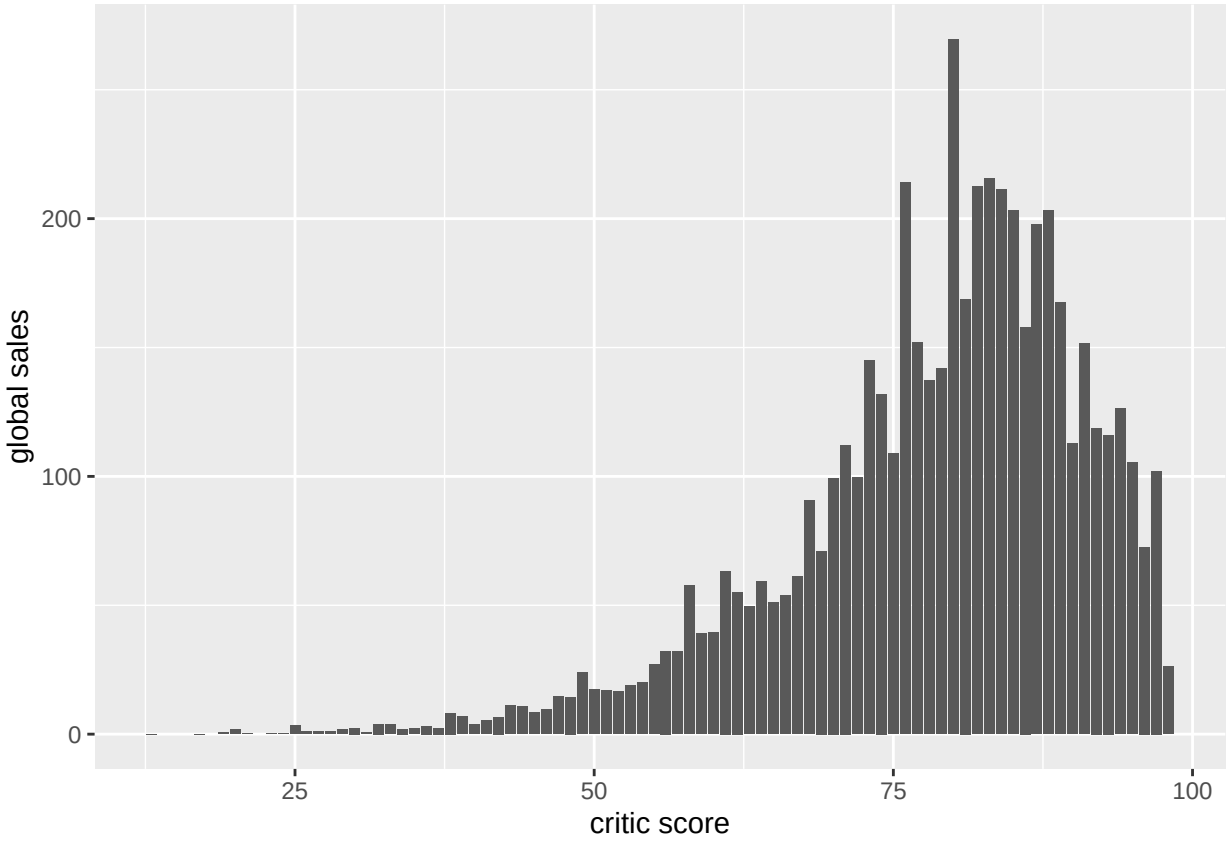
(iii) Global sales by rating: Global sales seem to be affected by rating category as well, with games rated "Everyone" having the highest sales, followed by "Mature", "Teen", and "Everyone 10+".

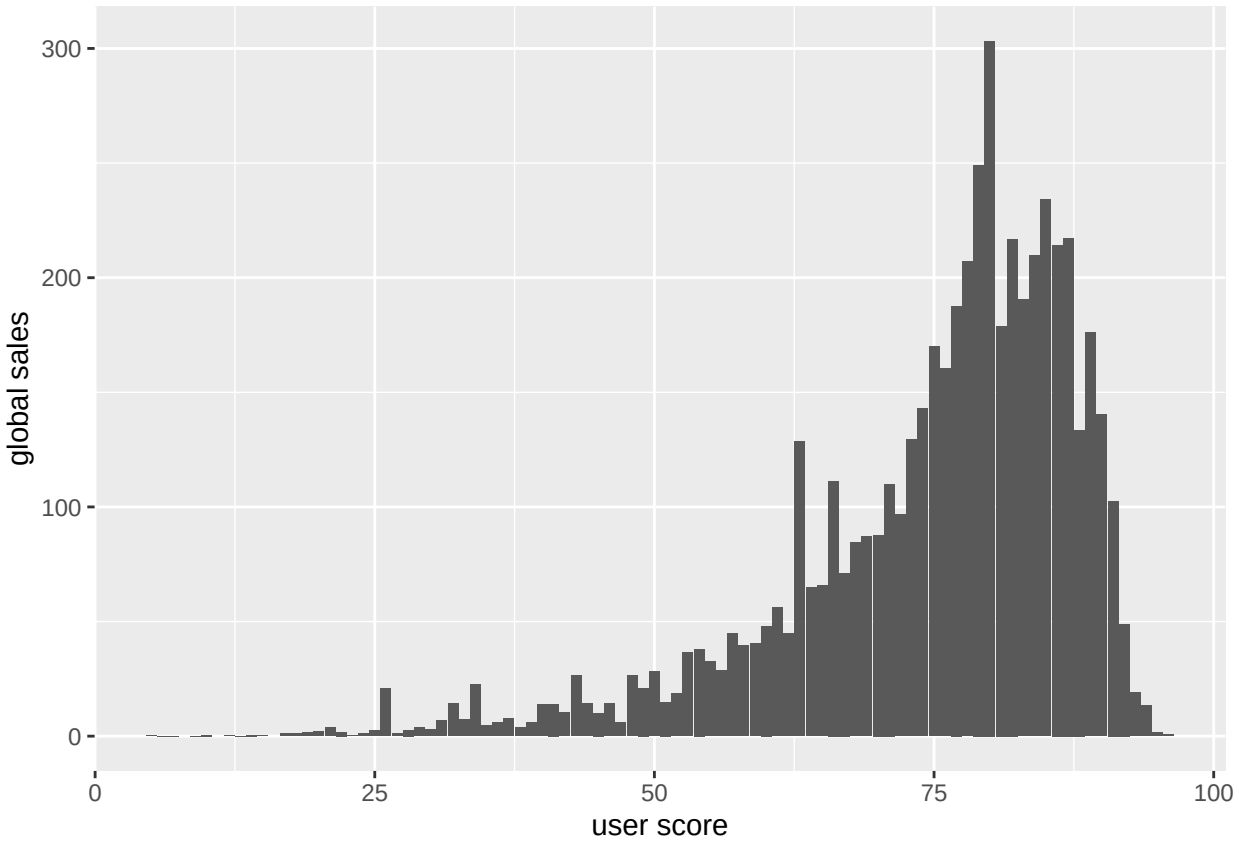


(iv) Global sales by year of release: Global sales of video games took off since 2000, and started seeing a steady decline since 2008.

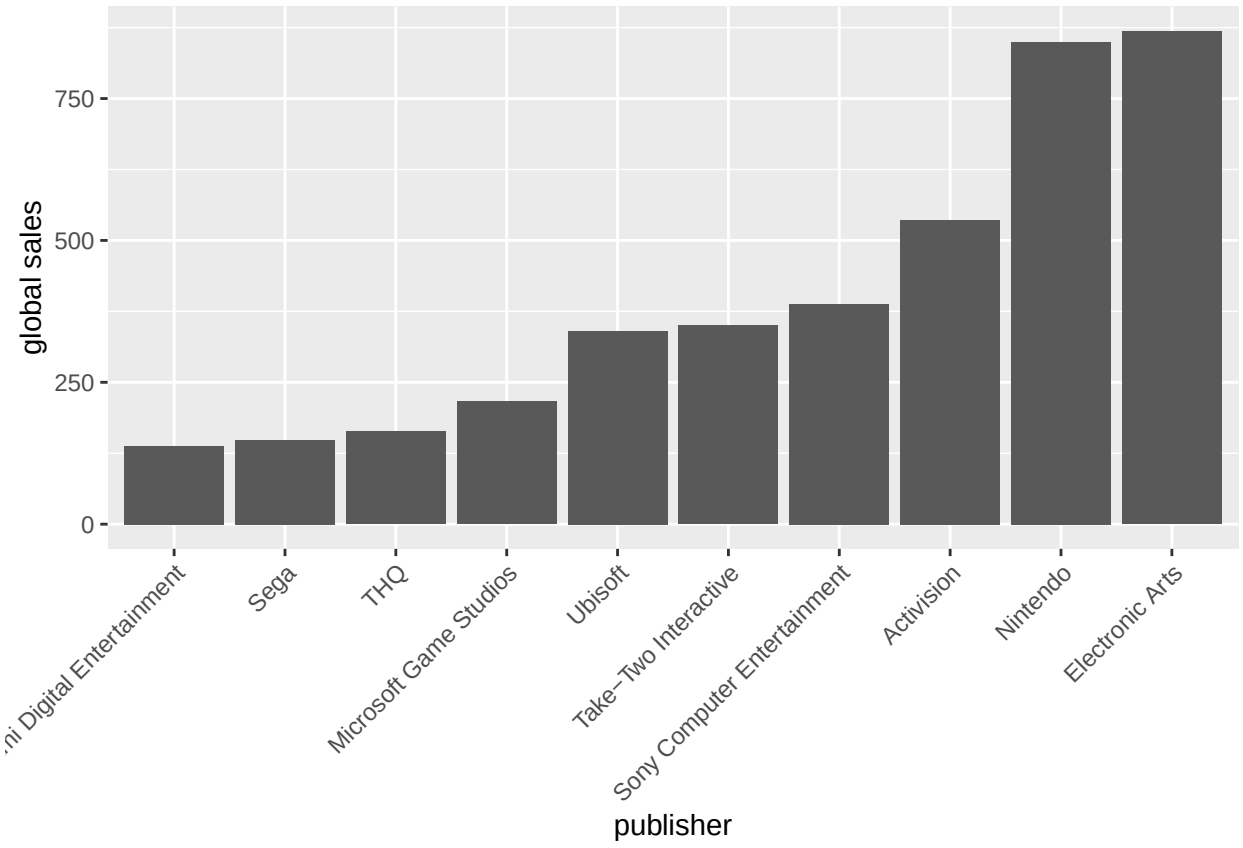


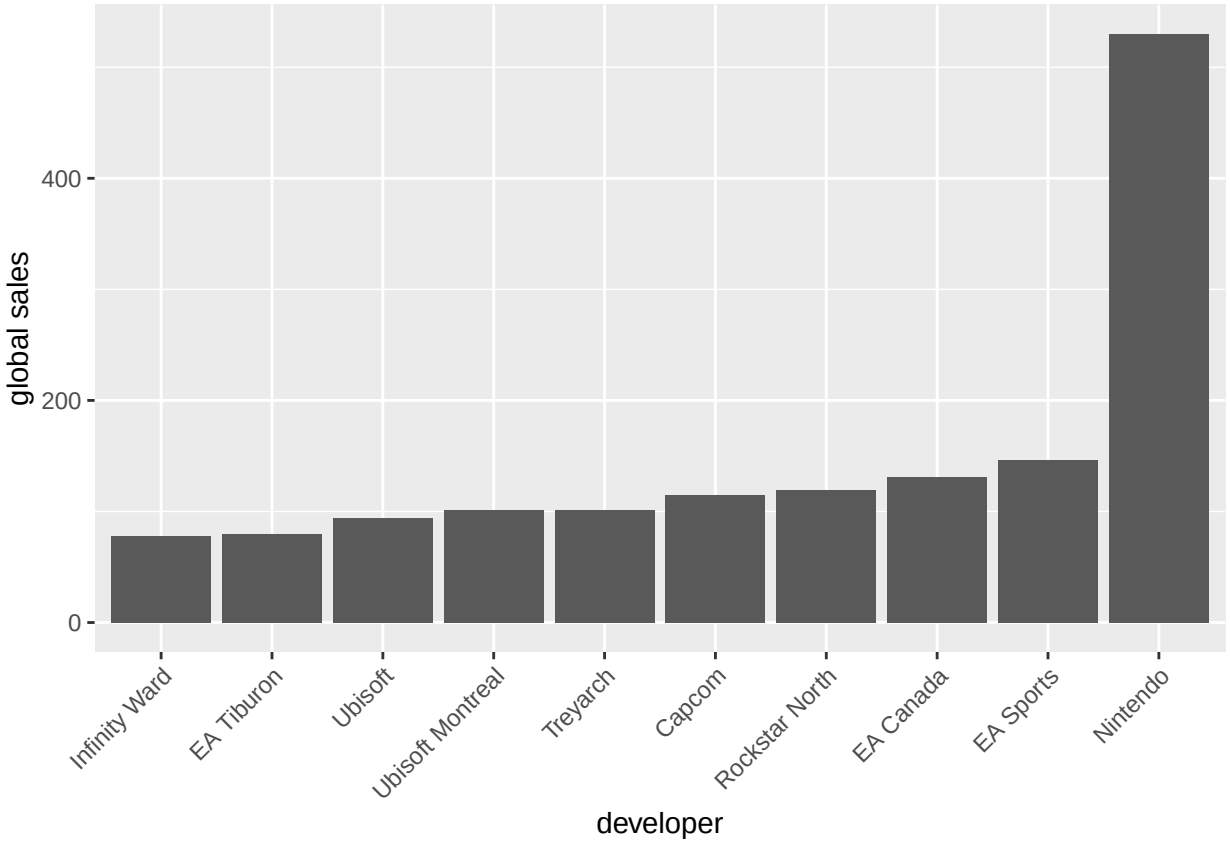
(v) Global sales by critic score & (vi) user score: Global sales appear to be determined by critic score and user score, with games having a score of above 70 accounting for bulk of the sales.


global sales
200
100
0
25
50
75
100
critic score



(vii) Global sales by publisher & (viii) developer: Global sales vary significantly across publisher and developer. Below are the global sales of the top 10 publishers and developers.


global sales
750
500
250
0
ni Digital Entertainment
Sega
THQ
Microsoft Game Studios
Ubisoft
Take−Two Interactive
Sony Computer Entertainment
Activision
Nintendo
Electronic Arts
publisher



Section 3: Results

We will now build our machine learning algorithms. Our analysis in Section 2 suggests that global sales is affected by the variables (i) platform, (ii) genre, (iii) rating, (iv) year of release, (v) critic score, (vi) user score, (vii) publisher, and (viii) developer.

Section 3.1: Splitting the Kaggle Video Game Sales dataset

We start by splitting our dataset into (i) validation, (ii) training, and (iii) testing sets.

The validation set will be 10% of the data. This is the final hold-out set that we will use to evaluate our final algorithm. It will not be used for developing our algorithms.

```
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(vg_sales$Global_Sales, p = 0.1, list = F)
temp <- vg_sales[test_index,]
remainder <- vg_sales[-test_index,]
```

We need to make sure all possible values of the categorical variables in the validation set are also in the remainder set. In particular, we need to make sure the publishers and developers are in both sets, as these two variables have many unique categories.

```
validation_set <- temp %>%
  semi_join(remainder, by = "Publisher") %>%
  semi_join(remainder, by = "Developer")
```

We then add the rows removed from the validation set back into the remainder set.

```
removed <- anti_join(temp, validation_set)
remainder <- rbind(remainder, removed)
```

We now repeat the above process to split the remainder set into training and testing sets. Likewise, the test set will be 10% of the remainder data.

```
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(remainder$Global_Sales, p = 0.1, list = F)
temp <- remainder[test_index,]
train_set <- remainder[-test_index,]
```

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

removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
```

Section 3.2: Linear regression

We then build a linear regression model, predicting global sales based on (i) platform, (ii) genre, (iii) rating, (iv) year of release, (v) critic score, (vi) user score, (vii) publisher, and (viii) developer. (Note: Remember to take the log of global sales, as discussed in Section 2.2.)

The performance of the algorithm will be determined by its root mean square error (RMSE), with smaller RMSE indicating better performance.

```
model_lm <- train(log(Global_Sales) ~ Platform + Genre +
                    Rating + Year_of_Release +
                    Critic_Score + User_Score +
                    Publisher + Developer,
                  method = "lm",
                  data = train_set)
model_lm
```

```
## Linear Regression
##
## 5652 samples
##    8 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 5652, 5652, 5652, 5652, 5652, 5652, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   1.444051  0.2914826  1.077342
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

The linear regression model has a RMSE of 1.13.

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

```
##               Method    RMSE
## 1 Linear regression 1.13313
```

Section 3.3: k-nearest neighbour (knn)

Next, we build a knn model with cross validation to see if we can improve our predictions.

```
set.seed(1, sample.kind = "Rounding")
control <- trainControl(method = "cv", number = 10, p = 0.9)
model_knn <- train(log(Global_Sales) ~ Platform + Genre +
                     Rating + Year_of_Release +
                     Critic_Score + User_Score +
                     Publisher + Developer,
                   method = "knn",
                   trControl = control,
                   tuneGrid = data.frame(k = seq(3, 31, 4)),
                   data = train_set)
model_knn
```

```
## k-Nearest Neighbors
##
## 5652 samples
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5086, 5087, 5085, 5087, 5087, 5086, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
##    3  1.346598  0.1554361  1.0629421
##    7  1.290972  0.1781550  1.0162809
##   11  1.274645  0.1895206  1.0042825
##   15  1.264111  0.1993469  0.9957697
##   19  1.263756  0.1989605  0.9959400
##   23  1.264624  0.1972497  0.9975301
##   27  1.266246  0.1951138  0.9985612
##   31  1.267096  0.1941970  0.9984500
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 19.
```

The predict function will use the best-performing model. The knn model has a RMSE higher than linear regression model, suggesting that knn might not be very suitable for our dataset.

```
y_hat_knn <- predict(model_knn, test_set)
rmse_results <- bind_rows(rmse_results,
                          data.frame(Method = "knn",
                                     RMSE = RMSE(log(test_set$Global_Sales), y_hat_knn)))
rmse_results
```

```
##               Method     RMSE
## 1 Linear regression 1.133130
## 2               knn 1.260367
```

Section 3.4: Random forest

Finally, we build a random forest model with cross validation to see if we can improve our predictions.

```
set.seed(1, sample.kind = "Rounding")
control <- trainControl(method = "cv", number = 5)
grid <- data.frame(mtry = c(1, 5, 10, 25, 50, 100))
model_rf <- train(log(Global_Sales) ~ Platform + Genre +
                    Rating + Year_of_Release +
                    Critic_Score + User_Score +
                    Publisher + Developer,
                  method = "rf",
                  trControl = control,
                  tuneGrid = grid,
                  data = train_set)
model_rf
```

```
## Random Forest
##
## 5652 samples
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4520, 4521, 4522, 4521, 4524
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##     1   1.407819  0.2380754  1.1308382
##     5   1.298508  0.3964654  1.0433140
##    10   1.216943  0.4199683  0.9749131
##    25   1.116668  0.4692248  0.8894014
##    50   1.050087  0.5027673  0.8311455
##   100   1.002332  0.5216945  0.7862689
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 100.
```

The predict function will use the best-performing model. The random forest model has a RMSE much smaller than the linear regression and knn models.

```
y_hat_rf <- predict(model_rf, test_set)
rmse_results <- bind_rows(rmse_results,
                          data.frame(Method = "Random forest",
                                     RMSE = RMSE(log(test_set$Global_Sales), y_hat_rf)))
rmse_results
```

```
##               Method      RMSE
## 1 Linear regression 1.1331302
## 2               knn 1.2603668
## 3     Random forest 0.9484296
```

Section 3.5: Testing the final model using the validation set

Our above results show that the random forest model has the best performance (i.e., the smallest RMSE). We will now test the model using the validation set, where we obtain a RMSE of around 0.92.

```
y_hat_final <- predict(model_rf, validation_set)
RMSE(log(validation_set$Global_Sales), y_hat_final)
```

```
## [1] 0.9219711
```

Section 4: Conclusion

We have successfully trained a machine learning algorithm to predict video game sales, using (i) platform, (ii) genre, (iii) rating, (iv) year of release, (v) critic score, (vi) user score, (vii) publisher, and (viii) developer as the predictors. Our final model is random forest with cross validation, which performed better than linear regression and knn with cross validation.

However, this model used the whole range of critic score and user score in its predictions, as well as the whole list of publishers and developers. It could be that higher critic score and user score, and top publishers and developers, have greater effect on video game sales. Future work can dive deeper into these aspects to refine the model. Future work can also look into the relative importance of each predictor to improve the model further.

Section 5: References

Being new to R and machine learning, I have consulted the following materials when doing up this video game sales project.

- https://rafalab.github.io/dsbook/index.html
- https://www.kaggle.com/datasets/rush4ratio/video-game-sales-with-ratings
- https://www.kaggle.com/code/yonatanrabinovich/video-games-sales-regression-techniques?scriptVersionId=55716523
- https://www.kaggle.com/code/najibmozahem/video-game-sales-and-reviews-with-r