finalProject

Sarah THEOULLE

2024-06-07

Final projet R-visualisation

The goal of this document is to expose the results of my analysis of the biggest categories watched on Twitch (twitch.tv) from the 22nd of April up until the 11th of July. I gathered all the data myself and the method will be detailed below before the analysis itself.

The technical report will be structured as follows :

- Overview of the project
- Gathering the data
- Analysis of the results
- Annexes

Overview of the project

The goal of this paper is to analyze the viewership trends and patterns of the top 100 categories streamed on Twitch between the 22nd of April and the 11th of July.

Twitch is a video live-streaming service that focuses on video game live streaming, including broadcasts of esport competitions, music broadcasts, creative content, and "in real life" streams. When a creator broadcasts content, they need to choose a category (Just Chatting, League of Legends, Music, etc.) according to their content.

Gathering the datas

To collect the data, I wrote several scripts in Python and connected them to the Twitch API. Twitch has a public API that anyone with an account and developer credentials can access (https://dev.twitch.tv/console/apps). I mainly used scripts. During my preliminary research, I collected the top 100 streamed categories and their IDs. Then I iterated over this data set every week to get the position of the top 100 as well as all their data. From week to week, the weekly data set got smaller and smaller because all the games that weren't in the top 100 anymore were deleted. This is due to the fact that some games can have a buzz for a week and then disappear completely, distorting the top 100 data. So I decided to remove any line where the game was 101 or worse.

```
library(dplyr)
library(ggplot2)
library(readr)
library(ggrepel)
```

```r
library(gridExtra)
library(reticulate)
library(circlize)
library(tidyr)

games_combined <- read_csv("games_combined.csv")
top20_17 <- read_csv("get_top100_17.csv")
top20_18 <- read_csv("get_top100_18.csv")
top20_19 <- read_csv("get_top100_19.csv")
top20_20 <- read_csv("get_top100_20.csv")
top20_21 <- read_csv("get_top100_21.csv")
top20_22 <- read_csv("get_top100_22.csv")
top20_26 <- read_csv("get_top100_26.csv")


top20_17$Week <- 17
top20_18$Week <- 18
top20_19$Week <- 19
top20_20$Week <- 20
top20_21$Week <- 21
top20_22$Week <- 22
top20_26$Week <- 26
```

Analysis of the results

Here are the three questions this report will be answering :

- How can abrupt changes in the hierarchy of top categories be interpreted?
- Is there a relationship between the average viewership of a category and the total hours watched?
- How does the viewer engagement trends vary among the top-ranked categories and the lowest categories ?

1- Detecting Significant Changes in Category Rankings

How can abrupt changes in the hierarchy of top categories be interpreted? For instance, can events like the release of a new DLC (e.g., Sea of Thieves or Elden Ring) or a major tournament (e.g. Marvel Snap or GTA V) be detected through the data?

In this section, we analyze the abrupt changes in the rankings of the most-watched Twitch categories. We investigate spikes in viewership or shifts in ranking that coincide with notable events like the end of popular events or new content releases (e.g., games or DLC).

Description of the statistical experiment

- **Population**:
  - **Definition**: All categories of content streamed on Twitch during the specified period.
  - **Size**: Includes all categories that were streamed on Twitch between April 22 and July 11.
- **Sample**:

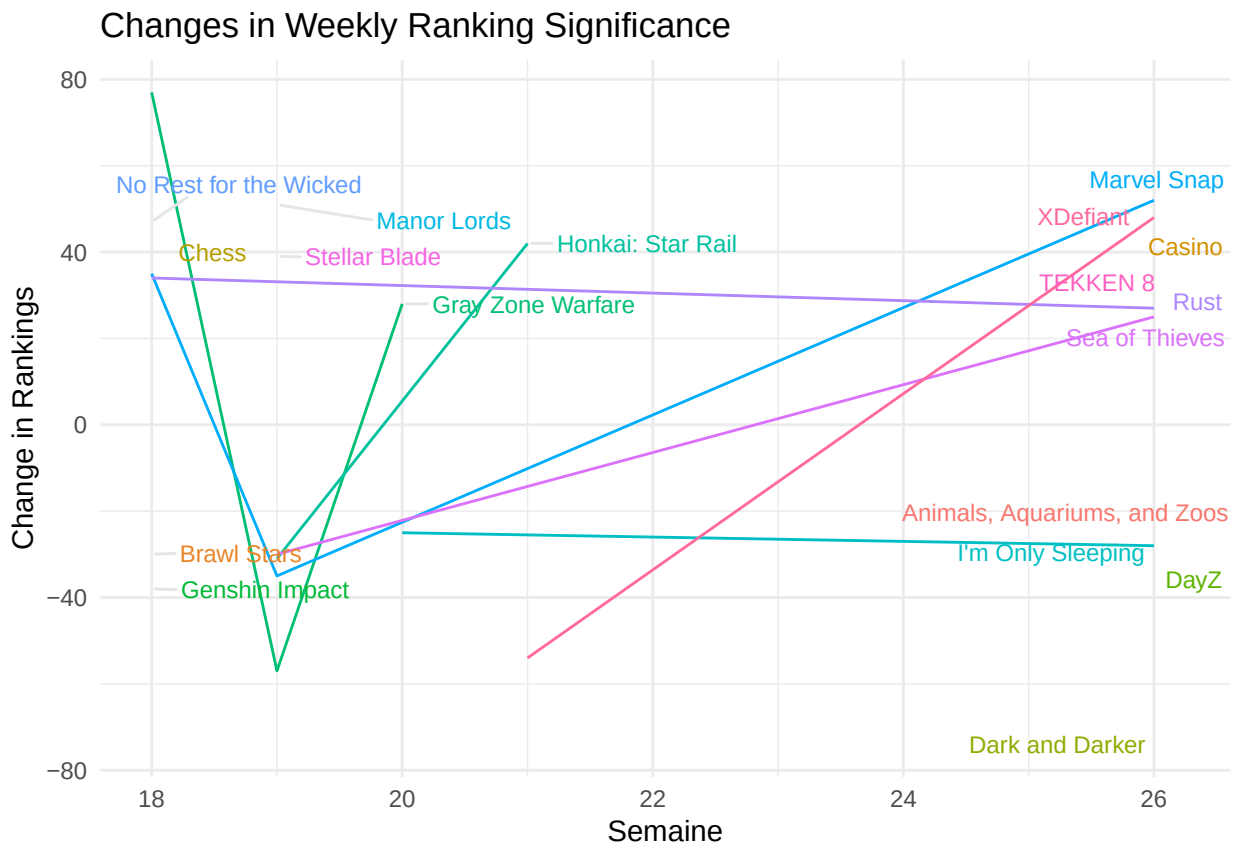
  - **Definition**: The top 100 most-watched categories on Twitch for each week during the specified period.
  - **Size**: The weekly data set includes 100 categories.

- **Variable**:
  - **Definition**: The rank of a category in the list of the top 100 most-watched categories.

Processus

1. **Data Collection**:
   - Gather weekly data on the top 100 most-watched categories on Twitch.
   - Record the rank of each category for each week.
2. **Analysis**:
   - Calculate changes in rankings from one week to the next.
   - Identify significant changes (e.g., a change in rank by 25 or more positions).
3. **Interpretation**:
   - Investigate the possible causes of abrupt changes, such as new game releases, major tournaments, or special events.

```r
# Collect all datas in one dataframe
all_data <- bind_rows(top20_17, top20_18, top20_19, top20_20, top20_21, top20_22, top20_26)

all_data <- all_data %>%
  arrange(Name, Week) %>%
  group_by(Name) %>%
  mutate(Rank_Change = Rank - lag(Rank))

# Filtering
significant_changes <- all_data %>%
  filter(!is.na(Rank_Change) & abs(Rank_Change) >= 25)

end_points <- significant_changes %>%
  group_by(Name) %>%
  filter(Week == max(Week))

significant_games <- significant_changes %>%
  pull(Name) %>%
  unique()

ggplot(significant_changes, aes(x = Week, y = Rank_Change, color = Name)) +
  geom_line() +
  geom_text_repel(data = end_points, aes(label = Name),
                  nudge_x = 0.2, hjust = 0, size = 3, segment.color = "grey90") +
  scale_color_manual(values = scales::hue_pal()(length(significant_games)),
                     breaks = significant_games,
                     guide = "none") +
  labs(title = "Changes in Weekly Ranking Significance", x = "Semaine", y = "Change in Rankings") +
  theme_minimal()
```



1. Detecting Significant Changes in Category Rankings

The chart "Changes in Weekly Ranking Significance" illustrates the significant changes in rankings for various Twitch categories from week 18 to week 26. Each line represents a game, showing how its ranking has evolved over the weeks.

Key Observations:

- **No Rest for the Wicked** and **Chess**: These categories saw a sharp increase in ranking around week 18 but declined shortly after.
- **Marvel Snap**: This category showed a consistent upward trend, indicating growing interest over the observed weeks.
- **Gray Zone Warfare** and **Manor Lords**: These categories had a significant jump in ranking between weeks 18 and 20, suggesting increased viewership or successful events/launches during that period.

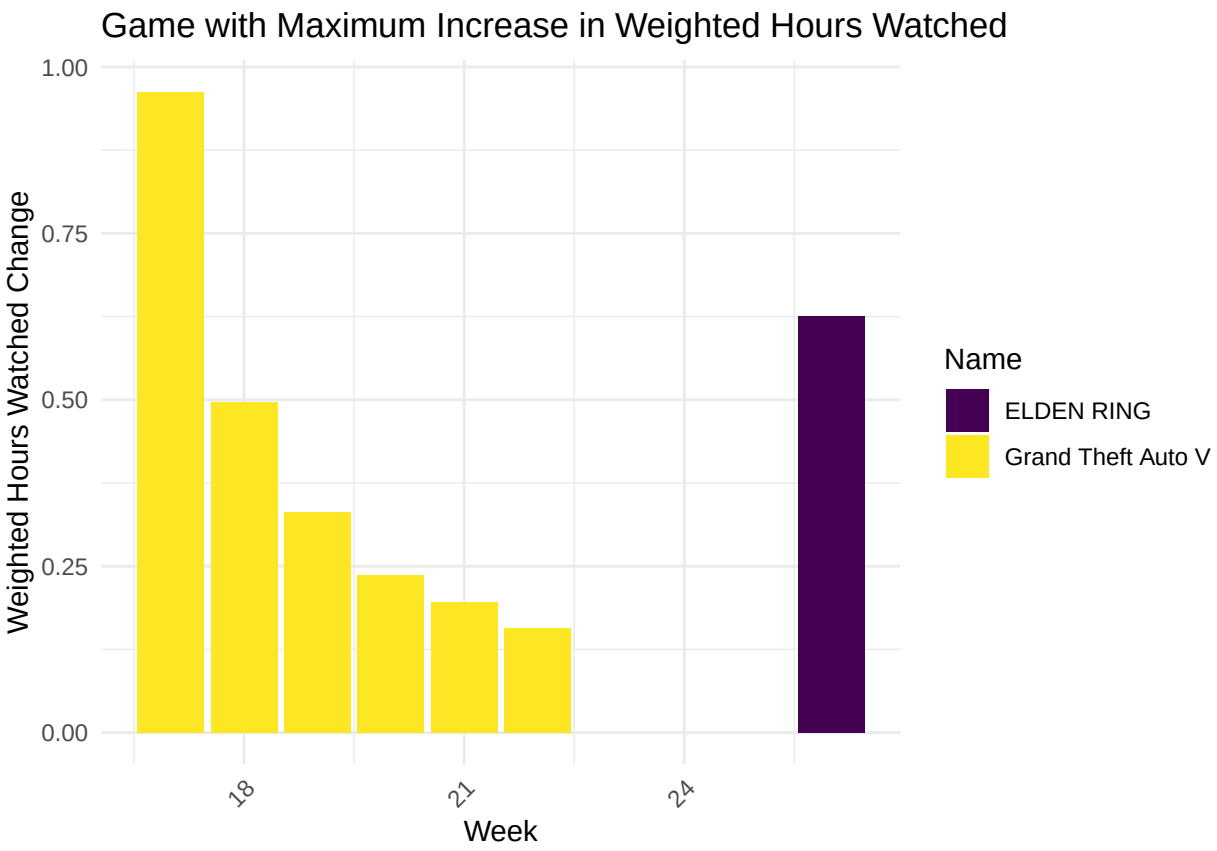
These sudden changes likely reflect reactions to specific events such as new content releases, updates, or major tournaments. For instance, spikes in rankings for **Sea of Thieves** and **Marvel Snap** could be linked to new DLC releases or updates observed during the data period.

```
# Calculate the change in hours watched from one week to the next
total_hours_by_game <- all_data %>%
  group_by(Week, Name) %>%
  summarise(Total_Hours_Watched = sum(`Hours Watched`, na.rm = TRUE)) %>%
  arrange(Week) %>%
```

```
  mutate(Hours_Watched_Change = Total_Hours_Watched - lag(Total_Hours_Watched),
         Hours_Watched_Change = ifelse(is.na(Hours_Watched_Change), 0, Hours_Watched_Change))
```

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

```
total_hours_by_game <- total_hours_by_game %>%
  arrange(Week, desc(Hours_Watched_Change))

# Group by Week and select the second highest Hours_Watched_Change
second_max_increase_game <- total_hours_by_game %>%
  group_by(Week) %>%
  slice(2)

# Merge to get the game with the maximum increase
total_hours_by_game <- total_hours_by_game %>%
  left_join(second_max_increase_game %>% select(Week, Max_Increase_Game = Name), by = "Week")

# Adjust the Contribution calculation
total_hours_by_game <- total_hours_by_game %>%
  group_by(Name) %>%
  mutate(Weighted_Hours_Watched_Change = Hours_Watched_Change / cumsum(Total_Hours_Watched),
         Contribution = ifelse(Name == Max_Increase_Game, 1, Weighted_Hours_Watched_Change / sum(Weight


# Filter to get only rows where the game is the one with the maximum increase
max_increase_games <- total_hours_by_game %>%
  filter(Name == Max_Increase_Game)

# Plot the bar chart with weighted hours watched change
ggplot(max_increase_games, aes(x = Week, y = Weighted_Hours_Watched_Change, fill = Name)) +
  geom_bar(stat = "identity") +
  labs(x = "Week", y = "Weighted Hours Watched Change", title = "Game with Maximum Increase in Weighted
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_viridis_d()  # You can change the color scale if needed
```



2. Viewer Engagement Trends for Top Games

The chart "Game with Maximum Increase in Weighted Hours Watched" focuses on games that experienced significant increases in weighted hours watched over the weeks. It highlights trends for **ELDEN RING** and **Grand Theft Auto V**.

Key Observations: **Grand Theft Auto V (GTA V)**: From weeks 18 to 22, GTA V consistently showed a substantial increase in weighted hours watched, indicating sustained viewer engagement. This surge is due to a major role-playing (RP) event during this period, which attracted a large audience and maintained high viewership.

ELDEN RING: In week 26, ELDEN RING saw a remarkable spike in weighted hours watched, surpassing all other categories for that week. This increase was driven by the release of a new DLC, generating renewed interest and capturing significant viewer attention.

These trends highlight games that effectively maintained viewer interest over time. GTA V's sustained popularity and ELDEN RING's sudden spike underscore the impact of strategic events and new content on viewer engagement.

Conclusion

Analyzing viewer engagement trends among top-ranked games reveals a dynamic landscape where strategic events and content updates play crucial roles in attracting and maintaining viewer interest. Games like GTA V and ELDEN RING exemplify different approaches to engaging audiences, from consistent performance to

periodic spikes driven by new content releases. This analysis underscores the importance of content strategy and community engagement in fostering sustained viewer interest on platforms like Twitch.

2- Correlation between Average Viewers and Hours Watched

Is there a relationship between the average viewership of a category and the total hours watched?

To explore this, we can examine a scatter plot to analyze the correlation between average viewership and total hours watched for each week.

Description of the statistical experiment

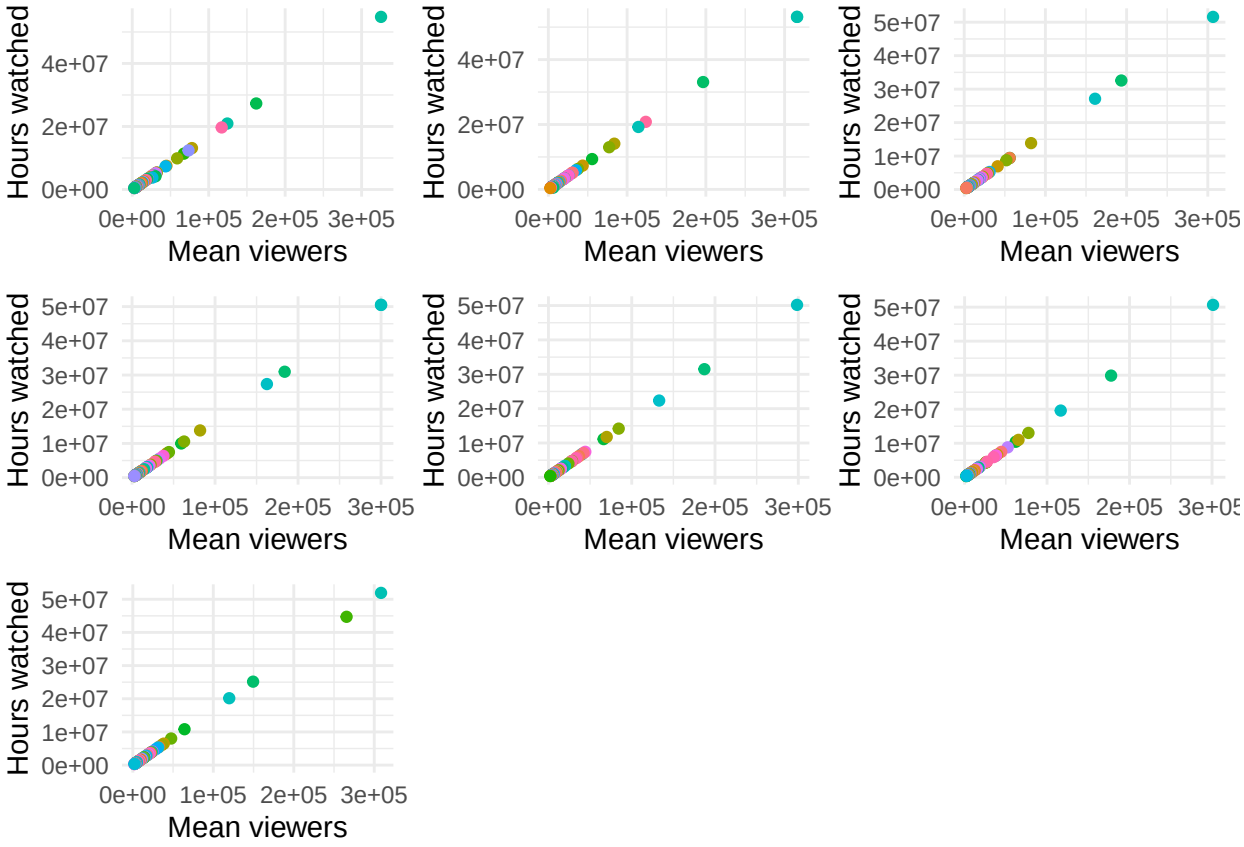
- **Population**:
  - **Definition**: All viewing data for categories streamed on Twitch during the specified period.
  - **Size**: Includes viewership data for all categories streamed on Twitch between April 22 and July 11.
- **Sample**:
  - **Definition**: Weekly data for the top 100 most-watched categories on Twitch.
  - **Size**: 100 categories per week.
- **Variables**:
  - **Average Viewership**:
    - **Definition**: The average number of viewers for a category during the specified week.
  - **Total Hours Watched**:
    - **Definition**: The total number of hours watched for a category during the specified week.

Processus

1. **Data Collection**:
   - Collect weekly data on the average viewership and total hours watched for each of the top 100 categories.
2. **Analysis**:
   - Plot a scatter plot with average viewership on the x-axis and total hours watched on the y-axis.
   - Calculate correlation coefficients to quantify the relationship.
3. **Interpretation**:
   - Determine if there is a linear relationship between average viewership and total hours watched.
   - Analyze any deviations or outliers to understand specific cases.

```
# List of datasets
week_datasets <- list(
  w17 = top20_17,
  w18 = top20_18,
  w19 = top20_19,
  w20 = top20_20,
  w21 = top20_21,
  w22 = top20_22,
  w26 = top20_26
```

```
)

# Create an empty list to store plots
plots <- list()

# Generate scatter plots for each week
for (week_name in names(week_datasets)) {
  week_data <- week_datasets[[week_name]]

  p <- ggplot(week_data, aes(x = `Average Viewers`, y = `Hours Watched`, color = Name)) +
    geom_point() +
    labs(
         x = "Mean viewers",
         y = "Hours watched") +
    theme_minimal() +
    theme(legend.position = "none")

  plots[[week_name]] <- p
}

# Grid layout
do.call("grid.arrange", c(plots, ncol = 3))
```



We notice that for each week, the distribution is perfectly linear. The linear distribution of average viewers and hours watched across different weeks on Twitch can be attributed to the inherent mathematical relationship between these two metrics, assuming consistent streaming times. This direct proportionality

underscores the predictable nature of viewer engagement metrics in live streaming platforms like Twitch, where higher average viewers naturally lead to higher total hours watched given steady streaming durations.

An exception could occur if a very large number of viewers were connected at a specific moment (notably when someone raids a channel) in a game and these viewers disconnect very quickly. This would then show that certain content does not retain viewers at all when users are sent to the channel without having clicked on the content themselves.

3 - Viewer engagement trends variations

How does the viewer engagement trends vary among the top-ranked categories and the lowest categories ?

We will answer to this question with the next plot, analyzing the viewer engagement trends among the top-ranked categories and the lowest categories. We will create line plots to visualize the evolution of the average number of viewers for both the most popular games and the least popular games over the observed weeks.

Description of the statistical experiment

- **Population**:
  - **Definition**: All viewership data for categories streamed on Twitch during the specified period.
  - **Size**: Includes viewership data for all categories streamed on Twitch between April 22 and July 11.
- **Sample**:
  - **Definition**: Weekly data for the top 10 and bottom 25 most-watched categories on Twitch.
  - **Size**: Top 10 categories and bottom 25 categories per week.
- **Variables**:
  - **Average Viewership**:
    - **Definition**: The average number of viewers for a category during the specified week.

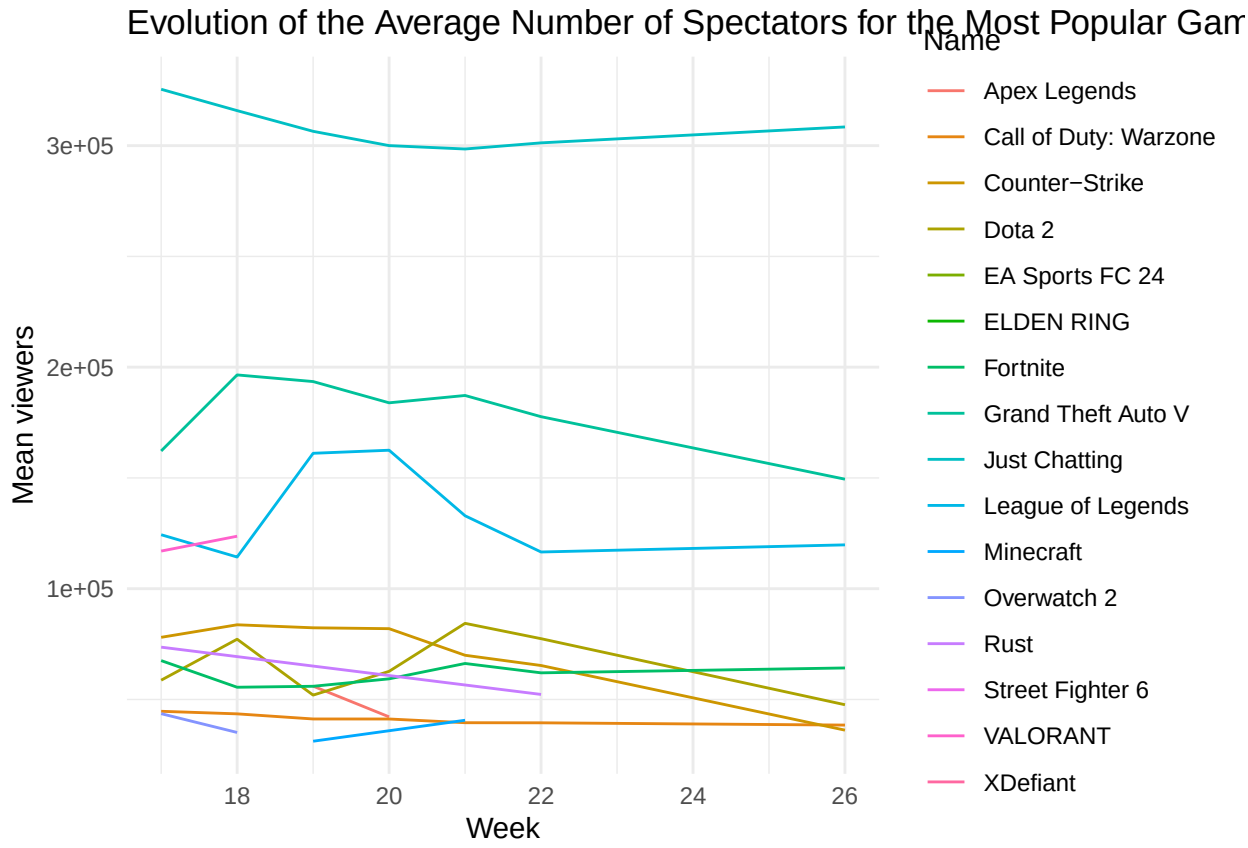
Processus

1. **Data Collection**:
   - Collect weekly data on the average viewership for the top 10 and bottom 25 most-watched categories.
2. **Analysis**:
   - Create line plots to visualize the evolution of average viewership over the weeks for the top 10 and bottom 25 categories separately.
   - Calculate summary statistics for each group.
3. **Interpretation**:
   - Compare the viewer engagement trends between the top-ranked and lowest-ranked categories.
   - Analyze factors contributing to the stability or volatility in viewership trends for different categories.

```r
# Most popular games
top_games <- all_data %>%
  filter(Rank <= 10)

# Mean viewers per week and game
average_viewers <- top_games %>%
  group_by(Name, Week) %>%
  summarise(Average_Viewers = mean(`Average Viewers`, na.rm = TRUE))
```

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

```r
ggplot(average_viewers, aes(x = Week, y = Average_Viewers, color = Name)) +
  geom_line() +
  labs(title = "Evolution of the Average Number of Spectators for the Most Popular Games", x = "Week", y
  theme_minimal()
```



Viewer Engagement Stability Among Top Games

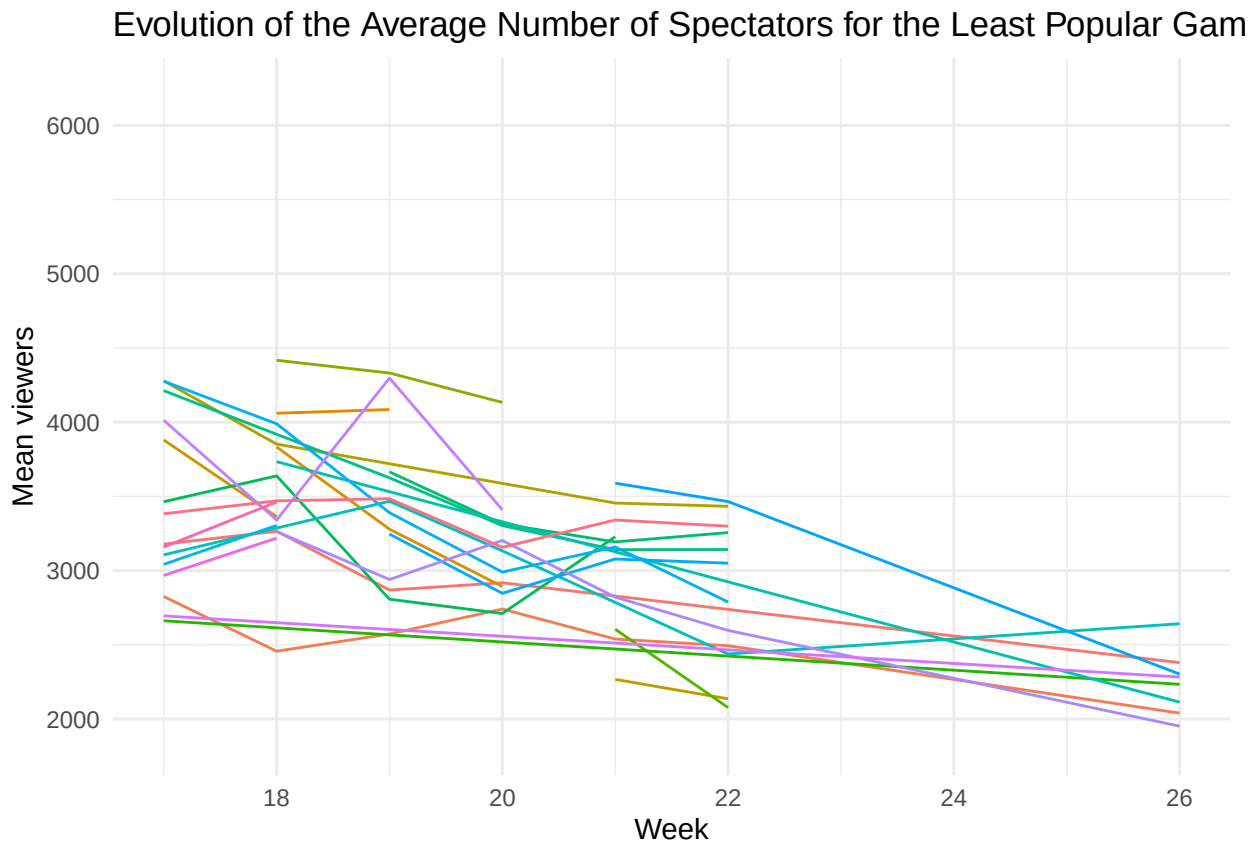
This graph illustrates the stability of the most-watched games throughout the week. The notable exception is the viewership peak during weeks 19 and 20 for League of Legends. This surge can be attributed to the final matches of the LEC competition, the European League of Legends Championship.

Games demonstrating consistent and substantial appeal on this graph include Just Chatting (live discussions with viewers without gameplay), Grand Theft Auto V, and League of Legends.

```r
# Least popular games
bot_games <- all_data %>%
  filter(Rank > 75)

# Mean viewers per week and game
average_viewers <- bot_games %>%
  group_by(Name, Week) %>%
  summarise(Average_Viewers = mean(`Average Viewers`, na.rm = TRUE))
```

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

```r
ggplot(average_viewers, aes(x = Week, y = Average_Viewers, color = Name)) +
  geom_line() +
  labs(title = "Evolution of the Average Number of Spectators for the Least Popular Games", x = "Week",
  theme_minimal() +
  theme(legend.position = "none")
```



Evolution of Average Viewership for Least Popular Games

This graph shows the evolution of the average number of viewers for the least popular games over the weeks. There is a general trend of steady decline for these games, indicating relative interest and a decreasing audience over time. This suggests that games at the bottom of the ranking are declining in popularity and are no longer in the hype, unlike top-ranked games which maintain strong attractiveness and prolonged hype.

Overall, this analysis underscores the importance of maintaining high-quality, engaging content and leveraging significant events to sustain and boost viewer engagement on Twitch.

Limitations of the analysis

The data collected for this project have several limitations. Firstly, using the public Twitch API may lead to restrictions in terms of request rate and data availability, which can affect the completeness of the gathered data. Additionally, removing games that fall out of the top 100 each week may introduce a bias, as temporarily popular categories are eliminated, making long-term trends difficult to track. Lastly, the analysis is limited to a specific period (from April 22 to July 11), which may not reflect seasonal trends or longer-term variations in viewing behaviors on Twitch.

Conclusion

This comprehensive analysis of Twitch viewership from April 22 to July 11 provides valuable insights into the dynamics of viewer engagement across different game categories. Through a detailed examination of the data, several key findings emerge:

1. **Significant Changes in Category Rankings**:
    - Notable spikes in rankings, such as those for "Marvel Snap" and "Sea of Thieves," are closely linked to specific events like new DLC releases or major tournaments. These events cause abrupt changes in viewer interest, indicating that timely content updates are crucial for maintaining high engagement levels.
2. **Viewer Engagement Trends for Top Games**:
    - Consistent viewer engagement for games like "Grand Theft Auto V" (GTA V) highlights the effectiveness of regular content updates and special events in sustaining audience interest. The periodic spikes for "ELDEN RING" during new DLC releases underscore the importance of fresh content in attracting viewers.
3. **Correlation Between Average Viewers and Hours Watched**:
    - A strong linear relationship between average viewers and total hours watched suggests that higher viewership naturally leads to more hours watched, given consistent streaming durations. This predictable pattern highlights the mathematical relationship between these two metrics in live streaming.
4. **Viewer Engagement Stability Among Top Games**:
    - Top games like "Just Chatting," "Grand Theft Auto V," and "League of Legends" show consistent viewer engagement, with occasional spikes during major events. This stability indicates these games' strong and lasting appeal.
5. **Evolution of Average Viewership for Least Popular Games**:
    - The steady decline in viewership for the least popular games suggests that these games struggle to retain their audience over time. This trend highlights the challenge for lower-ranked games to maintain viewer interest, unlike top-ranked games that consistently engage audiences.

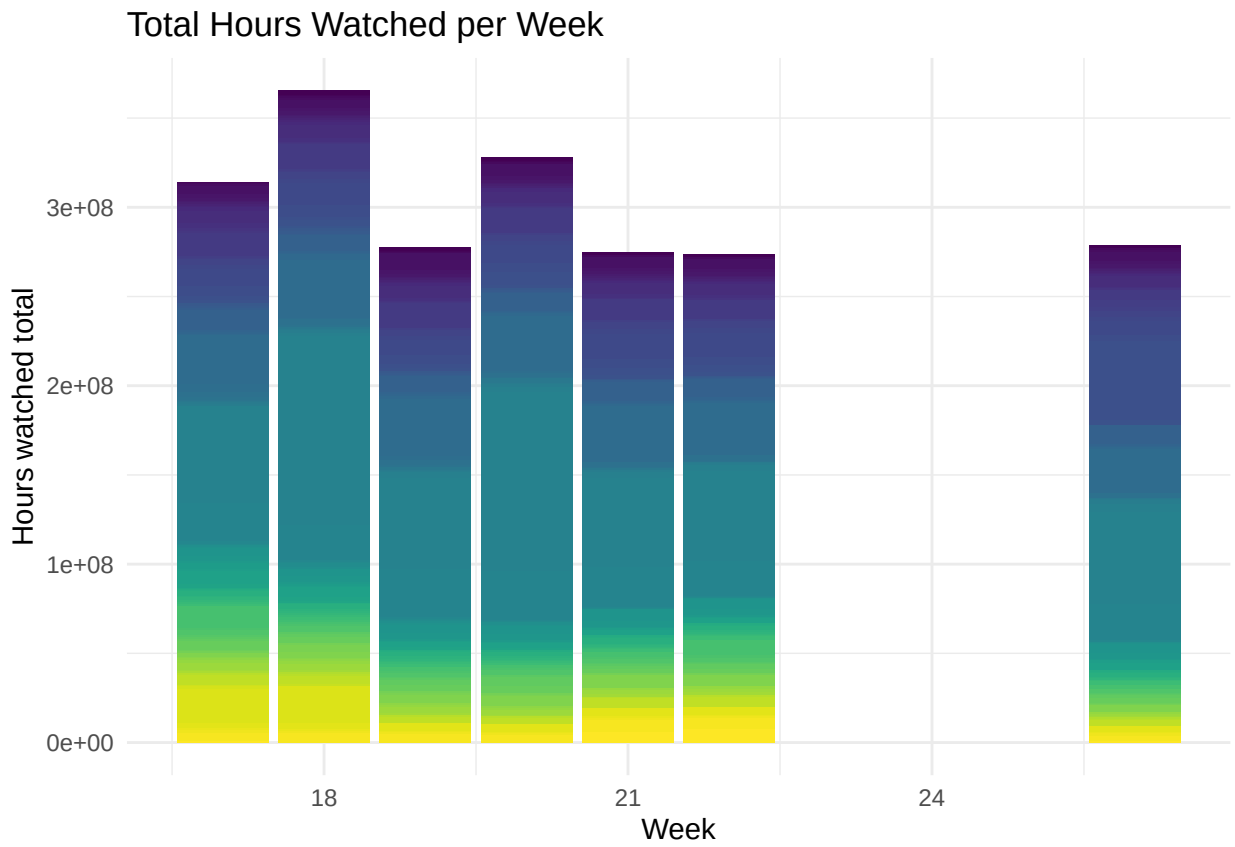
Overall, this analysis underscores the dynamic nature of viewer engagement on Twitch, driven by consistent content quality, strategic event planning, and the ability to adapt to audience preferences. These insights can guide content creators and platform managers in optimizing their strategies to enhance viewer retention and satisfaction.

Appendices

The first section contains additional graphs related to the datasets. Following that is the algorithm used for data retrieval.

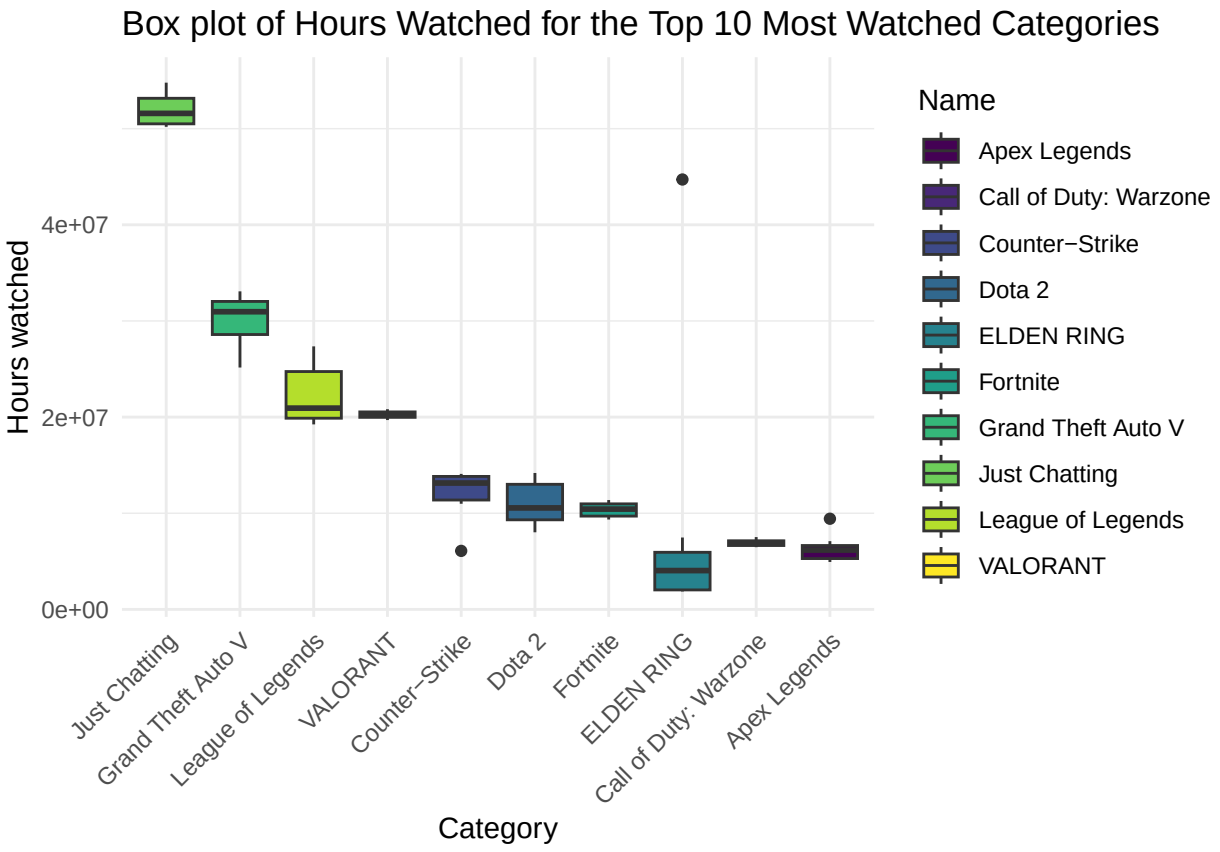
These graphs were created for personal enjoyment but are included in the document as appendices to provide comprehensive context to the reader.

```
ggplot(total_hours_by_game, aes(x = Week, y = Total_Hours_Watched, fill = Name)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Hours Watched per Week", x = "Week", y = "Hours watched total") +
  scale_fill_viridis_d() +
  theme_minimal() +
  theme(legend.position = "none")
```




```
top_categories <- all_data %>%
  group_by(Name) %>%
  summarise(Total_Hours_Watched = sum(`Hours Watched`, na.rm = TRUE)) %>%
  arrange(desc(Total_Hours_Watched)) %>%
  top_n(10, Total_Hours_Watched)

top_categories_data <- all_data %>%
  filter(Name %in% top_categories$Name)

ggplot(top_categories_data, aes(x = reorder(Name, -`Hours Watched`), y = `Hours Watched`, fill = Name)) +
  geom_boxplot() +
```

```
  labs(title = "Box plot of Hours Watched for the Top 10 Most Watched Categories",
       x = "Category",
       y = "Hours watched") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_viridis_d()
```



Algorithm to Retrieve Information for a List of Game IDs

Below is the main script used to retrieve all my data. There are other auxiliary scripts not included here that were used to obtain game category IDs and their names.

```
import json
import os
import csv
from time import sleep
import requests
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()

# Twitch API endpoints
url_twitch = "https://api.twitch.tv/helix/games"
```

```
url_twitchtracker = "https://twitchtracker.com/api/games/summary/"

# Set up headers with Twitch API credentials
headers_twitch = {
    "Client-ID": os.getenv("TWITCH_CLIENT_ID"),
    "Authorization": f"Bearer {os.getenv('TWITCH_ACCESS_TOKEN')}"
}

# Function to fetch games from Twitch API and additional data from TwitchTracker API
def fetch_and_write_data(start_id=13240):
    print("Fetching and writing data...")

    # Open the output CSV file in append mode
    with open('games_combined.csv', 'a', newline='') as csvfile:
        writer = csv.writer(csvfile)

        # Iterate over game IDs
        for game_id in range(start_id, 34282):
            params = {"id": game_id}
            sleep(0.3)

            # Fetch data from the first endpoint
            response_twitch = requests.get(url_twitch, headers=headers_twitch, params=params)
            data_twitch = response_twitch.json()

            # Check if request was successful
            if response_twitch.status_code == 200:
                games = data_twitch.get("data", [])

                # Iterate over games
                for game in games:
                    # Fetch data from the second endpoint
                    response_twitchtracker = requests.get(url_twitchtracker + str(game_id))

                    # Try to decode JSON response, handle JSONDecodeError
                    try:
                        data_twitchtracker = response_twitchtracker.json()
                    except json.decoder.JSONDecodeError:
                        print(f"No data found for game ID {game_id}")
                        continue  # Skip to the next iteration if no data is found

                    # Check if response is not empty
                    if data_twitchtracker:
                        # Extract required data
                        avg_viewers = data_twitchtracker['avg_viewers']
                        avg_channels = data_twitchtracker['avg_channels']
                        rank = data_twitchtracker['rank']
                        hours_watched = data_twitchtracker['hours_watched']

                        # Write to CSV file
                        writer.writerow([game["id"], game["name"], avg_viewers, avg_channels, rank, hou

            else:
```

```
                print(f"Error fetching games with ID {game_id}: {data_twitch.get('message', 'Unknown er

    print("Data has been successfully fetched and stored in games_combined.csv.")

# Call the function to fetch and write data
fetch_and_write_data()
```

Script Explanation:

1. **API Endpoints and Headers**:
   - `url_twitch`: Endpoint for Twitch API (`https://api.twitch.tv/helix/games`).
   - `url_twitchtracker`: Endpoint for TwitchTracker API (`https://twitchtracker.com/api/games/summary/`).
   - `headers_twitch`: Headers containing Twitch API credentials (`Client-ID` and `Authorization`).
2. **Fetch and Write Data Function (`fetch_and_write_data`)**:
   - **Function Purpose**: This function fetches game data from the Twitch API (`url_twitch`) and additional statistics from the TwitchTracker API (`url_twitchtracker`).
   - **Parameters**: `start_id` (optional) specifies the starting game ID from which data fetching begins.
   - **Data Fetching Process**:
     - Opens `games_combined.csv` in append mode for writing data.
     - For each game ID:
       * Makes a request to the Twitch API to fetch basic game information (`game_id`).
       * Checks if the request was successful (`status_code == 200`) and retrieves game data (`games`).
       * Iterates over each game and makes a request to the TwitchTracker API to fetch additional statistics (`avg_viewers`, `avg_channels`, `rank`, `hours_watched`).
       * Decodes the JSON response from TwitchTracker API and handles any potential errors (`JSONDecodeError`).
       * Writes the combined data (game ID, name, average viewers, average channels, rank, hours watched) to `games_combined.csv`.
   - **Rate Limit Handling**: Uses `time.sleep(0.3)` to introduce a delay between API requests to comply with rate limits imposed by Twitch API.

Notes:

- Ensure they are valid Twitch API credentials (`TWITCH_CLIENT_ID` and `TWITCH_ACCESS_TOKEN`) in the `.env` file.
- Adjust `start_id` and the range (`34282` in this case) based on the specific needs and available game IDs.
- Error handling (`JSONDecodeError`, API request errors) ensures robustness in data retrieval.
- This script combines data from multiple sources (`Twitch API` and `TwitchTracker API`) to provide comprehensive insights into Twitch game statistics.

By following this structure, you can efficiently gather and store detailed game information from Twitch for further analysis or reporting purposes.