Video Game Sales Investment Study

Matthew Afoakwah, Nathan Cummings, Jaye Evans, Nikola Ignjatovic, Kevin Mainello, Chris Weiner

March 28th, 2023

## Abstract

This Project is based on an Investors desire to invest into Video Games. Before the investor can invest it is our job to investigate where the investor should best invest their money in a means that will ultimately benefit the investor. In order to accomplish this we use a Video Game data set in the time frame of 2010 to 2016. We will prepare the data set by exploring it, cleaning it so that we are able to create proper visualizations and models for the investor. At the end of this project we will explain to the investor our findings based on our visualizations and models.

## 1. Scenerio

Suppose their is an investor who wants to invest in a gaming platform. However before the investor invests, they want to know which gaming platform between the years 2010 to 2016 has the best NA, EU, and Global sales. The investor also wants to know what video game category produces the most income between 2010 and 2016 while for each video game platform so that they can better invest their money to better profit themselves in the future.The investor also wants to see what developers create more profitable games and the video game categories those developers mainly develop. The investor wants an indepth analyst on the video game platforms, video game categories, and developers before they invest their money.

## 2.Preparation

The data we are using was uploaded to Kaggle by SID\_TWR. SID\_TWR stated that this dataset was scraped from VGChartz and Metacritic.

2.1 Load Tidyverse package Before we work on our data we first want to load it into R, but first we must add two R packages into R before we attempt to work with data.

# install.packages('Tidyverse')  
  
library(tidyverse) #Used to load tidyverse package into R.

## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'tidyr' was built under R version 4.2.3

## Warning: package 'readr' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## Warning: package 'forcats' was built under R version 4.2.3

## Warning: package 'lubridate' was built under R version 4.2.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.1 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

library(readr) #Used to load readr package in R to use to read CSV file.

2.2 Import dataset  
Now we want to use import our dataset into R using the read\_CSV function that is from our readr package.

# Reads the CSV file from the file location using read\_csv, and imports the  
# file into R, named Video\_Games\_Sales  
Video\_Games\_Sales <- read\_csv("data/Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv")

## Rows: 16719 Columns: 16  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (7): Name, Platform, Year\_of\_Release, Genre, Publisher, Developer, Rating  
## dbl (9): NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales, Global\_Sales, Critic\_Sco...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

2.3 Preview dataset Before we can work on our dataset, we must first see if the dataset is relivant to our task. In order to accomplish this we must explore the dataset in R.

# we use the head() function to get a preview of the dataset.  
head(Video\_Games\_Sales)

## # A tibble: 6 × 16  
## Name Platf…¹ Year\_…² Genre Publi…³ NA\_Sa…⁴ EU\_Sa…⁵ JP\_Sa…⁶ Other…⁷ Globa…⁸  
## <chr> <chr> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Wii Spo… Wii 2006 Spor… Ninten… 41.4 29.0 3.77 8.45 82.5  
## 2 Super M… NES 1985 Plat… Ninten… 29.1 3.58 6.81 0.77 40.2  
## 3 Mario K… Wii 2008 Raci… Ninten… 15.7 12.8 3.79 3.29 35.5  
## 4 Wii Spo… Wii 2009 Spor… Ninten… 15.6 10.9 3.28 2.95 32.8  
## 5 Pokemon… GB 1996 Role… Ninten… 11.3 8.89 10.2 1 31.4  
## 6 Tetris GB 1989 Puzz… Ninten… 23.2 2.26 4.22 0.58 30.3  
## # … with 6 more variables: Critic\_Score <dbl>, Critic\_Count <dbl>,  
## # User\_Score <dbl>, User\_Count <dbl>, Developer <chr>, Rating <chr>, and  
## # abbreviated variable names ¹​Platform, ²​Year\_of\_Release, ³​Publisher,  
## # ⁴​NA\_Sales, ⁵​EU\_Sales, ⁶​JP\_Sales, ⁷​Other\_Sales, ⁸​Global\_Sales

We have now seen some of the data set but before we can verify that the data set is relevant we must explore the data set further. To accomplish this we use the glimpse() function

# shows a glimpse of the data set along with the attributes.  
glimpse(Video\_Games\_Sales)

## Rows: 16,719  
## Columns: 16  
## $ Name <chr> "Wii Sports", "Super Mario Bros.", "Mario Kart Wii", "…  
## $ Platform <chr> "Wii", "NES", "Wii", "Wii", "GB", "GB", "DS", "Wii", "…  
## $ Year\_of\_Release <chr> "2006", "1985", "2008", "2009", "1996", "1989", "2006"…  
## $ Genre <chr> "Sports", "Platform", "Racing", "Sports", "Role-Playin…  
## $ Publisher <chr> "Nintendo", "Nintendo", "Nintendo", "Nintendo", "Ninte…  
## $ NA\_Sales <dbl> 41.36, 29.08, 15.68, 15.61, 11.27, 23.20, 11.28, 13.96…  
## $ EU\_Sales <dbl> 28.96, 3.58, 12.76, 10.93, 8.89, 2.26, 9.14, 9.18, 6.9…  
## $ JP\_Sales <dbl> 3.77, 6.81, 3.79, 3.28, 10.22, 4.22, 6.50, 2.93, 4.70,…  
## $ Other\_Sales <dbl> 8.45, 0.77, 3.29, 2.95, 1.00, 0.58, 2.88, 2.84, 2.24, …  
## $ Global\_Sales <dbl> 82.53, 40.24, 35.52, 32.77, 31.37, 30.26, 29.80, 28.92…  
## $ Critic\_Score <dbl> 76, NA, 82, 80, NA, NA, 89, 58, 87, NA, NA, 91, NA, 80…  
## $ Critic\_Count <dbl> 51, NA, 73, 73, NA, NA, 65, 41, 80, NA, NA, 64, NA, 63…  
## $ User\_Score <dbl> 8.0, NA, 8.3, 8.0, NA, NA, 8.5, 6.6, 8.4, NA, NA, 8.6,…  
## $ User\_Count <dbl> 322, NA, 709, 192, NA, NA, 431, 129, 594, NA, NA, 464,…  
## $ Developer <chr> "Nintendo", NA, "Nintendo", "Nintendo", NA, NA, "Ninte…  
## $ Rating <chr> "E", NA, "E", "E", NA, NA, "E", "E", "E", NA, NA, "E",…

Now we want to look at all the columns within the data set. We accomplish this using the colnames() function.

# Shows the name of the columns in the Video\_Games\_Sales data set.  
colnames(Video\_Games\_Sales)

## [1] "Name" "Platform" "Year\_of\_Release" "Genre"   
## [5] "Publisher" "NA\_Sales" "EU\_Sales" "JP\_Sales"   
## [9] "Other\_Sales" "Global\_Sales" "Critic\_Score" "Critic\_Count"   
## [13] "User\_Score" "User\_Count" "Developer" "Rating"

Now we want to determine the size of the data set so we use the dim() function to determine the data sets size

# After using the dim() function we see that the length of the data set is 16  
# and that there is 16719 rows.  
dim(Video\_Games\_Sales)

## [1] 16719 16

To continue our exploration we wish to see the data type of each column, in order to accomplish this we use the class function.

# We use the sapply() function to apply the class function on each attribute of  
# our data set.  
sapply(Video\_Games\_Sales, class)

## Name Platform Year\_of\_Release Genre Publisher   
## "character" "character" "character" "character" "character"   
## NA\_Sales EU\_Sales JP\_Sales Other\_Sales Global\_Sales   
## "numeric" "numeric" "numeric" "numeric" "numeric"   
## Critic\_Score Critic\_Count User\_Score User\_Count Developer   
## "numeric" "numeric" "numeric" "numeric" "character"   
## Rating   
## "character"

From viewing our data set we see that our data set is relevant to our task and has given us insight to our data set which will allow us to move on to the next step.

## 3. Data Cleaning

Though we can see that our data set has information that can be used for our problem there is also irrelevant data in the data set, in order to focus on the information that we need we will need to clean the data set using transformations.

3.1 Since we are focusing on the video game sales between 2010 to 2016 we will shrink the data set by filtering it.

# use the filter() function to remove all rows where the release year is not  
# between 2010 and 2016. Then we store it in a new data set called  
# gamesales2010\_2016  
gamesales2010\_2016 <- filter(Video\_Games\_Sales, Year\_of\_Release >= 2010)  
gamesales2010\_2016 <- filter(gamesales2010\_2016, Year\_of\_Release <= 2016)  
# We use the dim() function and see that the amount of rows in the data set.  
dim(gamesales2010\_2016)

## [1] 5277 16

3.2 We will now filter all rows where the Year\_of\_Release is unknown.

# The filter() function removes all values that are 'N/A' in the  
# Year\_of\_Release column  
gamesales2010\_2016 <- filter(gamesales2010\_2016, Year\_of\_Release != "N/A")  
# nrow() prints out the number of rows in gamesales2010\_2016.  
nrow(gamesales2010\_2016)

## [1] 5277

We have decided to remove all rows where the Year\_of\_Release is “N/A” because there are not too much “N/A” value’s missing in the Year\_of\_Release column which would greatly impact our data set, because of this we are able to easily remove the “N/A” values from the data set without having to worry about our results being greatly influenced.

3.3 We want to find and remove any duplicated files so we use the duplicated() function.

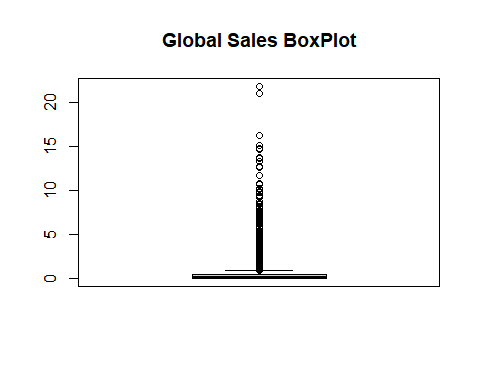
# We use the duplicated() to find all rows where they are repeated occurences.  
# #Then we use the ! to filter them out from the graph.  
gamesales2010\_2016 <- gamesales2010\_2016[!duplicated(gamesales2010\_2016), ]  
# We use #nrow() function to see the amount of rows left in the data set.  
nrow(gamesales2010\_2016)

## [1] 5277

After checking/removing all duplicated files in the data set we see view how many rows are in the data set. We notice that the number of rows did not decrease so there were no repeated values in the dataset.

3.4 We want to identify any outliers that are in our data set before we can continue any further. To accomplish this what we want to do is use a boxplot graph to identify any outliers in the graph. Because the numeric columns we will be focusing on are Global and NA Sales we will only look for outliers in these two columns.

# We create a boxplot for the Global Sales in gamesales\_2010\_2016 then set the  
# title to 'Global Sales BoxPlot'.  
boxplot(gamesales2010\_2016$Global\_Sales, main = "Global Sales BoxPlot")

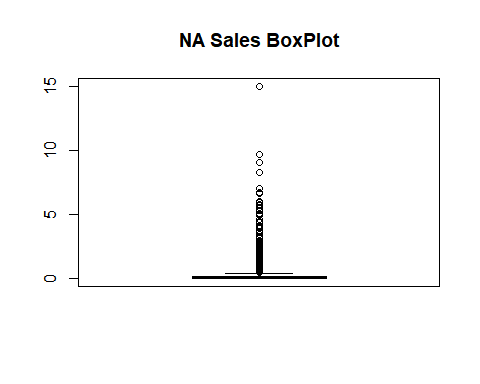


# We create a new variable outliers, then use boxplot.stats() to assign our  
# outliers from Global sales to outliers.  
outliers <- boxplot.stats(gamesales2010\_2016$Global\_Sales)$out

This boxplot creates a visual representation of the max, min, and median values in the Global sales of our data set.

We can see that there are some outliers in the Global Sales, this allows us to easily remove them from our data set. However we will not remove them as they will also not affect our progress moving forward. Now, we will check the outliers for the NA Sales.

# We create a boxplot for the NA Sales in gamesales\_2010\_2016 then set the  
# title to 'NA Sales BoxPlot'.  
boxplot(gamesales2010\_2016$NA\_Sales, main = "NA Sales BoxPlot")



# We create a new variable outliersNA, then use boxplot.stats() to assign our  
# outliers from NA sales to outliersNA.  
outliersNA <- boxplot.stats(gamesales2010\_2016$NA\_Sales)$out

This boxplot creates a visual representation of the max, min, and median values in the NA sales of our data set.

In this data set outliers are usually the top selling games or the ones that sell very little. This is why we did not remove any outliers, as it is valuable data and is relevant to what we will be searching for in the data set.

## 4. Data Exploration

We now develop questions to explore the data set even further.

1. Which video game platform made the most NA Sales from 2010 to 2016?
2. Which video game platform made the most Global Sales from 2010 to 2016?
3. Which video game category had the highest NA sales?
4. Which video game category made the highest Global sales?
5. Which top five video game developers have the highest Global video game sales?
6. Which video game genre is the top five developers most profitable in?

4.1 Before we can go and answer the questions we first want to explore the unique attributes in the platform column to see if there are any discrepancies that need to be fixed.

# unique function lists the unique characters in the platform column.  
unique(gamesales2010\_2016$Platform)

## [1] "X360" "PS3" "DS" "PS4" "3DS" "Wii" "XOne" "WiiU" "PC" "PSP"   
## [11] "PSV" "PS2"

From looking at the unique attributes in the platform column we do have irregularities so we need to recode the names in Platform using the recode() function.

# We use recode() to change the values of certain attributes to new attributes  
# such as Nintendo, PlayStation, and Xbox.  
gamesales2010\_2016$Platform <- recode(gamesales2010\_2016$Platform, X360 = "Xbox",  
 PS3 = "PlayStation", DS = "Nintendo", PS4 = "PlayStation", `3DS` = "Nintendo",  
 Wii = "Nintendo", XOne = "Xbox", WiiU = "Nintendo", PSP = "PlayStation", PSV = "PlayStation",  
 PS2 = "PlayStation")  
# We use unique()to list the unique attributes in the platform column.  
unique(gamesales2010\_2016$Platform)

## [1] "Xbox" "PlayStation" "Nintendo" "PC"

Now we have finished changing the attributes in the Platform column and can now move on to our next task.

4.2 In order to answer the first question we must calculate the NA revenue for each platform.

# We move the gamesales2010\_2016 data into a new data set.  
platform\_NA\_Sales <- gamesales2010\_2016 %>%  
 # We use the group\_by() to group all the platforms in the platform\_NA\_Sales  
 # together.  
group\_by(Platform) %>%  
 # We use the summarize() to summarize the sum of NA\_sales for each  
 # different platform.  
summarize(NA\_total = sum(NA\_Sales)) %>%  
 # We arrange the data set by decreasing order according to the NA\_total.  
arrange(desc(NA\_total))  
# We use the head() function to preview the new data set.  
head(platform\_NA\_Sales)

## # A tibble: 4 × 2  
## Platform NA\_total  
## <chr> <dbl>  
## 1 Xbox 427.   
## 2 PlayStation 362.   
## 3 Nintendo 302.   
## 4 PC 39.1

What this code does is that it sums up the total NA sales made between 2010 to 2016 for each video game platform. We create another table which calculates the NA and Global sales for each platform for every different year as well.

# We move the gamesales2010\_2016 data into a new data set.  
platform\_SalesbyYear <- gamesales2010\_2016 %>%  
 # We use the group\_by() to group all the Platform and Year\_of\_Release  
 # together.  
group\_by(Platform, Year\_of\_Release) %>%  
 # We use the summarize() to summarize the sum of NA\_sales for each  
 # different platform, and year.  
summarize(NA\_total = sum(NA\_Sales), glbl\_total = sum(Global\_Sales)) %>%  
 # We arrange the data set by decreasing order according to the glbl\_total.  
arrange(desc(glbl\_total))

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

# We use the head() function to preview the new data set.  
head(platform\_SalesbyYear)

## # A tibble: 6 × 4  
## # Groups: Platform [3]  
## Platform Year\_of\_Release NA\_total glbl\_total  
## <chr> <chr> <dbl> <dbl>  
## 1 Nintendo 2010 113. 213.  
## 2 PlayStation 2010 70.6 183.  
## 3 PlayStation 2011 68.6 180.  
## 4 Xbox 2010 107. 170.  
## 5 PlayStation 2014 53.1 160.  
## 6 PlayStation 2013 54.1 153.

What this code does is that it calculates the total NA and Global sales of each platform based on the year. This will allow us to find a trend in the profit of each platform.

4.3 Secondly, we calculate the global revenue for each platform.

# We move the gamesales2010\_2016 data into a new data set then pipe the  
# command.  
platform\_glbl\_sales <- gamesales2010\_2016 %>%  
 # We use the group\_by() to group all the platforms together.  
group\_by(Platform) %>%  
 # We use the summarize function to summarize the sum of Global\_sales for  
 # each different platform.  
summarize(glbl\_total = sum(Global\_Sales)) %>%  
 # We arrange the data set by decreasing order according to the glbl\_total.  
arrange(desc(glbl\_total))  
# We use the head() function to preview the table.  
head(platform\_glbl\_sales)

## # A tibble: 4 × 2  
## Platform glbl\_total  
## <chr> <dbl>  
## 1 PlayStation 1026.  
## 2 Xbox 710.  
## 3 Nintendo 687.  
## 4 PC 122.

4.4 Now, we want to calculate the NA revenue based on genre.but before we can do that we must see if there are any dependencies with the character values in genre.

# We use the unique to search for any irregular values in the categories  
# column.  
unique(gamesales2010\_2016$Genre)

## [1] "Misc" "Action" "Role-Playing" "Shooter" "Racing"   
## [6] "Platform" "Simulation" "Sports" "Fighting" "Strategy"   
## [11] "Adventure" "Puzzle"

After using the unique() function we see that there are not really any irregularities in the function which allows us to continue to calculate the NA revenue based on genre.

# We move the gamesales2010\_2016 data into a new data set.  
genre\_na\_sales <- gamesales2010\_2016 %>%  
 # We use the group\_by() to group all the Genres together.  
group\_by(Genre) %>%  
 # We use summarize() to summarize the sum of na\_sales for each different  
 # genre.  
summarise(genre\_na\_total = sum(NA\_Sales)) %>%  
 # We arrange the data set by decreasing order according to the  
 # genre\_na\_total.  
arrange(desc(genre\_na\_total))  
# We use the head() function to preview the table.  
head(genre\_na\_sales)

## # A tibble: 6 × 2  
## Genre genre\_na\_total  
## <chr> <dbl>  
## 1 Action 291.   
## 2 Shooter 237.   
## 3 Sports 157.   
## 4 Misc 124.   
## 5 Role-Playing 112.   
## 6 Platform 54.9

We now create another table that separates the NA sales by genre for each year.

# We move the gamesales2010\_2016 data into a new data set.  
genre\_SalesbyYear <- gamesales2010\_2016 %>%  
 # We use the group\_by() to group all the Genre and Year\_of\_Release  
 # together.  
group\_by(Genre, Year\_of\_Release) %>%  
 # We use the summarize() to summarize the sum of NA\_sales for each  
 # different Genre, and year.  
summarize(NA\_total = sum(NA\_Sales), glbl\_total = sum(Global\_Sales)) %>%  
 # We arrange the data set by decreasing order according to the glbl\_total.  
arrange(desc(glbl\_total))

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

# We use the head() function to preview the new data set.  
head(genre\_SalesbyYear)

## # A tibble: 6 × 4  
## # Groups: Genre [2]  
## Genre Year\_of\_Release NA\_total glbl\_total  
## <chr> <chr> <dbl> <dbl>  
## 1 Action 2013 53.5 123.   
## 2 Action 2012 51.8 119.   
## 3 Action 2011 53.1 117.   
## 4 Action 2010 59.7 115.   
## 5 Shooter 2011 49.7 98.2  
## 6 Action 2014 38.8 97.3

4.5 We now want to calculate the Global revenue based on genre.

# We move the gamesales2010\_2016 data into a new data set then pipe the  
# command.  
genre\_glbl\_sales <- gamesales2010\_2016 %>%  
 # We use group\_by() to group all the Genres in the genre\_glbl\_sales  
 # together.  
group\_by(Genre) %>%  
 # We use summarize() to summarize the sum of Global\_sales for each  
 # different genre.  
summarise(genre\_glbl\_total = sum(Global\_Sales)) %>%  
 # We arrange the data set by decreasing order according to the  
 # genre\_na\_total.  
arrange(desc(genre\_glbl\_total))  
# We use the head() function to preview the table.  
head(genre\_glbl\_sales)

## # A tibble: 6 × 2  
## Genre genre\_glbl\_total  
## <chr> <dbl>  
## 1 Action 673.  
## 2 Shooter 480.  
## 3 Sports 329.  
## 4 Role-Playing 315.  
## 5 Misc 235.  
## 6 Racing 123.

4.6 Before we can attempt to calculate NA and Global sales based on developer we must first determine whether there are any irregularities in the Developer column.

# We use unique() to take all unique values in the developer column then sort  
# them.  
unique\_dev <- sort(unique(gamesales2010\_2016$Developer))

After placing all the unique values from the Developer column of the data set gamesales2010\_2016 in a new set, we see that there are over 800 different values and there’s bound to be some missed spelled values so we review all the unique values and change their names to match what we want it to be. There are some values who may have be in a different region but are part of a company but because we want the overall of a company we change the different region companies to match its parent company as well.

# We use the recode() function from the dplyr package from our tidyverse  
# package in order change our variables.  
gamesales2010\_2016$Developer <- recode(gamesales2010\_2016$Developer, `1C: Maddox Games` = "1C Company",  
 `1C:Ino-Co` = "1C Company", `2K Australia` = "2K Games", `2K Czech` = "2K Games",  
 `2K Sports` = "2K Games", `2K Marin` = "2K Games", `2K Play` = "2K Games", `505 Games, Sarbakan Inc.` = "505 Games",  
 `Activision, Behaviour Interactive` = "Activision", `Activision, FreeStyleGames` = "Activision",  
 `Ambrella, The Pokemon Company` = "Ambrella", `Armature Studio, comcept` = "Armature Studio",  
 `Artificial Mind and Movement, EA Redwood Shores` = "Artificial Mind and Movement",  
 `Atari, Atari SA` = "Atari", `Atari, Slightly Mad Studios, Atari SA` = "Atari",  
 `Atlus, Dingo Inc.` = "Atlus", `Atomic Planet Entertainment` = "Atomic Games",  
 `Avalanche Software` = "Avalanche Studios", `Bandai Namco Games, Artdink` = "Bandai Namco Games",  
 `Beenox, Other Ocean Interactive` = "Beenox", `Big Blue Bubble Inc., Scholastic, Inc.` = "Big Blue Bubble Inc.",  
 `Big Blue Bubble Inc., Scholastic, Inc.` = "Big Blue Bubble Inc.", `Blitz Games Studios` = "Blitz Games",  
 `Blue Byte, Related Designs` = "Blue Byte", `Bungie Software, Bungie` = "Bungie",  
 `Capcom Vancouver` = "Capcom", `Capcom, Pipeworks Software, Inc.` = "Capcom")  
  
gamesales2010\_2016$Developer <- recode(gamesales2010\_2016$Developer, `Capcom, QLOC` = "Capcom",  
 `Climax Entertainment` = "Climax Studios", `Climax Group` = "Climax Studios",  
 `Climax Group, Climax Studios` = "Climax Studios", `Codemasters Birmingham` = "Codemasters",  
 `Compile Heart, GCREST` = "Compile Heart", `Crave, DTP Entertainment` = "Crave",  
 `Crystal Dynamics, Nixxes Software` = "Crystal Dynamics", `Cyanide, Cyanide Studios` = "Cyanide",  
 `CyberConnect2, Racjin` = "CyberConnect2", `CyberPlanet Interactive Public Co., Ltd., Maximum Family Games` = "CyberPlanet Interactive Public Co., Ltd.",  
 `Deep Silver Dambuster Studios` = "Deep Silver", `Deep Silver, Keen Games` = "Deep Silver",  
 `Dimps Corporation, Dream Execution` = "Dimps Corporation", `Dimps Corporation, SCE Japan Studio` = "Dimps Corporation",  
 `Disney Interactive Studios, Land Ho!` = "Disney Interactive Studios", `EA Black Box` = "EA Games",  
 `EA Bright Light` = "EA Games", `EA Canada` = "EA Games", `EA Canada, EA Vancouver` = "EA Games",  
 `EA DICE` = "EA Games", `EA DICE, Danger Close` = "EA Games")  
#'EA Sports' = 'EA Games'  
  
gamesales2010\_2016$Developer <- recode(gamesales2010\_2016$Developer, `EA Sports` = "EA Games",  
 `EA Montreal` = "EA Games", `EA Redwood Shores` = "EA Games", `EA Sports, EA Canada` = "EA Games",  
 `EA Sports, EA Vancouver` = "EA Games", `EA Tiburon` = "EA Games", `Eidos Montreal, Nixxes Software` = "Eidos Montreal",  
 `Engine Software, Re-Logic` = "Engine Software", `Epic Games, People Can Fly` = "Epic Games",  
 `Farsight Studios, Crave` = "Farsight Studios", `Gaijin Entertainment` = "Gaijin Games",  
 `Gearbox Software, 3D Realms` = "Gearbox Software", `Gearbox Software, WayForward` = "Gearbox Software",  
 `Guerilla Cambridge` = "Guerrilla Cambridge", Guerilla = "Guerilla Cambridge")  
  
  
gamesales2010\_2016$Developer <- recode(gamesales2010\_2016$Developer, `Harmonix Music Systems, Demiurge` = "Harmonix Music Systems",  
 `Headup Games, Crenetic Studios` = "Headup Games", `Konami Computer Entertainment Hawaii` = "Konami",  
 `Marvelous AQL` = "Marvelous Inc.", `Marvelous Entertainment` = "Marvelous Inc.",  
 `Midway Studios - Austin` = "Midway", `Monolith Soft` = "Monolith Productions",  
 `Monolith Soft, Banpresto` = "Monolith Productions")  
  
gamesales2010\_2016$Developer <- recode(gamesales2010\_2016$Developer, `Namco Bandai Games America, Namco Bandai Games` = "Namco Bandai Games",  
 `Namco Bandai Games, Bandai Namco Games` = "Namco Bandai Games", `Namco Bandai Games, Cellius` = "Namco Bandai Games",  
 `Namco Bandai Games, Monkey Bar Games` = "Namco Bandai Games", `NATSUME ATARI Inc.` = "Natsume",  
 `Nintendo EAD Tokyo` = "Nintendo", `Nintendo, Camelot Software Planning` = "Nintendo",  
 `Nintendo, Headstrong Games` = "Nintendo", `Nintendo, Intelligent Systems` = "Nintendo",  
 `Nintendo, Nd Cube` = "Nintendo", `Nintendo, Nintendo Software Technology` = "Nintendo",  
 `Nintendo, Spike Chunsoft` = "Nintendo", `Paradox Development Studio` = "Paradox Interactive")  
  
  
gamesales2010\_2016$Developer <- recode(gamesales2010\_2016$Developer, `PLAYGROUND, Playground Games` = "Playground Games",  
 `Retro Studios, Entertainment Analysis & Development Division` = "Retro Studios",  
 `Rockstar Leeds` = "Rockstar Studios", `Rockstar North` = "Rockstar Studios",  
 `Rockstar San Diego` = "Rockstar Studios", `Sanzaru Games, Sanzaru Games, Inc.` = "Sanzaru Games",  
 `SCE Japan Studio, comcept` = "SCE Studio", `SCE Santa Monica` = "SCE Studio",  
 `SCE Studio Cambridge` = "SCE Studio", `SCE Japan Studio` = "SCE Studio", `SCEA San Diego Studios` = "SCEA",  
 `SCEA, Zindagi Games` = "SCEA", `SCEE London Studio` = "SCEE")  
  
gamesales2010\_2016$Developer <- recode(gamesales2010\_2016$Developer, `Sega Studios San Francisco` = "Sega",  
 `Sega Toys` = "Sega", `Sega, Dimps Corporation` = "Sega", `Sega, French-Bread` = "Sega",  
 `Sega, Sonic Team` = "Sega", Snapdragon = "Snap Dragon Games", `Sonic Team` = "Sega",  
 `Sony Bend` = "Sony Interactive Entertainment", `Sony Online Entertainment` = "Sony Interactive Entertainment",  
 `Spike Chunsoft` = "Spike", `Spike Chunsoft Co. Ltd., Spike Chunsoft` = "Spike",  
 Tecmo = "Tecmo Koei Games", `Tecmo Koei Canada` = "Tecmo Koei Games", `THQ Australia` = "THQ",  
 `THQ Digital Studio Phoenix` = "THQ", `Ubisoft Casablanca` = "Ubisoft", `Ubisoft Milan` = "Ubisoft",  
 `Ubisoft Montpellier` = "Ubisoft", `Ubisoft Montreal` = "Ubisoft", `Ubisoft Osaka` = "Ubisoft",  
 `Ubisoft Paris` = "Ubisoft", `Ubisoft Paris, Ubisoft Montpellier` = "Ubisoft",  
 `Ubisoft Quebec` = "Ubisoft", `Ubisoft Reflections` = "Ubisoft", `Ubisoft Reflections, Ivory Tower` = "Ubisoft",  
 `Ubisoft Romania` = "Ubisoft", `Ubisoft Sofia` = "Ubisoft", `Ubisoft Toronto` = "Ubisoft",  
 `Ubisoft Vancouver` = "Ubisoft", `Ubisoft, FunHouse` = "Ubisoft", `Ubisoft, Ludia Inc.` = "Ubisoft",  
 `Ubisoft, Ubisoft Montreal` = "Ubisoft")

Because of how much values are in the Developer column there was many values that needed to be changed in the dataset.

4.7 Since we have recoded the values in the Developers column we can now calculate the global revenue based on the developer.

dev\_global <- gamesales2010\_2016 %>%  
 # We want to filter out any unknown Developers as there are so few NA  
 # developers that will affect our variables.  
filter(Developer != "N/A") %>%  
 # We will then group the columns according to the value in the developer  
 # columns.  
group\_by(Developer) %>%  
 # We sum up the total global sales and NA sales of each developers games  
 # earned.  
summarise(global\_total = sum(Global\_Sales)) %>%  
 # We then arrange each row in descending order based on the global\_total.  
arrange(desc(global\_total)) %>%  
 # We use the slice\_head() function to take ONLY the top five most  
 # profitable developers.  
slice\_head(n = 5)  
# We preview the data set.  
head(dev\_global)

## # A tibble: 5 × 2  
## Developer global\_total  
## <chr> <dbl>  
## 1 EA Games 209.   
## 2 Ubisoft 183.   
## 3 Nintendo 95.0  
## 4 Rockstar Studios 75.7  
## 5 Treyarch 58.4

4.8 Finally, we want to determine the video game categories that are mainly developed by the top three developers and how the revenue in these categories differ.

From the previous example we were able to already determine the top 5 developers, so if we use the information from the previous table we know that “EA Games”, “Ubisoft”, “Nintendo”, “Rockstar Studios”, and “Treyarch” are the top five developers, this allows us to create a new table which filters any developers that aren’t those three developers.

# We create a table which shows video games where the developers are the top  
# five developers.  
developer\_genre\_sale <- gamesales2010\_2016 %>%  
 filter(Developer %in% c("EA Games", "Ubisoft", "Nintendo", "Rockstar Studios",  
 "Treyarch"))  
# We use this to preview the data set.  
head(developer\_genre\_sale)

## # A tibble: 6 × 16  
## Name Platf…¹ Year\_…² Genre Publi…³ NA\_Sa…⁴ EU\_Sa…⁵ JP\_Sa…⁶ Other…⁷ Globa…⁸  
## <chr> <chr> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Grand T… PlaySt… 2013 Acti… Take-T… 7.02 9.09 0.98 3.96 21.0  
## 2 Grand T… Xbox 2013 Acti… Take-T… 9.66 5.14 0.06 1.41 16.3  
## 3 Call of… Xbox 2010 Shoo… Activi… 9.7 3.68 0.11 1.13 14.6  
## 4 Call of… PlaySt… 2012 Shoo… Activi… 4.99 5.73 0.65 2.42 13.8  
## 5 Call of… Xbox 2012 Shoo… Activi… 8.25 4.24 0.07 1.12 13.7  
## 6 Call of… PlaySt… 2010 Shoo… Activi… 5.99 4.37 0.48 1.79 12.6  
## # … with 6 more variables: Critic\_Score <dbl>, Critic\_Count <dbl>,  
## # User\_Score <dbl>, User\_Count <dbl>, Developer <chr>, Rating <chr>, and  
## # abbreviated variable names ¹​Platform, ²​Year\_of\_Release, ³​Publisher,  
## # ⁴​NA\_Sales, ⁵​EU\_Sales, ⁶​JP\_Sales, ⁷​Other\_Sales, ⁸​Global\_Sales

Now that we have isolated the video games that are developed by the top five developers we can now group all the data by genre and developer and calculate the global sale and NA sale based on the the the genre. but first we must confirm the amount of N/A values in our Genre column to see if those N/A’s will affect our results.

# We use is.na() to determine if any values in Genre is N/A. Then use sum() to  
# count the exact number of N/A values.  
sum(is.na(developer\_genre\_sale$Genre))

## [1] 0

Fortunately, there is no N/A variables in the Genre column so we can continue on with manipulating the data set.

# We store the developer\_genre\_sale data into df.  
df <- developer\_genre\_sale %>%  
 # We then group by Developer and Genre.  
group\_by(Developer, Genre) %>%  
 # We summarise the NA and Global total for each category based on  
 # developer.  
summarise(developer\_glbl\_total = sum(Global\_Sales), developer\_NA\_total = sum(NA\_Sales)) %>%  
 # We arrange the order in alphabetical order using the Developer column.  
arrange(Developer)

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

# We preview the data set.  
head(df)

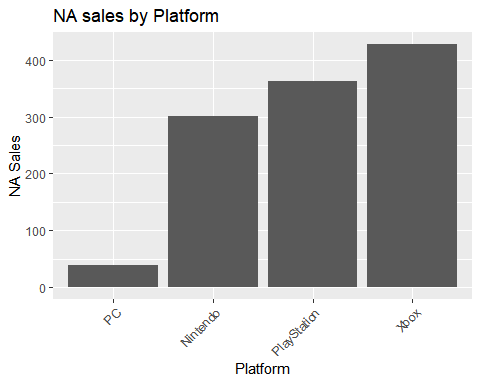
## # A tibble: 6 × 4  
## # Groups: Developer [1]  
## Developer Genre developer\_glbl\_total developer\_NA\_total  
## <chr> <chr> <dbl> <dbl>  
## 1 EA Games Action 5.3 3.55  
## 2 EA Games Adventure 0.32 0.17  
## 3 EA Games Fighting 2.61 1.24  
## 4 EA Games Platform 0.57 0.21  
## 5 EA Games Racing 1.56 0.62  
## 6 EA Games Shooter 49.7 23.1

## 5. Analyzing

Now that we have cleaned the data set and explored the data set we can now analyze the data set using visualization. In this section we will use multiple different graphs which helps us identify and analyze our data to help the client understand where they would receive the most profit.

5.1 In order to be able to answer our first question, we must first determine the most profitable platform and the rate of change between sales of each platform since 2010 to 2016.

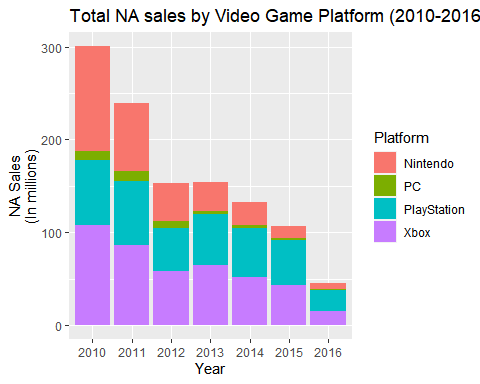
# We use the ggplot2 package to create our graph, the aes() function is used to  
# set the aesthetics of the plot. we set our x value to Platform and our Y  
# value to NA\_total. we also use the reorder() function to reorder the graph  
# from smallest to greatest.  
ggplot(data = platform\_NA\_Sales, aes(x = reorder(Platform, NA\_total), y = NA\_total)) +  
 geom\_bar(stat = "identity") + labs(title = "NA sales by Platform", x = "Platform",  
 y = "NA Sales") + theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# labs() is used to changes the labels in various parts of the chart. Both the  
# X and Y axis changing their names, as well as the title. theme() helps change  
# non-data elements of a chart. In this case adding a change to Platform’s axis  
# by utilizing element\_text() to change the look of texts, adding an angle and  
# using hjust to control horizontal justification)

This code produces the above graph in which it displays the sum of all Platforms in the 2010 to 2016 range.

# We use the ggplot2 package to create our graph, the aes() function is used to  
# set the aesthetics of the plot. we set our x value to Year\_of\_Release, our Y  
# value to NA\_total, and fill based on platform.  
ggplot(platform\_SalesbyYear, aes(x = Year\_of\_Release, y = NA\_total, fill = Platform)) +  
 geom\_bar(stat = "identity") + labs(title = "Total NA sales by Video Game Platform (2010-2016)",  
 x = "Year", y = "NA Sales\n(In millions)")



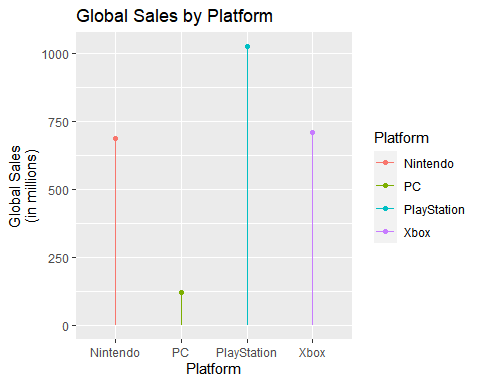
# labs() is used to changes the labels in various parts of the chart. Both the  
# X and Y axis changing their names, as well as the title.

This code creates a bar graph that breaks up the total amount of NA sales for each of the different years of each different Platform.

Analysis: Overall, from these graphs we can see that Xbox is the most profitable between all these platforms. However, compared to 2010 when it was at its highest, we are able to see that the sales for Xbox has been steadily declining same goes for they other platforms as well. But, compared to other platforms PC has had the least amount of sales, and have seen that by 2016 PlayStation had they highest sells.

5.2 To determine the most profitable platform globally we must first analyze the total sales made between 2010 and 2016 as well as analyzing the rate of change between these platforms.

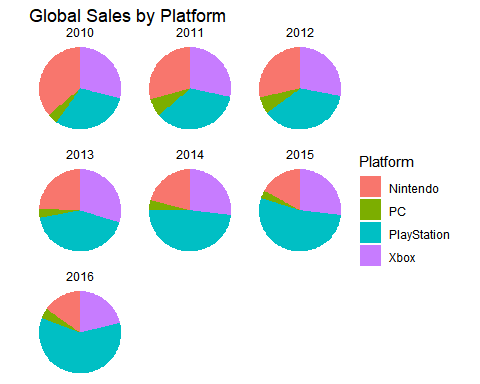
# We use the ggplot2 package to create our graph, the aes() function is used to  
# set the aesthetics of the plot. we set our x value to Platform, and our y  
# value to glbl\_total.  
ggplot(platform\_glbl\_sales, aes(x = Platform, y = glbl\_total, color = Platform)) +  
 geom\_point(stat = "identity") + geom\_segment(aes(x = Platform, xend = Platform,  
 y = 0, yend = glbl\_total)) + labs(title = "Global Sales by Platform", x = "Platform",  
 y = "Global Sales\n(in millions)")



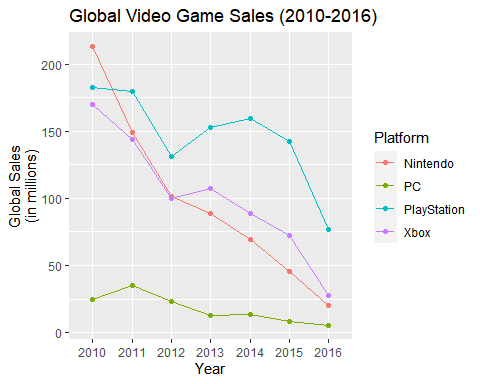
# geom\_point() sets a point at the value of each platforms global sales.  
# geom\_segment() creates lines at our platforms and reaches its total amount.  
# We use labs() to rename the x and y axis as well as our title.

This code generates a lollipop graph in which it displays the total global sales of each platform in 2010 to 2016.

ggplot(data = platform\_SalesbyYear, aes(x = 0, y = glbl\_total, fill = Platform)) +  
 geom\_col(position = "fill") + facet\_wrap(~Year\_of\_Release) + coord\_polar(theta = "y") +  
 theme\_void() + labs(title = "Global Sales by Platform")

 This code generates pie charts which shows the difference in sales between each platform from 2010 to 2016.

# ggplot() is used to create the graph, and set the aesthetics of the graph.  
# geom\_point() is used to create a scatter plot of each platforms sales from  
# 2010 to 2016.  
ggplot(data = platform\_SalesbyYear, aes(x = Year\_of\_Release, y = glbl\_total, color = Platform)) +  
 geom\_point() + geom\_line(aes(group = Platform)) + labs(title = "Global Video Game Sales (2010-2016)",  
 x = "Year", y = "Global Sales\n(in millions)")



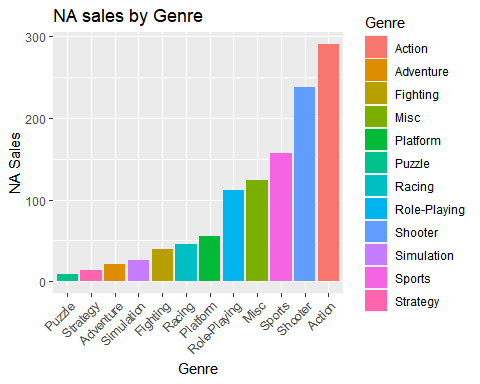
# Geom\_line is used to create a line graph and we set the aesthetic of the  
# graph to group by Platform. labs() is used to change the names of our x and y  
# axis as well as the title.

This code generates a line graph that shows the rate of increase/decrease of each Platform from 2010 to 2016.

Analysis: From our charts we can see that globally PlayStation has dominated with the most sales, and since 2010 the sales of PlayStation has had far more sales compared to other Platforms. We also see that the sales of PC has been low compared to other platforms and that the sales of Nintendo and Xbox has been steadily decreasing, with Nintendo decreasing the most in sales in the past 6 years.

5.3 We now wan to create some visualizations for our third question to better understand the NA sales for each genre in 2010 to 2016.

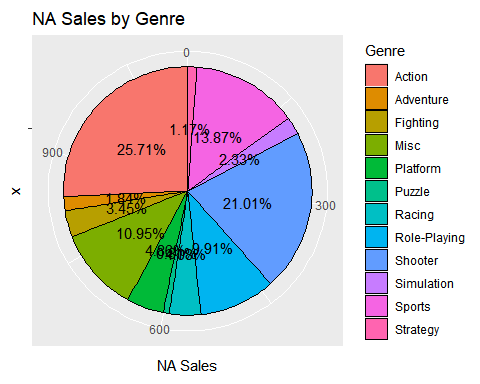
# We use the ggplot2 package to create a Bar graph that easily represents the  
# highest selling video game Genre for North American sales. We use the aes()  
# function to label the axis’, as well as using fill for discernment between  
# Genres, we use the reorder() function to arrange the attributes from least to  
# greatest. geom\_bar(stat=”identity”) makes a bar chart and makes the height  
# proportional to the number of cases in each group. Stat identity displays the  
# sum of values in the Sales(NA) column, grouped by Genre.  
ggplot(data = genre\_na\_sales, aes(x = reorder(Genre, genre\_na\_total), y = genre\_na\_total,  
 fill = Genre)) + geom\_bar(stat = "identity") + labs(title = "NA sales by Genre",  
 x = "Genre", y = "NA Sales") + theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# labs() is used to changes the labels in various parts of the chart. Both the  
# X and Y axis changing their names, as well as the title. theme() helps change  
# non-data elements of a chart. In this case adding a change to Genre’s axis by  
# utilizing element\_text() to change the look of texts, adding an angle and  
# using hjust to control horizontal justification)

This code creates a bar graph that displays the overall sum of sales of each genre between 2010 to 2016.

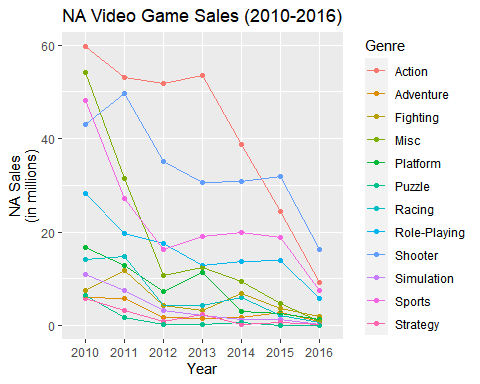
# We convert our numeric values in genre\_na\_sales into a percentage of the  
# entire sum and store it into pie\_labels.  
pie\_labels <- paste0(round(100 \* genre\_na\_sales$genre\_na\_total/sum(genre\_na\_sales$genre\_na\_total),  
 2), "%")  
# We use ggplot() to create the graph, the aes() function set’s the aesthetic.  
# Used nothing for X, but for Y we used NA Sales so that it would fill  
# proportionately.We use the reorder() function to reorder the different  
# Genre's from least to greatest. We then filled it by Genre using a rainbow  
# palette. the geom\_col() function is used to create black lines in between our  
# “Slices”.  
ggplot(genre\_na\_sales, aes(x = "", y = genre\_na\_total, fill = Genre)) + geom\_col(color = "black") +  
 geom\_text(aes(label = pie\_labels), position = position\_stack(vjust = 0.5)) +  
 coord\_polar(theta = "y") + labs(y = "NA Sales", title = "NA Sales by Genre")



# geom\_text is used along with aes() embedded to set labels to pie labels and  
# position them accordingly. coord\_polar is used because a pie chart is  
# actually just a stacked bar chart in polar coordinates. Then, we use the labs  
# function to change the name of the y axis and title on the graph.

This code creates a pie graph, then displays the difference in NA sales each genre made out of 100% of the total sales made between 2010 to 2016

# ggplot() is used to create the graph, and set the aesthetics of the graph.  
# geom\_point() is used to create a scatter plot of each platforms sales from  
# 2010 to 2016.  
ggplot(data = genre\_SalesbyYear, aes(x = Year\_of\_Release, y = NA\_total, color = Genre)) +  
 geom\_point() + geom\_line(aes(group = Genre)) + labs(title = "NA Video Game Sales (2010-2016)",  
 x = "Year", y = "NA Sales\n(in millions)")



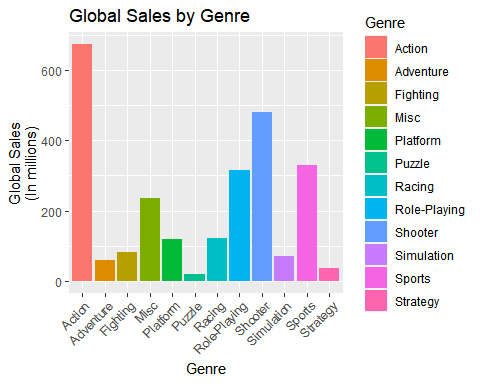
# Geom\_line is used to create a line graph and we set the aesthetic of the  
# graph to group by Genre. labs() is used to change the names of our x and y  
# axis as well as the title.

This code creates a line graph that displays the change of NA sales based on each genre over the past 6 years.

Analysis: Overall, we assume that Action is the most profitable video game genre compared to the other games, however the sales of action has dramatically decreased since 2013. While Shooter games has also decreased it was the most profitable video game genre by 2016.

5.4 We now see the total global revenue based on Genre to get a visual for which Genre was more profitable all throughout 2010-2016.

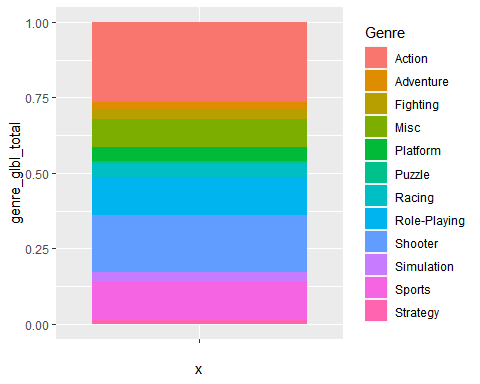
# We use the ggplot2 package to create a Bar graph that easily represents the  
# highest globally selling video game genre. We use the aes() function to label  
# the axis’, as well as using fill for discernment between Genres,  
# geom\_bar(stat=”identity”) makes a bar chart and makes the height proportional  
# to the number of cases in each group. Stat identity displays the sum of  
# values in the Sales(NA) column, grouped by Genre  
ggplot(data = genre\_glbl\_sales, aes(x = Genre, y = genre\_glbl\_total, fill = Genre)) +  
 geom\_bar(stat = "identity") + labs(title = "Global Sales by Genre", x = "Genre",  
 y = "Global Sales\n(In millions)") + theme(axis.text.x = element\_text(angle = 45,  
 hjust = 1))



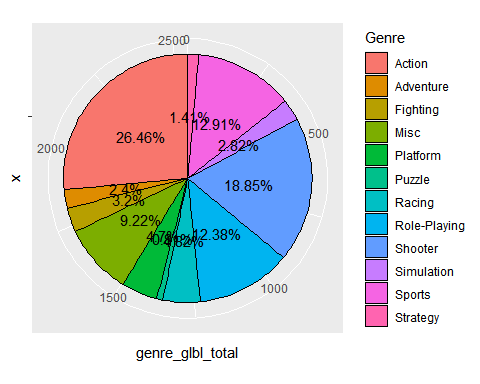
# labs() is used to changes the labels in various parts of the chart. Both the  
# X and Y axis changing their names, as well as the title. theme() helps change  
# non-data elements of a chart. In this case adding a change to Genre’s axis by  
# utilizing element\_text() to change the look of texts, adding an angle and  
# using hjust to control horizontal justification

This code generates a bar graph that displays the total Global sales of each video game genre between 2010 to 2016.

# We convert our numerical values into percentages, then store into pvgsg.  
PVGSG <- genre\_glbl\_sales %>%  
 mutate(percentage = paste0(round(genre\_glbl\_total/sum(genre\_glbl\_total) \* 100,  
 2), "%"))  
PVGSG <- PVGSG %>%  
 arrange(desc(genre\_glbl\_total))  
  
# We use ggplot() to create a bar chart, then use aes() to set the parameters  
# for x and y axis, and use geom\_bar() to create a bar chart and then set the  
# style for to fill.  
ggplot(PVGSG, aes(fill = Genre, y = genre\_glbl\_total, x = "")) + geom\_bar(position = "fill",  
 stat = "identity")

 This code displays a stacked bar graph with a couple of functions at play.

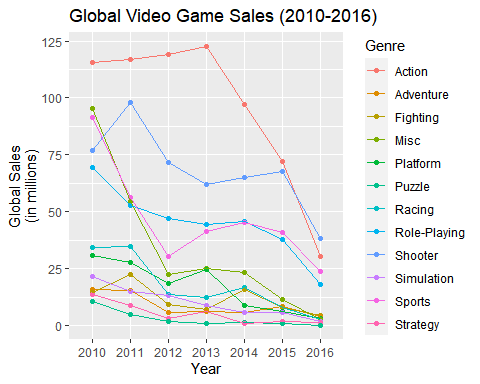
# We convert our numeric values in genre\_glbl\_sales into a percentage of the  
# entire sum and store it into pie\_labels2.  
pie\_labels2 <- paste0(round(100 \* genre\_glbl\_sales$genre\_glbl\_total/sum(genre\_glbl\_sales$genre\_glbl\_total),  
 2), "%")  
# We use ggplot() to create the graph, the aes() function set’s the aesthetic.  
# Used nothing for X, but for Y we used genre\_glbl\_total so that it would fill  
# proportionately.We use the reorder() function to reorder the different  
# Genre's from least to greatest. We then filled it by Genre using a rainbow  
# palette. the geom\_col() function is used to create black lines in between our  
# “Slices”.  
ggplot(genre\_glbl\_sales, aes(x = "", y = genre\_glbl\_total, fill = Genre)) + geom\_col(color = "black") +  
 geom\_text(aes(label = pie\_labels2), position = position\_stack(vjust = 0.5)) +  
 coord\_polar(theta = "y")



# geom\_text is used along with aes() embedded to set labels to pie labels and  
# position them accordingly. coord\_polar is used because a pie chart is  
# actually just a stacked bar chart in polar coordinates. Then, we use the labs  
# function to change the name of the y axis and title on the graph.

This code creates a pie chart in which it displays the sum of global sale of video games between 2010 to 2016 in a percentage of the total sales made by genre.

# ggplot() is used to create the graph, and set the aesthetics of the graph.  
# geom\_point() is used to create a scatter plot of each platforms sales from  
# 2010 to 2016.  
ggplot(data = genre\_SalesbyYear, aes(x = Year\_of\_Release, y = glbl\_total, color = Genre)) +  
 geom\_point() + geom\_line(aes(group = Genre)) + labs(title = "Global Video Game Sales (2010-2016)",  
 x = "Year", y = "Global Sales\n(in millions)")



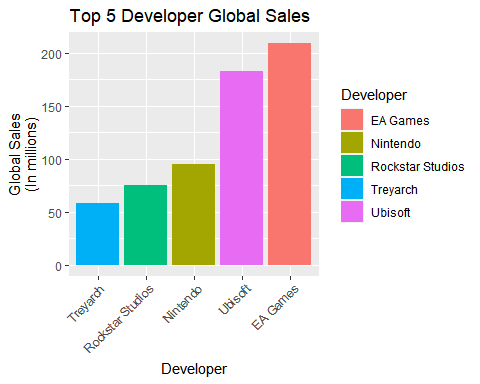
# Geom\_line is used to create a line graph and we set the aesthetic of the  
# graph to group by Genre. labs() is used to change the names of our x and y  
# axis as well as the title.

This line creates a line chart that shows the decrease/increase of gloabl video game sales each year.

Analysis: Just like our previous analysis with question 3, we see that similar to NA sales Action is also the most profitable genre, however over the course of 6 years the sales of Action games have decreased dramatically and eventually shooter has the most sales in 2016.

5.5 We know the global and NA revenue for each of the top five developers, but now we want to create some visuals, to get a better understanding of their profit within 2010 to 2016.

# We use the ggplot2 package to create our graph, the aes() function is used to  
# set the aesthetics of the plot. we set our x value to Developer and our Y  
# value to NA sale. we also use the reorder() function to reorder the graph  
# from smallest to greatest.  
  
ggplot(dev\_global, aes(x = reorder(Developer, global\_total), y = global\_total, fill = Developer)) +  
 geom\_bar(stat = "identity") + theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
 labs(title = "Top 5 Developer Global Sales", x = "Developer", y = "Global Sales\n(In millions)")

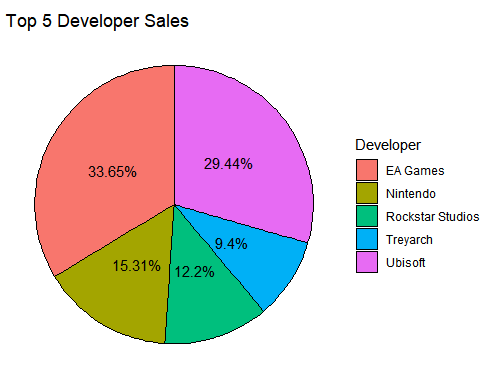


# The geom\_bar() function creates a bar plot using the 'identity' statistical  
# transformation.The theme() function is used to customize the appearance of  
# the plot. The axis.text.x argument is set to element\_text() to change the  
# angle of the x-axis labels to 45 degrees and set the horizontal justification  
# to 1 (right align).The labs() function sets the plot title to 'Top 5  
# Developers by Global Sales', and labels the x-axis 'Developer' and the y-axis  
# 'Global Game Sales (in millions)'.

Overall, this code generates a bar chart that ranks the top 5 video game developers by their total global sales across all regions. The bars are ordered by the sum of sales from all regions for each developer. The x-axis labels are angled and right aligned for better readability.

# we convert our numeric values in genre\_glbl\_sales into a percentage of the  
# entire sum and store it into pie\_labels2.  
pie\_labels3 <- paste0(round(100 \* dev\_global$global\_total/sum(dev\_global$global\_total),  
 2), "%")  
# The aes() function is setting the aesthetics of the plot. The x-axis is set  
# to an empty string, which means there will be no x-axis label. The y-axis is  
# set to the sum of sales from all regions. The fill aesthetic is set to the  
# Developer column. The geom\_col() function creates a pie chart using the  
# 'identity' statistical transformation. The width argument is set to 1 to  
# remove the space between the bars, and the color argument is set to 'black'  
# to add a white border around each bar. The coord\_polar() function is used to  
# convert the plot to a polar coordinate system. The 'y' argument specifies  
# that the y-axis values should be used to determine the radial distance of  
# each bar from the center of the plot. The start argument is set to 0 to align  
# the first bar with the 12 o'clock position.  
ggplot(dev\_global, aes(x = "", y = global\_total, fill = Developer)) + geom\_col(stat = "identity",  
 width = 1, color = "black") + coord\_polar(theta = "y") + theme\_void() + geom\_text(aes(label = pie\_labels3),  
 position = position\_stack(vjust = 0.5)) + labs(title = " Top 5 Developer Sales")

## Warning in geom\_col(stat = "identity", width = 1, color = "black"): Ignoring  
## unknown parameters: `stat`



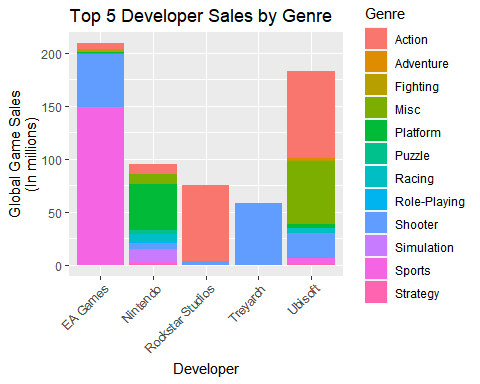
# The geom\_text() function is used to sum up the percentage each of these five  
# Developers make globally. The theme\_void() function removes all the axis  
# labels, ticks, and grid lines, leaving only the bars.The labs() function sets  
# the plot title to 'Top 5 Developer Sales'.

Overall, this code generates a pie chart that ranks the top 5 video game developers by their total global sales across all regions. The bars are arranged radially, with the outermost bar representing the developer with the highest total sales. The plot has no axis labels or grid lines, giving it a minimalist look.

Analysis: The first bar graph shows the order of the top 5 game developers in terms of the number of game copies sold. In ascending order, the top 5 game developers are Ubisoft, EA Sports, Nintendo, Treyarch, and Rockstar North. The pie chart provides another visualization for this ranking.

5.6 We are now going to create some visuals for our sixth question in order to analyze our findings. developer\_genre\_sales

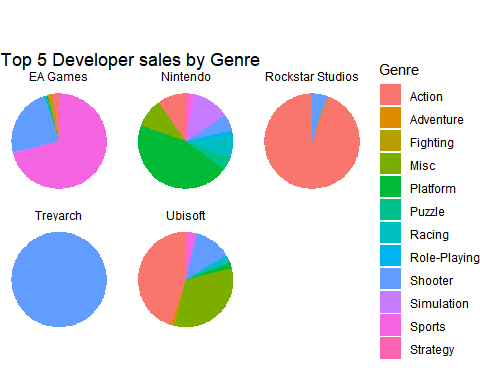
# We use the ggplot2 package in R to show the top 5 video game genres for each  
# of the top 5 video game developers in terms of global sales. The aes()  
# function is used to set the aesthetics of the plot. The x-axis is set to the  
# Developer column, the y-axis is set to the Global\_Total column, and the fill  
# aesthetic is set to the Genre column. The geom\_bar() function creates a bar  
# plot using the 'identity' statistical transformation, which means the bar  
# heights correspond to the values in the Global\_Total column.  
ggplot(df, aes(x = Developer, y = developer\_glbl\_total, fill = Genre)) + geom\_bar(stat = "identity") +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) + labs(title = "Top 5 Developer Sales by Genre",  
 x = "Developer", y = "Global Game Sales\n(In millions)")



# The theme() function sets the x-axis text to a 45-degree angle for better  
# readability, and the labs() function sets the plot title, x-axis label, and  
# y-axis label.

Overall, this code generates a stacked bar chart that shows the contribution of each video game genre to the global sales of the top 5 video game developers. Each bar is divided into segments corresponding to the sales of each genre, and the height of each bar corresponds to the total global sales of the developer.

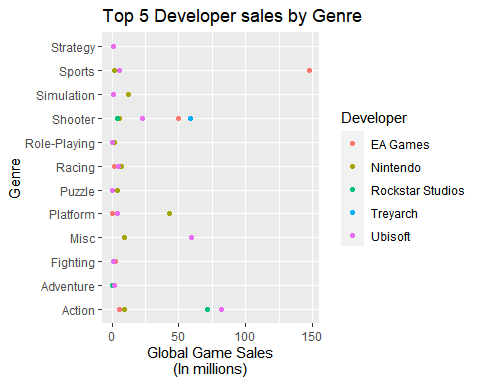
# The aes() function is used to set the aesthetics of the plot. The x-axis is  
# set to 0, the y-axis is set to the Global\_Total column, and the fill  
# aesthetic is set to the Genre column. The geom\_col() function creates a  
# column plot with position set to 'fill', which stacks the bars so that each  
# bar fills the available vertical space.  
ggplot(data = df, aes(x = 0, y = developer\_glbl\_total, fill = Genre)) + geom\_col(position = "fill") +  
 facet\_wrap(~Developer) + coord\_polar(theta = "y") + theme\_void() + labs(title = "Top 5 Developer sales by Genre")



# The facet\_wrap() function is used to split the plot into multiple panels, one  
# for each developer. The coord\_polar() function converts the plot to polar  
# coordinates, which makes it easier to compare the relative contributions of  
# each video game genre across the different developers. The theme\_void()  
# function sets the plot to have no background or axis labels, and the labs()  
# function sets the plot title.

Overall, this code generates a polar bar chart that shows the relative contribution of each video game genre to the global sales of the top 5 video game developers. The chart is split into multiple panels, one for each developer, making it easy to compare the contribution of each genre across the different developers.

# The aes() function is used to set the aesthetics of the plot. The x-axis is  
# set to Genre, the y-axis is set to the Global\_Total column, and the color  
# aesthetic is set to the Developer column. The geom\_point() function creates a  
# scatter plot graph.  
ggplot(data = df, aes(x = Genre, y = developer\_glbl\_total, colour = Developer)) +  
 geom\_point() + labs(title = "Top 5 Developer sales by Genre", x = "Genre", y = "Global Game Sales\n(In millions)") +  
 coord\_flip()



# We use the labs() function to label our title, x-axis, and y-axis. The  
# coordflip() flips the x and y axis, so that our x-axis value appear on the  
# y-axis and our y-axis values appear on the x-axis.

Overall this code creates a scatter plot that shows how most categories are similar in sales. However it also shows that When it comes to Sports games, EA Games make far more profit then any other categories.

Analysis: These visualizations represent the sales the top 5 game developers have earned per video game genre. Rockstar North and Treyarch develop exclusively Action and Shooter games respectively, meaning they have only sold games in those genres. EA Sports is almost the same however, they have developed at least one game in the racing genre. Nonetheless, most of the game copies EA Sports has sold are in the sports genre. Ubisoft has more diversity than the previous 3 developers, but it is visible that they have sold mostly action games or games that do not fall in any of the named genres. Nintendo is visibly the most diverse out of the 5 developers in terms of genre, with their most successful genre being platforms.

##6. Modelling

After creating our visualization what we want to do is create some models in order to understand the relationship between the models. We will be using five different models in this section, The models used will be Linear Regression, Multiple Linear Regression, Random Forest, Support Vector Regression (SVR),and Support Vector Machine (SVM).

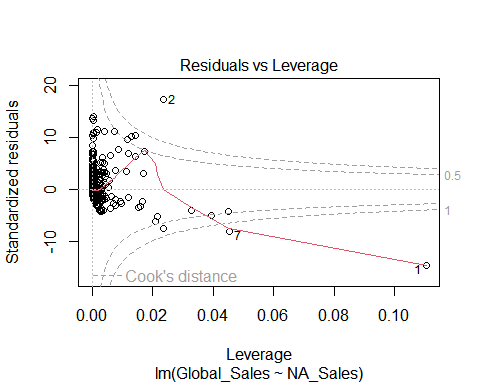
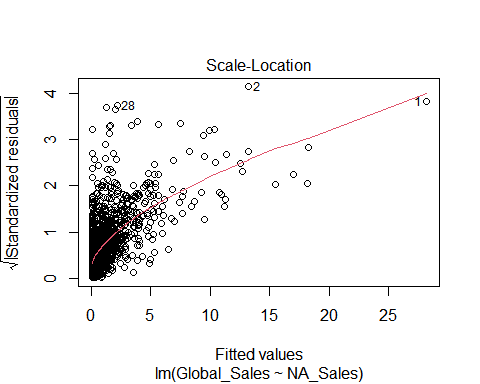
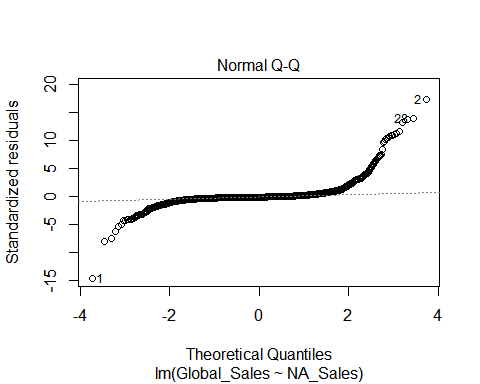
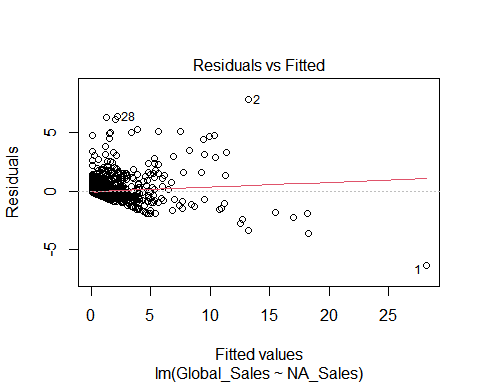
6.1 Linear Regression

The first model we will be using is linear regression and we will be using linear regression to establish a relationship between our dependent variable Global\_Sales with our independent variable NA\_Sales. What we want to predict is if we can calculate the Global\_Sales based on the NA\_sales. to start we first create a model using linear regression

# Using lm() to create a linear regression model using Global\_Sales as  
# dependent variable and NA\_sales as independent variable.  
global\_na\_sales\_model <- lm(formula = Global\_Sales ~ NA\_Sales, data = gamesales2010\_2016)  
# Summary() function used to get a summary of the model created.  
summary(global\_na\_sales\_model)

##   
## Call:  
## lm(formula = Global\_Sales ~ NA\_Sales, data = gamesales2010\_2016)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -6.3339 -0.0990 -0.0538 0.0146 7.8253   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.081547 0.006709 12.15 <2e-16 \*\*\*  
## NA\_Sales 1.870823 0.010331 181.09 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4601 on 5275 degrees of freedom  
## Multiple R-squared: 0.8614, Adjusted R-squared: 0.8614   
## F-statistic: 3.279e+04 on 1 and 5275 DF, p-value: < 2.2e-16

plot(global\_na\_sales\_model)



We can see that the p-value is less then our alpha 0.05 meaning that our independent variable is significant in our data set. we also see that our R-Squared is 86% which means that our model is a good fit for our data set.

We now will calculate the percentage error to see the error rate of this model. First we will find the Residual Standard Error(RSE) of the model then we will calculate the the mean of Global\_Sales. After which we divide our RSE by the mean and multiply by 100 to find our percentage error as shown in the code below.

# We calculate the RSE using sigma.  
rse <- summary(global\_na\_sales\_model)$sigma  
# We calculate the mean of Global\_Sales.  
y\_mean <- mean(gamesales2010\_2016$Global\_Sales)  
# We calculate our percentage error.  
perc\_error <- (rse/y\_mean) \* 100  
  
print(perc\_error)

## [1] 95.39015

After calculating the percentage error, we see that we have a high percentage error which indicates that our model is not accurate and is not good to help us predict our data set.

We are going to create another linear regression model where our dependent variable is NA\_Sales and our independent variable is Platform.

# Create a linear Regression model using Global\_Sales as dependent variable and  
# Platform as independent variable.  
lg\_model <- lm(formula = Global\_Sales ~ Platform, data = gamesales2010\_2016)  
summary(lg\_model)

##   
## Call:  
## lm(formula = Global\_Sales ~ Platform, data = gamesales2010\_2016)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.7558 -0.4052 -0.3052 -0.0652 21.0442   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.42520 0.03055 13.919 < 2e-16 \*\*\*  
## PlatformPC -0.16996 0.06388 -2.660 0.00783 \*\*   
## PlatformPlayStation 0.02970 0.04002 0.742 0.45804   
## PlatformXbox 0.34065 0.05059 6.733 1.84e-11 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.228 on 5273 degrees of freedom  
## Multiple R-squared: 0.01318, Adjusted R-squared: 0.01262   
## F-statistic: 23.47 on 3 and 5273 DF, p-value: 4.355e-15

After performing a linear regression we see that the model is significant as it is less then our alpha and that Xbox and PC are also significant as they are less then our alpha 0.05, however PlayStation is not significant in the model as its p-value is larger then our alpha.

Our R-squared is also very low showing that this model is not a good fit for our data set showing that linear regression is not a good model for this data set.

6.2 Multiple Linear Regression

We have tested the linear regression model, however now we are going to test the multiple linear regression model on our data set on specific independent variables such as Genre, Year\_of\_Release, and Platform. Since Developer has to many values we will restrict it from our model.

mreg <- lm(formula = Global\_Sales ~ Genre + Year\_of\_Release + Platform, data = gamesales2010\_2016)  
summary(mreg)

##   
## Call:  
## lm(formula = Global\_Sales ~ Genre + Year\_of\_Release + Platform,   
## data = gamesales2010\_2016)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.4844 -0.4115 -0.2275 0.0016 21.1845   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.42847 0.05074 8.444 < 2e-16 \*\*\*  
## GenreAdventure -0.33325 0.06038 -5.520 3.56e-08 \*\*\*  
## GenreFighting -0.07760 0.09127 -0.850 0.395244   
## GenreMisc -0.04880 0.06084 -0.802 0.422496   
## GenrePlatform 0.33299 0.10307 3.231 0.001242 \*\*   
## GenrePuzzle -0.20953 0.11943 -1.754 0.079415 .   
## GenreRacing 0.05764 0.08483 0.680 0.496837   
## GenreRole-Playing 0.11824 0.05966 1.982 0.047537 \*   
## GenreShooter 0.69947 0.06844 10.221 < 2e-16 \*\*\*  
## GenreSimulation -0.05293 0.08919 -0.593 0.552885   
## GenreSports 0.09503 0.05980 1.589 0.112116   
## GenreStrategy -0.14346 0.09954 -1.441 0.149587   
## Year\_of\_Release2011 -0.04196 0.04961 -0.846 0.397687   
## Year\_of\_Release2012 0.04443 0.05940 0.748 0.454492   
## Year\_of\_Release2013 0.12057 0.06273 1.922 0.054674 .   
## Year\_of\_Release2014 0.05487 0.06176 0.888 0.374344   
## Year\_of\_Release2015 -0.06118 0.06126 -0.999 0.317995   
## Year\_of\_Release2016 -0.26361 0.06533 -4.035 5.54e-05 \*\*\*  
## PlatformPC -0.22466 0.06544 -3.433 0.000601 \*\*\*  
## PlatformPlayStation 0.03198 0.04210 0.760 0.447477   
## PlatformXbox 0.24587 0.05211 4.718 2.44e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.203 on 5256 degrees of freedom  
## Multiple R-squared: 0.05561, Adjusted R-squared: 0.05202   
## F-statistic: 15.48 on 20 and 5256 DF, p-value: < 2.2e-16

In this model we are using multiple different attributes so we look at our adjusted R-squared, from looking at the adjusted R-squared we see that our R-squared is only 5%. This means the model is not a good fit for our data set and we do not have to proceed any further as multiple linear regression is not a good fit for our data set.

6.3 Random Forest Model

Now, We will attempt the Random forest model in order to see if we can use our data set to make predictions. First we will need to load two different packages

# Load randomForest package and caret packages  
library(randomForest)

## Warning: package 'randomForest' was built under R version 4.2.3

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(caret)

## Warning: package 'caret' was built under R version 4.2.3

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

After we will need to make sure we omit all N/A values and split the data set into our train data and our test data. Then we train our model. After we train our model we will test our model to evaluate its performance and identify any problems such as overfitting.

g <- gamesales2010\_2016  
# Remove N/A values  
g <- na.omit(g)  
# Make the Genre Column a factor  
g$Genre <- as.factor(g$Genre)  
  
  
set.seed(123)  
# Split the data into train and test  
trainIndex <- createDataPartition(g$Global\_Sales, p = 0.8, list = FALSE)  
train <- g[trainIndex, ]  
test <- g[-trainIndex, ]  
# Run the RandomForest model  
model <- randomForest(Genre ~ ., data = train, importance = TRUE, ntree = 30)  
# Create a prediction  
predictions <- predict(model, test)  
# use Confusion Matrix  
confusionMatrix(predictions, test$Genre)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Action Adventure Fighting Misc Platform Puzzle Racing  
## Action 86 7 15 15 8 1 10  
## Adventure 0 0 1 0 0 0 0  
## Fighting 2 1 2 1 0 0 0  
## Misc 2 0 0 4 1 0 0  
## Platform 1 0 0 2 1 0 0  
## Puzzle 1 0 0 1 0 0 0  
## Racing 6 1 0 0 3 0 8  
## Role-Playing 7 2 1 1 1 0 1  
## Shooter 24 4 4 2 2 0 4  
## Simulation 0 0 0 1 1 0 0  
## Sports 1 2 0 5 1 1 3  
## Strategy 0 0 0 0 0 0 0  
## Reference  
## Prediction Role-Playing Shooter Simulation Sports Strategy  
## Action 27 26 8 11 5  
## Adventure 1 0 0 0 0  
## Fighting 2 0 0 0 0  
## Misc 0 0 0 1 0  
## Platform 1 0 1 0 0  
## Puzzle 0 0 0 0 0  
## Racing 0 0 1 17 2  
## Role-Playing 13 1 0 0 0  
## Shooter 10 31 1 3 2  
## Simulation 0 0 0 2 0  
## Sports 0 0 1 9 1  
## Strategy 0 0 0 0 1  
##   
## Overall Statistics  
##   
## Accuracy : 0.3638   
## 95% CI : (0.3181, 0.4115)  
## No Information Rate : 0.3052   
## P-Value [Acc > NIR] : 0.005486   
##   
## Kappa : 0.1974   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: Action Class: Adventure Class: Fighting Class: Misc  
## Sensitivity 0.6615 0.000000 0.086957 0.12500  
## Specificity 0.5507 0.995110 0.985112 0.98985  
## Pos Pred Value 0.3927 0.000000 0.250000 0.50000  
## Neg Pred Value 0.7874 0.959906 0.949761 0.93301  
## Prevalence 0.3052 0.039906 0.053991 0.07512  
## Detection Rate 0.2019 0.000000 0.004695 0.00939  
## Detection Prevalence 0.5141 0.004695 0.018779 0.01878  
## Balanced Accuracy 0.6061 0.497555 0.536034 0.55742  
## Class: Platform Class: Puzzle Class: Racing  
## Sensitivity 0.055556 0.000000 0.30769  
## Specificity 0.987745 0.995283 0.92500  
## Pos Pred Value 0.166667 0.000000 0.21053  
## Neg Pred Value 0.959524 0.995283 0.95361  
## Prevalence 0.042254 0.004695 0.06103  
## Detection Rate 0.002347 0.000000 0.01878  
## Detection Prevalence 0.014085 0.004695 0.08920  
## Balanced Accuracy 0.521650 0.497642 0.61635  
## Class: Role-Playing Class: Shooter Class: Simulation  
## Sensitivity 0.24074 0.53448 0.00000  
## Specificity 0.96237 0.84783 0.99034  
## Pos Pred Value 0.48148 0.35632 0.00000  
## Neg Pred Value 0.89724 0.92035 0.97156  
## Prevalence 0.12676 0.13615 0.02817  
## Detection Rate 0.03052 0.07277 0.00000  
## Detection Prevalence 0.06338 0.20423 0.00939  
## Balanced Accuracy 0.60155 0.69115 0.49517  
## Class: Sports Class: Strategy  
## Sensitivity 0.20930 0.090909  
## Specificity 0.96084 1.000000  
## Pos Pred Value 0.37500 1.000000  
## Neg Pred Value 0.91542 0.976471  
## Prevalence 0.10094 0.025822  
## Detection Rate 0.02113 0.002347  
## Detection Prevalence 0.05634 0.002347  
## Balanced Accuracy 0.58507 0.545455

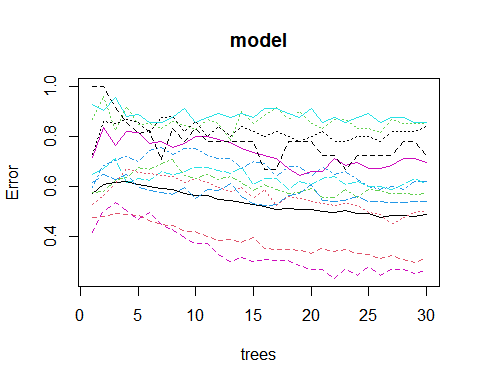
mean(predictions == test$Genre)

## [1] 0.3638498

summary(model)

## Length Class Mode   
## call 5 -none- call   
## type 1 -none- character  
## predicted 1712 factor numeric   
## err.rate 390 -none- numeric   
## confusion 156 -none- numeric   
## votes 20544 matrix numeric   
## oob.times 1712 -none- numeric   
## classes 12 -none- character  
## importance 210 -none- numeric   
## importanceSD 195 -none- numeric   
## localImportance 0 -none- NULL   
## proximity 0 -none- NULL   
## ntree 1 -none- numeric   
## mtry 1 -none- numeric   
## forest 14 -none- list   
## y 1712 factor numeric   
## test 0 -none- NULL   
## inbag 0 -none- NULL   
## terms 3 terms call

plot(model)

 From looking at the results above it can be see that Random Forest modelling is not a good model for our data set either. So we are not able to use this model to help us make predictions or classifications on the data set.

6.4 Support Vector Regression (SVR)

We will now try to use SVR to make predictions on our model. To do this what we do is add a new library, then we split our data set like we did with Random Forest. Then we will make all attributes numeric and attempt to do a linear kernel model and then we analyze the model.

library(e1071)

## Warning: package 'e1071' was built under R version 4.2.3

# Creating a separate data set with Year\_of\_release and Global\_Sales values  
gamesales2010\_2016\_model <- gamesales2010\_2016[, c("Year\_of\_Release", "Global\_Sales")]  
  
# Partition data into test and training by 80:20  
set.seed(123)  
train\_index <- sample(nrow(gamesales2010\_2016\_model), nrow(gamesales2010\_2016\_model) \*  
 0.8)  
train\_data <- gamesales2010\_2016\_model[train\_index, ]  
test\_data <- gamesales2010\_2016\_model[-train\_index, ]  
  
# Making sure all the values in the Year\_of\_Release column are numberic  
gamesales2010\_2016\_model$Year\_of\_Release <- as.numeric(gamesales2010\_2016\_model$Year\_of\_Release)  
  
# Calling a tune function to find optimal parameters for the model  
tuned <- tune(svm, Global\_Sales ~ Year\_of\_Release, data = train\_data, kernel = "radial",  
 ranges = list(cost = c(0.1, 1, 10), gamma = c(0.01, 0.1, 1)))  
# Building a model with radial kernel and tuned cost and gamma parameters  
model <- svm(Global\_Sales ~ Year\_of\_Release, data = train\_data, kernel = "radial",  
 cost = tuned$best.parameters$cost, gamma = tuned$best.parameters$gamma, type = "eps-regression")  
summary(model)

##   
## Call:  
## svm(formula = Global\_Sales ~ Year\_of\_Release, data = train\_data,   
## kernel = "radial", cost = tuned$best.parameters$cost, gamma = tuned$best.parameters$gamma,   
## type = "eps-regression")  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 1   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 2709

# Attempting a linear kernel model  
model2 <- svm(Global\_Sales ~ Year\_of\_Release, data = train\_data, kernel = "linear")  
summary(model2)

##   
## Call:  
## svm(formula = Global\_Sales ~ Year\_of\_Release, data = train\_data,   
## kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: linear   
## cost: 1   
## gamma: 0.1428571   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 2709

# Attempting a polynomial kernel with degree of 2  
model3 <- svm(Global\_Sales ~ Year\_of\_Release, data = train\_data, kernel = "polynomial",  
 degree = 2)  
summary(model3)

##   
## Call:  
## svm(formula = Global\_Sales ~ Year\_of\_Release, data = train\_data,   
## kernel = "polynomial", degree = 2)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: polynomial   
## cost: 1   
## degree: 2   
## gamma: 0.1428571   
## coef.0: 0   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 2713

pred <- predict(model3, newdata = test\_data)  
  
  
# Analyzing Metrics  
mse <- mean((pred - test\_data$Global\_Sales)^2)  
rsq <- 1 - sum((test\_data$Global\_Sales - pred)^2)/sum((test\_data$Global\_Sales - mean(test\_data$Global\_Sales))^2)  
mae <- mean(abs(test\_data$Global\_Sales - pred))  
print(paste0("Mean Squared Error: ", mse))

## [1] "Mean Squared Error: 1.67536601872764"

cat("R-squared:", rsq, "\n")

## R-squared: -0.06044289

cat("Mean Absolute Error:", mae, "\n")

## Mean Absolute Error: 0.4385891

From all the information above it is evident that this model does not fit our data. From doing SVR there is nothing to learn from our model. We can also see that our R-squared is 6% indicating that this model is not fit for our data set.

6.5 SVM

We are creating an SVM in order to help us predict our Genre based on sales, the code below shows the steps taken.

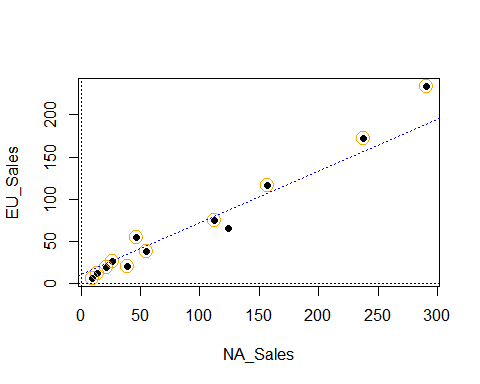
VGSales <- gamesales2010\_2016  
VGSales1 <- VGSales %>%  
 filter(Genre != "")  
  
# Aggregate Sum of NA and EU Sales  
VGSASum <- aggregate(cbind(NA\_Sales, EU\_Sales) ~ Genre, VGSales1, FUN = sum)  
  
# Create New Class Definition of 'Action Type' and 'Strategy Type'  
VGSASum %>%  
 mutate(Class = c(1, 1, 1, -1, -1, -1, 1, -1, 1, -1, 1, -1))

## Genre NA\_Sales EU\_Sales Class  
## 1 Action 290.64 233.63 1  
## 2 Adventure 20.84 18.88 1  
## 3 Fighting 39.05 20.33 1  
## 4 Misc 123.80 66.09 -1  
## 5 Platform 54.90 38.30 -1  
## 6 Puzzle 9.10 6.58 -1  
## 7 Racing 46.11 54.75 1  
## 8 Role-Playing 112.05 75.48 -1  
## 9 Shooter 237.47 171.45 1  
## 10 Simulation 26.39 26.39 -1  
## 11 Sports 156.81 116.84 1  
## 12 Strategy 13.25 12.49 -1

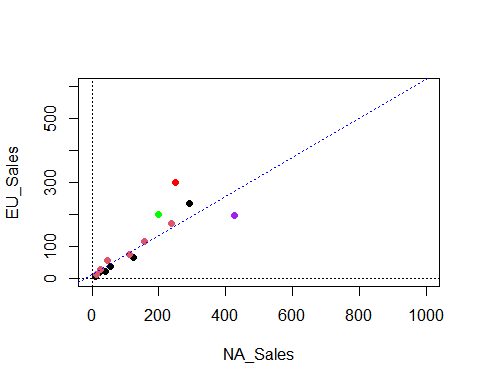
View(VGSASum)  
  
  
VGSASum <- VGSASum %>%  
 mutate(Class = c(1, 1, 1, -1, -1, -1, 1, -1, 1, -1, 1, -1))  
  
# Create Action Definition Variable  
isAction <- c(rep(-1, 6), rep(+1, 6))  
  
# Create Data Frame For Model  
my.data <- data.frame(NA\_Sales = VGSASum["NA\_Sales"], EU\_Sales = VGSASum["EU\_Sales"],  
 Class = as.factor(isAction))  
View(my.data)  
  
  
# Plot Data  
plot(my.data[, -3], col = (3)/2, pch = 19)  
abline(h = 0, v = 0, lty = 3)  
  
# Create SVM Model  
svm.model <- svm(Class ~ ., data = my.data, type = "C-classification", kernel = "linear",  
 scale = FALSE)  
summary(svm.model)

##   
## Call:  
## svm(formula = Class ~ ., data = my.data, type = "C-classification",   
## kernel = "linear", scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 11  
##   
## ( 5 6 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## -1 1

# Plot Points for Decision Paramters  
points(my.data[svm.model$index, c(1, 2)], col = "orange", cex = 2)  
  
w <- t(svm.model$coefs) %\*% svm.model$SV  
b <- -svm.model$rho  
abline(a = -b/w[1, 2], b = -w[1, 1]/w[1, 2], col = "blue", lty = 3)



# Set Test Data  
observations <- data.frame(NA\_Sales = c(200, 250, 425), EU\_Sales = c(200, 300, 195))  
  
# Plot Test Data  
plot(my.data[, -3], col = (isAction + 3)/2, pch = 19, xlim = c(0, 1000), ylim = c(0,  
 600))  
abline(h = 0, v = 0, lty = 3)  
points(observations[1, ], col = "green", pch = 19)  
points(observations[2, ], col = "red", pch = 19)  
points(observations[3, ], col = "purple", pch = 19)  
  
# Predict Data (Data is correct here, small mistake led to inversion of 1 and  
# -1 as assingments. Prediciton is correct  
abline(a = -b/w[1, 2], b = -w[1, 1]/w[1, 2], col = "blue", lty = 3)



predict(svm.model, observations)

## 1 2 3   
## 1 1 -1   
## Levels: -1 1

View(my.data)  
View(VGSASum)  
View(my.data)

Like the other models this model was terrible for this data set. In order to actually use this data set to create predictions using SVM, the data had to be aggregated and controlled to a point that would allow the model to make correct predictions. Though we had been able to make the model make some predictions, this model is not efficient for this data set at all.

##7. Conclusion

To sum up what we have done, from exploring this data set we can confirm that we are unable to use modelling to make any predictions or classifications. This shows that there exist some data sets which modelling can not help with. The reason for the failure in modelling is because there is no relation between any of the columns, this makes it hard to determine if we can use any dependent variables to determine our independent variable or vice versa.

Although we can see that modelling does not work on this data set, one thing that can be seen is that this data set works really well with visualizations. Though we can not use modelling to predict in this data set we are able to use visualizations to show the trends between different attributes in our data set. We are also able to determine different things from our data set as well.

In order to solve our clients problem, we attempted to use modelling to make predictions but failed as modelling does not fit this data set. However we were able to use visualizations, in these visualizations we were able to get some insight from the data and was able to determine that though Action games are the most profitable games between 2010 to 2016 in NA Sales and Global Sales we see a huge trend of the sales of Action Games declining, and if the client is interested in investing based on a genre the best genre to invest in would be Shooter since compared to other genres its sales globally and in North America has not decreased and was the most profitable genre in 2016.

We can also conclude that if the Developer is interested in which Platform is the most profitable to invest in we would suggest that PlayStation would be the best. The reason being that Though Xbox had the highest sales as a whole between 2010 and 2016, compared to PlayStation, the video game sales that are based on the platform Xbox has decreased dramatically while PlayStation has not decreased as much and by 2016 had been having more video game sales compared to other Platforms.

Also from what we have gathered about the game developers, though we found that EA Games had made the most global sales, we suggest it would be better to invest in Ubisoft which had the second highest sales. This conclusion was based of the fact that Though EA Games made the most money between 2010 to 2016, EA Games profit is mainly from Sports games rather then other different genres. So if a year were to happen where sports games were not popular, EA Games video game Sales would decreases dramatically. Unlike EA Games, Ubisoft has a diversified portfolio where if one video game genre is not doing too well it will not greatly affect its game sales.