CSGO eda

Jesse Coulson, Adam Sanden

2024-03-18

Load the data (Jesse)

```
df <-read.csv("csgo_players.csv")

#turn dataframe into tibble
csgo_data <- as_tibble(df)
csgo_data
```

```
# A tibble: 383,317 x 101
   date       player_name team      opponent country player_id match_id event_id
   <chr>      <chr>       <chr>     <chr>    <chr>       <int>    <int>    <int>
 1 2020-02-26 Brehze      Evil Gen~ Liquid   United~      9136  2339385     4901
 2 2020-02-26 CeRq        Evil Gen~ Liquid   Bulgar~     11219  2339385     4901
 3 2020-02-26 EliGE       Liquid    Evil Ge~ United~      8738  2339385     4901
 4 2020-02-26 Ethan       Evil Gen~ Liquid   United~     10671  2339385     4901
 5 2020-02-26 NAF         Liquid    Evil Ge~ Canada       8520  2339385     4901
 6 2020-02-26 Stewie2K    Liquid    Evil Ge~ United~      8797  2339385     4901
 7 2020-02-26 Twistzz     Liquid    Evil Ge~ Canada      10394  2339385     4901
 8 2020-02-26 nitr0       Liquid    Evil Ge~ United~      7687  2339385     4901
 9 2020-02-26 stanislaw   Evil Gen~ Liquid   Canada       8507  2339385     4901
10 2020-02-26 tarik       Evil Gen~ Liquid   United~      8523  2339385     4901
# i 383,307 more rows
# i 93 more variables: event_name <chr>, best_of <int>, map_1 <chr>,
#   map_2 <chr>, map_3 <chr>, kills <int>, assists <int>, deaths <int>,
#   hs <int>, flash_assists <dbl>, kast <dbl>, kddiff <int>, adr <dbl>,
#   fkdiff <int>, rating <dbl>, m1_kills <int>, m1_assists <int>,
#   m1_deaths <int>, m1_hs <int>, m1_flash_assists <dbl>, m1_kast <dbl>,
#   m1_kddiff <int>, m1_adr <dbl>, m1_fkdiff <int>, m1_rating <dbl>, ...
```

```
df2<-read.csv("csgo_results.csv")
results_data <-as_tibble(df2)


head(results_data)
```

```
# A tibble: 6 x 19
  date  team_1 team_2 X_map result_1 result_2 map_winner starting_ct  ct_1   t_2
  <chr> <chr>  <chr>  <chr>    <int>    <int>      <int>       <int> <int> <int>
1 2020~ Recon~ TeamO~ Dust2        0       16          2           2     0     1
2 2020~ Recon~ TeamO~ Infe~       13       16          2           2     8     6
3 2020~ New E~ Stati~ Infe~       12       16          2           1     9     6
4 2020~ Rugra~ Bad N~ Infe~        7       16          2           2     0     8
5 2020~ Rugra~ Bad N~ Vert~        8       16          2           2     4     5
6 2020~ Singu~ Endpo~ Over~       13       16          2           2     8     6
# i 9 more variables: t_1 <int>, ct_2 <int>, event_id <int>, match_id <int>,
#   rank_1 <int>, rank_2 <int>, map_wins_1 <int>, map_wins_2 <int>,
#   match_winner <int>
```

Since only some events have a best of 3 rule-set, we decided to remove the individual map statistics and just keep the overall statistics and take the average kills deaths, assists, hs, kddiff fkdiff for the whole match, otherwise any 3 game match would skew the data compared to a 1 game match. for the sake of the EDA we will be looking only at Single game matches to avoid match stats being skewed.

#remove unecessary columns

```
#get rid of individual round data (Jesse)
csgo_data <- df %>% select(player_name, team, opponent, country, player_id, match_id, even
head(csgo_data)
```

```
  player_name          team      opponent       country player_id match_id
1      Brehze Evil Geniuses        Liquid United States      9136  2339385
2        CeRq Evil Geniuses        Liquid      Bulgaria     11219  2339385
3       EliGE        Liquid Evil Geniuses United States      8738  2339385
4       Ethan Evil Geniuses        Liquid United States     10671  2339385
5         NAF        Liquid Evil Geniuses        Canada      8520  2339385
6    Stewie2K        Liquid Evil Geniuses United States      8797  2339385
  event_id        event_name best_of kills assists deaths hs kddiff fkdiff
1     4901 IEM Katowice 2020       3    57      14     61 29     -4      0
2     4901 IEM Katowice 2020       3    54      10     54 18      0      2
```

```
3     4901 IEM Katowice 2020       3    55      10     51 28      4      1
4     4901 IEM Katowice 2020       3    43       5     54 18    -11     -4
5     4901 IEM Katowice 2020       3    52      22     46 23      6     -1
6     4901 IEM Katowice 2020       3    56       8     54 31      2      1
  rating
1   1.04
2   0.98
3   1.08
4   0.83
5   1.08
6   1.08
```

```
#make a separate data set to focus just on 1 game matches
csgo_single <- df %>%
  filter(best_of == 1)
csgo_single <- csgo_single %>%
  select(player_name, team, opponent, country, player_id, match_id, event_id,event_name, b
head(csgo_single)
```

```
  player_name          team      opponent       country player_id match_id
1    Andersin Thunder Logic      Station7 United States     14038  2339816
2    FrostayK      Station7 Thunder Logic United States     12090  2339816
3   Inseaniac Thunder Logic      Station7        Canada     18623  2339816
4      JonahP      Station7 Thunder Logic        Canada     11445  2339816
5       PureR Thunder Logic      Station7 United States     10622  2339816
6     Sharkie Thunder Logic      Station7 United States     19476  2339816
  event_id                        event_name best_of kills assists deaths hs
1     5151 ESEA MDL Season 33 North America       1    21       3     10 12
2     5151 ESEA MDL Season 33 North America       1     9       0     16  8
3     5151 ESEA MDL Season 33 North America       1    20       4      7  6
4     5151 ESEA MDL Season 33 North America       1     5       2     18  2
5     5151 ESEA MDL Season 33 North America       1    18       3      9  7
6     5151 ESEA MDL Season 33 North America       1    13       5      7  4
  kddiff fkdiff rating
1     11      4   1.95
2     -7     -1   0.52
3     13      4   1.91
4    -13     -4   0.31
5      9      2   1.62
6      6      2   1.48
```

We got the average rating for each player and 1 standard deviation above and below. #Explore rating var

```
#get average rating
mean <- mean(csgo_data$rating)

#get one standrd deviation above average rating
sd_mean <- mean + sd(csgo_single$rating)
sd_mean
```

```
[1] 1.404905
```

```
sd_mean_2 <- sd_mean + sd(csgo_single$rating)
sd_mean_2
```

```
[1] 1.757098
```

```
sd_mean_negative_2 <- sd_mean - 2*  sd(csgo_single$rating)
sd_mean_negative_2
```

```
[1] 0.7005197
```

```
#check how many games dataset has
length(unique(csgo_single$match_id))
```

```
[1] 18526
```

We find outliers in our dataset #Show outliers in rating

```
#no missing values in player rating
sum(is.na(csgo_single$rating))
```

```
[1] 0
```

```
#missing values for games that did not reach max rounds
sum(is.na(csgo_single))
```

```
[1] 0
```

```r
ratings.outliers <- filter(csgo_single, rating < sd_mean_negative_2 | rating>sd_mean_2)
head(ratings.outliers)
```

```
  player_name          team      opponent       country player_id match_id
1    Andersin Thunder Logic      Station7 United States     14038  2339816
2    FrostayK      Station7 Thunder Logic United States     12090  2339816
3   Inseaniac Thunder Logic      Station7        Canada     18623  2339816
4      JonahP      Station7 Thunder Logic        Canada     11445  2339816
5       shonk      Station7 Thunder Logic United States     14022  2339816
6    sterling      Station7 Thunder Logic        Canada     14613  2339816
  event_id                        event_name best_of kills assists deaths hs
1     5151 ESEA MDL Season 33 North America       1    21       3     10 12
2     5151 ESEA MDL Season 33 North America       1     9       0     16  8
3     5151 ESEA MDL Season 33 North America       1    20       4      7  6
4     5151 ESEA MDL Season 33 North America       1     5       2     18  2
5     5151 ESEA MDL Season 33 North America       1     7       0     18  4
6     5151 ESEA MDL Season 33 North America       1     8       4     17  4
  kddiff fkdiff rating
1     11      4   1.95
2     -7     -1   0.52
3     13      4   1.91
4    -13     -4   0.31
5    -11     -2   0.32
6     -9     -5   0.44
```

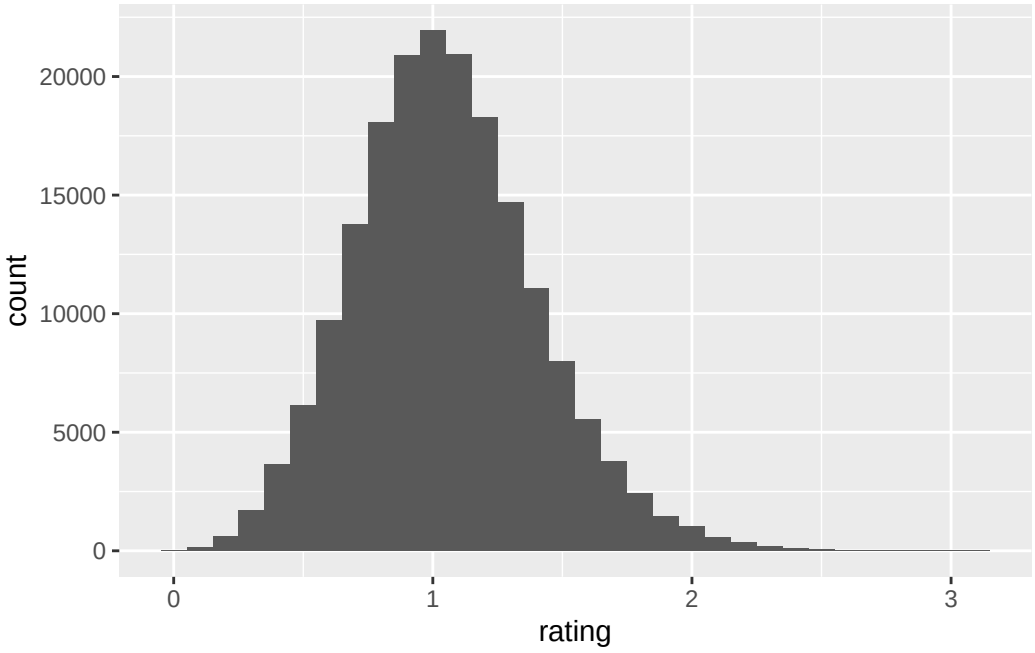
```r
csgo_arranged <- csgo_single%>%
  group_by(match_id) %>%
  arrange(match_id)

head(csgo_arranged)
```

```
# A tibble: 6 x 16
# Groups:   match_id [1]
  player_name team   opponent country player_id match_id event_id event_name
  <chr>       <chr>  <chr>    <chr>       <int>    <int>    <int> <chr>
1 AcilioN     SK     fnatic   Denmark      8151  2298484     1934 ESL ESEA Pro ~
2 Friis       SK     fnatic   Denmark         7  2298484     1934 ESL ESEA Pro ~
3 JW          fnatic SK       Sweden       3849  2298484     1934 ESL ESEA Pro ~
4 KRIMZ       fnatic SK       Sweden       7528  2298484     1934 ESL ESEA Pro ~
5 TENZKI      SK     fnatic   Denmark      5287  2298484     1934 ESL ESEA Pro ~
```

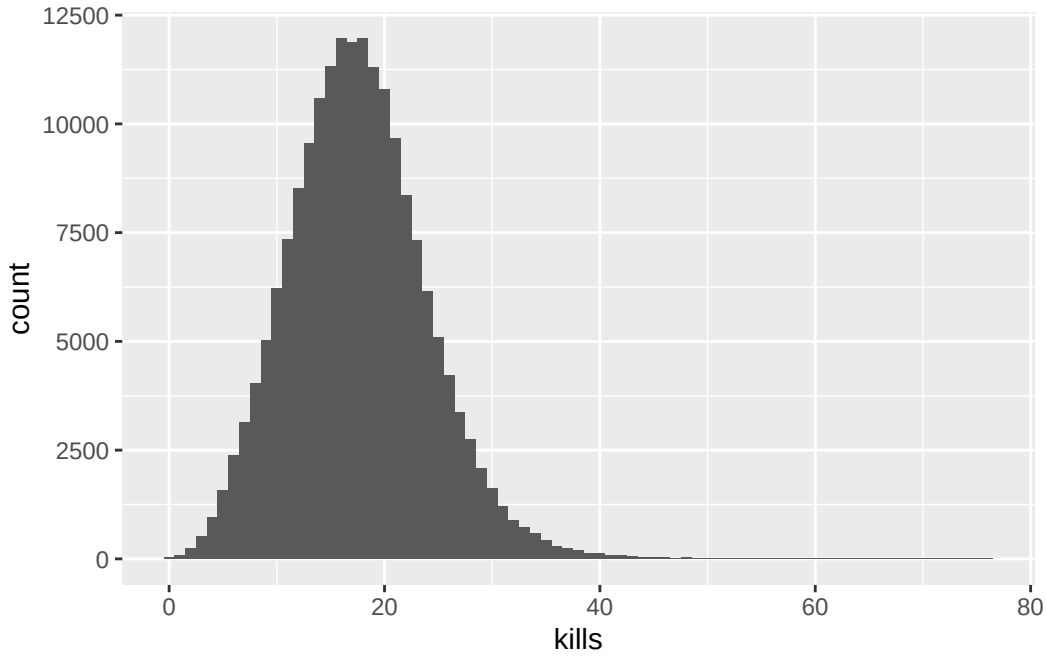
```
6 cadiaN      SK     fnatic   Denmark      7964  2298484     1934 ESL ESEA Pro ~
# i 8 more variables: best_of <int>, kills <int>, assists <int>, deaths <int>,
#   hs <int>, kddiff <int>, fkdiff <int>, rating <dbl>
```

Histograms of raw numebers #simple histograms

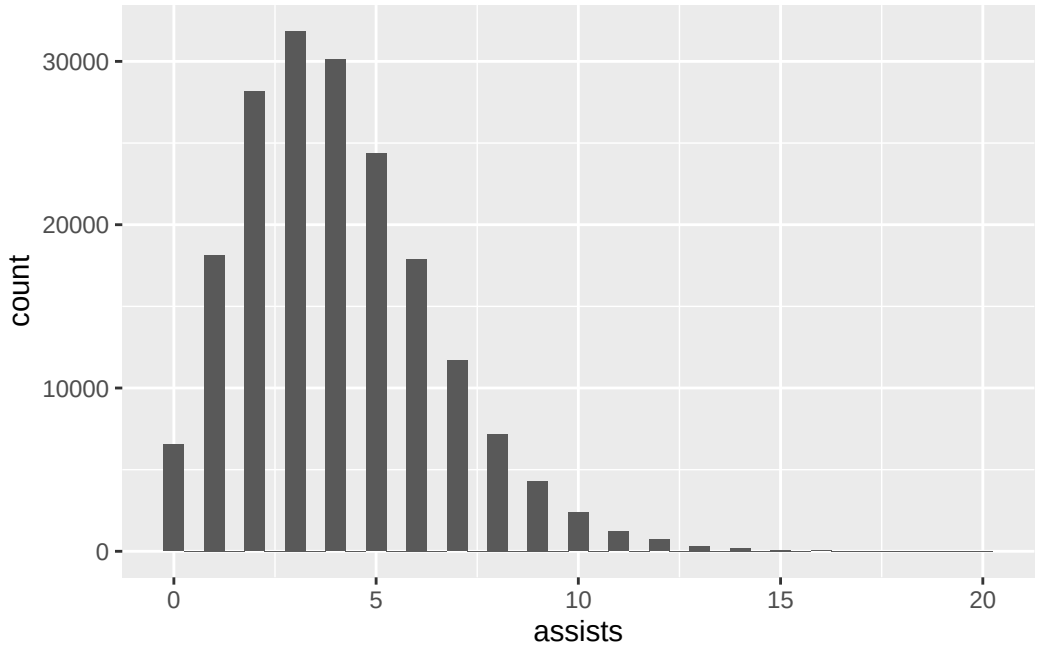
```
#rating
ggplot(csgo_single, aes(rating)) + geom_histogram(binwidth = 0.1)
```



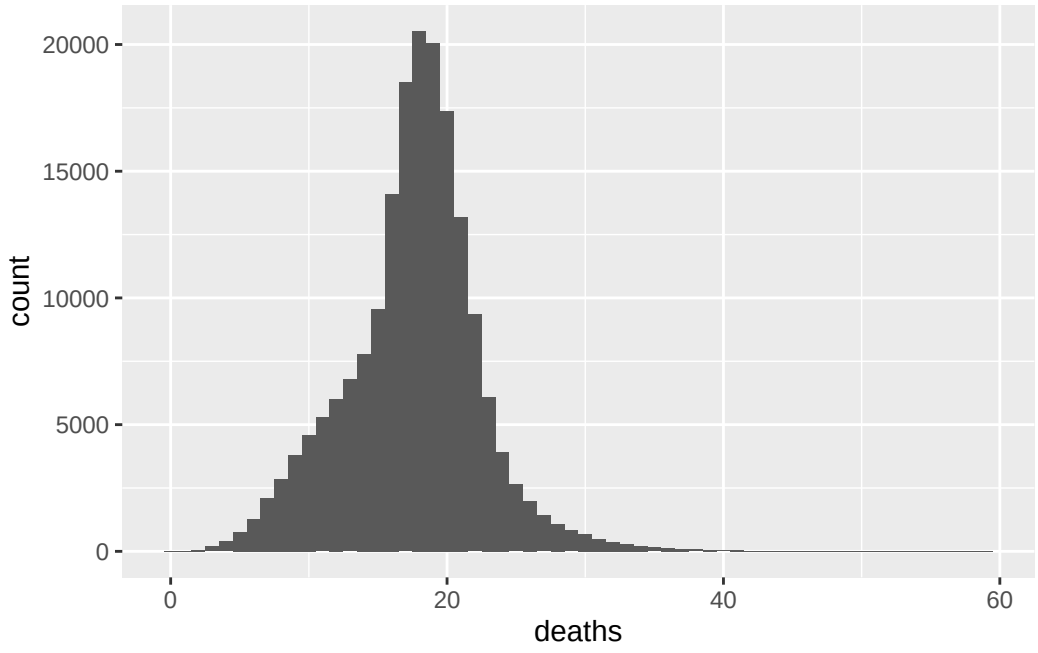

```
#kills
ggplot(csgo_single, aes(kills)) + geom_histogram(binwidth = 1)
```



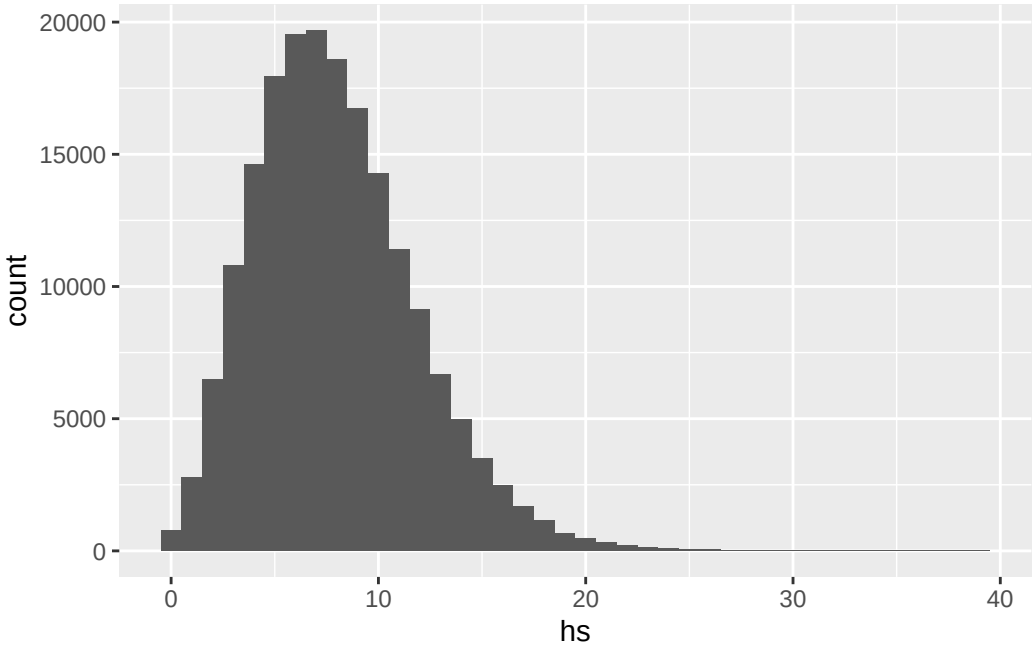

```
#assists
ggplot(csgo_single, aes(assists)) + geom_histogram(binwidth = .5)
```



```
#deaths
ggplot(csgo_single, aes(deaths)) + geom_histogram(binwidth = 1)
```

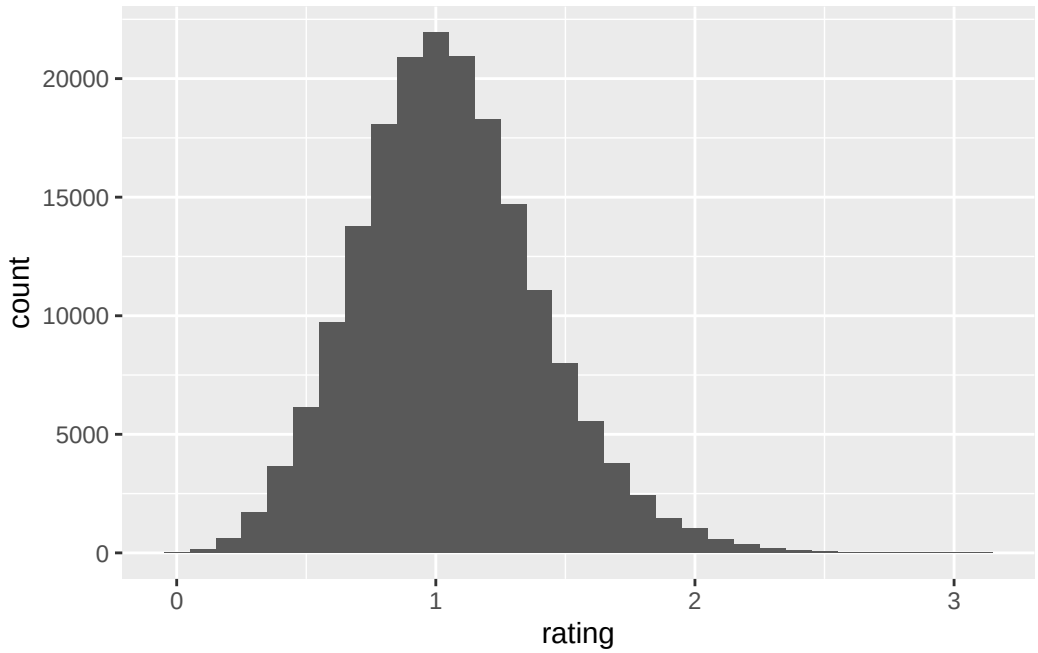



```
#headshot
ggplot(csgo_single, aes(hs)) + geom_histogram(binwidth = 1)
```

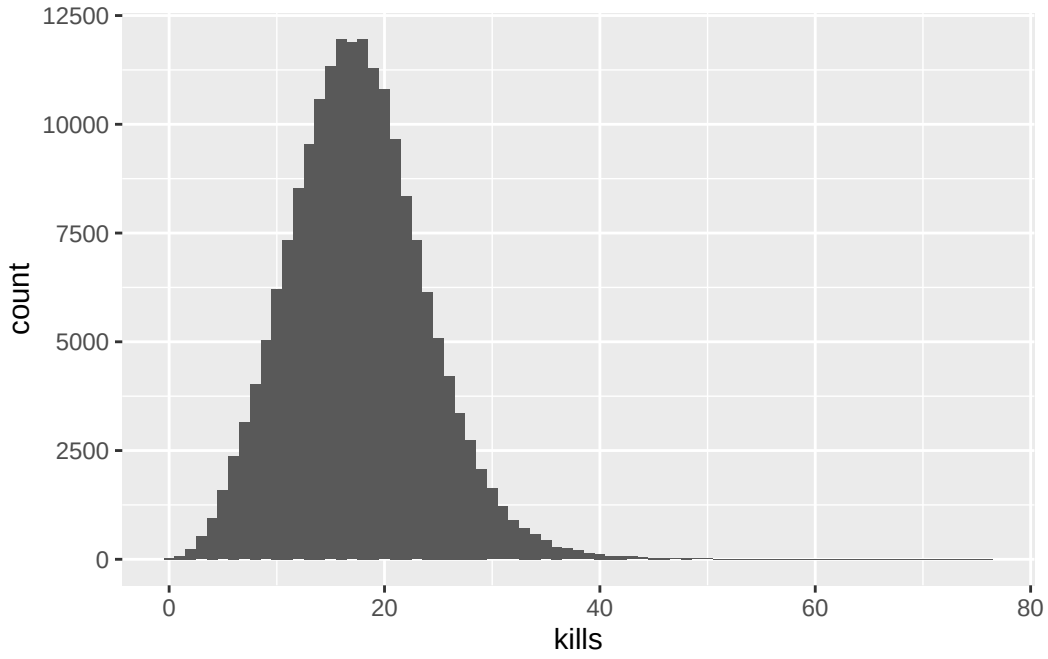


Histograms of average stats per game

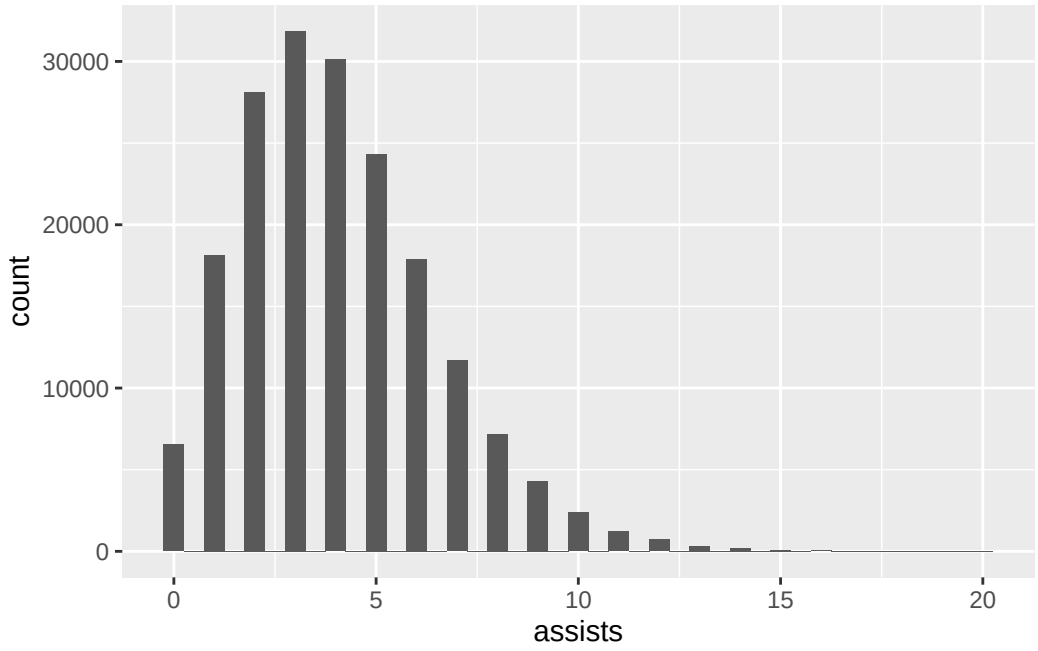
```
#rating
ggplot(csgo_single, aes(rating)) + geom_histogram(binwidth = 0.1)
```



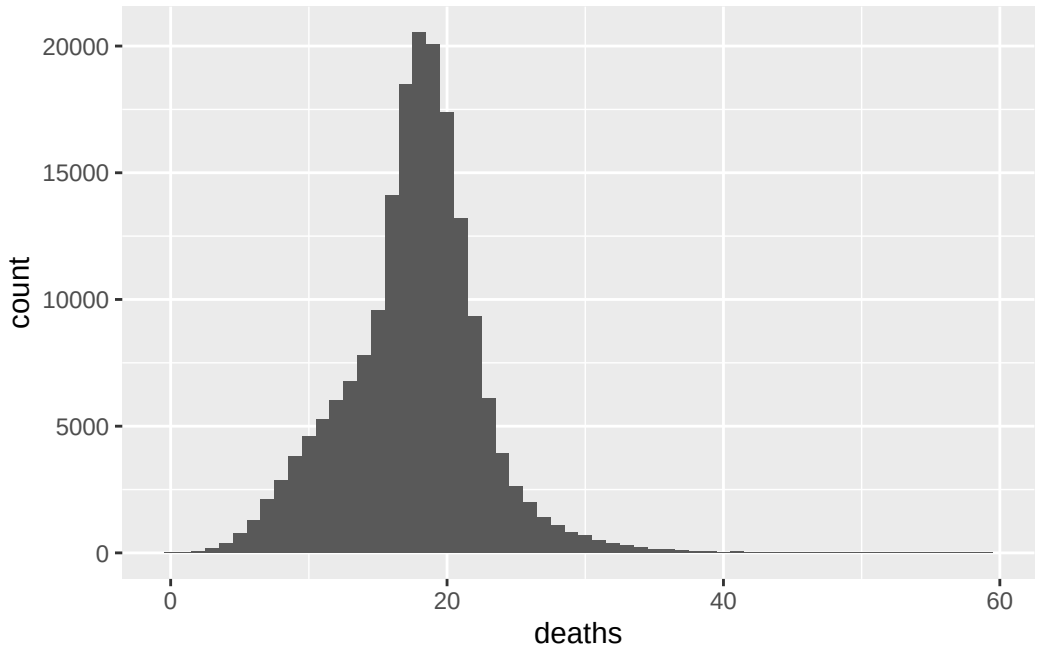

```
#avg kills
ggplot(csgo_single, aes(kills)) + geom_histogram(binwidth = 1)
```



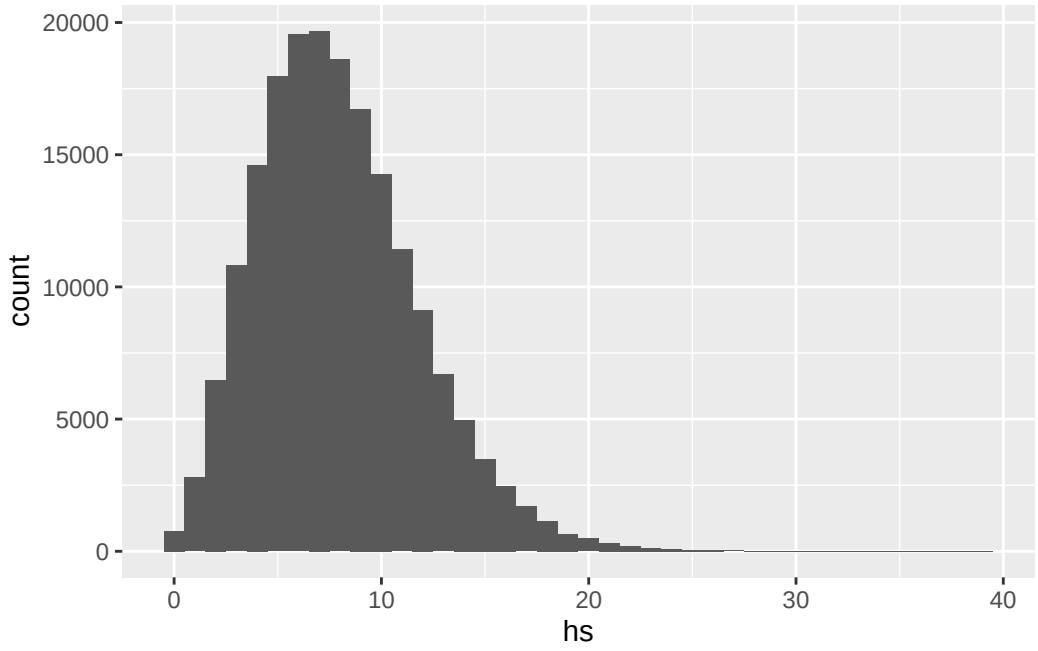
```r
#avg assists
ggplot(csgo_single, aes(assists)) + geom_histogram(binwidth = .5)
```




```r
#avg deaths
ggplot(csgo_single, aes(deaths)) + geom_histogram(binwidth = 1)
```




```
#avg headshot
ggplot(csgo_single, aes(hs)) + geom_histogram(binwidth = 1)
```



#calculating kda

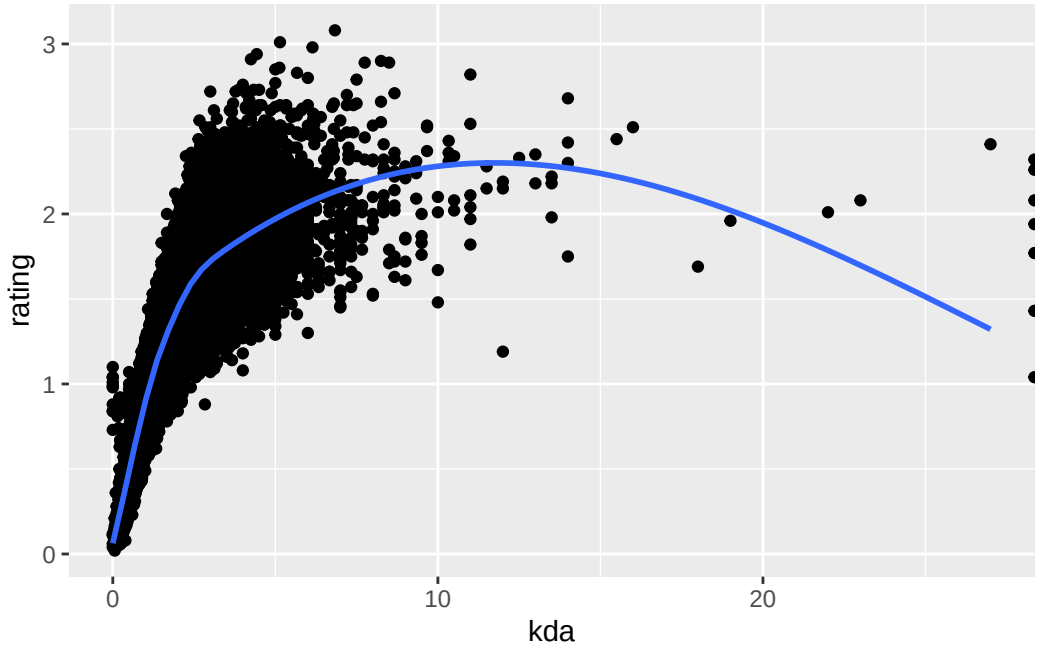
```
#Creating a column for kda
csgo_single <- csgo_single %>%
  mutate(kda = (kills + assists)/deaths)
```

#see relationship between kda and given rating

```
ggplot(csgo_single, aes(x = kda, y = rating)) +
  geom_point() +
 geom_smooth(aes(x = kda, y= rating ),se=FALSE)
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
Warning: Removed 7 rows containing non-finite outside the scale range
(`stat_smooth()`).
```



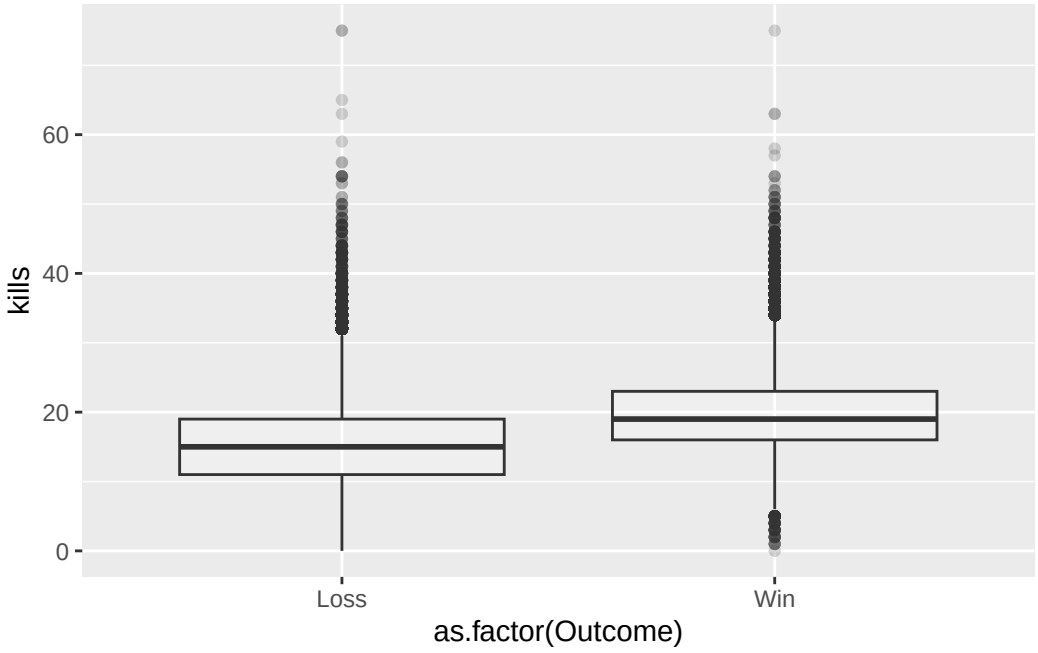
Join the match results data with the player statistics dataset in order to measure how good of a predictor each statistic is for predicitng whether a team wins or loses a match # Add win column using results dataset

```r
#find winning team from results columns
results_data <- results_data %>%
  mutate(winner = ifelse(result_1 > result_2, team_1, team_2))


csgo_single <- csgo_single %>%
  left_join(results_data, by = c("match_id"), relationship = 'many-to-many') %>%
  mutate(Outcome = ifelse(team == winner, "Win", "Loss"))


csgo_single <- csgo_single %>% select(player_name, date, team, opponent, country, player_i

#csgo_single
csgo_single <- na.omit(csgo_single) #removing na rows

head(csgo_single)
```

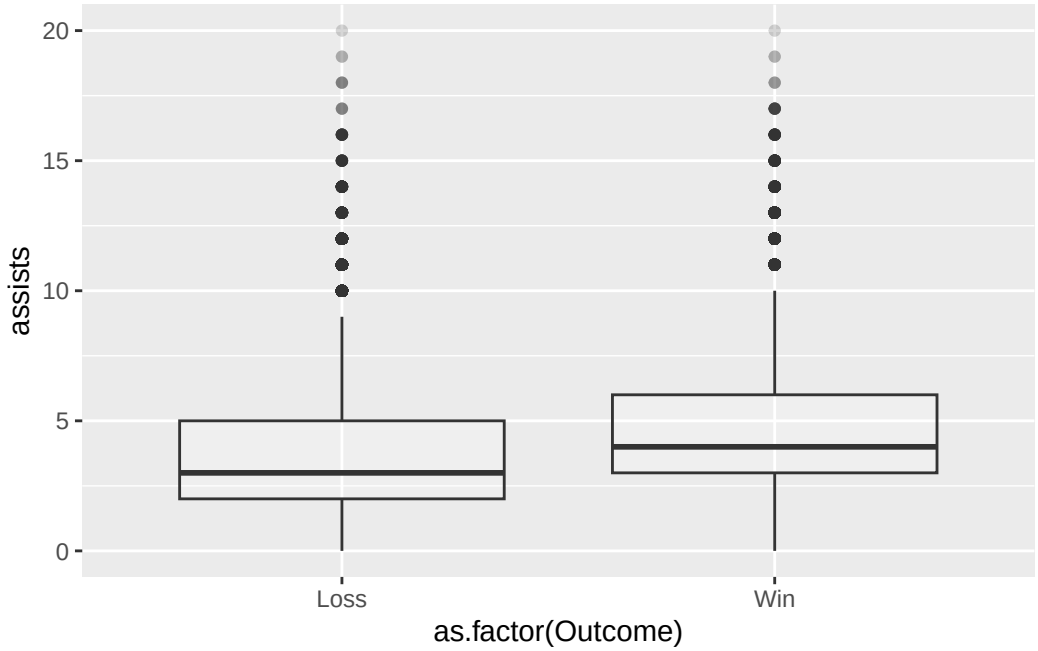
```
  player_name       date          team      opponent       country player_id
1    Andersin 2020-02-27 Thunder Logic      Station7 United States     14038
2    FrostayK 2020-02-27      Station7 Thunder Logic United States     12090
3   Inseaniac 2020-02-27 Thunder Logic      Station7        Canada     18623
4      JonahP 2020-02-27      Station7 Thunder Logic        Canada     11445
5       PureR 2020-02-27 Thunder Logic      Station7 United States     10622
6     Sharkie 2020-02-27 Thunder Logic      Station7 United States     19476
  match_id event_id.x                       event_name kills       kda assists
1  2339816       5151 ESEA MDL Season 33 North America    21 2.4000000       3
2  2339816       5151 ESEA MDL Season 33 North America     9 0.5625000       0
3  2339816       5151 ESEA MDL Season 33 North America    20 3.4285714       4
4  2339816       5151 ESEA MDL Season 33 North America     5 0.3888889       2
5  2339816       5151 ESEA MDL Season 33 North America    18 2.3333333       3
6  2339816       5151 ESEA MDL Season 33 North America    13 2.5714286       5
  deaths hs kddiff fkdiff rating        winner Outcome
1     10 12     11      4   1.95 Thunder Logic     Win
2     16  8     -7     -1   0.52 Thunder Logic    Loss
3      7  6     13      4   1.91 Thunder Logic     Win
4     18  2    -13     -4   0.31 Thunder Logic    Loss
5      9  7      9      2   1.62 Thunder Logic     Win
6      7  4      6      2   1.48 Thunder Logic     Win
```

Exploration of player statistics on game outome

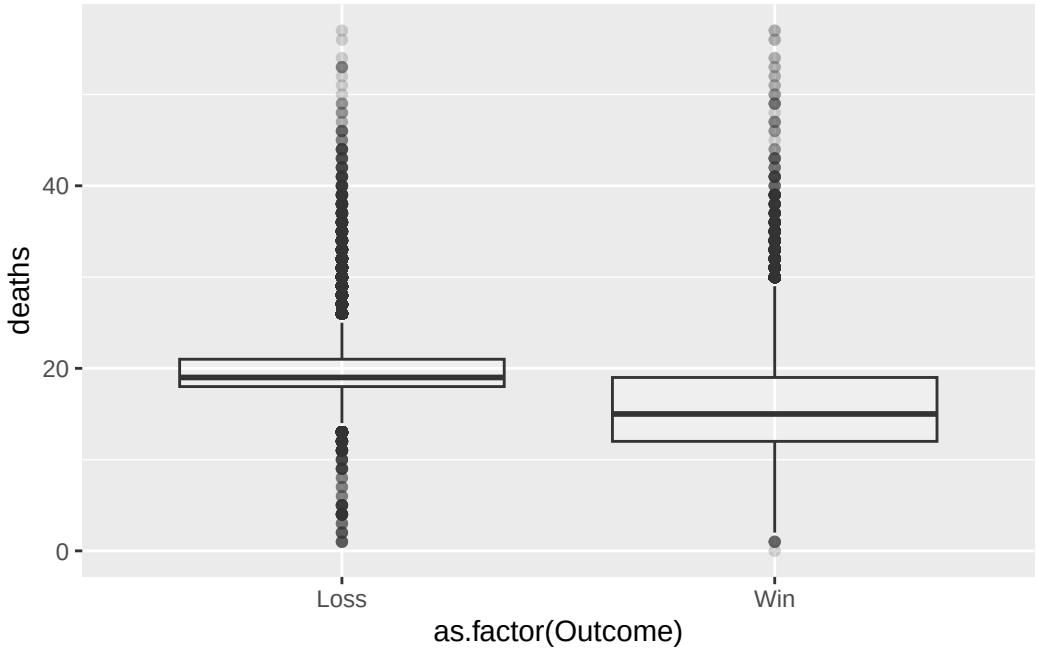
```
#kills
ggplot(csgo_single) +
  geom_boxplot(aes(y = kills, x=as.factor(Outcome)), alpha = 0.2)
```




```
#assists
ggplot(csgo_single) +
  geom_boxplot(aes(y = assists, x=as.factor(Outcome)), alpha = 0.2)
```




```
# deaths
ggplot(csgo_single) +
  geom_boxplot(aes(y = deaths, x=as.factor(Outcome)), alpha = 0.2)
```

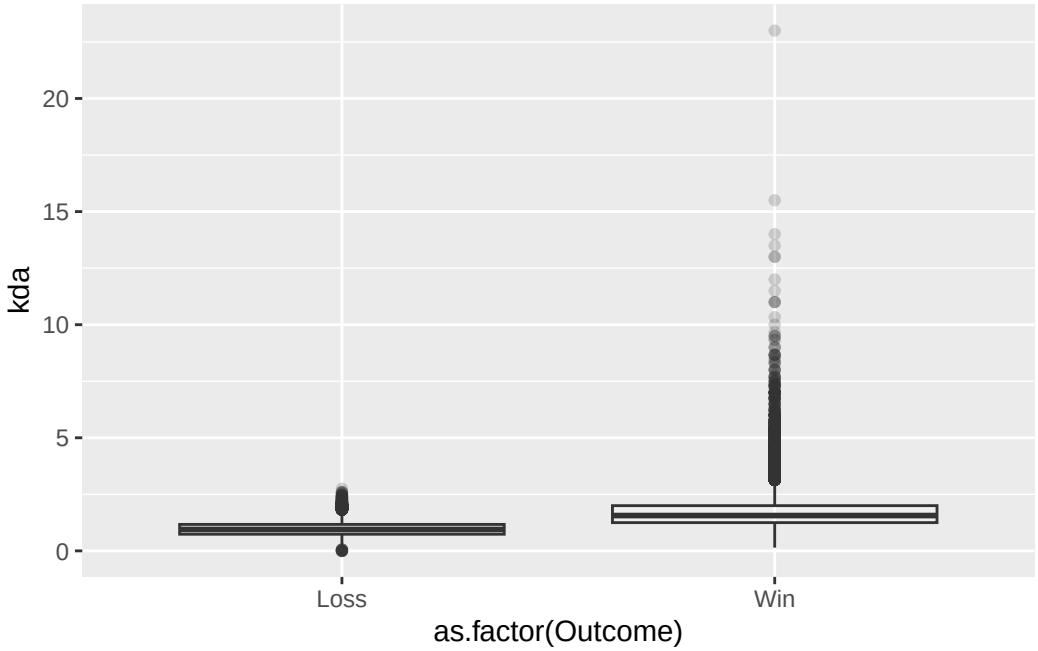



```
csgo_single %>%
  filter_all(any_vars(is.na(.)))
```

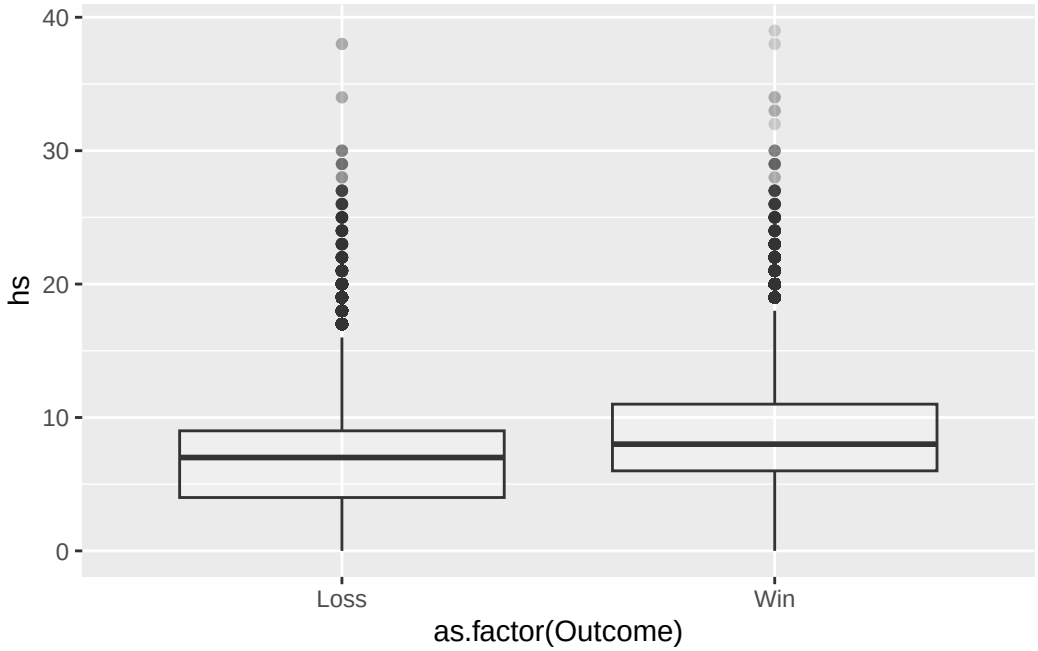
```
 [1] player_name date        team        opponent    country     player_id
 [7] match_id    event_id.x  event_name  kills       kda         assists
[13] deaths      hs          kddiff      fkdiff      rating      winner
[19] Outcome
<0 rows> (or 0-length row.names)
```

```
# kda
ggplot(csgo_single) +
  geom_boxplot(aes(y = kda, x=as.factor(Outcome)), alpha = 0.2)
```

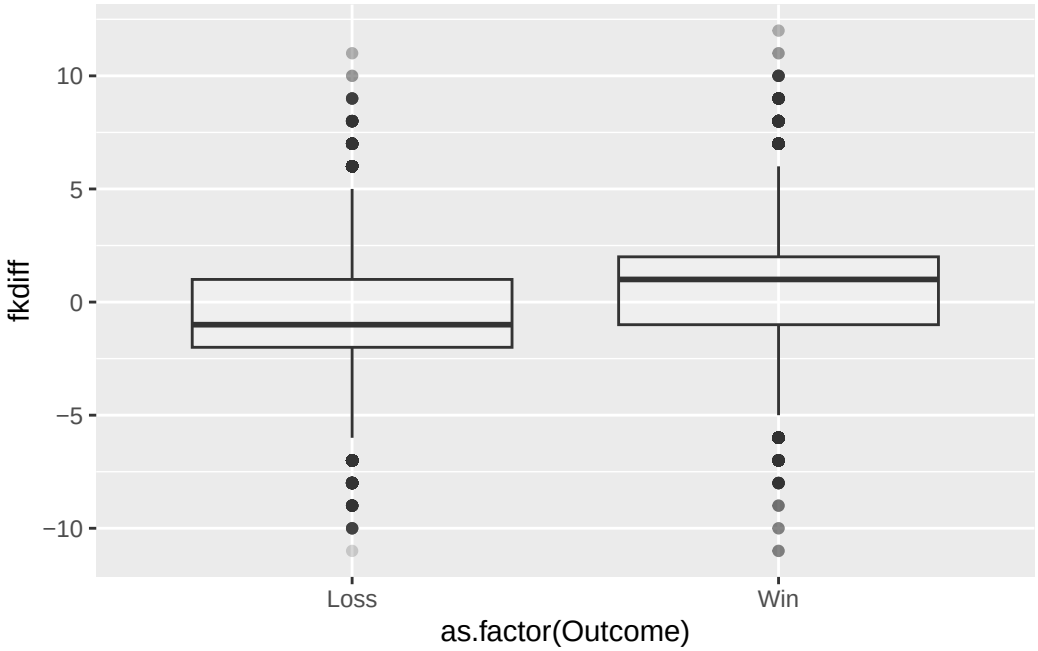
```
Warning: Removed 1 row containing non-finite outside the scale range
(`stat_boxplot()`).
```




```
# headshots
ggplot(csgo_single) +
  geom_boxplot(aes(y = hs, x=as.factor(Outcome)), alpha = 0.2)
```

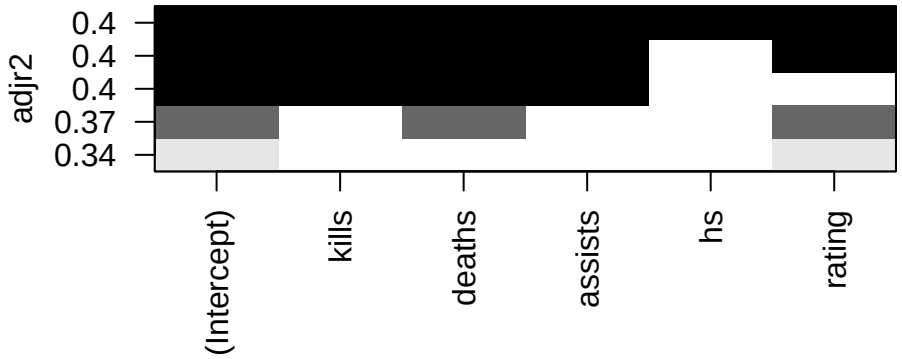



```
#fkdiff, firstkills-firstdeaths in the match
ggplot(csgo_single) +
  geom_boxplot(aes(y = fkdiff, x=as.factor(Outcome)), alpha = 0.2)
```

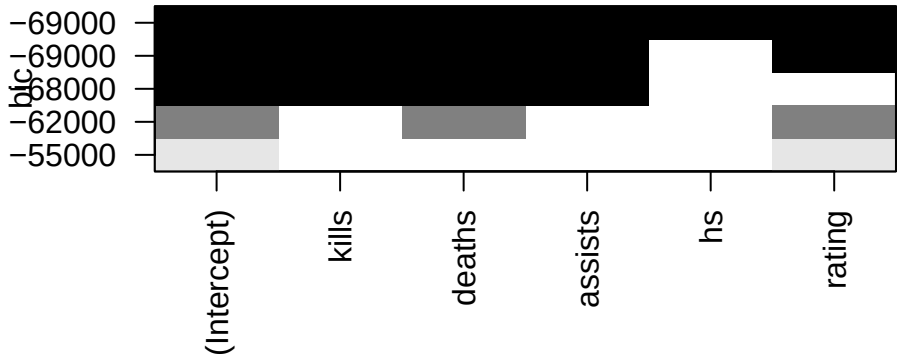


#best subset

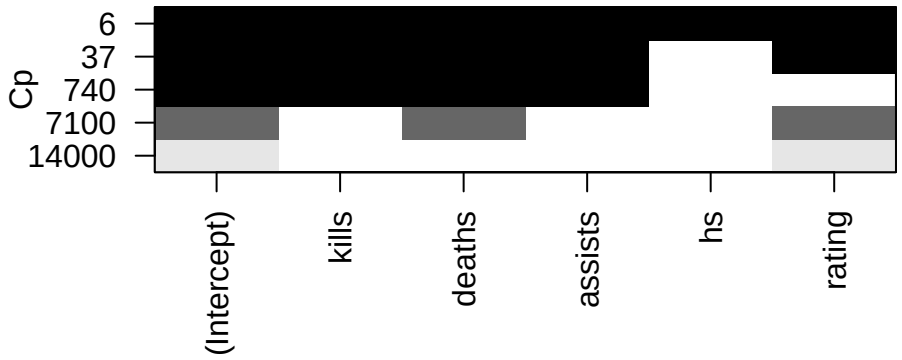
```
stats <- csgo_single %>% select(kills, deaths, assists,hs, Outcome,rating)
stats$Outcome <- ifelse(stats$Outcome == "Win", 1, 0)
pm.subsets <- regsubsets(data=stats, Outcome ~ .)
plot(pm.subsets, scale='adjr2')
```




```
plot(pm.subsets, scale='bic')
```



```
plot(pm.subsets, scale='Cp')
```



##Modelling for vars

```
mod1 <- glm(data = stats, formula = rating ~ kills + assists + deaths)
gtsummary::tbl_regression(mod1)
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	Beta	95% CI	p-value
kills	0.04	0.04, 0.04	<0.001
assists	0.02	0.02, 0.02	<0.001
deaths	-0.05	-0.05, -0.05	<0.001

```
mod2 <-glm(data = stats, formula = Outcome ~ kills + assists + deaths + hs + rating)
gtsummary::tbl_regression(mod1, exponentiate = TRUE)
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	exp(Beta)	95% CI	p-value
kills	1.04	1.04, 1.04	<0.001
assists	1.02	1.02, 1.02	<0.001
deaths	0.96	0.96, 0.96	<0.001

##Seperate by year, group by player, and take the mean of rating

```
#extract year into column
csgo_by_year <- csgo_single %>%
  mutate(year = as.integer(substr(date, 1, 4)))

#for csgo_single also add year
csgo_single <- csgo_single %>%
  mutate(year = as.integer(substr(date, 1, 4)))

#group into year and player name
csgo_by_year <- csgo_by_year %>%
  group_by(year, player_name)


#summarise mean rating
csgo_by_year <- csgo_by_year%>%
  summarise(mean_rating = mean(rating))
```

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

```
#display in descending mean rating order
csgo_by_year <- arrange(csgo_by_year, desc(mean_rating))
csgo_by_year
```

```
# A tibble: 7,773 x 3
# Groups:   year [6]
    year player_name mean_rating
```

```
   <int> <chr>             <dbl>
 1  2020 THRONE             2.25
 2  2020 fnx                2.08
 3  2017 imoRR              2.03
 4  2020 BORKUM             2.01
 5  2017 opportunity        1.98
 6  2018 JaKu               1.96
 7  2016 psycko             1.95
 8  2017 Art                1.93
 9  2020 witz               1.92
10  2020 RCF                1.86
# i 7,763 more rows
```

```
#sort into descending by year
csgo_by_year <- arrange(csgo_by_year, desc(year))
csgo_by_year
```

```
# A tibble: 7,773 x 3
# Groups:   year [6]
    year player_name mean_rating
   <int> <chr>             <dbl>
 1  2020 THRONE             2.25
 2  2020 fnx                2.08
 3  2020 BORKUM             2.01
 4  2020 witz               1.92
 5  2020 RCF                1.86
 6  2020 dumau              1.81
 7  2020 ritchiEE           1.77
 8  2020 VANITY             1.73
 9  2020 rallen             1.72
10  2020 flaw               1.68
# i 7,763 more rows
```

Find players one standard deviation above the mean mean_rating

```
overall_rating_mean <- mean(csgo_by_year$mean_rating, na.rm = TRUE)
overall_rating_sd <- sd(csgo_by_year$mean_rating, na.rm = TRUE)

#get all players who are over one standard deviation above avg
star_players <- csgo_by_year %>%
```

```
  filter(mean_rating > (overall_rating_mean + overall_rating_sd))

below_players <- csgo_by_year %>%
  filter(mean_rating < (overall_rating_mean - overall_rating_sd))

avg_players <- csgo_by_year %>%
  filter(mean_rating < (overall_rating_mean + overall_rating_sd) & mean_rating > (overall_

count_stars <- nrow(star_players)
count_avg <- nrow(avg_players)
count_below <- nrow(below_players)



#count of each type of player
count_stars
```

```
[1] 932
```

```
count_avg
```

```
[1] 5739
```

```
count_below
```

```
[1] 1102
```

```
print(paste("Star: ", count_stars))
```

```
[1] "Star:  932"
```

```
print(paste("Avg: ", count_avg))
```

```
[1] "Avg:  5739"
```

```r
print(paste("Below: ", count_below))
```

```
[1] "Below:  1102"
```

```r
head(star_players)
```

```
# A tibble: 6 x 3
# Groups:   year [1]
   year player_name mean_rating
  <int> <chr>             <dbl>
1  2020 THRONE             2.25
2  2020 fnx                2.08
3  2020 BORKUM             2.01
4  2020 witz               1.92
5  2020 RCF                1.86
6  2020 dumau              1.81
```

```r
head(avg_players)
```

```
# A tibble: 6 x 3
# Groups:   year [1]
   year player_name mean_rating
  <int> <chr>             <dbl>
1  2020 k0nfig             1.19
2  2020 ayken              1.19
3  2020 k1to               1.19
4  2020 El1an              1.19
5  2020 arT                1.19
6  2020 kaboose            1.19
```

```r
head(below_players)
```

```
# A tibble: 6 x 3
# Groups:   year [1]
   year player_name mean_rating
  <int> <chr>             <dbl>
1  2020 Dirty             0.75
```

```
2  2020 Jerry             0.75
3  2020 Slugy             0.75
4  2020 RZU               0.745
5  2020 OMG               0.74
6  2020 spamzzy1          0.74
```

```
head(csgo_single)
```

```
  player_name       date          team      opponent       country player_id
1    Andersin 2020-02-27 Thunder Logic      Station7 United States     14038
2    FrostayK 2020-02-27      Station7 Thunder Logic United States     12090
3   Inseaniac 2020-02-27 Thunder Logic      Station7        Canada     18623
4      JonahP 2020-02-27      Station7 Thunder Logic        Canada     11445
5       PureR 2020-02-27 Thunder Logic      Station7 United States     10622
6     Sharkie 2020-02-27 Thunder Logic      Station7 United States     19476
  match_id event_id.x                        event_name kills       kda assists
1  2339816       5151 ESEA MDL Season 33 North America    21 2.4000000       3
2  2339816       5151 ESEA MDL Season 33 North America     9 0.5625000       0
3  2339816       5151 ESEA MDL Season 33 North America    20 3.4285714       4
4  2339816       5151 ESEA MDL Season 33 North America     5 0.3888889       2
5  2339816       5151 ESEA MDL Season 33 North America    18 2.3333333       3
6  2339816       5151 ESEA MDL Season 33 North America    13 2.5714286       5
  deaths hs kddiff fkdiff rating        winner Outcome year
1     10 12     11      4   1.95 Thunder Logic     Win 2020
2     16  8     -7     -1   0.52 Thunder Logic    Loss 2020
3      7  6     13      4   1.91 Thunder Logic     Win 2020
4     18  2    -13     -4   0.31 Thunder Logic    Loss 2020
5      9  7      9      2   1.62 Thunder Logic     Win 2020
6      7  4      6      2   1.48 Thunder Logic     Win 2020
```

Create individual match data

```
#add the players to temp
t_star_players <- mutate(star_players, player_type = "star")
t_below_players <- mutate(below_players, player_type = "below_average")
t_avg_players <- mutate(avg_players, player_type = "average")

#combine into one tibble
all_players <- bind_rows(t_star_players, t_below_players, t_avg_players)
```

```r
#join with csgo data based on the player name and year
combined_data <- csgo_single %>%
  left_join(all_players, by = c("player_name", "year"))

#make new tibble that only has necessary variables, combining players into their teams tha
match_summary <- combined_data %>%
  group_by(match_id, team) %>%
  summarise(
    num_star = sum(player_type == "star"),
    num_average = sum(player_type == "average"),
    num_below_average = sum(player_type == "below_average"),
    outcome = first(Outcome)
  ) %>%
  filter(num_star + num_average+num_below_average == 5)
```

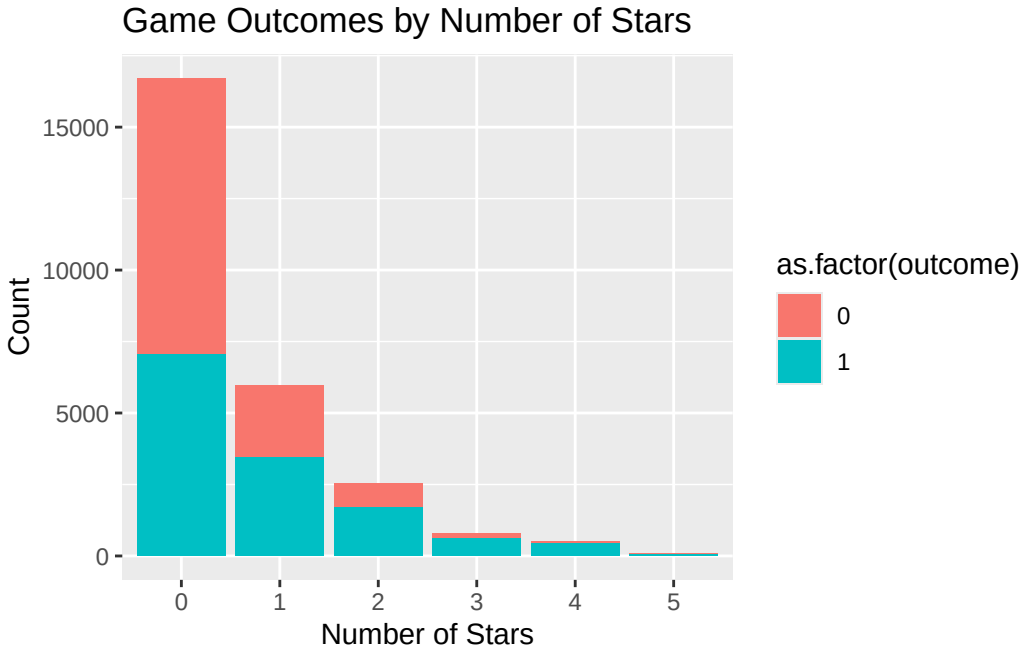
```
`summarise()` has grouped output by 'match_id'. You can override using the
`.groups` argument.
```

```r
head(match_summary)
```

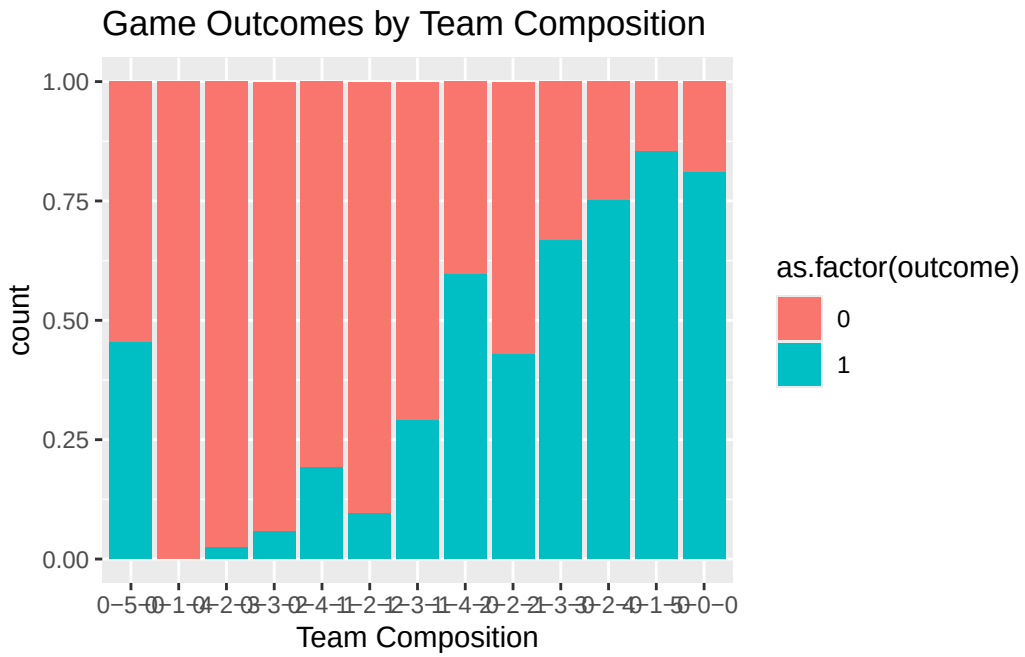
```
# A tibble: 6 x 6
# Groups:   match_id [3]
  match_id team     num_star num_average num_below_average outcome
     <int> <chr>       <int>       <int>             <int> <chr>
1  2299001 Dignitas        0           3                 2 Loss
2  2299001 NiP             1           4                 0 Win
3  2299003 Envy            0           5                 0 Loss
4  2299003 NiP             1           4                 0 Win
5  2299011 CLG             0           5                 0 Win
6  2299011 Liquid          0           5                 0 Loss
```

```r
match_summary <- match_summary %>%
  mutate(outcome = ifelse(outcome == "Win", 1, 0))

#make a column for outcome in binary form
match_summary <- match_summary %>%
  mutate(outcome = factor(outcome))

cols<-c('num_star', 'num_average', 'num_below_average')
```

```r
match_summary$composition <- do.call(paste, c(match_summary[cols], sep = "-"))
match_summary$composition <- relevel(factor(match_summary$composition), ref = "0-5-0")

tab <- table(match_summary$composition)
match_summary <- match_summary[match_summary$composition %in% names(tab)[tab>30],]

 ggplot(match_summary, aes(x = factor(num_star), fill = as.factor(outcome))) +
   geom_bar() +
   labs(x = "Number of Stars", y = "Count") +
   ggtitle("Game Outcomes by Number of Stars")
```




```r
ggplot(match_summary, aes(x = composition, fill = as.factor(outcome))) +
 geom_bar(position = "fill") +
 labs(x = "Team Composition") +
 ggtitle("Game Outcomes by Team Composition")
```



Final Models

```
match_summary
```

```
# A tibble: 26,612 x 7
# Groups:   match_id [13,334]
   match_id team      num_star num_average num_below_average outcome composition
      <int> <chr>        <int>       <int>             <int> <fct>   <fct>
 1  2299001 Dignitas         0           3                 2 0       0-3-2
 2  2299001 NiP              1           4                 0 1       1-4-0
 3  2299003 Envy             0           5                 0 0       0-5-0
 4  2299003 NiP              1           4                 0 1       1-4-0
 5  2299011 CLG              0           5                 0 1       0-5-0
 6  2299011 Liquid           0           5                 0 0       0-5-0
 7  2299040 CLG              0           5                 0 0       0-5-0
 8  2299040 Luminosi~        3           2                 0 1       3-2-0
 9  2299042 Liquid           0           5                 0 1       0-5-0
10  2299042 Luminosi~        3           2                 0 0       3-2-0
# i 26,602 more rows
```

```r
#split data into training and testing, 80 20
init_split <- initial_split(match_summary, prop = 0.8)
train <- training(init_split)
test  <- testing(init_split)

# #linear model
glm_star <- glm(data = train, formula = outcome ~ num_star, family =  "binomial")
glm_avg <- glm(data = train, formula = outcome ~ num_average, family =  "binomial")
glm_low <- glm(data = train, formula = outcome ~ num_below_average, family =  "binomial")

tbl_regression(glm_star, exponentiate = TRUE)
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	OR	95% CI	p-value
num_star	1.66	1.60, 1.71	<0.001

```r
tbl_regression(glm_avg, exponentiate = TRUE)
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	OR	95% CI	p-value
num_average	0.76	0.74, 0.78	<0.001

```r
tbl_regression(glm_low, exponentiate = TRUE)
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	OR	95% CI	p-value
num_below_average	0.25	0.22, 0.28	<0.001

```
 predicted_values <- predict(glm_star, newdata = test)


 glm_total_predictions <- ifelse(predicted_values > 0.5, 1, 0)

#display accuracy
 accuracy <- mean(glm_total_predictions == test$outcome)
 accuracy
```

```
[1] 0.5660342
```

```
glm_comp <- glm(data = train, formula = outcome ~ composition, family =  "binomial")
summary(glm_comp)
```

```
Call:
glm(formula = outcome ~ composition, family = "binomial", data = train)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -0.18801    0.01829 -10.281  < 2e-16 ***
composition0-1-4 -13.37806   85.73440  -0.156 0.876001
composition0-2-3  -3.31855    0.58623  -5.661 1.51e-08 ***
composition0-3-2  -2.55498    0.26701  -9.569  < 2e-16 ***
composition0-4-1  -1.24097    0.08684 -14.290  < 2e-16 ***
composition1-2-2  -1.85969    0.53159  -3.498 0.000468 ***
composition1-3-1  -0.81052    0.16811  -4.821 1.43e-06 ***
composition1-4-0   0.57604    0.03522  16.354  < 2e-16 ***
composition2-2-1  -0.33524    0.31602  -1.061 0.288775
composition2-3-0   0.87206    0.05103  17.090  < 2e-16 ***
composition3-2-0   1.29273    0.09221  14.020  < 2e-16 ***
composition4-1-0   1.95691    0.14203  13.778  < 2e-16 ***
composition5-0-0   1.74135    0.33238   5.239 1.61e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 29513  on 21288  degrees of freedom
Residual deviance: 27765  on 21276  degrees of freedom
AIC: 27791

Number of Fisher Scoring iterations: 12
```

```r
tbl_regression(glm_comp, exponentiate = TRUE)
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	**OR**	**95% CI**	**p-value**
composition			
0-5-0	—	—	
0-1-4	0.00	0.00, 0.00	0.9
0-2-3	0.04	0.01, 0.10	<0.001
0-3-2	0.08	0.04, 0.13	<0.001
0-4-1	0.29	0.24, 0.34	<0.001

Characteristic	OR	95% CI	p-value
1-2-2	0.16	0.05, 0.39	<0.001
1-3-1	0.44	0.32, 0.61	<0.001
1-4-0	1.78	1.66, 1.91	<0.001
2-2-1	0.72	0.38, 1.31	0.3
2-3-0	2.39	2.17, 2.64	<0.001
3-2-0	3.64	3.05, 4.38	<0.001
4-1-0	7.08	5.40, 9.44	<0.001
5-0-0	5.71	3.09, 11.5	<0.001

```r
phat.win <- predict(glm_comp, newdata = test ,type = 'response')


plot.mpp <- data.frame(pred.prob = phat.win,
                       pred = rbinom(n = length(phat.win),
                                     size = 1,
                                     p = phat.win),
                       truth = test$outcome)

plot.mpp <- plot.mpp %>%
  mutate(
    pred = factor(pred),
    truth = factor(truth))

print("Matrix:")
```

```
[1] "Matrix:"
```

```r
caret::confusionMatrix(plot.mpp$pred, plot.mpp$truth, positive="1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1447 1267
         1 1216 1393

               Accuracy : 0.5335
                 95% CI : (0.52, 0.547)
```

```
   No Information Rate : 0.5003
   P-Value [Acc > NIR] : 6.489e-07

                 Kappa : 0.0671

Mcnemar's Test P-Value : 0.3157

           Sensitivity : 0.5237
           Specificity : 0.5434
        Pos Pred Value : 0.5339
        Neg Pred Value : 0.5332
            Prevalence : 0.4997
        Detection Rate : 0.2617
  Detection Prevalence : 0.4901
     Balanced Accuracy : 0.5335

      'Positive' Class : 1
```