Dylan Rohan - a1844790

ASSESSMENT 3 – CASE STUDY AND DATA ANALYSIS

Summary to CEO

Hello Mr Russel,

Regarding the predicted sales of 'The Fatal Empire'. With the information we have about its title length, number of platforms released on, specific platform of release, game genre, year of release, and the sales figures of Japan, Europe, and other countries, our random forest model predicts sales of 1.92 million copies.

Taking the data as it is, and assuming the historical data is still representative, we can be around 90% sure in the model's accuracy. That being said, there are a few areas for improvement, such as a highly intensive clean of the data set and perhaps more data acquisition. I will cc you in on the manager's report were that will be discussed further.

Regards,

Dylan

Dylan Rohan - a1844790

Manager's report

1. Aim and Hypothesis:

The purpose of this report is to outline the process undertaken to predict the North American sales figure of the PS4 release of 'The Fatal Empire' using data available online via www.kaggle.com that was originally sourced from www.vgchatz.com. The data provided included information regarding a games Name, Year of release, Platform of release, Game genre, Sales figures of North America, Sales figures of Japan, Sales figures of Europe, Sales figures for the total of all other countries.

From this data, two additional pieces of information were acquired, the title length, as well as the 'number of platforms a game had been released on' as it was hypothesised that human behaviour relating to these two factors would be informative (Title length may influence a potential customers likelihood of picking up a game in-store, and more platforms influences 'word-of-mouth advertising' as well as the community/support available to a game). Two types of predictive models were trialled, a LASSO regression and a random forest. Their function and accuracy were then examined.

2. Data Cleaning

In this phase an approach to verify the Name and Publisher values was provided. The Year variable was properly formatted and NA values addressed. 270 NA values were reduced to 133 by setting them equal to the average year for rows with the same game Name. Five of the remaining titles had a year indicated in their title which made for a good approximation of its release year. The remaining values were given the mean year for rows with the same Platform. This was considered a better approximation on a case-by-case basis than simply setting them all equal to the global mean year. Some of the sales figures were representing weekly sales, these have been converted to annual figures where it was obvious. Of those, six were for US/American sales which would relate to the Other_Sales value. As there is no way to easily multiply this column (because it contains multiple countries and not just America) by 52.14 to find annual sales, they have been left as is with the knowledge that this will influence the accuracy of the model. Two incorrect Platform values (as indicated in the Name column) were corrected.

The issues that persist include, as mentioned above, the US weekly sales. The game names have not been changed to reflect the corrections made (this remains a source of error for the title length variable). There is no simple way to merge data pertaining to the same game because there is no linking variable/ID. And even if there were, it would need to be done individually and manually as some values would be written over in the process.

3. Exploratory Data Analysis (EDA)

The information most relevant to predicting North American sales was identified after careful examination. It was found that the title length, Platform, Year, Genre, all Sales figures, and the number of platforms a game had been released on were informative for the prediction of North American sales.

These assertions were identified through simple bivariate analysis, principal component analysis, and parallel coordinate plots. The parallel coordinate plot found that there was a relationship between platform and region; 'Nintendo' platforms generally selling better in Japan and 'Sony' platforms generally selling better outside of Japan. A similar trend could be found for some publishers; likely because some publishers produce games for Nintendo platforms/ Japanese audiences. The principal component analysis found that the numerical variables contained the most amount of variance in the data, but that is only saying that no specific category contained much variation on its own. The principal component analysis offered the best evidence at this stage that the additional variables would make good predictors. Their importance was also confirmed after the fact through variable importance analyses.

4. Data Processing

The data was modified as required. This involved a transformation of the sales figures (by log(x+1)) and normalizing all countable variables. These transformations improve the distribution of data (as much of it was pooled together) and values could only be reasonably compared in terms of proximity to the mean of their populations. Platform and genre were also treated in a manner that enabled a computer to interpret each category.

The data was then split into training and testing sets. This is done so that the model can be tested on data that was not intrinsically part of its production.

5. Model Fitting and Model Evaluation

The model fitting stage involved building four models and measuring their accuracy (this involves comparing the selected sample sets prediction to the true value of the unselected sets). The models were optimized by penalizing some of the information based on importance (in the case of LASSO regression) or by tree characteristics (in the case of random forest). A sufficient range of penalty values was decided upon, and the values that had the best improvement in accuracy were selected. A fifth model was created in an

attempt to improve/ verify the variable importance measures obtained by the best model, but it provided no benefit in terms of predictive accuracy.

The models created were:

1. LASSO regression model - Initial variables
2. LASSO regression model - initial variables + (Title length, platforms available to game)
3. Random forest model - Initial variables
4. Random forest model - initial variables + (Title length, platforms available to game)
5. Random forest model - initial variables + (Title length, platforms available to game) + preprocessing steps to reduce categories and correlation

There are a few assumptions involved in the LASSO regression model that were verified in this section. However, the relevance of the LASSO models was lacking. Of the above list, it was the fourth model that had the best accuracy as it was found to have the highest r-squared value and the lowest route mean squared. Here are the results calculated by that model:

Figure 26.1: The true values of the test set compared to the predicted values from the random forest model generated with the additional predictors

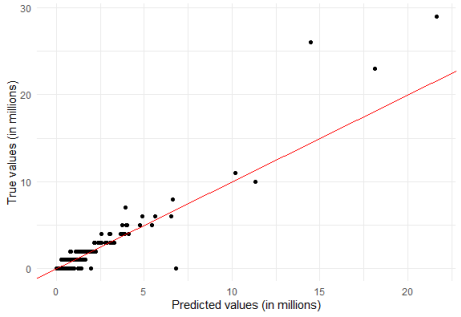


Figure 26.2:The errors associated with the predicted values of the model created with the additional predictors had a very near zero mean

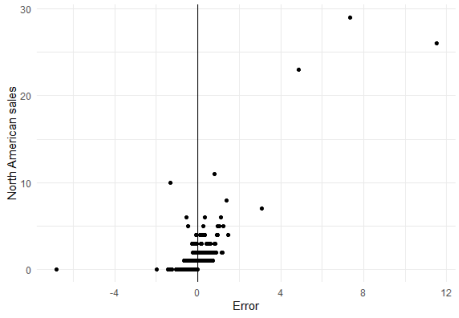
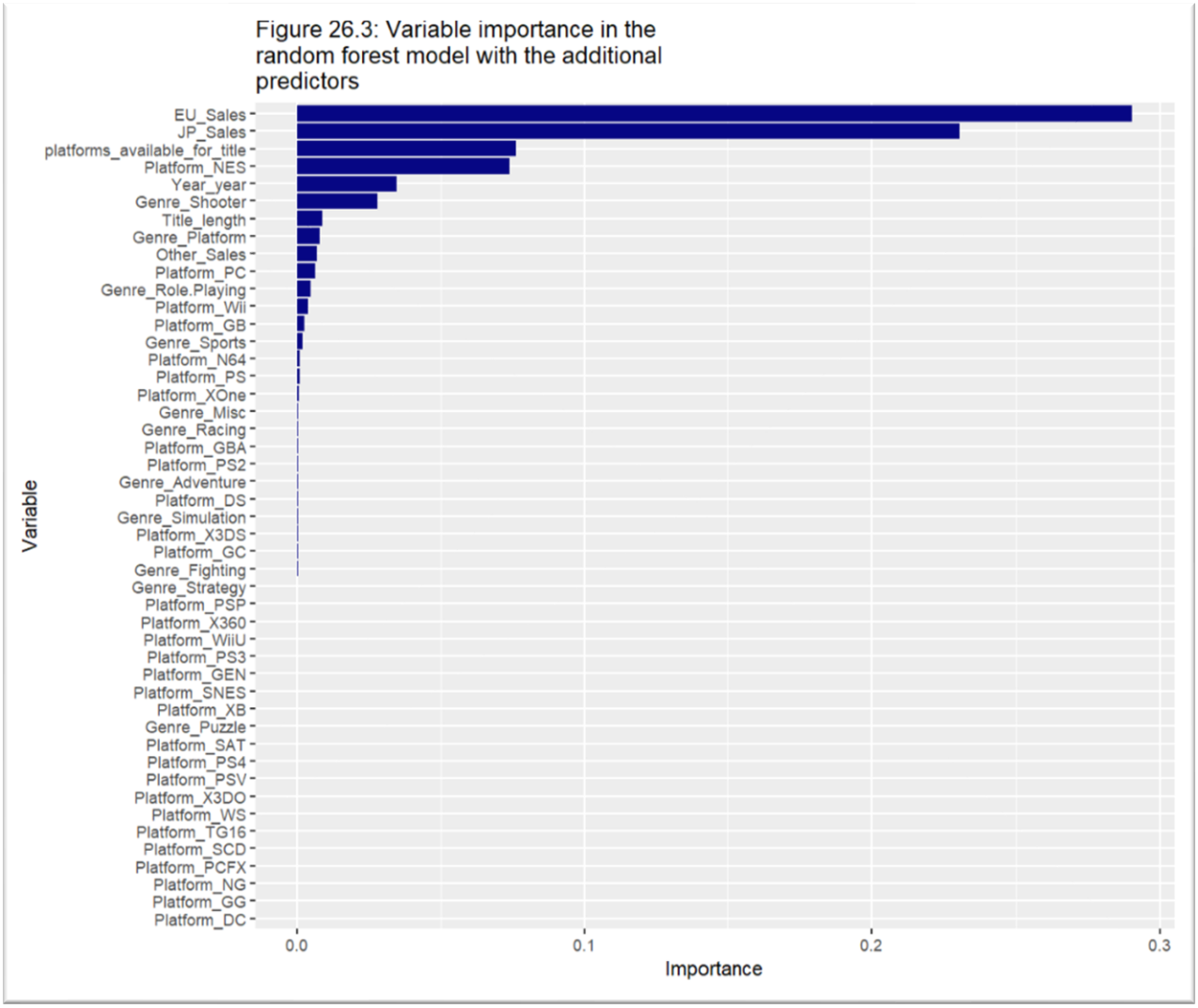


Table 14: The measures of model accuracy for the random forest model with the additional predictors

.metric	.estimator	.estimate
rmse	standard	0.2985364
rsq	standard	0.9195488
mae	standard	0.0542567



This model was then used to obtain the following prediction:

North American sales prediction (in millions) for PS4 role-playing game titled 'The Fatal Empire' with 2.58 million copies sold in Japan, 0.53 million copies in Europe, and 0.1 million copies in other parts of the world.
1.92

Dylan Rohan - a1844790

Statisticians report

Libraries Used

- library("tidyverse")
- library("dataMeta")
- library("caret")
- library("skimr")
- library("ggcorrplot")
- library("tidymodels")
- library("vip")
- library("ggpubr")
- library("usemodels")

(Dania M Rodriguez et al. 2017; Greenwell, Boehmke & McCarthy 2018; Kassambara 2019; Kassambara & Kassambara 2020; Kuhn 2015, 2022; Kuhn & Wickham 2020; McNamara et al. 2018; Wickham et al. 2019)

Data Clean

Part 1 - Import and skim

```
# Importing data
vgsales <- read.csv("vgsales.csv")
head(vgsales,6)
```

```
##                       Name Platform Year        Genre Publisher NA_Sales
## 1               Wii Sports      Wii 2006       Sports  Nintendo    41.49
## 2        Super Mario Bros.      NES 1985     Platform  Nintendo    29.08
## 3           Mario Kart Wii      Wii 2008       Racing  Nintendo    15.85
## 4        Wii Sports Resort      Wii 2009       Sports  Nintendo    15.75
## 5 Pokemon Red/Pokemon Blue       GB 1996 Role-Playing  Nintendo    11.27
## 6                   Tetris       GB 1989       Puzzle  Nintendo    23.20
##   EU_Sales JP_Sales Other_Sales
## 1    29.02     3.77        8.46
## 2     3.58     6.81        0.77
## 3    12.88     3.79        3.31
## 4    11.01     3.28        2.96
## 5     8.89    10.22        1.00
## 6     2.26     4.22        0.58
```

```
# Initial skim of Data
skim(vgsales)
```

Dylan Rohan - a1844790

Data summary

Name	vgsales
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	▇▁▁▁▁
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	▇▁▁▁▁

```
# No missing values identified here
```

The data has nine columns, with 16,598 rows. There are currently 5 character variables and 4 numeric variables. The variables include the name of the game (Name), the platform the game can be played on (Platform), the year the game was released (Year), the genre of the game (Genre), the publisher of the game (Publisher), as well as the number of copies sold in North America, Europe, Japan, and the total for all remaining countries (NA_Sales, EU_Sales, JP_Sales, Other_Sales, respectively). No missing values were made apparent in this step.

Dylan Rohan - a1844790

Data Clean

Part 2 - A Good Clean and Tidy

Summary of what has been accomplished below:

- A duplicate was removed
- An approach to verify the Name and Publisher values was provided
- The Year variable was properly formatted and NA values addressed. 270 NA values were reduced to 133 by setting them equal to the average year for rows with the same title. Five of the remaining titles had a year in their title which made for a good approximation of its release year. The remaining values were given the mean year for rows with the same Platform. This was considered a better approximation on a case-by-case basis than simply setting them all equal to the global mean year.
- A curiosity regarding Wii Sport was examined and validated
- Some of the sales figures were representing weekly sales, these have been converted to annual figures where obvious. Of those, six were for US/American sales which would relate to the Other_Sales value. As we cannot simply multiply this column (because it contains multiple countries and not just America) by 52, they have been left as is with the knowledge that this will influence the accuracy of the model
- Two incorrect Platform values were corrected

Issues Identified but not fixed:

- As mentioned above, the US weekly sales rows remain an issue
- I've not changed the names to reflect the corrections made (because Name is not a predictor)
- I could not find a simple way to merge data pertaining to the same game because there is no linking variable/ID. And even if there were, I would still need to go through each manually as some values would be written over in the process

```
# Check for Duplicates
filter(vgsales, duplicated(vgsales) == TRUE)

filter(vgsales, Name == "Wii de Asobu: Metroid Prime")
```

```
##                         Name Platform Year   Genre Publisher NA_Sales EU_Sales
## 1 Wii de Asobu: Metroid Prime      Wii  N/A Shooter  Nintendo        0        0
## 2 Wii de Asobu: Metroid Prime      Wii  N/A Shooter  Nintendo        0        0
…
```

```
clean <- distinct(vgsales)
# One duplicate removed

# Check for other mistakes/inconsistencies:
```

```
# Name check:
name_vgsales <- clean %>%
  group_by(Name) %>%
  tally(sort=TRUE)        # currently 11,492 Game titles
# Need to compare values against a Master list

# Platform check:
platform_vgsales <- clean %>%
  group_by(Platform) %>%
  tally(sort=TRUE)        # 31 Platforms
head(platform_vgsales,5)
```

```
## # A tibble: 5 x 2
##   Platform     n
##   <chr>    <int>
## 1 DS        2163
## 2 PS2       2161
## 3 PS3       1329
## 4 Wii       1324
## 5 X360      1265
```

```
# Clear

# Publisher check
publishers_vgsales <- clean %>%
  group_by(Publisher) %>%
  tally(sort=TRUE)        # 579 Publishers
head(publishers_vgsales,5)
```

```
## # A tibble: 5 x 2
##   Publisher                       n
##   <chr>                       <int>
## 1 Electronic Arts              1351
## 2 Activision                    975
## 3 Namco Bandai Games            932
## 4 Ubisoft                       921
## 5 Konami Digital Entertainment  832
```

```
# Need to compare values against Master list

      # Check 579 publishers against list found on wiki and at:           ht
tps://www.kaggle.com/datasets/andreshg/videogamescompaniesregions?resource=do
wnload) for quick spell-check I create a Master list to compare Publisher val
ues against:
dev1 <- read.csv("dev1.csv")
dev2 <- read.csv("dev2.csv")
wiki1 <- read_csv("table-2.csv")

dev1_publishers <- dev1$Developer
dev2_publishers <- dev2$Developer
wiki_publishers <- wiki1$Publisher
```

```
publisher_check <-  c(dev1_publishers, dev2_publishers, wiki_publishers)
publisher_check <- unique(publisher_check)
length(publisher_check) # 1535 Publishers to check against

## [1] 1535

publisher_check_df <- data.frame(matrix(unlist(publisher_check),
                                        nrow=length(publisher_check),
                                        byrow=TRUE))
      # Comparing lists
publishers_vgsales$Publisher %in% publisher_check_df$matrix.unlist.publisher_
check...nrow...length.publisher_check...

##   [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  T
RUE... etc.

# With more resources you could verify from a 'master list', mine is incomple
te and I have no way of knowing if it has errors itself. But it's a proof of
concept as to how you could go about checking the Qualitative variables.

# Year Check:
unique(clean$Year)
clean <- clean %>%
  mutate(Year = na_if(Year, "N/A"), Year = na_if(Year, ""), Year = na_if(Year
, " ") )
clean$Year[is.nan(clean$Year)]<-NA
sum(is.na(clean$Year))
clean$Year <- as.numeric(clean$Year)
# Has 270 NA values and is now numeric

# Could just give the average if a game is released  on multiple platforms
na_rows <- filter(clean, is.na(Year))
na_titles <- c(na_rows$Name)
# If a game is released on multiple platforms, let NA equal the average of
the year
for(i in na_titles){
  if(nrow(filter(clean, grepl(i, Name))) > 1){
    clean[clean$Name == i, "Year"] <- round(mean(filter(clean, grepl(i, Name)
)[,3], na.rm = TRUE),0)
  }
}
filter(clean, is.na(Year))
# Now only 133 NA values

# If the year is in the title, that's a good indication's likely to be within
a year of the true value. These 5 titles below only appear in the data once,
so we can alter by Name:
name_indicates_year <- filter(clean, grepl(" 20", Name))
Name_to_Year <- filter(name_indicates_year, !grepl("", Year))
clean[clean$Name == "wwe Smackdown vs. Raw 2006", "Year"] <- 2006
```

```
clean[clean$Name == "NFL GameDay 2003", "Year"] <- 2003
clean[clean$Name == "Tour de France 2011", "Year"] <- 2011
clean[clean$Name == "Sega Rally 2006", "Year"] <- 2006
clean[clean$Name == "Football Manager 2007", "Year"] <- 2006
filter(clean, is.na(Year))

# Now only 128 Na values

# The most accurate substitution for the remaining values would be the mean v
alue for the year for games on that platform type... this may take a second
na_key <- group_by(clean, Platform) %>%
    summarize(m = round(mean(Year, na.rm=TRUE))) # Average year for each plat
form
# Make a new row for 'mean year by platform', then back-fill with ifelse stat
ement
clean <- clean %>%
  left_join(na_key, by = "Platform")
clean$Year <- ifelse(is.na(clean$Year), clean$m, clean$Year)
clean <- select(clean, -m)
head(filter(clean, is.na(Year)))

# All clear



# Genre Check:
unique(clean$Genre)

genre_vgsales <- clean %>%
  group_by(Genre) %>%
  tally(sort=TRUE)
# All clear

# Sales Checks:
max(clean$NA_Sales)           ## [1] 41.49

max(clean$EU_Sales)           ## [1] 29.02

max(clean$JP_Sales)           ## [1] 10.22

max(clean$Other_Sales)        ## [1] 10.57

# Checkin for irregular values

head(filter(clean, NA_Sales <0 | NA_Sales > 20 ))

wii_sport <- filter(clean, NA_Sales <0 | NA_Sales > 40 )
# Wikipedia confirms Wii Sport actually did have 82 million copies sold by 20
17
# Clear

# Is total world wide sales equal to the sum of all sales?
```

```r
## filter(clean, (NA_Sales + EU_Sales + JP_Sales + Other_Sales) != Global_Sales)
# Apparently we're not using the Global column seen on kaggle in this assignment...

# Identified a re-occurring issue in sales:
dplyr::filter(clean, grepl("weekly", Name))

dplyr::filter(clean, grepl("Weekly", Name))

  # 20 rows report only weekly sales figures (multiply by 52)

# Altering all weekly Japanese sales figures to be annual
    # Creating list to sort
weekly_to_annual_all_rows <- filter(clean, grepl("weekly", Name) | grepl("Weekly", Name))
weekly_to_annual_jp_rows <- filter(weekly_to_annual_all_rows,
                                   grepl("JP", Name) | grepl("jp", Name)| grepl("Jp", Name))
weekly_to_annual_titles <- c(weekly_to_annual_jp_rows$Name)
    # looping through
for(i in weekly_to_annual_titles){
  clean[clean$Name == i, "JP_Sales"] <- clean[clean$Name == i, "JP_Sales"]*52.14 # weeks per year
}

# Investigating these other "weekly sales" figures
filter(weekly_to_annual_all_rows, !grepl("JP", Name) & !grepl("jp", Name)& !grepl("Jp", Name))

dplyr::filter(clean, grepl("Tony Hawk's American Wasteland", Name)) # Unclear

dplyr::filter(clean, grepl("NBA Live 06", Name)) # Unclear

dplyr::filter(clean, grepl("Ratchet & Clank: Up Your Arsenal", Name)) # Unclear

dplyr::filter(clean, grepl("Midnight Club 3", Name)) # Unclear

dplyr::filter(clean, grepl("Pokemon Mystery Dungeon: Red", Name)) # Unclear

dplyr::filter(clean, grepl("The Urbz: Sims In the City", Name)) # Unclear

# Doesn't appear to be marks for fixing this...

# As the name suggests, PS2 should be PS
filter(clean, grepl("wrong", Name))

clean[clean$Name=="Pachi-Slot Teiou: Golgo 13 Las Vegas (JP sales, but wrong system)", "Platform"] <- "PS"
  # Will leave it in, but unsure if this was even sold globally
clean[clean$Name=="Lunar 2: Eternal Blue(sales, but wrong system)", "Platform
```

```
"] <- "SCD" # in 1994, the platform should be sega CD
filter(clean, grepl("wrong", Name))

# Fixed

# Identified a re-occurring issue in name:
multirow <- dplyr::filter(clean, grepl("sales", Name))
  # Some of the observations have been split into two rows.
multirow <- filter(multirow, !grepl("all region sales", Name))
multirow <- filter(multirow, !grepl("All region sales", Name))
multirow <- filter(multirow, !grepl("All Region sales", Name))
multirow <- filter(multirow, !grepl("All Region Sales", Name))
multirow <- filter(multirow, !grepl("all regions sales", Name))
multirow <- filter(multirow, !grepl("wrong system", Name))
  # 132 problematic titles that represent at least as many rows but likely mo
re.
  # They each need to be assessed and bound to one row, or in the case of Ame
rican/US/us sales, removed. But this is an enormous undertaking, for so few m
arks so....
```

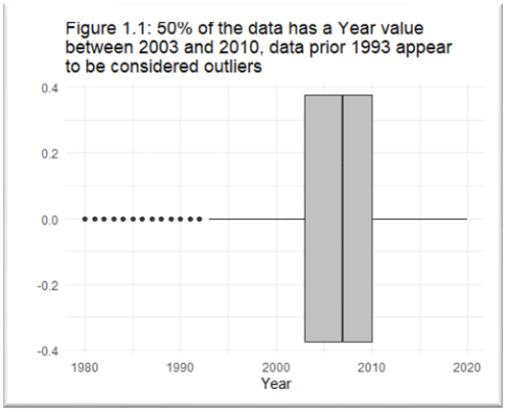
EDA

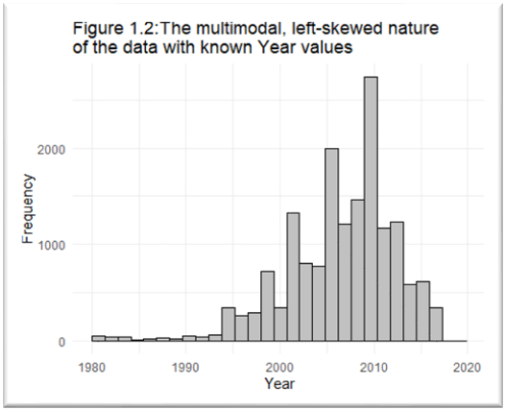
Part 1 - Initial variable inspections

An exploratory data analysis was essential because it provided an opportunity to confirm the validity of our data and ensure it made sense. It would be pointless creating a model from inaccurate or non-nonsensical data. Depending on the model type selected, this phase generally assists in determining what transformations (if any) may be necessary and clearly identifies important characteristics/trends in each variable as well as relationships between variables.

```
# Univariate analysis - Year
  # Histogram
ggplot(clean, aes(x = Year)) +
  geom_histogram(fill="grey", colour ="black") +
  ggtitle(str_wrap("Figure 1.2:The multimodal, left-skewed nature of the data
with known Year values", 70) ) +
  xlab("Year") +
  ylab("Frequency") +
  theme_minimal()

  # Boxplot
ggplot(clean, aes(x = Year)) +
  geom_boxplot(fill = "grey") +
  ggtitle(str_wrap("Figure 1.1: 50% of the data has a Year value between 2003
and 2010, data prior 1993 appear to be considered outliers", 70 )) +
  xlab("Year") +
  ylab("") +
  theme_minimal()
```

```
  # Summary statistics
summary(clean$Year)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1980    2003    2007    2006    2010    2020
```





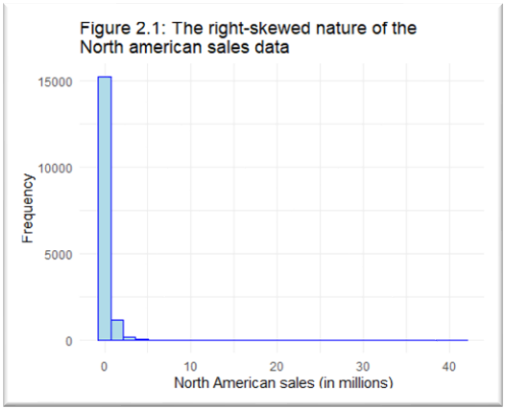
Year

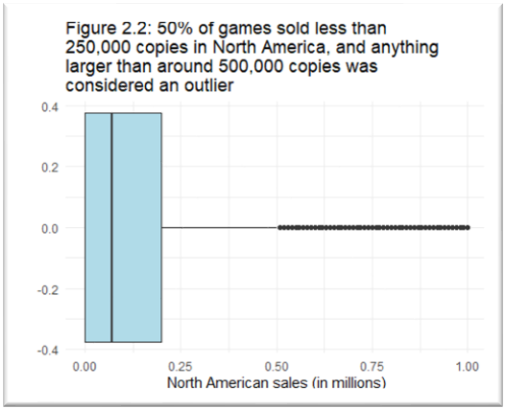
The data sits on a domain of 1980 to 2020, with the interquartile range from 2003 to 2010. It is multi-modal with very few games listed prior to the mid-90's or in the last few years; suggesting that data collection over the entire period has been very inconsistent. The left skew is not what one might expect to see when game development and play has been on the rise over the last few years. This data has been sourced from a public website, perhaps it has seen a decline in users since 2010 resulting in less frequent additions.

The data appears to be a poor representation of historical trends. As copies sold in North America is the outcome variable and not the dollar value, we could ignore the year entirely. However, this leads to a bit of a dangerous assumption. It assumes that wages have grown with inflation and the same proportion of the market can afford to buy a copy no matter the year. It also assumes population sizes (customer population size specifically) have been consistent. Essentially, it removes the 'timeline' element and the relationships observed in time (more on this later).

```
# Univariate analysis - NA_Sales
  # Histogram
ggplot(clean, aes(x = NA_Sales)) +
  geom_histogram(fill="lightblue", colour ="blue") +
  ggtitle(str_wrap("Figure 2.1: The right-skewed nature of the North american
sales data", 70 )) +
  xlab("North American sales (in millions)") +
  ylab("Frequency") +
  theme_minimal()

  # Boxplot
ggplot(clean, aes(x = NA_Sales)) +
  geom_boxplot(fill = "lightblue") +
```

```
  ggtitle(str_wrap("Figure 2.2: 50% of games sold less than 250,000 copies in
North America, \nand anything larger than around 500,000 copies was considere
d an outlier",70)) +
  xlab("North American sales (in millions)") +
  ylab("") +
  scale_x_continuous(limits = c(0, 1)) +
  theme_minimal()

  # Summary statistics
summary(clean$NA_Sales)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0800  0.2647  0.2400 41.4900
```



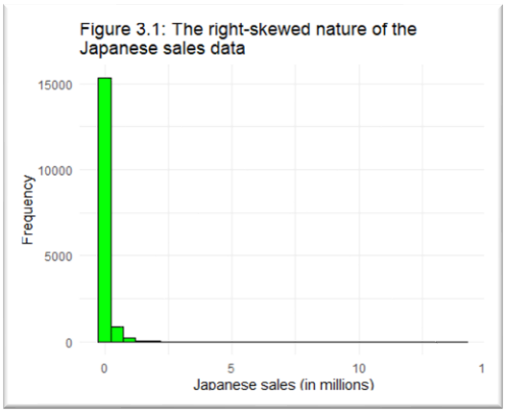


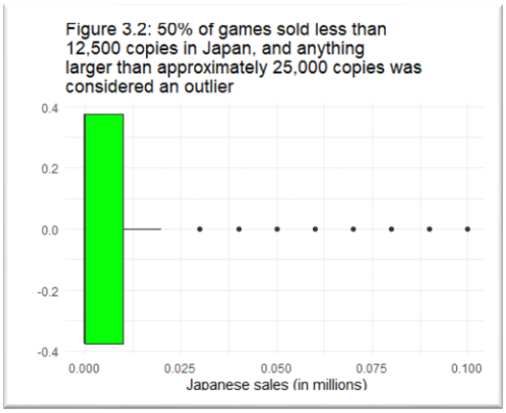
NA_Sales

The NA_Sales variable had a domain of 0 to 41.49 representing the number of copies sold in millions. The interquartile range was from 0 to 0.24 million. The mean number of copies sold was 0.2642 million, but the median was only 0.08 million. The data has a uni-modal shape with a clear right skew when looking at the histogram. The boxplot makes it more apparent that there are many small modes resulting from outliers across the domain.

```
# Univariate analysis - JP_Sales
  # Histogram
ggplot(clean, aes(x = JP_Sales)) +
  geom_histogram(fill="green", colour ="black") +
  ggtitle(str_wrap("Figure 3.1: The right-skewed nature of the Japanese sales
data", 70) ) +
  xlab("Japanese sales (in millions)") +
  ylab("Frequency") +
  theme_minimal()

  # Boxplot
ggplot(clean, aes(x = JP_Sales)) +
  geom_boxplot(fill = "green") +
  ggtitle(str_wrap("Figure 3.2: 50% of games sold less than 12,500  copies in
Japan, and anything larger than \napproximately 25,000 copies was considered
```

```
an outlier", 70)) +
  xlab("Japanese sales (in millions)") +
  ylab("") +
  scale_x_continuous(limits = c(0, .1)) +
  theme_minimal()

  # Summary statistics
summary(clean$JP_Sales)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00000  0.00000  0.00000  0.08321  0.04000 14.07780

filter(clean, JP_Sales>108)

## [1] Name         Platform     Year         Genre        Publisher    NA_Sales
## [7] EU_Sales     JP_Sales     Other_Sales
## <0 rows> (or 0-length row.names)
```



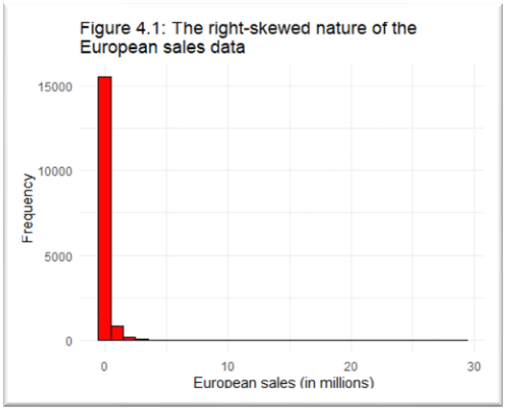


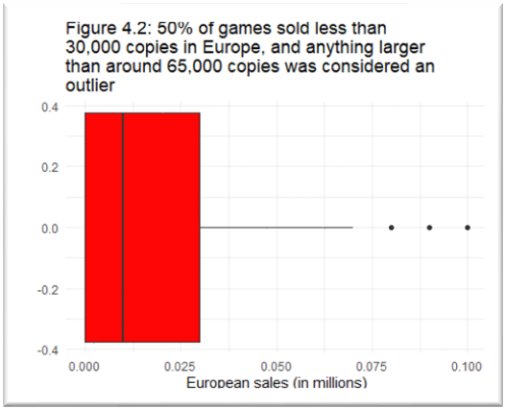
JP_Sales

JP_Sales had a domain from 0 to 14.08 million copies, (this changed because I converted a weekly sales figure to an annual sales figure in the data cleaning phase). The Median value was 0.00, while the mean value was 0.08 million. Similar to North America, the data has a right skew with a dominant mode with many small modes that represent outliers as observed in the boxplot. The median value of 0 indicates that many of the games were not sold in Japan.

```
# Univariate analysis - EU_Sales
  # Histogram
ggplot(clean, aes(x = EU_Sales)) +
  geom_histogram(fill = "red", colour ="black") +
  ggtitle(str_wrap("Figure 4.1: The right-skewed nature of the European sales
data", 70)) +
  xlab("European sales (in millions)") +
  ylab("Frequency") +
  theme_minimal()
```

```
  # Boxplot
ggplot(clean, aes(x = EU_Sales)) +
  geom_boxplot(fill = "red") +
  ggtitle(str_wrap("Figure 4.2: 50% of games sold less than 30,000 copies in
Europe, and anything larger than around 65,000 copies was considered an outli
er", 70)) +
  xlab("European sales (in millions)") +
  ylab("") +
  scale_x_continuous(limits = c(0, .1)) +
  theme_minimal()

  # Summary Statistics
 summary(clean$EU_Sales)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0200  0.1467  0.1100 29.0200
```



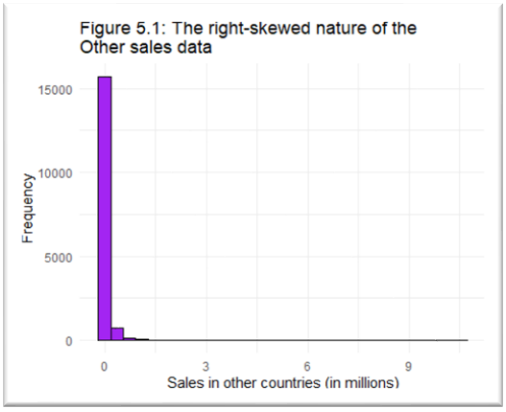


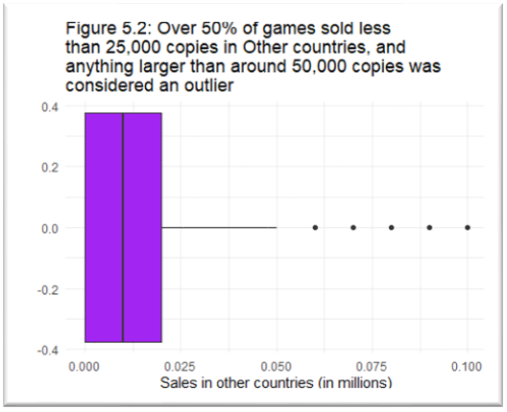
EU_Sales

The EU_Sales data had a domain of 0 to 29.02 million. The median value was 0.02 million, while the mean was 0.15 million. Much like the other sales figures, this too has a dominant mode and a right skew with several smaller modes scattered across the domain in positions identified as outliers.

```
# Univariate analysis - Other_Sales
  # Histogram
ggplot(clean, aes(x = Other_Sales)) +
  geom_histogram(fill = "purple", colour ="black") +
  ggtitle(str_wrap("Figure 5.1: The right-skewed nature of the Other sales da
ta", 70 ) ) +
  xlab("Sales in other countries (in millions)") +
  ylab("Frequency") +
  theme_minimal()

  # Boxplot
ggplot(clean, aes(x = Other_Sales)) +
  geom_boxplot(fill = "purple") +
```

```
  ggtitle(str_wrap("Figure 5.2: Over 50% of games sold less than 25,000 copie
s in Other countries, and anything larger than around 50,000 copies was consi
dered an outlier", 70 )) +
  xlab("Sales in other countries (in millions)") +
  ylab("") +
  scale_x_continuous(limits = c(0, .1)) +
  theme_minimal()

  # Summary statistics
summary(clean$Other_Sales)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00000  0.00000  0.01000  0.04807  0.04000 10.57000
```





Other_Sales

The Other_Sales data had a domain of 0 to 10.57 million. Its mean value was 0.05 million while it's median was 0.01 million. And it's the same story once more, dominant mode, right skew, several outliers.

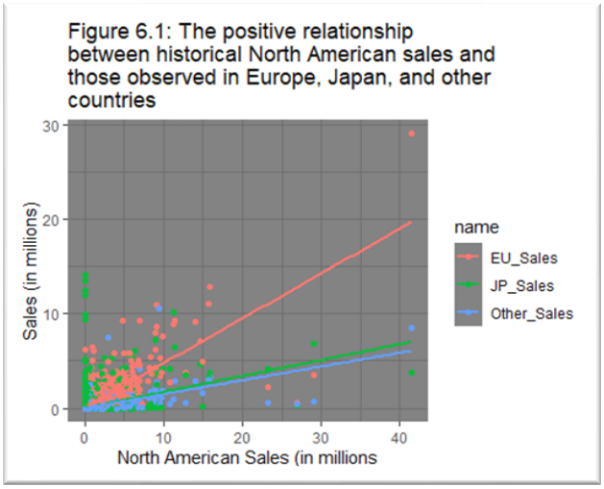
EDA

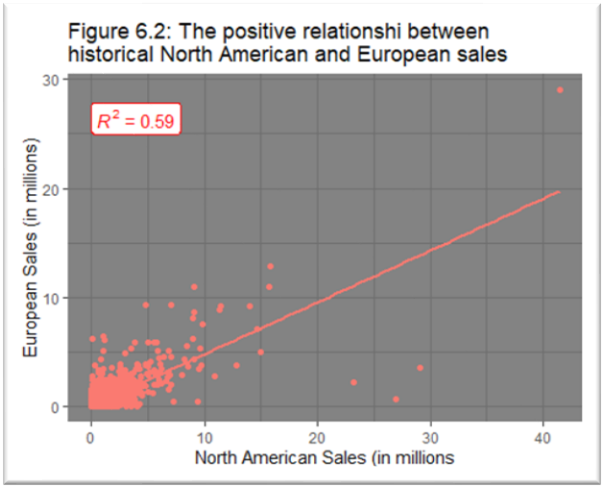
Part 2 - Improving the Information Exchange between Data and Model

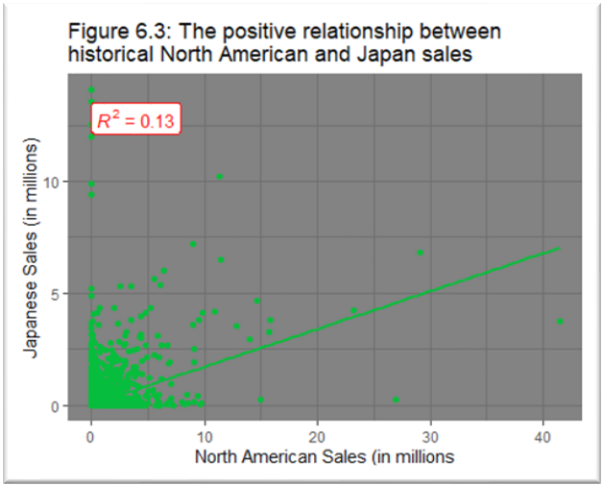
It is expected that the three most important predictors will be JP_Sales, EU_Sales, and Other_Sales because they represent the response to a game released in a given year on a given Platform of a given Genre; in a sense these variables contain latent information representing the other variables. While cultures, languages and preferences differ around the globe, a game worth buying in one part of the world is most likely worth buying in another:

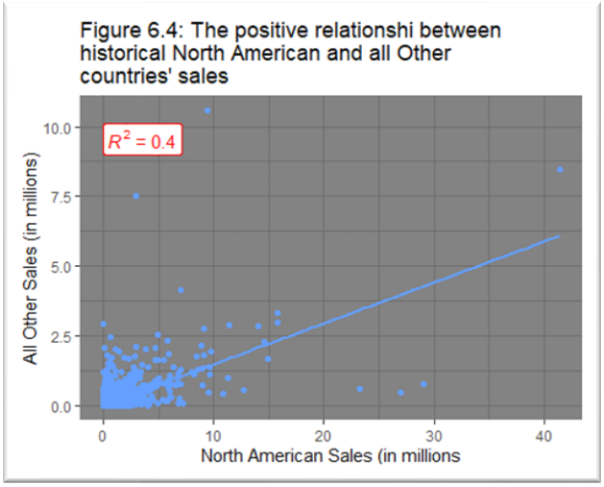
```
# Visualizing the relationship between NA_Sales against all sales variables
  # Comparison of JP_Sales, EU_Sales, and Other_Sales relationships with NA_S
ales
sales_stats <- clean %>%
  pivot_longer(cols = c(EU_Sales, JP_Sales, Other_Sales))
```

```
ggplot(sales_stats, aes(x = NA_Sales, y = value, colour = name)) +
  geom_point() +
  geom_smooth(method=lm, se=FALSE) +
  ggtitle(str_wrap("Figure 6.1: The positive relationship between historical
North American sales and those observed in Europe, Japan, and other countries
", 70))+
  xlab("North American Sales (in millions")+
  ylab("Sales (in millions)") +
  theme_dark()

  # NA_sales and EU_Sales
ggplot(clean, aes(x = NA_Sales, y = EU_Sales)) +
  geom_point(colour = '#F8766D', show.legend = FALSE) +
  geom_smooth(colour = '#F8766D', method = lm, se=FALSE, show.legend = FALSE)
+
  ggtitle(str_wrap("Figure 6.2: The positive relationshi between historical N
orth American and European sales", 70))+
  xlab("North American Sales (in millions")+
  ylab("European Sales (in millions)") +
  theme_dark() +
  ggpubr::stat_cor(aes(label = ..rr.label..), color = "red", geom = "label")

  # NA_sales and JP_Sales
ggplot(clean, aes(x = NA_Sales, y = JP_Sales)) +
  geom_point(colour = '#00BA38', show.legend = FALSE) +
  geom_smooth(colour = '#00BA38', method=lm, se=FALSE, show.legend = FALSE) +
  ggtitle(str_wrap("Figure 6.3: The positive relationship between historical
North American and Japan sales", 70))+
  xlab("North American Sales (in millions")+
  ylab("Japanese Sales (in millions)") +
  theme_dark() +
  ggpubr::stat_cor(aes(label = ..rr.label..), color = "red", geom = "label")

  # NA_sales and Other_Sales
ggplot(clean, aes(x = NA_Sales, y = Other_Sales)) +
  geom_point(colour = '#619CFF', show.legend = FALSE) +
  geom_smooth(colour = '#619CFF', method=lm, se=FALSE, show.legend = FALSE) +
  ggtitle(str_wrap("Figure 6.4: The positive relationshi between historical N
orth American and all Other countries' sales", 70))+
  xlab("North American Sales (in millions")+
  ylab("All Other Sales (in millions)") +
  theme_dark() +
  ggpubr::stat_cor(aes(label = ..rr.label..), color = "red", geom = "label")
```









Key predictors: EU_Sales, JP_Sales, and Other_Sales

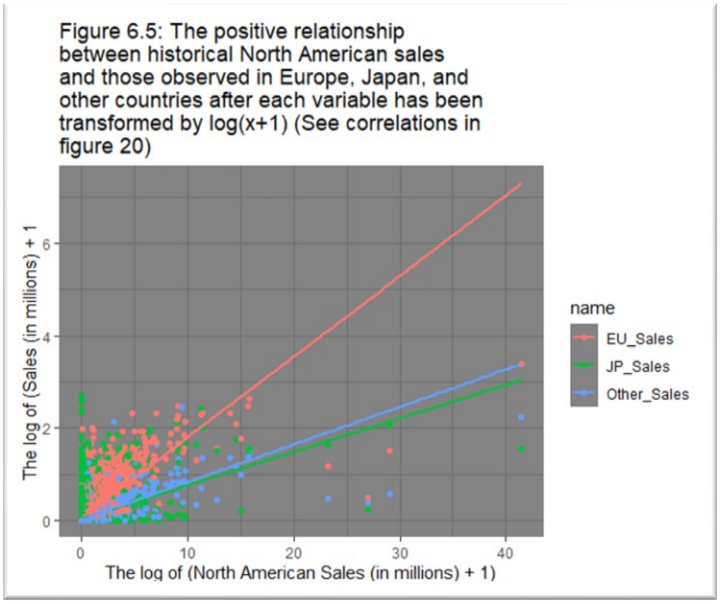
There does appear to be a positive correlation between these sales figures and NA_Sales. EU_Sales has a moderate positive relation, perhaps due to cultural similarities as touched on briefly above. The relationship between JP_Sales is weakly positive. When a game sells well in North America, it doesn't appear to do as well in Japan, which could be partially related to cultural differences but also to the number of customers in each country. Perhaps there are just more people to sell to in North America. Other_Sales follows a similar trend to that of JP_Sales, but it has a stronger correlation.

The right skew in the data should be addressed as a LASSO regression model is essentially still a linear model. Linear models make four key assumptions: linearity, Normality, Homoscedasticity, and Independence.

In the case of a LASSO model specifically though, each term in the otherwise linear model equation is given a penalty that relates to its importance. Essentially weighing each variable in the equation by importance, with a few being weighed by zero and effectively removed. When it comes to the assumptions of A LASSO linear model it is far simpler:

1. linearity - a straight line is expected to give the best fit.
2. Sparsity - only a small number of variables may be relevant
3. The Irrepresentable condition - the important variables are unrelated to the unimportant variables
4. The errors must have a finite variance and a mean of zero, but not necessarily be normally distributed (Hlaváčková-Schindler 2016)

The sales figures were transformed by log(x+1) in the modelling phase:




```
sales_stats <- clean %>%
  pivot_longer(cols = c(EU_Sales, JP_Sales, Other_Sales)) %>%
  mutate(value=log(value+1))
ggplot(sales_stats, aes(x = NA_Sales, y = value, colour = name)) +
  geom_point() +
  geom_smooth(method=lm, se=FALSE) +
  ggtitle(str_wrap("Figure 6.5: The positive relationship between historical
North American sales and those observed in Europe, Japan, and other countries
after each variable has been transformed by log(x+1) \n(See correlations in f
igure 20)", 70))+
  xlab("The log of (North American Sales (in millions) + 1)") +
  ylab("The log of (Sales (in millions) + 1") +
  theme_dark()
```

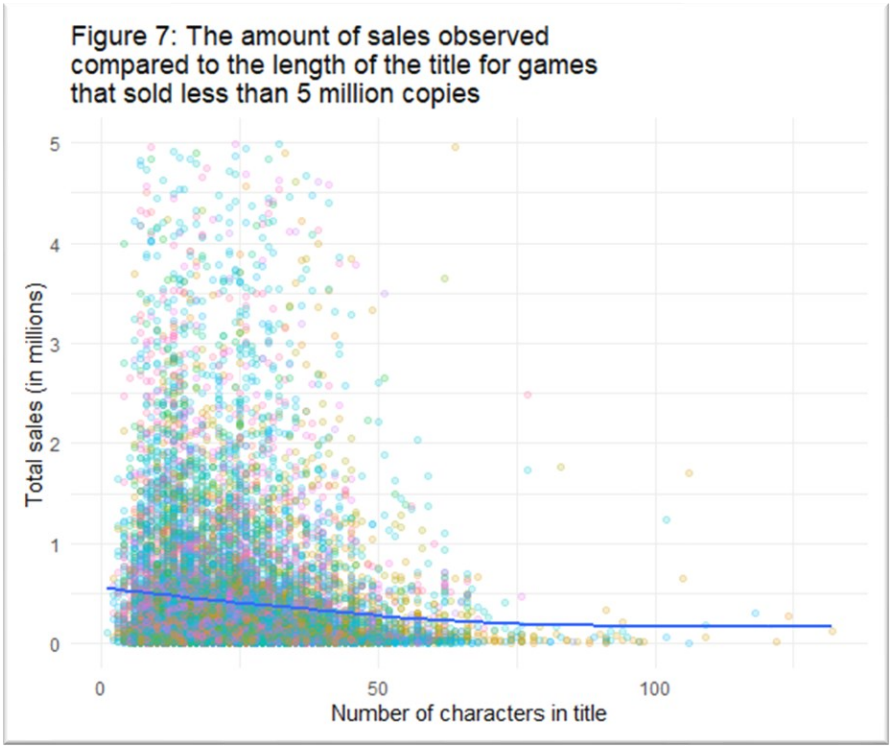
Improvement of Information exchange in other variables:

Name - The title length, Title language, or franchise indicators (such as ':', '-', 'II','2', reoccurring 'prefix' strings, etc.) is probably more informative than the exact String. The language indicates a proportion of the world that can effortlessly play the game and understand the title, and signs of a franchise/sequel indicate a pre-existing fan-base that will help drive sales. Longer titles might be less enticing to consumers; a trend observable in the data. The trend becomes less obvious when you include all the games, but when you take a subset (such as games that sell up to 5 million copies) you effectively remove pre-existing franchises and get a sense of how a new game with no pre-existing fan base sells. Also verified the assumption that no specific platform had guidelines regarding title length.

```
# Finding Name length and total sales
name_length <- clean %>%
  mutate(total_sales = (NA_Sales+EU_Sales+JP_Sales+Other_Sales))%>%
  group_by(Name,Platform, total_sales) %>%
  tally() %>%
  mutate(chr_length = nchar(Name))


# Creating table
name_length %>%
  ungroup() %>%
  select(c("Name", "chr_length")) %>%
  head(4) %>%
  knitr::kable(caption = "Table 1: The number of characters in each title")


# Revealing any relationship between title length and sales
ggplot(filter(name_length, total_sales < 5), aes(x = chr_length, y = total_sa
les)) +
  geom_point(aes(colour = Platform), alpha=0.2, show.legend = FALSE) +
  geom_smooth(se=FALSE)+
  labs(title = str_wrap("Figure 7: The amount of sales observed compared to t
he length of the title for games that sold less than 5 million copies", 70),
       x = "Number of characters in title",
       y = "Total sales (in millions)") +
  theme_minimal()
```

Table 1:Example of how incorrect names influenced title length.

Name	chr_length
'98 Koshien	11
.hack//G.U. Vol.1//Rebirth	26
.hack//G.U. Vol.2//Reminisce	28
.hack//G.U. Vol.2//Reminisce (jp sales)	39



Publisher - Although this variable was not a predictor itself, it is worthwhile investigating for trends that might provide some inspiration for the model building phase. For instance, some Publishers (such as 'Nintendo') are also the producers of the console. The name alone does not give an indication of how well established the company is nor the fan base they have in terms of market share. We can get an estimate of this if we instead consider the average sales per game for each Publisher. Below I've prepared a plot to approximate the average proportion of market share each Publisher has:

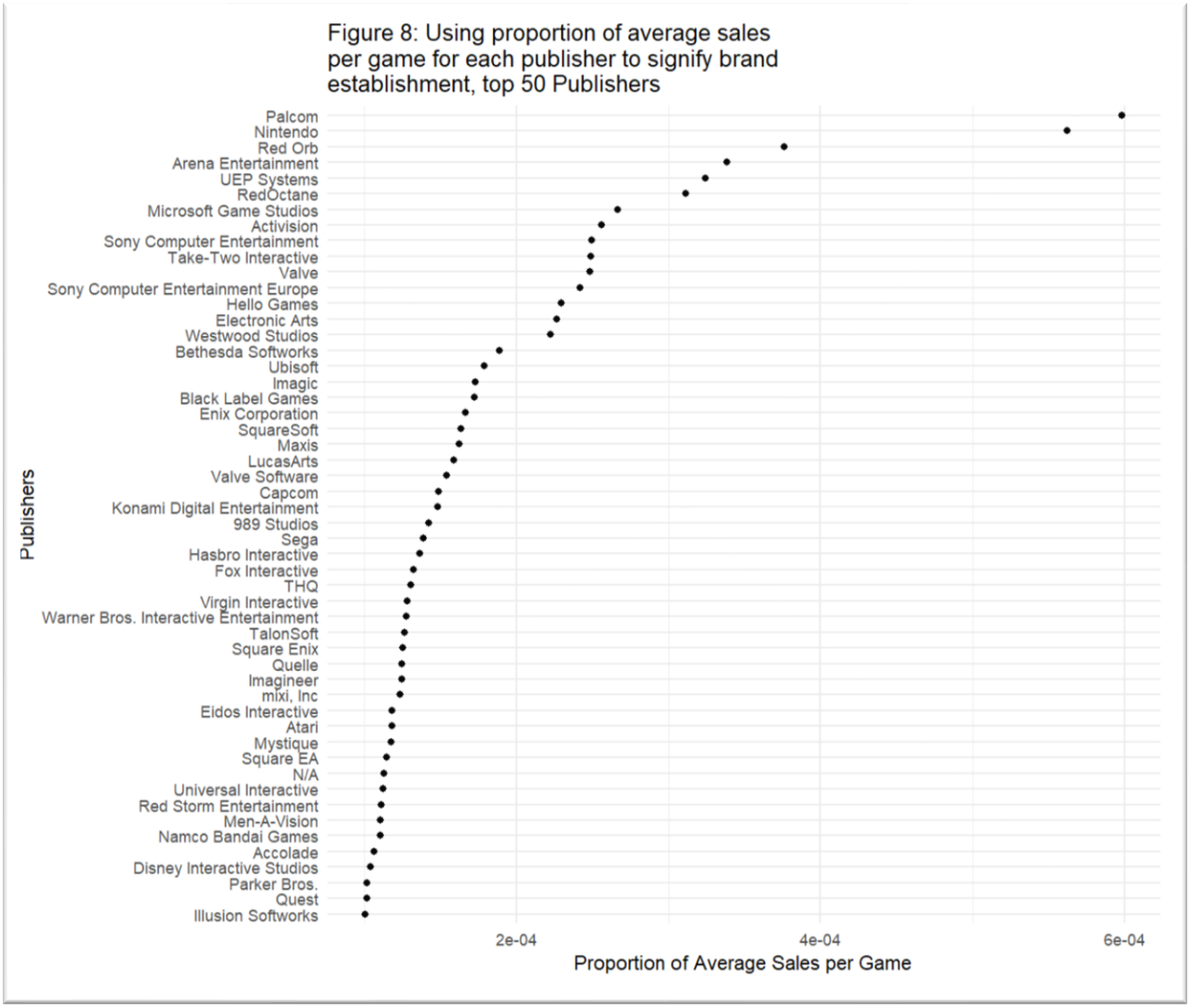
```
# Finding the total sales for each Publisher
clean$Publisher <- as.factor(clean$Publisher)
established <- clean %>%
  mutate(publisher_sales = (NA_Sales+EU_Sales+JP_Sales+Other_Sales),
         releases = 1) %>%
  group_by(Publisher, publisher_sales, releases) %>%
  tally() %>%
  group_by(Publisher)%>%
  summarise(releases = sum(releases),
            publisher_sales = sum(publisher_sales)) %>%
  mutate(estimated_market_share = ((publisher_sales/releases)/sum(publisher_s
ales)))

# Creating table

established %>%
  ungroup() %>%
  select(c("Publisher", "publisher_sales")) %>%
```

```
  arrange(desc(publisher_sales) ) %>%
  head(25) %>%
  knitr::kable(caption = "Table 2: The top 25 Publishers in terms of sales in millions")
```

Table 2: The top 25 Publishers in terms of sales (in millions). The size of some of these companies made it necessary to consider average sales numbers in figure 8 which gives a better overview of average performance.

Publisher	publisher_sales
Nintendo	1630.6862
Electronic Arts	675.0526
Activision	528.0100
Sony Computer Entertainment	487.6536
Take-Two Interactive	347.0400
Ubisoft	325.0714
Microsoft Game Studios	234.4098
Konami Digital Entertainment	199.3786
THQ	193.5200
Sega	175.1600
Capcom	166.7698
Warner Bros. Interactive Entertainment	128.2400
Namco Bandai Games	124.5300
Square Enix	118.7400
Atari	116.0900
Disney Interactive Studios	96.2800
Eidos Interactive	87.3700
LucasArts	82.8800
Bethesda Softworks	80.3000
SquareSoft	53.6200
Midway Games	50.6900
Acclaim Entertainment	46.4900
505 Games	44.9400
Vivendi Games	42.1700
Virgin Interactive	41.9500

```
# Plotting the Publishers that produced more than 20 million sales
ggplot(filter(established, estimated_market_share > 0.0001),
       aes( x = estimated_market_share, y = fct_reorder(Publisher, estimated_market_share))) +
  geom_point() +
  ggtitle(str_wrap(" Figure 8: Using proportion of average sales per game for each publisher to signify brand establishment, top 50 Publishers", 45)) +
  ylab("Publishers") +
  xlab("Proportion of Average Sales per Game") +
  theme_minimal()
```



Platform - Some of the consoles are no longer circulated, failed to capture a significant proportion of the market, or no longer exist. The information they provide is not all that informative. But there is something to be said about leaving them in the model. There are patterns when it comes to technology adoption (innovators, early adopters, early majority, late majority, laggards) and if technology that is at the end of its adoption curve were removed, then there is a risk the model won't detect those latent patterns. Whether or not such information will be provided is uncertain, however it's inclusion will alter probabilities/odds in a way that better represents reality. Below are the platforms that have a total of less than 100,000 sales, as the list is incomplete (does not contain every game released on that console over that period of time) the adoption curves are less apparent:

```
# Finding the amount of sales per Platform over time
meaningful_platform <- clean %>%
  mutate(total_sales = (NA_Sales+EU_Sales+JP_Sales+Other_Sales))%>%
  group_by(Platform, Year, total_sales) %>%
```

```
  tally() %>%
  summarise(platform_sales_for_year = sum(total_sales))



# Creating table
clean %>%
  mutate(total_sales = (NA_Sales+EU_Sales+JP_Sales+Other_Sales))%>%
  group_by(Platform, total_sales) %>%
  tally() %>%
  summarise(Total_platform_sales = sum(total_sales)) %>%
  arrange(desc(Total_platform_sales))
  knitr::kable(caption = "Table 3: Sales made for each category of platform")
```

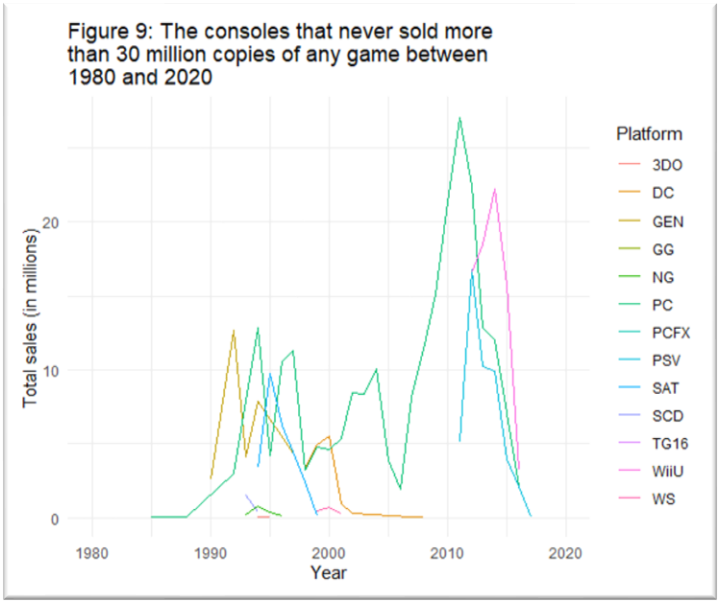
Table 3: Sales made for each category of platform over the time period. This table demonstrates the bias in Year as a result of poor data acquisition. PS4 sales up to the year 2020 are expected to be underestimated here.

Platform	Total_platform_sales
PS2	720.8162
Wii	687.7600
X360	677.1614
PS3	631.3200
DS	518.6700
PS	435.2500
GB	258.9122
PS4	249.5000
NES	236.6900
GBA	222.5942
3DS	202.9300
N64	200.8628
PC	183.1400
SNES	171.0800
PSP	160.5100
XB	146.1998

Platform	Total_platform_sales
GC	130.2400
XOne	125.2900
2600	75.0700
WiiU	71.6000
PSV	36.0700
GEN	31.5498
SAT	15.6200
DC	14.3700
SCD	1.8600
WS	1.4200
NG	1.2400
TG16	0.1600
3DO	0.0800
GG	0.0400
PCFX	0.0300

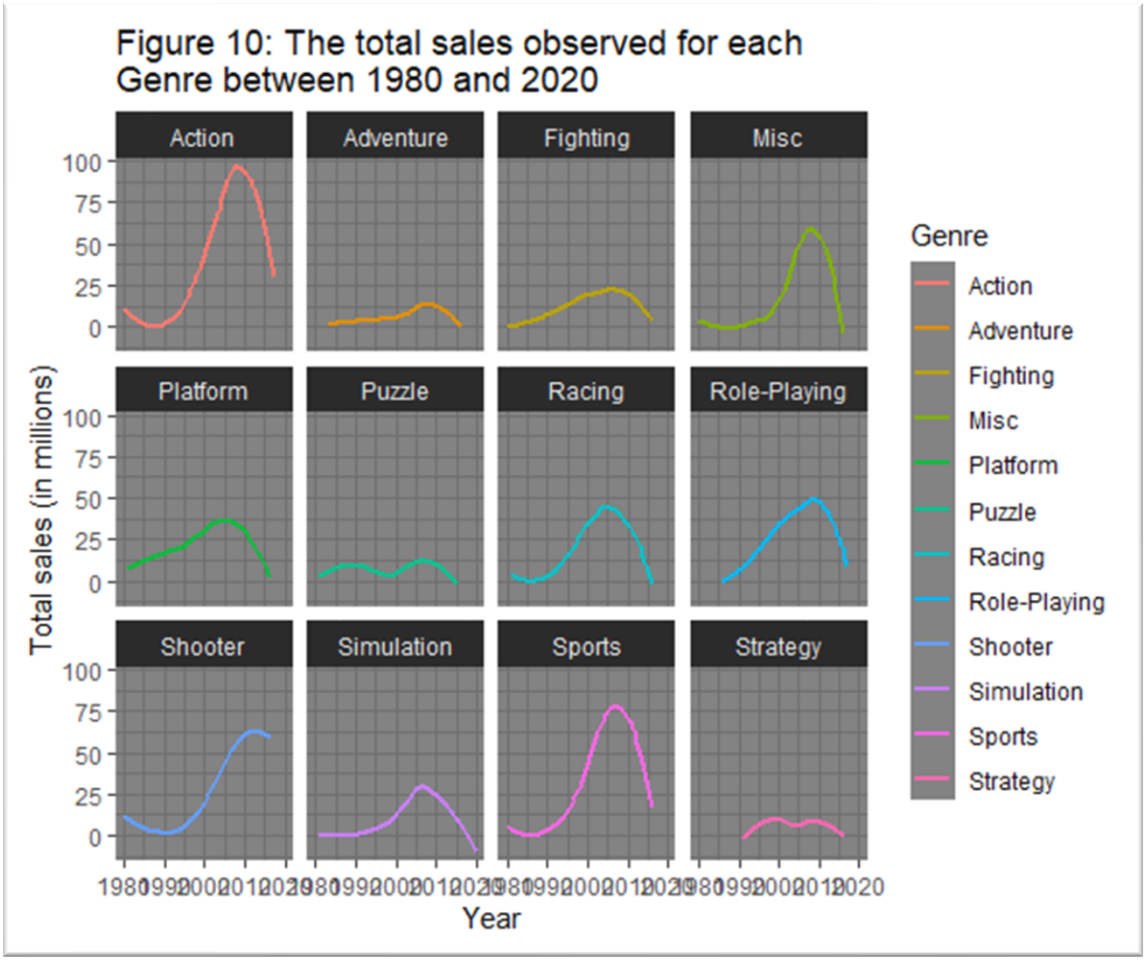
```


s# Plotting the above data set
ggplot(filter(meaningful_platform, max(platform_sales_for_year)<30),
       aes( x = Year,
            y = platform_sales_for_year,
            colour = Platform)) +
  geom_line() +
  ggtitle(str_wrap("Figure 9: The consoles that never sold more than 30 milli
```

```
on copies of any game between 1980 and 2020", 70)) +
  ylab("Total sales (in millions)") +
  xlab("Year") +
  xlim(c(1980,2020)) +
  theme_minimal()
```



Year - Useful in association with other variables like Genre, as it maps preference trends. (It might also be useful to measure the optimal release time for a sequel when taken in association with Name). It will most likely have more meaning as an interaction term than on its own. (See below and in the summary for more commentary on Year)

```
# Finding the total amount of sales for each Genre over time
historic_trend <- clean %>%
  mutate(total_sales = (NA_Sales+EU_Sales+JP_Sales+Other_Sales))%>%
  group_by(Genre, Year, total_sales) %>%
  tally() %>%
  summarise(total_sales = sum(total_sales))

#Plotting the trends and consumer preferences in Genre over time
ggplot(historic_trend, aes(x = Year, y = total_sales, colour = Genre)) +
  geom_smooth(se=FALSE) +
  facet_wrap(.~Genre) +
  ggtitle(str_wrap("Figure 10: The total sales observed for each Genre betwee
n 1980 and 2020", 70)) +
  ylab("Total sales (in millions)") +
  xlab("Year") +
  theme_dark()
```



This chart may be interpreted as indicating that all genres of game have been on the decline since around 2010. This is a deceptive plot because data acquisition has been irregular. Furthermore, the data itself is an incomplete list of all the games released/sold each year and perhaps not very representative. This trend may still exist as a result of the growth of Apple and Android mobile gaming services, but that is not an assertion that can be made with the data on hand. Furthermore, no binary predictor indicating a growing or declining/stagnation of genre could be provided to estimate current market preferences as had been considered initially.

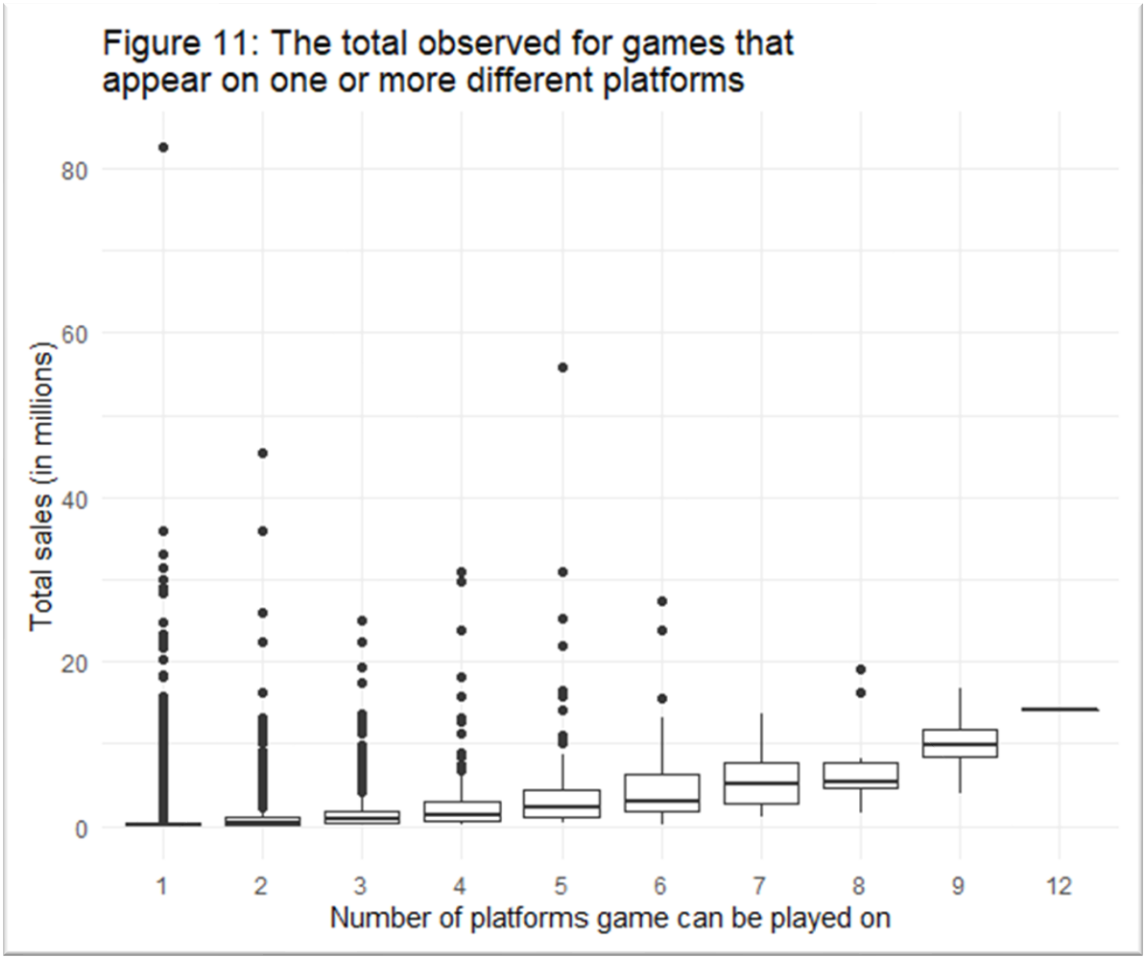
Platform -There is more information that can be squeezed from the data. The number of platforms the game has been released on. There are two opposing strategies when it comes to economically producing a game:

1. Put all the money into creating a great game for one console, or
2. Create an okay game and split the remaining funds for engineering it for each platform and the associated advertising costs.

There are of course different budgets and trade-offs each Publisher has to decide on, and that leads to a certain degree of variation. However, the assertion that a wider fan base will

increase sales because of the increased word of mouth and accessibility to game help/ community, etc seems to be fairly well identified in the data:

```
# Finding the Number of platforms a game can be played on and its sales data
platform_count <- clean %>%
  mutate(total_sales = (NA_Sales+EU_Sales+JP_Sales+Other_Sales)) %>%
  group_by(Name, total_sales) %>%
  tally(sort = TRUE) %>%
  summarise(total_sales = sum(total_sales), n=sum(n))

# Plotting sales vs number of platforms the game can be played on
ggplot(platform_count, aes(x = as.factor(n), y = (total_sales))) +
  geom_boxplot() +
  ggtitle(str_wrap("Figure 11: The total observed for games that appear on on
e or more different platforms", 70)) +
  ylab("Total sales (in millions)") +
  xlab("Number of platforms game can be played on") +
  theme_minimal()
```



```
filter(platform_count, n > 11) # it was 'Need for Speed: Most Wanted'

## # A tibble: 1 x 3
##   Name                          total_sales     n
##   <chr>                               <dbl> <int>
## 1 Need for Speed: Most Wanted          14.1    12

# Preparing tibble to merge the potentially meaningful variable later
platform_count <- platform_count[c("Name", "n")]
platform_count$platforms_available_for_title <- platform_count$n
platform_count$n <- NULL

# Creating table
platform_count %>%
  ungroup() %>%
  select(c("Name","platforms_available_for_title")) %>%
  head(20) %>%
  knitr::kable(caption = "Table 4: An example of the platform counts for each
title.")
```

Table 4: An example of the platform counts for each title.

Name	platforms_available_for_title
'98 Koshien	1
.hack//G.U. Vol.1//Rebirth	1
.hack//G.U. Vol.2//Reminisce	1
.hack//G.U. Vol.2//Reminisce (jp sales)	1
.hack//G.U. Vol.3//Redemption	1
.hack//Infection Part 1	1
.hack//Link	1
.hack//Mutation Part 2	1
.hack//Outbreak Part 3	1
.hack//Quarantine Part 4: The Final Chapter	1
.hack: Sekai no Mukou ni + Versus	1
[Prototype 2]	3
[Prototype]	2
007 Racing	1
007: Quantum of Solace	6
007: The World is not Enough	2
007: Tomorrow Never Dies	1
1 vs. 100	1
1/2 Summer +	1
10 Minute Solution	1

To summaries,

- Title length was more informative than the game Name. it is notable however that names had not been corrected prior to use and a few rows will have more characters than they should as a result of additions like "…(jp weekly sales)" (**exclude Name**, include Title_length)
- The Platform was included, despite the argument that the data should be subdivided to only include current platforms so as probabilities/odds could be better calculated to reflect the immediate present; however, including them all leads to a more conservative prediction that accounts for early adopters in competing new platforms. (**Include Platform**)
- Year was included as it contained latent information. The consequences of its exclusion have been carefully considered. As the data provided is irregular, incomplete, and without any reasonable way to tell if its representative of reality from year to year in terms of proportion of games per platform or game genre sales, there is an argument to exclude Year and instead aim to make an 'average' prediction. Average in the sense that it would interpret every row as being equivalent in weight and relevance. However, the amount of latent information Year contains may justify aiming for a more relevant time-dependent prediction. comparing the accuracy of the two opposing models may be impossible as they effectively predict two different things; sales in today's climate, and a time-independent sales average. (**Include Year**)
- Genre had enough variation between locations for it to be an informative predictor as will be seen below. (**Include Genre**)
- The Publisher variable did not supply enough information on its own, and as the predictive model will not be given publisher as a predictor anyway, it had to be excluded. (**Exclude Publisher**)
- The sales in locations other than North America were informative and were included. (**Include EU_Sales, JP_Sales, and Other_sales**)
- There was so much variation in the data that the information provided by the Number of Platforms predictor is not as precise as I'd thought it might be, but it may have some predictive value. (**Try with Platform_count**)

Data Cleaning

Part 3 - Consolidating Data for the modelling pre-processing phase

Below I have modified the data into the form that was taken into the next phase. Note: variables were not reduced until their inadequacy had been verified:

```
# Defining new data frame and altering missing values
vgsales_df <- clean %>%
# Adding total sales
```

```
  mutate(Total_sales = (NA_Sales+EU_Sales+JP_Sales+Other_Sales)) %>%
# Adding title length
  mutate(Title_length = nchar(Name)) %>%
# Adding Publisher's Total Sales
  left_join(clean, established, by = c("Name","Platform","Year","Genre",
                                       "Publisher","NA_Sales","EU_Sales",
                                       "JP_Sales","Other_Sales"))

# Adding platform dominance, note: Uses entire time period
vgsales_df <- left_join(vgsales_df, meaningful_platform, by = c("Platform","Y
ear"))
# Adding the Number of available platforms per game
vgsales_df <- left_join(vgsales_df, platform_count, by = "Name")

# Setting data types for all variables, regardless of inclusion
vgsales_df$Name <- as.factor(vgsales_df$Name)
vgsales_df$Platform <- as.factor(vgsales_df$Platform)
vgsales_df$Year <- as.Date(as.character(vgsales_df$Year), format = "%Y")
vgsales_df$Genre <- as.factor(vgsales_df$Genre)
vgsales_df$Publisher <- as.factor(vgsales_df$Publisher)
vgsales_df$NA_Sales <- as.integer(vgsales_df$NA_Sales)
vgsales_df$EU_Sales <- as.integer(vgsales_df$EU_Sales)
vgsales_df$JP_Sales <- as.integer(vgsales_df$JP_Sales)
vgsales_df$Other_Sales <- as.integer(vgsales_df$Other_Sales)
vgsales_df$Total_sales <- as.integer(vgsales_df$Total_sales)
vgsales_df$platform_sales_for_year<- as.integer(vgsales_df$platform_sales_for
_year)
vgsales_df$Title_length <- as.integer(vgsales_df$Title_length)
vgsales_df$platforms_available_for_title <- as.integer(vgsales_df$platforms_a
vailable_for_title)

# Preview
head(vgsales_df, 6)

# Final Skim
skim(vgsales_df)
```

Dylan Rohan - a1844790

Data summary

Name	vgsales_df
Number of rows	16597
Number of columns	13
_____	
Column type frequency:	
Date	1
factor	4
numeric	8
_____	
Group variables	None

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
Year	0	1	1980-04-22	2020-04-22	2007-04-22	39

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Name	0	1	FALSE	11493	Nee: 12, FIF: 9, LEG: 9, Mad: 9
Platform	0	1	FALSE	31	DS: 2163, PS2: 2160, PS3: 1329, Wii: 1324
Genre	0	1	FALSE	12	Act: 3316, Spo: 2346, Mis: 1739, Rol: 1488
Publisher	0	1	FALSE	579	Ele: 1351, Act: 975, Nam: 932, Ubi: 921

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
NA_Sales	0	1	0.11	0.75	0	0	0	0	41	▇▁▁▁▁
EU_Sales	0	1	0.05	0.45	0	0	0	0	29	▇▁▁▁▁
JP_Sales	0	1	0.03	0.34	0	0	0	0	14	▇▁▁▁▁
Other_Sales	0	1	0.01	0.15	0	0	0	0	10	▇▁▁▁▁
Total_sales	0	1	0.30	1.52	0	0	0	0	82	▇▁▁▁▁
Title_length	0	1	23.97	12.79	1	14	22	31	132	▇▃▁▁▁
platform_sales_for_year	0	1	72.86	51.49	0	29	59	114	177	▇▇▆▆▃
platforms_available_for_title	0	1	2.12	1.60	1	1	1	3	12	▇▁▁▁▁

At this point there were 13 columns and 16,327 rows. One of the variables was of a date class, 4 were factors and the remaining 8 were numeric. The details of each can be viewed in the data dictionary provided below:

```
# Creating Data dictionary
variable_description <- c("The Name of the game",
                          "The Platform of the games release",
                          "The Year of release, please note that only the yea
r is relevant. The day and month are not accurate.",
                          "The Genre of the game",
                          "The Publisher of the game",
                          "The copies sold in North America (in millions)",
                          "The copies sold in Europe (in millions)",
                          "The copies sold in Japan (in millions)",
                          "The copies sold in all other coutries (in millions
)",
"The sum of sales in North America, Europe, Japan, and Other",
"The number of characters in the title",
"The dominance of the platform as represented by the number of games sold on
that platform for each year",
"The number of platforms that game has been released on")

variable_type <- c(1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)

linker <- build_linker(my.data = vgsales_df,
                       variable_description = variable_description,
                       variable_type = variable_type)

data_dictionary <- build_dict(my.data = vgsales_df,
                              linker = linker,
                              option_description = NULL,
                              prompt_varopts = FALSE)

knitr::kable(data_dictionary, caption = "Table 5:The data dictionary")
```

Table 5: ***The data dictionary***

variable_name	variable_description	variable_options
EU_Sales	The copies sold in Europe (in millions)	0 to 29
Genre	The Genre of the game	Sports, Platform, Racing, Role-Playing, Puzzle, Misc, Shooter, Simulation, Action, Fighting, Adventure, Strategy
JP_Sales	The copies sold in Japan (in millions)	0 to 14
NA_Sales	The copies sold in North America (in millions)	0 to 41
Name	The Name of the game	Wii Sports…. (super long list) UIG Entertainment
Title_length	The number of characters in the title	1 to 132
Total_sales	The sum of sales in North America, Europe, Japan, and Other	0 to 82
Year	The Year of release, please note that only the year is relevant. The day and month are not accurate.	1980-04-22 to 2020-04-22

Exploratory Data Analysis

Part 2 - Inspecting the outcome and predictor variables

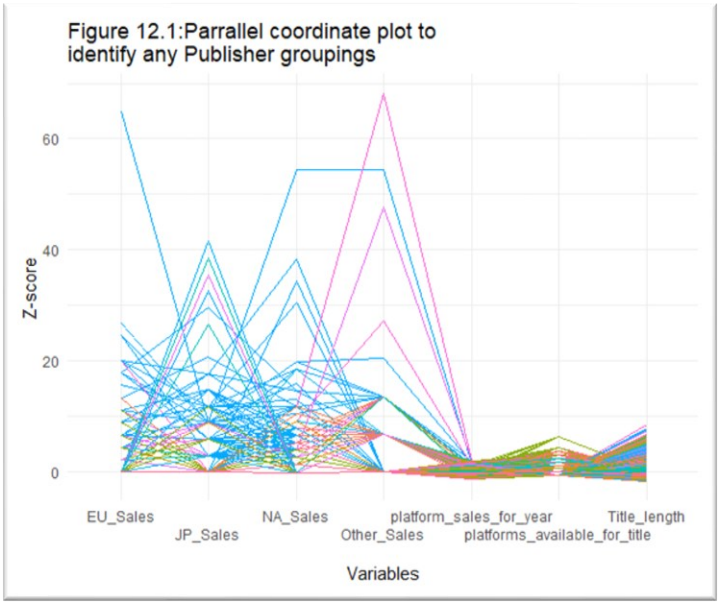
Is there any obvious natural grouping structure to the variables?

Some natural grouping structures are to be expected. For example, some game genres are more common on particular platforms or in particular regions, the sales in North America probably follow similar trends to what is observed in Europe but less similar to that in Japan, the year of release will likely group with the platform type, the year is also likely to group with the number of sales, etc. Parallel Coordinate Plots have been used to assess the natural grouping structures in the numerical data:

```
# Providing an indexation code
vgsales_id <- vgsales_df %>%
  mutate(id = row_number())
head(vgsales_id)
```

```
##                       Name Platform       Year        Genre Publisher NA_Sales
## 1               Wii Sports      Wii 2006-04-22       Sports  Nintendo       41
## 2        Super Mario Bros.      NES 1985-04-22     Platform  Nintendo       29
## 3           Mario Kart Wii      Wii 2008-04-22       Racing  Nintendo       15
## 4        Wii Sports Resort      Wii 2009-04-22       Sports  Nintendo       15
## 5 Pokemon Red/Pokemon Blue       GB 1996-04-22 Role-Playing  Nintendo       11
## 6                   Tetris       GB 1989-04-22       Puzzle  Nintendo       23
…
```

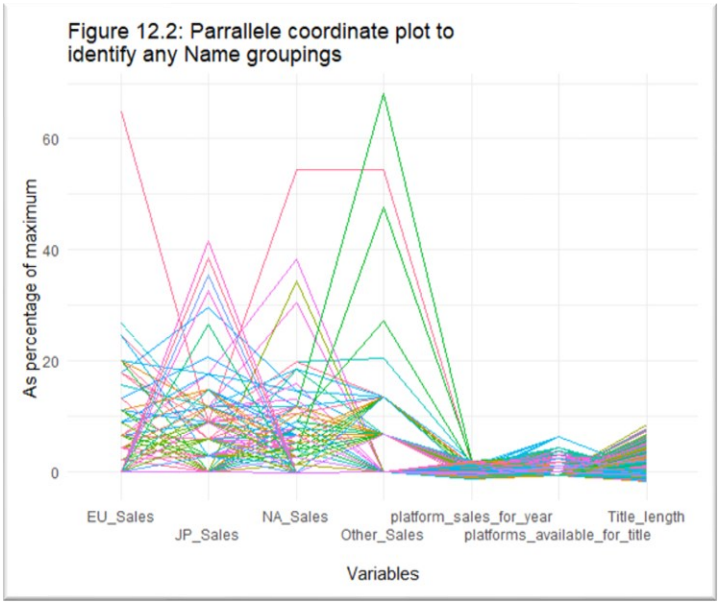
```
# Normalize Sales (removing range differences) then convert it to long format
  # grouping best observed with z-scores
df_long <-  vgsales_id %>%
  mutate(NA_Sales = scale(NA_Sales),
         EU_Sales = scale(EU_Sales),
         JP_Sales = scale(JP_Sales),
         Other_Sales = scale(Other_Sales),
         platform_sales_for_year = scale(platform_sales_for_year),
         platforms_available_for_title = scale(platforms_available_for_title)
,
         Title_length = scale(Title_length)) %>%
  pivot_longer(cols = c(NA_Sales, EU_Sales, JP_Sales, Other_Sales,
                        Title_length, platform_sales_for_year,
                        platforms_available_for_title))


# Parallel plot showing Platform groupings
df_long %>%
  ggplot(aes(x = name, y = value, colour = Publisher)) +
  geom_line(aes(group = id), show.legend = FALSE) +
  ggtitle(str_wrap("Figure 12.1:Parrallel coordinate plot to identify any Pub
lisher groupings", 70)) +
  xlab("\nVariables") +
  ylab("Z-score") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_minimal()
```



Publisher groupings

There appeared to be natural grouping structures when it came to Publishers in different locations. The sales in Japan seemed to be the opposite of those in Europe and North America. Platform_sales_for_year is uninformative because different publishers may make games for multiple platforms, there are no clear trends observed in platforms_available_for_title. There didn't appear to be any rules regarding title length from any Publisher as Publishers/colours didn't appear to stay below any specific value.

```
# Parallel plot showing Name groupings
df_long %>%
  ggplot(aes(x = name, y = value, colour = Name)) +
  geom_line(aes(group = id), show.legend = FALSE) +
  ggtitle(str_wrap("Figure 12.2: Parrallele coordinate plot to identify any N
ame groupings", 70)) +
  xlab("\nVariables") +
  ylab("As percentage of maximum") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_minimal()
```




```
# IF TIME, CREATE BINARY FRANCHISE COLUMN
```

Name groupings

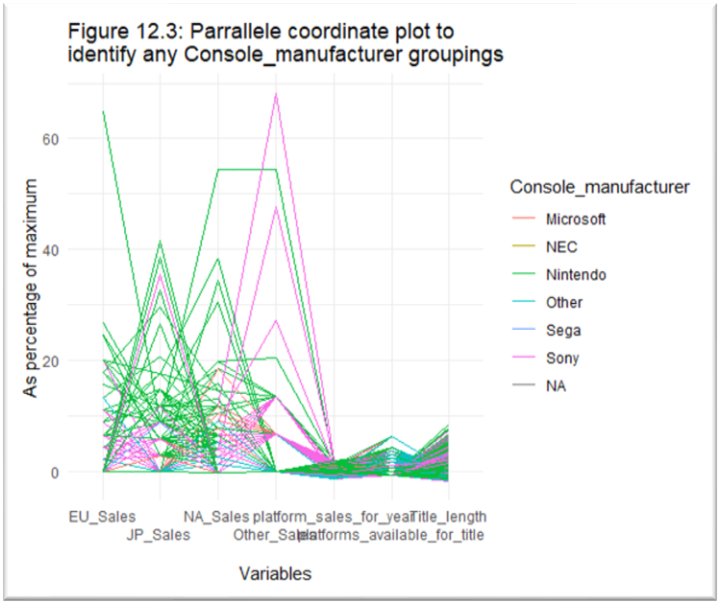
Nothing hidden in the name apart from maybe Japanese titles selling in Japan and English titles perhaps not selling as much. This was not directly useful, but for the sake of completeness and being thorough all avenues were considered.

```
# Clean up this next one a bit by grouping Platform manufacturers
Plat_mnfctr <- read.csv("console_producer.csv") # Faster in excel
head(Plat_mnfctr, 5)
```

```
##     Producer Platform
## 1   Nintendo      Wii
## 2   Nintendo      NES
## 3   Nintendo       GB
## 4   Nintendo       DS
## 5  Microsoft     X360
```

```
# Parallel plot showing Platform manufacturer groupings
df_long %>%
  left_join(Plat_mnfctr, by = "Platform") %>%
  rename(Console_manufacturer = Producer) %>%
  ggplot(aes(x = name, y = value, colour = Console_manufacturer))+
  geom_line(aes(group = id)) +
  ggtitle(str_wrap("Figure 12.3: Parrallele coordinate plot to identify any C
onsole_manufacturer groupings", 70)) +
  xlab("\nVariables") +
  ylab("As percentage of maximum") +
```

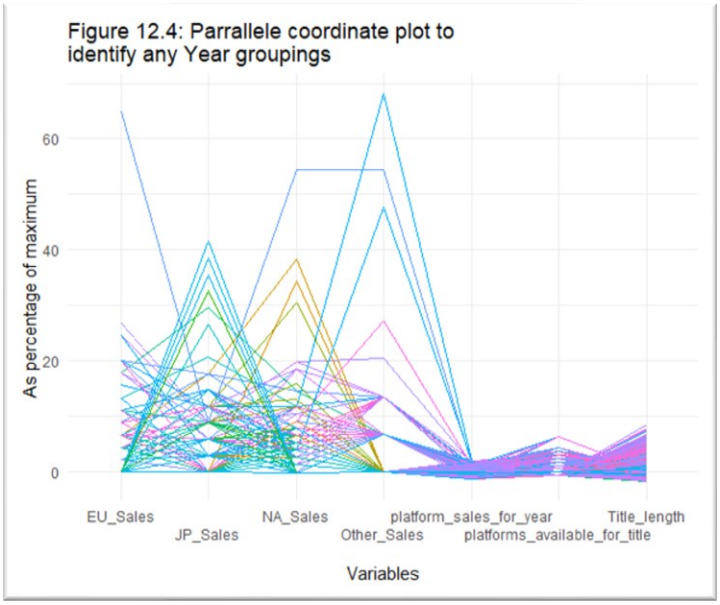
```
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_minimal()
```



Platform groupings

Clearly 'Nintendo' sold far better in Japan, and 'Sony' did far better in the other locations. Unsurprisingly, platform_sales_for_year and platforms_available_for_title offer no meaningful information and, again, there doesn't appear to be any regulations regarding game title lengths for any particular console_manufacturer either.

```
# Parallel plot showing Year groupings
df_long %>%
  ggplot(aes(x = name, y = value, colour = as.factor(Year))) +
  geom_line(aes(group = id), show.legend = FALSE) +
    ggtitle(str_wrap("Figure 12.4: Parrallele coordinate plot to identify any
Year groupings", 70)) +
  xlab("\nVariables") +
  ylab("As percentage of maximum") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_minimal()
```



Year groupings

With a more detailed analysis, year groupings may be uncovered as a result of wage growth, inflation, and delayed population growth (delayed because babies don't play video-games) from year to year, but it isn't apparent in this chart. Platform_sales_for_year, platforms_available_for_title, and Title-length does not appear to have obvious natural grouping structures with Year either.

Is a PCA analysis worthwhile?

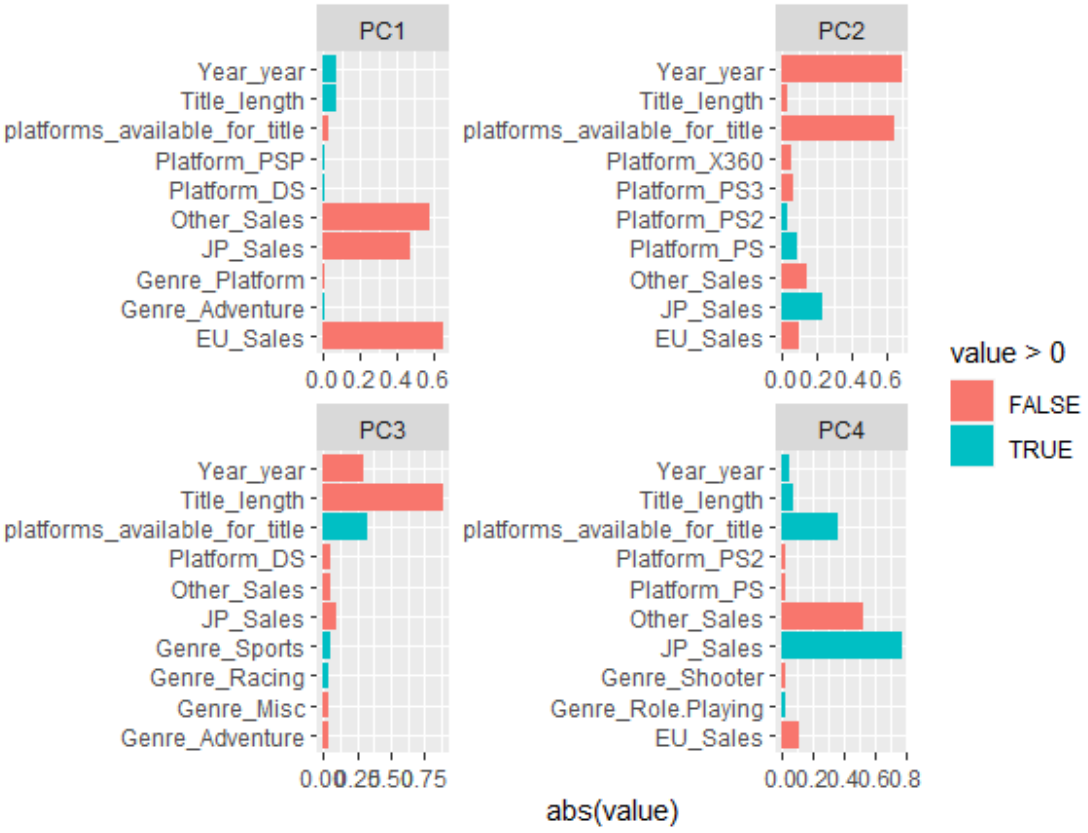
Principle component analysis, PCA, is useful when there is multi-collinearity, lots of variables, or if you want to remove noise or compress the data. It enables identification of the most, and least, important variables early on so that the variables best suited for predicting the outcome are well understood prior to pre-processing. Our selected models can handle multicollinearity and high variable count well, but PCA offers an opportunity to determine variable importance.

Two PCAs were computed, the first with all the additional predictor variables, the second with just the variables provided in the CSV:

```
# Start with the initial variables
  # Remove Name, only useful as a factor if we categorise by franchise
  # Remove publisher as we can't predict with it
  # Total sales has no predictive information
 vgsales_df_vars <- select(vgsales_df, c(Platform, Genre, NA_Sales, EU_Sales,
                                         JP_Sales, Other_Sales, Title_length,
                                         platforms_available_for_title,
                                         Year))

# Recipe for PCA
recipe_PCA <- recipe(NA_Sales ~., data = vgsales_df_vars ) %>%
```

```
  step_log(JP_Sales, EU_Sales, Other_Sales, offset = 1) %>% #Account for skew
  step_log(all_outcomes(), offset = 1, skip = TRUE) %>%
  step_date(Year, features = "year") %>% # creating time predictors
  step_rm(Year) %>%
  step_normalize(all_numeric(), -all_outcomes()) %>%
  step_dummy(Platform, Genre)%>%
  step_pca(all_predictors())

# Preping
prepped_PCA <- recipe_PCA %>%
  prep()
# View loadings
tidy(prepped_PCA) # All steps completed

tidy(prepped_PCA, 7) %>%
  dim() # 2209 = 47*47 (all possible PCs)

# Viewing relationship
tidy(prepped_PCA, 7 ) %>%
  filter( component %in% c("PC1", "PC2", "PC3", "PC4") ) %>%
  group_by( component ) %>%
  top_n(10, abs(value) ) %>%
  ungroup() %>%
  ggplot( aes( x = abs(value), y = terms, fill = value > 0 ) ) +
  geom_col(show.legend = TRUE) +
  facet_wrap( ~ component, scales = "free") +
  ggtitle(str_wrap("Figure 13:Importance of variables in the first four princ
ipal components", 45))+
  ylab(NULL) # We do not need the y axis label.
```

Figure 13:Importance of variables in the first four principal components



According to the first PCA:

- PC1 showed EU_sales and Other_sales contributed strongly, along with JP-Sales to a lesser degree (negative loadings)
- PC2 showed Year and platforms_available_for_title contributed strongly (negative loadings)
- PC3 convincingly showed Title_length was a strong contributor, as well as Year to a lesser degree (negative loadings)
- PC4 showed JP_Sales contributed strongly (positive loadings), and Other_Sales to a lesser degree (positive loadings).
- The dummy variables are difficult to comment on, but what we can say is no single Platform or genre on its own provided enough information to represent the data independently (values were distributed).

```
# Juice recipe to get principle components
juiced_pca <- juice( prepped_PCA )
juiced_pca %>%
  head()
```

```
## # A tibble: 6 x 6
##   NA_Sales   PC1   PC2     PC3    PC4    PC5
##      <dbl> <dbl> <dbl>   <dbl>  <dbl>  <dbl>
## 1     3.74 -38.7 -3.85 -2.54   -12.3  -0.977
## 2     3.40 -12.7  5.40 -0.0463  10.5   1.04
## 3     2.77 -27.8 -1.78 -2.13    -5.05  0.611
## 4     2.77 -24.8 -1.22 -2.12    -2.61  1.14
## 5     2.48 -23.8  3.28 -2.36     6.88  1.83
## 6     3.18 -10.4  4.49  0.787    8.60  1.36
```

```
# Plot PC1 and PC2 for each subject
juiced_pca %>%
  ggplot( aes( x = PC1, y = PC2, colour= NA_Sales)) +
  geom_point() +
  scale_colour_distiller()+
  ggtitle(str_wrap("Figure 14: Principal component 1 loadings mapped agains
t principal component 2 loadings with the additional variables, coloured by
the value of NA_Sales", 45))
```

Figure 14: Principal component 1 loadings mapped against principal component 2 loadings with the additional variables, coloured by the value of NA_Sales

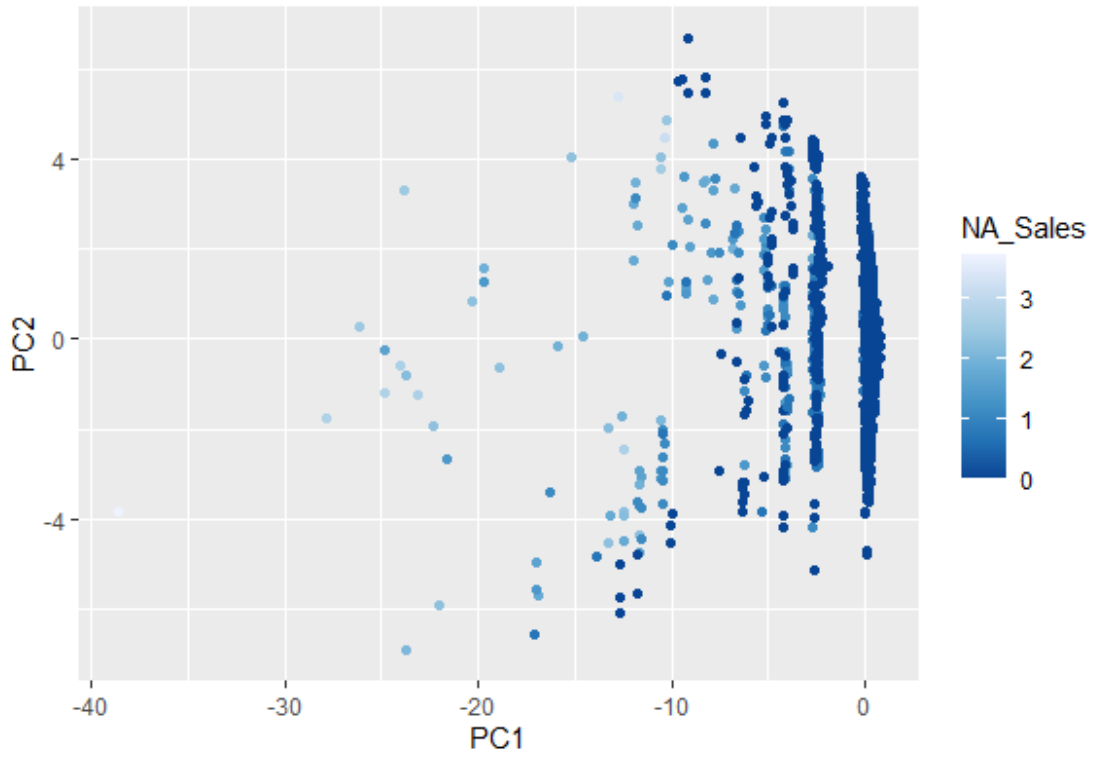
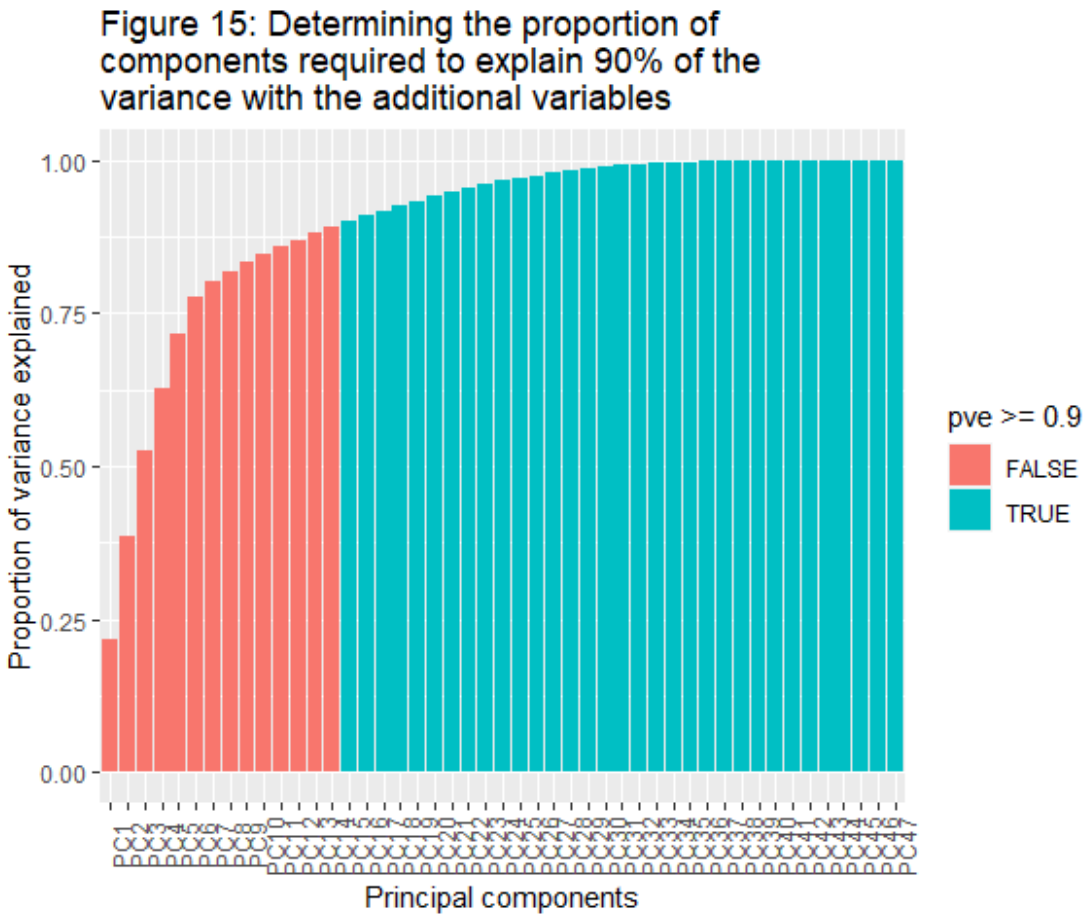


Figure 14 relates to figure 13. For PC1, loadings further to the left indicate above average values (due to normalization and negative loadings) of the sales figures, while for PC2 lower loading values indicate above average values of Year and platforms_available_for_title.

```
# Determining number of dimensions
sdev <- prepped_PCA$steps[[7]]$res$sdev
ve <- sdev^2 / sum(sdev^2)
ve # Total of 47

  # Getting the portion of variance explained
PC.pve <- tibble(
  pc = fct_inorder( str_c("PC", 1:47) ),
  pve = cumsum( ve ))

  # visualizing PVE for each component
PC.pve %>%
  ggplot( aes( x = pc,
               y = pve,
```

```
              fill = pve >= 0.9 ) ) + # Observing PCs that make up 90% of va
riability
  geom_col() +
  ggtitle(str_wrap("Figure 15: Determining the proportion of components requi
red to explain 90% of the variance with the additional variables", 45)) +
  ylab("Proportion of variance explained") +
  xlab("Principal components") +
  theme( axis.text.x = element_text( angle = 90 ) ) #rotate the x-axis labels
```




```
PC.pve %>%
  filter( pve >= 0.9)

  # 15 components to explain at least 90% of the variance
```

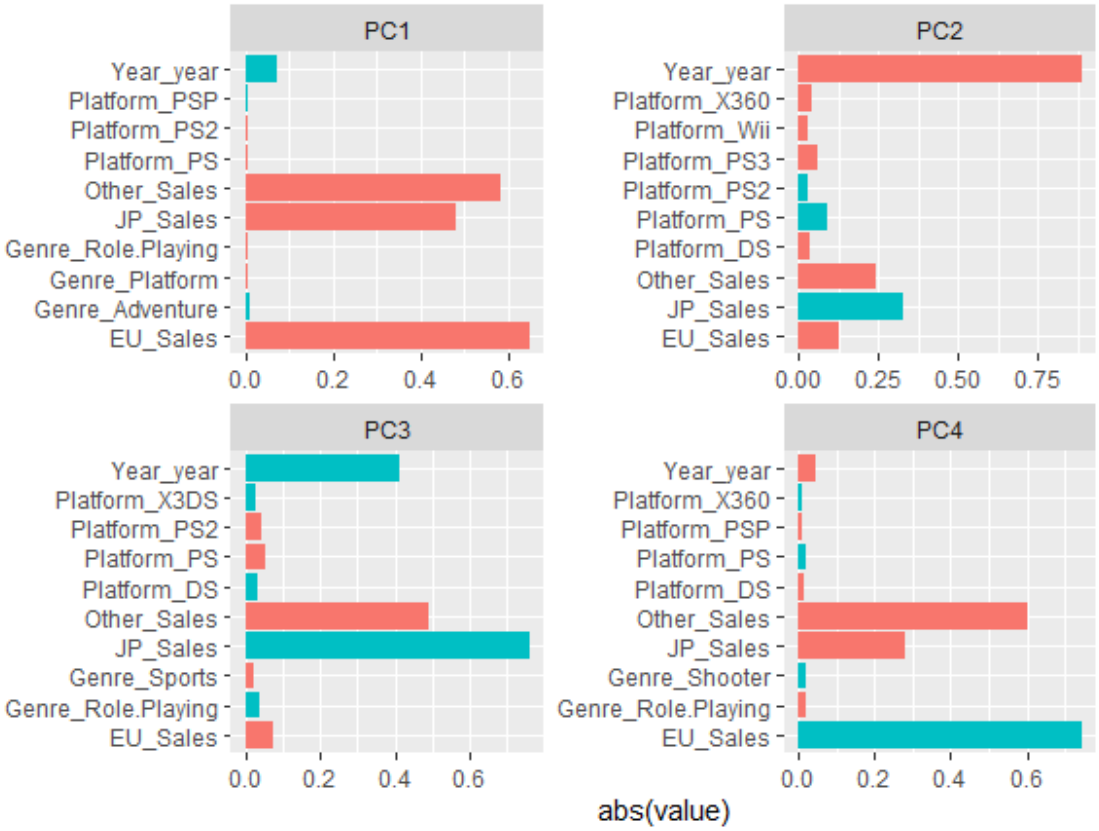
There were 47 dimensions, 15 of which could be used to explain at least 90% of the variance. This is how dimension reduction is achieved in a PCA. However, the meaning of a principal component is ambiguous. It's perhaps best thought of as the relationship between the variables that best explains the data. Each iteration is taken from a new plane, with

different key variables required to explain the positions. Each component should have a lower Root Mean Squared Error, RSME, then the last.

Comparing the above to the PCA with just the initial variables:

```
# Preparing data to suit the PCA
PCA_initial_vars <- select(vgsales_df,  c(Platform, Genre, NA_Sales, EU_Sales
,                                         JP_Sales, Other_Sales, Year ))

# Comparing against the initial variables
# Recipe for PCA
recipe_PCA_initial_vars <- recipe(NA_Sales ~., data = PCA_initial_vars ) %>%
  step_log(JP_Sales, EU_Sales, Other_Sales, offset = 1) %>% #Account for skew
  step_log(all_outcomes(), offset = 1, skip = TRUE) %>%
  step_date(Year, features = "year") %>% # creating time predictors
  step_rm(Year) %>%
  step_normalize(all_numeric(), -all_outcomes()) %>%
  step_dummy(Platform, Genre)%>%
  step_pca(all_predictors()) %>%
  prep()

# Viewing relationship
tidy(recipe_PCA_initial_vars, 7 ) %>%
  filter( component %in% c("PC1", "PC2", "PC3", "PC4") ) %>%
  group_by( component ) %>%
  top_n(10, abs(value) ) %>%
  ungroup() %>%
  ggplot( aes( x = abs(value), y = terms, fill = value > 0 ) ) +
  ggtitle(str_wrap("Figure 16: Importance of variables in the first four prin
ciple components", 45)) +
  geom_col(show.legend = F) +
  facet_wrap( ~ component, scales = "free") +
  ylab(NULL) # We do not need the y axis label.
```

Figure 16: Importance of variables in the first four principle components



When it comes to the initial predictors, we see a similar story to that observed with the added variables:

- PC1 – sales figures were contributed strongly (negative loadings)
- PC2 – Year contributed strongly (negative loadings), and JP_Sales to a much lesser degree (positive loadings)
- PC3 - JP_Sales (positive loadings), Other_Sales (negative loadings), year (positive loadings) contributed most
- PC4 - EU_Sales (positive loadings), and Other_Sales (negative loadings) were important contributors.
- No single dummy variable (genre or platform) was an important indicator of NA_Sales

```
# Juice recipe to get principle components
vgsales_juiced_initial_vars <- juice(recipe_PCA_initial_vars )
vgsales_juiced_initial_vars %>%
  head()
```

```
## # A tibble: 6 x 6
##   NA_Sales   PC1   PC2    PC3   PC4    PC5
##      <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>
## 1     3.74 -38.7 -7.49 -10.5  -8.70 -0.483
## 2     3.40 -12.8  7.22   9.26  2.08 -0.193
## 3     2.77 -27.9 -4.02  -3.71 -4.75 -0.155
## 4     2.77 -24.9 -3.00  -1.38 -2.39 -0.684
## 5     2.48 -23.9  3.38   6.99 -1.97 -0.218
## 6     3.18 -10.4  5.95   7.70  1.49 -0.160
```

```
#Plot PC1 and PC2 for each subject
vgsales_juiced_initial_vars %>%
  ggplot( aes( x = PC1, y = PC2, colour=NA_Sales ) ) +
  geom_point() +
  ggtitle(str_wrap( "Figure 17: Principal component 1 loadings mapped again
st principal component 2 loadings without the additional variables, coloure
d by the value of NA_Sales", 45 )) +
  scale_colour_distiller()
```

Figure 16 and 17 are also related. PC1 loadings further to the left indicate above average values of Other_Sales and EU_Sales, while lower PC2 loadings indiate above average values of year but below average values of JP_Sales.

```
# Determining number of dimensions
sdev <- PCA_prepped_initial_vars$steps[[5]]$res$sdev
ve <- sdev^2 / sum(sdev^2)
ve # Getting the portion of variance explained

PC.pve_initial_vars <- tibble( pc = fct_inorder( str_c("PC", 1:45) ),
                  pve = cumsum(ve))

# visualizing PVE for each component
PC.pve_initial_vars %>%
  ggplot( aes( x = pc,
               y = pve,
               fill = pve >= 0.9 ) ) +
   ggtitle(str_wrap("Figure 18: Determining the proportion of components requ
ired to explain 90% of the variance with just the initial variables", 70)) +
  ylab("Proportion of variance explained") +
  xlab("Principal components") +
  geom_col() +
  theme( axis.text.x = element_text( angle = 90 ) ) #rotate the x-axis labels
```

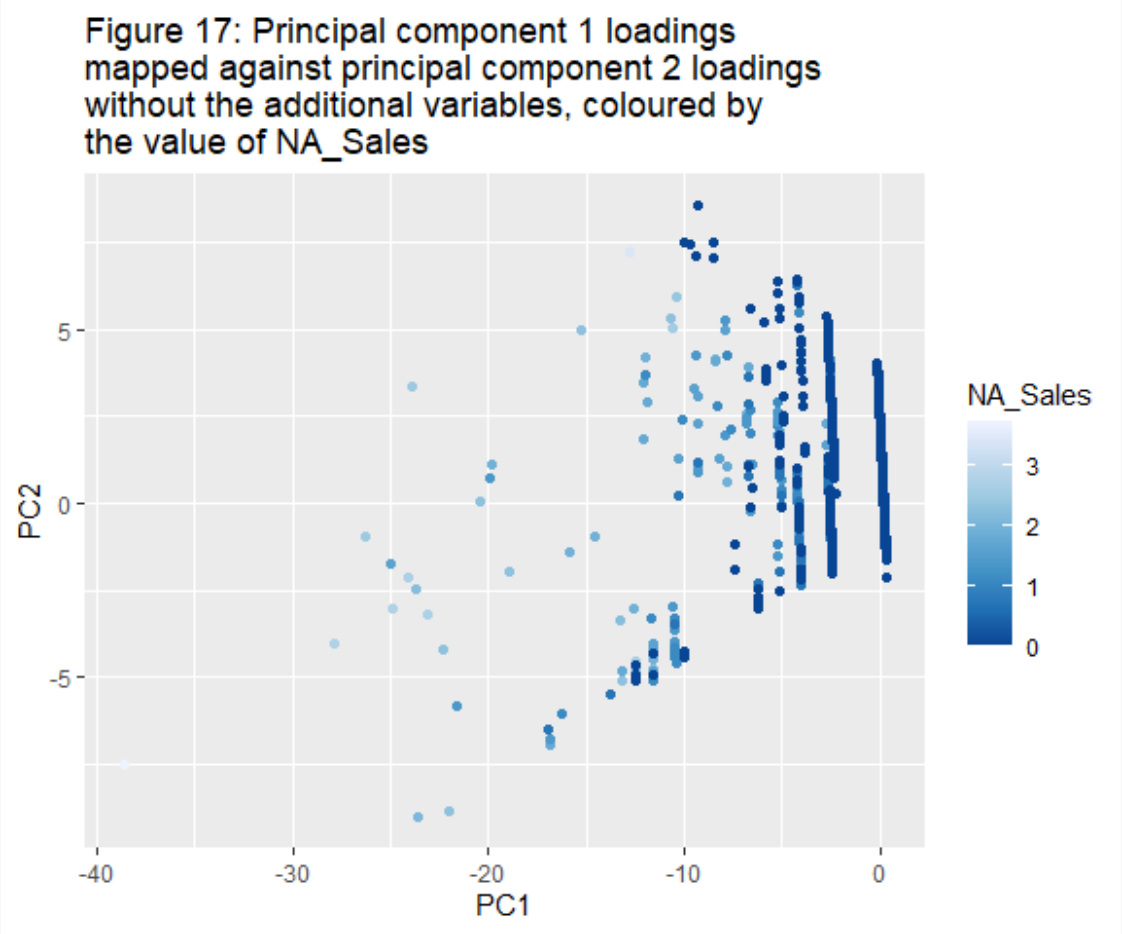
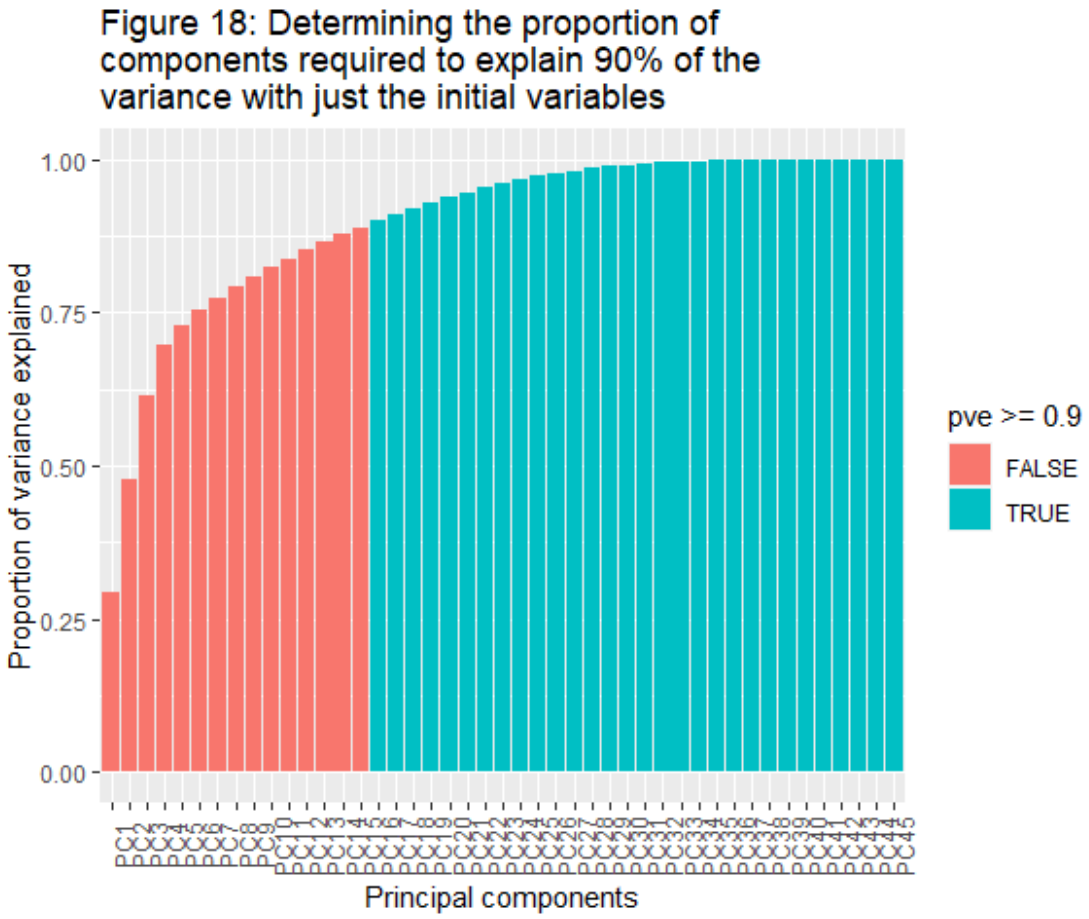


Figure 16 and 17 are also related for similar reasons as above. For low values of PC1,you wi ll have higher sales figures, and for low values of PC2 you will have higher values of Year.

```
#Determining number of dimensions
sdev <- recipe_PCA_initial_vars$steps[[7]]$res$sdev
ve <- sdev^2 / sum(sdev^2)
ve # Getting the portion of variance explained

PC.pve_initial_vars <- tibble( pc = fct_inorder( str_c("PC", 1:45) ),
                  pve = cumsum(ve))

# visualizing PVE for each component
PC.pve_initial_vars %>%
  ggplot( aes( x = pc,
               y = pve,
               fill = pve >= 0.9 ) ) +
   ggtitle(str_wrap("Figure 18: Determining the proportion of components requ
ired to explain 90% of the variance with just the initial variables", 45)) +
  ylab("Proportion of variance explained") +
```

```
  xlab("Principal components") +
  geom_col() +
  theme( axis.text.x = element_text( angle = 90 ) ) #rotate the x-axis labels
```




```
PC.pve_initial_vars %>%
  filter( pve > 0.9)  # Let's look at those explaining 90% or more

  # PC16 components to explain at least 90% of the variance
```

In this instance, there were 45 dimensions of which only 16 were needed to explain 90% of the variance. To be clear, the first PCA had 47 dimensions reduced to 15, then in the second PCA there were only 45 dimensions reduced to just 16. This implies that the added variables explained a good deal of the variance.

What are the relationships between the response variable and the categorical predictors?

First let's check the relatedness of the categorical predictors:

```
# Obtaining Categorical list
sapply(vgsales_df, class)
```

```
##                          Name                      Platform
##                      "factor"                      "factor"
##                          Year                         Genre
##                        "Date"                      "factor"
##                     Publisher                      NA_Sales
##                      "factor"                     "integer"
##                      EU_Sales                      JP_Sales
##                     "integer"                     "integer"
##                   Other_Sales                   Total_sales
##                     "integer"                     "integer"
##                  Title_length       platform_sales_for_year
##                     "integer"                     "integer"
## platforms_available_for_title
##                     "integer"
```

```
  # Only platform and genre

# Chi-squared test
chisq.test(vgsales_df$Platform, vgsales_df$Genre, simulate.p.value = TRUE)
```

```
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
## data:  vgsales_df$Platform and vgsales_df$Genre
## X-squared = 5912.8, df = NA, p-value = 0.0004998
```

As the p-value was lower than 0.05, the null hypothesis was rejected meaning the two variables were found to be dependent. However, Lasso and Random Forest models can handle multi-collinearity rather well.

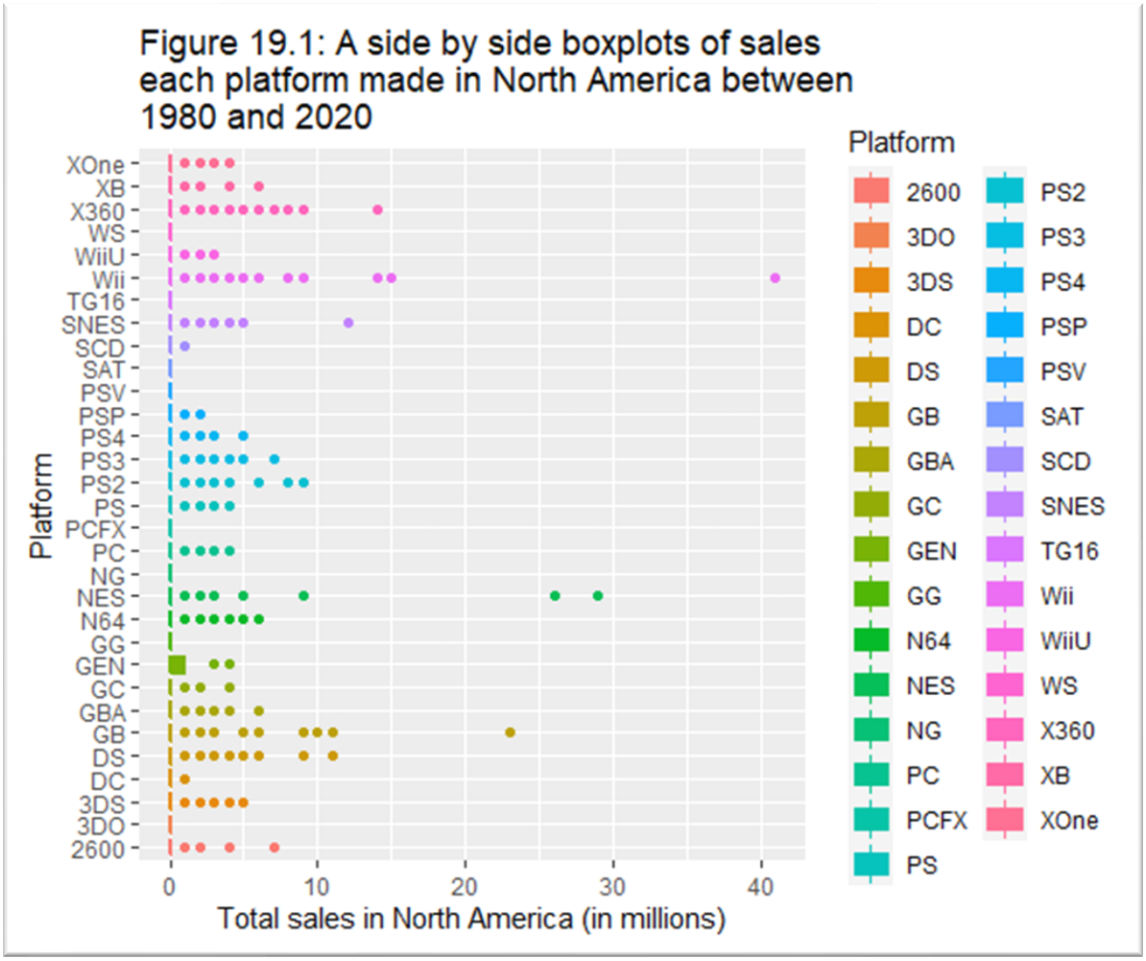
The relationships the factor variables (predictors) had with NA_Sales (response variable) were individually scrutinized below:

```
# Finding the number of sales in North America on each Platform
    # Thought there was rounding down problem, but persists even after NA_Sal
es is multiplied by a million. Must be fine.
NA_vs_Platform <- vgsales_df %>%
  group_by(Platform, NA_Sales) %>%
  tally(round(NA_Sales,3)) %>%
  summarise(NA_total = sum(NA_Sales*n)) %>%
  arrange(desc(NA_total))
knitr::kable(NA_vs_Platform, caption = "Table 6: The amount of total sales ma
de in North America from games on each Platform.")
```

Table 6: The number of total sales made in North America from games on each Platform. It contrasts the total platform sales above and identifies a difference in adoption curves regionally or poor data acquisition.

Platform	NA_total	Platform	NA_total	Platform	NA_total
Wii	2979	2600	111	SCD	1
NES	1670	3DS	104	3DO	0
X360	1208	PC	78	GG	0
GB	939	PS4	78	NG	0
PS2	543	XB	76	PCFX	0
DS	497	GC	70	PSV	0
PS3	319	XOne	67	SAT	0
PS	250	GEN	32	TG16	0
SNES	221	WiiU	28	WS	0
N64	167	PSP	16		
GBA	119	DC	4		

```
# Side by side boxplots
ggplot(vgsales_df, aes(y = Platform, x = NA_Sales, fill = Platform)) +
  geom_boxplot(aes(colour = Platform)) +
  ggtitle(str_wrap("Figure 19.1: A side by side boxplots of sales each platfo
rm made in North America between 1980 and 2020", 70)) +
  xlab("Total sales in North America (in millions)")
```



Platform

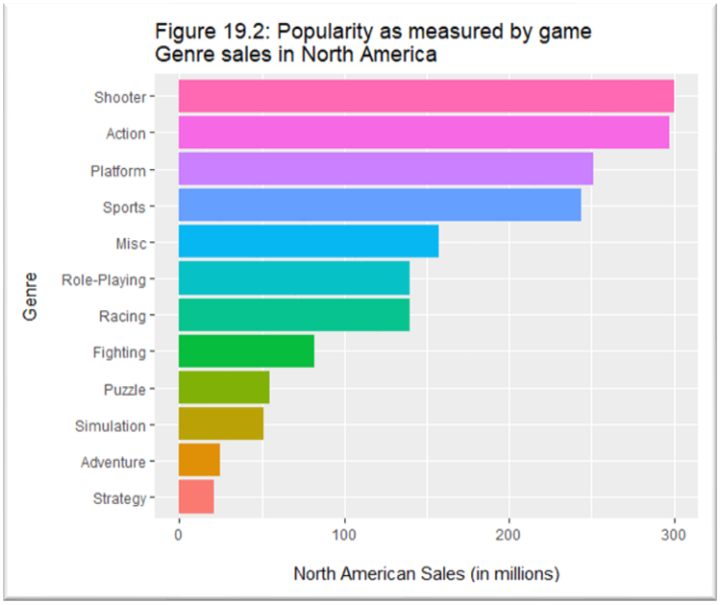
Figure 19.1 shows how poorly distributed our data is. The boxplots were barely visible (this was still the case when the x axis limits where set from 0 to 1). The data set is likely just incomplete, but it implies we are lacking the data required to be truly confident in the findings. It is more likely that the 'outliers' are samples of a population sitting on a more dispersed domain. This is just a suspicion, but it seems to complement other assertions made about the data rather well.

```
# Finding the amount of sales in North America in each Genre
NA_vs_Genre <- vgsales_df %>%
  group_by(Genre, NA_Sales) %>%
  tally() %>%
  summarise(NthA_total = sum(NA_Sales*n)) %>%
  arrange(desc(NthA_total))
knitr::kable(NA_vs_Genre)
```

Table 7: The number of total sales made in North America from games of a given Genre. It seems North America appreciates a fast-paced, and doesn't tend to choose a mentally challenging genre.

Genre	NthA_total
Shooter	300
Action	297
Platform	251
Sports	244
Misc	157
Racing	140
Role-Playing	140
Fighting	82
Puzzle	55
Simulation	51
Adventure	25
Strategy	21

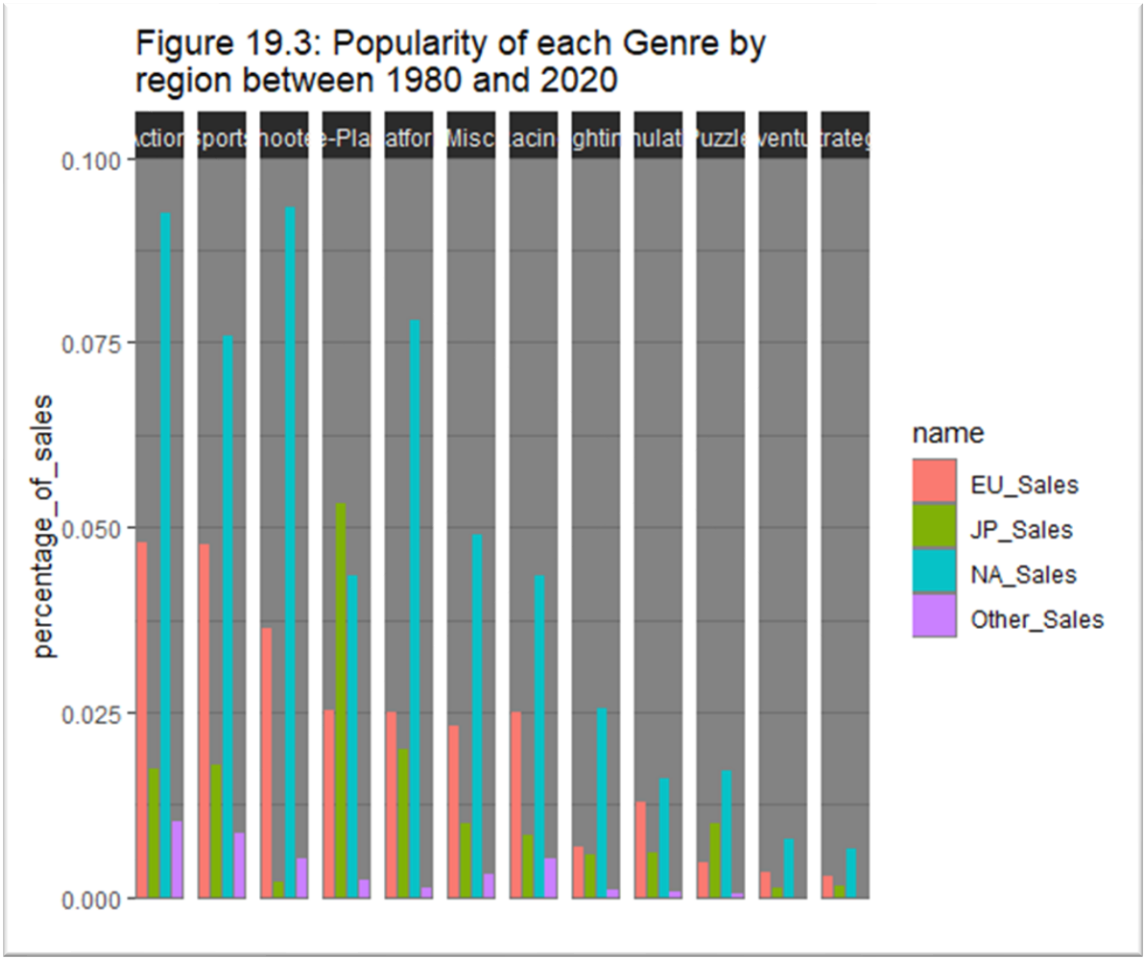
```
ggplot(NA_vs_Genre, aes(y = Genre, x = NthA_total, fill = Genre)) +
  geom_bar(stat="identity", show.legend = F) +
  ggtitle(str_wrap(" Figure 19.2: Popularity as measured by game Genre sales
in North America", 70)) +
  xlab("\nNorth American Sales (in millions)")+
  ylab("Genre\n")
```



Genre

Figure 19.2 gives a sense of genre popularity in North America over the period. If the sampled data was proportionally representative, then there was no issue in including Genre as a predictor.

```
# Finding the proportion of sales for each Genre in each place
NA_vs_Genre <- vgsales_df %>%
  pivot_longer(cols = NA_Sales:Other_Sales) %>%
  mutate(percentage_of_sales = (value)/(sum(value))) %>%
  select(Genre, name, percentage_of_sales)
NA_vs_Genre <- aggregate(.~ Genre + name, data = NA_vs_Genre, FUN = sum)

# Creating levels based on percentage of sales
Genre_popularity <- select(NA_vs_Genre, Genre, percentage_of_sales)
Genre_popularity <- aggregate(.~ Genre, data = Genre_popularity, FUN = sum)
Genre_popularity <- arrange(Genre_popularity, desc(percentage_of_sales))
NA_vs_Genre$Genre <- factor(NA_vs_Genre$Genre, levels = c("Action", "Sports",
"Shooter", "Role-Playing", "Platform", "Misc", "Racing", "Fighting", "Simulat
ion", "Puzzle", "Adventure", "Strategy"))

# Plotting sales of each genre for each region
ggplot(NA_vs_Genre, aes(x = name, y = percentage_of_sales, fill=name)) +
  geom_col(show.legend = T) +
  facet_grid(~Genre)+
  scale_x_discrete(labels = NULL, breaks = NULL) +
  ggtitle(str_wrap(" Figure 19.3: Popularity of each Genre by region between
1980 and 2020", 70)) +
  labs(x = "") +
  scale_y_continuous(limits=c(0,0.1),expand=c(0,0)) +
  theme_dark()
```

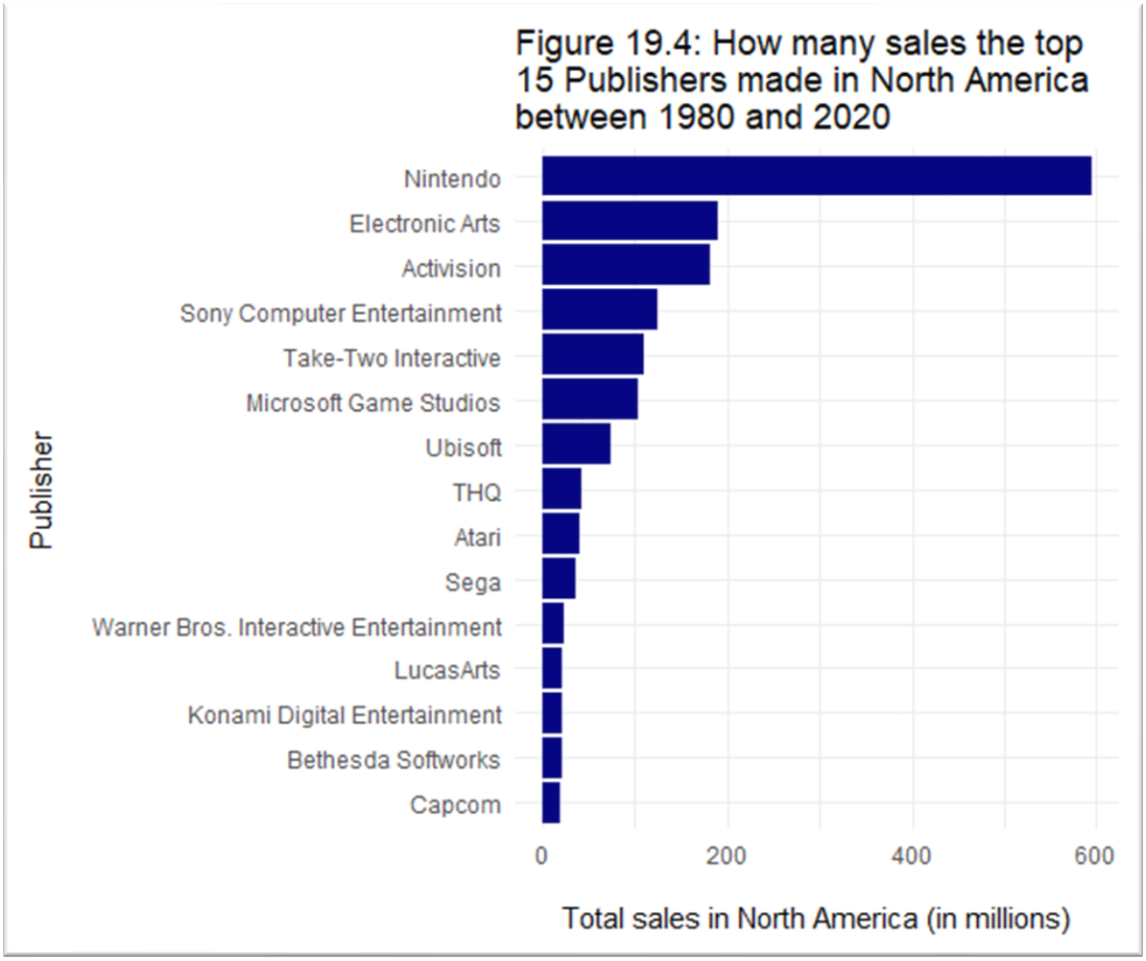



```
# Large proportion of sales are from North America, may need to address this
# Role playing games don't appear to capture the NA market as well as some ot
her Genres do.
```

Figure 19.3 enables comparisons of genre preferences to be drawn between locations. It was this plot that made the recipe step step_other() attractive; binding uncommon values under an 'Other' value.

```
# Finding the amount of sales in North America from each Publisher
NA_vs_Publisher <- vgsales_df %>%
  group_by(Publisher, NA_Sales) %>%
  tally() %>%
  summarise(NthA_total = sum(NA_Sales*n)) %>%
  mutate(Publisher = fct_reorder(Publisher, NthA_total ))
ggplot(filter(NA_vs_Publisher, NthA_total > 16), aes(y = Publisher, x = NthA_
total)) +
  geom_bar( fill = "navy", stat="identity", show.legend = F) +
  theme_minimal()+
  ggtitle(str_wrap(" Figure 19.4: How many sales the top 15 Publishers made i
n North America between 1980 and 2020", 45)) +
```

```
  xlab("\nTotal sales in North America (in millions)")+
  ylab("Publisher\n")
```

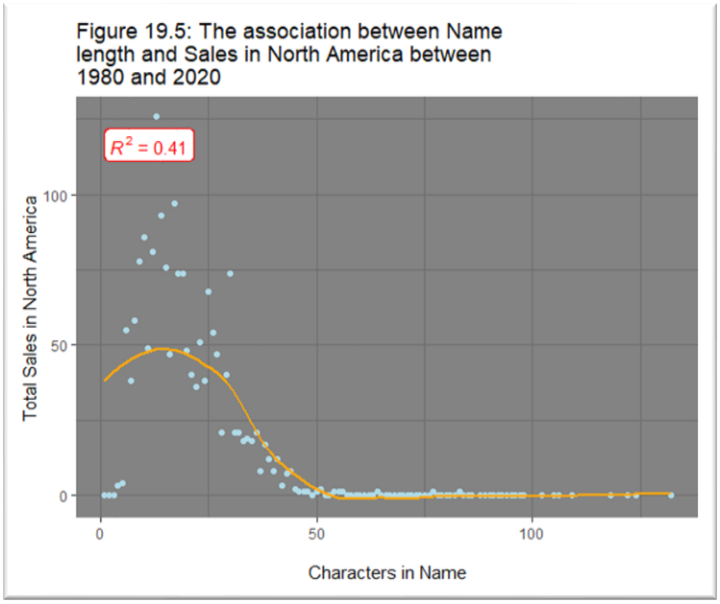


Publisher

For completeness's sake, figure 19.4 gives a sense of just how dominant some of the publishers have been in North America, and how small an impact the other publishers have had in comparison.

```
# Finding the amount of sales in North America given the Title length
NA_vs_TitleLength <- vgsales_df %>%
  group_by(Title_length, NA_Sales) %>%
  tally() %>%
  summarise(NthA_total = sum(NA_Sales*n)) %>%
  #mutate(Title_length = fct_reorder(Title_length, NthA_total )) %>%
  ggplot(aes(x = as.numeric(Title_length), y = NthA_total)) +
  geom_point(colour = "lightblue") +
  geom_smooth(colour = "orange", se=FALSE)+
  theme_dark() +
  ggtitle(str_wrap("Figure 19.5: The association between Name length and Sale
s in North America between 1980 and 2020", 70)) +
```

```
  xlab("\nCharacters in Name") +
  ylab("Total Sales in North America")+
  ggpubr::stat_cor(aes(label = ..rr.label..), color = "red", geom = "label")
NA_vs_TitleLength
```

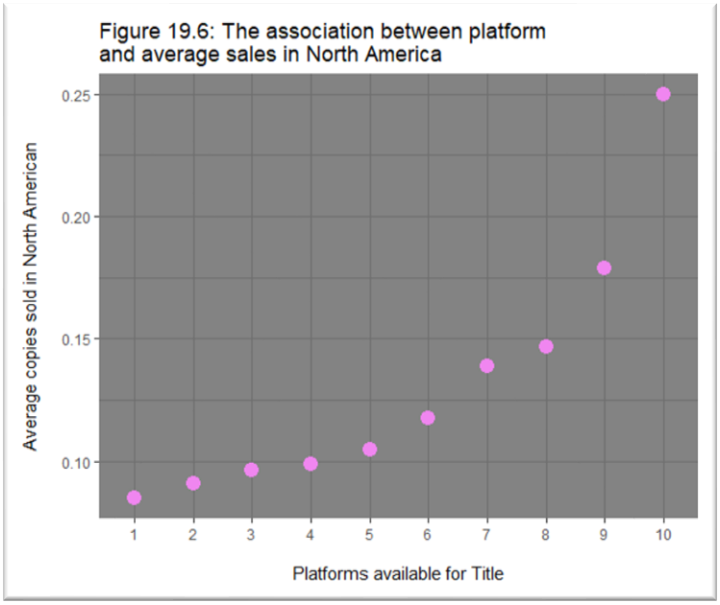


Title_length

Developers should take figure 19.5 into consideration and aim for a title length of between 12 and 20 characters. But for the purposes of predicting North American sales, a correlation appears to exist between the Title_length and how well a game sells in North America. Note: some of the values are knowingly erroneous as observed in *table 1*.

```
# Finding the amount of sales in North America based on the number of platfor
ms the game is available on. This was more difficult than it should have been
NA_vs_consoles <- vgsales_df %>%
  group_by(platforms_available_for_title, NA_Sales) %>%
  tally() %>%
  summarise(NthA_total = (sum(NA_Sales*n)/sum(n))) %>%
  mutate(platforms_available_for_title =
           fct_reorder(as.factor(platforms_available_for_title), NthA_total))
levels(NA_vs_consoles$platforms_available_for_title) <- c("1","2", "3",
                                                          "4", "5", "6",
                                                          "7", "8", "9",
                                                          "10", "11", "12")
ggplot(NA_vs_consoles, aes(x = platforms_available_for_title, y = NthA_total)
) +
  geom_point(colour="violet", size = 4) +
  theme_dark() +
  ggtitle(str_wrap("Figure 19.6: The association between platform and average
sales in North America", 70)) +
```

```
  xlab("\nPlatforms available for Title") +
  ylab("Average copies sold in North American\n")
```



platforms_available_for_title

The positive relationship hypothesized appeared to be present in the data when the right skew was account for (accomplished by summing the total number of rows with the same value of platforms_available_for_title, then dividing the total sales for those games by the number of games present, or the average sales per game for each value of platforms_available_for_title).

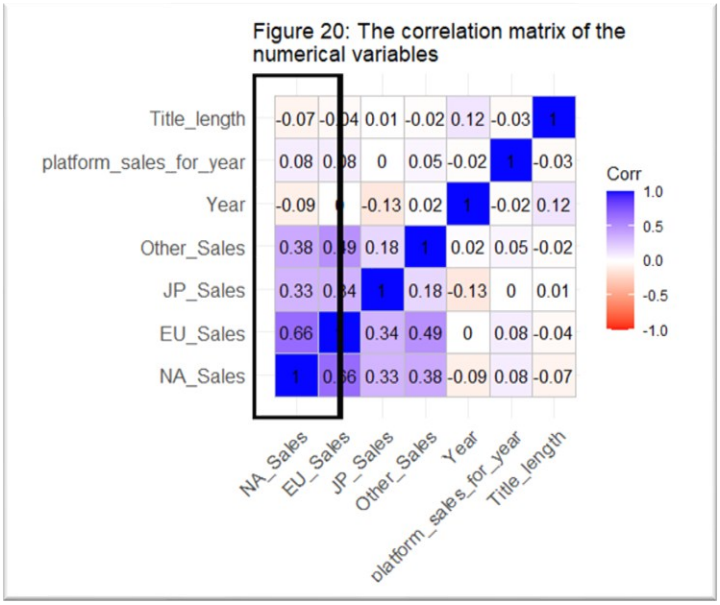
What are the relationships between the response variable and the numeric predictors?

See below how the response variable, NA_Sales, related each of the numerical predictors:

```
# Obtaining numerical variables
sapply(vgsales_df, class)
```

```
##                          Name                      Platform
##                      "factor"                      "factor"
##                          Year                         Genre
##                        "Date"                      "factor"
##                     Publisher                      NA_Sales
##                      "factor"                     "integer"
##                      EU_Sales                      JP_Sales
##                     "integer"                     "integer"
##                   Other_Sales                  Total_sales
##                     "integer"                     "integer"
##                  Title_length       platform_sales_for_year
##                     "integer"                     "integer"
## platforms_available_for_title
##                     "integer"
```

```
# Our model will process year as an integer so lets have a peak
vgsales_df2 <- vgsales_df
vgsales_df2$Year <- as.integer(vgsales_df2$Year)
# Plotting a correlation matrix for numerical variables
numeric_variables <- select(vgsales_df2,
                            c("NA_Sales", "EU_Sales", "JP_Sales", "Other_Sale
s","Year",
                              "platform_sales_for_year","Title_length")) %>%
mutate(NA_Sales = log(NA_Sales+1),
       JP_Sales = log(JP_Sales+1),
       EU_Sales = log(EU_Sales+1),
       Other_Sales = log(Other_Sales+1))

# Observing relationship
ggcorrplot(cor(numeric_variables), method = "square", colors = c("red", "whit
e", "Blue"), lab =TRUE) +
  geom_rect(aes(xmin =0, xmax=2, ymin=0, ymax= 8), size = 2, colour="Black",a
lpha = 0)+
  ggtitle(str_wrap("Figure 20: The correlation matrix of the numerical variab
les\n", 70))
```



Title_length and platform_sales_for_year didn't have the correlation observed in previous plots when comparing averages. The sales figures (log(x sales+1)) were positively related to NA_Sales (log(NA_Sales + 1)), a strong relationship in the case of EU_Sales, and weak-to-moderate for Other_Sales and JP_Sales. This is most likely due to differing cultural tastes and preferences, as well as population sizes as previously discussed. The Year variable had practically no correlation one way or the other with NA_Sales, as a result it is likely to have a large penalty in the lasso model.

Data Selection and Pre-processing

Here are the variables provided for the prediction of Sales in North America for the game ' The Fatal Empire':

```
Genre_levs <- c(levels(vgsales_df$Genre))
Platform_levs <- c(levels(vgsales_df$Platform))

prediction_data <-  tibble(Genre = factor("Role-Playing", levels = Genre_levs
),
                           Platform = factor('PS4', levels = Platform_levs),
                           Title_length = nchar('The Fatal Empire'),
                           JP_Sales = 2.58,
                           EU_Sales = 0.53,
                           Other_Sales = 0.1,
                           Year = as.Date('2022', format = "%Y"),
                           platforms_available_for_title = 1)
head(prediction_data) %>%
  knitr::kable(caption = "Table 8: The predictions set for 'The Fatal Empire'
")
```

Table 8: The prediction set for 'The Fatal Empire'

Genre	Platform	Title_length	JP_Sales	EU_Sales	Other_Sales	Year	platforms_available_for_title
Role-Playing	PS4	16	2.58	0.53	0.1	2022	1

Below the final data set was compiled including all the variables that would be Pre-processed and taken into the modelling phase. Name has been removed as it is more or less an ID key and may lead to over-fitting. Additionally, seeds were set for reproducibility, the data was split into testing and training sets for validation purposes:

```
# Selecting variables from initial set and from modified set
selected_vars_initial <- select(vgsales_df, c(Genre, Platform, Year, NA_Sales
,
                                              JP_Sales, EU_Sales, Other_Sales
))

selected_vars_final <- select(vgsales_df, c(Genre, Platform, Year, Title_leng
th,
                                            NA_Sales, JP_Sales, EU_Sales,
                                            Other_Sales,
                                            platforms_available_for_title))

# Making reproducible testing and training data
set.seed(2022)
index <- sample(1:nrow(vgsales_df))
repro_vgsales_df <-vgsales_df[index, ]
repro_vgsales_df

vgsales_df_split <- initial_split(vgsales_df, strata = NA_Sales)
vgsales_df_training <- training(vgsales_df_split)
```

```
# Creating an initial training set by selecting variables:
initial_training <- select(vgsales_df_training, c(Genre, Platform, NA_Sales,
Year,
                                                  JP_Sales, EU_Sales, Other_S
ales))
nrow(initial_training) # 12447 rows
```

```
## [1] 12447
```

```
final_vars_training <- select(vgsales_df_training, c(Genre, Platform, Year,
                                                     Title_length, NA_Sales,
JP_Sales,
                                                     EU_Sales, Other_Sales,
                                                     platforms_available_for_
title))
nrow(final_vars_training) # still 12447 rows
```

```
## [1] 12447
```

```
# Creating resamples
initial_folds_rf <- bootstraps( data = initial_training, times = 10)
final_vars_folds_rf <- bootstraps( data = final_vars_training, times = 10)

# Needed to use the same bootstraps, fixing without altering variable names
initial_folds <- initial_folds_rf
final_vars_folds <- final_vars_folds_rf

#Creating testing sets from the split data
vgsales_df_testing <- testing(vgsales_df_split)
initial_testing <- select(vgsales_df_testing, c(Genre, Platform, Year, NA_Sal
es,
                                                  JP_Sales, EU_Sales, Other_S
ales))
nrow(initial_testing) # still 4150 rows
```

```
## [1] 4150
```

```
final_vars_testing <- select(vgsales_df_testing, c(Genre, Platform, Year, Tit
le_length,
                                                   NA_Sales, JP_Sales, EU_Sal
es,
                                                   Other_Sales,
                                                   platforms_available_for_ti
tle))
nrow(final_vars_testing) # still 4150 rows
```

```
## [1] 4150
```

```
knitr::kable(head(final_vars_training), caption = "Table 9: The training set
with all predictors")
```

Table 9: The training set with all predictors

	Genre	Platform	Year	Title_length	NA_Sales	JP_Sales	EU_Sales	Other_Sales	platforms_available_for_title
1	Sports	Wii	2006-04-22	10	41	3	29	8	1
3	Racing	Wii	2008-04-22	14	15	3	12	3	1
4	Sports	Wii	2009-04-22	17	15	3	11	2	1
5	Role-Playing	GB	1996-04-22	24	11	10	8	1	1
8	Misc	Wii	2006-04-22	8	14	2	9	2	1
9	Platform	Wii	2009-04-22	25	14	4	7	2	1

```
knitr::kable(head(final_vars_testing), caption = "Table 10: The testing set with all predictors")
```

Table 10: The testing set with all predictors

	Genre	Platform	Year	Title_length	NA_Sales	JP_Sales	EU_Sales	Other_Sales	platforms_available_for_title
2	Platform	NES	1985-04-22	17	29	6	3	0	2
6	Puzzle	GB	1989-04-22	6	23	4	2	0	2
7	Platform	DS	2006-04-22	21	11	6	9	2	1
10	Shooter	NES	1984-04-22	9	26	0	0	0	1
14	Sports	Wii	2007-04-22	7	8	3	8	2	1
21	Role-Playing	DS	2006-04-22	29	6	6	4	1	1

A recipe was created using the 'tidymodels' package:

- The formula was set to use all predictors available in the training set above to predict NA_Sales. All these steps were essential, there were normalization, dummy, and transformation steps:

1. Step_log() was used to transform the sales figure predictors with an offset of 1.
2. Step_log() was applied separately to NA_Sales with an offset of 1 so that a skip argument could be implemented.
3. Step_date() was used to convert the Year into a numerical variable
4. Step_rm() was used to remove the remaining date class variable Year
5. Step_normalize() was used to make different scales more comparable with z-scores
6. Step_dummy() was used to create binary values for every factor value in the training data but one (which itself is indicated when all other dummy variables are equal to zero).

This recipe was applied to both the initial data, and the data with the additional variables:

```
# Recipe for initial variables
vgsales_recipe_initial <- recipe(NA_Sales ~., data = initial_training) %>%
  step_log(JP_Sales, EU_Sales, Other_Sales, offset = 1) %>% #Account for skew
  step_log(all_outcomes(), offset = 1, skip = TRUE) %>%
  step_date(Year, features = "year") %>% # creating time predictors
  step_rm(Year) %>%
  #step_num2factor(Year_year, ? consider year as a factor instead
  step_normalize(all_numeric(), -all_outcomes()) %>%
  step_dummy(all_nominal_predictors()) %>%
  #step_zv() %>% # not needed as I've set features in step_date
  #step_corr(all_numerical()) # largest is between eu and other, 0.58
  #step_other()
  prep()
# Interaction steps may benefit the lasso, but the random forrest can manage
on its own.
# Step_BoxCox seemed unhelpful


  # Preprocessing the training set
initial_training_prepro <- vgsales_recipe_initial %>%
  juice()           # Using clean_training data from recipe
initial_training_prepro
```

```
## # A tibble: 12,447 x 46
##    JP_Sales EU_Sales Other_Sales NA_Sales Year_year Genre_Adventure
##       <dbl>    <dbl>       <dbl>    <dbl>     <dbl>           <dbl>
##  1    11.4      20.2     33.6        3.74   -0.0632               0
##  2    11.4      15.2     21.2        2.77    0.277                0
##  3    11.4      14.7     16.8        2.77    0.446                0
##  4    19.8      13.0     10.5        2.48   -1.76                 0
##  5     8.99     13.6     16.8        2.71   -0.0632               0
##  6    13.2      12.3     16.8        2.71    0.446                0
##  7     5.63     14.7     16.8        2.30   -0.233                0
##  8    13.2      12.3     10.5        2.30   -0.233                0
##  9    17.1      11.5     -0.0650     2.30   -1.25                 0
## 10     8.99     13.0     10.5        2.30    0.446                0
## # ... with 12,437 more rows, and 40 more variables: 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>,
## #   Platform_X3DO <dbl>, Platform_X3DS <dbl>, Platform_DC <dbl>,
## #   Platform_DS <dbl>, Platform_GB <dbl>, Platform_GBA <dbl>,
## #   Platform_GC <dbl>, Platform_GEN <dbl>, Platform_GG <dbl>, ...
```

```
  # Preprocessing the testing set
initial_test_prepro <- vgsales_recipe_initial %>%
  bake(initial_testing)
initial_test_prepro
```

```
## # A tibble: 4,150 x 46
##    JP_Sales EU_Sales Other_Sales NA_Sales Year_year Genre_Adventure
##       <dbl>    <dbl>       <dbl>    <int>     <dbl>           <dbl>
##  1   16.0      8.13      -0.0650       29   -3.63                0
##  2   13.2      6.41      -0.0650       23   -2.95                0
##  3   16.0     13.6       16.8          11   -0.0632              0
##  4   -0.113   -0.158     -0.0650       26   -3.80                0
##  5   11.4     13.0       16.8           8    0.107               0
##  6   16.0      9.46      10.5           6   -0.0632              0
##  7   13.2      6.41      -0.0650       10   -2.95                0
##  8   14.7      8.13      -0.0650        6   -0.743               0
##  9   14.7      8.13      -0.0650        5    0.616               0
## 10    5.63    10.6       10.5           6   -0.913               0
## # ... with 4,140 more rows, and 40 more variables: 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>,
## #   Platform_X3DO <dbl>, Platform_X3DS <dbl>, Platform_DC <dbl>,
## #   Platform_DS <dbl>, Platform_GB <dbl>, Platform_GBA <dbl>,
## #   Platform_GC <dbl>, Platform_GEN <dbl>, Platform_GG <dbl>, ...
```

```
#recipe for initial and additional variables
vgsales_recipe_final <- recipe(NA_Sales ~ ., data = final_vars_training) %>%
  step_log(JP_Sales, EU_Sales, Other_Sales, offset = 1) %>% #Account for skew
  step_log(all_outcomes(), offset = 1, skip = TRUE) %>%
  step_date(Year, features = "year") %>% # creating time predictors
  step_rm(Year) %>%
  step_normalize(all_numeric(), -all_outcomes()) %>%
  step_dummy(all_nominal_predictors()) %>%
  #step_zv() %>% # not needed as I've set features in step_date
  #step_corr(all_numerical()) # largest is between eu and other, 0.58
  #step_other()
  prep()

  # Preprocessing the training set
final_training_prepro <- vgsales_recipe_final %>%
  juice()           # Using clean_training data from recipe
final_training_prepro
```

```
## # A tibble: 12,447 x 48
##    Title_length JP_Sales EU_Sales Other_Sales platforms_available_for_~ NA_Sales
##           <dbl>    <dbl>    <dbl>       <dbl>                    <dbl>    <dbl>
##  1   -1.09         11.4      20.2     33.6                      -0.706     3.74
##  2   -0.779        11.4      15.2     21.2                      -0.706     2.77
##  3   -0.545        11.4      14.7     16.8                      -0.706     2.77
##  4   -0.0000250    19.8      13.0     10.5                      -0.706     2.48
##  5   -1.25          8.99     13.6     16.8                      -0.706     2.71
##  6    0.0778       13.2      12.3     16.8                      -0.706     2.71
##  7   -1.09          5.63     14.7     16.8                      -0.706     2.30
##  8   -0.857        13.2      12.3     10.5                      -0.706     2.30
##  9    0.234        17.1      11.5     -0.0650                   -0.706     2.30
```

```
## 10   -0.934         8.99     13.0     10.5                        -0.706     2.30
## # ... with 12,437 more rows, and 42 more variables: Year_year <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>, Platform_X3DO <dbl>,
## #   Platform_X3DS <dbl>, Platform_DC <dbl>, Platform_DS <dbl>,
## #   Platform_GB <dbl>, Platform_GBA <dbl>, Platform_GC <dbl>, ...
```

```
  # Preprocessing the testing set
final_test_prepro <- vgsales_recipe_final %>%
  bake(final_vars_testing)
final_test_prepro
```

```
## # A tibble: 4,150 x 48
##    Title_length JP_Sales EU_Sales Other_Sales platforms_available_for_~ NA_Sales
##           <dbl>    <dbl>    <dbl>       <dbl>                    <dbl>    <int>
##  1       -0.545   16.0      8.13      -0.0650                  -0.0788       29
##  2       -1.40    13.2      6.41      -0.0650                  -0.0788       23
##  3       -0.234   16.0     13.6       16.8                     -0.706        11
##  4       -1.17    -0.113   -0.158     -0.0650                  -0.706        26
##  5       -1.32    11.4     13.0       16.8                     -0.706         8
##  6        0.389   16.0      9.46      10.5                     -0.706         6
##  7       -0.623   13.2      6.41      -0.0650                  -0.706        10
##  8        0.389   14.7      8.13      -0.0650                  -0.706         6
##  9        0.234   14.7      8.13      -0.0650                  -0.706         5
## 10       -0.156    5.63    10.6       10.5                     -0.706         6
## # ... with 4,140 more rows, and 42 more variables: Year_year <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>, Platform_X3DO <dbl>,
## #   Platform_X3DS <dbl>, Platform_DC <dbl>, Platform_DS <dbl>,
## #   Platform_GB <dbl>, Platform_GBA <dbl>, Platform_GC <dbl>, ...
```

```
# Preprocessing predictor set
initial_prediction_set <- vgsales_recipe_initial %>%
  bake(prediction_data)

final_prediction_set <- vgsales_recipe_final %>%
  bake(prediction_data)
```

The lasso regression model was set to tune the penalty parameter in order to find the value that produced the lowest RSME when comparing fitted values to true values from the testing set.

Dylan Rohan - a1844790

Model fitting and Model Evaluation

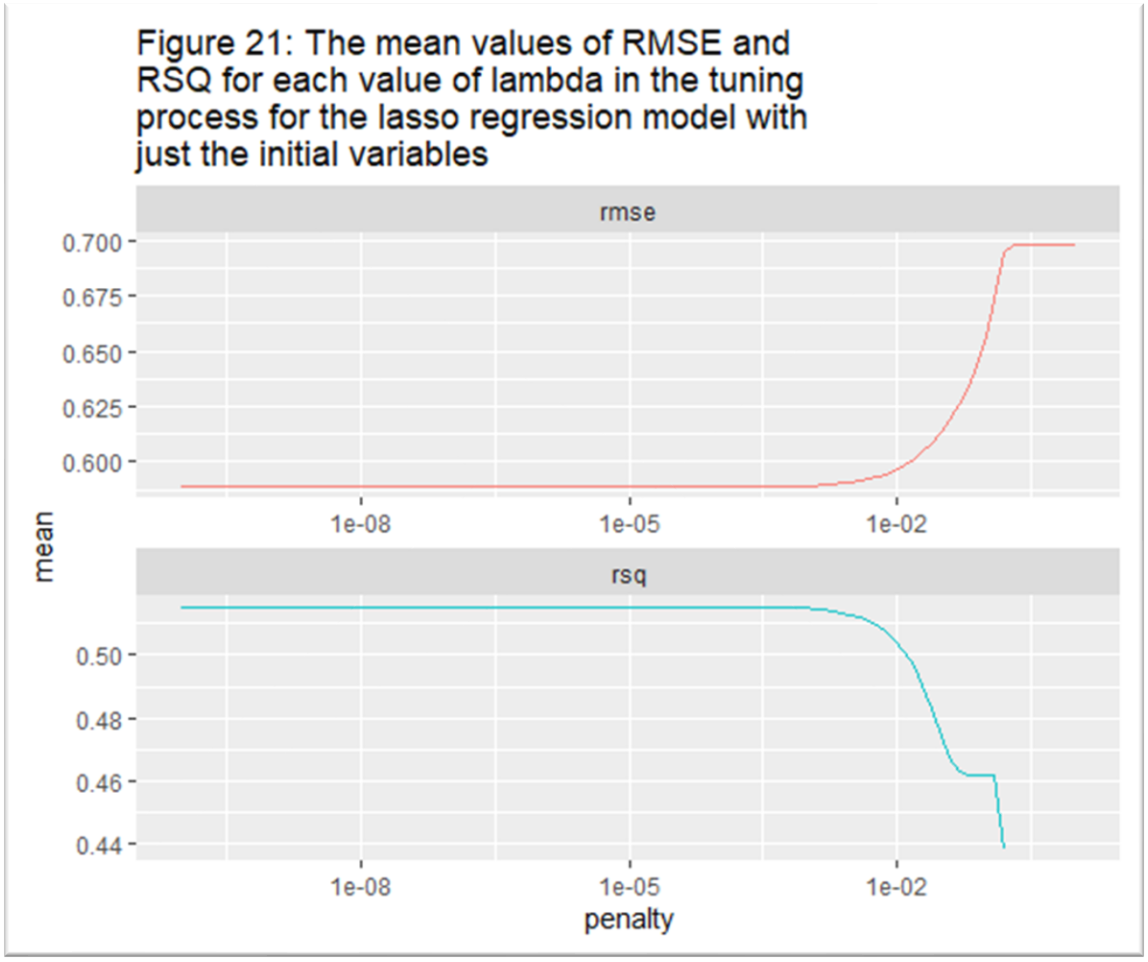
Lasso regression model

The general idea in model fitting is to create a mathematical relation between predictors. To minimize the errors and improve the accuracy, penalty values can be applied to each predictor term. Here the model was supplied with a grid of penalty values, or 'lambda' values. Every value was attempted with resampling and the value that produced the lowest RSME was chosen. RSME was considered the best metric for this assessment because it is a measure of how well a model can make a prediction. That being said the r-squared value offers an answer to 'what is the probability the model will make a correct prediction?' and should be taken into consideration when comparing models as well. It should be noted that the Lasso model has the ability to silence variable terms by shrinking them to zero.

The tidymodels linear_reg() function was used to create a regression model using the glmnet engine. Lasso was selected by setting mixture to one, and the optimal lambda value was obtained by tuning the penalty term. Two models were created, one with just the initial variables, and one with the additional variables:

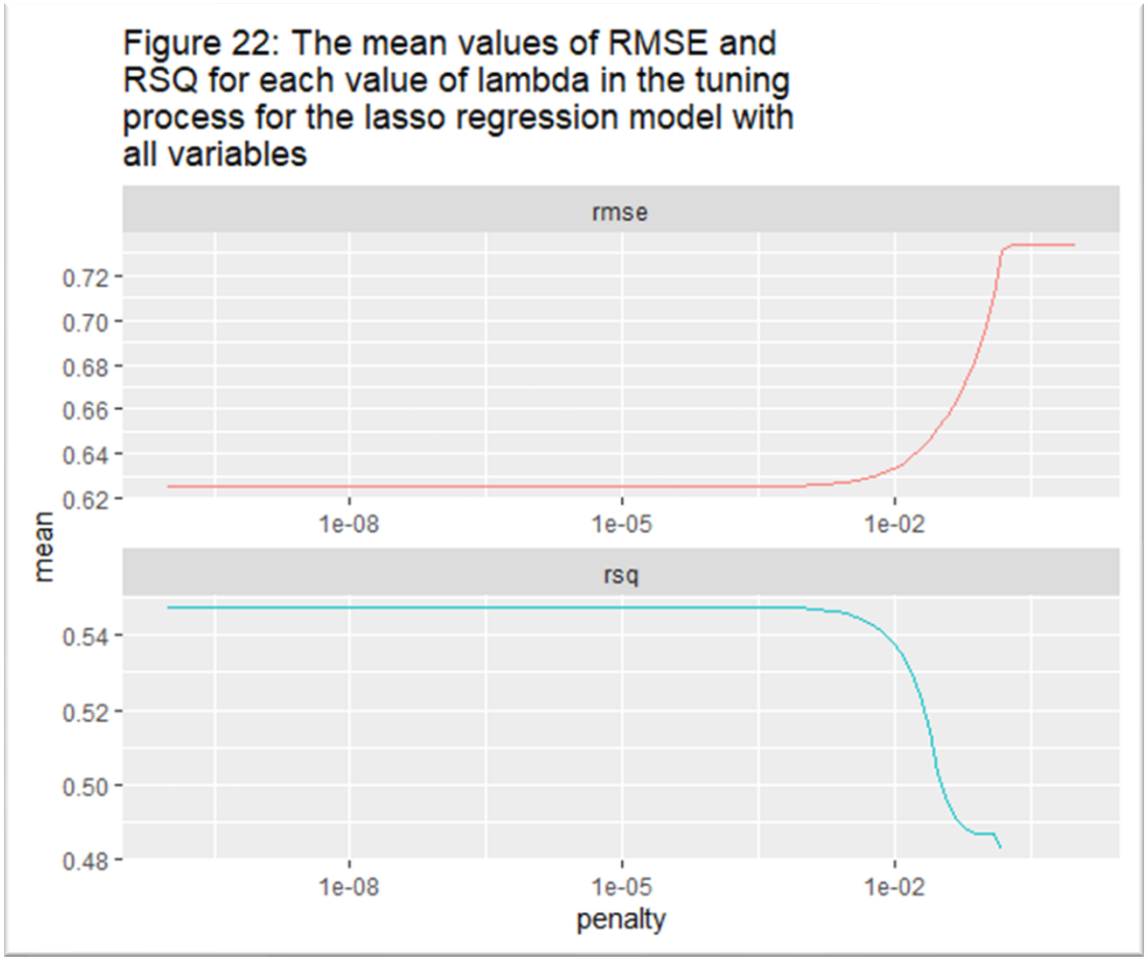
```
# Creating lasso model specification
lasso_spec <-  linear_reg(mode="regression",
                          mixture = 1, # set to lasso
                          penalty = tune()) %>% # Tune the penalty parameter
  set_engine("glmnet")

# Creating a grid of lambda values
lambda_grid <- grid_regular (penalty(),
                             levels = 100) # even when higher, it is always
1e-10

  # Workflow for initial model
initial_wf <-  workflow() %>%
  add_recipe(vgsales_recipe_initial) %>%
  add_model(lasso_spec) %>%
  fit(data = initial_training) # Juicing must be done under the hood

  # Workflow for extension model
final_vars_wf <-  workflow() %>%
  add_recipe(vgsales_recipe_final) %>%
  add_model(lasso_spec) %>%
  fit(data = final_vars_training) # Juicing must be done under the hood

  # Tuning models
lasso_grid_initial_vars <- tune_grid(initial_wf,
                                     resamples = initial_folds,
                                     grid = lambda_grid)
```

```
collect_metrics(lasso_grid_initial_vars) %>%
  ggplot( aes( x = penalty,
               y = mean,
               colour = .metric ) ) +
  geom_line(show.legend = FALSE) +
  facet_wrap( ~.metric,
              scales = "free",
              nrow = 2 ) +
  scale_x_log10() +
  ggtitle( str_wrap("Figure 21: The mean values of RMSE and RSQ for each valu
e of lambda in the tuning process for the lasso regression model with just th
e initial variables", 70))
```




```
lasso_grid_final_vars <- tune_grid(final_vars_wf,
                        resamples = final_vars_folds,
                        grid = lambda_grid)

collect_metrics(lasso_grid_final_vars) %>%
  ggplot( aes( x = penalty,
               y = mean,
               colour = .metric ) ) +
```

```
  geom_line(show.legend = FALSE) +
  facet_wrap( ~.metric,
              scales = "free",
              nrow = 2 ) +
  scale_x_log10() +
  ggtitle( str_wrap("Figure 22: The mean values of RMSE and RSQ for each valu
e of lambda in the tuning process for the lasso regression model with all var
iables", 70))
```




```
# Finding best rsme value
initial_best_rmse <- lasso_grid_initial_vars %>%
  select_best("rmse")
initial_best_rmse
```

```
## # A tibble: 1 x 2
##        penalty .config
##          <dbl> <chr>
## 1 0.0000000001 Preprocessor1_Model001
```

```
final_vars_best_rmse <- lasso_grid_final_vars %>%
  select_best("rmse")
final_vars_best_rmse
```

```
## # A tibble: 1 x 2
##        penalty .config
##          <dbl> <chr>
## 1 0.0000000001 Preprocessor1_Model001
```

```
# Finalizing the two models
initial_lasso_spec_finalized <- finalize_model(lasso_spec, initial_best_rmse
)

final_vars_lasso_spec_finalized <- finalize_model(lasso_spec, final_vars_best
_rmse )
```

The highest penalty value with the lowest RSME values were then selected and used to finalize the model, fit the training data, and validate the resulting fits as observed below. Note that this was 0.0000000001, the lowest penalty parameter available in the grid and an indicator that this model has been overfit, there may be too much complexity for the Lasso regression model.

```
# Fitting testing data
initial_lasso_finalized <- initial_lasso_spec_finalized %>%
  fit( NA_Sales~., data = initial_test_prepro  )

final_vars_lasso_spec <- final_vars_lasso_spec_finalized %>%
  fit( NA_Sales~., data = final_test_prepro)


# Checking model RMSE and RSQ
fit_resamples(initial_lasso_spec_finalized, NA_Sales ~ ., initial_folds) %>%
  collect_metrics() %>%
  knitr::kable(caption = "Table 11: The accuracy of the Lasso model (initial
variables only) from cross validation")

fit_resamples(final_vars_lasso_spec_finalized, NA_Sales ~ ., final_vars_folds
) %>%
  collect_metrics() %>%
  knitr::kable(caption = "Table 12: The accuracy of the Lasso model (with add
itional variables) from cross validation")
```

Table 11: The accuracy of the Lasso model (initial variables only) from cross validation

.metric	.estimator	mean	n	std_err	.config
Rmse	standard	0.4178118	10	0.0085855	Preprocessor1_Model1
Rsq	standard	0.6328408	10	0.0343982	Preprocessor1_Model1

Table 12: The accuracy of the Lasso model (with additional variables) from cross validation

.metric	.estimator	mean	n	std_err	.config
Rmse	standard	0.4076700	10	0.0067134	Preprocessor1_Model1
Rsq	standard	0.6781148	10	0.0402386	Preprocessor1_Model1

The initial lasso model has a larger RSME and a lower Root Mean square (RSQ) than the lasso model including the additional variables. This implies that the model with the additional variables has better predictive power, however it is not advisable to any prediction from this model with such a poor accuracy score. The importance of variables was then determined as follows:

```
# Testing model (initial)
initial_prediction <- initial_lasso_finalized %>%
  predict(new_data = initial_test_prepro) %>%
  bind_cols(initial_test_prepro %>%
              select(NA_Sales))

initial_prediction %>%
  ggplot( aes( x = .pred, y = NA_Sales ) ) +
  geom_point() +
  geom_abline( intercept = 0, slope = 1, colour = "red" ) +
  theme_minimal() +
  ggtitle(str_wrap("Figure 23.1: The values predicted by the lasso model with
just the initial predictors mapped against the True values of NA_Sales from t
he test set"))

# Check errors
initial_prediction %>%
  mutate(errors = NA_Sales - .pred) %>%
  ggplot(aes(x = errors, y = NA_Sales)) +
  geom_point() +
  geom_vline(aes(xintercept = mean(errors))) +
  xlab("Error") +
  ylab("Global sales") +
  ggtitle(str_wrap("Figure 23.2: The error associated witheach of the fitted
values from the LASSO model with just the initial variables with the mean ind
icated with a line. Errors have a mean of zero and a finite variance", 70)) +
  theme_minimal()

# Testing model (final)
final_vars_prediction <- final_vars_lasso_spec %>%
  predict(new_data = final_test_prepro)%>%
  bind_cols(final_test_prepro %>%
              select(NA_Sales))
# Check errors
final_vars_prediction %>%
  mutate(errors = NA_Sales - .pred) %>%
  ggplot(aes(x = errors, y = NA_Sales)) +
  geom_point() +
```

```
  geom_vline(aes(xintercept = mean(errors))) +
  xlab("Error") +
  ylab("Global sales") +
  ggtitle(str_wrap("Figure 24.2: The error associated witheach of the fitted
values from the LASSO model with just the initial variables with the mean ind
icated with a line. Errors have a mean of zero and a finite variance", 70)) +
  theme_minimal()

final_vars_prediction %>%
  ggplot( aes( x = .pred, y = NA_Sales ) ) +
  geom_point() +
  geom_abline( intercept = 0, slope = 1, colour = "red" ) +
  theme_minimal() +
  ggtitle(str_wrap("Figure 24.1: The values predicted by the lasso model with
just the initial predictors mapped against the True values of NA_Sales from t
he test set"))

# Checking variable importance
vip::vi(initial_lasso_finalized) %>%
    mutate( Importance = abs(Importance),
            Variable = fct_reorder(Variable, Importance)) %>%
    ggplot(aes(x=Importance, y= Variable, fill= Sign)) +
    geom_col() +
    ggtitle(str_wrap("Figure 23.3: Variable importance in the lasso model wit
h just the initial predictors", 70))

vip::vi(final_vars_lasso_spec ) %>%
    mutate( Importance = abs(Importance),
            Variable = fct_reorder(Variable, Importance)) %>%
    ggplot(aes(x = Importance, y= Variable, fill= Sign)) +
    geom_col() +
  ggtitle(str_wrap("Figure 24.3: Variable importance in the lasso model with
the additional predictors", 70))
```

Figure 23.1: The values predicted by the lasso model with just the initial predictors mapped against the True values of NA_Sales from the test set

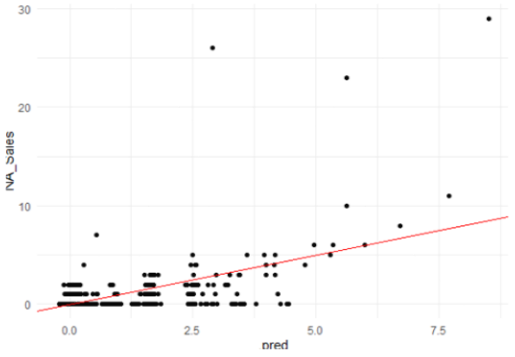
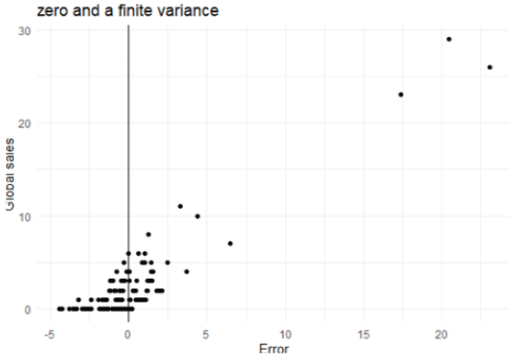
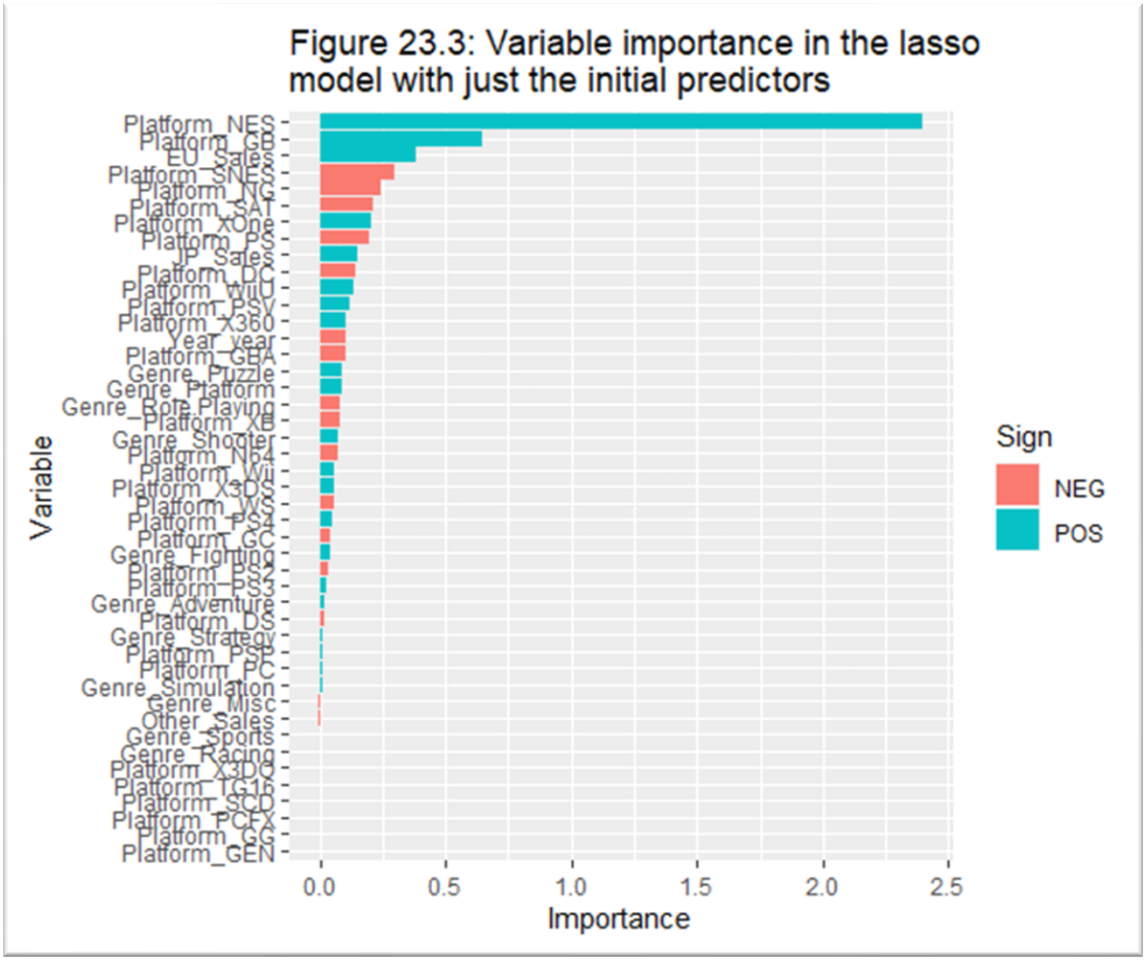
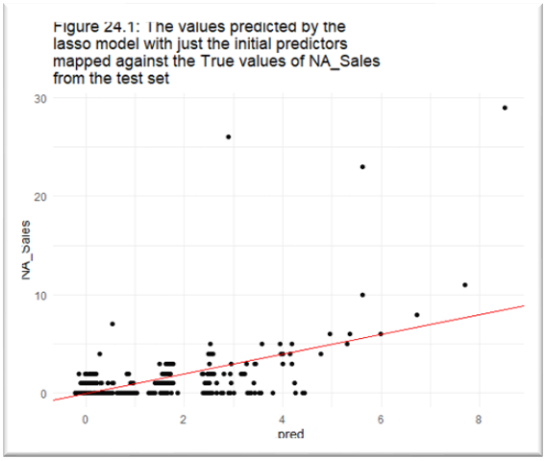
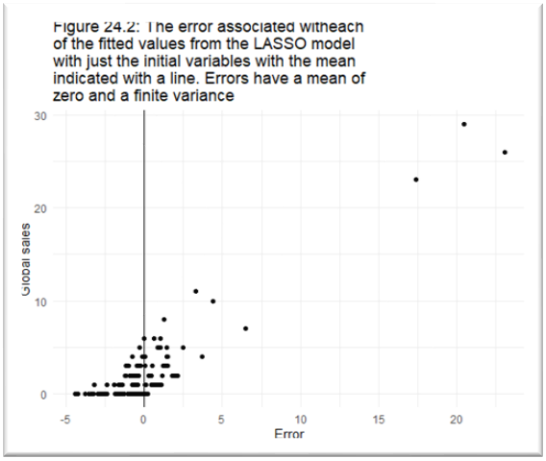


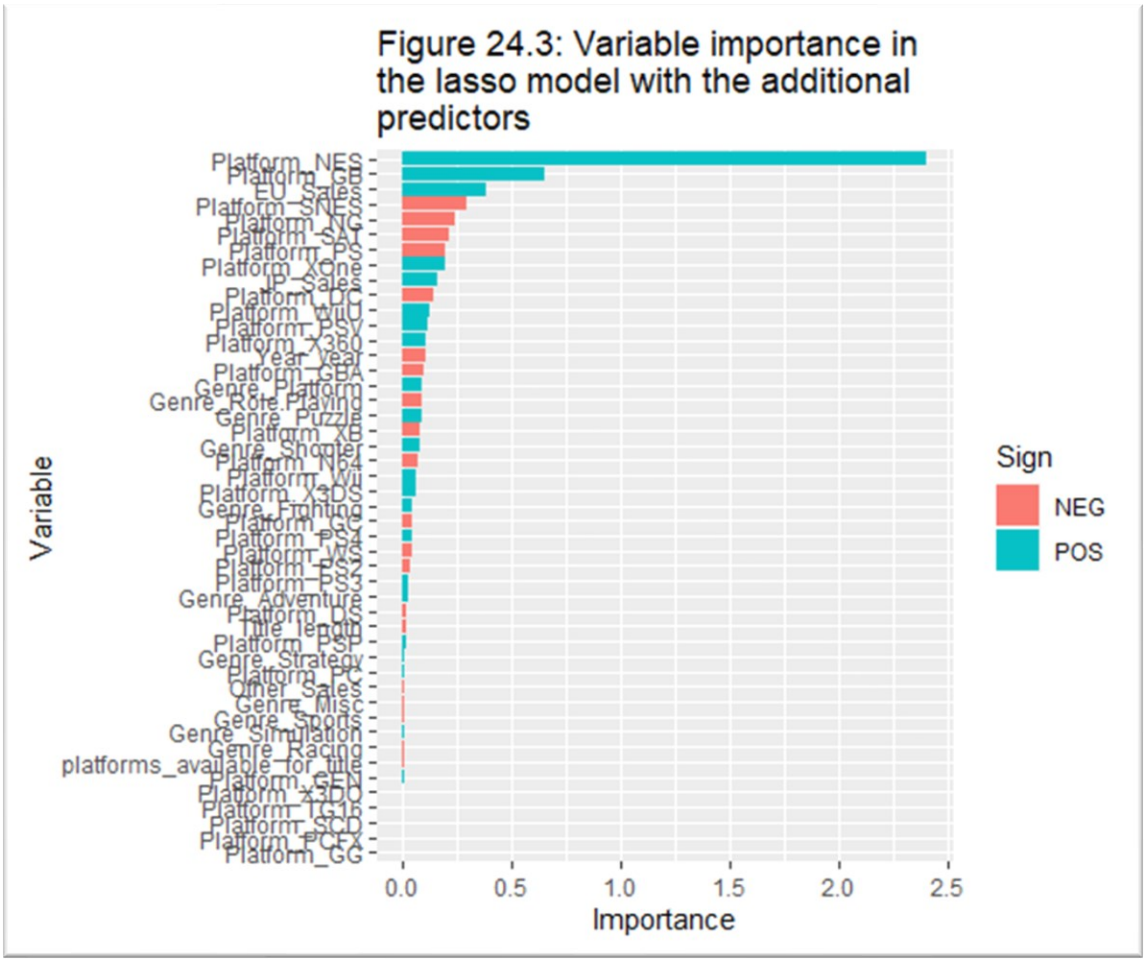
Figure 23.2: The error associated witheach of the fitted values from the LASSO model with just the initial variables with the mean indicated with a line. Errors have a mean of zero and a finite variance











When it comes to predicting North American sales, a linear model makes sense. The unimportant variables didn't appear to have any relation with the important variables, and it wasn't expected that every variable would have importance to begin with. The average of errors was found to be very close to zero in both cases (initial variables only and with

additional variables) and naturally the variance was finite. In terms of the assumptions, the LASSO regression model could be applied.

The additional variables were observed to hold some importance, enough to improve the accuracy of prediction. The other predictors maintained similar degrees and orders of importance. Of the numeric predictors, EU_Sales was the most important, followed by JP_sales, Year, Other_Sales, Title_length, and platforms_available_for_title. The EU_Sales and JP_Sales, had a positive influence on NA_Sales, while the others had a negative influence. In the case of Title_length, this was as hypothesized. This was interesting when it came to Year. It indicated that the higher the year, the lower the sales and was most likely a result of an incomplete data and reduction in data acquisition.

The dummy variables were quite interesting in that Japanese consoles 'Nintendo Entertainment System' and 'Game boy' were found to have high positive importance in predicting NA_Sales. It would be unsurprising to find that the period in time where most of the data was acquired by the source website aligns with the period in which these platforms were at the height of their technology adoption curve. On the whole, Genre appeared to have less importance than Platform. Six of the dummy variables appear to have been set equal to or very near zero, perhaps because their information has already been provided 'latent-ly' elsewhere.

Random Forest model

A random forest differs from a lasso model in a few ways. Random forests create many decision trees and determine the threshold value needed at each node by taking the average of all the trees. The random forest also has tuning capability, although now it is in 'mtry' and 'min_n'. Min_n represents the minimum node size, and mtry represents the number of predictors to be used in the random sampling. The model was supplied with a grid of penalty values for combinations of min_n and mtry. Every value was attempted with resampling and the value that produced the lowest RSME was chosen. RSME was considered the best metric for the same reasons as above. Resampling was done with replacement. This reduces over fitting as the input is constantly being replaced with other values in the set.

The tidymodels rand_forest() function was used to create a regression model using the ranger engine. the number of trees was set to 1000 (1000 samples), and the most optimal combination of min_n (nodes) and mtry (predictor variables) was obtained by tuning those parameters with a grid containing 25 combinations of values. Two models were created, one with just the initial variables, and one with the additional variables.

For those that follow, a useful package providing a template for many models has been provided:

```
#install.packages("usemodels")
library("usemodels")
use_ranger(NA_Sales~., data = initial_training)
```

```
set.seed(2020)
# Folds for the random forest model !!!!! MOVED ABOVE !!!!
#initial_folds_rf <- bootstraps( data = initial_training, times = 10 )
#final_vars_folds_rf <- bootstraps( data = final_vars_training, times = 10)

# Preparing a grid to optimize mtry and min_n
initial_tune_vals <- grid_regular(finalize( mtry(), initial_training_prepro),
                                  min_n(),
                                  levels = 5)

# Creating the model specification with mtry and min_n set for tuning
initial_rf_spec <- rand_forest(mode = "regression",
                               trees = 1000,
                               mtry = tune(),
                               min_n = tune()) %>%
  set_engine("ranger", importance = "permutation")

# Tune the mtry and min_n parameters
set.seed(12345)
doParallel::registerDoParallel()
rf_tuned <- tune_grid(initial_rf_spec,
                      vgsales_recipe_initial,
                      initial_folds_rf, #Same folds, different name
                      grid = initial_tune_vals,
                      metrics = metric_set(rmse, rsq, mae))

# Selecting the best tuned values for the rsme values
best_rmse <- select_best(rf_tuned, metric = "rmse")
best_rmse

## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1    34     2 Preprocessor1_Model04

# Finalizing the model with the tuned values
initial_rf_spec_finalized <- finalize_model(initial_rf_spec, best_rmse)
```

The tuning above found that mtry should be set to 34 and min_n be set to two in order to have the best reduction in the RMSE. The model was finalized on these values. Below we fit the model to the testing data then look at the variable importance and accuracy measure:

```
# Fitting with testing set
initial_rf_fin_fit <- initial_rf_spec_finalized %>%
  fit(NA_Sales ~ ., data = initial_test_prepro)
```

The accuracy of this model was assessed as follows:

```
rf_initial_preds <- predict(initial_rf_fin_fit, # Get class prediction
                            new_data = initial_test_prepro) %>%
  bind_cols(initial_test_prepro %>%
```

```
               select( NA_Sales))

rf_initial_preds %>%
  metrics(truth = NA_Sales, estimate = .pred) %>%
  knitr::kable(caption = "Table 13: The measures of model accuracy for the ra
ndom forest model built from just the initial predictors" )
```

Table 13: The measures of model accuracy for the random forest model built from just the initial predictors

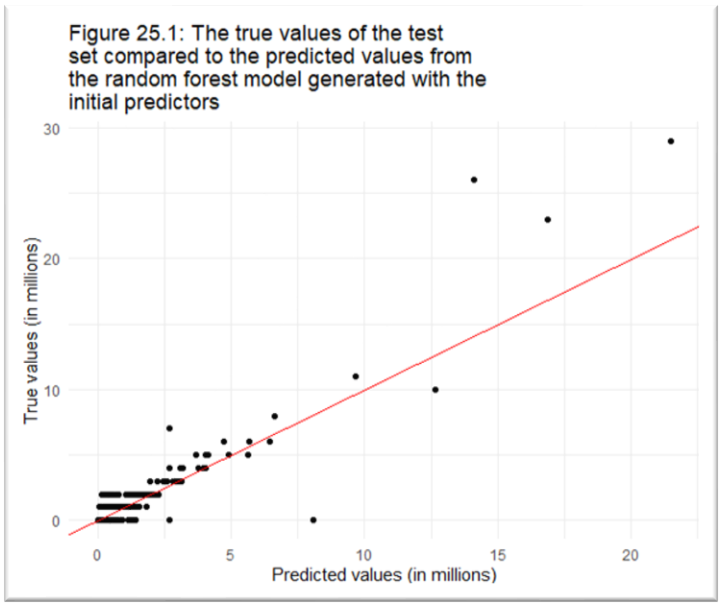
.metric	.estimator	.estimate
rmse	standard	0.3481123
rsq	standard	0.8656444
mae	standard	0.0742310

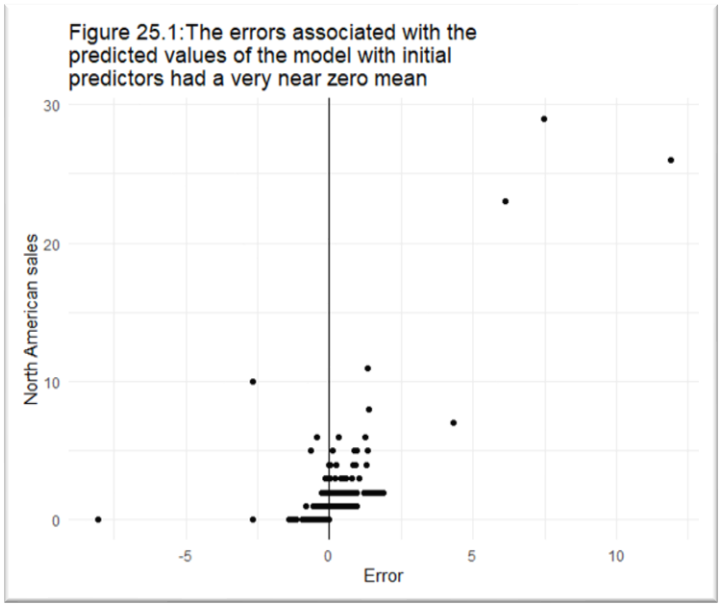
The model had quite a high RMSE, and a low RSQ with a considerable MAE. It outperformed both LASSO models.

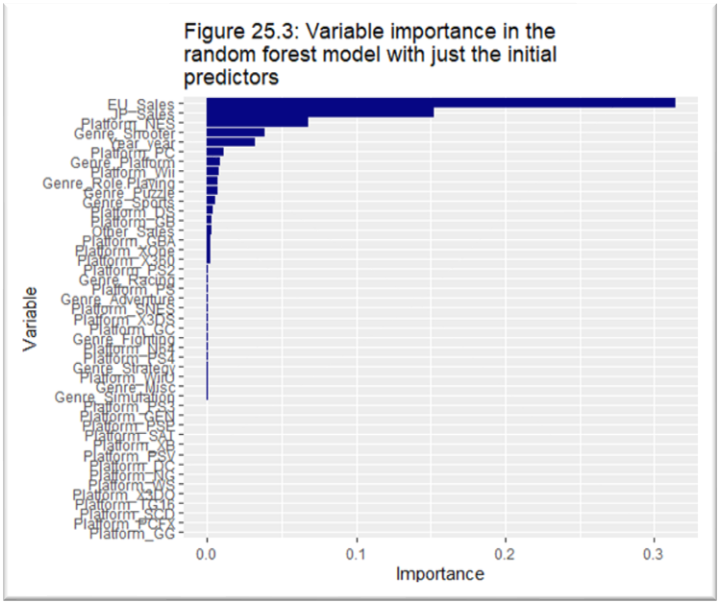
```
# Check errors
rf_initial_preds %>%
  mutate( errors = NA_Sales - .pred) %>%
  ggplot(aes(x = errors, y = NA_Sales)) +
  geom_point() +
  xlab("Error") +
  ylab("North American sales")+
  geom_vline(aes(xintercept = mean(errors))) +
  ggtitle(str_wrap("Figure 25.1:The errors associated with the predicted valu
es of the model with initial predictors had a very near zero mean "))

# Maoping predictions against fitted values
rf_initial_preds %>%
  ggplot( aes( x = .pred, y = NA_Sales ) ) +
  geom_point() +
  geom_abline( intercept = 0, slope = 1, colour = "red" ) +
  theme_minimal() +
  ggtitle(str_wrap("Figure 25.1: The true values of the test set compared to
the predicted values from the random forest model generated with the initial
predictors", 70)) +
  xlab("Predicted values (in millions)") +
  ylab("True values (in millions)")
```

```
# Variable importance
initial_rf_fin_fit %>%
```

```
  vip::vi() %>%
  mutate( Importance = abs(Importance),
          Variable = fct_reorder(Variable, Importance)) %>%
    ggplot(aes(x=Importance, y= Variable)) +
    geom_col(fill = "navy", show.legend = FALSE) +
    ggtitle(str_wrap("Figure 25.3: Variable importance in the random forest m
odel with just the initial predictors", 70))
```







The variable importance identified in the random forest was more in line with the hypotheses that the sales figures would be most important. Aside from that, 'Role Playing' and 'Puzzle' genres were found to be the most important of genres, and Year was also found to have comparatively high importance compared to other variables. Several dummy variables were found to have no importance (observable numerically through the vi() function alone).

The same process was then applied to the data including the additional variables:

```
# Preparing a grid to optimize min_n and cost_complexity
final_vars_tune_vals <- grid_regular(finalize( mtry(), final_training_prepro)
,
                           min_n(),
                           levels = 5)

# Creating the model specification with mtry and min_n set for tuning
final_vars_rf_spec <- rand_forest(mode = "regression",
                        trees = 1000,
                        mtry = tune(),
                        min_n = tune()) %>%
  set_engine("ranger", importance = "permutation")


# Tune the mtry and min_n parameters
set.seed(54321)
doParallel::registerDoParallel()
final_vars_rf_tuned <- tune_grid(final_vars_rf_spec,
                       vgsales_recipe_final,
                       final_vars_folds_rf, #Same folds, different name
                       grid = final_vars_tune_vals,
```

```
                      metrics = metric_set(rmse, rsq, mae))

# Selecting the best tuned values for the rmse value
best_rmse <- select_best(final_vars_rf_tuned, metric = "rmse")

#Finalizing the model with the tuned values
final_vars_rf_spec_finalized <- finalize_model(final_vars_rf_spec, best_rmse)

final_vars_rf_fin_fit <- final_vars_rf_spec_finalized %>%
  fit(NA_Sales ~ ., data = final_test_prepro)

# Comparing fitted values to true values
rf_final_vars_preds <- predict( final_vars_rf_fin_fit, # Get class prediction
                      new_data = final_test_prepro) %>%
  bind_cols(final_test_prepro %>%
              select( NA_Sales ))

# Measuring accuracy
rf_final_vars_preds %>%
  metrics(truth = NA_Sales, estimate = .pred) %>%
  head() %>%
  knitr::kable(caption = "Table 14: The measures of model accuracy f
  or the random forrest model built from just the initial predictors")

  ## # A tibble: 3 x 3
  ##   .metric .estimator .estimate
  ##   <chr>   <chr>          <dbl>
  ## 1 rmse    standard      0.292
  ## 2 rsq     standard      0.918
  ## 3 mae     standard      0.0513

  # Mapping predictions against fitted values
  rf_final_vars_preds %>%
    ggplot( aes( x = .pred, y = NA_Sales ) ) +
    geom_point() +
    geom_abline( intercept = 0, slope = 1, colour = "red" ) +
    theme_minimal() +
    ggtitle(str_wrap("Figure 26.1: The true values of the test set compared t
  o the predicted values from the random forest model generated with the addi
  tional predictors", 45)) +
    xlab("Predicted values (in millions)") +
    ylab("True values (in millions)")
```

```
# Check errors
rf_final_vars_preds %>%
  mutate(errors = NA_Sales - .pred) %>%
  ggplot(aes(x = errors, y = NA_Sales)) +
  geom_point() +
  xlab("Error") +
  ylab("North American sales") +
```

```
  geom_vline(aes(xintercept = mean(errors))) +
  ggtitle(str_wrap("Figure 26.2:The errors associated with the predicted valu
es of the model created with the additional predictors had a very near zero m
ean", 45))

# Checking Variable importance
final_vars_rf_fin_fit %>%
  vip::vi() %>%
  mutate( Importance = abs(Importance),
          Variable = fct_reorder(Variable, Importance)) %>%
    ggplot(aes(x=Importance, y= Variable)) +
    geom_col(fill = "navy", show.legend = FALSE) +
    ggtitle(str_wrap("Figure 26.3: Variable importance in the random forest m
odel with the additional predictors", 45))
```

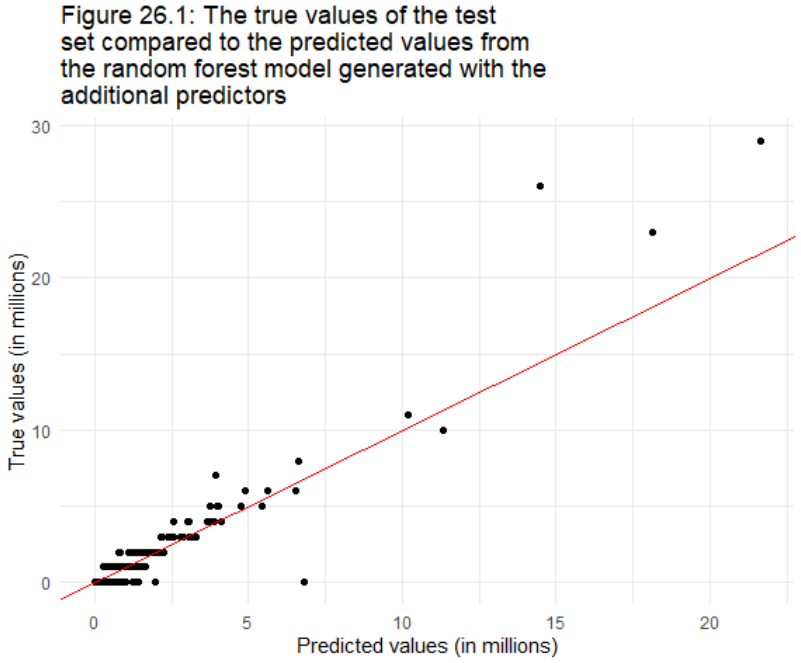


Table 14: The measures of model accuracy for the random forest model with the additional predictors

.metric	.estimator	.estimate
rmse	standard	0.2985364
rsq	standard	0.9195488
mae	standard	0.0542567

Figure 26.2:The errors associated with the predicted values of the
with the additional predictors had a very near zero mean

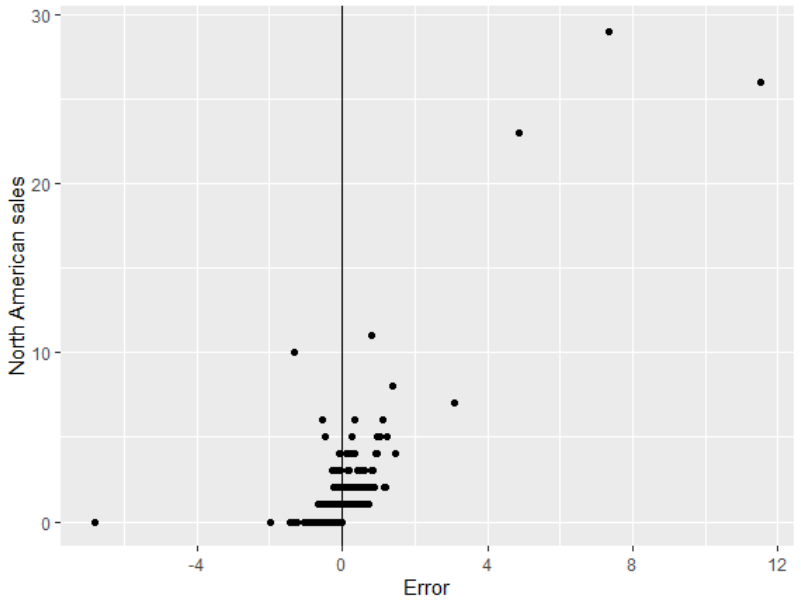
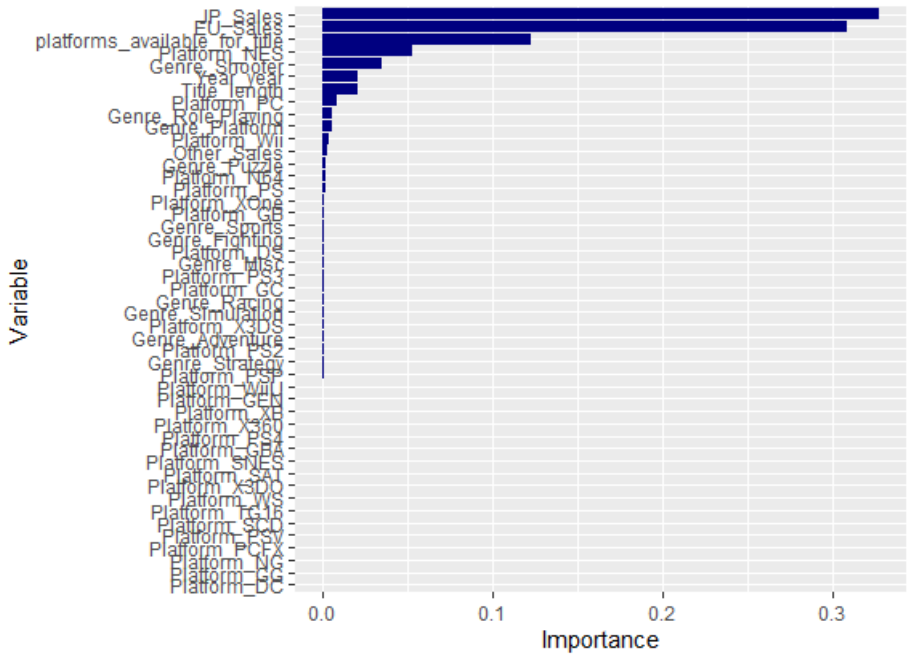


Figure 26.3: Variable importance in the random forest model with the additional predictors



In comparison with the initial random forest model, it appeared that the additional variables improved prediction accuracy. The RMSE was lower, the RSQ was higher, and the Mean absolute error (MAE) was also lowest.

The variable importance analysis identified the top 10 predictors were:

1. EU_Sales
2. JP_Sales, Platform_NES
3. platforms_available_for_title
4. Genre_Shooter
5. Year
6. Other_Sales
7. Genre_Platform
8. Platform_PC
9. Title_length
10. Platform_Wii

However, there is one line of particular importance in the 'vip' documentation:

'Note that both methods for constructing VI scores can be unreliable in certain situations; for example, when the predictor variables vary in their scale of measurement or their number of categories[...], or when the predictors are highly correlate'.

These phenomena have been thoroughly examined in the literature (Strobl et al. 2008; Strobl et al. 2007; Strobl, Hothorn & Zeileis 2009). The large number of categories/ scales, and an intrinsic bias towards correlated predictors may be influencing the variable importance observed above, and indeed all previous variable importance analyses, (as there were a considerable number of categories, and differing scales even after normalizing). The most important predictors may be exaggerated, and the importance of others may be underestimated.

The possibility that this was one of the situations in which the 'vip' package was unreliable was assessed. Step_corr was implemented to remove highly correlated variables (of which none were identified) and reduced the number of categorical variables with step_other (categories reduced to 10 Platforms, and 11 Genres). Correlation issues were not expected to make much of an impact as that only results in a bias of importance; it's unlikely to improve such a low-level predictor but it was tried all the same. Making the number of categories smaller may assist in reducing the scales and assist in the variable importance measurements:

```
set.seed(54321)
# Preparing a grid to optimize min_n and mtry
again_tune_vals <- grid_regular(finalize( mtry(), final_vars_training),
                                min_n(),
                                levels = 5)

# Creating the model specification with mtry and min_n set for tuning
again_rf_spec <- rand_forest(mode = "regression",
                             trees = 1000,
```

```
                        mtry = tune(),
                        min_n = tune()) %>%
  set_engine("ranger", importance = "permutation")

set.seed(246810)
# Recipe to test variable importance accuracy
again_recipe <- recipe(NA_Sales ~ ., data = final_vars_training) %>%
  step_log(JP_Sales, EU_Sales, Other_Sales, offset = 1) %>% #Account for skew
  step_log(all_outcomes(), offset = 1, skip = TRUE) %>%
  step_date(Year, features = "year") %>% # creating time predictors
  step_rm(Year) %>%
  step_normalize(all_numeric(), -all_outcomes()) %>%
  step_other(Platform, Genre) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_corr(all_predictors()) %>%
  prep()
knitr::kable(tidy(again_recipe, 6), caption = "Table 15: The retained
categorical variables remaining after step_other()")# 9 retained + Other
```

Table 15: The retained categorical variables remaining after step_other()

terms	retained	id
Platform	DS	other_Q6YZu
Platform	PC	other_Q6YZu
Platform	PS	other_Q6YZu
Platform	PS2	other_Q6YZu
Platform	PS3	other_Q6YZu
Platform	PSP	other_Q6YZu
Platform	Wii	other_Q6YZu
Platform	X360	other_Q6YZu
Platform	XB	other_Q6YZu
Genre	Action	other_Q6YZu
Genre	Adventure	other_Q6YZu
Genre	Fighting	other_Q6YZu
Genre	Misc	other_Q6YZu
Genre	Platform	other_Q6YZu
Genre	Racing	other_Q6YZu
Genre	Role-Playing	other_Q6YZu
Genre	Shooter	other_Q6YZu
Genre	Simulation	other_Q6YZu
Genre	Sports	other_Q6YZu

```
knitr::kable(tidy(again_recipe, 8)) # None at 0.9 threshold,

                                    # rf models pick up on interactions terms
knitr::kable(tidy(again_recipe, 5) ), caption = Table 16: The Mean and standa
rd deviation of each normalized numerical variable.")) # Discrepancy in range
s remains
```

Table 16: The Mean and standard deviation of each normalized numerical variable.

terms	statistic	value	id
Title_length	mean	24.0003214	normalize_arT9E
JP_Sales	mean	0.0136048	normalize_arT9E
EU_Sales	mean	0.0263797	normalize_arT9E
Other_Sales	mean	0.0042443	normalize_arT9E
platforms_available_for_title	mean	2.1255724	normalize_arT9E
Year_year	mean	2006.3719772	normalize_arT9E
Title_length	sd	12.8428179	normalize_arT9E
JP_Sales	sd	0.1207194	normalize_arT9E
EU_Sales	sd	0.1673213	normalize_arT9E
Other_Sales	sd	0.0653265	normalize_arT9E
platforms_available_for_title	sd	1.5940374	normalize_arT9E
Year_year	sd	5.8865469	normalize_arT9E

```
# Tune the mtry and min_n parameters
set.seed(54321)
doParallel::registerDoParallel()
again_rf_tuned <- tune_grid(again_rf_spec,
                      again_recipe,
                      final_vars_folds_rf, #Same folds, different name
                      grid = again_tune_vals,
                      metrics = metric_set(rmse, rsq, mae))

# Selecting the best tuned values for the rmse value
best_rmse <- select_best(again_rf_tuned, metric = "rmse")

#Finalizing the model with the tuned values
again_rf_spec_finalized <- finalize_model(again_rf_spec, best_rmse)

again_rf_fin_fit <- again_rf_spec_finalized %>%
  fit(NA_Sales ~ ., data = final_test_prepro)


# Comparing predictions and measuring accuracy
```

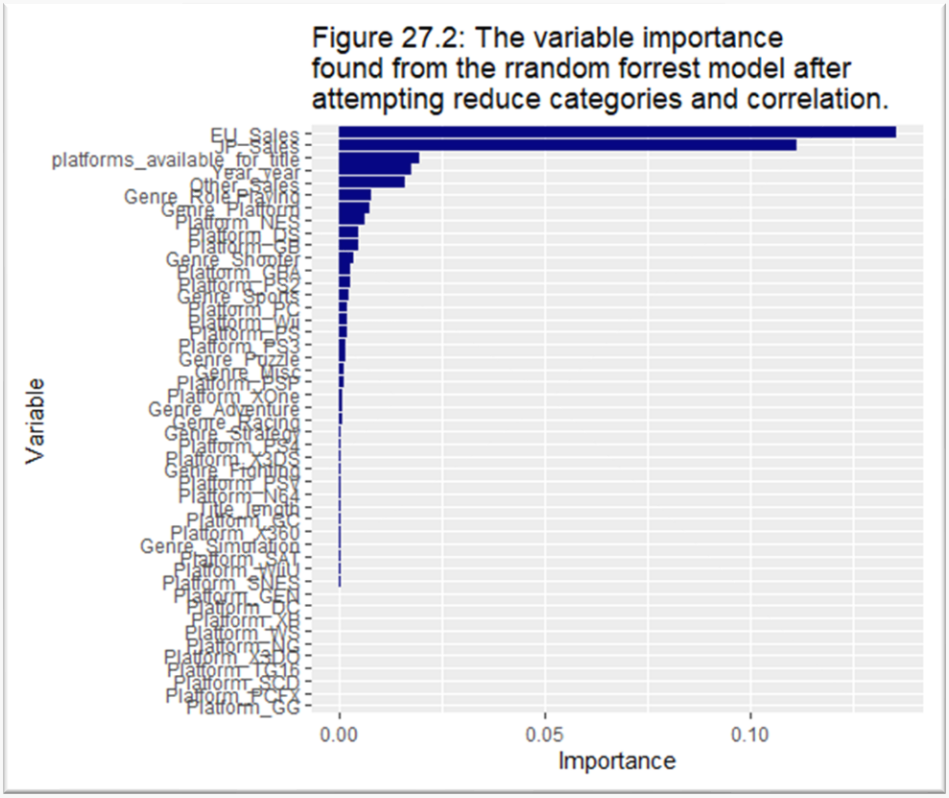
```
again_preds <- predict( again_rf_fin_fit, # Get class prediction
                        new_data = final_test_prepro) %>%
  bind_cols(final_test_prepro %>%
              select( NA_Sales ))

# Mapping fitted values against true values
again_preds %>%
  ggplot( aes( x = .pred, y = NA_Sales ) ) +
  geom_point() +
  geom_abline( intercept = 0, slope = 1, colour = "red" ) +
  theme_minimal() +
  ggtitle(str_wrap("Figure 27.1: The true values of the test set compared to
the predicted values from the random forest model generated with all predicto
rs but Title_length", 45)) +
  xlab("Predicted values (in millions)") +
  ylab("True values (in millions)")

# Measuring accuracy
again_preds %>%
  metrics( truth = NA_Sales, estimate = .pred) %>%
  knitr::kable(caption = "Table 17: The accuracy of the random forest model a
fter attempting reduce categories and correlation")

# Accessing variable importance
again_rf_fin_fit %>%
  vip::vi() %>%
  mutate( Importance = abs(Importance),
          Variable = fct_reorder(Variable, Importance)) %>%
    ggplot(aes(x = Importance, y = Variable)) +
    geom_col(fill = 'navy', show.legend = FALSE) +
  ggtitle(str_wrap("Figure 26.2: The variable importance found from the rando
m forest model excluding the Title_length variable but including all others,
45))
```

It is said the road to hell is paved with good intentions. In the final model, correlation and category size was accounted for in an attempt to improve variable importance accuracy. But in so doing, the accuracy of the model itself was sacrificed rather harshly. There were 32 none negative values, likely because mtry was set to 32.

The dummy variables had a range of 0 to 1, which was comparable to many of the numeric predictors, but quite far off others. This remains a source of error for the variable importance analyses. To more accurately measure variable importance in a classification setting, it has been suggested that the random forests be built from unbiased classification trees using sampling without replacement; available in the 'party' package (Strobl, Hothorn & Zeileis 2009). There may be a similar technique that could be employed here to be more confident in identifying importance. However, simply riding sampling-with-replacement will have detrimental over-fitting effects. By sampling with replacement, the trees in the model are more likely to be representative of reality.

Table 17: The accuracy of the random forest model after attempting reduce categories and correlation

.metric	.estimator	.estimate
rmse	standard	0.5959625
rsq	standard	0.7101636
mae	standard	0.1381563



The model selected is the random forest model containing the extra variables which had the best accuracy scores. Before the prediction is shared, here are some sources of error that need to be carefully considered:

- The data set has inconsistent sampling over the time period, the data will be weighted by the number of values in a given year, but to step_downsample by year would be akin to making a prediction for NA_Sales that was independent of the year.
- The data contains rows with questionable observations and values (such as US weekly sales)
- Missing data has been approximated as best as possible

- The data only provides context up until two years ago, the assumption is that it is still representative of a current context.

In any case, here is the predicted sales figure (in millions) for North America for 'The Fatal Empire':

```
# Making prediction
final_vars_prediction_rf <- final_vars_rf_fin_fit %>%
  predict(new_data = final_prediction_set) %>%
  mutate_if(is.numeric, round, digits=2) %>%
  rename("North american sales prediction (in millions) for PS4 role-playing
game titled 'The Fatal Empire' with 2.58 million copies sold in Japan, 0.53 m
illion copies in Europe, and 0.1 million copies in other parts of the world."
= .pred)

knitr::kable(final_vars_prediction_rf)
```

North American sales prediction (in millions) for PS4 role-playing game, titled 'The Fatal Empire' with 2.58 million copies sold in Japan, 0.53 million copies in Europe, and 0.1 million copies in other parts of the world.
1.92

(Improvements I'd make if the marks justified it):

- Check importance with the party package.
- Could use the 'textrecipes' library to add a step that creates a franchise predictor if any of :, -, 2, ii, etc. are present; a binary would be sufficient.
- Could obtain more data or find true missing values.
- Could account for the fact that the population size (effectively the pool of customers) has increased over time (add together all sales for a five-year period, then normalize to represent changes in the 'customer-scape').
- Account for population size differences in the sales figures. The number of gamers in one part of the world may exceed other parts of the world thus skewing sales figures.

References

Dania M Rodriguez, Michael A Johansson, Luis Mier-y-Teran-Romero, moiradillon2, eyq9, YoJimboDurant & Daniel Mietchen 2017, 'dataMeta: Making and appending a data dictionary to an R dataset', *Zenodo*, vol. 1, no. 1.

Greenwell, BM, Boehmke, BC & McCarthy, AJ 2018, 'A simple and effective model-based variable importance measure', *arXiv preprint arXiv:1805.04755*.

Hlaváčková-Schindler, K 2016, 'Prediction Consistency of Lasso Regression Does Not Need Normal Errors', *British Journal of Mathematics & Computer Science*, vol. 19, no. 4, 2016-01-10, pp. 1-7.

Kassambara, A 2019, 'ggcorrplot: Visualization of a Correlation Matrix using 'ggplot2'', vol. R package version 0.1.3, <https://CRAN.R-project.org/package=ggcorrplot>.

Kassambara, A & Kassambara, MA 2020, 'Package 'ggpubr'', *R package version 0.1*, vol. 6.

Kuhn, M 2015, 'Caret: classification and regression training', *Astrophysics Source Code Library*, p. ascl: 1505.1003.

Kuhn, M 2022, 'Boilerplate Code for 'Tidymodels' Analyses', vol. 1-5.

Kuhn, M & Wickham, H 2020, 'Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles', *Boston, MA, USA.[(accessed on 10 December 2020)]*.

McNamara, A, Arino de la Rubia, E, Zhu, H, Ellis, S & Quinn, M 2018, 'skimr: Compact and flexible summaries of data', *R package version*, vol. 1, no. 1.

Strobl, C, Boulesteix, A-L, Kneib, T, Augustin, T & Zeileis, A 2008, 'Conditional variable importance for random forests', *BMC Bioinformatics*, vol. 9, no. 1, pp. 1-11.

Strobl, C, Boulesteix, A-L, Zeileis, A & Hothorn, T 2007, 'Bias in random forest variable importance measures: Illustrations, sources and a solution', *BMC Bioinformatics*, vol. 8, no. 1, pp. 1-21.

Strobl, C, Hothorn, T & Zeileis, A 2009, 'Party on!'.

Wickham, H, Averick, M, Bryan, J, Winston, C, McGowan, LDA, François, R, Grolemund, G, Hayes, A, Henry, L, Hester, J, Kuhn, M, Pedersen, TL, Miller, E, Bache, SM, Müller, K, Ooms, J, Robinson, D, Seidel, DP, Spinu, V, Takahashi, K, Vaughan, D, Wilke, C, Woo, K & Yutani, H 2019, 'Welcome to the {tidyverse}', *Journal of Open Source Software*, vol. 4, no. 43, p. 1686.