2024 Reds Hackathon

Kai Franke

2024-01-19

Downloading the data

```
setwd("C:/Users/kaifr/OneDrive/Miscellaneous/Desktop/Productivity/Baseball Research/Reds Hackathon")

fangraph = read.csv("fangraphs_season_level.csv")
savant = read.csv("savant_pitch_level.csv")
```

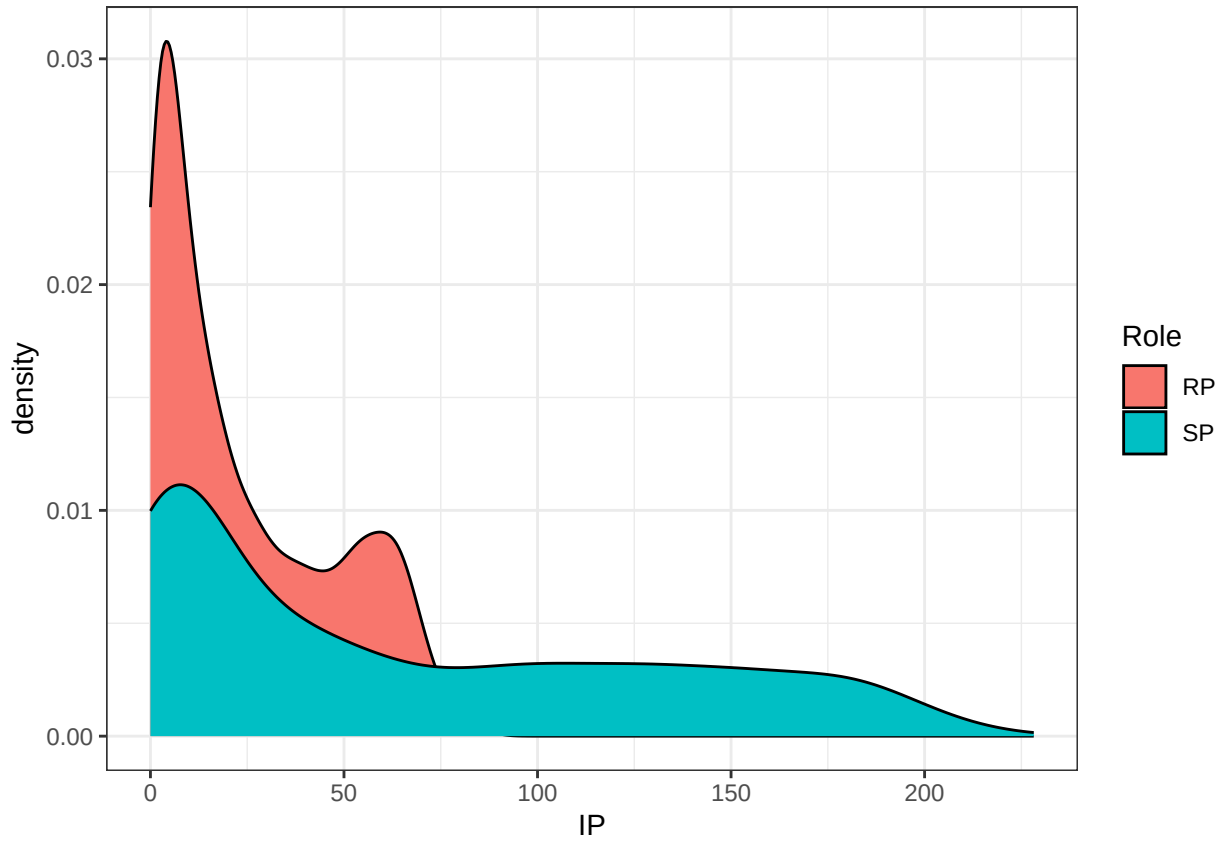
Initial Filtering

Made it so that each pitcher has a decent sample size. IP is greater than the median.

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
theme_set(theme_bw())

ggplot(fangraph, aes(IP, fill = Role)) +
  geom_density()
```




```
summary(filter(fangraph, Role == "SP")$IP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.10    6.10   42.20   64.96  118.55  228.20
```

```
summary(filter(fangraph, Role == "RP")$IP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   17.00   24.61   42.60   84.10
```

```
quantile(filter(fangraph, Role == "SP")$IP, probs = c(0.33, 0.5))
```

```
##  33%  50%
## 12.2 42.2
```

```
quantile(filter(fangraph, Role == "RP")$IP, probs = c(0.33, 0.5))
```

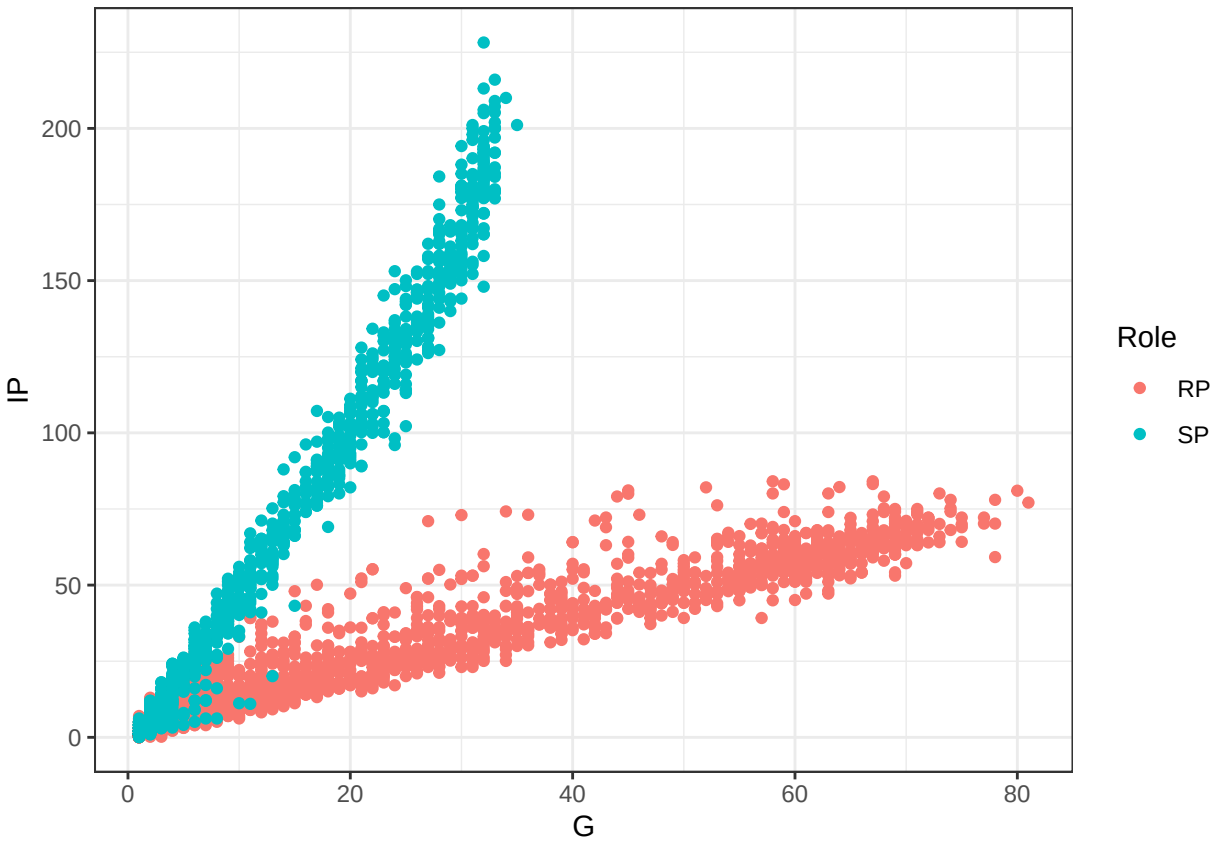
```
##    33%    50%
##  7.792 17.000
```

```
fangraph2 = filter(fangraph, ifelse(Role == "SP", IP >= 42, IP >= 17))
```

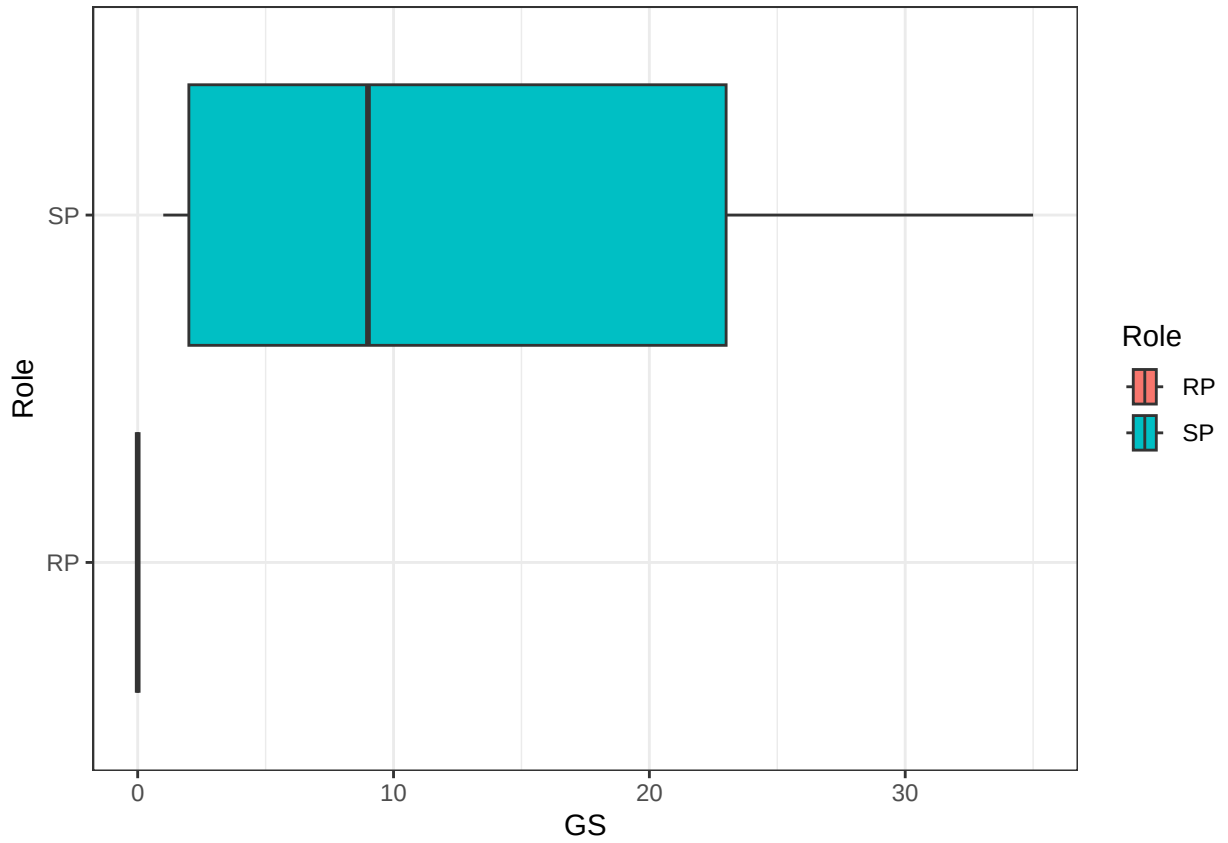
EDA

Understanding the dataset and making sure things make sense.

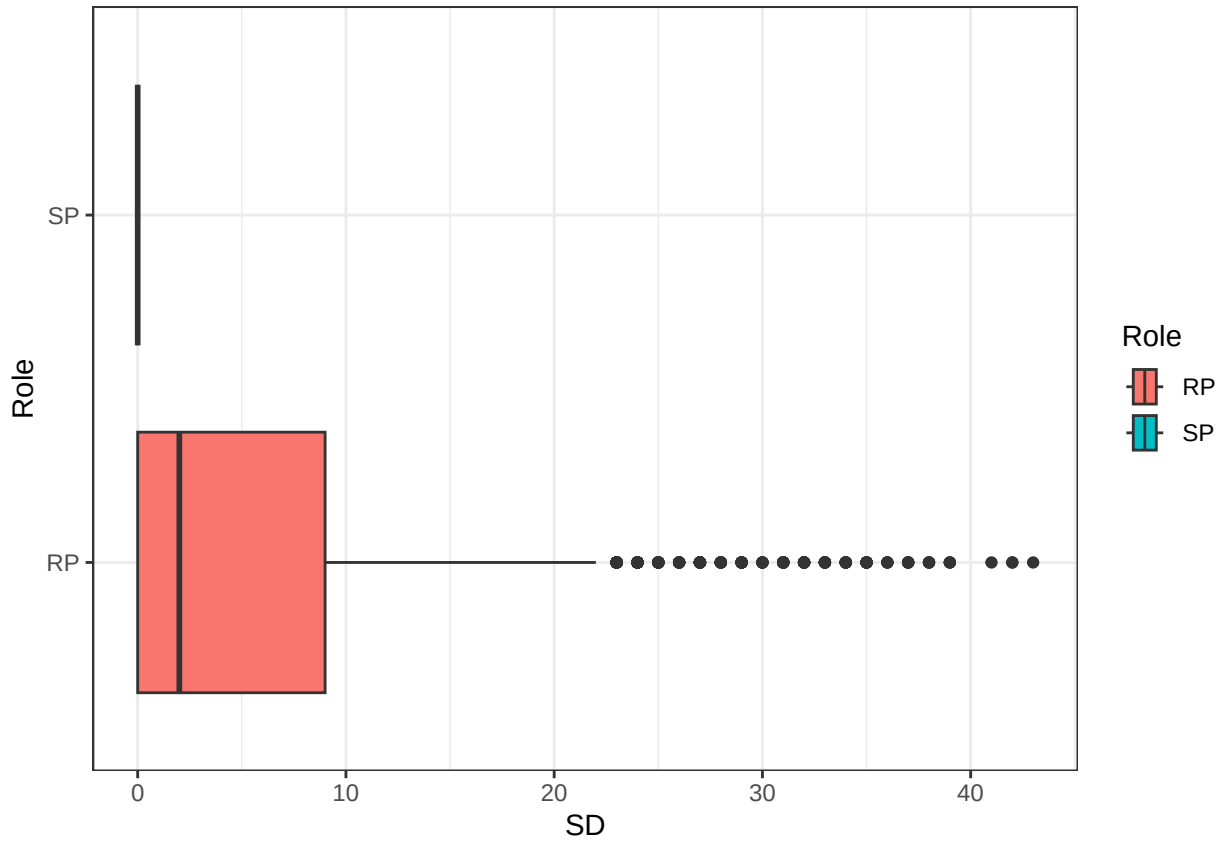
```
ggplot(fangraph, aes(G, IP, color = Role)) +
  geom_point()
```



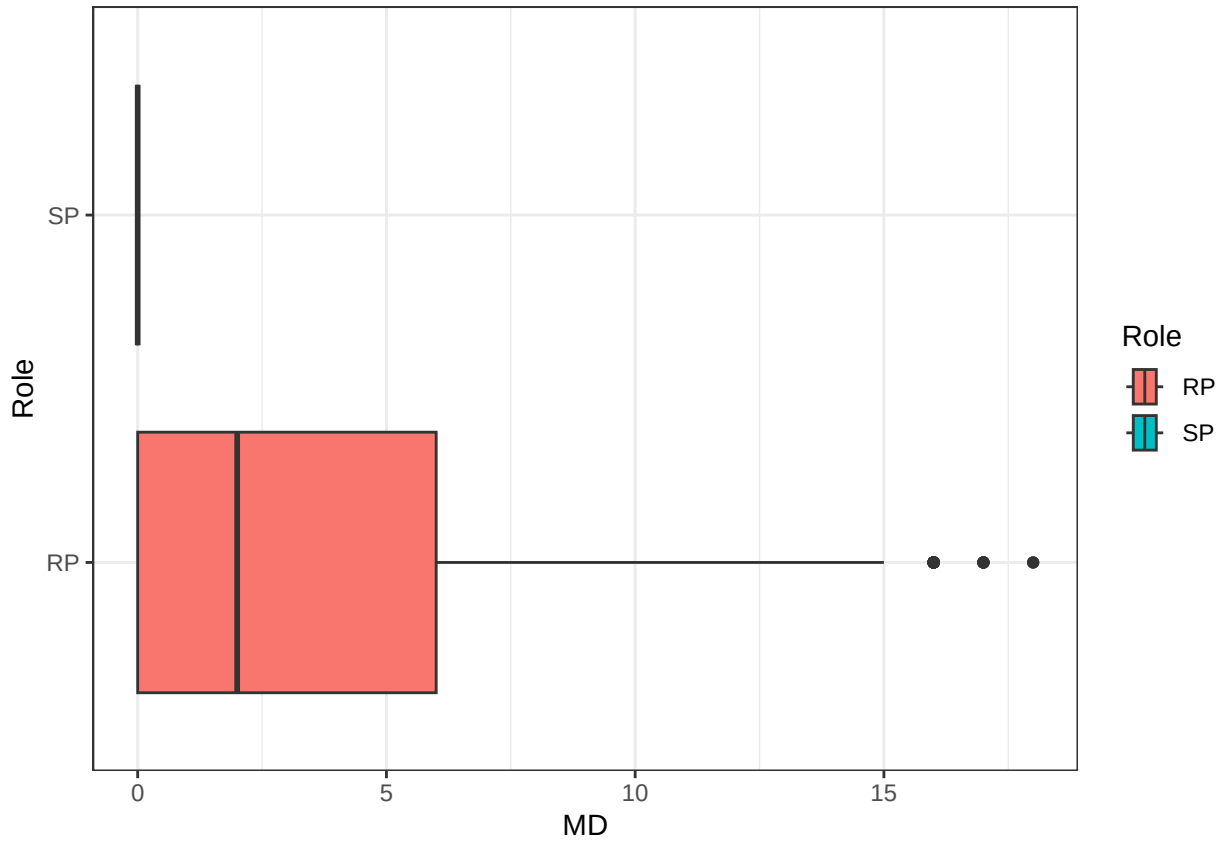

```
ggplot(fangraph, aes(GS, Role, fill = Role)) +
  geom_boxplot()
```



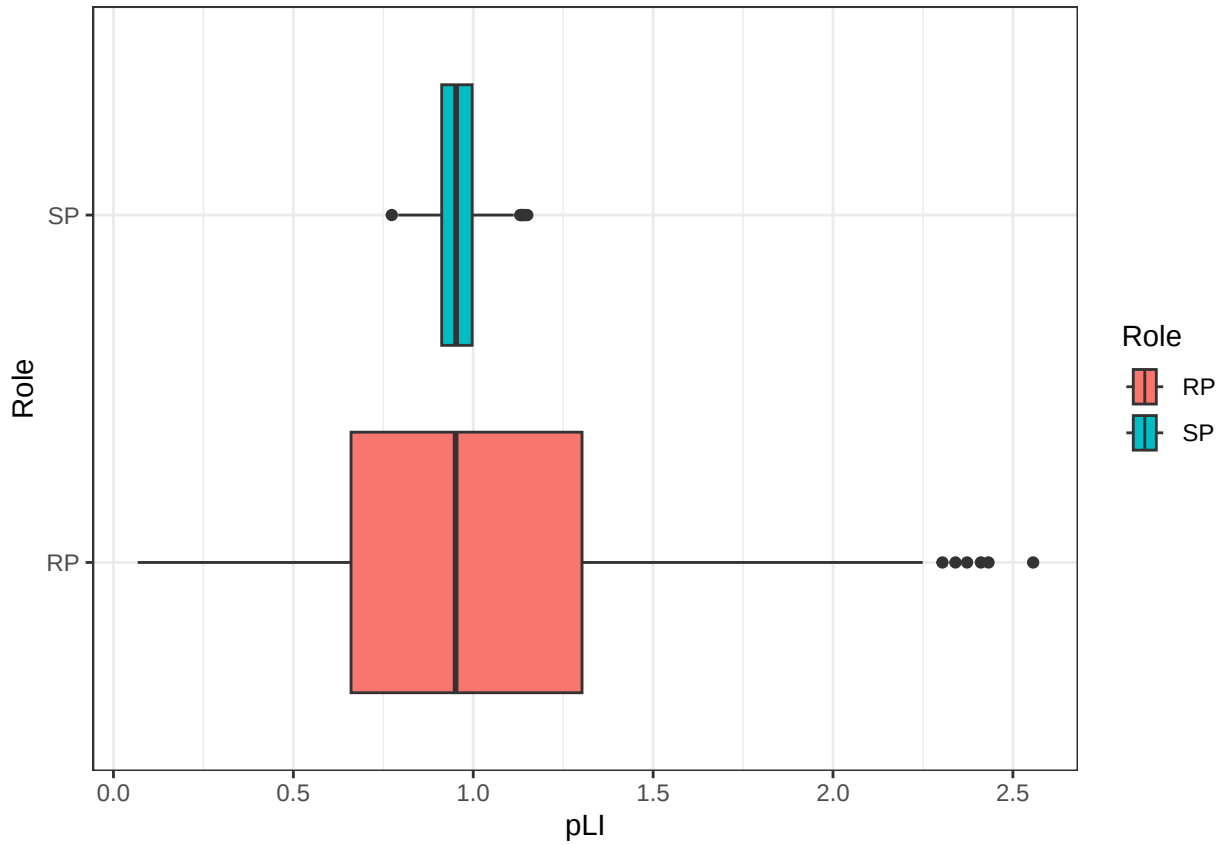

```
ggplot(fangraph, aes(SD, Role, fill = Role)) +
  geom_boxplot()
```



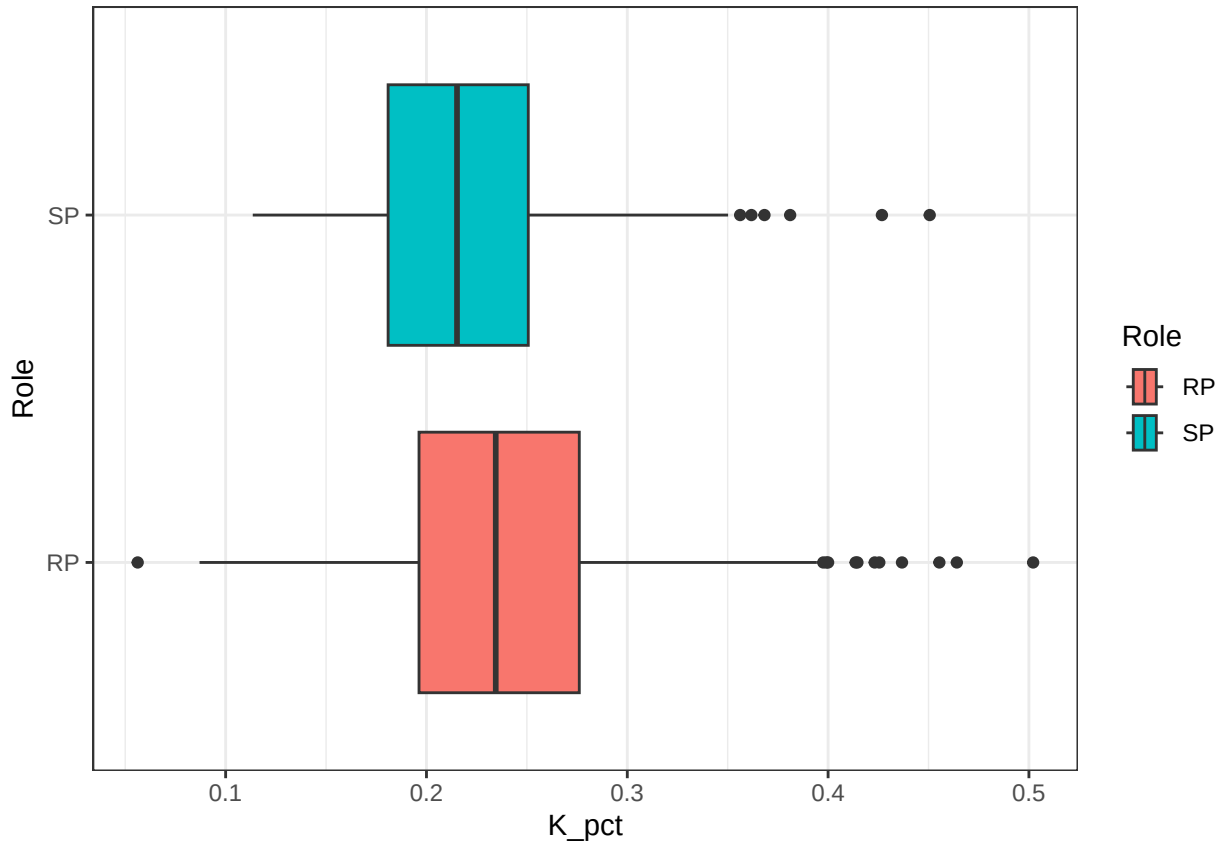

```
ggplot(fangraph, aes(MD, Role, fill = Role)) +
  geom_boxplot()
```



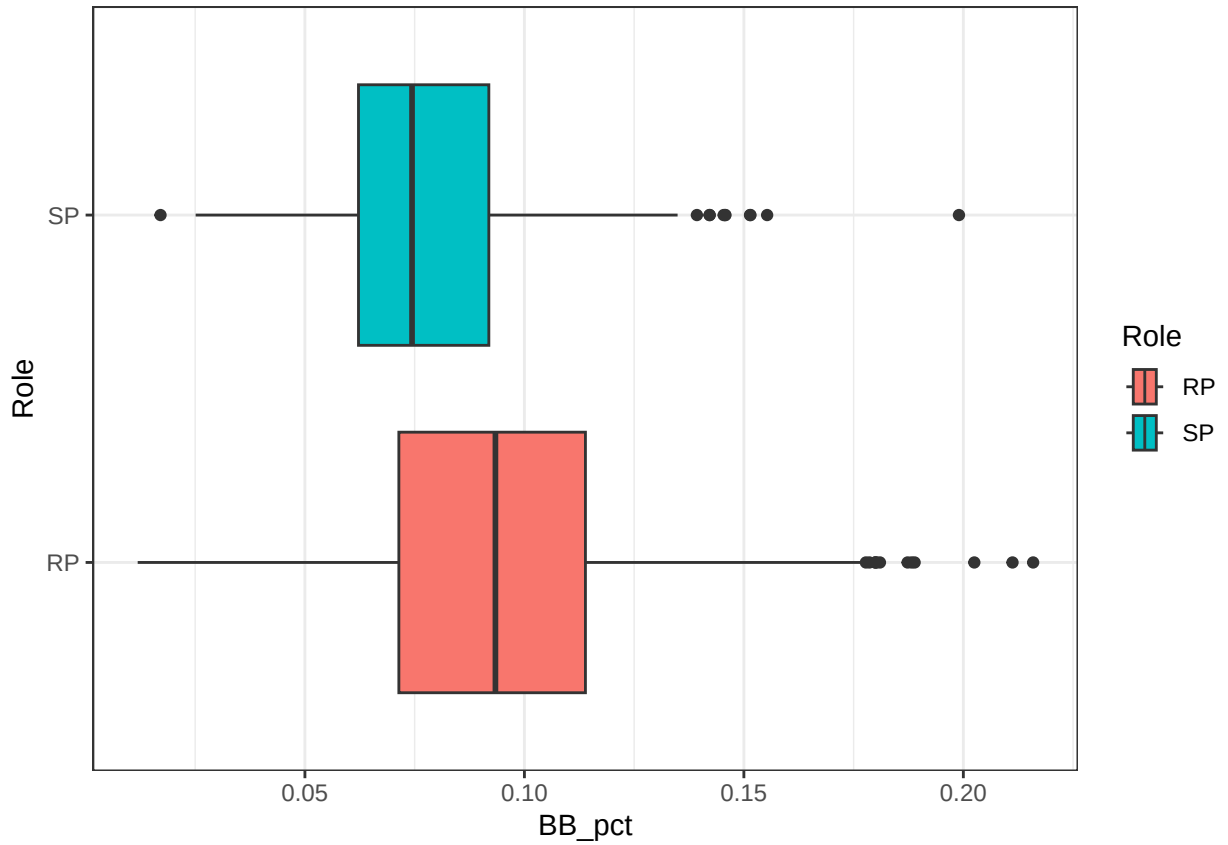

```
ggplot(fangraph2, aes(pLI, Role, fill = Role)) +
  geom_boxplot()
```



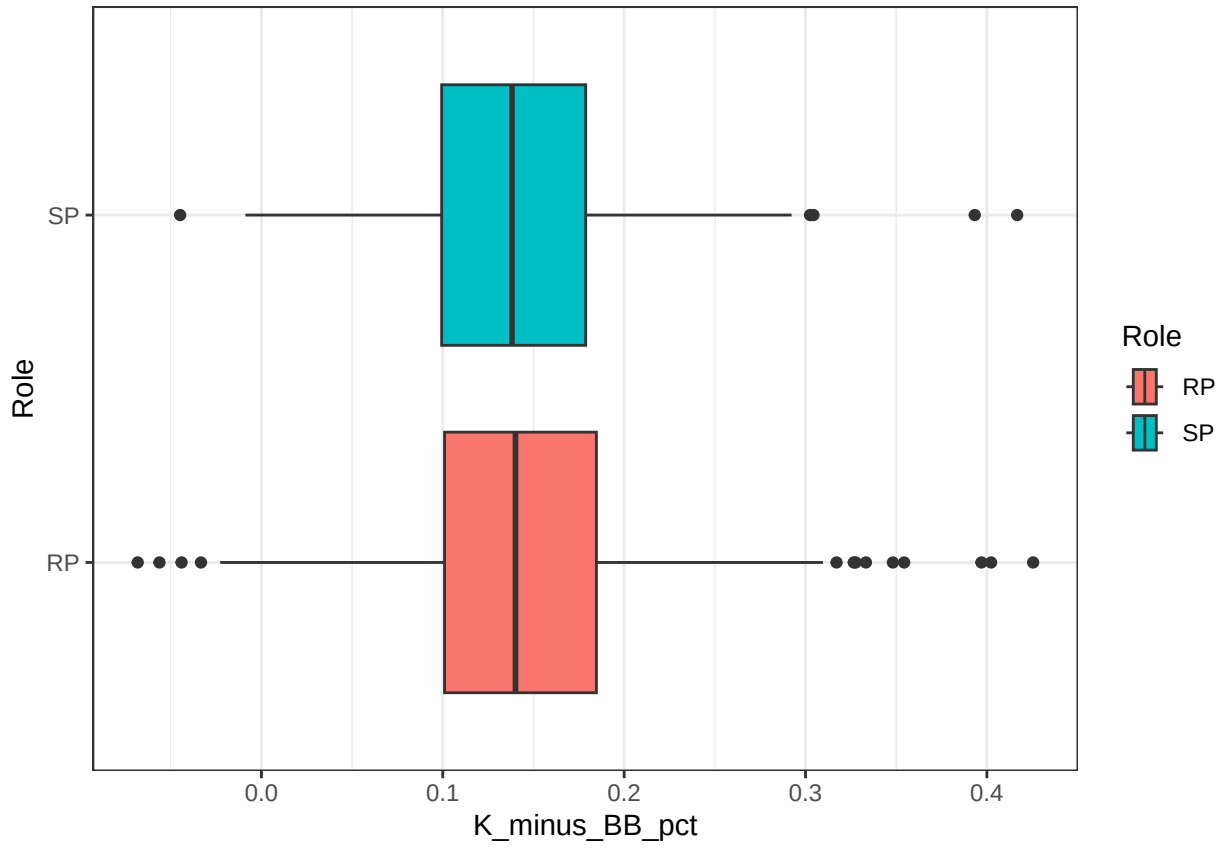

```
ggplot(fangraph2, aes(K_pct, Role, fill = Role)) +
  geom_boxplot()
```



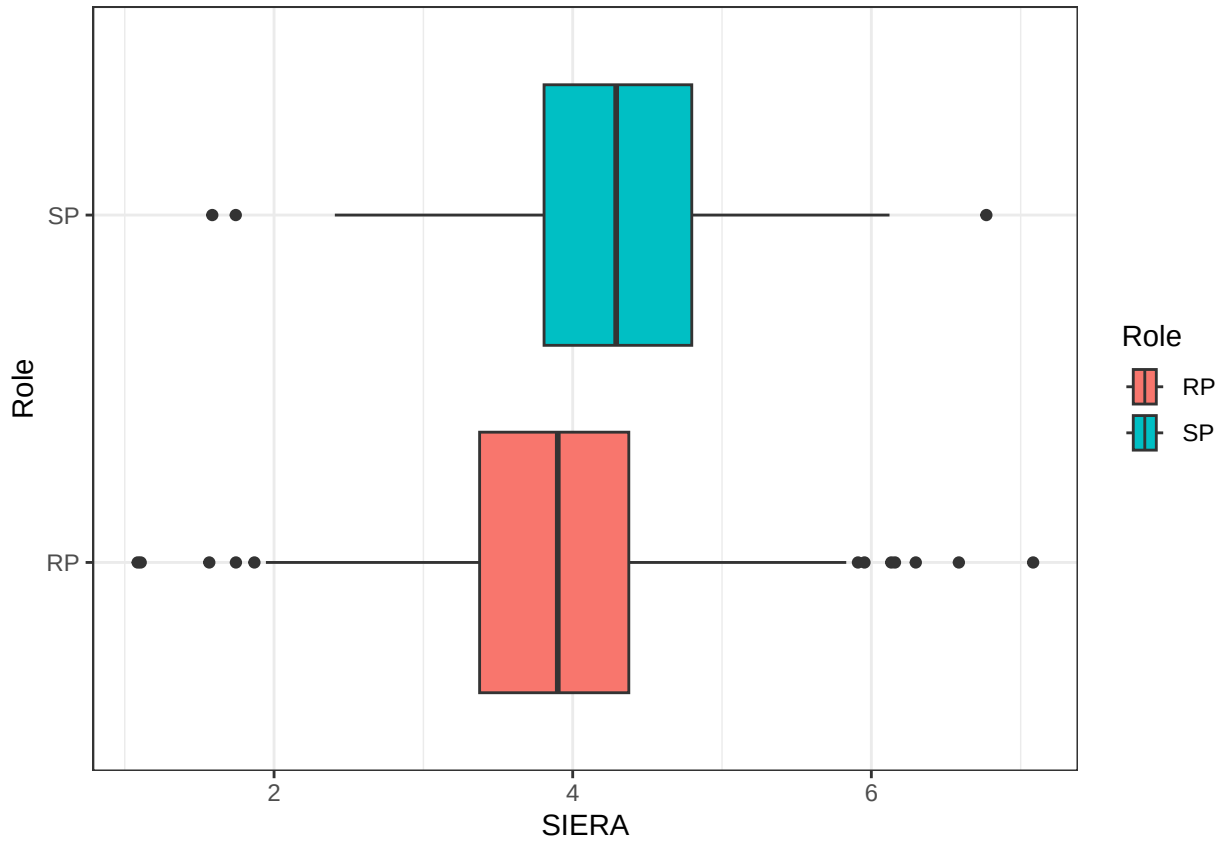

```
ggplot(fangraph2, aes(BB_pct, Role, fill = Role)) +
  geom_boxplot()
```



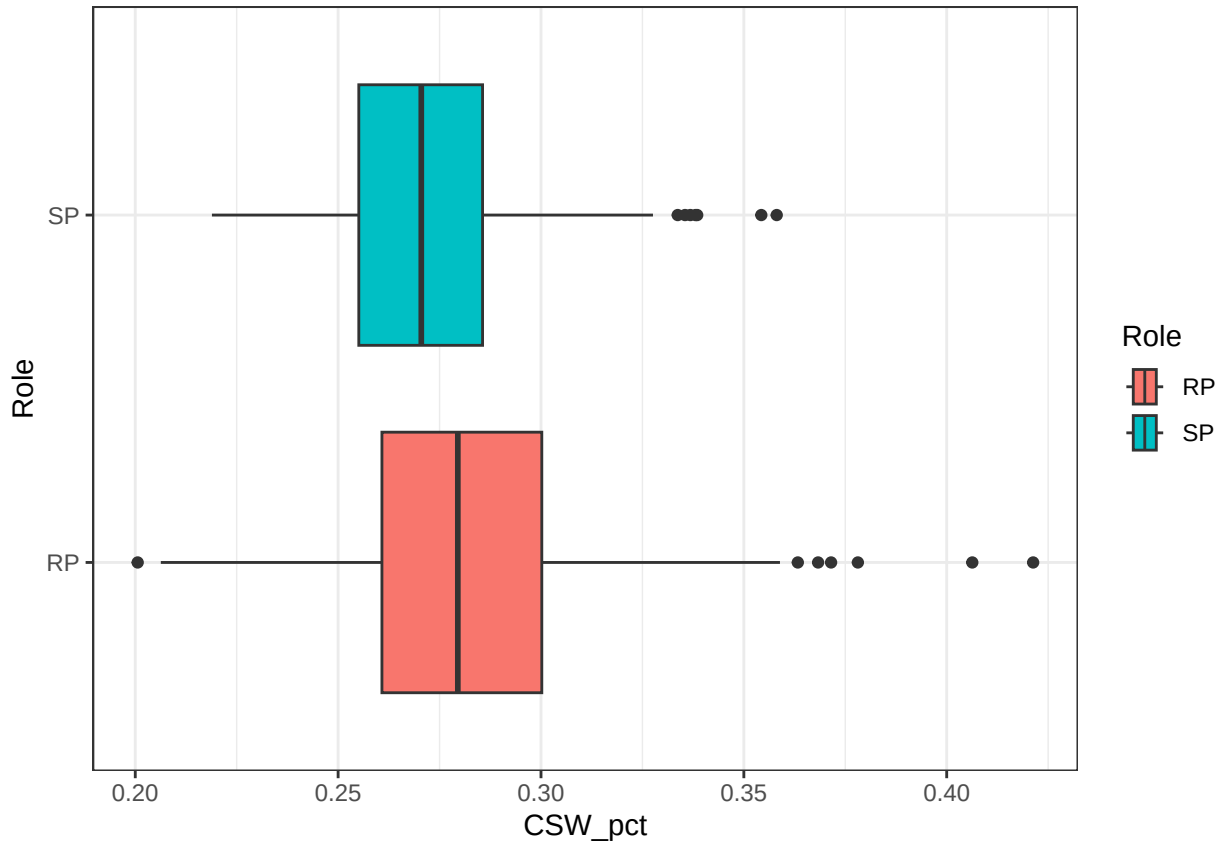

```
ggplot(fangraph2, aes(K_minus_BB_pct, Role, fill = Role)) +
  geom_boxplot()
```



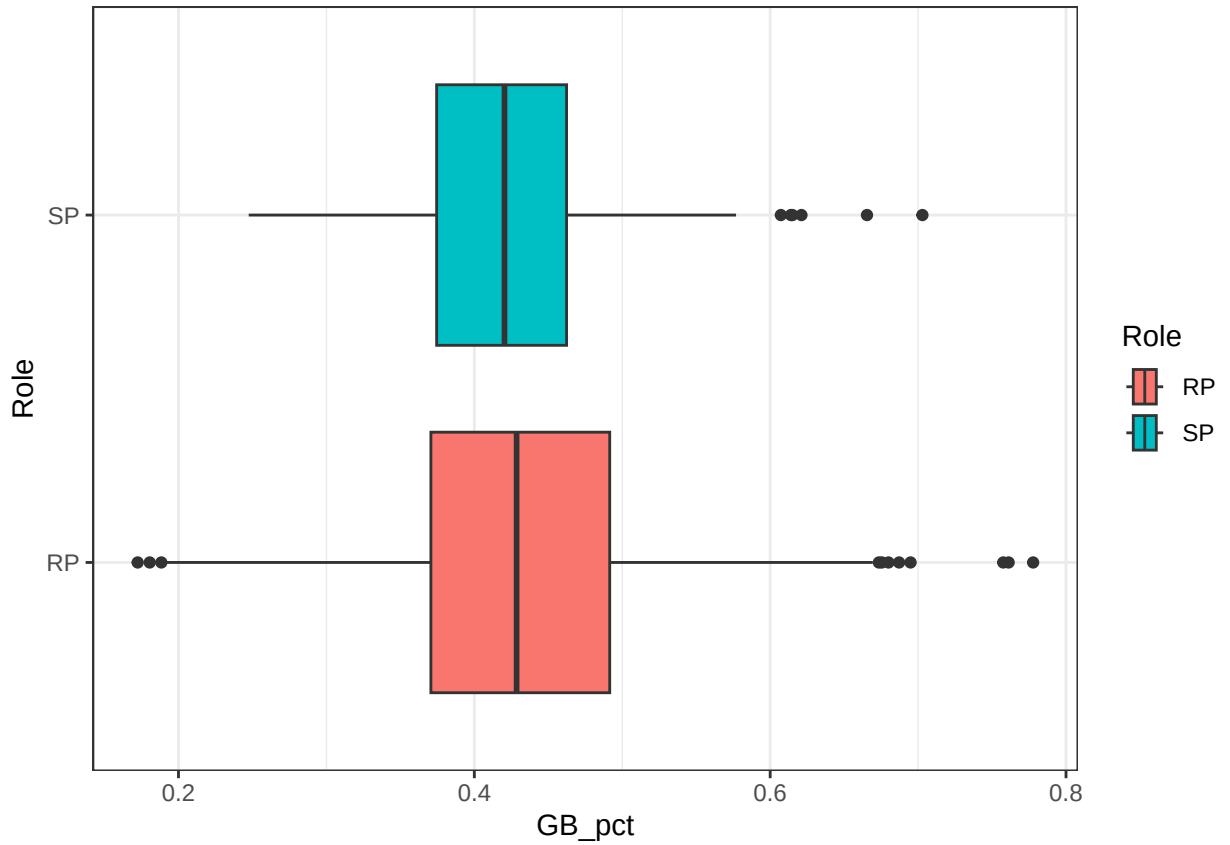

```
ggplot(fangraph2, aes(SIERA, Role, fill = Role)) +
  geom_boxplot()
```



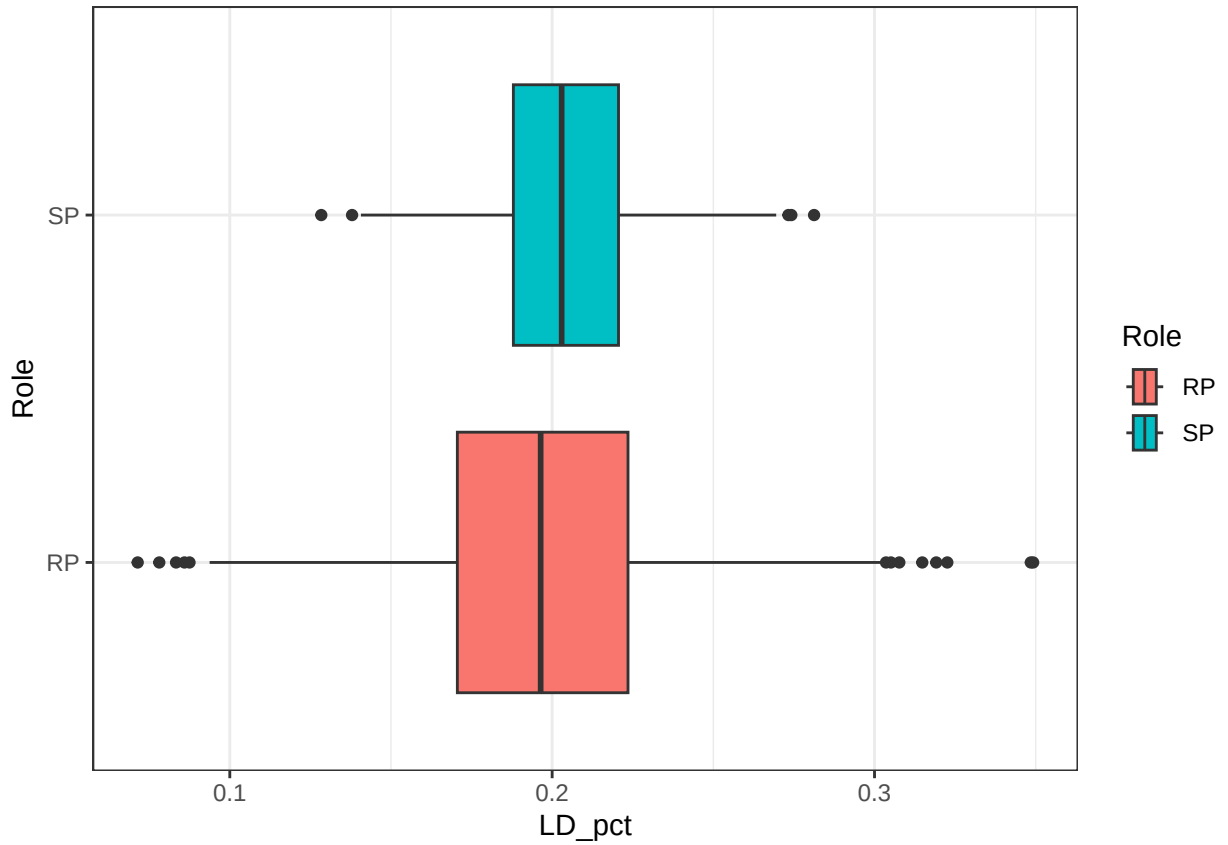

```
ggplot(fangraph2, aes(CSW_pct, Role, fill = Role)) +
  geom_boxplot()
```



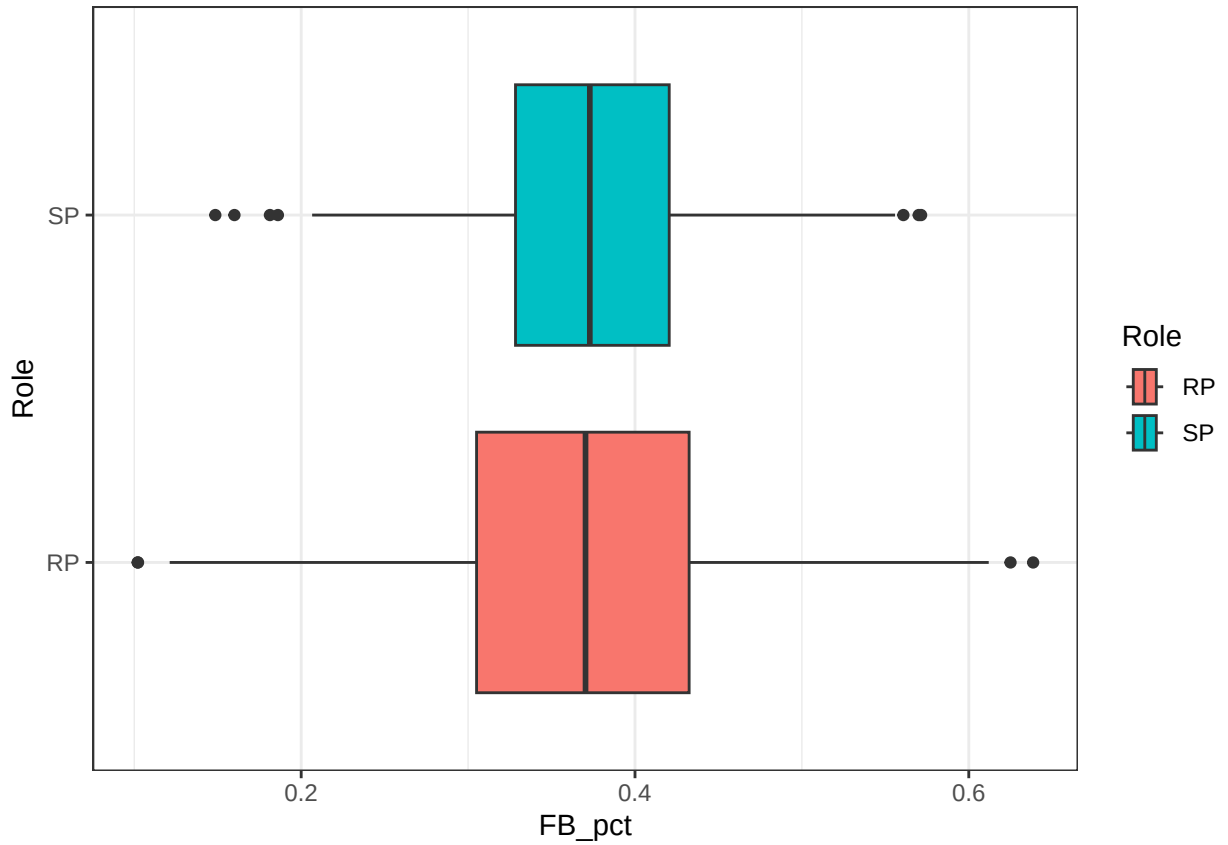

```
ggplot(fangraph2, aes(GB_pct, Role, fill = Role)) +
  geom_boxplot()
```



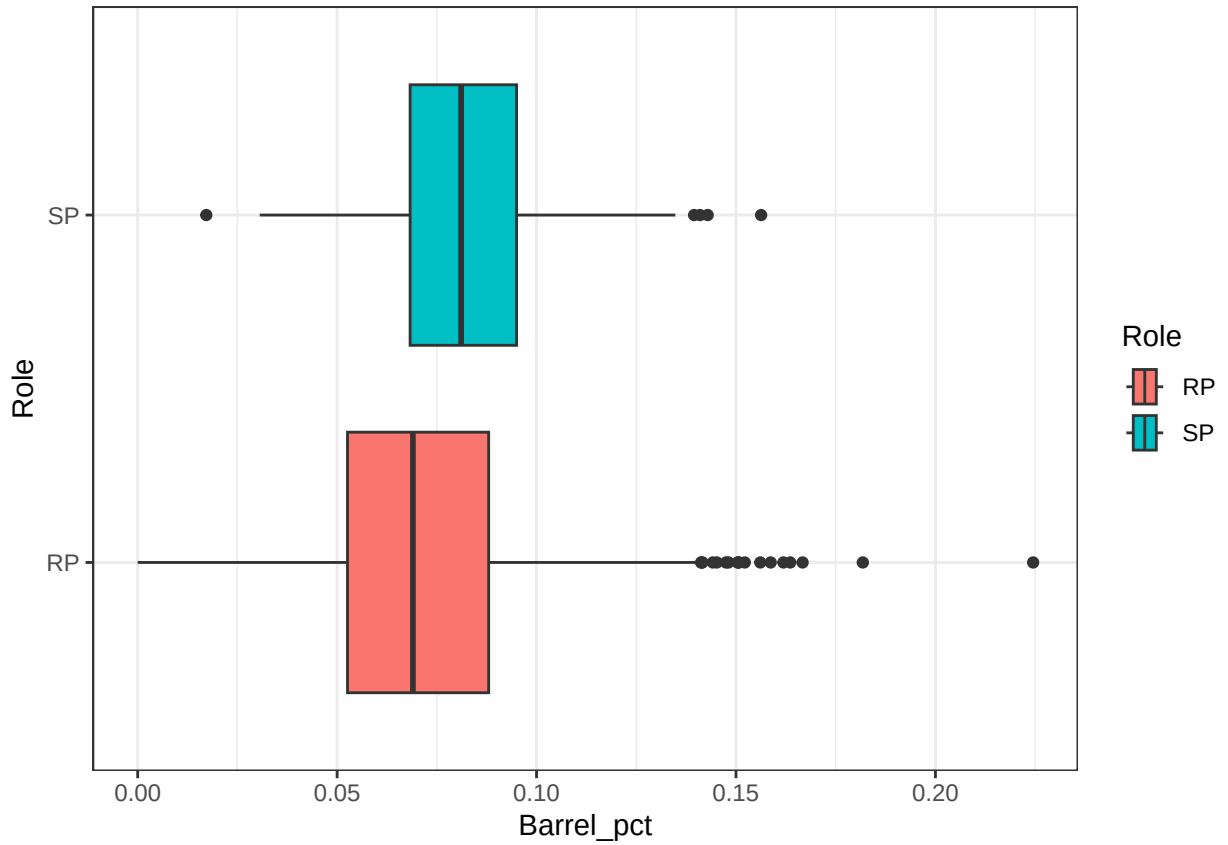

```
ggplot(fangraph2, aes(LD_pct, Role, fill = Role)) +
  geom_boxplot()
```




```
ggplot(fangraph2, aes(FB_pct, Role, fill = Role)) +
  geom_boxplot()
```




```
ggplot(fangraph2, aes(Barrel_pct, Role, fill = Role)) +
  geom_boxplot()
```




```
fangraph2 %>%
  group_by(Role) %>%
  summarise(Stuff_plus = round(mean(Stuff_plus)),
            Location_plus = round(mean(Location_plus)),
            Pitching_plus = round(mean(Pitching_plus)))
```

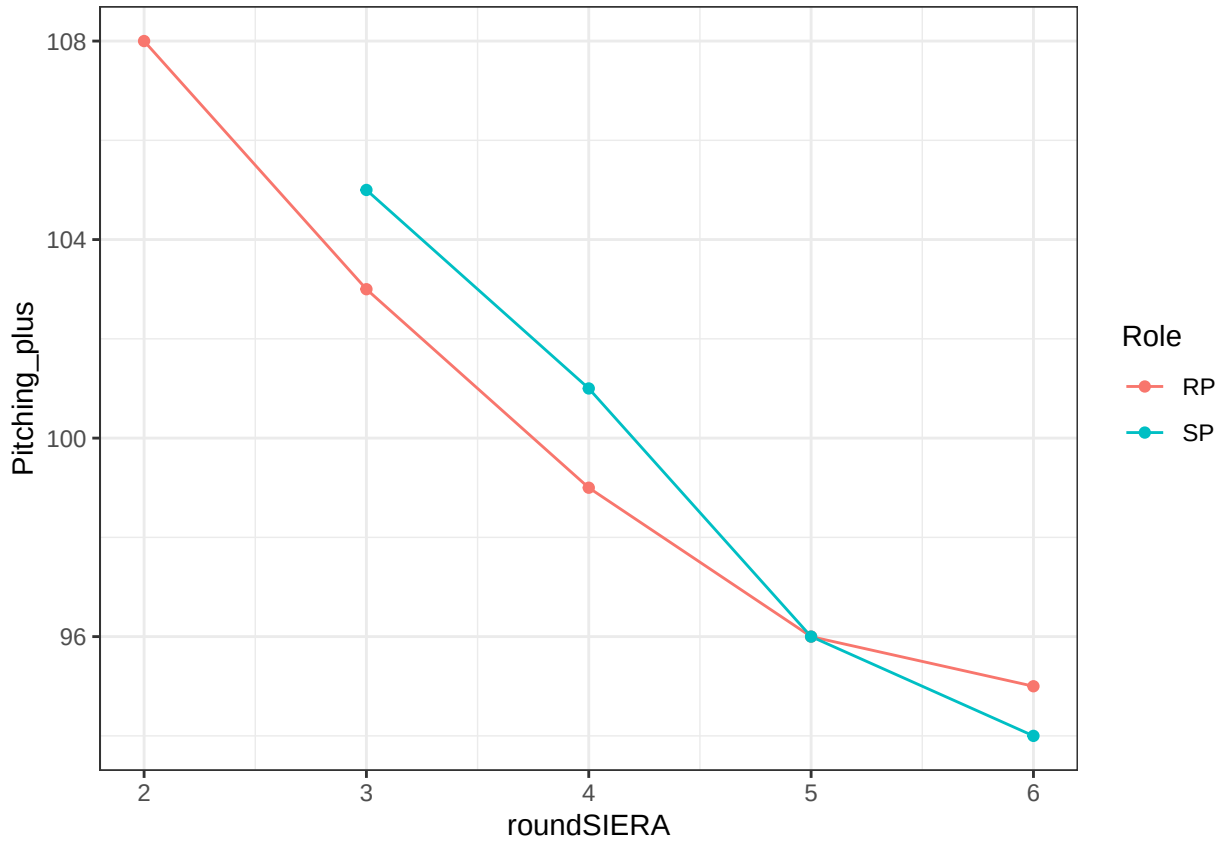
```
## # A tibble: 2 x 4
##   Role  Stuff_plus Location_plus Pitching_plus
##   <chr>      <dbl>         <dbl>         <dbl>
## 1 RP           103            99           100
## 2 SP            97           101           100
```

Seeing how SIERA relates to Stuff+, Location+, and Pitching+

```
sieraRole = fangraph2 %>%
  mutate(roundSIERA = round(SIERA)) %>%
  group_by(Role, roundSIERA) %>%
  summarise(n = n(),
            Stuff_plus = round(mean(Stuff_plus)),
            Location_plus = round(mean(Location_plus)),
            Pitching_plus = round(mean(Pitching_plus))) %>%
  filter(n > 10)
```

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

```
ggplot(sieraRole, aes(roundSIERA, Pitching_plus, color = Role)) +
  geom_line() +
  geom_point()
```

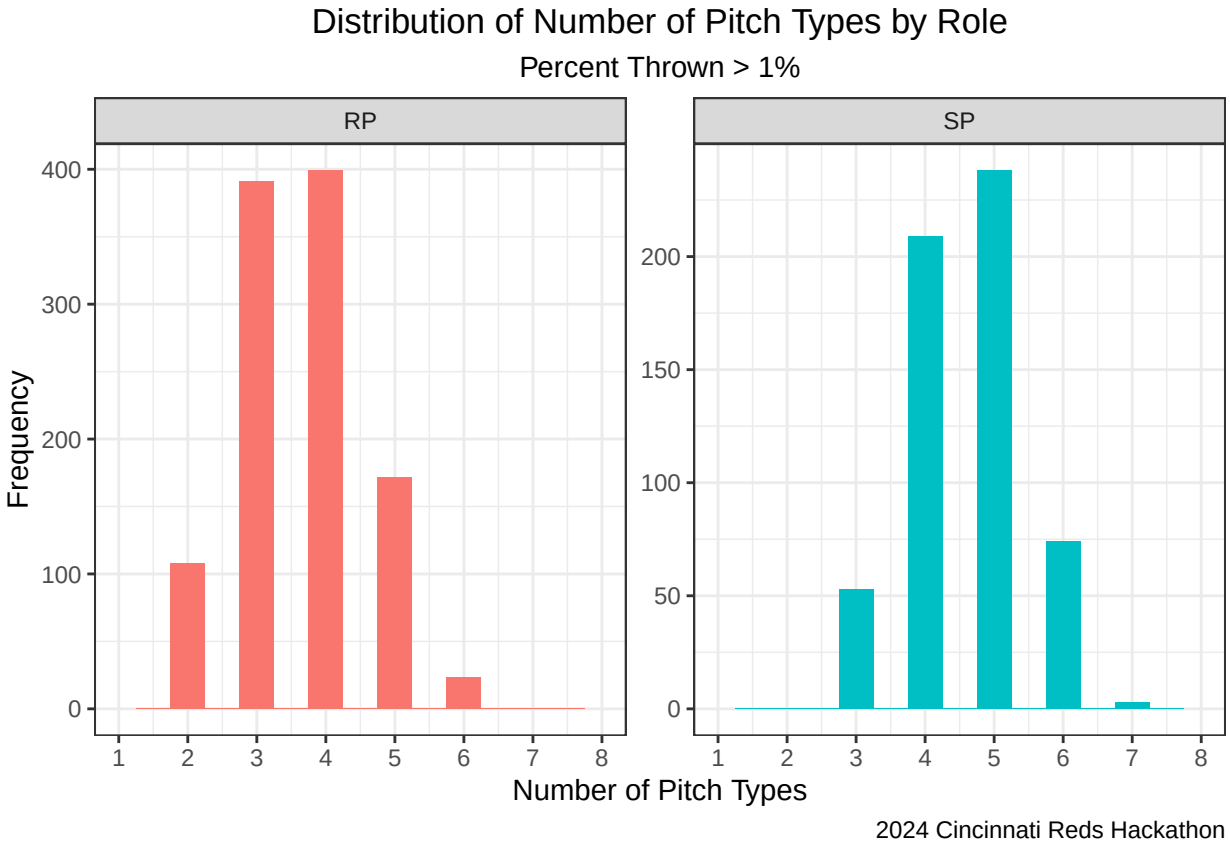


Adding Number of pitch types and number of great stuff

```
pitch_columns = select(fangraph2, FA_pct_sc:UN_pct_sc)

pitchTypeGr = function(row) {
  sum(ifelse(is.na(row), 0, ifelse(row > 0.01, 1, 0)))
}

pitch_sums = apply(pitch_columns, 1, pitchTypeGr)

fangraph4 = bind_cols(fangraph2, nPitchTypes = pitch_sums)

ggplot(fangraph4, aes(x = nPitchTypes, fill = Role)) +
  geom_histogram(binwidth = 0.5) +
  facet_wrap(~Role, scales = "free") +
  labs(title = "Distribution of Number of Pitch Types by Role",
```

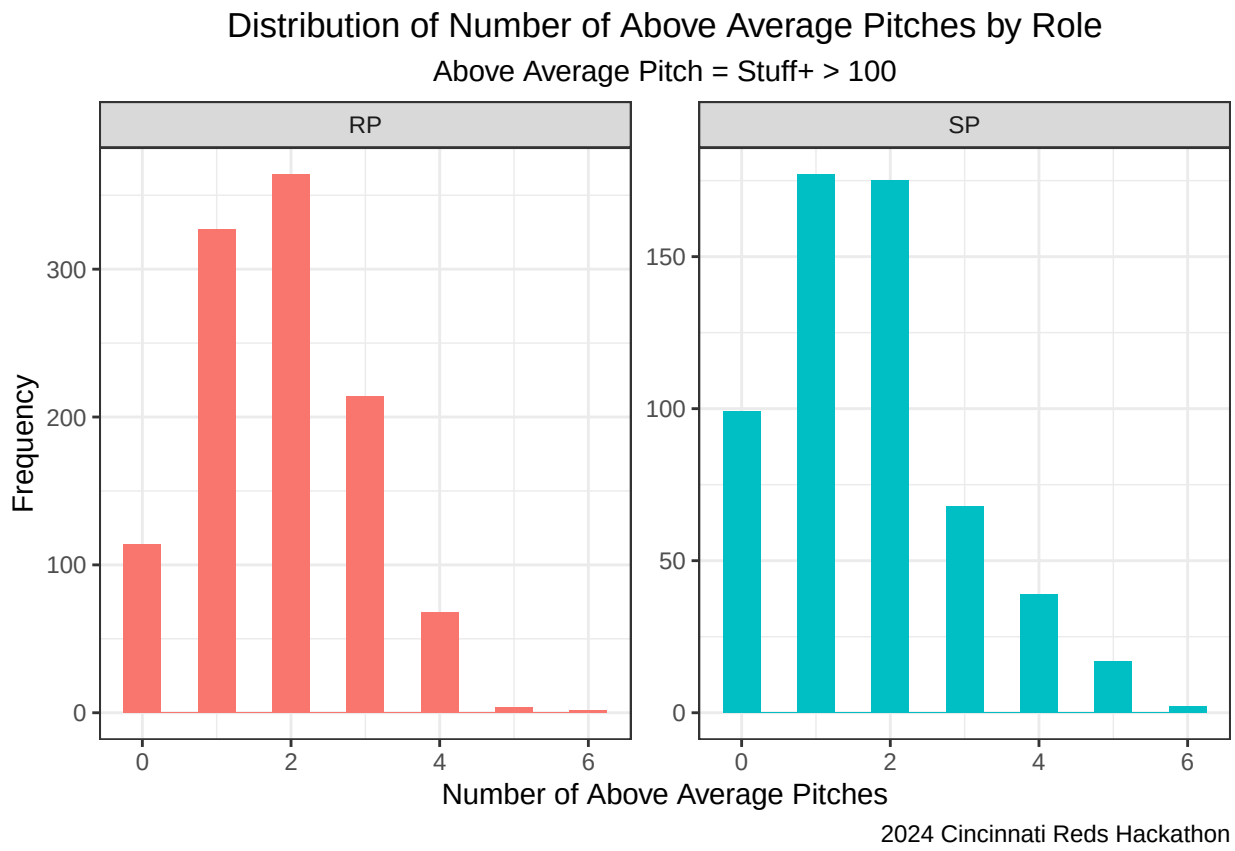
```
       x = "Number of Pitch Types", y = "Frequency",
       subtitle = "Percent Thrown > 1%",
       caption = "2024 Cincinnati Reds Hackathon") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = seq(1, 8, 1), limits = c(1,8))
```

```
## Warning: Removed 4 rows containing missing values (`geom_bar()`).
```




```
stuff_columns = select(fangraph2, starts_with("Stf_plus_"))

stuffTypeGr = function(row) {
  sum(ifelse(is.na(row), 0, ifelse(row >= 100, 1, 0)))
}

stuff_sums = apply(stuff_columns, 1, stuffTypeGr)

fangraph4 = bind_cols(fangraph4, nAboveAvgStuf = stuff_sums)

ggplot(fangraph4, aes(x = nAboveAvgStuf, fill = Role)) +
  geom_histogram(binwidth = 0.5) +
  facet_wrap(~Role, scales = "free") +
  labs(title = "Distribution of Number of Above Average Pitches by Role",
```

```
       x = "Number of Above Average Pitches", y = "Frequency",
       subtitle = "Above Average Pitch = Stuff+ > 100",
       caption = "2024 Cincinnati Reds Hackathon") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```



Looking at if more than 3 or 4 pitch types relates to performance

```
fangraph4 %>%
  mutate(more4PT = ifelse(nPitchTypes >= 4, 1, 0)) %>%
  group_by(Role, more4PT) %>%
  summarise(n = n(),
            SIERA = round(mean(SIERA),2),
            Barrel_pct = round(100 * mean(Barrel_pct), 1),
            HardHit_pct = round(100 * mean(Hard_pct), 1),
            CSW_pct = round(100 * mean(CSW_pct), 1),
            Stuff_plus = round(mean(Stuff_plus)),
            Location_plus = round(mean(Location_plus)),
            Pitching_plus = round(mean(Pitching_plus)))
```

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

```
## # A tibble: 4 x 10
## # Groups:   Role [2]
##   Role  more4PT     n SIERA Barrel_pct HardHit_pct CSW_pct Stuff_plus
##   <chr>   <dbl> <int> <dbl>      <dbl>       <dbl>   <dbl>      <dbl>
## 1 RP          0   499  3.8         7.5        31.1    28.7        106
## 2 RP          1   594  3.96        7          30.2    27.6        100
## 3 SP          0    53  4.05        8.3        32.5    28.1         98
## 4 SP          1   524  4.31        8.2        32.3    27           97
## # i 2 more variables: Location_plus <dbl>, Pitching_plus <dbl>
```

```
fangraph4 %>%
  mutate(more3ST = ifelse(nAboveAvgStuf >= 3, 1, 0)) %>%
  group_by(Role, more3ST) %>%
  summarise(n = n(),
            SIERA = round(mean(SIERA),2),
            Barrel_pct = round(100 * mean(Barrel_pct), 1),
            HardHit_pct = round(100 * mean(Hard_pct), 1),
            CSW_pct = round(100 * mean(CSW_pct), 1),
            Stuff_plus = round(mean(Stuff_plus)),
            Location_plus = round(mean(Location_plus)),
            Pitching_plus = round(mean(Pitching_plus)))
```

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

```
## # A tibble: 4 x 10
## # Groups:   Role [2]
##   Role  more3ST     n SIERA Barrel_pct HardHit_pct CSW_pct Stuff_plus
##   <chr>   <dbl> <int> <dbl>      <dbl>       <dbl>   <dbl>      <dbl>
## 1 RP          0   805  3.96        7.3        31      27.9         99
## 2 RP          1   288  3.68        7          29.7    28.7        114
## 3 SP          0   451  4.41        8.4        32.6    26.8         93
## 4 SP          1   126  3.85        7.6        31.1    28.4        110
## # i 2 more variables: Location_plus <dbl>, Pitching_plus <dbl>
```

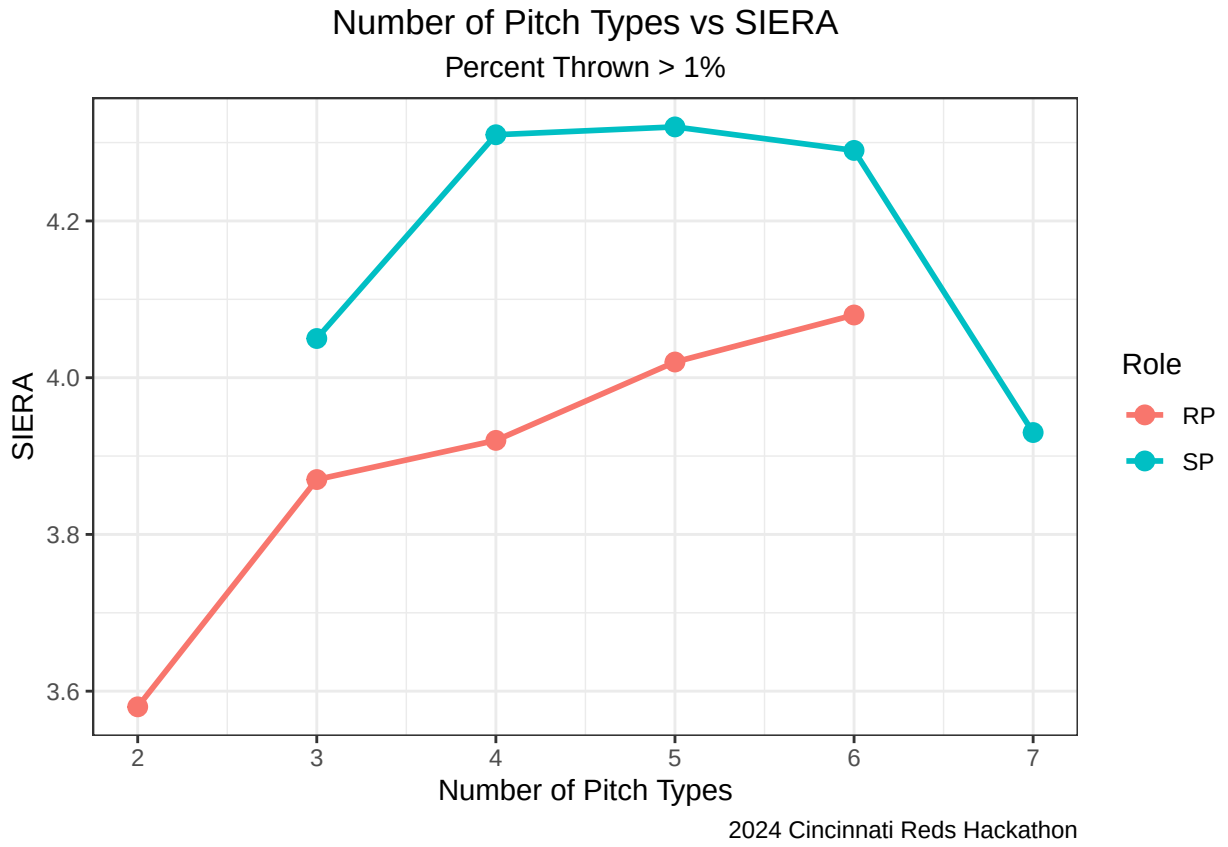
Number of pitch types vs SIERA

```
fangraph4 %>%
  group_by(Role, nPitchTypes) %>%
  summarise(G = sum(G),
            IP = round(sum(IP)),
            SIERA = round(mean(SIERA),2),
            Barrel_pct = round(100 * mean(Barrel_pct), 1),
            CSW_pct = round(100 * mean(CSW_pct), 1)) %>%
  ggplot(aes(nPitchTypes, SIERA, color = Role)) +
  geom_point(size = 3) +
  geom_line(size = 1) +
  labs(title = "Number of Pitch Types vs SIERA",
       x = "Number of Pitch Types",
       y = "SIERA",
       caption = "2024 Cincinnati Reds Hackathon",
```

```
       subtitle = "Percent Thrown > 1%") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

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

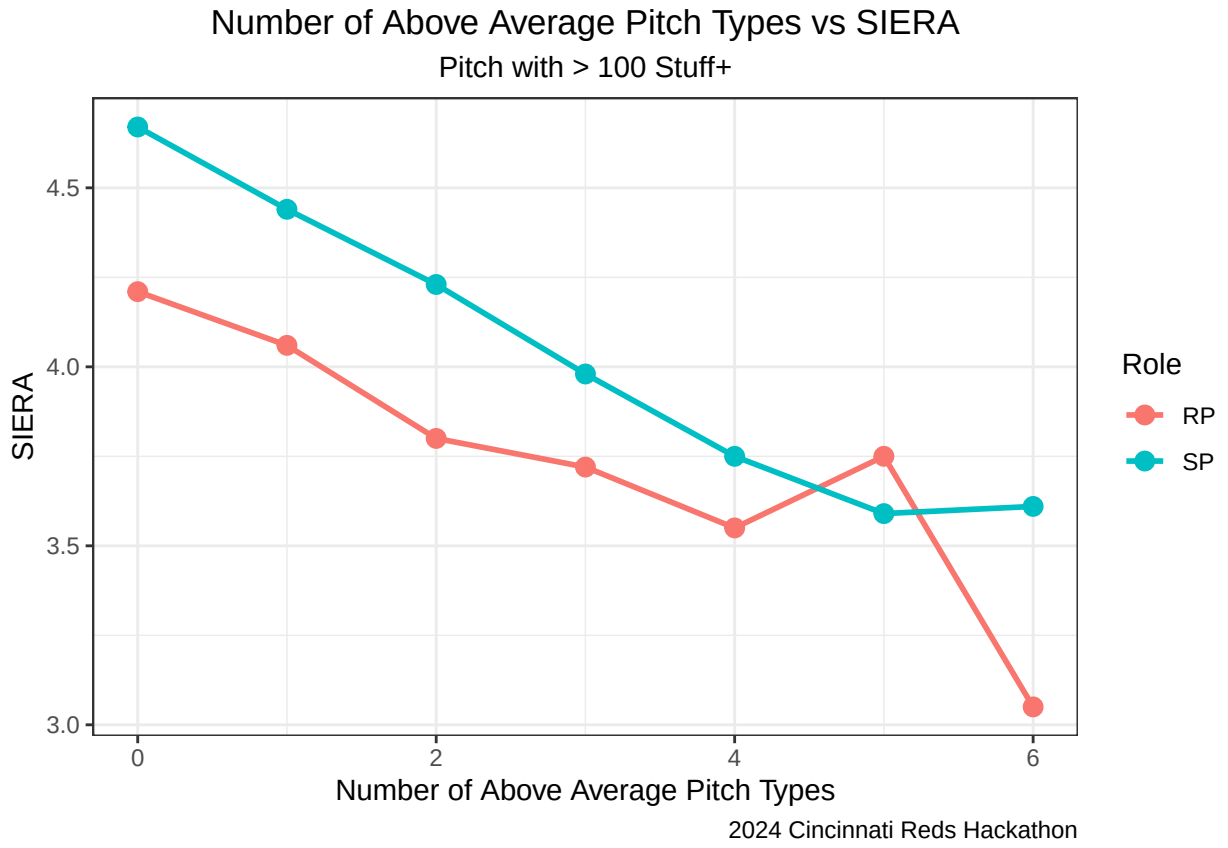
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
fangraph4 %>%
  group_by(Role, nAboveAvgStuf) %>%
  summarise(G = sum(G),
            IP = round(sum(IP)),
            SIERA = round(mean(SIERA),2),
            Barrel_pct = round(100 * mean(Barrel_pct), 1),
            CSW_pct = round(100 * mean(CSW_pct), 1)) %>%
  ggplot(aes(nAboveAvgStuf, SIERA, color = Role)) +
  geom_point(size = 3) +
  geom_line(size = 1) +
    labs(title = "Number of Above Average Pitch Types vs SIERA",
```

```
       x = "Number of Above Average Pitch Types",
       y = "SIERA",
       caption = "2024 Cincinnati Reds Hackathon",
       subtitle = "Pitch with > 100 Stuff+") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

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



Arsenal Similarity code

```
avgs = savant %>%
  group_by(player_name, pitch_type, game_year) %>%
  summarise(n = n(),
            velo = mean(release_speed, na.rm = T),
            pfx_x = mean(pfx_x, na.rm = T),
            pfx_z = mean(pfx_z, na.rm = T),
            pitcher = mean(pitcher)) %>%
  ungroup() %>%
  mutate(nameYear = paste0(player_name, "-", game_year))
```

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

```
pitchers = unique(avgs$nameYear)

pitcher_sim = matrix(NA, nrow = length(pitchers), ncol = 2)

normalize = function(x) {
  return((x - min(x, na.rm = T)) / (max(x, na.rm = T) - min(x, na.rm = T)))
}

normalized_avgs = as.data.frame(lapply(avgs[, c('velo', 'pfx_x', 'pfx_z')], normalize))

normalized_avgs = bind_cols(player_name = avgs$nameYear, normalized_avgs)

normalized_avgs = bind_cols(pitcher = avgs$pitcher, normalized_avgs)

for (i in 1:length(pitchers)) {

  selected_pitcher = pitchers[i]

  pitcher_subset = filter(normalized_avgs, player_name == selected_pitcher)

  feature_vectors_pitcher = pitcher_subset[, c('velo', 'pfx_x', 'pfx_z')]

  euclidean_distance_pitcher = as.matrix(dist(feature_vectors_pitcher, method = 'euclidean'))

  avg_dist = mean(euclidean_distance_pitcher)

  pitcher_sim[i,1] = selected_pitcher
  pitcher_sim[i,2] = avg_dist

}

pitchers2 = savant %>%
  mutate(nameYear = paste0(player_name, "-", game_year)) %>%
  group_by(nameYear) %>%
  summarise(n = n(),
            pitcher = mean(pitcher)) %>%
  arrange(nameYear)

pitcher_sim = as.data.frame(pitcher_sim) %>%
  arrange(V1)

pitcher_sim2 = bind_cols(pitchers2, similarity = pitcher_sim$V2)

library(stringr)

pitcher_sim3 = pitcher_sim2

pitcher_sim3$Year = NA
pitcher_sim3$player_name = NA

for (i in 1:nrow(pitcher_sim3)) {
```

```
  pitcher_sim3$Year[i] = unlist(str_split(pitcher_sim3$nameYear[i], "-"))[2]

  Name = unlist(str_split(pitcher_sim3$nameYear[i], "-"))[1]

  pitcher_sim3$player_name[i] = paste0(unlist(str_split(Name, ", "))[2], " ", unlist(str_split(Name, ",

}

pitcher_sim3$Year = as.numeric(pitcher_sim3$Year)
```

```
## Warning: NAs introduced by coercion
```

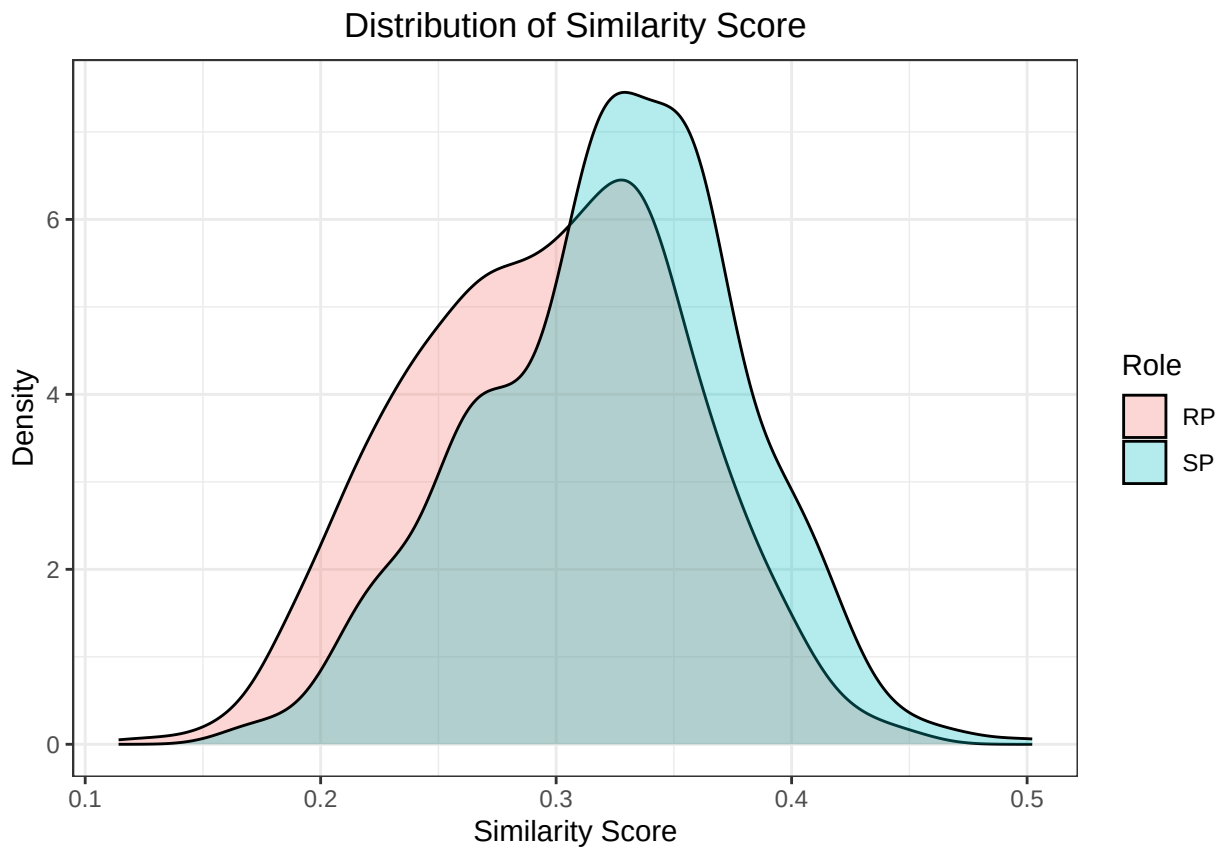
```
fangraph5 = inner_join(fangraph4, pitcher_sim3, by = c("MLBAMID" = "pitcher", "Season" = "Year"))
```

```
fangraph5$similarity = as.numeric(fangraph5$similarity)

fangraph5 %>%
  group_by(Role) %>%
  summarise(sim = mean(similarity, na.rm = T))
```

```
## # A tibble: 2 x 2
##   Role    sim
##   <chr> <dbl>
## 1 RP    0.296
## 2 SP    0.324
```

```
ggplot(fangraph5, aes(similarity, fill = Role)) +
  geom_density(alpha = 0.3) +
  labs(title = "Distribution of Similarity Score",
       x = "Similarity Score",
       y = "Density") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 69 rows containing non-finite values (‘stat_density()‘).
```

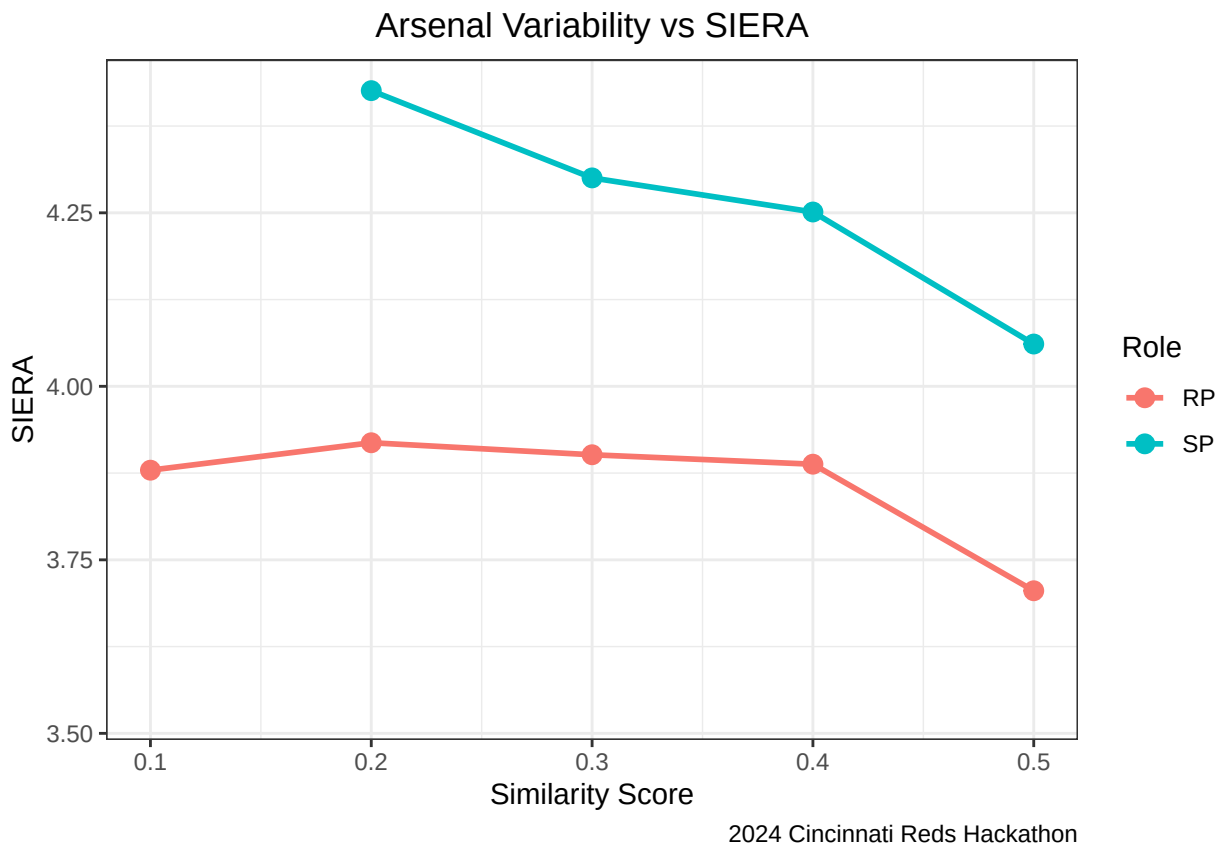



```
fangraph5 %>%
  mutate(roundSim = round(similarity, 1)) %>%
  group_by(Role, roundSim) %>%
  summarise(SIERA = mean(SIERA)) %>%
  ggplot(aes(roundSim, SIERA, color = Role)) +
  geom_point(size = 3) +
  geom_line(size = 1) +
  labs(title = "Arsenal Variability vs SIERA",
       x = "Similarity Score",
       caption = "2024 Cincinnati Reds Hackathon") +
  theme(plot.title = element_text(hjust = 0.5))
```

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

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```



TTO/Sustainability

```
unique(savant$pitch_type)
```

```
##  [1] "FF" "SL" "CU" "SI" "CH" "FS" "KC" "FC" "SV" "ST" "FA" "CS" "PO" "EP" ""
## [16] "SC" "KN" "FO"
```

```
savant %>%
  group_by(role_key, times_faced) %>%
  summarise(n = n(),
            velo = mean(release_speed[pitch_type %in% c("FF", "SI", "FS", "FC", "FA")], na.rm = T),
            xwOBA = mean(estimated_woba_using_speedangle, na.rm = T))
```

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

```
## # A tibble: 9 x 5
## # Groups:   role_key [2]
##   role_key times_faced      n  velo xwOBA
##   <chr>          <int>  <int> <dbl> <dbl>
## 1 RP                 1 870045  93.5 0.357
## 2 RP                 2  35106  91.7 0.365
```

```
## 3 RP                3   3017  90.7 0.373
## 4 RP                4     51  89.4 0.531
## 5 SP                1 519054  92.6 0.368
## 6 SP                2 463286  92.2 0.373
## 7 SP                3 238582  92.1 0.387
## 8 SP                4   7162  92.5 0.376
## 9 SP                5     34  92.7 0.312
```

Modeling

Initial Splitting

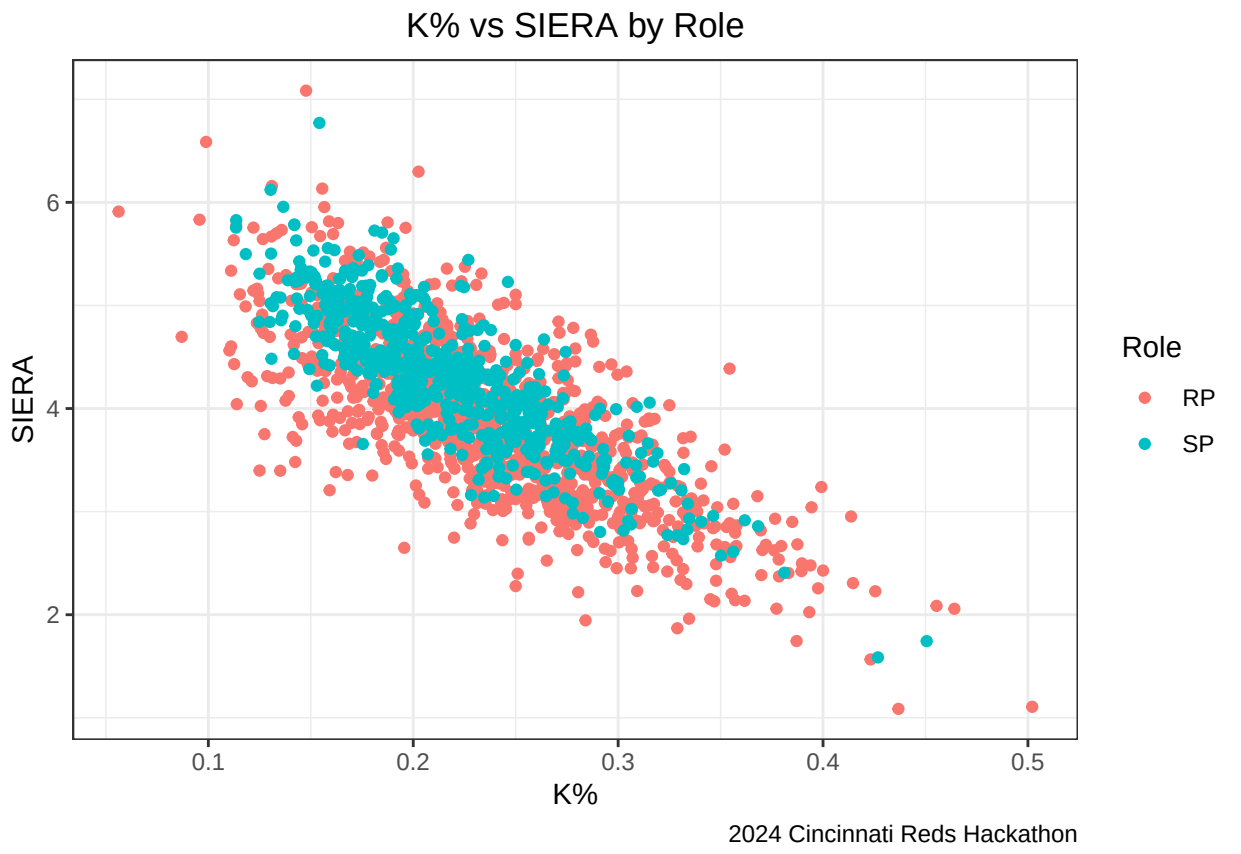
```
modBegin = fangraph5 %>%
  select(Name, Role, SIERA, K_pct, BB_pct,
         CSW_pct, OSwing_pct, OContact_pct, ZSwing_pct, ZContact_pct, Hard_pct, Barrel_pct,
         EV, Stuff_plus:Pitching_plus, nPitchTypes, nAboveAvgStuf, similarity)

modBegin$nPitchTypes = as.factor(modBegin$nPitchTypes)
modBegin$nAboveAvgStuf = as.factor(modBegin$nAboveAvgStuf)


starters = filter(modBegin, Role == "SP") %>%
  select(-Role)

relievers = filter(modBegin, Role == "RP") %>%
  select(-Role)


library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.1.1 --

## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tibble       3.2.1
## v infer        1.0.5     v tidyr        1.3.0
## v modeldata    1.2.0     v tune         1.1.2
## v parsnip      1.1.1     v workflows    1.1.3
## v purrr        1.0.2     v workflowsets 1.0.1
## v recipes      1.0.8     v yardstick    1.2.0

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x recipes::fixed() masks stringr::fixed()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

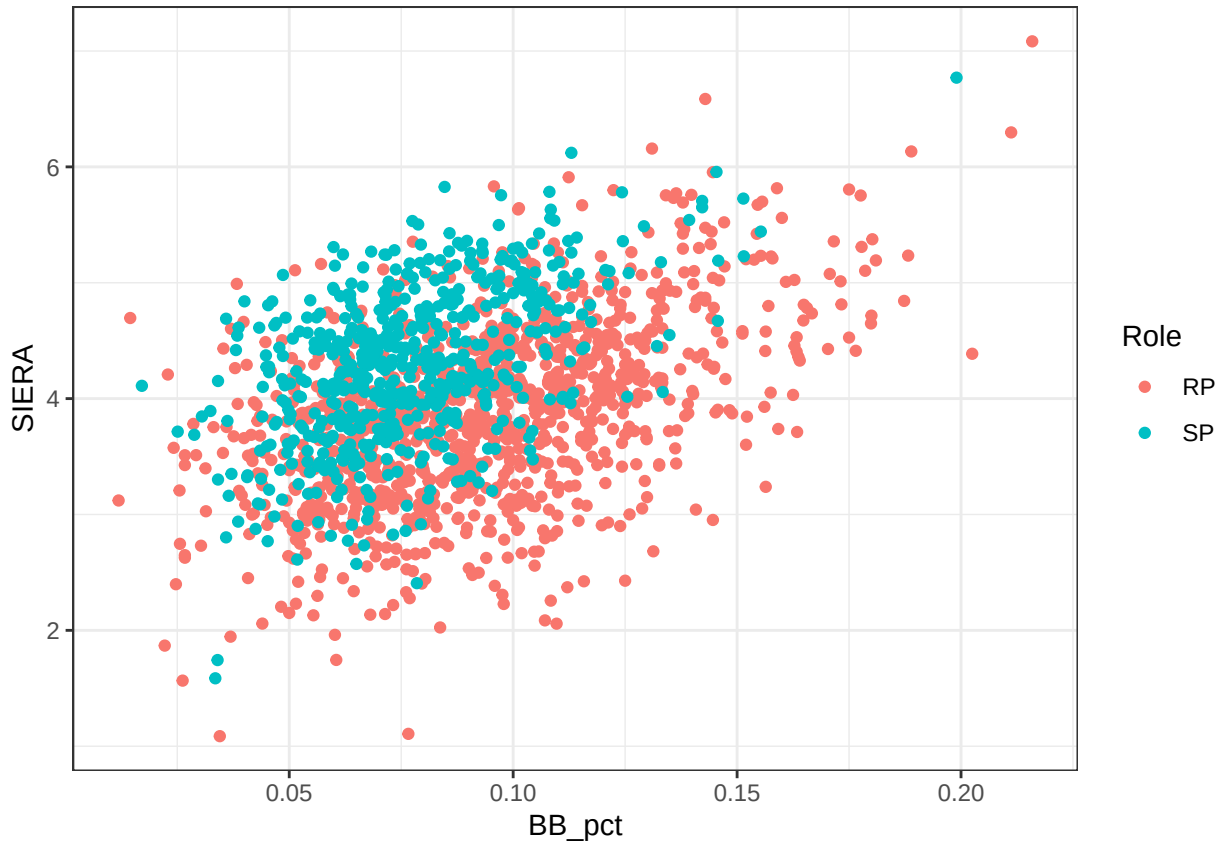
```
set.seed(123)
splitSP = initial_split(starters[-1], prop = 0.7, strata = SIERA)

set.seed(123)
splitRP = initial_split(relievers[-1], prop = 0.7, strata = SIERA)

trainSP = training(splitSP)
testSP = testing(splitSP)

trainRP = training(splitRP)
testRP = testing(splitRP)

set.seed(123)
sp_folds = vfold_cv(trainSP, strata = SIERA, v = 4)

set.seed(123)
rp_folds = vfold_cv(trainRP, strata = SIERA, v = 4)
```

EDA

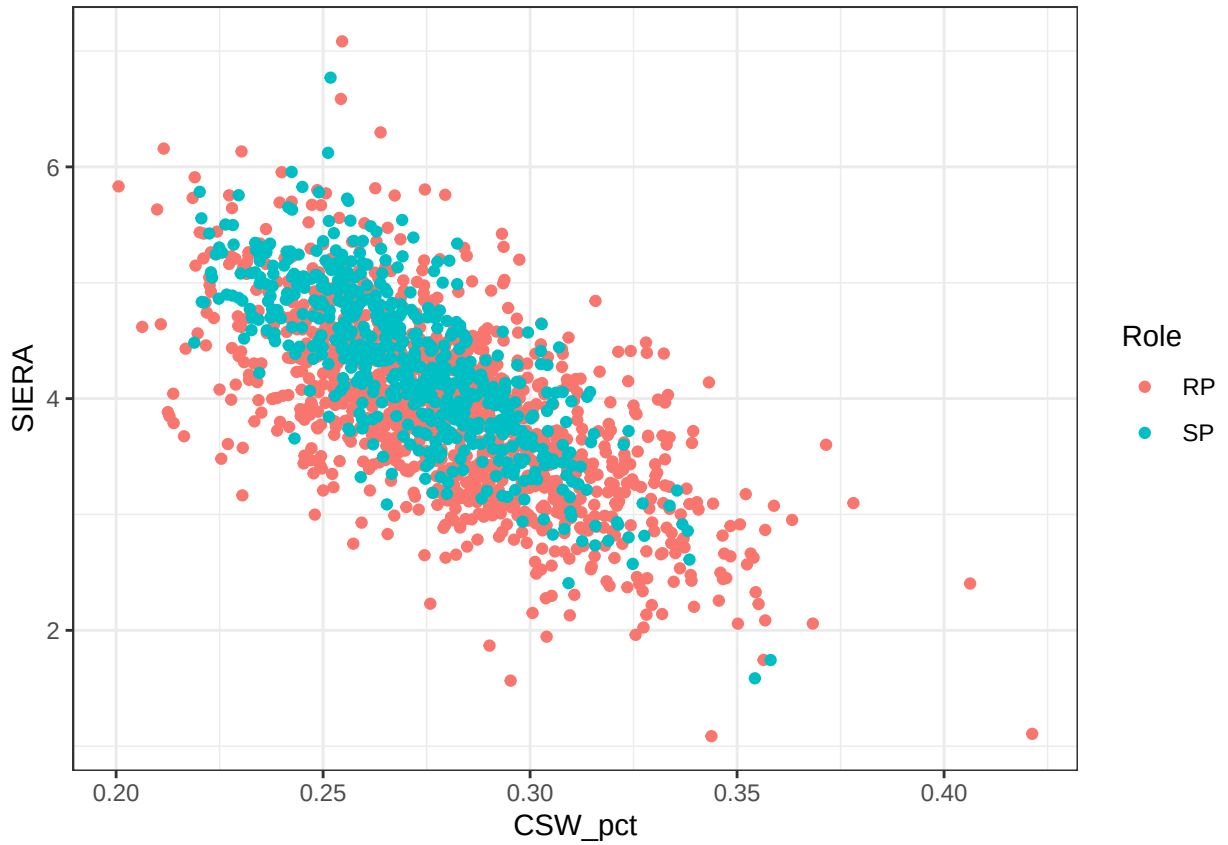
```
ggplot(modBegin, aes(K_pct, SIERA, color = Role)) +
  geom_point() +
  labs(title = "K% vs SIERA by Role", x = "K%", caption = "2024 Cincinnati Reds Hackathon") +
  theme(plot.title = element_text(hjust = 0.5))
```



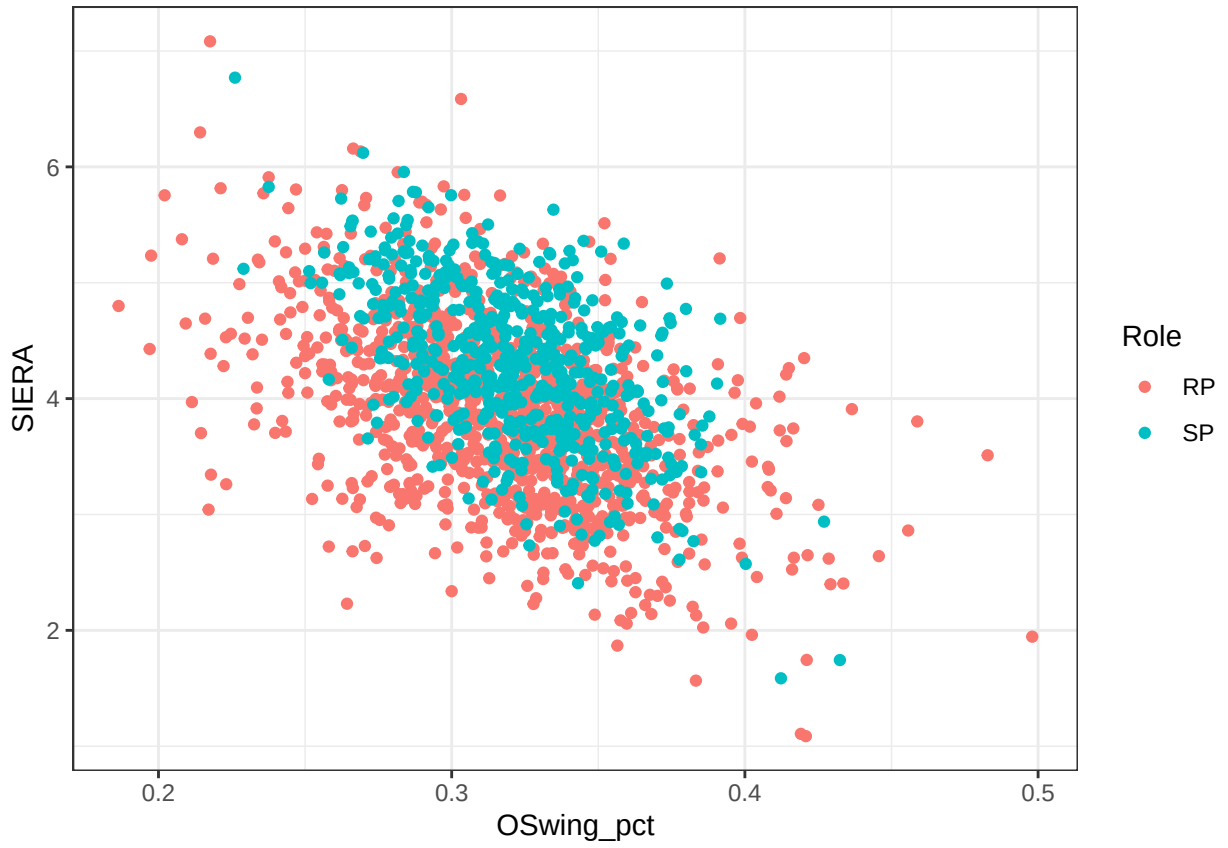

```
ggplot(modBegin, aes(BB_pct, SIERA, color = Role)) +
  geom_point()
```



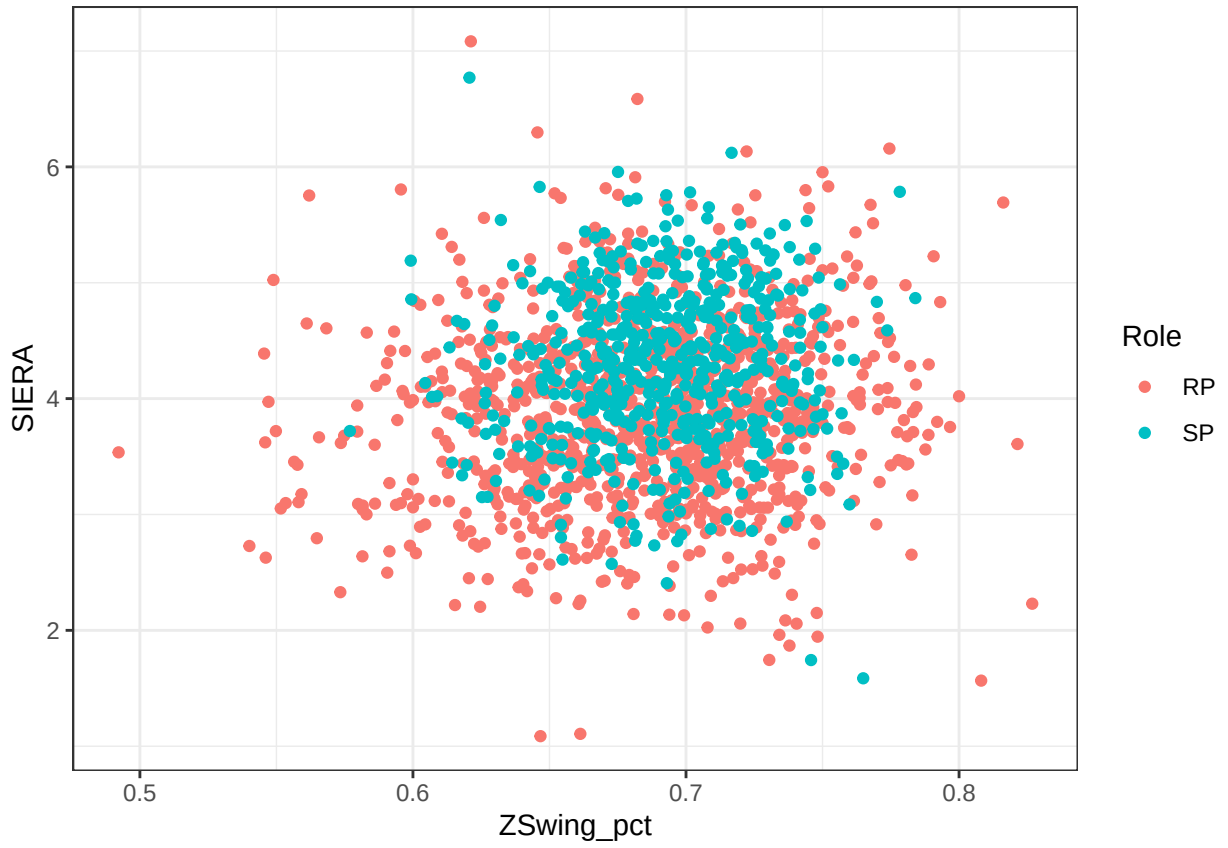

```
ggplot(modBegin, aes(CSW_pct, SIERA, color = Role)) +
  geom_point()
```



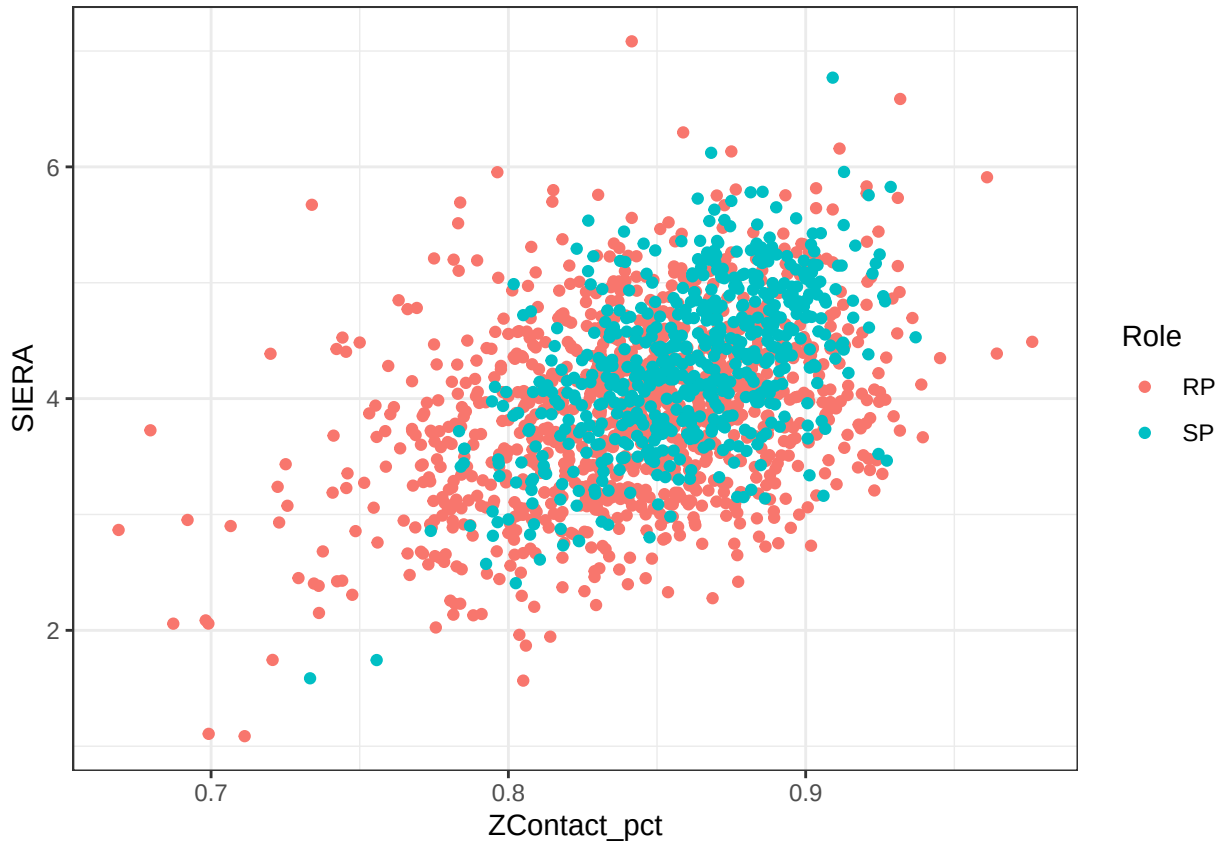

```
ggplot(modBegin, aes(OSwing_pct, SIERA, color = Role)) +
  geom_point()
```



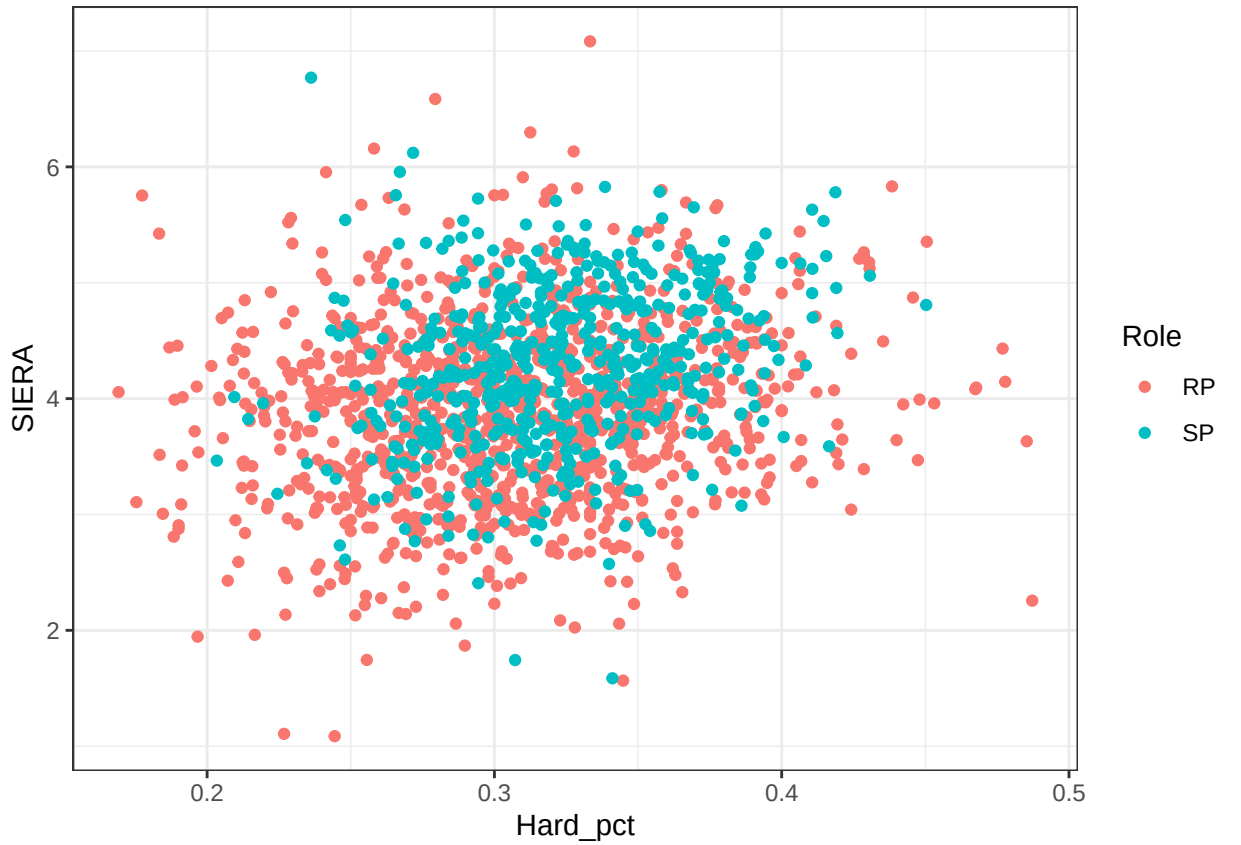

```
ggplot(modBegin, aes(ZSwing_pct, SIERA, color = Role)) +
  geom_point()
```



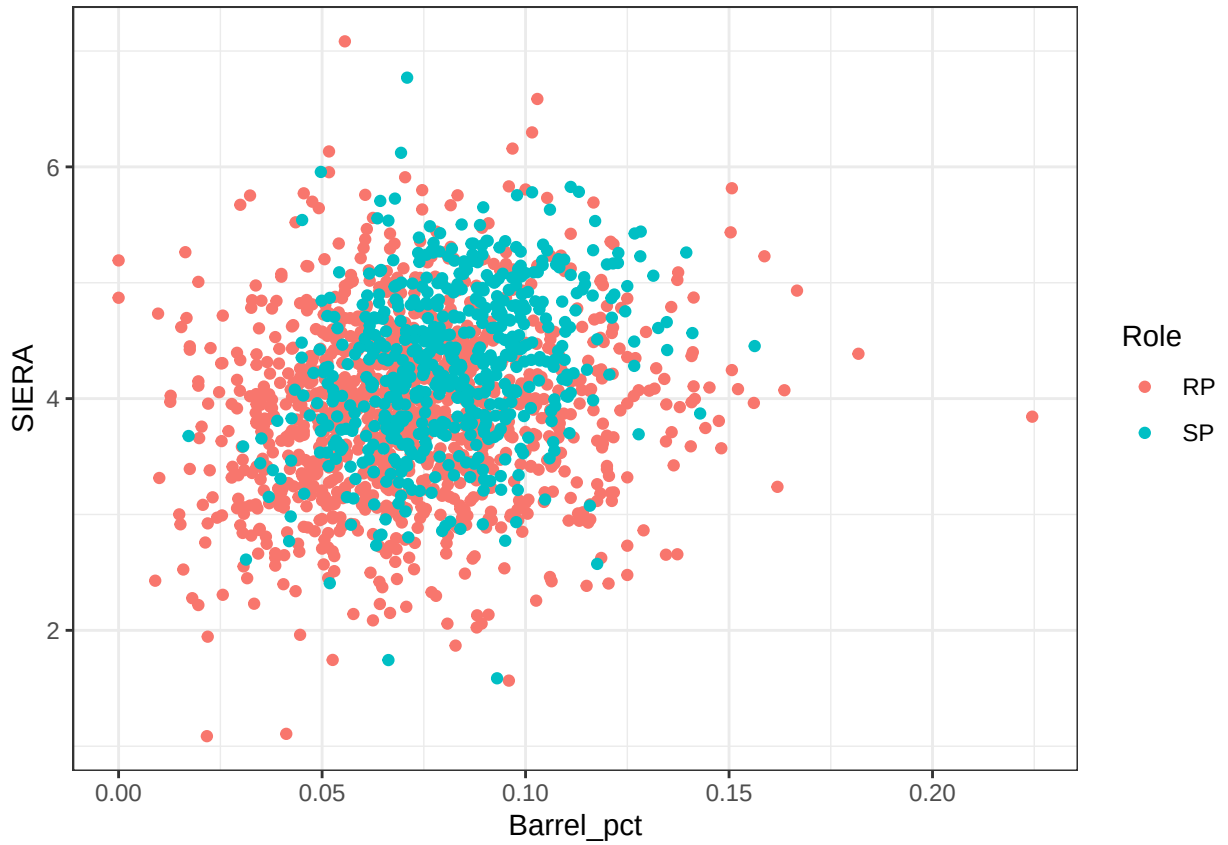

```
ggplot(modBegin, aes(ZContact_pct, SIERA, color = Role)) +
  geom_point()
```



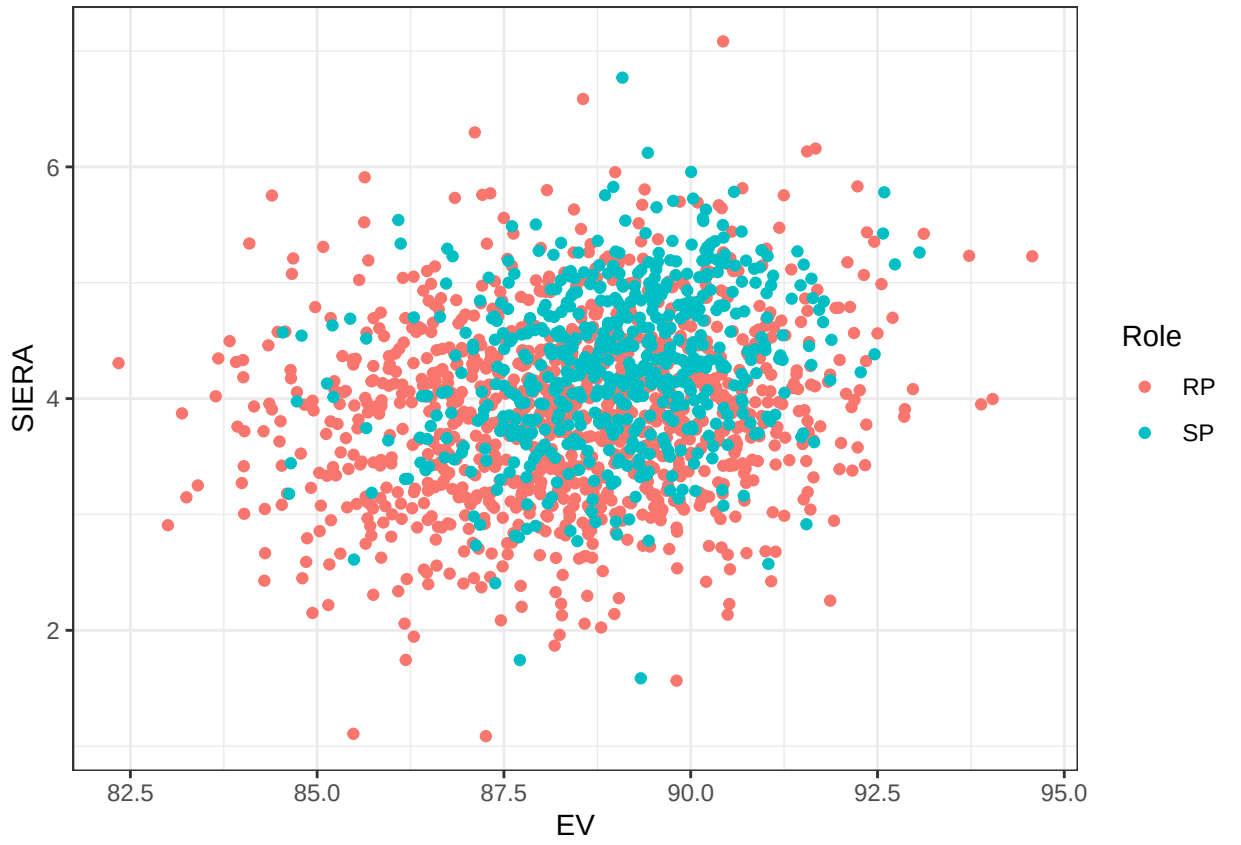

```
ggplot(modBegin, aes(Hard_pct, SIERA, color = Role)) +
  geom_point()
```



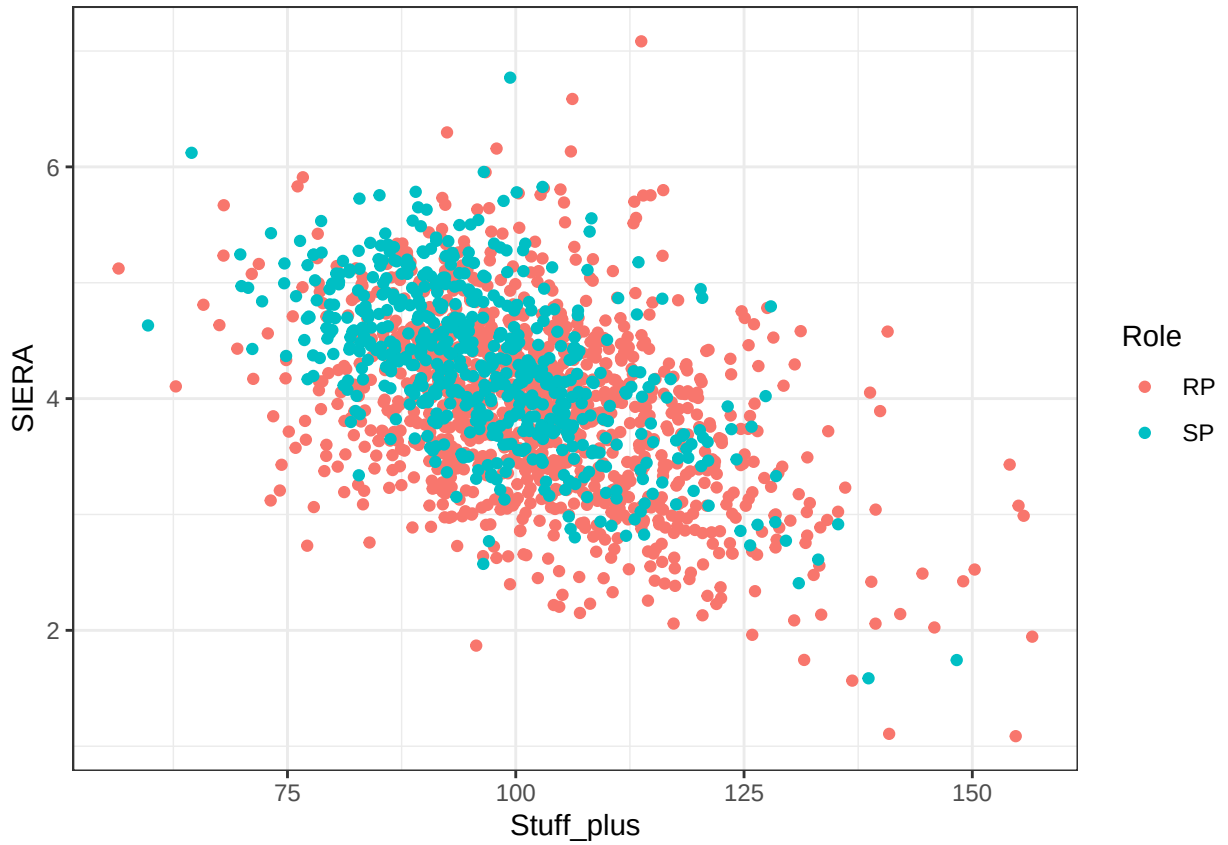

```
ggplot(modBegin, aes(Barrel_pct, SIERA, color = Role)) +
  geom_point()
```




```
ggplot(modBegin, aes(EV, SIERA, color = Role)) +
  geom_point()
```



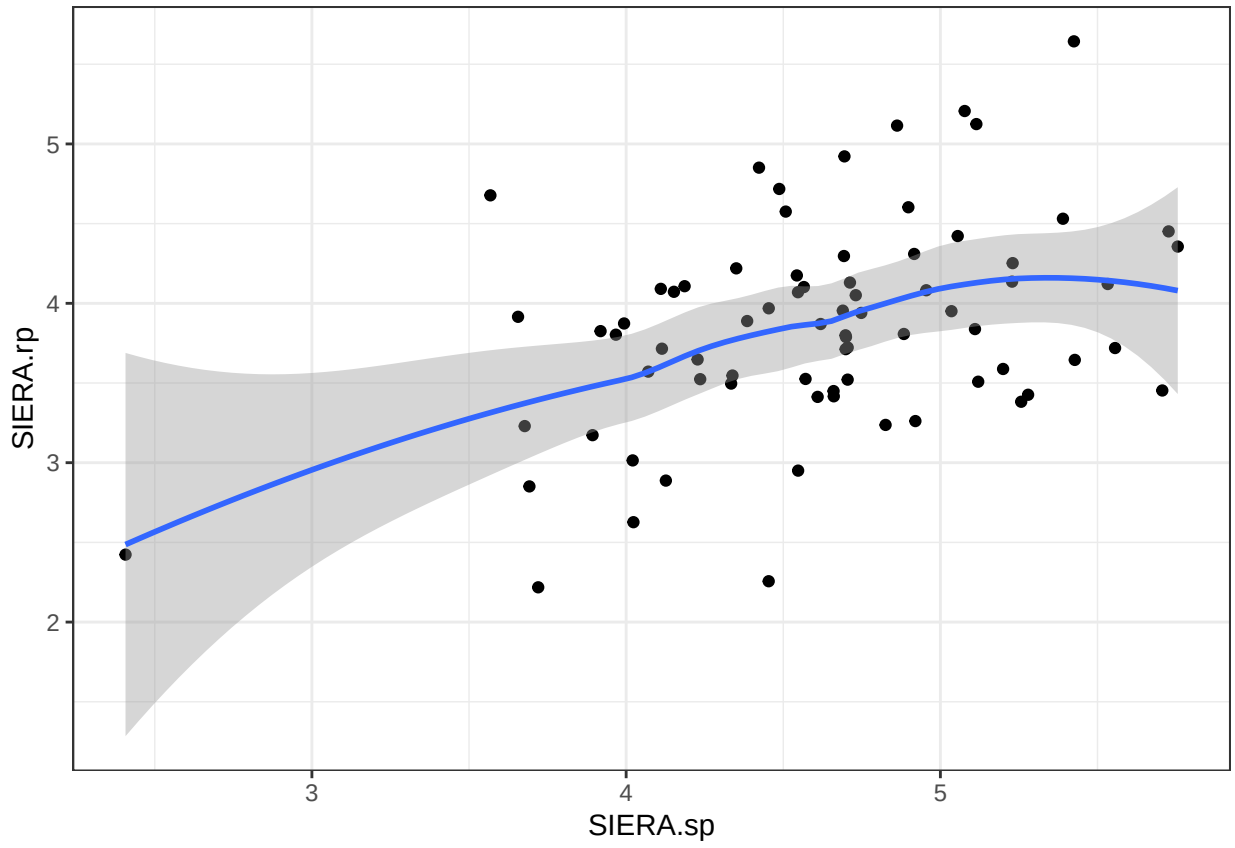

```
ggplot(modBegin, aes(Stuff_plus, SIERA, color = Role)) +
  geom_point()
```



Stat Adjustments

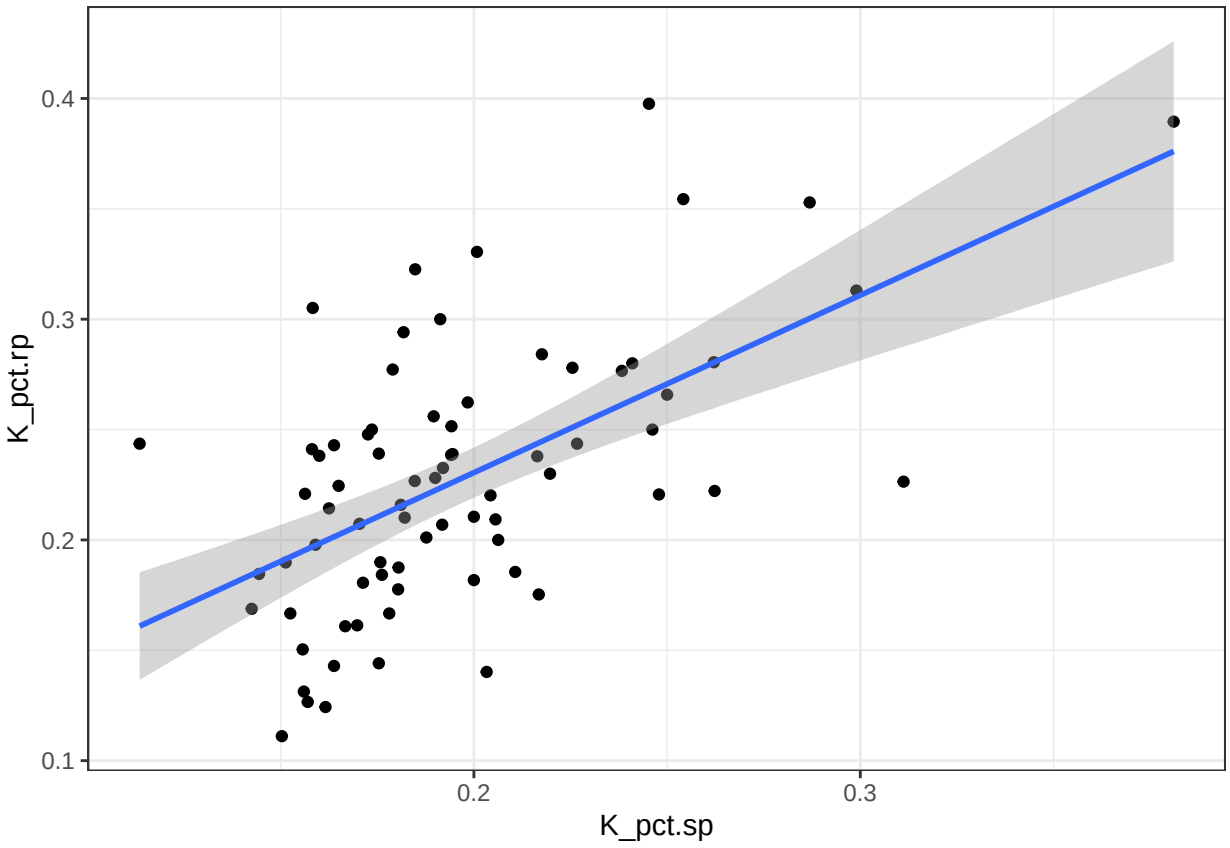
```
start2 = filter(fangraph5, Role == "SP")
relie2 = filter(fangraph5, Role == "RP")

comb = inner_join(start2, relie2, by = c("PlayerId", "MLBAMID", "Name", "Season", "Throws", "Age"), suf

ggplot(comb, aes(SIERA.sp, SIERA.rp)) +
  geom_point() +
  stat_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



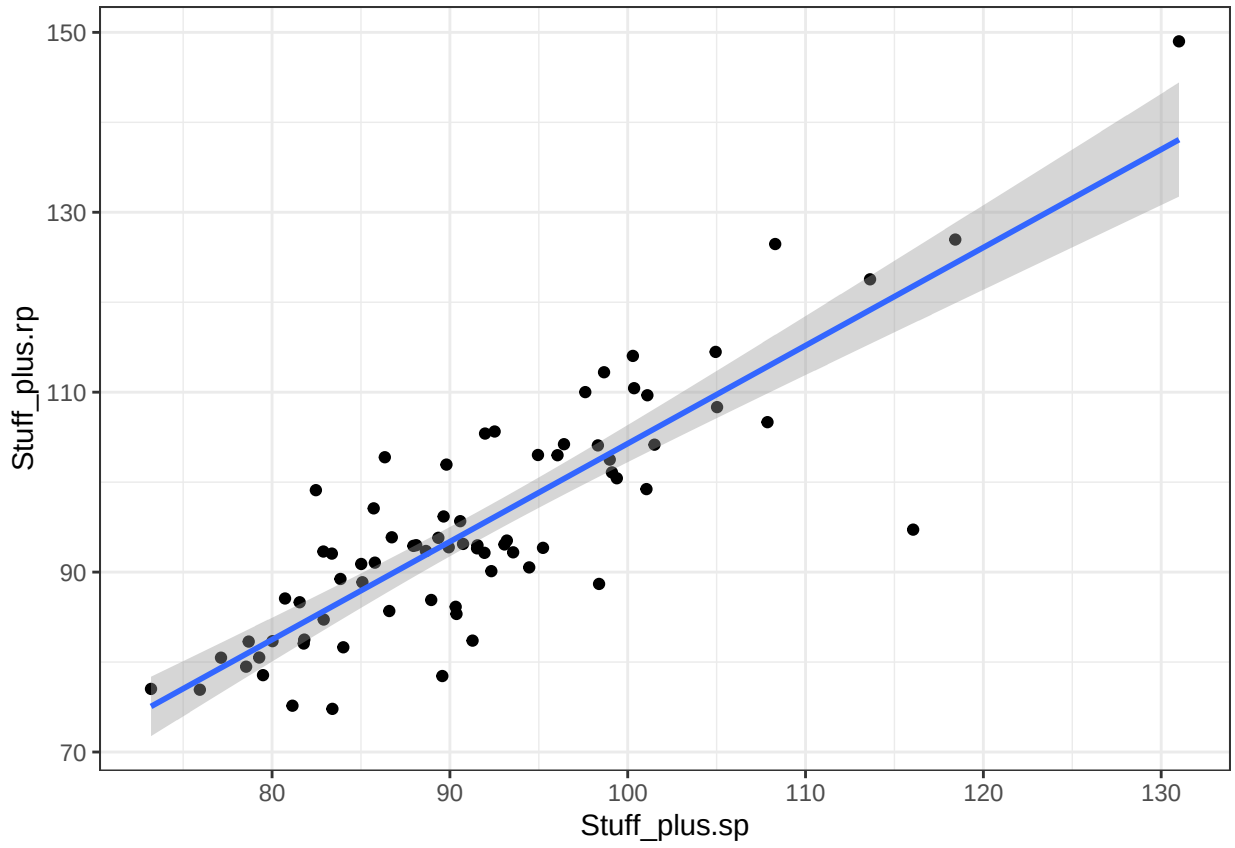

```
ggplot(comb, aes(K_pct.sp, K_pct.rp)) +
  geom_point() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



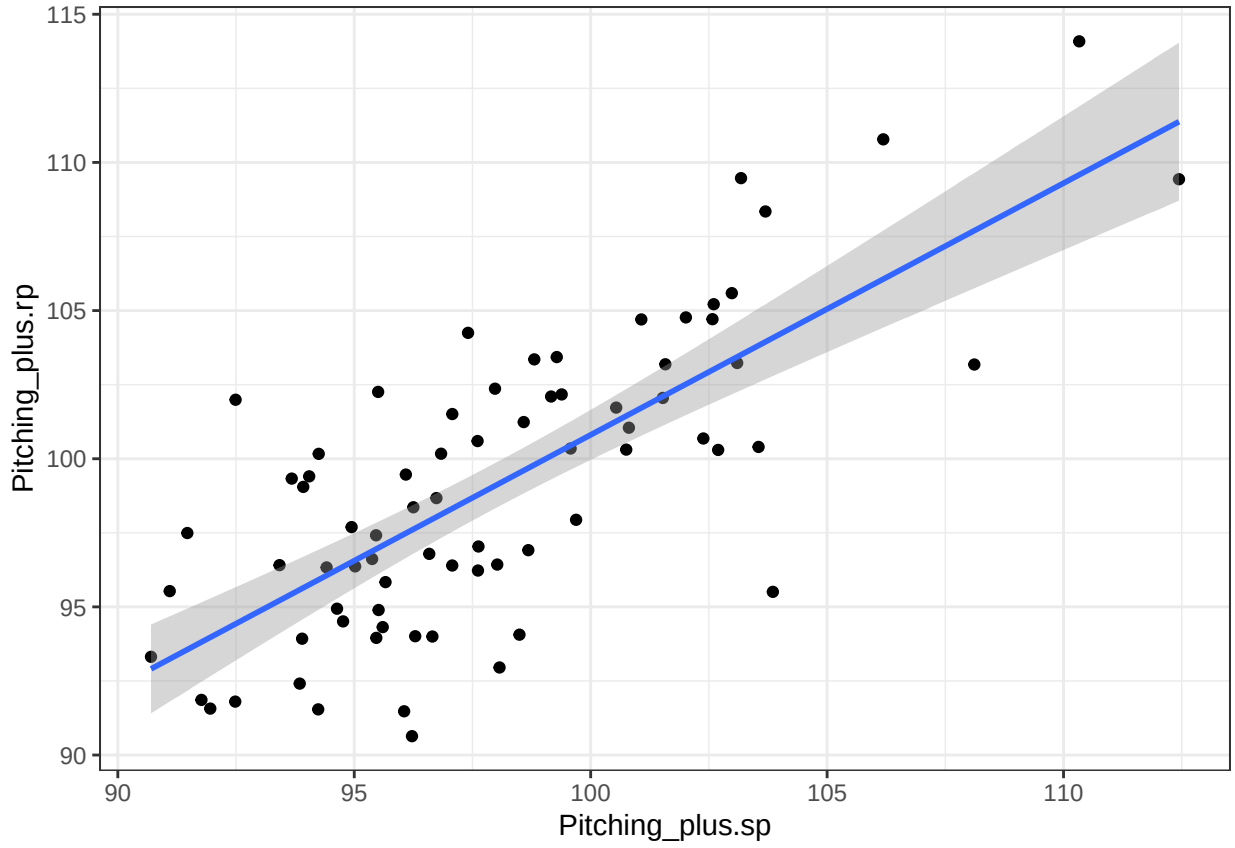

```
ggplot(comb, aes(Stuff_plus.sp, Stuff_plus.rp)) +
  geom_point() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



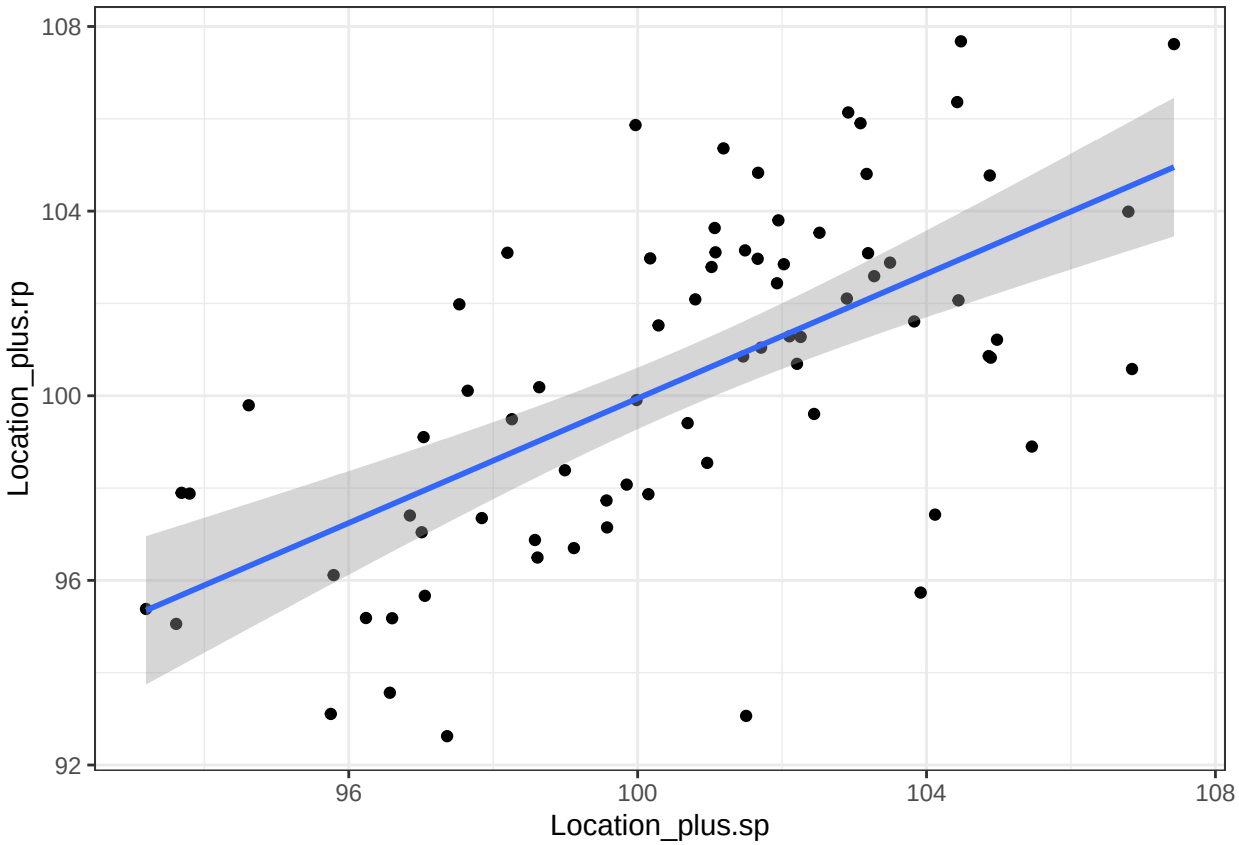

```
ggplot(comb, aes(Pitching_plus.sp, Pitching_plus.rp)) +
  geom_point() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



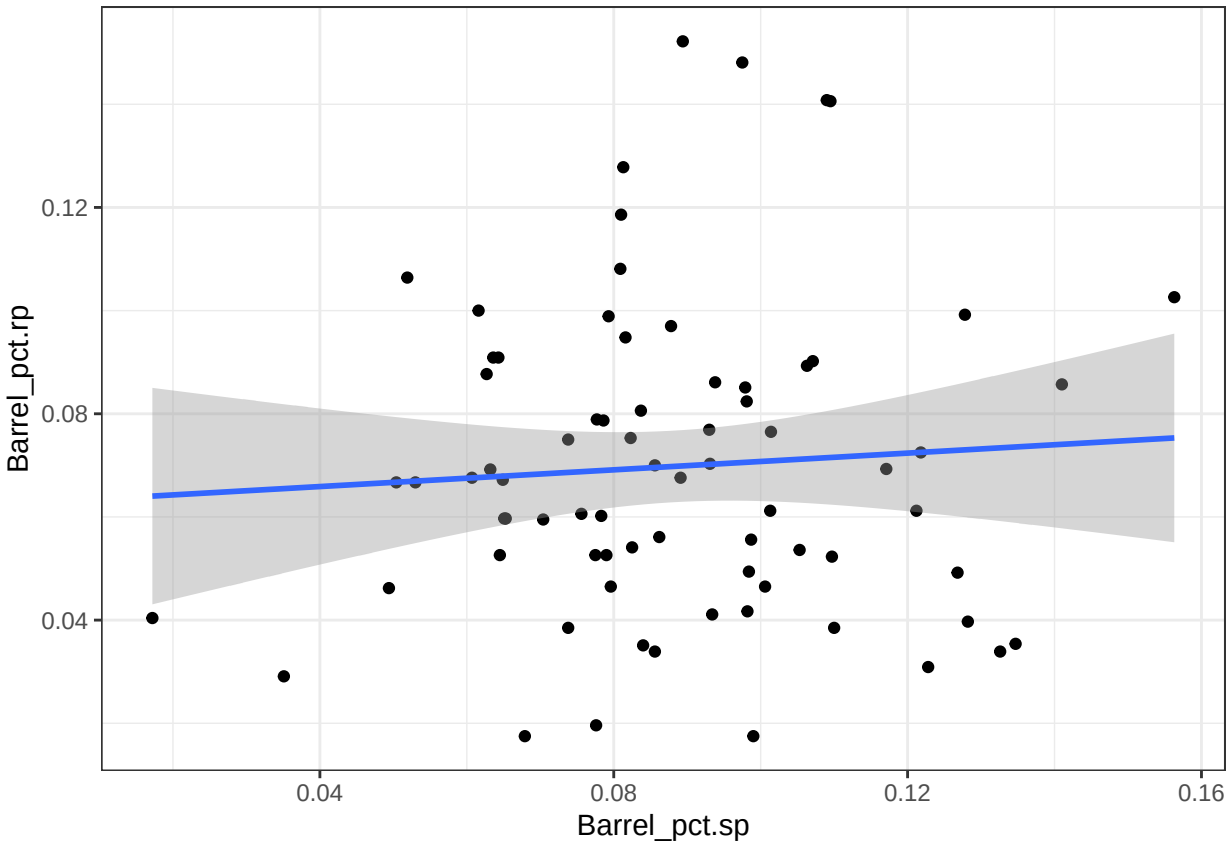

```
ggplot(comb, aes(Location_plus.sp, Location_plus.rp)) +
  geom_point() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



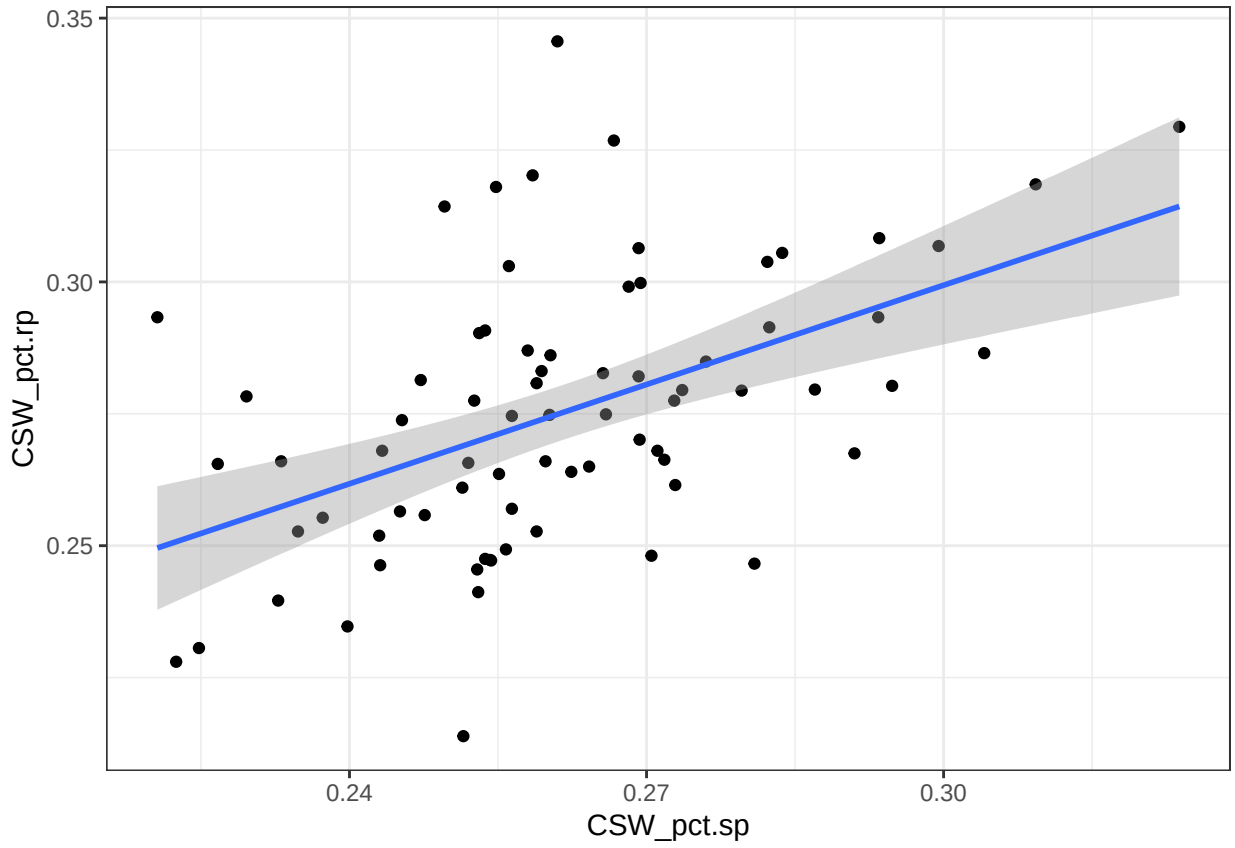

```
ggplot(comb, aes(Barrel_pct.sp, Barrel_pct.rp)) +
  geom_point() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



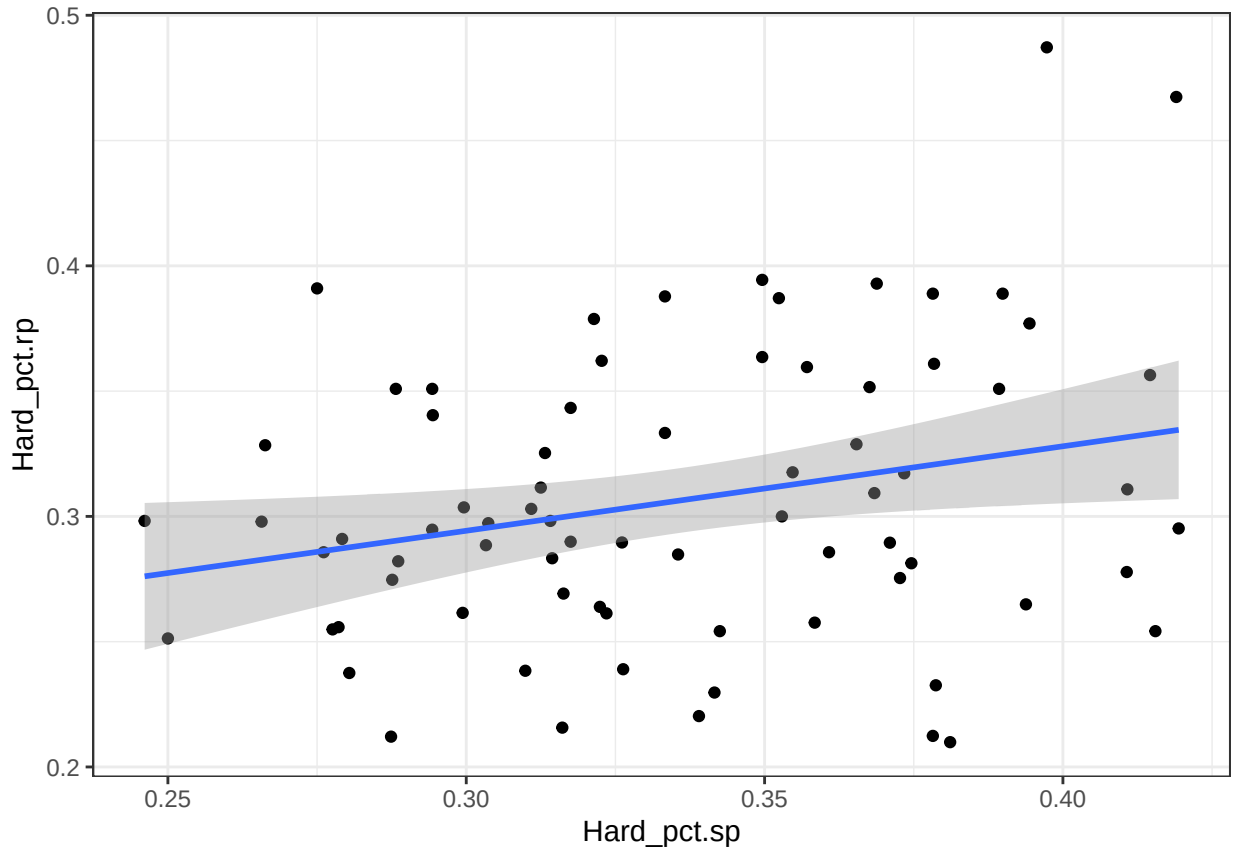

```
ggplot(comb, aes(CSW_pct.sp, CSW_pct.rp)) +
  geom_point() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(comb, aes(Hard_pct.sp, Hard_pct.rp)) +
  geom_point() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
comb2 = comb %>%
  select(Name, SIERA.sp, K_pct.sp, BB_pct.sp, CSW_pct.sp, OSwing_pct.sp, OContact_pct.sp, ZSwing_pct.sp


start_vars = colnames(comb2)[3:15]
reliev_vars = colnames(comb2)[20:32]

models = list()

set.seed(123)
combSplit = initial_split(comb2, prop = 0.8)

combTrain = training(combSplit)
combTest = testing(combSplit)

testRes = matrix(NA, nrow = 2, ncol = length(start_vars)+1)
colnames(testRes) = c("Role", start_vars)

testRes[1,1] = "SP"

for (i in 1:length(start_vars)) {

  start_var = start_vars[i]
  reliev_var = reliev_vars[i]

  model = lm(paste(reliev_var, "~", start_var), data = combTrain)
```

```
  preds = predict(model, combTest)

  testRes[1,1+i] = rmse_vec(combTest[, 19+i], preds)

  models[[reliev_var]] = model

}



SPadj = cbind(Name = starters$Name, data.frame(matrix(NA, nrow = nrow(starters), ncol = 13)))


colnames(starters) = paste0(colnames(starters), ".sp")
colnames(SPadj)[2:14] = colnames(starters)[3:15]

for (i in 1:length(start_vars)) {
  start_var = start_vars[i]
  reliev_var = reliev_vars[i]

  SPadj[[start_var]] = predict(models[[reliev_var]], newdata = starters)
}

SPadj = cbind(SPadj, starters[16:18])

colnames(SPadj) = str_remove(colnames(SPadj), paste0(".sp", "$"))







models = list()

testRes[2,1] = "RP"

for (i in 1:length(reliev_vars)) {

  start_var = start_vars[i]
  reliev_var = reliev_vars[i]

  model = lm(paste(start_var, "~", reliev_var), data = combTrain)
  preds = predict(model, combTest)

  testRes[2,1+i] = rmse_vec(combTest[, 19+i], preds)

  models[[start_var]] = model

}
```

```
RPadj = cbind(Name = relievers$Name, data.frame(matrix(NA, nrow = nrow(relievers), ncol = 13)))


colnames(relievers) = paste0(colnames(relievers), ".rp")
colnames(RPadj)[2:14] = colnames(relievers)[3:15]



for (i in 1:length(reliev_vars)) {
  start_var = start_vars[i]
  reliev_var = reliev_vars[i]

  RPadj[[reliev_var]] <- predict(models[[start_var]], newdata = relievers)
}

RPadj = cbind(RPadj, relievers[16:18])

colnames(RPadj) = str_remove(colnames(RPadj), paste0(".rp", "$"))
```

Modeling Starters

```
sp_rec =
  recipe(SIERA ~ ., data = trainSP) %>%
  step_dummy(all_nominal_predictors())

sp_rec %>%
  prep() %>%
  bake(new_data = NULL) %>%
  head()
```

```
## # A tibble: 6 x 26
##   K_pct BB_pct CSW_pct OSwing_pct OContact_pct ZSwing_pct ZContact_pct Hard_pct
##   <dbl>  <dbl>   <dbl>      <dbl>        <dbl>      <dbl>        <dbl>    <dbl>
## 1 0.286 0.0767   0.310      0.313        0.545      0.646        0.846    0.288
## 2 0.255 0.0458   0.299      0.338        0.595      0.71         0.840    0.271
## 3 0.451 0.034    0.358      0.432        0.451      0.746        0.756    0.307
## 4 0.317 0.0856   0.304      0.314        0.551      0.651        0.838    0.326
## 5 0.273 0.0738   0.315      0.341        0.553      0.627        0.852    0.314
## 6 0.292 0.0646   0.303      0.300        0.564      0.677        0.839    0.302
## # i 18 more variables: Barrel_pct <dbl>, EV <dbl>, Stuff_plus <dbl>,
## #   Location_plus <dbl>, Pitching_plus <dbl>, similarity <dbl>, SIERA <dbl>,
## #   nPitchTypes_X3 <dbl>, nPitchTypes_X4 <dbl>, nPitchTypes_X5 <dbl>,
## #   nPitchTypes_X6 <dbl>, nPitchTypes_X7 <dbl>, nAboveAvgStuf_X1 <dbl>,
## #   nAboveAvgStuf_X2 <dbl>, nAboveAvgStuf_X3 <dbl>, nAboveAvgStuf_X4 <dbl>,
## #   nAboveAvgStuf_X5 <dbl>, nAboveAvgStuf_X6 <dbl>
```

```
sp_lm_spec =
  linear_reg() %>%
  set_engine("lm")

sp_xgb_spec =
```

```
  boost_tree(
    trees = tune(),
    tree_depth = tune(),
    min_n = tune(),
    loss_reduction = tune(),
    sample_size = tune(),
    mtry = tune(),
    learn_rate = tune()
  ) %>%
  set_engine("xgboost") %>%
  set_mode("regression")

set.seed(123)
xgb_grid = grid_latin_hypercube(
  trees(),
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(), trainSP),
  learn_rate(),
  size = 100
)

lm_wf = workflow() %>%
  add_recipe(sp_rec) %>%
  add_model(sp_lm_spec)

xgb_wf = workflow() %>%
  add_recipe(sp_rec) %>%
  add_model(sp_xgb_spec)

set.seed(123)

library(finetune)
```

```
## Warning: package 'finetune' was built under R version 4.3.2
```

```
doParallel::registerDoParallel()

set.seed(123)

xgb_res = tune_grid(
  xgb_wf,
  resamples = sp_folds,
  grid = xgb_grid,
  control = control_grid(),
  metrics = metric_set(rmse)
)

best_rmse = select_best(xgb_res, "rmse")

final_sp_lm_mod = lm_wf %>%
```

```
  fit(data = trainSP)

final_sp_xgb_mod = finalize_workflow(
  xgb_wf,
  best_rmse
) %>%
  fit(data = trainSP)

testSP$predSIERAlm = predict(final_sp_lm_mod, testSP)$.pred
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response", : prediction from rank-deficient fit; consider predict(.,
## rankdeficient="NA")
```

```
testSP$predSIERAxgb = predict(final_sp_xgb_mod, testSP)$.pred

finSPlm = testSP %>%
  select(SIERA, predSIERAlm)

finSPxg = testSP %>%
  select(SIERA, predSIERAxgb)


rmse(finSPlm, truth = SIERA, estimate = predSIERAlm)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.197
```

```
rsq(finSPlm, truth = SIERA, estimate = predSIERAlm)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.914
```

```
rmse(finSPxg, truth = SIERA, estimate = predSIERAxgb)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.225
```

```
rsq(finSPxg, truth = SIERA, estimate = predSIERAxgb)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.887
```

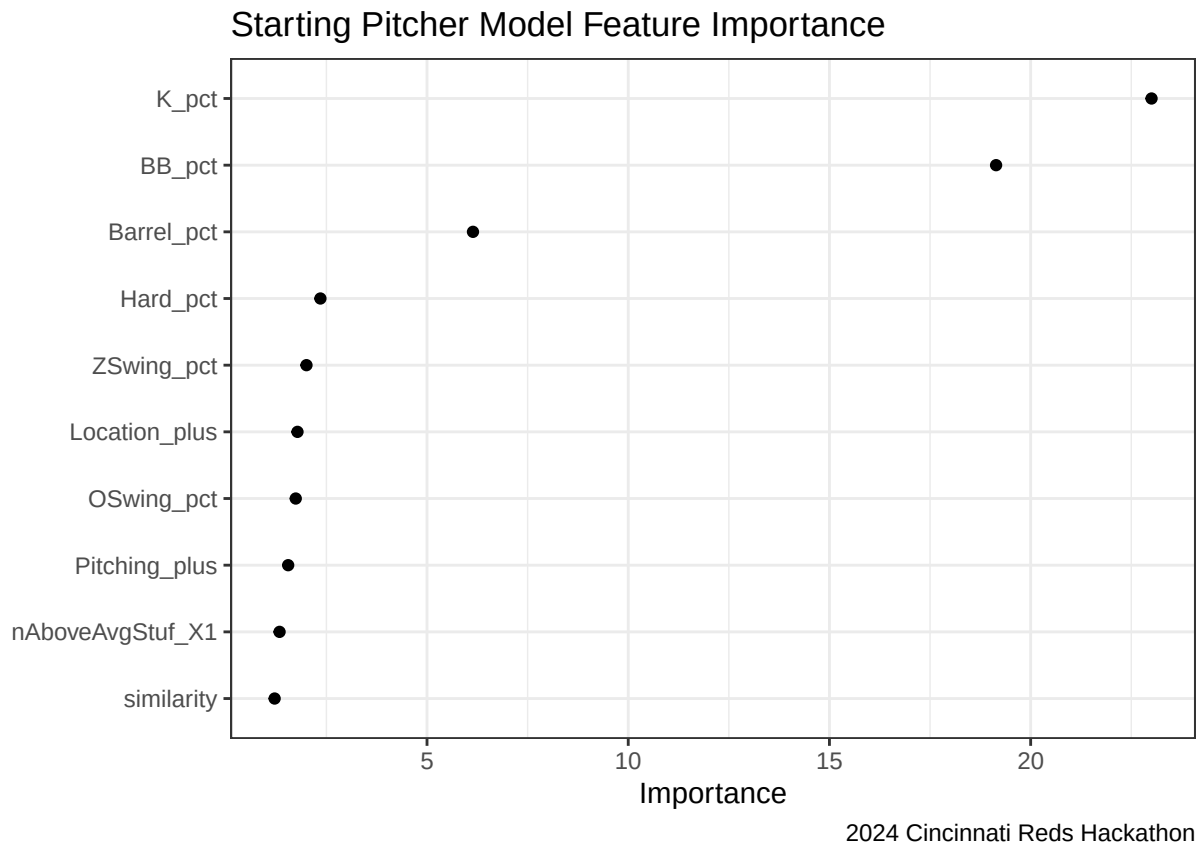
```r
library(vip)
```

```
## Warning: package 'vip' was built under R version 4.3.2
```

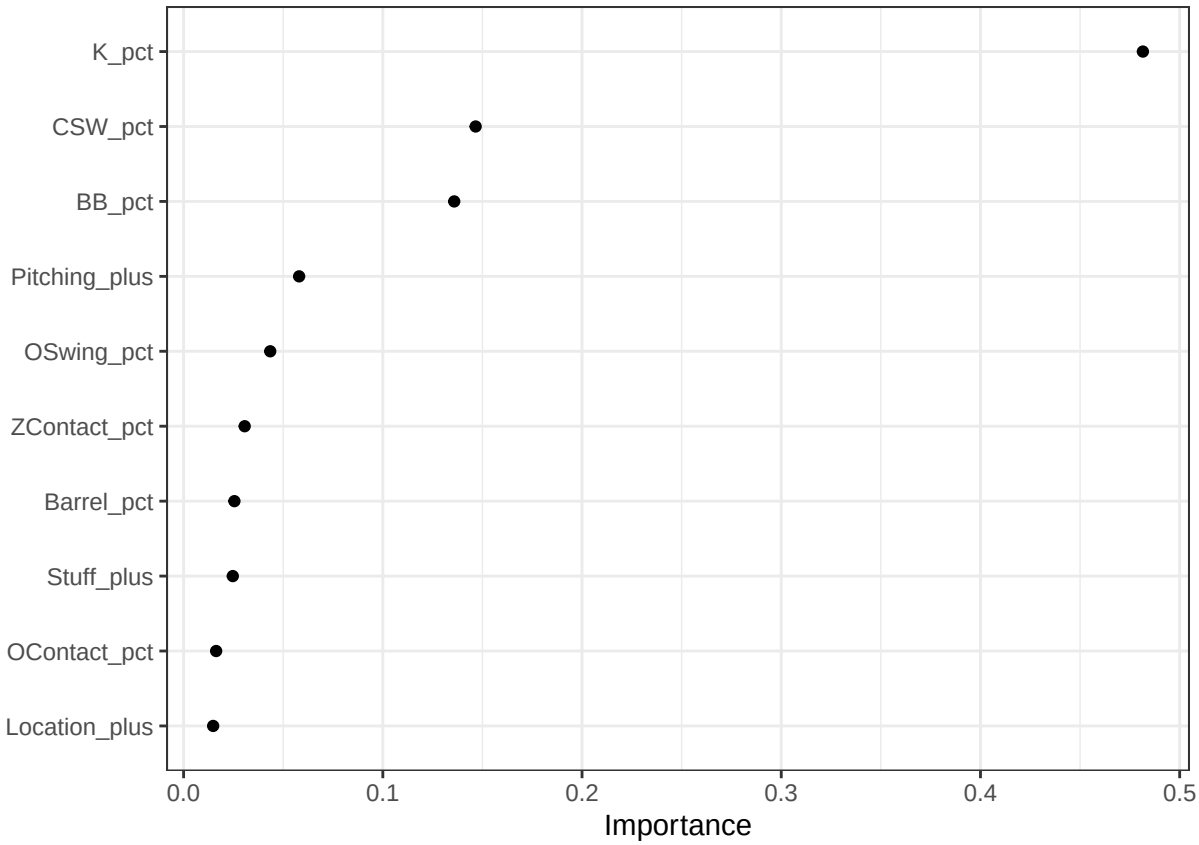
```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```r
final_sp_lm_mod %>%
  extract_fit_parsnip() %>%
  vip(geom = "point") +
  labs(title = "Starting Pitcher Model Feature Importance", caption = "2024 Cincinnati Reds Hackathon")
```

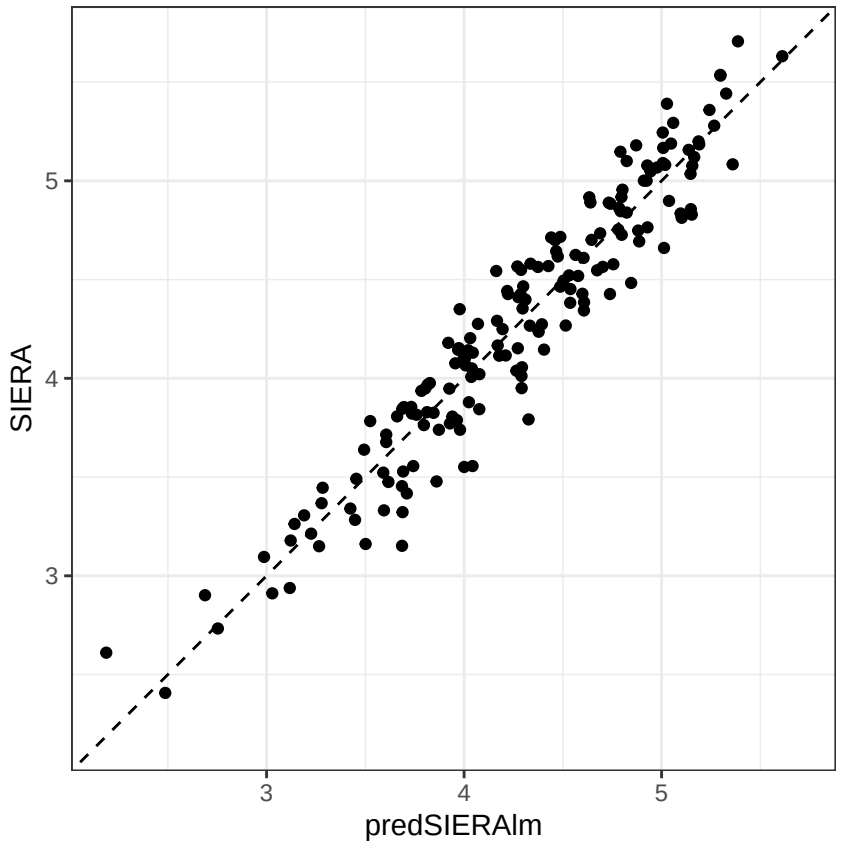



```r
final_sp_xgb_mod %>%
  extract_fit_parsnip() %>%
  vip(geom = "point")
```

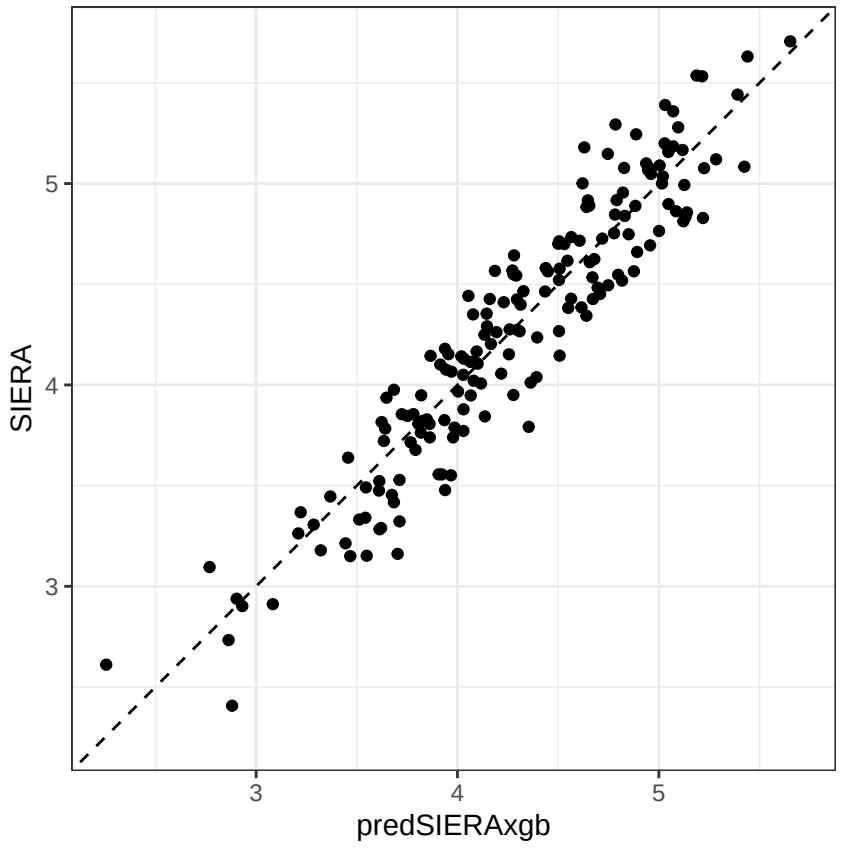



```
ggplot(finSPlm, aes(predSIERAlm, SIERA)) +
  geom_point() +
  geom_abline(lty = 2) +
  coord_obs_pred()
```

```
## Warning: Removed 5 rows containing missing values (‘geom_point()‘).
```




```
ggplot(finSPxg, aes(predSIERAxgb, SIERA)) +
  geom_point() +
  geom_abline(lty = 2) +
  coord_obs_pred()
```



Modeling Relievers

```
rp_rec =
  recipe(SIERA ~ ., data = trainRP) %>%
  step_dummy(all_nominal_predictors())

rp_rec %>%
  prep() %>%
  bake(new_data = NULL) %>%
  head()
```

```
## # A tibble: 6 x 26
##   K_pct BB_pct CSW_pct OSwing_pct OContact_pct ZSwing_pct ZContact_pct Hard_pct
##   <dbl>  <dbl>   <dbl>      <dbl>        <dbl>      <dbl>        <dbl>    <dbl>
## 1 0.423 0.0262   0.295      0.383        0.428      0.808        0.805    0.345
## 2 0.426 0.0979   0.355      0.328        0.374      0.661        0.782    0.349
## 3 0.252 0.0593   0.321      0.306        0.463      0.704        0.811    0.337
## 4 0.324 0.052    0.335      0.372        0.490      0.669        0.877    0.346
## 5 0.357 0.0762   0.282      0.328        0.515      0.783        0.802    0.328
## 6 0.245 0.0314   0.292      0.338        0.612      0.649        0.889    0.304
## # i 18 more variables: Barrel_pct <dbl>, EV <dbl>, Stuff_plus <dbl>,
## #   Location_plus <dbl>, Pitching_plus <dbl>, similarity <dbl>, SIERA <dbl>,
## #   nPitchTypes_X3 <dbl>, nPitchTypes_X4 <dbl>, nPitchTypes_X5 <dbl>,
## #   nPitchTypes_X6 <dbl>, nPitchTypes_X7 <dbl>, nAboveAvgStuf_X1 <dbl>,
```

```
## #   nAboveAvgStuf_X2 <dbl>, nAboveAvgStuf_X3 <dbl>, nAboveAvgStuf_X4 <dbl>,
## #   nAboveAvgStuf_X5 <dbl>, nAboveAvgStuf_X6 <dbl>
```

```
rp_lm_spec =
  linear_reg() %>%
  set_engine("lm")

rp_xgb_spec =
  boost_tree(
    trees = tune(),
    tree_depth = tune(),
    min_n = tune(),
    loss_reduction = tune(),
    sample_size = tune(),
    mtry = tune(),
    learn_rate = tune()
  ) %>%
  set_engine("xgboost") %>%
  set_mode("regression")


set.seed(123)
xgb_grid = grid_latin_hypercube(
  trees(),
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(), trainRP),
  learn_rate(),
  size = 100
)

lm_wf = workflow() %>%
  add_recipe(rp_rec) %>%
  add_model(rp_lm_spec)

xgb_wf = workflow() %>%
  add_recipe(rp_rec) %>%
  add_model(rp_xgb_spec)



doParallel::registerDoParallel()

set.seed(123)

xgb_res = tune_grid(
  xgb_wf,
  resamples = rp_folds,
  grid = xgb_grid,
  control = control_grid(),
  metrics = metric_set(rmse)
)
```

```
best_rmse = select_best(xgb_res, "rmse")

final_rp_lm_mod = lm_wf %>%
  fit(data = trainRP)

final_rp_xgb_mod = finalize_workflow(
  xgb_wf,
  best_rmse
) %>%
  fit(data = trainRP)

testRP$predSIERAlm = predict(final_rp_lm_mod, testRP)$.pred
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response", : prediction from rank-deficient fit; consider predict(.,
## rankdeficient="NA")
```

```
testRP$predSIERAxgb = predict(final_rp_xgb_mod, testRP)$.pred

finRPlm = testRP %>%
  select(SIERA, predSIERAlm)

finRPxg = testRP %>%
  select(SIERA, predSIERAxgb)


rmse(finRPlm, truth = SIERA, estimate = predSIERAlm)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.258
```

```
rsq(finRPlm, truth = SIERA, estimate = predSIERAlm)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.890
```

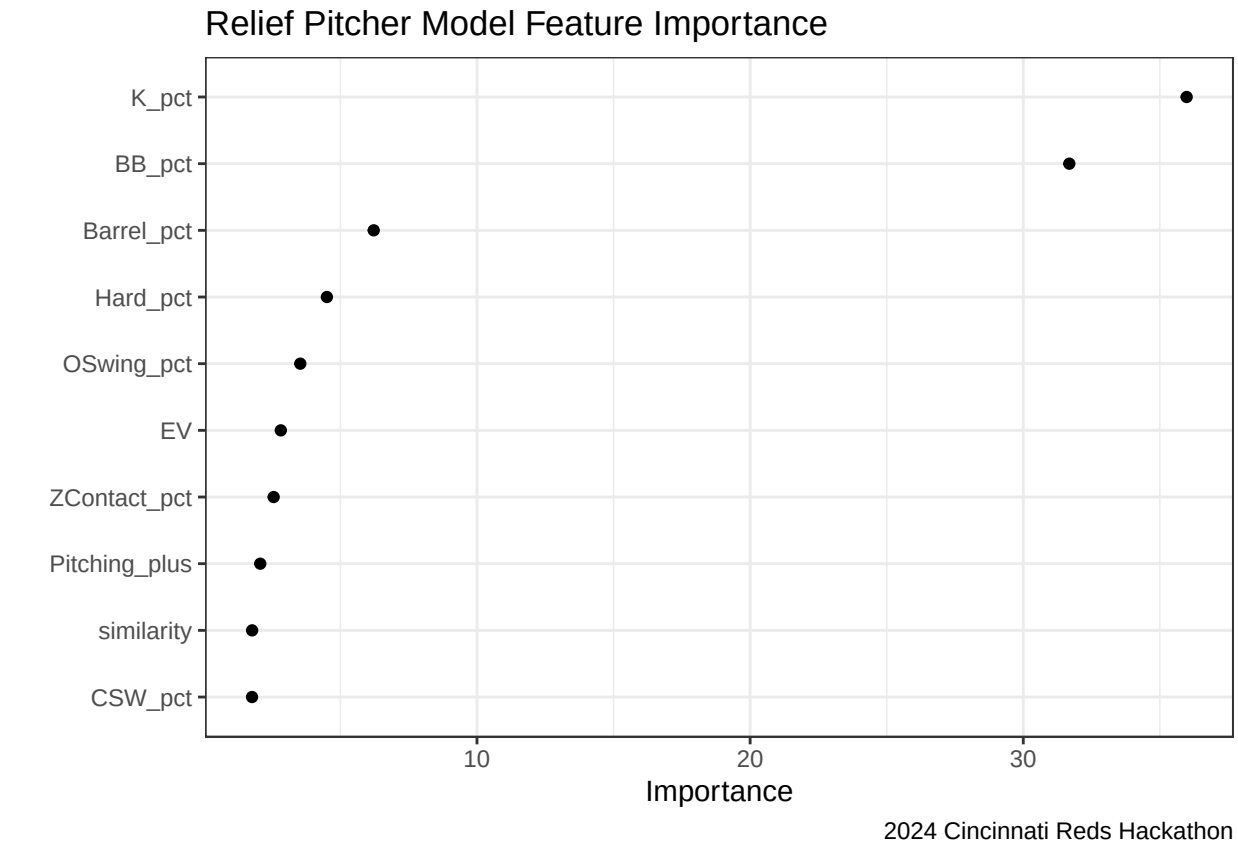
```
rmse(finRPxg, truth = SIERA, estimate = predSIERAxgb)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.291
```

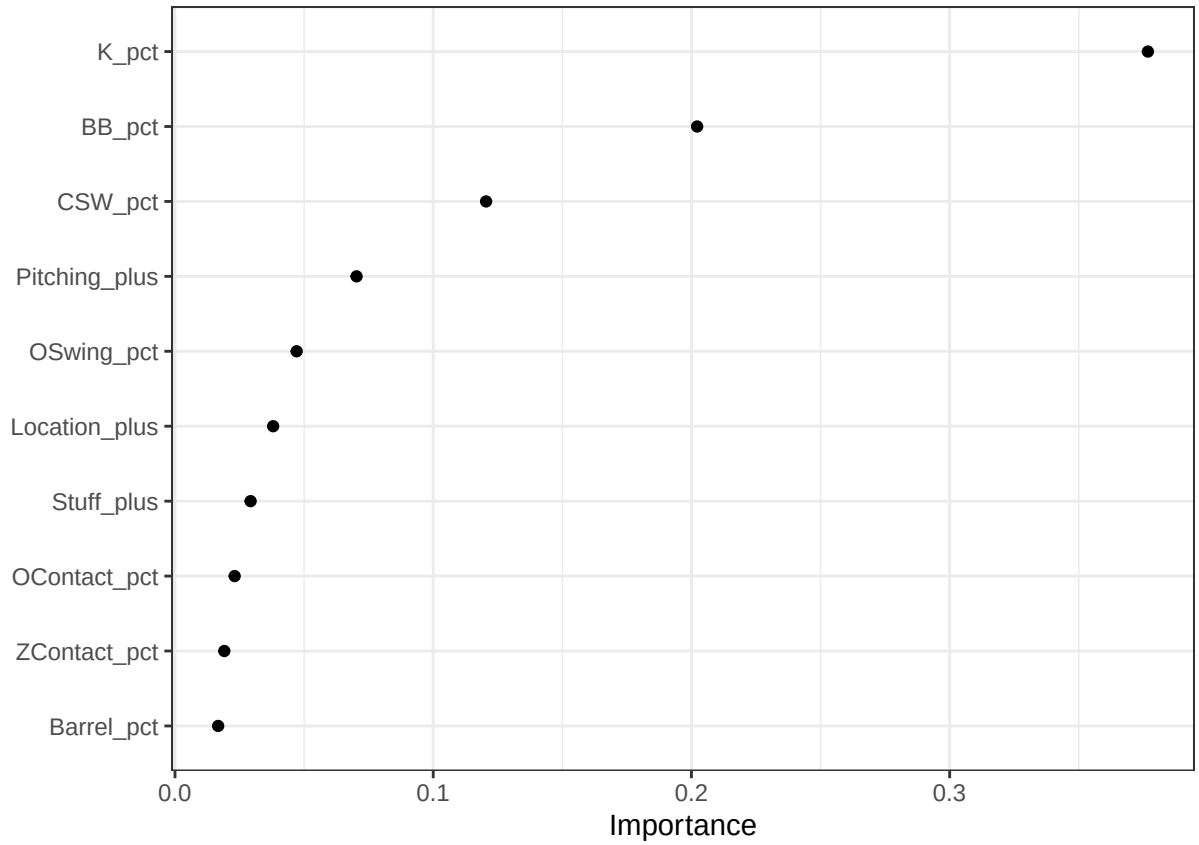
```
rsq(finRPxg, truth = SIERA, estimate = predSIERAxgb)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.861
```

```
library(vip)

final_rp_lm_mod %>%
  extract_fit_parsnip() %>%
  vip(geom = "point") +
  labs(title = "Relief Pitcher Model Feature Importance", caption = "2024 Cincinnati Reds Hackathon")
```

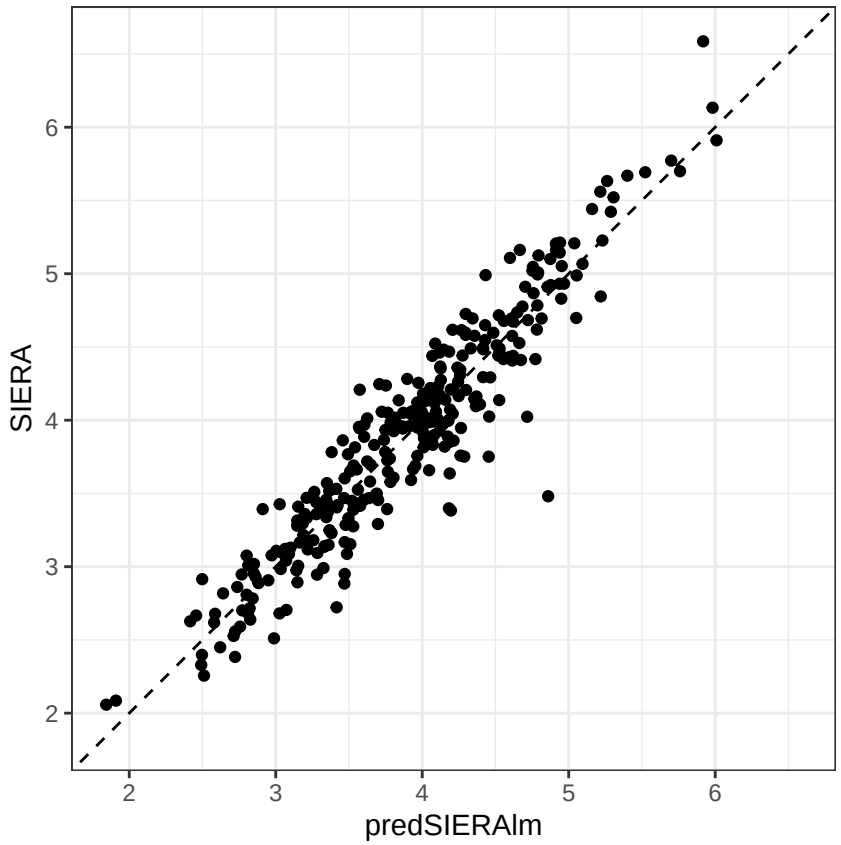



```
final_rp_xgb_mod %>%
  extract_fit_parsnip() %>%
  vip(geom = "point")
```

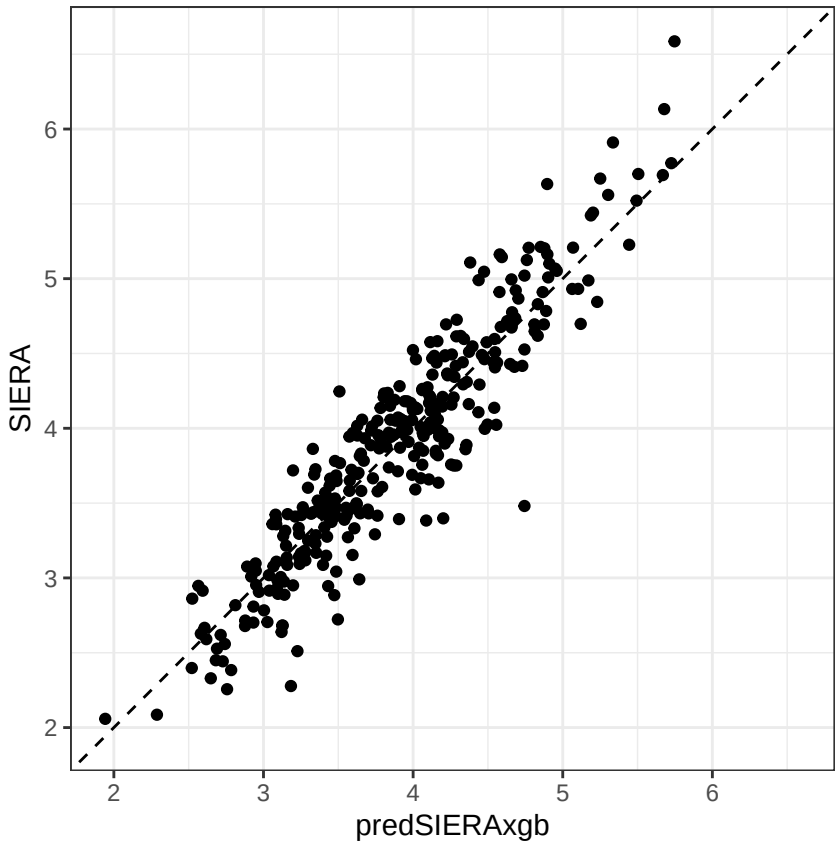



```
ggplot(finRPlm, aes(predSIERAlm, SIERA)) +
  geom_point() +
  geom_abline(lty = 2) +
  coord_obs_pred()
```

```
## Warning: Removed 18 rows containing missing values (`geom_point()`).
```




```
ggplot(finRPxg, aes(predSIERAxgb, SIERA)) +
  geom_point() +
  geom_abline(lty = 2) +
  coord_obs_pred()
```



Prediction

```
SIERAsp = predict(final_rp_lm_mod, new_data = SPadj)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response", : prediction from rank-deficient fit; consider predict(.,
## rankdeficient="NA")
```

```
SIERArp = predict(final_sp_lm_mod, new_data = RPadj)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response", : prediction from rank-deficient fit; consider predict(.,
## rankdeficient="NA")
```

```
newRPs = bind_cols(SPadj, cbind(SIERAsp, SIERAsp = starters$SIERA))
newSPs = bind_cols(RPadj, cbind(SIERArp, SIERArp = relievers$SIERA.rp))

newRPs = newRPs %>%
  mutate(diff = SIERAsp - .pred)
newSPs = newSPs %>%
  mutate(diff = SIERArp - .pred)
```