Video Game Sales Investment Study

March 28th, 2023

## 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.

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 1.0.1   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.5.0   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

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.

Video\_Games\_Sales <- read\_csv("C:/Users/Matthew Afoakwah/OneDrive/Documents/Bport/Bport Spring 2023/CSC435 Data Analytics/Project/CSC435\_Video\_Games\_Sales\_Project/data/Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv") #Reads the CSV file from the file location and imports the file into R, named Video\_Games\_Sales

## 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.

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

## # 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

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

## 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.

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

## [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

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

## [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.

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

## 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.

gamesales2010\_2016 <- filter(Video\_Games\_Sales, Year\_of\_Release >= 2010) #We 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  
   
dim(gamesales2010\_2016) #We use the dim() function and see that the amount of rows in the data set has decreased from 16719 to 5550.

## [1] 5550 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] 5281

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.

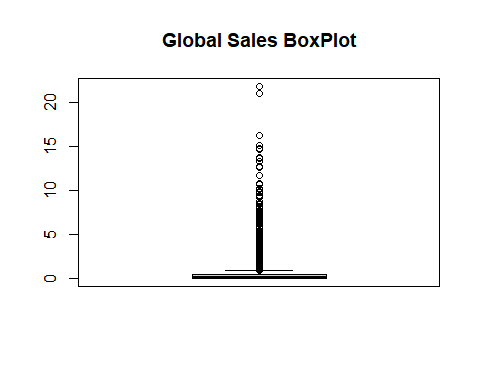
#We use the duplicated() to find all rows where they are repeated occurences then we use the ! to filter them out from the graph.  
gamesales2010\_2016 <- gamesales2010\_2016[!duplicated(gamesales2010\_2016), ]  
#We use #nrow() function to see the amount of rows left in the data set.  
nrow(gamesales2010\_2016)

## [1] 5281

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.

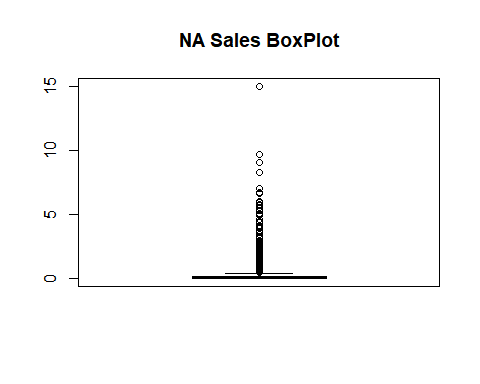
boxplot(gamesales2010\_2016$Global\_Sales, main = "Global Sales BoxPlot")



outliers <- boxplot.stats(gamesales2010\_2016$Global\_Sales)$out

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.

boxplot(gamesales2010\_2016$NA\_Sales, main = "NA Sales BoxPlot")



outliersNA <- boxplot.stats(gamesales2010\_2016$NA\_Sales)$out

Just like our Global Sales, we see that there are not many outliers in the NA sales either, because of how big the data set is having these outliers will hardly impact the progress of our data set and allows us to keep these outliers and move on to our next step.

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

## [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 not have any irregularities that might need to be rewritten and we can now continue to answer the first question.

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

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

## # A tibble: 6 × 2  
## Platform NA\_total  
## <chr> <dbl>  
## 1 X360 334.   
## 2 PS3 229.   
## 3 Wii 121.   
## 4 PS4 109.   
## 5 XOne 93.1  
## 6 3DS 82.6

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

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

## # A tibble: 6 × 2  
## Platform glbl\_total  
## <chr> <dbl>  
## 1 PS3 588.  
## 2 X360 550.  
## 3 PS4 314.  
## 4 3DS 258.  
## 5 Wii 223.  
## 6 XOne 159.

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.

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

## [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.

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

## # 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

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

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

## # A tibble: 6 × 2  
## Genre genre\_glbl\_total  
## <chr> <dbl>  
## 1 Action 674.  
## 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 deterine whether there are any irregularities in the Developer column.

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

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

## # 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.

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

## # 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… PS3 2013 Acti… Take-T… 7.02 9.09 0.98 3.96 21.0  
## 2 Grand T… X360 2013 Acti… Take-T… 9.66 5.14 0.06 1.41 16.3  
## 3 Call of… X360 2010 Shoo… Activi… 9.7 3.68 0.11 1.13 14.6  
## 4 Call of… PS3 2012 Shoo… Activi… 4.99 5.73 0.65 2.42 13.8  
## 5 Call of… X360 2012 Shoo… Activi… 8.25 4.24 0.07 1.12 13.7  
## 6 Call of… PS3 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.

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

## [1] 0

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

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

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

head(df) #We preview the data set.

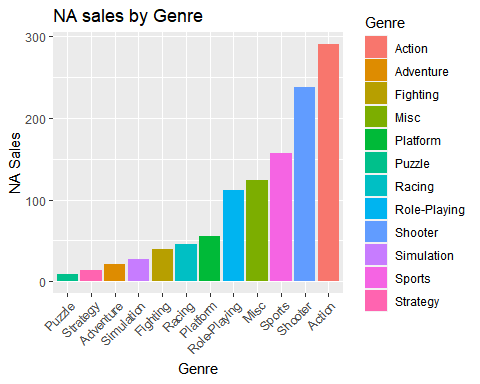
## # A tibble: 6 × 4  
## # Groups: Developer [1]  
## Developer Genre developer\_glbl\_total developer\_NA\_total  
## <chr> <chr> <dbl> <dbl>  
## 1 EA Games Action 5.3 3.55  
## 2 EA Games Adventure 0.32 0.17  
## 3 EA Games Fighting 2.61 1.24  
## 4 EA Games Platform 0.57 0.21  
## 5 EA Games Racing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

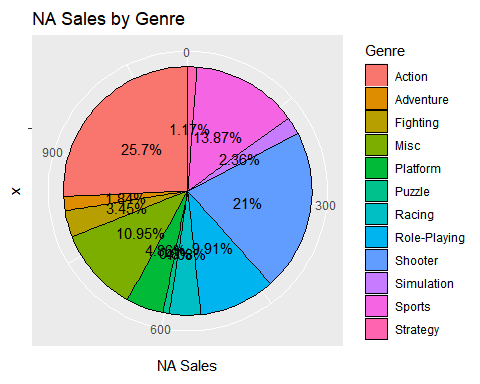
5.4 We now wan to create some visualizations for our third question to better understand the NA sales for each genre in 2010 to 2016.

#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)

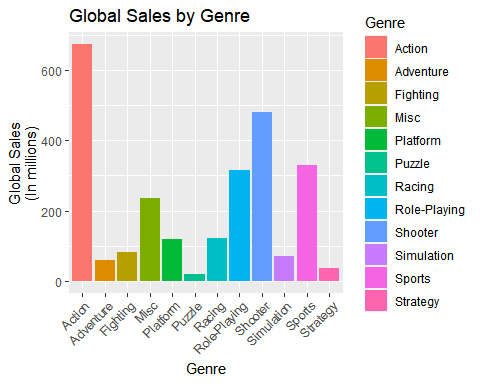
#we convert our numeric values in genre\_na\_sales into a percentage of the entire sum and store it into pie\_labels.  
pie\_labels <- paste0(round(100 \*genre\_na\_sales$genre\_na\_total/sum(genre\_na\_sales$genre\_na\_total), 2), "%")  
#we use ggplot() to create the graph, the aes() function set’s 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.

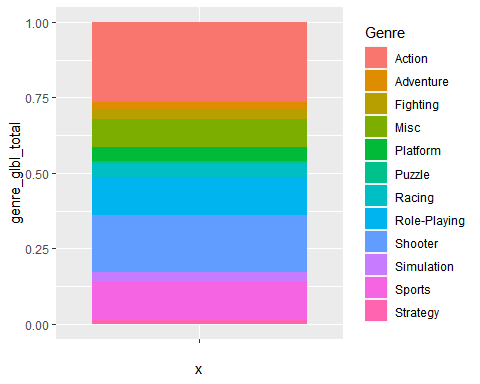
5.5 We now see the total global revenue based on Genre to get a visual for which Genre was more profitable all throughout 2010-2016.

#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))

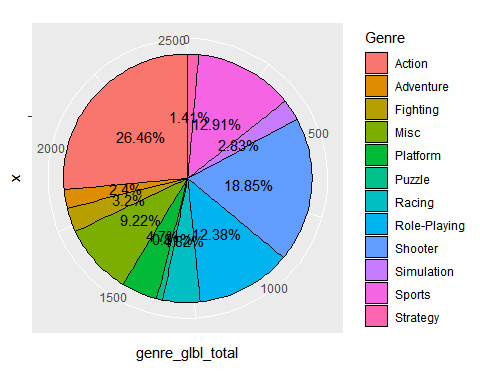


#labs() is used to changes the labels in various parts of the chart. Both the X and Y axis changing their names, as well as the title. theme() helps change non-data elements of a chart. In this case adding a change to Genre’s axis by utilizing element\_text() to change the look of texts, adding an angle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

#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))  
  
#This is simply a stacked bar chart with only a couple functions at play. 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")



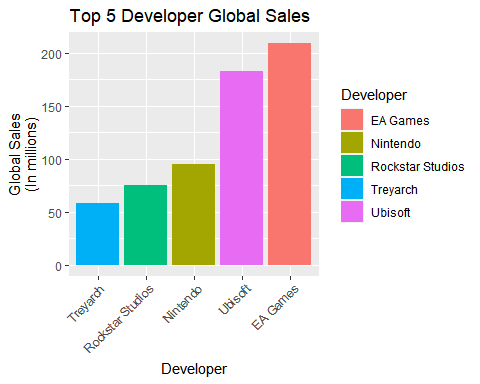
#we convert our numeric values in genre\_glbl\_sales into a percentage of the entire sum and store it into pie\_labels2.  
pie\_labels2 <- paste0(round(100\*genre\_glbl\_sales$genre\_glbl\_total/sum(genre\_glbl\_sales$genre\_glbl\_total), 2), "%")  
#we use ggplot() to create the graph, the aes() function 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.

5.6 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)")

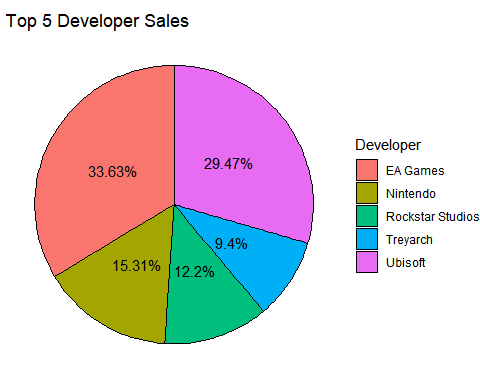


#The geom\_bar() function creates a bar plot using the "identity" statistical transformation.The theme() function is used to customize the appearance of the plot. The axis.text.x argument is set to element\_text() to change the angle of the x-axis labels to 45 degrees and set the horizontal justification 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`



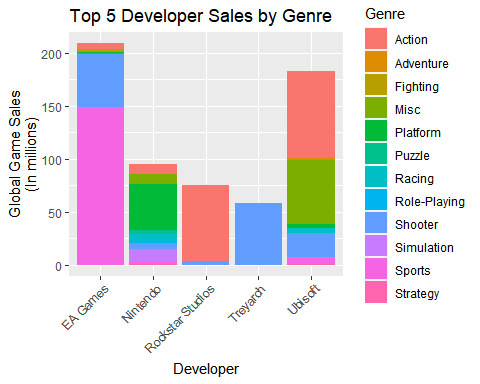
#The geom\_text() function is used to sum up the percentage each of these five Developers make globally. The theme\_void() function removes all the axis labels, ticks, and grid lines, leaving only the bars.The labs() function sets the plot title to "Top 5 Developer Sales".

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.

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

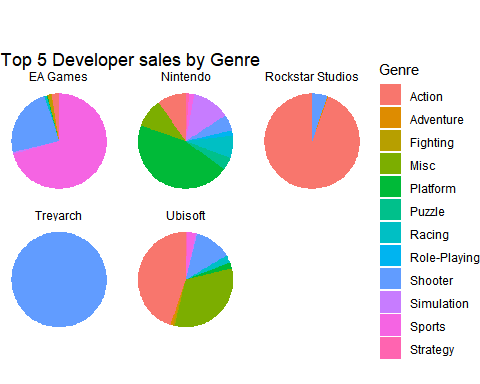
#We use the ggplot2 package in R to show the top 5 video game genres for each of the top 5 video game developers in terms of global sales. The aes() function is used to set the aesthetics of the plot. The x-axis is set to the Developer column, the y-axis is set to the Global\_Total column, and the fill 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.

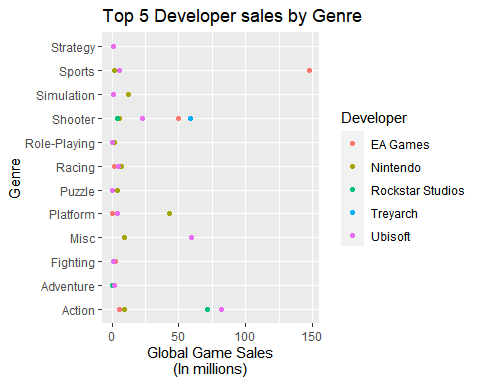
#The aes() function is used to set the aesthetics of the plot. The x-axis is set to 0, the y-axis is set to the Global\_Total column, and the fill aesthetic is set to the Genre column. The geom\_col() function creates a column plot with position set to "fill", which stacks the bars so that each bar fills 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.