Final Project: Video Games Analysis

Puttisan “Ninja” Mukneam

05/13/2022

```
#Loading the Libraries
library(tidyverse)
library(lubridate)
library(modelr)
library(ggplot2)
library(partykit)
options(na.action = na.warn)
```

In this final project, we are looking at various relationships between certain aspects of video games, such as their popularity defined by an average number of players during a specific period, genre, developers, global sales, critic scores, all-time player reviews, and a number of Twitch’s viewers. Before diving into more details, we should familiarize ourselves with some of the key terms presented here:

- Steam is a video game digital distribution service provided by Valve Corporation. It is an ideal “Amazon.com” not exclusively for a gamer but for anyone who thrives in digital industries since non-video game products are being distributed as well.
- With Steam being an open platform, any customer or gamer could leave their reviews of a product by giving them thump up or thump down accompanied by their written reviews. Then, the system will accumulate those scores and calculate a percentage of overall thumps ups as a summary.
- Global sales were gathered by summing the number of copies sold of a game from every region, namely North America, Europe, Japan, and other regions.
- Critic score is a score given by authentic video game review companies or organizations. These related groups would get their hands on a game before its official release date to produce their review.
- Lastly, Twitch is a video live streaming service that focuses mainly on video game live streaming, including broadcasts of e-sports competitions. Additionally, there are also other categories such as “in-real-life” content, music, events, and more being streamed as well.

Through out this project, gradually, we are trying to establish some understanding between theses variables by asking these 4 questions:

- what is/are the most popular genres of video games from 2012 to 2020, in term of average number of players through out the years; And how genres have changed from 2012 to 2020?
- Are there any relationship between video game’s critic score and all time player review as of June, 2021?
- Are there any relationship between video game’s number of Twitch’s viewer and its global sales as of 2020?
- Can we have a model predicting video’s global sales using its genre, developer, and critic score?

Ultimately, throughout the almost a decade long, Free to Play, which is usually followed by Action and Strategy games, had been dominated, standing at the top 1 to 3 for almost all of the year, except for 2018, which was a year of Massively Multiplayer, during which the newly popular genre of “Battle Royal” was born. Furthermore, despite no apparent relationship between the number of Twitch viewers and its global sales for most games, some hinted at non-linear growing relationships between video game critic scores and all-time player reviews. Lastly, we did not seem to be successful in having a model to predict a video game’s global sales using its genre, developer, and critic score. When we are looking at the difference of 100,000 copies sold between the actual data and our model, it seems to achieve merely 15% accuracy. Moreover,there were some patterns in some of the plots when looking at the residuals hinting that this is also not a good model.

Next looking at our data source, we have:

1) 80000 Steam Games DataSet ["steam_data_1"]

- Public URL: https://www.kaggle.com/datasets/deepann/80000-steam-games-dataset?select=steam_data.csv
- contains various information of games that was scrapable through Steam API link using (https://store.steampowered.com/appreviewhistogram/945360) and (https://store.steampowered.com/appreviews/945360?json=1)

2) Steam Store Games (Clean dataset) (May, 2019) ["steam_data_2"]

- Public URL: https://www.kaggle.com/datasets/nikdavis/steam-store-games?select=steam.csv
- contains various information of games that was scrapable through Steam Store and SteamSpy APIs using Python Libraries

3) Popularity of games on Steam ["steam_game_popularity"]

- Public URL: https://www.kaggle.com/datasets/michau96/popularity-of-games-on-steam
- contains various information of games such as average number of player at a certain time, its peak, and month-to-month difference in average. The data was obtained from https://steamcharts.com/ using webscraping through "rvesrt" package in R and Google Chrome extention called "SelectorGadet"

4) Top games on Twitch 2016 - 2021 ["twitch_game_popularity"]

- Public URL: https://www.kaggle.com/datasets/rankirsh/evolution-of-top-games-on-twitch?select=Twitch_game_data.csv
- contains various information of top 200 games on Twitch.tv such as average viewer, peak, and total hours watched per month from 2016 to February 2021. The data was obtained from a twitch analytics and statistics site https://sullygnome.com/

5) Video Games Dataset ["video_game_sales_1"]

- Public URL: https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2019/2019-07-30/video_games.csv
- contains various information of games including metascore (critic score). The data was obtained from Steam Spy scraped and provided by Liza Wood and modified by jthomasmock in his tidytuesday repository.

6) Video Game Sales ["video_game_sales_2"]

- Public URL: https://data.world/sumitrock/video-games-sales
- contains various information of games with sales more than 100,000 copies including global sales and critic score. The data was obtained from https://www.vgchartz.com/ by scraping using Python library "BeautifulSoup."

Even though these data sets contributed to establishing some understanding of various aspects of video games, we may notice that these data sets have never come from an official report by the publisher or developer that made or published the game since, most of the time, they were likely to be confidential or concealed within the companies. Indeed, for some data sets, such as those that came from the Steam database that is publicly available to obtain, the reviewers of products, for example, may not notice or ever give any fully aware consent to use their reviews on research or academic purposes. Although in this final project, for the most part, we are interested in the numbers related to the reviews, while the whole review section itself contains the written text of individuals as well, this may post some policies and ethic-related questions. Furthermore, some variables could be biased such as critic scores and players' reviews. There is no international agreement on the rubric that calculates the scores; it was subjective in most cases. Regarding, the analysis, the data set were never completed or deemed perfect. The analysis and the data sets do not represent the whole video games industry. Throughout the analysis, the data might have been deviated from the cleaning process by the uncertainty of joining and other complexity. Therefore, if someone uses these data sets or analyses in a misleading way to extract false information that these data are not capable of, there will be a wide range of repercussions in the long run inevitably. Moreover, since some of the information was real-time base, further in the future, their accuracy and relevancy may not remain the same as today. The misleading usage may also result negatively on related individuals. Lastly, the transparency issue also comes into our consideration. As our model involves more variable and becomes more and more complicated, the algorithm we used may exceed our perception to comprehensibly explains the process, which potentially raises transparency question whenever negative connotations were to be discovered.

```
#importing data
steam_data_1 <- read_csv("steam_data.csv") #w/o game's genre
steam_data_2 <- read_csv("steam.csv") # w game's genre
steam_game_popularity <- read_csv("SteamCharts.csv")
twitch_game_popularity <- read_csv("Twitch_game_data.csv")
video_game_sales_1 <- read_csv("video_gamesgit.csv")
video_game_sales_2 <- read_csv("Video_Games (2).csv")
```

Question 1: what is/are the most popular genres of video games from 2012 to 2020, in term of average number of players through out the years; And how genres have changed from 2012 to 2020?

a. **Cleaning relevant data sets**

```
#Cleaning dates
steam_data_1_cleaned_1 <- steam_data_1 %>%
  #filtering out duplicate rows with same names and same developers
  distinct(name, developer, .keep_all = TRUE) %>%
  select(date, name, developer) %>%
  filter(!is.na(name),
         !name == "-",
         !date == "-",
         #filtering out non-date value
         !str_detect(date, "\\w{3}", negate = TRUE)) %>%
  rename(game_name = name,
         release_date = date) %>%
  mutate(release_date = parse_date_time(release_date, orders = c("%b %d, %Y",
                                                                  "%d %b, %Y",
                                                                  "%m/%d/%Y",
                                                                  "%Y/%m/%d",
                                                                  "%Y-%m-%d",
                                                                  "%d.%m.%Y",
                                                                  "%m.%d.%Y"))) %>%
  filter(!is.na(release_date))
```

```
## Warning: 2815 failed to parse.
```

```
#Cleaning developer
steam_data_1_cleaned_2 <- steam_data_1_cleaned_1 %>%
  mutate(temp_dev = str_split(developer,
                              ", (?!L(?i)t(?i)d(?i).?|L(?i)C(?i)C(?i).?|I(?i)n(?i)c(?i).?|L(?i).?L(?i).?C(?i)
  # From this line, we are separating each developer into multiple columns
  unnest_legacy(temp_dev) %>%
  group_by(developer, game_name) %>%
  mutate(temp_id = row_number()) %>%
  pivot_wider(names_from = temp_id,
              values_from = temp_dev,
              names_prefix = "developer_") %>%
  ungroup() %>%
  select(-developer)

#Cleaning genres
steam_data_2_cleaned_1 <- steam_data_2 %>%
  #filtering out duplicate rows with same names and same developers
  distinct(name, developer,.keep_all = TRUE) %>%
  select(name, genres, developer) %>%
  #I do not think that Early Access counts as game genre.
  #So we are excluding them out here
  filter(genres != "Early Access") %>%
  rename(game_name = name) %>%
  mutate(genres = ifelse(str_detect(genres, ";Early Access"),
                         str_remove(genres, ";Early Access"),
                         str_remove(genres, "Early Access;"))) %>%
  mutate(temp_genres = str_split(genres, ";")) %>%
  # From this line, we are separating each genre into multiple columns
  unnest_legacy(temp_genres) %>%
  group_by(developer, game_name) %>%
  mutate(temp_id = row_number()) %>%
  pivot_wider(names_from = temp_id,
              values_from = temp_genres,
              names_prefix = "genre_") %>%
  ungroup() %>%
  select(- genres)

#Cleaning developer
steam_data_2_cleaned_2 <- steam_data_2_cleaned_1 %>%
  mutate(temp_dev = ifelse(str_detect(developer, ";"),
                           str_split(developer, ";"),
                           str_split(developer,
                                     ", (?!L(?i)t(?i)d(?i).?|L(?i)C(?i)C(?i).?|I(?i)n(?i)c(?i).?|L(?i).
  # From this line, we are separating each developer into multiple columns
  unnest_legacy(temp_dev) %>%
  group_by(developer, game_name) %>%
  mutate(temp_id = row_number()) %>%
  pivot_wider(names_from = temp_id,
              values_from = temp_dev,
              names_prefix = "developer_") %>%
  ungroup() %>%
  select(-developer)
```

b. **Joining tibbles**

```
#Preparing columns to join with steam_data_1
steam_data_2_cleaned_2_joinable <- steam_data_2_cleaned_2 %>%
  #low-casing and removing all special characters
  mutate(joined_game_name = str_replace_all(game_name, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name),
         #Using at most 2 developers as a joined column
         joined_dev = ifelse(is.na(developer_2),
                             developer_1,
                             str_c(developer_1, developer_2, sep = " ")),
         joined_dev = str_replace_all(joined_dev, "[^[:alnum:] ]", " "),
         joined_dev = str_to_lower(joined_dev)) %>%
  select(genre_1:genre_16, joined_game_name, joined_dev)

#Preparing columns to join with steam_data_2 and joining them
steam_data_1_with_genres <- steam_data_1_cleaned_2 %>%
  #preparing joined columns
  #low-casing and removing all special characters
  mutate(joined_game_name = str_replace_all(game_name, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name),
         joined_dev = ifelse(is.na(developer_2),
                             developer_1,
                             str_c(developer_1, developer_2, sep = " ")),
         joined_dev = str_replace_all(joined_dev, "[^[:alnum:] ]", " "),
         joined_dev = str_to_lower(joined_dev)) %>%
  select(release_date, game_name, joined_game_name, joined_dev) %>%
  filter(!is.na(joined_dev)) %>%
  #joining using inner_join so that we can work with games that have their
  #genres recorded
  inner_join(steam_data_2_cleaned_2_joinable,
                       by = c("joined_game_name", "joined_dev"))

#Cleaning and Preparing columns to join with steam_data_1_with_genres
steam_game_pop_joinable <- steam_game_popularity %>%
  mutate(year = as.integer(year),
         #low-casing and removing all special characters
         joined_game_name = str_replace_all(gamename, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name)) %>%
  select(gamename, year, month, avg, joined_game_name) %>%
  #Excluding year 2021 because there are only recordings from Jan and Feb
  filter(!year == 2021)

#Preparing columns to join with steam_game_pop_joinable
steam_data_1_with_1_genre <- steam_data_1_with_genres %>%
  #Pivot games' genres into rows so that we look at every genre instead of
  #only 1 genre and with a bunch of NAs, which will produce greater complexity
  pivot_longer(genre_1:genre_16, names_to = "genre", values_drop_na = TRUE) %>%
  select(- genre) %>%
  rename("genres" = value)

#joining steam_game_pop_joinable and steam_data_1_with_1_genre to achieve a
#tibble to answer question 1
q1 <- inner_join(steam_game_pop_joinable, steam_data_1_with_1_genre,
                 by = "joined_game_name") %>%
  select(-joined_game_name, - gamename)

q1 %>%
  filter(str_detect(genres, "Action"), year == 2014) %>%
  arrange(desc(avg))
```

```
## # A tibble: 2,504 x 7
##     year month          avg release_date        game_name joined_dev genres
##    <int> <chr>        <dbl> <dttm>              <chr>     <chr>      <chr>
##  1  2014 July      537019. 2013-07-09 00:00:00 Dota 2    valve      Action
##  2  2014 November  528790. 2013-07-09 00:00:00 Dota 2    valve      Action
##  3  2014 December  523940. 2013-07-09 00:00:00 Dota 2    valve      Action
##  4  2014 June      513783. 2013-07-09 00:00:00 Dota 2    valve      Action
##  5  2014 October   495693. 2013-07-09 00:00:00 Dota 2    valve      Action
##  6  2014 August    490884. 2013-07-09 00:00:00 Dota 2    valve      Action
##  7  2014 May       482387. 2013-07-09 00:00:00 Dota 2    valve      Action
##  8  2014 September 477998. 2013-07-09 00:00:00 Dota 2    valve      Action
##  9  2014 April     421710. 2013-07-09 00:00:00 Dota 2    valve      Action
## 10  2014 February  421114. 2013-07-09 00:00:00 Dota 2    valve      Action
## # ... with 2,494 more rows
```

c. Answering question

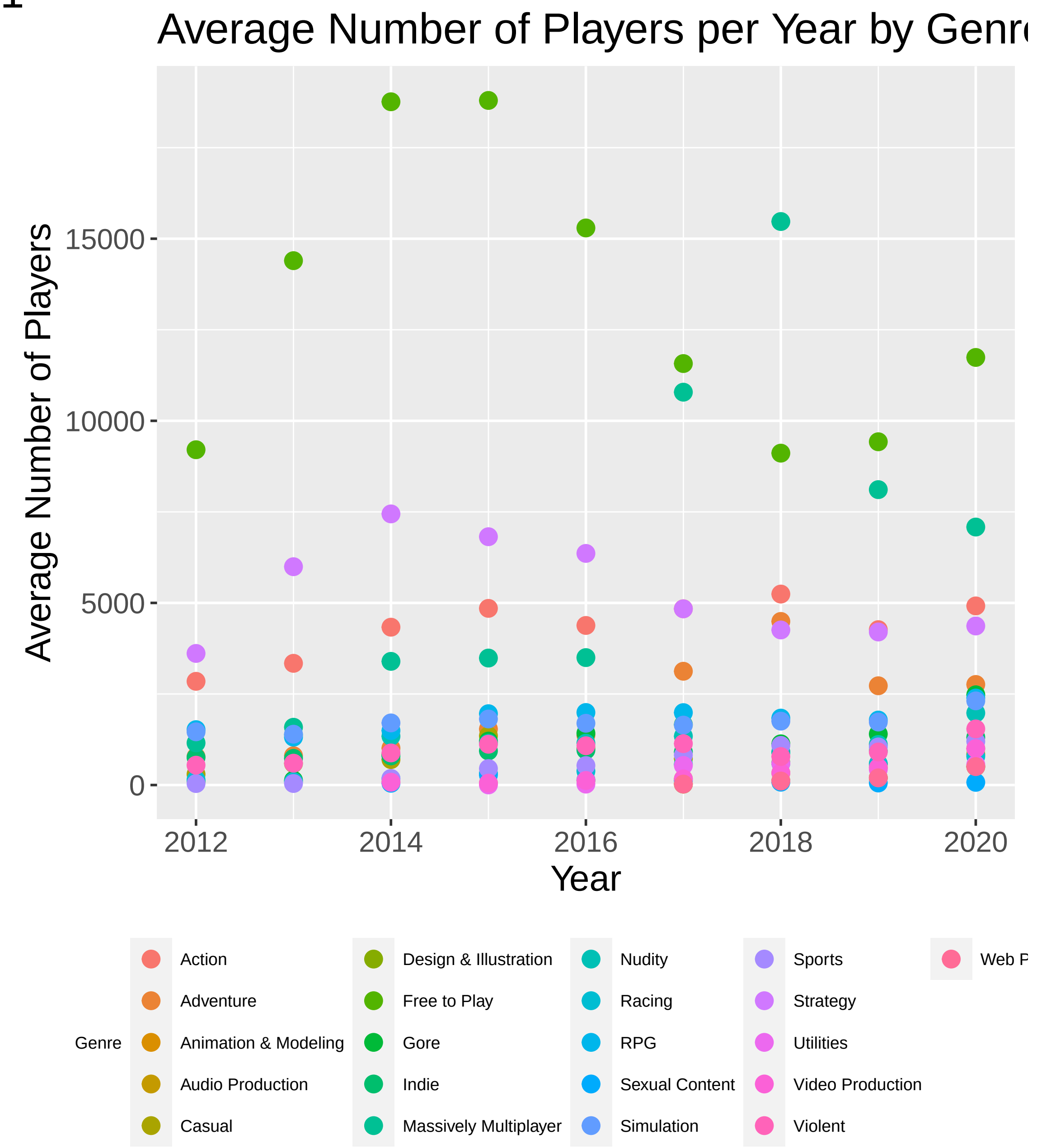
```
#average by year
#Collapsing all the months of that year by taking their average
 q1_avg_by_year <- q1 %>%
   group_by(year, genres) %>%
   summarise(avg_by_genre = mean(avg)) %>%
   ungroup()
```

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

After we have collapsed the observations to show only the top one genre of a particular year, we then do a scatter plot of the year vs. an average number of players across that year and group the color by genre.

```
q1_avg_by_year %>%
  ggplot() +
  geom_point(aes(x = year,
                 y = avg_by_genre,
                 color = genres),
             size = 3) +
  theme(text = element_text(size = 15),
        legend.position = "bottom",
        legend.title = element_text(size = 7),
        legend.text = element_text(size = 7)) +
  labs(title = "Average Number of Players per Year by Genre",
       colour = "Genre",
       tag = "Plot 1") +
  xlab("Year") +
  ylab("Average Number of Players")
```

Plot 1



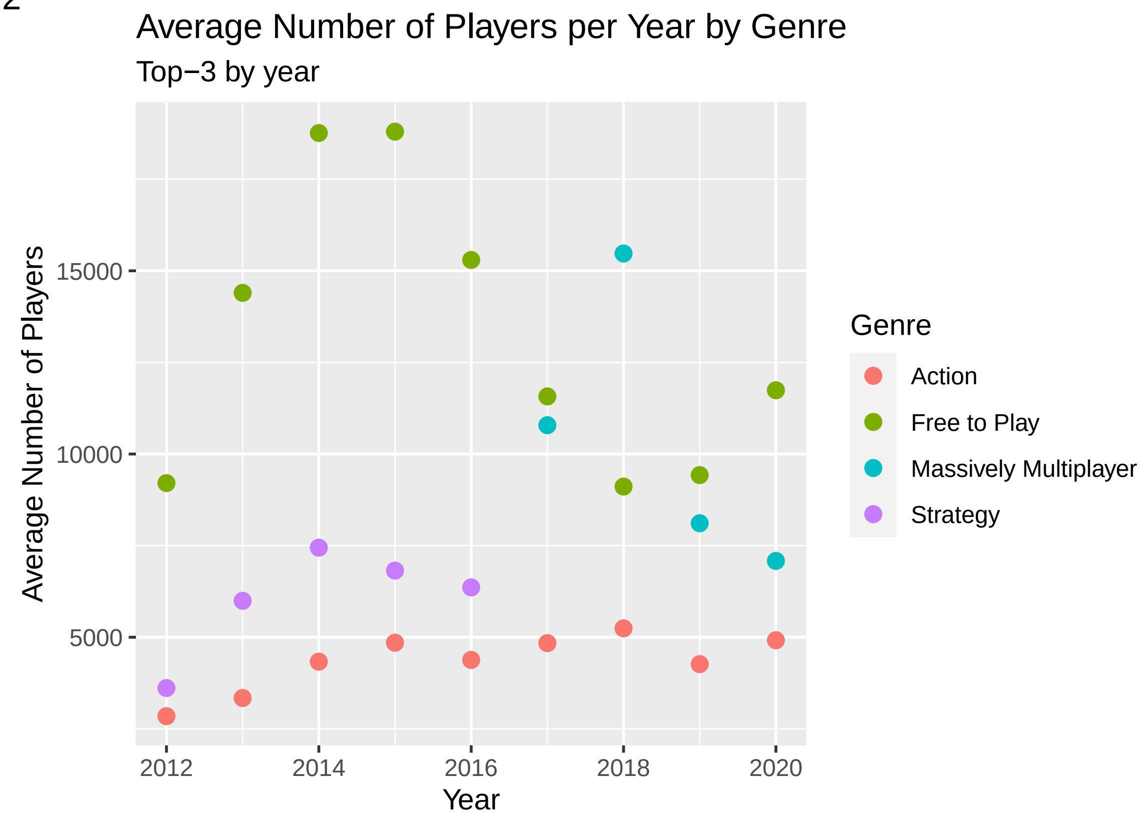
```
#table Top-1 each year
q1_avg_by_year %>%
  group_by(year) %>%
  filter(avg_by_genre == max(avg_by_genre))
```

```
## # A tibble: 9 x 3
## # Groups:   year [9]
##     year genres                  avg_by_genre
##    <int> <chr>                          <dbl>
## 1  2012 Free to Play                    9206.
## 2  2013 Free to Play                   14398.
## 3  2014 Free to Play                 18760.
## 4  2015 Free to Play                 18797.
## 5  2016 Free to Play                 15296.
## 6  2017 Free to Play                 11572.
## 7  2018 Massively Multiplayer       15472.
## 8  2019 Free to Play                  9425.
## 9  2020 Free to Play                 11740.
```

Looking at the plot and table above, "A Free to Play" genre has dominated other genres throughout the last decade, by a large margin. However, in 2018, Massively Multiplayer suddenly won over Free to Play by certain amount. We can observe further by looking into the top-3 genre of each year, which is represented by the plot below.

```
#Top-3
q1_avg_by_year %>%
  arrange(desc(avg_by_genre)) %>%
  group_by(year) %>%
  slice(1:3) %>%
  ggplot() +
  geom_point(aes(x = year,
                 y = avg_by_genre,
                 color = genres),
             size = 2.5) +
  labs(title = "Average Number of Players per Year by Genre",
       subtitle = "Top-3 by year",
       colour = "Genre",
       tag = "Plot 2") +
  xlab("Year") +
  ylab("Average Number of Players")
```

Plot 2



Firstly, we can see that there is not much variety in genres trying to compete among themselves. We can only see here in the top-3, a maximum of 2 genres are competing between themselves. Next, by observing the plot above, if we ignore the "Free to Play" genre and look at the second place, the first half of the last decade, 2012 to 2016, would be dominated by the Strategy genre, and the second half, 2017 onward, Massively Multiplayer would take the lead. When looking deeper into what games represent these genres, "Free to Play" and Strategy are occupied by two big names, such as Dota 2 and Counter-Strike. These games have maintained their popularity since their release date and have been loved by many people. On the other hand, a new "Battle Royal" genre was born. The beginning of PUBG and Fortnite marked the changes in the gaming genre after that. Consequently, in 2018, "Battle Royal" solely drove Massively Multiplayer to the top of the chart. This occurrence was phenomenal. Suddenly, everyone talked about it; even personal and celebrities participated in these "Battle Royal" games. Thus, there was no suspicion that this new genre would dominate the Steam chart in no time. Lastly, another genre that is also as popular as the top-3 and was able to maintain its ranking is the Action genre, which there is no surprise either since many popular games take Action as its sub-genre.

Question 2: Are there any relationship between video game's critic score and all time player review as of June, 2021?

a. **Cleaning relevant data sets**

```
steam_data_1_cleaned_3 <- steam_data_1 %>%
  distinct(name, .keep_all = TRUE) %>% #making names unique
  select(name, developer, user_reviews, all_reviews) %>%
  #filtering out "NAs"
  filter(!is.na(name),
         !name == "-",
```

```r
         !is.na(developer)) %>%
  rename(game_name = name,
         reviews_all_time = all_reviews,
         reviews_30_days = user_reviews)

#Cleaning developer
steam_data_1_cleaned_4 <- steam_data_1_cleaned_3 %>%
  mutate(d1 = str_split(developer,
                        ", (?!L(?i)t(?i)d(?i).?|L(?i)C(?i)C(?i).?|I(?i)n(?i)c(?i).?|L(?i).?L(?i).?C(?i)
  # From this line, we are separating each developer into multiple columns
  unnest_legacy(d1) %>% #faster version of unnest_longer
  group_by(game_name) %>%
  mutate(temp_id = row_number()) %>%
  pivot_wider(names_from = temp_id,
              values_from = d1,
              names_prefix = "developer_") %>%
  select(-developer) %>%
  ungroup()

#Cleaning all_reviews, we are interested in all time reviews only
steam_data_1_cleaned_5 <- steam_data_1_cleaned_4 %>%
  #filtering out non-review value
  filter(str_detect(reviews_30_days, "%")) %>%
  #filtering out non-review value in reviews_all_time
  mutate(reviews_all_time_pct = ifelse(!str_detect(reviews_all_time, "%"),
                                       NA,
                                       reviews_all_time),
         #it is the way the data been collected that if a game doesn't have
         #last 30 days reviews, their all time review will be in reviews_30_days
         #column
         reviews_all_time_pct = ifelse(is.na(reviews_all_time_pct),
                                       reviews_30_days,
                                       reviews_all_time_pct)) %>%
  select(-reviews_30_days, -reviews_all_time) %>%
  #Extracting the percentage of all time reviews
  mutate(reviews_all_time_pct = str_extract(reviews_all_time_pct,
                                            "[:digit:]+(?=%)"),
         reviews_all_time_pct = as.double(reviews_all_time_pct))

#Cleaning video game sales 1; gathering critic score
video_game_sales_1_cleaned_1_joinable<- video_game_sales_1 %>%
  select(game, metascore, developer, release_date) %>%
  filter(!is.na(metascore)) %>%
  #Preparing the joining columns with video game sales 2
  #low-casing and removing all special characters
  mutate(joined_game_name = str_replace_all(game, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name),
         temp_dev = str_split(developer,
                        ", (?!L(?i)t(?i)d(?i).?|L(?i)C(?i)C(?i).?|I(?i)n(?i)c(?i).?|L(?i).?L(?i).?C(?i)
  #From this line, we are separating each developer into multiple columns
  unnest_legacy(temp_dev) %>%
  group_by(release_date,game) %>%
  mutate(temp_id = row_number()) %>%
  pivot_wider(names_from = temp_id,
              values_from = temp_dev,
              names_prefix = "developer_") %>%
  ungroup() %>%
  select(-developer) %>%
  #Preparing the joining columns with video game sales 2 using 2 devs
  mutate(joined_dev = ifelse(is.na(developer_2),
                               developer_1,
                               str_c(developer_1, developer_2, sep = " ")),
         #low-casing and removing all special characters
         joined_dev = str_replace_all(joined_dev, "[^[:alnum:] ]", " "),
         joined_dev = str_to_lower(joined_dev)) %>%
  select(game, metascore, joined_game_name, joined_dev) %>%
  filter(!is.na(joined_dev))

#Cleaning video game sales 2; gathering more critic score
video_game_sales_2_cleaned_1_joinable <- video_game_sales_2 %>%
  select(Name, Critic_Score, Developer, NA_Sales) %>%
  filter(!is.na(Critic_Score)) %>%
  #Preparing the joining columns with video game sales 1
  #The rest transforming data the same way as video game sales 1
  mutate(joined_game_name = str_replace_all(Name, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name),
         temp_dev = str_split(Developer,
                        ", (?!L(?i)t(?i)d(?i).?|L(?i)C(?i)C(?i).?|I(?i)n(?i)c(?i).?|L(?i).?L(?i).?C(?i)
  unnest_legacy(temp_dev) %>%
  group_by(Developer, Name, NA_Sales) %>%
  mutate(temp_id = row_number()) %>%
  pivot_wider(names_from = temp_id,
              values_from = temp_dev,
              names_prefix = "developer_") %>%
  ungroup() %>%
  select(-Developer) %>%
  mutate(joined_dev = ifelse(is.na(developer_2),
                               developer_1,
                               str_c(developer_1, developer_2, sep = " ")),
         joined_dev = str_replace_all(joined_dev, "[^[:alnum:] ]", " "),
         joined_dev = str_to_lower(joined_dev)) %>%
  select(Name, Critic_Score, joined_game_name, joined_dev) %>%
  filter(!is.na(joined_dev))
```

b. **Joining tibbles**

```
#This is full join because we want to keep games/w their critics score from
#both data set.
full_join_video_game_sales <- full_join(video_game_sales_2_cleaned_1_joinable,
                                        video_game_sales_1_cleaned_1_joinable,
                                        by = c("joined_game_name", "joined_dev")) %>%
  #Keeps those game who have at least one of the two critic score
  filter(!is.na(Critic_Score) | !is.na(metascore))

#After gathering more scores, if a game has 2 critics scores, we then take their
#average
full_join_video_game_sales_critic <- full_join_video_game_sales %>%
```

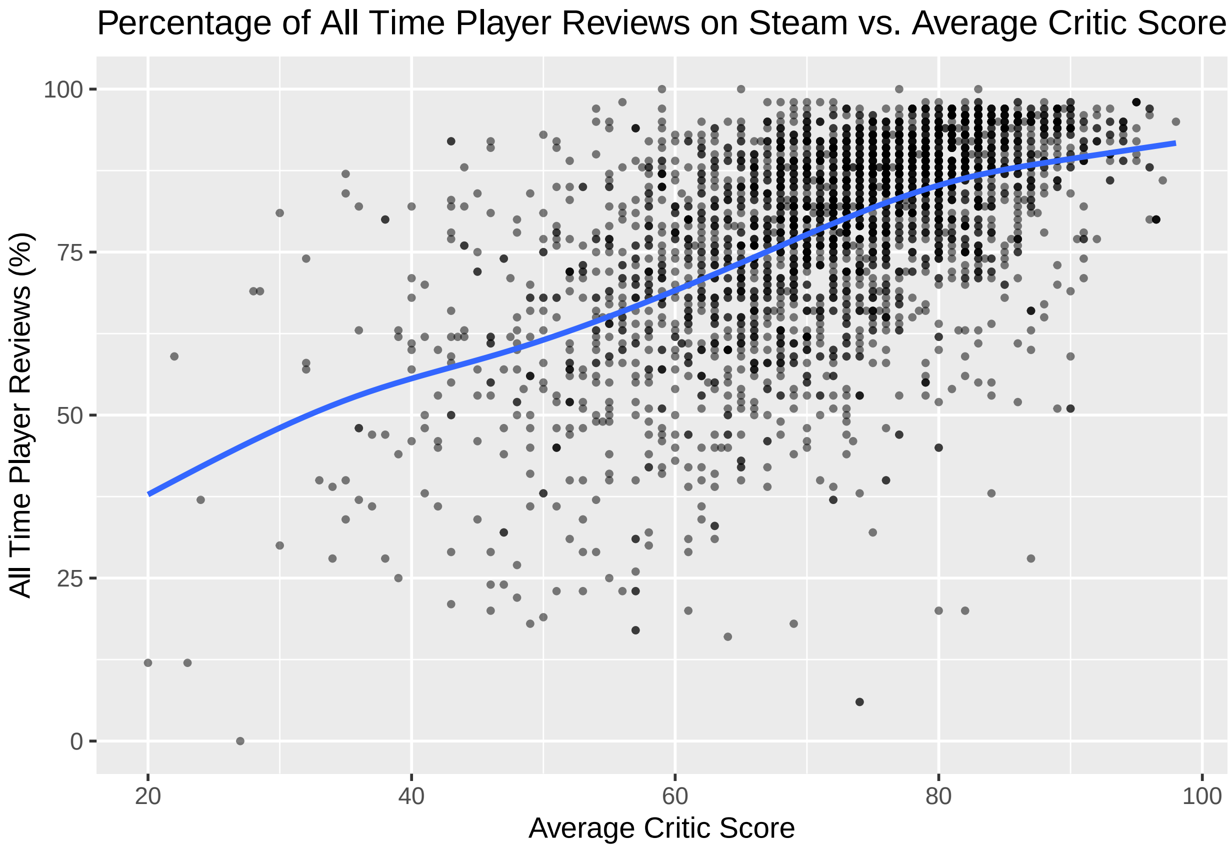
```r
  mutate(critic_score = rowMeans(full_join_video_game_sales[c('Critic_Score',
                                                               'metascore')],
                                 na.rm = TRUE)) %>%
  select(Name, critic_score, joined_game_name, joined_dev)

#Preparing player reviews to join with critic score
#Going through the same process as previous preparing for joining process
steam_data1_w_review_joinable <- steam_data_1_cleaned_5 %>%
  mutate(joined_game_name = str_replace_all(game_name, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name),
         joined_dev = ifelse(is.na(developer_2),
                             developer_1,
                             str_c(developer_1, developer_2, sep = " ")),
         joined_dev = str_replace_all(joined_dev, "[^[:alnum:] ]", " "),
         joined_dev = str_to_lower(joined_dev)) %>%
  select(game_name, joined_game_name, joined_dev, reviews_all_time_pct) %>%
  filter(!is.na(joined_dev))

#Joining critic score with player review
#Notice that we do not join using "joined_dev" since it produce less observation
#Because of the spelling of developer in these two data set.
#This attemptt was meant for keeping game_name that has the scores available as
#much as possible
joined_critic_all_review <- left_join(full_join_video_game_sales_critic,
                                      steam_data1_w_review_joinable,
                                      by = c("joined_game_name"))  %>%
  filter(!is.na(reviews_all_time_pct)) %>%
  #Keeping game name listed in video_game_sales (tend to be more accurate)
  select(game_name, critic_score, reviews_all_time_pct) %>%
  rename("avg_critic_score" = critic_score)

q2 <- joined_critic_all_review
q2
```

```
## # A tibble: 3,105 x 3
##    game_name                       avg_critic_score reviews_all_time_pct
##    <chr>                                      <dbl>                <dbl>
##  1 Grand Theft Auto V                          96.5                   80
##  2 Grand Theft Auto: San Andreas               95                     90
##  3 Grand Theft Auto V                          96.5                   80
##  4 Grand Theft Auto: Vice City                 95                     92
##  5 Grand Theft Auto III                        97                     86
##  6 Grand Theft Auto V                          96.5                   80
##  7 Halo 3                                      94                     89
##  8 Halo: Reach                                 91                     77
##  9 FINAL FANTASY VII                           92                     92
## 10 FINAL FANTASY VIII                          90                     88
## # ... with 3,095 more rows
```

c. **Answering question**

```r
#plotting
q2 %>%
  ggplot() +
  geom_point(aes(x = avg_critic_score,
               y = reviews_all_time_pct),
             size = 0.8,
             alpha = 0.5) +
  geom_smooth(aes(x = avg_critic_score,
                  y = reviews_all_time_pct),
              se = FALSE) +
  labs(title = "Percentage of All Time Player Reviews on Steam vs. Average Critic Score") +
  xlab("Average Critic Score") +
  ylab("All Time Player Reviews (%)")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Overall, observing the above plot shows a growing relationship between all-time player reviews and average critic Scores, as it seems to be a natural phenomenon. Furthermore, we can look at the outlier by observing the table below.

```
q2 %>%
  filter(avg_critic_score > 60 & reviews_all_time_pct < 30)
```

```
## # A tibble: 9 x 3
##   game_name                                 avg_critic_score reviews_all_time_pct
##   <chr>                                                <dbl>                <dbl>
## 1 The Darkness                                            80                   20
## 2 The Darkness                                            82                   20
## 3 Vietcong                                                74                    6
## 4 Command & Conquer 4: Tiberian Twilight                  64                   16
## 5 Razor2: Hidden Skies                                    61                   20
## 6 Dream Pinball 3D                                         61                    29
## 7 Basketball Pro Management 2015                            69                    18
## 8 NBA 2K18                                                  87                    28
## 9 Vietcong                                                  74                     6
```

From the table above, a few of these names might have surprised some audiences. For example, many sports fans and gamers have loved a triple-A NBA 2K series, but this time, NBA 2K18 got such a controversial review by its players. Looking deeper into the written reviews, it seems as if the model might have changed to "Pay to Win," which has always been a controversial shift of a game in this industry. However, this is not the first time the "Pay to Win" controversy has happened. The wide heard example is the controversy involved Electronic Arts (EA Games) due to their changed strategy to the "Pay to Win" approach, which got much hatred from players but seemed to be overlooked in many critics' reviews.

Question 3: Are there any relationship between video game's number of Twitch's viewer and its global sales as of 2020?

a. Cleaning relevant data sets

```r
#Cleaning twitch game popularity and preparing column to join with game sales
twitch_game_popularity_joined <- twitch_game_popularity %>%
  #Collapsing Twitch's viewer by averaging viewer by game
  group_by(Game) %>%
  summarise(avg_all_time_viwers = mean(Avg_viewers)) %>%
  #This time only using game name as a joined column
  #low-casing and removing all special characters
  mutate(joined_game_name = str_replace_all(Game, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name)) %>%
  rename("game_name" = Game) %>%
  ungroup()

#Cleaning video game sales and preparing column to join with Twitch's viewer
video_game_sales_2_cleaned_2_joinable <- video_game_sales_2 %>%
  group_by(Name, Developer) %>%
  #Collapsing Global sales by summing across all platform sales
  summarise(sum_global_sales = sum(Global_Sales)) %>%
  #low-casing and removing all special characters
  mutate(joined_game_name = str_replace_all(Name, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name),
         #Adjusting sales from a million unit to 100k unit for clearer plot later
         sum_global_sales = sum_global_sales * 10) %>% # 100k unit
  select(Name, joined_game_name, sum_global_sales) %>%
  rename("game_name" = Name) %>%
  ungroup()
```

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

b. Joining tibbles

```r
#joining using inner_join because we want to work with games that have both
#Twitch's viewer data and its sales data available
joined_avg_viwer_global_sales <- inner_join(twitch_game_popularity_joined,
                                            video_game_sales_2_cleaned_2_joinable,
                                            by = "game_name") %>%
  select(game_name, avg_all_time_viwers, sum_global_sales)
```

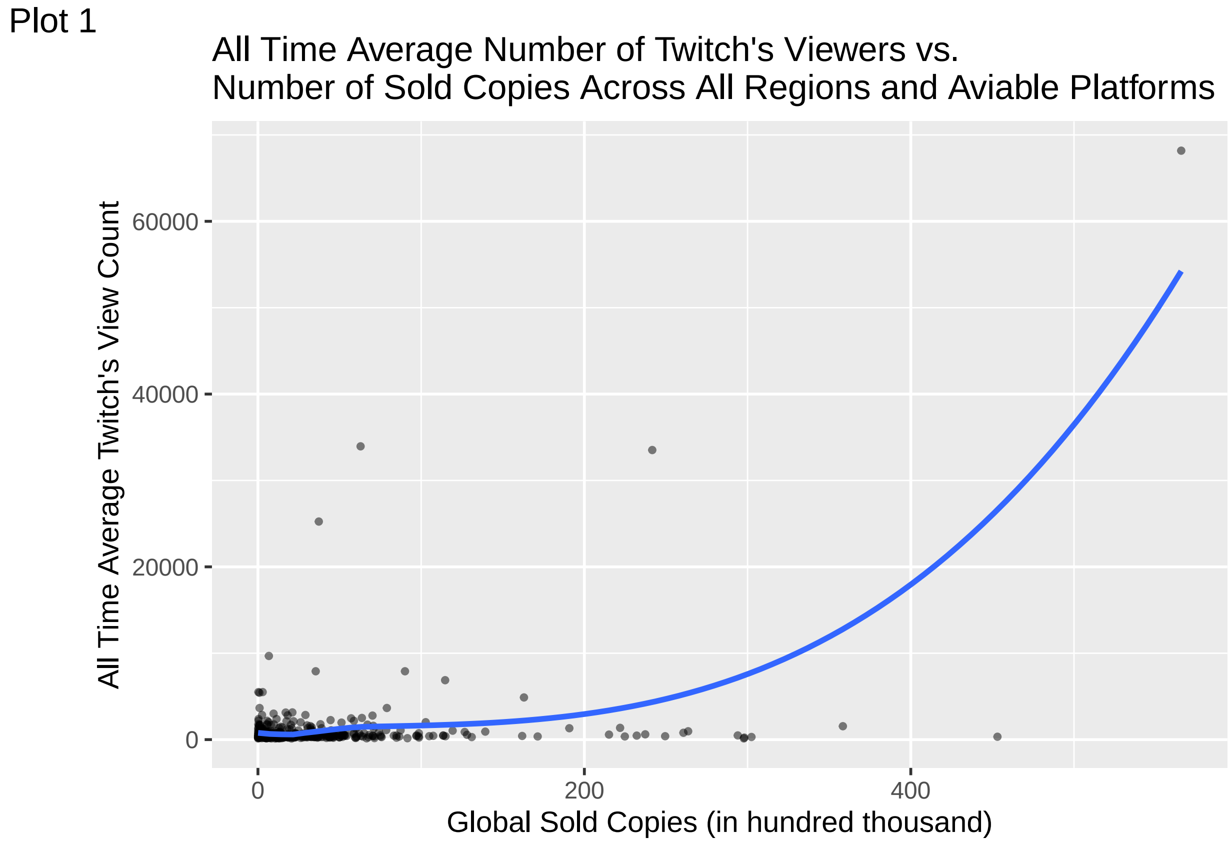
```
q3 <- joined_avg_viwer_global_sales
q3
```

```
## # A tibble: 472 x 3
##    game_name                  avg_all_time_viwers sum_global_sales
##    <chr>                                    <dbl>            <dbl>
##  1 7 Days to Die                            1190.              1.9
##  2 Alien: Isolation                          605.             23.9
##  3 Animal Crossing: New Leaf                 163              91.6
##  4 APB Reloaded                              399               0.4
##  5 Ape Escape                                313              16.3
##  6 Ape Escape                                313               0.4
##  7 Assassin's Creed II                       459             114. 
##  8 Assetto Corsa                            1090               0.8
##  9 Banjo-Kazooie                             515               0.1
## 10 Banjo-Kazooie                             515              36.5
## # ... with 462 more rows
```

c. Answering question

```
#plot 1: Overall
joined_avg_viwer_global_sales %>%
  ggplot() +
  geom_point(aes(x = sum_global_sales,
                 y = avg_all_time_viwers),
             size = 0.8,
             alpha = 0.5) +
  geom_smooth(aes(x = sum_global_sales,
                  y = avg_all_time_viwers),
              se = FALSE) +
  labs(title = "All Time Average Number of Twitch's Viewers vs. \nNumber of Sold Copies Across All Regi
       tag = "Plot 1") +
  ylab("All Time Average Twitch's View Count") +
  xlab("Global Sold Copies (in hundred thousand)")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

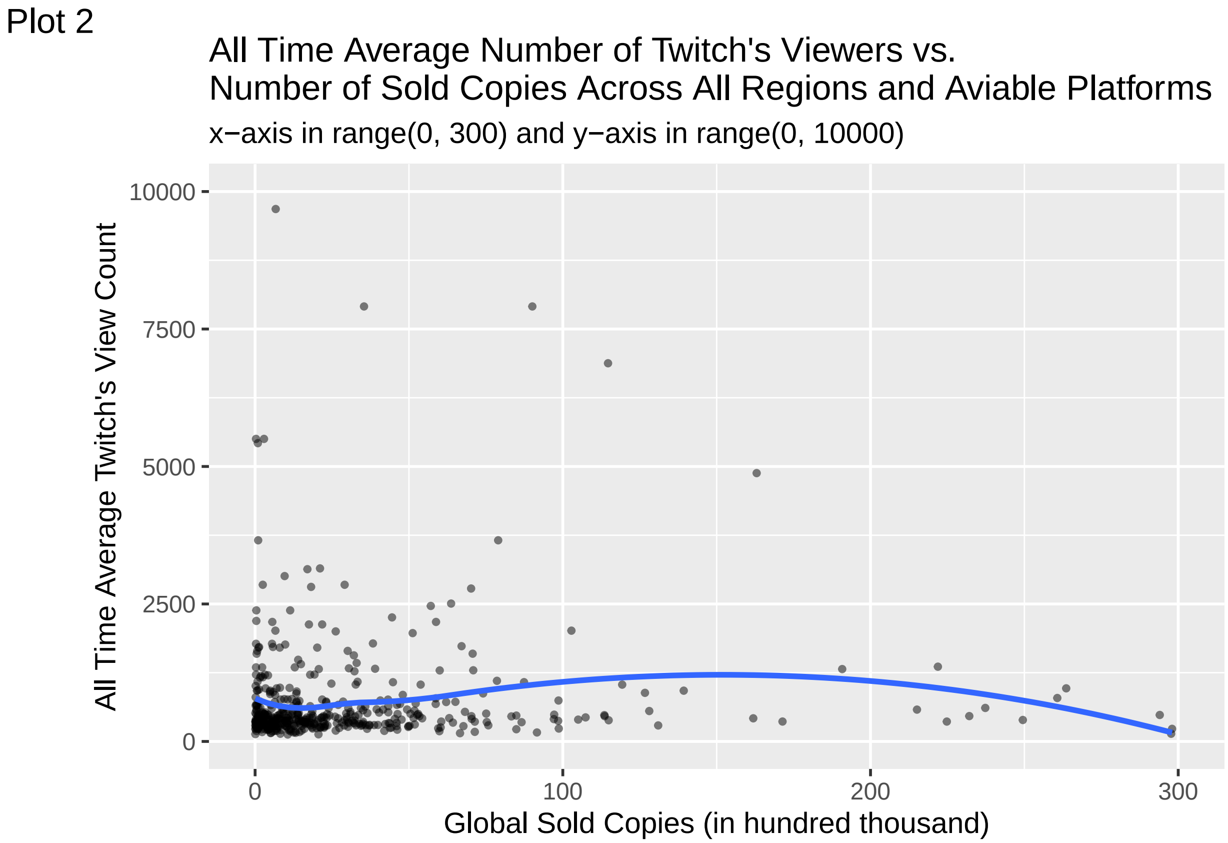


Looking at the above plot, there seems to be an increasing non-linear relationship between the two variables. However, this trend is only caused by the outlier and does not genuinely represent the whole data point. The outstanding game that held over the all-time average Twitch's view count of over 60,000 views and was sold over 50 million copies is Grand Theft Auto V, an action-adventure, open-world developed by Rockstar North, which, again, no other game could thoroughly beat this game, even though it has been released since 2013. Next, we would like to exclude the outlier and look at where most of the data points were.

```
#plot 2
joined_avg_viwer_global_sales %>%
  ggplot() +
  geom_point(aes(x = sum_global_sales,
                 y = avg_all_time_viwers),
             size = 0.8,
             alpha = 0.5) +
  geom_smooth(aes(x = sum_global_sales,
                  y = avg_all_time_viwers),
              se = FALSE) +
  labs(title = "All Time Average Number of Twitch's Viewers vs. \nNumber of Sold Copies Across All Regi
       subtitle = "x-axis in range(0, 300) and y-axis in range(0, 10000)",
       tag = "Plot 2") +
  ylab("All Time Average Twitch's View Count") +
  xlab("Global Sold Copies (in hundred thousand)") +
  xlim(0, 300) +
  ylim(0, 10000)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 7 rows containing non-finite values (stat_smooth).
## Warning: Removed 7 rows containing missing values (geom_point).
```



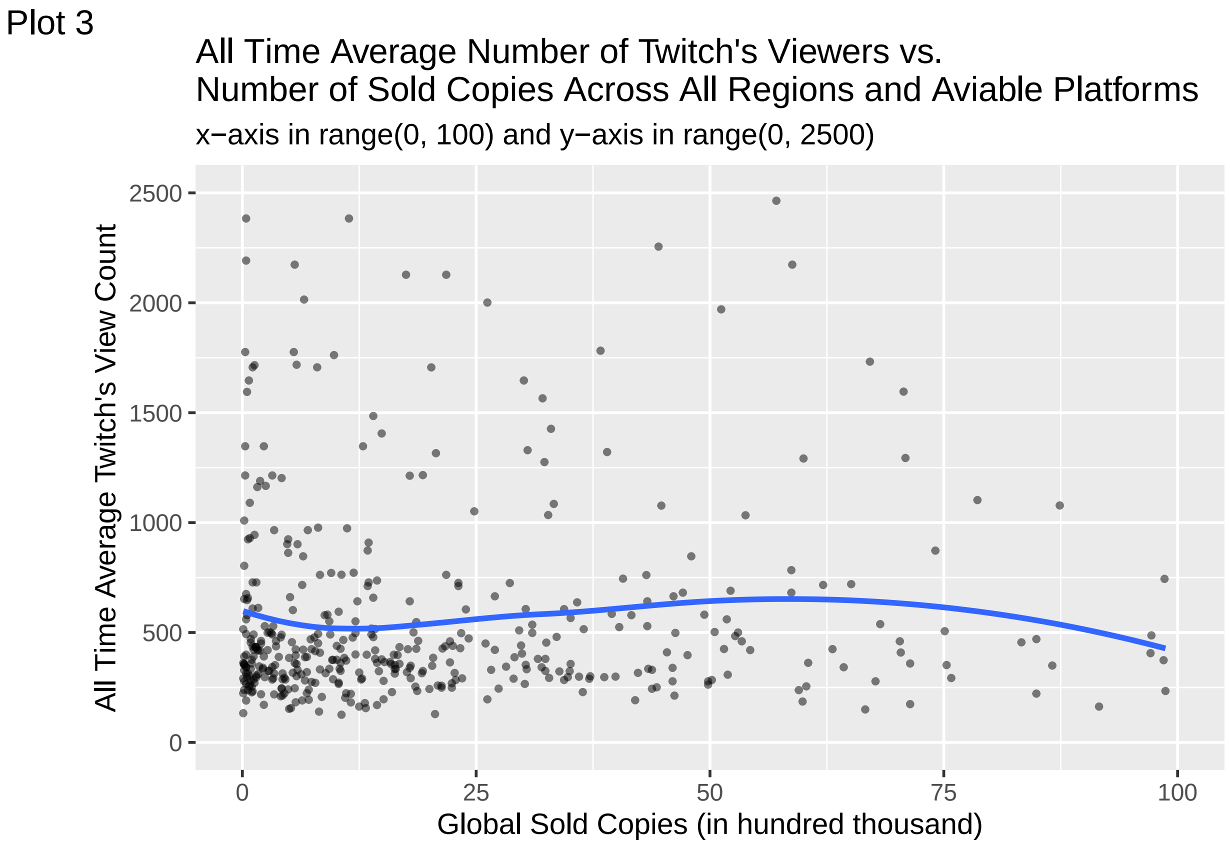
Here, there are no apparent patterns or relationships between the two variables by observing the second plot.

```
#plot 3
joined_avg_viwer_global_sales %>%
  ggplot() +
  geom_point(aes(x = sum_global_sales,
                 y = avg_all_time_viwers),
             size = 0.8,
             alpha = 0.5) +
  geom_smooth(aes(x = sum_global_sales,
                  y = avg_all_time_viwers),
              se = FALSE) +
  labs(title = "All Time Average Number of Twitch's Viewers vs. \nNumber of Sold Copies Across All Regi
       subtitle = "x-axis in range(0, 100) and y-axis in range(0, 2500)",
       tag = "Plot 3") +
  ylab("All Time Average Twitch's View Count") +
  xlab("Global Sold Copies (in hundred thousand)") +
  xlim(0, 100) +
  ylim(0, 2500)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 50 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 50 rows containing missing values (geom_point).
```



By restricting the boundary even smaller, they suggest no particular relationship or an increasing/decreasing linear pattern but only a partial glimpse of a non-linear relationship, which again does not contain much insightful interpretation. Thus, except for a few phenomenal games, there is no apparent relationship between global sales and the all-time average Twitch's view count.

Question 4: Can we have a model predicting video's global sales using its genre, developer, and critic score?

a. **Cleaning relevant data sets**

```
#Cleaning global sales 2 with the same process as in question 3
#But this time we add the first_developer column to use in modeling
video_game_sales_2_cleaned_3_joinable <- video_game_sales_2 %>%
  group_by(Name, Developer) %>%
  summarise(sum_global_sales = sum(Global_Sales)) %>%
  mutate(joined_game_name = str_replace_all(Name, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name),
         sum_global_sales = sum_global_sales * 10, #100k
         temp_dev = str_split(Developer,
                        ", (?!L(?i)t(?i)d(?i).?|L(?i)C(?i)C(?i).?|I(?i)n(?i)c(?i).?|L(?i).?L(?i).?C(?i)
  unnest_legacy(temp_dev) %>%
  group_by(Name, sum_global_sales) %>%
  mutate(temp_id = row_number()) %>%
  pivot_wider(names_from = temp_id,
              values_from = temp_dev,
              names_prefix = "developer_") %>%
  ungroup() %>%
  mutate(joined_dev = ifelse(is.na(developer_2),
                               developer_1,
                               str_c(developer_1, developer_2, sep = " ")),
         joined_dev = str_replace_all(joined_dev, "[^[:alnum:] ]", " "),
         joined_dev = str_to_lower(joined_dev),
         #Extracting each game's first developer to use in modeling
         first_developer = ifelse(is.na(developer_1),
                                  developer_2,
                                  developer_1)) %>%
  select(Name, joined_game_name, joined_dev, sum_global_sales, first_developer) %>%
  filter(!is.na(joined_dev))
```

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

```
#Preparing the steam_data_1 with each game's first gebre to use in modeling
steam_data1_joinable_first_genre <- steam_data_1_with_genres %>%
  select(game_name, joined_game_name, joined_dev, genre_1)

#Preparing Critic_score using full_join_video_game_sales_critic but only pulls
#out those that that the game name is present
full_join_video_game_sales_critic_joinable <- full_join_video_game_sales_critic %>%
  filter(!is.na(joined_game_name))
```

b. **Joining tibbles**

```
#joining all the data using inner_join because we would like to work with games
#that has all attribute present, namely: game_name, global_sales, genre, critic_score
#Notice that the first join didn't use joined_dev. This is the same reason that
#We want to produce as many observations as possible. Thus, we are avoiding
#the spelling of developer, which turned the whole thing into NA if not written
#the same way across all the data.
game_with_sales_genre_developer_cri_score <- inner_join(video_game_sales_2_cleaned_3_joinable,
                        steam_data1_joinable_first_genre,
                        by = "joined_game_name") %>%
  rename(joined_dev = joined_dev.x) %>%
  inner_join(full_join_video_game_sales_critic_joinable,
             by = c("joined_game_name", "joined_dev")) %>%
  unique() %>%
  select(game_name, sum_global_sales, first_developer, genre_1, critic_score)

q4 <- game_with_sales_genre_developer_cri_score
q4
```

```
## # A tibble: 702 x 5
##    game_name               sum_global_sales first_developer genre_1 critic_score
##    <chr>                              <dbl> <chr>           <chr>          <dbl>
##  1 15 Days                              0.1 DTP Entertainm~ Advent~           63
##  2 7 Days to Die                        1.9 The Fun Pimps ~ Nudity            45
##  3 7 Days to Die                        1.9 The Fun Pimps ~ Nudity            35
##  4 7 Wonders II                         3.1 MumboJumbo      Casual            60
##  5 7 Wonders of the Ancie~              1.3 Hot Lava Games  Casual            60
##  6 7 Wonders of the Ancie~              0.8 MumboJumbo      Casual            51
##  7 A Boy and His Blob                   2.1 WayForward      Advent~           80
##  8 A Vampyre Story                          0.2 Autumn Moon     Advent~           75
##  9 Aegis of Earth: Proton~                  0.8 Acquire         Strate~           57
## 10 Air Conflicts: Secret ~                  3   Games Farm      Action            51
## # ... with 692 more rows
```

c. **Answering question**

```
#since we cannot use string/character value in cforest,
# we are assigning artificial IDs to developer and genre
q4_with_ids <- game_with_sales_genre_developer_cri_score %>%
  group_by(genre_1) %>%
  mutate(genre_id = cur_group_id()) %>%
  group_by(first_developer) %>%
  mutate(developer_id = cur_group_id()) %>%
  ungroup()

#creating model
q4_mod <- cforest(sum_global_sales ~ developer_id + genre_id + critic_score,
                       data = q4_with_ids)

#add_prediction and residuals
q4_pred_resid <- q4_with_ids %>%
  add_predictions(q4_mod) %>%
  add_residuals(q4_mod)
```

We are trying to create a model to predict global sales of a game, knowing its developer, genre, and critic score using cforest. We are choosing cforest because there is a mix of categorical and numerical predictors, and there are not many functions/algorithms that could handle multiple types of predictors effectively. Furthermore, it is not easy to choose between a linear or non-linear model in the first place because we could not effectively visualize these variables simultaneously to choose their model. Thus, one of our best bets would be the cforest function.

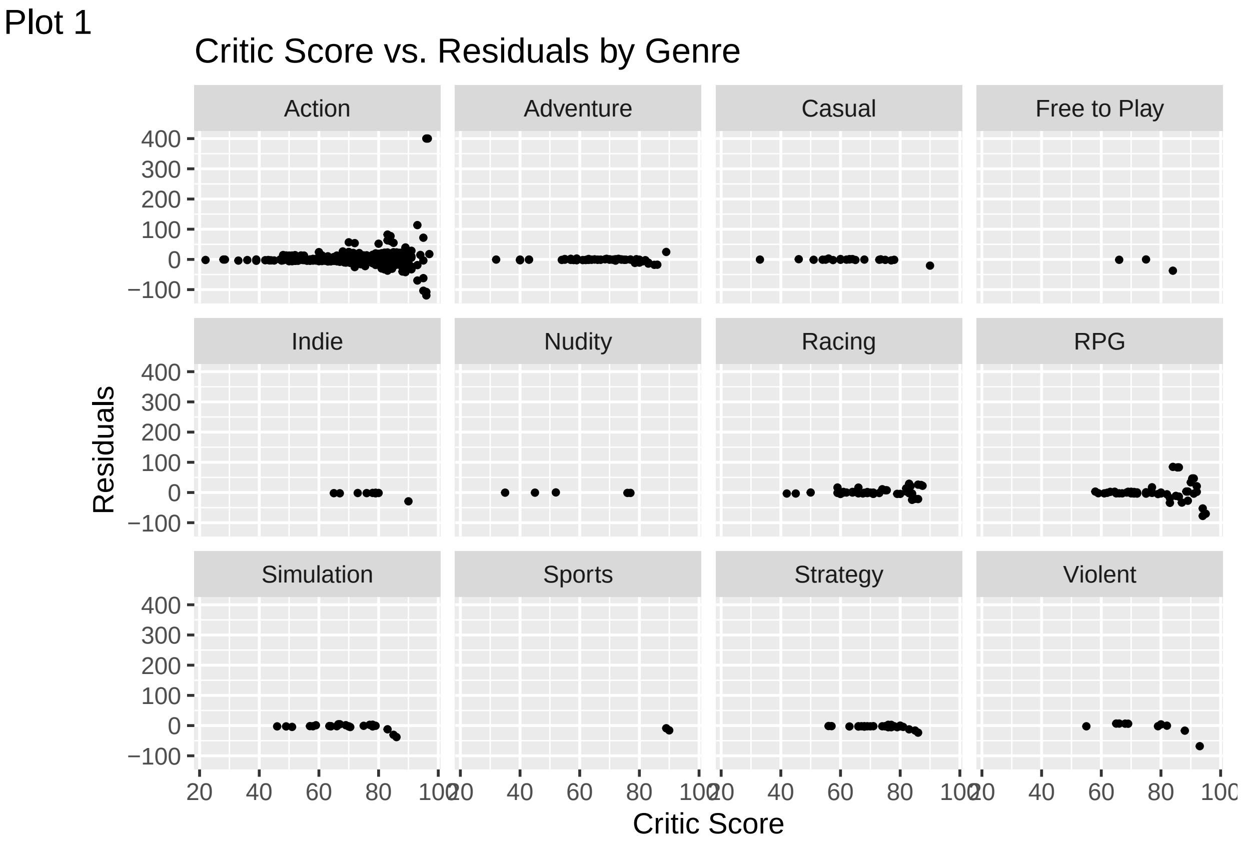
```
#Justify model
q4_pred_resid %>%
  mutate(correct = ifelse(abs(resid) <= 1,
                          1,
                          0)) %>%
  summarize(mean(correct))
```

```
## # A tibble: 1 x 1
##   `mean(correct)`
##             <dbl>
## 1           0.138
```

In order to justify our model, as tricky as it is to observe the residuals and see if there are any patterns, we are creating an artificial standard to justify our model. If the absolute differences between sales predicted by our model and the actual sales differ by 100,000 copies, we assume that we got a correct prediction, and vice versa. Unfortunately, as we notice from the above summary table, we achieved as little as 15% accuracy, which is significantly low.

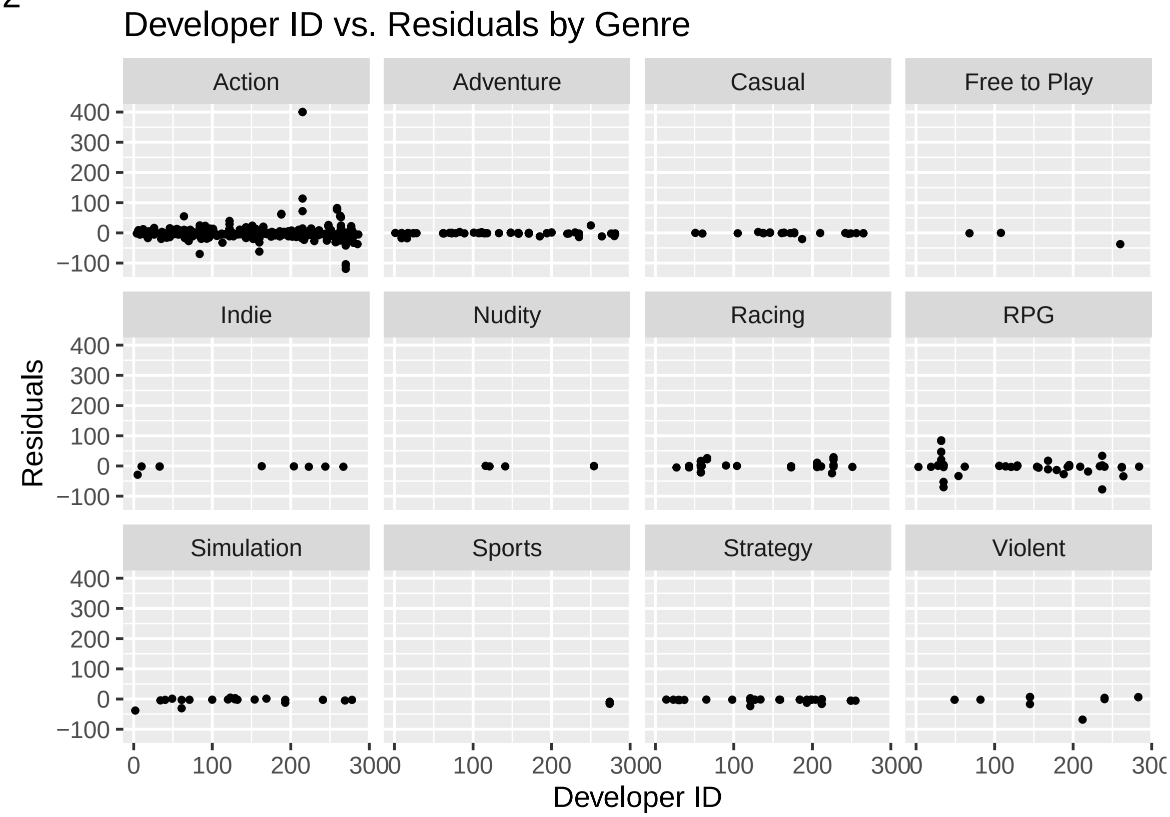
Now we are trying to create various residual plots and see if there are any patterns within these plots.

```
q4_pred_resid %>%
  ggplot() +
  geom_point(aes(x = critic_score,
                 y = resid),
             size = 0.8) +
  facet_wrap(~ genre_1) +
  labs(title = "Critic Score vs. Residuals by Genre",
       tag = "Plot 1") +
  xlab("Critic Score") +
  ylab("Residuals")
```



```
q4_pred_resid %>%
  ggplot() +
  geom_point(aes(x = developer_id,
                 y = resid),
             size = 0.8) +
  facet_wrap(~ genre_1) +
  labs(title = "Developer ID vs. Residuals by Genre",
       tag = "Plot 2") +
  xlab("Developer ID") +
  ylab("Residuals")
```

Plot 2



The first residual plot, Action, Free to Play, Indie, RPG, Simulation, Strategy, and Violent genre, show a decreasing non-linear pattern at the end of the data points, which suggests that this may not be a good model. However, in the second plot, we can observe a decreasing non-linear pattern at the tail-end of the Violent genre and a few at the starting of the Simulation genre, which further confirms that this may not be a good model.

Along the way, we may notice potential flaws during the cleaning and methods we used to represent these variable during the creation of our model, which likely led us to the given result. Therefore, without further correction, more effective method of joining the tibbles, and a better technique to model these variables, we couldn't accomplish a decent global sales prediction model using developer, genre, and critic score.

In conclusion, we have found many discoveries throughout every question. In the first question, the Free to Play genre stood at the top-1 of every year during the last decade, except for 2018, when a Massively Multiplayer genre took control, and the "Battle Royal" games were born. When we looked deeper into other genres that revolved around Free to Play, we noticed that Action and Strategy were also a part of many iconic games that influenced the Free to Play genre. Secondly, we found a growing relationship between the critic score and the player review score. Additionally, we observed a few expected outliers of a triple-A game, having an exceptionally high critic score but exclusively low when the reviews from the players came. Thirdly, we did not find a particular or obvious relationship between global sales and Twitch's view count of the overall data points but found an expected behavior of a few awesome made games. Finally, we did not successfully achieve our goal of creating a model to predict global sales, knowing a game's developer, genre, and critic score due to many difficulties and flaws along the way up to the modeling process.

As we discussed some of the limitations and difficulties along with the analysis, there were many of them. First of all, we claimed that we were doing video games analysis, but our database was based on the Steam database. There are many more digital stores similar to Steam with numerous great games that are not on

Steam. In future work, we could investigate more into different platforms and consoles for different results than today. Next, the data we were working with data that were not catching up with this time period, which may inevitably result in an outdated analysis. Thirdly, some of our final tibble did not have that many observations to work with or analyze due to many of them being excluded during the inefficient cleaning and joining process, which brings us to the next problem. Notice that some tibbles were joined with only a game name, which is a big loophole if there are games with the same name. However, when we tried to solve it by joining both the game name and its developer column, we ended up facing more problems with two names or developers spelled differently, such as "Valve" and "Valve software," resulting in losing a big chunk of data even though these data should not be excluded. In future work, we will definitely need a much better method to handle these complex situations, such as a game ID to identify them uniquely. Ultimately, regarding the modeling process, as discussed in question 4, we need an approach to figure out what the model of multiple cross-categories predictors should be to use a better tool to attain a decent model. Finally, we could have divided our Tibble into a training and testing data set to avoid overfitting.