Video Game Sales - CASE STUDY AND DATA ANALYSIS using PCA, Random Forest and Lasso Regression

Simon Kennedy

20th April 2022

The Fatal Empire

SUMMARY FOR CEO

Sales of The Fatal Empire in North America are predicted to be 1.082 million copies. Two different predictive models were created and tested with the most accurate one being implemented. The most influential factors for the prediction were the sales figures from Europe and other parts of the world and thirdly sales in Japan. The accuracy of the model is only fair with the actual sales being different by a margin of approximately 520,000 copies.

OVERVIEW FOR MANAGER

Introduction

This report assesses the optimum predictive model of either Lasso Regression or Random Forest, given the data provided to predict sales in North America of the recently released video game *The Fatal Empire*.

Data cleaning

The dataset provided (vgsales.csv) required some alterations to enable the most accurate analysis and modelling. Firstly, the variable *Year* required conversion from character to numeric type and Platform and Genre were converted to factors. There was some missing or incorrect data in the *Year* variable, which was dropped to prevent errors in the analysis, pre-processing and modelling.

The categorical variables of *Genre* and *Platform* had high levels of unique values and many of these had no North American sales equal to zero. These were filtered out to give a clearer analysis of the most important Genre's and Platform's when viewing the plots and tables. The variable of *Publisher* was not required for prediction as no data was given in the brief. This was removed prior to the pre-processing stage.

Relationships between variables

A Parallel Coordinates Plot (Fig 2.1) was used to assess for any natural grouping in the data. This did not yield and significant result except that the Wii platform had the highest sales in North America and Europe, 2nd most in other parts of the world, but not in Japan. Principal Component Analysis (PCA) was used and the PCA loadings of PC1 in Figure 2.2 highlighted European sales, followed by sales in other parts of the world and then Japanese sales as the most influential variables. Scatterplots confirmed these findings with European sales as having the strongest positive linear relationship (Fig 2.6) followed by Other parts of the world (Fig 2.8) and then Japan (Fig 2.7) with a weaker positive relationship. The relationship of North American sales with the genre and platform of release was not clear in the boxplots of Figures 2.4 and 2.5 respectively. The PCA revealed the DS platform as the most important platform as seen in the PC loadings of PC2 and still very influential in PC3 (Fig 2.2). Of further note, it was demonstrated that sales have dropped quite sharply across the world since their peak in 2009, particularly in North America. There appears to be a lack of data for the years since 2017.

Pre-processing steps

The data was split into testing and training data sets. A recipe was created using the training data and used to remove the Name and Year variables as these are not required for prediction. The categorical predictor variables of Platform and Genre were converted to dummy variables and then all the predictors could be normalised for the next step taken of PCA with 36 components. The first 36 of 44 components was able to account for 90% of the variance of the data. Finally, the recipe was prepped and juiced to extract the principal component values.

Results part 1

Bootstraps were prepared from the pre-processed training data and used to tune both the Lasso regression and the Random forest models. Each model's accuracy was measured using the cross-validated pre-processed training data. The accuracy measure used was the residual mean squared error (RMSE) which in this case refers to the average number of sales that the model will be incorrect by (lower is best), and R squared which is a measure between 0 and 1 for how well the variables used in the model account for the prediction, with higher the better. The Random forest model had the lowest average error of 0.42 million sales compared to Lasso regression of 0.48 million sales on the training data. The R squared for the Random forest model was 0.74 compared to the Lasso regression of 0.64. The Random forest model was the best model.

Results part 2

The most important component by far was PC1 (fig 5.1). When looking at PC1, it is clearly seen that European sales had the strongest influence followed by sales in other parts of the world and then sales in Japan. Platform and Genre types had far less influence. When tested on the testing data, the Random forest model perform slightly lower than on the cross-validated training data. The residual mean square error was approximately 0.52 million sales, i.e. 0.1 million sales greater error in the prediction. The variation in the model (R squared) was 0.11 worse at approximately 0.63. This suggests some bias in the model towards under prediction which can be seen in fig 3.4, with more of the data points above the red line.

Predicted sales

The final model predicts 1.082 million sales of The Fatal Empire in North America, with an average error range of 0.52 million sales.

TECHNICAL REPORT FOR STATISTICIAN

1. Data cleaning

Part 1. Data dictionary

```
data_dict <- tribble(
  ~ variable, ~ description,
  'Name', 'The games name',
  'Platform', 'Platform of the games release (i.e. PC,PS4, etc.)',
  'Year', 'Year of the game\'s release',
  'Genre', 'Genre of the game',
  'Publisher', 'Publisher of the game',
  'NA_Sales', 'Sales in North America (in millions)',
  'EU_Sales', 'Sales in Europe (in millions)',
  'JP_Sales', 'Sales in Japan (in millions)',
  'Other_Sales', 'Sales in the rest of the world (in millions)',
)
data_dict %>%
  kable(caption = "Table 1.1. Data dictionary")
```

Table 1.1. Data dictionary

variable	description
Name	The games name

Table 1.1. Data dictionary

variable	description
Platform	Platform of the games release (i.e. PC,PS4, etc.)
Year	Year of the game's release
Genre	Genre of the game
Publisher	Publisher of the game
NA_Sales	Sales in North America (in millions)
EU_Sales	Sales in Europe (in millions)
JP_Sales	Sales in Japan (in millions)
Other_Sales	Sales in the rest of the world (in millions)

Load the data and view

```
vgsales <- read.csv("C:\\Users\\skent\\Downloads\\vgsales.csv")

vgsales <- as_tibble(vgsales)

vgsales %>%

  head()

## # A tibble: 6 x 9

##   Name     Platform Year  Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales

##   <chr>    <chr>    <chr> <chr> <chr>        <dbl>    <dbl>    <dbl>       <dbl>

## 1 Wii Spo~ Wii      2006  Spor~ Nintendo      41.5    29.0      3.77        8.46

## 2 Super M~ NES      1985  Plat~ Nintendo      29.1     3.58     6.81        0.77

## 3 Mario K~ Wii      2008  Raci~ Nintendo      15.8    12.9      3.79        3.31

## 4 Wii Spo~ Wii      2009  Spor~ Nintendo      15.8    11.0      3.28        2.96

## 5 Pokemon~ GB       1996  Role~ Nintendo      11.3     8.89    10.2         1

## 6 Tetris   GB       1989  Puzz~ Nintendo      23.2     2.26     4.22        0.58
```

Skim the data

```
vgsales %>%

  skim()
```

Data summary

Name	Piped data
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

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
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	▇▁▁▁▁

Key findings from skimming the data

There are 16598 rows and 9 variables. There are 5 variables read in as character type and 4 as numeric.

There are 11493 unique video game names, published by 579 different publishers, across 31 different platforms over a 40 year time frame. The games have been categorised into 12 different genres. The sales of each game have been measured for 4 different regions across the world, North America, Europe, Japan and other parts of the world .

The response variable for this project is NA_Sales which is numeric. The predictor variables are potentially all other variables except for Name and Year. This includes the character variables of Platform, Genre, and Publisher, and the numeric variables of EU_Sales, JP_Sales and Other_Sales. Name is essentially an ID column. Year will be informative for the analysis stage, but like Publisher this variable has not been included in the in the in the brief so won't be used in the prediction phase.

There is no missing data indicated and no whitespace in the character variables.

Variables types

Some variables are not read in correctly for the project. Platform and Genre have been read in as characters but should be factors. Year has been read in as a character but should be numeric. Publisher should remain a character. All numeric variables are read in correctly.

Converting variables to correct type

```
vgsalesA <- vgsales %>%
  mutate(Platform = as_factor(Platform),
         Year = as.numeric(Year),
         Genre = as_factor(Genre))
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
vgsalesA %>% skim()
```

Data summary

Name	Piped data
Number of rows	16598
Number of columns	9
Column type frequency:	
character	2
factor	2
numeric	5
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
Publisher	0	1	3	38	0	579	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Platform	0	1	FALSE	31	DS: 2163, PS2: 2161, PS3: 1329, Wii: 1325
Genre	0	1	FALSE	12	Act: 3316, Spo: 2346, Mis: 1739, Rol: 1488

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Year	271	0.98	2006.41	5.83	1980	2003	2007.00	2010.00	2020.00	▁▁▂▇▂
NA_Sales	0	1.00	0.26	0.82	0	0	0.08	0.24	41.49	▇▁▁▁▁
EU_Sales	0	1.00	0.15	0.51	0	0	0.02	0.11	29.02	▇▁▁▁▁
JP_Sales	0	1.00	0.08	0.31	0	0	0.00	0.04	10.22	▇▁▁▁▁
Other_Sales	0	1.00	0.05	0.19	0	0	0.01	0.04	10.57	▇▁▁▁▁

By skimming the data again we can see that the type conversions have been performed. Of note is that there are now 271 missing entries in the 'Year' variable. This was not picked up when it was a character type. These NA's will now be removed from year so that further analysis and modeling can be performed. Removing missing values from the dataset is not necessarily best practice. Further imputation methods could be considered to address this.

```
vgsalesA <- vgsalesA %>%

  drop_na(Year)

vgsalesA %>%

  skim_without_charts()
```

Data summary

Name	Piped data
Number of rows	16327
Number of columns	9
_____	
Column type frequency:	
character	2
factor	2
numeric	5
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Name	0	1	1	132	0	11360	0
Publisher	0	1	3	38	0	577	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Platform	0	1	FALSE	31	DS: 2133, PS2: 2127, PS3: 1304, Wii: 1290
Genre	0	1	FALSE	12	Act: 3253, Spo: 2304, Mis: 1710, Rol: 1471

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Year	0	1	2006.41	5.83	1980	2003	2007.00	2010.00	2020.00
NA_Sales	0	1	0.27	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

There are no longer any missing values. The data set is ready for analysis.

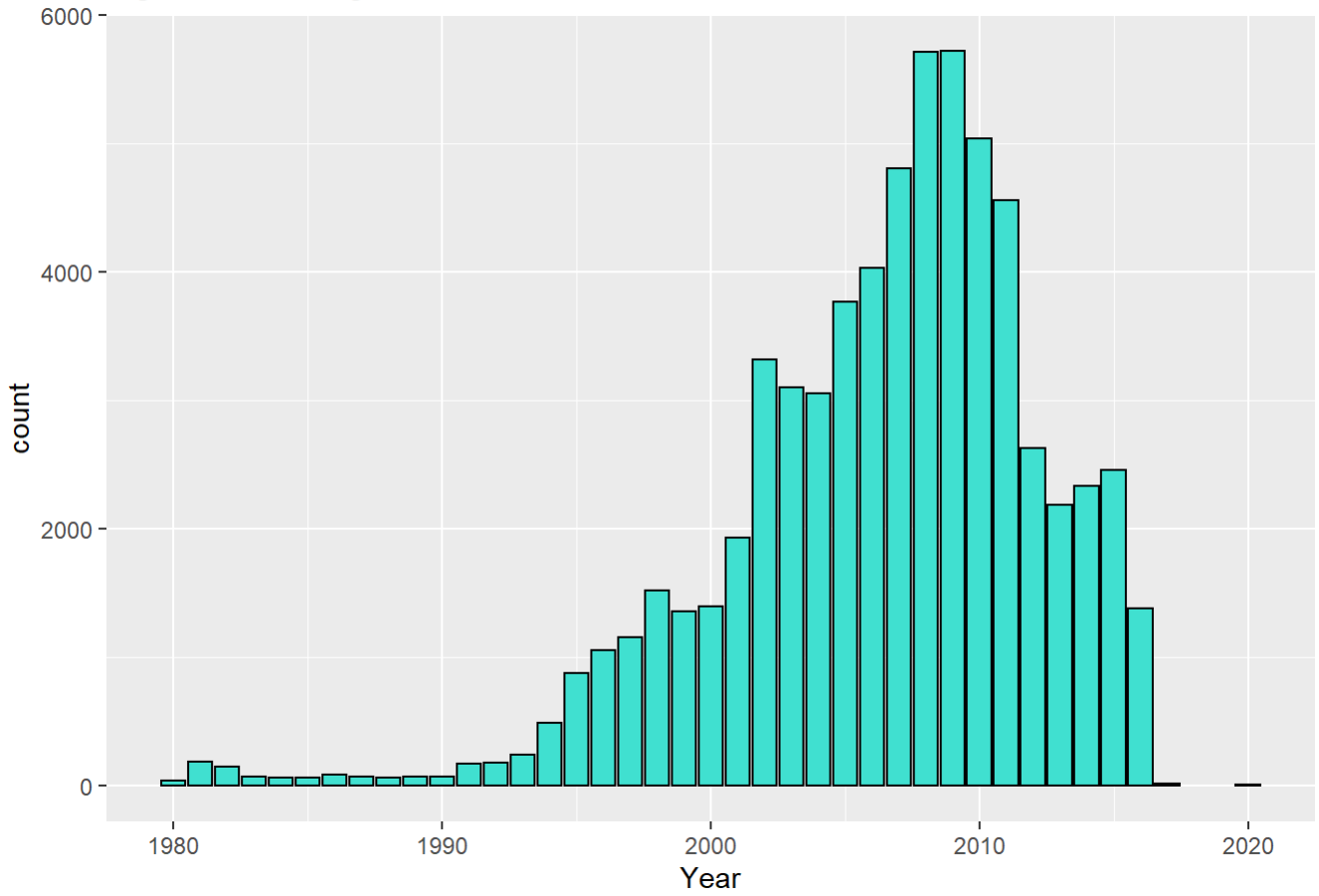
Part 2. Plotting variables and univariate analysis

Year

```
vgsales_long <- vgsalesA %>%
  pivot_longer(cols = NA_Sales:Other_Sales) %>%
  rename(Region = name,
         Sales = value)


vgsales_long %>%
  ggplot(aes(x = Year)) +
  geom_bar(col = 'black', fill = 'turquoise') + # geom_bar presents individual year
's better than geom_histogram
  labs(title = "Figure 1.1 Histogram of Year")
```

Figure 1.1 Histogram of Year



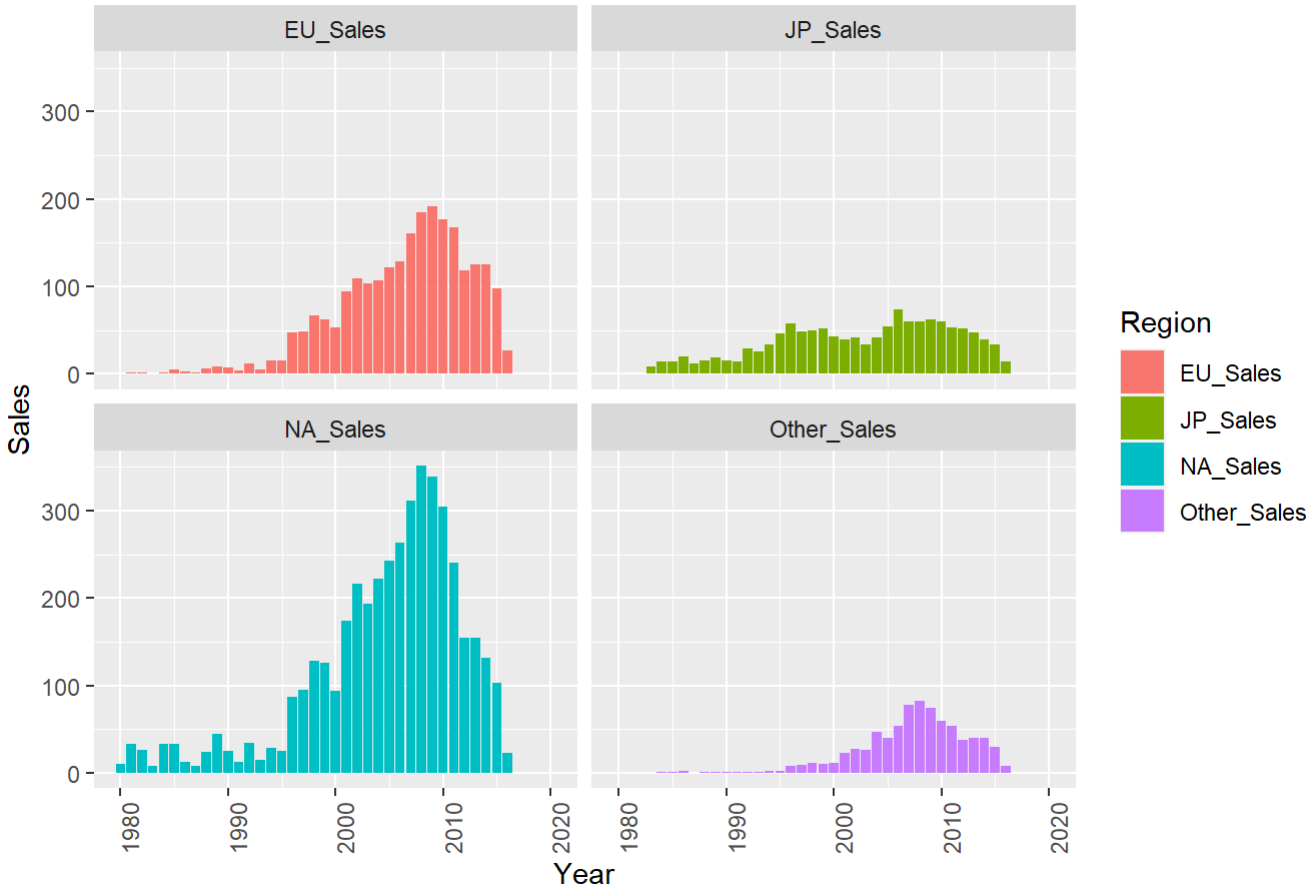
This chart shows a unimodal distribution with a L skew. The modal value is in 2009. Counts have increased steadily from the 1990's with a peak in 2009, then a sharp decline thereafter.

There is some outlier data from 2017 onward. This could suggest incomplete data and can be seen in the last 4 years of the data. This may impact modeling later on. We will keep this in mind when processing the data further.

Plots of key variables with explanation of findings

```
vgsales_long %>%
  ggplot(aes(x = Year, y = Sales, fill = Region)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~Region) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "Fig 1.2 Sales by Region")
```

Fig 1.2 Sales by Region

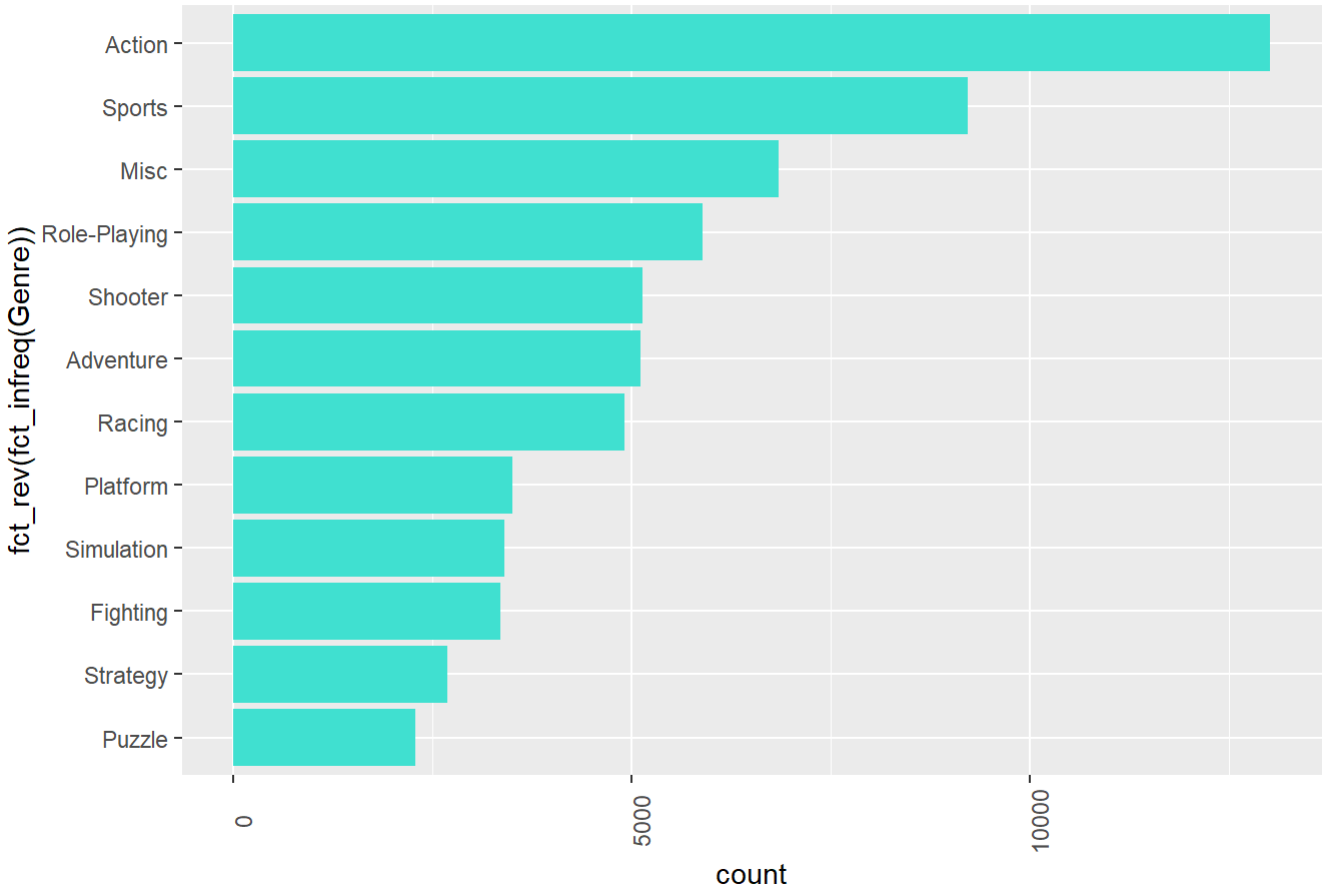


Sales in all regions except Japan demonstrate a unimodal distribution with a slight left skew and are consistent with Figure 1.1. Sales in Japan demonstrate a bimodal distribution. The mode for each region is at a similar year range of 2007 to 2009, with sales declining from apprximately 2010 onwards. The distribution of sales in Japan is more uniform and doesn't show the same growth that the other regions experienced through the 2000's however sales still decline more modestly from approximately 2010 onwards.

Genre

```
vgsales_long %>%
  group_by(Genre) %>%
  ggplot(aes(x = fct_rev(fct_infreq(Genre)))) +
  geom_bar(fill = 'turquoise') +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "Figure 1.3 Genres by count")
```

Figure 1.3 Genres by count



Of the 12 Genre's we can see in Figure 1.3 that Action is the most common Genre, followed by Sports, miscellaneous and then Role-Playing.

Part 3. Data tables of Platform and Publisher

Both Platform and Publisher variables have large unique values and so just the top 11 have been retained and the remaining levels have been collapsed into a new category of 'other' for a total of 12 categories which is consistent with the number of levels in the Genre variable.

Platform

```
# creating a new dataframe with the collapsed Platform categories
vgsalesOther <- vgsalesA %>% mutate(
  Platform_lump = fct_lump(Platform, n = 11)) # collapsing non top 11 into 'other"

# first creating a count table and then sorting
platform_table <- table(vgsalesOther$Platform_lump)

platform_table_sort <- platform_table %>%
  as.data.frame() %>%
  arrange(desc(Freq))
```

```
# creating summary data of the sales variables
vgsalesOther_table_data <- vgsalesOther %>%
  group_by(Platform_lump) %>%
  arrange(desc(NA_Sales)) %>%
  summarise(NA_mean = mean(NA_Sales),
            EU_mean = mean(EU_Sales),
            Other_mean = mean(Other_Sales),
            JP_mean = mean(JP_Sales)) %>%
  rename(Var1 = Platform_lump)


# joining the two data frames and creating the summary table
left_join(platform_table_sort,vgsalesOther_table_data, by="Var1") %>%
  kable(caption = "Table 1.2. Frequency and mean sales of games released by Platfor
m")
```

Table 1.2. Frequency and mean sales of games released by Platform

Var1	Freq	NA_mean	EU_mean	Other_mean	JP_mean
Other	2753	0.3121431	0.1488740	0.0384344	0.1913658
DS	2133	0.1821660	0.0909845	0.0282654	0.0820534
PS2	2127	0.2693559	0.1563846	0.0895487	0.0646638
PS3	1304	0.2982362	0.2610966	0.1079831	0.0607439
Wii	1290	0.3855581	0.2049225	0.0613953	0.0529302
X360	1235	0.4812389	0.2251093	0.0685587	0.0099595
PSP	1197	0.0894653	0.0561069	0.0346867	0.0634002
PS	1189	0.2815055	0.1786207	0.0342220	0.1175610
PC	943	0.0976670	0.1459597	0.0258006	0.0001803
GBA	811	0.2270284	0.0919729	0.0093835	0.0574106
XB	803	0.2267248	0.0742839	0.0105604	0.0017186
GC	542	0.2434317	0.0707011	0.0094649	0.0393727

Table 1.2 above shows that the DS Platform is the most common video game platform. It is however not the highest selling platform in North America. The highest selling platform in North America is X360 with mean North American sales of 0.481 million sales.

Publisher

```
# adding the collapsed Publisher categories
vgsalesOther <- vgsalesOther %>% mutate(
  Publisher_lump = fct_lump(Publisher, n = 11)) # collapsing non top 11 into 'other
"


# first creating a count table and then sorting
publisher_table <- table(vgsalesOther$Publisher_lump)


publisher_table_sort <- publisher_table %>%
  as.data.frame() %>%
  arrange(desc(Freq))
```

```
# creating summary data of the sales variables
PublisherOther_table_data <- vgsalesOther %>%
  group_by(Publisher_lump) %>%
  arrange(desc(NA_Sales)) %>%
  summarise(NA_mean = mean(NA_Sales),
            EU_mean = mean(EU_Sales),
            Other_mean = mean(Other_Sales),
            JP_mean = mean(JP_Sales)) %>%
  rename(Var1 = Publisher_lump)

# joining the two data frames and creating the summary table
left_join(publisher_table_sort,PublisherOther_table_data, by="Var1") %>%
  kable(caption = "Table 1.3. Frequency and mean sales of games released by Publish
er")
```

Table 1.3. Frequency and mean sales of games released by Publisher

Var1	Freq	NA_mean	EU_mean	Other_mean	JP_mean
Other	7843	0.1548935	0.0783871	0.0242076	0.0478363
Electronic Arts	1339	0.4363107	0.2743689	0.0953174	0.0104406
Activision	966	0.4410041	0.2212422	0.0774224	0.0067702
Namco Bandai Games	928	0.0747629	0.0459159	0.0157759	0.1366810
Ubisoft	918	0.2753922	0.1775926	0.0546405	0.0079847
Konami Digital Entertainment	823	0.1080316	0.0833779	0.0363426	0.1104860
THQ	712	0.2929775	0.1328652	0.0450983	0.0070365
Nintendo	696	1.7205460	0.6010057	0.1367672	0.6537213
Sony Computer Entertainment	682	0.3888856	0.2750000	0.1178886	0.1086510
Sega	632	0.1721203	0.1288133	0.0384494	0.0889082
Take-Two Interactive	412	0.5351214	0.2862864	0.1339806	0.0141505
Capcom	376	0.2086436	0.1041489	0.0394149	0.1792021

Table 1.3 above shows that Electronic Arts are the most common Publisher however Nintendo has by far the highest average number of sales in North America with 1.17 million sales.

The ‘other’ category is very large and demonstrates that there is a large spread of publishers.

Part 4, 5 and 6. Variables that will not be included in the final analysis

Variable’s were converted to the correct type earlier in Part 1.

Data for the variable ‘Publisher’ is not provided in the brief of the project and thus is not able to be used in the final prediction of North American sales for The Fatal Empire and so will be removed from the dataset.

Year is also not included in the brief and so will not form part of the modeling. However, I am choosing not to remove this variable until the pre-processing stage.

Appling changes to the dataset

```
vgsalesB <-  vgsalesA %>%
  dplyr::select(-Publisher)
vgsalesB %>%
  head()
```

```
## # A tibble: 6 x 8
##   Name            Platform  Year Genre    NA_Sales EU_Sales JP_Sales Other_Sales
##   <chr>           <fct>    <dbl> <fct>       <dbl>    <dbl>    <dbl>       <dbl>
## 1 Wii Sports      Wii       2006 Sports       41.5    29.0      3.77        8.46
## 2 Super Mario Br~ NES       1985 Platform     29.1     3.58     6.81        0.77
## 3 Mario Kart Wii  Wii       2008 Racing       15.8    12.9      3.79        3.31
## 4 Wii Sports Res~ Wii       2009 Sports       15.8    11.0      3.28        2.96
## 5 Pokemon Red/Po~ GB        1996 Role-Pl~     11.3     8.89    10.2         1
## 6 Tetris          GB        1989 Puzzle       23.2     2.26     4.22        0.58
```

We can see here that Publisher has been removed and Platform and Genre are factors, Year is numeric. Thus all variables are of the correct type.

2. Exploratory data analysis

Part 1.

Exploratory data analysis(EDA) will now be performed to firstly assess for any natural grouping in the data using parallel coordinates plot, the Principal Component Analysis (PCA) will be used to reduce the dimensiality of the data.

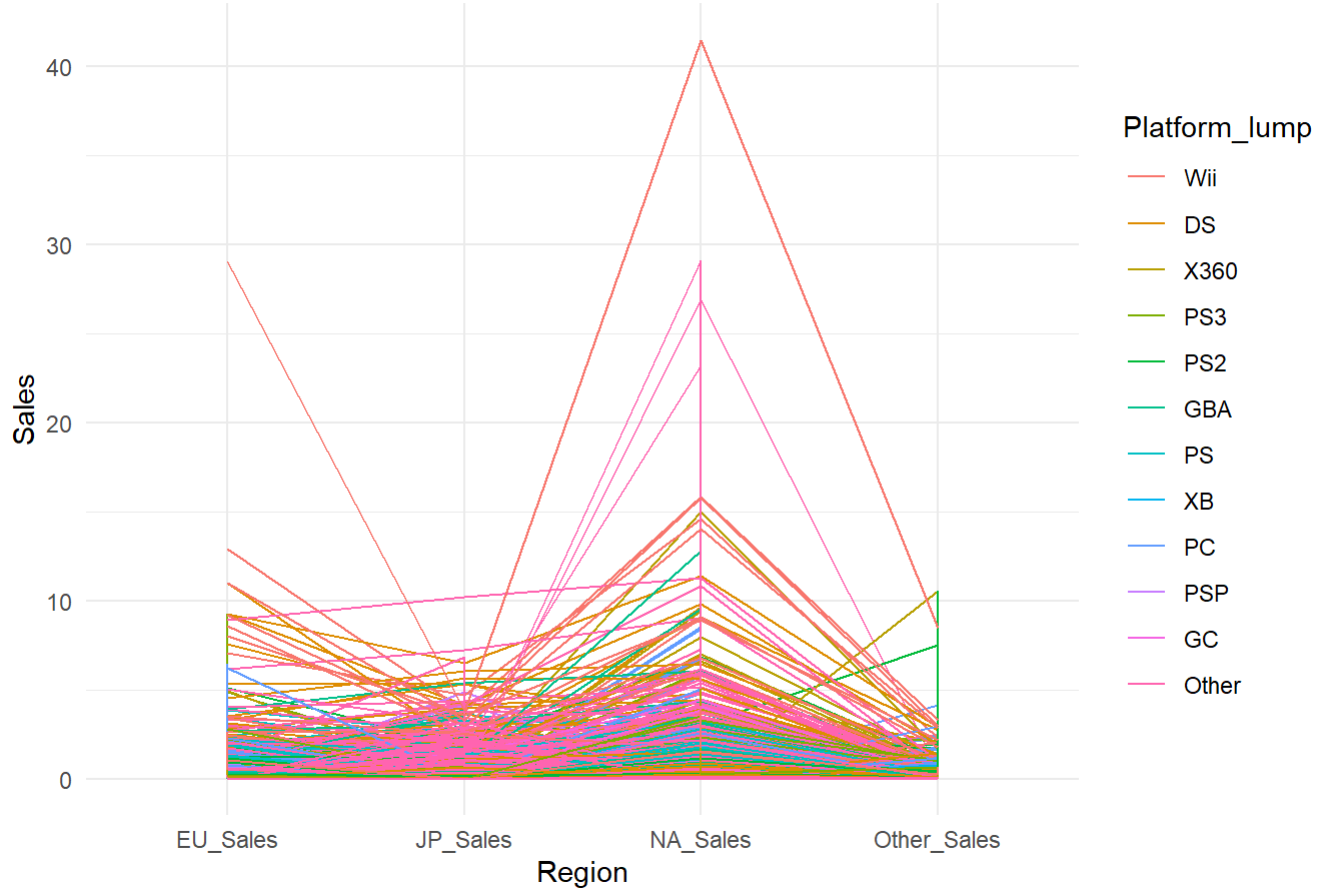
Plots will also be used to visualise the relationships of the key remaining predictor variables of Genre, Platform European, Japanese and sales from Other parts of the world.

EDA will allow us to identify the most important variables to use in our models.

Part 2. Parallel coordinates plot

```
vgsalesOther %>% # the lumped data reduces the number of levels
  pivot_longer(cols = NA_Sales:Other_Sales) %>%
  rename(Region = name,
         Sales = value) %>%
  ggplot( aes( x = Region, y = Sales, colour = Platform_lump ) ) +
  geom_line( aes( group = Name) ) +
  theme_minimal() +
  labs(title = "Figure 2.1. Parallel Coordinates plot for Sales by Region")
```

Figure 2.1. Parallel Coordinates plot for Sales by Region



The parallel coordinates plot (Figure 2.1) is not very conclusive. There are no obvious groupings mostly due to the large number of levels despite the lumping of the data. It can be seen that there are more outliers in the North American sales data especially, but also the European sales, with fewer in the Japanese and Other sales.

Part 3. Principal Component Analysis (PCA)

We are going to consider the categorical variables of Genre and Platform and all the numerical variables of EU_Sales, JP_Sales and Other_Sales. NA_Sales is the outcome variable. I will be excluding Year and Name.

Genre and Platform will need to be converted to dummy variables with step_dummy and the function all_nominal_predictors will select for these. The numeric predictors will need to be normalised so that they are on the same scale using step_normalize and the function all_predictors will select these and the new dummy variables.

Finally the pca step can be applied to all the predictors variables.

Recipe

```
vgs_recipe <- recipe(NA_Sales ~ EU_Sales + JP_Sales + Other_Sales + Genre + Platfor
m, data = vgsalesB) %>%

  step_dummy(all_nominal_predictors()) %>% # convert categorical variabels to a dum
my variable

  step_normalize(all_predictors()) %>% # normalise our predictors

  step_pca(all_predictors()) # do the PCA

vgs_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          5
##
## Operations:
##
## Dummy variables from all_nominal_predictors()
## Centering and scaling for all_predictors()
## No PCA components were extracted.
```

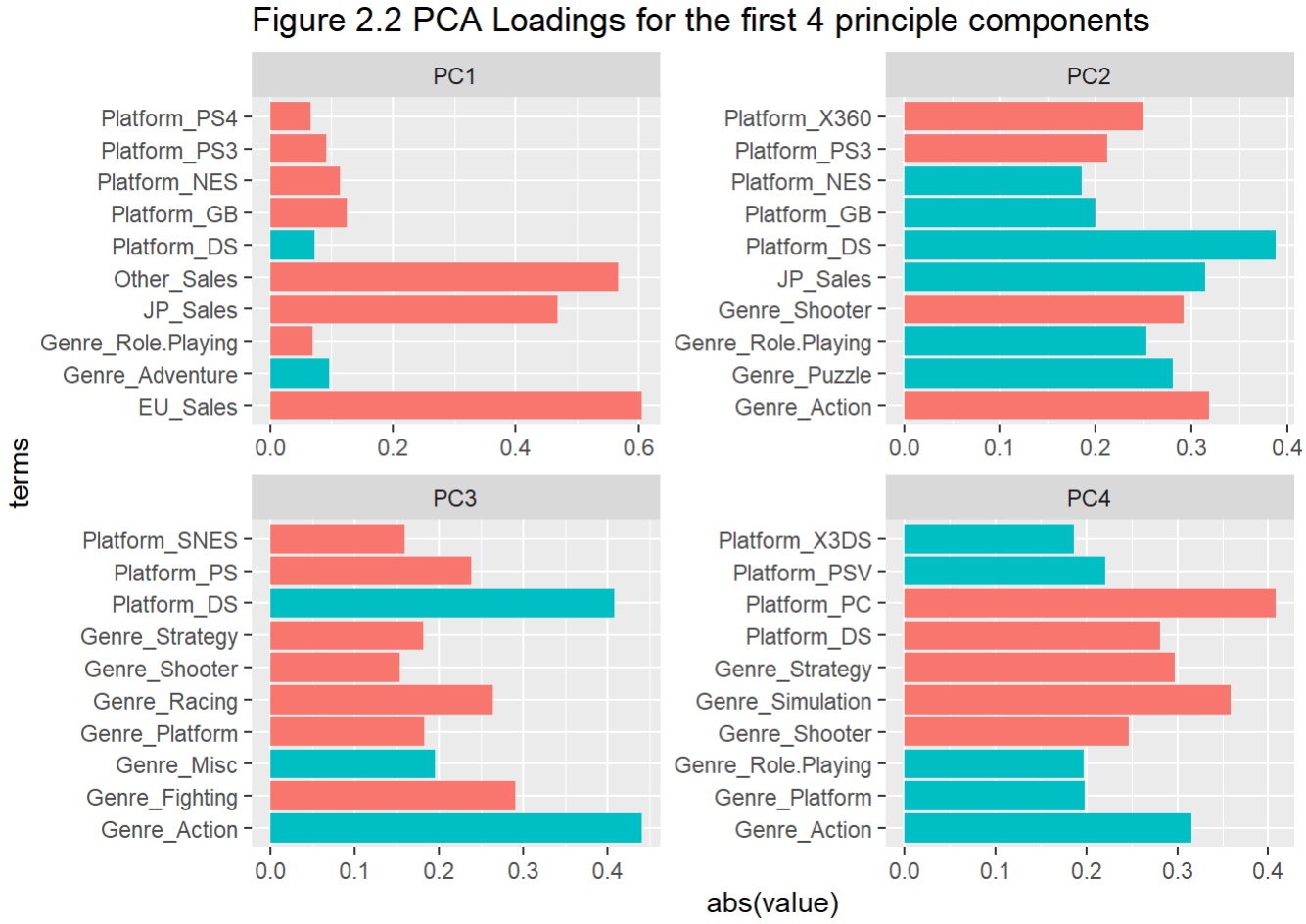
Prep the recipe

```
vgs_prepped <- vgs_recipe %>%
  prep()
vgs_prepped
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          5
##
## Training data contained 16327 data points and no missing data.
##
## Operations:
##
## Dummy variables from Genre, Platform [trained]
## Centering and scaling for EU_Sales, JP_Sales, Other_Sales, Genre_Platform... [tr
ained]
## PCA extraction with EU_Sales, JP_Sales, Other_Sales, Genre_Platform,... [trained
]
```

We can see that the dummy variables have been created and normalisation of the numeric and dummy variables has occured correctly. There are 5 predictors and 1 outcome.

Plots to visualise the important variables in each of the first 4 componenets

```
tidy( vgs_prepped, 3 ) %>% # pca is step 3 in the recipe
  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 = F) +
  facet_wrap( ~ component, scales = "free") +
  ylab(NULL) %>% # We do not need the y axis label.
  labs(title = "Figure 2.2 PCA Loadings for the first 4 principle components")
```



The above plot shows EU_Sales followed by Other_Sales and JP_Sales have the greatest influence in PC1 and the DS platform the most influence in PC2. For PC3 it is Genre_Action and in PC4, Platform_PC has the greatest contribution.

Getting the principle components

```
vgs_pca <- juice(vgs_prepped)
vgs_pca
## # A tibble: 16,327 x 6
##    NA_Sales     PC1     PC2      PC3     PC4     PC5
```

```
##       <dbl>  <dbl>  <dbl>    <dbl>   <dbl>  <dbl>
##  1     41.5 -65.0  -6.93   7.77    -11.3    0.463
##  2     29.1 -18.1   8.69  -3.29      3.80   0.556
##  3     15.8 -30.5  -1.20   1.51     -4.20   0.490
##  4     15.8 -26.4  -0.598  2.20     -3.43  -0.230
##  5     11.3 -30.4  11.6   -2.37      2.28  -3.57
##  6     23.2 -11.8   7.58  -0.702     0.757 -0.411
##  7     11.4 -29.0   4.69   1.34     -1.74   0.876
##  8     14.0 -23.3  -0.190  2.62     -3.34   0.792
##  9     14.6 -22.1   2.53  -0.00911  -0.608  0.612
## 10     26.9  -3.79  1.15  -1.67      0.549  0.641
## # ... with 16,317 more rows
```

Variance Explained - Getting the standard deviation and variance explained

```
sdev <- vgs_prepped$steps[[3]]$res$sdev


ve <- sdev^2 / sum(sdev^2)
ve
##  [1] 0.047495920 0.034268756 0.030956539 0.030484353 0.029822415 0.029068348
##  [7] 0.026824427 0.026321548 0.026105531 0.025795303 0.025118481 0.024640274
## [13] 0.024379827 0.024027807 0.023801842 0.023654588 0.023584022 0.023470030
## [19] 0.023205657 0.023111184 0.023026456 0.022913536 0.022891522 0.022830949
## [25] 0.022819574 0.022774772 0.022753205 0.022746957 0.022742121 0.022564018
## [31] 0.022478541 0.022362146 0.022143331 0.021885678 0.021278696 0.020135386
## [37] 0.019377596 0.018987326 0.018031165 0.016975432 0.011274555 0.005522251
## [43] 0.003458879 0.001889056
```

There are 44 principle components.

Proportion of variance explained

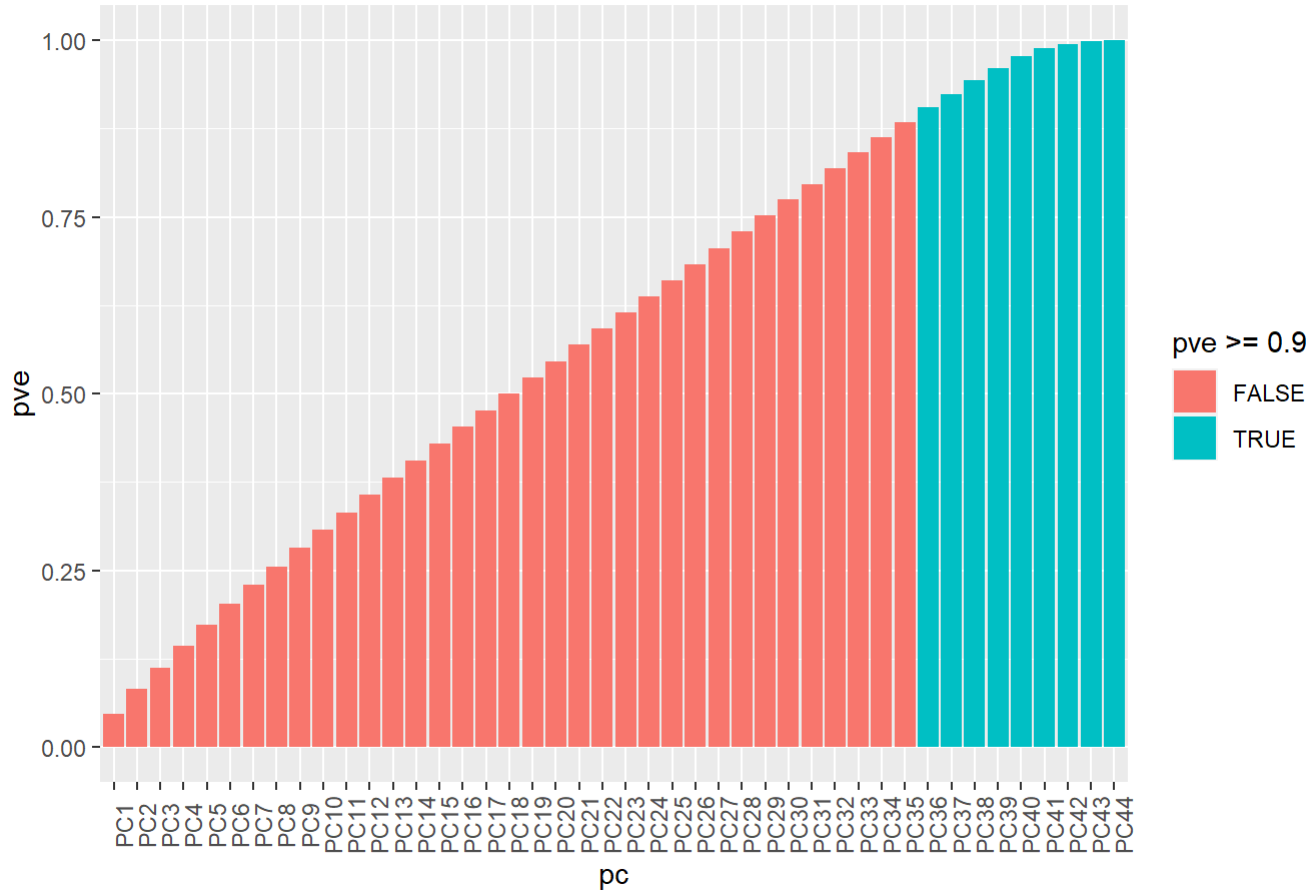
```
PC.pve <- tibble(
  pc = fct_inorder( str_c("PC", 1:44) ), # This will create PC1, PC2, ..., PC44
  pve = cumsum( ve ) # Takes the cumulative sum to get our PVE
)
PC.pve
## # A tibble: 44 x 2
##    pc       pve
##    <fct>  <dbl>
##  1 PC1   0.0475
```

```
##  2 PC2    0.0818
##  3 PC3    0.113
##  4 PC4    0.143
##  5 PC5    0.173
##  6 PC6    0.202
##  7 PC7    0.229
##  8 PC8    0.255
##  9 PC9    0.281
## 10 PC10   0.307
## # ... with 34 more rows
```

Visualising the pve for 90%

```
PC.pve %>%
  ggplot( aes( x = pc,
               y = pve,
               fill = pve >= 0.9 ) ) + # Finding the PC's that explains 90% of our
variance
  geom_col() +
  theme( axis.text.x = element_text( angle = 90 ) ) + #rotate the x-axis labels
  labs(title = "Figure 2.3 Cumulative PVE for each principle component")
```

Figure 2.3 Cumulative PVE for each principle component



This chart shows that 90% of variance is explained by the first 35 components. Let's check this.

```
PC.pve %>%
  filter( pve >= 0.9)  # Let's look at those explaining 90% or more
## # A tibble: 9 x 2
##   pc      pve
##   <fct> <dbl>
## 1 PC36  0.904
## 2 PC37  0.924
## 3 PC38  0.943
## 4 PC39  0.961
## 5 PC40  0.978
## 6 PC41  0.989
## 7 PC42  0.995
## 8 PC43  0.998
## 9 PC44  1
```

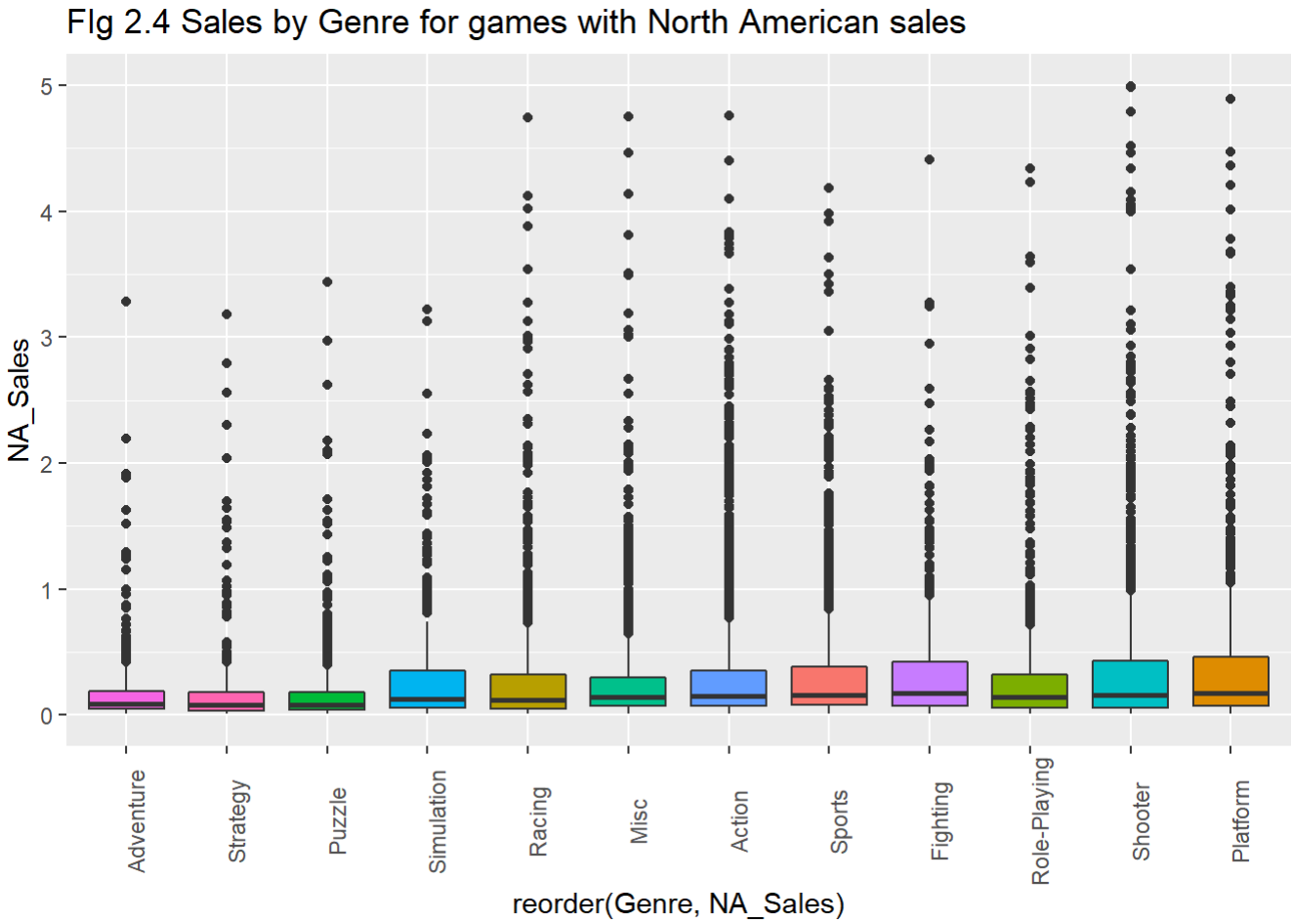
To get at least 90% we need 36 components. From a total of 44 components, 36 is an 18% reduction, which is worthwhile.

Part 4. Bivariate analysis

North American Sales by Genre

Filtering for Genre's with North American sales greater than 0. There are a lot of 0 values in the data. This is to assist in visualising those Genre's that actually have sales in North America.

```
vgsalesB %>%
  filter(NA_Sales > 0) %>%
  ggplot(aes(x = reorder(Genre, NA_Sales), y = NA_Sales, fill = Genre)) +
  geom_boxplot(show.legend = FALSE) +
  ylim(0,5) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "FIg 2.4 Sales by Genre for games with North American sales")
## Warning: Removed 58 rows containing non-finite values (stat_boxplot).
```

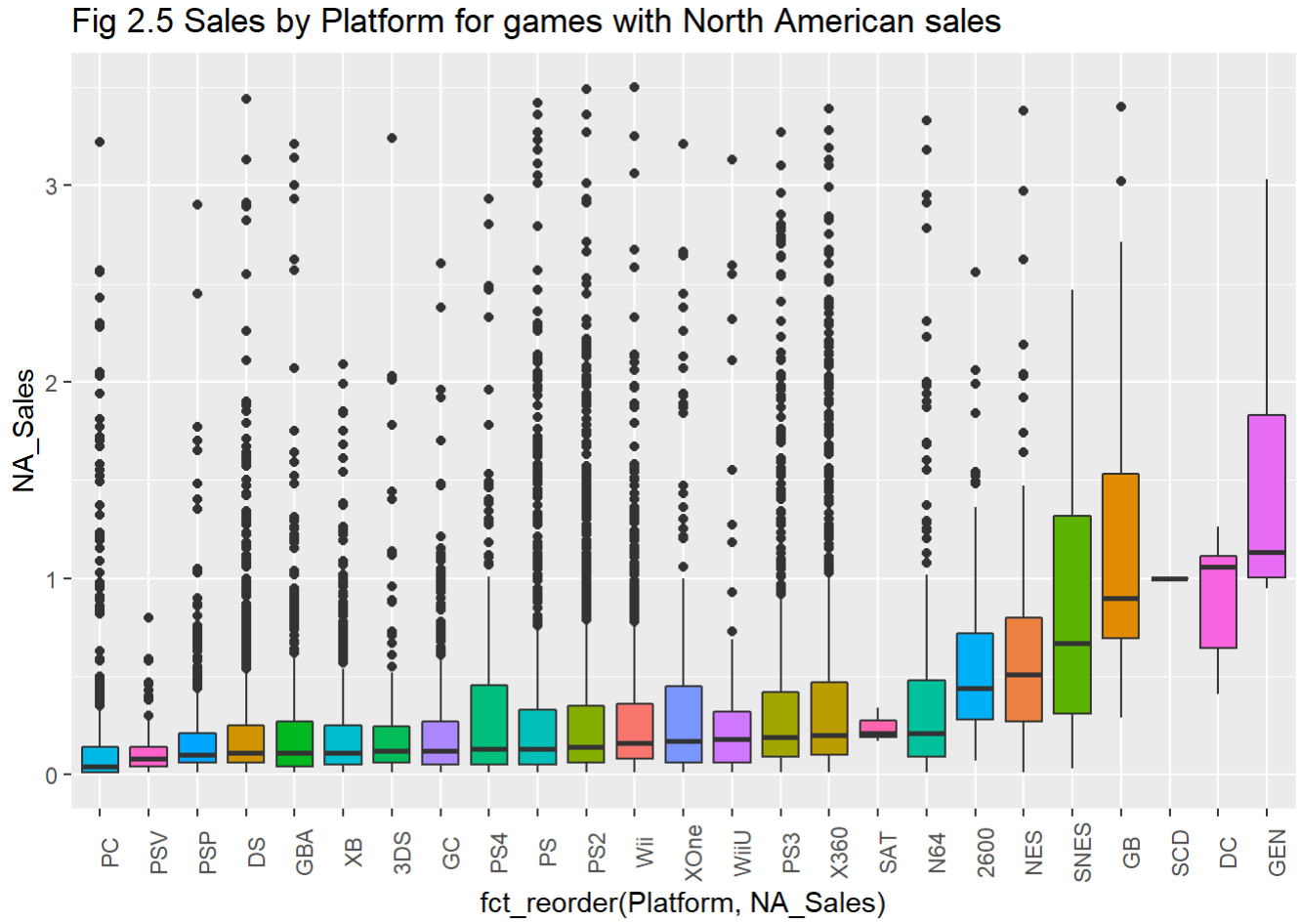


This chart shows only slight variation of NA_Sales by Genre when considering the median values and inter-quartile ranges. Shooter, Role-Playing and Fighting make up the top categories with the generic Platform ordered as the highest. Importantly there are a lot of outliers and long tails for each Genre. Median values appear to range from just above zero to approximately 0.15 million copies.

North American sales by Platform

```
vgsalesB %>%
  filter(NA_Sales > 0) %>%
```

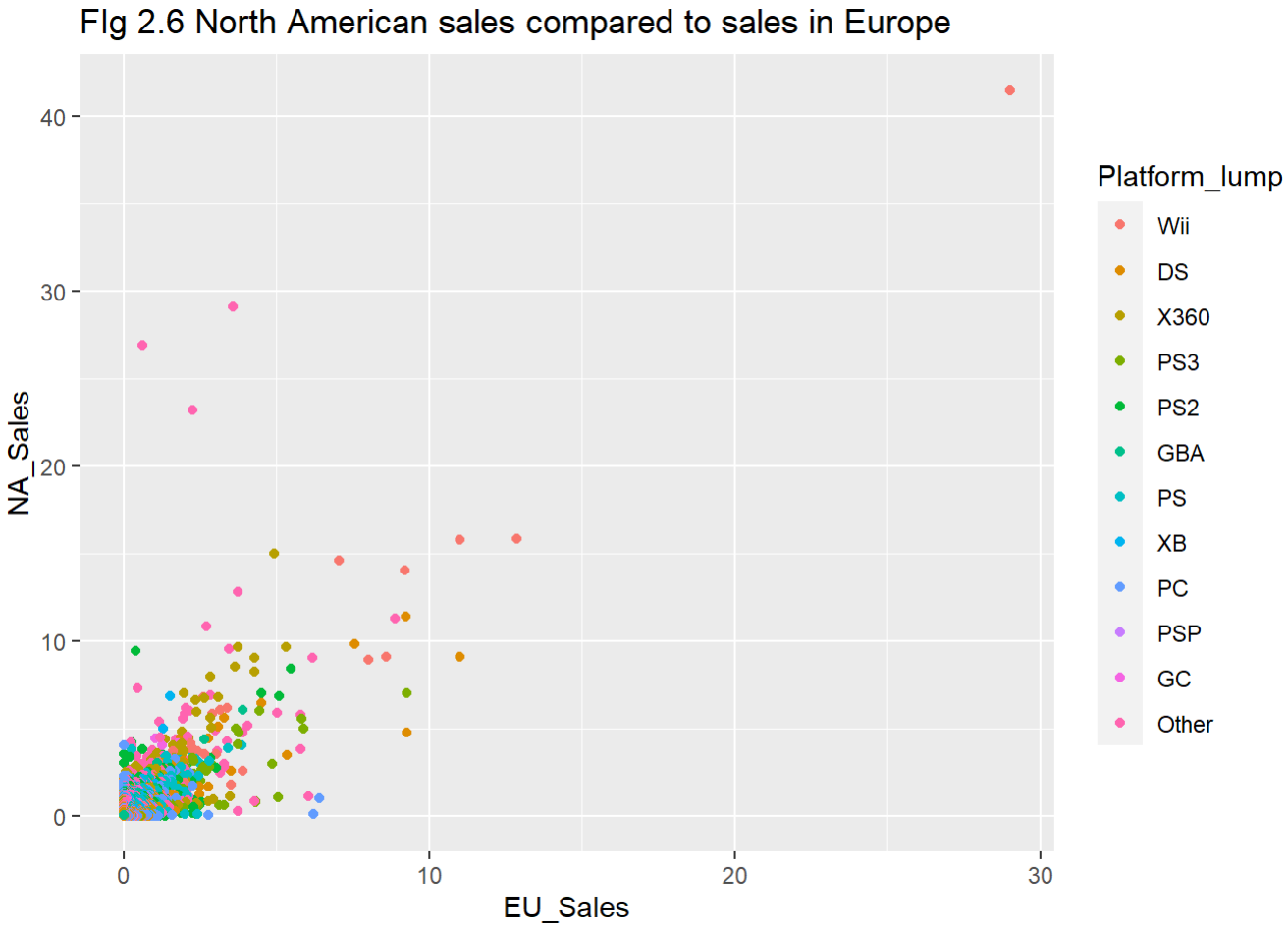
```
  ggplot(aes(x = fct_reorder(Platform, NA_Sales), y = NA_Sales, fill = Platform)) +
  geom_boxplot(show.legend = FALSE) +
  ylim(0,3.5) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "Fig 2.5 Sales by Platform for games with North American sales")
## Warning: Removed 109 rows containing non-finite values (stat_boxplot).
```



There is much more variation of North American sales by Platform. The top 7 median values showing considerably more sales than the rest. Many of the Platforms have long tails and or outliers, but particularly PS, PS2, PS3, Wii and X360. This suggests these platforms have some very high selling games on their platform.

North American Sales by EU_Sales

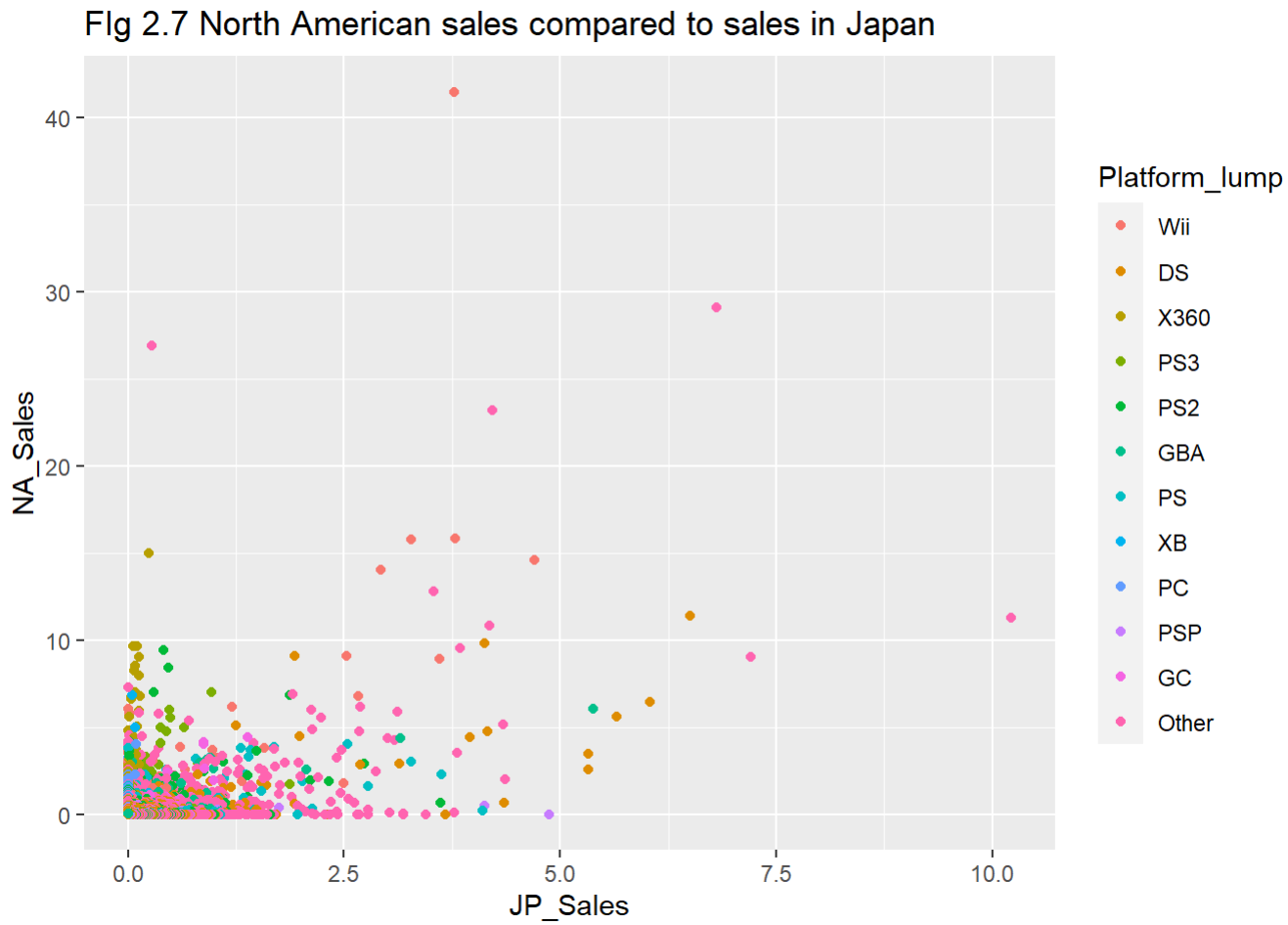
```
vgsalesOther %>%
  ggplot(aes(y = NA_Sales, x = EU_Sales, col = Platform_lump)) +
  geom_point() +
  labs(title = "FIg 2.6 North American sales compared to sales in Europe")
```



There is a moderate positive linear relationship of sales in Europe with North American sales. There are 3 notable outliers from the Other category, with the highest North American sales, but relatively lower European sales.

North American Sales by JP_Sales

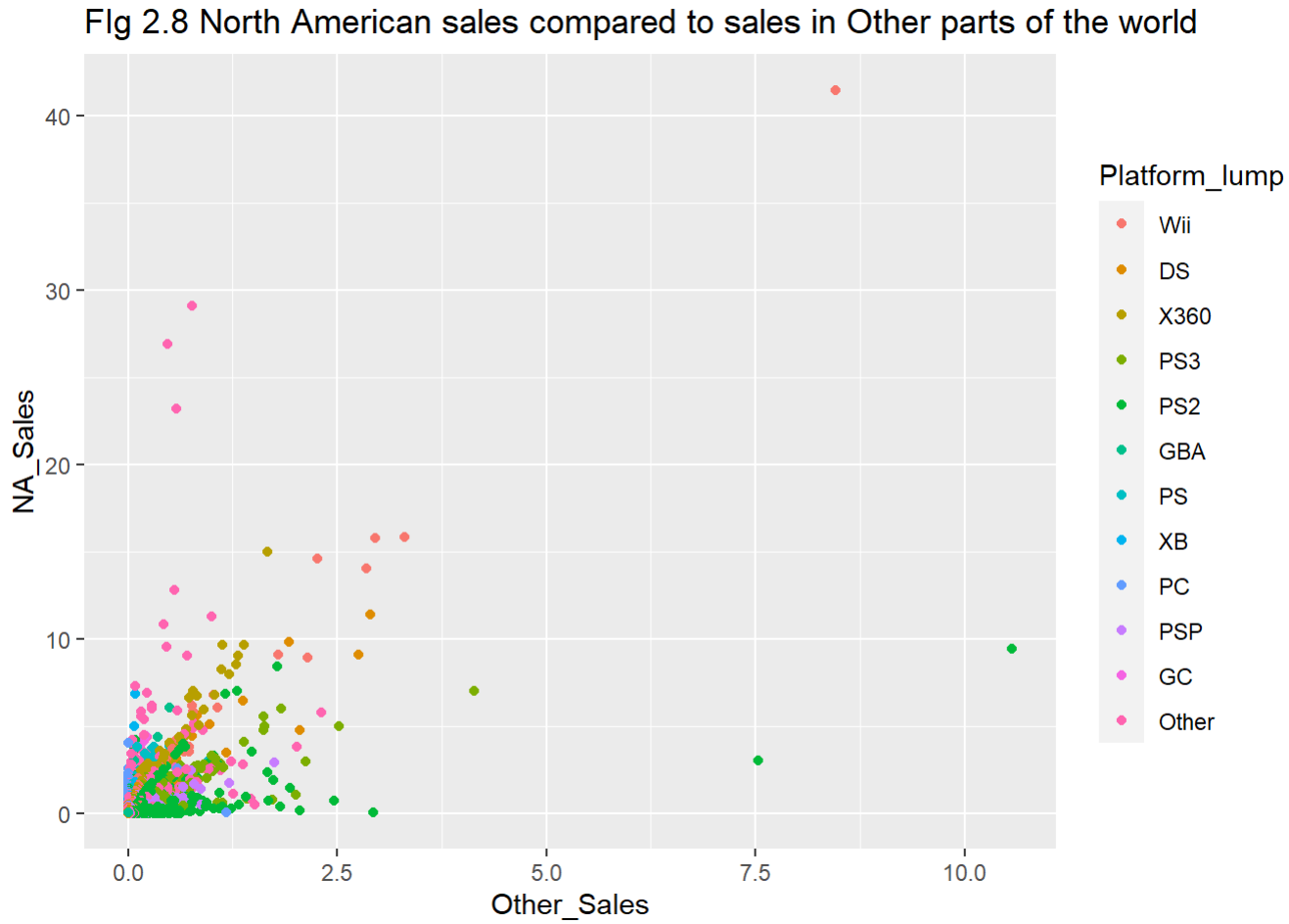
```
vgsalesOther %>%
  ggplot(aes(y = NA_Sales, x = JP_Sales, col = Platform_lump)) +
  geom_point() +
  labs(title = "FIg 2.7 North American sales compared to sales in Japan")
```



There is a much weaker positive relationship of North American sales with Japanese sales. There are some outliers in both the high sales of North America as well as Japanese sales.

North American Sales by Other_Sales

```
vgsalesOther %>%
  ggplot(aes(y = NA_Sales, x = Other_Sales, col = Platform_lump)) +
  geom_point() +
  labs(title = "FIg 2.8 North American sales compared to sales in Other parts of th
e world")
```

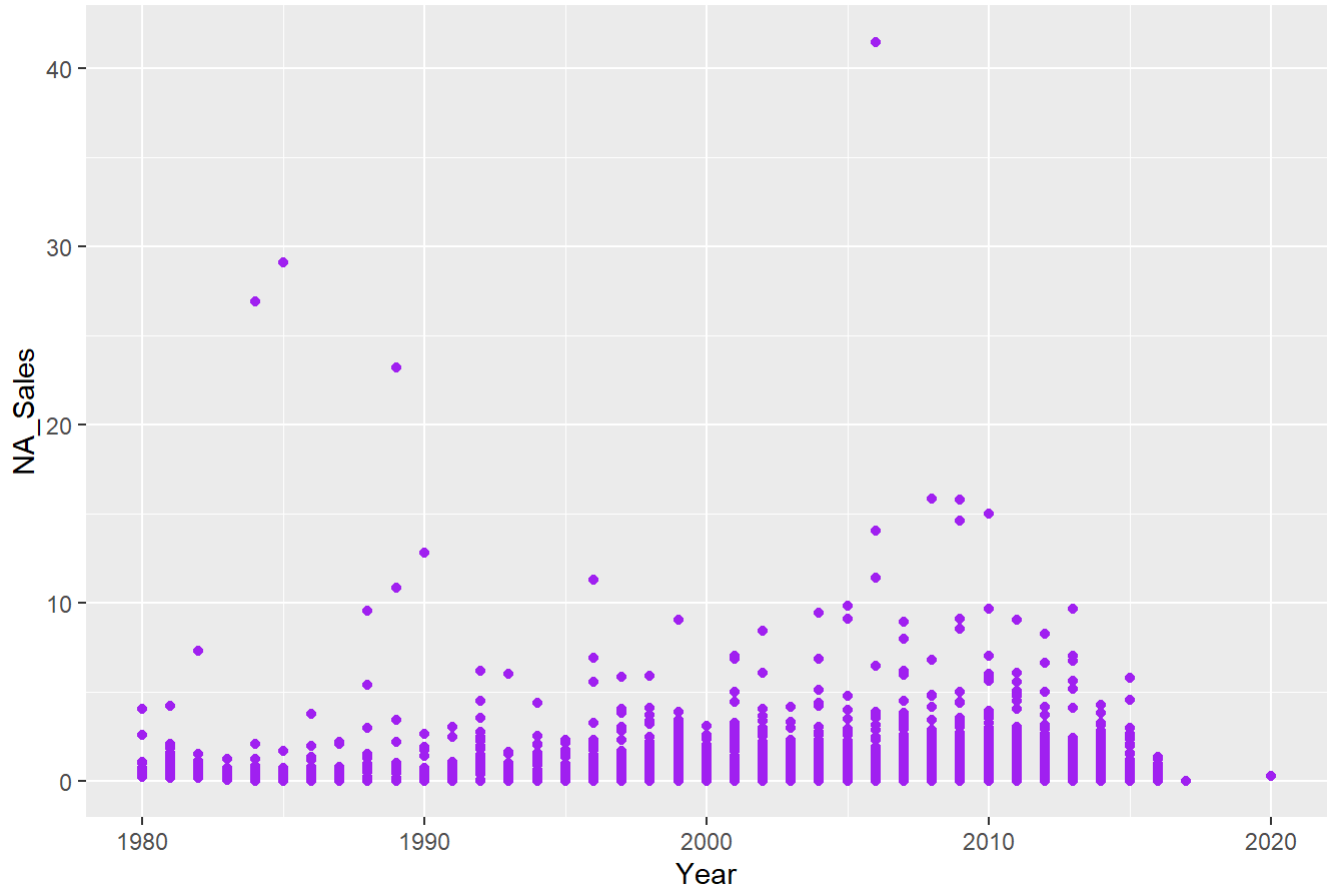


This plot shows a moderately strong linear relationship of North American sales with sales from Other parts of the world. Outliers are present agin in both high North American sales and Other sales.

North American Sales by Year

```
vgsalesB %>%
  ggplot(aes(y = NA_Sales, x = Year)) +
  geom_point(col = "Purple") +
  labs(title = "FIg 2.9 North American sales by Year")
```

FIg 2.9 North American sales by Year



This chart shows a gradual increase in sales in North America up to 2010 and then a relatively steep decline to 2017. There are notable outliers across multiple years suggesting there are particular games that are far more popular than the rest. The lack of sales from 2017 has been discussed as potential incomplete data and thus needs to be interpreted with caution.

EDA Summary

Analysis of the data suggests the strongest relationship of North American Sales is a positive one with sales from firstly Europe, then Other parts of the world and thirdly Japan. This is seen in the scatterplots Fgures 2.6, 2.7, 2.8 and confirmed by the PCA loadings of PC1 in Fig 2.2.

The DS Platform and Action Genre are the next most influential predictors as shown in the PCA loadings in PC2 and PC3 but is less clear in the boxplots of Figures 2.5 and 2.4 respectively.

3. Data Pre-processing

Part 1. Split for testing and training sets

```
set.seed(1234)
vgs_split <- initial_split( vgsalesB )
vgs_train <- training( vgs_split )
vgs_test <- testing( vgs_split )
```

```
vgs_split
## <Analysis/Assess/Total>
## <12245/4082/16327>
```

There are 12245 observations in the training set and 4082 in the testing set.

Part 2. Removal of variables

The variable Publisher has already been removed, so does not need to be removed in this section. I will now remove the Name variable as it is only an ID column, and Year as it is not necessary for the modeling.

Part 3. Apply recipe to training set

step_rm is used firstly to remove the variables Name and Year.

The categorical predictors of Genre and Platform need to be processed with step_dummy, with the use of all_nominal predictors() function.

All the predictors can then be normalised to enable the modeling process to work best with step_normalize and using the all_predictors function to make sure the outcome is not normalised.

step_pca will be used with the number of components defined as 36 with in num_comp.

```
vgs_recipe2 <- recipe(NA_Sales ~ ., data = vgs_train) %>%
  step_rm(Name, Year) %>%
  step_dummy(all_nominal_predictors()) %>% # convert categorical variables to a dum
my variable
  step_normalize(all_predictors()) %>% # normalise our predictors
  step_pca(all_predictors(), num_comp = 36) %>%
  prep()
```

Now to juice the recipe.

```
vgs_preproc <- vgs_recipe2 %>%
  juice()
vgs_preproc %>%
  head()
## # A tibble: 6 x 37
##   NA_Sales   PC01   PC02    PC03    PC04    PC05   PC06   PC07    PC08   PC09
##      <dbl>  <dbl>  <dbl>   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>   <dbl>  <dbl>
## 1     0.17  0.169 -1.94  -0.164  -0.664   0.108   0.195 -2.15   1.31   -0.664
## 2     0.08 -0.631  0.384  0.0778  0.0697 -0.462  -1.67   0.283 -0.310  -1.52
## 3     0.08 -0.156  1.45  -0.491   0.612  -2.24   -1.29  -0.494  1.19   -0.796
## 4     0    -0.176  0.901  1.14    0.850  -0.0163  0.259  0.169  1.54    2.09
## 5     0     0.366  1.63  -2.78    1.24   -0.359   0.102 -2.44  -0.542   2.79
## 6     0    -0.442  0.302 -0.379   0.193  -0.190  -2.66   1.46   0.0659 -0.906
## # ... with 27 more variables: PC10 <dbl>, PC11 <dbl>, PC12 <dbl>, PC13 <dbl>,
```

```
## #   PC14 <dbl>, PC15 <dbl>, PC16 <dbl>, PC17 <dbl>, PC18 <dbl>, PC19 <dbl>,
## #   PC20 <dbl>, PC21 <dbl>, PC22 <dbl>, PC23 <dbl>, PC24 <dbl>, PC25 <dbl>,
## #   PC26 <dbl>, PC27 <dbl>, PC28 <dbl>, PC29 <dbl>, PC30 <dbl>, PC31 <dbl>,
## #   PC32 <dbl>, PC33 <dbl>, PC34 <dbl>, PC35 <dbl>, PC36 <dbl>
```

Juicing (or baking) the recipe applies the estimates produced by the recipe to the training data creating new values for the variables, in this case for the principal components. We can see that 36 components have been processed.

Checking that normalisation has occurred:

```
vgs_preproc %>%
  select_if( is.numeric ) %>% # Get only the numeric variables
  summarise_all( list( mean = mean, sd = sd ) ) %>%  # Get the standard deviation a
nd the mean of each variable
  pivot_longer( cols = everything(), names_to = "measure", values_to = "value" ) %>
% #make it look prettier
  mutate( value = round( value, 2 )) # Finally, round all our values to two decimal
places
## # A tibble: 74 x 2
##    measure       value
##    <chr>         <dbl>
##  1 NA_Sales_mean  0.26
##  2 PC01_mean      0
##  3 PC02_mean      0
##  4 PC03_mean      0
##  5 PC04_mean      0
##  6 PC05_mean      0
##  7 PC06_mean      0
##  8 PC07_mean      0
##  9 PC08_mean      0
## 10 PC09_mean      0
## # ... with 64 more rows
```

This shows that everything except NA_Sales has a mean of 0 and a standard deviation of 1 and so the normalisation has occured.

4. Model Fitting

Now that the data has been pre-processed the two different models of Lasso regression and Random forest will be fitted. Firstly, model specifications will be created for each model. Then bootstrapped data will be created for the resampling as well as a penalty grid for each model. The tuning process will identify the penalty value for the Lasso regression model and combination of mtry and min_n values for the Random forest model that give the lowest residual mean square error (RMSE).

Seeds will be set to ensure reproducibility.

These parameters will be used to finalise each model ready for comparison using cross-validated pre-processed training data. The model with the best RMSE and R squared will then be chosen and fitted.

Part 1. Lasso regression

Model specification

```
lasso_spec <- linear_reg( mode = "regression", mixture = 1, penalty = tune()  ) %>%
  set_engine( "glmnet")
lasso_spec
## Linear Regression Model Specification (regression)
##
## Main Arguments:
##   penalty = tune()
##   mixture = 1
##
## Computational engine: glmnet
```

Setting a seed and creating the bootstraps

```
set.seed( 1234 )
vgs_boots <- bootstraps( vgs_preproc, times = 10)
vgs_boots
## # Bootstrap sampling
## # A tibble: 10 x 2
##    splits               id
##    <list>               <chr>
##  1 <split [12245/4558]> Bootstrap01
##  2 <split [12245/4534]> Bootstrap02
##  3 <split [12245/4511]> Bootstrap03
##  4 <split [12245/4576]> Bootstrap04
##  5 <split [12245/4503]> Bootstrap05
##  6 <split [12245/4495]> Bootstrap06
##  7 <split [12245/4527]> Bootstrap07
##  8 <split [12245/4524]> Bootstrap08
##  9 <split [12245/4516]> Bootstrap09
## 10 <split [12245/4531]> Bootstrap10
```

Creating the penalty grid

```
lambda_grid <- grid_regular( penalty(),
                             levels = 50)
```

```
lambda_grid
```

```
## # A tibble: 50 x 1
##     penalty
##       <dbl>
##  1 1   e-10
##  2 1.60e-10
##  3 2.56e-10
##  4 4.09e-10
##  5 6.55e-10
##  6 1.05e- 9
##  7 1.68e- 9
##  8 2.68e- 9
##  9 4.29e- 9
## 10 6.87e- 9
## # ... with 40 more rows
```

Tuning

```
set.seed(4321)
lasso_grid <- tune_grid(lasso_spec,
                        NA_Sales ~ .,
                        resamples = vgs_boots,
                        grid = lambda_grid)
```

```
## Warning: package 'rlang' was built under R version 4.0.5
## Warning: package 'vctrs' was built under R version 4.0.5
## ! Bootstrap01: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap02: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap03: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap04: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap05: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap06: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap07: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap08: internal: A correlation computation is required, but `estimate` i
s const...
## ! Bootstrap09: internal: A correlation computation is required, but `estimate` i
s const...
```

```
## ! Bootstrap10: internal: A correlation computation is required, but `estimate` i
s const...
```

Visualising the metrics

```
lasso_grid %>%
  collect_metrics() %>%
  ggplot( aes( penalty, mean, color = .metric ) ) +
  geom_line() +
  facet_wrap( ~.metric, scales = "free", nrow = 2) +
  scale_x_log10() + # it looks better on a log_10 scale
  labs(title = "Figure 4.1. Lasso metrics for different penalty values")
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

Figure 4.1. Lasso metrics for different penalty values

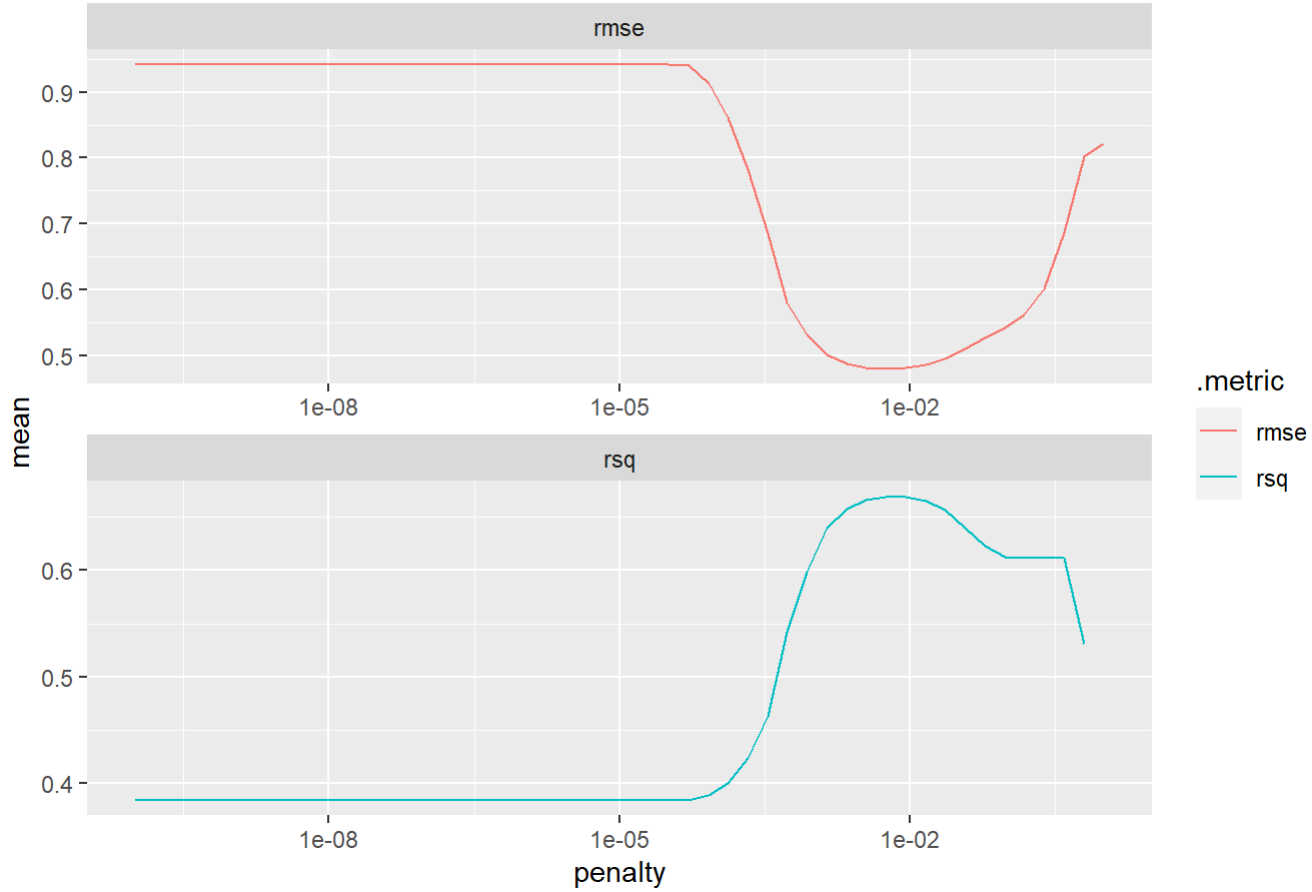


Figure 3.1 demonstates that the penalty value with the lowest rmse also correlates to the highest R squared value. The select_best function will be used to extract the best value.

```
best_lasso_rmse <- select_best( lasso_grid, "rmse" )
best_lasso_rmse
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
```

```
## 1 0.00569 Preprocessor1_Model39
```

Using the best penalty value to finalise the model.

```
final_lasso <- finalize_model( lasso_spec, best_lasso_rmse )
final_lasso
## Linear Regression Model Specification (regression)
##
## Main Arguments:
##   penalty = 0.00568986602901831
##   mixture = 1
##
## Computational engine: glmnet
```

We can see that the best penalty value of 0.00569 has been used in the Lasso model.

Part 2. Random Forest model

Model specification

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

Creating a grid to tune over

```
rand_spec_grid <- grid_regular(
  finalize( mtry(),
            vgs_preproc %>%
              dplyr::select( - NA_Sales ) ),
  min_n(),
  levels = 5 )
rand_spec_grid
## # A tibble: 25 x 2
##     mtry min_n
##    <int> <int>
##  1     1     2
##  2     9     2
##  3    18     2
##  4    27     2
##  5    36     2
##  6     1    11
```

```
##  7     9    11
##  8    18    11
##  9    27    11
## 10    36    11
## # ... with 15 more rows
```

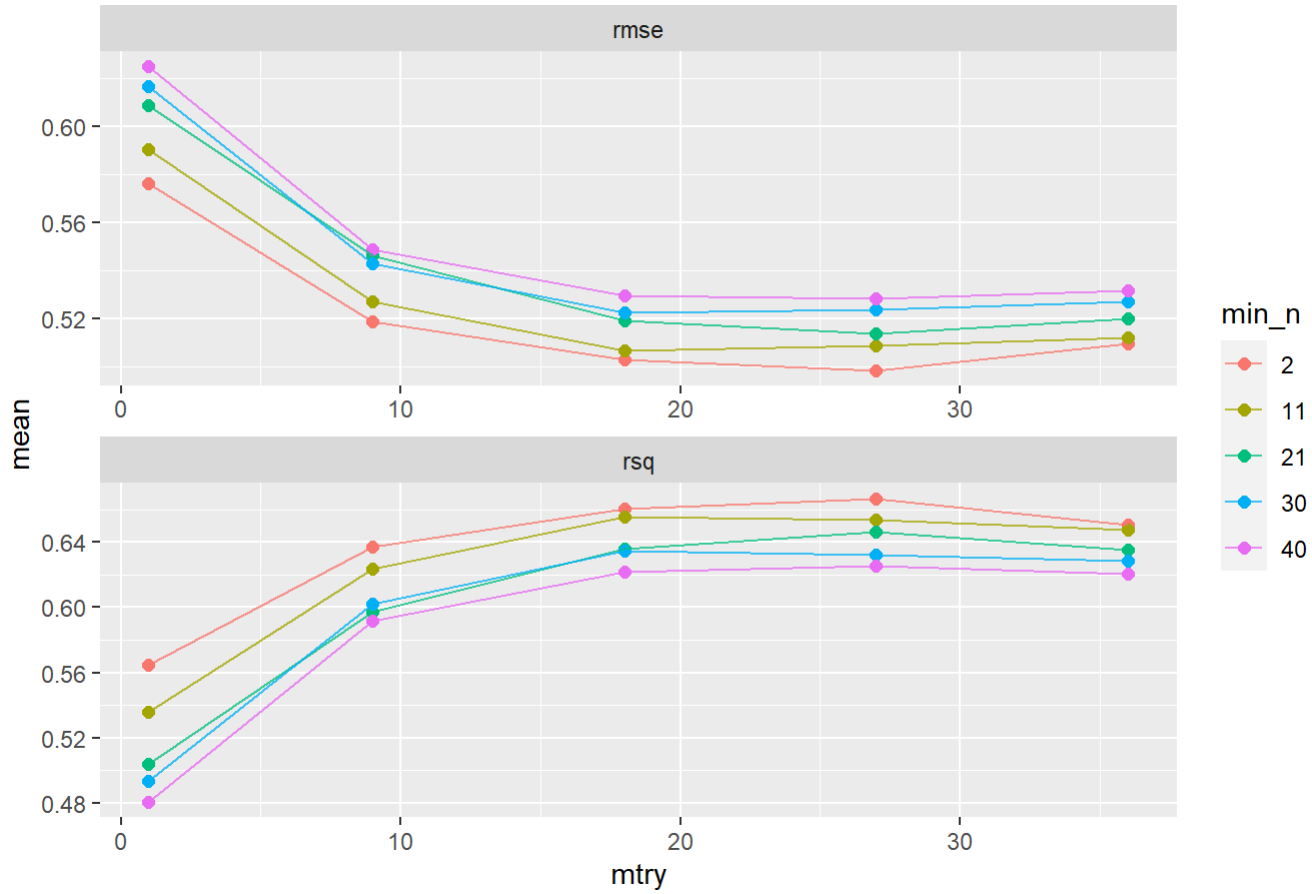
Tuning

```
set.seed( 4321 )
rf_grid <- tune_grid( rf_spec,
                      NA_Sales ~ .,
                      vgs_boots,
                      grid = rand_spec_grid )
```

Visualising the metrics

```
rf_grid %>%
  collect_metrics() %>%
  mutate( min_n = as.factor( min_n ) ) %>%
  ggplot( aes( x = mtry, y = mean, colour = min_n ) ) +
  geom_point( size = 2 ) +
  geom_line( alpha = 0.75 ) +
  facet_wrap( ~ .metric, scales = "free", nrow = 3 ) +
  labs(title = "Figure 4.2. Random forest metrics for different penalty values")
```

Figure 4.2. Random forest metrics for different penalty values



The plot suggests mtry of approximately 27 and a min_n of 2 gives the best rmse and R squared. Using select_best function to confirm.

```
best_rf_rmse <- select_best( rf_grid, "rmse" )
best_rf_rmse
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1    27     2 Preprocessor1_Model04
```

Finalise the random forest model using the best mtry of 27 and min_n of 2.

```
final_rf <- finalize_model( rf_spec, best_rf_rmse )
final_rf
## Random Forest Model Specification (regression)
##
## Main Arguments:
##   mtry = 27
##   trees = 100
##   min_n = 2
##
```

```
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

Step 3. Comparing models using cross-validation

```
set.seed( 4321 )
vgs_cv <- vfold_cv( vgs_preproc, v = 10)
```

Cross validation estimates for the LASSO regression model

```
lasso_cv <- fit_resamples( final_lasso, NA_Sales ~ ., vgs_cv )
lasso_cv %>%
  collect_metrics()
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.476    10  0.0443 Preprocessor1_Model1
## 2 rsq     standard   0.638    10  0.0332 Preprocessor1_Model1
```

RMSE of 0.476 is not great, with better R squared of 0.638.

Cross validation estimates for the RANDOM FOREST model

```
set.seed(4321)
rf_cv <- fit_resamples(final_rf, NA_Sales ~ ., vgs_cv )
rf_cv %>%
  collect_metrics()
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.422    10  0.0677 Preprocessor1_Model1
## 2 rsq     standard   0.737    10  0.0191 Preprocessor1_Model1
```

The Random Forest model has an RMSE of 0.422 which is 0.054 million copies less than the Lasso regression model and the R squared of 0.737 is better by approx. 0.1 which is very good.

Thus the winning model is the Random Forest.

Fitting the best model - RANDOM FOREST

```
set.seed( 1234 )
vgsales_rf <- final_rf %>%
```

```
  fit( NA_Sales ~ . , data = vgs_preproc )
```

5. Model Evaluation

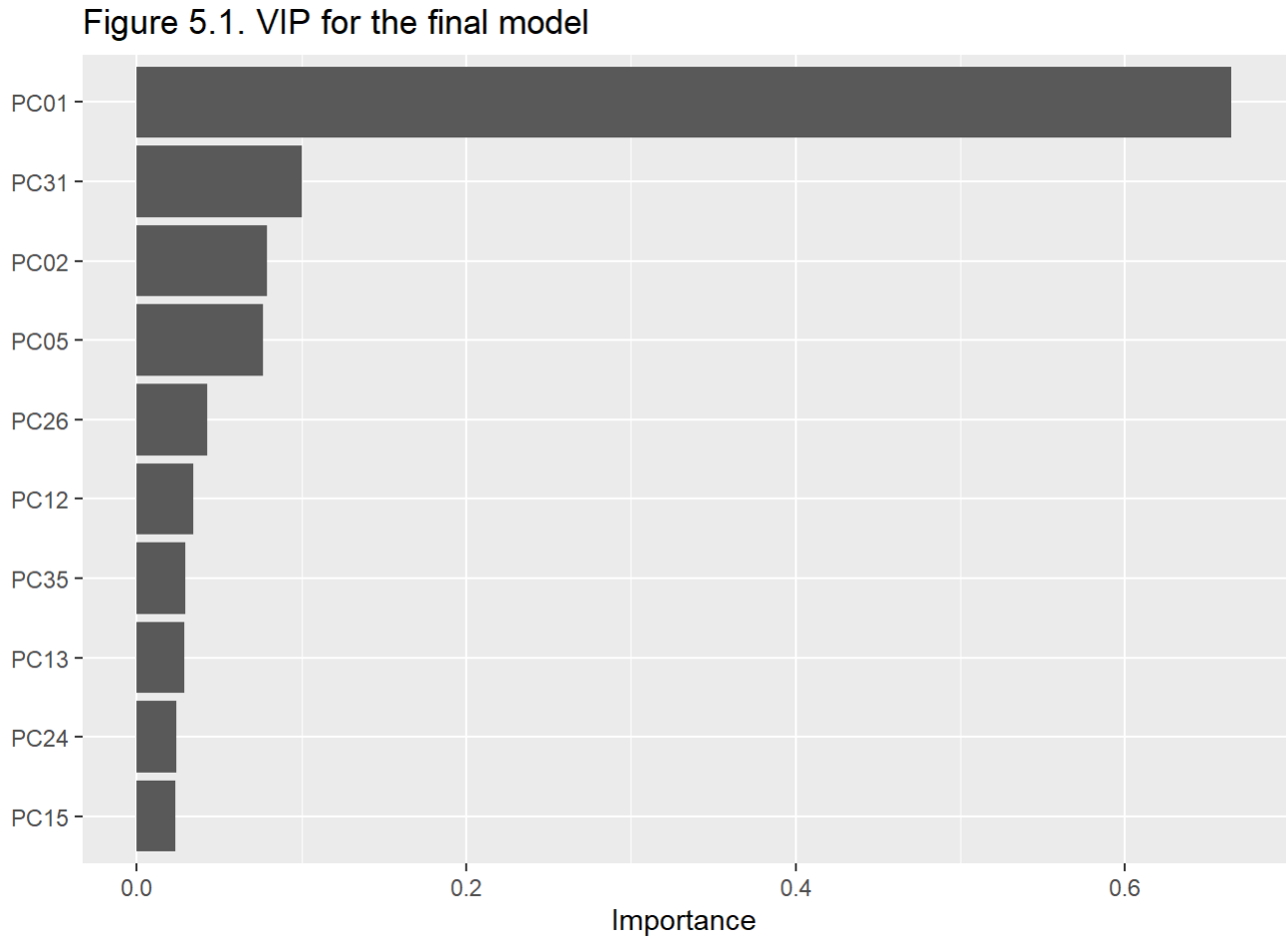
The model will be evaluated by firstly identifying the most important variables. The testing data will be used for evaluation which will be pre-processed using the bake function. The model precitions from the test data will then be compared to the true values and the accuracy metrics of RMSE and R squared will be gained.

These accuracy metrics will be compared to the accuracy of the cross-validated training data.

Finally the data provided in the brief for sales of the Fatal Empire will be pre-processed (using bake) and used to produce a prediction of sales in North America.

Part 1. Variable Importance

```
vgsales_rf %>%
  vip() +
  labs(title = "Figure 5.1. VIP for the final model")
```



Because we have used PCA in the pre-processing the variables of the importance metric are the different principle components rather than the original individual variables in the vgsales dataset.

Principle component 1 (figure 5.1) is by far the most important variable. It has the greatest influence on outcome. Interestingly PC31 is the next most important followed by PC2 and then PC5.

As detailed earlier in the PCA loadings (fig 2.5), the individual variables that contributed mostly to PC1 were EU_Sales, Other_Sales, and JP_Sales. Thus it is sales in other parts of the world that has the greatest influence on North American sales.

Part 2. How well is the model predicting?

Baking the test data

```
vgs_test_preproc <- bake( vgs_recipe2, vgs_test )
vgs_test_preproc
## # A tibble: 4,082 x 37
##    NA_Sales  PC01   PC02   PC03   PC04    PC05   PC06    PC07   PC08    PC09
##       <dbl> <dbl>  <dbl>  <dbl>  <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>
##  1    15.8  29.8  -1.25   1.65  -3.53   2.55   -3.05   2.94   -3.11  -2.58
##  2    11.4  28.4   4.73   1.43  -1.61   1.84   -0.281  1.82   -1.91  -1.92
##  3    14.0  22.7  -0.297  2.71  -2.65   2.55   -2.89   1.90   -0.407 -0.549
##  4    26.9   3.87  1.37  -1.81   0.271  0.0462  2.42  -2.62    0.710 -2.59
##  5     9.07 22.8  -0.245  4.46  -5.33   3.32   -2.66   2.12   -3.22  -1.10
##  6    12.8  11.6   4.69  -2.39   1.23  -0.243   2.32  -0.0884  0.423  0.0699
##  7     6.42 17.8   6.62   0.526 -0.851 -1.17   -0.328 -1.02   -0.302 -0.273
##  8    10.8  12.3   6.92  -2.26   1.10  -0.677   3.27   0.0182  0.102 -1.99
##  9     8.41 11.9  -2.81   2.40  -1.15   0.905  -2.26   2.82   -1.28  -0.0430
## 10     3.44 16.8   6.44   1.79  -1.61   1.34    0.546 -1.46   -2.46  -1.73
## # ... with 4,072 more rows, and 27 more variables: PC10 <dbl>, PC11 <dbl>,
## #   PC12 <dbl>, PC13 <dbl>, PC14 <dbl>, PC15 <dbl>, PC16 <dbl>, PC17 <dbl>,
## #   PC18 <dbl>, PC19 <dbl>, PC20 <dbl>, PC21 <dbl>, PC22 <dbl>, PC23 <dbl>,
## #   PC24 <dbl>, PC25 <dbl>, PC26 <dbl>, PC27 <dbl>, PC28 <dbl>, PC29 <dbl>,
## #   PC30 <dbl>, PC31 <dbl>, PC32 <dbl>, PC33 <dbl>, PC34 <dbl>, PC35 <dbl>,
## #   PC36 <dbl>
```

Getting predictions and adding the truth

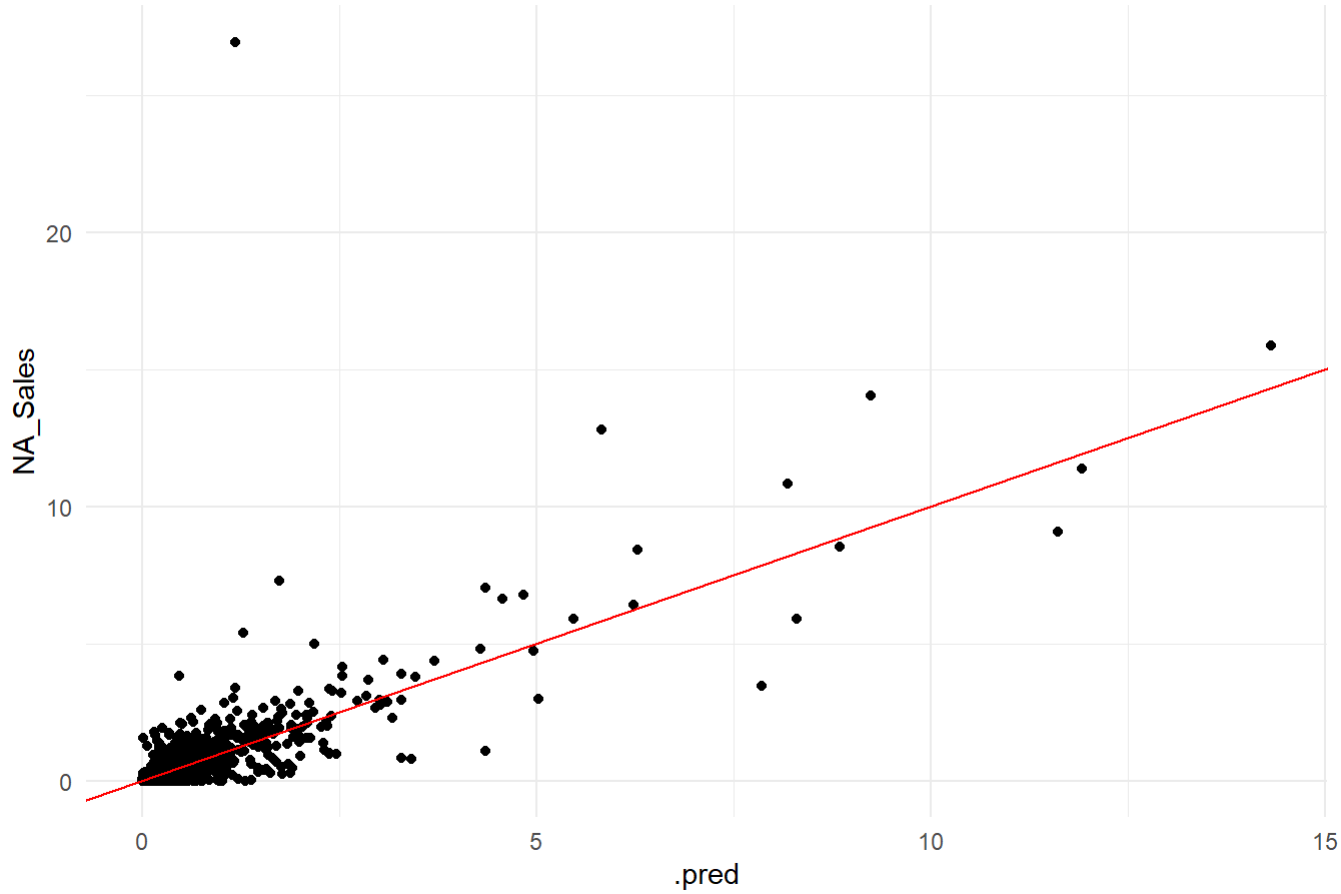
```
vgs_preds <- predict( vgsales_rf, # The predictions
                      new_data = vgs_test_preproc ) %>%
  bind_cols( vgs_test_preproc %>%  # Add the truth
               dplyr::select( NA_Sales ) )
vgs_preds
## # A tibble: 4,082 x 2
##    .pred NA_Sales
```

```
##    <dbl>    <dbl>
##  1 14.3     15.8
##  2 11.9     11.4
##  3  9.24    14.0
##  4  1.19    26.9
##  5 11.6      9.07
##  6  5.82    12.8
##  7  6.24     6.42
##  8  8.19    10.8
##  9  6.28     8.41
## 10  7.86     3.44
## # ... with 4,072 more rows
```

Some predictions look pretty good whilst others are way off. Visualising this will be better.

```
vgs_preds %>%
  ggplot( aes( x = .pred, y = NA_Sales ) ) +
  geom_point() +
  geom_abline( intercept = 0, slope = 1, colour = "red" ) +
  theme_minimal() +
  labs(title = "Figure 5.2. Truth versus predicted values")
```

Figure 5.2. Truth versus predicted values



There is a lot bunching of data points early in the plot making accurate interpretation difficult. There are more points above the red line than under it, suggesting there is some bias towards under prediction. On larger NA_Sales the model has less relative variation than the lower sales range and so looks to be predicting better with larger sales numbers.

Part 3. Accuracy metrics on the pre-processed testing data

```
vgs_preds %>%
  metrics( truth = NA_Sales, estimate = .pred ) %>%
  kable(caption = "Table 5.1. Accuracy metrics for testing data")
```

Table 5.1. Accuracy metrics for testing data

.metric	.estimator	.estimate
rmse	standard	0.5185396
rsq	standard	0.6262872
mae	standard	0.1126785

The RMSE of 0.519 is larger than the cross_validated RMSE and the R squared is lower then the cross-validated R squared.

Although the predictions on the test data are expected to be different to the cross validated training data due to these accuracy metrics being estimates only, I do expect some contributing factors. Firstly there is outlier data, some video games have substantially greater sales than average and the random sampling process will not distribute these equally. Different seeds could be used on the data splitting process to assess for any difference.

Part 4. The Fatal Empire North American sales prediction and interpretation.

Creating the dataframe

```
TheFatalEmpire_data <- tibble(
  Name = 'The Fatal Empire',
  Genre = 'Role-Playing',
  Platform = 'PS4',
  JP_Sales = 2.58,
  EU_Sales = 0.53,
  Other_Sales = 0.1
)

TheFatalEmpire_data <- TheFatalEmpire_data %>%
  mutate(Genre = as_factor(Genre),
         Platform = as_factor(Platform))
TheFatalEmpire_data
## # A tibble: 1 x 6
##   Name             Genre        Platform JP_Sales EU_Sales Other_Sales
##   <chr>            <fct>        <fct>       <dbl>    <dbl>       <dbl>
## 1 The Fatal Empire Role-Playing PS4          2.58     0.53         0.1
```

Pre-processing the Fatal Empire Data using bake()

```
TFE_data_pp <- bake(vgs_recipe2, TheFatalEmpire_data)
TFE_data_pp
## # A tibble: 1 x 36
##    PC01  PC02   PC03  PC04  PC05  PC06   PC07  PC08  PC09  PC10  PC11   PC12
##   <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1  5.16  2.72 -0.222 0.863 -2.67 0.423 -0.846 0.588 0.102 -1.61 0.178 -0.237
## # ... with 24 more variables: PC13 <dbl>, PC14 <dbl>, PC15 <dbl>, PC16 <dbl>,
## #   PC17 <dbl>, PC18 <dbl>, PC19 <dbl>, PC20 <dbl>, PC21 <dbl>, PC22 <dbl>,
## #   PC23 <dbl>, PC24 <dbl>, PC25 <dbl>, PC26 <dbl>, PC27 <dbl>, PC28 <dbl>,
## #   PC29 <dbl>, PC30 <dbl>, PC31 <dbl>, PC32 <dbl>, PC33 <dbl>, PC34 <dbl>,
## #   PC35 <dbl>, PC36 <dbl>
```

Prediction of North American Sales for The Fatal Empire

```
predict(vgsales_rf,
```

```
        new_data = TFE_data_pp)
## # A tibble: 1 x 1
##   .pred
##   <dbl>
## 1  1.08
```

The Random Forest model has predicted North American sales of 1.082 million copies for The Fatal Empire.

This sales figure is influenced mostly by the sales figures from around the world rather than the platform that the game is released on or the genre of the game.

The accuracy of the model is only fair. The RMSE being approximately 0.52 million sales, which is a large amount of difference on average between sales predicted and actual sales. The R squared of 0.626, suggests that the predictor variables in this model can explain only about 63% of the variation in sales in North America.

The model certainly gives us some idea of the expected sales in North America given the data provided, however a potential variation of half a million sales needs to be taken into account. Further feature engineering of the model could achieve better results and should be pursued when resources allow.