Supercell Data Intern Test

27-11-2023

Tuan Tran

tuan.tranquanganh@aalto.fi

I. Data description

The dataset contains 3 tables:

- account: This table contains the accounts information. Each row of this table is an account with the associated id, the created time, device, platform, country, and the app store id.

- iap_purchase: This table registers information of the purchases made in game. Each row is a purchase with the associated account id, purchase time, package id, price, and the app store id.

- account_date_session: This table registers players sessions' information. Each row is a player who plays on a specific date with the associated account id, date, number of sessions, and the sessions' duration.

II. Daily active users analysis

1. Monthly DAU analysis

DAU (daily active users) represent the number of unique users who are active in a game during a day. This metric can help identify trends and measure the game's overall popularity. The plot below shows the DAU during the year.

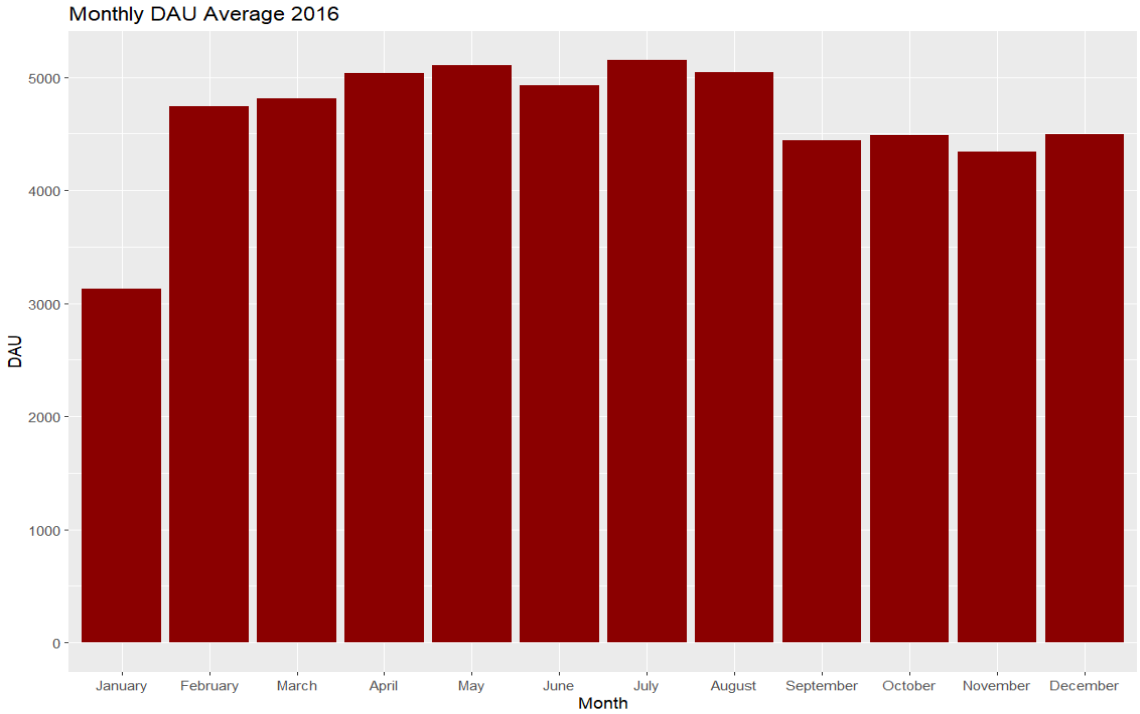

Figure 1. Monthly DAU average

DAU increases sharply in February. It has an upwards trend from January till March, peaking in May at approximately 5100 users. The user count stabilizes from April till August. It decreases from July to September, reaching around 4400 users and maintains that level till the end of the year.

The rise from January to February is rather abnormal. The sudden increase may be caused by the game introduction or increased marketing. The drop in July may be caused by the game losing attraction. It is noticed that the daily user count decreases more quickly between August and September. This could also be a result of the beginning of the school year, which lowers the players play time.

2. Seasonality

To explore short term trends and seasonality, we inspect the day by day active users count in the time series plot below.

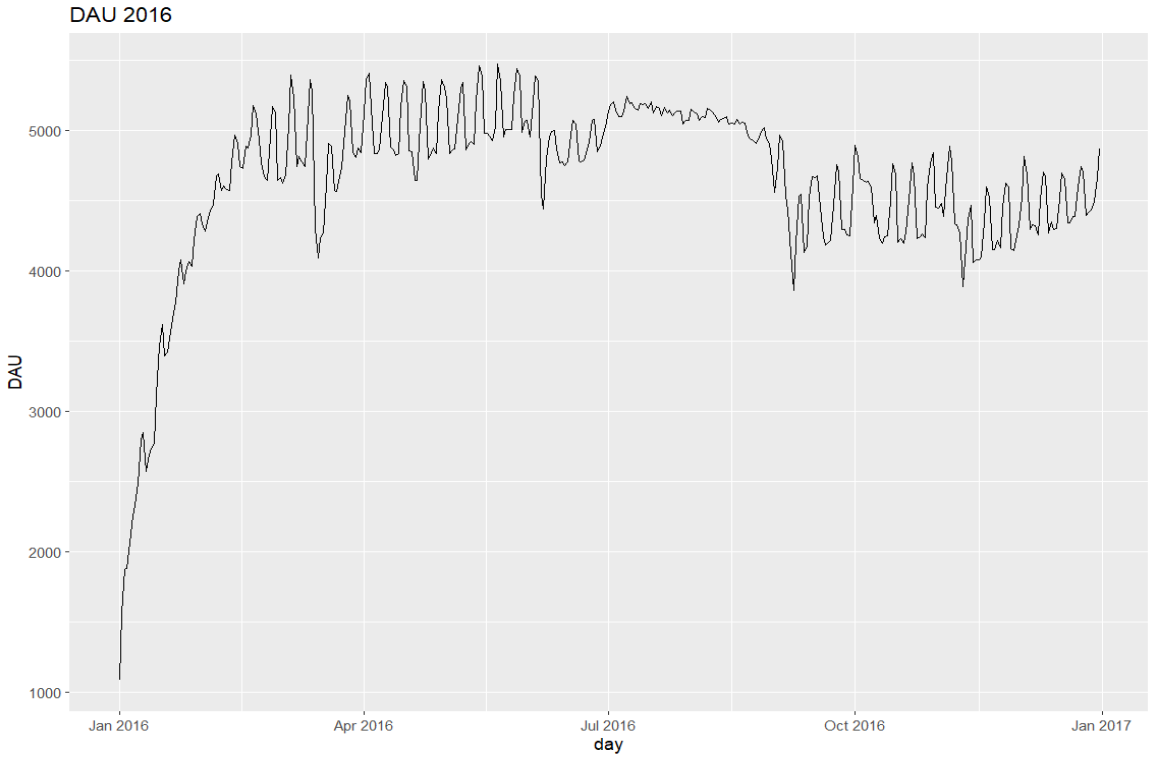

Figure 2. Daily active users

There appear to be seasonalities during certain months We next plot the DAU separately by months to explore this.

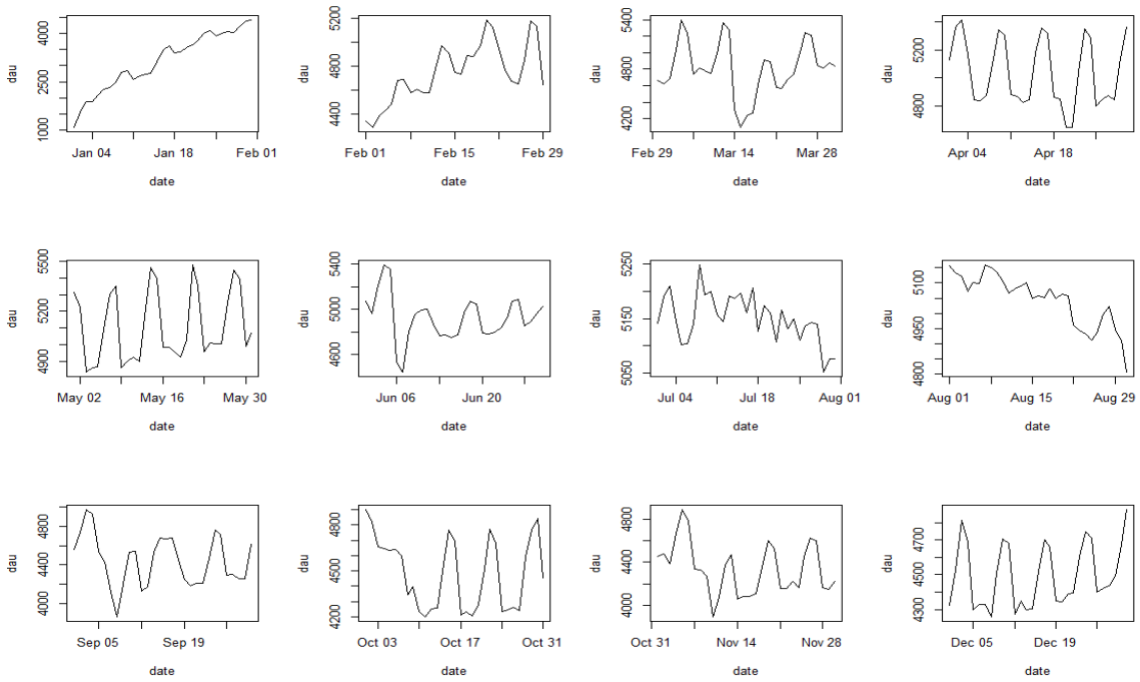

Figure 3. Plots of DAU in each month

As seen above, seasonality clearly exists in April, May, October, and December. Their durations are 7 days. We may assume that the seasonality happens on a weekly basis, caused by the increases during the weekend. The lack of seasonality from June to September can also be explained by the impact of the summer holiday, where the players play the game similarly throughout the week. To confirm this assumption, we calculate the average DAU on each weekday throughout the year.

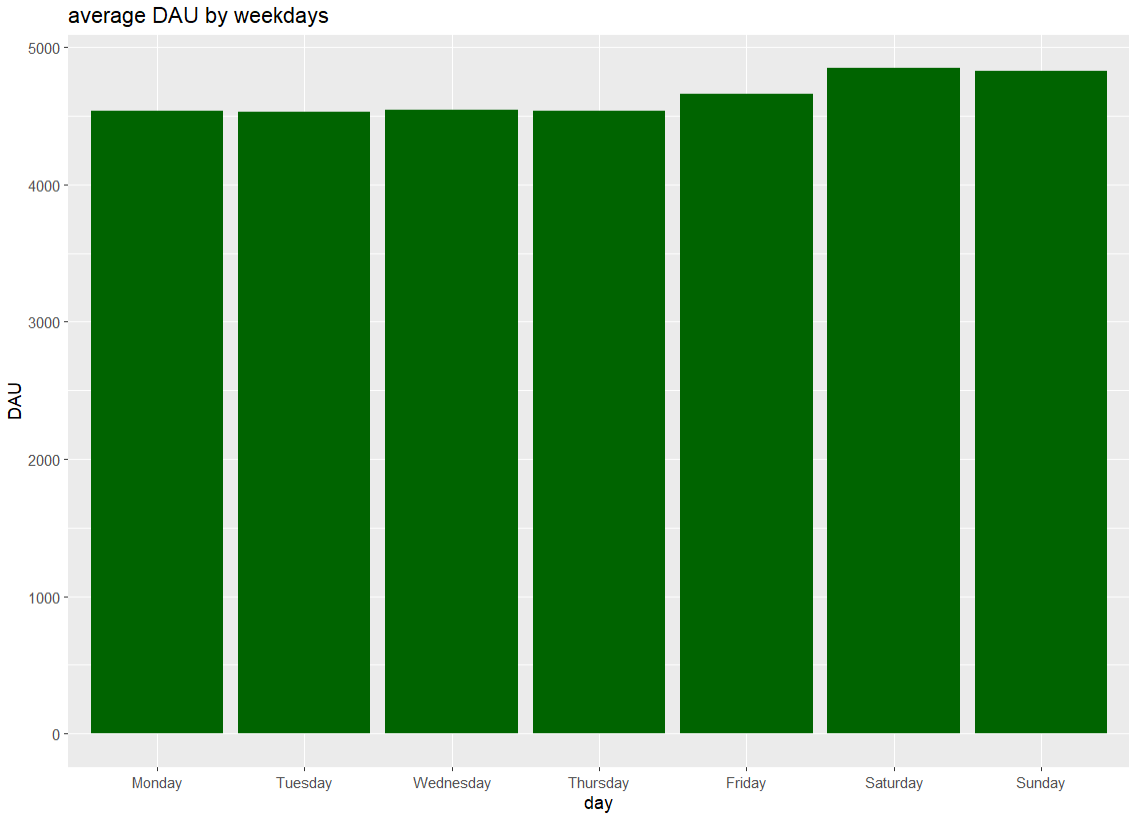

Figure 4. Average weekdays DAU year round

It appears that the weekend has the highest active users count, as expected.

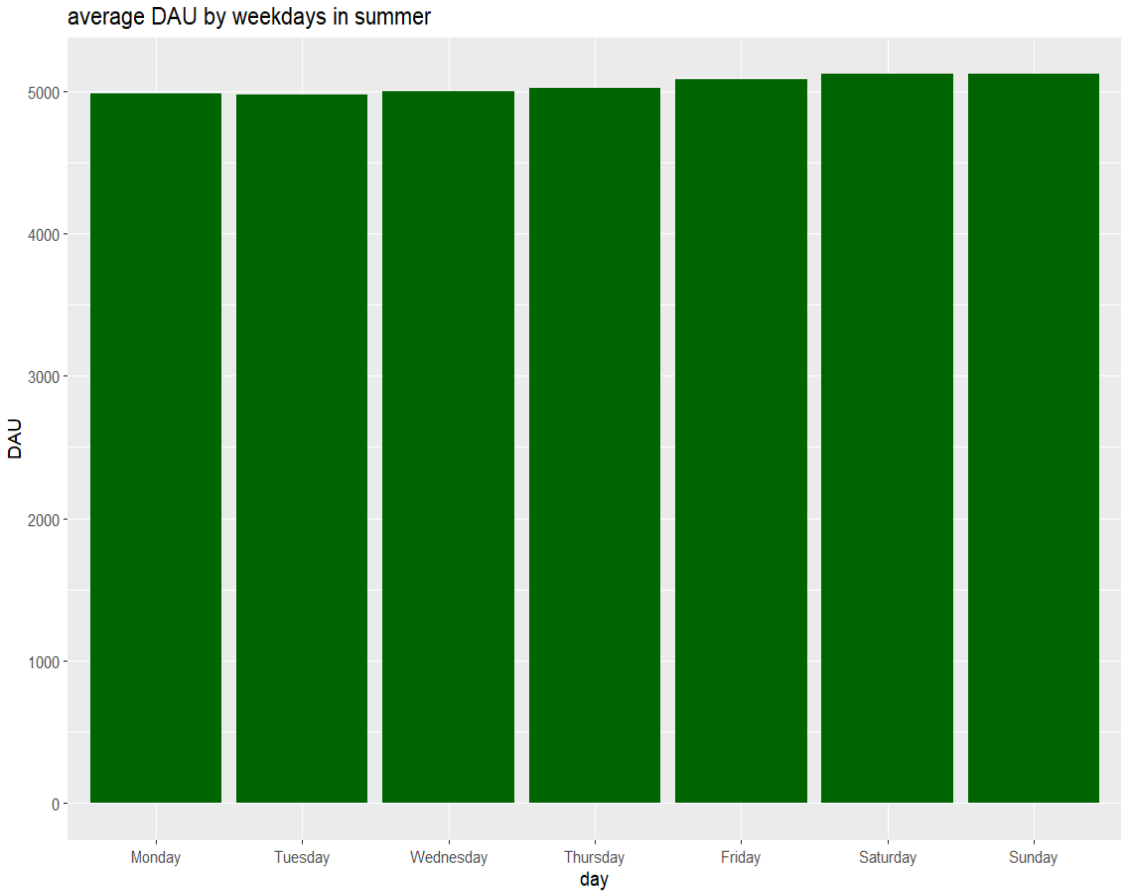

Figure 5. Average weekdays DAU on summer months

In comparison, the average DAU in the summer month has less variance during the week. This confirms our theory that the players tend to play more regularly during the summer. Understanding this disparity in user behavior at different times helps the developers plan their promotions and events more appropriately.

In summary, DAU in the year 2016 increased. However, there is a drop in July that requires further investigation. There exists weekly seasonality across the months, except for summer months

3. Extra: MAU analysis

Although the DAU can show the game overall daily usage, there is usually more to users' analytics than the number of active users. In particular, the developers would also like to know how the player base changes over time. This could be measured by the MAU (monthly active users) metrics, which count the number of unique users during a month. Considering both DAU and MAU is important to understand the players' return rate. It is safe to assume that, for free to play games, returning players are more likely to make ingame purchases, thus are more valuable customers.

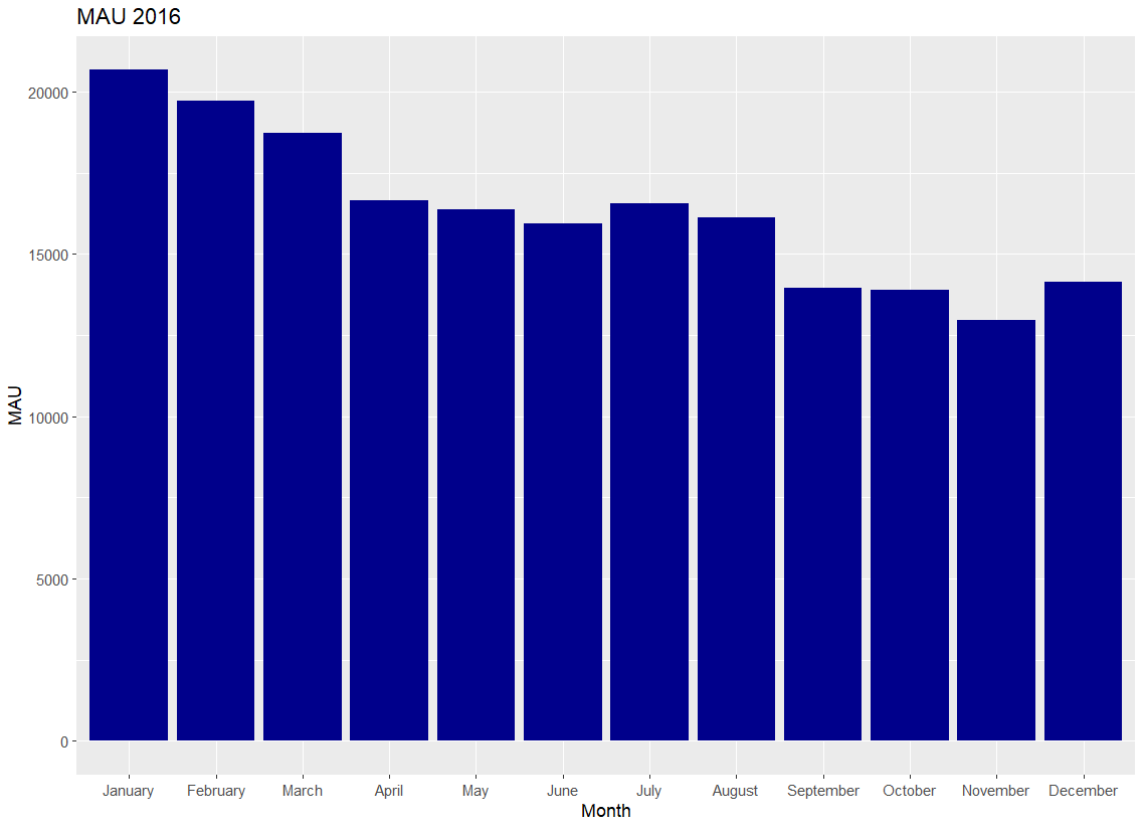

Figure 5. MAU 2016

The plot above shows the MAU of the game during 2016. It can be seen that although the number of daily unique users increases rapidly from January to May, the number of monthly unique users actually decreases. This indicates that in the beginning of the year, many players tried the game, but did not return often, or quitted. During the later months, DAU and MAU behave similarly, which indicates that the player base has stabilized. However, it is also noted that these metrics only count the number of users, without actually knowing the similarity between the player base of each period. This means that although the count may stay the same, the player base may have changed completely.

III. Sales analysis

Next, we will analyze the geographic split of users count and revenue. The total number of unique users and total revenue across countries are calculated.

NOTE: Map plots are used to visualize the global user and revenue distribution. The log scale of user count, revenue, and average revenue per user (ARPU) is used to achieve better color contrast between levels. The color "gray" represents countries with zero or NA values. The darker the color, the higher the value, and vice versa. This applies to all 3 map plots below.

1. Users geographic split

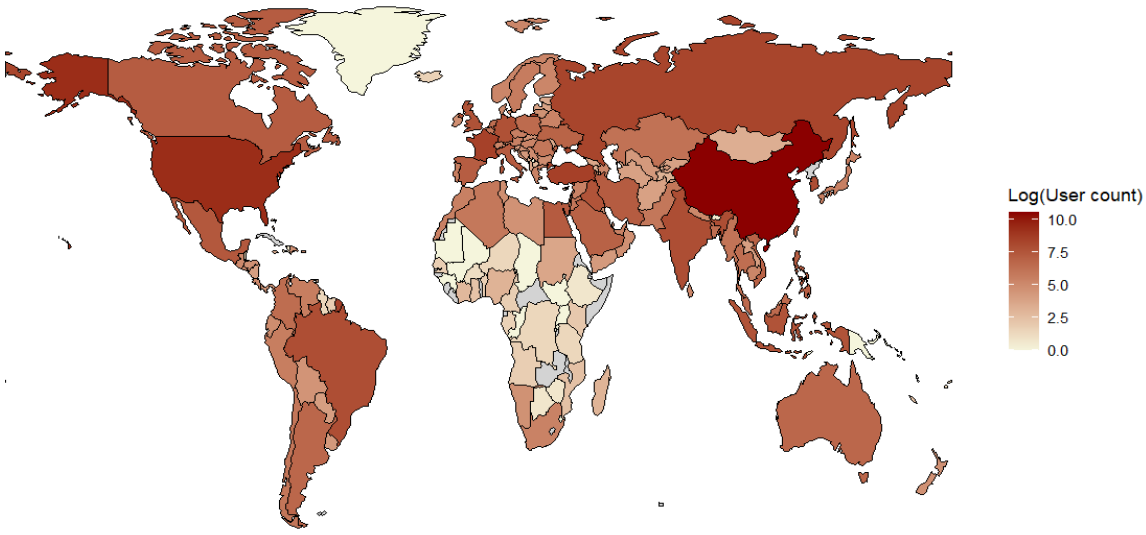

Figure 6. Global User Distribution

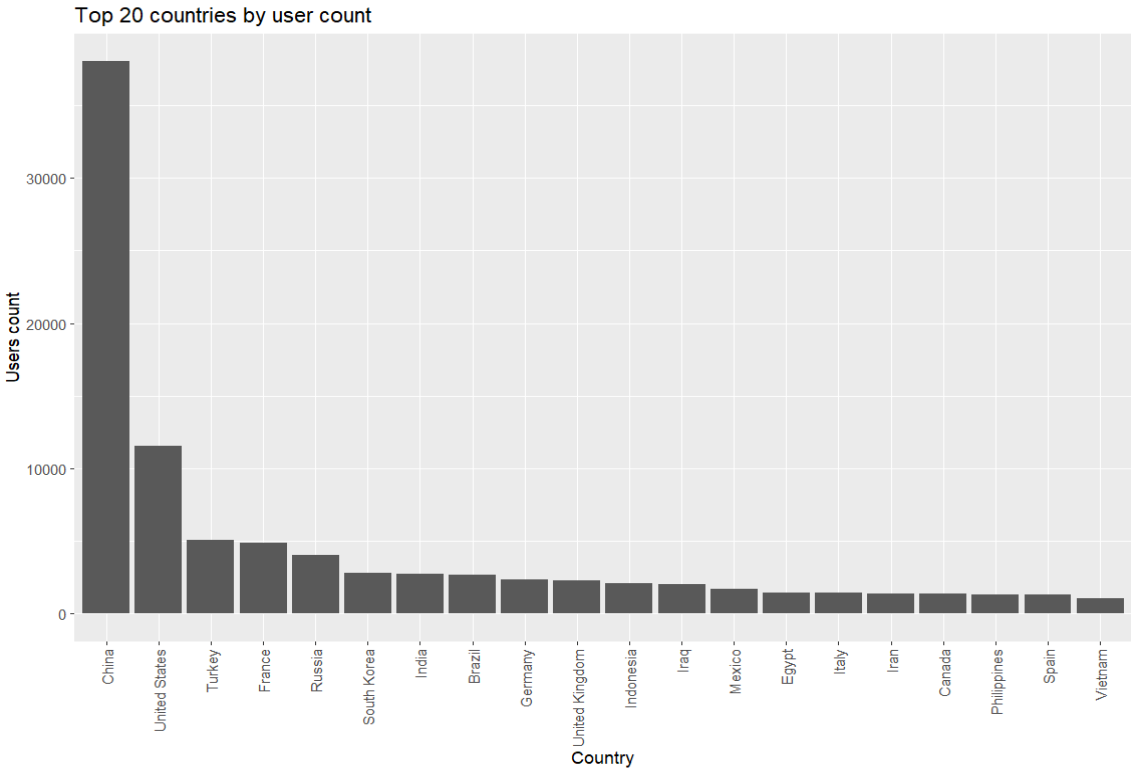

Figure 7. Top 20 Countries by User Count

As seen in figure 6, the game has large user bases in Europe, North, South America, and Asia. The user count is low or zero in most of Africa. Figure 7 shows the top 20 countries with the highest user count. The largest user base is in China, followed by the US. It can be seen that the top 20 mostly consist of countries with large populations.

2. Revenue geographic split

Next, we analyze the geographic split of revenue.

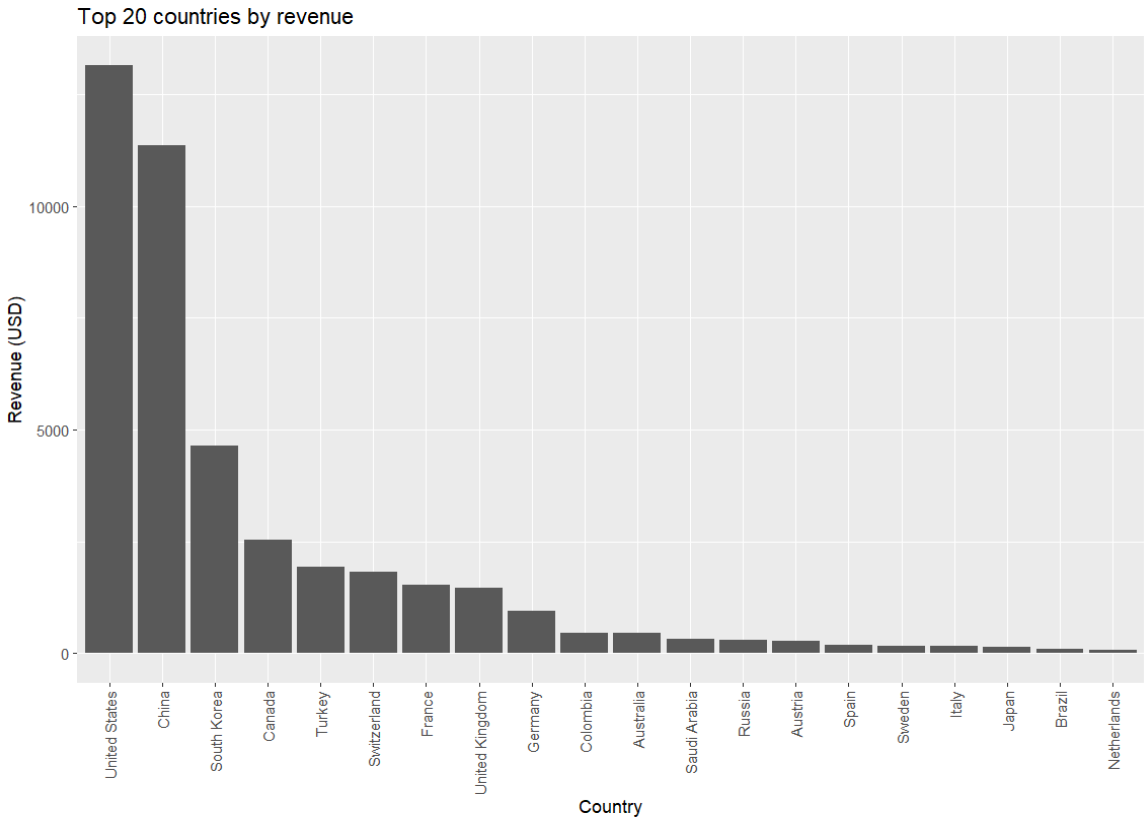

Figure 8. Top 20 Countries by Revenue

It can be seen that the top 20 countries by revenue consist mostly of developed countries in Europe, North America, and East Asia. The highest revenue comes from the US, followed by China. We continue by inspecting the global revenue distribution to understand the revenue behavior in other parts of the world.

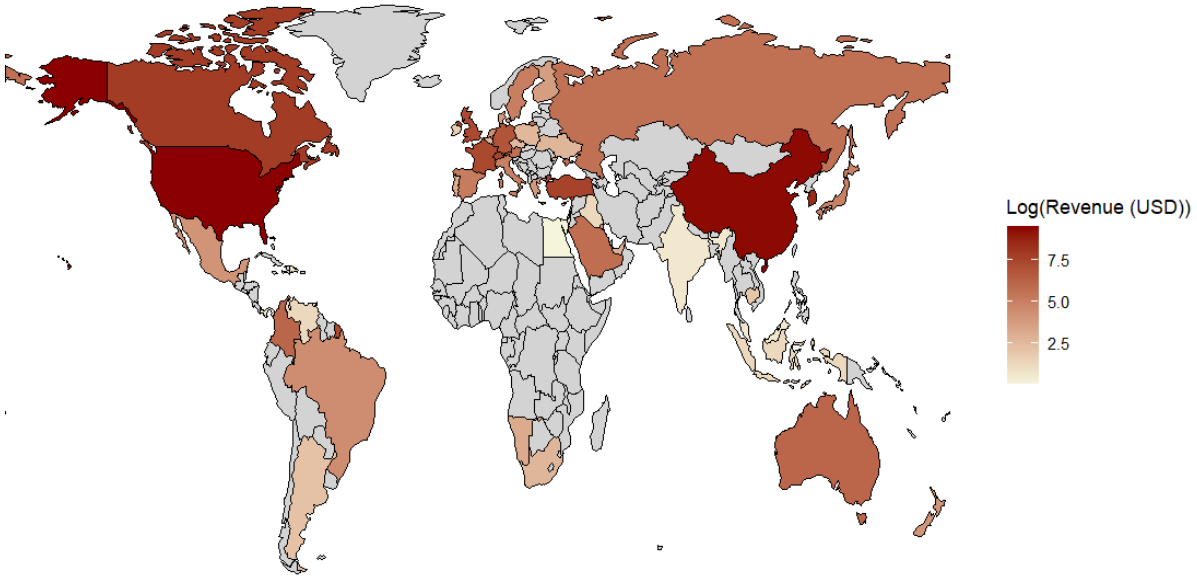

Figure 9. Global Revenue Distribution

In certain regions, although the user base is large, the revenue is low or zero, such as in South Asia, South East Asia, Eastern Europe, and in South America. It is noted that although many countries from these regions are present in the top 20 countries by user count, they have from really low to zero revenue streams. This is unwanted behavior, which shows that the game is not monetizing well in these regions. The problem can be caused by inappropriate pricing, regional payment error, or regional laws. Note that it could also be caused by economic factors. Most of these countries are developing countries which could explain the difference in purchasing behavior compared to developed countries. There are certain exceptions, such as Norway. The game generates zero revenue from users in Norway. Understanding these patterns requires further investigation into these markets.

3. **Average revenue per user**

The ARPU for each market is calculated by formula:

ARPU = Revenue / User count

The plot below shows the top 20 countries with highest ARPU.

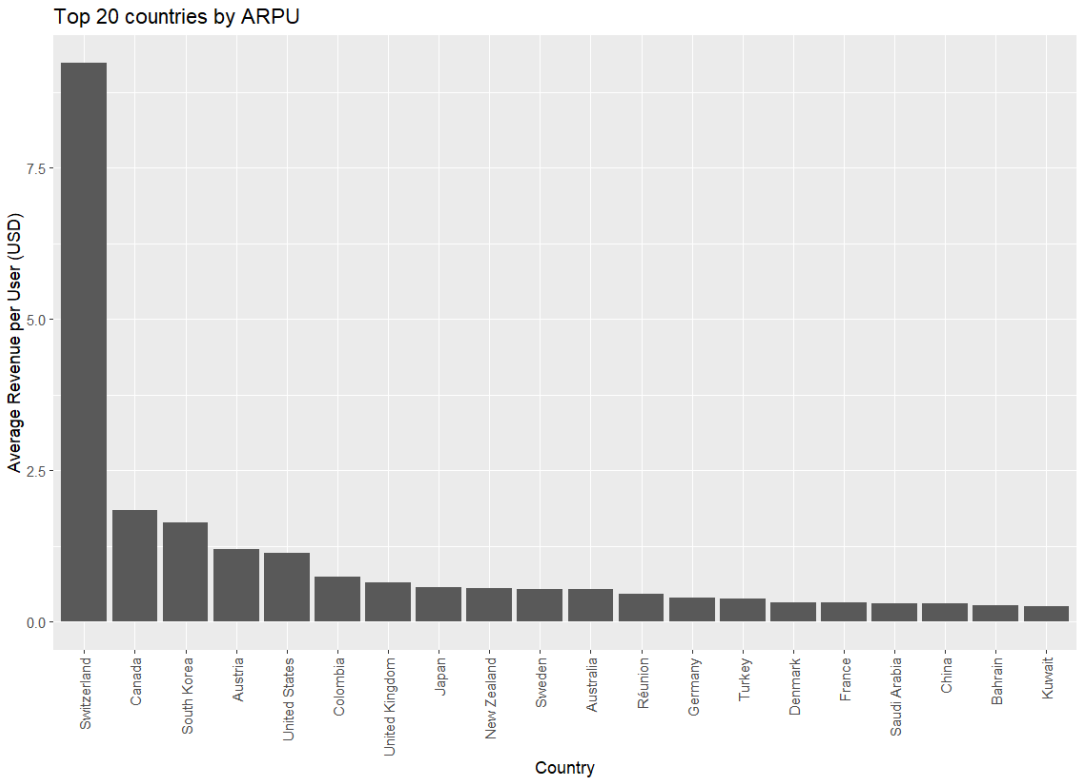

Figure 9. Top 20 countries by ARPU

Most countries in this list are highly developed countries. Switzerland has the highest ARPU, which significantly surpasses other countries. The average ARPU across all countries is approximately 0.124 USD per user, with standard deviation roughly 0.71. This is relatively low ARPU average with rather high variability. This suggests that the game's monetization ability is vastly different in different countries. This may indicate that many markets are not optimally monetized. However, economic differences across countries must also be taken into account.

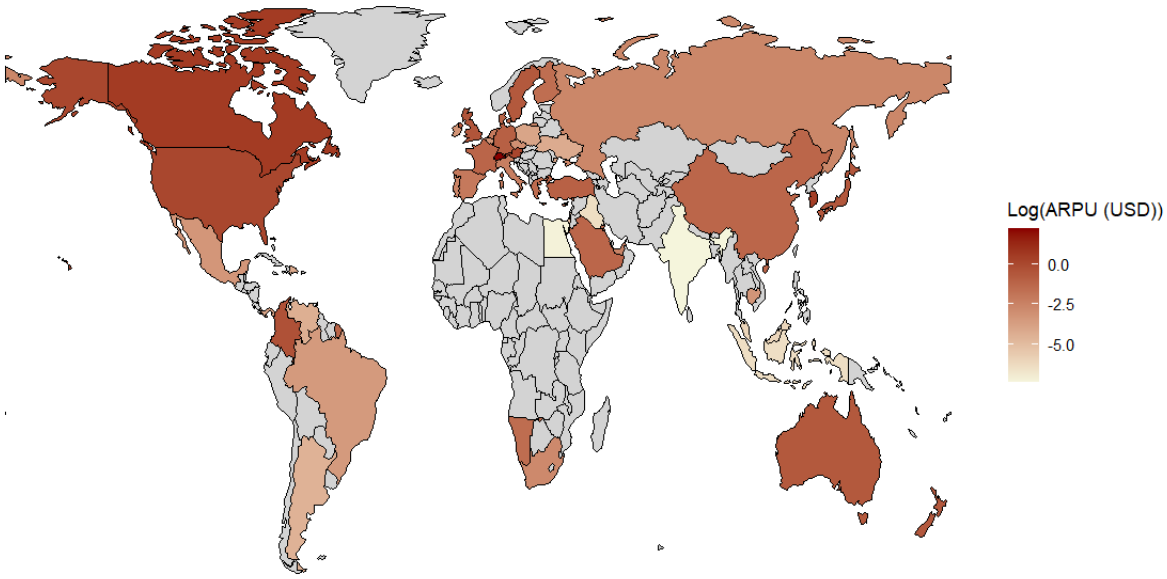

Figure 10. Global Average Revenue per User

We would like to now inspect which specific countries are not monetized well. A simple approach to this is to use ARPU. However, for countries with zero revenue, the ARPU is zero regardless of user count. As a result, we also add a small constant plus 0.01 term to avoid the rate of countries with zero revenue and varying user count having the same rate zero. The formula is:

Revenue rate = (Revenue + 0.01) / User Count

It is best to choose the smallest constant term possible to avoid affecting the result ranking. Different constant terms are tested and it shows that for terms smaller than 0.01, the result produced is the same.

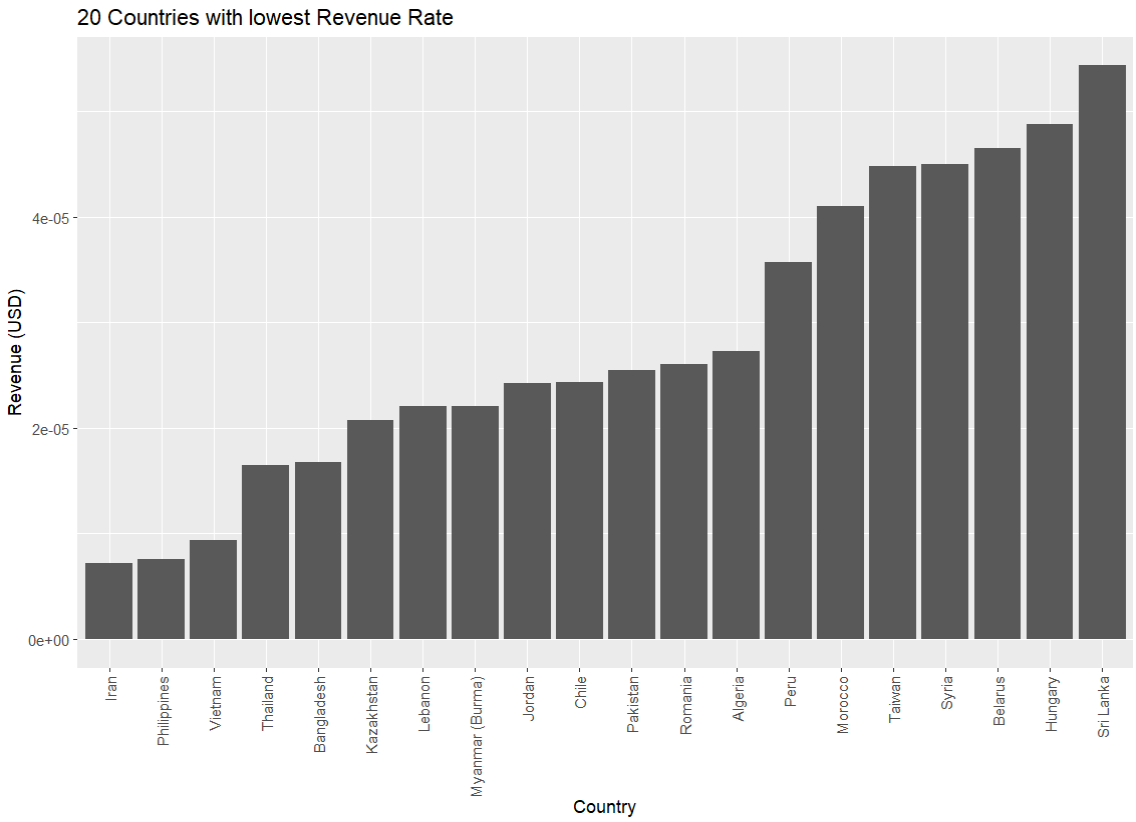

Figure 10. 20 Countries with lowest Revenue Rate

It can be seen that Iran, Philippines, and Vietnam are leading the list of lowest revenue rate while also appearing in the top user count list. Further investigation into these markets are required to improve the monetization rate of the game.

4. Conclusion

In conclusion, the game has high variability in ARPU across countries. Furthermore, most developing countries with a large player base have quite low revenue, which may point to pricing problems in these areas. This can be partly explained by the economic difference of countries. However, further research into these regions may help improve the revenue of the game.

IV. Appendix

supercell

2023-11-27

Imports

```
getwd()

#install.packages("RSQLite")
#install.packages("gridExtra")
#install.packages("countrycode")
#install.packages("rnaturalearth")
#install.packages("rnaturalearthdata")
library(RSQLite)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(countrycode)
library(rnaturalearth)
library(rnaturalearthdata)
library(sf)
```

DAU analysis

Preprocessing

```
db_file <- "sample.sqlite"

# Connect to the SQLite database
con <- dbConnect(SQLite(), dbname = db_file)


# ------------------------DAILY ACTIVE USERS ANALYSIS----------------------------

# Count the unique account id on each date and return a dataframe with 2 columns
# date and DAU
dau_df <- dbGetQuery(con, "SELECT
                      COUNT(distinct account_id) AS dau,
                      DATE
                      FROM account_date_session
                      group by date
                      Order by date;")

# Create columns weekday and month for grouping dau_df
dau_df <- dau_df %>%
```

```
  mutate(
    date = as.Date(date),
    weekday = weekdays(date),
    month = months(date))

# Create a df for average monthly DAU, contains 12 averages for each month
average_dau_df <- dau_df %>%
  group_by(month) %>%
  summarize(avg_dau= mean(dau), ) %>%
  arrange(factor(month, levels = month.name))

average_dau_df$changes <- c(0,diff(average_dau_df$avg_dau)) /
                          lag(average_dau_df$avg_dau) * 100
average_dau_df$changes[is.na(average_dau_df$changes)] <- 0
# Count the number of unique ids in each month and returns a dataframe with 2
# columns
# MAU: monthly active users
# month: months in the year
mau_df <- dbGetQuery(con, "SELECT
                      COUNT(distinct account_id) AS mau,
                      SUBSTR(date,1,7) as month
                      FROM account_date_session
                      group by month
                      Order by month;")

#Use the average_dau_df's month column as months for mau_df
mau_df$month <- average_dau_df$month

#Calculate DAU/MAU ratio as dm_ratio column in mau_df
mau_df$dm_ratio <- average_dau_df$avg_dau / mau_df$mau

#Rearrange the months for visualization
mau_df <- mau_df %>% arrange(factor(month, levels = month.name))


#Create a string vector of days in the week
custom_weekday_order <- c("Monday", "Tuesday", "Wednesday", "Thursday",
                          "Friday", "Saturday", "Sunday")

#Create a df where each row is a weekday with columns:
# avg_dau: average DAU on a weekday throughout the year
# weekday: day in the week
weekday_dau_df <- dau_df %>%
  group_by(weekday) %>%
  summarize(avg_dau= mean(dau), ) %>%
  arrange(factor(weekday, levels = custom_weekday_order))

weekday_dau_summer <-
  dau_df[dau_df$month %in% c("June", "July", "August"), ] %>%
  group_by(weekday) %>%
  summarize(avg_dau= mean(dau), ) %>%
  arrange(factor(weekday, levels = custom_weekday_order))
```

Plots

```
#Plotting the DAU
ggplot(dau_df, aes(x = date, y = dau)) +
  geom_line(stat = "identity") +
  labs(x = "day", y = "DAU") +
  ggtitle("DAU 2016")
```

```
#Plotting monthly average DAU
ggplot(average_dau_df, aes(x = factor(month, levels = month.name),
                           y = avg_dau)) +
  geom_bar(stat = "identity", fill = "darkred") +
  labs(x = "Month", y = "DAU") +
  ggtitle("Monthly DAU Average 2016")
```

```
#Plotting DAU trendline for each month to explore seasonality
layout(matrix(1:12, nrow = 3, byrow = TRUE))
for (m in average_dau_df$month) {
  dt <- dau_df %>%
    filter(month == m)
  plot(dt$date,dt$dau, type = "l",xlab = "date",ylab="dau")
}
```

```
layout(matrix(1, 1))

#Plotting weekday average DAU
ggplot(weekday_dau_df, aes(x = factor(weekday, levels = custom_weekday_order),
                           y = avg_dau)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  labs(x = "day", y = "DAU") +
  ggtitle("average DAU by weekdays")
```

```
ggplot(weekday_dau_summer,
       aes(x = factor(weekday, levels = custom_weekday_order),
                           y = avg_dau)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  labs(x = "day", y = "DAU") +
  ggtitle("average DAU by weekdays in summer")
```

```
var(weekday_dau_df$avg_dau)
```

```
## [1] 20292.65
```

```
var(weekday_dau_summer$avg_dau)
```

```
## [1] 4061.995
```

```
#Plotting MAU
ggplot(mau_df, aes(x = factor(month, levels = month.name), y = mau)) +
  geom_bar(stat = "identity", fill = "darkblue") +
  labs(x = "Month", y = "MAU") +
  ggtitle("MAU 2016")
```

```r
#Plotting DAU/MAU ratio
ggplot(mau_df, aes(x = factor(month, levels = month.name), y = dm_ratio)) +
  geom_bar(stat = "identity", fill = "darkorange") +
  labs(x = "Month", y = "DAU/MAU") +
  ggtitle("DAU/MAU ratio 2016 ")
```

Sales analysis

Preprocessing

```r
# ------------------------------SALES ANALYSIS----------------------------------

# Query and preprocessing

region_users_count <- dbGetQuery(con, "SELECT
                  COUNT(account_id) as user_counts, country_code
                  FROM account
                  GROUP by country_code
                  ORDER by user_counts desc;")
region_revenue <- dbGetQuery(con, "SELECT
                                  Sum(ip.iap_price_usd_cents) as revenue,
                                      a.country_code
                                  FROM iap_purchase ip JOIN account a
                                  ON ip.account_id = a.account_id
                                  GROUP BY a.country_code
                                  ORDER BY revenue DESC;")

users_revenue_df <- merge(region_revenue, region_users_count,
                      by.x = "country_code", by.y = "country_code", all = TRUE)
revenue_na_indices <- which(is.na(users_revenue_df[["revenue"]]))
users_revenue_df[revenue_na_indices,"revenue"] <- 0
users_revenue_df <- users_revenue_df %>%
  mutate(country = countrycode(country_code, "iso2c", "country.name"))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `country = countrycode(country_code, "iso2c", "country.name")`.
## Caused by warning:
## ! Some values were not matched unambiguously: XK
```

```r
row_to_fix <- which(users_revenue_df[["country_code"]]=="XK")
users_revenue_df[row_to_fix,"country"] = "Kosovo"

rows_with_null <- which(is.na(users_revenue_df[["country_code"]]))
users_revenue_df[rows_with_null,"country"] = "Unknown"
users_revenue_df[rows_with_null,"country_code"] = "NA"

# Convert from 1 cents unit to 1 USD
users_revenue_df[["revenue"]] <- users_revenue_df[["revenue"]] / 100
```

```r
#Calculate ARPU
users_revenue_df$avg_revenue <- users_revenue_df$revenue /
                                users_revenue_df$user_counts

# Calculate average ARPU and standard deviation
mean(users_revenue_df$avg_revenue)
```

```
## [1] 0.1238976
```

```r
sd(users_revenue_df$avg_revenue)
```

```
## [1] 0.7075529
```

```r
#Calculate revenue rate
users_revenue_df$revenue_rate <- (users_revenue_df$revenue + 0.01) /
                                     users_revenue_df$user_counts


world_map <- ne_countries(scale = "small", returnclass = "sf")
world_map$iso_a2[world_map$iso_a2 == -99] <- NA
world_map$iso_a2_eh[world_map$iso_a2_eh == -99] <- NA

world_map$country_code <- coalesce(world_map$iso_a2_eh, world_map$iso_a2)
world_map$country_code[world_map$name == "N. Cyprus"] <- "CY"
world_map_data <- merge(world_map, users_revenue_df,
                    by.x = "country_code", by.y = "country_code", all.x = TRUE)

world_map_data <- world_map_data[world_map_data$name != "Antarctica", ]
#Use log scale for better color division in the map plots
world_map_data$log_user_counts <- log(world_map_data$user_counts)
world_map_data$log_revenue <- log(world_map_data$revenue)
```

Plots

```r
# Plotting
# Users count

#Map plot of user count
ggplot() +
  geom_sf(data = world_map_data, aes(fill = log_user_counts), color = "black") +
  scale_fill_gradient(low = "beige", high = "darkred", na.value = "lightgray",
                      name = "Log(User count)") +
  theme_void() +
  theme(
    aspect.ratio = 0.6,
    plot.margin = margin(0, 5, 0, 0)
  )
```

```r
#Bar plot of top 20 by user count
ggplot(head(users_revenue_df[order(users_revenue_df$user_counts,
                                   decreasing = TRUE),],20),
       aes(x = reorder(country,- user_counts), y = user_counts )) +
  geom_bar(stat = "identity") +
  labs(title = "Top 20 countries by user count",
       x = "Country",
       y = "Users count") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```r
# Revenue

#Map plot
ggplot() +
  geom_sf(data = world_map_data, aes(fill = log_revenue), color = "black") +
  scale_fill_gradient(low = "beige", high = "darkred", na.value = "lightgray",
                      name = "Log(Revenue (USD)) ")+
  theme_void() +
  theme(
    aspect.ratio = 0.6,
    plot.margin = margin(0, 5, 0, 0)
  )
```

```r
#Bar plot of top 20 by revenue
ggplot(head(users_revenue_df[order(users_revenue_df$revenue,
                                   decreasing = TRUE),],20),
       aes(x = reorder(country,- revenue), y = revenue )) +
  geom_bar(stat = "identity") +
  labs(title = "Top 20 countries by revenue",
       x = "Country",
       y = "Revenue (USD)") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```r
# Average revenue per user per market

#Map plot of ARPU
world_map_data$log_avg_revenue <- log(world_map_data$avg_revenue)
ggplot() +
  geom_sf(data = world_map_data, aes(fill = log_avg_revenue), color = "black") +
  scale_fill_gradient(low = "beige", high = "darkred", na.value = "lightgray",
                      name = "Log(ARPU (USD)) ")+
  theme_void()+
  theme(
    aspect.ratio = 0.6,
    plot.margin = margin(0, 5, 0, 0)
  )
```

```r
#Bar plot of top 20 by ARPU
ggplot(head(users_revenue_df[order(users_revenue_df$avg_revenue,
                                   decreasing = TRUE),],20),
       aes(x = reorder(country,- avg_revenue), y = avg_revenue )) +
  geom_bar(stat = "identity") +
  labs(title = "Top 20 countries by ARPU",
```

```
       x = "Country",
       y = "Average Revenue per User (USD)") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
#Bar plot of top 20 lowest revenue rate
ggplot(head(users_revenue_df[order(users_revenue_df$revenue_rate),],20),
       aes(x = reorder(country,revenue_rate), y = revenue_rate)) +
  geom_bar(stat = "identity") +
  labs(title = "20 Countries with lowest Revenue Rate",
       x = "Country",
       y = "Revenue Rate (USD)") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```