EDA_Basketball

```r
# Load in packages
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.0     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(leaps)
library(randomForest)
```

```
randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

```r
library(tidymodels)
```

```
-- Attaching packages -------------------------------------- tidymodels 1.1.1 --
v broom        1.0.5      v rsample      1.2.0
v dials        1.2.1      v tune         1.1.2
v infer        1.0.6      v workflows    1.1.4
v modeldata    1.3.0      v workflowsets 1.0.1
v parsnip      1.2.0      v yardstick    1.3.0
v recipes      1.0.10
-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x randomForest::combine() masks dplyr::combine()
x scales::discard()       masks purrr::discard()
x dplyr::filter()         masks stats::filter()
x recipes::fixed()        masks stringr::fixed()
x dplyr::lag()            masks stats::lag()
x randomForest::margin()  masks ggplot2::margin()
x yardstick::spec()       masks readr::spec()
x recipes::step()         masks stats::step()
* Learn how to get started at https://www.tidymodels.org/start/
```

```r
library(gtsummary)
```

```
Attaching package: 'gtsummary'

The following objects are masked from 'package:recipes':

    all_double, all_factor, all_integer, all_logical, all_numeric
```

Note: 'game_details' looks like the most promising data set.

```r
# Load in data (Jhet)
games <- read.csv("basketball_games.csv")
games_details <- read.csv("basketball_games_details.csv")
players <- read.csv("basketball_players.csv")
ranking <- read.csv("basketball_ranking.csv")
teams <- read.csv("basketball_teams.csv")
```

```r
# Var names converted to lower case
names(games_details) <- tolower(names(games_details))
names(games) <- tolower(names(games))

# Joining games with game details by game-id to get information necessary to make the outc
full_games_details <- inner_join(games_details, games, by = 'game_id', relationship = 'man

full_games_details <- full_games_details %>%
  select(-c(nickname, comment))

full_games_details <- full_games_details %>% separate(min, c("minutes", "seconds"), sep =
  mutate(min_played = as.numeric(minutes) + (as.numeric(seconds) / 60))
```

```
Warning: Expected 2 pieces. Missing pieces filled with `NA` in 133543 rows [14, 52, 63,
64, 65, 66, 75, 76, 77, 90, 112, 113, 124, 125, 126, 127, 138, 139, 140, 151,
...].
```

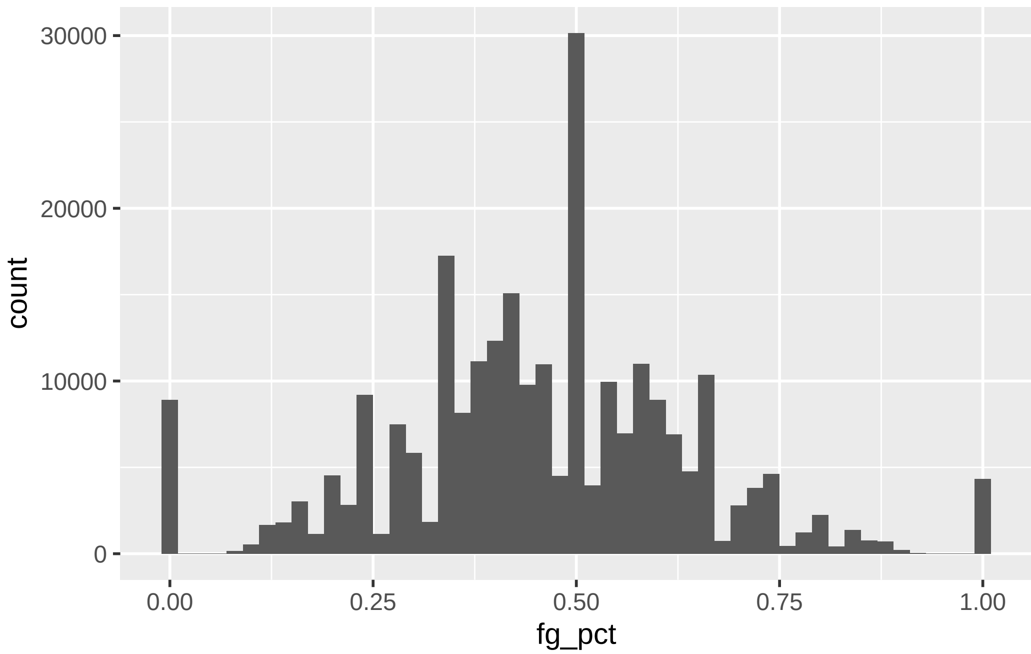
```r
# Data frame that only contains the players starting in the game (allows us to keep the po
starters <- full_games_details %>%
  filter(start_position != "")
```

```r
starters_sum <- starters %>%
  group_by(player_name) %>%
  arrange(player_name) %>%
  mutate(
    total_fgm = sum(fgm),
    total_fga = sum(fga),
    total_fg3m = sum(fg3m),
    total_fg3a = sum(fg3a),
    total_min_played = sum(min_played),
    overall_fg_pct = total_fgm / total_fga,
    total_pts = sum(pts),
    overall_pm = sum(plus_minus)
  )

starters_unique <- starters %>%
  distinct(player_name)

starters_final <- starters_sum %>%
  inner_join(starters_unique, by = "player_name") %>%
```

```
  slice(1)

starters <- starters %>%
  select(c(player_name, start_position, team_city, season, home_team_id, visitor_team_id,

## Creating a binary variable indicating if the player won or loss the game (1 indicates a
starters <- starters %>%
  mutate(player_game_outcome = ifelse((team_id == home_team_id & home_team_wins == 1) | (t
```

Univariate exploration (Jhet)

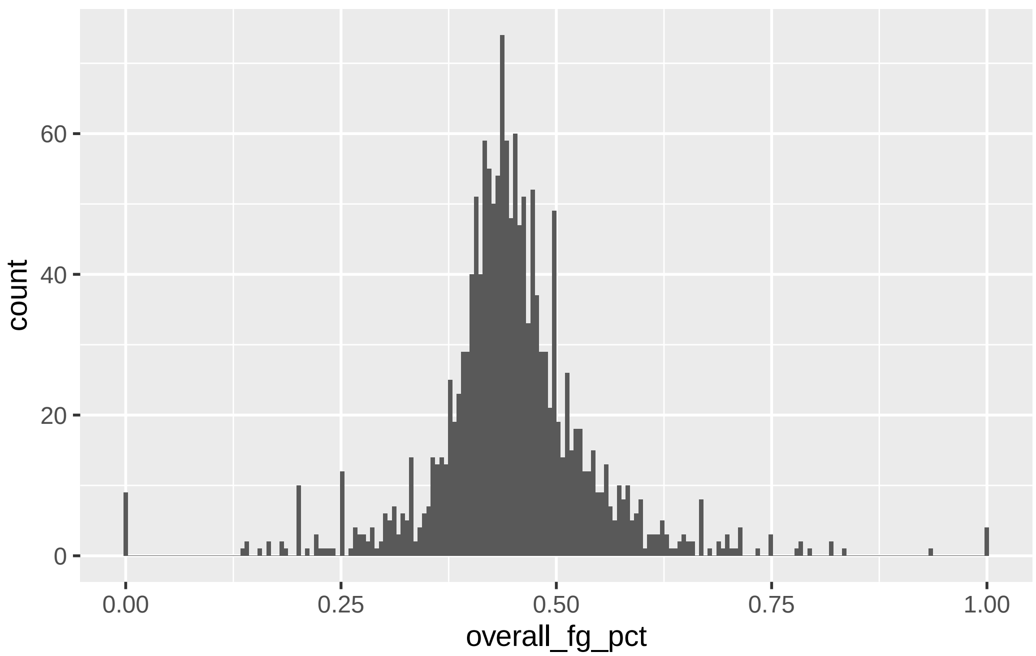
```
# Field goal percentage
ggplot(starters, aes(fg_pct)) + geom_histogram(binwidth = .02)
```



In the field goal percentage variable (% of field goal attempts that were successful), we see a vague shadow of a normal distribution. The distribution is far from smooth, but most of these weird spikes can be explained quite easily. Because each observation is a player's stats from a single game, there will be a lot of repeating values in cases where the player attempts very few shots.

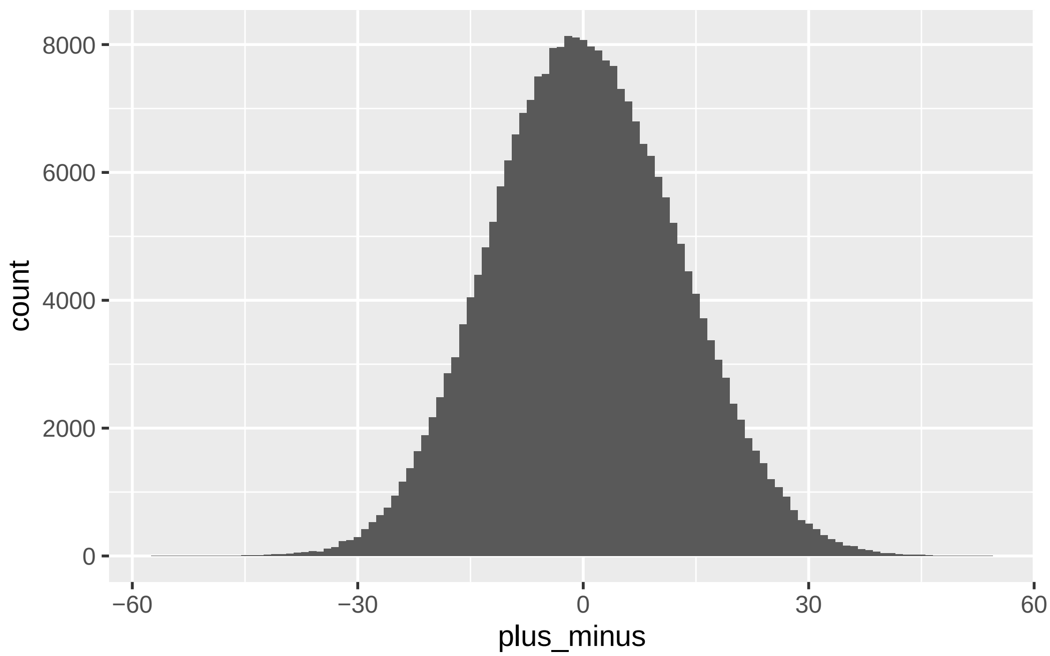
```r
# Field goal percentage (career)
ggplot(starters_final, aes(overall_fg_pct)) + geom_histogram(bins = 200)
```

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



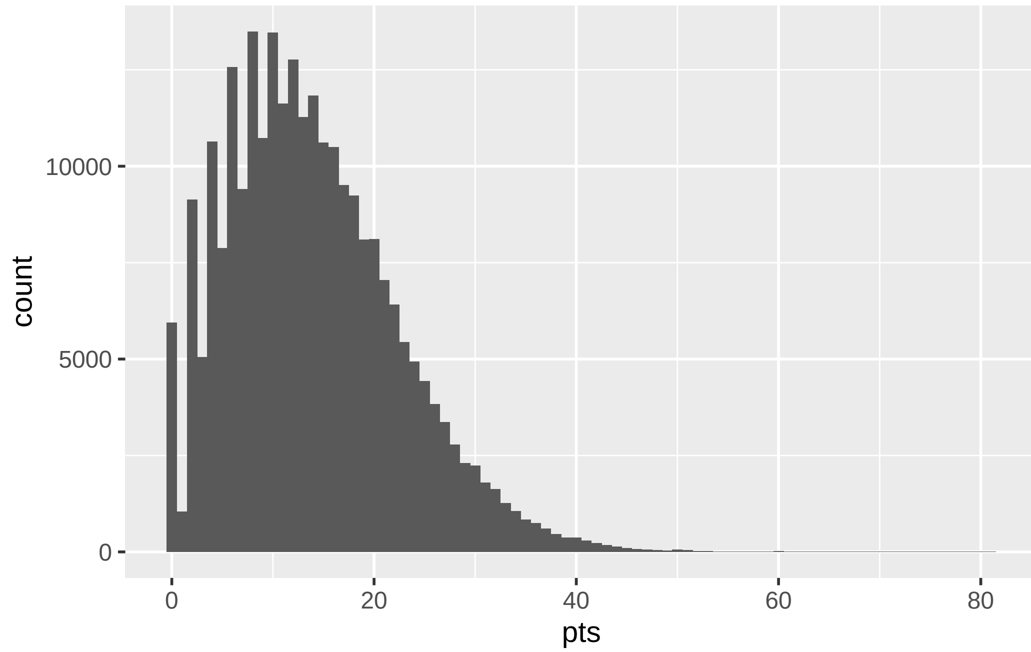
Thus, when we instead look at the field goal percentage over the player's entire career, the distribution becomes a lot less discrete, though it does still have spikes at common values among low shot volumes such as 0%, 100%, and 50%.

```r
# Plus minus variable -- Measure of player impact, looks at a team's point differential wh
ggplot(starters, aes(plus_minus)) + geom_histogram(binwidth = 1)
```



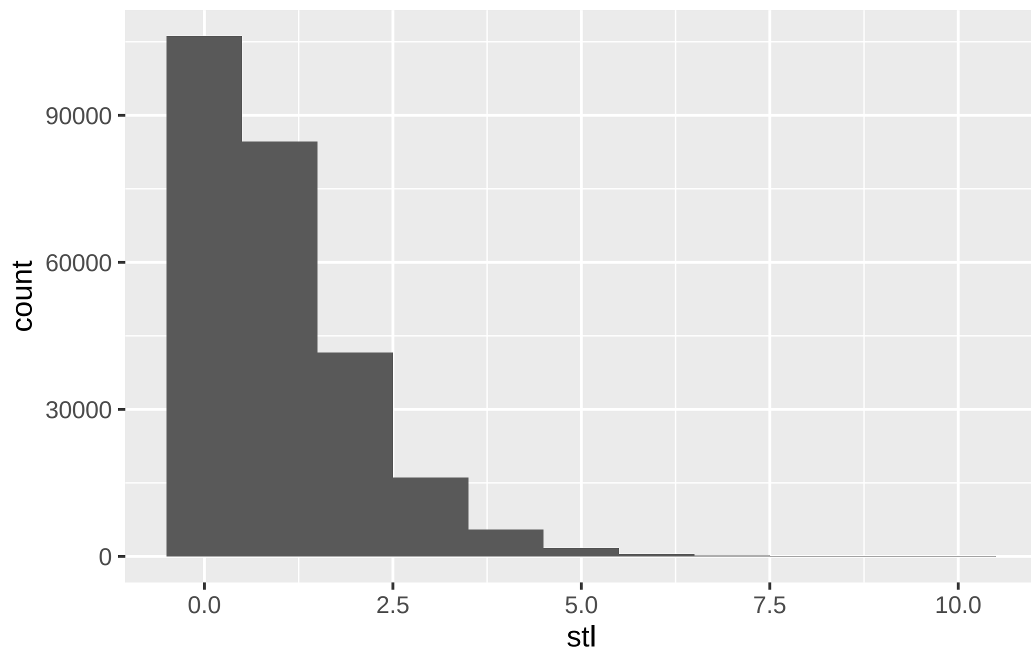
The “plus minus” stat is a measure of the team’s performance while a player is on the court. It is the difference between the amount of points gained by both teams while the player is active. This variable has a very nice normal distribution about 0, which makes sense as the plus minus of each team will be the inverse of the other at any given point. So even though the variable is based on the player’s active time, it is reasonable to expect a relatively (but not exactly) symmetrical distribution around 0.

```
# Points scored
ggplot(starters, aes(pts)) + geom_histogram(binwidth = 1)
```



The distribution of the points variable is also very reasonable. There is a spike at 0, as not every player's role is focused on scoring, and then a steep upward trend until around 10 points. After the 10 point mark the frequency starts dropping off quickly, and observations greater than ~45 are very rare. An interesting feature of this graph is that there seem to be regularly appearing spikes throughout the curve, showing much more in the lower point ranges. On closer examination, we can see that these spikes occur on multiples of 2. This is because scoring an even amount of points is much more likely than an odd amount, as the only way to score an odd amount is with a free throw or a 3 pointer.

```
# Steals
ggplot(starters, aes(stl)) + geom_histogram(binwidth = 1)
```

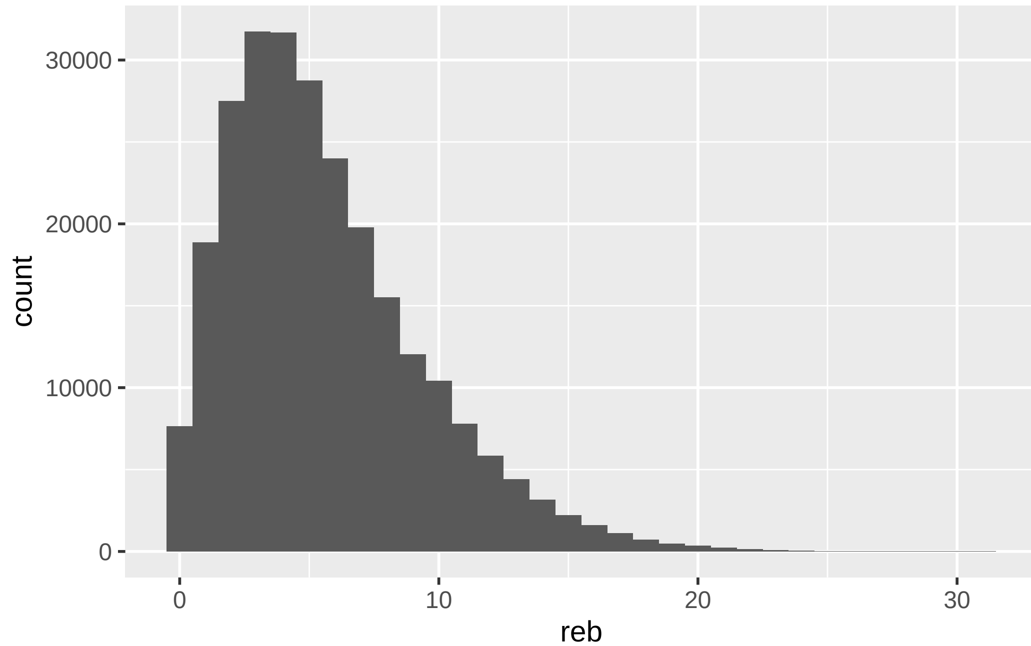



```r
summary(starters$stl)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  1.0000  0.9757  2.0000 10.0000 
```

The distribution of the steals variable is a very predictable downward curve. Steals are not very common, so most players get 0 during their time in a game. In fact, the maximum number of steals in a game in the data set is 10, with the average being just under 1.

```r
# Total rebounds
ggplot(starters, aes(reb)) + geom_histogram(binwidth = 1)
```



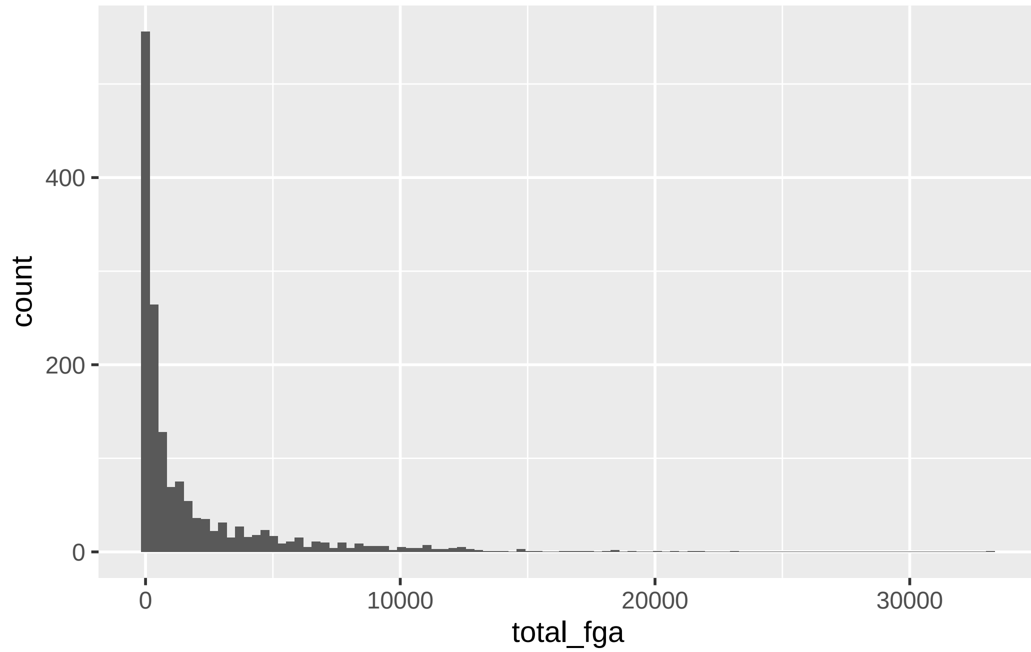
Finally, the rebounds variable is very similar to the points variable, with a dramatic right skew to the histogram. The data peaks at around 3-4 and begins dropping off quickly, with observations greater than 20 being very rare.

Trend exploration (Jhet)

Shot Volume, Percentage, and Points Scored

The first possible trend we wanted to investigate was how shot volume (# of shots attempted) influenced the shot percentage and number of points.

```
# Investigating how shot volume affects shot percentage and points scored
# Field goal percentage
ggplot(starters_final, aes(total_fga)) + geom_histogram(bins = 100)
```




```r
arrange(starters_final, desc(total_fga)) %>% head(5) %>% select(player_name, total_fga)
```

```
# A tibble: 5 x 2
# Groups:   player_name [5]
  player_name       total_fga
  <chr>                 <dbl>
1 LeBron James          33169
2 Carmelo Anthony       23034
3 Kevin Durant          21900
4 Russell Westbrook     21564
5 Kobe Bryant           20722
```

```r
summary(starters_final$total_fga)
```

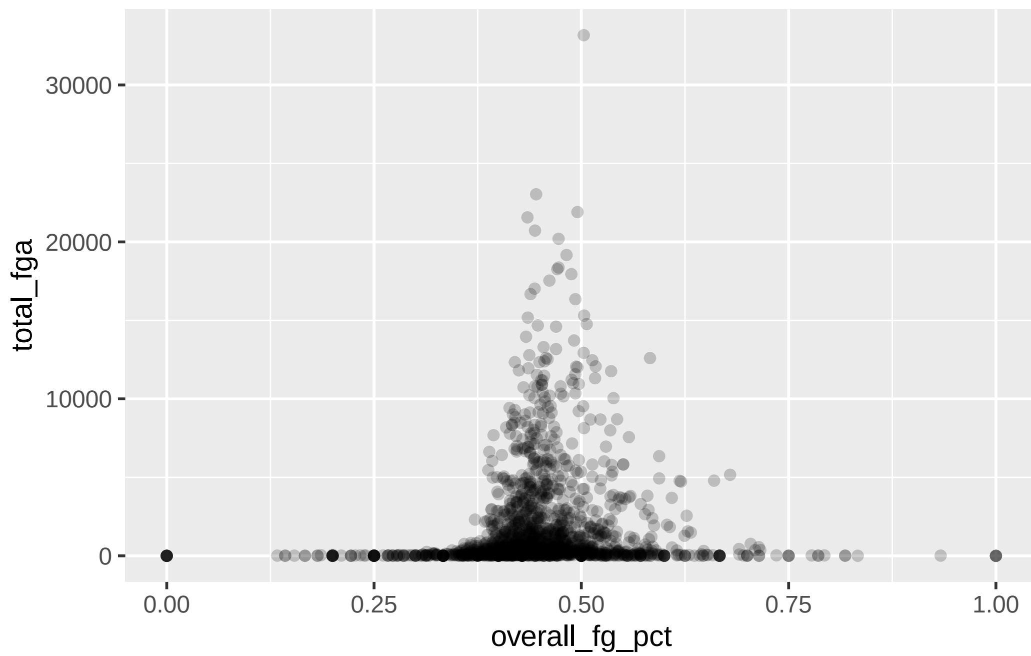
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    63.0   439.5  1845.3  2033.5 33169.0
```

First, the distribution of the total_fga (total field goals attempted over the player's career) variable is incredibly right skewed (blame LeBron). The overwhelming majority of players lie on the extreme left end of the distribution with an average value of 1845, and a median of 439.

This means that the average NBA player is over 30 thousand points behind LeBron James in this stat.

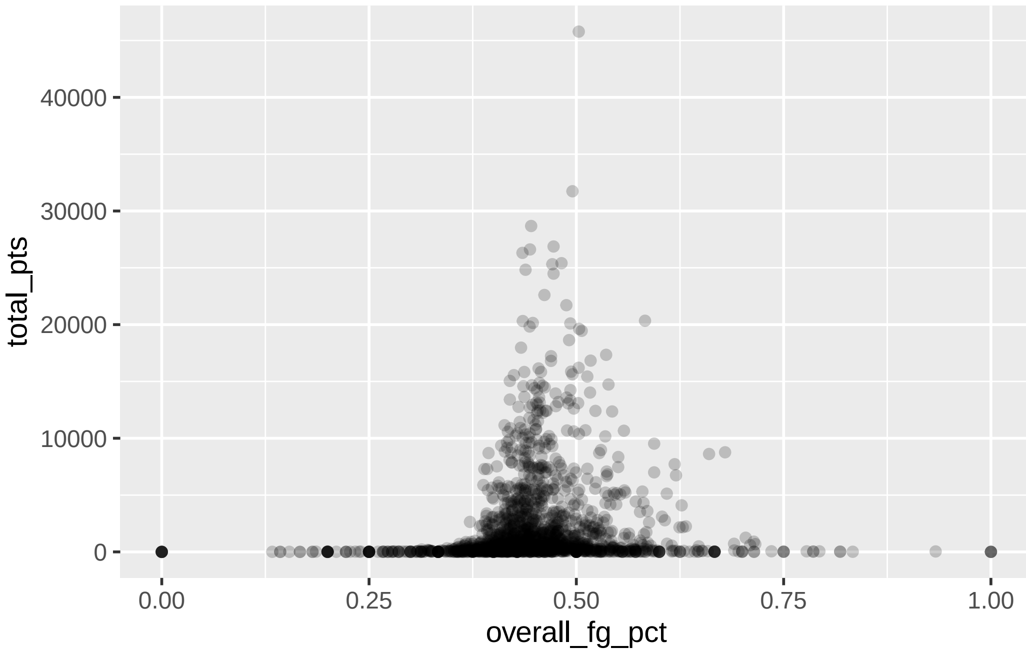
```
ggplot(starters_final) +
  geom_point(aes(x=overall_fg_pct, y=total_fga), alpha=0.2)
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```



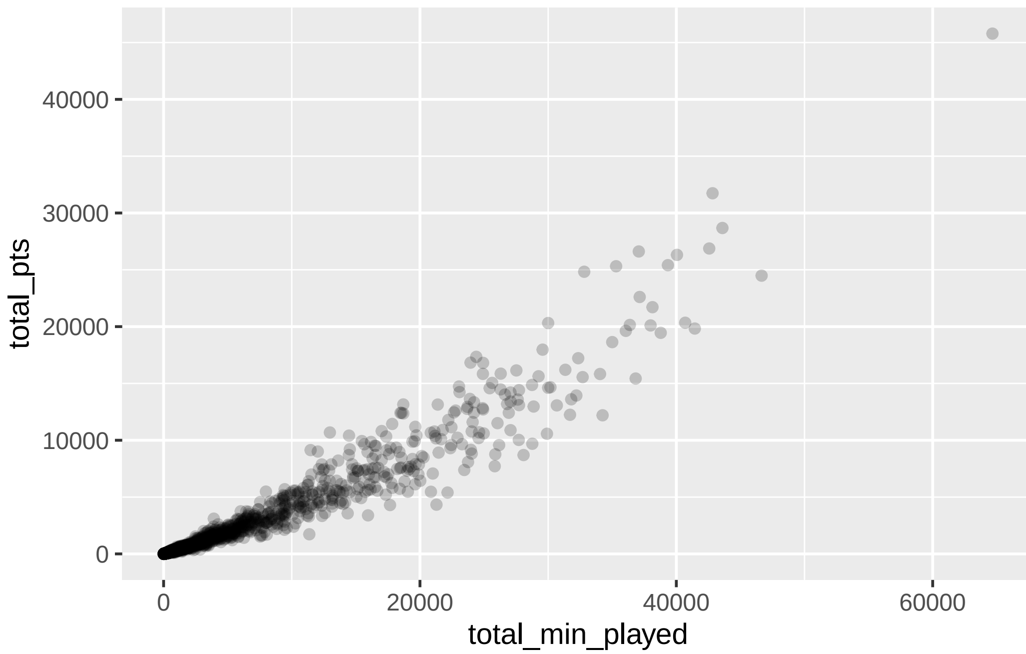

```
ggplot(starters_final) +
  geom_point(aes(x=overall_fg_pct, y=total_pts), alpha=0.2)
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```



The charts above compare the total_fga and total_pts (total points over a player's career) against overall_fg_pct (the player's field goal percentage over their career). Both distributions are very similar, with high concentration at 0% and 100%, and a very dense grouping around 40-50%. Overall, it seems that shot percentage has little relationship with shot volume or total points scored, with LeBron James leading both y variables by a large margin yet sitting very close to a 50% success rate. In fact, a higher shot volume in general trends towards a sub-50 shot percentage.

```
ggplot(starters_final) +
  geom_point(aes(x=total_min_played, y=total_pts), alpha=0.2)
```

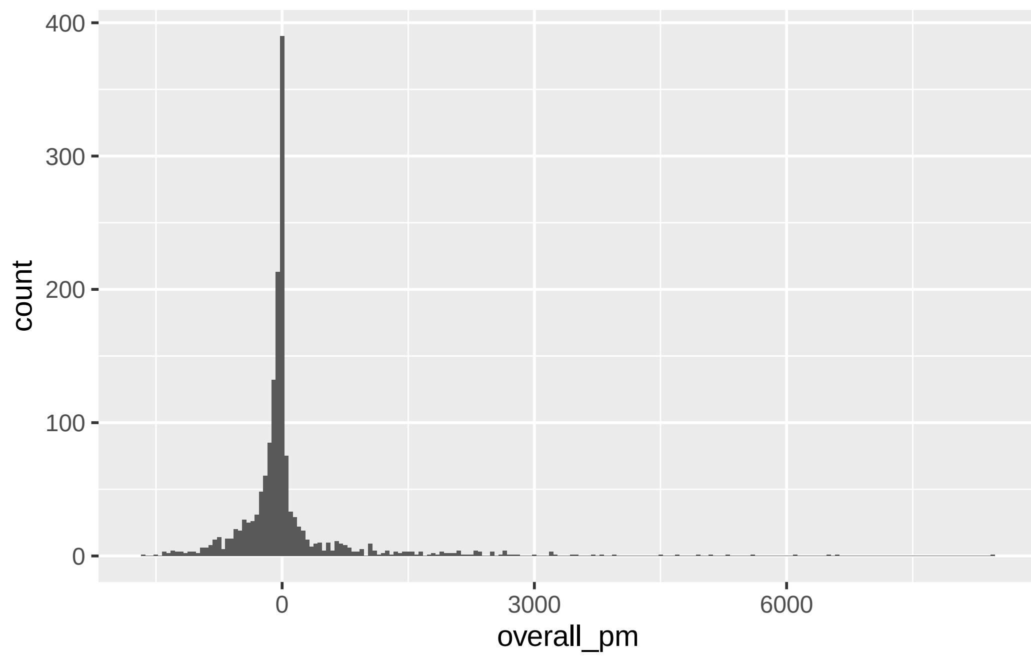


The similarity of the previous two charts is to be expected, as the total amount of points scored is likely to increase alongside the amount of shots attempted. This expectations is very predictably demonstrated in the above chart, with an incredibly linear relationship between the two being shown.

Plus Minus vs. the World

Next, the plus minus stat stands out as a potentially valuable measure of player performance, so we examined how it interacts with other stats.

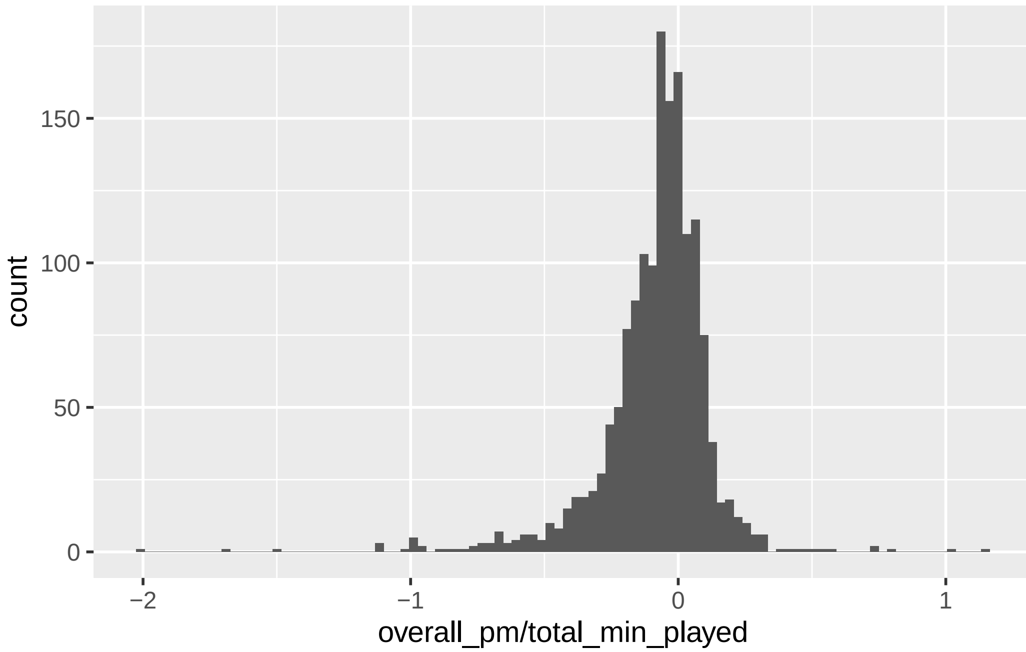
```
#Investigating how different stats affect plus minus
ggplot(starters_final, aes(overall_pm)) + geom_histogram(binwidth = 50)
```



Compared to the gentle and symmetrical distribution of the game-wise plus minus variable, the career-wise measure is very reminiscent of the total_fga variable (thanks LeBron). There is an incredibly large spike around 0, the majority of the data being centered around this spike.

In order to make this distribution more normal, we next looked at plus minus per minute (career-wise):

```
#Plus minus per min
ggplot(starters_final, aes(overall_pm/total_min_played)) + geom_histogram(bins = 100)
```

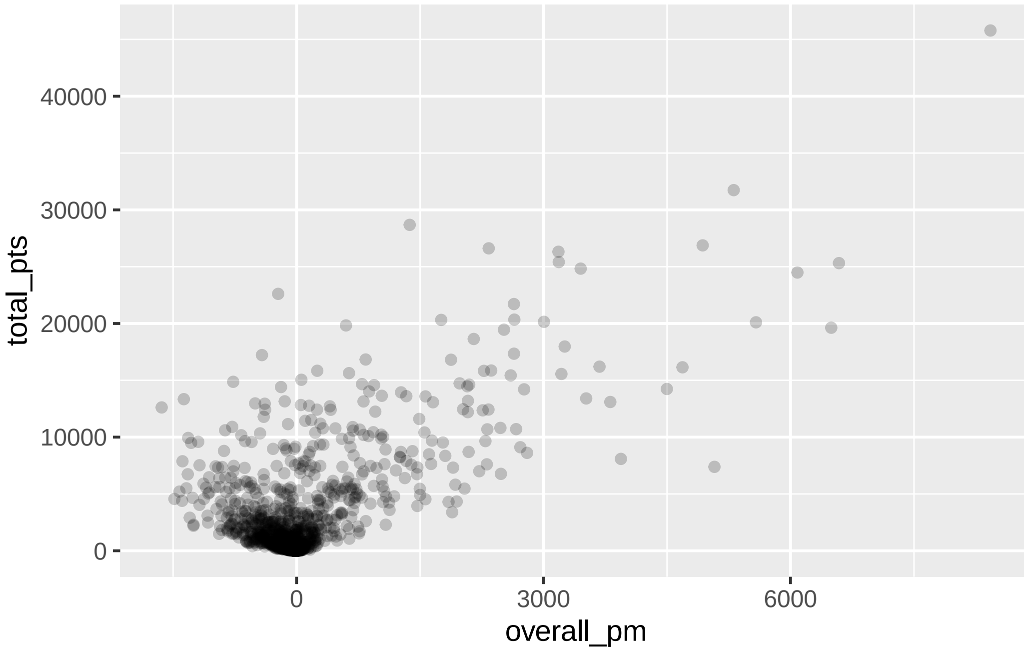



```
arrange(starters_final, desc(overall_pm/total_min_played)) %>%
  head(10) %>% select(player_name, overall_pm, total_min_played)
```

```
# A tibble: 10 x 3
# Groups:   player_name [10]
   player_name          overall_pm total_min_played
   <chr>                     <dbl>            <dbl>
 1 Marcus Derrickson            24             21.1
 2 Nicolas Claxton              21             20.6
 3 Dalen Terry                  17             21.4
 4 Brandon Boston Jr.           15             20.1
 5 Frank Williams               56             76.8
 6 Sasha Kaun                   23             39.9
 7 Pierre Jackson                7             13.0
 8 James Michael McAdoo         35             67.0
 9 Juwan Morgan                 19             39.6
10 Kennedy Chandler             13             29.4
```

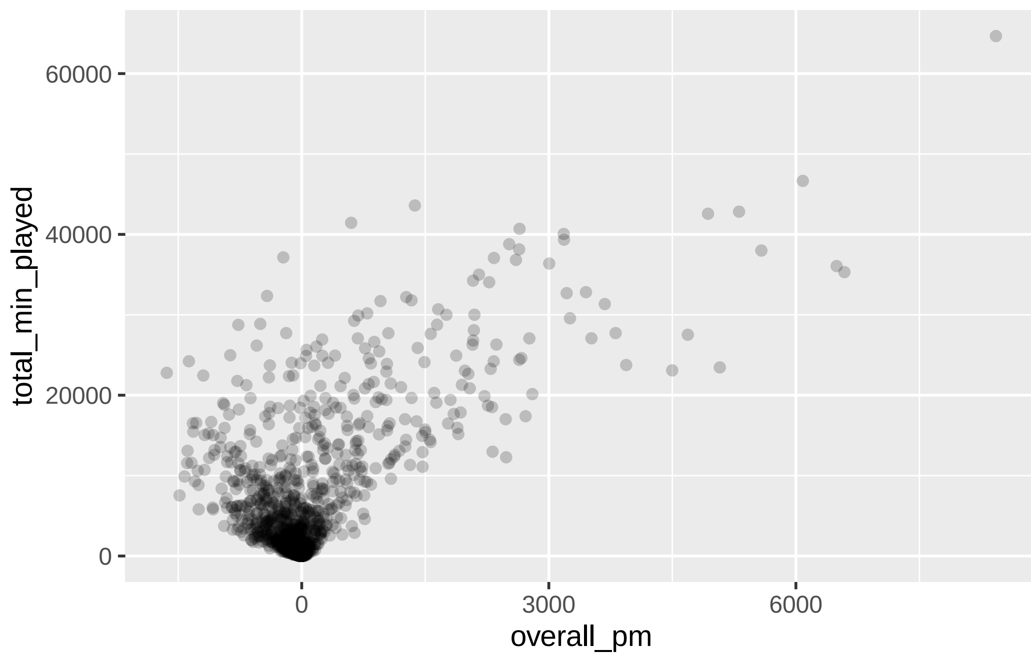
While the distribution is slightly more reasonable, the top players by this measure are names we had never heard of, all of whom had relatively low total minutes values. Thus, plus minus per minute is a very flawed performance measure.

```
ggplot(starters_final) +
  geom_point(aes(x=overall_pm, y=total_pts), alpha=0.2)
```



Overall plus minus and total points have an interestingly almost parabolic distribution. Most of the observations are crowded around 0 and under 5000 total points, but in general the outliers in the total points stat have incredibly high overall plus minuses.

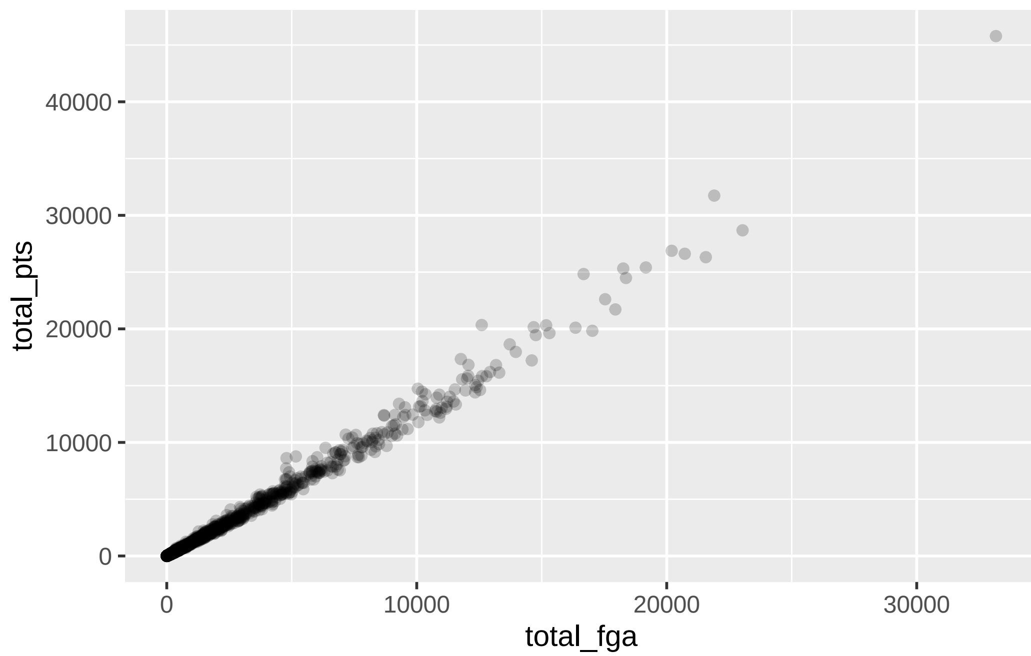
```
ggplot(starters_final) +
  geom_point(aes(x=overall_pm, y=total_min_played), alpha=0.2)
```



In plotting total minutes against total plus minus, we get a very similar distribution, with crowding around 0 in the lower minute ranges, with outliers having large plus minuses.

The similarities between these two graphs can again be explained by a linear relationship between the y variables:

```
ggplot(starters_final) +
  geom_point(aes(x=total_fga, y=total_pts), alpha=0.2)
```



Just like total points and shot volume, the total points scored by a player has a very strong linear relationship with their total minutes played, which makes intuitive sense.

Investigation of starters data set (Saul)

```
# Data looks good
summary(starters)
```

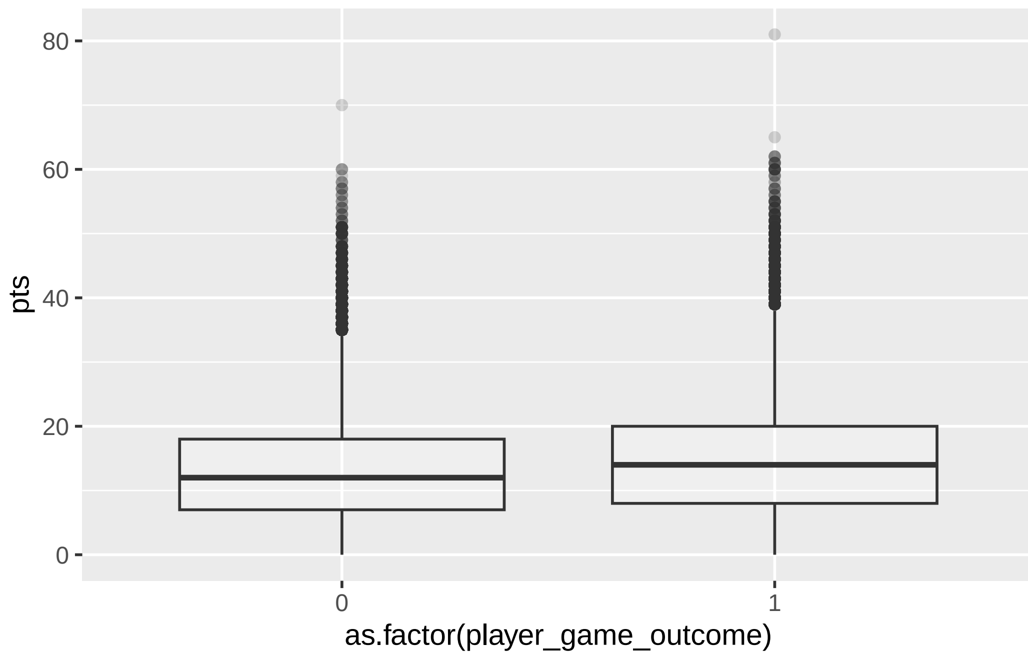
```
 player_name        start_position       team_city              season    
 Length:256104      Length:256104      Length:256104      Min.   :2003  
 Class :character   Class :character   Class :character   1st Qu.:2007  
 Mode  :character   Mode  :character   Mode  :character   Median :2012  
                                                          Mean   :2012  
                                                          3rd Qu.:2017  
                                                          Max.   :2022  
  home_team_id       visitor_team_id        team_id            game_id        
 Min.   :1.611e+09   Min.   :1.611e+09   Min.   :1.611e+09   Min.   :11400001  
 1st Qu.:1.611e+09   1st Qu.:1.611e+09   1st Qu.:1.611e+09   1st Qu.:20700808  
 Median :1.611e+09   Median :1.611e+09   Median :1.611e+09   Median :21300071  
 Mean   :1.611e+09   Mean   :1.611e+09   Mean   :1.611e+09   Mean   :22207301  
```

```
 3rd Qu.:1.611e+09   3rd Qu.:1.611e+09   3rd Qu.:1.611e+09   3rd Qu.:21800324
 Max.   :1.611e+09   Max.   :1.611e+09   Max.   :1.611e+09   Max.   :52100211
 home_team_wins      pts_home        pts_away       min_played
 Min.   :0.0000   Min.   : 59.0   Min.   : 54.0   Min.   : 0.06667
 1st Qu.:0.0000   1st Qu.: 95.0   1st Qu.: 92.0   1st Qu.:25.85000
 Median :1.0000   Median :103.0   Median :100.0   Median :31.86667
 Mean   :0.5892   Mean   :103.7   Mean   :100.9   Mean   :30.99513
 3rd Qu.:1.0000   3rd Qu.:112.0   3rd Qu.:110.0   3rd Qu.:36.73333
 Max.   :1.0000   Max.   :168.0   Max.   :168.0   Max.   :64.96667
      fgm              fga             fg_pct            fg3m
 Min.   : 0.000   Min.   : 0.00   Min.   :0.0000   Min.   : 0.000
 1st Qu.: 3.000   1st Qu.: 7.00   1st Qu.:0.3330   1st Qu.: 0.000
 Median : 5.000   Median :11.00   Median :0.4550   Median : 0.000
 Mean   : 5.184   Mean   :11.21   Mean   :0.4569   Mean   : 1.056
 3rd Qu.: 7.000   3rd Qu.:15.00   3rd Qu.:0.5710   3rd Qu.: 2.000
 Max.   :28.000   Max.   :50.00   Max.   :1.0000   Max.   :14.000
      fg3a           fg3_pct            ftm              fta
 Min.   : 0.000   Min.   :0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 0.000   1st Qu.:0.000   1st Qu.: 0.000   1st Qu.: 0.000
 Median : 2.000   Median :0.000   Median : 2.000   Median : 2.000
 Mean   : 2.918   Mean   :0.234   Mean   : 2.542   Mean   : 3.303
 3rd Qu.: 5.000   3rd Qu.:0.429   3rd Qu.: 4.000   3rd Qu.: 5.000
 Max.   :24.000   Max.   :1.000   Max.   :26.000   Max.   :39.000
     ft_pct            oreb             dreb             reb
 Min.   :0.0000   Min.   : 0.000   Min.   : 0.00   Min.   : 0.000
 1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.: 2.00   1st Qu.: 3.000
 Median :0.6670   Median : 1.000   Median : 4.00   Median : 5.000
 Mean   :0.5572   Mean   : 1.373   Mean   : 4.23   Mean   : 5.602
 3rd Qu.:1.0000   3rd Qu.: 2.000   3rd Qu.: 6.00   3rd Qu.: 8.000
 Max.   :1.0000   Max.   :18.000   Max.   :25.00   Max.   :31.000
      ast              stl               blk                to
 Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
 1st Qu.: 1.000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 1.000
 Median : 2.000   Median : 1.0000   Median : 0.0000   Median : 2.000
 Mean   : 3.084   Mean   : 0.9757   Mean   : 0.6336   Mean   : 1.836
 3rd Qu.: 4.000   3rd Qu.: 2.0000   3rd Qu.: 1.0000   3rd Qu.: 3.000
 Max.   :25.000   Max.   :10.0000   Max.   :12.0000   Max.   :12.000
       pf             pts          plus_minus       player_game_outcome
 Min.   :0.000   Min.   : 0.00   Min.   :-57.0000   Min.   :0.0000
 1st Qu.:1.000   1st Qu.: 8.00   1st Qu.: -8.0000   1st Qu.:0.0000
 Median :2.000   Median :13.00   Median :  0.0000   Median :0.0000
 Mean   :2.446   Mean   :13.97   Mean   :  0.3626   Mean   :0.4999
 3rd Qu.:3.000   3rd Qu.:19.00   3rd Qu.:  9.0000   3rd Qu.:1.0000
```

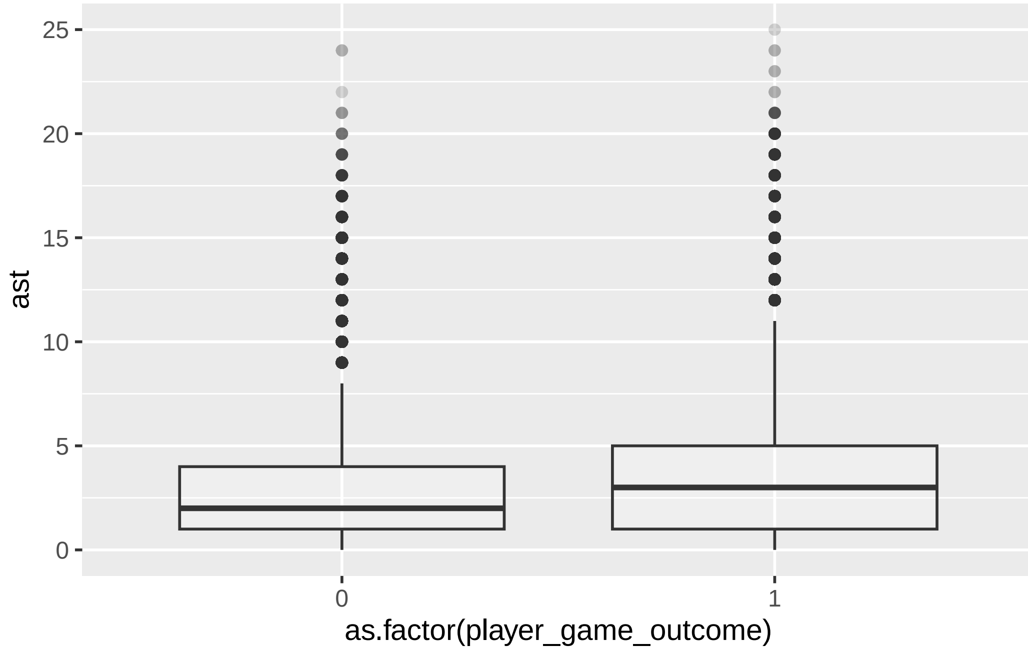
```
Max.   :6.000   Max.   :81.00   Max.   : 54.0000   Max.   :1.0000
```

An exploration of the big 5 and their relationship to game outcomes (Points, Assists, Steals, Rebounds, and Turnovers)

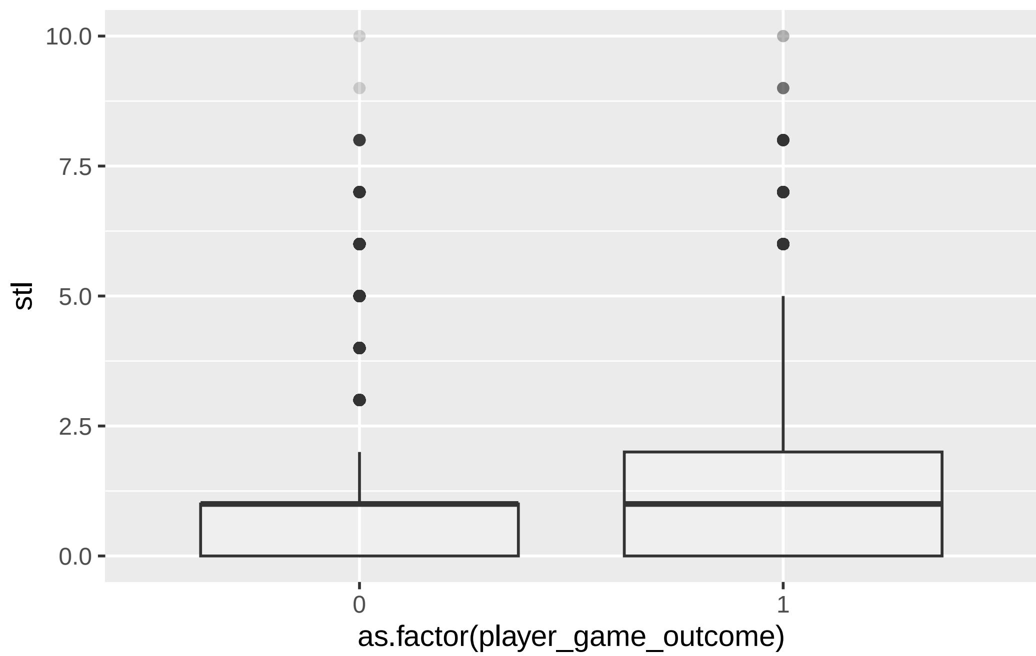
```
#Points
ggplot(starters) +
  geom_boxplot(aes(y=pts, x=as.factor(player_game_outcome)), alpha=0.2)
```



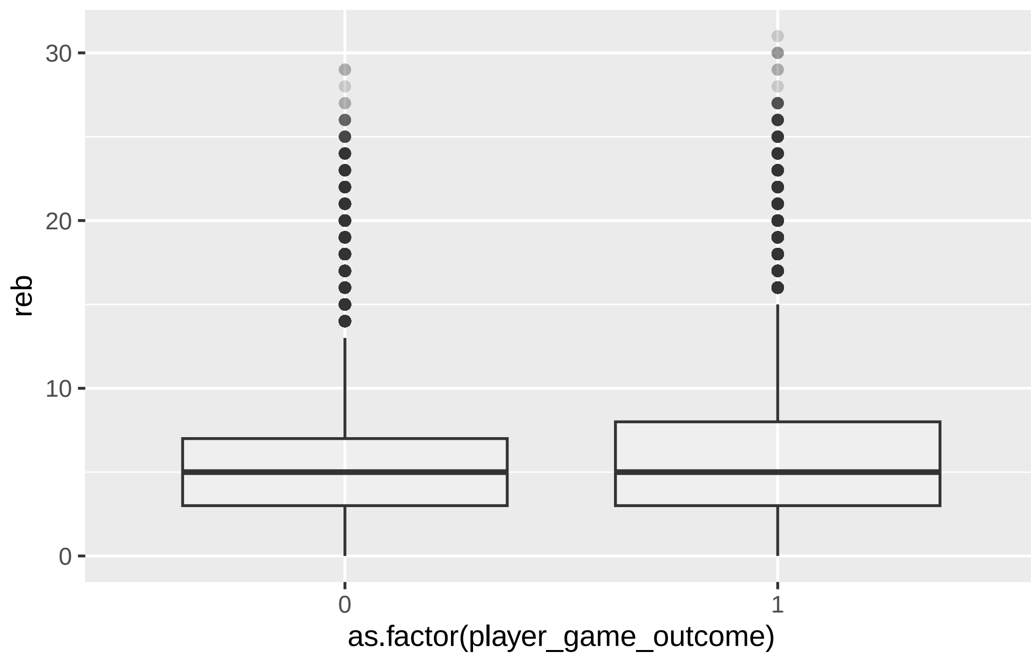

```
# Assists
ggplot(starters) +
  geom_boxplot(aes(y=ast, x=as.factor(player_game_outcome)), alpha=0.2)
```



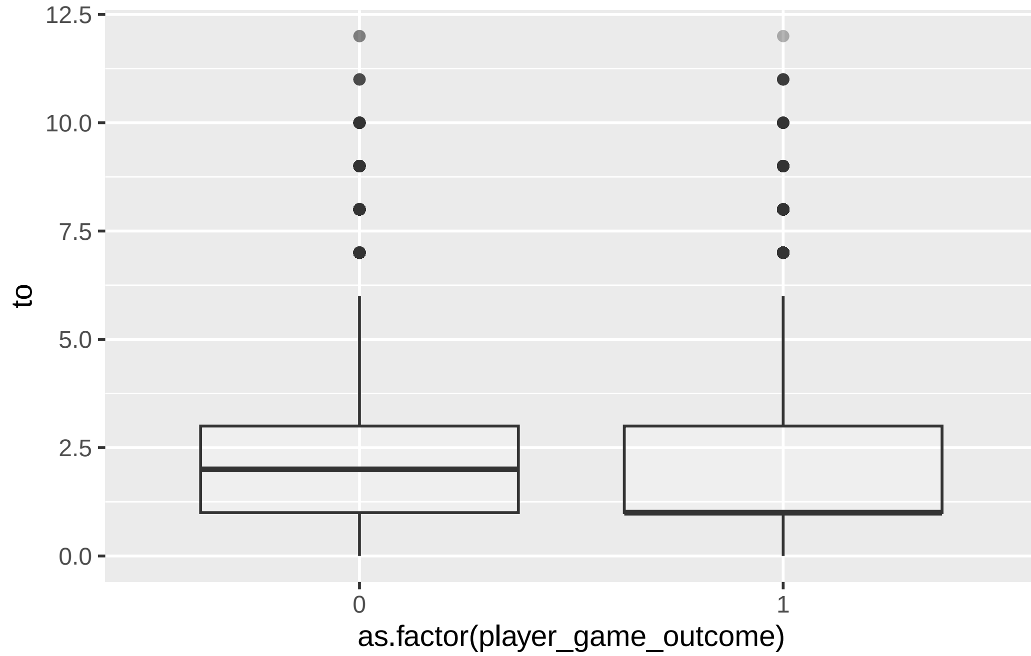

```
# Steals
ggplot(starters) +
  geom_boxplot(aes(y=stl, x=as.factor(player_game_outcome)), alpha=0.2)
```




```
# Rebounds
ggplot(starters) +
  geom_boxplot(aes(y=reb, x=as.factor(player_game_outcome)), alpha=0.2)
```



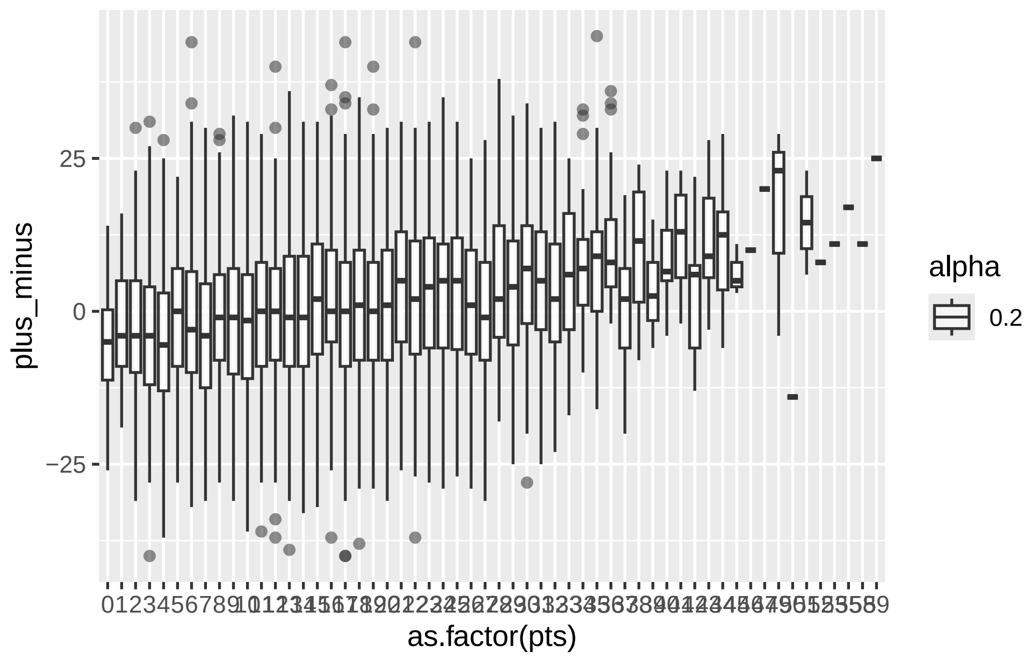

```
# Turnovers
ggplot(starters) +
  geom_boxplot(aes(y=to, x=as.factor(player_game_outcome)), alpha=0.2)
```



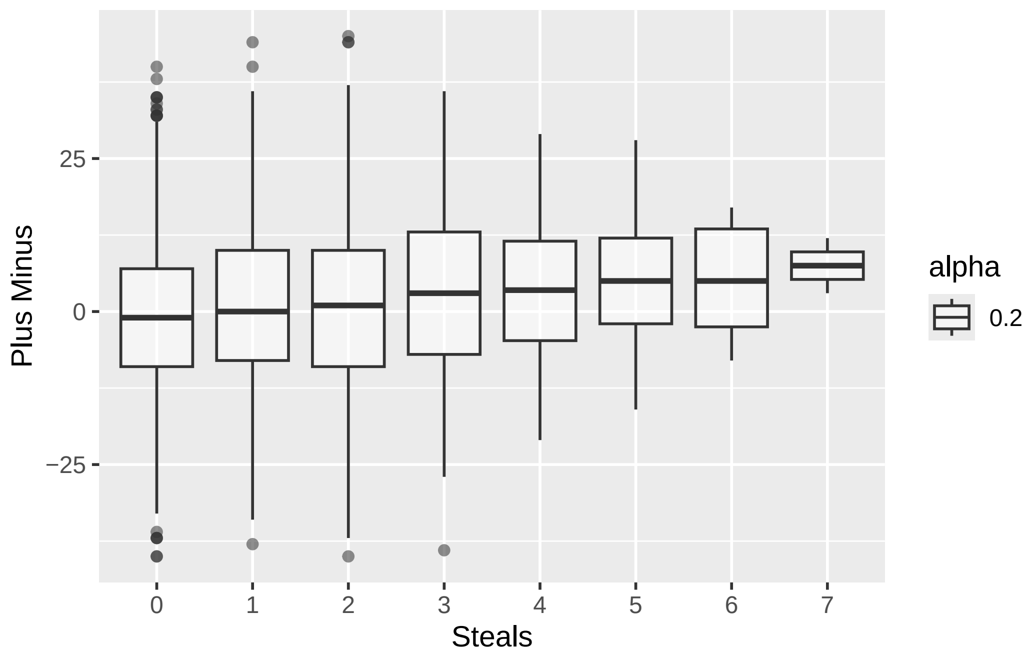
The big 5 and the plus minus

Filtering by season to deal with long wait times observations

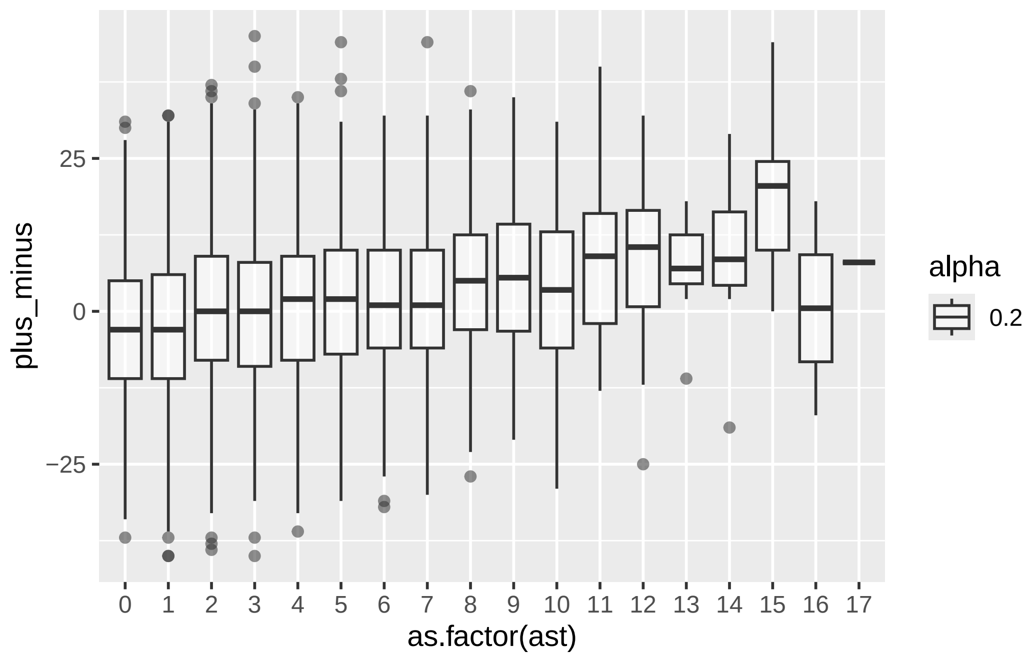
```
# Points
starters2 <- filter(starters, season == 2022)
ggplot(starters2, aes(x=as.factor(pts), y=plus_minus, alpha=0.2)) +
  geom_boxplot()
```



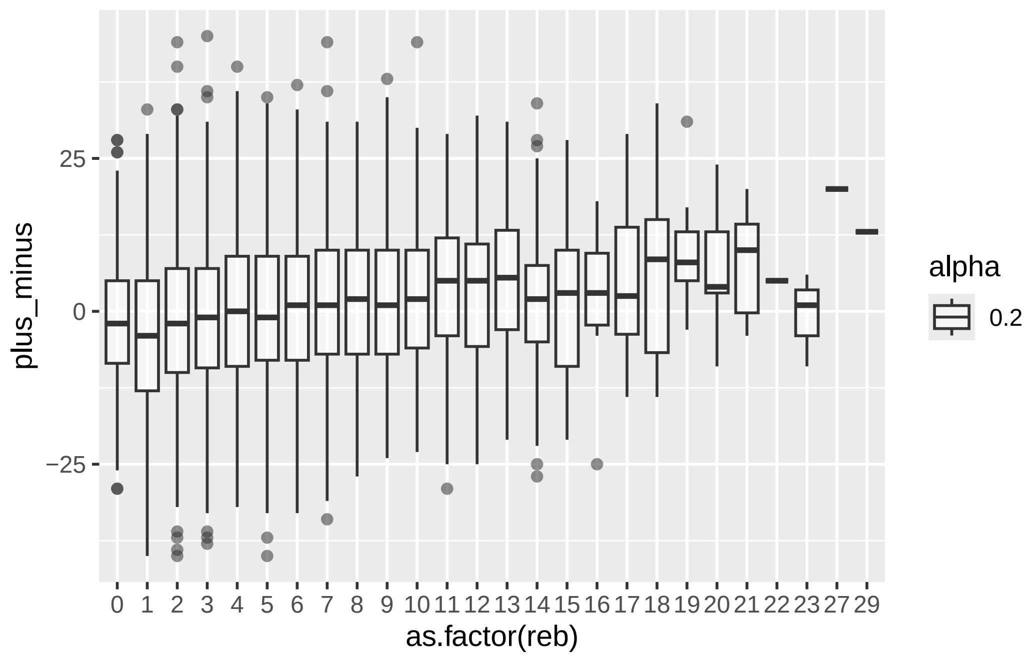

```
# Steals
ggplot(starters2, aes(x=as.factor(stl), y=plus_minus, alpha=0.2)) +
  geom_boxplot() + xlab('Steals') + ylab('Plus Minus')
```



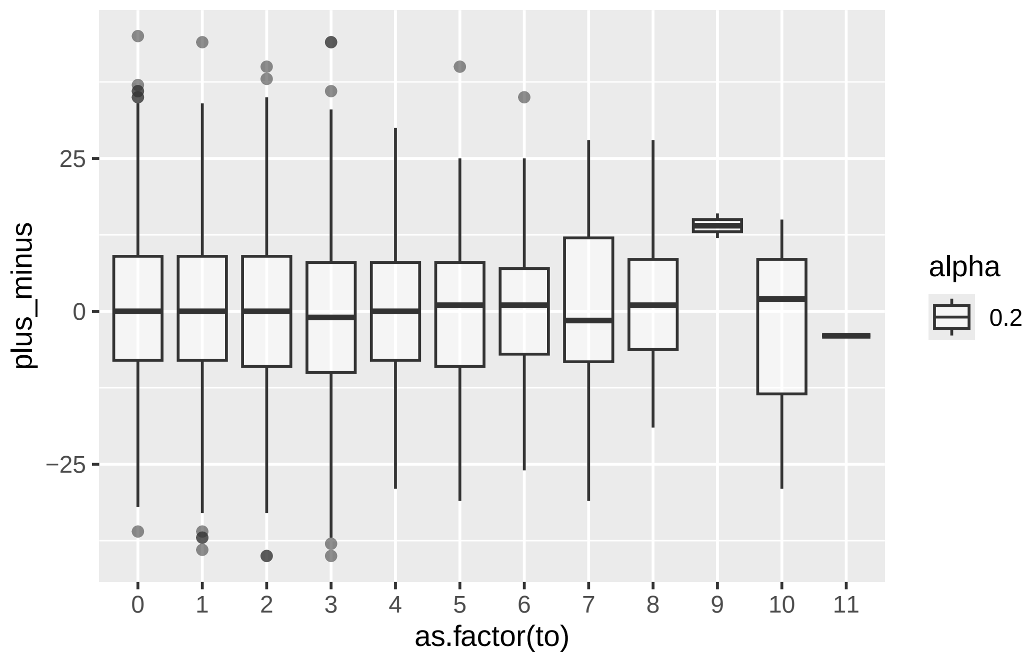

```
# Assists
ggplot(starters2, aes(x=as.factor(ast), y=plus_minus, alpha=0.2)) +
  geom_boxplot()
```



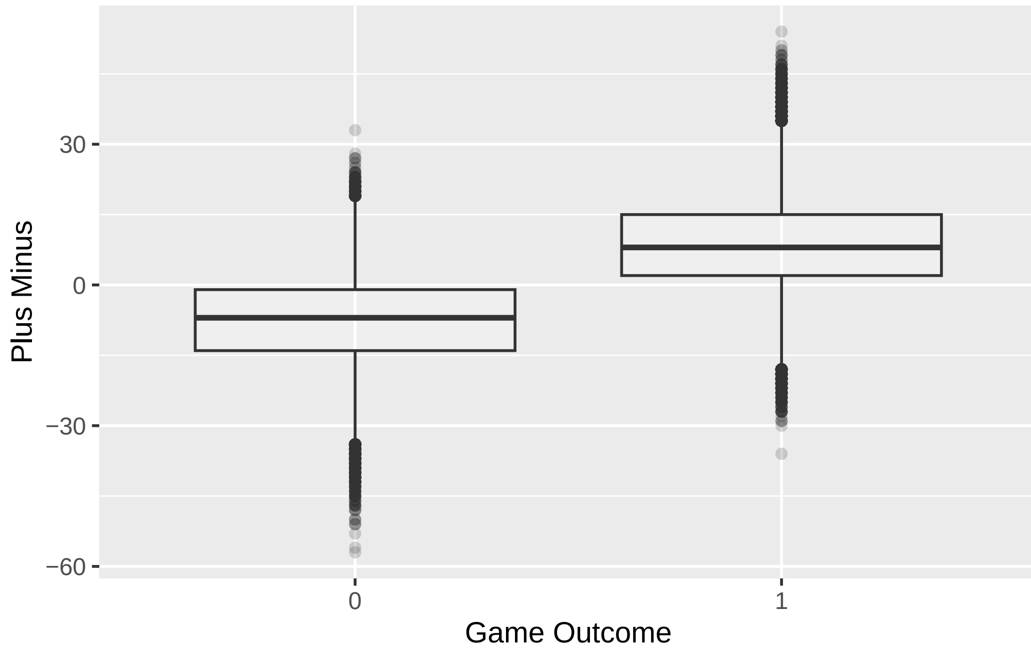

```
# Rebounds
ggplot(starters2, aes(x=as.factor(reb), y=plus_minus, alpha=0.2)) +
  geom_boxplot()
```




```
# Turnovers
ggplot(starters2, aes(x=as.factor(to), y=plus_minus, alpha=0.2)) +
  geom_boxplot()
```




```r
# Game outcome
ggplot(starters) +
  geom_boxplot(aes(y=plus_minus, x=as.factor(player_game_outcome)), alpha=0.2) +
  xlab('Game Outcome') + ylab('Plus Minus')
```




```
#yeah that makes sense
```

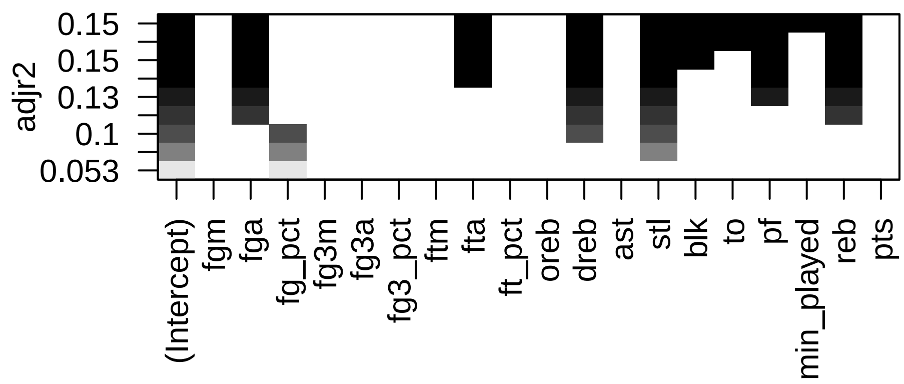
What if we use statistical model building techniques to determine what factors have the greatest influence on the plus minus variable

```
#BEST SUBSETS RAAAAAAHHHH
stats <- starters %>% select(fgm:plus_minus, min_played)
pm.subsets <- regsubsets(data=stats, plus_minus ~ .)
```

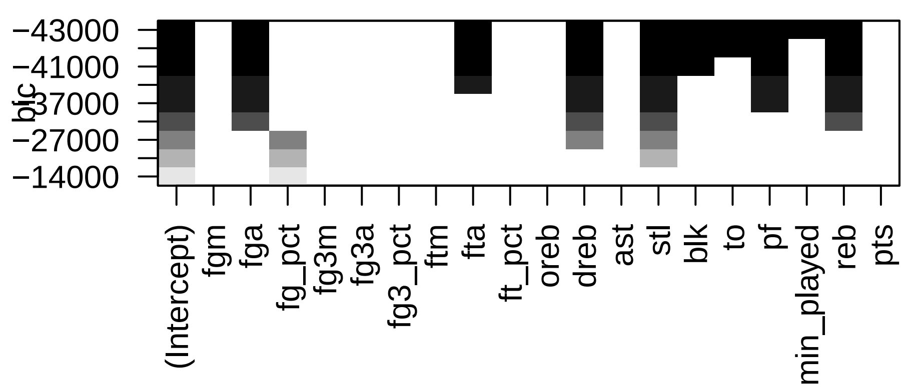
```
Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
force.in, : 2 linear dependencies found
```

```
Reordering variables and trying again:
```

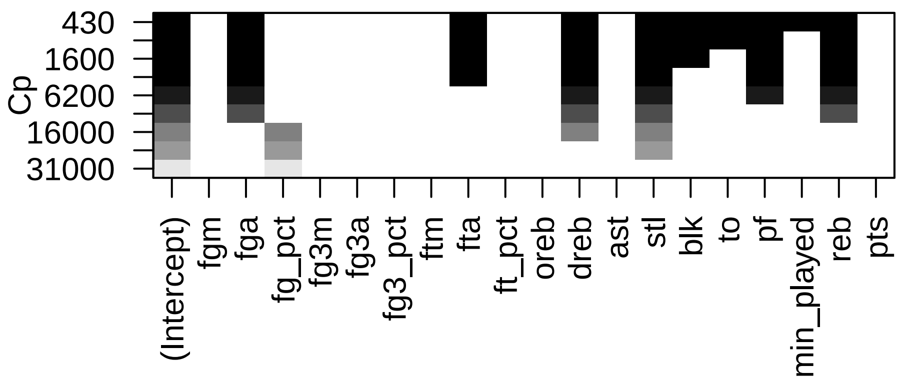
```
plot(pm.subsets, scale='adjr2')
```




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



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




```r
# stepwise?
# null.model <- lm(plus_minus ~ 1, data=stats)
# full.model <- lm(plus_minus ~ ., data=stats)
#
# step(null.model,
#      scope=list(lower=null.model, upper=full.model),
#      direction='both', trace=1) |> summary()
```

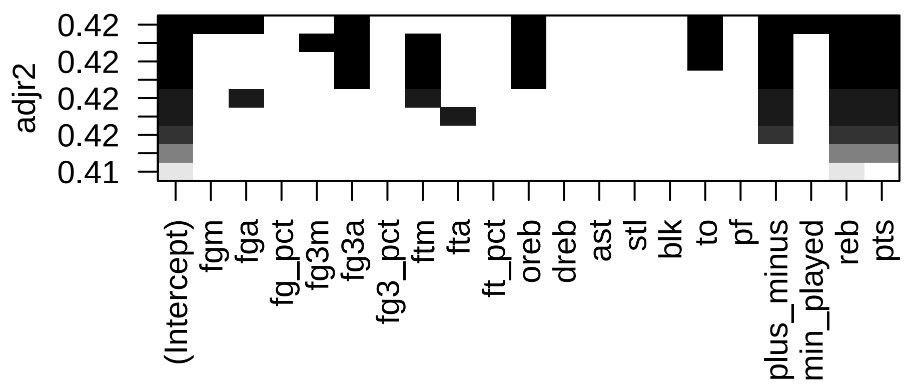
From best subsets there are a few standout variables for predicting plus minus: field goals attempted, defensive rebounds, steals, and rebounds to name a few. Interestingly, points seems to have very little effect on the model's accuracy.

```r
stats2 <- starters %>% select(fgm:plus_minus, min_played, player_game_outcome)
outcome.subsets <- regsubsets(data=stats2, player_game_outcome ~ .)
```

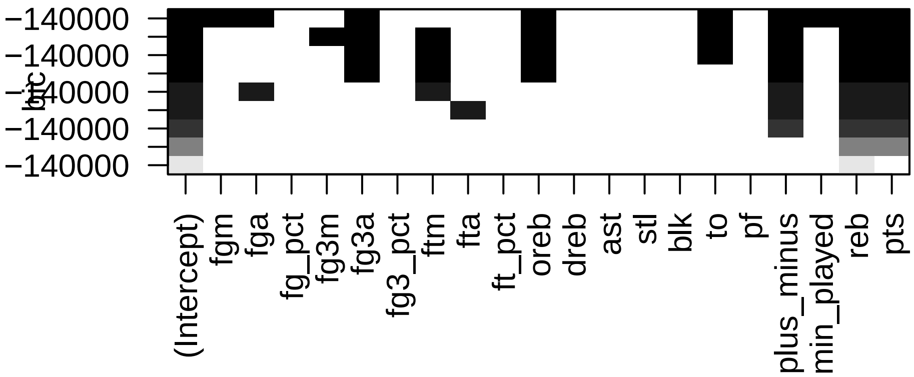
```
Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
force.in, : 2 linear dependencies found
```

```
Reordering variables and trying again:
```

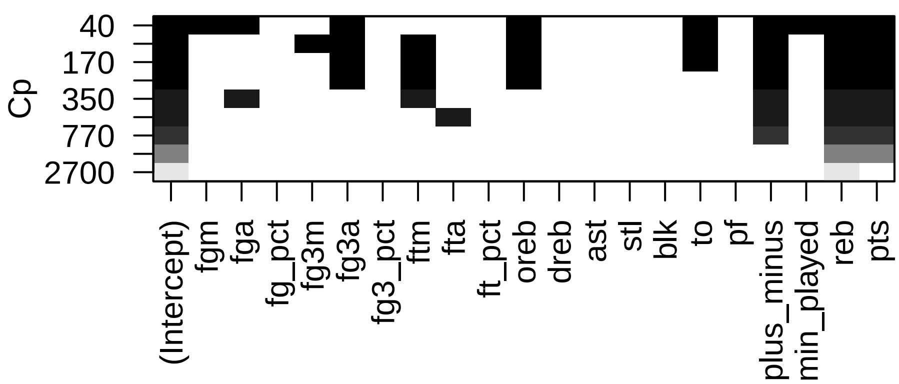
```r
plot(outcome.subsets, scale='adjr2')
```




```r
plot(outcome.subsets, scale='bic')
```




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



Initial Modelling

```
mod1 <- lm(data = starters, formula = plus_minus ~ stl + reb + pts + fg_pct + blk + pf)
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
stl	1.0	0.97, 1.1	<0.001
reb	0.28	0.26, 0.29	<0.001
pts	0.16	0.15, 0.16	<0.001
fg_pct	11	11, 11	<0.001
blk	0.50	0.45, 0.54	<0.001
pf	-0.47	-0.50, -0.44	<0.001

Stratifying by season

```
starters_season <- starters %>%
  group_by(player_name, season) %>%
  arrange(player_name) %>%
  summarise(
    total_min_played = sum(min_played),
    overall_pm = sum(plus_minus),
    pm_per_min = overall_pm / total_min_played,
  )
```

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

```
# for(i in 2003:2022){
#   starters_season %>% filter(season == i) %>% arrange(desc(overall_pm)) %>% head(n=20) %

# }

starters_season <- starters_season %>%
  group_by(season) %>%
```

```r
    mutate(
      season_avg_pm = mean(overall_pm),
      season_pm_sd = sd(overall_pm),
    )

  starters_season <- starters_season %>%
    group_by(player_name, season) %>%
    mutate(
      performance = ifelse(overall_pm > season_avg_pm + season_pm_sd, "high",
                      ifelse(overall_pm < season_avg_pm - season_pm_sd, "low", "avg"))
    )
```

```r
  ## Expand starters season by joining starters and starters_season df
  starters_season2 <- merge(starters, starters_season, by = c('player_name', 'season'), rela
```

```r
  starters_season2020 <- filter(starters_season2, season == 2020 & team_city == 'Los Angeles
```

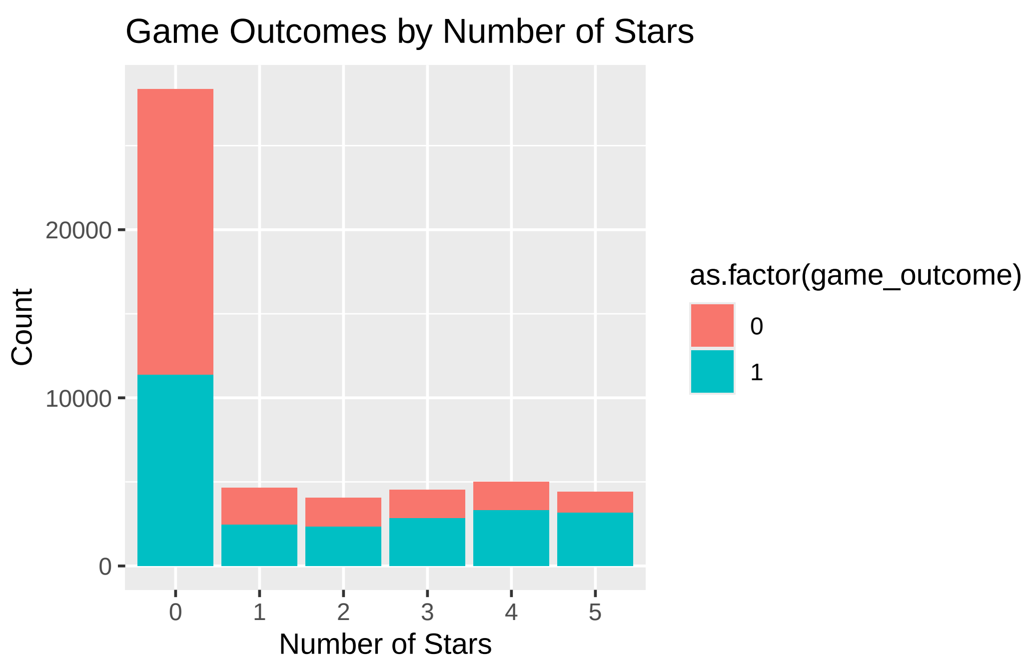
```r
  starters_season2 <- starters_season2 %>%
    mutate(
      star = ifelse(performance == "high", 1, 0),
      avg = ifelse(performance == "avg", 1, 0),
      low = ifelse(performance == "low", 1, 0),
    )

  team_games <- starters_season2 %>%
    group_by(team_id, game_id) %>%
    summarise(
      num_star = sum(star),
      num_avg = sum(avg),
      num_low = sum(low))%>%
    filter(num_star + num_avg + num_low == 5)
```

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

```r
  team_season_final <- starters_season2 %>%
    select(home_team_id, visitor_team_id, home_team_wins, game_id, team_id) %>%
    group_by(game_id, team_id) %>%
    unique()
```

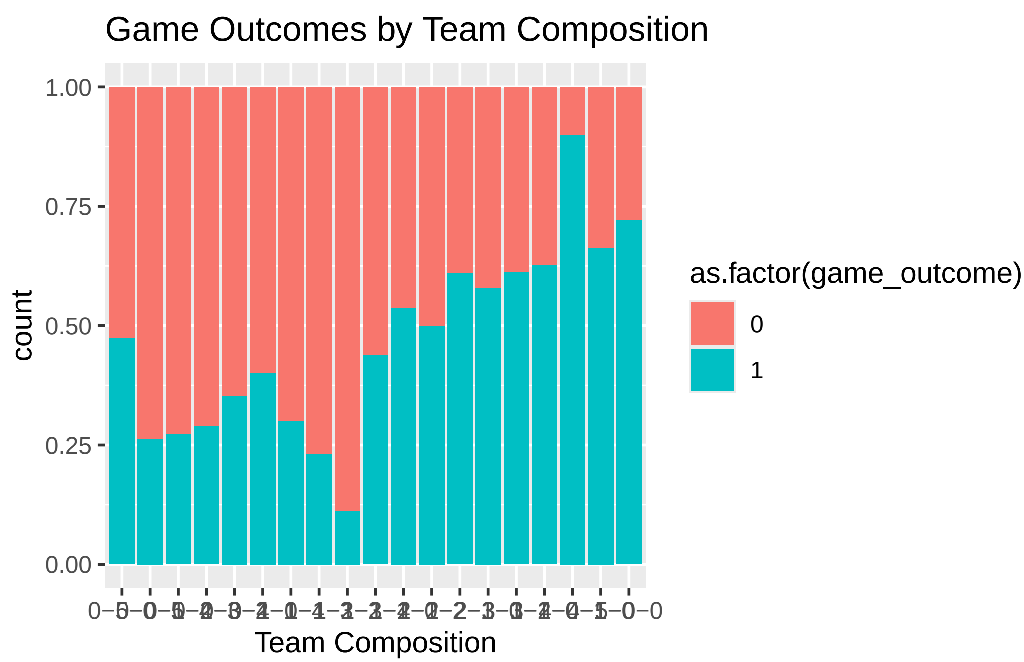
```r
team_games <- inner_join(team_games, team_season_final, by = c('game_id', 'team_id'), rela

team_games <- team_games %>%
  mutate(game_outcome = ifelse((team_id == home_team_id & home_team_wins == 1) | (team_id

cols<-c('num_star', 'num_avg', 'num_low')
team_games$composition <- do.call(paste, c(team_games[cols], sep = "-"))

team_games$composition <- relevel(factor(team_games$composition), ref = "0-5-0")

tab <- table(team_games$composition)
team_games_filtered <- team_games[team_games$composition %in% names(tab)[tab>1000],]
```

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

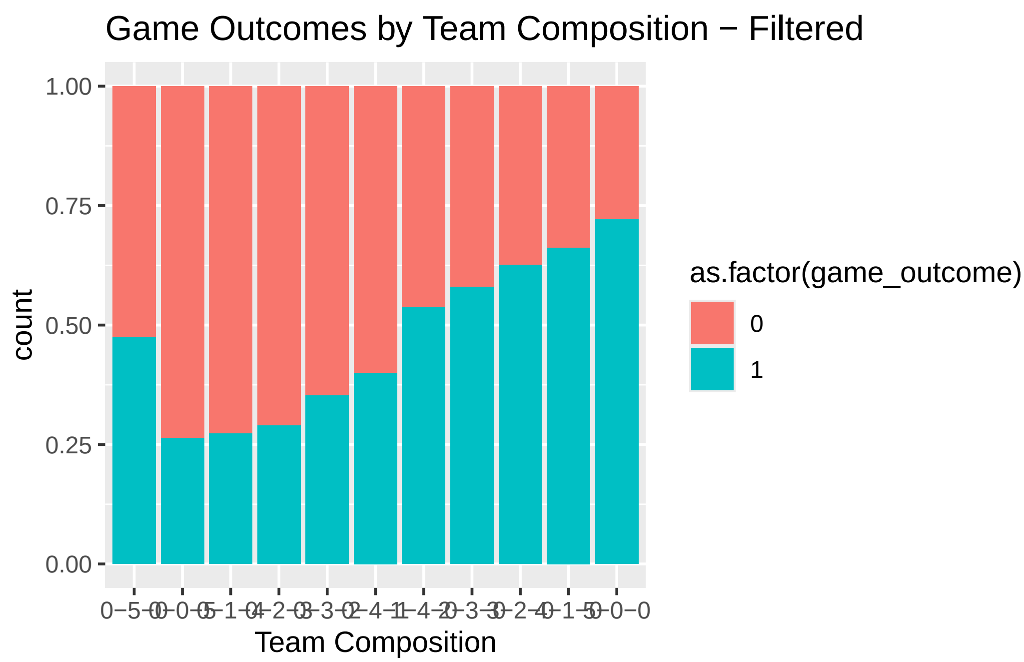



```r
# ggplot(cabbage_exp, aes(x = Date, y = Weight, fill = Cultivar)) +
#   geom_col(position = "fill")
```

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




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




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

#linear model
glm_star <- glm(data = train, formula = game_outcome ~ num_star, family =  "binomial")
glm_avg <- glm(data = train, formula = game_outcome ~ num_avg, family =  "binomial")
glm_low <- glm(data = train, formula = game_outcome ~ num_low, family =  "binomial")

tbl_regression(glm_star)
```

```
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	**log(OR)**	**95% CI**	**p-value**
num_star	0.28	0.27, 0.29	<0.001

```r
tbl_regression(glm_avg)
```

```
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	**log(OR)**	**95% CI**	**p-value**
num_avg	-0.09	-0.10, -0.08	<0.001

```r
tbl_regression(glm_low)
```

```
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	**log(OR)**	**95% CI**	**p-value**
num_low	-0.36	-0.37, -0.34	<0.001

```r
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$game_outcome)
accuracy
```

```
[1] 0.5715406
```

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

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

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)    
(Intercept)       -0.10757    0.01946  -5.527 3.26e-08 ***
composition0-0-5  -1.00303    0.07996 -12.544  < 2e-16 ***
composition0-1-4  -0.92833    0.05458 -17.007  < 2e-16 ***
composition0-2-3  -0.77351    0.04866 -15.895  < 2e-16 ***
composition0-3-2  -0.48489    0.04448 -10.902  < 2e-16 ***
composition0-4-1  -0.28272    0.03745  -7.549 4.39e-14 ***
composition1-0-4  -0.40325    0.73056  -0.552   0.5810    
composition1-1-3  -0.84794    0.52659  -1.610   0.1073    
composition1-2-2 -10.45845   48.77263  -0.214   0.8302    
composition1-3-1  -0.16507    0.16330  -1.011   0.3121    
composition1-4-0   0.25310    0.03887   6.511 7.45e-11 ***
composition2-1-2  -0.07475    0.60584  -0.123   0.9018    
composition2-2-1   0.46859    0.27232   1.721   0.0853 .  
composition2-3-0   0.43974    0.04087  10.760  < 2e-16 ***
composition3-1-1   0.37584    0.36895   1.019   0.3084    
composition3-2-0   0.63129    0.03958  15.951  < 2e-16 ***
composition4-0-1   2.18702    1.06084   2.062   0.0392 *  
composition4-1-0   0.79059    0.03877  20.393  < 2e-16 ***
composition5-0-0   1.04109    0.04226  24.634  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56580  on 40813  degrees of freedom
Residual deviance: 53536  on 40795  degrees of freedom
AIC: 53574

Number of Fisher Scoring iterations: 9
```

```
tbl_regression(glm_comp)
```

```
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	**log(OR)**	**95% CI**	**p-value**
composition			
0-5-0	—	—	
0-0-5	-1.0	-1.2, -0.85	<0.001
0-1-4	-0.93	-1.0, -0.82	<0.001
0-2-3	-0.77	-0.87, -0.68	<0.001
0-3-2	-0.48	-0.57, -0.40	<0.001
0-4-1	-0.28	-0.36, -0.21	<0.001
1-0-4	-0.40	-2.0, 1.0	0.6
1-1-3	-0.85	-2.0, 0.13	0.11
1-2-2	-10		0.8
1-3-1	-0.17	-0.49, 0.15	0.3
1-4-0	0.25	0.18, 0.33	<0.001
2-1-2	-0.07	-1.3, 1.1	>0.9
2-2-1	0.47	-0.06, 1.0	0.085
2-3-0	0.44	0.36, 0.52	<0.001
3-1-1	0.38	-0.34, 1.1	0.3
3-2-0	0.63	0.55, 0.71	<0.001
4-0-1	2.2	0.49, 5.1	0.039
4-1-0	0.79	0.71, 0.87	<0.001
5-0-0	1.0	0.96, 1.1	<0.001

```r
blorr::blr_model_fit_stats(glm_comp)
```

```
                              Model Fit Statistics
---------------------------------------------------------------------------------
Log-Lik Intercept Only:     -28290.108    Log-Lik Full Model:           -26768.045
Deviance(40795):             53536.091    LR(18):                         3044.125
                                          Prob > LR:                         0.000
MCFadden's R2                    0.054    McFadden's Adj R2:                 0.053
ML (Cox-Snell) R2:               0.072    Cragg-Uhler(Nagelkerke) R2:        0.096
McKelvey & Zavoina's R2:         0.094    Efron's R2:                        0.073
Count R2:                        0.612    Adj Count R2:                      0.224
BIC:                         53737.809    AIC:                           53574.091
---------------------------------------------------------------------------------
```

```r
predicted_values <- predict(glm_comp, newdata = test)


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

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

```
[1] 0.5921207
```

```r
init_split <- initial_split(team_games_filtered, prop = 0.8)
train <- training(init_split)
test  <- testing(init_split)

glm_comp_filtered <- glm(data = train, formula = game_outcome ~ composition, family =  "bi
tbl_regression(glm_comp_filtered)
```

```
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	**log(OR)**	**95% CI**	**p-value**
composition			
0-5-0	—	—	
0-0-5	-0.89	-1.1, -0.74	<0.001
0-1-4	-0.87	-0.98, -0.77	<0.001
0-2-3	-0.83	-0.92, -0.73	<0.001
0-3-2	-0.49	-0.58, -0.41	<0.001
0-4-1	-0.30	-0.37, -0.23	<0.001
1-4-0	0.23	0.15, 0.30	<0.001
2-3-0	0.43	0.35, 0.51	<0.001
3-2-0	0.61	0.53, 0.68	<0.001
4-1-0	0.74	0.67, 0.82	<0.001
5-0-0	1.1	0.99, 1.2	<0.001

```r
blorr::blr_model_fit_stats(glm_comp_filtered)
```

```
                               Model Fit Statistics
---------------------------------------------------------------------------------
Log-Lik Intercept Only:     -28091.856    Log-Lik Full Model:            -26615.442
Deviance(40517):             53230.885    LR(10):                          2952.828
                                          Prob > LR:                          0.000
MCFadden's R2                    0.053    McFadden's Adj R2:                  0.052
ML (Cox-Snell) R2:               0.070    Cragg-Uhler(Nagelkerke) R2:         0.094
McKelvey & Zavoina's R2:         0.088    Efron's R2:                         0.071
Count R2:                        0.610    Adj Count R2:                       0.220
BIC:                         53347.592    AIC:                            53252.885
---------------------------------------------------------------------------------
```