CYO Vgdata

Juan Pablo Mendoza

9 de junio de 2019

I would like to thank the course's staff and stack overflow, whom I just met and where I found solace.

1. Executive Summary.

Greetings dear reviewer.

This report documents the approach taken to fulfil the "Chose Your Own Project Submission" for the HarvardX Data Science Capstone Course. The dataset used in this report can be found in the following link link

I decided to use this videogame dataset because it was something I was able to understand. While I was looking for it I understood that it would be impossible for me to take any value out of it without understanding what I was handling. My knowledge regarding biology are not great, so that was a big filter as well considering a lot of the datasets there are.

Unfortunetly this dataset doesn't include the N64 games, and it's only up to 2017. I don't know the reasons for that. But we can still gain some really useful insights from it. Everything shown in the following report is up to debate. It's perfectly natural to find disagreements with it, and I do not intend to establish an unarguable result. I mean, I don't even know the things I still need to learn! But oh well, let's do our best. As you may have seen from my user name, English is not my first language so I hope you may excuse me for any language mistakes along the way.

This report consists of four different courses of action: +The first section of the following code consists of Data cleaning and Shaping looking to find the top 10 games of our dataset. +The second section we will analyze performance of the Publishers against each other in sales. +The third section, we will see the console that perform the best from our dataset. +Lastly, in the fourth section will see how good can we predict the Users acceptance of a game by checking the Critics Scores.

2. Methodology.

2.1 Setting up the dataset.

We will require the following packages in order to run our analysis.

The report follow the next steps: +First, we will use the "tidyverse" package to filter unuseful information in each section and create a new column as a "True Score" resulting from the Scores of both Users and Critics under which to evaluate each game. +Then, we will then order the values under certain parameters in order to find the "best games" of our dataset. +Third, We will make use of the "ggplot2" package to plot the results of our filterings and evaluate our results for each section. +After that We will judge the performance of different agents and establish a list of the best in each section. +For our last step we will implement Machine Learning algorithms to analyse the performance of the Critics Score to predict how the Users will judge each game.

3. Results.

3.1 Data cleaning and Shaping: Looking for the top 10.

Let's create an object called "vgdata" to contain the dataset.

```
url <- "https://raw.githubusercontent.com/juanpmendoza/Test/master/Video_Games_Sales_as_at_22_Dec_2016.c
vgdata <- read.csv(url)
```

Now let's check how many different inputs we have for the following categories:

```
n_vgdata <- vgdata %>% summarize(n_Games = n_distinct(Name),
                                 n_Platforms = n_distinct(Platform),
                                 n_Genres = n_distinct(Genre),
                                 n_Publishers = n_distinct(Publisher))
n_vgdata
```

```
##   n_Games n_Platforms n_Genres n_Publishers
## 1   11563          31       13          582
```

```
options(max.print = 320)

head(vgdata)
```

```
##                        Name Platform Year_of_Release        Genre Publisher
## 1                Wii Sports      Wii            2006       Sports  Nintendo
## 2         Super Mario Bros.      NES            1985     Platform  Nintendo
## 3            Mario Kart Wii      Wii            2008       Racing  Nintendo
## 4         Wii Sports Resort      Wii            2009       Sports  Nintendo
## 5 Pokemon Red/Pokemon Blue       GB            1996 Role-Playing  Nintendo
## 6                    Tetris       GB            1989       Puzzle  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score
## 1    41.36    28.96     3.77        8.45        82.53           76
## 2    29.08     3.58     6.81        0.77        40.24           NA
## 3    15.68    12.76     3.79        3.29        35.52           82
## 4    15.61    10.93     3.28        2.95        32.77           80
## 5    11.27     8.89    10.22        1.00        31.37           NA
## 6    23.20     2.26     4.22        0.58        30.26           NA
##   Critic_Count User_Score User_Count Developer Rating
## 1           51          8        322  Nintendo      E
## 2           NA                    NA
## 3           73        8.3        709  Nintendo      E
## 4           73          8        192  Nintendo      E
## 5           NA                    NA
## 6           NA                    NA
```

We can get an idea of how the data is structured and the columns it contains. For the purposes of our analysis it will be most convenient for us to commence shaping the data by Publishers.

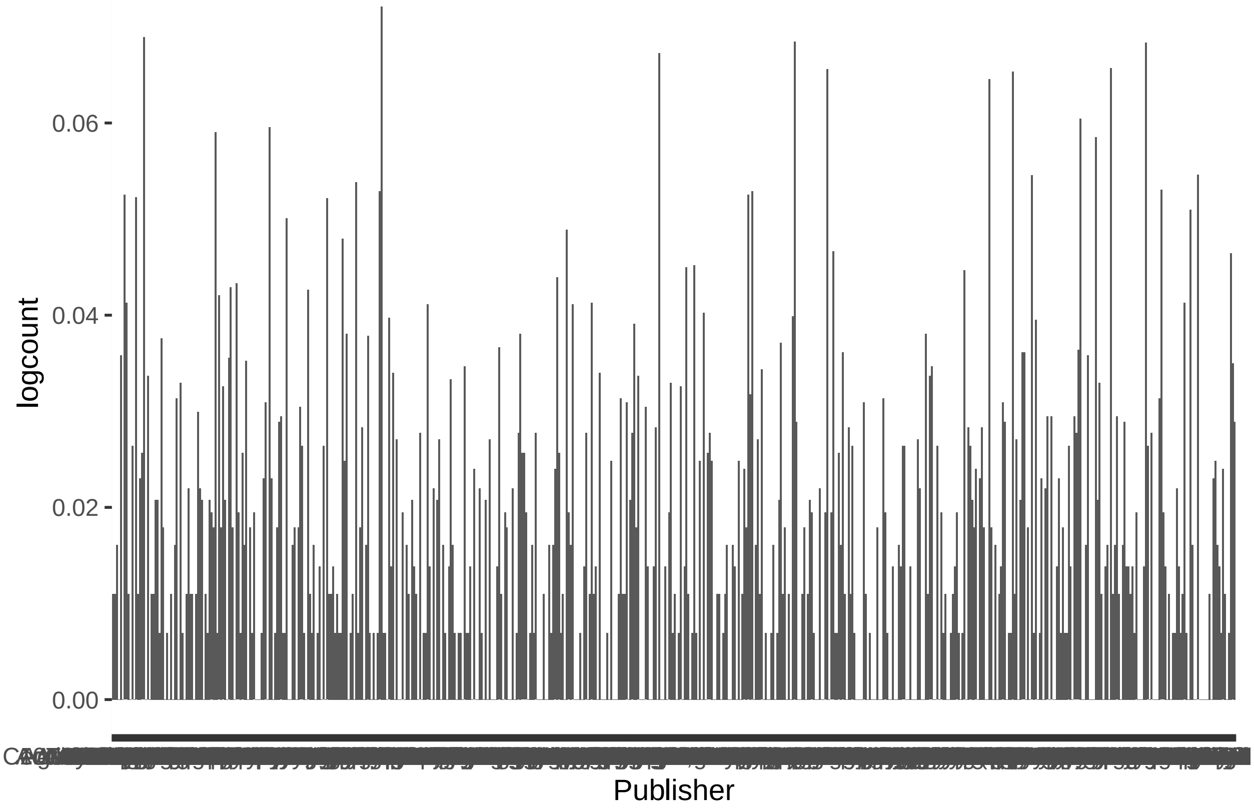
We can see that there are a lot of Publishers:

```
n_vgdata$n_Publishers
```

```
## [1] 582
```

So trying to plot them all would be quite unproductive, not only because of the amount of them, but also the huge disparity between as we can see in the following plot, where even if we take a log scale of 100, the distance between publishers is gigantic!

```
vg_by_publisher <- vgdata %>% group_by(Publisher) %>% summarize(count = n()) %>%
  mutate(logcount = log(count, exp(100)))
vg_by_publisher %>% ggplot(aes(Publisher, logcount)) + geom_bar(stat = "identity")
```



So we are going to have to take just a part of them into consideration, specially since there are some indie ones that are not really much of a match in terms of quality when compared to others (no offense). When checking the top 20 Publishers, we find the following:

```
vgdata %>% group_by(Publisher) %>% summarize(count = n()) %>%  arrange(desc(count)) %>% print(n = 20)
```

```
## # A tibble: 582 x 2
##    Publisher                              count
##    <fct>                                  <int>
##  1 Electronic Arts                         1356
##  2 Activision                               985
##  3 Namco Bandai Games                       939
##  4 Ubisoft                                 933
##  5 Konami Digital Entertainment            834
##  6 THQ                                     715
##  7 Nintendo                                706
##  8 Sony Computer Entertainment             687
##  9 Sega                                    638
## 10 Take-Two Interactive                    422
## 11 Capcom                                  386
## 12 Atari                                   367
## 13 Tecmo Koei                              348
## 14 Warner Bros. Interactive Entertainment   235
## 15 Square Enix                             234
## 16 Disney Interactive Studios              218
## 17 Unknown                                 201
## 18 Eidos Interactive                       198
## 19 Midway Games                            198
## 20 505 Games                               191
## # ... with 562 more rows
```

We can see that the #17 publisher in the list is “Unknown”, we must then delete this for the sake of the analysis. We can do so with the following code:

```
vgdata <- vgdata %>% filter(!Publisher%in%c("Unknown"))
```

We can then check and see that the games with Unknown Publishers have indeed been deleted.

```
vgdata %>% group_by(Publisher) %>% summarize(count = n()) %>%  arrange(desc(count)) %>% print(n = 20)
```

```
## # A tibble: 581 x 2
##    Publisher                              count
##    <fct>                                  <int>
##  1 Electronic Arts                         1356
##  2 Activision                               985
##  3 Namco Bandai Games                       939
##  4 Ubisoft                                  933
##  5 Konami Digital Entertainment             834
##  6 THQ                                      715
##  7 Nintendo                                 706
##  8 Sony Computer Entertainment              687
##  9 Sega                                     638
## 10 Take-Two Interactive                     422
## 11 Capcom                                   386
## 12 Atari                                    367
## 13 Tecmo Koei                               348
## 14 Warner Bros. Interactive Entertainment   235
## 15 Square Enix                              234
## 16 Disney Interactive Studios               218
## 17 Eidos Interactive                        198
## 18 Midway Games                             198
## 19 505 Games                                191
## 20 Microsoft Game Studios                   191
## # ... with 561 more rows
```

Now there's another issue to adress beforehand, and it's about the NAs present in both the Critic and User scores, since they will be important for our analysis. Let's check for NAs:

```r
is.na(vgdata) %>% print()
```

```
##           Name Platform Year_of_Release Genre Publisher NA_Sales EU_Sales
##     [1,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [2,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [3,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [4,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [5,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [6,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [7,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [8,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##     [9,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [10,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [11,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [12,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [13,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [14,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [15,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [16,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [17,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [18,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [19,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##    [20,] FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##          JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
##     [1,]    FALSE       FALSE        FALSE        FALSE        FALSE
##     [2,]    FALSE       FALSE        FALSE         TRUE         TRUE
##     [3,]    FALSE       FALSE        FALSE        FALSE        FALSE
##     [4,]    FALSE       FALSE        FALSE        FALSE        FALSE
##     [5,]    FALSE       FALSE        FALSE         TRUE         TRUE
##     [6,]    FALSE       FALSE        FALSE         TRUE         TRUE
##     [7,]    FALSE       FALSE        FALSE        FALSE        FALSE
##     [8,]    FALSE       FALSE        FALSE        FALSE        FALSE
##     [9,]    FALSE       FALSE        FALSE        FALSE        FALSE
##    [10,]    FALSE       FALSE        FALSE         TRUE         TRUE
##    [11,]    FALSE       FALSE        FALSE         TRUE         TRUE
##    [12,]    FALSE       FALSE        FALSE        FALSE        FALSE
##    [13,]    FALSE       FALSE        FALSE         TRUE         TRUE
##    [14,]    FALSE       FALSE        FALSE        FALSE        FALSE
##    [15,]    FALSE       FALSE        FALSE        FALSE        FALSE
##    [16,]    FALSE       FALSE        FALSE        FALSE        FALSE
##    [17,]    FALSE       FALSE        FALSE        FALSE        FALSE
##    [18,]    FALSE       FALSE        FALSE        FALSE        FALSE
##    [19,]    FALSE       FALSE        FALSE         TRUE         TRUE
##    [20,]    FALSE       FALSE        FALSE        FALSE        FALSE
##          User_Score User_Count Developer Rating
##     [1,]     FALSE      FALSE     FALSE  FALSE
##     [2,]     FALSE       TRUE     FALSE  FALSE
##     [3,]     FALSE      FALSE     FALSE  FALSE
##     [4,]     FALSE      FALSE     FALSE  FALSE
##     [5,]     FALSE       TRUE     FALSE  FALSE
##     [6,]     FALSE       TRUE     FALSE  FALSE
##      [7,]       FALSE      FALSE     FALSE  FALSE
##      [8,]       FALSE      FALSE     FALSE  FALSE
##      [9,]       FALSE      FALSE     FALSE  FALSE
##     [10,]       FALSE       TRUE     FALSE  FALSE
##     [11,]       FALSE       TRUE     FALSE  FALSE
##     [12,]       FALSE      FALSE     FALSE  FALSE
##     [13,]       FALSE       TRUE     FALSE  FALSE
##     [14,]       FALSE      FALSE     FALSE  FALSE
##     [15,]       FALSE      FALSE     FALSE  FALSE
##     [16,]       FALSE      FALSE     FALSE  FALSE
##     [17,]       FALSE      FALSE     FALSE  FALSE
##     [18,]       FALSE      FALSE     FALSE  FALSE
##     [19,]       FALSE       TRUE     FALSE  FALSE
##     [20,]       FALSE      FALSE     FALSE  FALSE
##  [ reached getOption("max.print") -- omitted 16498 rows ]
```

We can see that there are indeed NAs present in the scores, we can delete them and then check again with the following code:

```
vgdata<- na.omit(vgdata)
is.na(vgdata) %>% print()
```

```
##        Name Platform Year_of_Release Genre Publisher NA_Sales EU_Sales
## 1     FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 3     FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 4     FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 7     FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 8     FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 9     FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 12    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 14    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 15    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 16    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 17    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 18    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 20    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 24    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 25    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 27    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 29    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 30    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 33    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
## 35    FALSE    FALSE           FALSE FALSE     FALSE    FALSE    FALSE
##       JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
## 1        FALSE       FALSE        FALSE        FALSE        FALSE
## 3        FALSE       FALSE        FALSE        FALSE        FALSE
## 4        FALSE       FALSE        FALSE        FALSE        FALSE
## 7        FALSE       FALSE        FALSE        FALSE        FALSE
## 8        FALSE       FALSE        FALSE        FALSE        FALSE
## 9        FALSE       FALSE        FALSE        FALSE        FALSE
## 12       FALSE       FALSE        FALSE        FALSE        FALSE
## 14       FALSE       FALSE        FALSE        FALSE        FALSE
## 15       FALSE       FALSE        FALSE        FALSE        FALSE
## 16       FALSE       FALSE        FALSE        FALSE        FALSE
## 17       FALSE       FALSE        FALSE        FALSE        FALSE
## 18       FALSE       FALSE        FALSE        FALSE        FALSE
## 20       FALSE       FALSE        FALSE        FALSE        FALSE
## 24       FALSE       FALSE        FALSE        FALSE        FALSE
## 25       FALSE       FALSE        FALSE        FALSE        FALSE
## 27       FALSE       FALSE        FALSE        FALSE        FALSE
## 29       FALSE       FALSE        FALSE        FALSE        FALSE
## 30       FALSE       FALSE        FALSE        FALSE        FALSE
## 33       FALSE       FALSE        FALSE        FALSE        FALSE
## 35       FALSE       FALSE        FALSE        FALSE        FALSE
##       User_Score User_Count Developer Rating
## 1          FALSE      FALSE     FALSE  FALSE
## 3          FALSE      FALSE     FALSE  FALSE
## 4          FALSE      FALSE     FALSE  FALSE
## 7          FALSE      FALSE     FALSE  FALSE
## 8          FALSE      FALSE     FALSE  FALSE
## 9          FALSE      FALSE     FALSE  FALSE
## 12         FALSE      FALSE     FALSE  FALSE
## 14         FALSE      FALSE     FALSE  FALSE
## 15         FALSE      FALSE     FALSE  FALSE
## 16         FALSE      FALSE     FALSE  FALSE
## 17         FALSE      FALSE     FALSE  FALSE
## 18         FALSE      FALSE     FALSE  FALSE
## 20         FALSE      FALSE     FALSE  FALSE
## 24         FALSE      FALSE     FALSE  FALSE
## 25         FALSE      FALSE     FALSE  FALSE
## 27         FALSE      FALSE     FALSE  FALSE
## 29         FALSE      FALSE     FALSE  FALSE
## 30         FALSE      FALSE     FALSE  FALSE
## 33         FALSE      FALSE     FALSE  FALSE
## 35         FALSE      FALSE     FALSE  FALSE
##  [ reached getOption("max.print") -- omitted 6928 rows ]
```

Let's create a column with the average of the proportion of Score over Count, so we can use it as an estimate for the following step.

```r
#Now we're going to divide the Critic_Count column by 10 for it to be on a 10 scale, like the
#User_Count and create an AvgP column with the result.
vgdata <- vgdata %>% mutate(AvgP=(Critic_Count/10)/User_Count)
#Now let's create a column with the average of all the proportions, an "Average TOtal" (AvgT)
vgdata <- vgdata %>% mutate(AvgT = sum(AvgP)/(length(AvgP)))
```

Now the reason we created the AvgT column is because we need it to balance the weight of the Critics' rate, since the values are too volatile.

One would think that the critics are specialized people in the segment in which they operate, however, when it comes down to video games there usually are great discrepancies between them and the users, a particular example would be the following:

```r
vgdata %>% filter(Name == "Call of Duty: Modern Warfare 3")
```

```
##                              Name Platform Year_of_Release   Genre
## 1 Call of Duty: Modern Warfare 3     X360            2011 Shooter
## 2 Call of Duty: Modern Warfare 3      PS3            2011 Shooter
## 3 Call of Duty: Modern Warfare 3       PC            2011 Shooter
## 4 Call of Duty: Modern Warfare 3      Wii            2011 Shooter
##    Publisher NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales
## 1 Activision     9.04     4.24     0.13        1.32        14.73
## 2 Activision     5.54     5.73     0.49        1.57        13.32
## 3 Activision     0.41     0.98     0.00        0.32         1.72
## 4 Activision     0.55     0.20     0.00        0.08         0.82
##   Critic_Score Critic_Count User_Score User_Count
## 1           88           81        3.4       8713
## 2           88           39        3.2       5234
## 3           78           26        2.5       5664
## 4           70           16        1.8        442
##                            Developer Rating         AvgP      AvgT
## 1 Infinity Ward, Sledgehammer Games      M 0.0009296454 0.1162633
## 2 Infinity Ward, Sledgehammer Games      M 0.0007451280 0.1162633
## 3 Infinity Ward, Sledgehammer Games      M 0.0004590395 0.1162633
## 4      Treyarch, Sledgehammer Games      M 0.0036199095 0.1162633
```

Notice the huge difference between audience and critic score. Critics usually get too technical (if they even try to do a proper critic), and I personally believe the users are more to trust since they focus more on the "Fun" of the game. Regardless of what may be true, the AvgT column works to standardize a fixed proportion so we can aim to assess each game similarly. I'm honestly not 100% sure if this is correct, but I sincerely hope it's not too wrong.

Now let's create a "True_Score" column with the weighted average of the two. We are giving the users' score a weight of almost twice the one of the critics' score. Of course this is a consideration opened to debate.

```
vgdata <- vgdata %>%
  mutate(True_Score= ((Critic_Score/10)*AvgT*3)+(as.numeric(as.character(User_Score))*(1-AvgT*3)))
#I had to use the as.numeric(as.character(User_Score)) part because User_Score was a factor and
#for reasons that go way beyond my current understandment, it had to be done like that when
#working with factors, god bless stackoverflow.com.
```

Now we have a column with values over which to base our judgement.

Ok now let's reorganize the data frame from highest to lowest True_Score.

```
vgdata_ordered <- vgdata %>%  arrange(desc(True_Score))
```

However, we must notice that on the Name column we may find the same game more than once, this is because the very same game can be ported to different consoles, and the data set shows each. It makes sense to asume that a game is not enjoyable if it can't run properly, so let's keep just the best performing of each according to our True_Score, this would also make it more "fair" for the games that were not ported. Since we had them already ordered by True_Score it suffices with just filtering the repeated names.

```
vgdata_ordered <- vgdata_ordered[!duplicated(vgdata_ordered$Name),]
```

Now let's check the first 20 games sorted by True_score:

```
vgdata_ordered %>% group_by(Name, True_Score) %>% summarise() %>%
  arrange(desc(True_Score)) %>% print(n = 20)
```

```
## # A tibble: 4,435 x 2
## # Groups:   Name [4,435]
##    Name                                    True_Score
##    <fct>                                        <dbl>
##  1 Resident Evil 4                               9.47
##  2 Metroid Prime                                 9.44
##  3 The Orange Box                                9.40
##  4 Metal Gear Solid                              9.4
##  5 Castlevania: Symphony of the Night            9.37
##  6 Skies of Arcadia                              9.37
##  7 The Legend of Zelda: Twilight Princess        9.34
##  8 Metal Gear Solid 3: Subsistence               9.33
##  9 Super Mario Galaxy 2                          9.31
## 10 Okami                                         9.3
## 11 The Witcher 3: Wild Hunt                      9.3
## 12 Half-Life                                     9.27
## 13 Half-Life 2                                   9.27
## 14 Tekken 3                                      9.27
## 15 Star Wars: Knights of the Old Republic        9.27
## 16 The Last of Us                                9.24
## 17 The Legend of Zelda: A Link to the Past       9.24
## 18 Persona 4: Golden                             9.23
## 19 Golden Sun                                    9.23
## 20 Metal Gear Solid 3: Snake Eater               9.23
## # ... with 4,415 more rows
```

We can see that there are some parts of the ranking that don't really make sense. Let's see an example.

```
vgdata_ordered %>% filter(Name%in%c("Boktai: The Sun is in Your Hand"))
```

```
##                                Name Platform Year_of_Release        Genre
## 1 Boktai: The Sun is in Your Hand      GBA            2003 Role-Playing
##                        Publisher NA_Sales EU_Sales JP_Sales Other_Sales
## 1 Konami Digital Entertainment      0.1     0.04        0           0
##   Global_Sales Critic_Score Critic_Count User_Score User_Count Developer
## 1         0.15           83           31        9.6         16      KCEJ
##   Rating     AvgP      AvgT True_Score
## 1      E 0.19375 0.1162633   9.146573
```

We can see that the reason this game did so great in our analysis was because it had a tiny amount of User reviews with great score, which is a similar scenario with the cult movies in the Movielens Project, so it makes sense to see cases like this as cult games, with great score among its cult. This means that we must adjust our analysis and avoid games with minuscular amount of User reviews.

Perhaps it makes sense to think that a great game would sell a lot, but this isn't a very safe assumption from what we saw with the Call of Duty: MOdern Warfare check, some games can do great in sales thanks to it's marketing campaign or just because of its name. Maybe it is safer to think that a great game would motivate a lot of users to leave great reviews.

Let's check both and see how we do!

Let's check first at the top 50% games in global sales performance and order them by True_Score. We are pickign this amount of games for further analysis.

```r
vgdata_ordered_GSUC <- vgdata_ordered %>% arrange(desc(Global_Sales))
#Let's arrange the games from top to bottom in Global Sales.
```

We'll call this object "vgdata_ordered_SC" since we are first filtering by global sales (GS) and then User Counts (UC)

```r
n <- length(vgdata_ordered_GSUC$Name) #Let's see how many games we have.
Q2 <- as.integer(n*0.5) #We now grab the half of them
vgdata_ordered_GSUC <- vgdata_ordered_GSUC[1:Q2,]
#Since we had already ordered them by sales, we are picking the first half.
vgdata_ordered_GSUC <- vgdata_ordered_GSUC %>% arrange(desc(User_Count))
#And now we arrange him by User_COunt
n <- length(vgdata_ordered_GSUC$Name)
P90 <- as.integer(n*0.1) #We now grab the first 10% of the User_Counts
#since there are greatdisparities between  the amount of them,
#and we want the best of the best here. It's not anything to be the game that
#was able to pull the biggest amount of User reviews for example.
vgdata_ordered_GSUC <- vgdata_ordered_GSUC[1:P90,]
vgdata_ordered_GSUC %>% group_by(Name, True_Score) %>% summarise() %>%
  arrange(desc(True_Score)) %>% print(n = 20)
```

```
## # A tibble: 221 x 2
## # Groups:   Name [221]
##    Name                                True_Score
##    <fct>                                    <dbl>
##  1 Resident Evil 4                           9.47
##  2 Metroid Prime                             9.44
##  3 Metal Gear Solid                          9.4
##  4 Super Mario Galaxy 2                      9.31
##  5 The Witcher 3: Wild Hunt                  9.3
##  6 Half-Life                                 9.27
##  7 Half-Life 2                               9.27
##  8 The Last of Us                            9.24
##  9 Persona 4: Golden                         9.23
## 10 Metal Gear Solid 3: Snake Eater           9.23
## 11 Final Fantasy VII                         9.2
## 12 Super Mario Galaxy                        9.18
## 13 Grand Theft Auto: San Andreas             9.17
## 14 Red Dead Redemption                       9.17
## 15 BioShock                                  9.14
## 16 Mass Effect 2                             9.14
## 17 The Legend of Zelda: The Wind Waker       9.14
## 18 Fire Emblem: Awakening                    9.13
## 19 Super Mario 3D World                      9.10
## 20 Shadow of the Colossus                    9.1
## # ... with 201 more rows
```

Now let's try it the other order around

```r
vgdata_ordered_CUSG <- vgdata_ordered %>% arrange(desc(User_Count))
n <- length(vgdata_ordered_CUSG$Name)
Q2 <- as.integer(n*0.1)
```

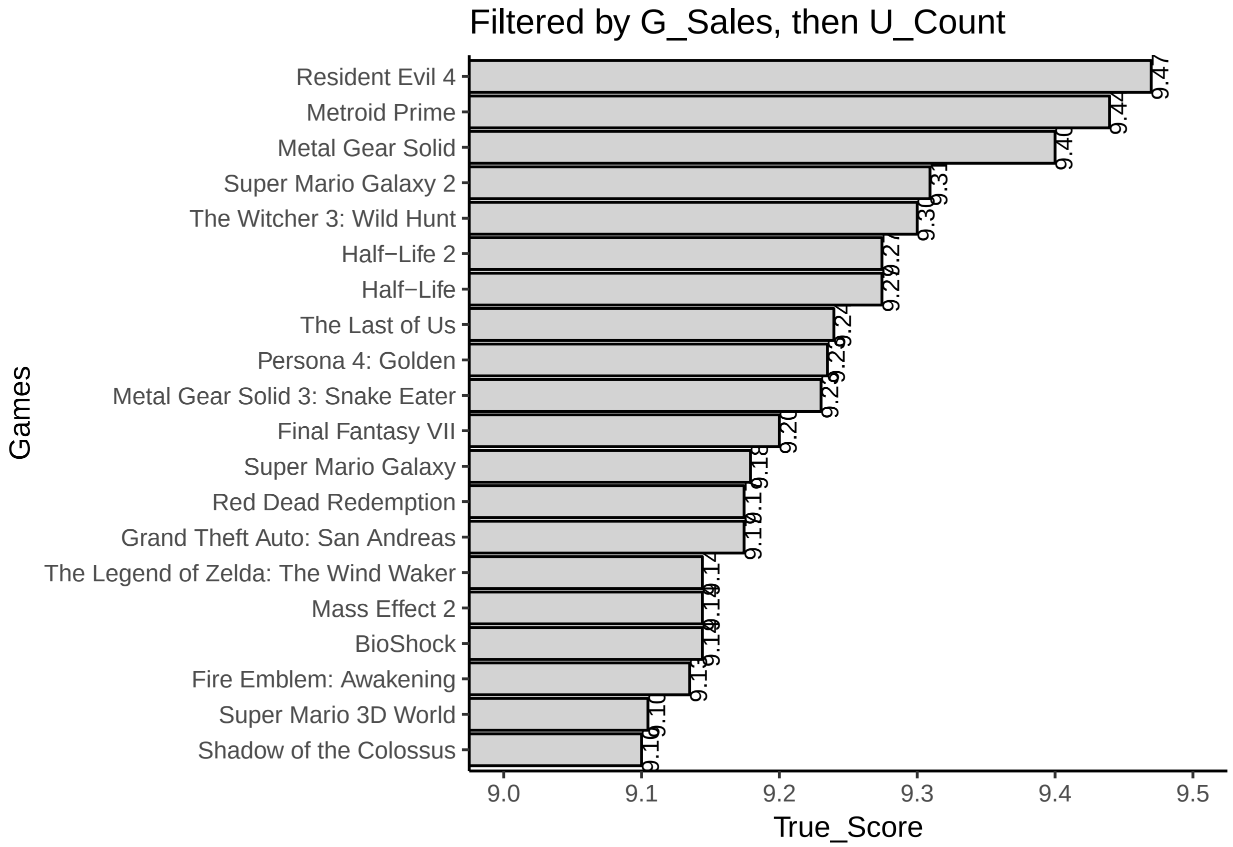
```
vgdata_ordered_CUSG <- vgdata_ordered_CUSG[1:Q2,]
vgdata_ordered_CUSG <- vgdata_ordered_CUSG %>% arrange(desc(Global_Sales))
n <- length(vgdata_ordered_CUSG$Name)
P90 <- as.integer(n*0.5)
vgdata_ordered_CUSG <- vgdata_ordered_CUSG[1:P90,]
vgdata_ordered_CUSG %>% group_by(Name, True_Score) %>% summarise() %>%
  arrange(desc(True_Score)) %>% print(n = 20)
```

```
## # A tibble: 221 x 2
## # Groups:   Name [221]
##    Name                                   True_Score
##    <fct>                                       <dbl>
##  1 Resident Evil 4                              9.47
##  2 Metroid Prime                                9.44
##  3 Metal Gear Solid                             9.4
##  4 The Legend of Zelda: Twilight Princess       9.34
##  5 Super Mario Galaxy 2                         9.31
##  6 Half-Life                                    9.27
##  7 Half-Life 2                                  9.27
##  8 Tekken 3                                     9.27
##  9 Star Wars: Knights of the Old Republic       9.27
## 10 The Last of Us                               9.24
## 11 Metal Gear Solid 3: Snake Eater              9.23
## 12 Final Fantasy VII                            9.2
## 13 Super Mario Galaxy                           9.18
## 14 Grand Theft Auto: San Andreas                9.17
## 15 Red Dead Redemption                          9.17
## 16 BioShock                                     9.14
## 17 Mass Effect 2                                9.14
## 18 The Legend of Zelda: The Wind Waker          9.14
## 19 Fire Emblem: Awakening                       9.13
## 20 Super Smash Bros. Melee                      9.13
## # ... with 201 more rows
```

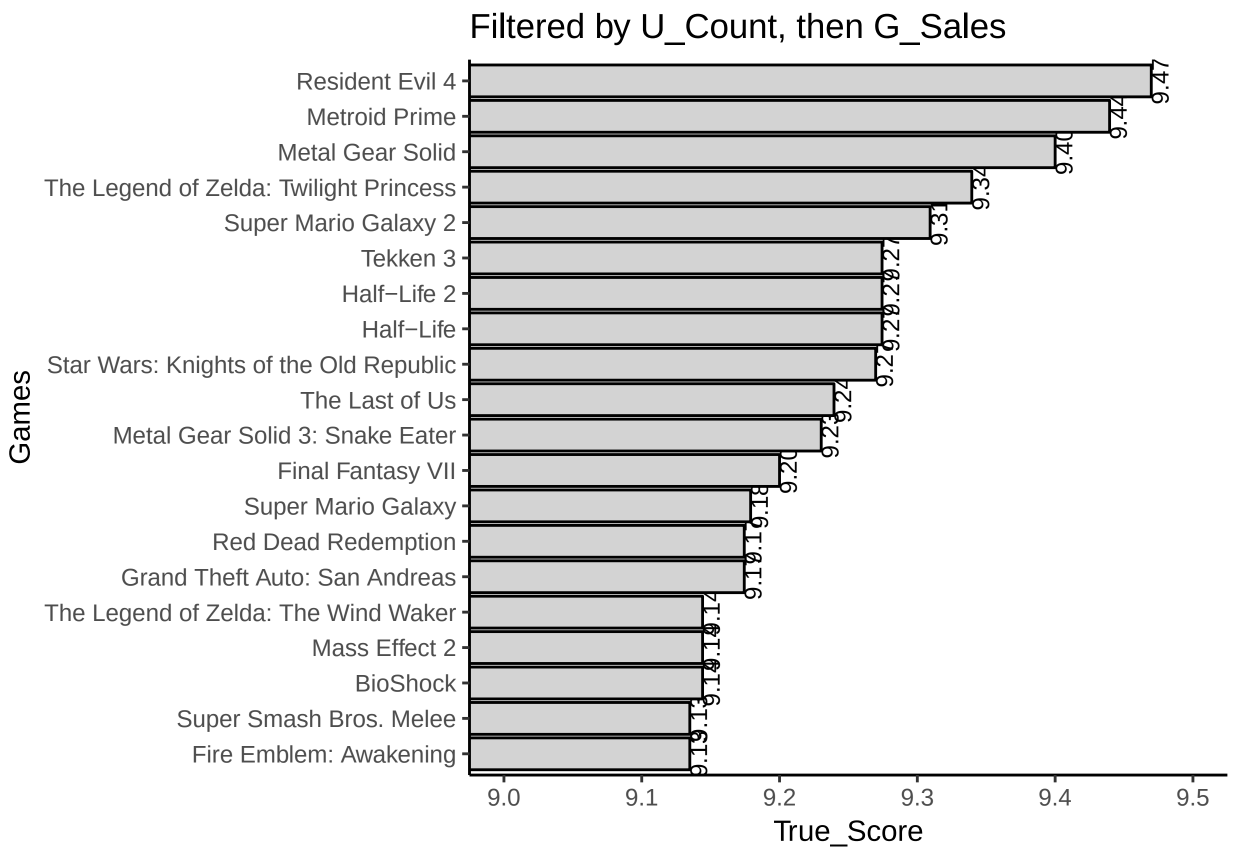
The dataset at hand didn't have N64 games, so that's kind of a bummer, and it's only up to 2017. Let's plot them together and see how they turned out.

```
vgdata_ordered_GSUC_20 <- vgdata_ordered_GSUC %>%  arrange(desc(True_Score)) %>% .[1:20,]
P_GSUC_Bar <- vgdata_ordered_GSUC_20 %>% mutate(reorder = reorder(Name, True_Score))
P_GSUC_Bar <- vgdata_ordered_GSUC_20 %>% mutate(reorder = reorder(Name, True_Score)) %>%
  ggplot(aes(reorder, True_Score))+
  geom_text(aes(label=format(round(True_Score, 2), nsmall = 2)), size = 3, vjust = 1, angle = 90)+
  geom_bar(stat = "identity", color = "black", fill = "lightgray") +
  coord_flip(ylim = c(9, 9.5)) +
  theme(axis.text.y = element_text(size = 6)) +
  xlab("Games")+
  ylab("True_Score")+
  ggtitle("Filtered by G_Sales, then U_Count") +
  scale_color_discrete(name = "Platform") +
  theme_classic()

vgdata_ordered_CUSG_20 <- vgdata_ordered_CUSG %>%  arrange(desc(True_Score)) %>% .[1:20,]
P_CUSG_Bar <- vgdata_ordered_CUSG_20 %>% mutate(reorder = reorder(Name, True_Score))
```

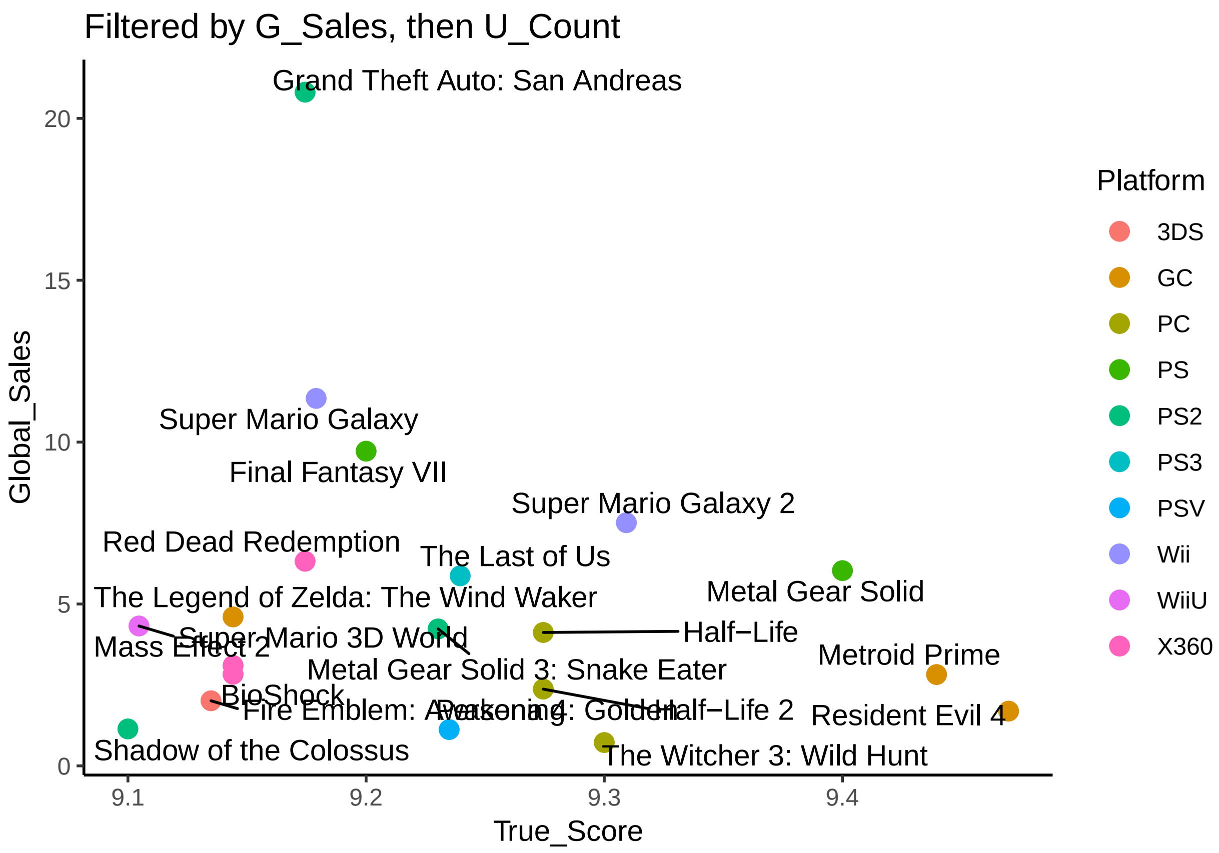
```r
P_CUSG_Bar <- vgdata_ordered_CUSG_20 %>% mutate(reorder = reorder(Name, True_Score)) %>%
  ggplot(aes(reorder, True_Score))+
  geom_text(aes(label=format(round(True_Score, 2), nsmall = 2)), size = 3, vjust = 1, angle = 90)+
  geom_bar(stat = "identity", color = "black", fill = "lightgray") +
  coord_flip(ylim = c(9, 9.5)) +
  theme(axis.text.y = element_text(size = 6)) +
  xlab("Games")+
  ylab("True_Score")+
  ggtitle("Filtered by U_Count, then G_Sales") +
  scale_color_discrete(name = "Platform") +
  theme_classic()

P_GSUC_Point <- vgdata_ordered_GSUC_20 %>% ggplot(aes(True_Score, Global_Sales, label = Name))+
  geom_point(aes(col= Platform), size = 3) +
  geom_text_repel() +
  ggtitle("Filtered by G_Sales, then U_Count") +
  xlab("True_Score") +
  ylab("Global_Sales") +
  scale_color_discrete(name = "Platform") +
  theme_classic()

P_CUSG_Point <- vgdata_ordered_CUSG_20 %>% ggplot(aes(True_Score, Global_Sales, label = Name))+
  geom_point(aes(col= Platform), size = 3) +
  geom_text_repel() +
  ggtitle("Filtered by U_Count, then G_Sales") +
  xlab("True_Score") +
  ylab("Global_Sales") +
  scale_color_discrete(name = "Platform") +
  theme_classic()

#Let's plot them and see!.
P_GSUC_Bar
```


Filtered by G_Sales, then U_Count
Games
Resident Evil 4
Metroid Prime
Metal Gear Solid
Super Mario Galaxy 2
The Witcher 3: Wild Hunt
Half−Life 2
Half−Life
The Last of Us
Persona 4: Golden
Metal Gear Solid 3: Snake Eater
Final Fantasy VII
Super Mario Galaxy
Red Dead Redemption
Grand Theft Auto: San Andreas
The Legend of Zelda: The Wind Waker
Mass Effect 2
BioShock
Fire Emblem: Awakening
Super Mario 3D World
Shadow of the Colossus
9.47
9.44
9.40
9.31
9.30
9.27
9.27
9.24
9.23
9.23
9.20
9.18
9.17
9.17
9.14
9.14
9.14
9.13
9.10
9.10
9.0
9.1
9.2
9.3
9.4
9.5
True_Score

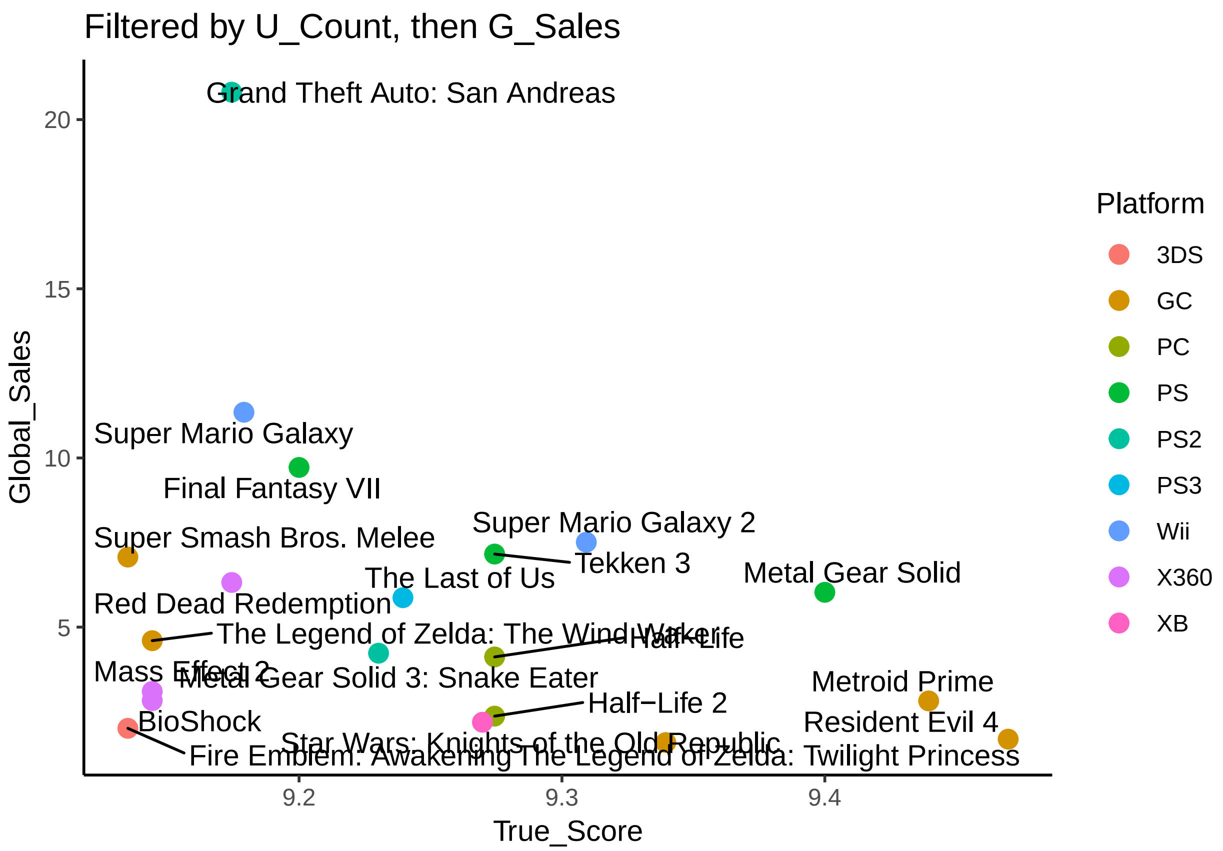
```
P_CUSG_Bar
```



```
#And let's see their performance in sales for more insight.
P_GSUC_Point
```


Filtered by G_Sales, then U_Count
Global_Sales
20
15
10
5
0
9.1
9.2
9.3
9.4
True_Score
Grand Theft Auto: San Andreas
Super Mario Galaxy
Final Fantasy VII
Super Mario Galaxy 2
Red Dead Redemption
The Last of Us
The Legend of Zelda: The Wind Waker
Metal Gear Solid
Half−Life
Super Mario 3D World
Mass Effect 2
Metroid Prime
Metal Gear Solid 3: Snake Eater
BioShock
Fire Emblem: Awakening
Persona 4: Golden
Half−Life 2
Resident Evil 4
Shadow of the Colossus
The Witcher 3: Wild Hunt
Platform
3DS
GC
PC
PS
PS2
PS3
PSV
Wii
WiiU
X360

```
P_CUSG_Point
```



Now I am aware this could get into a deeply subjective argument. I personally have some disagreements with the results but I believe the parameters we used have a certain degree of being right. This is all of course opened to discussion (please remember there weren't any N64 games in the dataset). Under my personal judgement I believe that the more prudent resulting arrangements would be the ones produced by filtering first by Global_Sales and then by User_Counts, since the games in the first one seem to fit more than the ones in the second.

I would then believe the following to be the top 10 games from our dataset:

```
vgdata_ordered_GSUC_top10 <- vgdata_ordered_GSUC %>% arrange(desc(True_Score)) %>% .[1:10,]
top10vg <- vgdata_ordered_GSUC_top10 %>%
  group_by(Name, Platform, Publisher, Year_of_Release, True_Score) %>%
  summarise() %>% arrange(desc(True_Score))

top10vg %>% knitr::kable()
```

Name	Platform	Publisher	Year_of_Release	True_Score
Resident Evil 4	GC	Capcom	2005	9.469758
Metroid Prime	GC	Nintendo	2002	9.439516
Metal Gear Solid	PS	Konami Digital Entertainment	1998	9.400000
Super Mario Galaxy 2	Wii	Nintendo	2010	9.309274
The Witcher 3: Wild Hunt	PC	Namco Bandai Games	2015	9.300000
Half-Life	PC	Vivendi Games	1997	9.274395
Half-Life 2	PC	Vivendi Games	2004	9.274395
The Last of Us	PS3	Sony Computer Entertainment Europe	2013	9.239516
Persona 4: Golden	PSV	Atlus	2012	9.234879

Name	Platform	Publisher	Year_of_Release	True_Score
Metal Gear Solid 3: Snake Eater	PS2	Konami Digital Entertainment	2004	9.230242

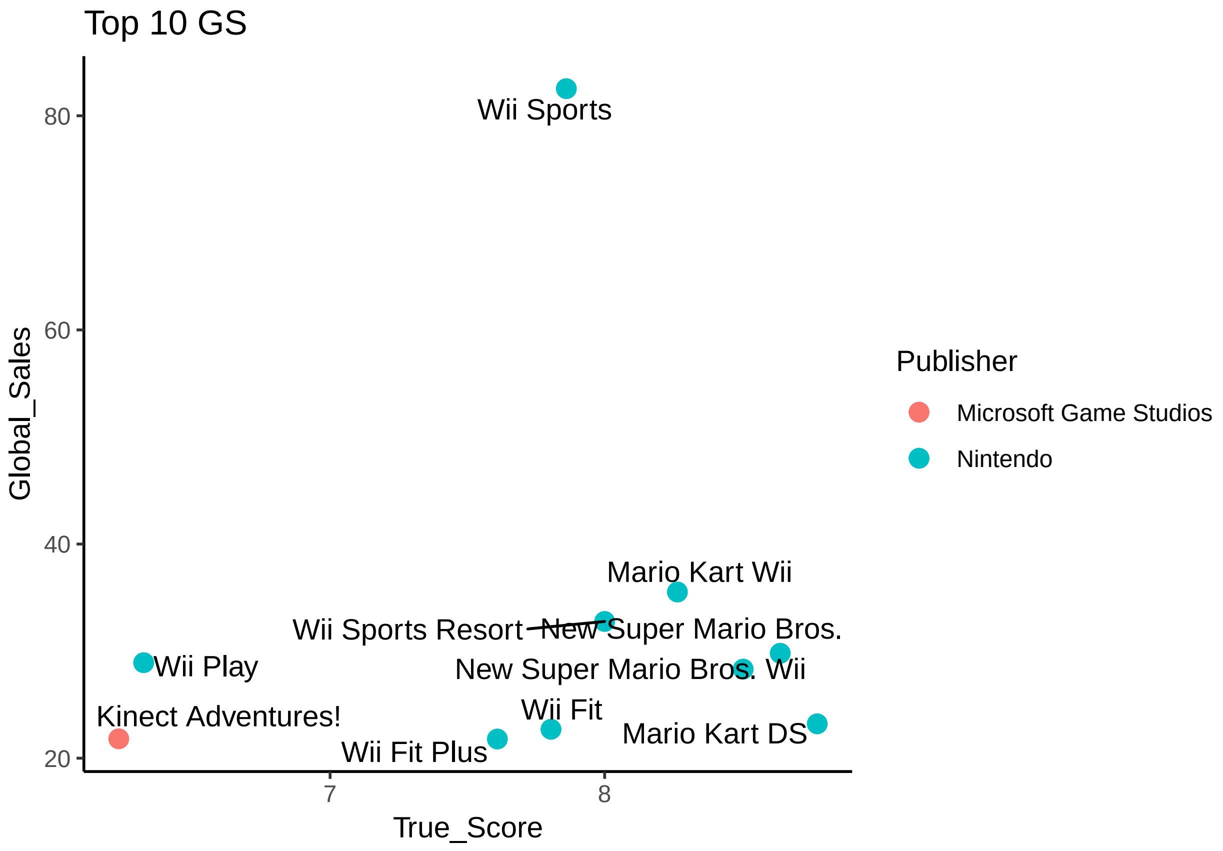
And there we go! The Top 10 games from the data we have.

3.2 Publisher wars!

Let's check the results of every region when choosing the top 10 selling games and their True_Score

Let's begin globally.

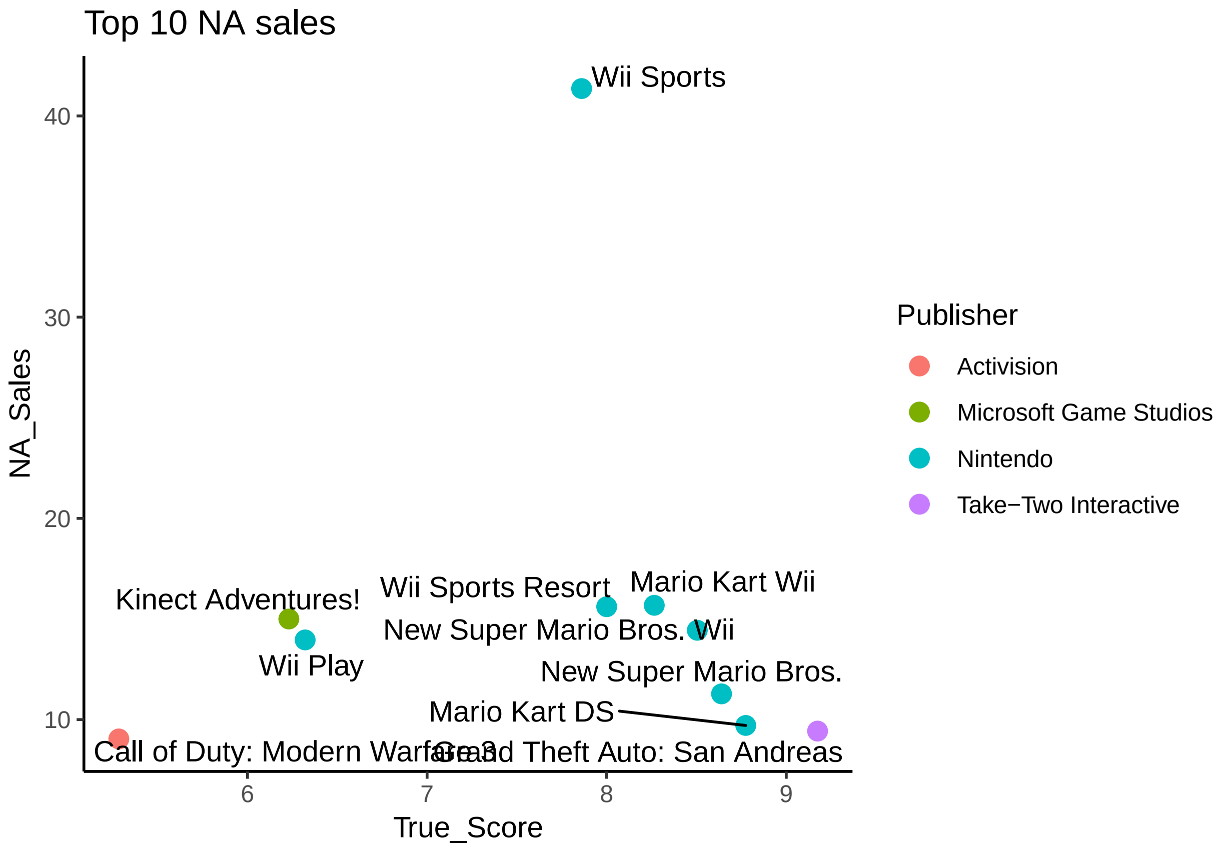
```
vgdata_sales_global <- vgdata_ordered %>% arrange(desc(Global_Sales)) %>% .[1:10,]
Global_Sales_Point <- vgdata_sales_global %>% ggplot(aes(True_Score, Global_Sales, label = Name))+
  geom_point(aes(col= Publisher), size = 3) +
  geom_text_repel() +
  ggtitle("Top 10 GS ") +
  xlab("True_Score") +
  ylab("Global_Sales") +
  scale_color_discrete(name = "Publisher") +
  theme_classic()
Global_Sales_Point
```



```
#We can see that Nintendo has a strong position with high True Scores
```

Now let's check in North America

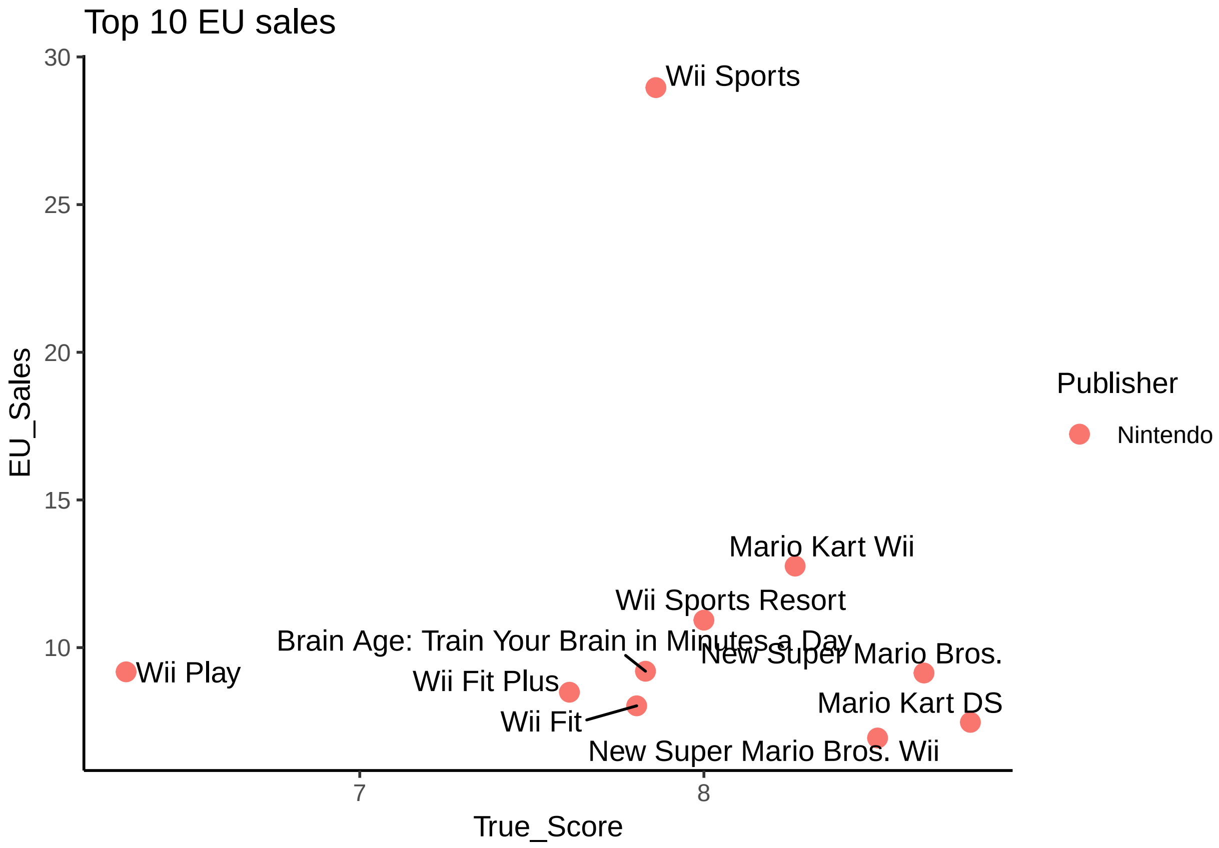
```
vgdata_sales_NA <- vgdata_ordered %>% arrange(desc(NA_Sales)) %>% .[1:10,]

NA_Sales_Point <- vgdata_sales_NA %>% ggplot(aes(True_Score, NA_Sales, label = Name))+
  geom_point(aes(col= Publisher), size = 3) +
  geom_text_repel() +
  ggtitle("Top 10 NA sales ") +
  xlab("True_Score") +
  ylab("NA_Sales") +
  scale_color_discrete(name = "Publisher") +
  theme_classic()
NA_Sales_Point
```



```
#Nintendo also has a strong position in North American sales, although we can see two more
#publishers than before.
```

And now with the European Union

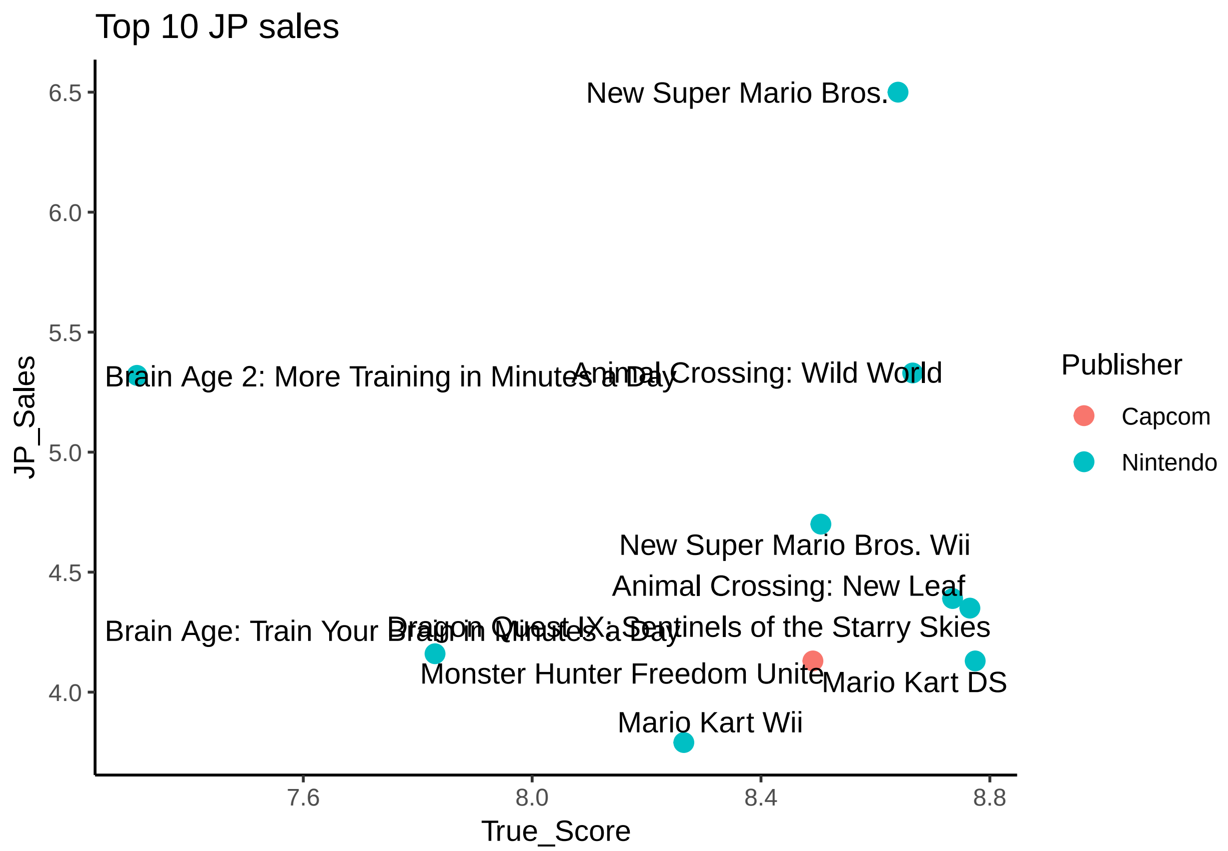
```
vgdata_sales_EU <- vgdata_ordered %>% arrange(desc(EU_Sales)) %>% .[1:10,]

EU_Sales_Point <- vgdata_sales_EU %>% ggplot(aes(True_Score, EU_Sales, label = Name))+
  geom_point(aes(col= Publisher), size = 3) +
  geom_text_repel() +
  ggtitle("Top 10 EU sales ") +
  xlab("True_Score") +
  ylab("EU_Sales") +
  scale_color_discrete(name = "Publisher") +
  theme_classic()
EU_Sales_Point
```



```
#We can see that Nintendo dominates the top 10 selling games in the EU
```

And last but not at all least, Japan

```
vgdata_sales_JP <- vgdata_ordered %>% arrange(desc(JP_Sales)) %>% .[1:10,]

JP_Sales_Point <- vgdata_sales_JP %>% ggplot(aes(True_Score, JP_Sales, label = Name))+
  geom_point(aes(col= Publisher), size = 3) +
  geom_text_repel() +
  ggtitle("Top 10 JP sales ") +
  xlab("True_Score") +
  ylab("JP_Sales") +
  scale_color_discrete(name = "Publisher") +
  theme_classic()
JP_Sales_Point
```



```
#The top 10 games sold in Japan are also dominated by Nintendo, Capcom is the only Publisher
#besides it with one game.
```

Now let's check who the top 10 Publishers would be ordered by Global Sales

```
Top_Pub <- vgdata_ordered %>%
  group_by(Publisher) %>%
  summarise(Global_Sales_Per_Pub = sum(Global_Sales), Games_Published = n(), Average_Score = sum(True_S
  arrange(desc(Global_Sales_Per_Pub)) %>% .[1:10,]
Top_Pub
```

```
## # A tibble: 10 x 4
##    Publisher              Global_Sales_Per_~ Games_Published Average_Score
##    <fct>                               <dbl>           <int>         <dbl>
##  1 Nintendo                            844.              292          7.81
##  2 Electronic Arts                     402.              398          7.55
##  3 Sony Computer Entertai~             373.              286          7.64
##  4 Activision                          235.              213          7.30
##  5 Microsoft Game Studios              216.              133          7.54
##  6 Ubisoft                             190.              280          7.05
##  7 Take-Two Interactive                173.              132          7.45
##  8 Konami Digital Enterta~             106.              177          7.33
##  9 Sega                                101.              191          7.53
## 10 THQ                                  90.2             180          7.29
```

```
#Nintendo wins in the sales area, and it's also the one with the highest Average Score among the 10.
```

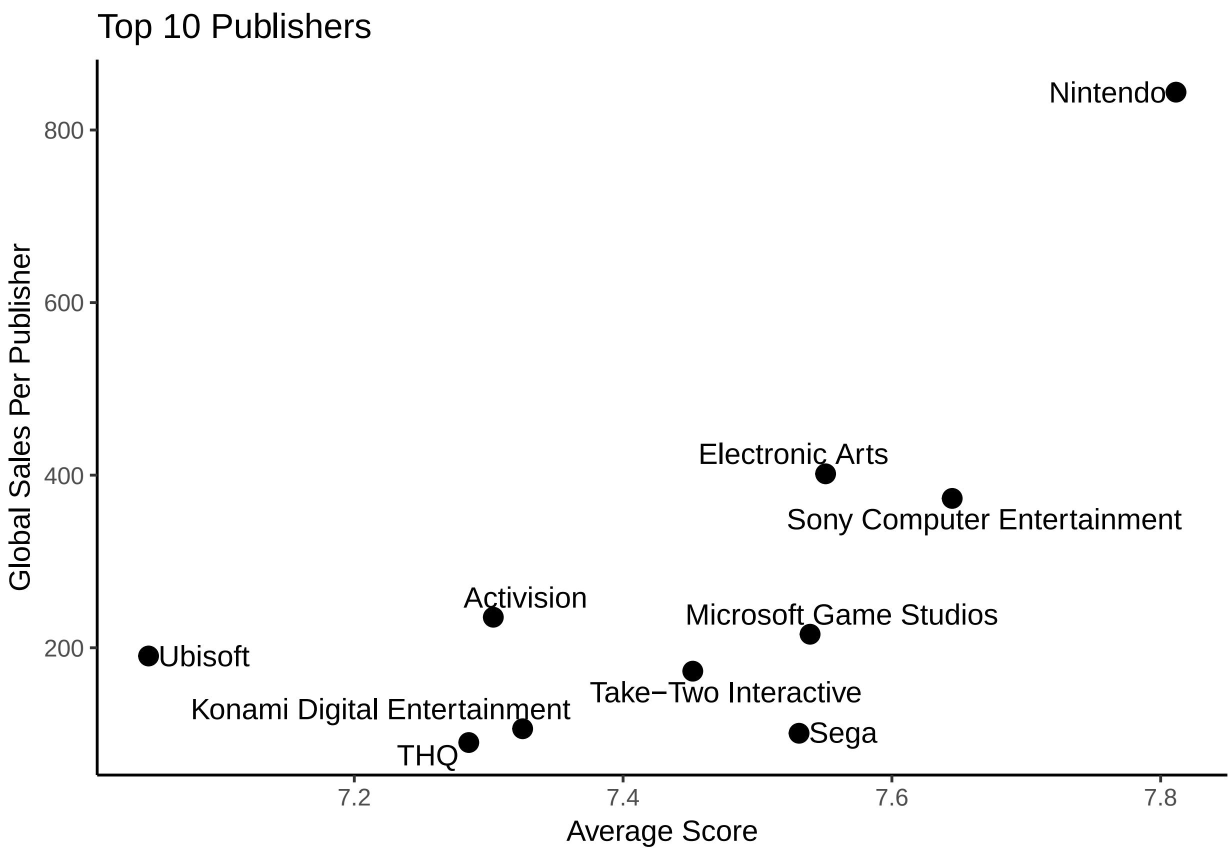
We could check the order by Score but we run once again into the segmentation issue. The publisher with the best score has only one game published.

```
Top_Pub2 <- vgdata_ordered %>%
  group_by(Publisher) %>%
  summarise(Global_Sales_Per_Pub = sum(Global_Sales), Games_Published = n(), Average_Score = sum(True_S
  arrange(desc(Average_Score)) %>% .[1:10,]
Top_Pub2
```

```
## # A tibble: 10 x 4
##    Publisher              Global_Sales_Per_P~ Games_Published Average_Score
##    <fct>                                <dbl>           <int>         <dbl>
##  1 Valve Software                        0.76               1          9.04
##  2 Square                                0.52               1          9.03
##  3 Blue Byte                             0.01               1          8.90
##  4 Havas Interactive                     0.13               1          8.8
##  5 Graphsim Entertainment                0.02               1          8.74
##  6 Number None                           0.03               1          8.74
##  7 Kadokawa Shoten                       0.9                2          8.72
##  8 GT Interactive                        8.5                3          8.71
##  9 2D Boy                                0.04               1          8.67
## 10 Psygnosis                             1.24               4          8.67
```

Let's graph the performance of the Publishers ordered by sales.

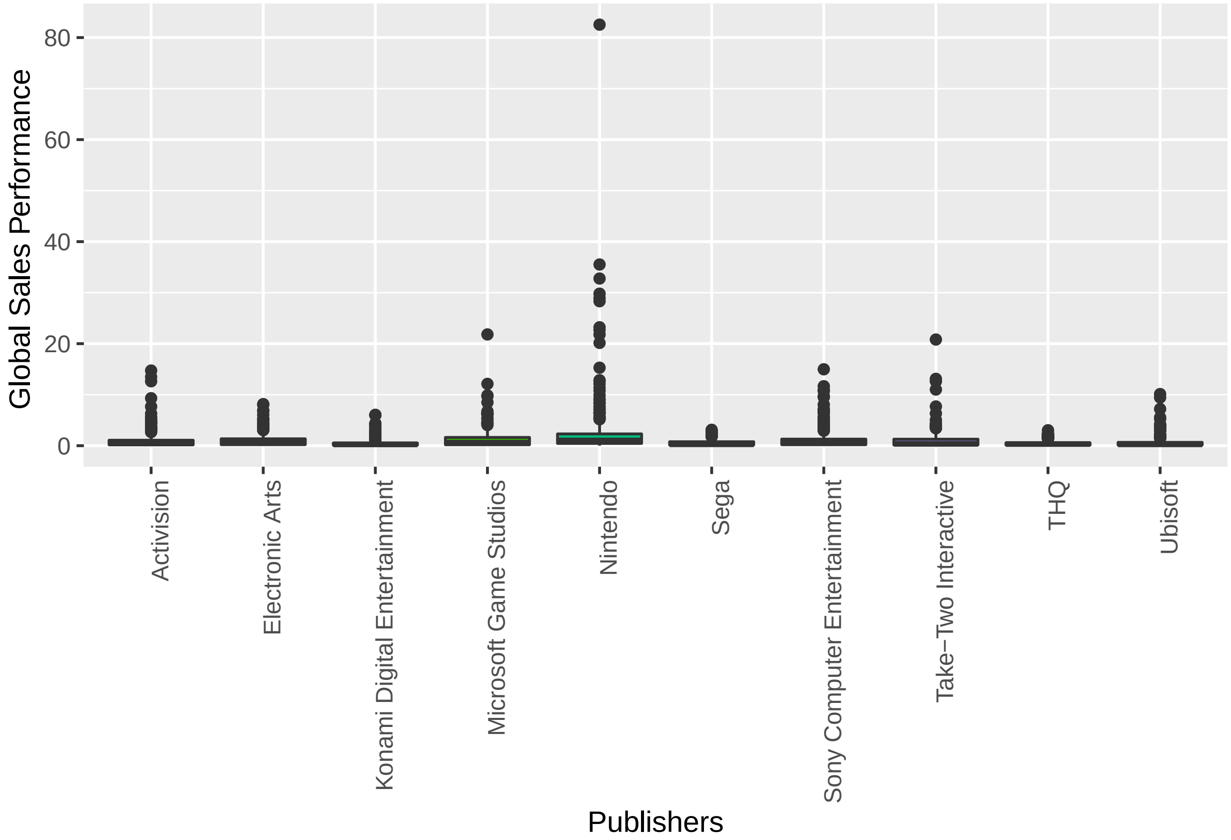
```
Top_Pub_Point <-Top_Pub %>% ggplot(aes(Average_Score, Global_Sales_Per_Pub, label = Publisher))+
  geom_point(size = 3) +
  geom_text_repel() +
  ggtitle("Top 10 Publishers") +
  xlab("Average Score") +
  ylab("Global Sales Per Publisher") +
  scale_color_discrete(name = "Publisher") +
  theme_classic()
Top_Pub_Point
```



```
#So yeah, Nintendo has an overwhelming position when noticing that it has almost twice the sales
#of the 2 Publisher, while maintaining a high Score.
```

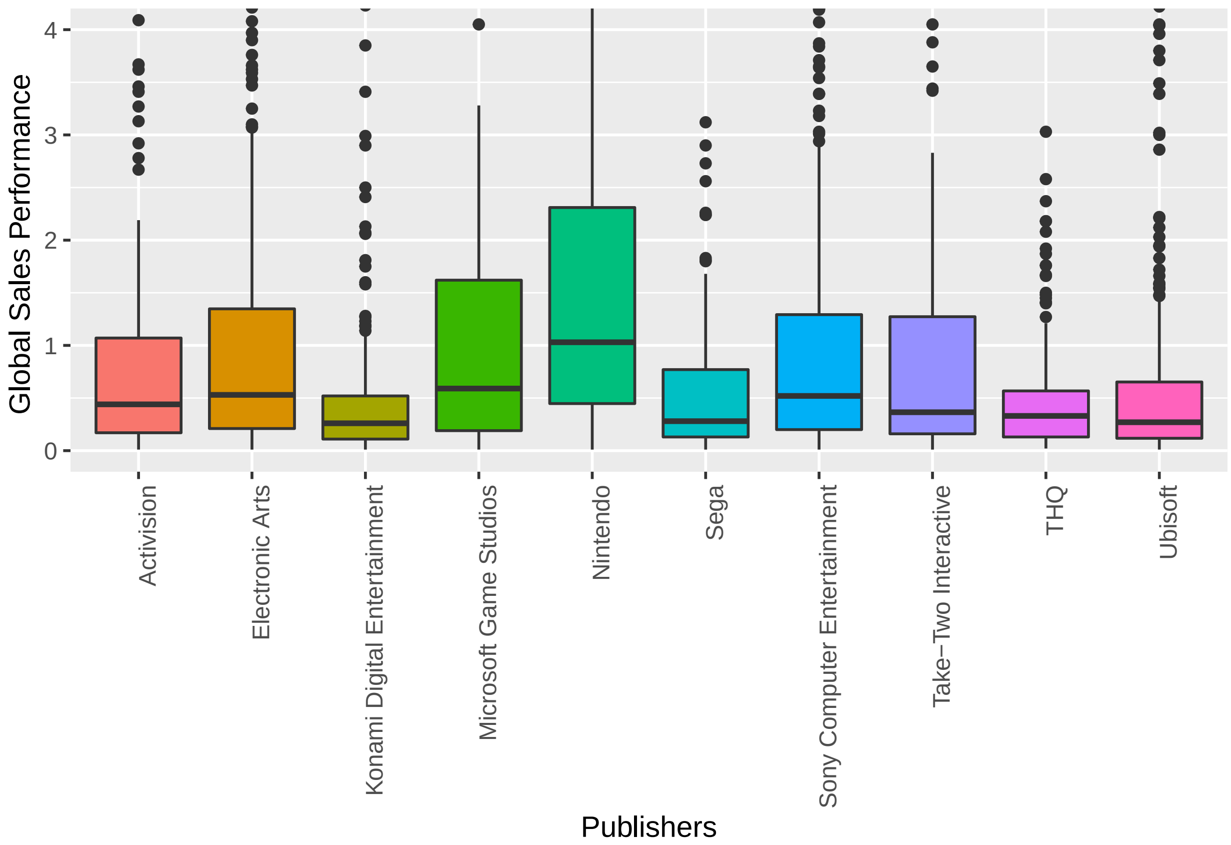
Now let's check how every Publisher did in Sales using a boxplot, to check the statistical distribution of each Publisher's performance.

```
Pub_List <- Top_Pub$Publisher

Top_Pub_Box <- vgdata_ordered %>% dplyr::filter(Publisher %in% Pub_List) %>%
  mutate(median = median(Global_Sales)) %>% mutate(reordered = reorder(Publisher, median, order = TRUE)
Top_Pub_Box %>%
  ggplot(aes(reordered, Global_Sales, fill = Publisher)) +
  geom_boxplot(show.legend = FALSE) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Publishers") +
  ylab("Global Sales Performance ")
```



Well that's not so informatieve, although we can see that the disparity of the distribution is huge, and that the most selling game that each Publisher has greatly outperforms the rest. It's quite a heterogeneous distribution. Let's use coord_cartesian() to zoom in:

```
Top_Pub_Box %>%
  ggplot(aes(reordered, Global_Sales, fill = Publisher)) +
  geom_boxplot(show.legend = FALSE) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  coord_cartesian(
    ylim = c(0,4)
  )+
  xlab("Publishers") +
  ylab("Global Sales Performance ")
```



```
#Better! We can now have a better glimpse of the usual performance that each Publisher has.
#Nintendo still has a superior performance.
```

I know there may be disagreements around these results, but we have to keep in mind that the dataset doesn't have the N64 games, which would give Nintendo a huge boost. So for the time being and with the data at hand, I believe Nintendo would be the winner of the "Publisher Wars".

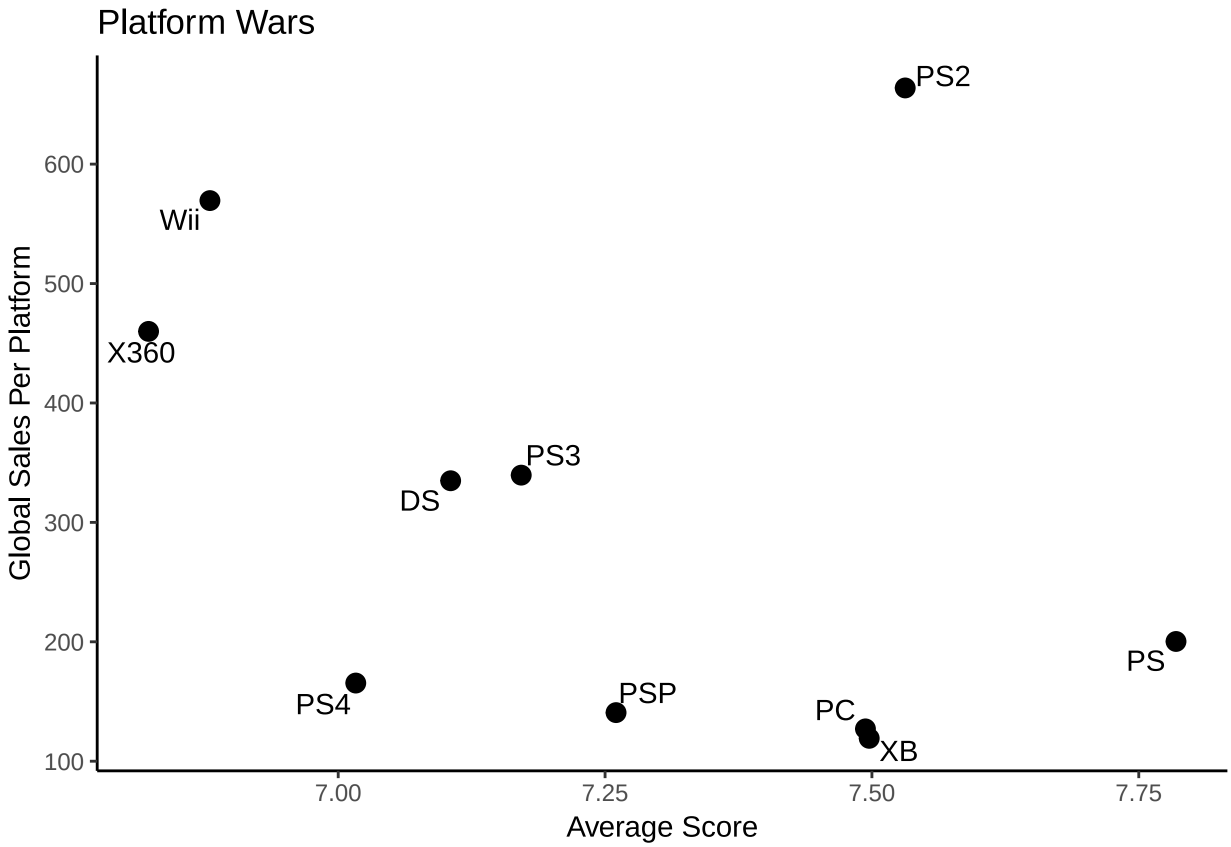
3.3 Console wars!

Let's see how each console performed when comparing Global Sales and Average Score

```
Platform_wars <- vgdata_ordered %>%
  group_by(Platform) %>%
  summarise(Global_Sales_Per_Platform = sum(Global_Sales), Games_Published = n(), Average_Score = sum(T
  arrange(desc(Global_Sales_Per_Platform)) %>% .[1:10,]
Platform_wars
```

```
## # A tibble: 10 x 4
##    Platform Global_Sales_Per_Platform Games_Published Average_Score
##    <fct>                        <dbl>           <int>         <dbl>
##  1 PS2                           664.             860          7.53
##  2 Wii                           569.             340          6.88
##  3 X360                          460.             480          6.82
##  4 PS3                           340.             334          7.17
##  5 DS                               335.            372          7.11
##  6 PS                               200.            145          7.78
##  7 PS4                              165.            154          7.02
##  8 PSP                              141.            279          7.26
##  9 PC                               127.            405          7.49
## 10 XB                               119.            325          7.50
```

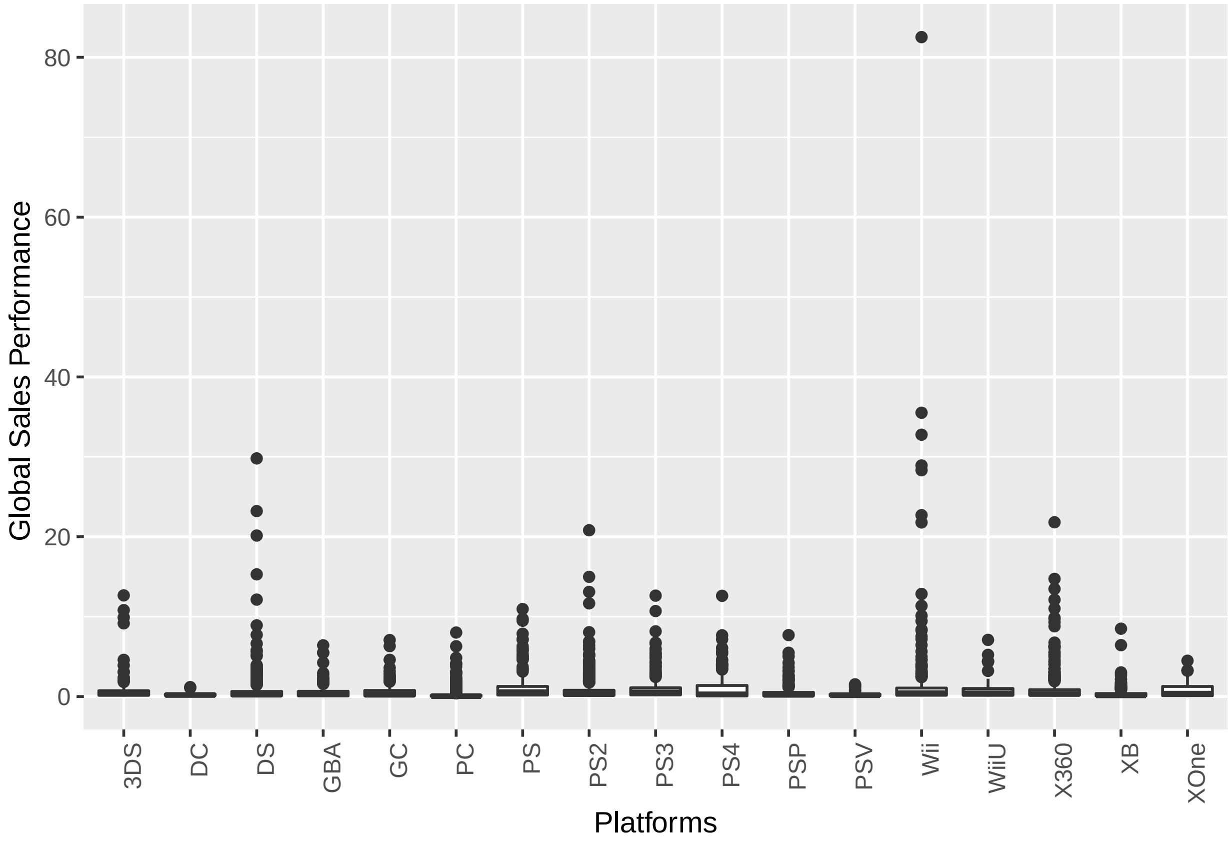
```
Platform_wars_point <-Platform_wars %>%
  ggplot(aes(Average_Score, Global_Sales_Per_Platform, label = Platform))+
  geom_point(size = 3) +
  geom_text_repel() +
  ggtitle("Platform Wars") +
  xlab("Average Score") +
  ylab("Global Sales Per Platform") +
  scale_color_discrete(name = "Platform") +
  theme_classic()
Platform_wars_point
```



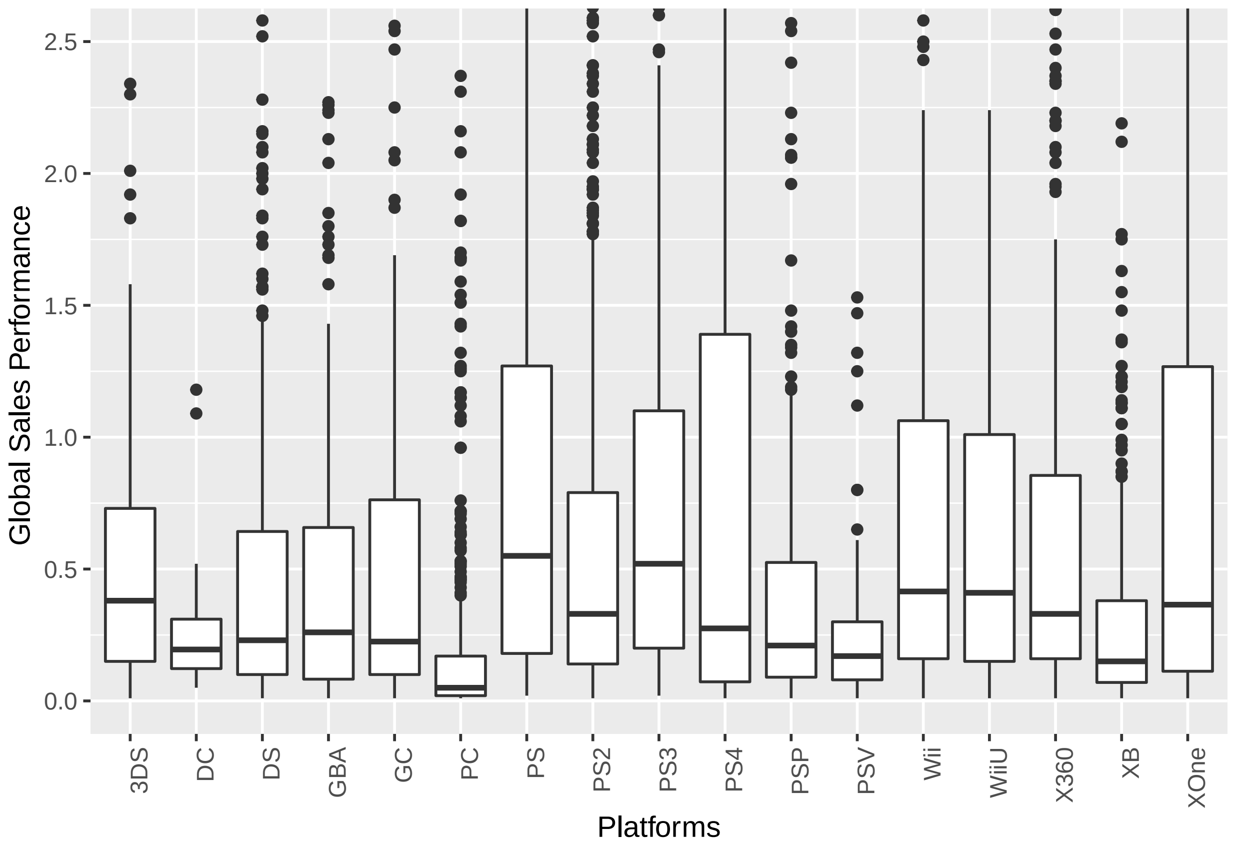
```
#Now despite Nintendo's predominance in sales, we can see that when we check by console,
#the PS2 has a great advantage over the rest.
```

Now let's see how each Platform performed using a boxplot.

```r
Platform_List <- as.character(unique(vgdata_ordered$Platform))

Platform_wars_box <- vgdata_ordered %>% dplyr::filter(Platform %in% Platform_List)

Platform_wars_box%>%
  ggplot(aes(Platform, Global_Sales)) +
  geom_boxplot(show.legend = FALSE) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Platforms") +
  ylab("Global Sales Performance ")
```



```r
#Let's use coord() to zoom up to 2.5M so we can see the distributions better

Platform_wars_box%>%
  ggplot(aes(Platform, Global_Sales)) +
  geom_boxplot(show.legend = FALSE) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  coord_cartesian(
    ylim = c(0,2.5)
  )+
  xlab("Platforms") +
  ylab("Global Sales Performance ")
```



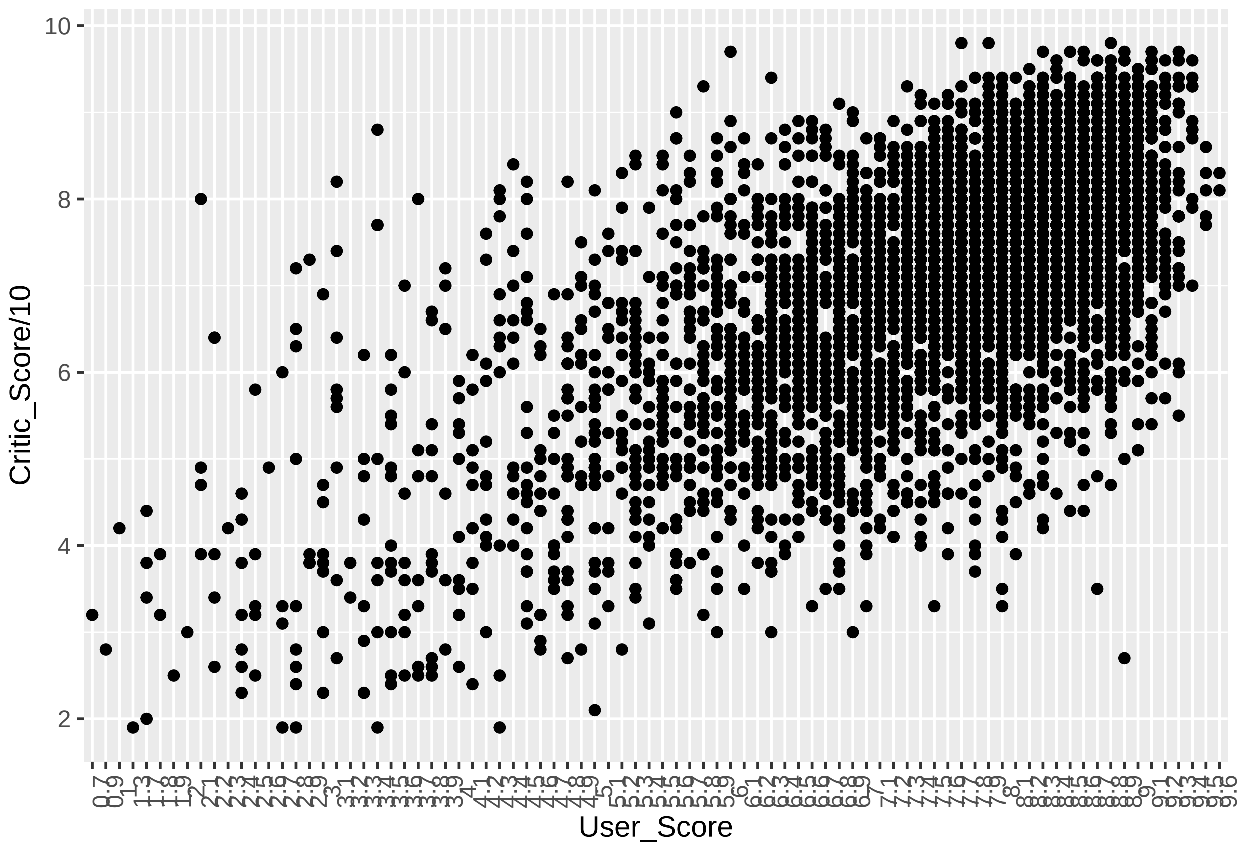
There we go. We can see that even if the distribution of the Ps2 doesn't seem to be that big of a deal, the amount of games it had pushed it to the top in sales. It could be said that it was because of a nice timing for the PS2 in the game industry, but when we see that despite its great amount of games it manages to keep the second best score among the top 10, we have to give it some credit.

The winner of the console wars would be the PS2.

3.4 Machine Learning Algorithms.

We can see that there is a certain tendency between the score provided by the users and the one provided by the Critics. Let's see if we can make an algorithm for this.

```r
vgdata_ordered %>% arrange(desc(True_Score)) %>% ggplot(aes(User_Score, Critic_Score/10)) +
  geom_point() + theme(axis.text.x = element_text(angle = 90))
```



Let's try to predict user score using critic's score, since it's how it's supposed to work right? Let's check out! I know I'm not supposed to use lm here, but I thought it would be of use to have its result as a reference, and to see if it was possible to do this.

```
set.seed(2)

vgdataml <- vgdata_ordered
vgdataml$User_Score <- as.numeric(as.character(vgdataml$User_Score))
vgdataml$Critic_Score <- as.numeric(vgdataml$Critic_Score)/10

y <- as.numeric(vgdataml$User_Score)
test_index <- createDataPartition(y, times = 1, p = 0.5, list = FALSE)

train_set <- vgdataml %>% slice(-test_index)
test_set <- vgdataml %>% slice(test_index)
fit <- lm(User_Score ~ Critic_Score, data = train_set)
y_hat <- predict(fit, test_set)
mean((y_hat - test_set$User_Score)^2)
```

```
## [1] 1.127688
```

Well that didn't go too bad.

Now let's try with other different approaches! Please keep in mind that the $User_Score column is a factor, hence why I transform it frequently.

Before going any further, both partitions must be on the same length, I dont really know why but the code doesn't cut them exactly by half so I'm deleting the excessing rows.

```
train_set$User_Score <- as.numeric(train_set$User_Score)
test_set<- test_set[-c(2217:2219),]
```

Trying Knn.

Let's begin using Knn.

Now I must clarify something. When I was making this code I tried several ML algorithms, but none of them seemed to work for me. There was a huge amount of different errors. I tried different ways to solve them, but none really seemed to work. I had no idea how to proceed, until I found the following code in Stack Overflow that arranged the data without mistakes.

```
Accuracy_knn
```

```
##   Accuracy
## 0.03700361
```

Trying Rpart

Now let's see how we do with Rpart.

```
Accuracy_rpart
```

```
##   Accuracy
## 0.04602888
```

Trying RandomForests

I tried using Random Forest but I couldn't get it to work…

```
set.seed(2)
train_rf <- train(User_Score ~ Critic_Score,
                  method = "Rborist",
                  trControl = trainControl(
                    method = "cv", number = 15,
                    verboseIter = TRUE
                  ),
                  tuneGrid = data.frame(predFixed = 2, minNode = c(3, 50)),
                  data = train_set)
```

It gets stopped and I honestly don't know why. It's pointless to run the rest of the code since it would just use the "my_data1" and the rest from Rpart.

And so our results are…

```
Results <- data_frame(method = "Knn",
                      Accuracy = Accuracy_knn)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
Results <- bind_rows(Results,
                     data_frame(method="Rpart",
                                Accuracy = Accuracy_rpart))

Results <- bind_rows(Results,
                     data_frame(method="Random Forest",
                                Notes = "Couldn't make it work :c"))

Results %>% knitr::kable()
```

method	Accuracy	Notes
Knn	0.0370036	NA
Rpart	0.0460289	NA
Random Forest	NA	Couldn't make it work :c

There we have it everyone! I guess Critics Score are not that much of a useful way to see how the users will judge a game!

4. Conclusion.

As we have seen, each of our quests have given illustrative results: +We were able to filter and organize the data helping us find the top 10 games of our data set by using a True Score and seeing it's effects over the Users. +We noticed how Nintendo's long line of performance has granted it a solid first place among the different Publishers. +We saw how despite Nintendo's dominance, the PS2 manages to out perform it, reaching the top as the best console to date considering its success and score. +We were able to see how through different kinds of ML analysis, the opinion of the Critics is very far from resembling the opinion of the Users when it comes to videogames.

Thank you dear reviewer for checking my submission. Once again every result provided here is of course up to debate. If by any reason you do not consider my code to be up to expectations, I will appreaciate your comments and keep them in mind for the future.

Thank you for your time and best of lucks for you too on the Capstone course, may we all make the best of it!