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 --
## v ggplot2 3.4.0      v purrr   1.0.1
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x 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 (Fi
```

```
## 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 x 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
```

```
## [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 attr
```

```
##            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

dim(gamesales2010_2016) #We use the dim() function and see that the amount of rows in the data set has
```

```
## [1] 5550   16
```

3.2 We will now filter all rows where the Year_of_Release is unknown.************************

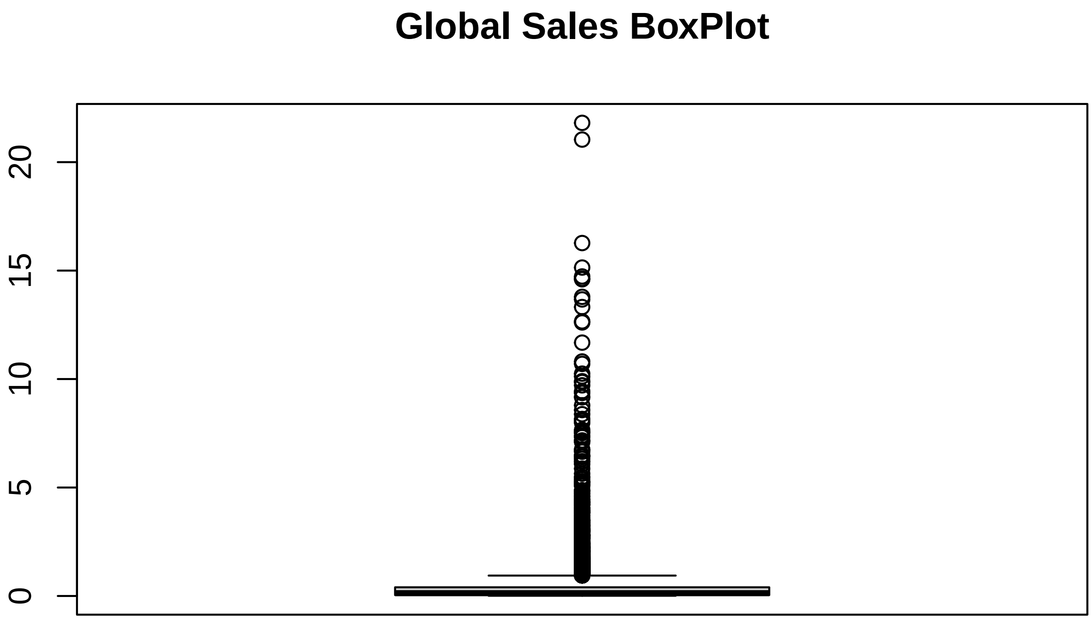
```
gamesales2010_2016 <- filter(gamesales2010_2016, Year_of_Release != "N/A") #The filter() function remov
nrow(gamesales2010_2016) #nrow() prints out the number of rows in gamesales2010_2016.
```

```
## [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 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.

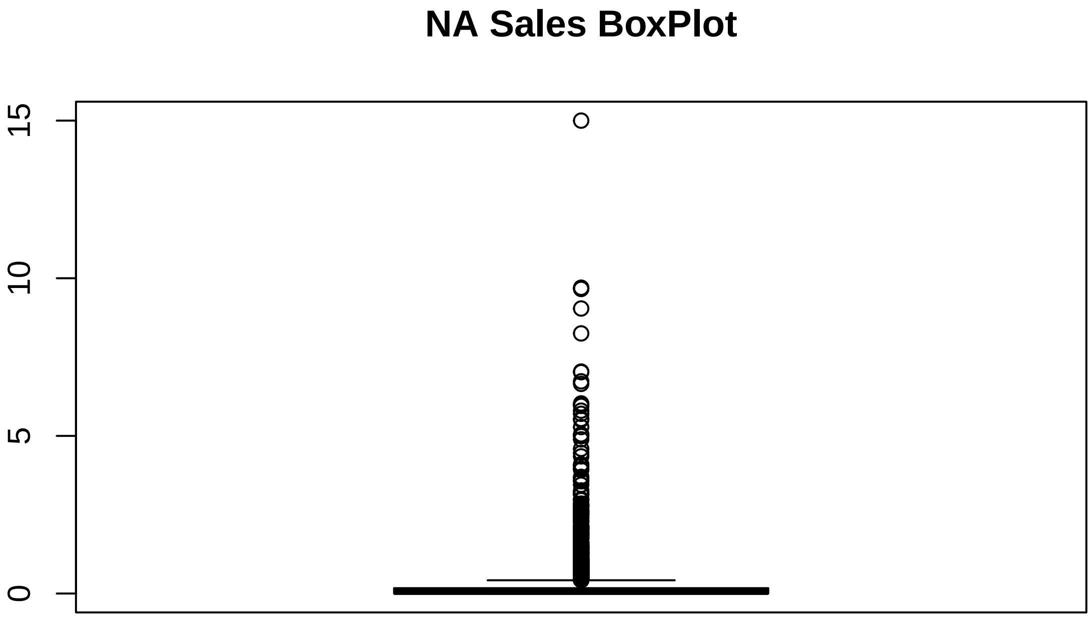
```
boxplot(gamesales2010_2016$Global_Sales, main = "Global Sales BoxPlot")
```



```
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. * Which video game platform made the most NA Sales from 2010 to 2016? * Which video game platform made the most Global Sales from 2010 to 2016? * Which video game category had the highest NA sales? * Which video game category made the highest Global sales? * Which top five video game developers have the highest Global video game sales? * 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 th
  group_by(Platform) %>% # we use the group_by() function to group all the platforms in the platform_NA_
  summarize(NA_total = sum(NA_Sales)) %>% #we use the summarize function to summarize the sum of NA_sal
  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 x 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
  group_by(Platform) %>% # we use the group_by() function to group all the platforms in the platform_gl
  summarize(glbl_total = sum(Global_Sales)) %>% #we use the summarize function to summarize the sum of 
  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 x 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
```

```
##  [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
  group_by(Genre) %>% # we use the group_by() function to group all the Genres in the genre_na_sales to
  summarise(genre_na_total = sum(NA_Sales)) %>% #we use the summarize function to summarize the sum of
  arrange(desc(genre_na_total))#We arrange the data set by decreasing order according to the genre_na_t
head(genre_na_sales) #We use the head() function to preview the table.
```

```
## # A tibble: 6 x 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
  group_by(Genre) %>%  # we use the group_by() function to group all the Genres in the genre_glbl_sales
  summarise(genre_glbl_total = sum(Global_Sales)) %>% #we use the summarize function to summarize the s
  arrange(desc(genre_glbl_total))#We arrange the data set by decreasing order according to the genre_na
head(genre_glbl_sales)#We use the head() function to preview the table.
```

```
## # A tibble: 6 x 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 co
```

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 var

gamesales2010_2016$Developer <- recode(gamesales2010_2016$Developer,"1C: Maddox Games" ="1C Company", "

gamesales2010_2016$Developer <- recode(gamesales2010_2016$Developer, "Capcom, QLOC" = "Capcom", "Climax
#"EA Sports" = "EA Games"
```

```
gamesales2010_2016$Developer <- recode(gamesales2010_2016$Developer, "EA Sports" = "EA Games", "EA Mont


gamesales2010_2016$Developer <- recode(gamesales2010_2016$Developer,"Harmonix Music Systems, Demiurge" 

gamesales2010_2016$Developer <- recode(gamesales2010_2016$Developer,"Namco Bandai Games America, Namco 


gamesales2010_2016$Developer <- recode(gamesales2010_2016$Developer,"PLAYGROUND, Playground Games" = "P

gamesales2010_2016$Developer <- recode(gamesales2010_2016$Developer, "Sega Studios San Francisco" = "Seg
```

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 d
  group_by(Developer) %>% #We will then group the columns according to the value in the developer colum
  summarise(global_total = sum(Global_Sales)) %>% #We sum up the total global sales and NA sales of eac
  arrange(desc(global_total)) %>% #We then arrange each row in descending order based on the global_tot
  slice_head(n = 5) #We use the slice_head() function to take ONLY the top five most profitable develop
head(dev_global) #We preview the data set.
```

```
## # A tibble: 5 x 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 creat
  head(developer_genre_sale) # We use this to preview the data set.
```

```
## # A tibble: 6 x 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
```

```
## [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 summa
  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.
```

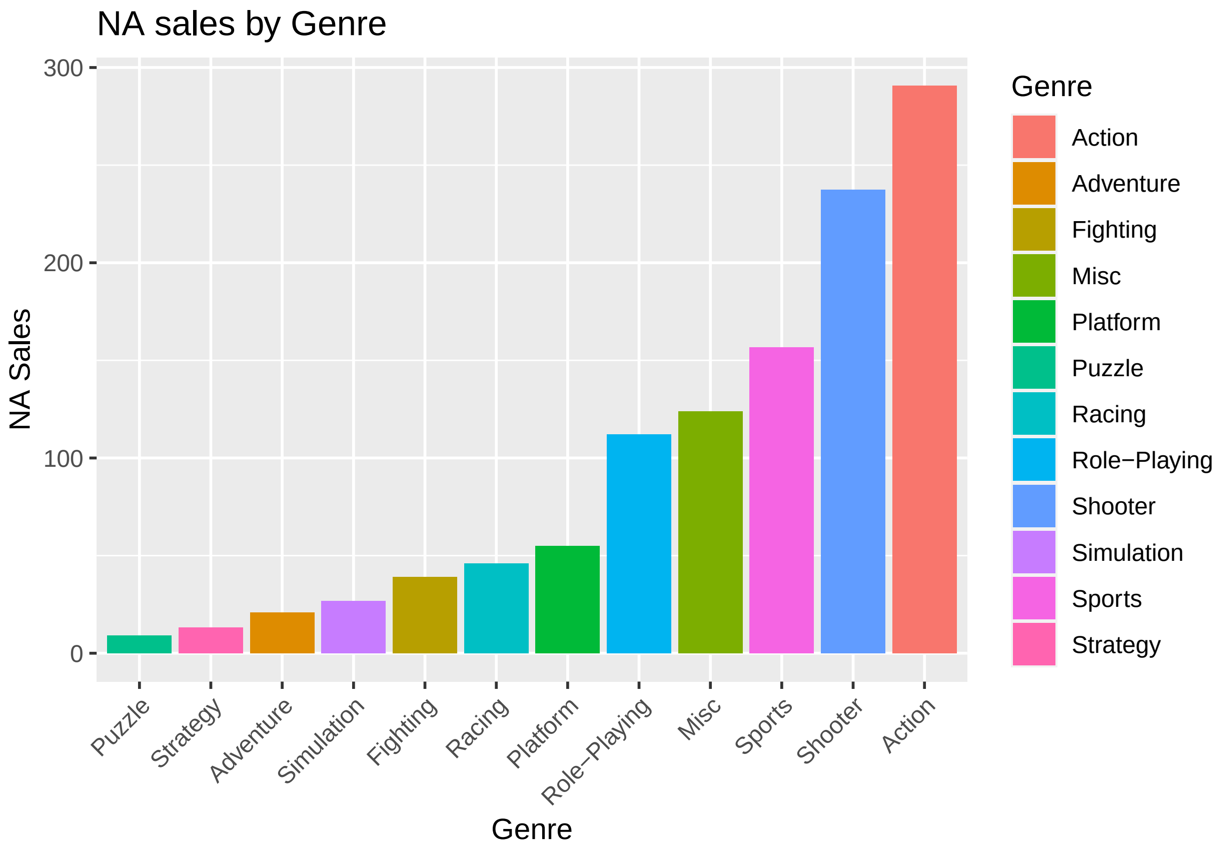
```
## # A tibble: 6 x 4
## # Groups:   Developer [1]
##   Developer Genre     developer_glbl_total developer_NA_total
##   <chr>     <chr>                    <dbl>              <dbl>
## 1 EA Games  Action                    5.3                3.55
## 2 EA Games  Adventure                 0.32               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.

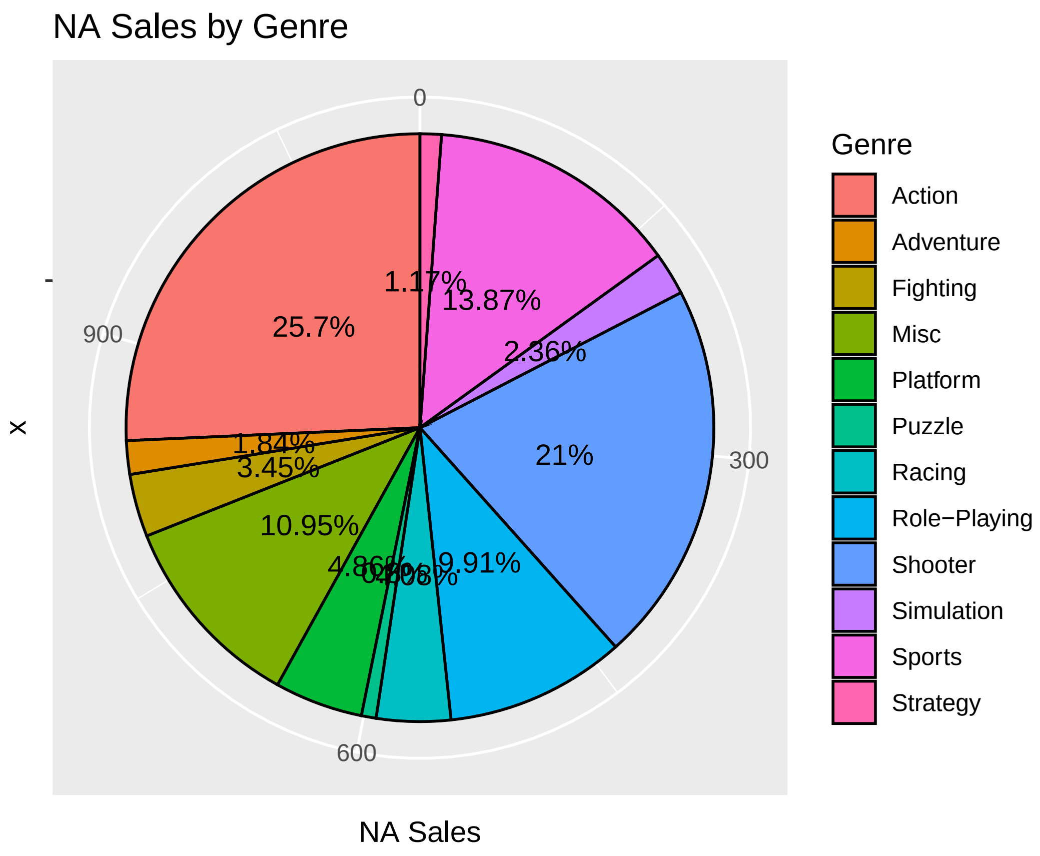
5.4 We now wan to create some visualizations for our third question to better understand the NA sales for each genre in 2010 to 2016.

```
#We use the ggplot2 package to create a Bar graph that easily represents the highest selling video game
ggplot(data = genre_na_sales, aes(x = reorder(Genre,genre_na_total), y = genre_na_total, fill = Genre))
```



```
#labs() is used to changes the labels in various parts of the chart. Both the X and Y axis changing the
```

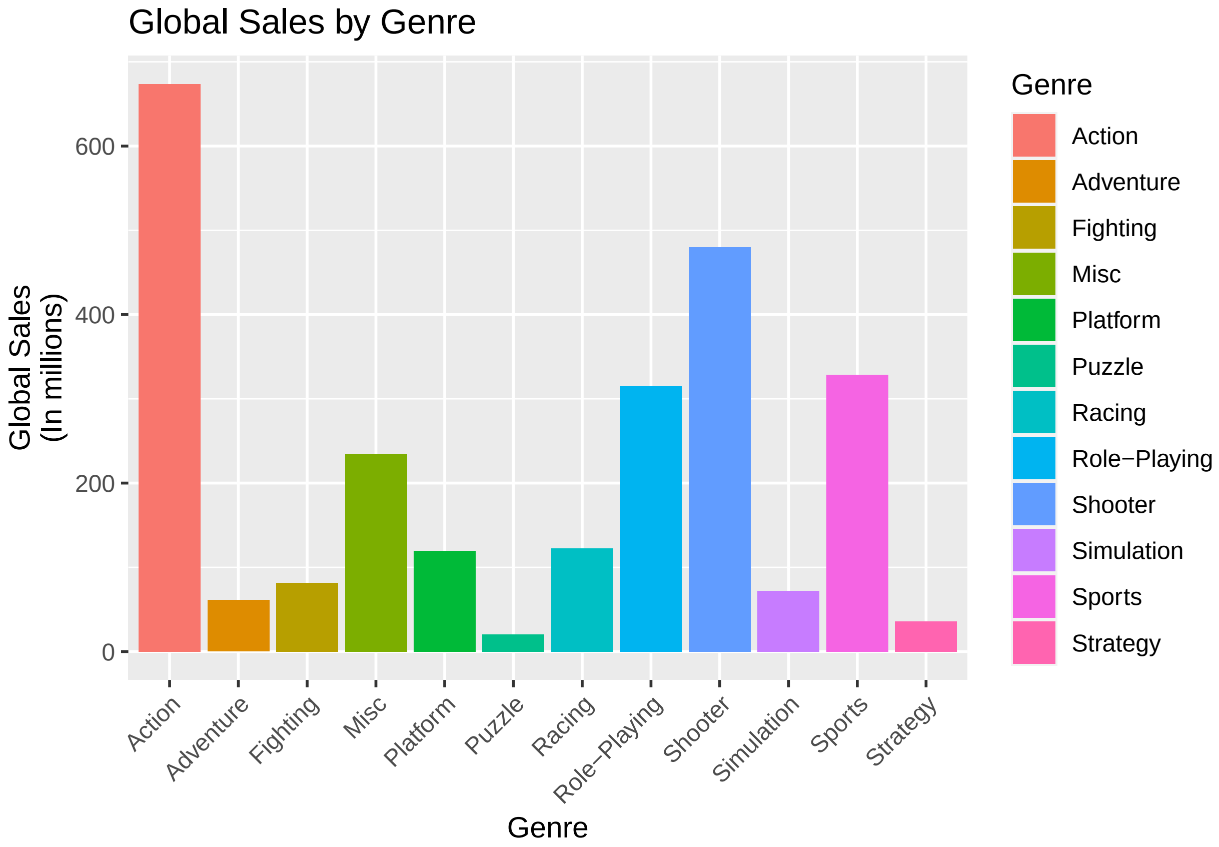
```
#we convert our numeric values in genre_na_sales into a percentage of the entire sum and store it into p
pie_labels <- paste0(round(100 *genre_na_sales$genre_na_total/sum(genre_na_sales$genre_na_total), 2), "
#we use ggplot() to create the graph, the aes() function set's the aesthetic. Used nothing for X, but f
ggplot(genre_na_sales, aes(x = "", y = genre_na_total, fill = Genre)) + geom_col(color = "black") + geom
```



```
#geom_text is used along with aes() embedded to set labels to pie labels and position them accordingly.
```

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.

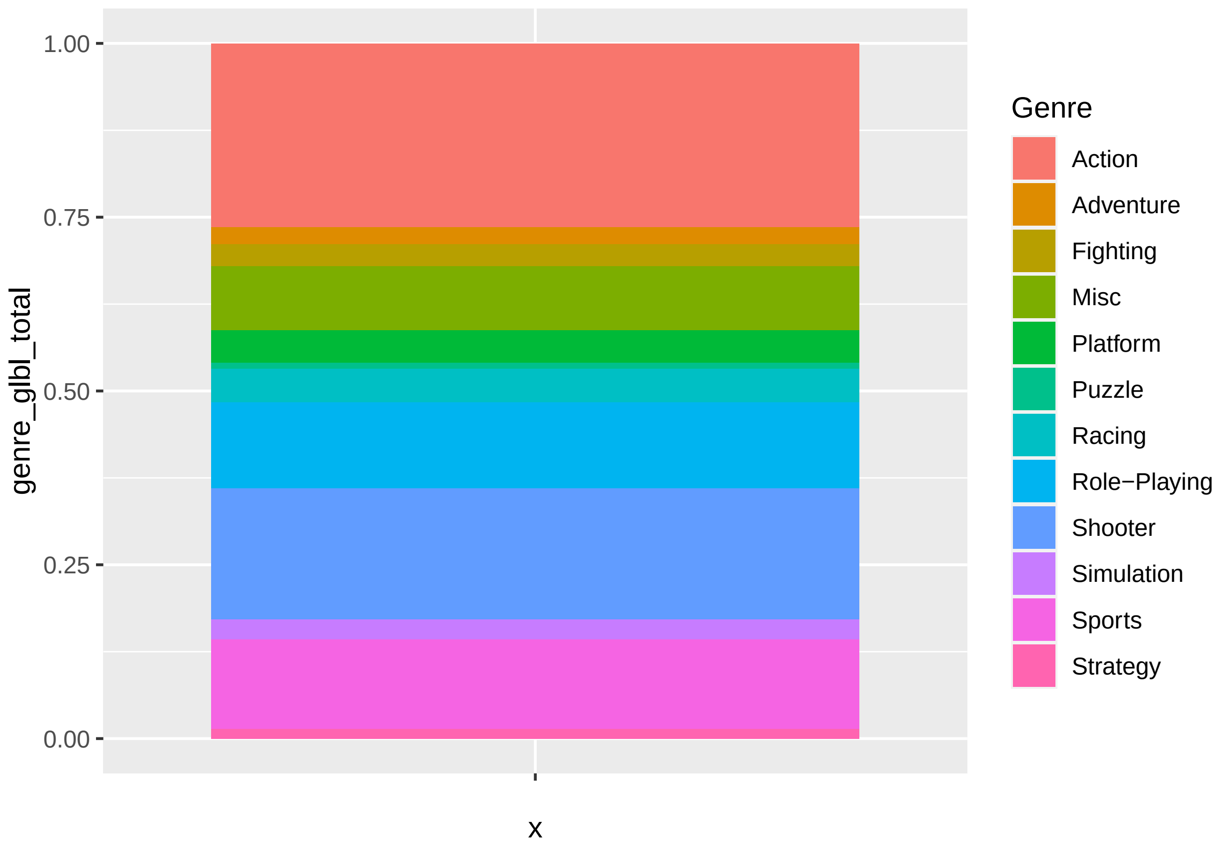
```
#We use the ggplot2 package to create a Bar graph that easily represents the highest globally selling v
ggplot(data = genre_glbl_sales, aes(x = Genre, y = genre_glbl_total, fill = Genre)) + geom_bar(stat = "
```



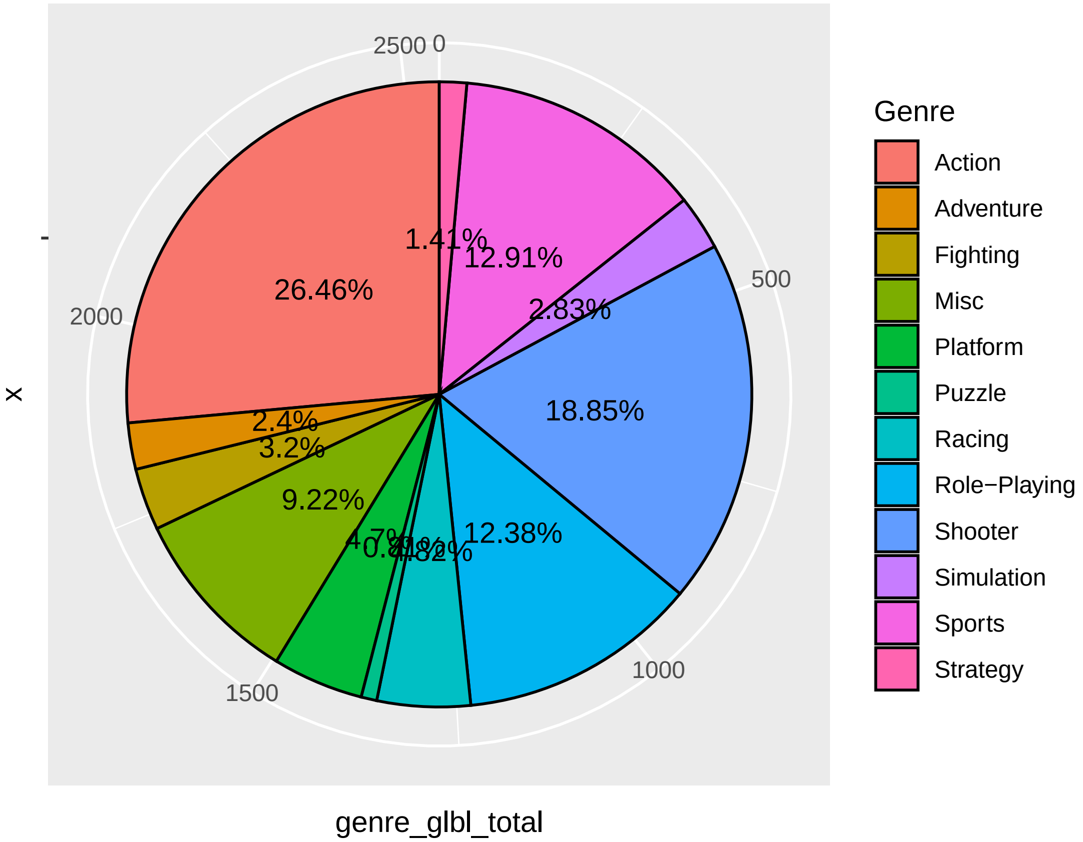
```
#labs() is used to changes the labels in various parts of the chart. Both the X and Y axis changing the
```

```
#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
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 b
ggplot(PVGSG, aes(fill = Genre, y = genre_glbl_total, x = "")) + geom_bar(position = "fill", stat = "id
```



```
#we convert our numeric values in genre_glbl_sales into a percentage of the entire sum and store it int
pie_labels2 <- paste0(round(100*genre_glbl_sales$genre_glbl_total/sum(genre_glbl_sales$genre_glbl_total
#we use ggplot() to create the graph, the aes() function set's the aesthetic. Used nothing for X, but f
ggplot(genre_glbl_sales, aes(x = "", y = genre_glbl_total, fill = Genre)) + geom_col(color = "black") +
```

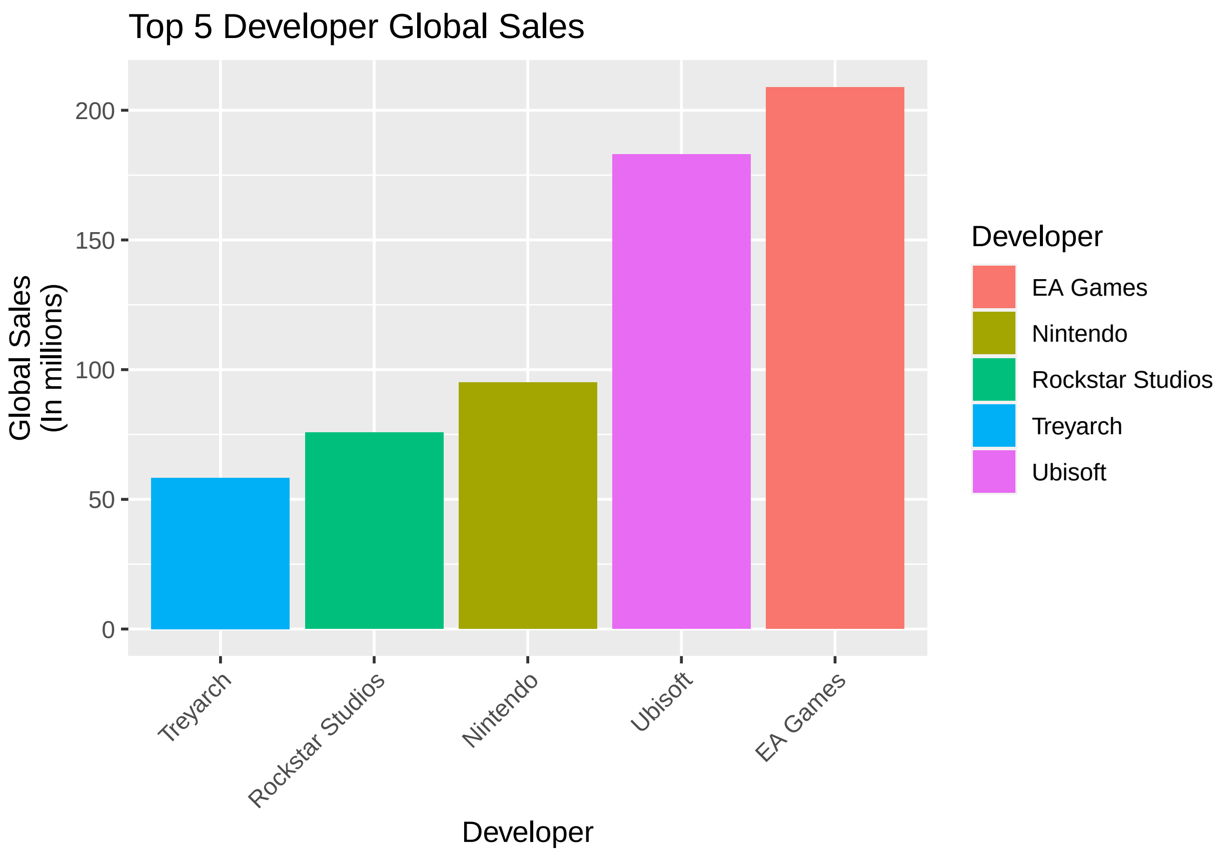


```
#geom_text is used along with aes() embedded to set labels to pie labels and position them accordingly.
```

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 th

ggplot(dev_global, aes(x = reorder(Developer, global_total), y = global_total, fill = Developer)) + geon
```

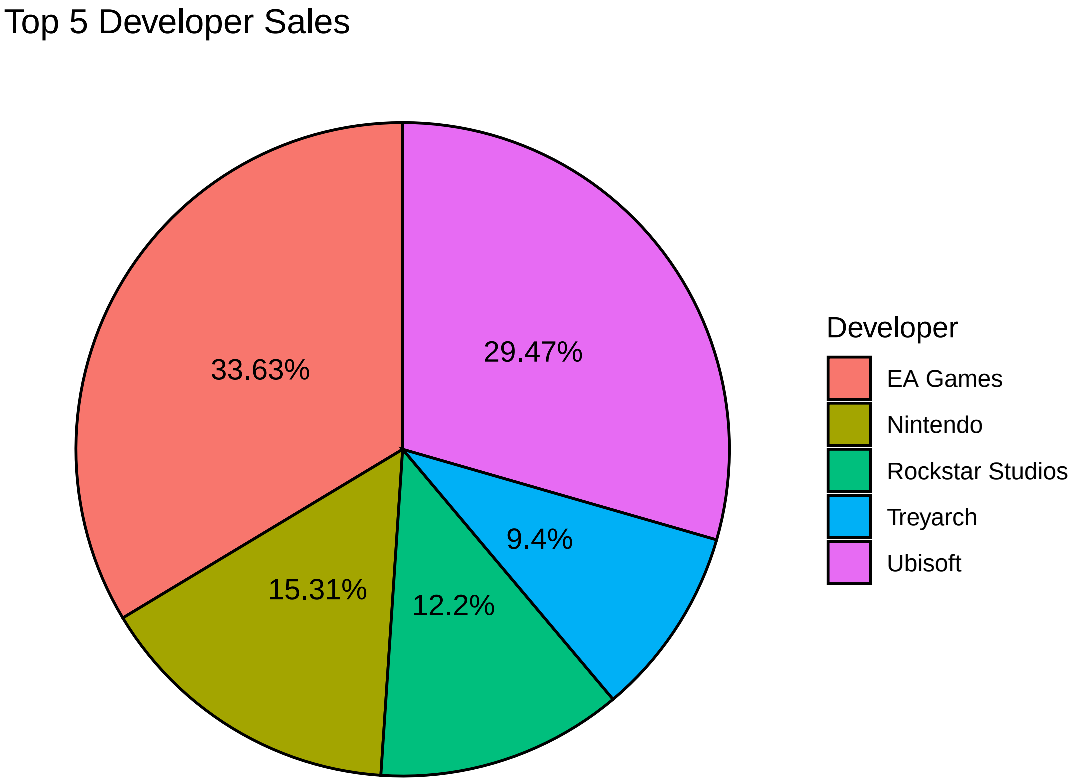


```
#The geom_bar() function creates a bar plot using the "identity" statistical transformation.The theme()
```

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 int
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 
ggplot(dev_global, aes(x = "", y = global_total, fill = Developer)) + geom_col(stat = "identity", width
```

```
## Warning in geom_col(stat = "identity", width = 1, color = "black"): Ignoring
## unknown parameters: ‘stat‘
```



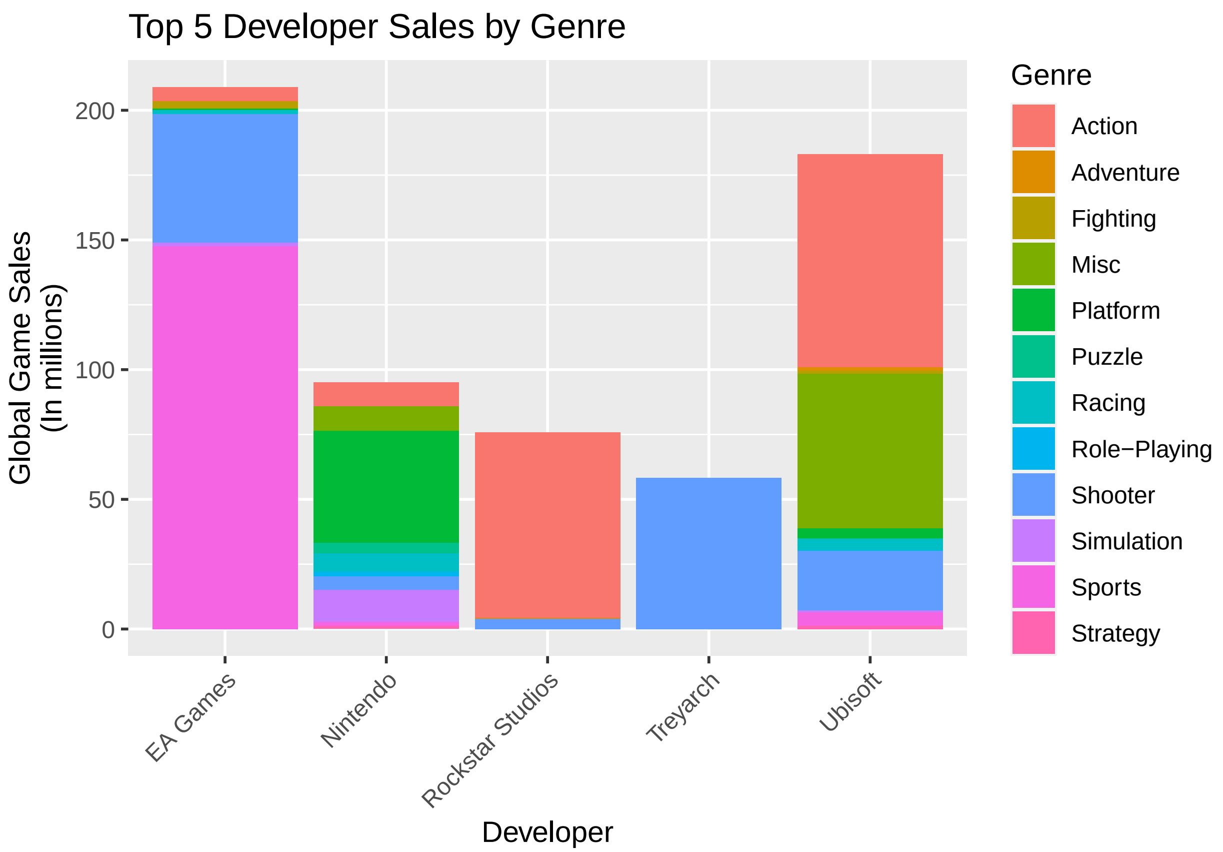
```
#The geom_text() function is used to sum up the percentage each of these five Developers make globally.
```

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

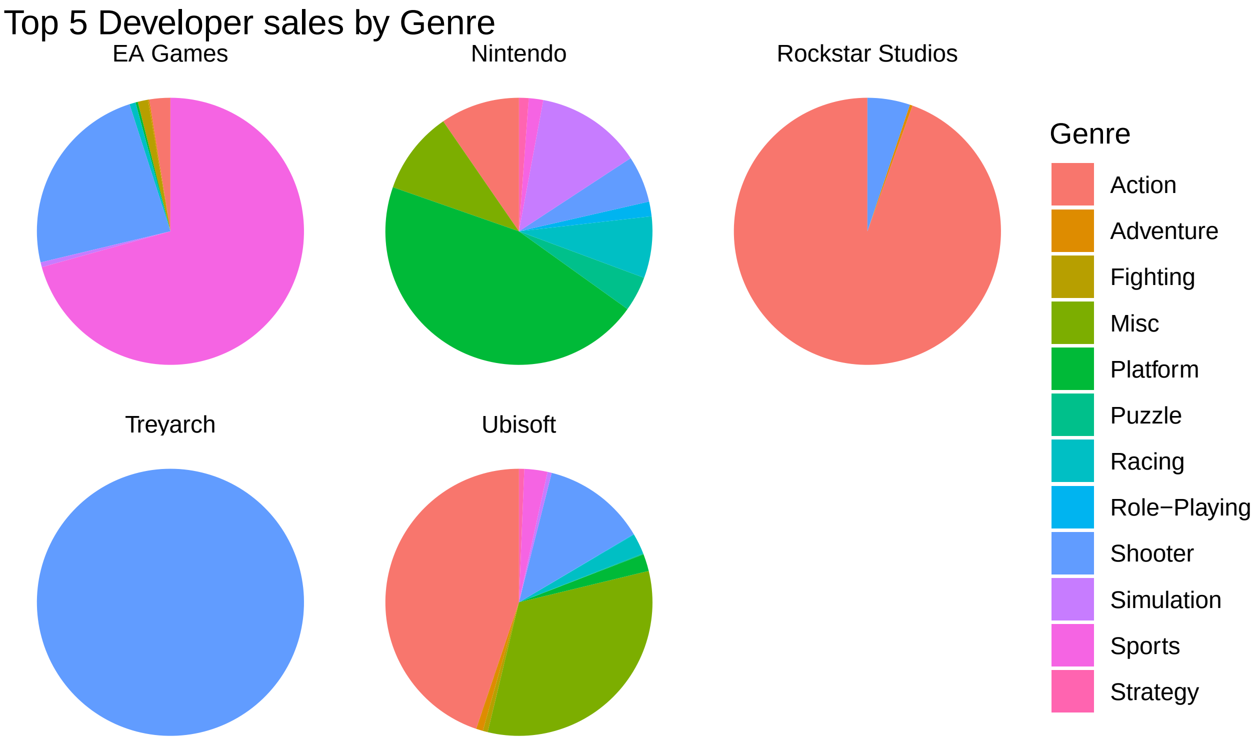
```
#We use the ggplot2 package in R to show the top 5 video game genres for each of the top 5 video game d
ggplot(df, aes( x = Developer, y = developer_glbl_total, fill = Genre)) + geom_bar(stat = "identity") +
```



```
#The theme() function sets the x-axis text to a 45-degree angle for better readability, and the labs()
```

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.

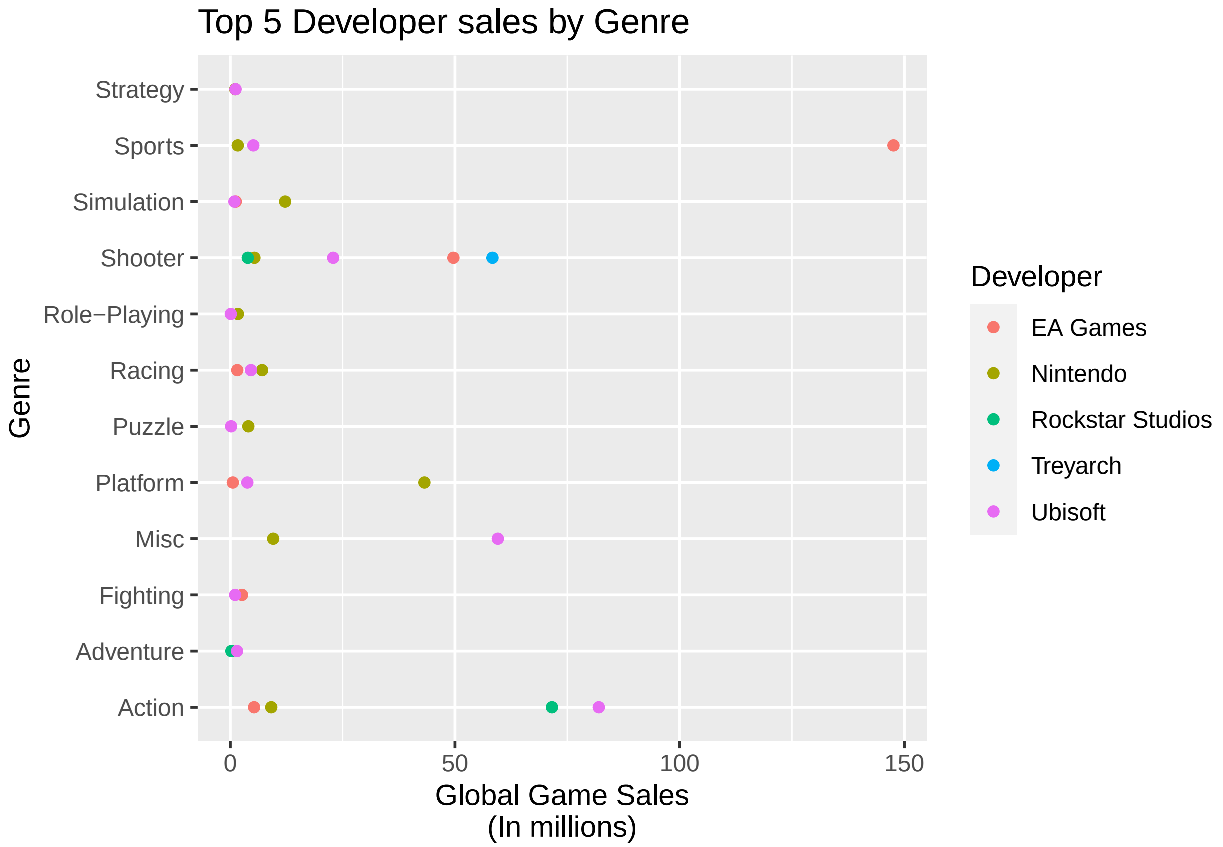
```
#The aes() function is used to set the aesthetics of the plot. The x-axis is set to 0, the y-axis is se
ggplot(data = df, aes(x = 0, y = developer_glbl_total, fill = Genre)) + geom_col(position = "fill") + fa
```



```
#The facet_wrap() function is used to split the plot into multiple panels, one for each developer. The
```

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 i
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)") +
```



```
#We use the labs() function to label our title, x-axis, and y-axis. The coordflip() flips the x and y a
```

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.