Baseball_script.R

William

2019-04-04

```r
all_pitches <- read.csv("all_pitches.csv", header = T)
all_stats <- read.csv("all_stats.csv", header = T)
all_players <- read.csv("all_players.csv", header = T)

str(all_pitches)
```

```
## 'data.frame':    2944 obs. of  52 variables:
##  $ pitch_no          : int  131 130 129 128 127 126 125 124 123 122 ...
##  $ game_date         : Factor w/ 32 levels "4/13/2018","4/14/2018",..: 32 32 32 32 32 32 32 32 32 32
##  $ pa_of_inning      : int  7 7 7 7 7 7 6 6 6 6 ...
##  $ pitch_of_pa       : int  6 5 4 3 2 1 5 4 3 2 ...
##  $ pitcher           : Factor w/ 3 levels "Player A","Player B",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ pitcher_id        : int  111112 111112 111112 111112 111112 111112 111112 111112 111112 111112 ..
##  $ pitcher_throws    : Factor w/ 1 level "Right": 1 1 1 1 1 1 1 1 1 1 ...
##  $ batter_side       : Factor w/ 2 levels "Left","Right": 1 1 1 1 1 1 2 2 2 2 ...
##  $ pitcher_set       : Factor w/ 3 levels "Stretch","Undefined",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ inning            : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ top_bottom        : Factor w/ 2 levels "Bottom","Top": 1 1 1 1 1 1 1 1 1 1 ...
##  $ outs              : int  2 2 2 2 2 2 1 1 1 1 ...
##  $ balls             : int  3 3 2 1 0 0 2 1 1 0 ...
##  $ strikes           : int  2 1 1 1 1 0 2 2 1 1 ...
##  $ tagged_pitch_type : Factor w/ 8 levels "ChangeUp","Curveball",..: 4 4 4 4 4 4 4 4 4 2 ...
##  $ auto_pitch_type   : Factor w/ 7 levels "ChangeUp","Curveball",..: 3 3 3 3 3 3 3 3 3 2 ...
##  $ pitch_call        : Factor w/ 8 levels "BallCalled","BallIntentional",..: 1 3 1 1 1 6 7 1 6 1 ...
##  $ k_or_bb           : Factor w/ 3 levels "Strikeout","Undefined",..: 3 2 2 2 2 2 1 2 2 2 ...
##  $ hit_type          : Factor w/ 6 levels "Bunt","FlyBall",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ play_result       : Factor w/ 8 levels "Double","Error",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ outs_on_play      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ runs_scored       : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ notes             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ rel_speed         : num  93.7 93.5 93 94.3 93.6 ...
##  $ vert_rel_angle    : num  -2.95 -1.34 -1.02 -2.13 -1.17 ...
##  $ horz_rel_angle    : num  -2.55 -3.72 -4.63 -2.52 -2.78 ...
##  $ spin_rate         : num  2379 2225 2352 2322 2405 ...
##  $ spin_axis         : num  205 200 199 201 192 ...
##  $ tilt              : Factor w/ 33 levels "","1:00","1:15",..: 11 11 11 11 10 11 10 2 11 24 ...
##  $ rel_height        : num  6.06 6.18 6.09 6.23 6.18 ...
##  $ rel_side          : num  2.93 2.98 2.93 2.97 3.16 ...
##  $ extension         : num  6.56 6.47 6.55 6.35 6.43 ...
##  $ vert_break        : num  -15.2 -16.1 -15.1 -12.4 -14.6 ...
##  $ induced_vert_break: num  15.7 15.1 16.3 18.5 16.5 ...
##  $ horz_break        : num  7.89 6.06 6.35 7.68 4.3 ...
##  $ plate_loc_height  : num  1.86 3.38 3.67 3.01 3.66 ...
##  $ plate_loc_side    : num  1.1296 -0.0535 -0.9086 1.1786 0.8502 ...
##  $ zone_speed        : num  87.5 87.1 86.7 87.5 87.2 ...
##  $ vert_appr_angle   : num  -6.03 -4.59 -4.09 -4.69 -4.15 ...
##  $ horz_appr_angle   : num  -1.31 -2.82 -3.68 -1.31 -2.21 ...
```

```
##  $ zone_time         : num  0.397 0.399 0.401 0.397 0.399 ...
##  $ pfxx              : num  -4.21 -3.47 -3.83 -4.09 -2.29 ...
##  $ pfxz              : num  8.35 7.61 8.24 9.65 8.31 ...
##  $ x0                : num  -2.75 -2.71 -2.6 -2.78 -2.96 ...
##  $ y0                : int  50 50 50 50 50 50 50 50 50 50 ...
##  $ z0                : num  5.73 5.95 5.89 5.95 5.97 ...
##  $ vx0               : num  5.9 8.73 10.83 5.84 6.54 ...
##  $ vy0               : num  -137 -137 -136 -138 -137 ...
##  $ vz0               : num  -7.58 -3.77 -2.93 -5.58 -3.31 ...
##  $ ax0               : num  -8.12 -6.67 -7.27 -7.96 -4.41 ...
##  $ ay0               : num  25.7 25.6 24.9 27.2 25.6 ...
##  $ az0               : num  -16.1 -17.6 -16.5 -13.4 -16.2 ...
```

```
View(all_pitches)

library("tidyverse")
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages ---------------------------------------------------------------------------- t

## v ggplot2 3.1.0       v purrr   0.3.1
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## -- Conflicts ------------------------------------------------------------------------------- tidyver
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

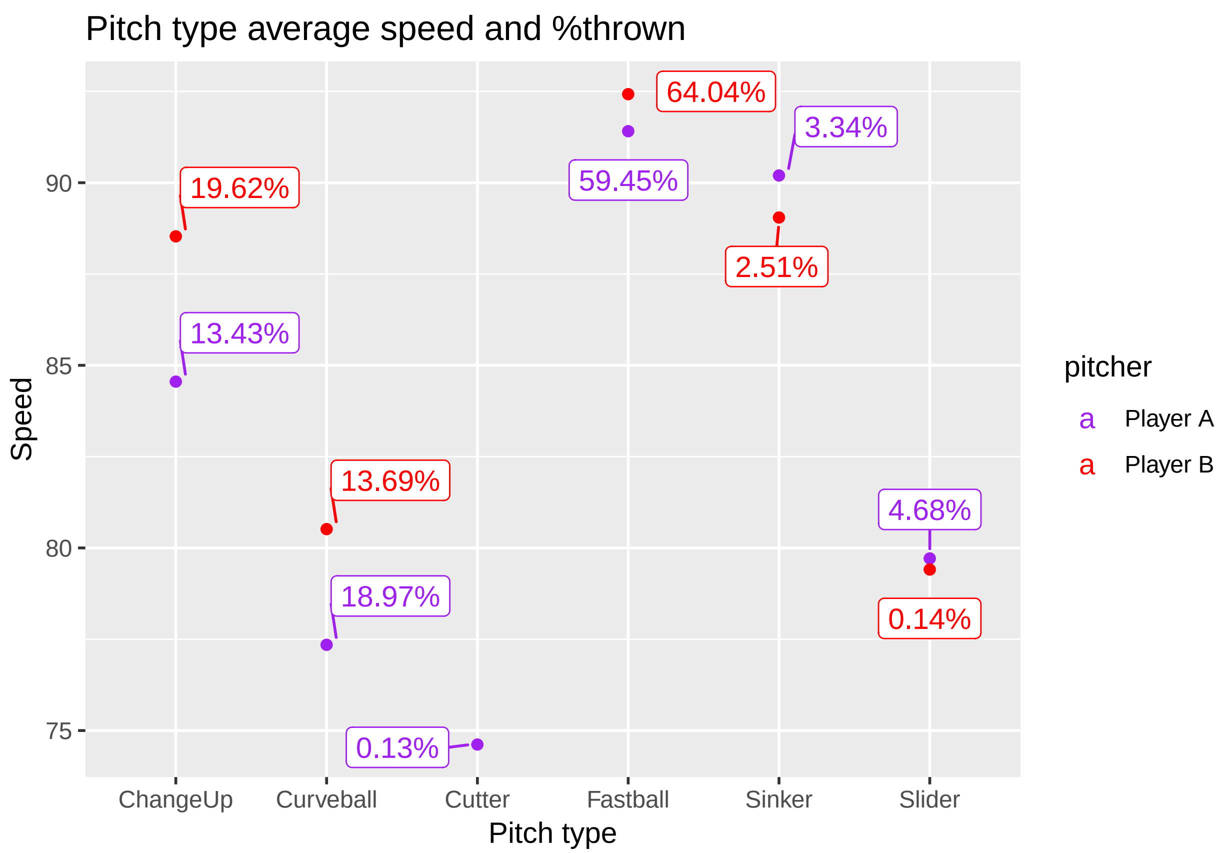
```
col_drop <- c("notes")
require(dplyr)

all_pitches <- all_pitches[,!(names(all_pitches) %in% col_drop)]

all_pitches$pitcher <- str_replace(all_pitches$pitcher, "Povse", "Player A")
all_pitches$pitcher <- as.factor(all_pitches$pitcher)

all_pitches$game_date <- as.character(all_pitches$game_date)

all_pitches$game_date <- strptime(as.character(all_pitches$game_date), "%d/%m/%Y")

all_pitches$inning <- as.factor(all_pitches$inning)

unique(all_pitches$tagged_pitch_type)
```

```
## [1] Fastball  Curveball ChangeUp  Undefined Slider    Sinker    Cutter
## [8] Other
## 8 Levels: ChangeUp Curveball Cutter Fastball Other Sinker ... Undefined
```

```r
unique(all_pitches$auto_pitch_type)
```

```
## [1] Fastball  Curveball ChangeUp  Other     Undefined Sinker    Splitter
## Levels: ChangeUp Curveball Fastball Other Sinker Splitter Undefined
```

```r
#Decided to use tagged pitch type and would use the auto pitch type to fill in undefined
cols.factor <- c("tagged_pitch_type", "auto_pitch_type")

all_pitches[cols.factor] <- sapply(all_pitches[cols.factor], as.character)

change_pitches <- c("Undefined", "Other")

all_pitches$tagged_pitch_type <- ifelse(all_pitches$tagged_pitch_type %in% change_pitches,
                                        all_pitches$auto_pitch_type, all_pitches$tagged_pitch_type)

all_pitches[cols.factor] <- lapply(all_pitches[cols.factor], as.factor)


all_pitches <- all_pitches[!(all_pitches$tagged_pitch_type %in% change_pitches),]

#1. Which of the two pitchers will likely get injured first based on previous research?

quick_summaries <- all_pitches %>%
  group_by(pitcher, tagged_pitch_type) %>%
  summarize(mean_speed = mean(rel_speed, na.rm = T))

times_thrown <- group_by(all_pitches,pitcher, tagged_pitch_type) %>%
  count()

quick_summaries <- merge(quick_summaries, times_thrown, by = c("pitcher", "tagged_pitch_type"))

quick_summaries <- quick_summaries %>%
  group_by(pitcher) %>%
  mutate(pct_pitched = n / sum(n))

quick_summaries$pct_pitched <- quick_summaries$pct_pitched * 100
quick_summaries$pct_pitched <- round(quick_summaries$pct_pitched, 2)
quick_summaries$pct_pitched <- paste(quick_summaries$pct_pitched, sep = "", "%")

library("ggrepel")

ggplot(quick_summaries, aes(tagged_pitch_type, mean_speed, color = pitcher)) +
  geom_point() +
  scale_color_manual(breaks = c("Player A", "Player B"),
                     values = c("purple", "red")) +
  labs(title = "Pitch type average speed and %thrown", x = "Pitch type", y = "Speed") +
  geom_label_repel(aes(label = pct_pitched),
                   box.padding = 0.25,
                   point.padding = 0.5)
```

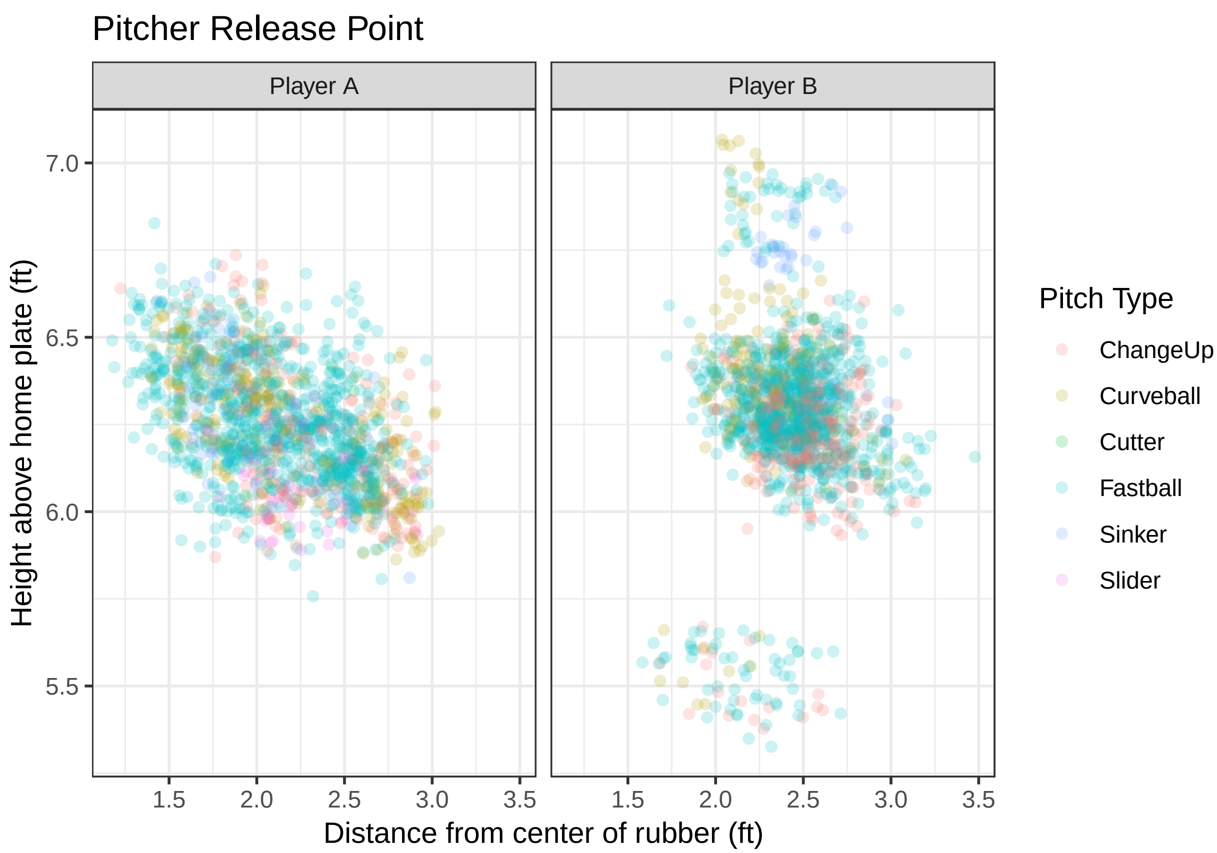



```
speed_cors <- cor(all_pitches[, c(23:27, 29:ncol(all_pitches))], use = "pairwise.complete.obs")
```

```
## Warning in cor(all_pitches[, c(23:27, 29:ncol(all_pitches))], use =
## "pairwise.complete.obs"): the standard deviation is zero
```

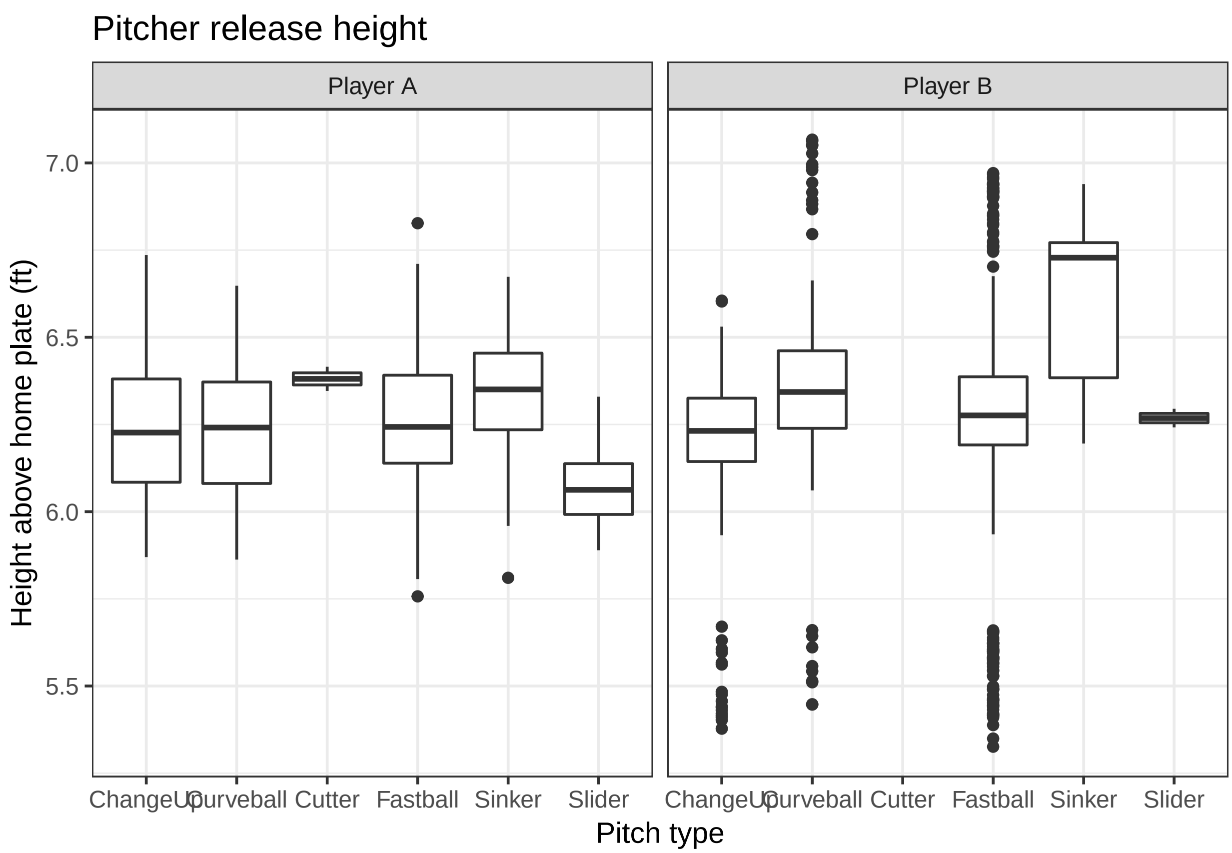
```
pitcher_dif <- aggregate(all_pitches[, c(23:27, 29:ncol(all_pitches))], by = list(all_pitches$tagged_pi
          FUN = mean, na.rm = T)

pitcher_dif <- pitcher_dif %>%
  arrange(Group.1)

pitcher_dif <- pitcher_dif[-5, ]

ggplot(all_pitches, aes(x = rel_side, y = rel_height, color = tagged_pitch_type)) +
  geom_jitter(alpha = .2) +
  facet_grid(~ pitcher) +
  labs(title = "Pitcher Release Point", x = "Distance from center of rubber (ft)",
       y = "Height above home plate (ft)",
       color = "Pitch Type") +
  theme_bw()
```

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




```r
ggplot(all_pitches, aes(x = tagged_pitch_type, y = rel_height)) +
  geom_boxplot() +
  facet_grid(~ pitcher) +
  labs(title = "Pitcher release height", x = "Pitch type", y = "Height above home plate (ft)") +
  theme(legend.position = "None") +
  theme_bw()
```

```
## Warning: Removed 2 rows containing non-finite values (stat_boxplot).
```




```r
#2. During which pitch of the plate appearance will the pitcher most likely throw a specific pitch?

all_pitches$Pitch_count <- paste(all_pitches$balls, all_pitches$strikes, sep = "-")

all_pitches <- all_pitches[,c(1:14, 52, 15:ncol(all_pitches))]
all_pitches <- all_pitches[, -53]


all_pitches$Pitch_count <- as.factor(all_pitches$Pitch_count)



all_pitches$tagged_pitch_type <- droplevels(all_pitches)$tagged_pitch_type


library("caret")
```

```
## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
```

```
##
##     lift
```

```r
library("nnet")
library("e1071")
```

```
## Warning: package 'e1071' was built under R version 3.5.3
```

```r
training.samples <- all_pitches$tagged_pitch_type %>%
  createDataPartition(p = 0.8, list = F)

train.data <- all_pitches[training.samples,]
test.data <- all_pitches[-training.samples,]

myCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

model <- train(tagged_pitch_type ~ strikes + balls + outs + pa_of_inning + pitcher + batter_side,
               data = train.data,
               method = "multinom",
               trControl = myCtrl,
               na.action = na.omit)
```

```
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2329.514917
## iter  20 value 2142.152365
## iter  30 value 2100.471303
## iter  40 value 2097.158447
## iter  50 value 2096.449110
## iter  60 value 2096.099808
## iter  70 value 2096.046271
## final  value 2096.046129
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2333.103064
## iter  20 value 2142.819490
## iter  30 value 2104.426885
## iter  40 value 2102.278096
## iter  50 value 2102.128425
## final  value 2102.096857
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2329.518465
## iter  20 value 2142.156226
## iter  30 value 2100.474678
## iter  40 value 2097.147566
## iter  50 value 2096.457505
## iter  60 value 2096.116069
## final  value 2096.072058
## converged
## # weights:  48 (35 variable)
```

```
## initial  value 3778.820721
## iter  10 value 2360.162752
## iter  20 value 2149.626640
## iter  30 value 2117.128143
## iter  40 value 2111.703569
## iter  50 value 2110.399290
## iter  60 value 2110.201501
## iter  70 value 2109.919584
## final  value 2109.914605
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2362.776291
## iter  20 value 2160.469210
## iter  30 value 2120.501783
## iter  40 value 2116.615482
## iter  50 value 2115.797124
## iter  60 value 2115.762710
## final  value 2115.762642
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2360.165293
## iter  20 value 2149.654710
## iter  30 value 2117.138872
## iter  40 value 2111.710418
## iter  50 value 2110.407016
## iter  60 value 2110.212454
## iter  70 value 2110.026867
## final  value 2110.008974
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2369.439383
## iter  20 value 2146.454299
## iter  30 value 2107.463835
## iter  40 value 2102.868544
## iter  50 value 2102.093924
## iter  60 value 2101.786473
## iter  70 value 2101.720949
## final  value 2101.720767
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2371.675349
## iter  20 value 2153.825454
## iter  30 value 2111.216863
## iter  40 value 2108.127403
## iter  50 value 2107.773519
## final  value 2107.755508
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2369.441489
```

```
## iter  20 value 2146.461888
## iter  30 value 2107.467340
## iter  40 value 2102.875035
## iter  50 value 2102.101958
## iter  60 value 2101.799068
## final  value 2101.747420
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2348.532365
## iter  20 value 2150.354702
## iter  30 value 2119.523127
## iter  40 value 2113.985791
## iter  50 value 2112.954705
## iter  60 value 2112.580480
## iter  70 value 2112.556027
## final  value 2112.555962
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2352.555272
## iter  20 value 2152.545720
## iter  30 value 2123.278264
## iter  40 value 2118.746687
## iter  50 value 2118.576163
## final  value 2118.540152
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2348.536379
## iter  20 value 2150.356725
## iter  30 value 2119.527599
## iter  40 value 2113.991698
## iter  50 value 2112.962675
## iter  60 value 2112.598402
## iter  70 value 2112.582810
## final  value 2112.581889
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2373.506408
## iter  20 value 2160.882970
## iter  30 value 2119.938448
## iter  40 value 2115.158921
## iter  50 value 2114.312431
## iter  60 value 2113.974225
## iter  70 value 2113.909476
## final  value 2113.909233
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2376.275901
## iter  20 value 2166.671081
## iter  30 value 2124.356395
```

```
## iter  40 value 2120.553430
## iter  50 value 2120.099366
## final  value 2120.056295
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2373.509097
## iter  20 value 2160.888183
## iter  30 value 2119.943110
## iter  40 value 2115.165594
## iter  50 value 2114.320499
## iter  60 value 2113.987681
## final  value 2113.937179
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2353.226266
## iter  20 value 2159.943422
## iter  30 value 2118.711755
## iter  40 value 2114.267620
## iter  50 value 2113.431612
## iter  60 value 2113.101657
## iter  70 value 2113.044288
## final  value 2113.044193
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2356.359835
## iter  20 value 2164.941839
## iter  30 value 2124.338183
## iter  40 value 2119.755048
## iter  50 value 2119.080935
## iter  60 value 2119.047731
## iter  60 value 2119.047721
## iter  60 value 2119.047721
## final  value 2119.047721
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2353.229318
## iter  20 value 2159.947548
## iter  30 value 2118.717897
## iter  40 value 2114.274283
## iter  50 value 2113.439462
## iter  60 value 2113.114829
## final  value 2113.071755
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2363.811662
## iter  20 value 2154.116553
## iter  30 value 2104.484730
## iter  40 value 2099.824546
## iter  50 value 2098.723688
```

```
## iter  60 value 2098.066430
## iter  70 value 2097.695889
## final  value 2097.692859
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2366.517638
## iter  20 value 2151.244383
## iter  30 value 2112.898226
## iter  40 value 2105.996864
## iter  50 value 2105.409429
## iter  60 value 2105.360563
## iter  60 value 2105.360546
## iter  60 value 2105.360546
## final  value 2105.360546
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2363.814314
## iter  20 value 2154.119745
## iter  30 value 2104.490470
## iter  40 value 2099.834076
## iter  50 value 2098.737412
## iter  60 value 2098.114700
## iter  70 value 2097.766149
## final  value 2097.764362
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2377.367230
## iter  20 value 2142.297676
## iter  30 value 2111.866171
## iter  40 value 2106.387902
## iter  50 value 2105.626978
## iter  60 value 2105.412546
## iter  70 value 2105.249401
## final  value 2105.249062
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2378.596656
## iter  20 value 2147.242164
## iter  30 value 2117.976866
## iter  40 value 2111.803885
## iter  50 value 2111.460259
## final  value 2111.412450
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2377.368304
## iter  20 value 2142.297515
## iter  30 value 2111.878130
## iter  40 value 2106.395261
## iter  50 value 2105.635255
```

```
## iter  60 value 2105.422836
## final  value 2105.274552
## converged
## # weights:  48 (35 variable)
## initial  value 3785.987758
## iter  10 value 2372.485306
## iter  20 value 2162.696136
## iter  30 value 2123.679179
## iter  40 value 2115.771254
## iter  50 value 2114.739336
## iter  60 value 2114.384534
## iter  70 value 2114.306436
## final  value 2114.306302
## converged
## # weights:  48 (35 variable)
## initial  value 3785.987758
## iter  10 value 2375.620594
## iter  20 value 2167.407498
## iter  30 value 2128.188239
## iter  40 value 2120.645045
## iter  50 value 2120.237830
## iter  60 value 2120.225750
## final  value 2120.225725
## converged
## # weights:  48 (35 variable)
## initial  value 3785.987758
## iter  10 value 2372.488415
## iter  20 value 2162.701081
## iter  30 value 2123.684121
## iter  40 value 2115.778065
## iter  50 value 2114.747080
## iter  60 value 2114.397955
## iter  70 value 2114.337677
## final  value 2114.337603
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2372.587243
## iter  20 value 2157.736987
## iter  30 value 2119.075129
## iter  40 value 2114.087741
## iter  50 value 2113.417357
## iter  60 value 2113.077998
## iter  70 value 2113.036157
## final  value 2113.036009
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2374.245558
## iter  20 value 2161.008886
## iter  30 value 2122.457467
## iter  40 value 2119.370151
## iter  50 value 2119.062535
## final  value 2119.033646
```

```
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2372.588800
## iter  20 value 2157.740075
## iter  30 value 2119.079338
## iter  40 value 2114.094445
## iter  50 value 2113.425398
## iter  60 value 2113.093179
## iter  70 value 2113.062965
## final  value 2113.062106
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2354.670672
## iter  20 value 2143.851898
## iter  30 value 2111.591468
## iter  40 value 2107.981891
## iter  50 value 2106.973151
## iter  60 value 2106.695211
## iter  70 value 2106.586876
## final  value 2106.586255
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2357.232602
## iter  20 value 2149.440339
## iter  30 value 2115.328501
## iter  40 value 2113.078335
## iter  50 value 2112.688530
## final  value 2112.656302
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2354.673135
## iter  20 value 2143.930523
## iter  30 value 2111.592443
## iter  40 value 2107.996592
## iter  50 value 2106.981268
## iter  60 value 2106.707654
## iter  70 value 2106.616937
## final  value 2106.616874
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2360.730358
## iter  20 value 2148.390516
## iter  30 value 2114.312537
## iter  40 value 2110.486914
## iter  50 value 2109.680127
## iter  60 value 2109.419045
## iter  70 value 2109.313694
## final  value 2109.313567
## converged
```

```
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2363.407434
## iter  20 value 2152.813018
## iter  30 value 2118.612117
## iter  40 value 2115.700673
## iter  50 value 2115.280452
## final  value 2115.258031
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2360.732951
## iter  20 value 2148.394565
## iter  30 value 2114.316958
## iter  40 value 2110.493898
## iter  50 value 2109.688066
## iter  60 value 2109.430985
## iter  70 value 2109.340661
## final  value 2109.338609
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2363.306579
## iter  20 value 2159.921209
## iter  30 value 2118.995734
## iter  40 value 2110.872845
## iter  50 value 2109.717589
## iter  60 value 2108.970799
## iter  70 value 2108.814359
## final  value 2108.814040
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2366.411534
## iter  20 value 2169.010143
## iter  30 value 2122.060424
## iter  40 value 2115.783788
## iter  50 value 2115.330715
## iter  60 value 2115.290522
## final  value 2115.290194
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2363.309644
## iter  20 value 2159.927293
## iter  30 value 2119.002925
## iter  40 value 2110.878584
## iter  50 value 2109.728011
## iter  60 value 2109.019038
## iter  70 value 2108.907929
## iter  80 value 2108.898904
## iter  90 value 2108.897904
## iter 100 value 2108.892862
## final  value 2108.892862
```

```
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2355.550095
## iter  20 value 2159.690011
## iter  30 value 2114.045762
## iter  40 value 2108.987520
## iter  50 value 2108.152312
## iter  60 value 2107.872432
## iter  70 value 2107.773907
## final  value 2107.773608
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2358.222054
## iter  20 value 2160.365338
## iter  30 value 2117.403964
## iter  40 value 2114.068080
## iter  50 value 2113.720629
## final  value 2113.679629
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2355.552686
## iter  20 value 2159.690400
## iter  30 value 2114.049807
## iter  40 value 2108.993269
## iter  50 value 2108.159825
## iter  60 value 2107.882415
## final  value 2107.799697
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2369.766956
## iter  20 value 2164.878829
## iter  30 value 2126.312640
## iter  40 value 2117.091712
## iter  50 value 2116.051315
## iter  60 value 2115.673594
## iter  70 value 2115.596323
## final  value 2115.596234
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2372.988721
## iter  20 value 2167.716315
## iter  30 value 2127.947694
## iter  40 value 2122.238496
## iter  50 value 2121.876215
## iter  60 value 2121.856851
## iter  60 value 2121.856835
## iter  60 value 2121.856835
## final  value 2121.856835
## converged
```

```
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2369.770145
## iter  20 value 2164.887022
## iter  30 value 2126.336128
## iter  40 value 2117.098886
## iter  50 value 2116.059434
## iter  60 value 2115.686293
## iter  70 value 2115.622813
## final  value 2115.622450
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2376.599338
## iter  20 value 2177.316608
## iter  30 value 2119.061219
## iter  40 value 2114.295964
## iter  50 value 2113.489372
## iter  60 value 2113.331471
## iter  70 value 2113.127453
## final  value 2113.126622
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2378.086539
## iter  20 value 2173.590906
## iter  30 value 2123.323563
## iter  40 value 2119.596326
## iter  50 value 2119.170236
## final  value 2119.105769
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2376.600725
## iter  20 value 2177.320485
## iter  30 value 2119.092333
## iter  40 value 2114.274239
## iter  50 value 2113.475890
## iter  60 value 2113.297261
## final  value 2113.157324
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2356.643492
## iter  20 value 2150.165196
## iter  30 value 2105.954509
## iter  40 value 2100.547953
## iter  50 value 2099.597573
## iter  60 value 2099.250284
## iter  70 value 2099.193318
## final  value 2099.193104
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
```

```
## iter  10 value 2358.993511
## iter  20 value 2153.540741
## iter  30 value 2109.780311
## iter  40 value 2105.903508
## iter  50 value 2105.349182
## final  value 2105.322016
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2356.645734
## iter  20 value 2150.168379
## iter  30 value 2105.959541
## iter  40 value 2100.554658
## iter  50 value 2099.605577
## iter  60 value 2099.265808
## iter  70 value 2099.223054
## final  value 2099.222184
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2409.036156
## iter  20 value 2166.732166
## iter  30 value 2101.811609
## iter  40 value 2092.346905
## iter  50 value 2089.932353
## iter  60 value 2089.717377
## iter  70 value 2089.339314
## final  value 2089.318982
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2405.370634
## iter  20 value 2162.897227
## iter  30 value 2104.482500
## iter  40 value 2096.235339
## iter  50 value 2095.371542
## iter  60 value 2095.359727
## final  value 2095.357960
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2409.031969
## iter  20 value 2166.732625
## iter  30 value 2101.802247
## iter  40 value 2092.346750
## iter  50 value 2089.939425
## iter  60 value 2089.726021
## iter  70 value 2089.392487
## final  value 2089.389490
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2355.102244
## iter  20 value 2159.798032
```

```
## iter  30 value 2119.415942
## iter  40 value 2115.330649
## iter  50 value 2114.266555
## iter  60 value 2113.960903
## iter  70 value 2113.840318
## final  value 2113.839867
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2358.126809
## iter  20 value 2156.998898
## iter  30 value 2123.241432
## iter  40 value 2120.306116
## iter  50 value 2119.758848
## final  value 2119.732147
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2355.105209
## iter  20 value 2159.799084
## iter  30 value 2119.419580
## iter  40 value 2115.336661
## iter  50 value 2114.274136
## iter  60 value 2113.971576
## final  value 2113.868127
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2345.606826
## iter  20 value 2161.771767
## iter  30 value 2117.873971
## iter  40 value 2111.995148
## iter  50 value 2111.375811
## iter  60 value 2110.997802
## iter  70 value 2110.846786
## final  value 2110.846303
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2349.799611
## iter  20 value 2165.973929
## iter  30 value 2123.371115
## iter  40 value 2118.359155
## iter  50 value 2117.970134
## final  value 2117.950647
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2345.611013
## iter  20 value 2161.775971
## iter  30 value 2117.879487
## iter  40 value 2112.004450
## iter  50 value 2111.387151
## iter  60 value 2111.027578
```

```
## iter  70 value 2110.913198
## final  value 2110.911008
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2354.398686
## iter  20 value 2137.342132
## iter  30 value 2114.921407
## iter  40 value 2110.749925
## iter  50 value 2109.775267
## iter  60 value 2109.120724
## iter  70 value 2109.021423
## final  value 2109.021194
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2357.302578
## iter  20 value 2145.246874
## iter  30 value 2117.816633
## iter  40 value 2115.378280
## iter  50 value 2115.294708
## iter  60 value 2115.276181
## iter  60 value 2115.276172
## iter  60 value 2115.276172
## final  value 2115.276172
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2354.401519
## iter  20 value 2137.342176
## iter  30 value 2114.928368
## iter  40 value 2110.755020
## iter  50 value 2109.786020
## iter  60 value 2109.242385
## iter  70 value 2109.163423
## final  value 2109.098710
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2358.054165
## iter  20 value 2156.595012
## iter  30 value 2112.085144
## iter  40 value 2105.508188
## iter  50 value 2104.559084
## iter  60 value 2104.359474
## iter  70 value 2104.084240
## final  value 2104.080466
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2360.879473
## iter  20 value 2158.386632
## iter  30 value 2114.010587
## iter  40 value 2110.624212
```

```
## iter  50 value 2110.264122
## final  value 2110.218466
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2358.056899
## iter  20 value 2156.595277
## iter  30 value 2112.100957
## iter  40 value 2105.514464
## iter  50 value 2104.567236
## iter  60 value 2104.372591
## iter  70 value 2104.198213
## iter  80 value 2104.154324
## iter  90 value 2104.150371
## iter 100 value 2104.148349
## final  value 2104.148349
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2345.132550
## iter  20 value 2148.433934
## iter  30 value 2113.869638
## iter  40 value 2108.859331
## iter  50 value 2107.766207
## iter  60 value 2107.422346
## iter  70 value 2107.304357
## final  value 2107.303972
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2349.192559
## iter  20 value 2160.068394
## iter  30 value 2118.214214
## iter  40 value 2114.002439
## iter  50 value 2113.534558
## iter  60 value 2113.507301
## iter  60 value 2113.507297
## iter  60 value 2113.507297
## final  value 2113.507297
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2345.136620
## iter  20 value 2148.440910
## iter  30 value 2113.874111
## iter  40 value 2108.865651
## iter  50 value 2107.774228
## iter  60 value 2107.433581
## final  value 2107.330080
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2380.634641
## iter  20 value 2146.511072
```

```
## iter  30 value 2115.009864
## iter  40 value 2109.623203
## iter  50 value 2108.549810
## iter  60 value 2108.370231
## iter  70 value 2108.134506
## final  value 2108.133893
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2381.717021
## iter  20 value 2148.172432
## iter  30 value 2119.100175
## iter  40 value 2114.548972
## iter  50 value 2114.069354
## final  value 2114.040262
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2380.635545
## iter  20 value 2146.510843
## iter  30 value 2115.013739
## iter  40 value 2109.628947
## iter  50 value 2108.557377
## iter  60 value 2108.378879
## iter  70 value 2108.159428
## iter  70 value 2108.159425
## iter  70 value 2108.159425
## final  value 2108.159425
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2365.888963
## iter  20 value 2168.885506
## iter  30 value 2114.790083
## iter  40 value 2109.822427
## iter  50 value 2108.822050
## iter  60 value 2108.558043
## iter  70 value 2108.430609
## final  value 2108.430250
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2368.956486
## iter  20 value 2168.491299
## iter  30 value 2119.826377
## iter  40 value 2114.979632
## iter  50 value 2114.439133
## final  value 2114.401032
## converged
## # weights:  48 (35 variable)
## initial  value 3784.195999
## iter  10 value 2365.891973
## iter  20 value 2168.885126
## iter  30 value 2114.793897
```

```
## iter  40 value 2109.828473
## iter  50 value 2108.829846
## iter  60 value 2108.569127
## iter  70 value 2108.457736
## final  value 2108.457266
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2378.468657
## iter  20 value 2148.035016
## iter  30 value 2115.067098
## iter  40 value 2110.028577
## iter  50 value 2109.031289
## iter  60 value 2108.782410
## iter  70 value 2108.618723
## final  value 2108.618107
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2379.485229
## iter  20 value 2150.259142
## iter  30 value 2118.186373
## iter  40 value 2115.062689
## iter  50 value 2114.566215
## final  value 2114.540215
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2378.469500
## iter  20 value 2148.033082
## iter  30 value 2115.071715
## iter  40 value 2110.033763
## iter  50 value 2109.038808
## iter  60 value 2108.792157
## iter  70 value 2108.643229
## iter  70 value 2108.643213
## iter  70 value 2108.643212
## final  value 2108.643212
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2376.695537
## iter  20 value 2169.100367
## iter  30 value 2125.174603
## iter  40 value 2119.620363
## iter  50 value 2118.550976
## iter  60 value 2118.326888
## iter  70 value 2118.170294
## final  value 2118.169556
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2378.656104
## iter  20 value 2175.262566
```

```
## iter  30 value 2129.539051
## iter  40 value 2124.996932
## iter  50 value 2124.273675
## final  value 2124.238781
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2376.697346
## iter  20 value 2169.105771
## iter  30 value 2125.178797
## iter  40 value 2119.626840
## iter  50 value 2118.558893
## iter  60 value 2118.336943
## iter  70 value 2118.203565
## final  value 2118.203259
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2367.605108
## iter  20 value 2162.904429
## iter  30 value 2113.215375
## iter  40 value 2109.319996
## iter  50 value 2108.345244
## iter  60 value 2108.079339
## iter  70 value 2107.927592
## final  value 2107.927111
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2370.457974
## iter  20 value 2169.005680
## iter  30 value 2117.962077
## iter  40 value 2114.401503
## iter  50 value 2113.919002
## final  value 2113.909157
## converged
## # weights:  48 (35 variable)
## initial  value 3780.612480
## iter  10 value 2367.607903
## iter  20 value 2162.910378
## iter  30 value 2113.220194
## iter  40 value 2109.326466
## iter  50 value 2108.352871
## iter  60 value 2108.089360
## iter  70 value 2107.957442
## final  value 2107.957309
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2382.575064
## iter  20 value 2153.044418
## iter  30 value 2114.297290
## iter  40 value 2109.728088
## iter  50 value 2109.047839
```

```
## iter  60 value 2108.721905
## iter  70 value 2108.677027
## final  value 2108.676863
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2383.165733
## iter  20 value 2155.927159
## iter  30 value 2118.544003
## iter  40 value 2115.098437
## iter  50 value 2114.725299
## final  value 2114.700631
## converged
## # weights:  48 (35 variable)
## initial  value 3782.404240
## iter  10 value 2382.575504
## iter  20 value 2153.047117
## iter  30 value 2114.301618
## iter  40 value 2109.734824
## iter  50 value 2109.055951
## iter  60 value 2108.737206
## iter  70 value 2108.705502
## final  value 2108.705201
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2333.086638
## iter  20 value 2136.196461
## iter  30 value 2101.024419
## iter  40 value 2096.041415
## iter  50 value 2095.293004
## iter  60 value 2094.932358
## iter  70 value 2094.859111
## final  value 2094.858142
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2336.450495
## iter  20 value 2140.812895
## iter  30 value 2105.808499
## iter  40 value 2102.539351
## iter  50 value 2102.255975
## final  value 2102.247485
## converged
## # weights:  48 (35 variable)
## initial  value 3778.820721
## iter  10 value 2333.089953
## iter  20 value 2136.200988
## iter  30 value 2101.029439
## iter  40 value 2096.050852
## iter  50 value 2095.307146
## iter  60 value 2094.999106
## iter  70 value 2094.962958
## final  value 2094.951640
```

```
## converged
## # weights:  48 (35 variable)
## initial  value 4201.675955
## iter  10 value 2590.636341
## iter  20 value 2403.494411
## iter  30 value 2356.896784
## iter  40 value 2351.178445
## iter  50 value 2349.884939
## iter  60 value 2349.621556
## final  value 2349.620501
## converged
```

```
model
```

```
## Penalized Multinomial Regression
##
## 2345 samples
##    6 predictor
##    6 classes: 'ChangeUp', 'Curveball', 'Cutter', 'Fastball', 'Sinker', 'Slider'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 2110, 2109, 2110, 2112, 2110, 2109, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0e+00  0.6161920  0.0499983
##   1e-04  0.6161920  0.0499983
##   1e-01  0.6166175  0.0504403
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.
```

```
predicted.pitch <- model %>% predict(test.data)

confusionMatrix(predicted.pitch, test.data$tagged_pitch_type)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  ChangeUp Curveball Cutter Fastball Sinker Slider
##   ChangeUp         0         0      0        0      0      0
##   Curveball        2         8      0        8      1      1
##   Cutter           0         0      0        0      0      0
##   Fastball        94        88      0      353     16     13
##   Sinker           0         0      0        0      0      0
##   Slider           0         0      0        0      0      0
##
## Overall Statistics
##
##                Accuracy : 0.6182
##                  95% CI : (0.5774, 0.6577)
##     No Information Rate : 0.6182
```

```
##     P-Value [Acc > NIR] : 0.5183
##
##                   Kappa : 0.0391
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: ChangeUp Class: Curveball Class: Cutter
## Sensitivity                   0.0000          0.08333            NA
## Specificity                   1.0000          0.97541             1
## Pos Pred Value                   NaN          0.40000            NA
## Neg Pred Value                0.8356          0.84397            NA
## Prevalence                    0.1644          0.16438             0
## Detection Rate                0.0000          0.01370             0
## Detection Prevalence          0.0000          0.03425             0
## Balanced Accuracy             0.5000          0.52937            NA
##                      Class: Fastball Class: Sinker Class: Slider
## Sensitivity                  0.97784       0.00000       0.00000
## Specificity                  0.05381       1.00000       1.00000
## Pos Pred Value               0.62589           NaN           NaN
## Neg Pred Value               0.60000       0.97089       0.97603
## Prevalence                   0.61815       0.02911       0.02397
## Detection Rate               0.60445       0.00000       0.00000
## Detection Prevalence         0.96575       0.00000       0.00000
## Balanced Accuracy            0.51583       0.50000       0.50000
```

```
#Simplify whether it will be a fastball or not

all_pitches$fball_or_not <- ifelse(all_pitches$tagged_pitch_type == "Fastball", "Yes", "No")
all_pitches$fball_or_not <- as.factor(all_pitches$fball_or_not)
fball_or_not <- all_pitches$fball_or_not

set.seed(1)

training.samples2 <- all_pitches$tagged_pitch_type %>%
  createDataPartition(p = 0.8, list = F)

train.data2 <- all_pitches[training.samples2,]
test.data2 <- all_pitches[-training.samples2,]

myCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

log_model <- train(fball_or_not ~ strikes + balls + outs + inning + pitch_of_pa +
                     pa_of_inning + batter_side + pitcher,
                   data = train.data2,
                   method = "pls",
                   trControl = myCtrl,
                   na.action = na.omit)


log_model
```

```
## Partial Least Squares
##
```

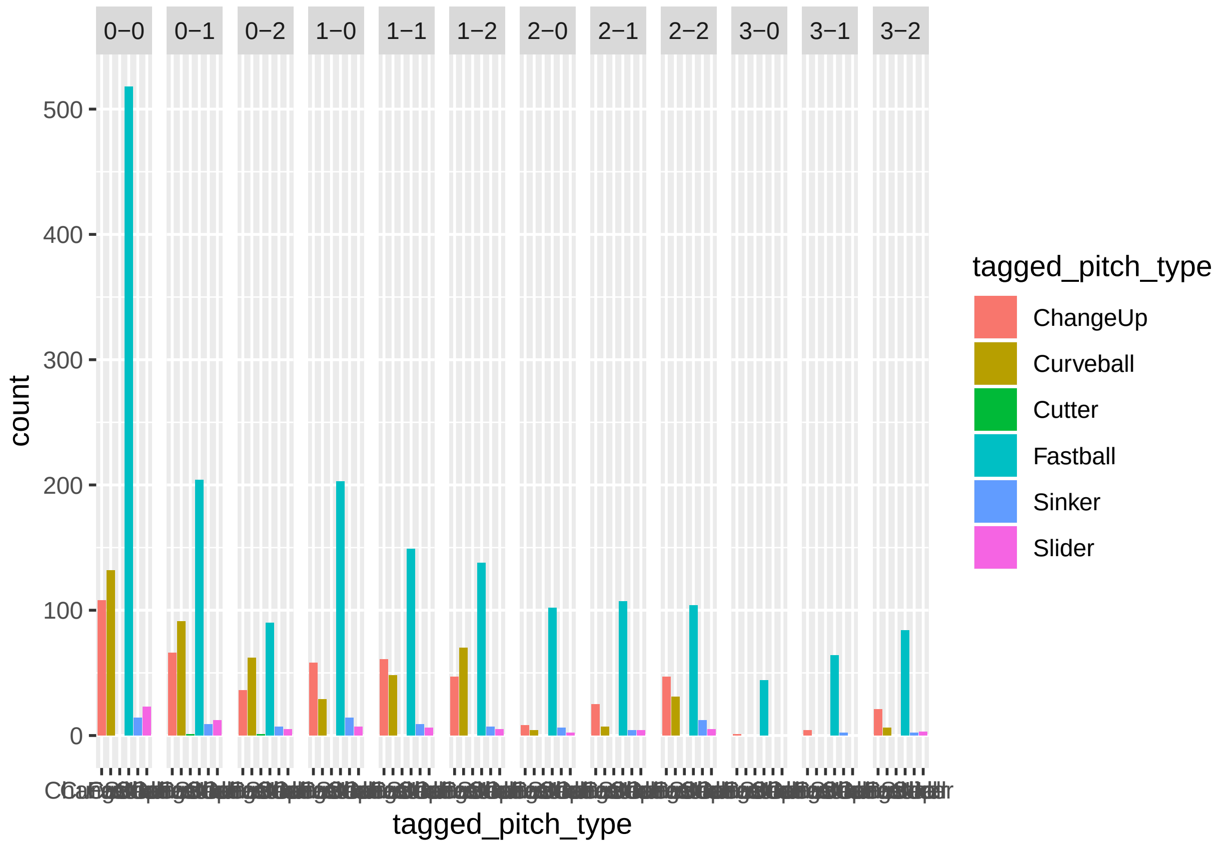
```
## 2345 samples
##    8 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 2111, 2110, 2112, 2111, 2110, 2110, ...
## Resampling results across tuning parameters:
##
##   ncomp  Accuracy   Kappa
##   1      0.6126378  0.05883657
##   2      0.6201774  0.10736310
##   3      0.6213043  0.11241763
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 3.
```

```r
predicted.pitch2 <- log_model %>% predict(test.data2)
confusionMatrix(predicted.pitch2, test.data2$fball_or_not)
```

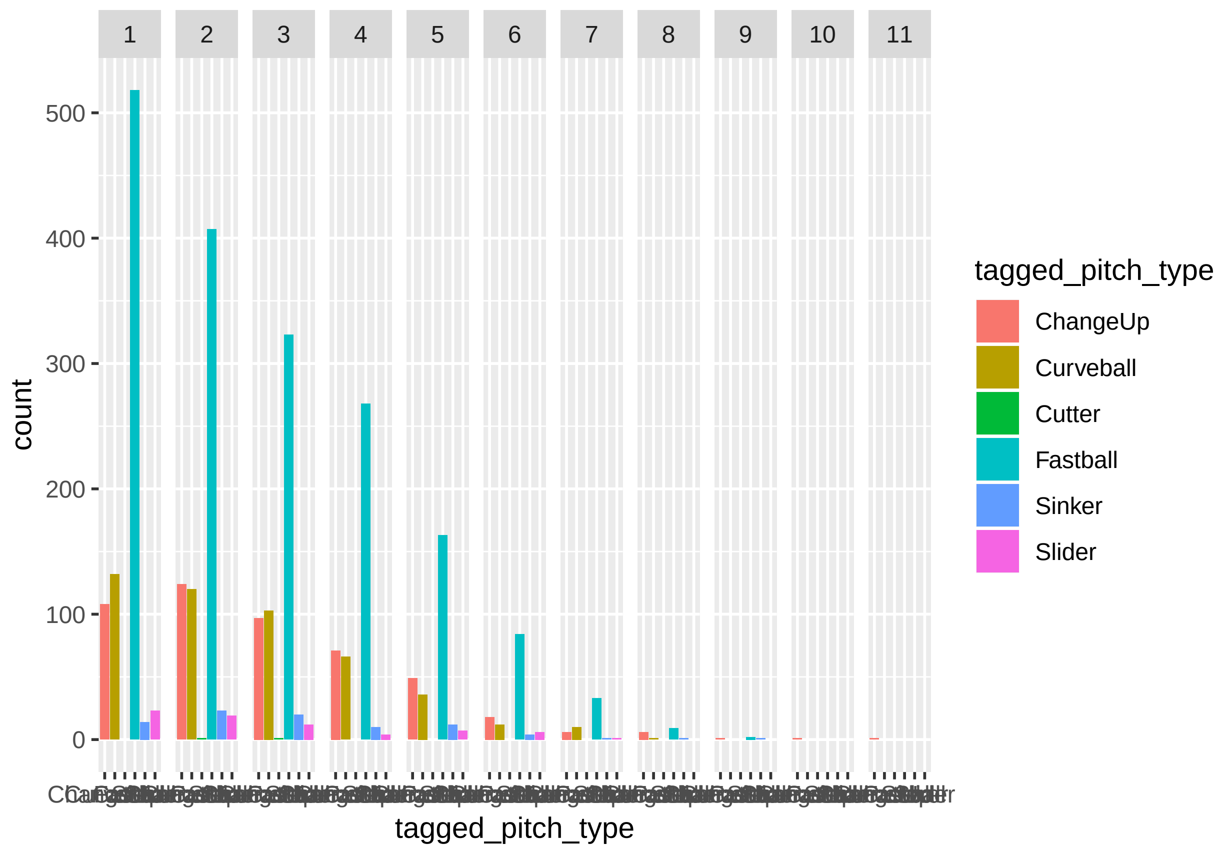
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No   46  42
##        Yes 177 319
##
##                Accuracy : 0.625
##                  95% CI : (0.5843, 0.6644)
##     No Information Rate : 0.6182
##     P-Value [Acc > NIR] : 0.384
##
##                   Kappa : 0.1017
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.20628
##             Specificity : 0.88366
##          Pos Pred Value : 0.52273
##          Neg Pred Value : 0.64315
##              Prevalence : 0.38185
##          Detection Rate : 0.07877
##    Detection Prevalence : 0.15068
##       Balanced Accuracy : 0.54497
##
##        'Positive' Class : No
##
```

```r
#There doesn't appear to be good predictive power with this dataset to determine when a pitcher
#will throw a fastball or on what specific count. Based on the amount of times the pitchers throw a fas
#best to simply guess they will throw a fastball.
```

```r
ggplot(all_pitches, aes(x = tagged_pitch_type, fill = tagged_pitch_type)) +
  geom_bar() +
  facet_grid(~Pitch_count)
```



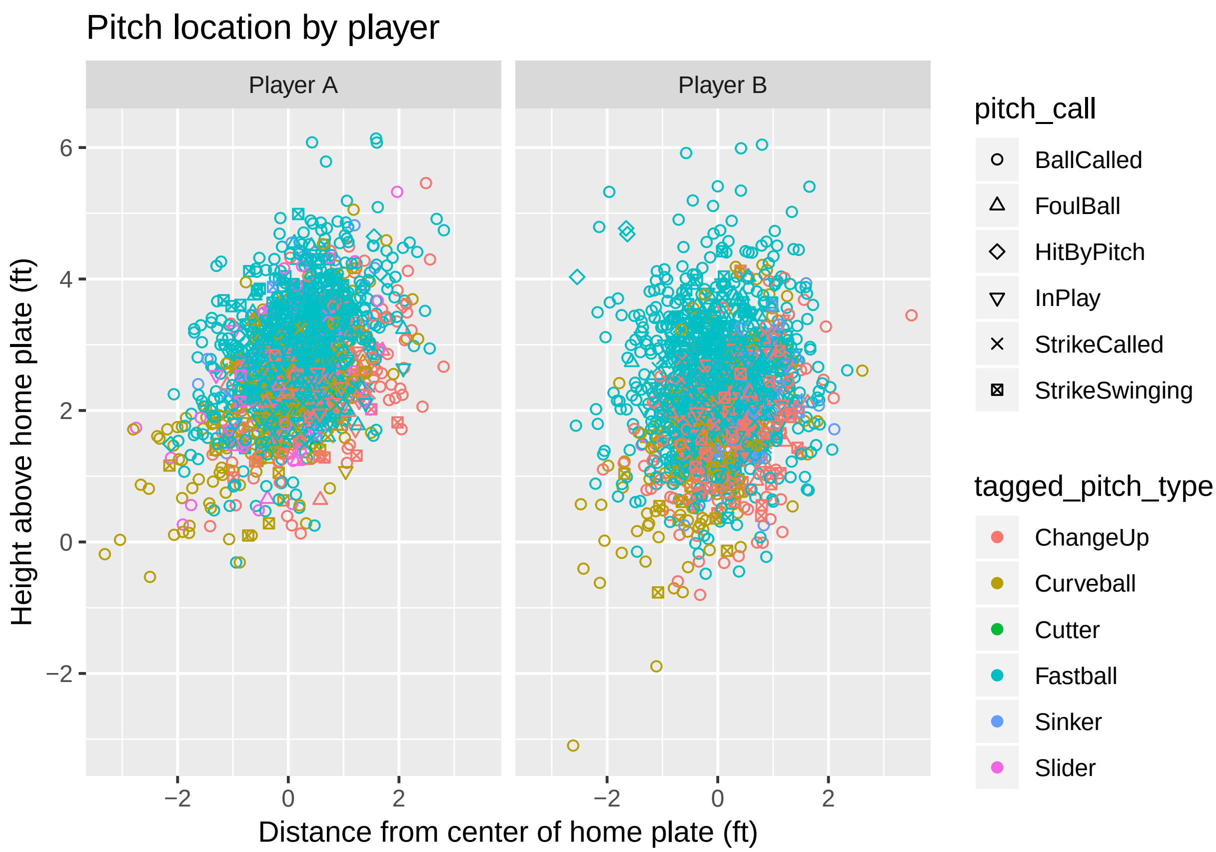

```r
ggplot(all_pitches, aes(x = tagged_pitch_type, fill = tagged_pitch_type)) +
  geom_bar() +
  facet_grid(~pitch_of_pa)
```




```
pitch_by_count <- all_pitches %>%
  group_by(pitcher, Pitch_count) %>%
  count(tagged_pitch_type)


pitch_by_pct <- pitch_by_count %>%
  group_by(pitcher, Pitch_count) %>%
  mutate(pct_pitched = n / sum(n))

pitch_by_pct$pct_pitched <- as.numeric(pitch_by_pct$pct_pitched)

pitch_by_pct$pct_pitched <- round(pitch_by_pct$pct_pitched * 100, 2)


pitch_by_pct <- pitch_by_pct[, -4]

pitch_by_pct <- pitch_by_pct %>%
  spread(key = Pitch_count, value = pct_pitched) %>%
  arrange(pitcher, desc(`0-0`))

ggplot(data = all_pitches, aes(x = plate_loc_side,
                               y = plate_loc_height,
                               color = tagged_pitch_type,
                               shape = pitch_call)) +
  geom_jitter() +
  facet_grid(~pitcher) +
```

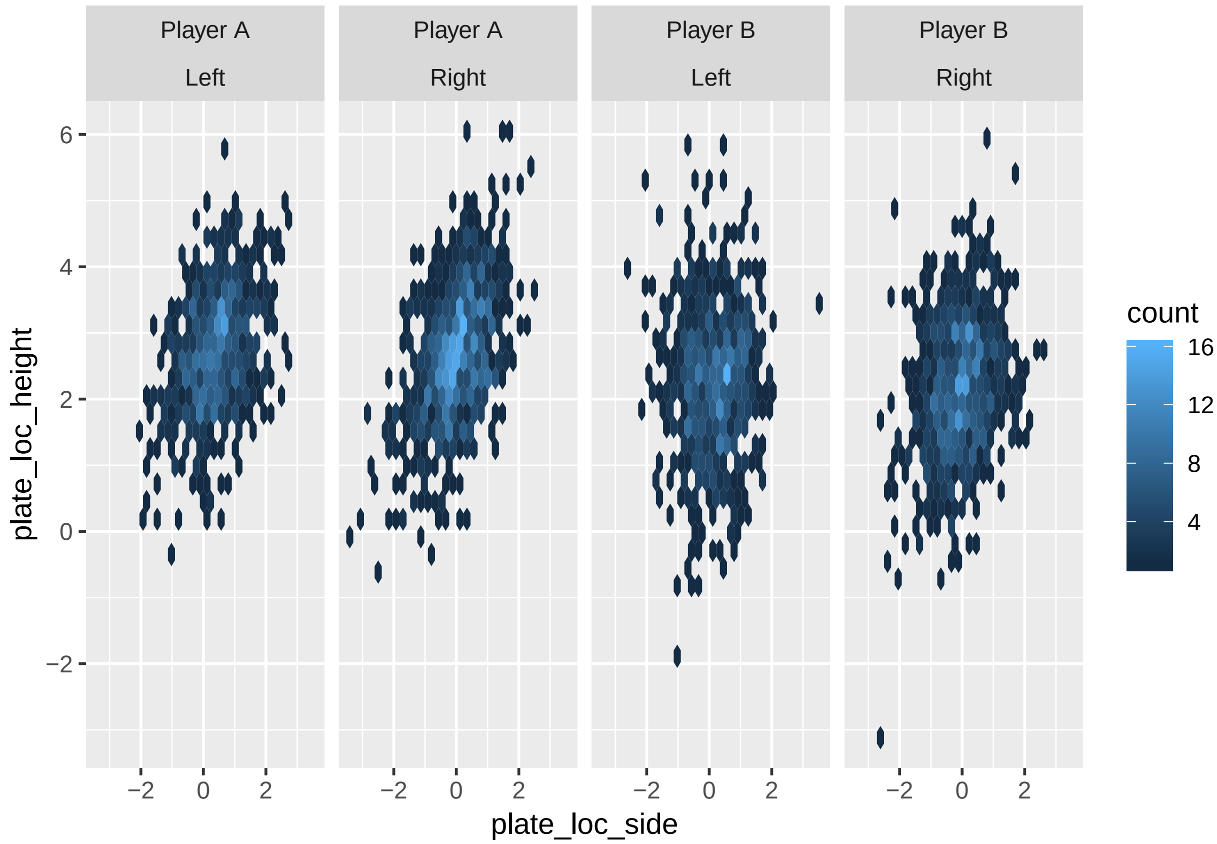
```r
  labs(title = "Pitch location by player",
       x = "Distance from center of home plate (ft)",
       y = "Height above home plate (ft)") +
  scale_shape_manual(values = c(1, 2, 5, 6, 4, 7))
```

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



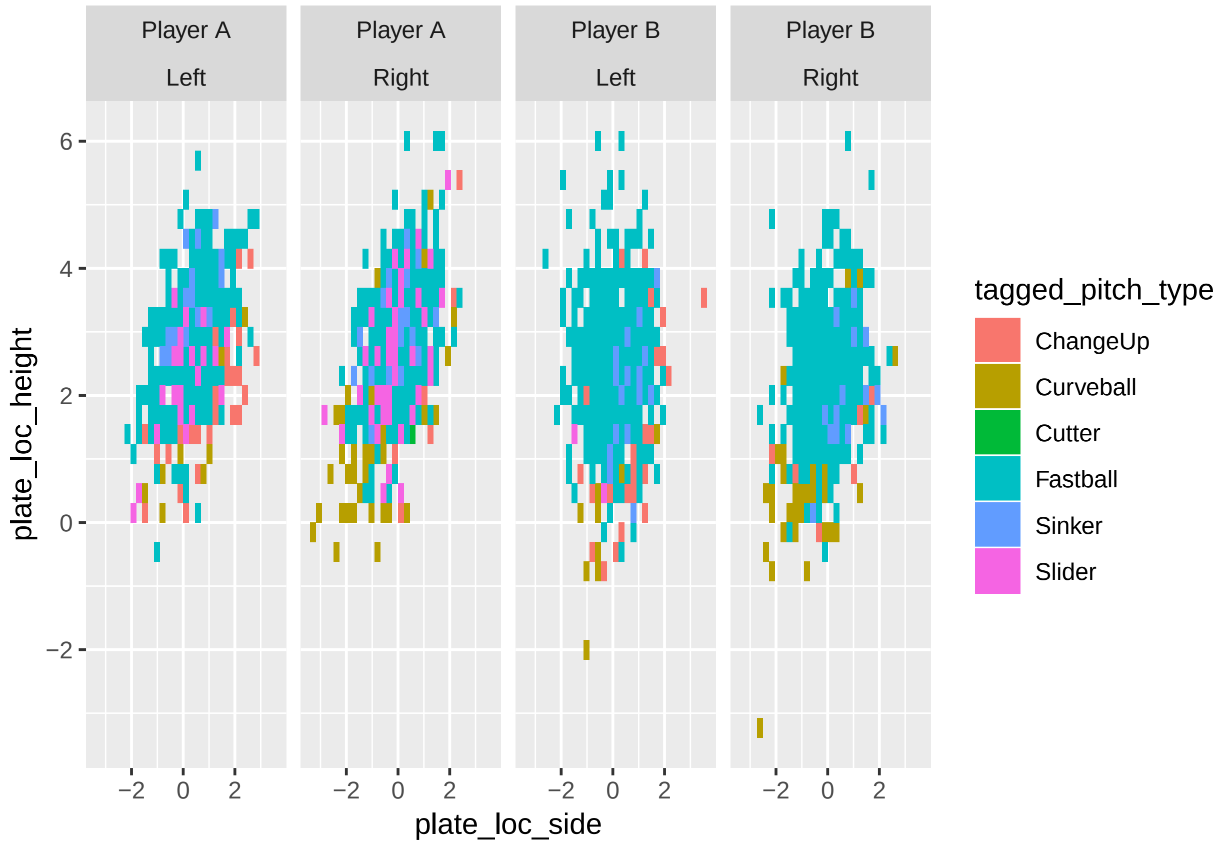

```r
ggplot(all_pitches, aes(x = plate_loc_side,
                        y = plate_loc_height)) +
  geom_hex() +
  facet_grid(~pitcher + batter_side)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_binhex).
```




```r
ggplot(all_pitches, aes(x = plate_loc_side,
                        y = plate_loc_height,
                        fill = tagged_pitch_type)) +
  geom_bin2d() +
  facet_grid(~pitcher + batter_side)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin2d).
```




```
all_pitches$game_date <- as.POSIXct(all_pitches$game_date)



#5. Effectiveness of stretch vs. windup in pitching


pitcher_sets <- c("Stretch","Windup")

set_effectiveness <- all_pitches %>%
  filter(pitcher_set %in% pitcher_sets) %>%
  group_by(pitcher, pitcher_set) %>%
  count(pitch_call, play_result)

set_effectiveness <- spread(set_effectiveness, key = pitcher_set, value = n) %>%
  arrange(pitcher, desc(Stretch))

ks_bb <- all_pitches %>%
  filter(pitcher_set %in% pitcher_sets) %>%
  group_by(pitcher, pitcher_set) %>%
  count(k_or_bb) %>%
  spread(key = pitcher_set, value = n) %>%
  arrange(pitcher, desc(`Stretch`)) %>%
  filter(k_or_bb != "Undefined") %>%
  ungroup()
```

```r
#7. Predictive modeling for stolen bases based on factors

str(all_players)
```

```
## 'data.frame':    581 obs. of  6 variables:
##  $ player_code              : int  343043 383431 384144 386053 386260 393367 393988 439850 609613 609
##  $ year                     : int  2010 2010 2010 2010 2010 2010 2016 2010 2011 2012 ...
##  $ vertical_jump_average    : num  29.4 NA 25.9 25.9 31.4 ...
##  $ rl_lat_broad_jump_average: num  100.8 NA 90.3 87.2 110.3 ...
##  $ grip_strength_average    : num  NA NA NA NA NA ...
##  $ ttest_average            : num  NA NA NA NA NA NA NA NA 6.72 9.38 ...
```

```r
str(all_stats)
```

```
## 'data.frame':    1581 obs. of  8 variables:
##  $ player_code   : int  386053 2224800 1254096 2769244 2215500 2694494 2629680 2634602 2694494 27822
##  $ year          : int  2010 2014 2010 2015 2014 2012 2010 2010 2013 2013 ...
##  $ tpa           : int  668 629 616 612 612 610 608 608 608 599 ...
##  $ sb            : int  7 26 3 12 9 33 12 14 17 10 ...
##  $ avg           : num  0.29 0.276 0.268 0.295 0.269 ...
##  $ slg           : num  0.506 0.465 0.409 0.428 0.344 ...
##  $ avg_exit_speed: Factor w/ 1204 levels "","100.5312693",..: 1 957 1 926 670 1 1 1 122 942 ...
##  $ hard_hit_perc : Factor w/ 694 levels "","0","0.004048583",..: 1 312 1 116 161 1 1 1 2 298 ...
```

```r
cols.numeric <- c("avg_exit_speed", "hard_hit_perc")

all_stats[cols.numeric] <- sapply(all_stats[cols.numeric], as.character)
all_stats[cols.numeric] <- lapply(all_stats[cols.numeric], as.numeric)
```

```
## Warning in lapply(all_stats[cols.numeric], as.numeric): NAs introduced by
## coercion
```

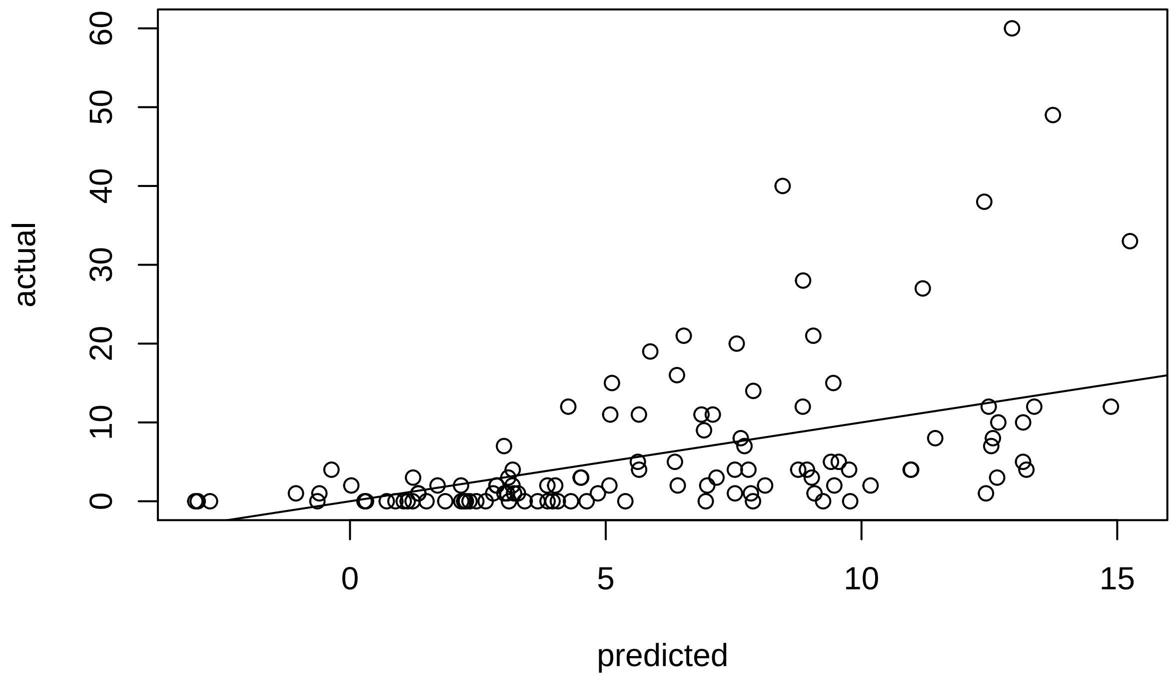
```
## Warning in lapply(all_stats[cols.numeric], as.numeric): NAs introduced by
## coercion
```

```r
dummy.vars <- dummyVars(~ ., data = all_stats[, -2])
train.dummy <- predict(dummy.vars, all_stats[, -2])
View(train.dummy)

pre.process <- preProcess(train.dummy, method = "bagImpute")
imputed.data <- predict(pre.process, train.dummy)
View(imputed.data)


all_stats$avg_exit_speed <- imputed.data[, 6]
all_stats$hard_hit_perc <- imputed.data[, 7]




all_players_stats <- merge(all_stats, all_players, by = c("player_code", "year"))
```

```r
cols.numeric <- c("avg_exit_speed", "hard_hit_perc")

all_players_stats[cols.numeric] <- sapply(all_players_stats[cols.numeric], as.character)
all_players_stats[cols.numeric] <- lapply(all_players_stats[cols.numeric], as.numeric)

summary(all_players_stats[,7:12])
```

```
##  avg_exit_speed  hard_hit_perc     vertical_jump_average
##  Min.   :68.68   Min.   :0.00000   Min.   :20.00
##  1st Qu.:84.03   1st Qu.:0.08589   1st Qu.:25.12
##  Median :86.00   Median :0.16971   Median :27.50
##  Mean   :85.59   Mean   :0.14862   Mean   :27.64
##  3rd Qu.:87.60   3rd Qu.:0.20495   3rd Qu.:30.00
##  Max.   :93.96   Max.   :0.66667   Max.   :36.00
##                                    NA's   :453
##  rl_lat_broad_jump_average grip_strength_average ttest_average
##  Min.   : 54.00            Min.   :31.00         Min.   : 6.160
##  1st Qu.: 73.28            1st Qu.:50.50         1st Qu.: 9.273
##  Median : 78.89            Median :57.00         Median : 9.570
##  Mean   : 80.65            Mean   :56.65         Mean   : 9.532
##  3rd Qu.: 84.31            3rd Qu.:62.00         3rd Qu.: 9.883
##  Max.   :128.17            Max.   :85.00         Max.   :11.390
##  NA's   :207               NA's   :204           NA's   :290
```

```r
train.samples <- createDataPartition(all_players_stats$sb, p = 0.8, list = F)

myTrain <- all_players_stats[train.samples,]
myTest <- all_players_stats[-train.samples,]

myCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10)

lmModel <- train(sb ~ tpa + avg + slg,
                 data = myTrain,
                 preProcess = c("center", "scale"),
                 method = "glmnet",
                 trControl = myCtrl)

sb_predict <- predict(lmModel, myTest)

plot(sb_predict, myTest$sb,
     xlab = "predicted", ylab = "actual")
abline(a=0,b=1)
```




```r
RMSE(sb_predict, myTest$sb)
```

```
## [1] 8.758567
```

```r
train_predict <- predict(lmModel, myTrain)

RMSE(train_predict, myTrain$sb)
```

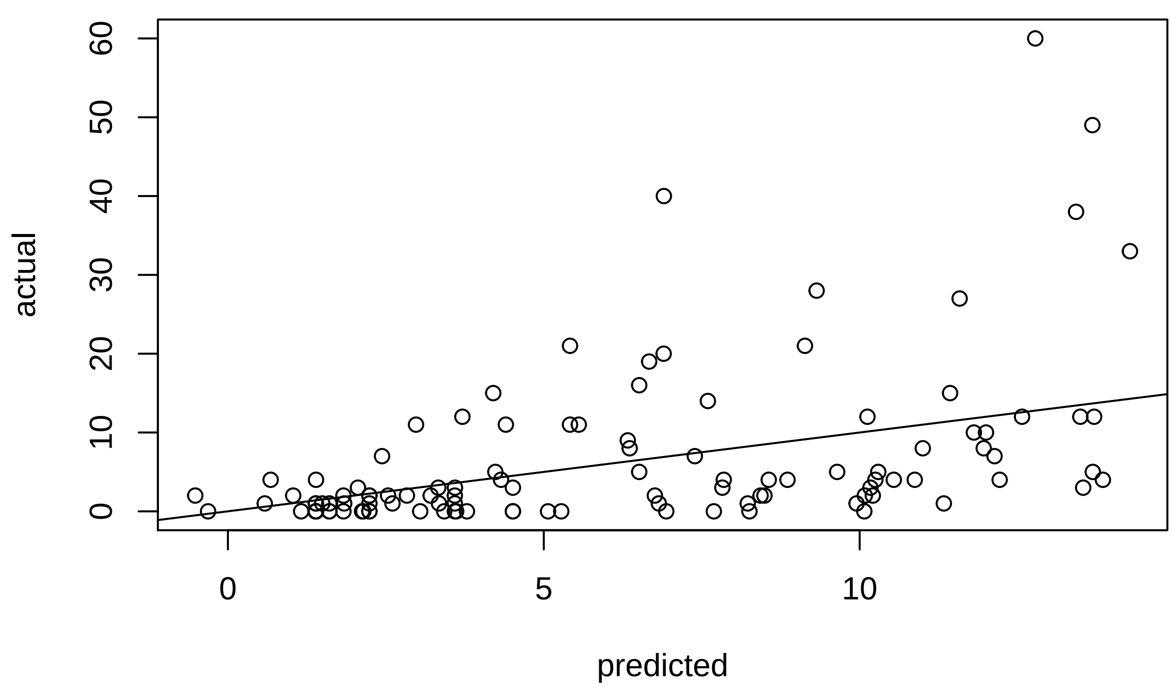
```
## [1] 7.385308
```

```r
arrange(lmModel$results, RMSE) %>% head
```

```
##   alpha      lambda     RMSE  Rsquared      MAE   RMSESD RsquaredSD
## 1  0.10 0.080299588 7.297339 0.2769227 4.923873 1.383993 0.08685573
## 2  0.10 0.008029959 7.298249 0.2767280 4.940559 1.377077 0.08624309
## 3  0.55 0.008029959 7.298319 0.2767212 4.940529 1.377032 0.08625197
## 4  0.55 0.080299588 7.298325 0.2768640 4.909975 1.389720 0.08750577
## 5  1.00 0.008029959 7.298406 0.2767220 4.940421 1.377060 0.08626706
## 6  1.00 0.080299588 7.300476 0.2766430 4.896707 1.395590 0.08818712
##       MAESD
## 1 0.6748692
## 2 0.6756645
## 3 0.6756298
## 4 0.6747428
```

```
## 5 0.6757156
## 6 0.6749907
```

```r
#___________________________________________________
#GM Boosted
#___________________________________________________

tg <- expand.grid(shrinkage = seq(0.1, 1, by = 0.2),
                  interaction.depth = c(1, 3, 7, 10),
                  n.minobsinnode = c(2, 5, 10),
                  n.trees = c(100, 300, 500, 1000))

gbm_model <- train(sb ~ tpa + avg + slg,
                   data = myTrain,
                   method = "gbm",
                   trControl = myCtrl,
                   tuneGrid =tg,
                   verbose = FALSE)

xgpredict <- predict(gbm_model, myTest)
plot(xgpredict, myTest$sb,
     xlab = "predicted", ylab = "actual")
abline(a=0,b=1)
```



```r
RMSE(xgpredict, myTest$sb)
```

```
## [1] 8.870921
```

```r
#Most of the models are not particularly good at predicting sb.
#To improve on the model I would want their speed, the count of the pitch,
#the score of the game, basically as much situational information as possible,
#and this dataset doesn't have enough of that.

#8. Correlations between physiological metrics and on-field performance

pc <- cor(all_players_stats[,9:12], all_players_stats[,5:8], use = "pairwise.complete.obs")

#Very little to no correlations between physiological metrics and on field
#performance metrics
```