Steam Exploring Project

Mira Tang

12/10/2018

Data Pre-processing

Scraping Raw Data

ATTENTION: The result of following code may change through seconds. In order to keep the reproducibility of this project, saving all data as csv files and DON'T run it(set "eval = FALSE").

```
# Scraping the data of current players from Steam Charts
## Specifying the url
url <- "https://store.steampowered.com/stats/"
## Reading the HTML
webpage <- read_html(url)
currentplayers_1 <- html_nodes(webpage, "td:nth-child(1) .currentServers")
currentplayers_1 <- html_text(currentplayers_1)
currentplayers_1 <- gsub(",", "", currentplayers_1)
currentplayers_1 <- as.data.frame(as.numeric(currentplayers_1))

currentplayers_2 <- html_nodes(webpage, "td+ td .currentServers")
currentplayers_2 <- html_text(currentplayers_2)
currentplayers_2 <- gsub(",", "", currentplayers_2)
currentplayers_2 <- as.data.frame(as.numeric(currentplayers_2))

currentplayers_3 <- html_nodes(webpage, ".gameLink")
currentplayers_3 <- html_text(currentplayers_3)
currentplayers_3 <- as.data.frame(currentplayers_3)

currentplayers <- cbind(currentplayers_3, currentplayers_1, currentplayers_2)
rm(currentplayers_1, currentplayers_2, currentplayers_3)
## Change the names of variables
colnames(currentplayers) <- c("Game", "Current.Players", "Peak.Today")
## Save data
write.csv(currentplayers, file = "Raw-Data/currentplayers.csv")
rm(url, wrbpage, currentplayers)

# Scraping the appid from SteamDB
left <- "https://steamdb.info/apps/page"
page <- 1:1400
url <- str_c(left, page, "/")

get_appid <- function(html) {
  appid <- html %>%
    html_nodes(".applogo+ td a") %>%
    html_text()
  return(appid)
}
```

```r
get_name <- function(html) {
  name <- html %>%
    html_nodes(".muted , .b") %>%
    html_text()
  return(name)
}

AppID <- data.frame()
Sys.time()
for (i in 1:length(url)) {
  html <- read_html(url[i])
  appid[(1 + 80 * (i - 1)):(80 + 80 * (i - 1))] <- get_appid(html)
  name[(1 + 80 * (i - 1)):(80 + 80 * (i - 1))] <- get_name(html)
}
Sys.time()
AppID <- as.data.frame(cbind(appid, name))
## Save data
write.csv(AppID, file = "Raw-Data/AppID.csv")

# Scraping the price data of each app from SteamDB
Current_players <- read.csv("Raw-Data/currentplayers.csv")
Current_players_top30 <- Current_players[1:30, ]
## In order to keep the accuracy of following scraping, manually input the appid of top30
Current_players_top30$AppID <- c(
  578080, 570, 730, 359550, 359550, 230410, 238960, 582010, 440, 622590,
  582010, 346110, 252950, 8930, 289070, 755790, 281990, 4000, 872790, 381210,
  105600, 218620, 377160, 304930, 594570, 841370, 221100, 251570, 812140, 413150
)

url <- "https://store.steampowered.com/app/578080/"

get_currency <- function(html) {
  Currency <- html %>%
    html_nodes(".price-line") %>%
    html_text()
  return(Currency)
}

get_currentprice <- function(html) {
  CurrentPrice <- html %>%
    html_nodes(".price-line+ td") %>%
    html_text()
  return(CurrentPrice)
}

get_covtertedprice <- function(html) {
  CovtertedPrice <- html %>%
    html_nodes("td.table-prices-converted:nth-child(3)") %>%
    html_text()
  return(CurrentPrice)
}

get_compare <- function(html) {
```

```
  Compare <- html %>%
    html_nodes(".table-prices-converted+ .table-prices-converted") %>%
    html_text()
  return(CurrentPrice)
}

get_lowestrecordedprice <- function(html) {
  LowestRecordedPrice <- html %>%
    html_nodes(".price-discount-minor+ td , .owned+ tr .table-prices-converted+ td ,
               .text-center+ td , .price-discount+ td , .price-discount-major+ td") %>%
    html_text()
  return(LowestRecordedPrice)
}

url <- "https://steamdb.info/app/578080/"
html <- read_html(url)

Currency <- html_text(html_nodes(html, ".price-line"))

CurrentPrice <- html_text(html_nodes(html, ".price-line+ td"))

ConvtertedPrice_raw <- html_text(html_nodes(html,
                                            "td.table-prices-converted:nth-child(3)"))
ConvtertedPrice <- NULL
ConvtertedPrice[1] <- CurrentPrice[1]
ConvtertedPrice[2:41] <- ConvtertedPrice_raw[1:40]
rm(ConvtertedPrice_raw)

Compare_raw <- html_text(html_nodes(html, ".table-prices-converted+
                                    .table-prices-converted"))
Compare <- NULL
Compare[c(1, 41)] <- 0
Compare[2:40] <- Compare_raw[1:39]
rm(Compare_raw)

LowestRecordedPrice_raw <- html_text(html_nodes(html, ".price-discount-major+ td ,
                                                .price-discount+ td , .text-center+ td ,
                                                .owned+ tr .table-prices-converted+ td ,
                                                .price-discount-minor+ td"))
LowestRecordedPrice[1] <- LowestRecordedPrice_raw[1]
LowestRecordedPrice[2:40] <- LowestRecordedPrice_raw[3:41]
LowestRecordedPrice[41] <- "$10.49"
rm(LowestRecordedPrice_raw)

StorePrices_PUBG <- data.frame()
StorePrices_PUBG <- as.data.frame(cbind(Currency, CurrentPrice, ConvtertedPrice,
                                        Compare, LowestRecordedPrice))
write.csv(StorePrices_PUBG, file = "Raw-Data/StorePrices_PUBG.csv")

# Scraping top10 reviews from top30 games based on play time
# Some games have limitation of age thus can not open the url without logging in
AppID <- c(578080,570,730,359550,230410,238960,582010,440,
           346110,252950,8930,289070,755790,281990,4000,872790,381210,
```

```
          105600,218620,377160,304930,594570,841370,221100,251570,812140,413150)

left <- "https://steamcommunity.com/app/"
id <- AppID
url <- str_c(left, id, "/reviews/?browsefilter=toprated&snr=1_5_100010_")

Review <- matrix(nrow = 10, ncol = 27)
for(i in 1:27){
  html <- read_html(url[i])
  Review_html <- html_nodes(html, ".apphub_CardTextContent")
  Review[c(1:length(html_text(Review_html))),i] <-  html_text(Review_html)
}
Review <- as.data.frame(Review)

raw.text <- matrix(" ", nrow =27, ncol = 1)
for(i in 1:27){
  for(j in 1:10){
    raw.text[i] <- paste(raw.text[i],Review[j,i])
  }
}
raw.text <- as.data.frame(raw.text)
colnames(raw.text) <- "text"
raw.text$text <- as.character(raw.text$text)
write.csv(raw.text, file = "Raw-Data/raw.text.csv")
```

Import datasets

```
Steam_sales <- read.csv("Raw-Data/Steam sales.csv")
Steam_playtime <- read.csv("Raw-Data/Steam top by playtime.csv")
Current_players <- read.csv("Raw-Data/currentplayers.csv")
StorePrices_PUBG <- read.csv("Raw-Data/StorePrices_PUBG.csv")
raw.text <- read.csv("Raw-Data/raw.text.csv")
```

Data Cleaning

```
# Steam
colnames(Steam_sales)[1] <- "Sales.rank"
colnames(Steam_playtime)[1] <- "Playtime.rank"
colnames(Current_players)[1] <- "CurrentPlayers.rank"

## Merging datasets
Steam <- merge(Steam_playtime, Steam_sales, by = "Game") # Free games don't have sale data
Steam <- merge(Steam, Current_players, by = "Game")

ScoreRank_raw <- unlist(strsplit(
  unlist(strsplit(
    unlist(strsplit(
      unlist(strsplit(
        as.character(Steam$Score.rank.Userscore...Metascore.), "[%]"
      )), "[(]"
```

```
    )), "[)]"
  )), "[/]"
))
ScoreRank_raw <- na.omit(as.data.frame(as.numeric(ScoreRank_raw)))
ScoreRank <- NULL
ScoreRank[c(4 * 3, 16 * 3, 19 * 3, 23 * 3, 34 * 3, 38 * 3, 39 * 3, 44 * 3)] <- NA
ScoreRank[c(1:11, 13:47, 49:56, 58:101, 103:113, 115:116, 118:131)] <- ScoreRank_raw[1:124, 1]

PlayTime <- unlist(strsplit(
  unlist(strsplit(
    as.character(Steam$Playtime..Median.), "[(]"
  )), "[)]"
))

Discount <- unlist(strsplit(
  unlist(strsplit(
    as.character(Steam$Max.discount), "[(]"
  )), "[)]"
))

for (i in 1:44) {
  Steam$ScoreRank[i] <- ScoreRank[1 + 3 * (i - 1)]
  Steam$UserScore[i] <- ScoreRank[2 + 3 * (i - 1)]
  Steam$MetaScore[i] <- ScoreRank[3 + 3 * (i - 1)]
  Steam$Playtime.Mean[i] <- PlayTime[1 + 2 * (i - 1)]
  Steam$Playtime.Median[i] <- PlayTime[2 + 2 * (i - 1)]
  Steam$MaxDiscountPercentage[i] <- Discount[1 + 2 * (i - 1)]
  Steam$LowestPrice[i] <- Discount[2 + 2 * (i - 1)]
}

Steam$PlayersPercentage <- as.numeric(gsub("[%]", "", Steam$Players))
Steam$Owners.before <- as.numeric(gsub("[,]", "", Steam$Owners.before))
Steam$Owners.after <- as.numeric(gsub("[,]", "", Steam$Owners.after))
Steam$Sales <- as.numeric(gsub("[,]", "", Steam$Sales))
Steam$IncreasePercentage <- as.numeric(gsub("[%]", "", Steam$Increase))
Steam$Price.y <- as.numeric(gsub("[$]", "", Steam$Price.y))
Steam$MaxDiscountPercentage <- as.numeric(gsub("[%]", "", Steam$MaxDiscountPercentage))
Steam$LowestPrice <- as.numeric(gsub("[$]", "", Steam$LowestPrice))
Steam$Release.date <- as.Date(Steam$Release.date, format = "%b %d,%Y")
Steam$Release.year <- year(Steam$Release.date)
Steam$Release.month <- month(Steam$Release.date)
Steam$Release.day <- day(Steam$Release.date)

## Selecting and renaming useful variables from orginial dataset
Steam <- Steam[, c(1, 2, 9, 17, 3, 6, 10:12, 14, 28, 25:26, 18:24, 27, 29:31)]
colnames(Steam) <- c("Game", "Rank of Playtime", "Rank of Sales",
                     "Rank of CurrentPlayers", "Release Date", "Owners Level",
                     "Owners Before", "Owners After", "Sales",
                     "Price", "Increase(Percentage)",
                     "Max Discount(Percentage of Original)", "Lowest Price",
                     "Current Players","Peak Today", "Score Rank(Percentage)",
                     "User Score", "Meta Score", "Average of Playtime",
                     "Median of Playtime", "Actural Players(Percentage)",
```

```
                    "Release Year", "Release Month", "Release Day")

# Steam Sales
Steam_sales$Owners.before <- as.numeric(gsub("[,]", "", Steam_sales$Owners.before))
Steam_sales$Owners.after <- as.numeric(gsub("[,]", "", Steam_sales$Owners.after))
Steam_sales$Sales <- as.numeric(gsub("[,]", "", Steam_sales$Sales))
Steam_sales$IncreasePercentage <- as.numeric(gsub("[%]", "", Steam_sales$Increase))
Steam_sales$Price <- as.numeric(gsub("[$]", "", Steam_sales$Price))

Discount <- unlist(strsplit(
  unlist(strsplit(
    as.character(Steam_sales$Max.discount), "[(]"
  )), "[)]"
))
for (i in 1:9623) {
  Steam_sales$MaxDiscountPercentage[i] <- Discount[1 + 2 * (i - 1)]
  Steam_sales$LowestPrice[i] <- Discount[2 + 2 * (i - 1)]
}
Steam_sales$MaxDiscountPercentage <- as.numeric(gsub("[%]", "",
                                                     Steam_sales$MaxDiscountPercentage))
Steam_sales$LowestPrice <- as.numeric(gsub("[$]", "", Steam_sales$LowestPrice))

## Selecting and renaming useful variables from orginial dataset
Steam_sales <- Steam_sales[, c(1:5, 7, 10:12)]
colnames(Steam_sales) <- c("Rank of Sales", "Game", "Owners Before",
                           "Owners After", "Sales", "Price", "Increase(Percentage)",
                           "Max Discount(Percentage of Original)", "Lowest Price")

# Steam Playtime
PlayTime <- unlist(strsplit(
  unlist(strsplit(
    as.character(Steam_playtime$Playtime..Median.), "[(]"
  )), "[)]"
))
for (i in 1:100) {
  Steam_playtime$`Average of Playtime`[i] <- PlayTime[1 + 2 * (i - 1)]
  Steam_playtime$`Median of Playtime`[i] <- PlayTime[2 + 2 * (i - 1)]
}
Steam_playtime$Release.date <- as.Date(Steam_playtime$Release.date, format = "%b %d,%Y")
Steam_playtime$Players <- as.numeric(gsub("[%]", "", Steam_playtime$Players))
colnames(Steam_playtime)[c(1,3,7)] <- c("Rank of Playtime", "Release Date",
                                        "Actural Players(Percentage)")

# Store Prices of Game "PUBG"
StorePrices_PUBG$`Converted Price(USD)` <- as.numeric(gsub("[$]", "",
                                                           StorePrices_PUBG$ConvtertedPrice))
StorePrices_PUBG$`Lowest Recorded Price(USD)` <- as.numeric(
  gsub("[$]", "", StorePrices_PUBG$LowestRecordedPrice)
)
StorePrices_PUBG$`Compare Percentage` <- as.numeric(gsub("[%]", "", StorePrices_PUBG$Compare))
StorePrices_PUBG <- StorePrices_PUBG[,-c(1,3:6)]

# Raw Text
```

```
raw.text$X <- NULL
raw.text$text <- as.character(raw.text$text)

# Current Players
colnames(Current_players) <- c("Rank of Current Players", "Game",
                               "Current Players", "Peak Today")
```

EDA

```
# Top 10 Games of Play Time
Steam_playtime_top10 <-
  Steam_playtime %>%
  select(
    `Rank of Playtime`, Game, `Release Date`, Price, `Actural Players(Percentage)`,
    `Average of Playtime`, `Median of Playtime`
  ) %>%
  top_n(10, -`Rank of Playtime`)
colnames(Steam_playtime_top10)[1] <- "Rank"

kable(Steam_playtime_top10, "latex",
      caption = "Top 10 Games of Play Time", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down"))
```

Table 1: Top 10 Games of Play Time

Rank	Game	Release Date	Price	Actural Players(Percentage)	Average of Playtime	Median of Playtime
1	Counter-Strike: Global Offensive	2012-08-21	Free	39.69	13:38	05:08
2	Dota 2	2013-07-09	Free	16.71	26:23	13:45
3	PLAYERUNKNOWN'S BATTLEGROUNDS	2017-12-21	$29.99	31.51	11:58	03:54
4	Tom Clancy's Rainbow Six Siege	2015-12-01	$14.99	29.00	11:49	04:57
5	Team Fortress 2	2007-10-10	Free	5.97	20:02	03:08
6	Grand Theft Auto V	2015-04-13	$14.99	18.06	07:52	02:16
7	Euro Truck Simulator 2	2012-10-12	$19.99	22.77	08:24	03:31
8	Ring of Elysium	2018-09-19	Free	29.91	06:04	01:47
9	Rocket League	2015-07-07	$19.99	20.92	08:56	03:25
10	Rust	2018-02-08	$34.99	11.82	24:30	09:18

```
# Top 10 Games of Sales
Steam_sales_top10 <-
  Steam_sales %>%
  select(`Rank of Sales`, Game, `Owners After`, Price, `Lowest Price`) %>%
  top_n(10, -`Rank of Sales`)

kable(Steam_sales_top10, "latex",
      caption = "Top 10 Games of Sales", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
# Top 10 Games of Current Players
Current_players_top10 <-
  Current_players %>%
  top_n(10, -`Rank of Current Players`)
kable(Current_players_top10, "latex",
      caption = "Top 10 Games of Current Players", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

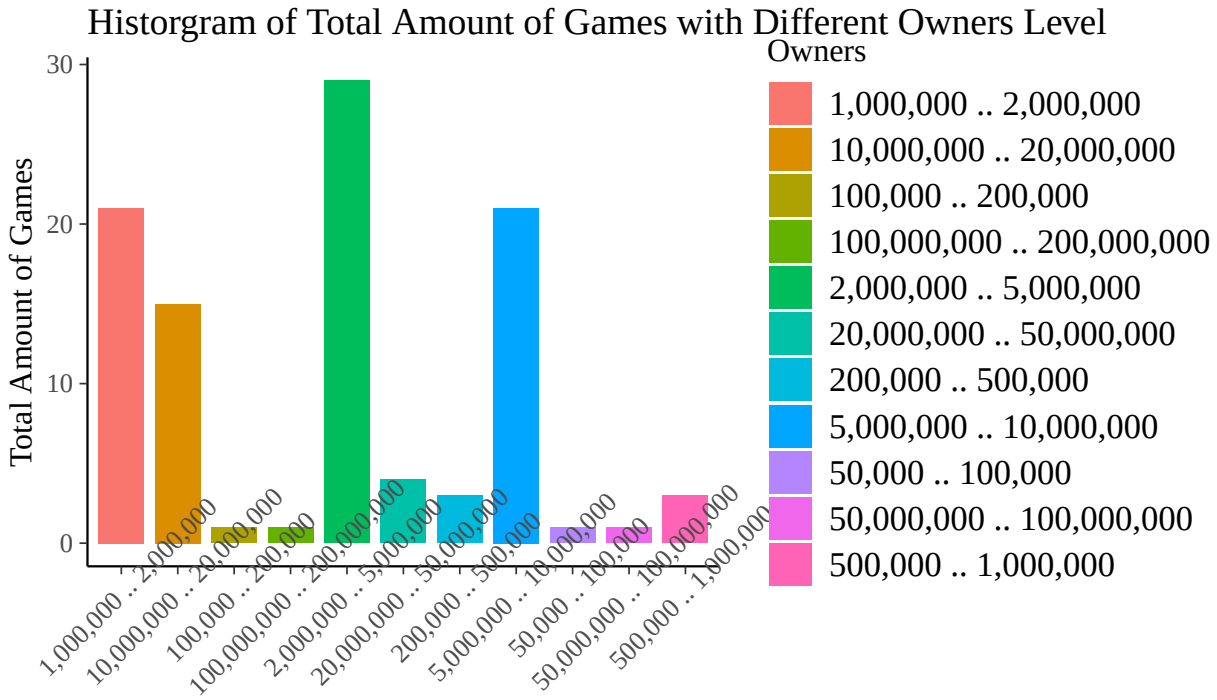
Table 2: Top 10 Games of Sales

Rank of Sales	Game	Owners After	Price	Lowest Price
1	PLAYERUNKNOWN'S BATTLEGROUNDS	64106000	29.99	19.99
2	Left 4 Dead 2	19143000	9.99	1.99
3	PAYDAY 2	15410000	9.99	4.99
4	Half-Life 2: Deathmatch	14725000	4.99	0.79
5	Garry's Mod	14518000	9.99	2.49
6	Counter-Strike: Condition Zero	11379000	9.99	0.99
7	Portal 2	10830000	9.99	1.99
8	Portal	10633000	9.99	0.99
9	Grand Theft Auto V	9954000	29.99	14.99
10	Half-Life 2	9878000	9.99	0.99

Table 3: Top 10 Games of Current Players

Rank of Current Players	Game	Current Players	Peak Today
1	PLAYERUNKNOWN'S BATTLEGROUNDS	581958	926855
2	Dota 2	434313	700240
3	Counter-Strike: Global Offensive	290873	685353
4	Tom Clancy's Rainbow Six Siege	80229	114102
5	Grand Theft Auto V	61282	94685
6	Warframe	57119	73052
7	Path of Exile	54273	103108
8	MONSTER HUNTER: WORLD	45868	63955
9	Team Fortress 2	43207	56308
10	PUBG: Test Server	41305	54635

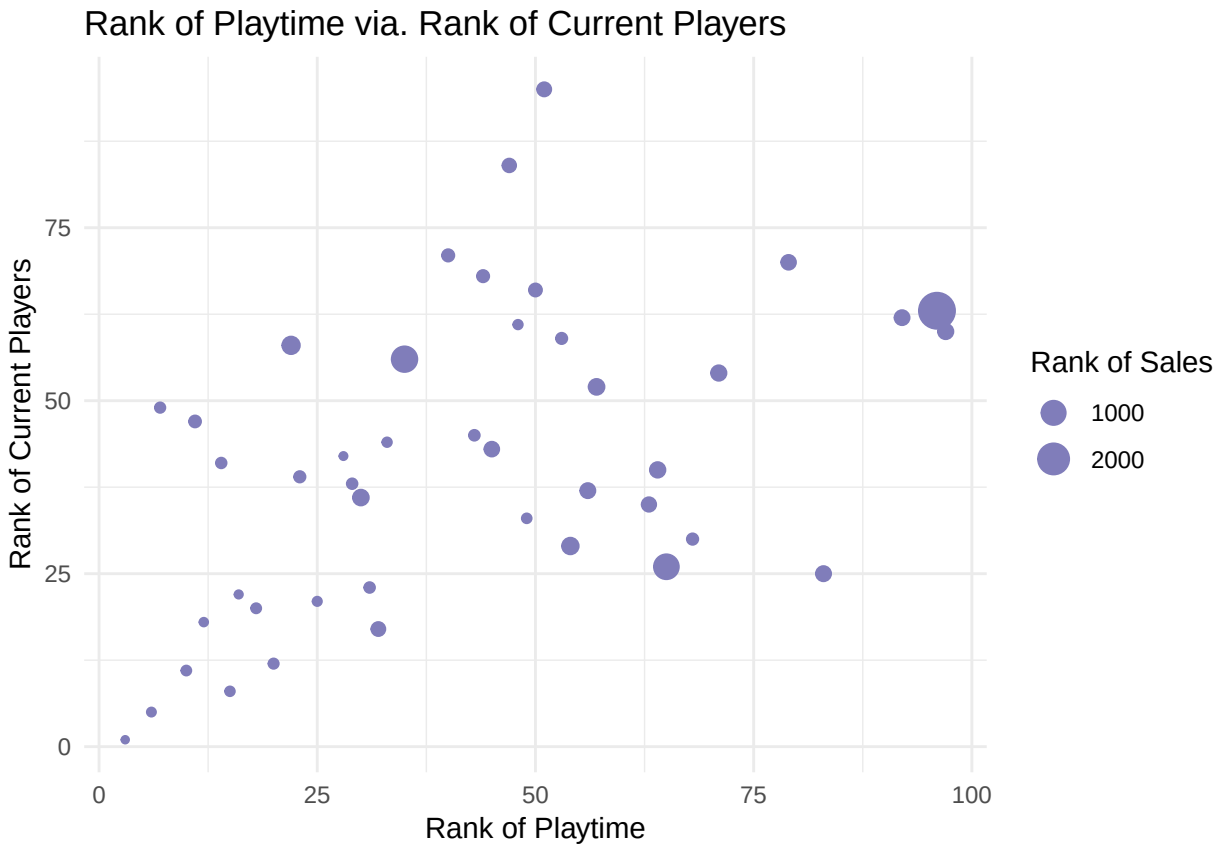
```
# Historgram of Total Amount of Games with Different Owners Level
ggplot(data = Steam_playtime %>% group_by(Owners) %>% count(),
       aes(x = Owners, y = n)) +
  geom_bar(aes(fill = Owners), stat = "identity",
           position = "stack", width = 0.8) +
  labs(x = "Owners Level",
       y = "Total Amount of Games",
       title = "Historgram of Total Amount of Games with Different Owners Level") +
  theme(axis.text.x = element_text(angle = 45))
```



Owners Level

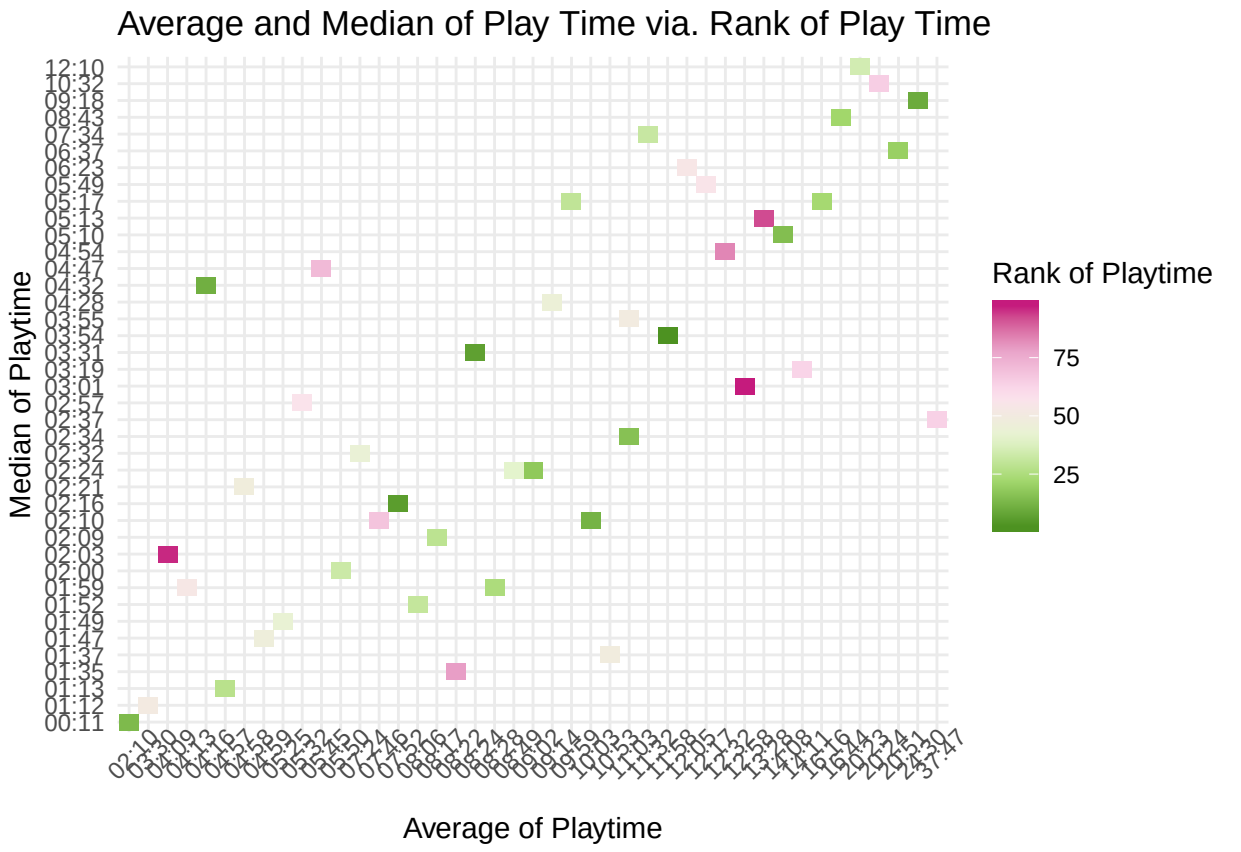
The owners level of the majority games among top 100 based on play time either between 100,000,000 to 200,000,000 which is very high or between 200,000 to 500,000 or 1,000,000 to 2,000,000 which are lower than other levels. So those favorite games drive people pay more time on are either have tons of owners or have less owners but attract each owner to spent more time on it.

```
# Rank of Playtime via. Rank of Current Players
ggplot(data = Steam) +
  aes(x = `Rank of Playtime`, y = `Rank of CurrentPlayers`, size = `Rank of Sales`) +
  geom_point(color = "#807dba") +
  labs(
    title = "Rank of Playtime via. Rank of Current Players",
    y = "Rank of Current Players"
  ) +
  theme_minimal()
```



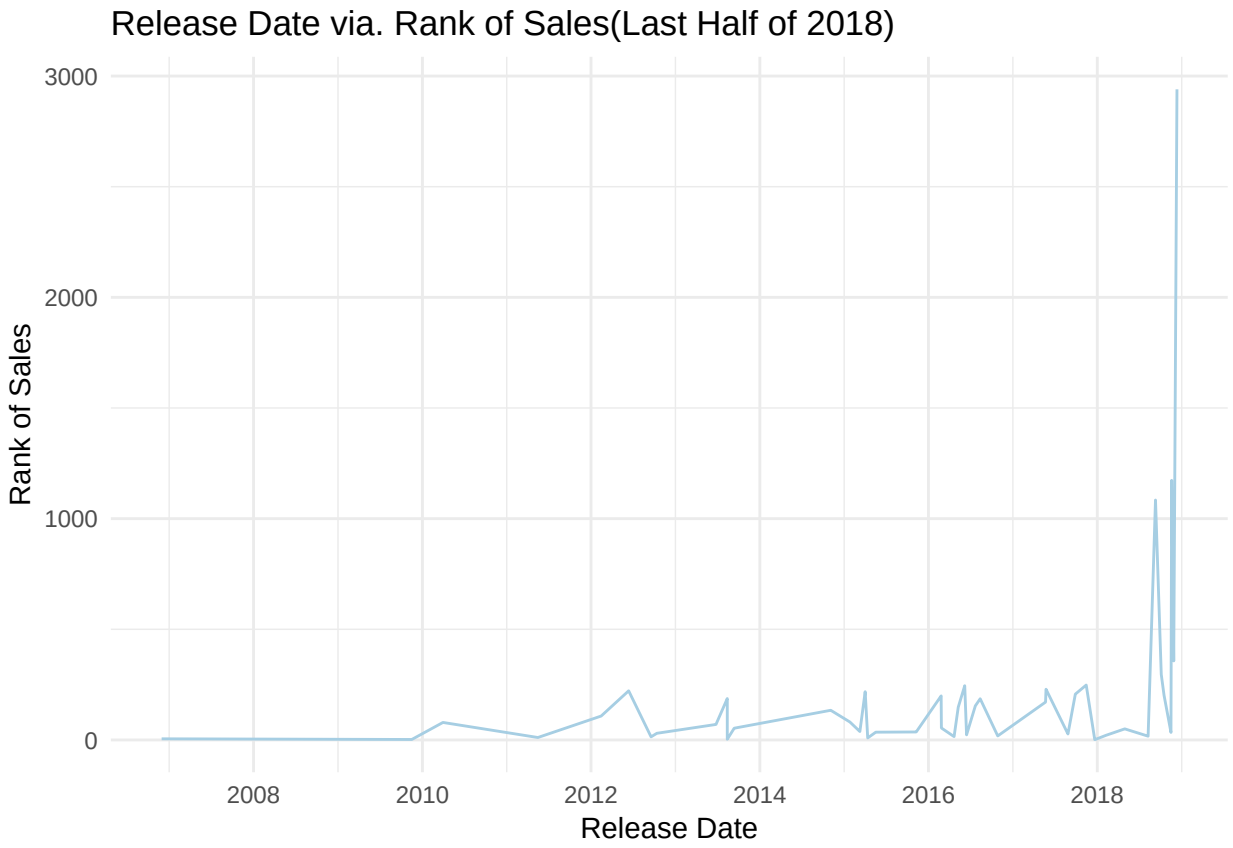
From the rank of play time and current players, there is a weak positive correlated relationship between play time and the amount of current players. According to the size of each point, represented by rank of sales, there is no apparent relationship between sales volume and play time either or current players. The bigger point means the more that game sold.

```
# Average and Median of Play Time via. Rank of Play Time
ggplot(data = Steam) +
  aes(x = `Average of Playtime`, y = `Median of Playtime`, fill = `Rank of Playtime`) +
  geom_tile() +
  scale_fill_distiller(palette = "PiYG") +
  labs(title = "Average and Median of Play Time via. Rank of Play Time") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45))
```



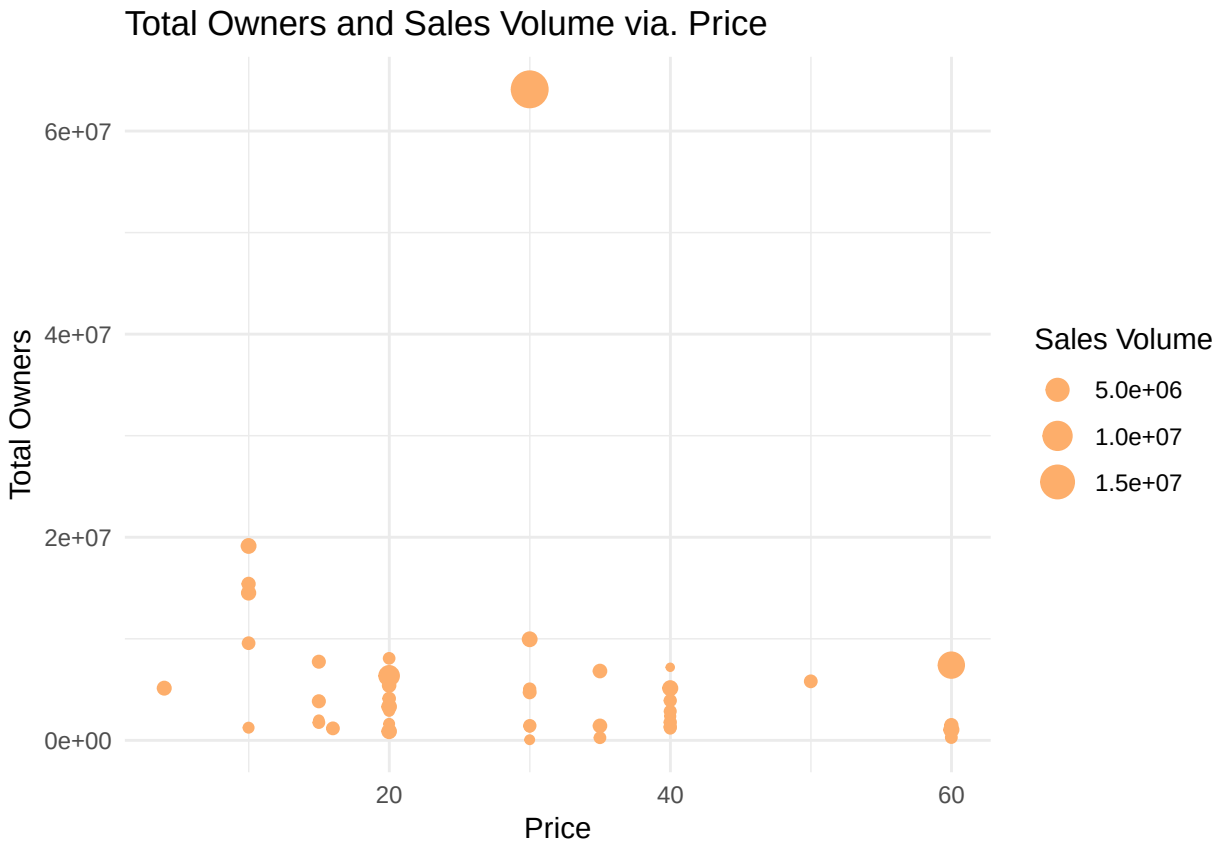
Average play time and median of play time are positive related. However, either of these two indicators seem have no apparent relationship to the rank of playtime. Even though the average and median play time of some games are low, the rank of playtime is still high. For these type of games, there are more users have been playing this game for a long time, on the same time, there are also more users might only played for few seconds. Hence, the average play time is low instead the total play time still on the top of the rank.

```
# Release Date via. Rank of Sales(Last Half of 2018)
ggplot(data = Steam) +
  aes(x = `Release Date`, y = `Rank of Sales`) +
  geom_line(color = "#a6cee3") +
  labs(
    title = "Release Date via. Rank of Sales(Last Half of 2018)",
    y = "Rank of Sales"
  ) +
  theme_minimal()
```



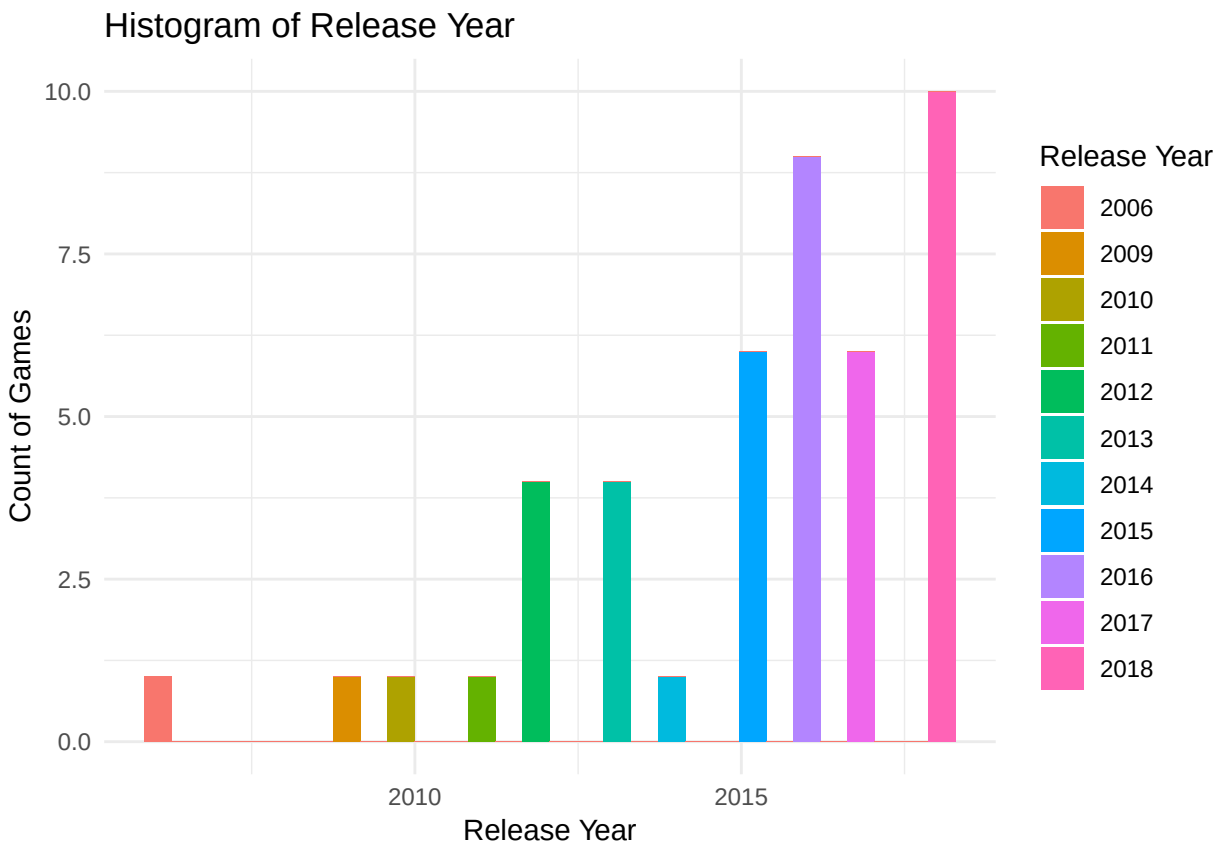
The best seller among games are usually release recently. For users, latest games can keep novelty and follow the development of the computer. No one would like play a game with pixels.

```
# Total Owners and Sales via. Price
ggplot(data = Steam) +
  aes(x = Price, y = `Owners After`, size = Sales) +
  geom_point(color = "#fdae6b") +
  labs(
    title = "Total Owners and Sales Volume via. Price",
    y = "Total Owners",
    size = "Sales Volume"
  ) +
  theme_minimal()
```



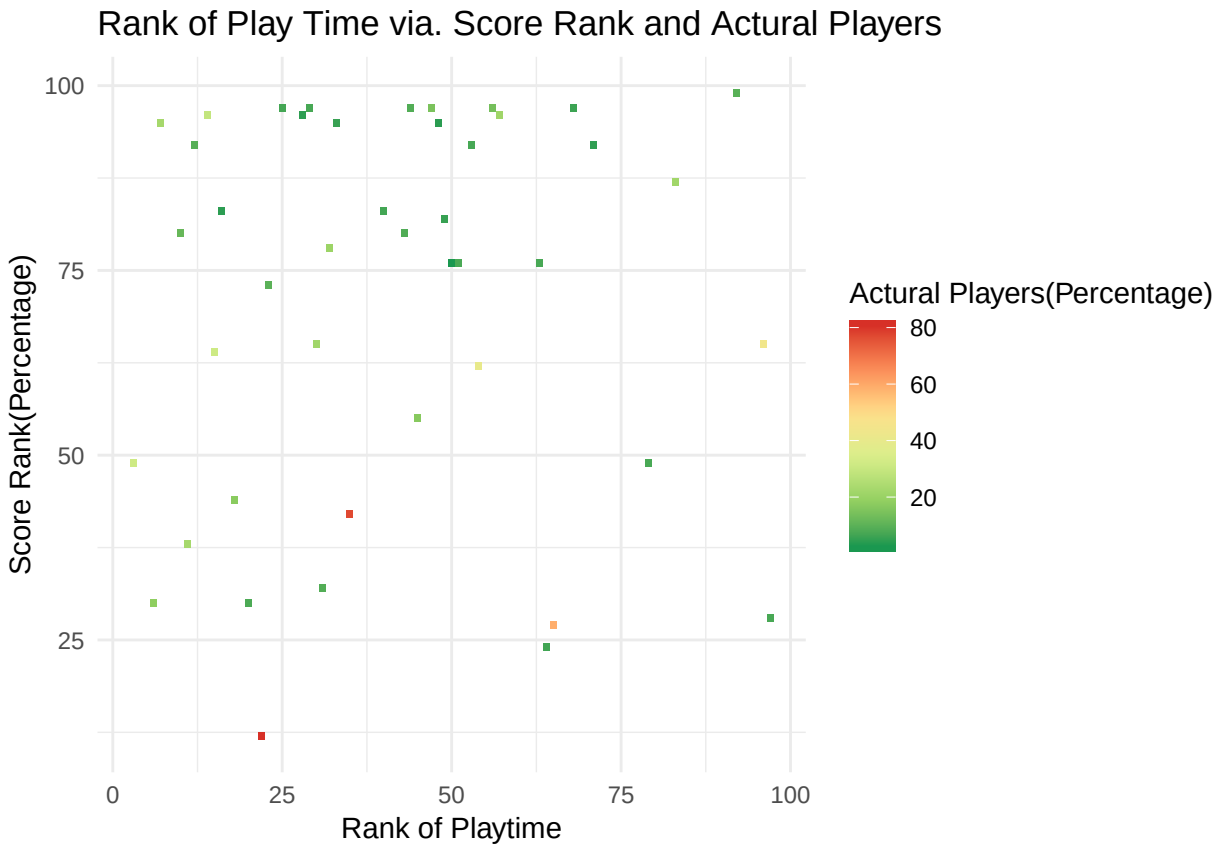
The sales volume doesn't always for granted to have relationship of price. The best seller(biggest point on plot) is actually positioning the price of its produce at average level which can be accepted by most of people.

```
# Historgram of Release Year
ggplot(data = Steam) +
  aes(x = `Release Year`, fill = as.factor(`Release Year`)) +
  geom_histogram(bins = 30) +
  labs(
    title = "Histogram of Release Year",
    y = "Count of Games",
    fill = "Release Year"
  ) +
  theme_minimal()
```



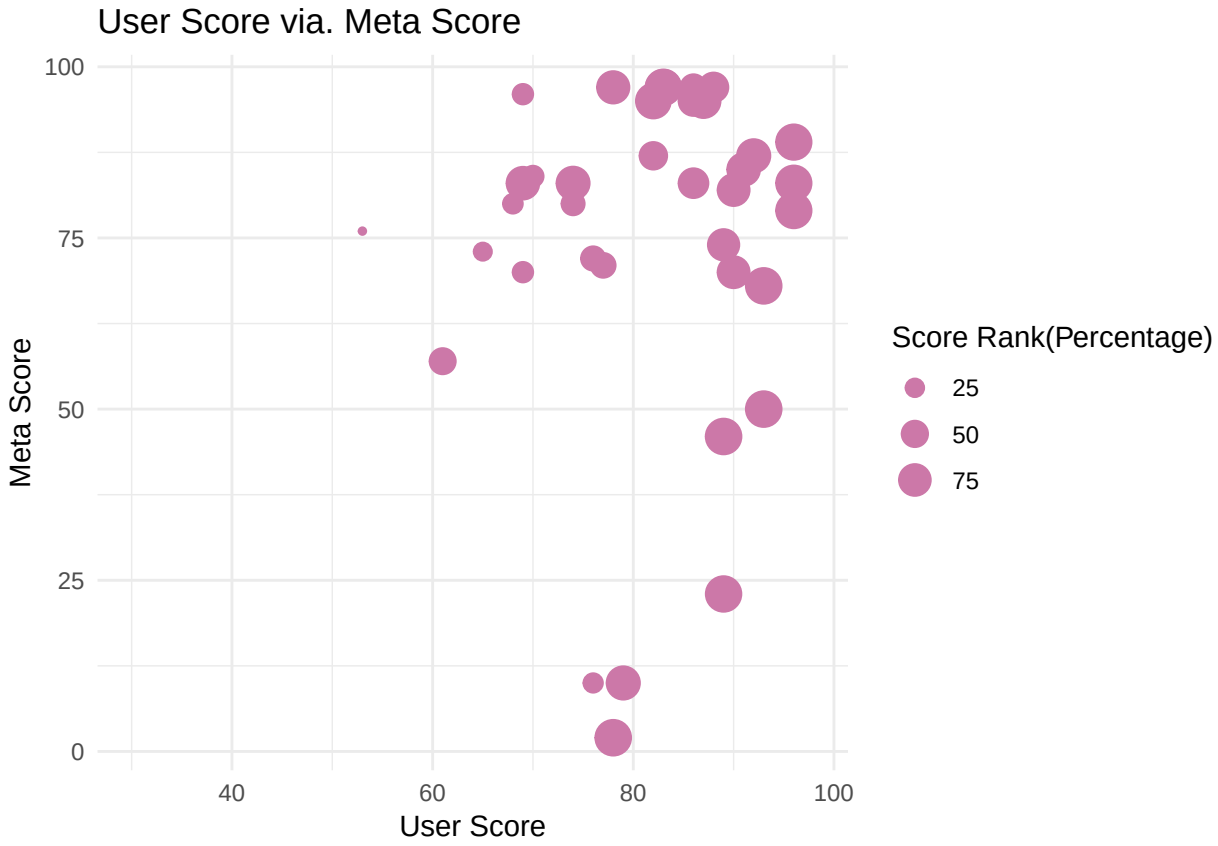
Only few games released before 2010 are still on the top of rank, which is impressed.

```
# Rank of Play Time via. Score Rank and Actural Players
ggplot(data = Steam) +
  aes(x = `Rank of Playtime`, y = `Score Rank(Percentage)`,
      fill = `Actural Players(Percentage)`) +
  geom_tile() +
  scale_fill_distiller(palette = "RdYlGn") +
  labs(
    title = "Rank of Play Time via. Score Rank and Actural Players",
    y = "Score Rank(Percentage)"
  ) +
  theme_minimal()
```



The weird thing is, the more actual players(the percentage of people who are really playing among total owners) are, the lower of score rank(the user score of this games over how many percentage other games on Steam). Is this a common phenomenon with popular games which have more actual players but more complains?

```
# User Score via. Meta Score
ggplot(data = Steam) +
  aes(x = `User Score`, y = `Meta Score`, size = `Score Rank(Percentage)`) +
  geom_point(color = "#cc78a8") +
  labs(
    title = "User Score via. Meta Score",
    y = "Meta Score"
  ) +
  theme_minimal()
```



This plot shows a U turn trend on the right side. In general, higher user score related to higher meta score. But even if higher meta score, there are few games have lower score rank compared to other games on Steam.

```
# Store Prices Data of PUBG through Different Countries
StorePrices_PUBG_top10 <-
  StorePrices_PUBG %>%
  arrange(`Lowest Recorded Price(USD)`) %>%
  top_n(10, -`Lowest Recorded Price(USD)`)

kable(StorePrices_PUBG_top10, "latex",
      caption = "Top 10 Games of Play Time", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down"))
```

Table 4: Top 10 Games of Play Time

Currency	Converted Price(USD)	Lowest Recorded Price(USD)	Compare Percentage
Argentine Peso	14.46	6.18	-51.76
Turkish Lira	16.19	8.60	-46.01
Russian Ruble	13.48	9.03	-55.04
Indonesian Rupiah	15.08	9.19	-49.69
Indian Rupee	13.89	9.30	-53.68
Brazilian Real	14.29	9.57	-52.34
Philippine Peso	14.31	9.59	-52.27
Chinese Yuan Renminbi	14.18	9.69	-52.69
Vietnamese Dong	14.62	9.80	-51.23
South Asia - U.S. Dollar	29.99	10.49	0.00

PUBG(PLAYERUNKNOWN'S BATTLEGROUNDS) as a favorite game not only on sales but also on players, the prices of it among different countries are different. The original price in Unite States is $29.99. The cheapest recorded price among all different countries is 6.18 dollars in Argentine Peso, which means if you have a Steam account in Argentine Peso, you can easily own this famous game with 80% off.

Benford Analysis

In this part, this project used sales data of games on Steam over past half of year(from Jun 2018 to Dec 2018) for verifying. This data including sales records of 9623 games in total.

```
# Choose sales volume data of each game
bfd.cp <- benford(Steam_sales$Sales)
bfd.cp
```

```
##
## Benford object:
##
## Data: Steam_sales$Sales
## Number of observations used = 9623
## Number of obs. for second order = 345
## First digits analysed = 2
##
## Mantissa:
##
##    Statistic  Value
##         Mean  0.475
##          Var  0.061
##  Ex.Kurtosis -0.810
##     Skewness  0.152
##
##
## The 5 largest deviations:
##
##   digits absolute.diff
## 1     20       2148.10
## 2     30       1224.96
## 3     40        806.80
## 4     50        587.24
## 5     60        382.92
##
## Stats:
##
##  Pearson's Chi-squared test
##
## data:  Steam_sales$Sales
## X-squared = 53214, df = 89, p-value < 2.2e-16
##
##
##  Mantissa Arc Test
##
## data:  Steam_sales$Sales
## L2 = 0.051898, df = 2, p-value < 2.2e-16
##
```

```
## Mean Absolute Deviation: 0.01346692
## Distortion Factor: -10.20232
##
## Remember: Real data will never conform perfectly to Benford's Law. You should not focus on p-values!
```

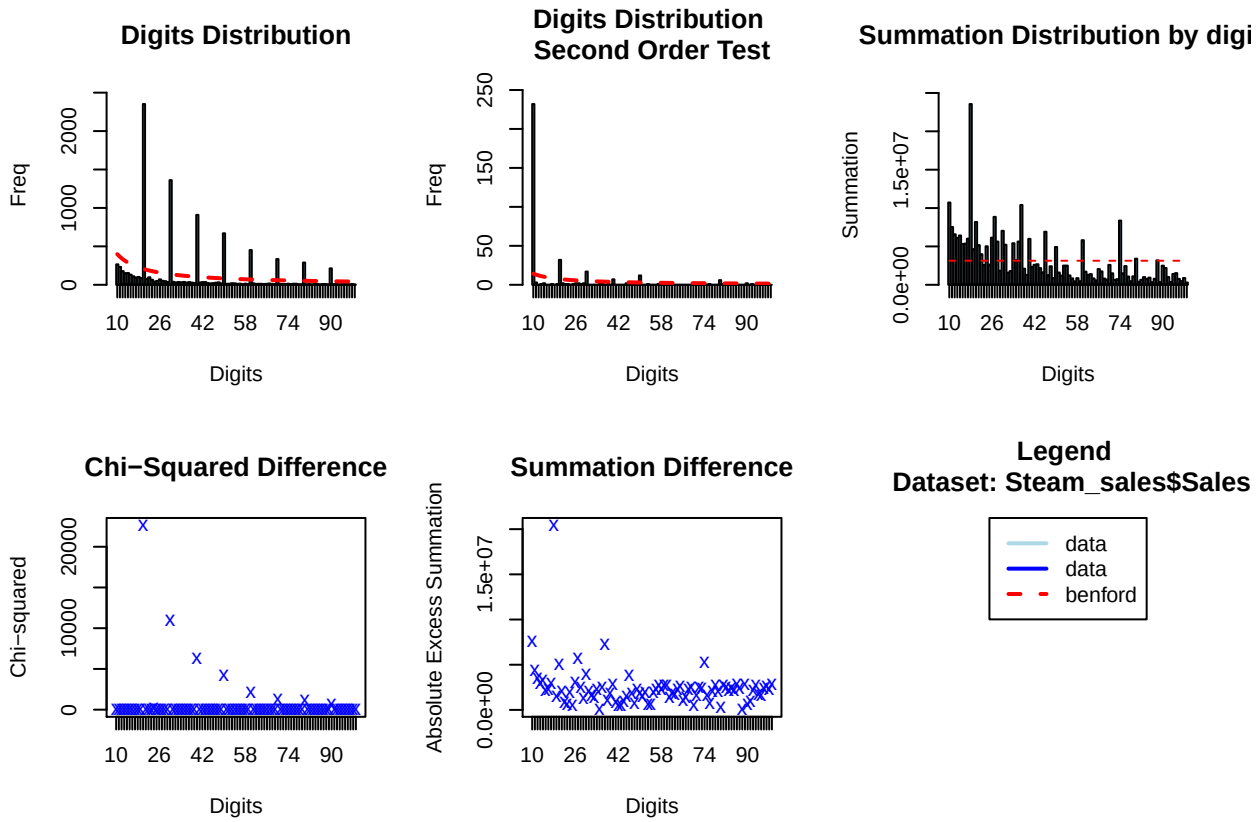
```
# gets the Mean Absolute Deviation
MAD(bfd.cp)
```

```
## [1] 0.01346692
```

```
# gets the Chi-squared test
chisq(bfd.cp)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Steam_sales$Sales
## X-squared = 53214, df = 89, p-value < 2.2e-16
```

```
# plot results from benford analysis
plot(bfd.cp)
```




```
# get suspects data
suspects <- getSuspects(bfd.cp, Steam_sales)
# suspects data by ranking
suspects_ranked <- suspectsTable(bfd.cp)
kable(suspects_ranked[1:10], "latex", caption = "Top 10 of Suspects Data") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
# get duplicates data
duplicates <- getDuplicates(bfd.cp, Steam_sales)
```

Table 5: Top 10 of Suspects Data

digits	absolute.diff
20	2148.0954
30	1224.9643
40	806.8042
50	587.2405
60	382.9205
70	275.7193
80	238.0836
90	167.8203
13	157.7132
12	156.5157

```
# duplicates data by ranking
duplicates_ranked <- duplicatesTable(bfd.cp)
kable(duplicates_ranked[1:10], "latex", caption = "Top 10 of Duplicates Data") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 6: Top 10 of Duplicates Data

number	duplicates
2000	2271
3000	1314
4000	875
5000	655
6000	435
7000	331
8000	285
10000	221
9000	207
11000	185

According to the plot, we can get the conclusion that there are some inconsistencies on this data. The majority data obey the distribution of benford, but it shows 8 large discrepancies. From the table of duplicates data, the most probably duplicated number is 2000, we can see that over 2271 games which is bigger than 20% of data sold 2000 times over the past half of year which is extremely weird. Excepted that, 3000 - 11000 are also among the top 10 duplicates data. Interesting. It might caused by recording in round or other way or the sales data appears on Steam is not convincing.

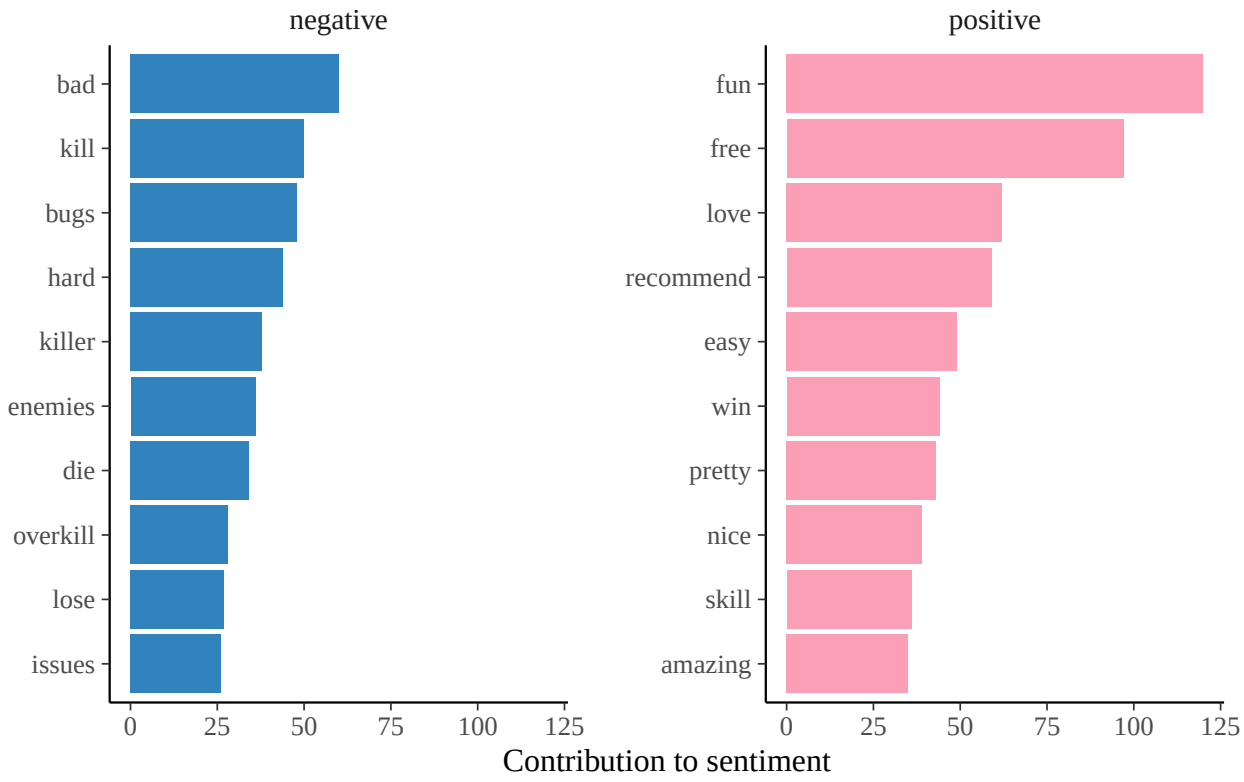
Text Mining

In text mining, this project choose top 10 reviews of top 30 games based on playing time of each game as representative. Armed at exploring what factors of a game play an important role for users.

```
tidy_review <- raw.text %>%
  unnest_tokens(word, text) %>%
  mutate(word = str_extract(word, "[a-z'\\s]+")) %>%
  anti_join(stop_words, by = "word") %>%
  filter(
```

```
    !word %in% "[\\s]+",
    !word %in% "",
    !word %in% NA,
    !word %in% "Posted",
    !word %in% "posted",
    !word %in% "game"
  )

# Import sentiment dictionaires
bing <- get_sentiments("bing")

sentitext <- raw.text %>%
  unnest_tokens(sentence, text, token = "sentences") %>%
  ungroup() %>%
  unnest_tokens(word, sentence) %>%
  anti_join(stop_words, by = "word") %>%
  filter(
    !word %in% "[\\s]+",
    !word %in% "",
    !word %in% NA,
    !word %in% "Posted",
    !word %in% "posted",
    !word %in% "game"
  )

bing_counts <- sentitext %>%
  inner_join(bing, by = "word") %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

# Wordclound of all games
tidy_review %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```


```
# Top 10 Hot Words on Positive and Negative Attitudes
bing_counts %>%
  group_by(sentiment) %>%
  top_n(10, n) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(
    y = "Contribution to sentiment",
    x = NULL,
    title = "Top 10 Hot Words on Positive and Negative Attitudes"
  ) +
  coord_flip() +
  scale_fill_manual(values = c("#3182bd", "#fa9fb5"))
```




```
# Wordclound of Positive and Negative Attitudes
sentitext %>%
  inner_join(bing, by = "word") %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(
    colors = c("#3182bd", "#fa9fb5"),
    max.words = 100
  )
```

From the wordcloud of all reviews, gamers focus more on words like “play”, “time”, “people” etc., which means what they really matter about a game are how this game preformed, how many times they devoted in it and how was the entire community of this game. And from the top10 words on negative attitudes, bing dictionary may be not a good choice of sentiment analyzing because there are several words like “killer”, “die” are not actually negative emotions when it comes to games. But it still shows that when the users have actual negative emotions about a game are usually because of bugs and other issues of game itself. Vice versa, for positive part, “fun”, “free”, “pretty” etc. are expected to see.

Shinyapps