video game analysis

Kashish Pandey

Importing Libraries

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.2
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.3     ✓ purrr   0.3.4
## ✓ tibble  3.0.5     ✓ dplyr   1.0.3
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
## Warning: package 'tibble' was built under R version 4.0.2
```

```
## Warning: package 'tidyr' was built under R version 4.0.2
```

```
## Warning: package 'readr' was built under R version 4.0.2
```

```
## Warning: package 'purrr' was built under R version 4.0.2
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
## Warning: package 'stringr' was built under R version 4.0.2
```

```
## Warning: package 'forcats' was built under R version 4.0.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.2
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(elasticnet)
```

```
## Warning: package 'elasticnet' was built under R version 4.0.2
```

```
## Loading required package: lars
```

```
## Warning: package 'lars' was built under R version 4.0.2
```

```
## Loaded lars 1.2
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.2
```

```
## corrplot 0.84 loaded
```

```
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 4.0.2
```

```
##
## Attaching package: 'kernlab'
## The following object is masked from 'package:purrr':
## 
##     cross
```

```
## The following object is masked from 'package:ggplot2':
## 
##     alpha
```

EDA

na.strings is removing null/blank values within the dataset

```
vg_sales <- read.csv("dataset/Video_Games_Sales_as_at_22_Dec_2016.csv",
                        sep=",",na.strings=c(""," ","NA","N/A"))
```

viewing the first 5 lines of the csv file

```
head(vg_sales)
```

```
##                       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        8.0        322  Nintendo      E
## 2         NA         NA      <NA>   <NA>
## 3        8.3        709  Nintendo      E
## 4        8.0        192  Nintendo      E
## 5         NA         NA      <NA>   <NA>
## 6         NA         NA      <NA>   <NA>
```

general summary of dataset

```
# summary(vg_sales)
```

checking number of null values within the dataset

```
colSums(is.na(vg_sales))
```

```
##            Name        Platform Year_of_Release           Genre       Publisher 
##               2               0             269               2              54 
##        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            9129            9129            6623 
##          Rating 
##            6769
```

dropping NA values from dataset

many missing values within this dataset

it is the combination of 2 different datasets and many of the original observations

did not match the data from the second set

```
vg_sales <- vg_sales[complete.cases(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
```

getting internal structure of each feature

```
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     : num  8 8.3 8 8.5 6.6 8.4 8.6 7.7 6.3 7.4 ...
##  $ 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" ...
```

examining outlier data for sales

```
summary(vg_sales$NA_Sales)
```

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

```
summary(vg_sales$EU_Sales)
```

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

```
summary(vg_sales$JP_Sales)
```

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

```
summary(vg_sales$Other_Sales)
```

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

```
summary(vg_sales$Global_Sales)
```

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

examining outlier data for score/count

```
summary(vg_sales$Critic_Score)
```

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

```
summary(vg_sales$Critic_Count)
```

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

```
summary(vg_sales$User_Count)
```

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

```
summary(vg_sales$User_Score)
```

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

critic score is int and user score is num

changing user score to int to keep it consistent

```
vg_sales$User_Score <- as.integer(vg_sales$User_Score)
summary(vg_sales$User_Score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   6.000   7.000   6.737   8.000   9.000
```

putting critic score and user score on the same scale

user score was only out of 10; critic was out of 100 both are out of 100 now

```
vg_sales$User_Score <- vg_sales$User_Score * 10
```

rating variable there is only 1 occurrence of AO, K-A, and RP going to add AO, K-A, and RP into Mature rating and Everyone rating

```
vg_sales %>% count(Rating)
```

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

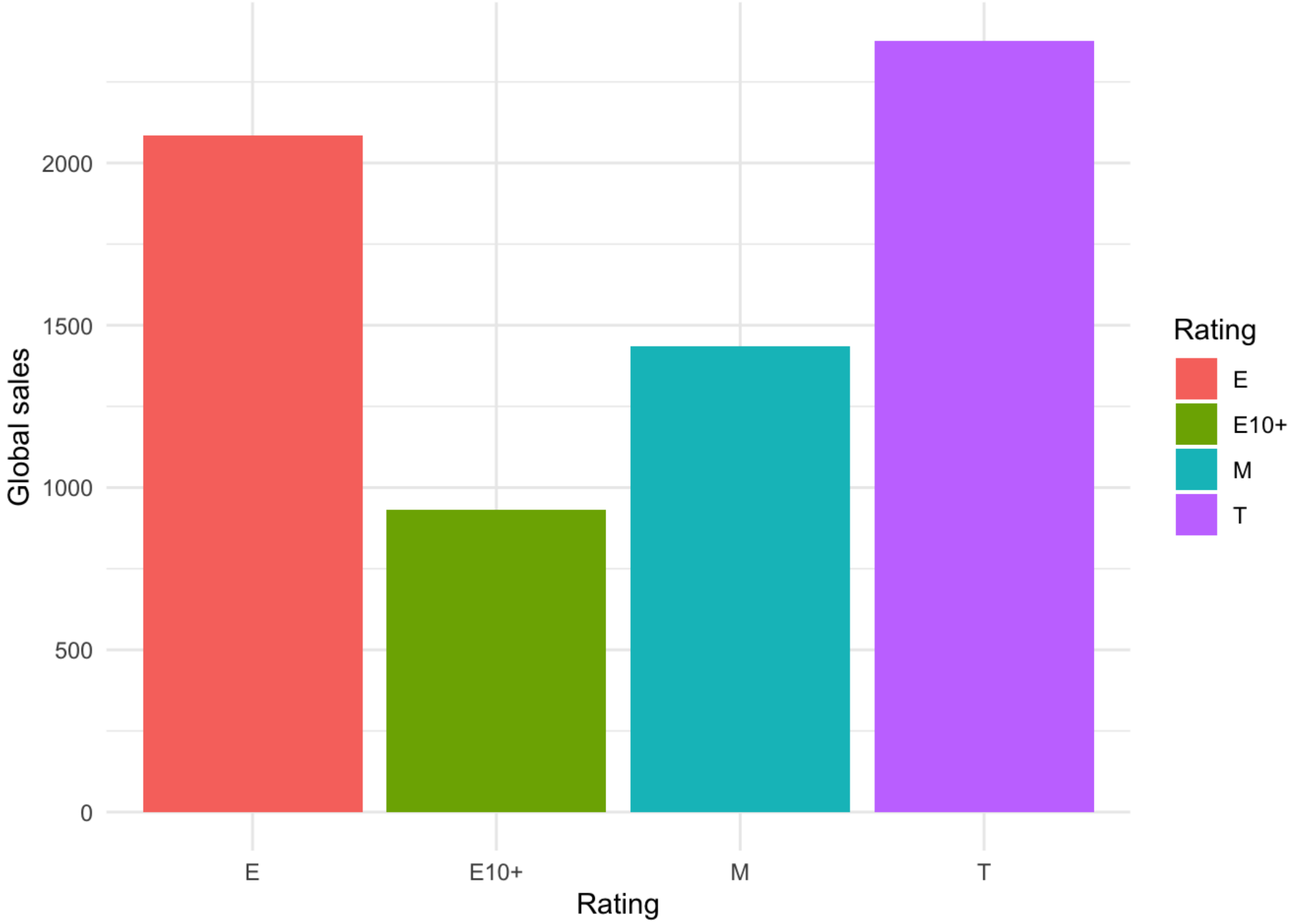
```
vg_sales <- vg_sales %>% mutate(Rating = ifelse(Rating == "AO", "M", Rating))
vg_sales <- vg_sales %>% mutate(Rating = ifelse(Rating == "K-A", "E", Rating))
vg_sales <- vg_sales %>% mutate(Rating = ifelse(Rating == "RP", "E", Rating))
vg_sales %>% count(Rating)
```

```
##   Rating    n
## 1      E 2084
## 2   E10+  930
## 3      M 1434
## 4      T 2377
```

Data Visualization

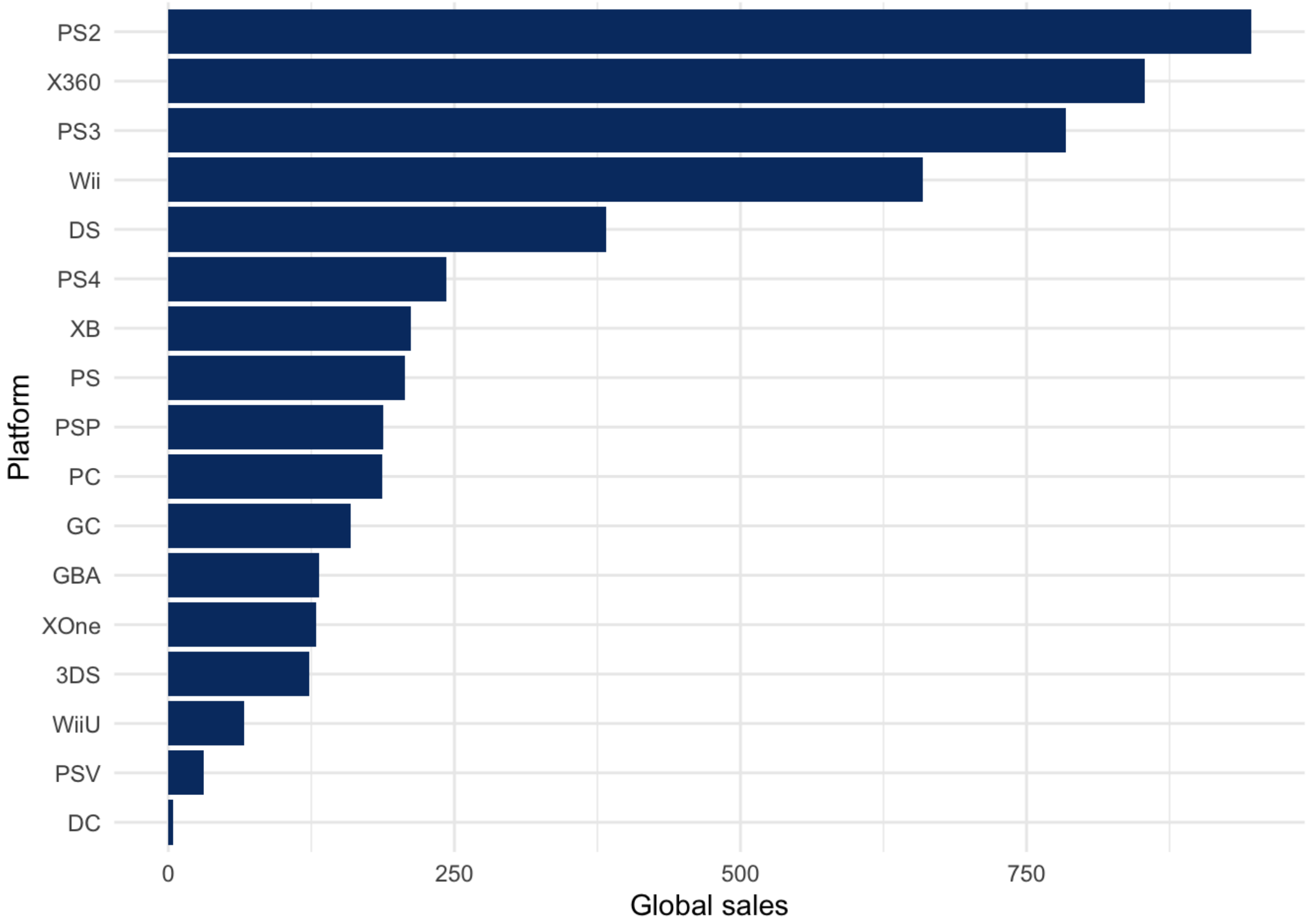
number of games per rating teen games have the highest global sales

```
rating_games_bar <- ggplot(vg_sales, aes(x=Rating,fill =Rating)) + geom_bar() +
      theme(text = element_text(size=10)) + xlab("Rating") + ylab("Global sales")+
  theme_minimal()
rating_games_bar
```



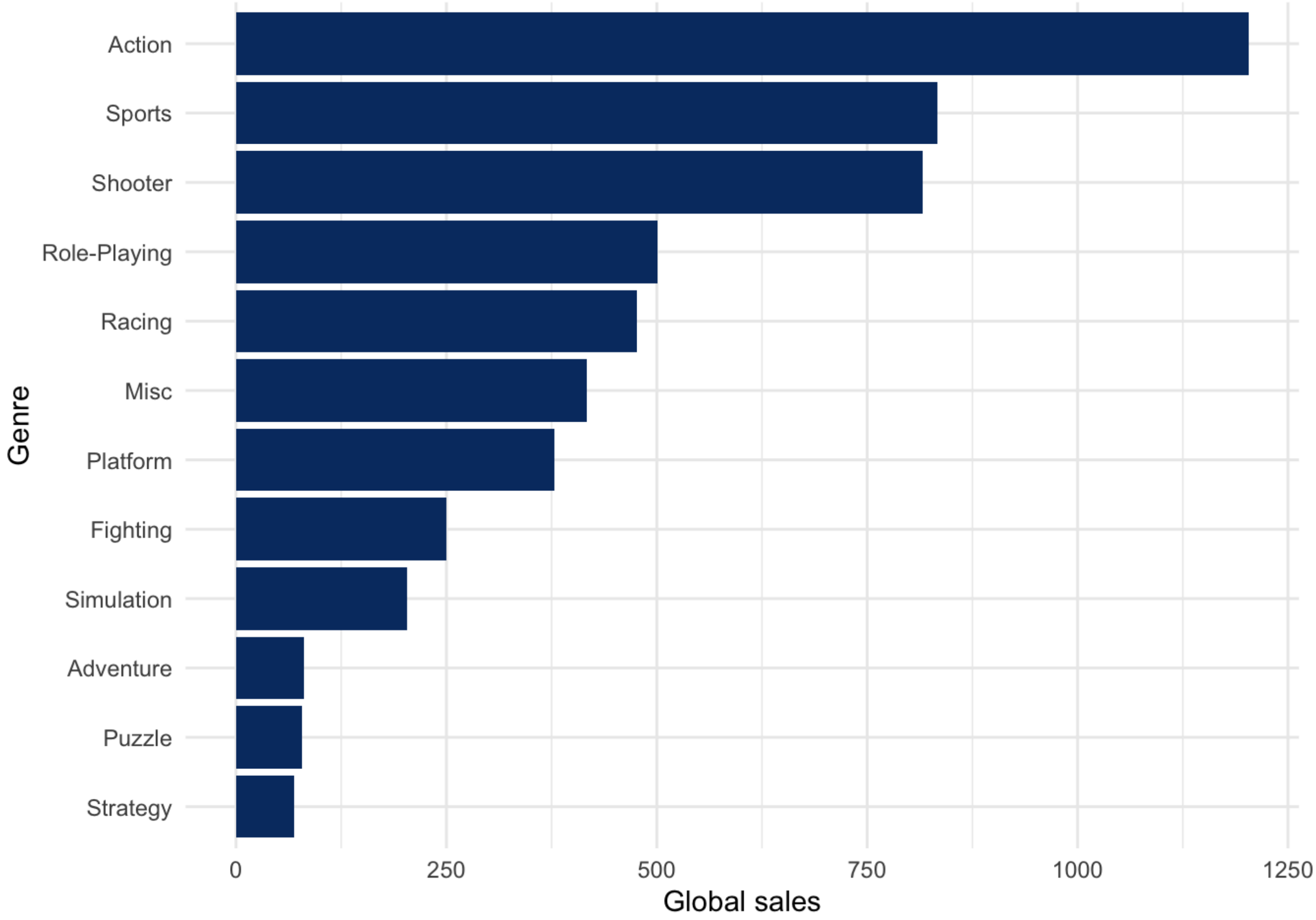
sales for each platform biggest sales from playstation 2 and xbox360

```
vg_sales %>% group_by(Platform) %>%
  summarise(vg_sales = sum(Global_Sales)) %>% ggplot() +
  geom_bar(aes(reorder(Platform, vg_sales), vg_sales), stat = "identity",
           fill = "#063970") +
  xlab("Platform") + ylab("Global sales") +
  coord_flip() + theme_minimal()
```



sales by genre action, sports, shooters are the top genres

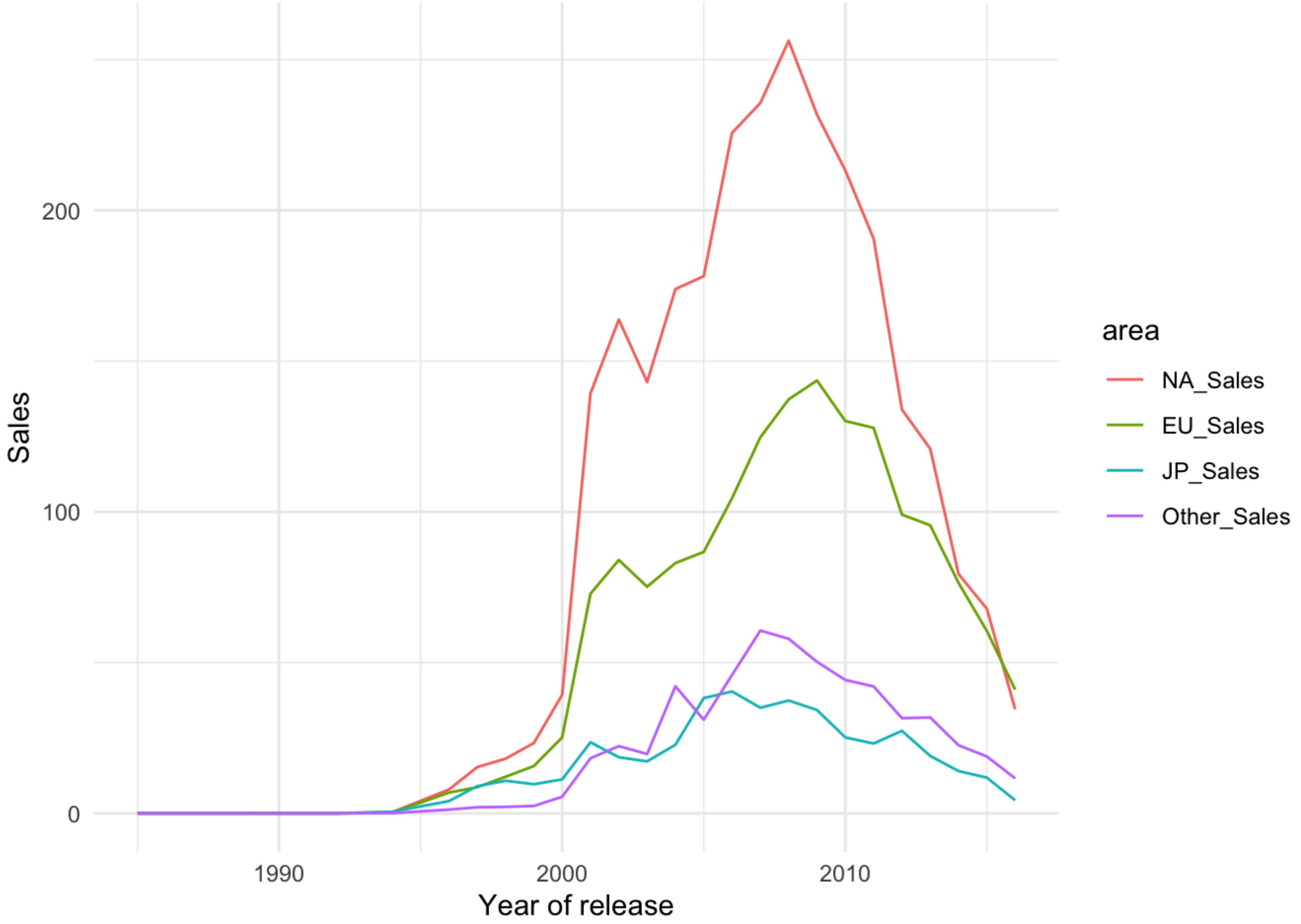
```
vg_sales %>% group_by(Genre) %>%
  summarise(vg_sales = sum(Global_Sales)) %>% ggplot() +
  geom_bar(aes(reorder(Genre, vg_sales), vg_sales), stat = "identity",
           fill = "#063970") +
  xlab("Genre") + ylab("Global sales") +
  coord_flip() + theme_minimal()
```



sales in North America, Europe, Japan, Other north america had the highest overall sales from 1990-2016

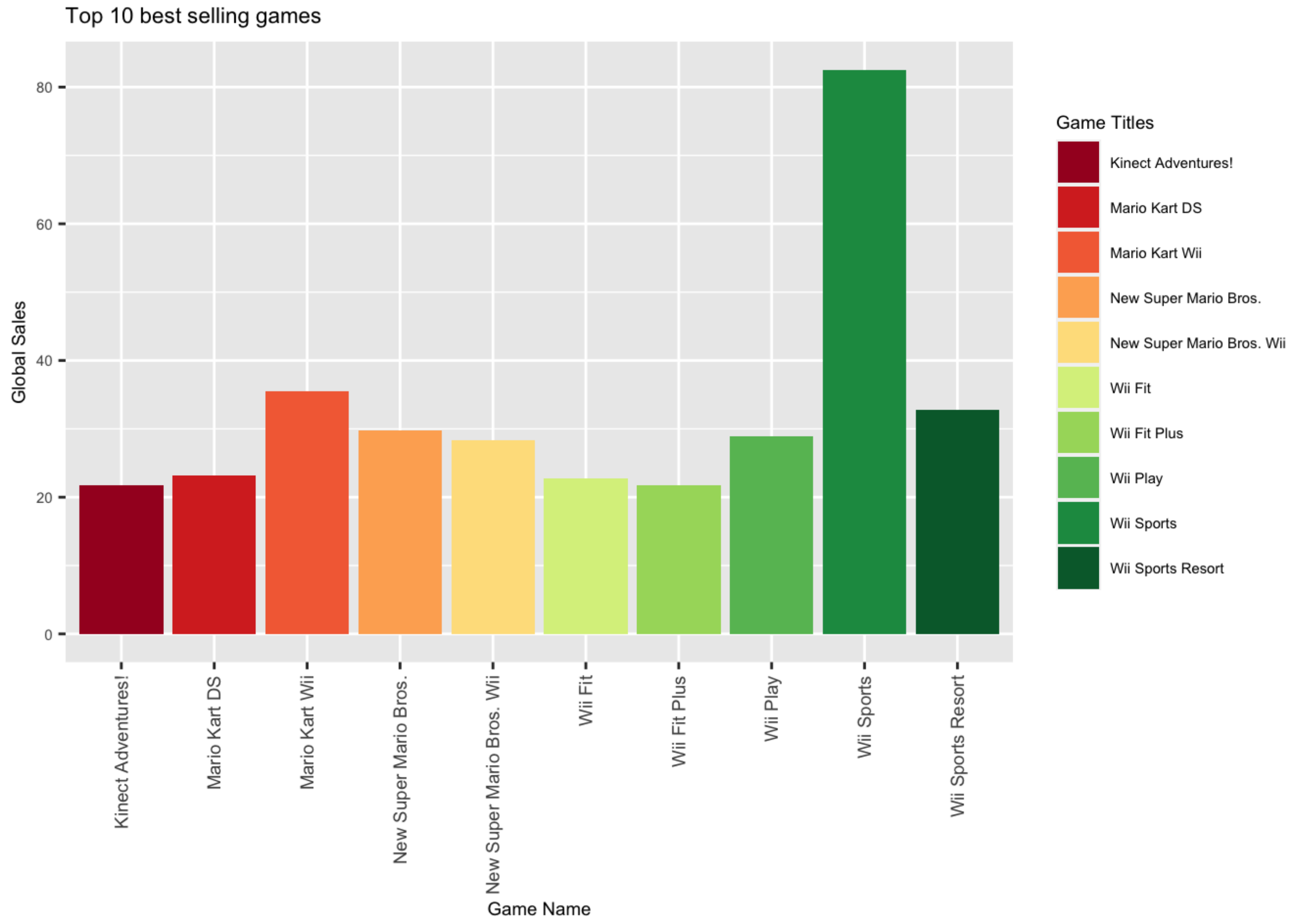
```
vg_sales %>% gather(area, vg_sales, NA_Sales:Other_Sales,
                    factor_key = TRUE) %>%
  group_by(area,Year_of_Release) %>%
  summarise(vg_sales = sum(vg_sales)) %>% ggplot() +
  xlab("Year of release") + ylab("Sales") +
  geom_line(aes(Year_of_Release, vg_sales, group = area, color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

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



top 10 best selling games globally wii sports is the #1 game sold globally

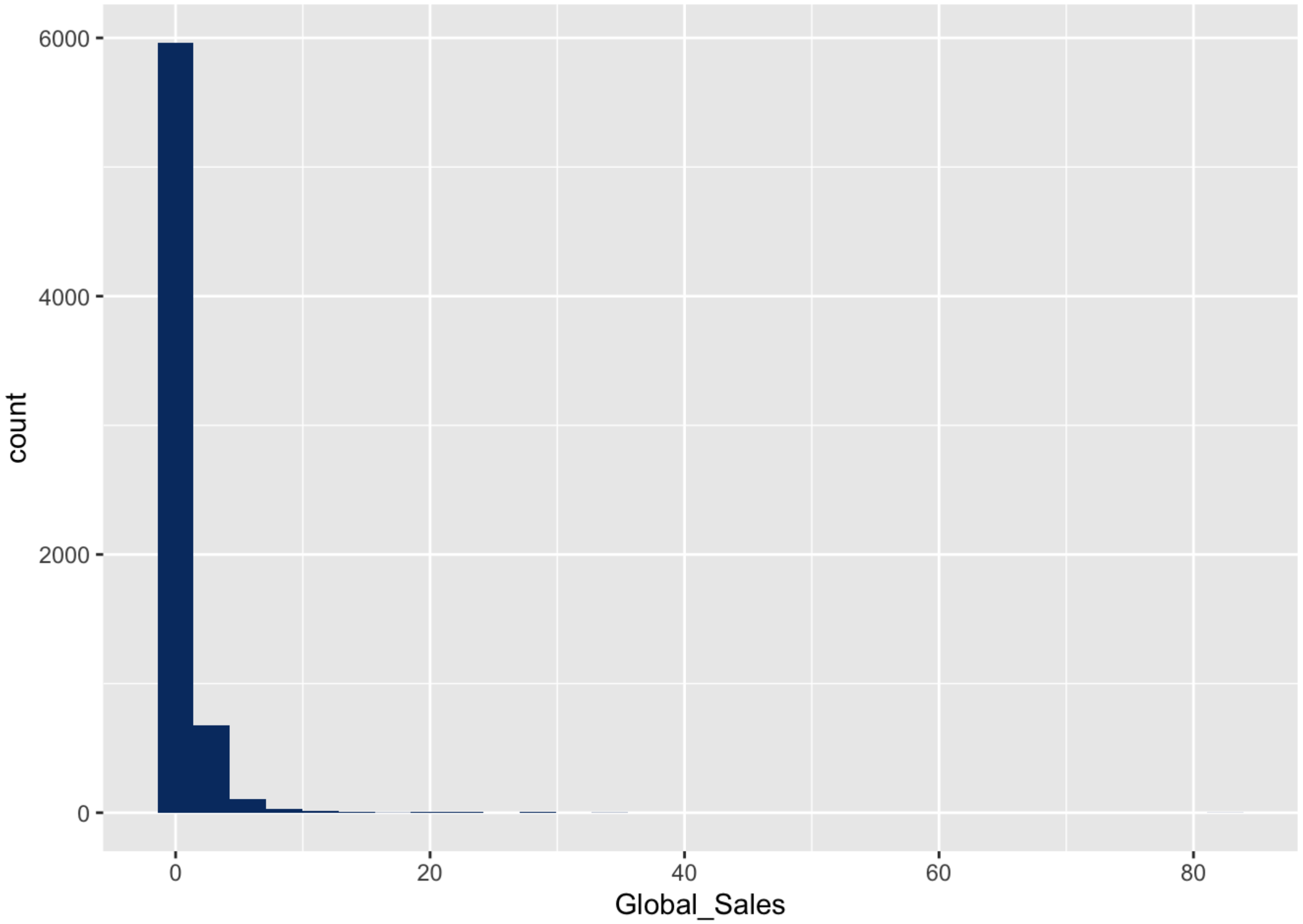
```
vg_sales %>% select(Name,Global_Sales) %>% arrange(desc(Global_Sales))%>% head(10)%>%
  ggplot(aes(x=Name,y=Global_Sales,fill= Name))+geom_bar(stat="identity")+
  labs(x="Game Name",y="Global Sales",
       title="Top 10 best selling games")+
  theme(text = element_text(size=7),legend.position="right",
        axis.text.x=element_text(angle = 90,vjust = 0.5,hjust = 1,size=7))+
  scale_fill_brewer(name= "Game Titles",palette="RdYlGn")
```



bar plot of global sales extremely skewed plot, need to change x axis to log axis

```
ggplot(vg_sales) + geom_histogram(aes(Global_Sales), fill = "#063970")
```

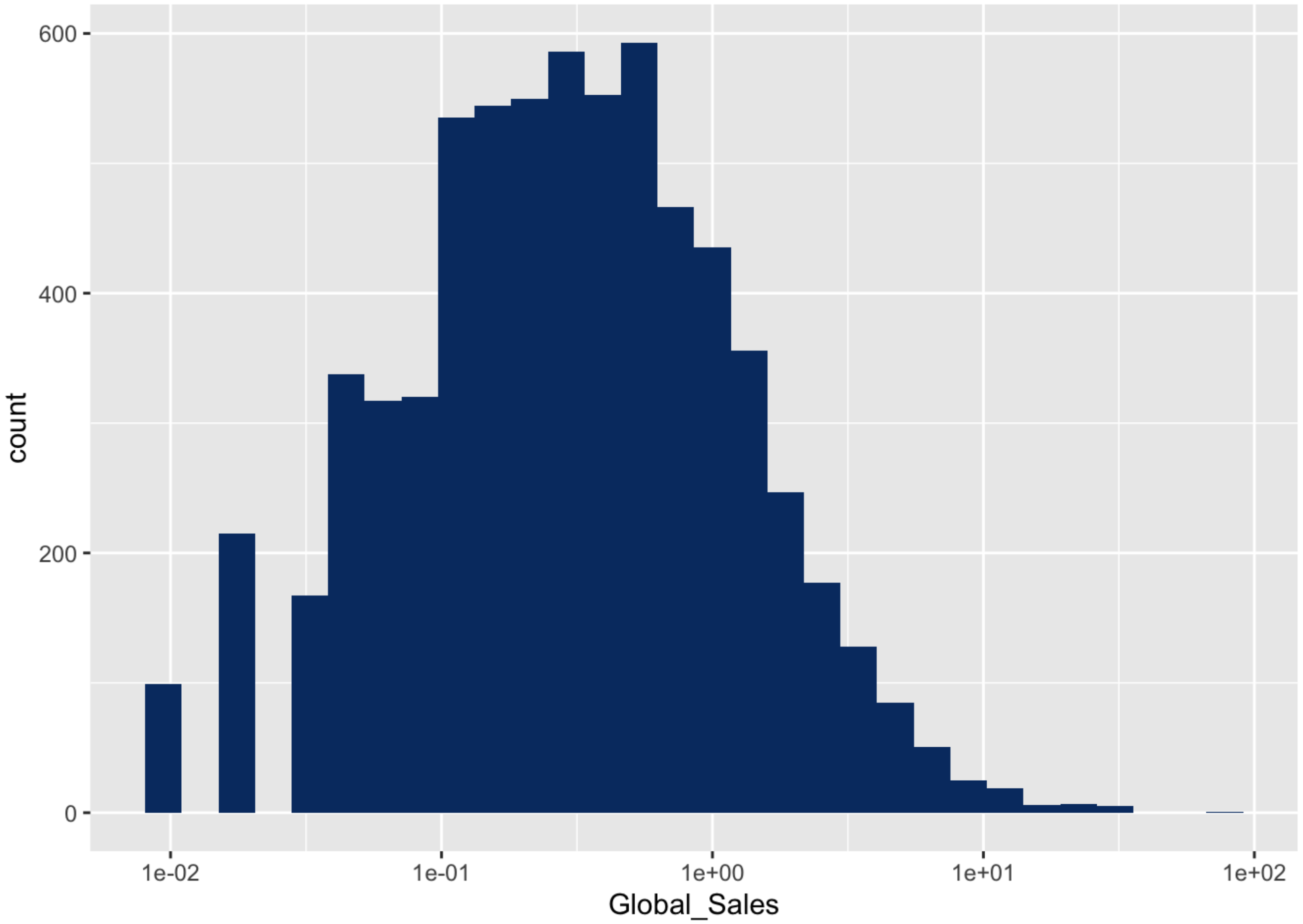
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



fixing axis, better distribution - similar to gaussian distribution

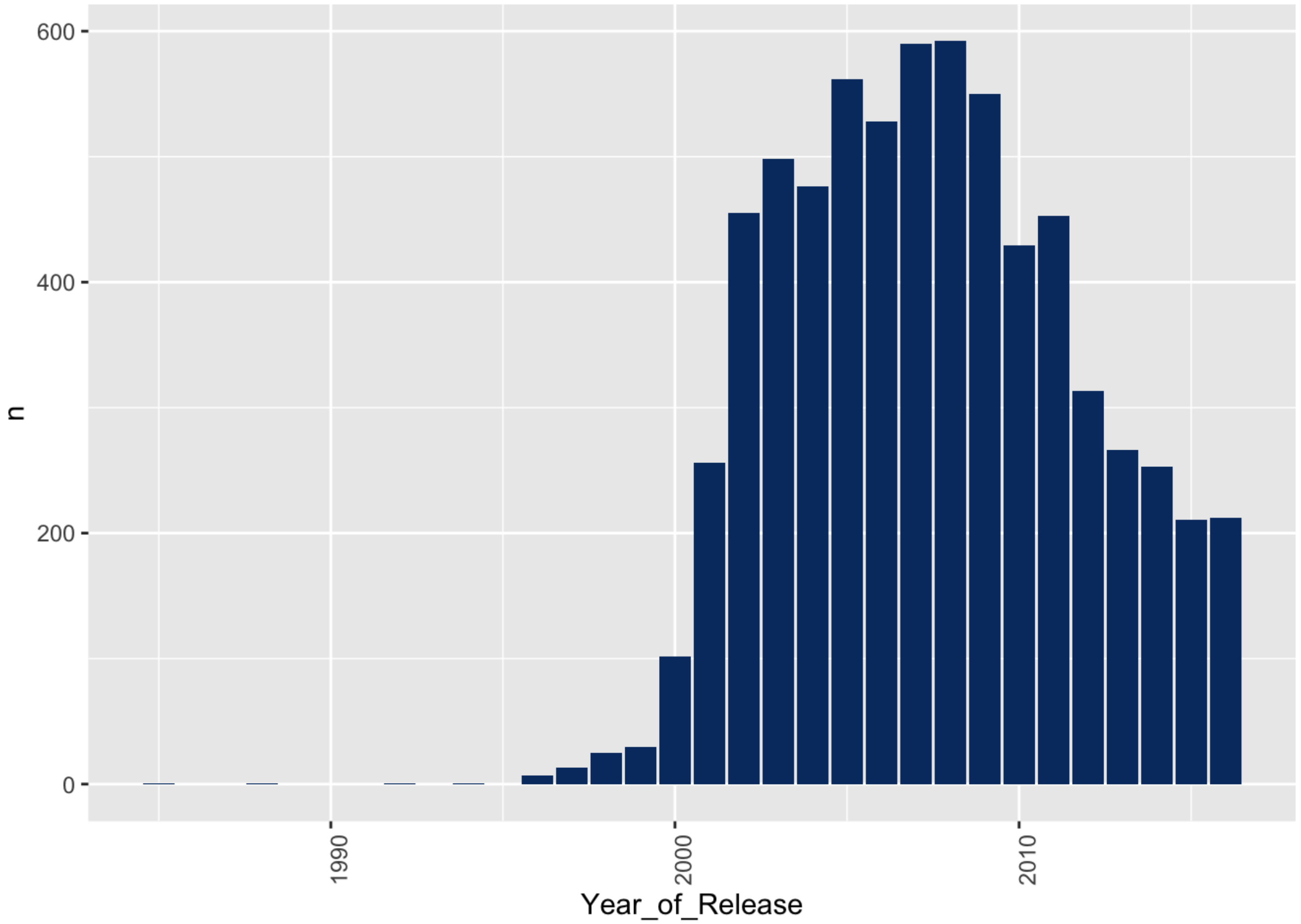
```
ggplot(vg_sales) + geom_histogram(aes(Global_Sales), fill = "#063970") +
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



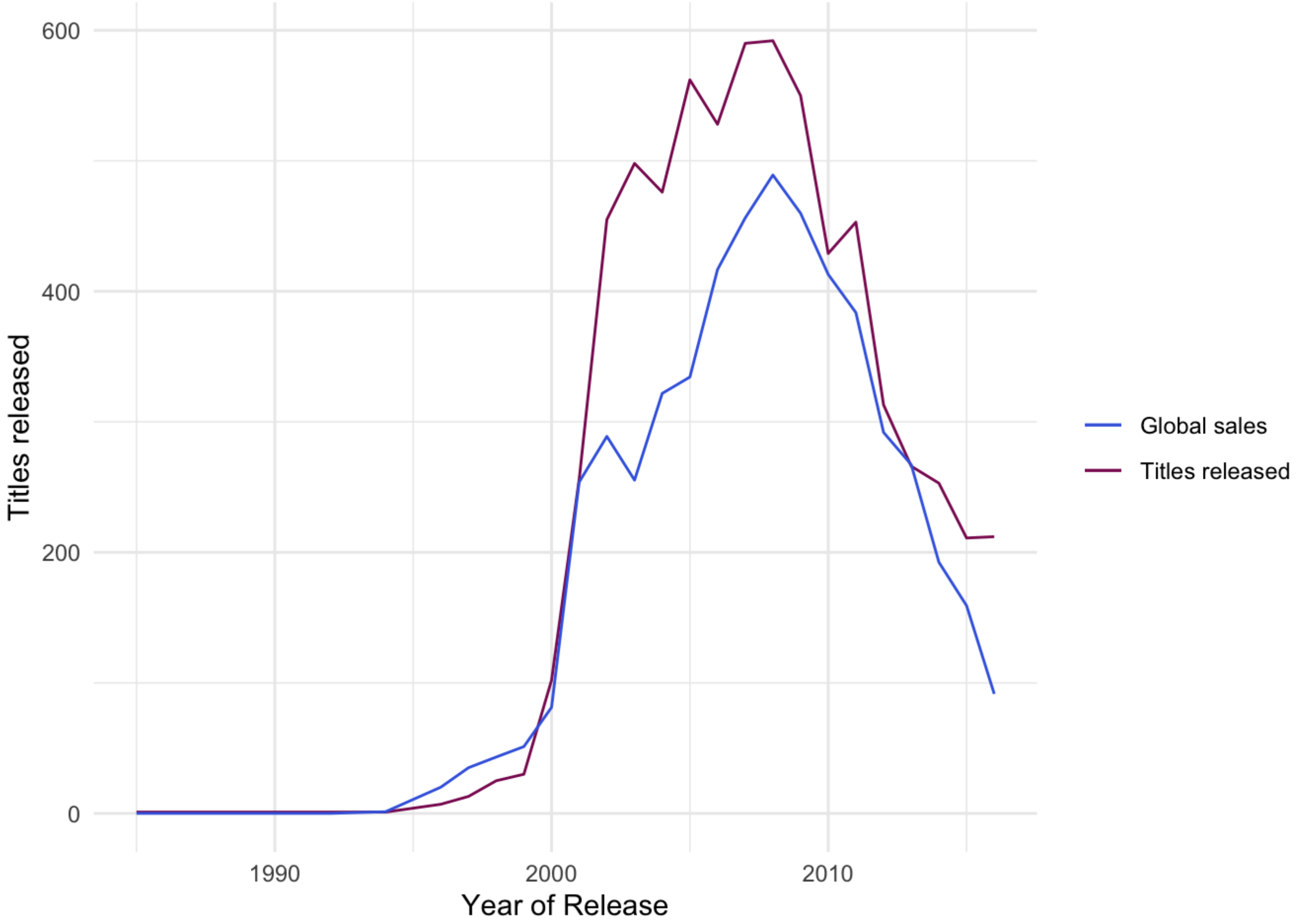
number of titles released each year there seems to be a peak within the data

```
vg_sales %>% group_by(Year_of_Release) %>%
  count() %>% ggplot() +
  geom_bar(aes(Year_of_Release, n), stat = "identity",
           fill = "#063970") + theme(axis.text.x = element_text(angle = 90))
```



sales each yr vs number of releases more revenue when more titles are released

```
color <- c("Titles released" = "maroon4", "Global sales" = "royalblue")
vg_sales %>% group_by(Year_of_Release) %>%
  summarise(vg_sales = sum(Global_Sales), count = n()) %>%
  ggplot() + xlab("Year of Release") + ylab("Titles released") +
  geom_line(aes(Year_of_Release, count, group = 1, color = "Titles released")) +
  geom_line(aes(Year_of_Release, vg_sales, group = 1, color = "Global sales")) +
  theme(axis.text.x = element_text(angle = 90), legend.position = "bottom") +
  scale_color_manual(name="",values = color) + theme_minimal()
```



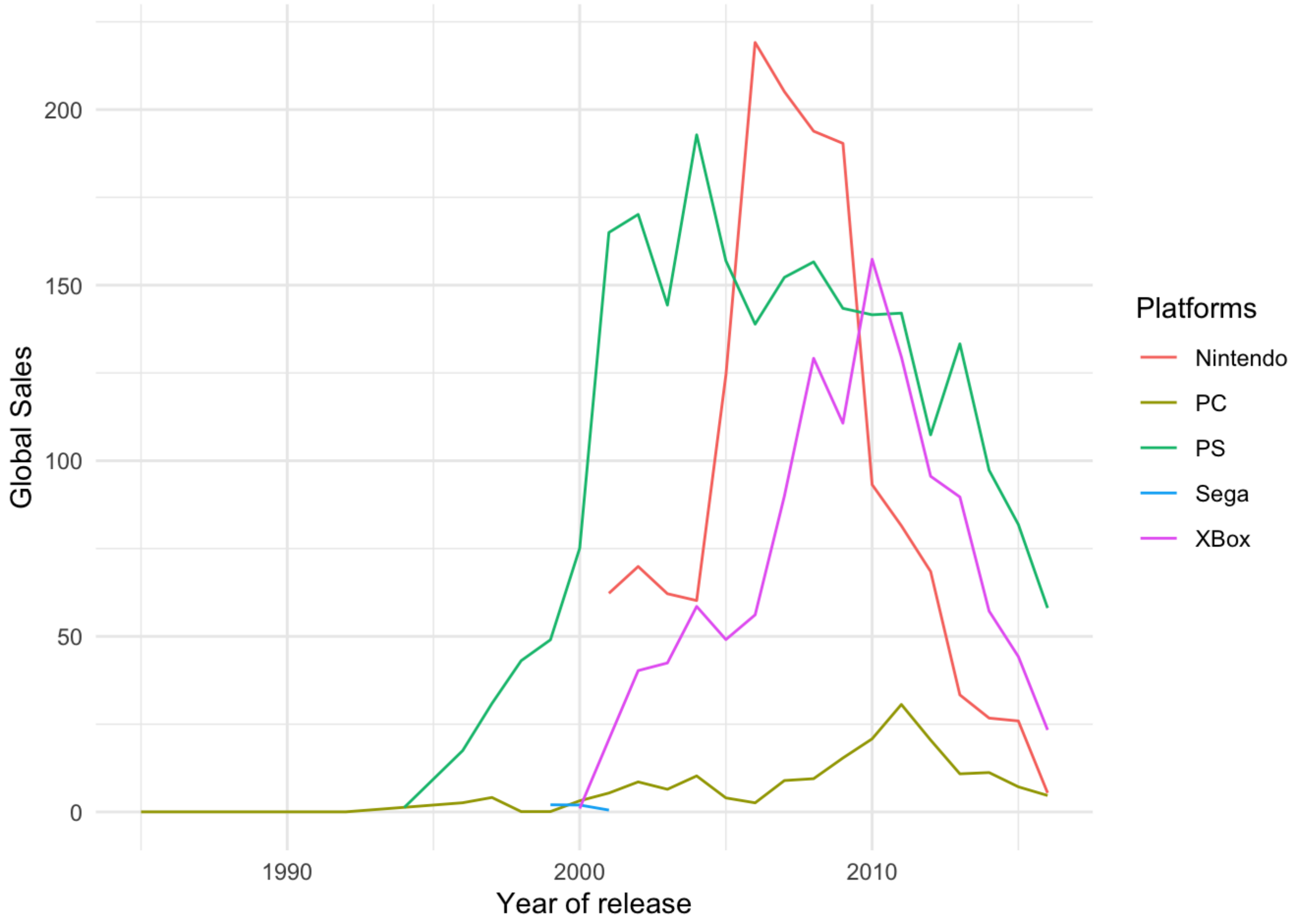
combining platform by company - to simplify all these platforms

```
vg_sales <- vg_sales %>% mutate(platform2 = case_when(
  Platform %in% c("Wii", "DS", "3DS", "WiiU", "GC", "GBA") ~ "Nintendo",
  Platform %in% c("X360", "XB", "XOne") ~ "XBox",
  Platform %in% c("PS3", "PS4", "PS2", "PS", "PSP", "PSV") ~ "PS",
  Platform == "PC" ~ "PC",
  Platform == "DC" ~ "Sega"
))
```

global sales each year for each platform nintendo and playstation both peaked around the same time

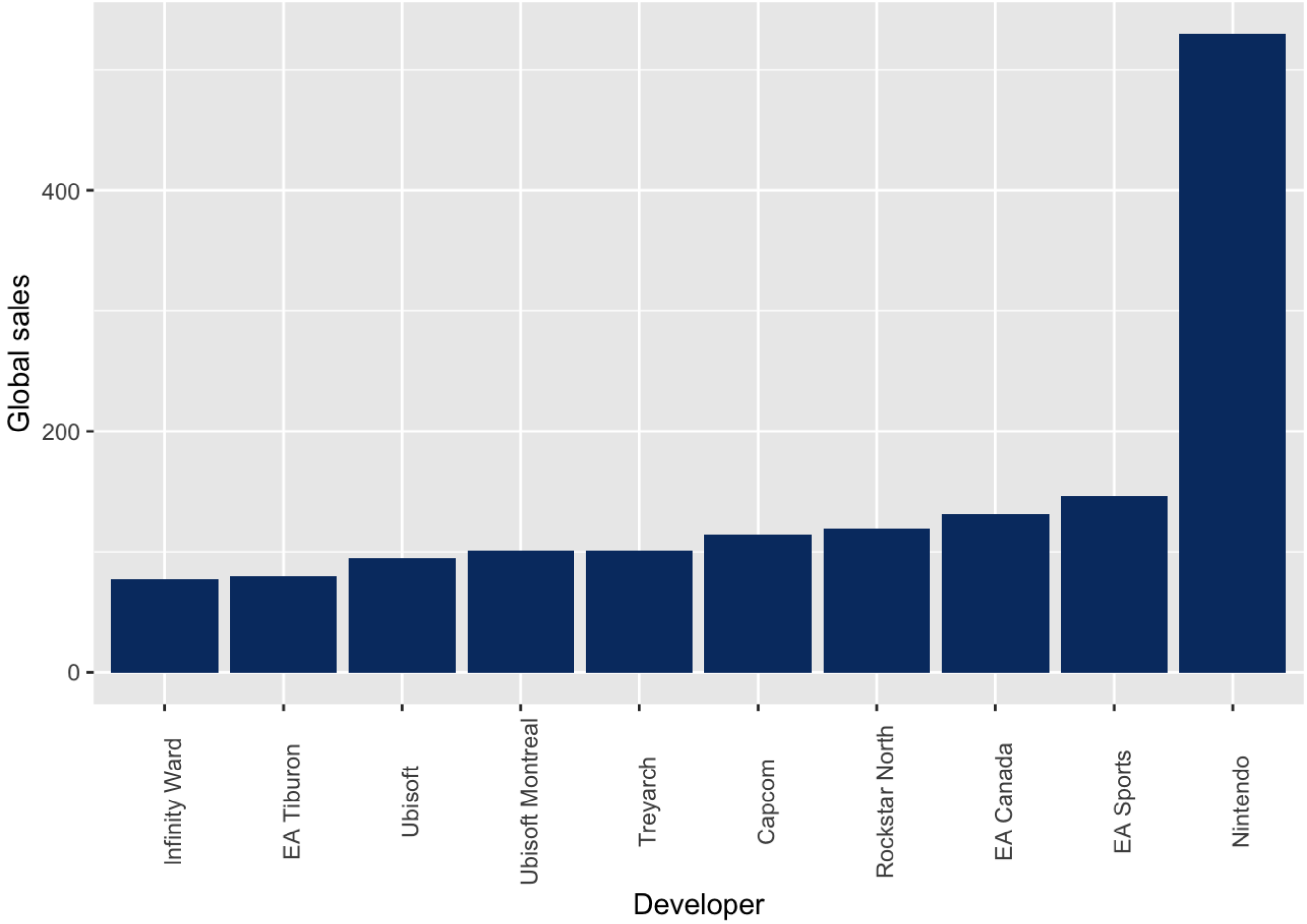
```
vg_sales %>% group_by(platform2, Year_of_Release) %>%
  summarise(vg_sales = sum(Global_Sales)) %>%
  ggplot() + xlab("Year of release") + ylab("Global Sales") +
  geom_line(aes(Year_of_Release, vg_sales, group = platform2, color = platform2)) +
  theme(legend.text = element_text(size = 7),
    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

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



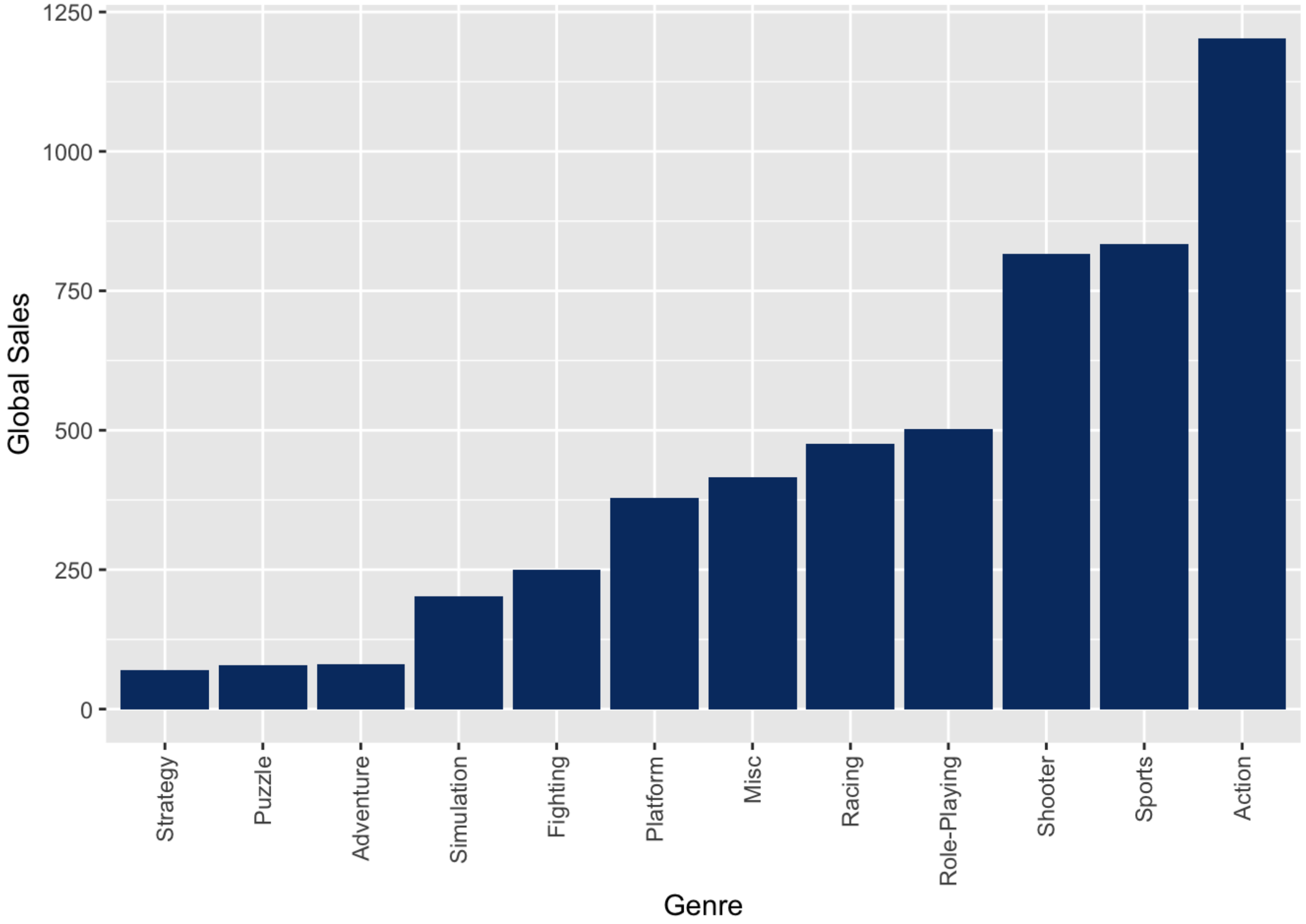
sales for each developer need to change individual bar colors

```
vg_sales %>% group_by(Developer) %>%
  summarise(vg_sales = sum(Global_Sales)) %>%
  arrange(desc(vg_sales)) %>% slice(1:10) %>%
  ggplot() + xlab("Developer") + ylab("Global sales")+
  geom_bar(aes(reorder(Developer, vg_sales), vg_sales),
           stat = "identity", fill = "#063970") +
  theme(axis.text.x = element_text(angle = 90))
```



sales for each gaming genre need to change individual bar colors

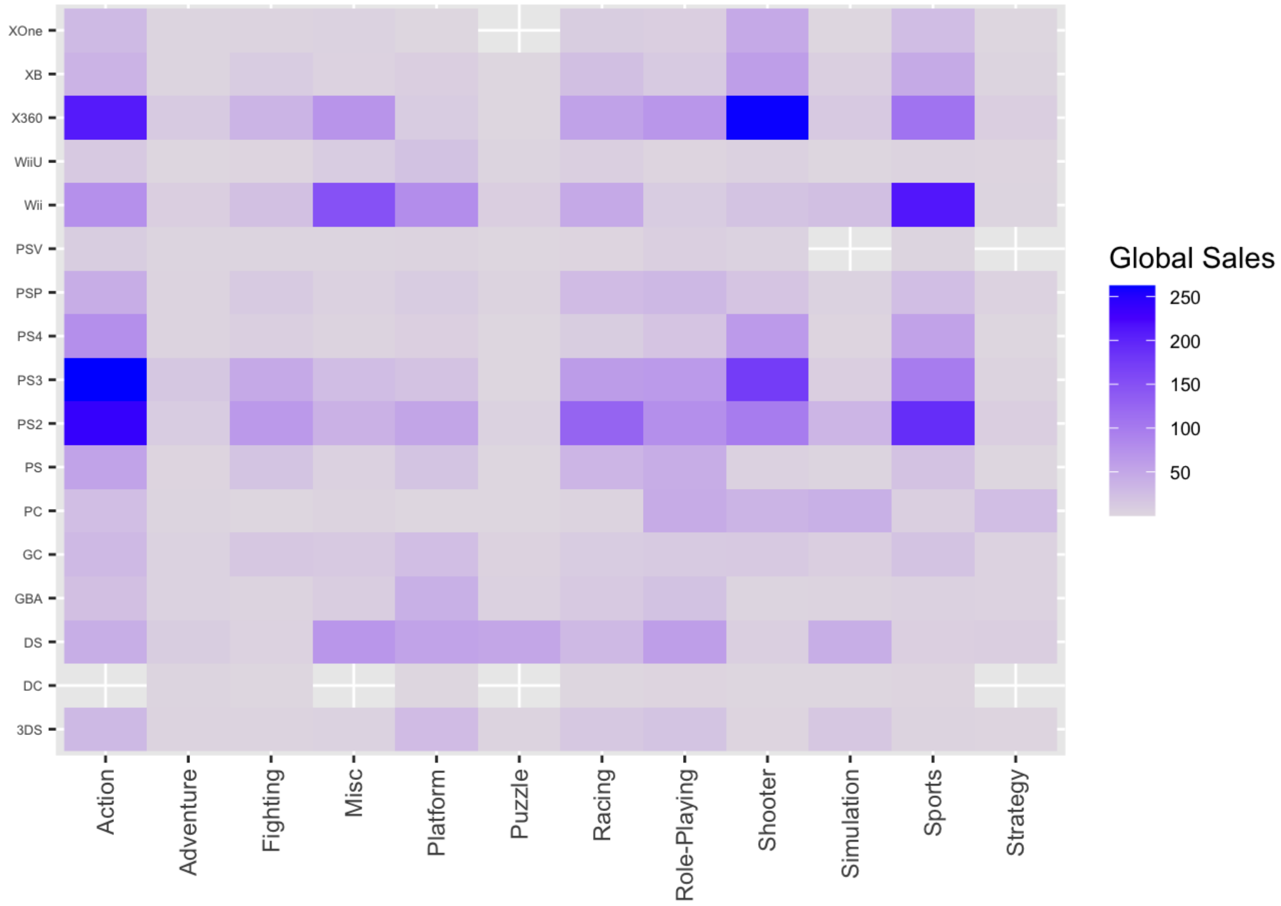
```
vg_sales %>% group_by(Genre) %>%
  summarise(vg_sales = sum(Global_Sales)) %>%
  ggplot() +
  geom_bar(aes(reorder(Genre, vg_sales), vg_sales), stat = "identity",
           fill = "#063970") +
  ylab("Global Sales") + xlab("Genre") +
  theme(axis.text.x = element_text(angle = 90,
                                   hjust = 1, vjust = 0.5))
```



sales for each platform in each genre TOP TWO: xbox 360 - shooter ps3 - action

```
vg_sales %>% group_by(Platform, Genre) %>%
  summarise(vg_sales = sum(Global_Sales)) %>%
  ggplot() + geom_raster(aes(Genre, Platform, fill = vg_sales)) +
  ylab("") + xlab("") +
  scale_fill_gradient(low = "#e4dee5", high = "blue") +
  theme(axis.text.x = element_text(angle = 90,
                                   vjust = 0.5, hjust = 1),
        axis.text.y = element_text(size = 5),
        legend.text = element_text(size = 7)) + labs(fill = "Global Sales")
```

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



MODELS for results

Overall, the sales vary depending on the platform, release year, and developer

The top developers had the highest sales

publishers is categorical but has many values

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

```
## Selecting by vg_sales
```

developers is categorical but has many values

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

```
## Selecting by vg_sales
```

creating new variable for whether a game is created by a top developer/publisher - making it binary(0,1)

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

whether games are exclusively launched on a specific platform

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

training and testing data sets

```
set.seed(2000)
```

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

including categorical data as well

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

creating RMSE function

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

linear regression model

base line model

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

# predicted values and RMSE
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_l
m))
```

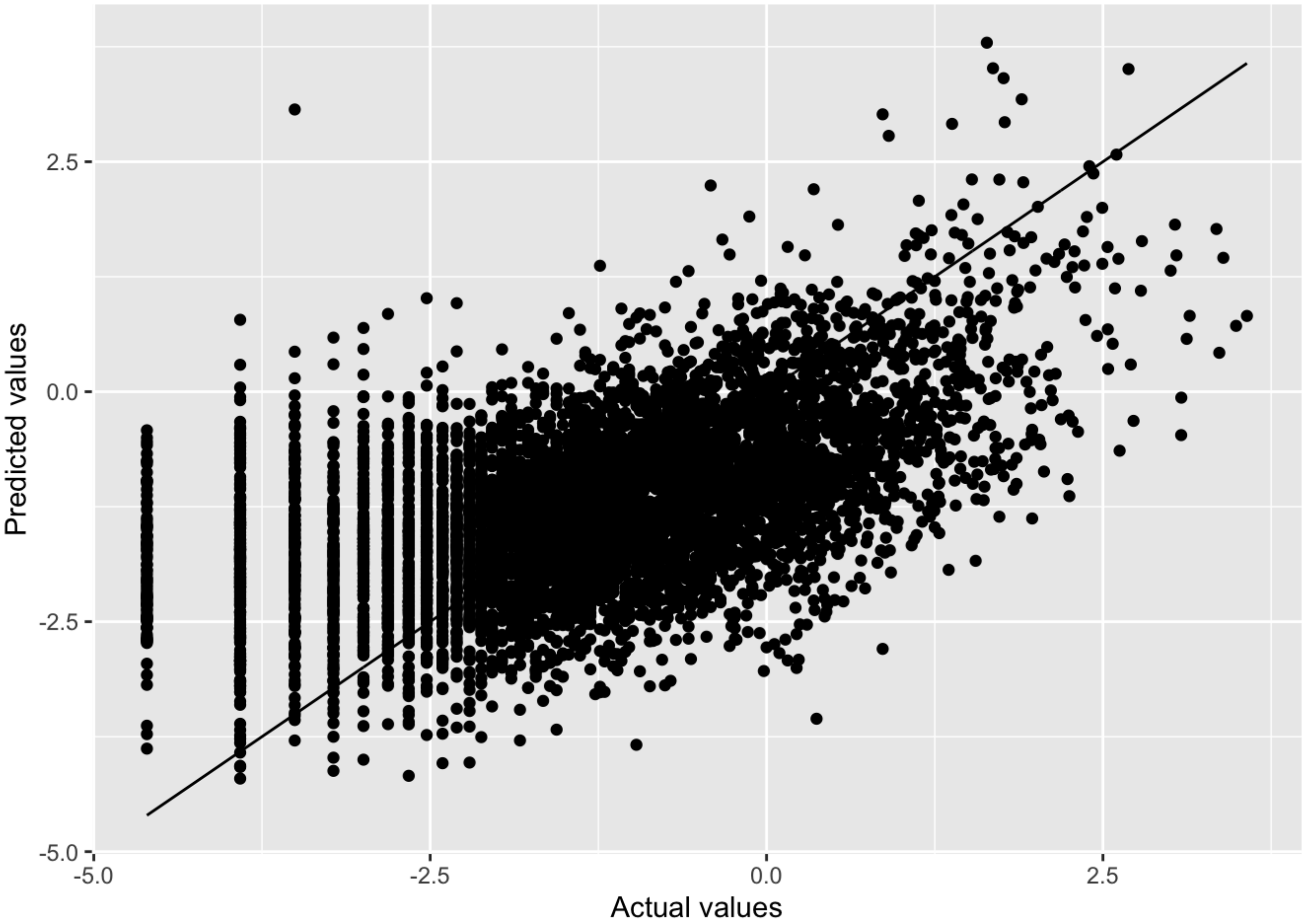
summary of linear regression model r^2: 0.5316

```
summary(model_lm)
```

```
## 
## Call:
## lm(formula = .outcome ~ ., data = dat)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -3.5492 -0.6550  0.0338  0.7429  4.2654 
## 
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)    
## (Intercept)         98.1741865 24.1566937   4.064 5.41e-05 ***
## Critic_Score         0.0224176  0.0043304   5.177 3.00e-07 ***
## User_Score          -0.0077227  0.0035801  -2.157 0.031359 *  
## GenreAdventure      -0.3436190  0.2611977  -1.316 0.188783    
## GenreFighting        0.0901682  0.2012717   0.448 0.654306    
## GenreMisc            0.4713842  0.2053026   2.296 0.021987 *  
## GenrePlatform        0.1350117  0.2261371   0.597 0.550690    
## GenrePuzzle         -0.5111645  0.4376765  -1.168 0.243268    
## GenreRacing         -0.5712306  0.1746461  -3.271 0.001128 ** 
## `GenreRole-Playing` -0.1108867  0.1657300  -0.669 0.503679    
## GenreShooter        -0.1578747  0.1460226  -1.081 0.280019    
## GenreSimulation      0.3908662  0.2308355   1.693 0.090878 .  
## GenreSports         -0.4196574  0.1762289  -2.381 0.017534 *  
## GenreStrategy       -1.5527957  0.2499323  -6.213 9.23e-10 ***
## Year_of_Release     -0.0504934  0.0120083  -4.205 2.97e-05 ***
## Critic_Count         0.0305187  0.0028713  10.629  < 2e-16 ***
## User_Count           0.0003169  0.0001085   2.921 0.003607 ** 
## `RatingE10+`        -0.2095308  0.1479506  -1.416 0.157184    
## RatingM             -0.6380651  0.1649593  -3.868 0.000121 ***
## RatingT             -0.4071647  0.1319754  -3.085 0.002120 ** 
## publisher_topTRUE    0.3950275  0.0936635   4.218 2.82e-05 ***
## developer_topTRUE    0.4678356  0.1379839   3.391 0.000739 ***
## num_of_platforms     0.0756197  0.0327896   2.306 0.021409 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.102 on 658 degrees of freedom
## Multiple R-squared:  0.4077, Adjusted R-squared:  0.3879 
## F-statistic: 20.59 on 22 and 658 DF,  p-value: < 2.2e-16
```

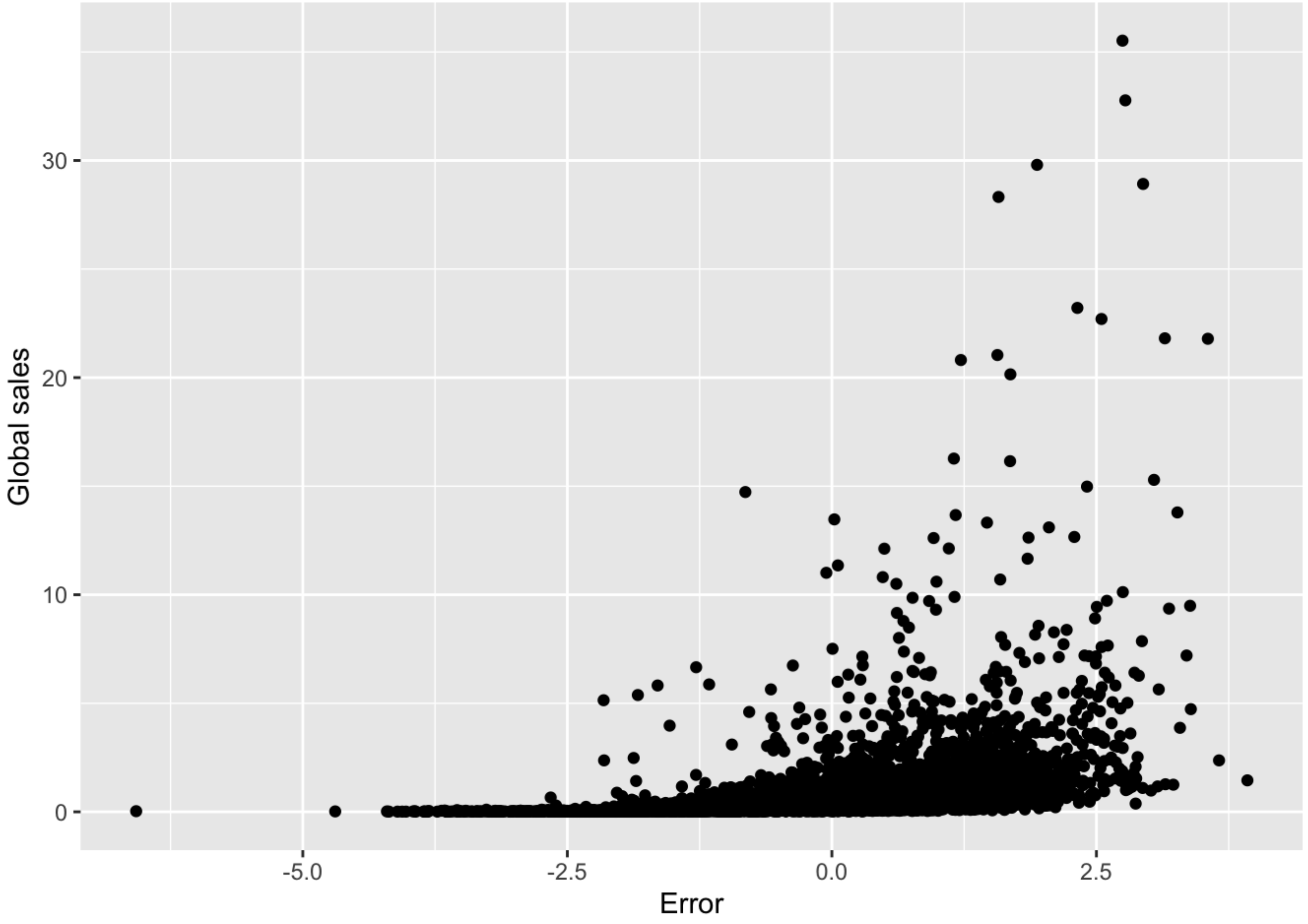
actual vs preds graph

```
ggplot(test_set) +
  geom_point(aes(log(Global_Sales), predicted_lm)) +
  geom_line(aes(log(Global_Sales), log(Global_Sales))) +
  xlab("Actual values") + ylab("Predicted values")
```



residual plot (error vs predicted) errors are largest for larger values of global sales - heteroskedacity present

```
ggplot(test_set) + geom_point(aes(log(Global_Sales) - predicted_lm, Global_Sales)) +
  xlab("Error") + ylab("Global sales")
```



SVM Linear model

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

# predicted value and RMSE
test_set$predicted_svm_linear <- predict(model_svm_linear, test_set)
rmse_results <- rmse_results %>%
  add_row(Method = "SVM Linear",
          RMSE = RMSE(log(test_set$Global_Sales),
                      test_set$predicted_svm_linear))
```

summary of SVM linear model

```
summary(model_svm_linear)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

SVM poly

might take several minutes to run because it is more mathematically complex (polynomial function)

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

# predicted value and RMSE
test_set$predicted_svm_poly <- predict(model_svm_poly, test_set)
rmse_results <- rmse_results %>%
  add_row(Method = "SVM Polynomial",
          RMSE = RMSE(log(test_set$Global_Sales),
                      test_set$predicted_svm_poly))
```

SVM poly model summary

```
summary(model_svm_poly)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

SVM radial

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

# predicted value and RMSE
test_set$predicted_svm_rad<- predict(model_svm_rad, test_set)
rmse_results <- rmse_results %>%
  add_row(Method = "SVM Radial",
          RMSE = RMSE(log(test_set$Global_Sales),
                      test_set$predicted_svm_rad))
```

SVM Radial summary

```
summary(model_svm_rad)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

L1 - lasso model

```
model_l1 <- train(log(Global_Sales) ~ Critic_Score +
                    User_Score + Genre +
                    Year_of_Release +  Critic_Count +
                    User_Count + Rating +
                    publisher_top + developer_top +
                    num_of_platforms, method = "lasso", data = train_set)
```

```
## Warning: model fit failed for Resample24: fraction=0.9 Error in elasticnet::enet(as.m
atrix(x), y, lambda = 0, ...) :
##   Some of the columns of x have zero variance
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
# predicted values and RMSE
test_set$predicted_l1 <- predict(model_l1, test_set)
rmse_results <- rmse_results %>% add_row(Method = "L1 Lasso",
                                         RMSE = RMSE(log(test_set$Global_Sales),
                                                     test_set$predicted_l1))
```

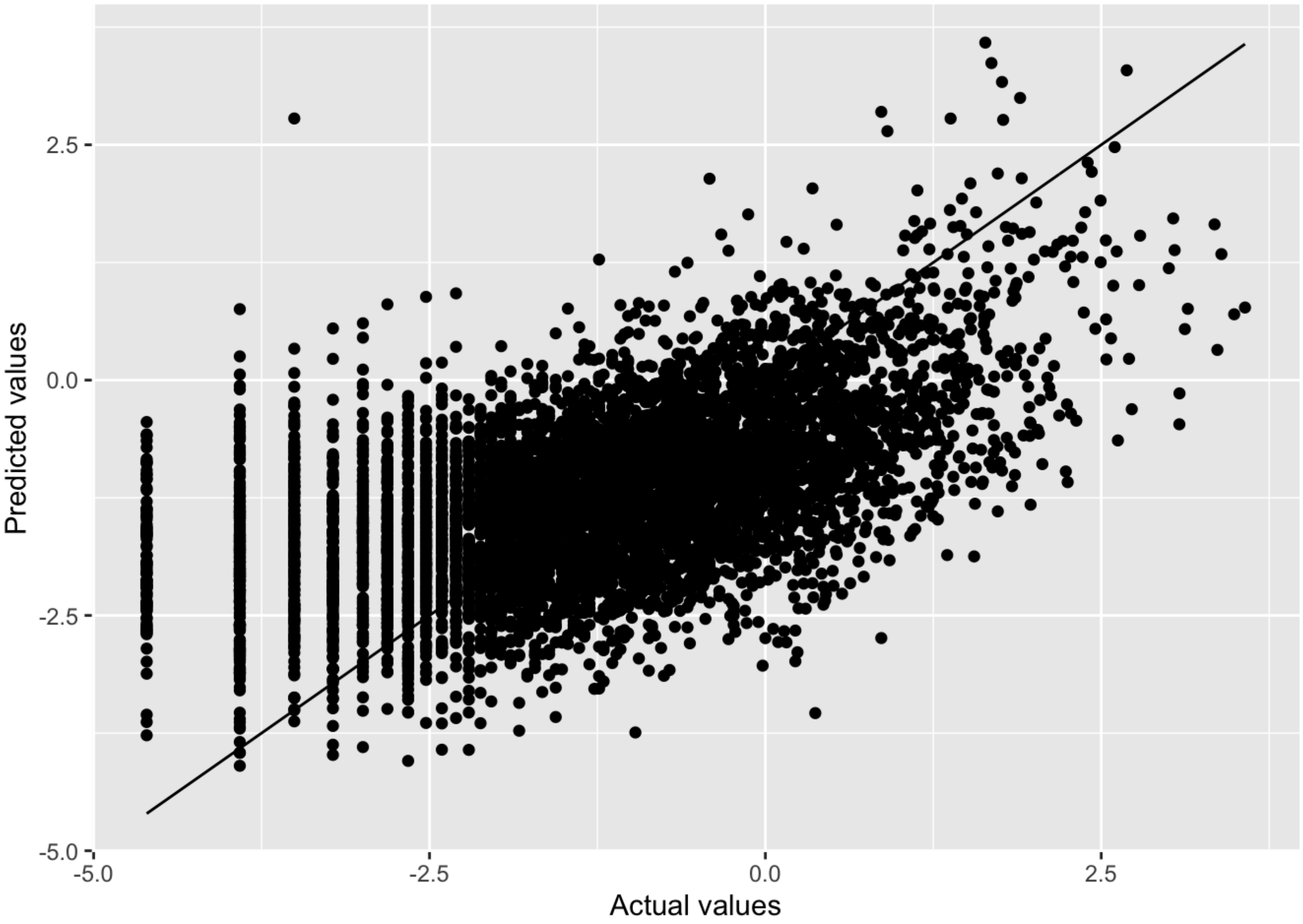
summary of lasso

```
summary(model_l1)
```

```
##              Length Class      Mode     
## call           4    -none-     call     
## actions       23    -none-     list     
## allset        22    -none-     numeric  
## beta.pure    506    -none-     numeric  
## vn            22    -none-     character
## mu             1    -none-     numeric  
## normx         22    -none-     numeric  
## meanx         22    -none-     numeric  
## lambda         1    -none-     numeric  
## L1norm        23    -none-     numeric  
## penalty       23    -none-     numeric  
## df            23    -none-     numeric  
## Cp            23    -none-     numeric  
## sigma2         1    -none-     numeric  
## xNames        22    -none-     character
## problemType    1    -none-     character
## tuneValue      1    data.frame list     
## obsLevels      1    -none-     logical  
## param          0    -none-     list     
```

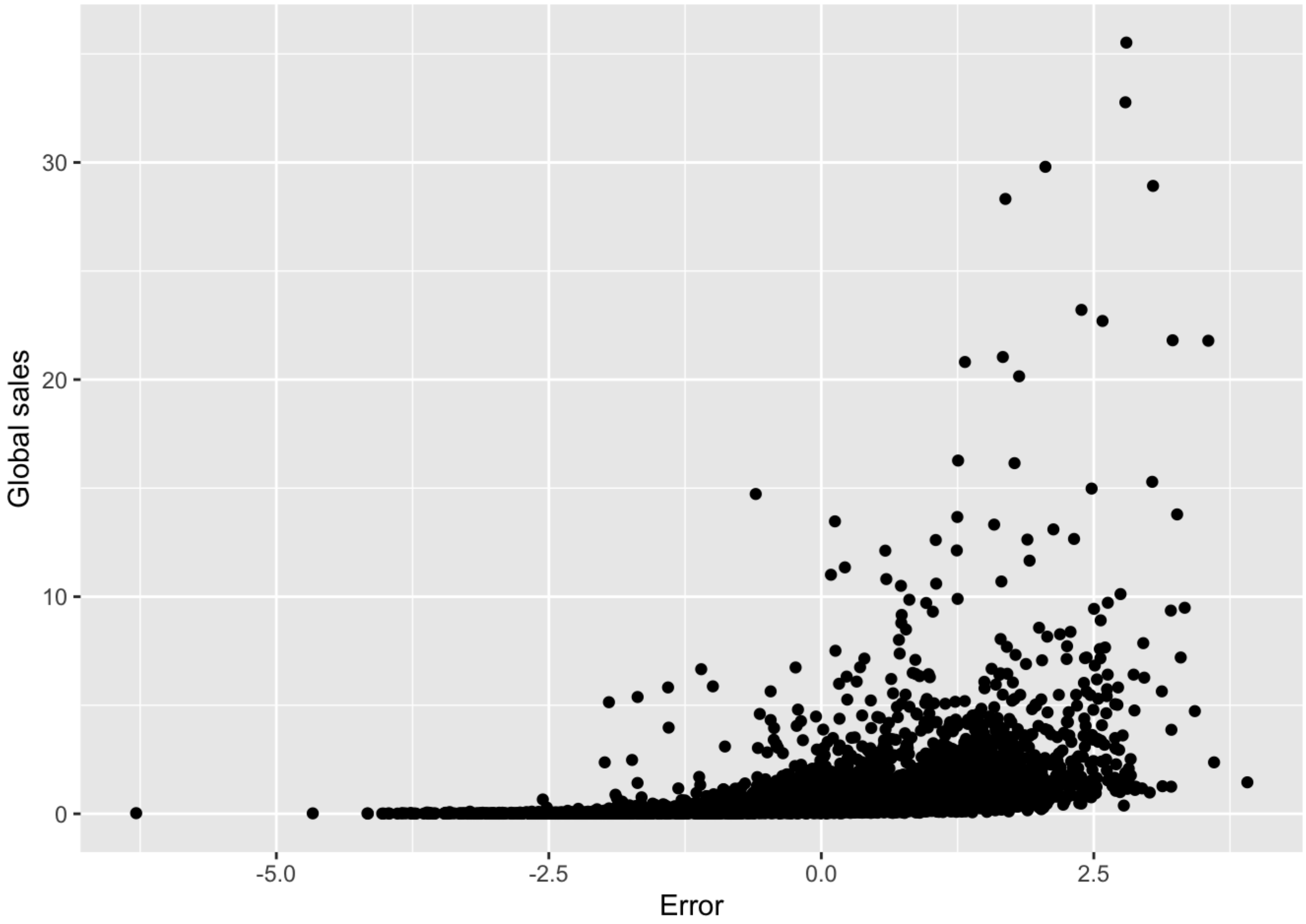
actual vs preds graph

```
ggplot(test_set) +
  geom_point(aes(log(Global_Sales), predicted_l1)) +
  geom_line(aes(log(Global_Sales), log(Global_Sales))) +
  xlab("Actual values") + ylab("Predicted values")
```



error vs sales

```
ggplot(test_set) + geom_point(aes(log(Global_Sales) - predicted_l1, Global_Sales)) +
  xlab("Error") + ylab("Global sales")
```



L2 - ridge model

```
model_l2 <- train(log(Global_Sales) ~ Critic_Score +
                    User_Score + Genre +
                    Year_of_Release +  Critic_Count +
                    User_Count + Rating +
                    publisher_top + developer_top +
                    num_of_platforms, method = "ridge", data = train_set)

# predicted values and RMSE
test_set$predicted_l2 <- predict(model_l2, test_set)
rmse_results <- rmse_results %>% add_row (Method = "L2 Ridge",
                           RMSE = RMSE(log(test_set$Global_Sales), test_set$predicted_l
2))
```

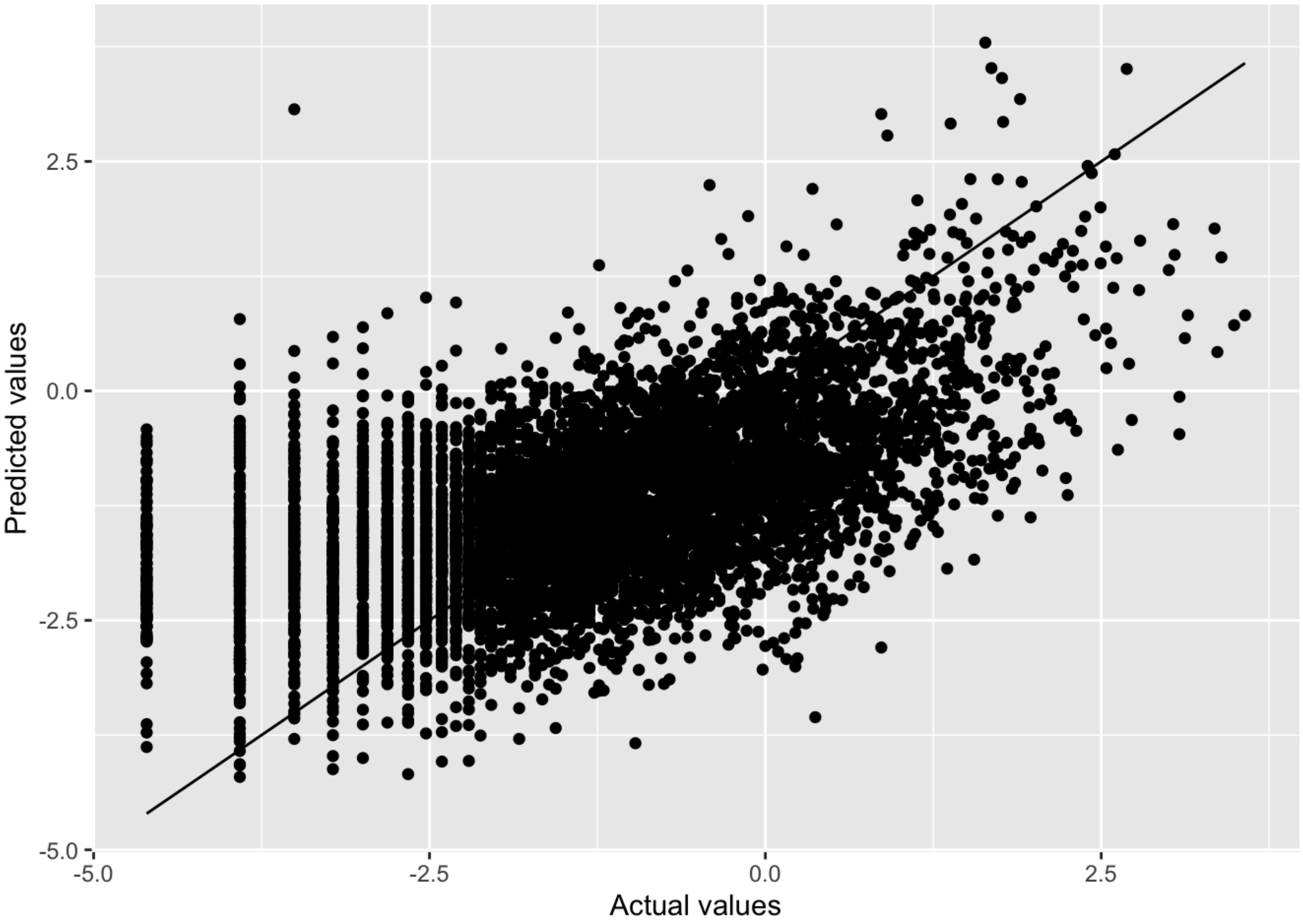
l2 model summary

```
summary(model_l2)
```

```
##              Length Class      Mode     
## call           4    -none-     call     
## actions       23    -none-     list     
## allset        22    -none-     numeric  
## beta.pure    506    -none-     numeric  
## vn            22    -none-     character
## mu             1    -none-     numeric  
## normx         22    -none-     numeric  
## meanx         22    -none-     numeric  
## lambda         1    -none-     numeric  
## L1norm        23    -none-     numeric  
## penalty       23    -none-     numeric  
## df            23    -none-     numeric  
## Cp            23    -none-     numeric  
## sigma2         1    -none-     numeric  
## xNames        22    -none-     character
## problemType    1    -none-     character
## tuneValue      1    data.frame list     
## obsLevels      1    -none-     logical  
## param          0    -none-     list
```

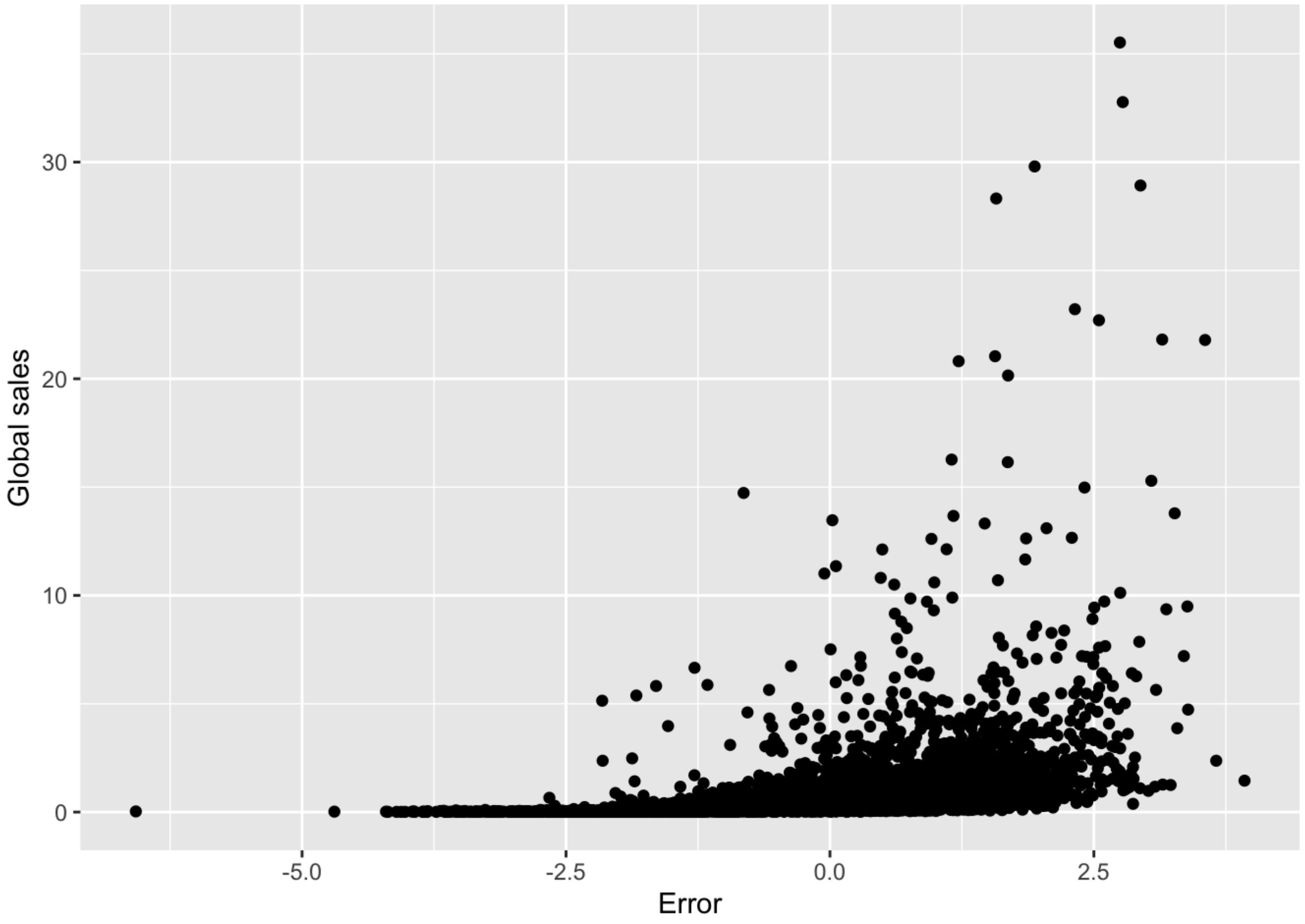
error vs preds

```
ggplot(test_set) +
  geom_point(aes(log(Global_Sales), predicted_l2)) +
  geom_line(aes(log(Global_Sales), log(Global_Sales))) +
  xlab("Actual values") + ylab("Predicted values")
```



error vs sales

```
ggplot(test_set) + geom_point(aes(log(Global_Sales) - predicted_l2, Global_Sales)) +
  xlab("Error") + ylab("Global sales")
```



random forest

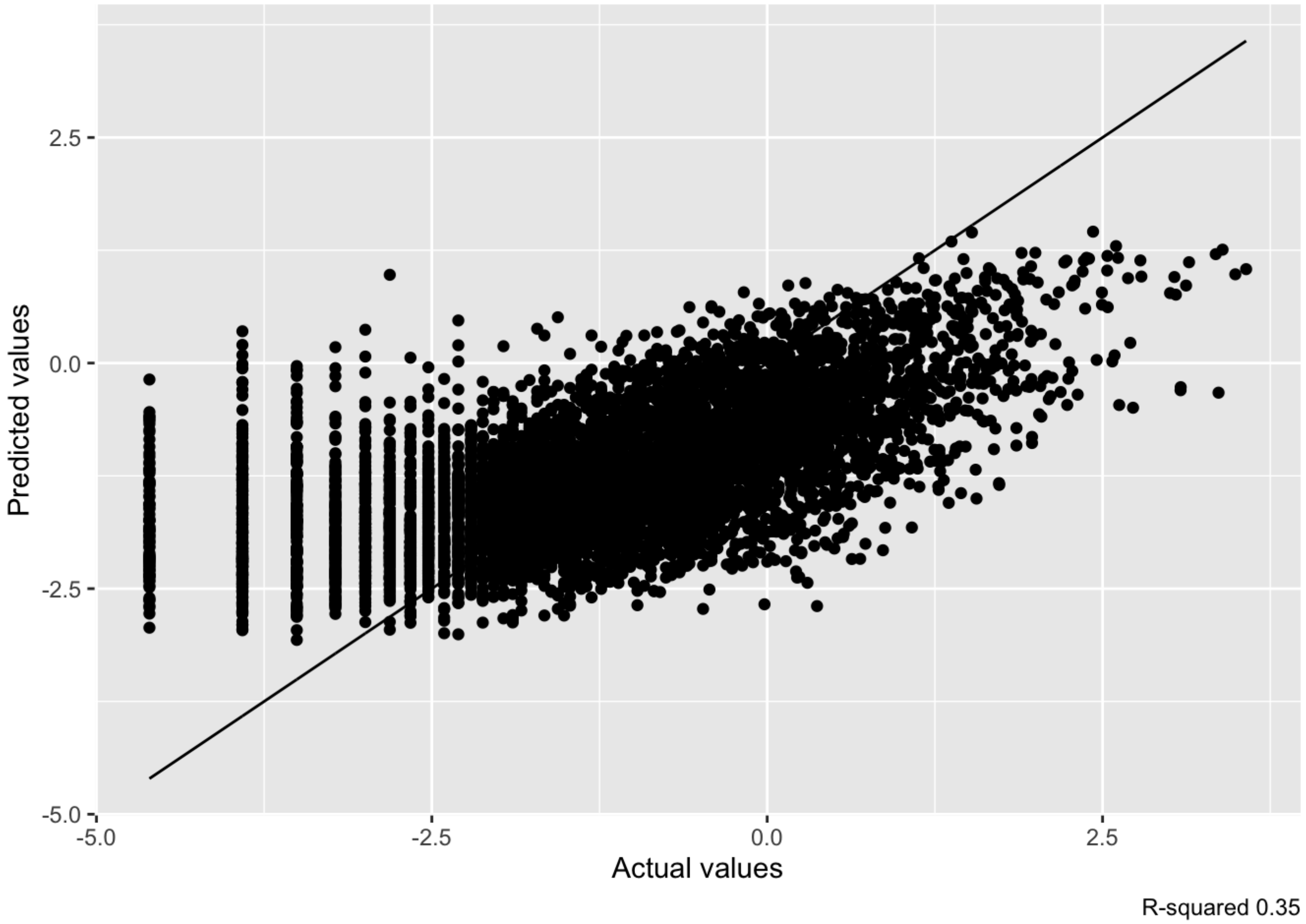
this one will also take a few minutes to run

```
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 + Critic_Count +
                    User_Count + Rating +
                    publisher_top + developer_top +
                    num_of_platforms, data = train_set,
                  method = "ranger", trControl = cntrl,
                  tuneGrid = tunegrid)

# predicted and RMSE
test_set$predicted_rf <- predict(model_rf, test_set)
rmse_results <- rmse_results %>% add_row(Method = "Random Forest",
                RMSE = RMSE(log(test_set$Global_Sales), test_set$predicted_rf))
```

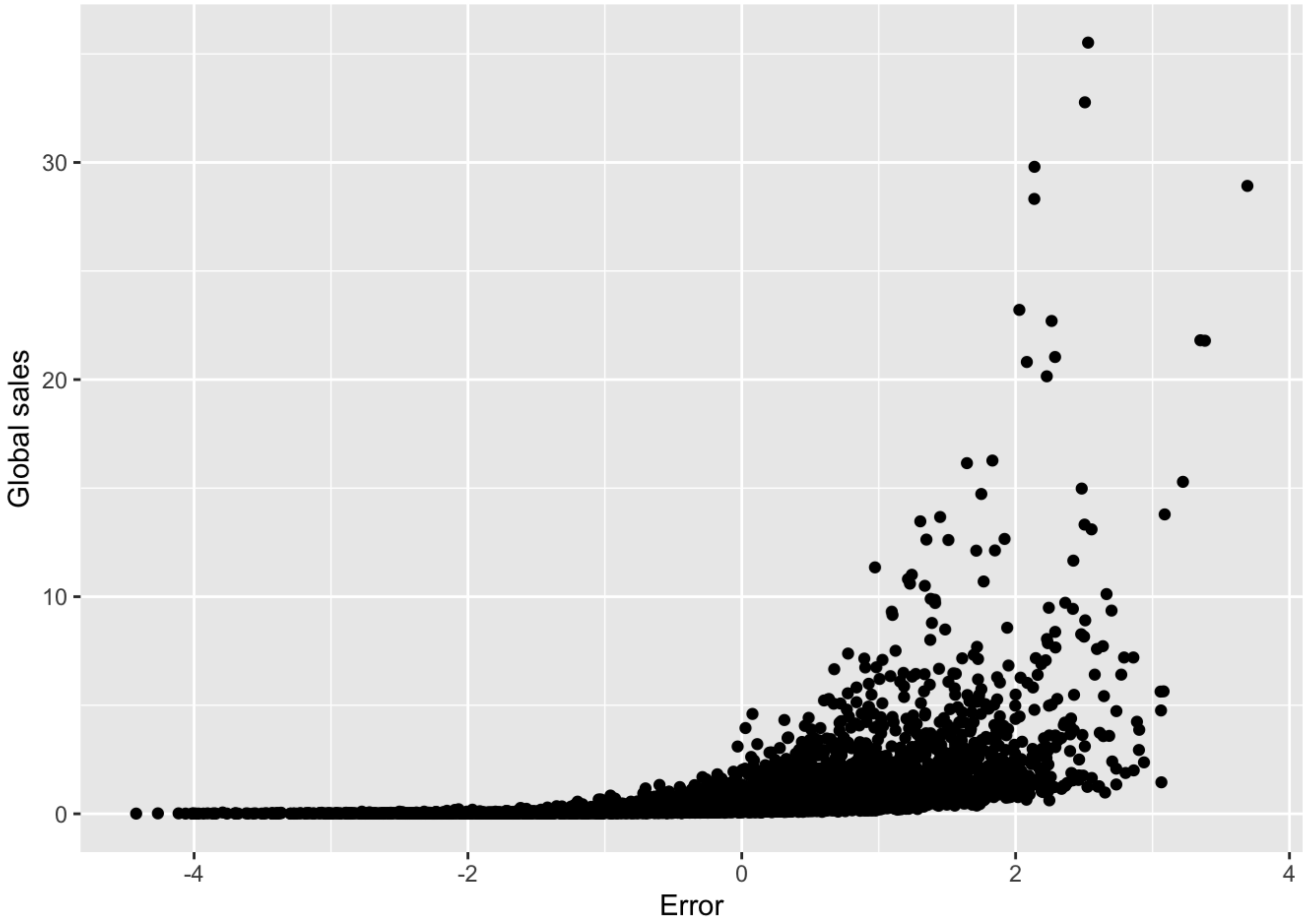
actual vs preds

```
ggplot(test_set) +
  geom_point(aes(log(Global_Sales), predicted_rf)) +
  geom_line(aes(log(Global_Sales), log(Global_Sales))) +
  xlab("Actual values") + ylab("Predicted values") +
  labs(caption =
         paste("R-squared",
               format(model_rf$finalModel$r.squared,
                      digits = 2)))
```



error vs global

```
ggplot(test_set) + geom_point(aes(log(Global_Sales) - predicted_rf, Global_Sales)) +
  xlab("Error") + ylab("Global sales")
```



compare the RMSE values of each model

```
rmse_results
```

```
##              Method     RMSE
## 1 Linear Regression 1.165805
## 2        SVM Linear 1.193608
## 3    SVM Polynomial 1.173235
## 4        SVM Radial 1.183304
## 5          L1 Lasso 1.159685
## 6          L2 Ridge 1.165798
## 7     Random Forest 1.092527
```

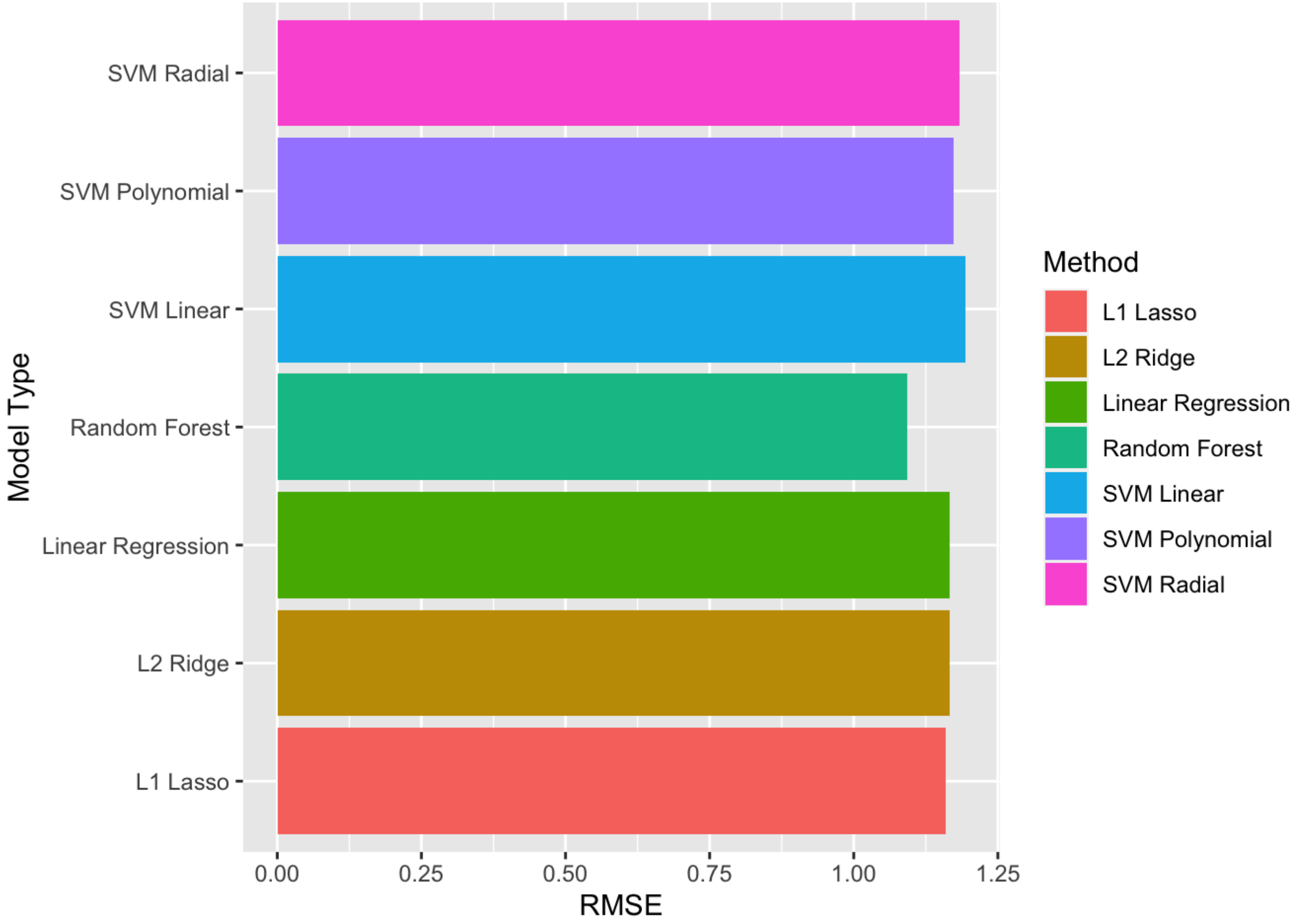
plotting and comparing all the models RMSE’s

random forest did best!

```
rmse_plot <- ggplot(rmse_results, aes(x=RMSE,y=Method, fill = Method))+geom_bar(stat="id
entity")+
  xlab("RMSE") + ylab("Model Type")
theme(text = element_text(size=10),
      legend.position="right",
      axis.text.x=element_text(angle = 90,vjust = 0.5,hjust = 1,size=8))
```

```
## List of 3
##  $ text           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : num 10
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : num 8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.position: chr "right"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi FALSE
##  - attr(*, "validate")= logi TRUE
```

```
rmse_plot
```


Model Type
SVM Radial
SVM Polynomial
SVM Linear
Random Forest
Linear Regression
L2 Ridge
L1 Lasso
0.00
0.25
0.50
0.75
1.00
1.25
RMSE
Method
L1 Lasso
L2 Ridge
Linear Regression
Random Forest
SVM Linear
SVM Polynomial
SVM Radial