The Fatal Empire Sales Prediction

Summary to the CEO

We have been tasked with building a predictive model to forecast the quantity of sales, within the North American Market, for “The Fatal Empire” video game. It must be noted that a video games market performance in three key regions (Europe, Japan, and Other) all play a varying role in terms of a games success in the North American market. Notably, market performance in “other” regions significantly influences North American Market success. Additionally, a video games genre, platform release, and global market inclusivity are all influential factors but to a lesser degree. Therefore, based upon “The Fatal Empire” being a role-playing game released on PS4 and the current market performance, selling 2.58 million copies in Japan, 0.53 million copies in Europe, and 0.1 million copies in other parts of the world, our model forecasts the sale

of 0.526838 million copies (526,838) in the North American Market. However, we can expect that value to vary by 240,000 copies on average.

Managerial Overview

We are tasked with building a regression model to predict the expected quantity of sales of “The Fatal Empire” video game in the North American market based upon six predictor variables. We have chosen to compare two different predictor models. Since our task is a supervised learning problem, meaning our data set contains labelled North American sales data, the chosen model is the one that most accurately predicts North American sales compared to its true label.

Our data set originally contained 9 variables, 4 numeric (JP_Sales, NA_Sales, EU_Sales, and Other_Sales) and 5 character variables (Name, Year, Publisher, Genre, and Platform). Our data set alterations came in the form of removing uninformative variables such as Name, Year, and Publisher. Since a games “Name” is just a form of identification and a games “Publisher” simply acts as a subgroup of a video games “Name” these variable would not be informative to the analysis. Additionally, because Platforms are released in sequential order corresponding to specific Years, a Year variable does not provided any additional information than the Platform variable already does. Furthermore, our data set contains Platforms that have either been discontinued, meaning no further games will be made for those consoles and newer games will not be compatible with their operating systems, or have given way for the evolution of newer Platforms within the company’s console offerings. Therefore, removing these Platforms creates a more up to date data set accurately reflecting current market trends which is reflected in our model. Notably, the 9 chosen Platforms are the only Platforms still in the current market or are anticipated for a resurgence according to Platform providers. Finally, market trends such as ‘hype’, popularity, and exposure of games drive sales. Exclusive market games or games not sold globally lacks this exposure and could potentially impact local and global market sales. Therefore, we created a “global” market variable.

The relationship between our numeric variables and NA_Sales can be determined by examining the strength of their relationship. Specifically, a change in one of our numeric variables is generally associated with a change in a specific direction in our NA_Sales variable. We see a strong positive relationship between NA_Sales and Other_Sales meaning when the value of Other_Sales increases, the value of the NA_Sales tends to increase as well. Additionally, we see a moderate positive relationship between NA_Sales and EU_Sales meaning when the value of EU_Sales increases, the value of the NA_Sales tends to increase as well but this relationship is not as strong as compared to Other_Sales. Finally, we see a small weak negative relationship between NA_Sales and JP_Sales meaning when the value of JP_Sales increases, the value of the NA_Sales does not follow suite as there is no relationship between the two variables. Furthermore, The relationship between categorical variables and our NA_Sales can be examined through visualizations and statistical tests. Notably, our Platform, Genre, and Global variables all showed signs of having an influential relationship with NA_Sales.

By visualizing our data through paralell coordinate plots we discovered some natural groupings within our genre and platform variables. We evidenced some segregation in NA_Sales in terms of the genre category with the shooter and role-playing genre seemingly distinguishable from sports and strategy. Additionally, we evidenced some segregation in JP_Sales in terms of the platform category with the 3DS and DS seemingly distinguishable from the playstation platforms. However, any natural groupings seems to be driven by outliers and the imbalance of observations within our genre and platform variable. Additionally, the relatively moderate cardinality of these two variables makes parallel coordinate plots very hard to visualize and interpret (only through genre filtering by color scale are groupings visable).

Through Principal Component analysis we are able to understand which variables contribute the most to the variation within our data. Notably, EU_Sales, Other_Sales, Global_yes and JP_Sales account for the most variation of our data (within first component) and are additionally seen as being within the top 6 most influential variables in our final model. Subsequently, our PCA plot seems to illustate some separation of our platforms (DS and Playstation Platoforms within first two components). However, since PCA is a dimentionality reduction technique and we would only lose 5 components in order to explain at least 90% of the variation we have chosen to forgo using PCA for modelling purposes.

Pre-processing our data is an essential step towards ensuring our model runs smoothly and more accurately. Since our models require only numeric data we first had to convert our categorical variables into numeric data (dummy encoding) and ensure all our predictors were on the same scale (normalization).

Our lasso regression model could only explain 66% of the variance in our NA_Sales and on average made an error of 367,000 copies in terms of predicting our NA_Sales. In contrast, our random forest regression model could explain 85.3% of the variance in our NA_Sales and on average only made an error of 250,000 copies in terms of predicting our NA_Sales. Since our random forest regression model explains NA_Sales more accurately and makes smaller prediction errors in terms of copies sold it is clearly our optimal model.

In terms of evaluating our random forest regression model on unseen test data, our model now explains 87.6% of the variance in NA_Sales_sales and on average only makes an error of 240,000 in terms of predicting NA_Sales. Notably, it seems our model is performing even better on unseen data which is exactly what we wanted to see. However, it must be noted that our model seems to perform relatively well in predicting our lower value NA_Sales but progressively worse on average at predicting our higher valued NA_Sales.

Based upon "The Fatal Empire" being a role-playing game released on PS4 and the current market performance, selling 2.58 million copies in Japan, 0.53 million copies in Europe, and 0.1 million copies in other parts of the world, our model forecasts the sale of 0.526838 million copies (526,838) in the North American Market. However, we can expect that value to vary by 240,000 copies on average.

Detailed Analysis

Data Cleaning

Importing libraries

```
pacman::p_load(tidyverse, rio,ggcorrplot, reshape2,olsrr,lmtest,car, skimr,
               naniar,e1071, tidycomm, janitor,corrplot,ggpubr, rmarkdown,
               rstatix,GGally,ggplot2,viridis,tidyr,tidymodels,parsnip,
               factoextra,devtools,ggbiplot,rsample,Boruta,caret,glmnet,vip,
               dataMeta,explore, knitr,dplyr,GGally,pheatmap,gt,tinytex)
```

Data dictionary

```
variable_type <- c("character","Factor","Factor","Factor","character",
                   "Numeric","Numeric","Numeric","Numeric","Factor")

variable_description <- c("The video games name","Platform of the games release",
                          "Year of the game's release", "Genre of the game",
                          "Publisher of the game",
                          "North America sales (in millions)",
                          "Sales in Europe (in millions)",
                          "Sales in Japan (in millions)",
                          "Rest of world sales (in millions)",
                          "Whether sold globally")

variable_names <- c("Name","Platform","Year","Genre",
```

```r
"Publisher","NA_Sales","EU_Sales","JP_Sales","Other_Sales","*Global*")

variable_details <- c("Call of Duty","Playstation, Xbox, etc",
                      "Released  in 2015", "Shooting, Racing, etc",
                      "Nintendo, PLaystation",
                      "Units sold in N.America not revenue",
                      "Units sold in EU not revenue",
                      "Units sold in Japan not revenue",
                      "Units sold rest of world not revenue",
                      "'No' if sold exclusively in certain regions")

data.dict <- data.frame(
  "Variable" = variable_names,
  "Type" = variable_type,
  "Description" = variable_description,
  "Details" = variable_details
)
data.dict%>%
gt()%>%
  tab_header(
    title = "Data Dictionary",
    subtitle = "Videogames Data Set"
  )
```

Data Dictionary
Videogames Data Set

Variable	Type	Description	Details
Name	character	The video games name	Call of Duty
Platform	Factor	Platform of the games release	Playstation, Xbox, etc
Year	Factor	Year of the game's release	Released in 2015
Genre	Factor	Genre of the game	Shooting, Racing, etc
Publisher	character	Publisher of the game	Nintendo, PLaystation
NA_Sales	Numeric	North America sales (in millions)	Units sold in N.America not revenue
EU_Sales	Numeric	Sales in Europe (in millions)	Units sold in EU not revenue
JP_Sales	Numeric	Sales in Japan (in millions)	Units sold in Japan not revenue
Other_Sales	Numeric	Rest of world sales (in millions)	Units sold rest of world not revenue
Global	Factor	Whether sold globally	'No' if sold exclusively in certain regions

Read in csv as tibble using rio

```r
games.df <- rio::import('/Users/racharon/Desktop/Other/vgsales.csv')%>%
  as_tibble()
```

Display first 6 rows using head

```r
head(games.df)%>%
  dplyr::select(-Name)%>%
  gt()%>%
```

```
  tab_header(
    title = "Videogames Data Set",
    subtitle = "First 6 rows of data presented"
  )
```

Videogames Data Set
First 6 rows of data presented

Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales
Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46
NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77
Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31
Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96
GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1.00
GB	1989	Puzzle	Nintendo	23.20	2.26	4.22	0.58

Skimming our Data

```
skim(games.df)
```

Data summary

Name	games.df
Number of rows	16598
Number of columns	9
Column type frequency:	
character	5
numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Name	0	1	1	132	0	11493	0
Platform	0	1	2	4	0	31	0
Year	0	1	3	4	0	40	0
Genre	0	1	4	12	0	12	0
Publisher	0	1	3	38	0	579	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
NA_Sales	0	1	0.26	0.82	0	0	0.08	0.24	41.49	
EU_Sales	0	1	0.15	0.51	0	0	0.02	0.11	29.02	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
JP__Sales	0	1	0.08	0.31	0	0	0.00	0.04	10.22	
Other__Sales	0	1	0.05	0.19	0	0	0.01	0.04	10.57	

Our data set is comprised of 16,598 observations and 9 variables. Of our 9 variables 4 are numeric (JP__Sales, NA__Sales, EU__Sales, and Other__Sales), all being read in correctly in their numeric form, with the remaining 5 being read in as character data types (Name, Year, Publisher, Genre, and Platform). However, we would expect Year, Genre, and Platform to be read in as factors and as such will require conversion. We are satisfied with Name and Publisher being read in as character data types due to their high-cardinality if converted into factors and their likelihood of being dropped. Notably, NA__Sales is our outcome/independent variable with the remaining variables acting as our explanatory variables.

Our data set spans over 40 years (39 plus "N/A"), comprising of 11,493 unique video games ("Names"). This indicates that our data set contains observations for certain "Names" across multiple platforms of which there are 31 in total. Additionally, these video games, created by 579 unique Publishers, can be subdivided into 12 unique genres.

Despite our skimmed data indicating an absence of null values, the inspection of our Year and Publisher variables paint a different story. For our Year variable we instantly notice a min of 3 characters within our summary statistic meaning we either have erroneous observations within our data set or null values have been read in as strings i.e., "N/A". Notably, the latter correctly answers our assumption for both Year and Publisher. Our year column contains 272 "N/A" values. Additionally, our Publisher column contains 58 "N/A" values with 203 "Unknown" publishers within this column. We could convert the "Unknown" data to null values or convert our null values to "Unknown" as there could be a correlation between NA__Sales and the absence of a Publisher. However, the Year and Publisher columns will likely be dropped.

Missing and Unknown Observations

For Year & Publisher variable

Variable	N/A	Unknown
Year	271	0
Publisher	58	203

Univariate Quantitative Variable Analysis

```
EU.zero <- games.df%>%
  filter(!(EU_Sales==0),
         JP_Sales == 0,
         Other_Sales == 0,
         NA_Sales == 0)%>%
  dplyr::count()

JP.zero <- games.df%>%
  filter(!(JP_Sales==0),
         EU_Sales == 0,
         Other_Sales == 0,
         NA_Sales == 0)%>%
  dplyr::count()

Other.zero <- games.df%>%
  filter(!(Other_Sales==0),
         JP_Sales == 0,
```

```
         NA_Sales == 0,
         EU_Sales == 0)%>%
  dplyr::count()

NA.zero <- games.df%>%
  filter(!(NA_Sales==0),
         JP_Sales == 0,
         Other_Sales == 0,
         EU_Sales == 0)%>%
  dplyr::count()

exclusive.sales <- tribble(
  ~name, ~exclusive,
  'NA_Sales', NA.zero$n,
  'EU_Sales',EU.zero$n,
  'JP_Sales',JP.zero$n,
  'Other_Sales', Other.zero$n
)

#Skewness & Kurtosis
skew.table <- tribble(
  ~name, ~skewness,
  'NA_Sales', skewness(games.df$NA_Sales),
  'EU_Sales',skewness(games.df$EU_Sales),
  'JP_Sales',skewness(games.df$JP_Sales),
  'Other_Sales', skewness(games.df$Other_Sales)
)

stable <-games.df%>%
  dplyr::select(NA_Sales:Other_Sales)%>%
  pivot_longer(NA_Sales:Other_Sales)%>%
  desc_statby(measure.var = "value",
              grps = "name")

EU.outliers <- games.df%>%
  filter(EU_Sales>0.165)%>%
  dplyr::count()

NA.outliers <- games.df%>%
  filter(NA_Sales>0.36)%>%
  dplyr::count()

JP.outliers <- games.df%>%
  filter(JP_Sales>0.06)%>%
  dplyr::count()

Other.outliers <- games.df%>%
  filter(Other_Sales>0.06)%>%
  dplyr::count()

EU.zero <- games.df%>%
  filter(EU_Sales==0)%>%
  dplyr::count()
```

```r
NA.zero <- games.df%>%
  filter(NA_Sales==0)%>%
  dplyr::count()

JP.zero<- games.df%>%
  filter(JP_Sales==0)%>%
  dplyr::count()

Other.zero <- games.df%>%
  filter(Other_Sales==0)%>%
  dplyr::count()


stable <- stable[, c("name", "mean","median")]%>%as_tibble()
stable <- stable%>%add_column(outliers = c(EU.outliers$n,JP.outliers$n,
                                           NA.outliers$n,Other.outliers$n),
  zero = c(EU.zero$n,JP.zero$n,NA.zero$n,Other.zero$n))

stable <- merge(stable, skew.table,by = "name")
stable <- merge(stable, exclusive.sales,by = "name")
colnames(stable)[1] <- 'region'

stable <- stable%>%
  as_tibble()

stable <- stable%>%
  gt()%>%
  tab_header(
    title = "Summary Statistics for Quantitative Variables",
    subtitle = "Analysis of Distributions"
  )

stable
```

Summary Statistics for Quantitative Variables

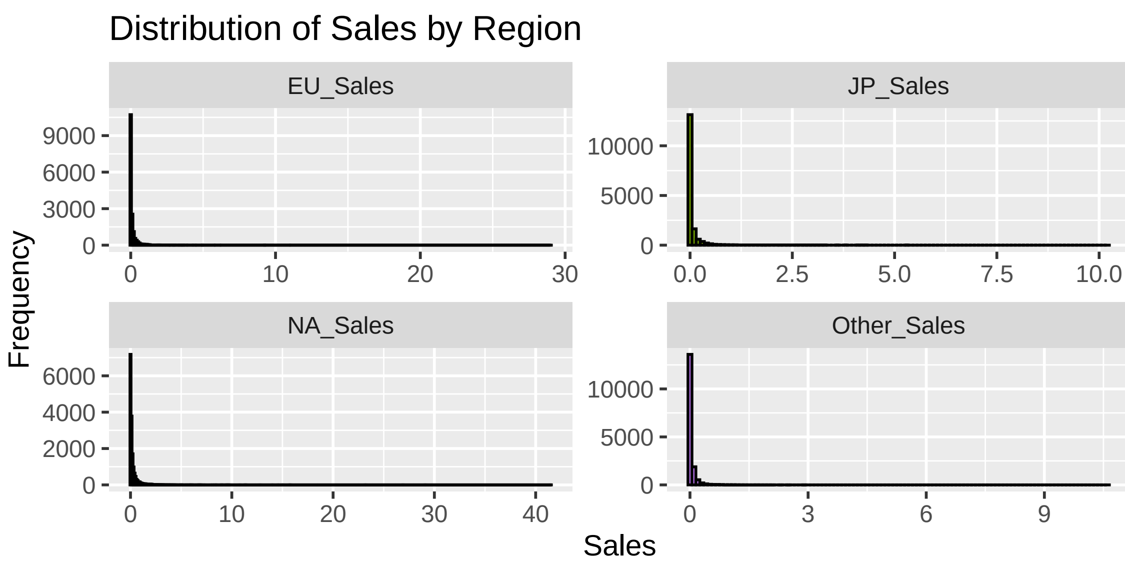
Analysis of Distributions

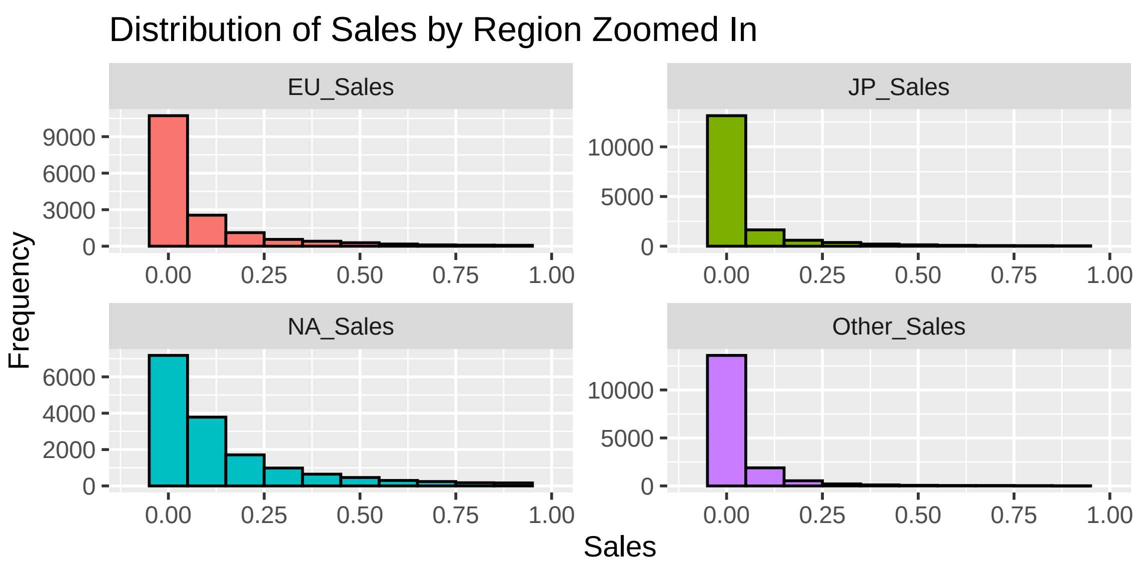
region	mean	median	outliers	zero	skewness	exclusive
EU_Sales	0.14665201	0.02	3158	5730	18.87212	545
JP_Sales	0.07778166	0.00	3175	10455	11.20443	3136
NA_Sales	0.26466743	0.08	2866	4499	18.79623	950
Other_Sales	0.04806302	0.01	2600	6477	24.22954	1

```r
hist.numeric <- games.df%>%
  dplyr::select(NA_Sales:Other_Sales)%>%
  pivot_longer(NA_Sales:Other_Sales)%>%
  ggplot(aes(x = value))+
  geom_histogram(col = "black",binwidth = 0.1, aes(fill = name),
                 show.legend = FALSE)+
  facet_wrap(~name,scales = "free")+
  labs(x = "Sales",y = 'Frequency', title = "Distribution of Sales by Region")
```

```r
hist.zoom <- games.df%>%
  dplyr::select(NA_Sales:Other_Sales)%>%
  pivot_longer(NA_Sales:Other_Sales)%>%
  ggplot(aes(x = value))+
  geom_histogram(col = "black",binwidth = 0.1, aes(fill = name),
                 show.legend = FALSE)+
  facet_wrap(~name,scales = "free")+
  xlim(-0.1, 1)+
  labs(x = "Sales",y = 'Frequency',
      title = "Distribution of Sales by Region Zoomed In")

ggarrange(hist.numeric,hist.zoom,
          nrow = 2)
```

```
## Warning: Removed 1681 rows containing non-finite values (stat_bin).

## Warning: Removed 8 rows containing missing values (geom_bar).
```

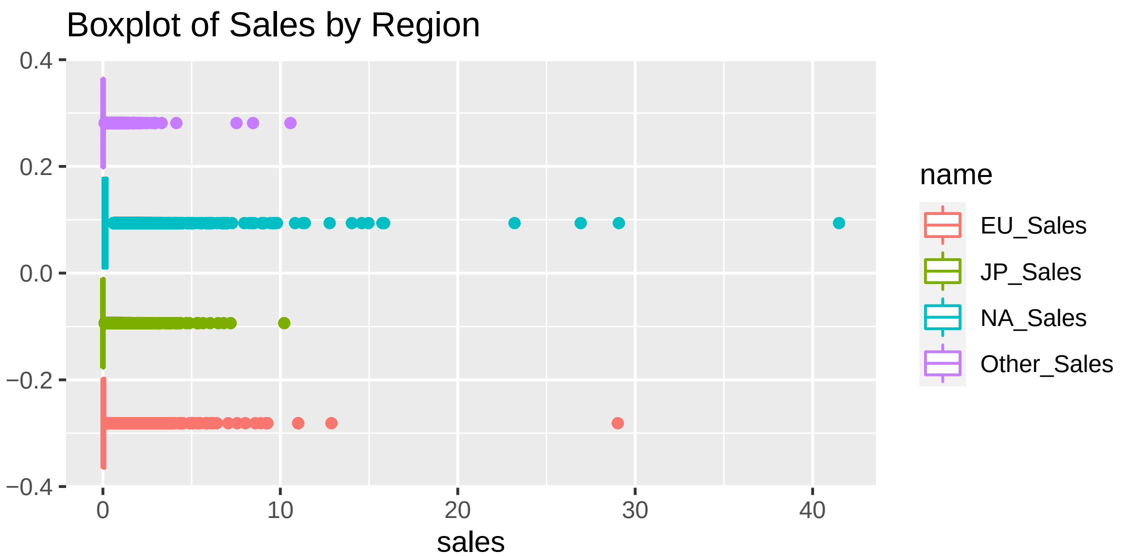


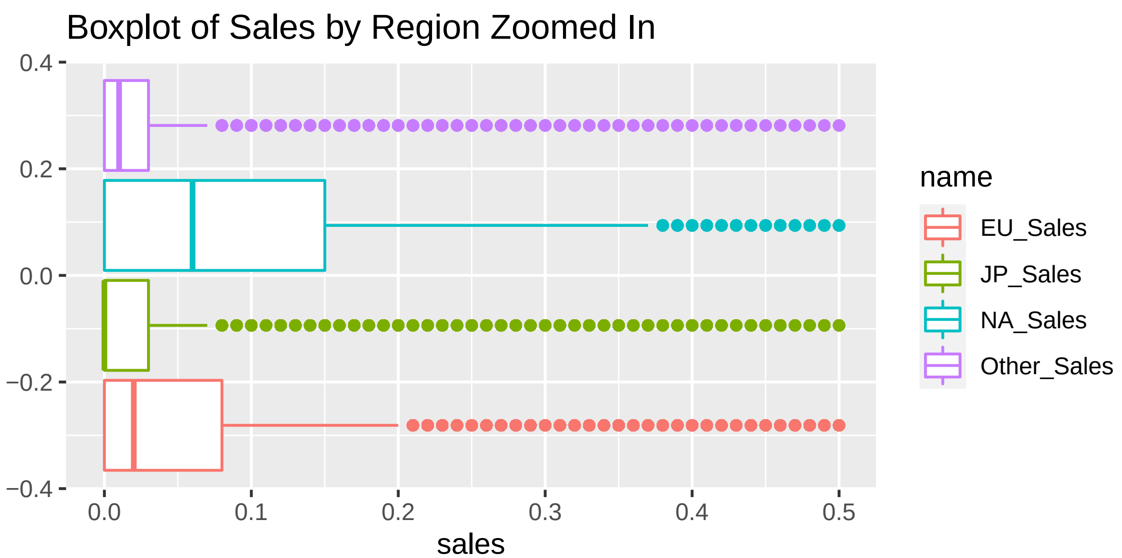



```
boxplot.zoomed <-games.df%>%
  dplyr::select(NA_Sales:Other_Sales)%>%
  pivot_longer(NA_Sales:Other_Sales)%>%
  ggplot(aes(x = value))+
  geom_boxplot(aes(color = name))+
  xlim(-0.001,0.5)+
  labs(x = "sales", title = "Boxplot of Sales by Region Zoomed In")

boxplot.spread <- games.df%>%
  dplyr::select(NA_Sales:Other_Sales)%>%
  pivot_longer(NA_Sales:Other_Sales)%>%
  ggplot(aes(x = value))+
  geom_boxplot(aes(color = name))+
  labs(x = "sales", title = "Boxplot of Sales by Region")

ggarrange(boxplot.spread,boxplot.zoomed,
```

```
          nrow = 2)
```

```
## Warning: Removed 3921 rows containing non-finite values (stat_boxplot).
```





Market Exclusivity

We know video games that depict drugs, sexual themes, or defamation of certain governments are almost always banned in specific regions. Additionally, various games are intended to be released exclusively in specific markets for various reasons. Therefore, it is inevitable that our data set contains certain games that are exclusively sold in specific markets or are banned in other markets (exclusive column in table above).

Our data set contains 545 observations/games (3.3% of our data) that were exclusively sold in the European market, 3126 observations (18.9% of our data) that were exclusively sold in the Japanese market, 950 observations (5.7% of our data) that were exclusively sold in the North American market, and 1 observation that was exclusively sold in other markets. Notably, this brings to light the potential correlation between games sold across all markets and NA_Sales which might be a potential feature engineering variable.

Summary Analysis

All four sales variables follow unimodal right skew distributions. More than 17% of each variables observations lie 1.5 times their upper Q3 indicating a significant amount of outliers present. We may want to consider the impact these outliers will have on our final model. Finally, there is a disproportional amount of zero values in each variable ranging from the lowest in NA_Sales of 27% to the highest in JP_Sales of 63%.

European Sales Analysis

Upon the inspection of our EU_Sales variable, we have a significant proportion of our observations at the lower end of the sales scale (0 and 0.1). We do see a disproportionate amount of sales in the zero bin (5730 amounting to 35%). This is because sales is obviously a bounded value, with zero being its lower bound. Additionally, we see mean sales (0.1467) greater than its median (0.02) resulting in EU_Sales following a right skewed unimodal distribution (18.9 skewness). Additionally, there are obvious outliers at the other end of the scale with 3,158 observations (19%) laying 1.5 times above Q3 as shown in the Boxplots graph.

Japan Sales Analysis

Upon the inspection of our JP_Sales variable, we have a significant proportion of our observations at the lower end of the sales scale (0 and 0.1). We do see a disproportionate amount of sales in the lower bound zero bin (10,455 amounting to 63%). Additionally, we see mean sales (0.078) greater than its median (0) resulting in JP_Sales following a right skewed unimodal distribution (11.2 skewness). Additionally, there are obvious outliers at the other end of the scale with 3,175 observations (19.13%) laying 1.5 times above Q3 as shown in the Boxplots graph.

Other Markets Sales Analysis

Upon the inspection of our Other_Sales variable, we have a significant proportion of our observations at the lower end of the sales scale (0 and 0.1). We do see a disproportionate amount of sales in the lower bound zero bin (6477 amounting to 39%). Resultantly, Other_Sales follows a right skewed unimodal distribution. Additionally, there are obvious outliers at the other end of the scale with 2,600 observations (17.2%) laying 1.5 times above Q3.

North America Sales Analysis

Upon the inspection of our NA_Sales variable, we have a significant proportion of our observations at the lower end of the sales scale (0 and 0.1). We do see a disproportionate amount of sales in the lower bound zero bin (4499 amounting to 27%). Resultantly, NA_Sales follows a right skewed unimodal distribution. Additionally, there are obvious outliers at the other end of the scale with 2,866 observations (17.3%) laying 1.5 times above Q3 as shown in the Boxplots graph.

Univariate Qualitative Variable Analysis

Platform Variable Analysis

```r
games.df <- games.df  %>% replace_with_na(replace = list(Year = "N/A"))

platform.table <- games.df%>%
  group_by(Platform)%>%
```

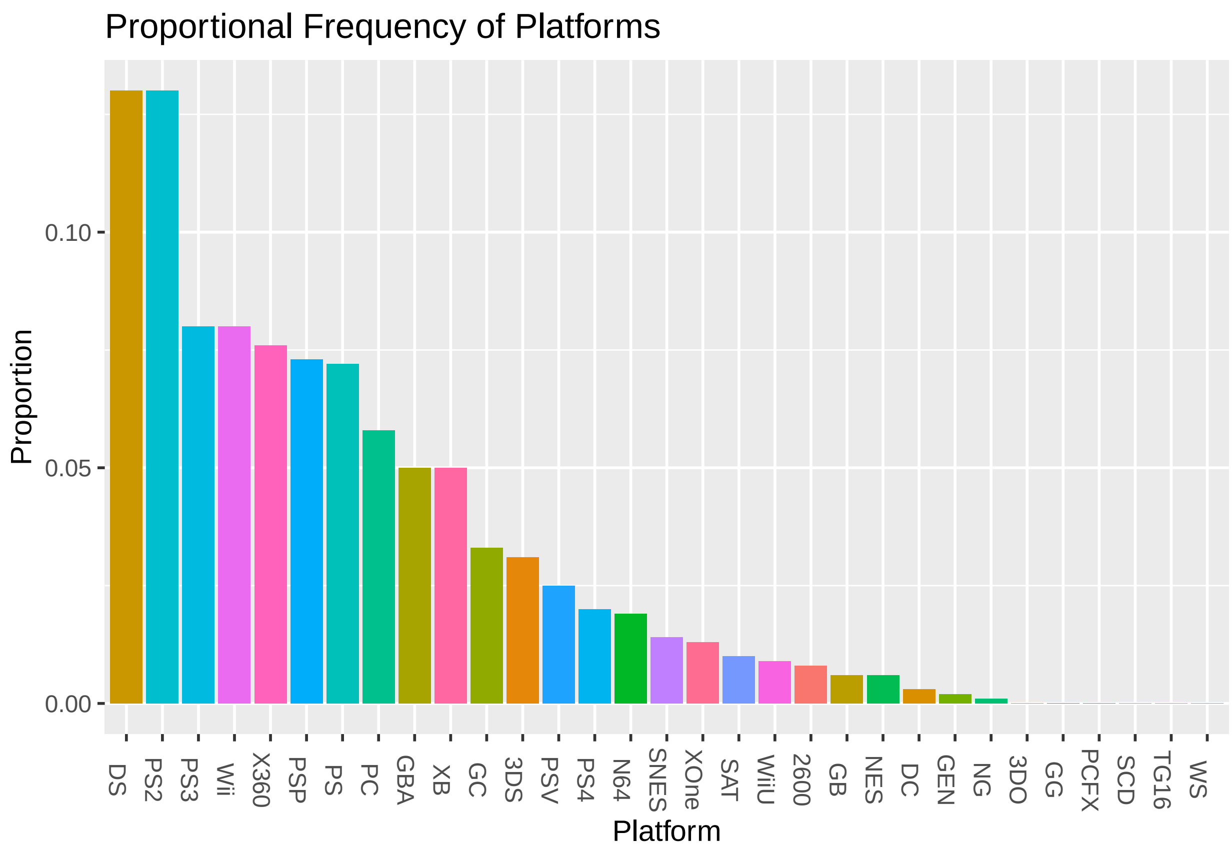
```
  dplyr::summarise(Frequency = n(),
            min_year = min(as.integer(Year),  na.rm = TRUE),
            max_year = max(as.integer(Year),  na.rm = TRUE))%>%
  arrange(desc(Frequency))%>%
  mutate("Proportion" = Frequency/16598,
         "CumSum" = cumsum(Proportion))%>%
  dplyr::select(Platform,Frequency,Proportion,CumSum,everything())%>%
  mutate(Proportion = round(Proportion,3),
         CumSum = round(CumSum,3))

platform.table %>%
  gt()%>%
  tab_header(
    title = "Platform Data Table",
    subtitle = "All 31 Platforms are presented"
  )
```

Platform Data Table

All 31 Platforms are presented

Platform	Frequency	Proportion	CumSum	min_year	max_year
DS	2163	0.130	0.130	1985	2020
PS2	2161	0.130	0.261	2000	2011
PS3	1329	0.080	0.341	2006	2016
Wii	1325	0.080	0.420	2006	2015
X360	1265	0.076	0.497	2005	2016
PSP	1213	0.073	0.570	2004	2015
PS	1196	0.072	0.642	1994	2003
PC	960	0.058	0.700	1985	2016
XB	824	0.050	0.749	2000	2008
GBA	822	0.050	0.799	2000	2007
GC	556	0.033	0.832	2001	2007
3DS	509	0.031	0.863	2011	2016
PSV	413	0.025	0.888	2011	2017
PS4	336	0.020	0.908	2013	2017
N64	319	0.019	0.927	1996	2002
SNES	239	0.014	0.942	1990	1999
XOne	213	0.013	0.955	2013	2016
SAT	173	0.010	0.965	1994	1999
WiiU	143	0.009	0.974	2012	2016
2600	133	0.008	0.982	1980	1989
GB	98	0.006	0.987	1988	2001
NES	98	0.006	0.993	1983	1994
DC	52	0.003	0.997	1998	2008
GEN	27	0.002	0.998	1990	1994
NG	12	0.001	0.999	1993	1996
SCD	6	0.000	0.999	1993	1994
WS	6	0.000	1.000	1999	2001
3DO	3	0.000	1.000	1994	1995
TG16	2	0.000	1.000	1995	1995
GG	1	0.000	1.000	1992	1992
PCFX	1	0.000	1.000	1996	1996

```
platform.table%>%
  ggplot(aes(x = reorder(Platform,desc(Proportion)), y = Proportion))+
  geom_bar(aes(fill =Platform),show.legend = FALSE, stat = 'Identity')+
  theme(axis.text.x = element_text(angle = -90))+
  labs(x = "Platform", y = "Proportion",title = "Proportional Frequency of Platforms")
```

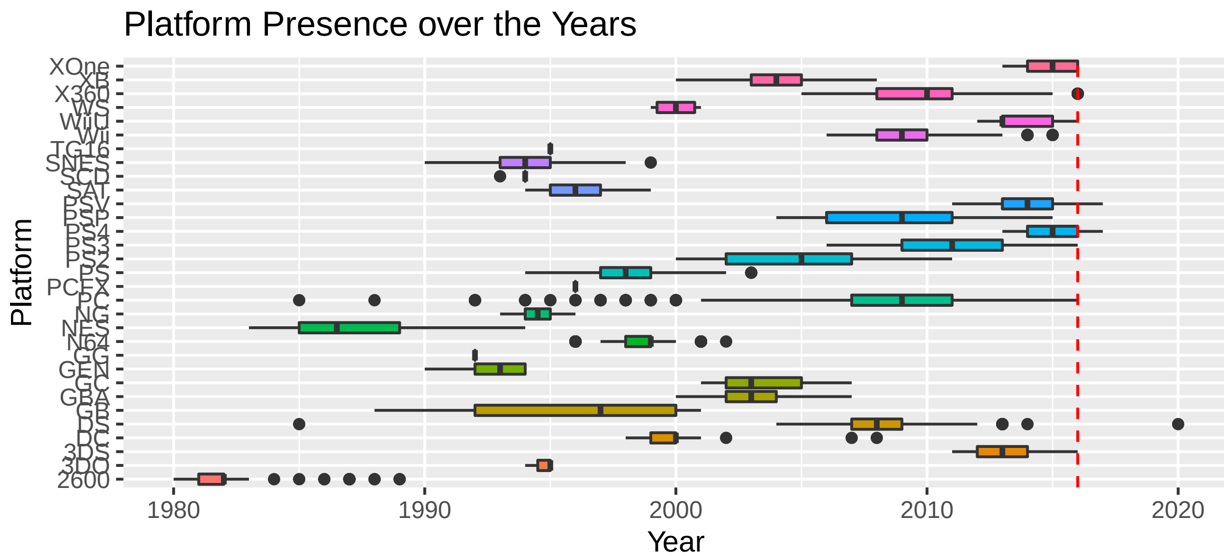


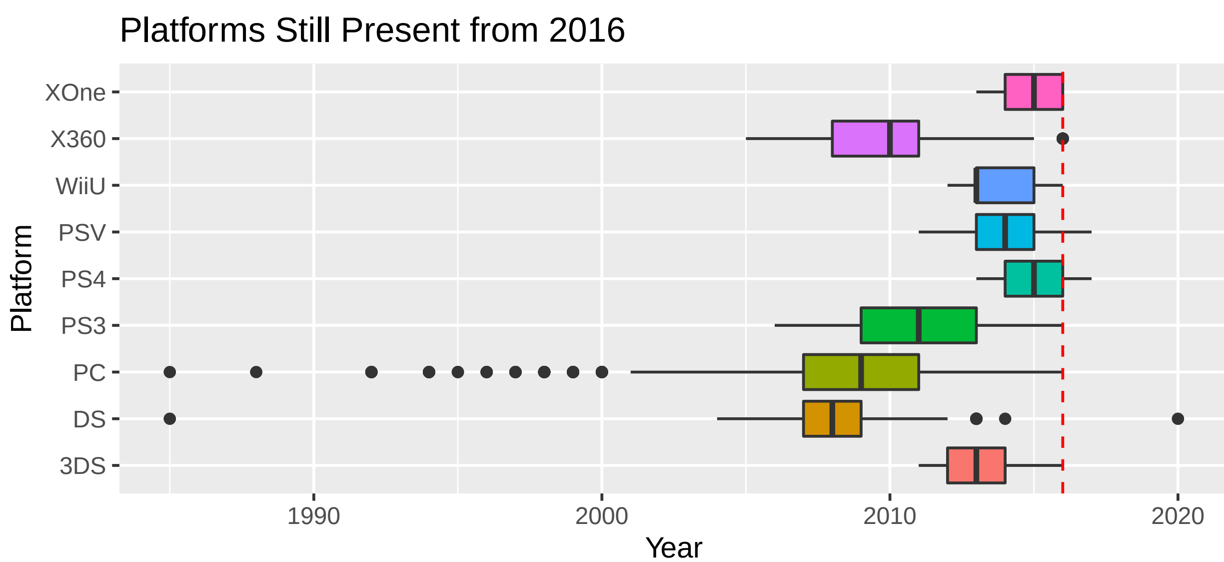

```
#boxplot
platform_box <- games.df%>%
  ggplot(aes(x= as.integer(Year), y = factor(Platform)))+
  geom_boxplot(aes(fill = Platform), show.legend = FALSE)+
  geom_vline(xintercept=2016, linetype="dashed", color = "red")+
  labs(x = 'Year',y = 'Platform', title = "Platform Presence over the Years")

#Filtered Platforms
current.platforms <- platform.table%>%
  filter(max_year>=2016)

#Filtered boxplot
filtered_box <- games.df%>%
  filter(Platform %in% c(current.platforms$Platform))%>%
  ggplot(aes(x= as.integer(Year), y = factor(Platform)))+
  geom_boxplot(aes(fill = Platform), show.legend = FALSE)+
  geom_vline(xintercept=2016, linetype="dashed", color = "red")+
  labs(x = 'Year',y = 'Platform', title = "Platforms Still Present from 2016")
```

```r
ggarrange(platform_box,filtered_box,
          nrow = 2)
```

```
## Warning: Removed 271 rows containing non-finite values (stat_boxplot).
```

```
## Warning: Removed 112 rows containing non-finite values (stat_boxplot).
```





Upon the inspection of our Platform variable, we have 31 unique Platforms with no missing values. However, there are obvious outliers in terms of the frequency/value counts of certain platforms especially on the lower end. We can see a significant tailing off of Platform frequency illustrated in our data table and Proportional frequency plot. Notably, there are 6 platforms that contain less than 7 observations each and only account for 0.145% of our data. This indicates that we might be able to reduce our Platform levels to ensure low-cardinality. Although our boxplot is a crud representation of Platform frequency over the years (indicating both life span and occurrence within our data set), it simply acts as a visual representation of the discontinuation of certain consoles over the span of our data set.

Whether the inclusion of discontinued platforms are pertinent to include in our modelling requires further analysis but i am of the option that they should be removed. In terms of tidying our data set, we have 31 unique Platforms, some of which have either been discontinued, meaning no further games will be made for those consoles and newer games will not be compatible with their operating systems, or have given way for the evolution of newer Platforms within the company's console offerings (PS2 -> PS3). Additionally, lower frequency Platforms (less than 7 observations) would carry less influential/predictive power (too few observations to train). Therefore, removing these aforementioned Platforms could create a more up to date data set accurately reflecting current market trends which will be reflected in our model and ensures we have low platform cardinality. Notably, the 9 identified Platforms above are the only Platforms out of the 31 still in the current market or are anticipated for a resurgence according to Platform providers.

Publisher Variable Analysis

```
games.df <- games.df%>%
  replace_with_na(replace = list(Publisher = "N/A"))

Publisher.table <- games.df%>%
  dplyr::count(Publisher, name = "Frequency")%>%
  arrange(desc(Frequency))%>%
  mutate("Proportion" = Frequency/sum(Frequency),
         "CumSum" = cumsum(Proportion))%>%
  mutate(Proportion = round(Proportion,3),
         CumSum = round(CumSum,3))

Publisher.table %>%
  gt()%>%
  tab_header(
    title = "Publisher Data Table",
    subtitle = "All 579 Publishers are presented"
  )
```

Publisher Data Table

All 579 Publishers are presented

Publisher	Frequency	Proportion	CumSum
Electronic Arts	1351	0.081	0.081
Activision	975	0.059	0.140
Namco Bandai Games	932	0.056	0.196
Ubisoft	921	0.055	0.252
Konami Digital Entertainment	832	0.050	0.302
THQ	715	0.043	0.345
Nintendo	703	0.042	0.387
Sony Computer Entertainment	683	0.041	0.428
Sega	639	0.038	0.467
Take-Two Interactive	413	0.025	0.492
Capcom	381	0.023	0.515
Atari	363	0.022	0.537
Tecmo Koei	338	0.020	0.557
Square Enix	233	0.014	0.571
Warner Bros. Interactive Entertainment	232	0.014	0.585
Disney Interactive Studios	218	0.013	0.598
Unknown	203	0.012	0.610

Eidos Interactive	198	0.012	0.622
Midway Games	198	0.012	0.634
505 Games	192	0.012	0.646
Microsoft Game Studios	189	0.011	0.657
Acclaim Entertainment	184	0.011	0.668
D3Publisher	184	0.011	0.679
Vivendi Games	164	0.010	0.689
Codemasters	152	0.009	0.698
Idea Factory	129	0.008	0.706
Deep Silver	122	0.007	0.714
Nippon Ichi Software	105	0.006	0.720
Zoo Digital Publishing	104	0.006	0.726
Majesco Entertainment	92	0.006	0.732
LucasArts	90	0.005	0.737
Rising Star Games	86	0.005	0.742
Hudson Soft	81	0.005	0.747
Banpresto	73	0.004	0.752
Bethesda Softworks	71	0.004	0.756
Crave Entertainment	71	0.004	0.760
Atlus	67	0.004	0.764
Infogrames	62	0.004	0.768
Virgin Interactive	62	0.004	0.772
5pb	61	0.004	0.775
Ignition Entertainment	61	0.004	0.779
Focus Home Interactive	58	0.003	0.783
NA	58	0.003	0.786
Marvelous Interactive	56	0.003	0.789
Empire Interactive	52	0.003	0.793
SquareSoft	52	0.003	0.796
Kadokawa Shoten	50	0.003	0.799
Destineer	45	0.003	0.801
DTP Entertainment	45	0.003	0.804
GT Interactive	45	0.003	0.807
Alchemist	43	0.003	0.809
MTV Games	41	0.002	0.812
Global Star	39	0.002	0.814
PQube	39	0.002	0.817
SouthPeak Games	37	0.002	0.819
Spike	37	0.002	0.821
Takara Tomy	37	0.002	0.823
3DO	36	0.002	0.825
TDK Mediactive	36	0.002	0.828
BAM! Entertainment	35	0.002	0.830
Nordic Games	35	0.002	0.832
Black Bean Games	34	0.002	0.834
Zoo Games	33	0.002	0.836
Game Factory	32	0.002	0.838
Mindscape	32	0.002	0.840
Psygnosis	32	0.002	0.842
Enix Corporation	30	0.002	0.843
Interplay	30	0.002	0.845
Activision Value	29	0.002	0.847
Kalypso Media	29	0.002	0.849
FuRyu	27	0.002	0.850

Level 5	27	0.002	0.852
Prototype	27	0.002	0.854
Arc System Works	26	0.002	0.855
Avanquest	26	0.002	0.857
Little Orbit	26	0.002	0.858
Telltale Games	25	0.002	0.860
Midas Interactive Entertainment	24	0.001	0.861
Aqua Plus	23	0.001	0.863
Jaleco	23	0.001	0.864
Paradox Interactive	23	0.001	0.865
Universal Interactive	23	0.001	0.867
Broccoli	22	0.001	0.868
JoWood Productions	22	0.001	0.869
Oxygen Interactive	22	0.001	0.871
SNK	22	0.001	0.872
Kemco	21	0.001	0.873
ASCII Entertainment	20	0.001	0.875
Compile Heart	20	0.001	0.876
City Interactive	19	0.001	0.877
Storm City Games	19	0.001	0.878
Success	19	0.001	0.879
Taito	19	0.001	0.880
Titus	19	0.001	0.881
ChunSoft	18	0.001	0.883
SNK Playmore	18	0.001	0.884
Tomy Corporation	18	0.001	0.885
Zushi Games	18	0.001	0.886
DreamCatcher Interactive	17	0.001	0.887
Koch Media	17	0.001	0.888
Natsume	17	0.001	0.889
O-Games	17	0.001	0.890
Rocket Company	17	0.001	0.891
SCi	17	0.001	0.892
Falcom Corporation	16	0.001	0.893
GSP	16	0.001	0.894
Hasbro Interactive	16	0.001	0.895
Imagineer	16	0.001	0.896
Mastiff	16	0.001	0.897
Milestone S.r.l.	16	0.001	0.898
Takara	16	0.001	0.899
UFO Interactive	16	0.001	0.900
Enterbrain	15	0.001	0.901
From Software	15	0.001	0.901
Ghostlight	15	0.001	0.902
Kadokawa Games	15	0.001	0.903
PopCap Games	15	0.001	0.904
Sony Computer Entertainment Europe	15	0.001	0.905
989 Studios	14	0.001	0.906
Conspiracy Entertainment	14	0.001	0.907
CyberFront	14	0.001	0.908
Ocean	14	0.001	0.908
Play It	14	0.001	0.909
Playlogic Game Factory	14	0.001	0.910
Quinrose	14	0.001	0.911

Rondomedia	14	0.001	0.912
System 3 Arcade Software	14	0.001	0.913
Ubisoft Annecy	14	0.001	0.914
Acquire	13	0.001	0.914
Bigben Interactive	13	0.001	0.915
GungHo	13	0.001	0.916
Gust	13	0.001	0.917
Human Entertainment	13	0.001	0.917
Mastertronic	13	0.001	0.918
Nobilis	13	0.001	0.919
Destination Software, Inc	12	0.001	0.920
Irem Software Engineering	12	0.001	0.920
Marvelous Entertainment	12	0.001	0.921
Mattel Interactive	12	0.001	0.922
Metro 3D	12	0.001	0.923
XS Games	12	0.001	0.923
Hudson Entertainment	11	0.001	0.924
Sammy Corporation	11	0.001	0.925
Yeti	11	0.001	0.925
Ackkstudios	10	0.001	0.926
Brash Entertainment	10	0.001	0.927
Cave	10	0.001	0.927
Microids	10	0.001	0.928
Popcorn Arcade	10	0.001	0.928
Scholastic Inc.	10	0.001	0.929
Starfish	10	0.001	0.930
Sunsoft	10	0.001	0.930
Xplosiv	10	0.001	0.931
ArtDink	9	0.001	0.931
ASCII Media Works	9	0.001	0.932
Avanquest Software	9	0.001	0.932
Foreign Media Games	9	0.001	0.933
Gathering of Developers	9	0.001	0.933
Gremlin Interactive Ltd	9	0.001	0.934
NewKidCo	9	0.001	0.935
Sting	9	0.001	0.935
Victor Interactive	9	0.001	0.936
Agetec	8	0.000	0.936
Aksys Games	8	0.000	0.937
Asgard	8	0.000	0.937
Aspyr	8	0.000	0.938
Evolved Games	8	0.000	0.938
Fox Interactive	8	0.000	0.939
GameMill Entertainment	8	0.000	0.939
Genki	8	0.000	0.940
JVC	8	0.000	0.940
MTO	8	0.000	0.940
NEC Interchannel	8	0.000	0.941
RTL	8	0.000	0.941
Sony Online Entertainment	8	0.000	0.942
Telegames	8	0.000	0.942
Tru Blu Entertainment	8	0.000	0.943
Valcon Games	8	0.000	0.943
Big Ben Interactive	7	0.000	0.944

BMG Interactive Entertainment	7	0.000	0.944
Epoch	7	0.000	0.945
Gotham Games	7	0.000	0.945
Hackberry	7	0.000	0.945
LEGO Media	7	0.000	0.946
Neko Entertainment	7	0.000	0.946
Nihon Falcom Corporation	7	0.000	0.947
Nippon Columbia	7	0.000	0.947
Parker Bros.	7	0.000	0.948
Rage Software	7	0.000	0.948
Reef Entertainment	7	0.000	0.948
Alternative Software	6	0.000	0.949
Astragon	6	0.000	0.949
Asylum Entertainment	6	0.000	0.950
Avalon Interactive	6	0.000	0.950
Benesse	6	0.000	0.950
Blast! Entertainment Ltd	6	0.000	0.951
CDV Software Entertainment	6	0.000	0.951
Comfort	6	0.000	0.951
Compile	6	0.000	0.952
DSI Games	6	0.000	0.952
Funbox Media	6	0.000	0.952
Graffiti	6	0.000	0.953
Idea Factory International	6	0.000	0.953
Kaga Create	6	0.000	0.953
Microprose	6	0.000	0.954
Mumbo Jumbo	6	0.000	0.954
NCSoft	6	0.000	0.955
Paon	6	0.000	0.955
Russel	6	0.000	0.955
Screenlife	6	0.000	0.956
Seta Corporation	6	0.000	0.956
Square	6	0.000	0.956
Swing! Entertainment	6	0.000	0.957
20th Century Fox Video Games	5	0.000	0.957
AQ Interactive	5	0.000	0.957
bitComposer Games	5	0.000	0.958
Coleco	5	0.000	0.958
Crystal Dynamics	5	0.000	0.958
dramatic create	5	0.000	0.959
ESP	5	0.000	0.959
Happinet	5	0.000	0.959
Hip Interactive	5	0.000	0.959
Home Entertainment Suppliers	5	0.000	0.960
IE Institute	5	0.000	0.960
Media Works	5	0.000	0.960
Mentor Interactive	5	0.000	0.961
Mojang	5	0.000	0.961
Nordcurrent	5	0.000	0.961
Pinnacle	5	0.000	0.962
Shogakukan	5	0.000	0.962
TDK Core	5	0.000	0.962
The Adventure Company	5	0.000	0.962
Time Warner Interactive	5	0.000	0.963

Tommo	5	0.000	0.963
Wanadoo	5	0.000	0.963
7G//AMES	4	0.000	0.964
Culture Brain	4	0.000	0.964
Daito	4	0.000	0.964
Encore	4	0.000	0.964
Excalibur Publishing	4	0.000	0.965
Funsta	4	0.000	0.965
Gamecock	4	0.000	0.965
Global A Entertainment	4	0.000	0.965
Imagic	4	0.000	0.966
Interchannel	4	0.000	0.966
KID	4	0.000	0.966
Knowledge Adventure	4	0.000	0.966
Laguna	4	0.000	0.967
LSP Games	4	0.000	0.967
Mercury Games	4	0.000	0.967
Phenomedia	4	0.000	0.967
Pioneer LDC	4	0.000	0.967
PlayV	4	0.000	0.968
RedOctane	4	0.000	0.968
Slitherine Software	4	0.000	0.968
Sunrise Interactive	4	0.000	0.968
System Soft	4	0.000	0.969
TGL	4	0.000	0.969
TopWare Interactive	4	0.000	0.969
Touchstone	4	0.000	0.969
Trion Worlds	4	0.000	0.970
U.S. Gold	4	0.000	0.970
ValuSoft	4	0.000	0.970
Video System	4	0.000	0.970
Xseed Games	4	0.000	0.971
10TACLE Studios	3	0.000	0.971
1C Company	3	0.000	0.971
Accolade	3	0.000	0.971
Agatsuma Entertainment	3	0.000	0.971
Angel Studios	3	0.000	0.972
Arika	3	0.000	0.972
Aruze Corp	3	0.000	0.972
ASC Games	3	0.000	0.972
Ascaron Entertainment GmbH	3	0.000	0.972
Asmik Ace Entertainment	3	0.000	0.972
Creative Core	3	0.000	0.973
Daedalic	3	0.000	0.973
Daedalic Entertainment	3	0.000	0.973
Data Design Interactive	3	0.000	0.973
DHM Interactive	3	0.000	0.973
Essential Games	3	0.000	0.973
Experience Inc.	3	0.000	0.974
Focus Multimedia	3	0.000	0.974
GN Software	3	0.000	0.974
Hect	3	0.000	0.974
Iceberg Interactive	3	0.000	0.974
Indie Games	3	0.000	0.975

Insomniac Games	3	0.000	0.975
Jack of All Games	3	0.000	0.975
Jester Interactive	3	0.000	0.975
Jorudan	3	0.000	0.975
Licensed 4U	3	0.000	0.975
Mad Catz	3	0.000	0.976
Maxis	3	0.000	0.976
MC2 Entertainment	3	0.000	0.976
Media Rings	3	0.000	0.976
Micro Cabin	3	0.000	0.976
Minato Station	3	0.000	0.977
Mud Duck Productions	3	0.000	0.977
Myelin Media	3	0.000	0.977
NCS	3	0.000	0.977
NEC	3	0.000	0.977
NovaLogic	3	0.000	0.977
O3 Entertainment	3	0.000	0.978
P2 Games	3	0.000	0.978
Princess Soft	3	0.000	0.978
Red Storm Entertainment	3	0.000	0.978
Slightly Mad Studios	3	0.000	0.978
Sony Computer Entertainment America	3	0.000	0.979
System 3	3	0.000	0.979
Telstar	3	0.000	0.979
Tigervision	3	0.000	0.979
Tivola	3	0.000	0.979
Tradewest	3	0.000	0.979
Valve Software	3	0.000	0.980
Vir2L Studios	3	0.000	0.980
Xicat Interactive	3	0.000	0.980
Yacht Club Games	3	0.000	0.980
Yuke’s	3	0.000	0.980
Aerosoft	2	0.000	0.980
Alawar Entertainment	2	0.000	0.981
Alvion	2	0.000	0.981
Arena Entertainment	2	0.000	0.981
Asmik Corp	2	0.000	0.981
Athena	2	0.000	0.981
Big Fish Games	2	0.000	0.981
Blue Byte	2	0.000	0.981
BPS	2	0.000	0.981
Cloud Imperium Games Corporation	2	0.000	0.982
Coconuts Japan	2	0.000	0.982
Core Design Ltd.	2	0.000	0.982
Crimson Cow	2	0.000	0.982
CTO SpA	2	0.000	0.982
Data Age	2	0.000	0.982
Data East	2	0.000	0.982
Datam Polystar	2	0.000	0.982
Devolver Digital	2	0.000	0.983
Dorart	2	0.000	0.983
Dusenberry Martin Racing	2	0.000	0.983
Easy Interactive	2	0.000	0.983
Edia	2	0.000	0.983

Electronic Arts Victor	2	0.000	0.983
Elf	2	0.000	0.983
Flashpoint Games	2	0.000	0.983
Flight-Plan	2	0.000	0.983
Funcom	2	0.000	0.984
G.Rev	2	0.000	0.984
Gainax Network Systems	2	0.000	0.984
Gakken	2	0.000	0.984
Game Life	2	0.000	0.984
Gamebridge	2	0.000	0.984
Groove Games	2	0.000	0.984
Hamster Corporation	2	0.000	0.984
Harmonix Music Systems	2	0.000	0.985
HMH Interactive	2	0.000	0.985
HuneX	2	0.000	0.985
imageepoch Inc.	2	0.000	0.985
Lexicon Entertainment	2	0.000	0.985
Liquid Games	2	0.000	0.985
Magix	2	0.000	0.985
Mamba Games	2	0.000	0.985
Media Factory	2	0.000	0.986
Milestone S.r.l	2	0.000	0.986
Misawa	2	0.000	0.986
Moss	2	0.000	0.986
NetRevo	2	0.000	0.986
Nippon Telenet	2	0.000	0.986
Nitroplus	2	0.000	0.986
Office Create	2	0.000	0.986
Pack-In-Video	2	0.000	0.987
Performance Designed Products	2	0.000	0.987
Rebellion	2	0.000	0.987
Rebellion Developments	2	0.000	0.987
Red Orb	2	0.000	0.987
responDESIGN	2	0.000	0.987
Revolution Software	2	0.000	0.987
Sonnet	2	0.000	0.987
Sweets	2	0.000	0.987
Syscom	2	0.000	0.988
Vatical Entertainment	2	0.000	0.988
Vic Tokai	2	0.000	0.988
Views	2	0.000	0.988
Virtual Play Games	2	0.000	0.988
Yamasa Entertainment	2	0.000	0.988
Zenrin	2	0.000	0.988
2D Boy	1	0.000	0.988
49Games	1	0.000	0.988
989 Sports	1	0.000	0.988
Abylight	1	0.000	0.989
Activision Blizzard	1	0.000	0.989
Adeline Software	1	0.000	0.989
Altron	1	0.000	0.989
American Softworks	1	0.000	0.989
Answer Software	1	0.000	0.989
Aques	1	0.000	0.989

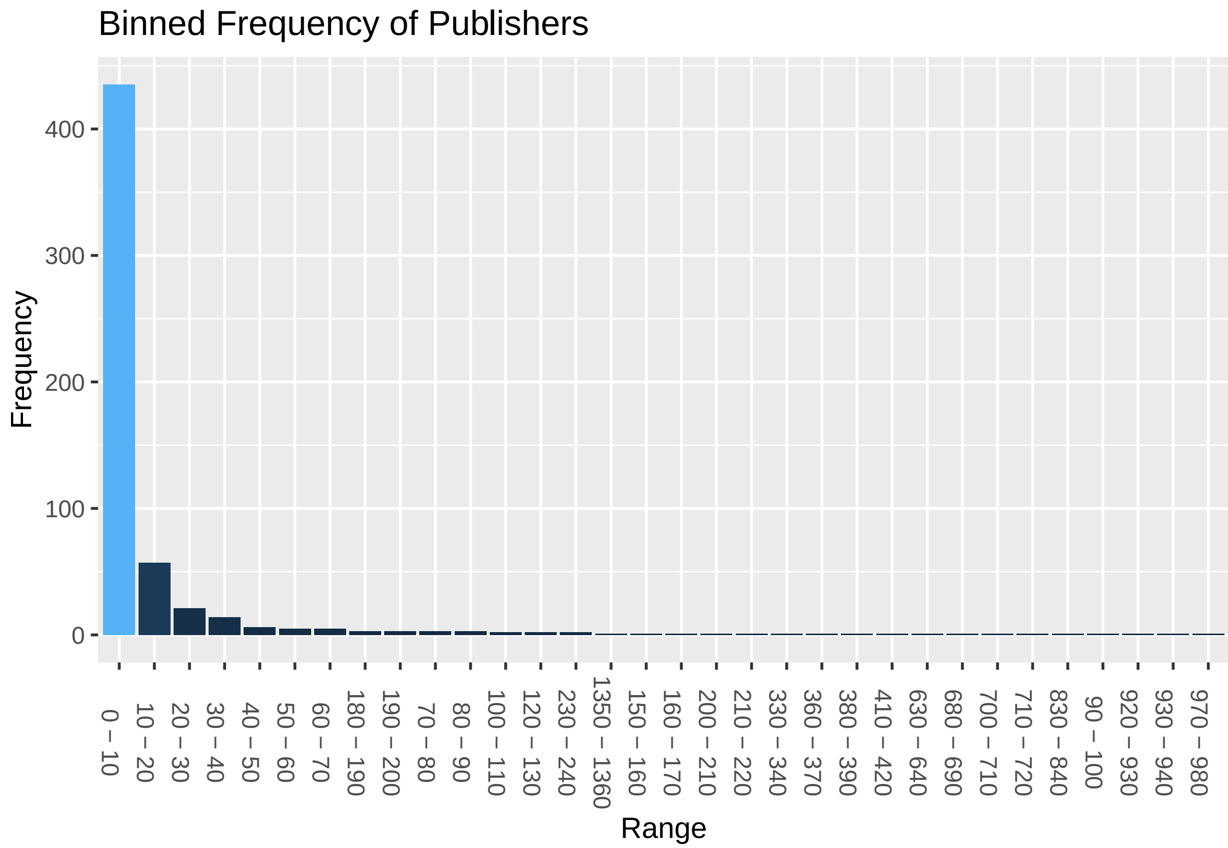
Aria	1	0.000	0.989
Ascaron Entertainment	1	0.000	0.989
ASK	1	0.000	0.989
Axela	1	0.000	0.989
Berkeley	1	0.000	0.989
Black Label Games	1	0.000	0.989
Bohemia Interactive	1	0.000	0.989
Bomb	1	0.000	0.989
Boost On	1	0.000	0.989
BushiRoad	1	0.000	0.990
CBS Electronics	1	0.000	0.990
CCP	1	0.000	0.990
Codemasters Online	1	0.000	0.990
CokeM Interactive	1	0.000	0.990
Commseed	1	0.000	0.990
CPG Products	1	0.000	0.990
Culture Publishers	1	0.000	0.990
Cygames	1	0.000	0.990
Detn8 Games	1	0.000	0.990
DigiCube	1	0.000	0.990
DreamWorks Interactive	1	0.000	0.990
EA Games	1	0.000	0.990
Ecole	1	0.000	0.990
Elite	1	0.000	0.990
Enjoy Gaming ltd.	1	0.000	0.990
EON Digital Entertainment	1	0.000	0.990
Epic Games	1	0.000	0.991
Ertain	1	0.000	0.991
Evolution Games	1	0.000	0.991
Extreme Entertainment Group	1	0.000	0.991
Fields	1	0.000	0.991
fonfun	1	0.000	0.991
Fortyfive	1	0.000	0.991
Fuji	1	0.000	0.991
FunSoft	1	0.000	0.991
FuRyu Corporation	1	0.000	0.991
Gaga	1	0.000	0.991
Game Arts	1	0.000	0.991
Gameloft	1	0.000	0.991
GameTek	1	0.000	0.991
General Entertainment	1	0.000	0.991
Genterprise	1	0.000	0.991
Giga	1	0.000	0.992
Giza10	1	0.000	0.992
Glams	1	0.000	0.992
GOA	1	0.000	0.992
Grand Prix Games	1	0.000	0.992
Graphsim Entertainment	1	0.000	0.992
Griffin International	1	0.000	0.992
HAL Laboratory	1	0.000	0.992
Havas Interactive	1	0.000	0.992
Headup Games	1	0.000	0.992
Hearty Robin	1	0.000	0.992
Hello Games	1	0.000	0.992

Her Interactive	1	0.000	0.992
id Software	1	0.000	0.992
Illusion Softworks	1	0.000	0.992
Imadio	1	0.000	0.992
Image Epoch	1	0.000	0.992
Imageworks	1	0.000	0.993
Imax	1	0.000	0.993
Interchannel-Holon	1	0.000	0.993
Intergrow	1	0.000	0.993
Interplay Productions	1	0.000	0.993
Interworks Unlimited, Inc.	1	0.000	0.993
Inti Creates	1	0.000	0.993
Introversion Software	1	0.000	0.993
inXile Entertainment	1	0.000	0.993
ITT Family Games	1	0.000	0.993
Ivolgamus	1	0.000	0.993
iWin	1	0.000	0.993
Just Flight	1	0.000	0.993
Kamui	1	0.000	0.993
Kando Games	1	0.000	0.993
Karin Entertainment	1	0.000	0.993
Kids Station	1	0.000	0.993
King Records	1	0.000	0.994
Kokopeli Digital Studios	1	0.000	0.994
Kool Kizz	1	0.000	0.994
KSS	1	0.000	0.994
Legacy Interactive	1	0.000	0.994
Lighthouse Interactive	1	0.000	0.994
Locus	1	0.000	0.994
Magical Company	1	0.000	0.994
Marvel Entertainment	1	0.000	0.994
Marvelous Games	1	0.000	0.994
Masque Publishing	1	0.000	0.994
Max Five	1	0.000	0.994
Maximum Family Games	1	0.000	0.994
Media Entertainment	1	0.000	0.994
MediaQuest	1	0.000	0.994
Men-A-Vision	1	0.000	0.994
Merscom LLC	1	0.000	0.995
Michaelsoft	1	0.000	0.995
Milestone	1	0.000	0.995
Mirai Shounen	1	0.000	0.995
Mitsui	1	0.000	0.995
mixi, Inc	1	0.000	0.995
MLB.com	1	0.000	0.995
Monte Christo Multimedia	1	0.000	0.995
Mycom	1	0.000	0.995
Mystique	1	0.000	0.995
Navarre Corp	1	0.000	0.995
Naxat Soft	1	0.000	0.995
NDA Productions	1	0.000	0.995
New	1	0.000	0.995
New World Computing	1	0.000	0.995
Nexon	1	0.000	0.995

Nichibutsu	1	0.000	0.995
Nippon Amuse	1	0.000	0.996
Number None	1	0.000	0.996
On Demand	1	0.000	0.996
Ongakukan	1	0.000	0.996
Origin Systems	1	0.000	0.996
Otomate	1	0.000	0.996
Pacific Century Cyber Works	1	0.000	0.996
Pack In Soft	1	0.000	0.996
Palcom	1	0.000	0.996
Panther Software	1	0.000	0.996
Paon Corporation	1	0.000	0.996
Paradox Development	1	0.000	0.996
Phantagram	1	0.000	0.996
Phantom EFX	1	0.000	0.996
Phoenix Games	1	0.000	0.996
Piacci	1	0.000	0.996
Playmates	1	0.000	0.997
Playmore	1	0.000	0.997
Plenty	1	0.000	0.997
PM Studios	1	0.000	0.997
Pony Canyon	1	0.000	0.997
PopTop Software	1	0.000	0.997
Pow	1	0.000	0.997
Quelle	1	0.000	0.997
Quest	1	0.000	0.997
Quintet	1	0.000	0.997
Rain Games	1	0.000	0.997
RED Entertainment	1	0.000	0.997
Revolution (Japan)	1	0.000	0.997
Riverhillsoft	1	0.000	0.997
Saurus	1	0.000	0.997
SCS Software	1	0.000	0.997
Sears	1	0.000	0.997
Seventh Chord	1	0.000	0.998
Simon & Schuster Interactive	1	0.000	0.998
Societa	1	0.000	0.998
Sold Out	1	0.000	0.998
Sony Music Entertainment	1	0.000	0.998
SPS	1	0.000	0.998
Square EA	1	0.000	0.998
SSI	1	0.000	0.998
Stainless Games	1	0.000	0.998
Starpath Corp.	1	0.000	0.998
Strategy First	1	0.000	0.998
Summitsoft	1	0.000	0.998
Sunflowers	1	0.000	0.998
T&E Soft	1	0.000	0.998
Takuyo	1	0.000	0.998
TalonSoft	1	0.000	0.998
Team17 Software	1	0.000	0.998
Technos Japan Corporation	1	0.000	0.999
TechnoSoft	1	0.000	0.999
Tetris Online	1	0.000	0.999

The Learning Company	1	0.000	0.999
TOHO	1	0.000	0.999
Tripwire Interactive	1	0.000	0.999
Tryfirst	1	0.000	0.999
TYO	1	0.000	0.999
Type-Moon	1	0.000	0.999
UEP Systems	1	0.000	0.999
UIG Entertainment	1	0.000	0.999
Ultravision	1	0.000	0.999
Universal Gamex	1	0.000	0.999
Valve	1	0.000	0.999
Vap	1	0.000	0.999
Visco	1	0.000	0.999
Warashi	1	0.000	1.000
Wargaming.net	1	0.000	1.000
Warp	1	0.000	1.000
WayForward Technologies	1	0.000	1.000
Westwood Studios	1	0.000	1.000
White Park Bay Software	1	0.000	1.000
Wizard Video Games	1	0.000	1.000
Xing Entertainment	1	0.000	1.000
Yumedia	1	0.000	1.000

```
br <- seq(0,1400,by=10)
ranges = paste(head(br,-1), br[-1], sep=" - ")
freq <- hist(Publisher.table$Frequency, breaks = br,include.lowest=TRUE, plot = FALSE)

tibble(range = ranges, frequency = freq$counts)%>%
  filter(frequency>0)%>%
  ggplot(aes(x= reorder(range,desc(frequency)),y =frequency ))+
  geom_bar(stat = "identity",aes(fill = frequency), show.legend = FALSE)+
  theme(axis.text.x = element_text(angle = -90))+
  labs(x= 'Range', y = 'Frequency', title = "Binned Frequency of Publishers")
```



Upon the inspection of our Publisher variable there are obvious outliers in terms of the frequency/value counts of certain Publishers especially on the lower end. Looking at both the frequency bar plot and our data table we can ascertain that there is a disproportional number of observations in the 0-10 bin. Notably, there are 435 Publishers that contain less than 10 observations each. Additionally, the top 10 most frequently occurring Publishers account for 50% of our data. This exposes the high-cardinality of our Publisher variable if we were to use it in our modelling. Therefore, in terms of the exorbitant amount of Publishers making dummy encoding impractical (could cut top 6 and bin rest as "Other" category creating 7 Publishers) and the fact that the Publisher variable acts simply as a subgroup of our video game names, we should drop the Publisher variable.

Identify the variables that will not to be included in the final analysis.

For our analysis, we are not interested in the names of our video games, so we are going to drop the Name variables from our data set. Name is just a way for us to identify the video games we are looking at (similar to an ID) and as a result will not be informative to the analysis. We can also drop the Publisher variable, since this is simply acting as a subgroup (each Publisher has the rights to a licensed game franchise) of our video game names adding no new information. The final thing we are going to drop is the Year variable. There could be some interesting behaviours with the Year a game was released (best performing Genre released by Year or highest grossing Year for specific Platforms). However, Platforms are released in sequential order corresponding to specific dates and as such Year won't give us much more information than the Platform already does.

Feature Engineering

Current video game market trends suggest the 'hype', popularity, and exposure of games drive sales. Inevitably we would expect games exposed to multiple markets have a greater chance of increasing its consumer popularity. Exclusive market games or games not sold globally lacks this exposure and could potentially impact local and global market sales. Therefore, we will consider the "global" variable.

```
#Feature Engineering
games.df <- games.df%>%
  mutate('global' = ifelse(EU_Sales != 0 & Other_Sales != 0 & JP_Sales!= 0,"Yes","No"))

#Filtered Platforms
current.platforms <- platform.table%>%
  filter(max_year>=2016)
```

Apply changes to the data and skim the dataset.

Our clean data set will contain a reduced number of observations due to the filtering of our Platform variable to only include the 9 aforementioned Platforms. Finally, the conversion of Platform, Genre and Glboal into factor data types is illustrated below.

```
Vidgames.df <- games.df%>%
  dplyr::select(Platform,Genre,global,EU_Sales:Other_Sales,NA_Sales)%>%
  filter(Platform %in% c(current.platforms$Platform))%>%
  mutate_if(is.character,as.factor)%>% #data types convert
  janitor::clean_names()

skim(Vidgames.df)
```

Data summary

Name	Vidgames.df
Number of rows	7331
Number of columns	7
Column type frequency:	
factor	3
numeric	4
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
platform	0	1	FALSE	9	DS: 2163, PS3: 1329, X36: 1265, PC: 960
genre	0	1	FALSE	12	Act: 1789, Mis: 795, Spo: 766, Rol: 734
global	0	1	FALSE	2	No: 6018, Yes: 1313

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
eu_sales	0	1	0.17	0.50	0	0	0.02	0.13	11.00	
jp_sales	0	1	0.06	0.27	0	0	0.00	0.03	6.50	
other_sales	0	1	0.05	0.16	0	0	0.01	0.04	4.14	
na_sales	0	1	0.24	0.64	0	0	0.07	0.22	14.97	

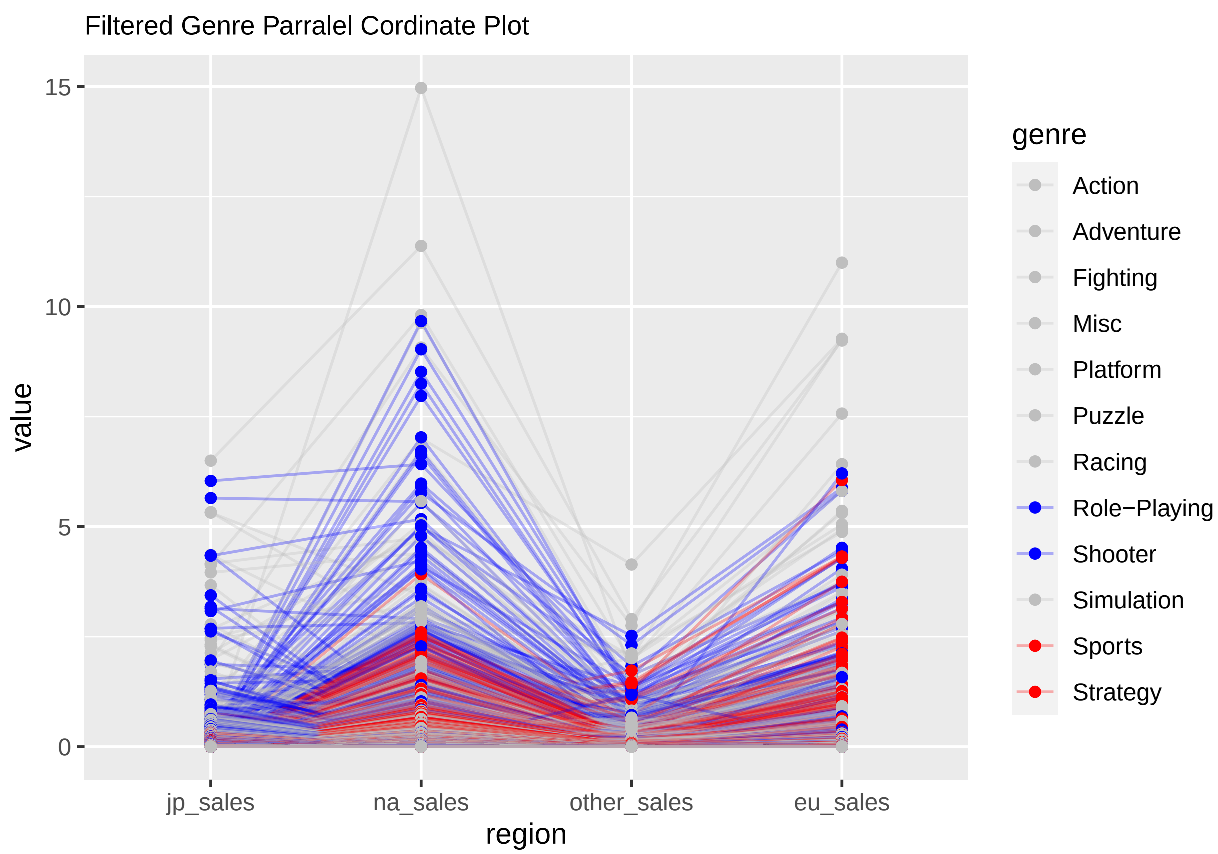
Exploratory Data Anaalysis

The Exploratory Data Analysis process refers to the critical process of performing initial investigations on our data. This allows us to uncover hidden patterns, identify certain anomalies, test our hypothesis, identify important variables, and to check assumptions with the assistance of statistical tests and graphical representations. Through our EDA process we will attempt to uncover natural groupings through parallel coordinate plots, discover the principal axis of variation within our data set by projecting our data onto lower dimensional space spanned by the maximum variance direction through PCA, investigate the relationship between our response variable and numerical predictors via spearman correlation, and finally investigate the relationship between our response variable and non-numerical predictors via ANOVA/Kruskal-Wallis tests, box plots and heatmaps.

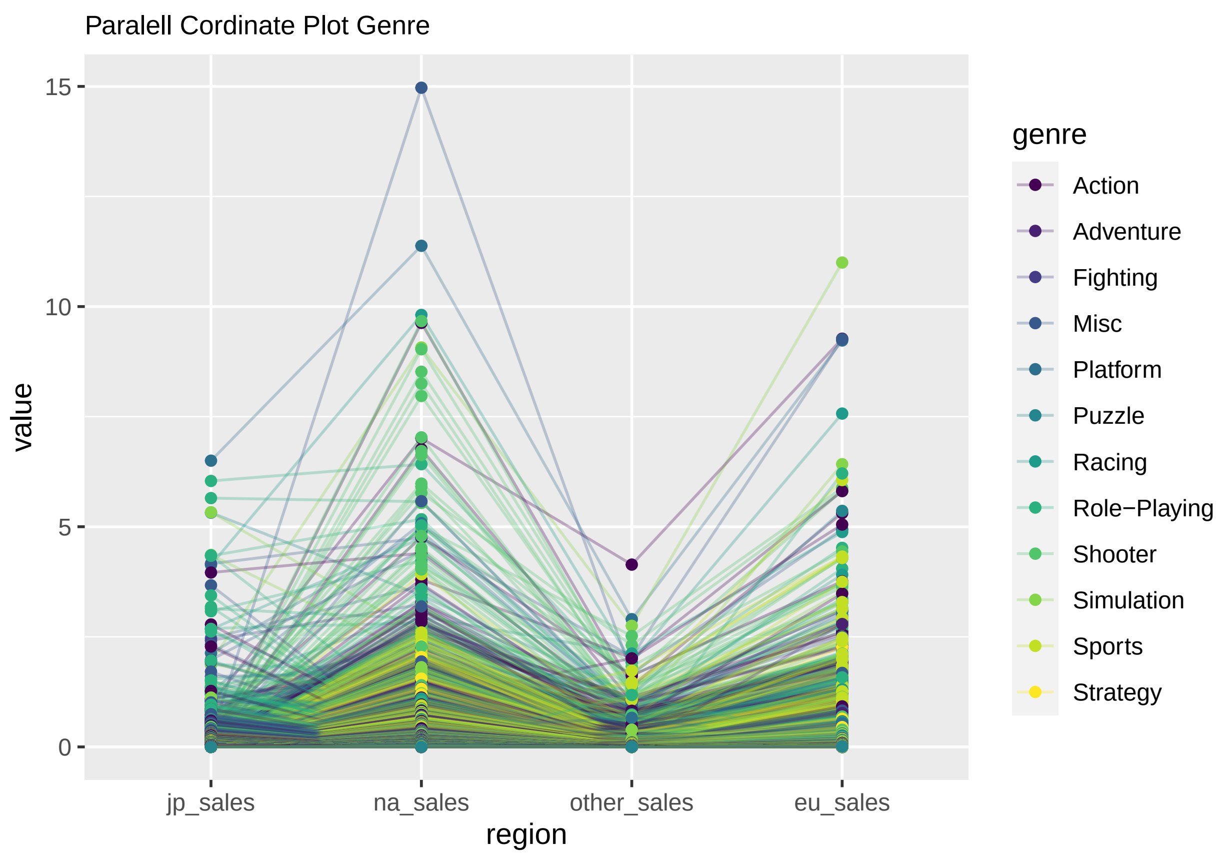
Parralel Cordinate Plot

There appears to be some segregation in NA_Sales in terms of the genre category. Arguably the shooter and role-playing genre is distinguishable from sports and strategy. Therefore, a case could be made for the existence of natural groupings within these genres. However, we cannot see any conclusive segregation as the groupings seem to be driven by outliers and the imbalance of observations within our genre variable as seen in our genre frequency heatmap. Additionally, the relatively moderate cardinality of our genre makes parallel coordinate plots very hard to visualize and interpret (only through genre filtering by color scale are groupings visable).

```
set.seed(1007)

Vidgames.df %>%
  ggparcoord(
    scale = 'globalminmax',
    columns = 4:7, groupColumn = 2, order = "anyClass",
    showPoints = TRUE,
    title = "Filtered Genre Parralel Cordinate Plot",
    alphaLines = 0.3
  ) +
  scale_color_manual(values=c( "grey", "grey", "grey", "grey", "grey", "grey",
                               "grey","blue", "blue", "grey", "red","red") )+
  theme(
    plot.title = element_text(size=10)
  ) +
  xlab("region")
```

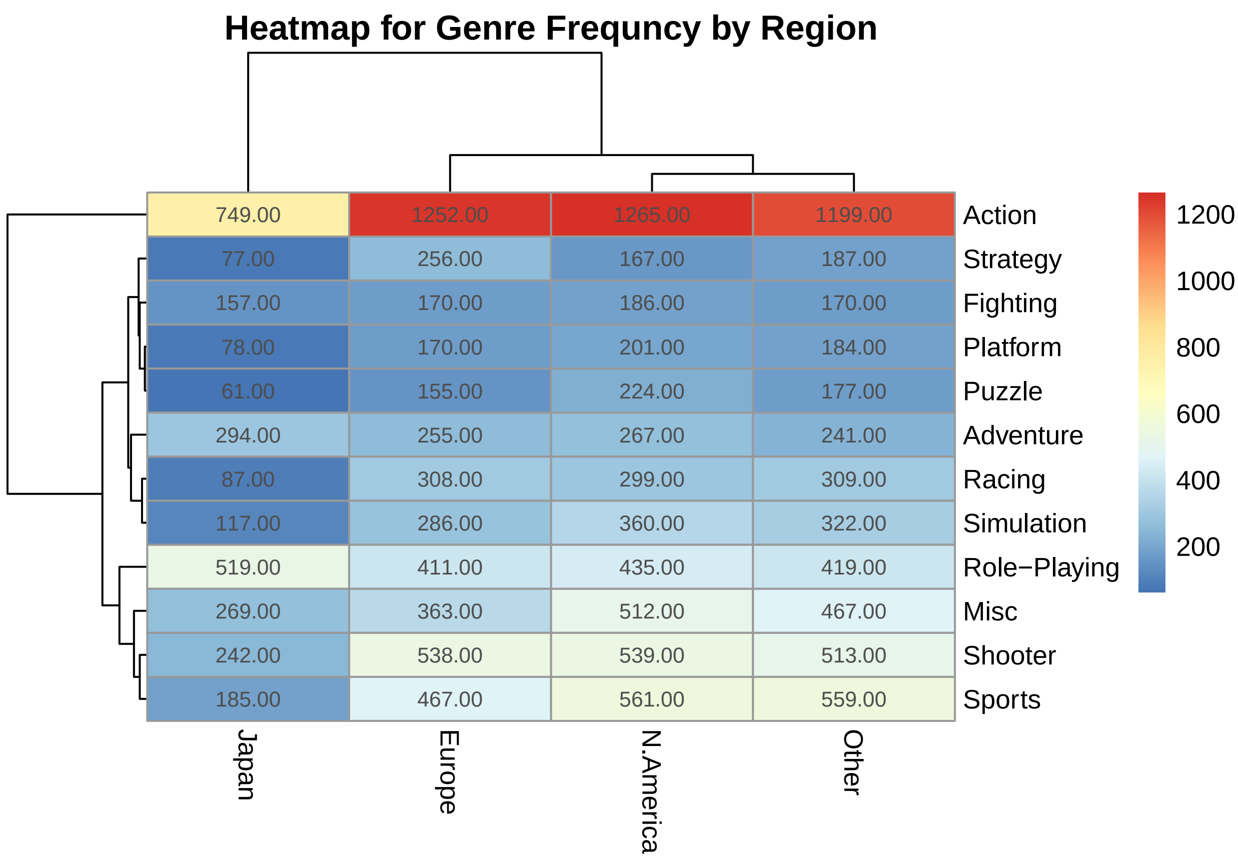



```
set.seed(1007)

Vidgames.df %>%
  ggparcoord(
    scale = 'globalminmax',
    columns = 4:7, groupColumn = 2, order = "anyClass",
    showPoints = TRUE,
    title = "Paralell Cordinate Plot Genre",
    alphaLines = 0.3
  ) +
  scale_color_viridis(discrete=TRUE)+
  theme(
    plot.title = element_text(size=10)
  ) +
  xlab("region")
```




```r
japan1 <- Vidgames.df%>%
  filter(jp_sales>0)%>%
  dplyr::select(genre, jp_sales)%>%
  dplyr::count(genre, name = "Japan")

europe1 <- Vidgames.df%>%
  filter(eu_sales>0)%>%
  dplyr::select(genre, eu_sales)%>%
  dplyr::count(genre, name = "Europe")

other1 <- Vidgames.df%>%
  filter(other_sales>0)%>%
  dplyr::select(genre, other_sales)%>%
  dplyr::count(genre, name = "Other")

NAmerica1 <- Vidgames.df%>%
  filter(na_sales>0)%>%
  dplyr::select(genre, na_sales)%>%
  dplyr::count(genre, name = "N.America")

genre.heatmap <- merge(NAmerica1,other1, by = 'genre')%>%
  merge(europe1,by = 'genre')%>%
  merge(japan1,by = 'genre')

cn <-c(genre.heatmap$genre)
heatmap1 <- genre.heatmap%>%
```

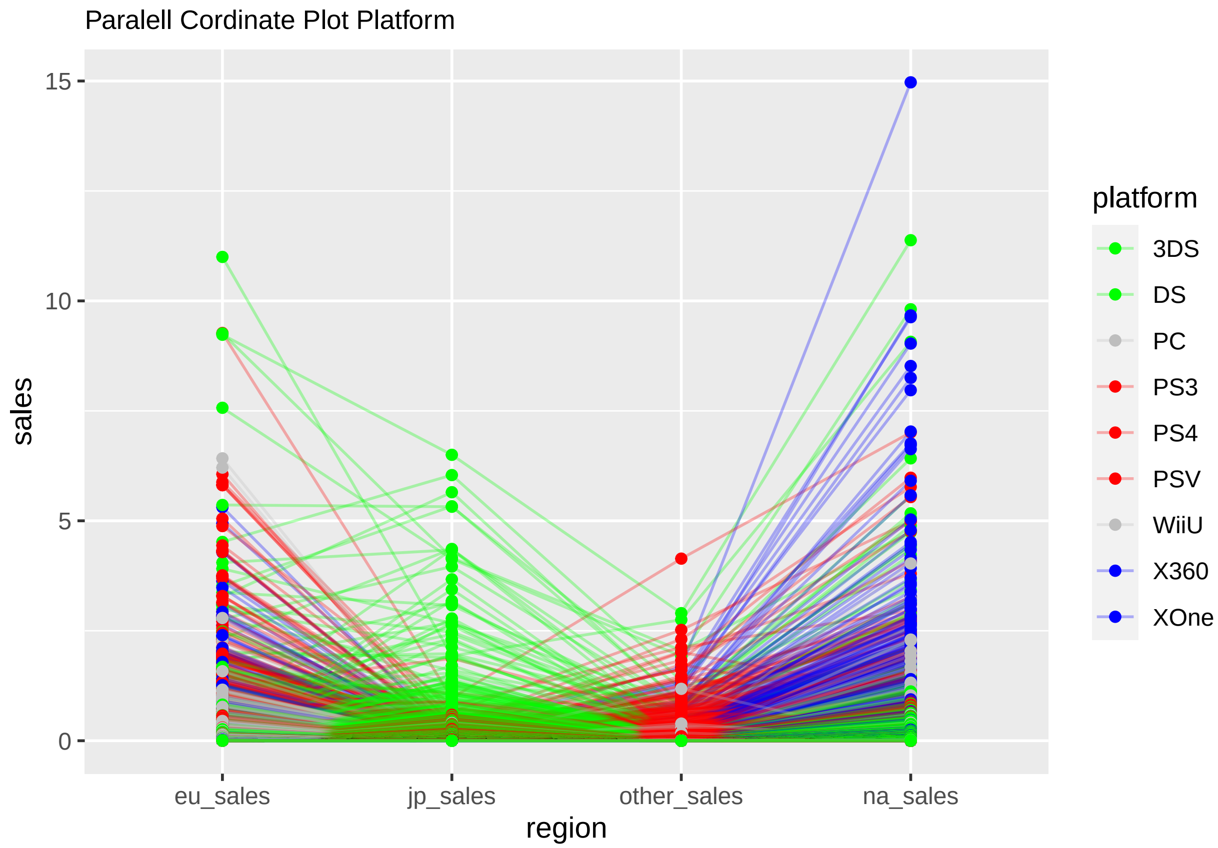
```
  dplyr::select(N.America:Japan)

pheatmap(heatmap1,display_numbers = T,
         labels_row = cn,
         main = "Heatmap for Genre Frequncy by Region")
```



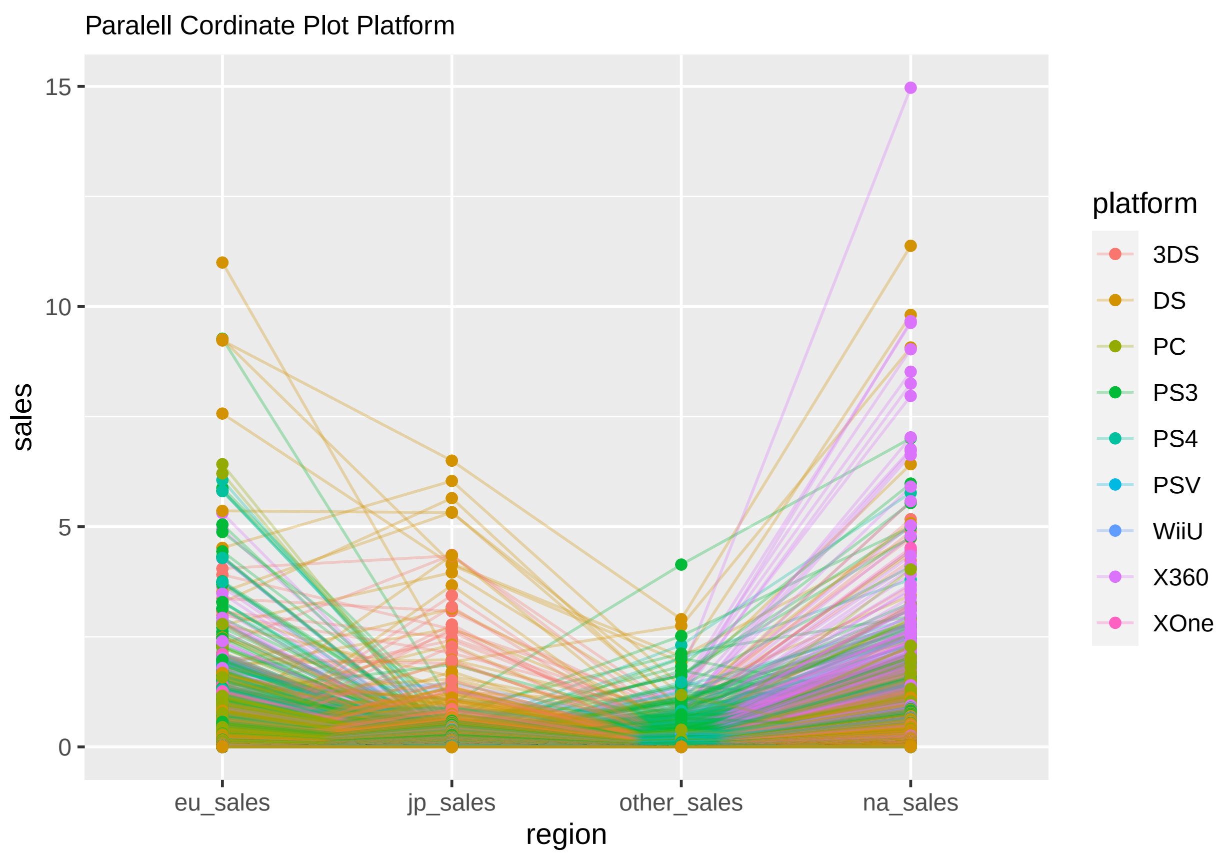
There appears to be some segregation in JP_Sales in terms of the platform category. Arguably the 3DS and DS are distinguishable from playstation platforms and a case could be made for the existence of natural groupings within these platforms. Additionally, there appears to be some segregation in Other_Sales in terms of the platform category. Arguably the playstation platforms are distinguishable from other platforms and a case could be made for the existence of natural groupings within these platforms. However, we cannot see any conclusive segregation as the groupings seem to be driven by outliers and the imbalance of observations within our platform variable as seen in the platform heatmap below. Most notably this plot brings to light the imbalance of platform presence within different regions and the grouping of outliers within these specific platforms in specific regions and this is seemingly driving these "visual groupings". Additionally, the relatively moderate cardinality of our platform makes parallel coordinate plots very hard to visualize and interpret.

```
Vidgames.df %>%
  ggparcoord(
    columns = 4:7, groupColumn = 1,
    scale = 'globalminmax',
    showPoints = TRUE,
    title = "Paralell Cordinate Plot Platform",
    alphaLines = 0.3
  ) +
```

```
  scale_color_manual(values=c( "green", "green", "grey", "red", "red", "red",
                               "grey", "blue","blue") )+
  theme(
    plot.title = element_text(size=10)
  ) +
  xlab("region")+
  ylab("sales")
```

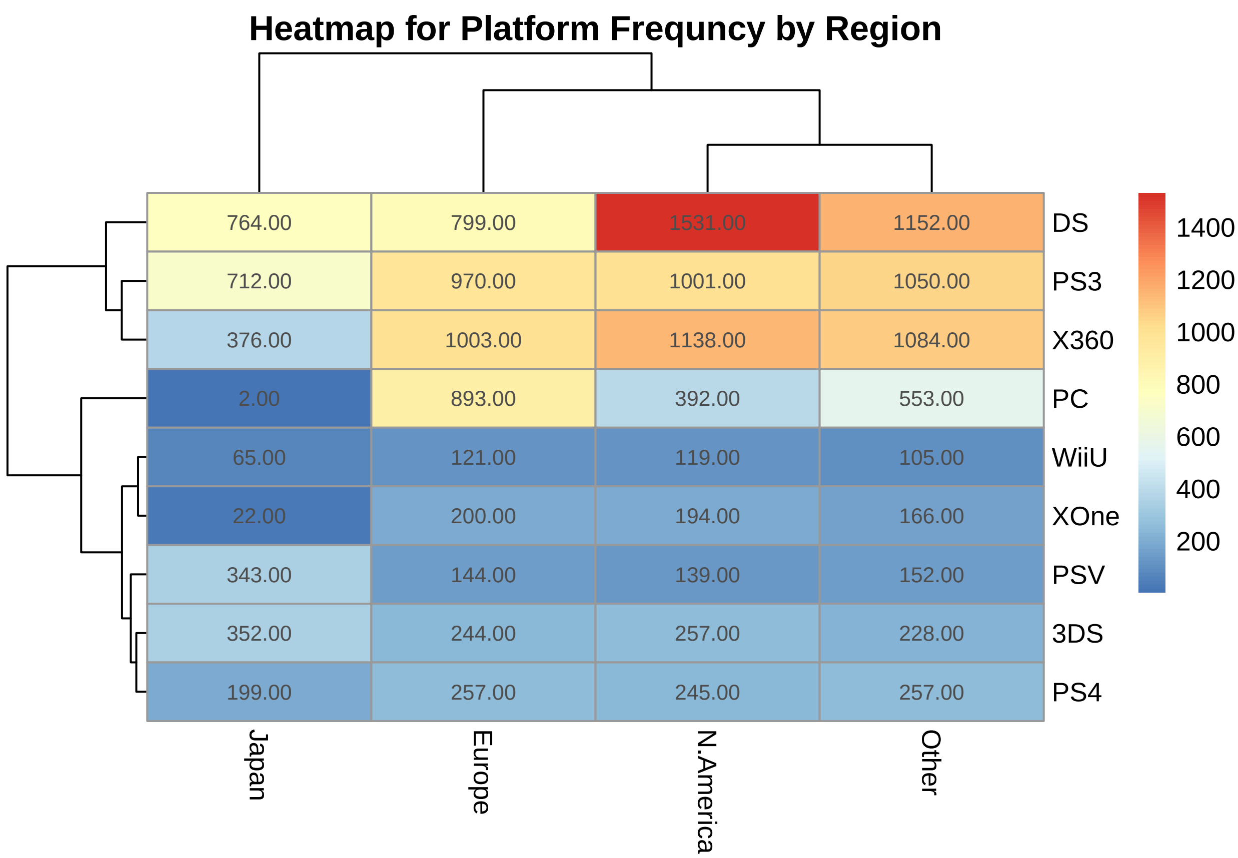



```
Vidgames.df %>%
  ggparcoord(
    columns = 4:7, groupColumn = 1,
    scale = 'globalminmax',
    showPoints = TRUE,
    title = "Paralell Cordinate Plot Platform",
    alphaLines = 0.3
  ) +
  theme(
    plot.title = element_text(size=10)
  ) +
  xlab("region")+
  ylab("sales")
```




```
japan <- Vidgames.df%>%
  filter(jp_sales>0)%>%
  dplyr::select(platform, jp_sales)%>%
  dplyr::count(platform, name = "Japan")

europe <- Vidgames.df%>%
  filter(eu_sales>0)%>%
  dplyr::select(platform, eu_sales)%>%
  dplyr::count(platform, name = "Europe")

other <- Vidgames.df%>%
  filter(other_sales>0)%>%
  dplyr::select(platform, other_sales)%>%
  dplyr::count(platform, name = "Other")

NAmerica <- Vidgames.df%>%
  filter(na_sales>0)%>%
  dplyr::select(platform, na_sales)%>%
  dplyr::count(platform, name = "N.America")

platform.heatmap <- merge(NAmerica,other, by = 'platform')%>%
  merge(europe,by = 'platform')%>%
  merge(japan,by = 'platform')

cn <-c(platform.heatmap$platform)
heatmap <- platform.heatmap%>%
```

```
  dplyr::select(N.America:Japan)

pheatmap(heatmap,display_numbers = T,
         labels_row = cn,
         main = "Heatmap for Platform Frequncy by Region")
```



Outline the variables to be considered in the PCA

For our Principal component analysis we will consider all 7 variables within our data set. Our 4 numeric variables (JP_Sales, NA_Sales, EU_Sales, and Other_Sales) and our 3 factor variables (Platform, Genre, and Global). We will need to convert our categorical variables (Platform, Genre, and Global) to dummy variables because PCA works purely for numerical variables and not categorical variables. Additionally, our predictor variables are all on different scales (they exhibit different means and different standard deviations) and will need to be normalized. This is because PCA works better with data that is all on the same scale. Normalization requires the centering of our variables (mean subtracted from the data) then the scaling of our variables (standard deviation of a variable is divided out of the data). Therefore, we will normalize our predictors to have a standard deviation of one and a mean of zero. Our final step is to perform the actual PCA.

PCA Analysis

```
games_recipe <- recipe(na_sales ~ ., data = Vidgames.df ) %>%
  step_dummy(platform,genre,global)%>%
```

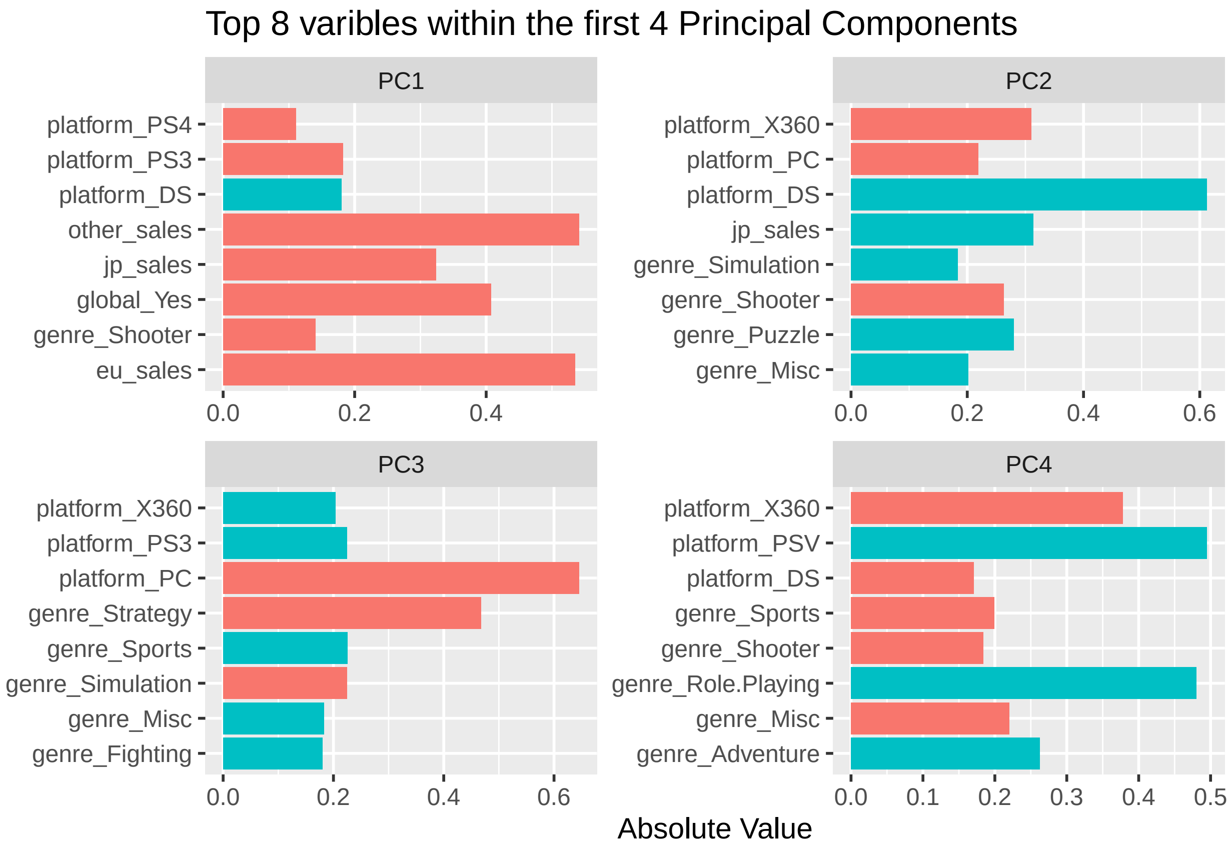
```r
  step_normalize(all_predictors())%>%
  step_pca(all_predictors()) # Do the PCA.

#prepping data
game_prepped <- games_recipe%>%
  prep()

tidy( game_prepped ) %>%
  gt()%>%
  tab_header(
    title = "PCA Steps",
    subtitle = "All 3 Steps Provided"
  )
```

PCA Steps

All 3 Steps Provided

number	operation	type	trained	skip	id
1	step	dummy	TRUE	FALSE	dummy_AZ61Q
2	step	normalize	TRUE	FALSE	normalize_22RtM
3	step	pca	TRUE	FALSE	pca_IYeqo

```r
#Plotting PCA
tidy( game_prepped, 3 ) %>%
  filter( component %in% c("PC1", "PC2", "PC3", "PC4") ) %>%
  mutate( component = fct_inorder( component ) )%>%
  group_by( component ) %>%
  top_n(8, abs(value) ) %>%
  ungroup() %>%
  ggplot( aes( x = abs(value), y = terms, fill = value > 0 ) ) +
  geom_col(show.legend = F) +
  facet_wrap( ~ component, scales = "free") +
  ylab(NULL)+ # We do not need the y axis label.
  xlab("Absolute Value")+
  labs(title = "Top 8 varibles within the first 4 Principal Components")
```



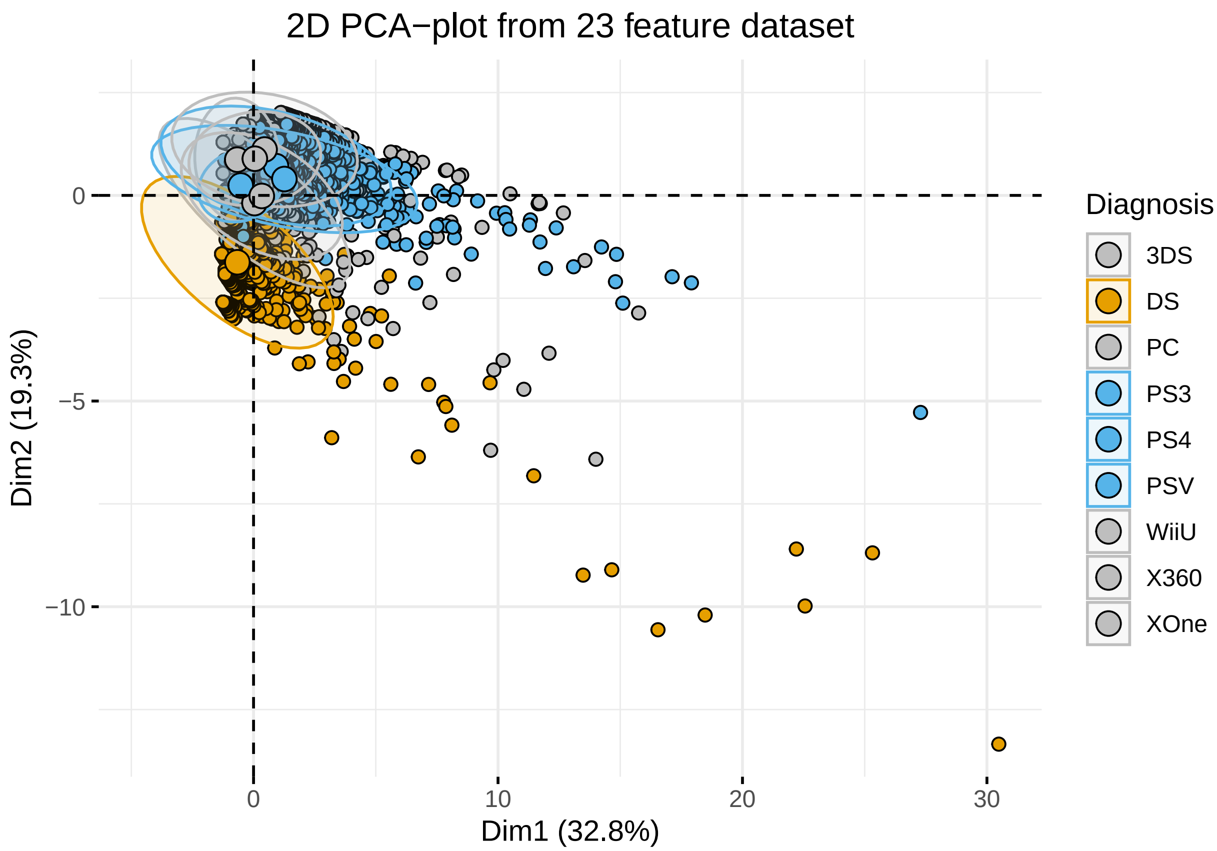
Our plots above allow us to understand which variables are the highest contributing predictors in the first four principal components in terms of contributing to the variance in our data.

Notably, EU_Sales, Other_Sales, Global_yes and JP_Sales are the most important in the first component. Additionally, platform_DS, platform_X360, JP_sales, and genre_puzzle are the most important in the second component. Subsequently, platform_PC and genre_strategy are the most important in the third component. Finally, platform_PSV, genre_Role.Playing, platform_X360, and genre_Adventure are the most important in the fourth component.

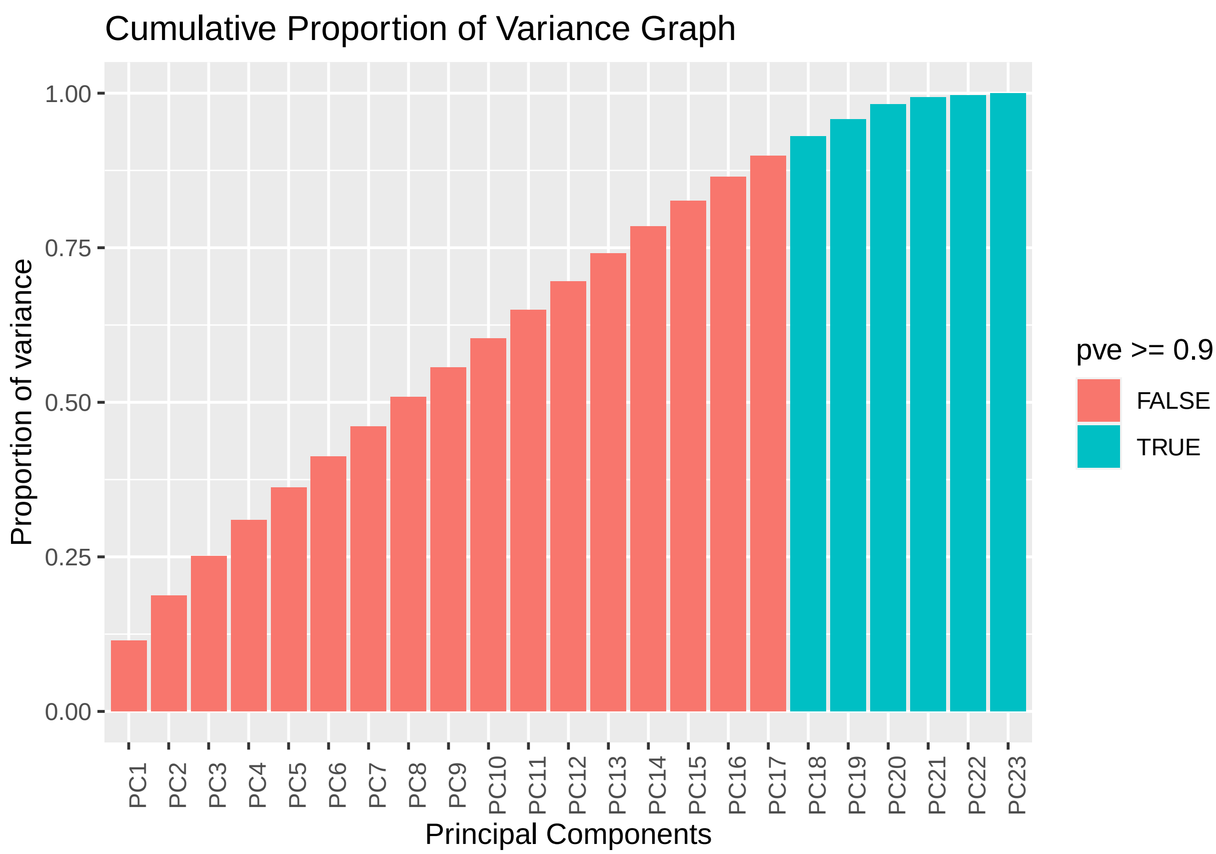
There seems to be some separation of the platforms based on the first two principal components. Notably, the plot below seems to indicate some separation between DS and playstation platforms. This looks interesting for our regression analysis. However, we must determine whether we want to use these PCA components in our model by examining the proportion of variation explained by the components.

```
games_juiced <- juice(game_prepped)

pca <- prcomp(games_juiced)

fviz_pca_ind(pca,geom.ind = "point", pointshape = 21,
             fill.ind = Vidgames.df$platform,
             pointsize = 2,
             col.ind = "black",
             palette = c("grey", "#E69F00", "grey", "#56B4E9", "#56B4E9",
                         "#56B4E9", "grey", "grey","grey"),
             addEllipses = TRUE,
             label = "var",
             col.var = "black",
```

```r
              repel = TRUE,
              legend.title = "Diagnosis") +
  ggtitle("2D PCA-plot from 23 feature dataset") +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
#Proportion of variance
sdev <- game_prepped$steps[[3]]$res$sdev
ve <- sdev^2 / sum(sdev^2)

PC.pve <- tibble(
  pc = fct_inorder( str_c("PC", 1:23) ),
  pve = cumsum( ve )
)

PC.pve %>%
  ggplot( aes( x = pc,
               y = pve,
               fill = pve >= 0.9 ) ) +
  geom_col() +
  labs(x = "Principal Components", y = "Proportion of variance",
       title = "Cumulative Proportion of Variance Graph")+
  theme( axis.text.x = element_text( angle = 90 ) )
```




```
PC.pve%>%
  gt()%>%
  tab_header(
    title = "Cumulative Proportion of Variance Explained",
    subtitle = "By each Principal Component"
  )
```

Cumulative Proportion of Variance Explained

By each Principal Component

pc	pve
PC1	0.1143621
PC2	0.1874545
PC3	0.2512518
PC4	0.3098823
PC5	0.3621224
PC6	0.4120742
PC7	0.4609451
PC8	0.5089074
PC9	0.5566817
PC10	0.6030866
PC11	0.6493775
PC12	0.6952654
PC13	0.7405745
PC14	0.7845047

PC15	0.8259150
PC16	0.8646040
PC17	0.8990426
PC18	0.9306143
PC19	0.9580488
PC20	0.9824079
PC21	0.9934391
PC22	0.9969599
PC23	1.0000000

For the first four components we are only explaining 31% of the variation in our data. To explain at least 90% of the variation, we need to consider 18 principal components. Since this only reduces our dimension by 5, and complicates our analysis as well as our ability to explain our model, we have chosen to forgo using these components for modelling purposes.

Discuss the relationships between the response variable and the numeric predictors.

Firstly, we need to determine which correlation test to use. Notably, is NA_Sales and each individual numeric predictor bivariate normal (follow a bivatiarte normal distribution)? In our data cleaning section we illustrated the distributions of all our numeric predictors and outcome variable as having unimodal right skews (non-normal distributions). Additionally, a shapiro-wilk test and QQ-plot visualizations confirm the non-normality of our variables. Notably, our scatter plot shows evidence of heteroscedasticity and the presence of numerous outliers. Despite all these tests and assumptions, what we are most concerned with is the influence of outliers on our Pearson Correlation. The Spearman correlation is less sensitive than the Pearson correlation to strong outliers which make up a significant proportion of our data set. Therefore, due to Spearman correlation being robust to outliers we have chosen this non-parametric test to describe the relationships between the response variable and the numeric predictor.

```r
#Sampling data to be used for shapiro wilks
sampled <- Vidgames.df%>%
  dplyr::select(eu_sales:na_sales)%>%
  sample_n(4999)

# Shapiro-Wilk normality test for na_sales
sw.na <- shapiro.test(sampled$na_sales)

# Shapiro-Wilk normality test for eu_sales
sw.eu <-shapiro.test(sampled$eu_sales)

# Shapiro-Wilk normality test for jp_sales
sw.jp <-shapiro.test(sampled$jp_sales)

# Shapiro-Wilk normality test for other_sales
sw.other <-shapiro.test(sampled$other_sales)

eu_sales.corr <- cor.test(Vidgames.df$eu_sales,Vidgames.df$na_sales,method = "spearman")
```

```
## Warning in cor.test.default(Vidgames.df$eu_sales, Vidgames.df$na_sales, : Cannot
## compute exact p-value with ties
```

```r
jp_sales.corr <- cor.test(Vidgames.df$jp_sales,Vidgames.df$na_sales,method = "spearman")
```

```
## Warning in cor.test.default(Vidgames.df$jp_sales, Vidgames.df$na_sales, : Cannot
## compute exact p-value with ties
```

```r
other_sales.corr <- cor.test(Vidgames.df$other_sales,Vidgames.df$na_sales,method = "spearman")
```

```
## Warning in cor.test.default(Vidgames.df$other_sales, Vidgames.df$na_sales, :
## Cannot compute exact p-value with ties
```

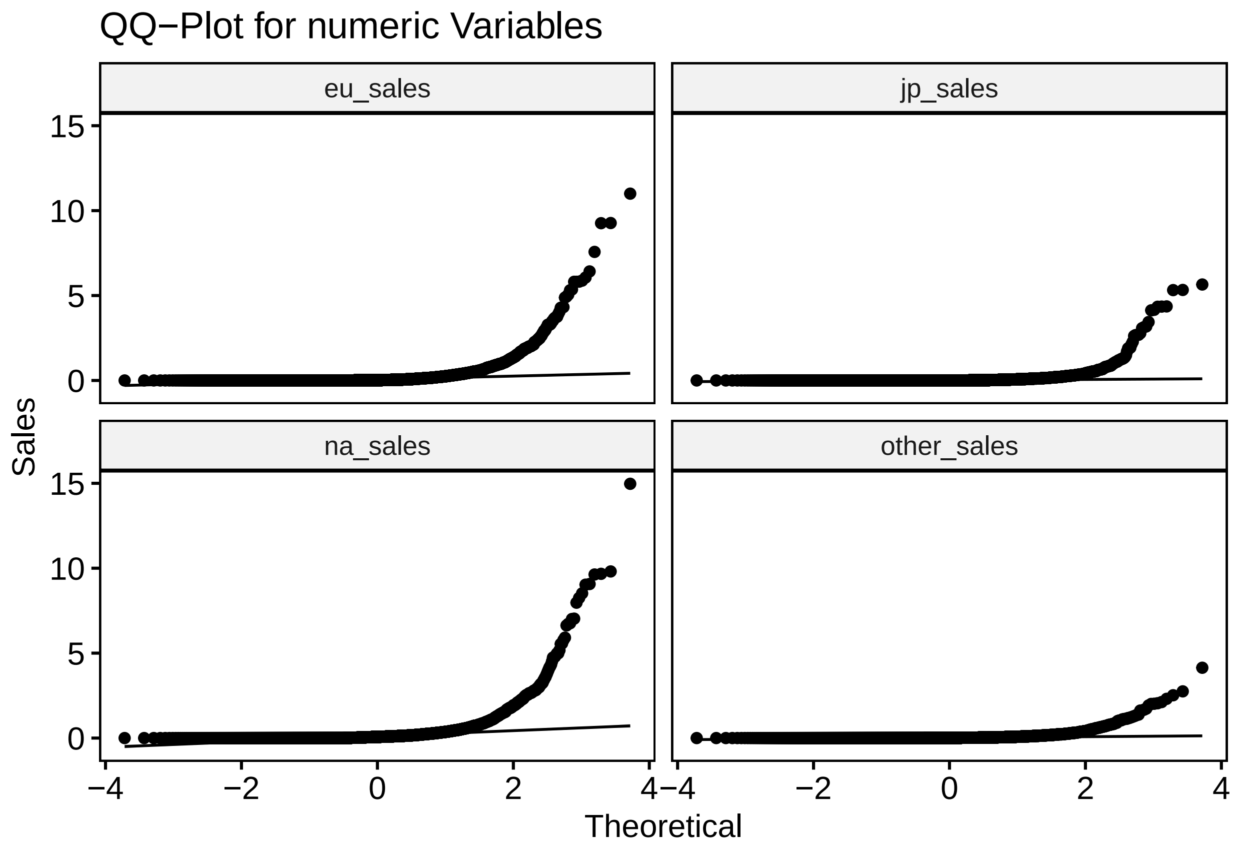
```r
spearman.table <- tribble(
  ~"Region",~"Correllation",~"Normality_Test",~"P-Value",
  'eu_sales',eu_sales.corr$estimate,"Shapiro-Wilk",sw.eu$p.value,
  'jp_sales',jp_sales.corr$estimate,"Shapiro-Wilk",sw.jp$p.value,
  'other_sales',other_sales.corr$estimate,"Shapiro-Wilk",sw.other$p.value
)
spearman.table%>%
  gt()%>%
  tab_header(
    title = "Examining Normality and Corellation",
    subtitle = "Non-Parametric Tests  (Spearman & Shapiro-Wilks)"
  )
```

Examining Normality and Corellation

Non-Parametric Tests (Spearman & Shapiro-Wilks)

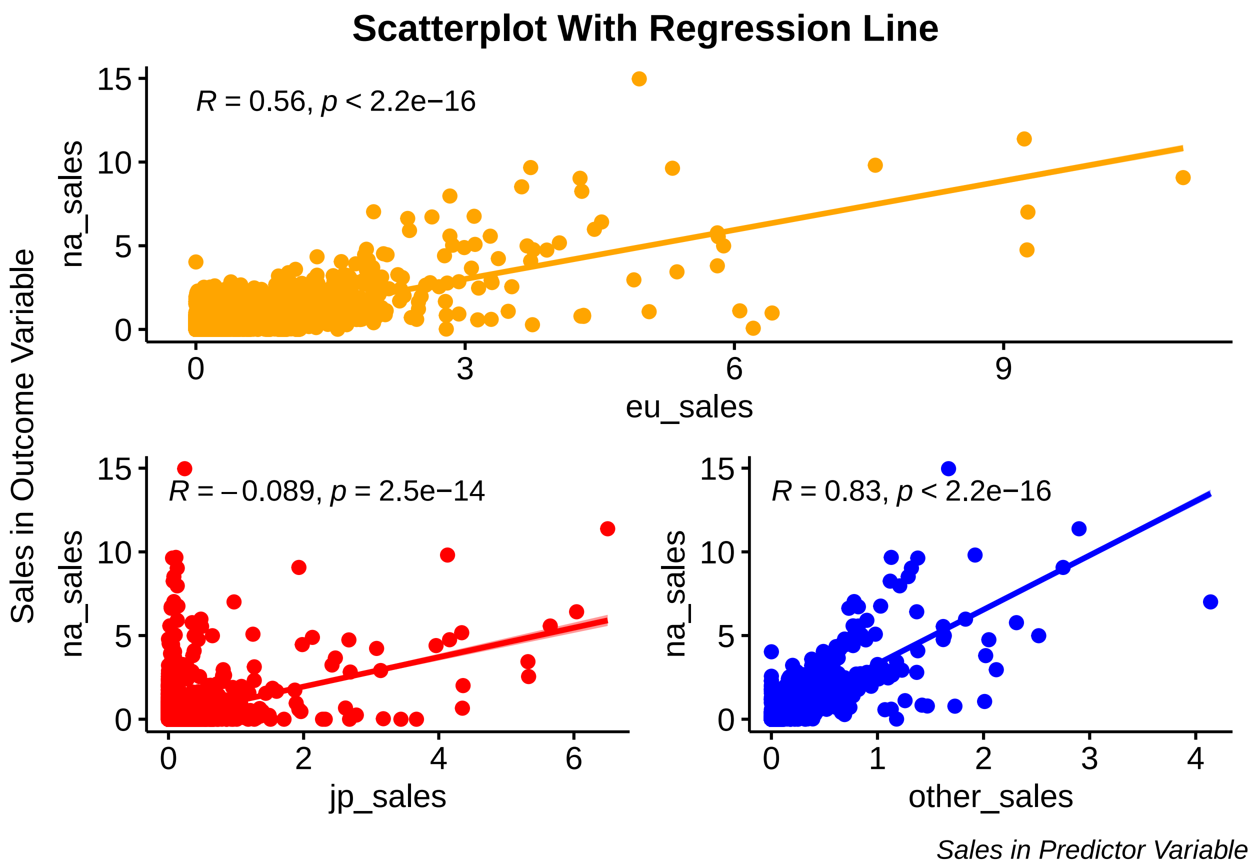
Region	Correllation	Normality_Test	P-Value
eu_sales	0.5649370	Shapiro-Wilk	5.517951e-87
jp_sales	-0.0888493	Shapiro-Wilk	4.988288e-91
other_sales	0.8299318	Shapiro-Wilk	3.408515e-87

```r
sampled%>%
  pivot_longer(eu_sales:na_sales)%>%
  ggqqplot('value', ylab = "Sales",
           title = "QQ-Plot for numeric Variables",
           facet.by = 'name')
```




```
EU_NA.corr <- Vidgames.df%>%
  dplyr::select(eu_sales:na_sales)%>%
  ggscatter(x = 'eu_sales', y = 'na_sales',
            add = "reg.line", conf.int = TRUE,
            cor.coef = TRUE, cor.method = "spearman",
            xlab = "eu_sales", ylab = "na_sales",color = "orange")

JP_NA.corr <- Vidgames.df%>%
  dplyr::select(eu_sales:na_sales)%>%
  ggscatter(x = 'jp_sales', y = 'na_sales',
            add = "reg.line", conf.int = TRUE,
            cor.coef = TRUE, cor.method = "spearman",
            xlab = "jp_sales", ylab = "na_sales",color = "red")

Other_NA.corr <- Vidgames.df%>%
  dplyr::select(eu_sales:na_sales)%>%
  ggscatter(x = 'other_sales', y = 'na_sales',
            add = "reg.line", conf.int = TRUE,
            cor.coef = TRUE, cor.method = "spearman",
            xlab = "other_sales", ylab = "na_sales", color = "blue")

# Arrange the plots on the same page
figure <- ggarrange(EU_NA.corr,
          ggarrange(JP_NA.corr, Other_NA.corr,
                    ncol= 2),
          nrow = 2)
```

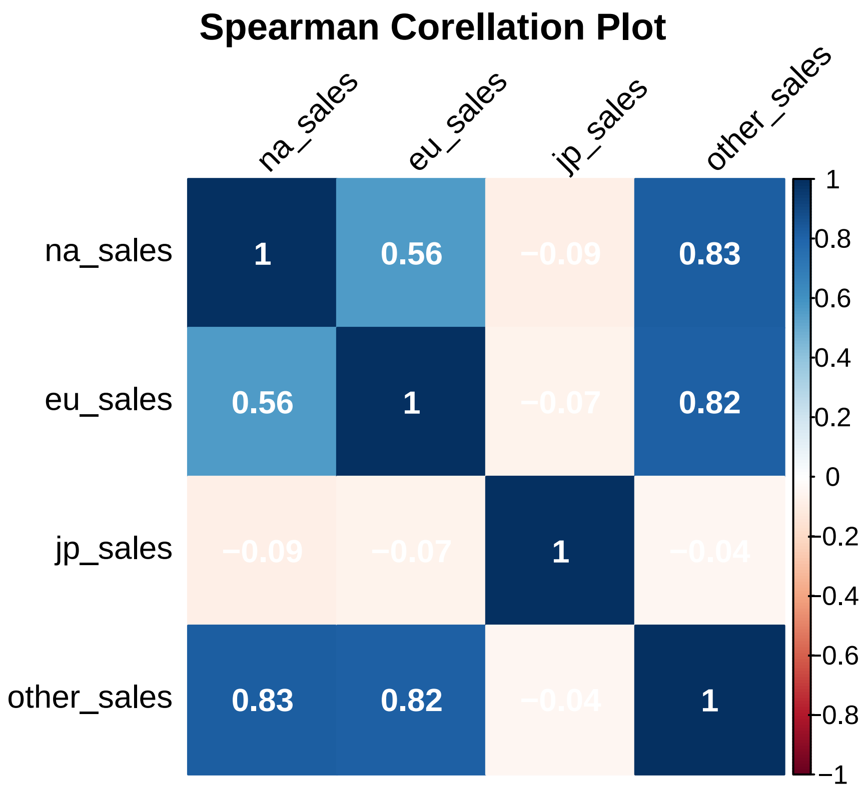
```
annotate_figure(figure,
                top = text_grob("Scatterplot With Regression Line",
                                color = "black", face = "bold", size = 14),
                bottom = text_grob("Sales in Predictor Variable",
                                   color = "black",
                                   hjust = 1, x = 1, face = "italic", size = 10),
                left = text_grob("Sales in Outcome Variable",
                                 color = "black", rot = 90),
                fig.lab.face = "bold"
)
```



We see a strong positive monotonic relationship between NA_Sales and Other_Sales (0.83). We see a moderate positive monotonic relationship between NA_Sales and EU_Sales (0.56). Finally, we see a weak negative monotonic relationship between NA_Sales and JP_Sales (-0.089). Notably, our three numeric variables indicate monotonic relationships with NA_Sales vary quite considerable with other_sales illustrating the strongest monotonic relationship with NA_Sales. Additionally, the weak monotonic relationship of JP_Sales and NA_Sales indicates that we should probably remove this variable.

```
#Correlation Matrix
correllation <-Vidgames.df%>%
  dplyr::select(na_sales,eu_sales:other_sales)%>%
  cor(method = "spearman",)

#Correlation Plot
corrplot(correllation, method="color",
```

```r
         addCoef.col = "white", # Add coefficient of correlation
         tl.col="black", tl.srt=45, title = "Spearman Corellation Plot",
         mar = c(2, 0, 1, 0))
```



Notably, we do see evidence of multicolinearity between our EU_Sales and Other_Sales. However, because lasso is a regularized linear model as squishes large coefficients closer to zero we do not have to worry too much about this.

```r
#Multicollinearity
Multicollinearity<- lm(na_sales ~ eu_sales + jp_sales + other_sales,
                       data = Vidgames.df)
ols_vif_tol(Multicollinearity)%>%
  gt()%>%
  tab_header(
    title = "Multicollinearity of Numeric Predictors",
    subtitle = "Variance Inflation Factor"
  )
```

Multicollinearity of Numeric Predictors

Variance Inflation Factor

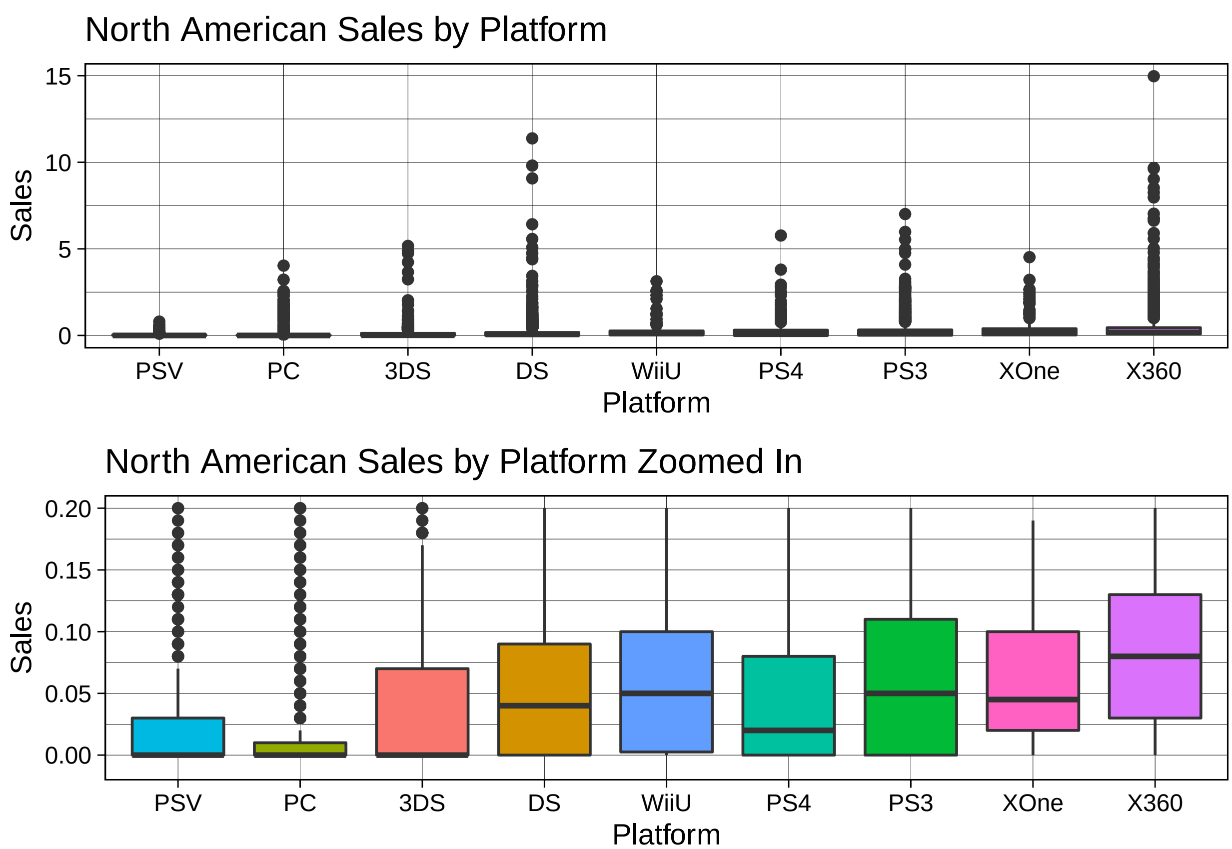
Variables	Tolerance	VIF
eu_sales	0.1394263	7.172247
jp_sales	0.8161126	1.225321

other_sales	0.1474062	6.783975

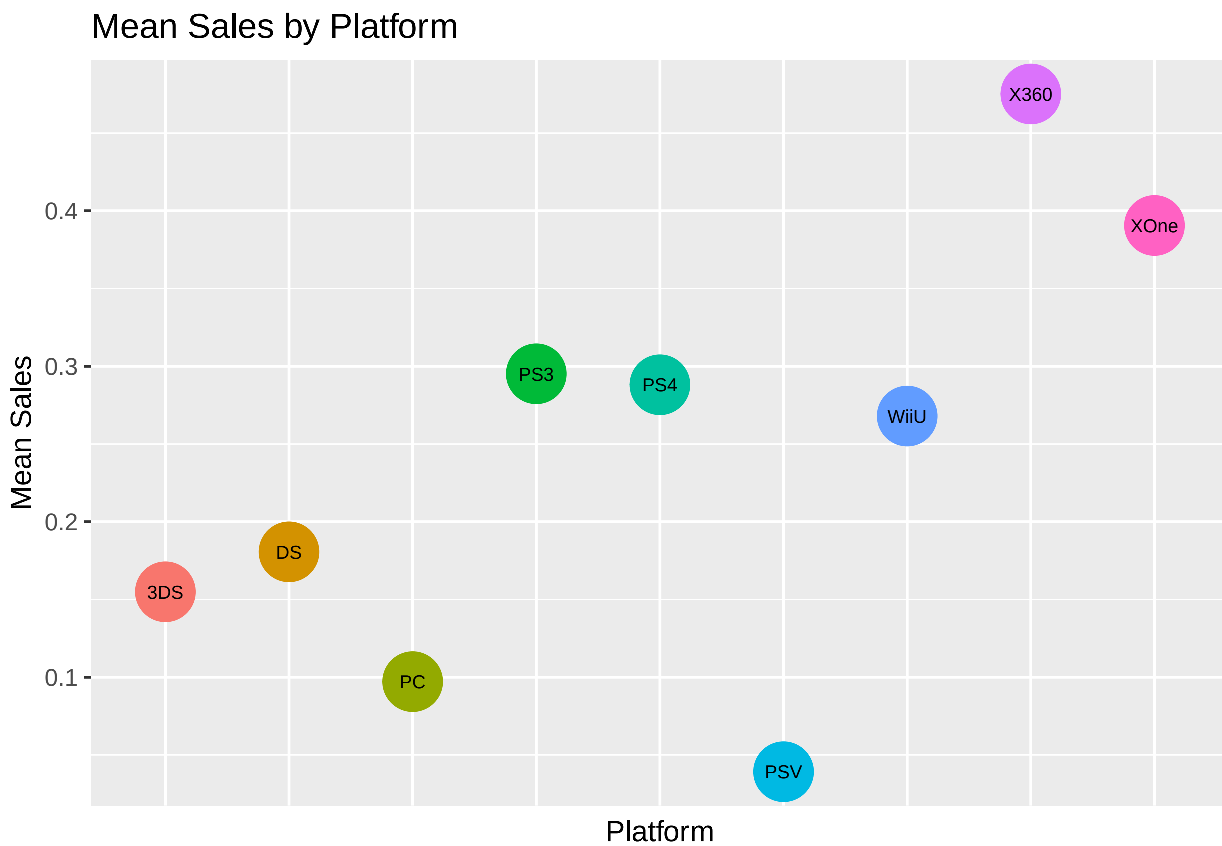
Relationships between the response and Platform.

```
logged <- Vidgames.df%>%
  ggplot(aes(x = reorder(platform,na_sales, FUN = 'mean'), y = na_sales))+
  geom_boxplot(aes(fill = platform), show.legend = FALSE)+
  labs(x = "Platform", y = "Sales",
       title = "North American Sales by Platform Zoomed In ")+
  theme_linedraw()+
  ylim(-0.01,0.2)

unlogged <- Vidgames.df%>%
  ggplot(aes(x = reorder(platform,na_sales,FUN = 'mean'), y = na_sales))+
  geom_boxplot(aes(fill = platform), show.legend = FALSE)+
  labs(x = "Platform", y = "Sales",
       title = "North American Sales by Platform")+
  theme_linedraw()

ggarrange(unlogged,logged,
          ncol = 1,nrow = 2,
          heights = c(2,2))
```

```
## Warning: Removed 1927 rows containing non-finite values (stat_boxplot).
```



```
Vidgames.df %>%
  group_by(platform) %>%
  dplyr::summarise(mean_sales = mean( na_sales ) ) %>%
  ggplot( aes( x = platform, y = mean_sales, colour = platform ) ) +
  geom_point( size = 10, show.legend = FALSE) +
  geom_text( aes( label = platform ), colour = "black", size = 2.5)+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())+
  labs(x =  "Platform",y = "Mean Sales", title = "Mean Sales by Platform")
```



The significant amount of outliers in our data may influence the performance of our model. Therefore, we might consider dropping outliers. We can see a clear standout of Xbox consoles (X360 & XOne) in terms of mean NA_Sales closely followed by our playstation Platforms (PS3 & PS4). Most notably PSV seems to have the lowest mean NA_Sales out of all platforms. Interestingly we do seeboth Nintendo platforms grouping together (3DS and DS). There appears to be significant differences among the means of our platforms in terms of NA_Sales which we will investigate using the non-parametric Kruskal Wallis Test.

```
#Kruskal Wallis Test
res.kruskal <- kruskal.test(na_sales ~ platform, data = Vidgames.df)
res.kruskal
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  na_sales by platform
## Kruskal-Wallis chi-squared = 1320.6, df = 8, p-value < 2.2e-16
```

```r
#Post ad hoc test
pwc <- dunn_test(na_sales ~ platform, data = Vidgames.df, p.adjust.method = "bonferroni")
pwc%>%
  filter(p.adj.signif!="ns")%>%
  dplyr::select(-p)
```

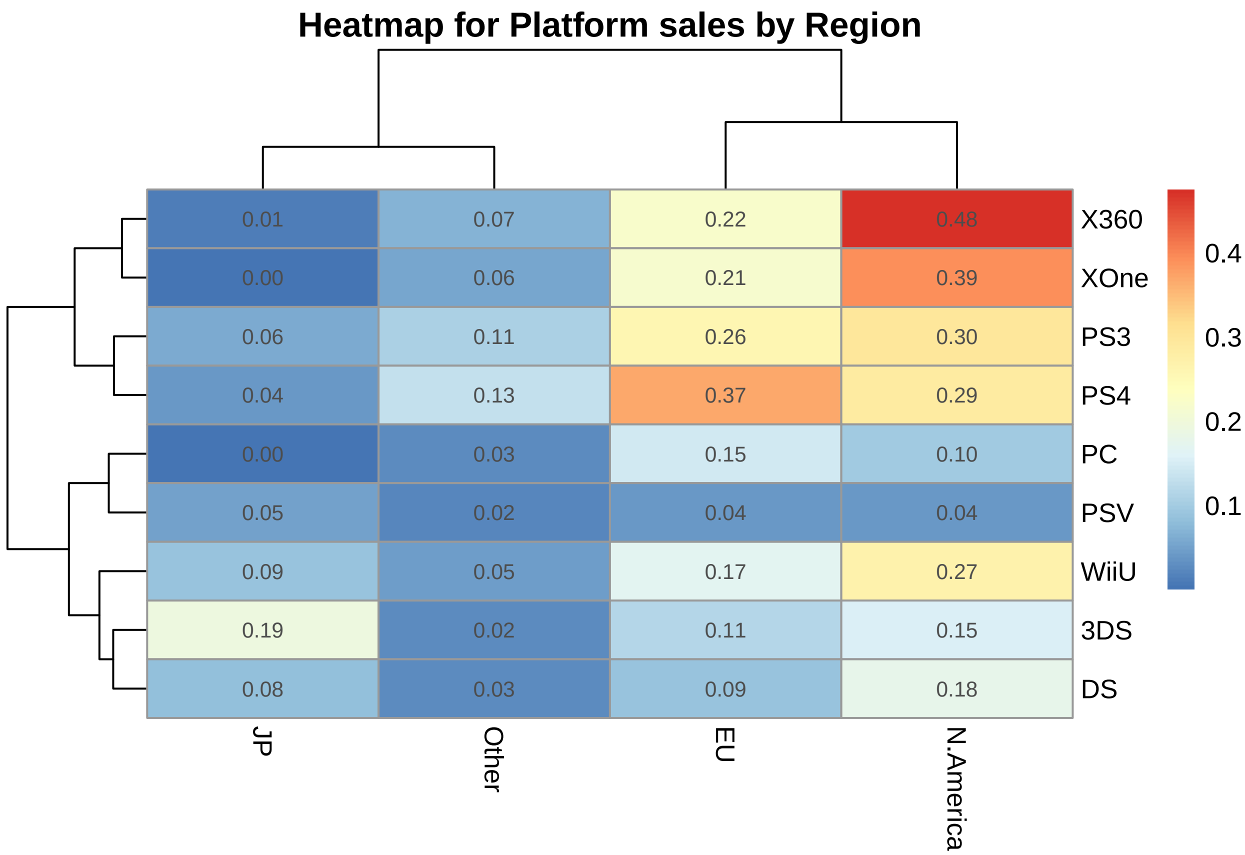
```
## # A tibble: 28 x 8
##    .y.      group1 group2    n1    n2 statistic    p.adj p.adj.signif
##    <chr>    <chr>  <chr>  <int> <int>     <dbl>    <dbl> <chr>
##  1 na_sales 3DS    DS       509  2163      6.46 3.80e- 9 ****
##  2 na_sales 3DS    PC       509   960     -5.82 2.16e- 7 ****
##  3 na_sales 3DS    PS3      509  1329     11.8  1.93e-30 ****
##  4 na_sales 3DS    PS4      509   336      6.22 1.82e- 8 ****
##  5 na_sales 3DS    PSV      509   413     -5.63 6.42e- 7 ****
##  6 na_sales 3DS    WiiU     509   143      6.76 4.97e-10 ****
##  7 na_sales 3DS    X360     509  1265     18.1  1.13e-71 ****
##  8 na_sales 3DS    XOne     509   213     10.2  8.18e-23 ****
##  9 na_sales DS     PC      2163   960    -16.4  4.30e-59 ****
## 10 na_sales DS     PS3     2163  1329      8.48 8.21e-16 ****
## # ... with 18 more rows
```

Interpretation

As the p-value is less than the significance level 0.05, we can conclude that there are significant differences between the NA_Sales means of our platforms.

Therefore, there is a statistically significant difference between platform groups as assessed using the Kruskal-Wallis test ($p = <2.2e-16$). Dunn's post hoc test showed that only 28 out of the possible 36 permutations were significant (Table Above). Notably, we do have an imbalanced design within our platform groups which may affect our test statistics.

```r
platform.pivot <- Vidgames.df %>%
  dplyr::select(platform,eu_sales:na_sales)%>%
  group_by(platform)%>%
  dplyr::summarise(EU = mean(eu_sales),
                   JP = mean(jp_sales),
                   N.America = mean(na_sales),
                   Other = mean(other_sales))

cn <-c(platform.pivot$platform)

heat <- platform.pivot%>%
  dplyr::select(EU:Other)

pheatmap(heat,display_numbers = T,
         labels_row = cn,
         main = "Heatmap for Platform sales by Region")
```



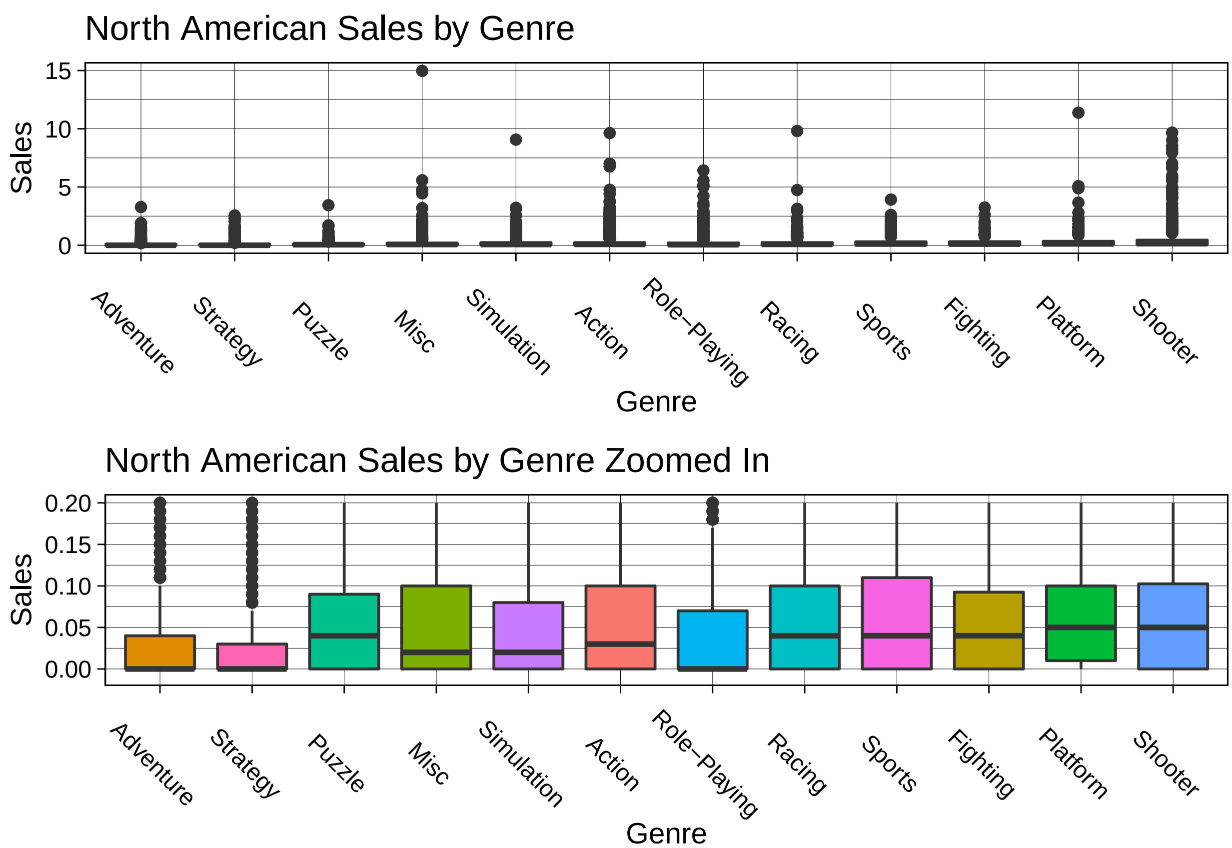
The table above shows the performance of each platform in different regions based upon mean sales. We can see that X360 is the top platform in N.America, PS4 performs best in Europe and Other, while unsurprisingly (as a Nintendo product) 3DS performs best in Japan. Notably, PS3 and PS4 performs relatively well across all markets. However, in terms of our outcome variable (N.America) Xbox and XOne seem to be the dominate platform.

Relationships between the response and Genre.

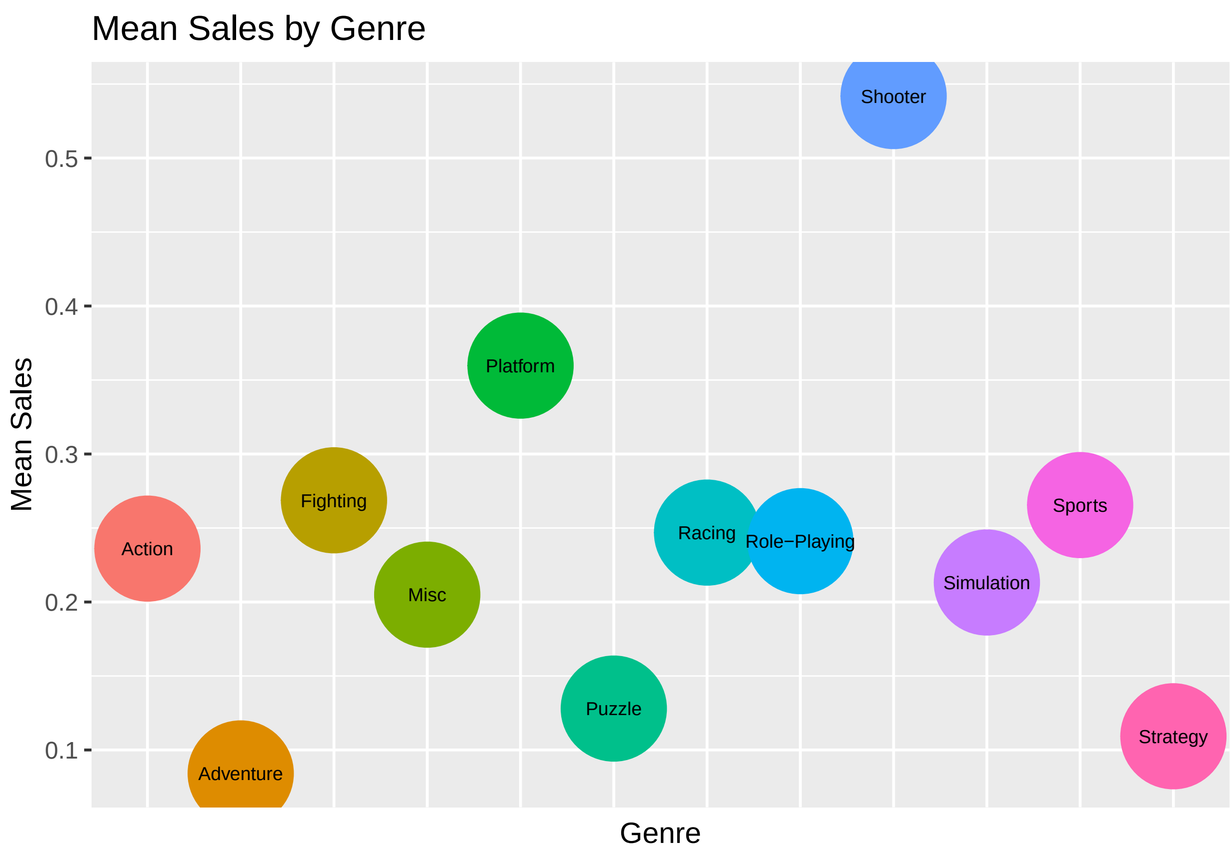
```
genre_log <- Vidgames.df%>%
  ggplot(aes(x = reorder(genre,na_sales), y = na_sales))+
  geom_boxplot(aes(fill = genre), show.legend = FALSE)+
  labs(x = "Genre", y = "Sales",
       title = "North American Sales by Genre Zoomed In")+
  theme_linedraw()+
  theme(axis.text.x = element_text(angle = -45))+
  ylim(-0.01,0.2)

genre_unlog <- Vidgames.df%>%
  ggplot(aes(x = reorder(genre,na_sales), y = na_sales))+
  geom_boxplot(aes(fill = genre), show.legend = FALSE)+
  labs(x = "Genre", y = "Sales",
       title = "North American Sales by Genre")+
  theme_linedraw()+
  theme(axis.text.x = element_text(angle = -45))
```

```
ggarrange(genre_unlog,genre_log,
          ncol = 1,nrow = 2,
          heights = c(3,3))
```

```
## Warning: Removed 1927 rows containing non-finite values (stat_boxplot).
```




```
Vidgames.df %>%
  group_by( genre ) %>%
  dplyr::summarise( mean_sales = mean( na_sales ) ) %>%
  ggplot( aes( x = genre, y = mean_sales, colour = as.factor( genre ) ) ) +
  geom_point( size = 18, show.legend = FALSE ) +
  geom_text( aes( label = genre ), colour = "black", size = 2.5)+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())+
  labs(x =  "Genre",y = "Mean Sales", title = "Mean Sales by Genre")
```



We can see a clear standout of shooter consoles in terms of mean NA_Sales. Most notably Puzzle, Strategy, and Adventure seems to have the lowest mean NA_Sales out of all genres. There appears to be significant differences among the means of our genres in terms of NA_Sales which we will investigate using the non-parametric Kruskal Wallis Test.

```
genre.kruskal <- kruskal.test(na_sales ~ genre, data = Vidgames.df)
genre.kruskal
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  na_sales by genre
## Kruskal-Wallis chi-squared = 470.81, df = 11, p-value < 2.2e-16
```

```
#Post ad hoc test
genre.pwc <- dunn_test(na_sales ~ genre, data = Vidgames.df, p.adjust.method = "bonferroni")
genre.pwc%>%
  filter(p.adj.signif!="ns")%>%
  dplyr::select(-p)
```

```
## # A tibble: 42 x 8
##    .y.      group1    group2          n1    n2 statistic    p.adj p.adj.signif
##    <chr>    <chr>     <chr>        <int> <int>     <dbl>    <dbl> <chr>
##  1 na_sales Action    Adventure     1789   583     -12.2  1.61e-32 ****
##  2 na_sales Action    Platform      1789   233       4.36 8.72e- 4 ***
```

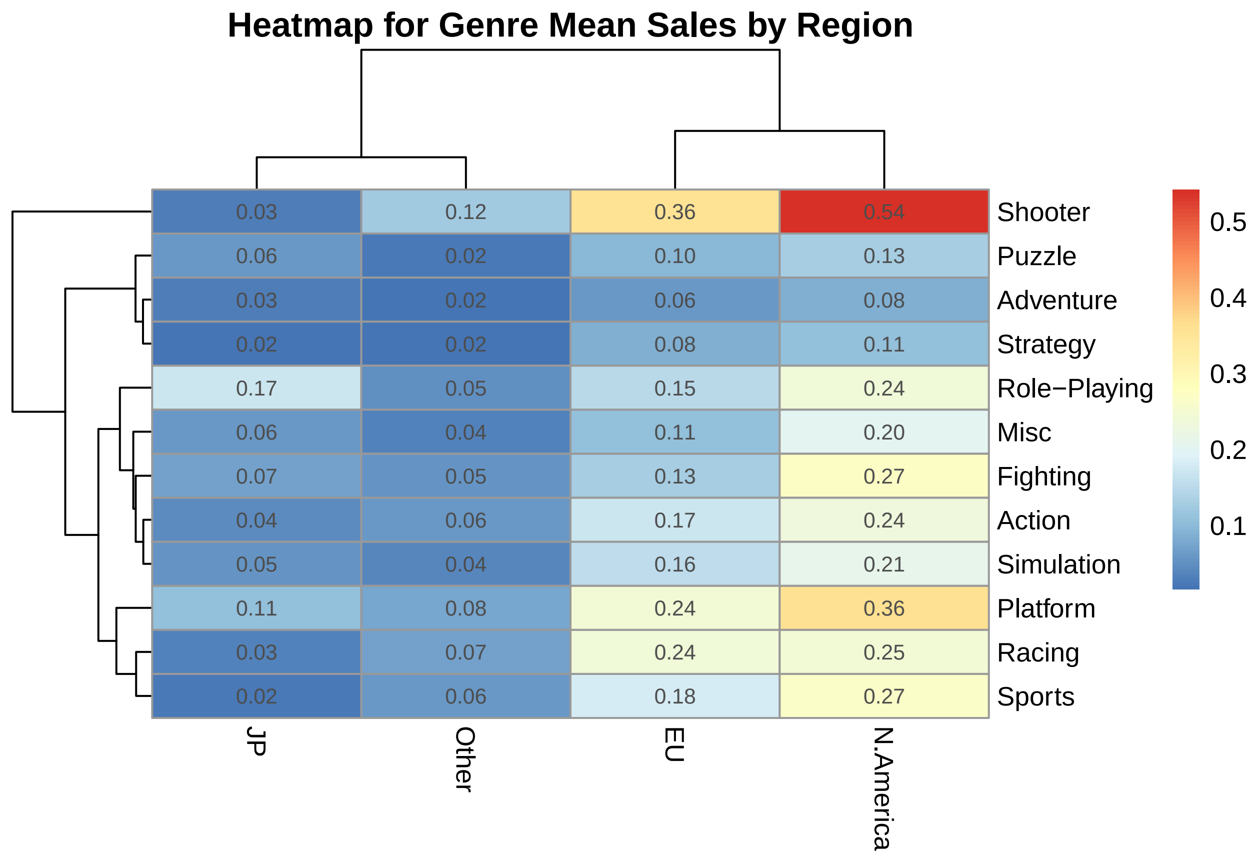
```
##  3 na_sales Action    Role-Playing  1789   734     -4.33 9.98e- 4 ***
##  4 na_sales Action    Shooter       1789   638      7.92 1.59e-13 ****
##  5 na_sales Action    Strategy      1789   352     -9.74 1.28e-20 ****
##  6 na_sales Adventure Fighting       583   242      9.86 3.90e-21 ****
##  7 na_sales Adventure Misc           583   795      8.12 3.14e-14 ****
##  8 na_sales Adventure Platform       583   233     11.4  1.87e-28 ****
##  9 na_sales Adventure Puzzle         583   301      5.56 1.78e- 6 ****
## 10 na_sales Adventure Racing         583   385      9.97 1.33e-21 ****
## # ... with 32 more rows
```

Interpretation

As the p-value is less than the significance level 0.05, we can conclude that there are significant differences between the NA_Sales means of our genre's

Therefore, there is a statistically significant difference between genre groups as assessed using the Kruskal-Wallis test (p = <2.2e-16). Dunn's post hoc test showed that only 42 out of the possible 66 permutations were significant (Table Above). Notably, we do have an imbalanced design within our genre groups which may affect our test statistics.

```r
genre.pivot <- Vidgames.df %>%
  dplyr::select(genre,eu_sales:na_sales)%>%
  group_by(genre)%>%
  dplyr::summarise(EU = mean(eu_sales),
                   JP = mean(jp_sales),
                   N.America = mean(na_sales),
                   Other = mean(other_sales))

cn <-c(genre.pivot$genre)

heat <- genre.pivot%>%
  dplyr::select(EU:Other)

pheatmap(heat,display_numbers = T,
         labels_row = cn,
         main = "Heatmap for Genre Mean Sales by Region")
```



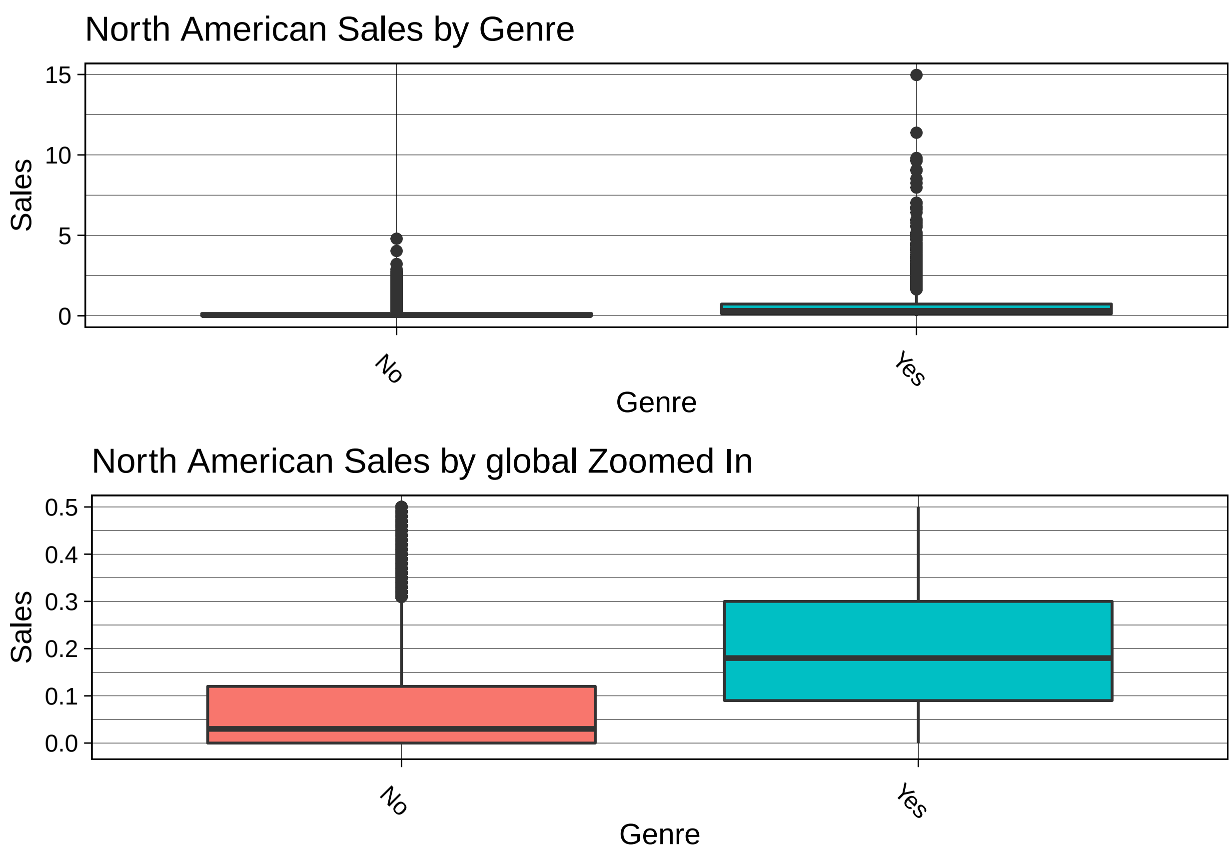
The table above shows the performance of each genre in different regions based upon mean sales. We can see that shooter is the top platform in N.America, EU, and other. Additionally, role-playing performs best in Japan. Notably, role-playing and platform genres perform relatively well across all markets. However, in terms of our outcome variable (N.America) all genres perform better compared to other markets.

Global

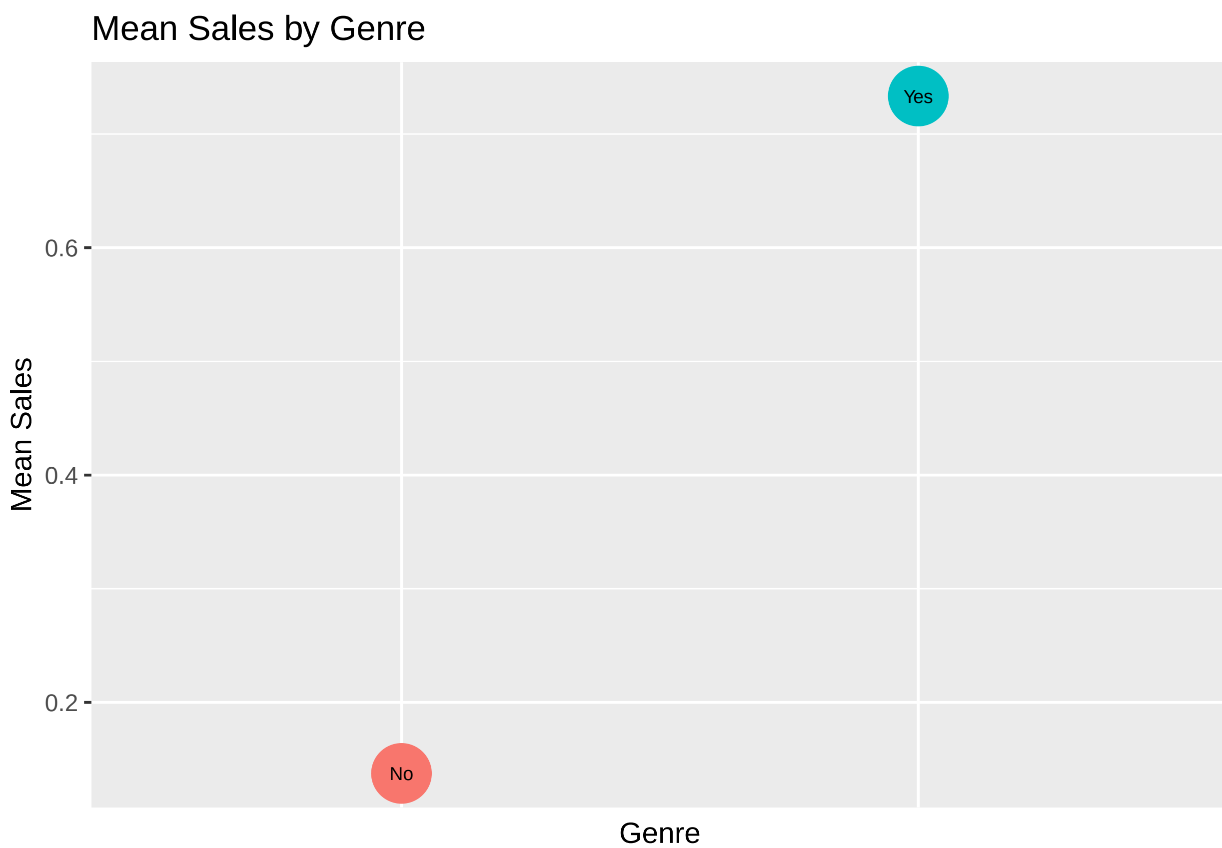
```
genre_log <- Vidgames.df%>%
  ggplot(aes(x = reorder(global,na_sales), y = na_sales))+
  geom_boxplot(aes(fill = global), show.legend = FALSE)+
  labs(x = "Genre", y = "Sales",
       title = "North American Sales by global Zoomed In")+
  theme_linedraw()+
  theme(axis.text.x = element_text(angle = -45))+
  ylim(-0.01,0.5)

genre_unlog <- Vidgames.df%>%
  ggplot(aes(x = reorder(global,na_sales), y = na_sales))+
  geom_boxplot(aes(fill = global), show.legend = FALSE)+
  labs(x = "Genre", y = "Sales",
       title = "North American Sales by Genre")+
  theme_linedraw()+
  theme(axis.text.x = element_text(angle = -45))

ggarrange(genre_unlog,genre_log,
```

```
        ncol = 1,nrow = 2,
        heights = c(3,3))
```

```
## Warning: Removed 831 rows containing non-finite values (stat_boxplot).
```




```
Vidgames.df %>%
  group_by( global ) %>%
  dplyr::summarise( mean_sales = mean( na_sales ) ) %>%
  ggplot( aes( x = global, y = mean_sales, colour = as.factor( global ) ) ) +
  geom_point( size = 10, show.legend = FALSE ) +
  geom_text( aes( label = global ), colour = "black", size = 2.5)+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())+
  labs(x =  "Genre",y = "Mean Sales", title = "Mean Sales by Genre")
```



We can see a clear difference between the means and distributions of our global variable. There appears to be significant differences among the means of our levels in terms of NA_Sales which we will investigate using the non-parametric Kruskal Wallis Test.

```
#Kruskal Wallis Test
res.kruskal <- kruskal.test(na_sales ~ global, data = Vidgames.df)
res.kruskal
```

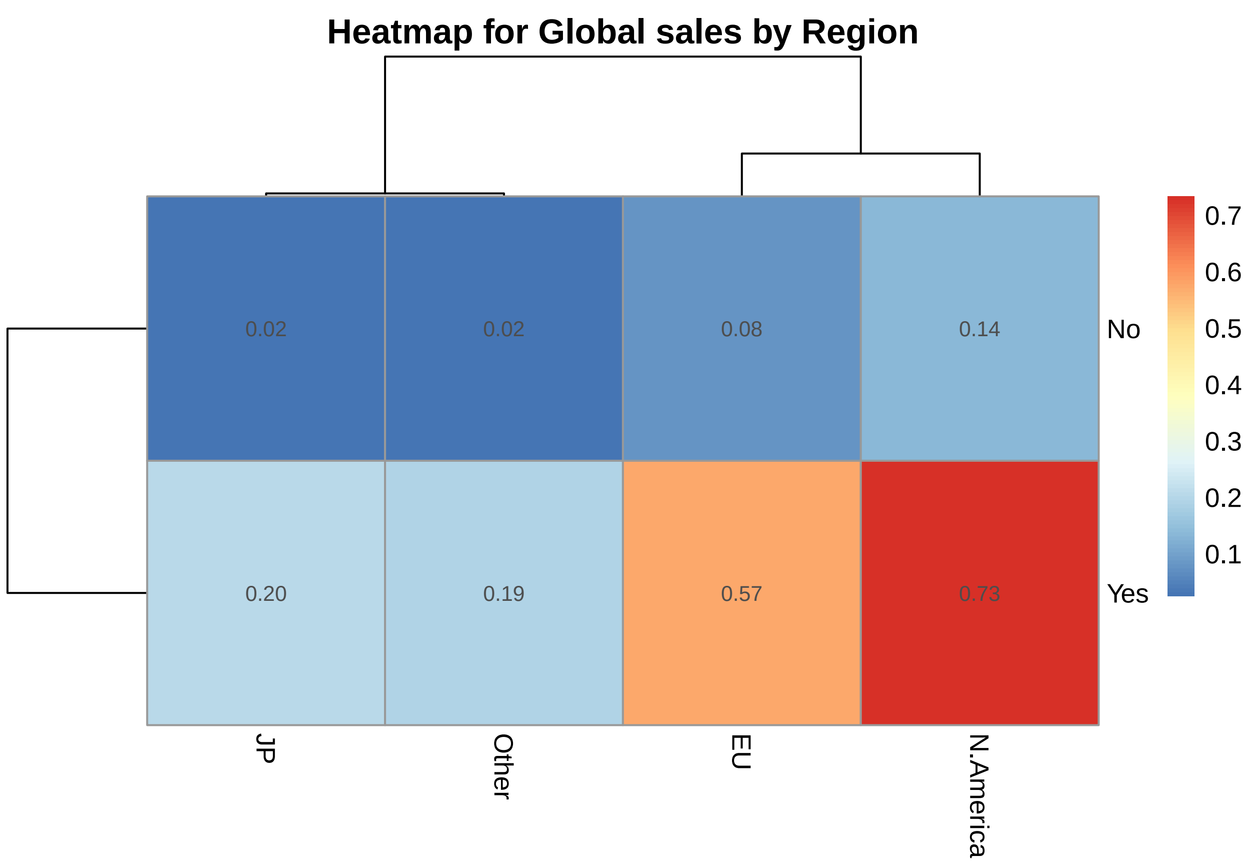
```
##
##  Kruskal-Wallis rank sum test
##
## data:  na_sales by global
## Kruskal-Wallis chi-squared = 1367.8, df = 1, p-value < 2.2e-16
```

Interpretation: As the p-value is less than the significance level 0.05, we can conclude that there are significant differences between the NA_Sales means of our global variable. However,we should be aware of the imbalanced design of our global variable.

Therefore, there is a statistically significant difference between global groups as assessed using the Kruskal-Wallis test ($p = <2.2e\text{-}16$).

```
global.pivot <- Vidgames.df %>%
  dplyr::select(global,eu_sales:na_sales)%>%
  group_by(global)%>%
  dplyr::summarise(EU = mean(eu_sales),
                   JP = mean(jp_sales),
```

```
                   N.America = mean(na_sales),
                   Other = mean(other_sales))

cn <-c(global.pivot$global)

heat <- global.pivot%>%
  dplyr::select(EU:Other)

pheatmap(heat,display_numbers = T,
         labels_row = cn,
         main = "Heatmap for Global sales by Region")
```



The table above shows performance in terms of mean sales in different regions based on whether games are sold across all markets. Accordingly, games that are sold in all markets perform better. For example we would expect to see games that are only sold in two regions to perform worse in all regions (including there own region) as compared to games that are sold globally.

Preprocessing

Train-Test Split

Splitting our dataset is essential for an unbiased evaluation of our models predictive performance and the validation of our model (ensuring we are not under/over-fitting). Therefore, we will initialize a train-test split of our data and then create training and testing sets based upon that random initialization. Notably, we have a 5,498 observations in our training split and 1,833 observations in our testing set which amounts to a 75%/25% train-test split.

```
set.seed(777)
games.split <- initial_split(Vidgames.df)
games.train <- training(games.split)
games.test <- testing(games.split)

#Number of observations in training and testing set
games.split
```

```
## <Analysis/Assess/Total>
## <5498/1833/7331>
```

Identify and remove any variables that could lead to overfitting.

We are fairly confident that our EDA has not exposed any variables that need to be removed. Notably, we have filtered our dataset down from 31 unique platforms to 9 and created a new "Global" variable.

Outline the three preprocessing steps that need to be performed on the data.

We have 23 predictors and one outcome. No variables had zero variance. We dummy encoded our three categorical variables (Platform, Genre, and Global), and all of our predictors were normalised.

Dummy Encoding: Through dummy encoding we expect our three categorical variables to be encoded into dichotomous variables while ensuring multicollinearity is avoided by dropping one of the resultant columns. Many machine learning algorithms cannot operate on categorical data directly. They require all input variables and output variables to be numeric. This means that our categorical variables (Platform, Genre, and Global) must be converted to a numerical form through dummy encoding.

Normalization: Normalization requires the centering of our variables (mean subtracted from the data) then the scaling of our variables (standard deviation of a variable is divided out of the data) whereby ensuring a standard deviation of one and a mean of zero. Notably, if our predictor variables are on different scales we could encounter a situation where our model prescribes greater importance to those variables with higher values. Our predictor variables are all on different scales (they exhibit different means and different standard deviations) and will need to be normalized. This will improve the performance of our model.

Zero Variance: This will remove any predictors that have zero variance.

Step_corr: This will remove highly correlated predictor variables. Highly correlated variables can cause some issues with fitting the model. I have forgone using the step_corr due to the fact that Multicollinearity does not affect the accuracy of predictive models (feature importance does become trickier though) as we are primarily interested in obtaining highly accurate models.

```
gamesVid.recipe <- recipes::recipe(na_sales ~ ., data = games.train)%>%
  step_zv( all_predictors() ) %>%
  step_dummy(platform,genre,global)%>%
  step_normalize(all_predictors())%>%
  prep()

gamesVid.recipe
```

```
## Data Recipe
```

```
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Training data contained 5498 data points and no missing data.
##
## Operations:
##
## Zero variance filter removed no terms [trained]
## Dummy variables from platform, genre, global [trained]
## Centering and scaling for eu_sales, jp_sales, other_sales, ... [trained]
```

Model fitting

As a supervised learning problem we have labelled outcome data. Therefore, our model fitting process requires the selection of two models upon which we will fit our prepossessed training data. As this is labelled data we are seeking to minimize the cost functions of our Lasso regression algorithm and our Random Forest Algorithm (outcome - predicted). Using our fitted model we will make model predictions on our training data, compare and access our model predictions to the training data's labelled outcomes (outcome - predicted), and finally use hyper-parameter tuning to find the best fitting parameter for each model (reduces our cost function the most) and fit that initalized model accordingly.

Lasso Model Initialization

We have initialized a lasso regression model with our hyper-parameter tuning parameter set only on our penalty parameter. As the mixture parameter controls the mixture between lasso and ridge (avoiding the use of elastic net) and we are only interested in a lasso regression model this parameter will remain at 1 (corresponds to lasso regression).

We have used bootstrapped data (resampling with replacement) as it allows us to estimate the predictive performance on our training data while applying our model to unobserved held out data (out of bag data from our bootstrapping). Since this is resampling with replacement our OOB data will obviously contain a larger quantity of unseen data when compared to cross validation which uses resampling without replacement. Therefore, there might be instances of more bias and less variance which is exactly what we are looking for. Additionally, we have bootstrapped our preprocessed training data 5 times due to 5-fold or 10-fold cross validation being common practice and we would like our model to train a bit faster as compared to 10 bootstrapped samples.

For our penalty parameter tuning we will try out 100 different penalty parameters due to the relatively small size of our training data, the fact we only used 5 bootstrapped samples, and we are searching for a highly accurate model. This penalty parameter controls the amount of regularization we want in our model i.e how much do we want to penalize large coefficients.

```r
vidgames.preproc <- gamesVid.recipe%>%
  juice()

lasso_spec <- linear_reg( mode = "regression", mixture = 1, penalty = tune())%>%
  set_engine( "glmnet" )
```

Boot strapping our Preproccessed training data

```
set.seed( 107 )

games_boots <- bootstraps( vidgames.preproc, times = 5)
```

Creating a penalty grid

```
penalty_grid <- grid_regular( penalty(),
                              levels = 100 )
penalty_grid
```

```
## # A tibble: 100 x 1
##     penalty
##       <dbl>
##  1 1   e-10
##  2 1.26e-10
##  3 1.59e-10
##  4 2.01e-10
##  5 2.54e-10
##  6 3.20e-10
##  7 4.04e-10
##  8 5.09e-10
##  9 6.43e-10
## 10 8.11e-10
## # ... with 90 more rows
```

```
set.seed(1007)

grid_lasso <- tune_grid(lasso_spec, na_sales ~ . , games_boots,
                        grid = penalty_grid )
```

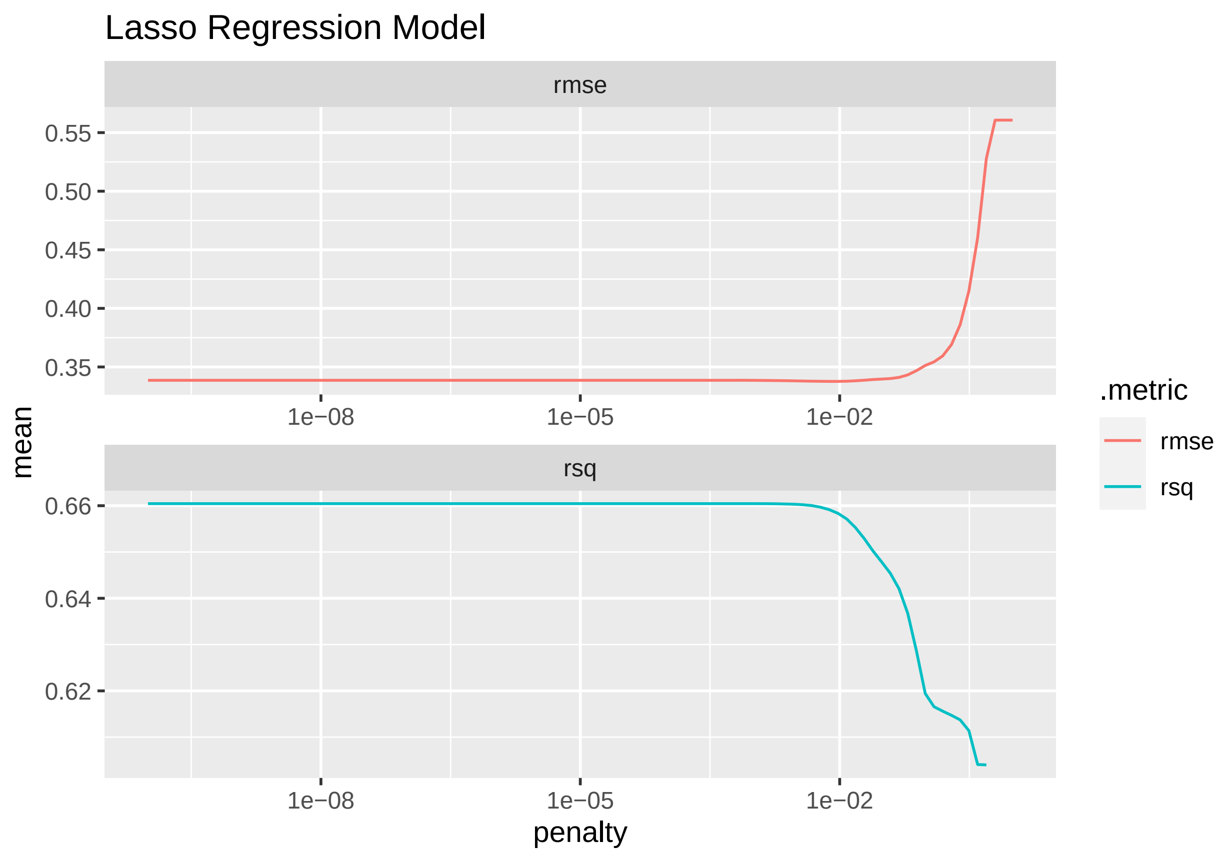
```
## ! Bootstrap1: internal: A correlation computation is required, but `estimate` is const...

## ! Bootstrap2: internal: A correlation computation is required, but `estimate` is const...

## ! Bootstrap3: internal: A correlation computation is required, but `estimate` is const...

## ! Bootstrap4: internal: A correlation computation is required, but `estimate` is const...

## ! Bootstrap5: internal: A correlation computation is required, but `estimate` is const...
```

RMSE and R squared plots

The plots below depict the RMSE and R-squared of our lasso regression model as the penalty term increases. We know a penalty of zero is simply linear regression and the higher the penalty the more we are penalizing large coefficients. Our plots below depict r-squared (variation of na_sales explained by our model) decreasing as our penalty value gets to 0.01. Notably, this seems to indicate that our model coefficients do not require significant shrinkage due to the consistent r-squared above 66% for the lower penalty values.

```
grid_lasso %>%
  collect_metrics() %>%
  ggplot( aes( penalty, mean, color = .metric ) ) +
  geom_line() +
  facet_wrap( ~.metric, scales = "free", nrow = 2) +
  scale_x_log10()+
  labs(title = "Lasso Regression Model")
```

```
## Warning: Removed 3 row(s) containing missing values (geom_path).
```



Best Penalty Value In terms of minimizing the rmse cost function, our hyper-parameter tuning found that a penalty of 0.009545485 produced the lowest rsme.

```
best_lasso_rmse <- select_best( grid_lasso, "rmse" )
best_lasso_rmse%>%
  gt()%>%
  tab_header(
    title = "Select Best Output",
    subtitle = "Best Tunned Penalty Value"
  )
```

Select Best Output

Best Tunned Penalty Value

penalty	.config

0.009545485	Preprocessor1_Model080

```r
games_lasso <- finalize_model( lasso_spec, best_lasso_rmse )
games_lasso
```

```
## Linear Regression Model Specification (regression)
##
## Main Arguments:
##   penalty = 0.00954548456661833
##   mixture = 1
##
## Computational engine: glmnet
```

Random forest Model

We have initialized a random forest regression model with our hyper-parameter tuning parameter set on our mtry and min_n parameters. While more "Trees" would have slight benefits in prediction performance, this benefit will be lower than the cost in computation time from learning significantly more trees. Additionally, 100 trees is common practice (especially for our sample size).

Since the random forest algorithm uses bootstrapped sampling with the number of variables considered at each split randomly selected, our mtry parameter controls the number of variables at each split to be considered. The default for regression is normally variables/3 (rounded down) which would leave us with 7. However, this needs to be tuned as it plays a vital role in model performance and at each new iteration after the model is tested on the OOB samples new trees are normally built using updated mtry in order to find the best model in terms of lowering its cost function.

Min_n controls the minimum number of observations in a node that are required for the node to be split further. Notably, tuning this parameter can assist us in avoiding overfitting and obtaining more accurate models.

Random Forest Fitting

```r
rf_spec <- rand_forest(
  mode = "regression",
  mtry = tune(),
  trees = 100,
  min_n = tune()
) %>%
  set_engine( "ranger", importance = "permutation"  )
```

```r
rand_spec_grid <- grid_regular(
  finalize( mtry(),
            vidgames.preproc %>%
              select( -na_sales ) ),
  min_n(),
  levels = c(mtry  = 11 ,min_n = 4))
rand_spec_grid
```

```
## # A tibble: 44 x 2
```

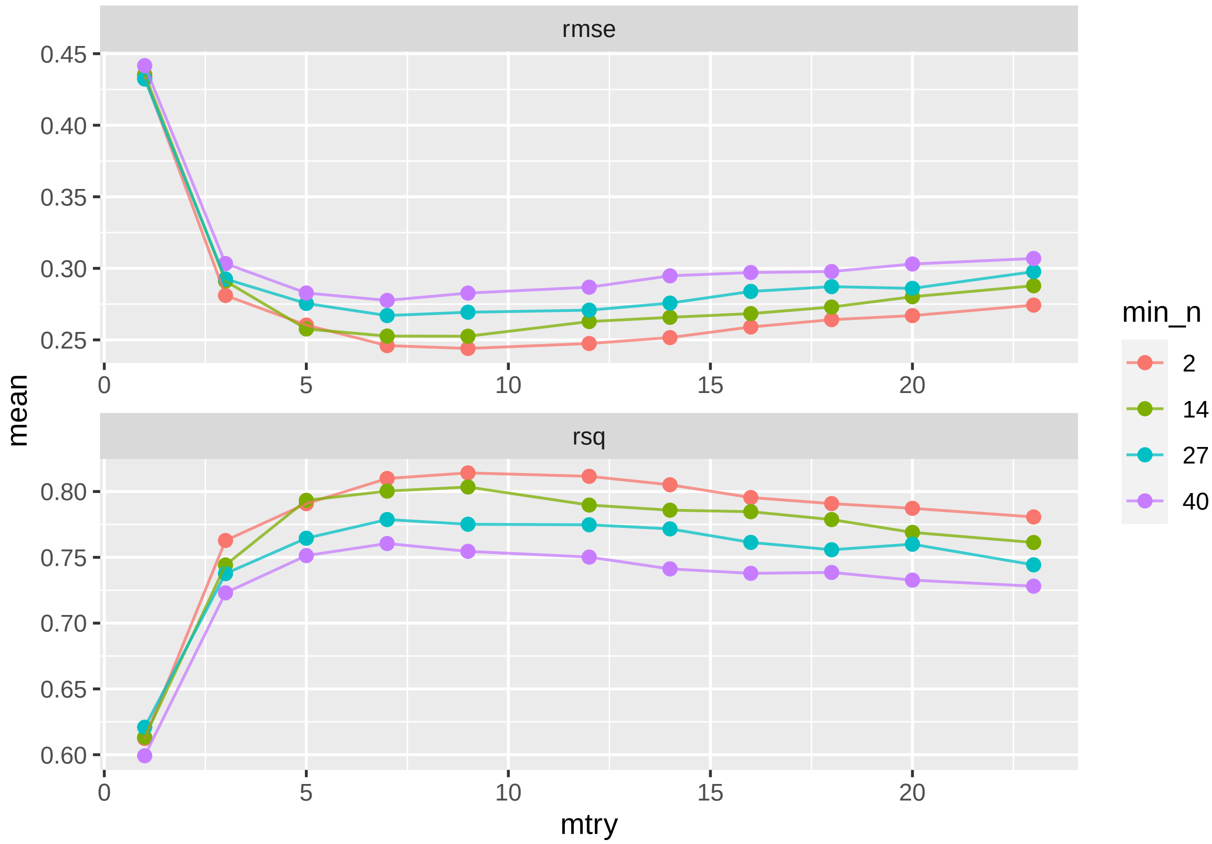
```
##     mtry min_n
##    <int> <int>
##  1     1     2
##  2     3     2
##  3     5     2
##  4     7     2
##  5     9     2
##  6    12     2
##  7    14     2
##  8    16     2
##  9    18     2
## 10    20     2
## # ... with 34 more rows
```

```
set.seed(1959)
doParallel::registerDoParallel()

rf_grid <- tune_grid( rf_spec, na_sales ~ .,
                      games_boots,
                      grid = rand_spec_grid )
```

RMSE and R squared plots

The plots below depict the RMSE and R-squared of our random forest regression model for the various interactions between mtry and min_n. Additionally, as the number of mtry increases we see a resultant increase in our models r-squared value. The min number of observations in each node (min_n) corresponding to 2 and 14 seem to constantly produce better metrics compared to the remaining 2. Notably, the value of 2 for min_n seems to constantly produce the best metrics and when combined with a mtry of 9 they produce the most accurate model.

```
rf_grid %>%
 collect_metrics() %>%
  mutate( min_n = as.factor( min_n ) ) %>%
  ggplot( aes( x = mtry, y = mean, colour = min_n ) ) +
  geom_point( size = 2 ) +
  geom_line( alpha = 0.75 ) +
  facet_wrap( ~ .metric, scales = "free", nrow = 3 )
```




```
best_rf_rmse <- select_best( rf_grid, "rmse" )
best_rf_rmse
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     9     2 Preprocessor1_Model05
```

```
final_rf <- finalize_model( rf_spec, best_rf_rmse )
final_rf
```

```
## Random Forest Model Specification (regression)
##
## Main Arguments:
##   mtry = 9
##   trees = 100
##   min_n = 2
##
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

Best Performing Model via Cross-Validation

Cross-validation uses resampling with out replacement and allows us to test the performance of our model on a hold out set (held out by CV). Therefore, the model with the best cross-validation results in terms of our rmse metric should perform best on our test set. We have used 10-fold cross-validation on our pre-processed training data due to the size of our data set and the fact that 5-fold or 10-fold cross-validation is common practice. Secondly, we have fit both models using this cross-validated training data. Finally, we have generated the appropriate accuracy metrics (rmse and rsq) for both models. Although we are primarily interested in that higher rsq value although high rsq values could still produce bad models.

```
set.seed(107)
vidgames.preproc_cv <- vfold_cv(vidgames.preproc,
                               v  = 10)
```

So our tunned lasso regression model has an rsq of 66% meaning that our models explains 66% of the variance in na_sales. A rmse of 0.367 means that on average our model has an error of 0.367 units in terms of predicting na_sales. This is relatively high when we consider the mean value of na_sales is relatively low. Overall this model is performing adequately although we can significantly improve our predictions with our random forest model.

```
games_lasso_rs <- fit_resamples( games_lasso,
                                  na_sales ~ .,
                                  vidgames.preproc_cv)
games_lasso_rs %>%
  collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.367    10  0.0240 Preprocessor1_Model1
## 2 rsq     standard   0.665    10  0.0244 Preprocessor1_Model1
```

So our tunned random forest regression model has an rsq of 85.3% meaning that our models explains 85.3% of the variance in na_sales. This is a significant improvement from our lasso regression model. A rmse of 0.25 means that on average our model has an error of 0.25 units in terms of predicting na_sales. Notably, our rmse has dropped 0.116 and our rsq has increased by 19% which is significant and conclusive. Therefore, based upon our cross-validation, the random forest model is our chosen model.

```
games_rf <- fit_resamples( final_rf,
                                  na_sales ~ .,
                                  vidgames.preproc_cv)
games_rf %>%
  collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.251    10  0.0285 Preprocessor1_Model1
## 2 rsq     standard   0.853    10  0.0161 Preprocessor1_Model1
```

Model evaluation

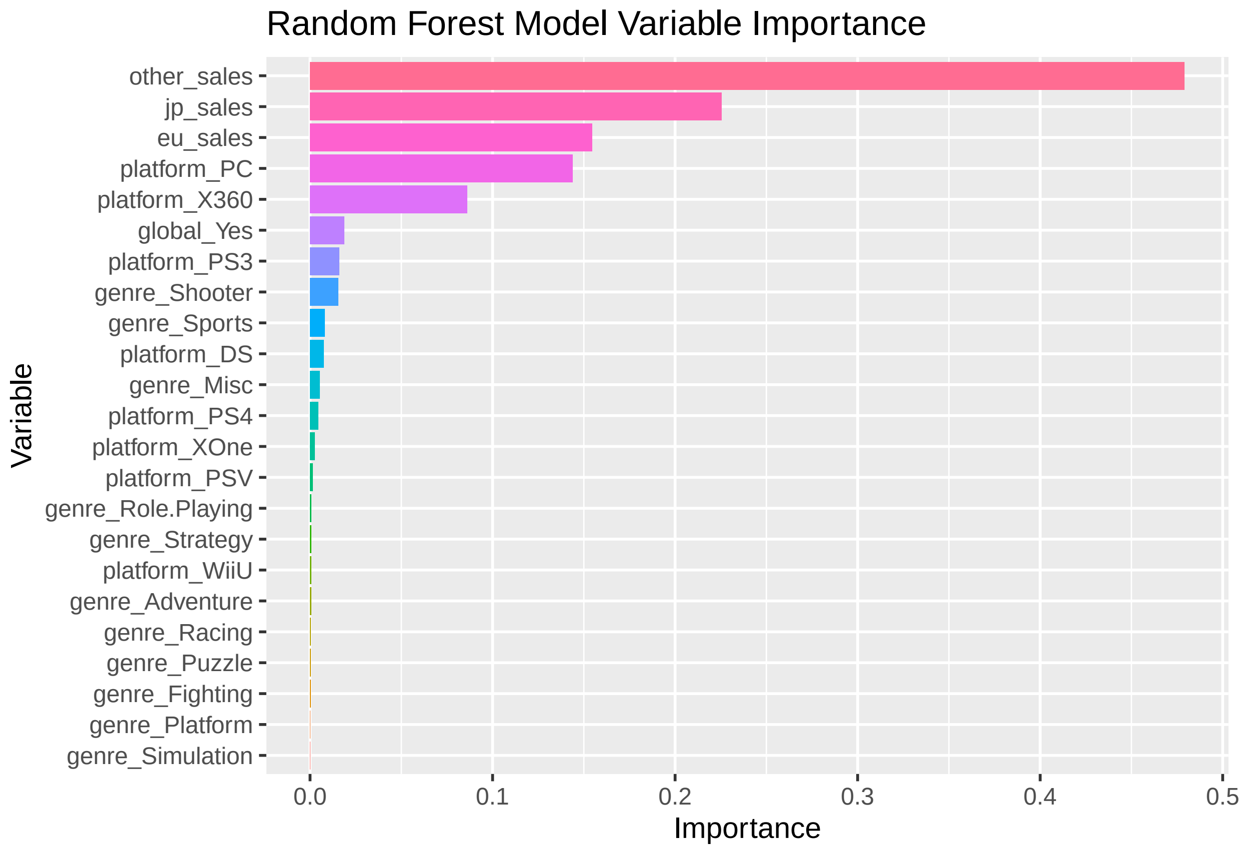
In order to validate our model for it to be deploy-able we must ensure this model generalizes well to unseen data (low variance). We will use r-squared and rmse to quantify the performance of our model on data it has not yet seen (our test set). We are aiming for a model that produces similar performance metrics as compared to our training set. What we do not want to see is a model that does not generalize well on our test set and produces an r-squared value significanlty below our 85% as seen above.

```
set.seed(777)
randomForest.games <- final_rf %>%
  fit( na_sales ~ . , data = vidgames.preproc )
```

Variable Importance Plot

Unsurprisingly, other_sales, eu_sales, and jp_sales are our top three most important variables. Notably, our EDA exposed the high monotonic correlation between other_sales and na_sales and this seem to be evident in our model. Additionally, we saw jp_sales had a weak monotonic relationship with sales and interestingly enough we now see it plays an important rolein terms of variable importance. Our decision to leave in jp_sales seems to be validated here. Additionally, we see X360 and PC as our top performing platforms in terms of variable importance. Interestingly, PC did not show up much in our EDA. Additionally, we see sports and shooter as our top performing genres in terms of variable importance although i am surprised with role-playing as i thought it would have been more influential based upon our EDA. However, from Xone onwards the variable importance of those remaining variables seem negligible. Notably, our global variable was our 6th most important variable which seems to provide justification for our inclusion of it. Overall other_sales is by far the most important predictor of the quantity of na_sales. However, please not that we did not perform step_corr to remove high correlation among our predictors (for the sake of high accuracy) as this impacts our confidence in our variable importance measures especially with our other_sales variable.

```
vi( randomForest.games ) %>%
  mutate( Importance = abs( Importance ),
          Variable = fct_reorder( Variable, Importance ) ) %>%
  ggplot( aes( x = Importance, y = Variable)) +
  geom_col(aes(fill = Variable))+
  theme(legend.position="none")+
  labs(title = 'Random Forest Model Variable Importance')
```



Preparing our test set

```
games_test_preproc <- bake( gamesVid.recipe, games.test )
games_test_preproc
```

```
## # A tibble: 1,833 x 24
##    eu_sales jp_sales other_sales na_sales platform_DS platform_PC platform_PS3
##       <dbl>    <dbl>       <dbl>    <dbl>       <dbl>       <dbl>        <dbl>
##  1    17.7   15.1          12.3      4.75       1.54       -0.398       -0.465
##  2    10.0    0.00913       8.14     9.63      -0.650      -0.398       -0.465
##  3     6.06 20.6            4.70     5.57       1.54       -0.398       -0.465
##  4     8.00  0.267          7.77     9.03      -0.650      -0.398       -0.465
##  5     6.93  0.193          6.60     9.67      -0.650      -0.398       -0.465
##  6     8.04  0.0460         6.54     8.25      -0.650      -0.398       -0.465
##  7     5.06 14.4            4.40     4.4        1.54       -0.398       -0.465
##  8     6.86  1.19           9.67     4.99      -0.650      -0.398        2.15
##  9     6.99  1.41           9.61     4.76      -0.650      -0.398        2.15
## 10     4.27 -0.0646         4.15     6.63      -0.650      -0.398       -0.465
## # ... with 1,823 more rows, and 17 more variables: platform_PS4 <dbl>,
## #   platform_PSV <dbl>, platform_WiiU <dbl>, platform_X360 <dbl>,
## #   platform_XOne <dbl>, genre_Adventure <dbl>, genre_Fighting <dbl>,
## #   genre_Misc <dbl>, genre_Platform <dbl>, genre_Puzzle <dbl>,
## #   genre_Racing <dbl>, genre_Role.Playing <dbl>, genre_Shooter <dbl>,
## #   genre_Simulation <dbl>, genre_Sports <dbl>, genre_Strategy <dbl>,
```

```
## #   global_Yes <dbl>
```

Creating our predictions

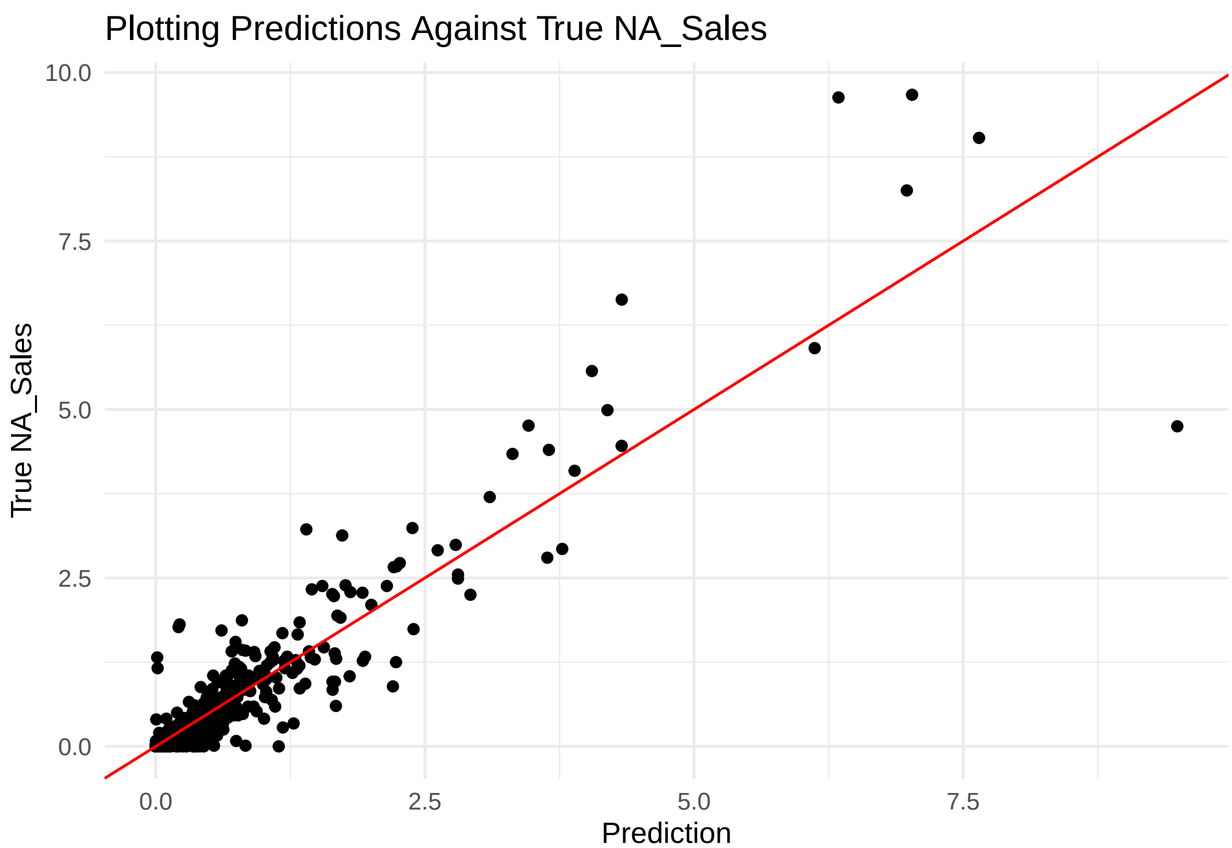
```
games_preds <- predict( randomForest.games,
                        new_data = games_test_preproc ) %>%
  bind_cols( games_test_preproc %>%
               select( na_sales ) )
games_preds
```

```
## # A tibble: 1,833 x 2
##    .pred na_sales
##    <dbl>    <dbl>
##  1  9.49     4.75
##  2  6.34     9.63
##  3  4.05     5.57
##  4  7.65     9.03
##  5  7.02     9.67
##  6  6.98     8.25
##  7  3.65     4.4
##  8  4.20     4.99
##  9  3.46     4.76
## 10  4.33     6.63
## # ... with 1,823 more rows
```

Plotting model predictions against true na_sales values

If our model is predicting accurately we would expect our scattered data points to lie along our red regression line. It is evident that most points are not lying on that line. We can almost see a heteroscedastic display of our points. This means we are performing relatively well in predicting our lower value na_sales (large cluster on the lower end) but progressively worse on average at predicting our higher valued na_sales. Our training set had a few outliers and it is clear the same is true in our testing set. Majority of those outliers values are lying above our line meaning that we are under-predicting our na_sales value more often than not which is what we would expect. Obviously we cannot expect our model to catch/predict all these outliers (thats why they are outliers) but we are relatively happy about our models ability to seemingly predict accurately our lower values which is the bulk of our data.

```
games_preds %>%
  ggplot( aes( x = .pred, y = na_sales ) ) +
  geom_point() +
  geom_abline( intercept = 0, slope = 1, colour = "red" ) +
  theme_minimal()+
  labs(x = "Prediction", y = "True NA_Sales",
       title = "Plotting Predictions Against True NA_Sales")
```



Prediction Evaluation

In terms of evaluating our random forest regression model on unseen test data, our model produced a rsq of 87.6% meaning that our models explains 87.6% of the variance in na_sales. A rmse of 0.24 means that on average our model has an error of 0.24 units in terms of predicting na_sales. Notably, our RMSE is smaller than our cross-validated RMSE, and our r-squared is higher than our cross-validated. This is exactly what we wanted to see. A higher RSQ means we have not overfit our training data and our model generalizes well on unseen data.

```
games_preds %>%
  metrics( truth = na_sales, estimate = .pred )%>%
  gt()%>%
  tab_header(
    title = "Random Froest Model Evaluation (Test)",
    subtitle = "R-Squared and RMSE metrics"
  )
```

Random Froest Model Evaluation (Test)

R-Squared and RMSE metrics

.metric	.estimator	.estimate
rmse	standard	0.24066249
rsq	standard	0.87625581
mae	standard	0.07829924

Prediction on Fatal Empire Data

```
newdata <- tibble(
  platform = "PS4",
  genre = "Role-Playing",
  global = "Yes",
  jp_sales = 2.58,
  eu_sales = 0.53,
  other_sales = 0.1
)

fatal.empire <- bake( gamesVid.recipe, newdata )
```

```
## Warning:  There were 3 columns that were factors when the recipe was prepped:
##  'platform', 'genre', 'global'.
##  This may cause errors when processing new data.
```

```
randomForest.games%>%
  predict(new_data = fatal.empire)%>%
  gt()%>%
  tab_header(
    title = "The Fatal Empire Predicted Sales",
    subtitle = "North American Sales"
  )
```

The Fatal Empire Predicted Sales

North American Sales

.pred
0.5268382

Model Summary

The top 6 most important predictors within our data set is other_sales, jp_sales, eu_sales, platform_PC, platform_X360, and global_yes. Our three sales variables are clearly the most influential predictors but overall other_sales is by far the most important predictor of the quantity of na_sales. In terms of model performance, our models explains 87.6% of the variance in na_sales which is significant. However, we are still receiving a RMSE of 0.24 meaning that on average we have a mean error in our predictions of 240,000 copies of sales. This is quite significant when you consider mean NA_Sales in our training set was 0.22. Notably, it is evident that our NA_Sales contained numerous outliers and a relatively high standard deviation. Therefore, we are not surprised that our model performed relatively well in predicting our lower value na_sales but progressively worse on average at predicting our higher valued na_sales.

According to our model we forecast North American sales for "The Fatal Empire" to be 0.526838 million copies (526,838) with an mean error of 240,000.