Final Project: Game Sales Analysis

Puttisan “Ninja” Mukneam

05/13/2022

```r
#Loading the Libraries
library(tidyverse)
library(lubridate)
library(modelr)
library(ggplot2)
library(partykit)
options(na.action = na.warn)
knitr::opts_chunk$set(dev = 'pdf')
```

```r
#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")
```

——————————Delete——————————

Question 1 (18 points)

Consider the data Ca-Vac-Jan-Jun.csv. On the last homework, we fit this data with a quadratic and a quartic model. Now, we’ll try splines!

a. (3 points) **Create a model for this data that is a spline with one degree of freedom. Make predictions for your model, and draw your model on top of the data points. What do you observe?**

——————————Delete——————————

Questions to answer

- What is the most popular genre, in term of the number of players, over the available data period, and how genres has change over time.
- Can we have a model predicting global sales of a game using its genre, developer and critic score.
- Are there any relationship between critics score and player review (ratio of a game?)
- Are there any relationship between Twitch’s viewer and global sale of a game?

——————————Delete——————————

Question 1: What is the most popular genre, in term of the number of players, over the available data period, and how genres has change over time.

Cleaning relevant data

a. **steam__data__1**

```r
#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.
```

```r
#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)
```

b. **steam_data_2**

```r
#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 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 developer 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)
```

c. **Joining tibbles**

```r
#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
  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),
         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
```

```
## # A tibble: 150,530 x 7
##     year month         avg release_date        game_name       joined_dev genres
##    <int> <chr>       <dbl> <dttm>              <chr>           <chr>      <chr>
##  1  2020 December  717804. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Action
##  2  2020 December  717804. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  3  2020 November  668755. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Action
##  4  2020 November  668755. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  5  2020 October   613667. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Action
##  6  2020 October   613667. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  7  2020 September 606850. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Action
##  8  2020 September 606850. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  9  2020 August    639958. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Action
## 10  2020 August    639958. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
## # ... with 150,520 more rows
```

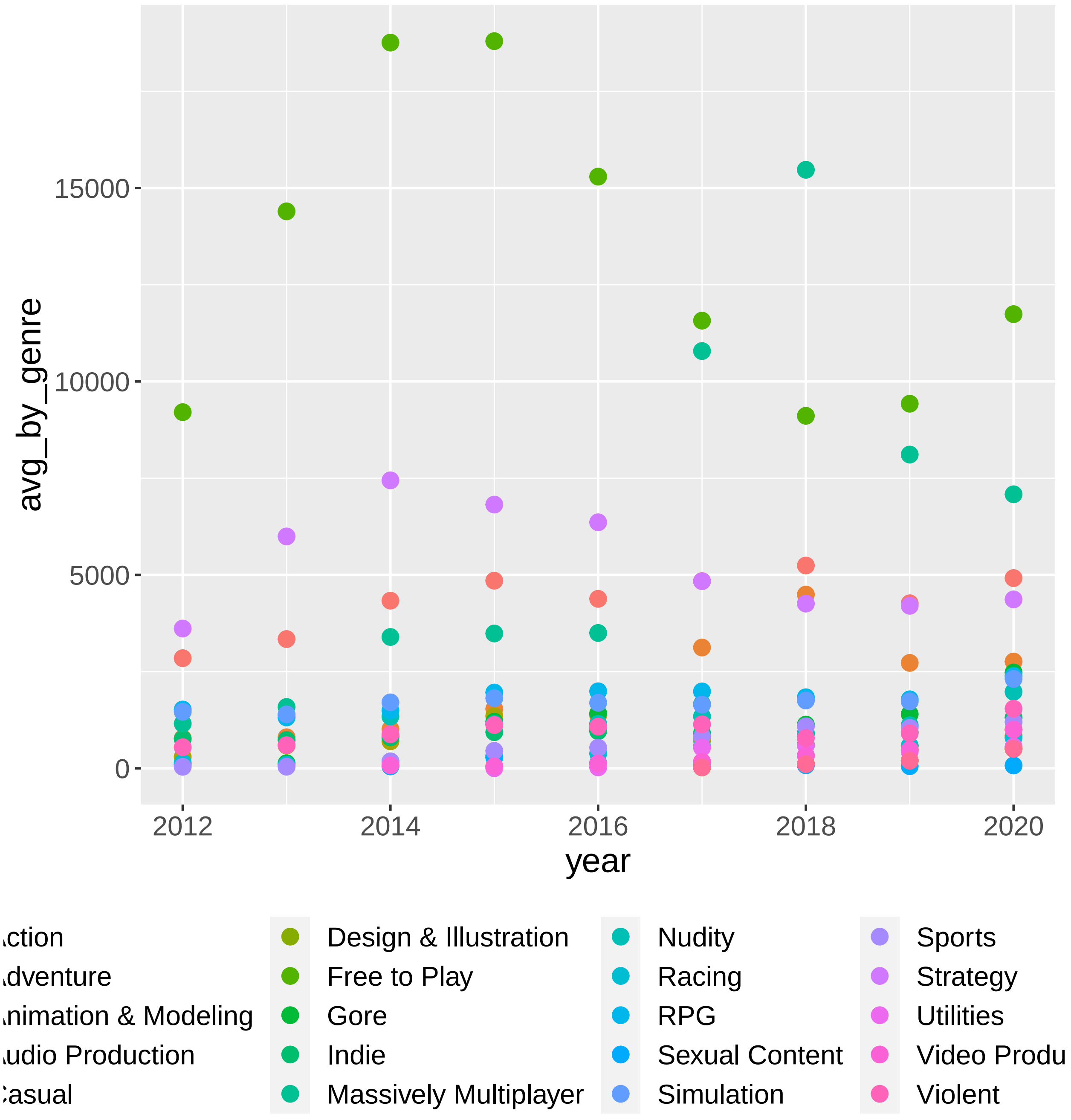
Answering question

a. **genres vs time**Delete maybe

```
#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.
```

```
#plot
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")
```



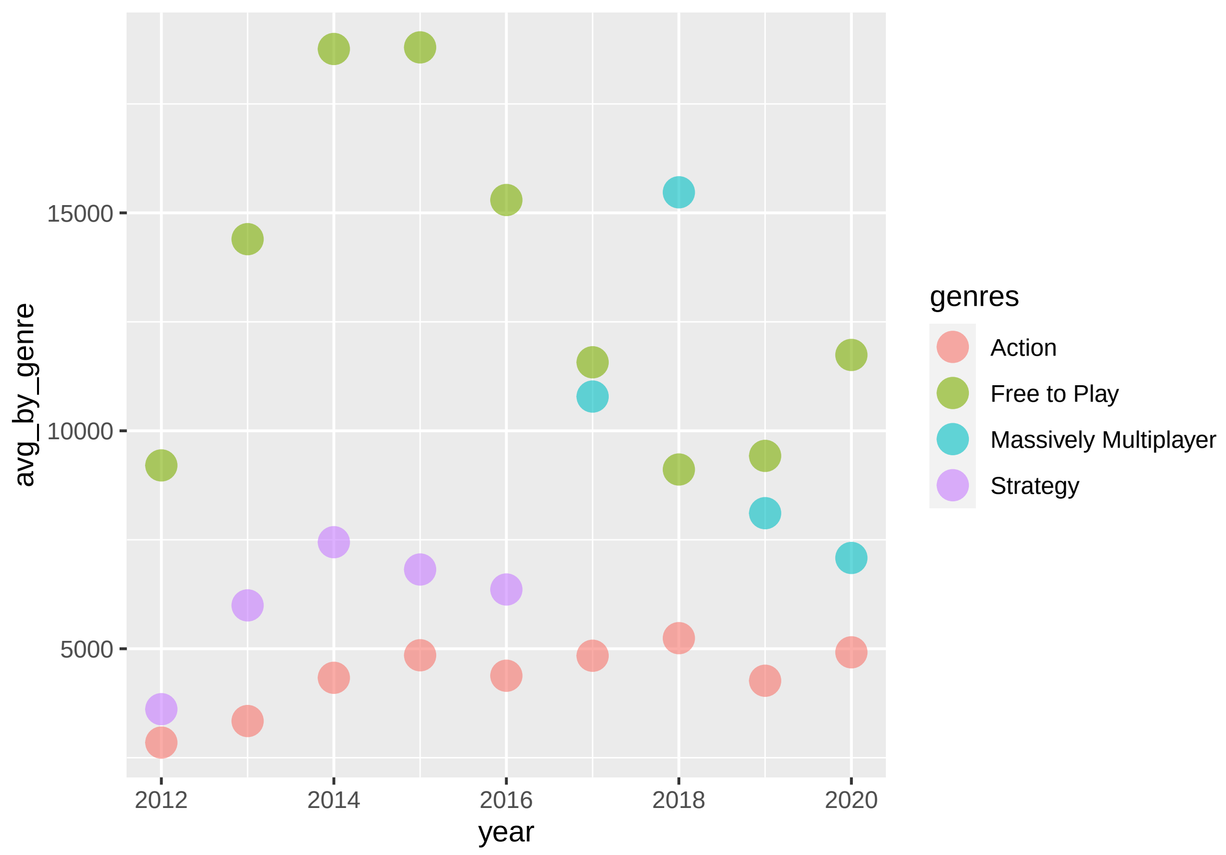
```
#table Top-1 each year
q1_avg_by_year %>%
  group_by(year) %>%
  filter(avg_by_genre == max(avg_by_genre))
```

```
## # 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.
```

```
#anlsys

q1 %>%
  filter(year == 2020, genres == "Free to Play") %>%
  arrange(desc(avg))
```

```
## # A tibble: 1,428 x 7
##     year month        avg release_date        game_name       joined_dev genres
##    <int> <chr>      <dbl> <dttm>              <chr>           <chr>      <chr>
##  1  2020 April     857604. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  2  2020 May       768795. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  3  2020 December  717804. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  4  2020 June      671647. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  5  2020 March     671033. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  6  2020 November  668755. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  7  2020 August    639958. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  8  2020 July      625901. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
##  9  2020 October   613667. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
## 10  2020 September 606850. 2012-08-21 00:00:00 Counter-Strike~ valve hid~ Free ~
## # ... with 1,418 more rows
```

```
#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 = 5,
             alpha = 0.6)
```



Question 2:Are there any relationship between critics score and player review (ratio of a game?)

Cleaning relevant data

a. **steam_data_1**

```
#Cleaning dates
steam_data_1_cleaned_1 <- steam_data_1 %>%
  distinct(name, .keep_all = TRUE) %>%
  select(date, name, developer, user_reviews, all_reviews) %>%
  filter(!is.na(name),
         !name == "-",
         !is.na(developer),
         !date == "-",
         !str_detect(date, "\\w{3}", negate = TRUE)) %>%
  rename(game_name = name,
         release_date = date,
         reviews_all_time = all_reviews,
         reviews_30_days = user_reviews) %>%
  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))
```

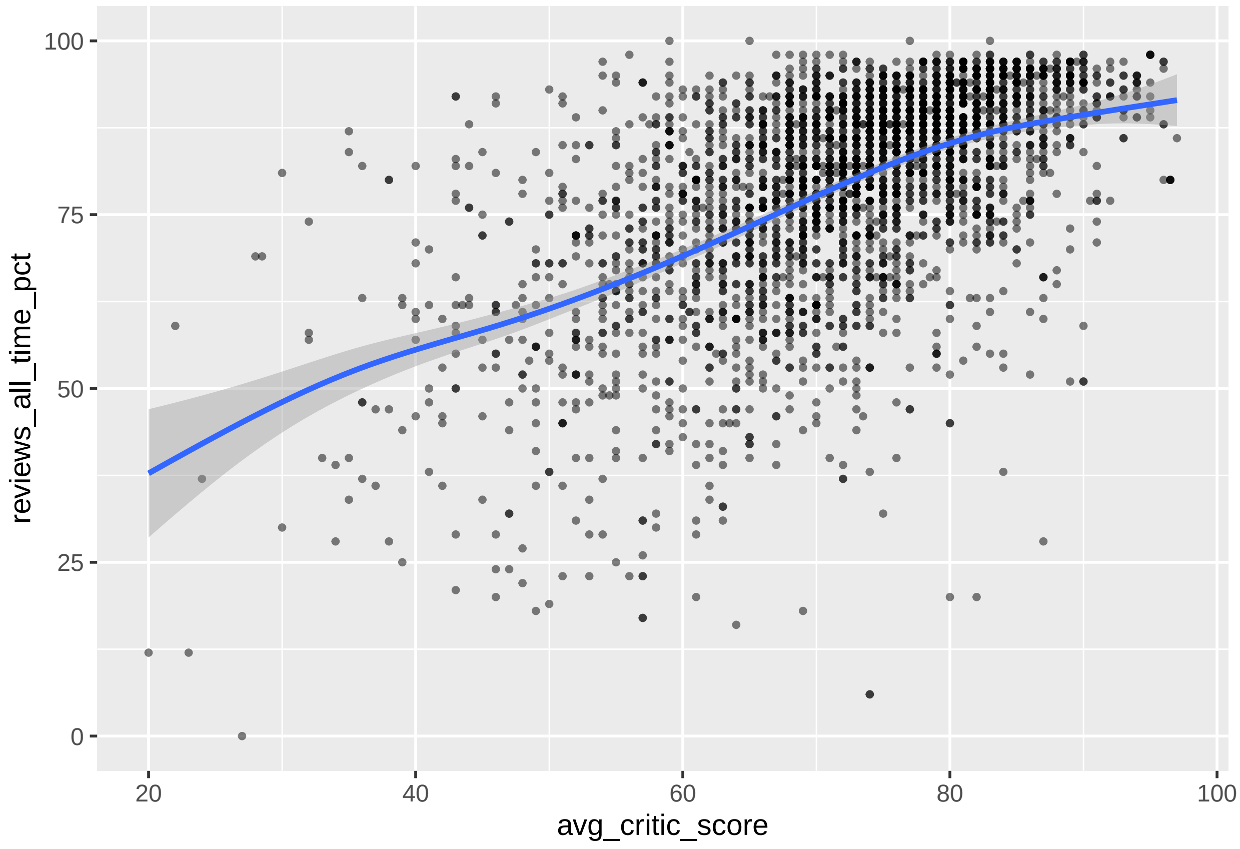
```
## Warning: 2805 failed to parse.
```

```
#Cleaning developer
steam_data_1_cleaned_2 <- steam_data_1_cleaned_1 %>%
  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)
  unnest_legacy(d1) %>%
  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
steam_data1_cleaned_3 <- steam_data_1_cleaned_2 %>%
  filter(str_detect(reviews_30_days, "%")) %>%
  mutate(reviews_all_time_pct = ifelse(!str_detect(reviews_all_time, "%"),
                                       NA,
                                       reviews_all_time),
         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) %>%
  mutate(reviews_all_time_pct = str_extract(reviews_all_time_pct, "[:digit:]+(?=%)"),
         reviews_all_time_pct = as.double(reviews_all_time_pct))


temp5 <- video_game_sales_1 %>%
  select(game, metascore, developer, release_date) %>%
  filter(!is.na(metascore)) %>%
  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)
  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) %>%
  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),
         first_developer = ifelse(is.na(developer_1),
                                  developer_2,
                                  developer_1)) %>%
```

```r
  select(game, metascore, joined_game_name, joined_dev, first_developer) %>%
  filter(!is.na(joined_dev))


temp4 <- video_game_sales_2 %>%
  select(Name, Critic_Score, Developer, NA_Sales) %>%
  filter(!is.na(Critic_Score)) %>%
  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),
         first_developer = ifelse(is.na(developer_1),
                                  developer_2,
                                  developer_1)) %>%
  select(Name, Critic_Score, joined_game_name, joined_dev, first_developer) %>%
  filter(!is.na(joined_dev))



full_join_video_game_sales <- full_join(temp4, temp5, by = c("joined_game_name", "joined_dev")) %>%
  filter(!is.na(Critic_Score) | !is.na(metascore))

full_join_video_game_sales_critic <- full_join_video_game_sales %>%
  mutate(critic_score = rowMeans(full_join_video_game_sales[c('Critic_Score', 'metascore')], na.rm = TRU
  select(Name, critic_score, joined_game_name, joined_dev)

# Join with player score

steam_data1_joined_3 <- steam_data1_cleaned_3 %>%
  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, reviews_all_time_pct) %>%
  filter(!is.na(joined_dev))
```

```r
#Fin
joined_critic_all_review <- left_join(full_join_video_game_sales_critic, steam_data1_joined_3, by = c("
  filter(!is.na(reviews_all_time_pct)) %>%
  select(Name, critic_score, reviews_all_time_pct) %>%
  rename("game_name" = Name,
         "avg_critic_score" = critic_score)


#ploting
joined_critic_all_review %>%
  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),
              method = NULL)
```

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



Question 4: Are there any relationship between Twitch's viewer and global sale of a game?

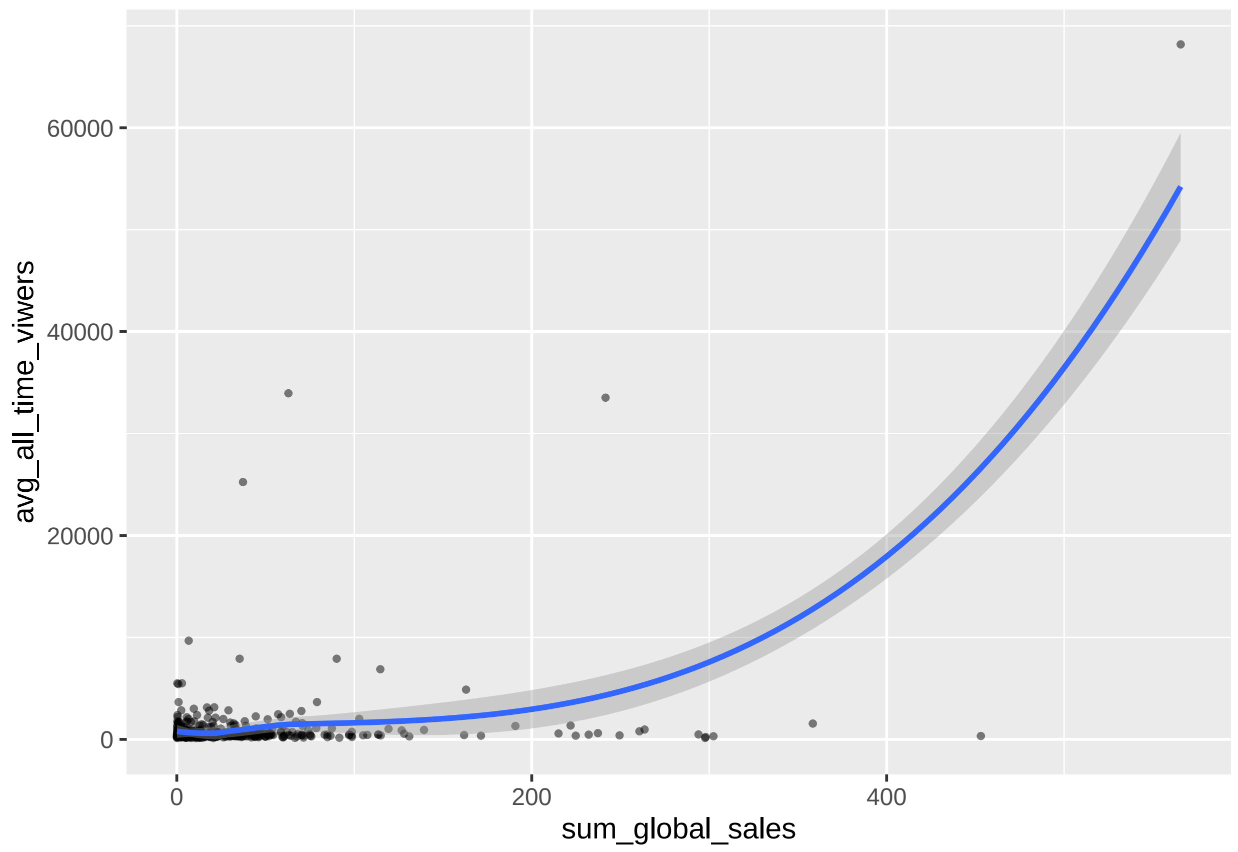
- cleaning twitch

```r
twitch_game_popularity_joined <- twitch_game_popularity %>%
  group_by(Game) %>%
  summarise(avg_all_time_viwers = mean(Avg_viewers)) %>%
  mutate(joined_game_name = str_replace_all(Game, "[^[:alnum:] ]", " "),
         joined_game_name = str_to_lower(joined_game_name)) %>%
  rename("game_name" = Game)



###############################################################################################
video_game_sales2_joined <- 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
  select(Name, joined_game_name, sum_global_sales) %>%
  rename("game_name" = Name)
```

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

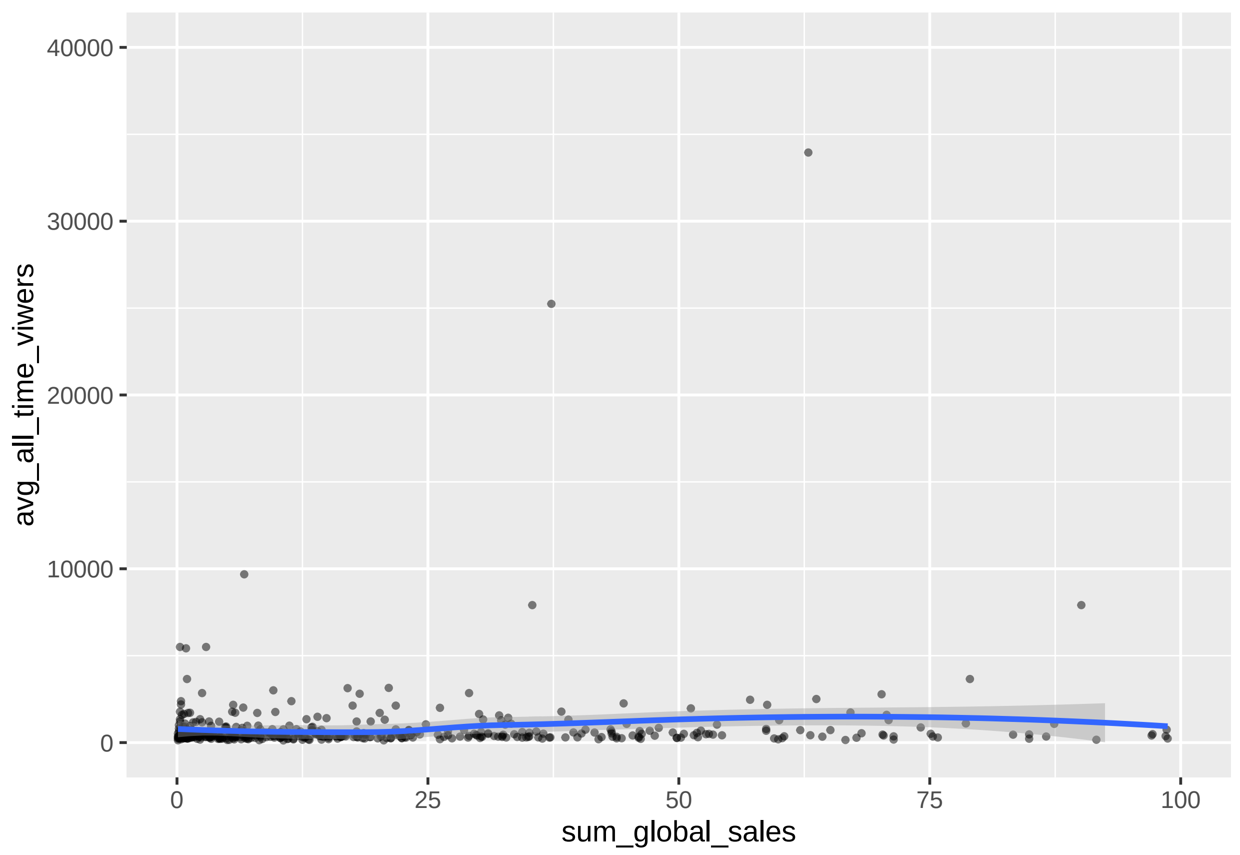
```r
#join
joined_avg_viwer_global_sales <- inner_join(twitch_game_popularity_joined, video_game_sales2_joined, by
  select(game_name, avg_all_time_viwers, sum_global_sales)

#plot 1
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),
              method = NULL) #+
```

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



```
 # ylim(0, 20000) +
  #xlim(0, 100)

#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),
              method = NULL) +
  ylim(0, 40000) +
  xlim(0, 100)
```

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

## Warning: Removed 32 rows containing non-finite values (stat_smooth).

## Warning: Removed 32 rows containing missing values (geom_point).
```



Question 5: Can we have a model predicting global sales of a game using its genre, developer and critic score.

```
#global sales
video_game_sales2_joinable_dev <- 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),
         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.
```

```
#Genre
steam_data1_joinable_first_genre <- steam_data_1_with_genres %>%
  select(game_name, joined_game_name, joined_dev, genre_1)

#Critic_score
full_join_video_game_sales_critic_joinable <- full_join_video_game_sales_critic %>%
  filter(!is.na(joined_game_name))

#fin
joined_q4 <- inner_join(video_game_sales2_joinable_dev, steam_data1_joinable_first_genre, by = "joined_
  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)


#assigning artificial IDs to developer and genre
joined_q4_with_ids <- joined_q4 %>%
  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 = joined_q4_with_ids)

#add_prediction and residuals
q4_pred_resid <- joined_q4_with_ids %>%
  add_predictions(q4_mod) %>%
  add_residuals(q4_mod)

#Justify model
q4_pred_resid %>%
  mutate(correct = ifelse(abs(resid) <= 0.5,
                          1,
                          0)) %>%
  summarize(mean(correct))
```

```
## # A tibble: 1 x 1
##   `mean(correct)`
##             <dbl>
## 1          0.0684
```

```
###############################V222222222#################################################################

#assigning artificial IDs to developer and genre
joined_q4_with_ids_2 <- joined_q4 %>%
  group_by(genre_1) %>%
  mutate(genre_id = cur_group_id()) %>%
  ungroup()


#creating model
q4_mod_2 <- cforest(sum_global_sales ~ genre_id + critic_score,
                       data = joined_q4_with_ids_2)

#add_prediction and residuals
q4_pred_resid_2 <- joined_q4_with_ids_2 %>%
  add_predictions(q4_mod_2) %>%
  add_residuals(q4_mod_2)

#Justify model
q4_pred_resid_2 %>%
  mutate(correct = ifelse(abs(resid) <= 1,
                          1,
                          0)) %>%
  summarize(mean(correct))
```

```
## # A tibble: 1 x 1
##   `mean(correct)`
##             <dbl>
## 1           0.155
```

```
q4_pred_resid_2 %>%
  select(sum_global_sales, pred)
```

```
## # A tibble: 702 x 2
##    sum_global_sales  pred
##               <dbl> <dbl>
##  1              0.1  1.20
##  2              1.9  1.82
##  3              1.9  1.84
##  4              3.1  1.70
##  5              1.3  1.70
##  6              0.8  2.09
##  7              2.1  1.72
##  8              0.2  1.25
##  9              0.8  1.99
## 10              3    7.74
## # ... with 692 more rows
```

###Storage###############
' ' '

#Cleaning dates steam_data_1_cleaned_1 <- steam_data_1 %>% distinct(name, .keep_all = TRUE) %>% select(date, name, developer, user_reviews, all_reviews) %>% filter(!is.na(name), !name == "-", !is.na(developer), !date == "-", !str_detect(date, "\w{3}", negate = TRUE)) %>% rename(game_name = name, release_date = date, reviews_all_time = all_reviews, reviews_30_days = user_reviews) %>% 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))

#Cleaning developer steam_data_1_cleaned_2 <- steam_data_1_cleaned_1 %>% 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).?|.dat|s. r. o.)”)) %>% unnest_legacy(d1) %>% 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 steam_data1_cleaned_3 <- steam_data_1_cleaned_2 %>% filter(str_detect(reviews_30_days, “%”)) %>% mutate(reviews_all_time_pct = ifelse(!str_detect(reviews_all_time, “%”), NA, reviews_all_time), 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) %>% mutate(reviews_all_time_pct = str_extract(reviews_all_time_pct, “[:digit:]+(?=%)”), reviews_all_time_pct = as.double(reviews_all_time_pct))

#joining all these information test <- stringdist_inner_join(steam_data1_joinable_first_genre, video_game_sales2_joinable_dev, by = c(“joined_game_name”)) %>%

test2 <- test %>% select(Name, joined_game_name.y, genre_1, sum_global_sales, first_developer) %>% rename(“joined_game_name” = “joined_game_name.y”)

test3 <- stringdist_inner_join(test2, full_join_video_game_sales_critic_joinable, by = “joined_game_name”)

video_game_sales2_joinable_dev <- video_game_sales_2 %>% group_by(Name) %>% 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
r. o.)“)) %>% 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), 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))