

STA 518 Final Reflection

Question 1: URL for Final Project

Github Code and Website Respository: <https://github.com/Robert-Bilyk/Final-Project>

Website Link: <https://robert-bilyk-sta518project.netlify.app/>

Question 2: Did you Work with a Group?

I worked alone on this project.

Question 3: Demonstration of Skills Learned

To demonstrate what I have learned, I will mainly reference 2 projects I worked on this semester. From these projects I will pull specific examples of the skills I employed to answer the questions the projects posed.

- The first project is the final project that I submitted. In that project I scraped data from the game World of Tanks for each individual tank and how well each individual tank performs. This was all done to see if there were any noticeable differences between the different playable nations, server the game is played on, tank tiers, and lastly the effect of premium tanks on the game.
- The second project was done to answer a question about dice rolling and what strategy is superior. In a game called dungeons and dragons there are two options you can choose. When you hit an enemy, you will roll a specific die to determine how much damage you deal. In the first option you can choose to simply take a flat +2 added to any damage roll. The second option, you can re roll any die that is a 2 or 1 but you have to take the second roll. To determine which of these was better I created a simulation for all of the different die combinations and simulated both options to see what choice appears to be better. I ran 1000 dice added them up and then repeated that 1000 times to get an adequately sized dataset.

Import, manage, and clean data:

For the World of Tanks project, the data I acquired was pulled from the Wot-News website. To do so, I wrote a function to pull the data. There were 4 different servers that I needed data from so I ran the function 4 times and then compiled them into a list. Finally, I used `reduce()` to combine the data horizontally into a single data set. Note that I also imported “other_info” which included premium tank status as I had to manually pull that data as there wasn’t any website that I could scrape this data from.

I also created a second data set that’s grouped on the specific tanks rather than the servers using the code below. This time, I combined them vertically using the `rbind()` function and added a new variable distinguish what server it came from. The code used here was almost identical to the server data set.

```
wotscrapewr <- function(x,y){
  tableda <- x %>%
    read_html() %>%
    html_nodes("table#stat_veh_all4") %>%
    html_table() %>%
    .[[1]] %>%
    setNames(c('Name','Tier','Type','Nation',paste('Total Played',y, sep=" "),
    paste('Wins',y, sep=" "), paste('Win %',y, sep=" "),
    paste('Unique Players',y, sep=" "), 'Region'))
```

```

}

wot_tableeu <- wotscrapewr("https://wot-news.com/stat/server/eu/norm/en/", "EU")
wot_tableus <- wotscrapewr("https://wot-news.com/stat/server/us/norm/en/", "US")
wot_tableru <- wotscrapewr("https://wot-news.com/stat/server/ru/norm/en/", "RU")
wot_tablesea <- wotscrapewr("https://wot-news.com/stat/server/sea/norm/en/", "SEA")
other_info <- read_csv("~/STA 518/Final-Project/tank_stats.csv") %>%
  select("Name", "Premium")

wot_list <- list(wot_tableeu, wot_tableus, wot_tableru, wot_tablesea, other_info)

tank_statstot <- wot_list %>% reduce(inner_join, by="Name")

```

After creating these data sets, I did a lot of work to make them presentable for graphs and tables. This includes:

- Removing Unnesesary Variables and Renaming Remaining ones

```

tank_statstot <- tank_statstot %>%
  select(-ends_with(".y"),-ends_with("x.x")) %>%
  rename("Tier"=`Tier.x`, "Nation"=`Nation.x`, "Type"=`Type.x`)

```

- Adding up all the servers together and calculating the average win rate for each tank globally

```

tank_statstot <- tank_statstot %>%
  mutate("Total Played" = `Total Played US` + `Total Played EU` +
    `Total Played SEA` + `Total Played RU`, "Total Wins" = `Wins US` +
    `Wins EU` + `Wins SEA` + `Wins RU`,
    "Total Unique Players" = `Unique Players US` + `Unique Players EU` +
    `Unique Players SEA` + `Unique Players RU`,
    "Win Rate %" = (`Total Wins`/`Total Played`)*100)

```

- Removing any duplicated variables and removing all tanks with less than 6,000 total battles

```

tank_statstot <- tank_statstot[!duplicated(tank_statstot$Name),] %>%
  filter(`Total Played` > 6000)

```

- Changing variables to character variables and reordering them for better visuals in the future

```

tank_statstot$`Tier` <- as.character(tank_statstot$`Tier`) %>%
  factor(levels=c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10"))
tank_statstot$`Premium` <- as.character(tank_statstot$`Premium`)

```

- Renaming the responses for certain categories for better visuals

```

tank_statstot <- tank_statstot%>%
  mutate(Type = fct_recode(Type,
    "Medium Tank" = "Medium Tanks",
    "Heavy Tank" = "Heavy Tanks",
    "Light Tank" = "Light Tanks",
    "Tank Destroyer" = "TD",
    "Artillery" = "SPG")) %>%

  mutate(Nation = fct_recode(Nation,
    "USA" = "Usa",
    "USSR" = "Ussr")) %>%

  mutate(Premium = fct_recode(Premium,

```

```
"No" = "0",
"Yes" = "1"))
```

All of this created the data sets shown below that could easily be called to create tables and graphs for the different servers and the different tanks.

Data Set for Tanks

```
## # A tibble: 971 x 30
##   Name      Tier.x Type.x  Nation.x `Total Played E~` `Wins EU` `Win % EU`
##   <chr>      <int> <chr>    <chr>      <int>      <int> <chr>
## 1 Tiger II      8 Heavy Ta~ Germany    605267    283901 46.91 %
## 2 Maus          10 Heavy Ta~ Germany    228165    112397 49.26 %
## 3 VK 36.01 (H)   6 Heavy Ta~ Germany    431108    216392 50.19 %
## 4 G.W. E 100     10 SPG      Germany    335550    165057 49.19 %
## 5 Hummel         6 SPG      Germany    327391    159269 48.65 %
## 6 G.W. Tiger     9 SPG      Germany    228558    112219 49.1 %
## 7 VK 45.02 (P)~  9 Heavy Ta~ Germany    423070    207047 48.94 %
## 8 E 50           9 Medium T~ Germany    466249    236179 50.66 %
## 9 E 100          10 Heavy Ta~ Germany    561181    269958 48.11 %
## 10 Panther II    8 Medium T~ Germany    243038    108550 44.66 %
## # ... with 961 more rows, and 23 more variables: Unique Players EU <int>,
## #   Tier.y <int>, Type.y <chr>, Nation.y <chr>, Total Played US <int>,
## #   Wins US <int>, Win % US <chr>, Unique Players US <int>, Tier.x.x <int>,
## #   Type.x.x <chr>, Nation.x.x <chr>, Total Played RU <int>, Wins RU <int>,
## #   Win % RU <chr>, Unique Players RU <int>, Tier.y.y <int>, Type.y.y <chr>,
## #   Nation.y.y <chr>, Total Played SEA <int>, Wins SEA <int>, Win % SEA <chr>,
## #   Unique Players SEA <int>, Premium <dbl>
```

Data Set for Servers

```
## # A tibble: 2,811 x 10
##   Name      Tier Type  Nation `Total played`  Win `Vehicles amoun~` Server
##   <chr>      <fct> <chr>  <chr>      <int>  <int>      <int> <chr>
## 1 Tiger II    8   Heavy ~ Germa~    605267 283901      3061 EU
## 2 Maus        10   Heavy ~ Germa~    228165 112397      1292 EU
## 3 VK 36.01 ~ 6   Heavy ~ Germa~    431108 216392      4430 EU
## 4 G.W. E 100  10   SPG    Germa~    335550 165057       803 EU
## 5 Hummel       6   SPG    Germa~    327391 159269      3044 EU
## 6 G.W. Tiger  9   SPG    Germa~    228558 112219      1025 EU
## 7 VK 45.02 ~ 9   Heavy ~ Germa~    423070 207047      6455 EU
## 8 E 50         9   Medium~ Germa~    466249 236179      1995 EU
## 9 E 100        10   Heavy ~ Germa~    561181 269958      2405 EU
## 10 Panther II  8   Medium~ Germa~    243038 108550       2091 EU
## # ... with 2,801 more rows, and 2 more variables: Winrate <dbl>, Premium <fct>
```

Lastly, using the dice project functions I created (will be shown later in this reflection), I ended up with this data set which wasn't very good for visuals so I pivoted it.

```
##   Normal_d6 With_GWF_d6 Normal_d8 With_GWF_d8 Normal_d10 With_GWF_d10
## 1      3550      4161      4548      5304      5484      6240
## 2      3478      4219      4609      5325      5395      6362
## 3      3583      4125      4510      5261      5384      6523
## 4      3466      4175      4541      5167      5633      6230
## 5      3464      4144      4606      5264      5464      6304
## 6      3414      4189      4544      5261      5533      6212
```

```
## 7      3454      4205      4543      5272      5572      6321
## 8      3580      4233      4571      5237      5556      6254
## 9      3505      4235      4621      5275      5506      6136
## 10     3590      4157      4551      5164      5524      6193
```

```
final <- dataframe %>%
  pivot_longer(
    cols = c(Normal_d6, With_GWF_d6, Normal_d8, With_GWF_d8, Normal_d10,
              With_GWF_d10, Normal_2d6, With_GWF_2d6),
    names_to = "Roll_Type",
    values_to = "Sum_of_Rolls"
  )
```

This changed the data set to the following so visuals could be easily created.

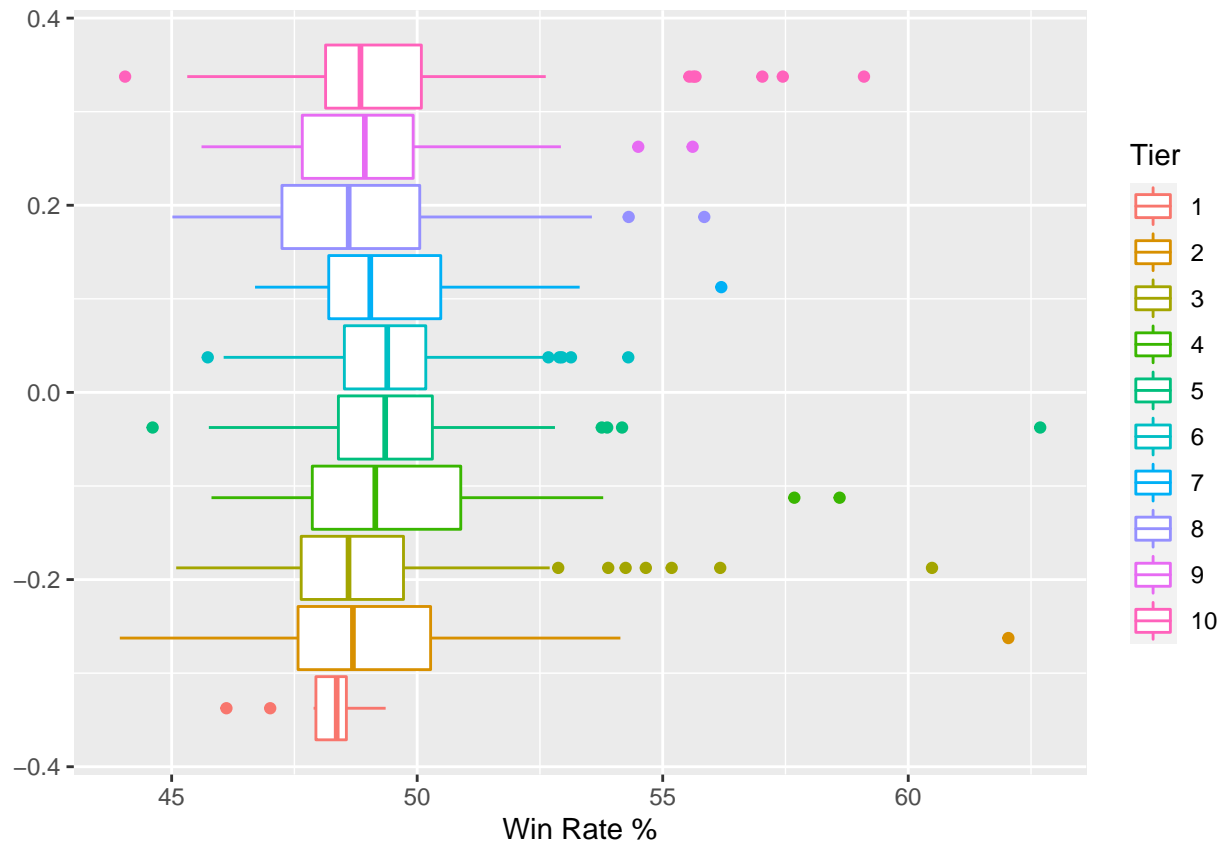
```
## # A tibble: 10 x 2
##   Roll_Type    Sum_of_Rolls
##   <fct>         <int>
## 1 Normal_d6      3550
## 2 With_GWF_d6   4161
## 3 Normal_d8     4548
## 4 With_GWF_d8   5304
## 5 Normal_d10    5484
## 6 With_GWF_d10  6240
## 7 Normal_2d6    7023
## 8 With_GWF_2d6  8276
## 9 Normal_d6     3478
## 10 With_GWF_d6  4219
```

Creating Visuals and Numerical Summaries

I created numerous summaries and visuals for the final project. Here are a few examples.

Comparative Boxplots

```
tank_statstot %>%
  ggplot(mapping=aes(x=`Win Rate %`, group=Tier, color=Tier)) +
  geom_boxplot()
```



Numerical Summaries of Means

```
tank_statstot %>%
  group_by(Tier) %>%
  summarise("Mean Percent by Tank Type" = mean(`Win Rate %`),
            "Total Number of Tier" = length(`Win Rate %`))
```

```
## # A tibble: 10 x 3
##   Tier `Mean Percent by Tank Type` `Total Number of Tier`
##   <dbl> <dbl> <int>
## 1 1 48.1 12
## 2 2 49.1 51
## 3 3 49.3 56
## 4 4 49.6 53
## 5 5 49.7 67
## 6 6 49.5 79
## 7 7 49.5 74
## 8 8 48.7 156
## 9 9 49.0 69
## 10 10 49.4 79
```

ANOVA Tests

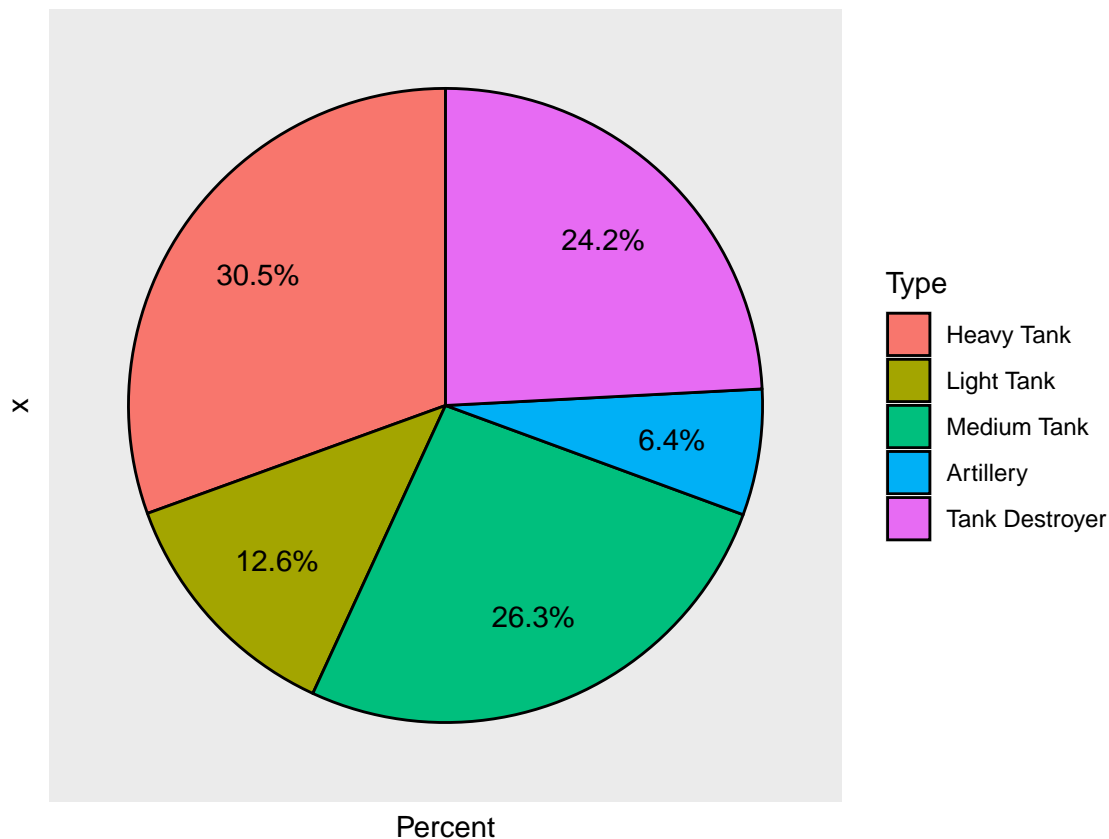
```
aov(`Win Rate %` ~ Tier, data = tank_statstot) %>%
  summary()
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
```

```
## Tier          9      95 10.541   2.107 0.0269 *
## Residuals    686    3432   5.002
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pie Charts

```
tank_statstot %>%
  group_by(Type) %>%
  summarise(`Type Played` = sum(`Total Played`)) %>%
  mutate(Percent = `Type Played` / sum(`Type Played`)) %>%
  mutate(labels = scales::percent(Percent)) %>%
  group_by(Type)%>%
  ggplot(aes(x = "", y = Percent, fill = Type)) +
  geom_col(color="black") +
  geom_text(aes(x=1.2, label = labels),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        panel.grid = element_blank())
```



Comparative Pie Charts

```
Server_pct <- function(x, y){
  tank_statsreg %>%
    filter(Server == x) %>%
```

```

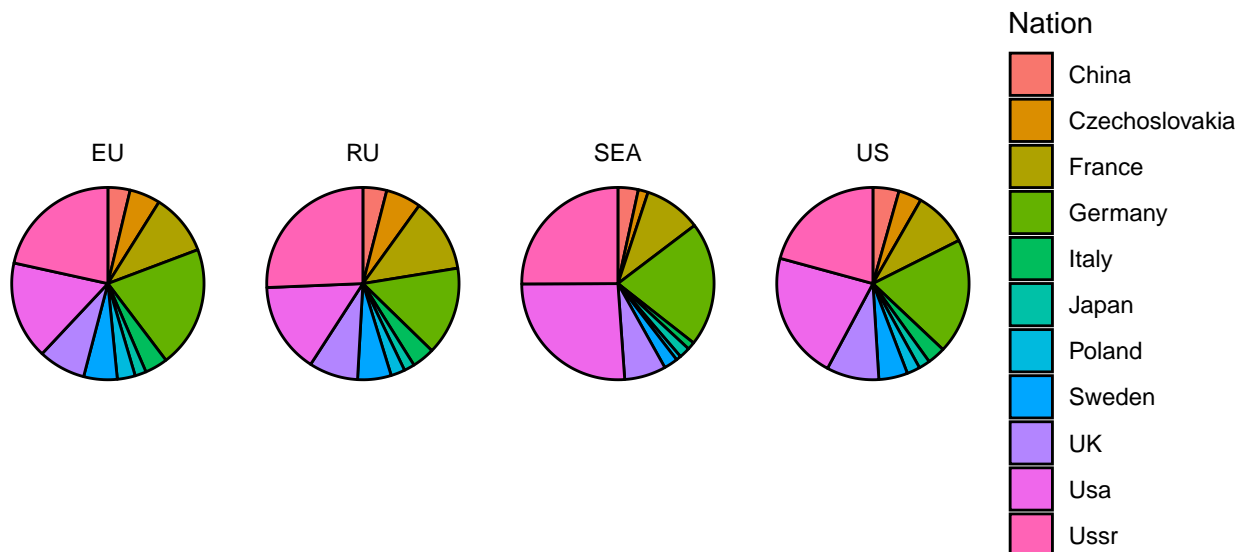
group_by({{y}}) %>%
  summarise(hold = sum(`Total played`), Server = {{x}}) %>%
  mutate(Percent = hold / sum(hold)) %>%
  mutate(labels = scales::percent(Percent))
}

NSEA <- Server_pct("SEA", Nation)
NRU <- Server_pct("RU", Nation)
NUS <- Server_pct("US", Nation)
NEU <- Server_pct("EU", Nation)

regnat <- list(NSEA, NRU, NUS, NEU)
regpnat <- do.call(rbind, regnat)

regpnat %>%
  ggplot(aes(x = "", y = Percent, fill = Nation, group = Server)) +
  geom_col(color="black") +
  coord_polar(theta = "y") +
  facet_grid(~ Server) + theme_void() +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        panel.grid = element_blank())

```



Write R Programs for Simulations w/ Randomization Based Experiments

Moving back to the dice project, I used the following code to create a function that first simply rolled a certain number of dice that had a certain number of sides. This is because in D&D many different sided die are used.

```

normal <- function(x,y){
  Normal_2d <- 1:1000; Normal_2d
  for(l in 1:1000){
    sumofall2d <- 1:1000; sumofall2d
    for(i in 1:1000){
      sum2d <- 0
      roll2d <- sample(1:x, y, replace=TRUE)

```

```

        sum2d <- sum(roll2d)
        sumofall2d[i] <- sum2d
    }
    Normal_2d[1] <- sum(sumofall2d)
}
return(Normal_2d)
}

```

Then I wrote a separate function to re roll any 1's and 2's and take the new roll. I wrote separate functions for one and two die.

1 Dice _

```

gwf_rolls <- function(x){
With_GWF <- 1:1000; With_GWF
for(l in 1:1000){
    sum <- 0
    roll <- sample(1:x, 1000, replace=TRUE)
    for(i in 1:1000){
        if(roll[i]<3){
            roll[i] <- sample(1:x, 1, replace=TRUE)
        }
    }
    sum <- sum(roll)
    With_GWF[l] <- sum
}
return(With_GWF)
}

```

2 Die _

```

gwf_2dice <- function(x){
totalroll <- 1:1000; totalroll
With_GWF_2d <- 1:1000; With_GWF_2d
for(l in 1:1000){
    for(i in 1:1000){
        roll2d <-sample(1:x, 2, replace=TRUE)
        if(roll2d[1]<3){
            roll2d[1] <-sample(1:x, 1, replace=TRUE)
        }
        if(roll2d[2]<3){
            roll2d[2] <-sample(1:x, 1, replace=TRUE)
        }
        sumroll <- sum(roll2d)
        totalroll[i] <- sumroll
    }
    finalroll <- sum(totalroll)
    With_GWF_2d[l] <- finalroll
}
return(With_GWF_2d)
}

```

With these functions written I ran them for all of the different dice combinations that would be encountered. Putting these all into a data frame and pivoting it as shown earlier, I was able to create the following box plot and table describing the average damage improvement from taking option 2.

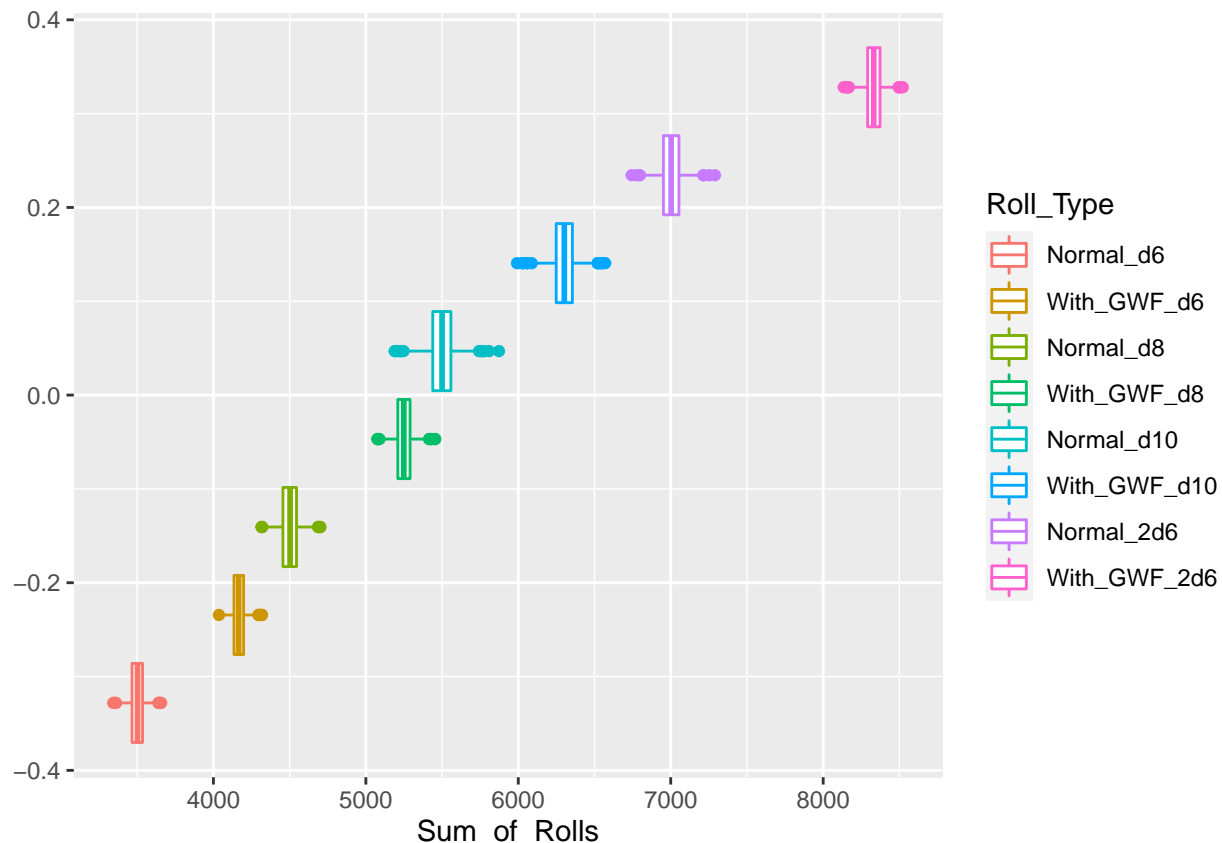


Table of Improvement from Picking Option 2

```
## # A tibble: 1 x 4
##   d6dmgincrease d8dmgincrease d10dmgincrease d2d6dmgincrease
##           <dbl>           <dbl>           <dbl>           <dbl>
## 1           0.958           1.14           1.27           1.92
```

This all appears to indicate that picking option 1 for the flat +2 damage increase is better than taking the second option as all common dice combinations result in a damage increase of less than 2.

Using Source Documentation to Troubleshoot and Extend R Programs

I had to use numerous sources both in R and using outside sources to accomplish tasks.

Stack Overflow was easily the most helpful for general advice on issues and simple syntax errors: <https://stackoverflow.com/>

For web scraping, I found a fantastic youtube series which really helped me learn to web scrape: https://www.youtube.com/watch?v=v8Yh_4oE-Fs&list=PLr5uaPu5L7xLEclrT0-2TWaz5FTkfdUiW

Finally, another helpful source for setting up my website was this web article: <https://lisalendway.netlify.app/posts/2020-12-09-buildingdistill/>

All of this, allowed me to improve my skills in web scraping and web design which really allowed me to enhance my project and any future projects I will do in R.

Write Clear, Efficient and Well Documented R Programs

I generally make well organized R code that is easy to read and has comments clearly explaining what each section of code does. On top of this, all edits are available through looking at the github history. The

following is the github files for both the dice project and World of Tanks project for you to see how it is generally organized.

World of Tanks: https://github.com/Robert-Bilyk/Final-Project/blob/main/FinalProject_Code.R

Dice: https://github.com/Robert-Bilyk/Final-Project/blob/main/Final-Reflection_Examples/dicetest.R

Question 4: Grade I Think I Deserve

I would overall give myself an A. I am very confident in all of the topics we covered and I know that I can adequately accomplish any tasks related to modifying data and displaying it. My biggest issue is forgetting the necessary syntax with different R commands. However, using the ? in R and using online resources mostly covers any major issues I have on that front.

Question 5: Thoughts or Reflections

My personal preference is to have more structure to classes and actually having grades but this class was certainly effective in teaching me the necessary skills needed to code in R. I don't have any major criticisms with how the class is run. I did have a lot of fun learning to use R and I really valued the time and effort I put into the class.