```r
# Name: Christopher Kingston
# Date: 04/24/2023
# Assignment Final Project

# Setup
getwd()
setwd('C:/Users/PC/Documents/Final Project')
list.files()
library(tidyverse)
library(ggplot2)
library(dplyr)
library(gridExtra)
library(fmsb)

# Overwatch 2 has had two complete seasons in the data sets and one incomplete
# Season 3 stretched from February 7th to April 3 and the data was acquired
# March 12th.

season3start = as.Date('2023-2-7')
season3end = as.Date('2023-4-3')
season3length <- length(seq(from=season3start, to=season3end, by='day')) - 1
datapulldate = as.Date('2023-3-12')
length(seq(from=season3start, to=datapulldate, by='day')) - 1
datacompletion = 33/55
datacompletion


# Season 1 October 4, 2022 to December 6, 2022
season1start = as.Date('2022-10-4')
season1end = as.Date('2022-12-6')
season1length <- length(seq(from=season1start, to=season1end, by='day')) - 1
# Season 2 December 6, 2022 to February 7, 2023
season1start = as.Date('2022-12-6')
season1end = as.Date('2023-2-7')
season2length <- length(seq(from=season1start, to=season1end, by='day')) - 1

# 60% of season 3 was complete
# Season 1 and 2 were 63 days long and Season 3 was scheduled for only 55 days.
# Be aware that though season 3 data is incomplete, the season is shorter than
# previous seasons so captured more fo the percent of the season than had they
# been the previous season length

# Load data sets
Season1 <- read.csv("ow2_season_01_FINAL_heroes_stats__2023-03-12.csv")
View(Season1)
Season2 <- read.csv("ow2_season_02_FINAL_heroes_stats__2023-03-12.csv")
View(Season2)
Season3 <- read.csv("ow_heroes_data_season3_2023-03-08.csv")
View(Season3)

# Clean data sets of unnecessary columns for analysis
colnames(Season1)
Season1Cleaned <- Season1 |>
  select(Hero, Skill.Tier, KDA.Ratio, Pick.Rate...,Win.Rate...,
         Eliminations...10min, Objective.Kills...10min, Objective.Time...10min,
         Damage...10min, Healing...10min, Deaths...10min, Role)
View(Season1Cleaned)
Season2Cleaned <- Season2 |>
  select(Hero, Skill.Tier, KDA.Ratio, Pick.Rate...,Win.Rate...,
         Eliminations...10min, Objective.Kills...10min, Objective.Time...10min,
         Damage...10min, Healing...10min, Deaths...10min, Role)
Season3Cleaned <- Season3 |>
  select(Hero, Skill.Tier, KDA.Ratio, Pick.Rate...,Win.Rate...,
         Eliminations...10min, Objective.Kills...10min, Objective.Time...10min,
```

```r
         Damage...10min, Healing...10min, Deaths...10min, Role)

# Seasons Grouped by Role
Season1Role <- Season1Cleaned |>
  group_by(Role) |>
  summarise(Average_win_rate_S1 = mean(Win.Rate...),
                                    Average_pick_rate_S1 = mean(Pick.Rate...))
View(Season1Role)
Season2Role <- Season2Cleaned |>
  group_by(Role) |>
  summarise(Average_win_rate_S2 = mean(Win.Rate...),
            Average_pick_rate_S2 = mean(Pick.Rate...))
View(Season2Role)
Season3Role <- Season3Cleaned |>
  group_by(Role) |>
  summarise(Average_win_rate_S3 = mean(Win.Rate...),
            Average_pick_rate_S3 = mean(Pick.Rate...))
View(Season3Role)

# Join Season Roles
SeasonalRoles <- full_join(Season1Role, Season2Role,by='Role')
SeasonalRoles <- full_join(SeasonalRoles, Season3Role,by='Role')

# Wins data frame
SeasonalWinRole <- SeasonalRoles |>
  select(Role, Average_win_rate_S1, Average_win_rate_S2, Average_win_rate_S3)
View(SeasonalWinRole)

# Isolating win rates by role
# Season 1
S1WinS <- Season1Cleaned |>
  filter(Role == 'Support') |>
  select(Win.Rate...)
S1WinT <- Season1Cleaned |>
  filter(Role == 'Tank') |>
  select(Win.Rate...)
S1WinD <- Season1Cleaned |>
  filter(Role == 'Damage') |>
  select(Win.Rate...)
# Season 2
S2WinS <- Season2Cleaned |>
  filter(Role == 'Support') |>
  select(Win.Rate...)
S2WinT <- Season2Cleaned |>
  filter(Role == 'Tank') |>
  select(Win.Rate...)
S2WinD <- Season2Cleaned |>
  filter(Role == 'Damage') |>
  select(Win.Rate...)
# Season 3
S3WinS <- Season3Cleaned |>
  filter(Role == 'Support') |>
  select(Win.Rate...)
S3WinT <- Season3Cleaned |>
  filter(Role == 'Tank') |>
  select(Win.Rate...)
S3WinD <- Season3Cleaned |>
  filter(Role == 'Damage') |>
  select(Win.Rate...)

# Testing Win Means

# Setup
xmean <- mean(S1WinS$Win.Rate...)
ymean <- mean(S1WinD$Win.Rate...)
```

```
xmean
ymean
xmean <- mean(S1WinS$Win.Rate...)
ymean <- mean(S1WinT$Win.Rate...)

# Function to compare means
CompareMeans <- function(x,y){
  if (mean(x)==mean(y)) return('same')
  if (mean(x)>mean(y)) return('positive')
  if (mean(x)<mean(y))return('negative')
}
CompareMeans(xmean,ymean)

# Compare Season One Support to Damage than Support to Tank and Damage win rates were
positive
# S1S - 46.82844
# S1T - 46.80025
# S1D - 46.19618

# Compare Season Two Support to Damage than Support to Tank and Damage win rates were
positive
# S2S - 46.8925
# S2T - 46.47125
# S2D - 46.17471

# Compare Season Three Support to Damage than Support to Tank and Damage win rates were
positive
# S3S - 48.98531
# S3T - 46.92761
# S3D - 47.98544

# Concludes that between all three season Support roles have greater mean win rates than
Tank and Damage roles

S1WinSplot <- ggplot() +
  geom_hex(Season1Cleaned,
                mapping=aes(x=Role, y=Win.Rate..., color=Role))
S1WinSplot

S1 <- ggplot(SeasonalWinRole,
       aes(x=Role, y=Average_win_rate_S1, color=Role)) +
  geom_point(size=6) +
  ggtitle('Season One') +
  ylim(44,50)
S1
S2 <- ggplot(SeasonalWinRole,
             aes(x=Role, y=Average_win_rate_S2, color=Role)) +
  geom_point(size=6) +
  ggtitle('Season Two') +
  ylim(44,50)
S2
S3 <- ggplot(SeasonalWinRole,
             aes(x=Role, y=Average_win_rate_S3, color=Role)) +
  geom_point(size=6) +
  ggtitle('Season Three') +
  ylim(44,50)
S3
# Win Visual by Role
WinRates <- grid.arrange(S1 + S2 + S3)
WinRates
# Pick data fame
SeasonalPickRole <- SeasonalRoles |>
  select(Role, Average_pick_rate_S1, Average_pick_rate_S2, Average_pick_rate_S3)
View(SeasonalPickRole)
```

```
S1P <- ggplot(SeasonalPickRole,
             aes(x=Role, y=Average_pick_rate_S1, color=Role)) +
  geom_point(size=6) +
  ggtitle('Season One') +
  ylim(0,7)
S1P
S2P <- ggplot(SeasonalPickRole,
             aes(x=Role, y=Average_pick_rate_S2, color=Role)) +
  geom_point(size=6) +
  ggtitle('Season Two') +
  ylim(0,7)
S2P
S3P <- ggplot(SeasonalPickRole,
             aes(x=Role, y=Average_pick_rate_S3, color=Role)) +
  geom_point(size=6) +
  ggtitle('Season Three') +
  ylim(0,7)
# Pick Visual
PickRates <- S1P + S2P + S3P
PickRates
# Support has a higher average win rate and pick rate in all three seasons

# Count depreciation of pick rate
HeroCount <- Season3Cleaned |>
  group_by(Role) |>
  summarise(HeroCountByRole = unique(Hero))
View(HeroCount)
# Out of the 36 hero selections: 17 are Damage, 11 Tank, and 8 Support
# Depreciate by pool size to show over inflated support pick rate
Role <- c("Damage","Tank","Support")
RoleNum <- c("17","11","8")
PickDeflator <- c("0.472","0.305","0.222")
Season_One <- c("0.769880616","1.876846170","0.4840155")
Season_Two <- c("1.077548152","1.574324295","0.49273904")
Season_Three <- c("1.4811081152","2.068948285","0.638123904")
Deflated_Pick <- data.frame(Role, PickDeflator, Season_One, Season_Two, Season_Three)
View(Deflated_Pick)
# Despite having the highest pick, when deflated for the pool in each role Support is
picked the least to a staggering degree.

# Conclusion of the first section is that a player selecting to play Support over the
other two roles will experience more success and less competition.

# Look at relationship between league ranks
# Rank ordering
RankOrdering <- data.frame(Skill.Tier = c('All','Bronze','Silver','Gold',
'Platinum','Diamond','Master','Grandmaster'),
                           Ranknum = c(1:8))
# Season1
Season1ST <- Season1Cleaned |>
  group_by(Skill.Tier) |>
  summarise(S1_Average_win_rate = mean(Win.Rate...),
            S1_Average_pick_rate = mean(Pick.Rate...),
            S1_Count_pick_rate = sum(Pick.Rate...))
View(Season1ST)
# Order by rank
Season1ST <- Season1ST |>
  left_join(RankOrdering,
            by = 'Skill.Tier')
Season1ST <- Season1ST[order(Season1ST$Ranknum),]

# Season2
Season2ST <- Season2Cleaned |>
  group_by(Skill.Tier) |>
  summarise(S2_Average_win_rate = mean(Win.Rate...),
```

```
            S2_Average_pick_rate = mean(Pick.Rate...),
            S2_Count_pick_rate = sum(Pick.Rate...))
View(Season2ST)
# Order by rank
Season2ST <- Season2ST |>
  left_join(RankOrdering,
            by = 'Skill.Tier')
Season2ST <- Season2ST[order(Season2ST$Ranknum),]

# Season3
Season3ST <- Season3Cleaned |>
  group_by(Skill.Tier) |>
  summarise(S3_Average_win_rate = mean(Win.Rate...),
            S3_Average_pick_rate = mean(Pick.Rate...),
            S3_Count_pick_rate = sum(Pick.Rate...))
View(Season3ST)
# Order by rank
Season3ST <- Season3ST |>
  left_join(RankOrdering,
            by = 'Skill.Tier')
Season3ST <- Season3ST[order(Season3ST$Ranknum),]

# Bulk of the players move up in the ranks each season
# Unusual that lower ranks get thinner, perhaps player's skill has grown, but fewer new
players are participating
SeasonalRanks <- list(Season1ST, Season2ST, Season3ST)
SeasonalRanks |> reduce(full_join, by='Skill.Tier')
print(SeasonalRanks)
SeasonalRanks <- data.frame(SeasonalRanks)
print(SeasonalRanks)

# look at objective time/kill to win rate by role
# Season1
Season1O <- Season1Cleaned |>
  group_by(Role) |>
  summarise(S1_Average_Objective_Time = mean(Objective.Time...10min),
            S1_Average_Objective_Kills = mean(Objective.Kills...10min))
View(Season1O)
Season2O <- Season2Cleaned |>
  group_by(Role) |>
  summarise(S2_Average_Objective_Time = mean(Objective.Time...10min),
            S2_Average_Objective_Kills = mean(Objective.Kills...10min))
View(Season2O)
Season3O <- Season3Cleaned |>
  group_by(Role) |>
  summarise(S3_Average_Objective_Time = mean(Objective.Time...10min),
            S3_Average_Objective_Kills = mean(Objective.Kills...10min))
View(Season3O)
# Interestingly Damage lost Objective kills to Tanks and Tanks also carried objective
times
# Tank objective performance does, while having the lowest win and pick rates in all
seasons, suggests objectives are not significant to overall wins
# This also means that the lower objective time of supports and damage means they are
roaming outside the objectives
# The latter might be due to roaming ahead and creating spaces for objective advance that
are not recognized in the stats.
# Dueling is another option where players are seeking smaller engagements, but that would
have the effect of creating barrier areas around objectives anyway.

# by league rank

# dive into characters like pick and win rates top ten and their roles

# Season1
Top10HeroPicksS1 <- Season1Cleaned |>
```

```
  group_by(Hero) |>
  summarise(S1_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S1_Win_Rate) |>
  head(10)
Top10HeroPicksS1$Role <- c('Damage',
'Damage','Damage','Damage','Tank','Support','Tank','Damage','Damage','Support')
View(Top10HeroPicksS1)
# Not as strong a Support representation as I expected

# Season2
Top10HeroPicksS2 <- Season2Cleaned |>
  group_by(Hero) |>
  summarise(S2_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S2_Win_Rate) |>
  head(10)
Top10HeroPicksS2$Role <-
c('Damage','Damage','Tank','Tank','Damage','Support','Support','Tank','Support','Tank')
View(Top10HeroPicksS2)
# Still an even distribution

# Season3
Top10HeroPicksS3 <- Season3Cleaned |>
  group_by(Hero) |>
  summarise(S3_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S3_Win_Rate) |>
  head(10)
Top10HeroPicksS3$Role <-
c('Suport','Tank','Support','Damage','Tank','Support','Tank','Support','Damage','Support')
View(Top10HeroPicksS3)
Season_One_role <- c("6","2","2")
Season_Two_role <- c("3","4","3")
Season_Three_role <- c("2","3","5")
Top_Ten_Trend <- data.frame(Role, Season_One_role, Season_Two_role, Season_Three_role)
View(Top_Ten_Trend)
# Finally support influenced, but not as heavily as I expected
# This may be a delay in the player base to adapt to Support win rates


#Look at top and bottom ranks

# Season1
Top10HeroPicksS1STBronze <- Season1Cleaned |>
  group_by(Hero) |>
  filter(Skill.Tier == 'Bronze') |>
  summarise(S1_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S1_Win_Rate) |>
  head(10)
View(Top10HeroPicksS1STBronze)
Top10HeroPicksS1STGM <- Season1Cleaned |>
  group_by(Hero) |>
  filter(Skill.Tier == 'Grandmaster') |>
  summarise(S1_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S1_Win_Rate) |>
  head(10)
View(Top10HeroPicksS1STGM)

# Season2
Top10HeroPicksS2STBronze <- Season2Cleaned |>
  group_by(Hero) |>
  filter(Skill.Tier == 'Bronze') |>
  summarise(S2_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S2_Win_Rate) |>
  head(10)
View(Top10HeroPicksS2STBronze)
Top10HeroPicksS2STGM <- Season2Cleaned |>
```

```
  group_by(Hero) |>
  filter(Skill.Tier == 'Grandmaster') |>
  summarise(S2_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S2_Win_Rate) |>
  head(10)
View(Top10HeroPicksS2STGM)

# Season 3
Top10HeroPicksS3STBronze <- Season3Cleaned |>
  group_by(Hero) |>
  filter(Skill.Tier == 'Bronze') |>
  summarise(S3_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S3_Win_Rate) |>
  head(10)
View(Top10HeroPicksS3STBronze)
Top10HeroPicksS3STGM <- Season3Cleaned |>
  group_by(Hero) |>
  filter(Skill.Tier == 'Grandmaster') |>
  summarise(S3_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S3_Win_Rate) |>
  head(10)
View(Top10HeroPicksS3STGM)

# Top Support Heroes
# Season1
Top5HeroPicksS1Support <- Season1Cleaned |>
  group_by(Hero) |>
  filter(Role == 'Support') |>
  summarise(S1_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S1_Win_Rate) |>
  head(5)
View(Top5HeroPicksS1Support)

# Season2
Top5HeroPicksS2Support <- Season2Cleaned |>
  group_by(Hero) |>
  filter(Role == 'Support') |>
  summarise(S2_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S2_Win_Rate) |>
  head(5)
View(Top5HeroPicksS2Support)

# Season3
Top5HeroPicksS3Support <- Season3Cleaned |>
  group_by(Hero) |>
  filter(Role == 'Support') |>
  summarise(S3_Win_Rate = mean(Win.Rate...)) |>
  arrange(-S3_Win_Rate) |>
  head(5)
View(Top5HeroPicksS3Support)

# Within the top five support fo each season, three heroes repeated: Mercy, Lucio,
Brigitte.
# Final conclusion, to win more in Overwatch 2 and have a more successful tome climbing
the league ranks
# play support, stay off the objectives, and play specifically Mercy, Lucio, Brigitte
```