Lab 4

Nate Beebe

Due on 4/12/25 at 11:59 pm

For this lab assignment you will have to create a data set called `sc_bip_small`. This data set should contain statcast data for all balls in play from 2017-2021. Statcast data can be obtained from this link:

https://uofi.app.box.com/file/1449126291821?s=we34tcz4wqdu063zpzuwjvb4r6u9j21s

There is a script called `create_sc_bip_small.R` in the **stat430sp25** repo to aid you in this task of creating the `sc_bip_small` data set.

```
library(tidyverse)
library(Lahman)

## get relevant path to statcast data set
sc = read_csv("C:/Users/STP/Downloads/statcast.csv")



# get balls in play and isolate relevant variables
# (feel free to play around with relevant variables)
sc_bip = sc %>% filter(type == "X") %>%
  dplyr::select(game_date, events, batter_name, stand, p_throws, pitcher_name, pitch_type,
                launch_speed, launch_angle, hc_x, hc_y, release_speed, release_spin_rate,
                #spin_dir,
                pfx_x, pfx_z, plate_x, plate_z, if_fielding_alignment,
                estimated_ba_using_speedangle, estimated_woba_using_speedangle,
                of_fielding_alignment, batter, pitcher)

# store data set, correcting paths
write_csv(sc_bip, file = "C:/Users/STP/Downloads/sc_bip_small.csv")
```

Question 1 Do the following for a year of your choice with the exception of 2020:

(a) List the batters with the ten highest average exit velocities on batted balls.

```
sc_bip_small = read.csv("C:/Users/STP/Downloads/sc_bip_small.csv")
sc_bip_small1 = sc_bip_small %>%
  mutate(exit_velo = launch_speed) %>%
  select(-launch_speed)
batters = sc_bip_small1 %>%
  mutate(year = year(game_date)) %>%
  filter(year == 2019) %>%
  group_by(batter_name)
```

```r
batter_ev = batters %>%
  summarise(n = n(),
            avg_ev = mean(exit_velo, na.rm = T)) %>%
  filter(n >= 300) %>%
  select(-n) %>%
  arrange(desc(avg_ev))
batter_ev %>%
  head(10)
```

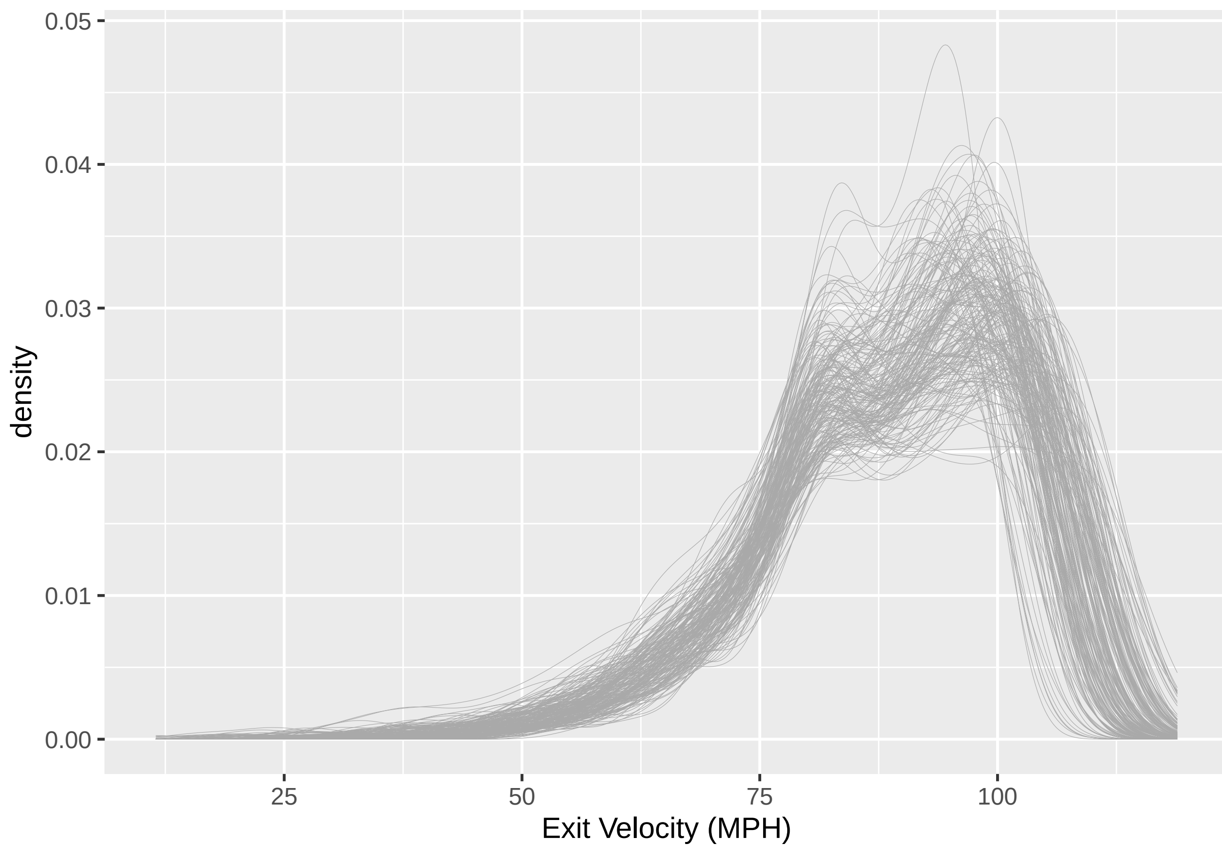
```
## # A tibble: 10 x 2
##    batter_name       avg_ev
##    <chr>              <dbl>
##  1 Nelson Cruz         93.6
##  2 Christian Yelich    93.4
##  3 Franmil Reyes       93.3
##  4 Yoán Moncada        93.1
##  5 Kyle Schwarber      92.8
##  6 Josh Donaldson      92.8
##  7 Matt Chapman        92.6
##  8 Jorge Soler         92.6
##  9 Josh Bell           92.4
## 10 Rafael Devers       92.3
```

(b) Plot the distribution of exit velocities across batters. Does exit velocity vary significantly across batters? Explain your reasoning.

```r
sc_bip_small_ = sc_bip_small %>% mutate(year = year(game_date)) %>%
  filter(year == 2019)

sc_regulars = sc_bip_small_ %>%
    filter(!is.na(launch_speed)) %>%
    inner_join(batter_ev, by = c("batter_name"))

ev_plot1 = ggplot(sc_regulars,
aes(x = launch_speed, group = batter_name)) +
geom_density(linewidth = 0.1, color = "darkgray") +
scale_x_continuous("Exit Velocity (MPH)")
ev_plot1
```



I would say it does vary. There appear to be two peaks in the data, one a little lower where I would say league average is, and one a tad higher with the leaders.

(c) List the pitchers with the ten highest average exit velocities allowed on batted balls.

```
pitchers = sc_bip_small1 %>%
  mutate(year = year(game_date)) %>%
  filter(year == 2019) %>%
  group_by(pitcher_name)

pitcher_ev = pitchers %>%
  summarise(n = n(),
            avg_ev = mean(exit_velo, na.rm = T)) %>%
  filter(n >= 300) %>%
  select(-n) %>%
  arrange(desc(avg_ev))
pitcher_ev %>%
  head(10)
```

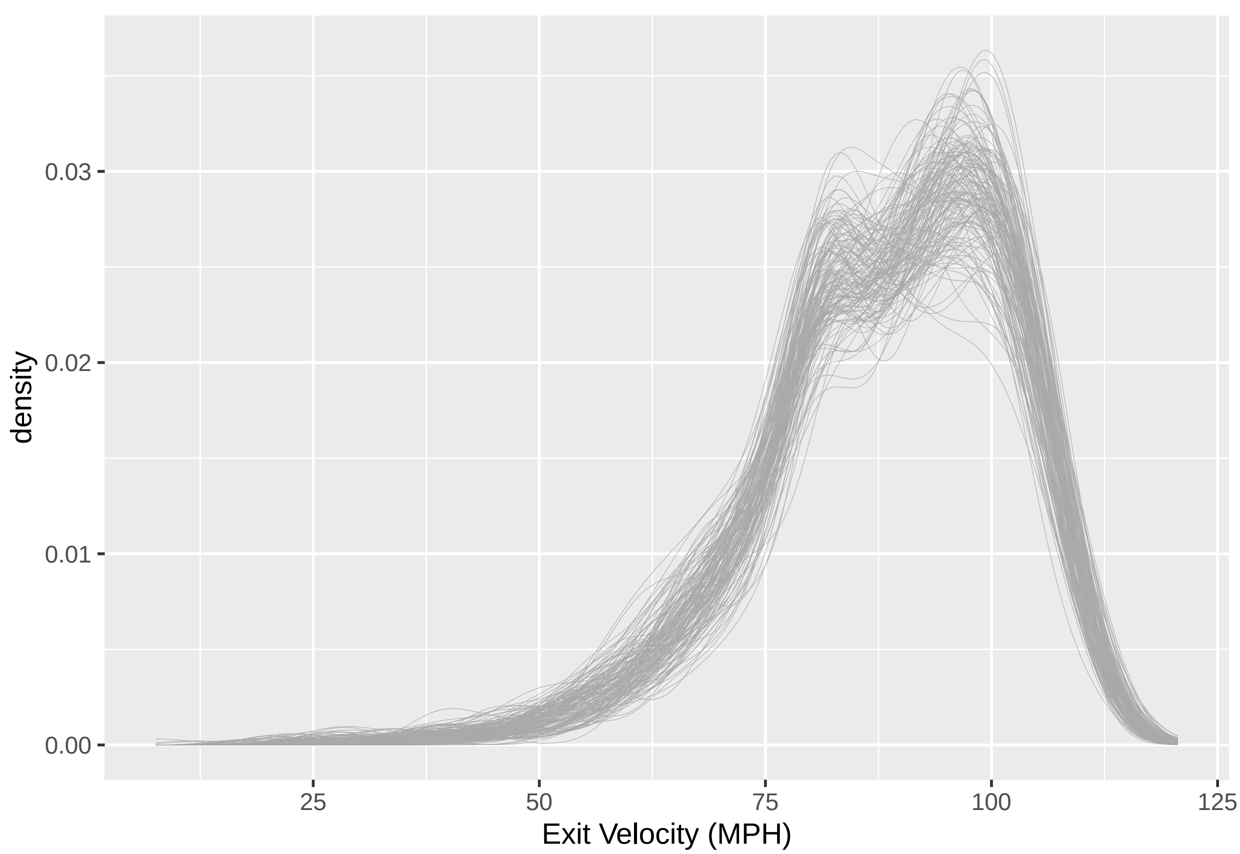
```
## # A tibble: 10 x 2
##    pitcher_name    avg_ev
##    <chr>            <dbl>
##  1 David Hess        91.8
##  2 Tyler Beede       90.9
##  3 Germán Márquez    90.9
##  4 Adrian Sampson    90.8
```

```
##   5 Mike Leake        90.8
##   6 Glenn Sparkman    90.7
##   7 Daniel Norris     90.7
##   8 Shane Bieber      90.6
##   9 Jorge López       90.4
## 10 Vince Velasquez   90.4
```

(d) Plot the distribution of exit velocities allowed across pitchers. Does exit velocity allowed vary significantly across pitchers? Explain your reasoning.

```
sc_bip_small_ = sc_bip_small %>% mutate(year = year(game_date)) %>%
  filter(year == 2019)
sc_regulars_ = sc_bip_small_ %>%
    filter(!is.na(launch_speed)) %>%
    inner_join(pitcher_ev, by = c("pitcher_name"))

ev_plot2 = ggplot(sc_regulars_,
aes(x = launch_speed, group = pitcher_name)) +
geom_density(linewidth = 0.1, color = "darkgray") +
scale_x_continuous("Exit Velocity (MPH)")
ev_plot2
```



Similar to the batter graph, there are multiple peaks. It varies between the two groups but they seem centered on those two peaks. There are a few wild densities however that don't really fit into either peak.

(e) Compute the correlation for exit velocity for batters between the first and second half of the season you chose.

The first half and second half of the season commonly references the All-Star Game and not actually splitting the season into two 81 game halves, so I will use the 2019 All-Star Game (July 9th) as the cutoff.

```
second_half = batters %>%
  filter(game_date > '2019-07-09')
first_half = batters %>%
  filter(game_date < '2019-07-09')
first_half = first_half %>%
  summarise(n = n(),
            avg_ev = mean(exit_velo, na.rm = T)) %>%
  filter(n>100) %>%
  select(-n)
second_half = second_half %>%
  summarise(n = n(),
            avg_ev2 = mean(exit_velo, na.rm = T)) %>%
  filter(n>50) %>%
  select(-n)

halves = first_half %>%
  left_join(second_half) %>%
  filter(avg_ev < 200 & avg_ev > 0 & avg_ev2 < 200 & avg_ev > 0)
view(halves)
cor(halves$avg_ev, halves$avg_ev2)
```

```
## [1] 0.7184131
```

(f) On the basis of your calculations, do you believe exit velocity us a batter skill? Explain.

There is a strong correlation between the average exit velocites for each half of the season. This leads me to believe that this is a skill and is not simply luck.

Question 2 In this question we will try to predict next years slugging percentage using several variables including statcast variables. Load in the data set `sc_bip_small` and run the code below to calculate some possibly important variables that encode launch angle and exit velocity distributional information for each player.

```
foo = sc_bip_small %>%
  mutate(yearID = year(game_date)) %>%
  group_by(batter_name, yearID) %>%
  summarise(N = n(), launch_angle = launch_angle, launch_speed = launch_speed) %>%
  filter(N >= 10) %>%
  summarise(avg_la = mean(launch_angle, na.rm = TRUE),
            sd_la = sd(launch_angle, na.rm = TRUE),
            la10 = quantile(launch_angle, prob = c(0.10), na.rm = TRUE),
            la25 = quantile(launch_angle, prob = c(0.25), na.rm = TRUE),
            la50 = quantile(launch_angle, prob = c(0.50), na.rm = TRUE),
            la75 = quantile(launch_angle, prob = c(0.75), na.rm = TRUE),
            la90 = quantile(launch_angle, prob = c(0.90), na.rm = TRUE),
            avg_ev = mean(launch_speed, na.rm = TRUE),
            sd_ev = sd(launch_speed, na.rm = TRUE),
            ev10 = quantile(launch_speed, prob = c(0.10), na.rm = TRUE),
            ev25 = quantile(launch_speed, prob = c(0.25), na.rm = TRUE),
            ev50 = quantile(launch_speed, prob = c(0.50), na.rm = TRUE),
```

```r
           ev75 = quantile(launch_speed, prob = c(0.75), na.rm = TRUE),
           ev90 = quantile(launch_speed, prob = c(0.90), na.rm = TRUE)) %>%
  rename(name = batter_name)
```

- Create a data frame for batters that contains slugging percentage (SLG) for each player. Call this data frame `bat_stat`. This data frame should contain the following variables: `name`, `yearID`, `teamID`, `AB`, and `SLG`. You can restrict attention to batters who had at least 200 ABs and who only played on a single team (`stint = 1` in the `Batting` data frame in the `Lahman` package).

```r
bat_stat = Batting %>%
  filter(stint == 1 & AB >= 200) %>%
  mutate(X1B = H - X2B - X3B - HR) %>%
  mutate(SLG = round((X1B + 2*X2B + 3*X3B + 4*HR)/AB, 3)) %>%
  left_join(People) %>%
  unite("name", nameFirst, nameLast, sep = " ") %>%
  select(name, yearID, teamID, AB, SLG) %>%
  filter(yearID != 2020)
```

- Merge `foo` into `bat_stat` using `inner_join` or a similar function. Run the following code which creates new variables `SLG_next` and `team_next` which are a player's slugging percentage and team for the next season. The code also creates a categorical variable `COL` which indicates whether the player's next season is with the Rockies. Note that you may have to change the `by` argument in the `inner_join` call below to get it to work.

```r
bat_stat = inner_join(bat_stat, foo, by = c("name", "yearID"))

bar = bat_stat %>% mutate(yearID = ifelse(yearID == 2021, 2020, yearID)) %>%
  mutate(yearID = ifelse(yearID == 2022, 2021, yearID)) %>%
  group_by(name, yearID) %>%
  summarise(SLG, teamID) %>%
  mutate(SLG_next = SLG[match(yearID, yearID-1)]) %>%
  mutate(team_next = teamID[match(yearID, yearID-1)]) %>%
  mutate(yearID = ifelse(yearID == 2021, 2022, yearID)) %>%
  mutate(yearID = ifelse(yearID == 2020, 2021, yearID)) %>%
  select(-SLG,-teamID)

bat_stat = inner_join(bat_stat, bar, by = c("name", "yearID")) %>%
  mutate(COL = ifelse(team_next == "COL",1,0)) %>%
  filter(complete.cases(.))
```

- We are going to use a simple procedure to assess predictive performance. Run the code below to split `bat_stat` into a model training data set `train` and a model testing data set `test`.

```r
set.seed(13)
ind = sample(1:nrow(bat_stat), size = 400, replace = FALSE)
train = bat_stat[ind, ]
test = bat_stat[-ind, ]
```

- Fit and compare the following models. Which model would you select for predicting slugging percentage (root mean squared prediction error is a good metric for assessing predictive performance)? Are statcast variables important for predicting slugging percentage? Explain. Try to find a model which offers better predictive performance than the best model below. Comment on the success of your efforts.

```r
m_big = lm(SLG_next ~ SLG + avg_la + avg_ev + team_next + sd_la + sd_ev +
           sd_la*avg_la + sd_ev*avg_ev +
           la10 + la25 + la50 + la75 + la90 +
           ev10 + ev25 + ev50 + ev75 + ev90,
         data = train)
summary(m_big)
```

```
##
## Call:
## lm(formula = SLG_next ~ SLG + avg_la + avg_ev + team_next + sd_la +
##     sd_ev + sd_la * avg_la + sd_ev * avg_ev + la10 + la25 + la50 +
##     la75 + la90 + ev10 + ev25 + ev50 + ev75 + ev90, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.170384 -0.039232  0.000236  0.034104  0.178585
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.690e-01  1.157e+00   0.405 0.685463
## SLG            2.195e-01  6.235e-02   3.520 0.000488 ***
## avg_la         1.733e-02  1.105e-02   1.568 0.117842
## avg_ev        -1.071e-02  2.335e-02  -0.459 0.646814
## team_nextATL   3.250e-02  2.517e-02   1.291 0.197440
## team_nextBAL  -1.461e-02  2.263e-02  -0.645 0.519048
## team_nextBOS   6.470e-02  2.399e-02   2.697 0.007329 **
## team_nextCHA  -5.256e-03  2.549e-02  -0.206 0.836754
## team_nextCHN  -2.501e-02  2.151e-02  -1.163 0.245612
## team_nextCIN   3.520e-02  2.404e-02   1.464 0.144005
## team_nextCLE  -1.497e-03  2.479e-02  -0.060 0.951900
## team_nextCOL   1.972e-02  2.465e-02   0.800 0.424225
## team_nextDET  -2.291e-02  2.338e-02  -0.980 0.327737
## team_nextHOU   2.719e-02  2.343e-02   1.160 0.246645
## team_nextKCA  -1.079e-02  2.168e-02  -0.498 0.619059
## team_nextLAA  -2.637e-02  2.273e-02  -1.160 0.246933
## team_nextLAN   8.609e-03  2.230e-02   0.386 0.699708
## team_nextMIA  -6.274e-03  2.122e-02  -0.296 0.767622
## team_nextMIL   1.549e-02  2.463e-02   0.629 0.529745
## team_nextMIN   3.561e-02  2.028e-02   1.755 0.080062 .
## team_nextNYA   5.594e-03  2.039e-02   0.274 0.783960
## team_nextNYN   1.996e-02  2.173e-02   0.918 0.359062
## team_nextOAK   1.212e-02  2.264e-02   0.535 0.592770
## team_nextPHI   2.042e-02  2.380e-02   0.858 0.391330
## team_nextPIT   2.264e-02  2.842e-02   0.797 0.426085
## team_nextSDN  -1.306e-03  2.131e-02  -0.061 0.951166
## team_nextSEA   2.224e-04  2.310e-02   0.010 0.992323
## team_nextSFN   4.508e-03  2.280e-02   0.198 0.843371
## team_nextSLN   1.391e-02  2.313e-02   0.601 0.548053
## team_nextTBA  -1.061e-02  2.184e-02  -0.486 0.627550
## team_nextTEX  -6.591e-04  2.257e-02  -0.029 0.976726
## team_nextTOR   7.149e-03  2.133e-02   0.335 0.737735
## team_nextWAS   3.904e-02  2.203e-02   1.772 0.077232 .
## sd_la         -1.979e-03  5.822e-03  -0.340 0.734096
```

```
## sd_ev        -9.204e-02  8.242e-02  -1.117 0.264906
## la10         -1.786e-03  1.624e-03  -1.100 0.272294
## la25         -5.484e-03  1.970e-03  -2.784 0.005662 **
## la50         -1.369e-03  2.935e-03  -0.466 0.641207
## la75          3.592e-03  2.616e-03   1.373 0.170669
## la90         -2.475e-03  1.816e-03  -1.363 0.173641
## ev10          1.971e-04  2.797e-03   0.070 0.943868
## ev25         -1.893e-05  3.982e-03  -0.005 0.996210
## ev50          8.155e-04  6.883e-03   0.118 0.905756
## ev75         -9.450e-03  7.140e-03  -1.323 0.186527
## ev90          1.578e-02  6.300e-03   2.505 0.012684 *
## avg_la:sd_la -3.273e-04  3.629e-04  -0.902 0.367792
## avg_ev:sd_ev  9.694e-04  9.286e-04   1.044 0.297211
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06058 on 353 degrees of freedom
## Multiple R-squared:  0.3912, Adjusted R-squared:  0.3119
## F-statistic: 4.932 on 46 and 353 DF,  p-value: < 2.2e-16
```

```r
sqrt(mean(m_big$residuals^2))
```

```
## [1] 0.05690739
```

```r
m_small = lm(SLG_next ~ SLG + avg_la + avg_ev + COL,
         data = train)
summary(m_small)
```

```
##
## Call:
## lm(formula = SLG_next ~ SLG + avg_la + avg_ev + COL, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.214065 -0.042412 -0.003204  0.035498  0.210221
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.6047670  0.1356063  -4.460 1.07e-05 ***
## SLG          0.2479468  0.0565098   4.388 1.47e-05 ***
## avg_la       0.0003126  0.0007201   0.434    0.664
## avg_ev       0.0103583  0.0016735   6.190 1.51e-09 ***
## COL          0.0040326  0.0202208   0.199    0.842
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06293 on 395 degrees of freedom
## Multiple R-squared:  0.2648, Adjusted R-squared:  0.2574
## F-statistic: 35.57 on 4 and 395 DF,  p-value: < 2.2e-16
```

```r
sqrt(mean(m_small$residuals^2))
```

```
## [1] 0.06253731
```

```r
m_smaller = lm(SLG_next ~ SLG  + COL, data = train)
summary(m_smaller)
```

```
##
## Call:
## lm(formula = SLG_next ~ SLG + COL, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.190885 -0.042238 -0.005746  0.039846  0.217942
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.225258   0.020839  10.809   <2e-16 ***
## SLG         0.451207   0.046323   9.740   <2e-16 ***
## COL         0.002187   0.021086   0.104    0.917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06575 on 397 degrees of freedom
## Multiple R-squared:  0.1935, Adjusted R-squared:  0.1895
## F-statistic: 47.63 on 2 and 397 DF,  p-value: < 2.2e-16
```

```r
sqrt(mean(m_smaller$residuals^2))
```

```
## [1] 0.06550002
```

Of these three models, the "m_big" model has both the highest R^2 value as well as the smallest MSE suggesting that it is the best fit. I would say that the Statcast data is important in predicting next season's SLG percentage.

```r
new_set = train %>%
  select(-c(name, yearID))
m_new = lm(SLG_next ~ ., data = new_set)

step(m_new, direction = "both")
```

```
## Start:  AIC=-2184.12
## SLG_next ~ teamID + AB + SLG + avg_la + sd_la + la10 + la25 +
##     la50 + la75 + la90 + avg_ev + sd_ev + ev10 + ev25 + ev50 +
##     ev75 + ev90 + team_next + COL
##
##
## Step:  AIC=-2184.12
## SLG_next ~ teamID + AB + SLG + avg_la + sd_la + la10 + la25 +
##     la50 + la75 + la90 + avg_ev + sd_ev + ev10 + ev25 + ev50 +
##     ev75 + ev90 + team_next
##
##             Df Sum of Sq    RSS     AIC
## - teamID    29  0.128141 1.2972 -2200.5
## - team_next 29  0.135257 1.3043 -2198.3
```

```
## - ev25       1  0.000149 1.1692 -2186.1
## - ev10       1  0.000203 1.1692 -2186.1
## - avg_ev     1  0.000240 1.1693 -2186.0
## - ev50       1  0.001719 1.1707 -2185.5
## - sd_la      1  0.002330 1.1714 -2185.3
## - la10       1  0.003432 1.1725 -2184.9
## - sd_ev      1  0.003823 1.1728 -2184.8
## - la50       1  0.004866 1.1739 -2184.4
## <none>                  1.1690 -2184.1
## - la75       1  0.006525 1.1756 -2183.9
## - ev75       1  0.007111 1.1761 -2183.7
## - AB         1  0.007357 1.1764 -2183.6
## - avg_la     1  0.009483 1.1785 -2182.9
## - la90       1  0.009819 1.1788 -2182.8
## - la25       1  0.026471 1.1955 -2177.2
## - ev90       1  0.030014 1.1990 -2176.0
## - SLG        1  0.036382 1.2054 -2173.9
##
## Step:  AIC=-2200.51
## SLG_next ~ AB + SLG + avg_la + sd_la + la10 + la25 + la50 + la75 +
##     la90 + avg_ev + sd_ev + ev10 + ev25 + ev50 + ev75 + ev90 +
##     team_next
##
##             Df Sum of Sq    RSS     AIC
## - team_next 29  0.143498 1.4407 -2216.5
## - ev10       1  0.000036 1.2972 -2202.5
## - ev50       1  0.000037 1.2972 -2202.5
## - ev25       1  0.000052 1.2972 -2202.5
## - avg_ev     1  0.000114 1.2973 -2202.5
## - sd_ev      1  0.001397 1.2986 -2202.1
## - la50       1  0.001445 1.2986 -2202.1
## - AB         1  0.005140 1.3023 -2200.9
## - la75       1  0.005272 1.3024 -2200.9
## - sd_la      1  0.005492 1.3027 -2200.8
## - ev75       1  0.005928 1.3031 -2200.7
## - la10       1  0.006295 1.3035 -2200.6
## <none>                  1.2972 -2200.5
## - avg_la     1  0.007355 1.3045 -2200.2
## - la90       1  0.008783 1.3059 -2199.8
## - ev90       1  0.020905 1.3181 -2196.1
## - la25       1  0.026367 1.3235 -2194.5
## - SLG        1  0.040129 1.3373 -2190.3
## + teamID    29  0.128141 1.1690 -2184.1
##
## Step:  AIC=-2216.54
## SLG_next ~ AB + SLG + avg_la + sd_la + la10 + la25 + la50 + la75 +
##     la90 + avg_ev + sd_ev + ev10 + ev25 + ev50 + ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - ev25       1  0.000024 1.4407 -2218.5
## - la50       1  0.000156 1.4408 -2218.5
## - ev10       1  0.000202 1.4409 -2218.5
## - ev50       1  0.000233 1.4409 -2218.5
## - avg_ev     1  0.000291 1.4410 -2218.5
```

```
## - sd_ev      1  0.002124 1.4428 -2217.9
## - la90       1  0.002548 1.4432 -2217.8
## - avg_la     1  0.002665 1.4433 -2217.8
## - la10       1  0.004444 1.4451 -2217.3
## - ev75       1  0.005466 1.4461 -2217.0
## - AB         1  0.007102 1.4478 -2216.6
## <none>                   1.4407 -2216.5
## - la75       1  0.008254 1.4489 -2216.3
## - sd_la      1  0.010754 1.4514 -2215.6
## + COL        1  0.001228 1.4394 -2214.9
## - ev90       1  0.016786 1.4574 -2213.9
## - la25       1  0.022683 1.4633 -2212.3
## - SLG        1  0.059601 1.5003 -2202.3
## + team_next 29  0.143498 1.2972 -2200.5
## + teamID    29  0.136382 1.3043 -2198.3
##
## Step:  AIC=-2218.54
## SLG_next ~ AB + SLG + avg_la + sd_la + la10 + la25 + la50 + la75 +
##     la90 + avg_ev + sd_ev + ev10 + ev50 + ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - la50       1  0.000171 1.4409 -2220.5
## - ev10       1  0.000187 1.4409 -2220.5
## - ev50       1  0.000321 1.4410 -2220.4
## - avg_ev     1  0.000330 1.4410 -2220.4
## - la90       1  0.002606 1.4433 -2219.8
## - avg_la     1  0.002703 1.4434 -2219.8
## - sd_ev      1  0.002938 1.4436 -2219.7
## - la10       1  0.004449 1.4451 -2219.3
## - ev75       1  0.006907 1.4476 -2218.6
## - AB         1  0.007078 1.4478 -2218.6
## <none>                   1.4407 -2218.5
## - la75       1  0.008244 1.4489 -2218.2
## - sd_la      1  0.010809 1.4515 -2217.6
## + COL        1  0.001234 1.4395 -2216.9
## + ev25       1  0.000024 1.4407 -2216.5
## - ev90       1  0.021586 1.4623 -2214.6
## - la25       1  0.023068 1.4638 -2214.2
## - SLG        1  0.059601 1.5003 -2204.3
## + team_next 29  0.143470 1.2972 -2202.5
## + teamID    29  0.136392 1.3043 -2200.3
##
## Step:  AIC=-2220.49
## SLG_next ~ AB + SLG + avg_la + sd_la + la10 + la25 + la75 + la90 +
##     avg_ev + sd_ev + ev10 + ev50 + ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - ev10       1  0.000176 1.4410 -2222.4
## - avg_ev     1  0.000290 1.4411 -2222.4
## - ev50       1  0.000338 1.4412 -2222.4
## - la90       1  0.002510 1.4434 -2221.8
## - sd_ev      1  0.003249 1.4441 -2221.6
## - avg_la     1  0.003290 1.4442 -2221.6
## - la10       1  0.004290 1.4451 -2221.3
```

```
## - ev75       1  0.006783 1.4477 -2220.6
## - AB         1  0.007069 1.4479 -2220.5
## <none>                   1.4409 -2220.5
## - la75       1  0.008557 1.4494 -2220.1
## - sd_la      1  0.011658 1.4525 -2219.3
## + COL        1  0.001308 1.4396 -2218.8
## + la50       1  0.000171 1.4407 -2218.5
## + ev25       1  0.000039 1.4408 -2218.5
## - ev90       1  0.022464 1.4633 -2216.3
## - la25       1  0.022974 1.4638 -2216.2
## - SLG        1  0.059438 1.5003 -2206.3
## + team_next 29  0.142138 1.2987 -2204.0
## + teamID    29  0.135966 1.3049 -2202.1
##
## Step:  AIC=-2222.44
## SLG_next ~ AB + SLG + avg_la + sd_la + la10 + la25 + la75 + la90 +
##     avg_ev + sd_ev + ev50 + ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - avg_ev     1  0.000139 1.4412 -2224.4
## - ev50       1  0.000614 1.4417 -2224.3
## - la90       1  0.002595 1.4436 -2223.7
## - sd_ev      1  0.003142 1.4442 -2223.6
## - avg_la     1  0.003317 1.4444 -2223.5
## - la10       1  0.004456 1.4455 -2223.2
## - ev75       1  0.006619 1.4477 -2222.6
## - AB         1  0.007120 1.4482 -2222.5
## <none>                   1.4410 -2222.4
## - la75       1  0.008457 1.4495 -2222.1
## - sd_la      1  0.011610 1.4526 -2221.2
## + COL        1  0.001255 1.4398 -2220.8
## + ev10       1  0.000176 1.4409 -2220.5
## + la50       1  0.000159 1.4409 -2220.5
## + ev25       1  0.000020 1.4410 -2220.4
## - la25       1  0.022844 1.4639 -2218.2
## - ev90       1  0.024576 1.4656 -2217.7
## - SLG        1  0.059625 1.5007 -2208.2
## + team_next 29  0.142258 1.2988 -2206.0
## + teamID    29  0.135639 1.3054 -2204.0
##
## Step:  AIC=-2224.4
## SLG_next ~ AB + SLG + avg_la + sd_la + la10 + la25 + la75 + la90 +
##     sd_ev + ev50 + ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - la90       1  0.002496 1.4437 -2225.7
## - avg_la     1  0.003209 1.4444 -2225.5
## - la10       1  0.004350 1.4455 -2225.2
## - ev50       1  0.004567 1.4458 -2225.1
## - AB         1  0.007169 1.4484 -2224.4
## <none>                   1.4412 -2224.4
## - ev75       1  0.007442 1.4486 -2224.3
## - la75       1  0.008715 1.4499 -2224.0
## - sd_la      1  0.011740 1.4529 -2223.2
```

```
## + COL        1  0.001249 1.4399 -2222.8
## + avg_ev     1  0.000139 1.4410 -2222.4
## + la50       1  0.000133 1.4410 -2222.4
## + ev10       1  0.000025 1.4411 -2222.4
## + ev25       1  0.000022 1.4412 -2222.4
## - la25       1  0.022898 1.4641 -2220.1
## - sd_ev      1  0.025429 1.4666 -2219.4
## - ev90       1  0.050488 1.4917 -2212.6
## - SLG        1  0.060710 1.5019 -2209.9
## + team_next 29  0.142290 1.2989 -2208.0
## + teamID    29  0.135765 1.3054 -2206.0
##
## Step:  AIC=-2225.71
## SLG_next ~ AB + SLG + avg_la + sd_la + la10 + la25 + la75 + sd_ev +
##     ev50 + ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - avg_la     1  0.001079 1.4447 -2227.4
## - la10       1  0.005069 1.4487 -2226.3
## - ev50       1  0.005509 1.4492 -2226.2
## - AB         1  0.006754 1.4504 -2225.8
## <none>                   1.4437 -2225.7
## - ev75       1  0.007753 1.4514 -2225.6
## + la90       1  0.002496 1.4412 -2224.4
## - la75       1  0.012367 1.4560 -2224.3
## + COL        1  0.001352 1.4423 -2224.1
## + ev10       1  0.000105 1.4436 -2223.7
## + la50       1  0.000095 1.4436 -2223.7
## + avg_ev     1  0.000040 1.4436 -2223.7
## + ev25       1  0.000000 1.4437 -2223.7
## - la25       1  0.020416 1.4641 -2222.1
## - sd_ev      1  0.023448 1.4671 -2221.3
## - sd_la      1  0.041417 1.4851 -2216.4
## - ev90       1  0.049365 1.4930 -2214.3
## - SLG        1  0.063287 1.5070 -2210.6
## + team_next 29  0.137525 1.3061 -2207.8
## + teamID    29  0.135562 1.3081 -2207.2
##
## Step:  AIC=-2227.41
## SLG_next ~ AB + SLG + sd_la + la10 + la25 + la75 + sd_ev + ev50 +
##     ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - la10       1  0.003990 1.4487 -2228.3
## - ev50       1  0.004663 1.4494 -2228.1
## - AB         1  0.006760 1.4515 -2227.5
## <none>                   1.4447 -2227.4
## - ev75       1  0.007927 1.4527 -2227.2
## + COL        1  0.001135 1.4436 -2225.7
## + avg_la     1  0.001079 1.4437 -2225.7
## + la50       1  0.000699 1.4441 -2225.6
## + la90       1  0.000366 1.4444 -2225.5
## + ev10       1  0.000111 1.4446 -2225.4
## + avg_ev     1  0.000020 1.4447 -2225.4
```

```
## + ev25       1  0.000000 1.4447 -2225.4
## - sd_ev      1  0.026445 1.4712 -2222.2
## - la25       1  0.028078 1.4728 -2221.7
## - sd_la      1  0.040384 1.4851 -2218.4
## - la75       1  0.042478 1.4872 -2217.8
## - ev90       1  0.052699 1.4974 -2215.1
## - SLG        1  0.067125 1.5119 -2211.2
## + team_next 29  0.137510 1.3072 -2209.4
## + teamID    29  0.136386 1.3084 -2209.1
##
## Step:  AIC=-2228.31
## SLG_next ~ AB + SLG + sd_la + la25 + la75 + sd_ev + ev50 + ev75 +
##     ev90
##
##             Df Sum of Sq    RSS     AIC
## - ev50       1  0.004885 1.4536 -2229.0
## - AB         1  0.007182 1.4559 -2228.3
## <none>                   1.4487 -2228.3
## - ev75       1  0.008044 1.4568 -2228.1
## + la10       1  0.003990 1.4447 -2227.4
## + la90       1  0.001741 1.4470 -2226.8
## + COL        1  0.000975 1.4478 -2226.6
## + la50       1  0.000569 1.4482 -2226.5
## + ev10       1  0.000363 1.4484 -2226.4
## + ev25       1  0.000015 1.4487 -2226.3
## + avg_ev     1  0.000001 1.4487 -2226.3
## + avg_la     1  0.000000 1.4487 -2226.3
## - sd_ev      1  0.024997 1.4737 -2223.5
## - la25       1  0.034090 1.4828 -2221.0
## - la75       1  0.039419 1.4882 -2219.6
## - sd_la      1  0.040412 1.4892 -2219.3
## - ev90       1  0.051962 1.5007 -2216.2
## - SLG        1  0.067684 1.5164 -2212.0
## + team_next 29  0.136186 1.3126 -2209.8
## + teamID    29  0.134664 1.3141 -2209.3
##
## Step:  AIC=-2228.96
## SLG_next ~ AB + SLG + sd_la + la25 + la75 + sd_ev + ev75 + ev90
##
##             Df Sum of Sq    RSS     AIC
## - ev75       1  0.003375 1.4570 -2230.0
## <none>                   1.4536 -2229.0
## - AB         1  0.007365 1.4610 -2228.9
## + ev50       1  0.004885 1.4487 -2228.3
## + la10       1  0.004213 1.4494 -2228.1
## + avg_ev     1  0.003577 1.4501 -2227.9
## + la90       1  0.003266 1.4504 -2227.9
## + ev25       1  0.001249 1.4524 -2227.3
## + COL        1  0.000781 1.4528 -2227.2
## + la50       1  0.000495 1.4531 -2227.1
## + avg_la     1  0.000263 1.4534 -2227.0
## + ev10       1  0.000004 1.4536 -2227.0
## - la25       1  0.030232 1.4839 -2222.7
## - la75       1  0.035453 1.4891 -2221.3
```

```
## - sd_ev      1  0.037343 1.4910 -2220.8
## - sd_la      1  0.040856 1.4945 -2219.9
## - ev90       1  0.047471 1.5011 -2218.1
## - SLG        1  0.070821 1.5245 -2211.9
## + team_next 29  0.139525 1.3141 -2211.3
## + teamID    29  0.134757 1.3189 -2209.9
##
## Step:  AIC=-2230.03
## SLG_next ~ AB + SLG + sd_la + la25 + la75 + sd_ev + ev90
##
##             Df Sum of Sq    RSS     AIC
## <none>                   1.4570 -2230.0
## - AB         1  0.008488 1.4655 -2229.7
## + la10       1  0.004151 1.4528 -2229.2
## + ev75       1  0.003375 1.4536 -2229.0
## + la90       1  0.002053 1.4549 -2228.6
## + COL        1  0.000743 1.4563 -2228.2
## + la50       1  0.000738 1.4563 -2228.2
## + ev10       1  0.000401 1.4566 -2228.1
## + ev50       1  0.000216 1.4568 -2228.1
## + ev25       1  0.000200 1.4568 -2228.1
## + avg_ev     1  0.000011 1.4570 -2228.0
## + avg_la     1  0.000010 1.4570 -2228.0
## - sd_ev      1  0.033968 1.4910 -2222.8
## - la25       1  0.034956 1.4920 -2222.6
## - la75       1  0.040771 1.4978 -2221.0
## - sd_la      1  0.042078 1.4991 -2220.6
## - SLG        1  0.067470 1.5245 -2213.9
## + team_next 29  0.134158 1.3229 -2210.7
## + teamID    29  0.132511 1.3245 -2210.2
## - ev90       1  0.183593 1.6406 -2184.6
```

```
##
## Call:
## lm(formula = SLG_next ~ AB + SLG + sd_la + la25 + la75 + sd_ev +
##     ev90, data = new_set)
##
## Coefficients:
## (Intercept)           AB          SLG        sd_la         la25         la75
##  -5.759e-01    4.058e-05    2.483e-01   -7.533e-03   -3.632e-03    4.489e-03
##       sd_ev         ev90
##  -8.770e-03    1.003e-02
```

```r
m_newer = lm(SLG_next ~ AB + SLG + sd_la + la75 + sd_ev + ev90, data = new_set)
summary(m_newer)
```

```
##
## Call:
## lm(formula = SLG_next ~ AB + SLG + sd_la + la75 + sd_ev + ev90,
##     data = new_set)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.192688 -0.040817 -0.001906  0.033137  0.208478
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.821e-01  1.251e-01  -4.652 4.51e-06 ***
## AB           4.945e-05  2.698e-05   1.833 0.067562 .
## SLG          1.996e-01  5.668e-02   3.522 0.000478 ***
## sd_la       -3.290e-03  1.779e-03  -1.849 0.065144 .
## la75         8.940e-04  6.875e-04   1.300 0.194272
## sd_ev       -9.042e-03  2.931e-03  -3.085 0.002177 **
## ev90         1.042e-02  1.436e-03   7.257 2.14e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06161 on 393 degrees of freedom
## Multiple R-squared:  0.2989, Adjusted R-squared:  0.2881
## F-statistic: 27.92 on 6 and 393 DF,  p-value: < 2.2e-16
```

```r
mean(m_newer$residuals^2)
```

```
## [1] 0.003729901
```

```r
AIC(m_big)
```

```
## [1] -1061.913
```

```r
AIC(m_newer)
```

```
## [1] -1085.399
```

I wasn't able to find a model better than "m_big" in terms of predictive performance, however the AIC is lower in my new model which could suggest a better overall fit.

Question 3 The 2021 San Francisco Giants certainly surprised a lot of people when they won 107 games with a rotation led by Kevin Gausman, Logan Webb, Anthony DeSclafani, and Alex Wood. Coming into the 2021 season, I think it is fair to say that this is a shaky rotation. One commentator said that the Giants have developed reputation as an organization that can make players better, but that reputation will be tested with a risky experiment in 2021. Let's investigate the success of the 2021 San Francisco Giants. In this question we will look at the 2021 San Francisco Giants pitching staff from a recent historical perspective. In the next question we will examine specific Giants pitchers. As an aside, anyone can go down the rabbit hole that your professor went down as this problem was developed:

- Logan Webb, As Advertised
- What the Heck Is a Flat Sinker, Anyway?
- The Giants Took a New Angle With Sinkers
- The Seam-Shifted Revolution Is Headed for the Mainstream
- Pitch Movement, Spin Efficiency, and All That
- Prospectus Feature: All Spin Is Not Alike
- Determining the 3D Spin Axis from Statcast Data

First create a data frame that contains the following variables on team pitching statistics: `yearID`, `teamID`, `frac_junk`, `ERA`, `HAp9`, `HRAp9`, and `WAR`. This data frame only needs to be created for the 2017, 2018, 2019, and 2021 baseball seasons. The variables `HAp9` and `HRAp9` are, respectively, hits allowed per 9 innings and home runs allowed per 9 innings. The variable `frac_junk` is the fraction of team pitches that are sinkers (SI), splitters (FS), sliders (SL), or change ups (CH) for a given season. Calculation of `frac_junk` involves the use of all statcast data for the 2017, 2018, 2019, and 2021 baseball seasons. The statcast data set is massive and a Rmd document might not compile if this data set is directly loaded in and manipulated. I recommend that you first perform your data manipulations to statcast data in an active R session, then save a much smaller data set which contains your data manipulations onto your computer, then load this smaller data set into the Rmd document corresponding to your lab assignment. The code below obtains pitching WAR.

```
bwar_pit =
  readr::read_csv("https://www.baseball-reference.com/data/war_daily_pitch.txt",
                  na = "NULL") %>%
    filter(year_ID >= 2017) %>%
    select(team_ID, year_ID, WAR) %>%
    rename(teamID = team_ID, yearID = year_ID)

bwar_pit = bwar_pit %>%
  filter(WAR > -100 & WAR < 100) %>%
  group_by(teamID, yearID) %>%
  summarise(WAR = sum(WAR, na.rm = T)) %>%
  filter(yearID >= 2017 & yearID <= 2021 & yearID != 2020)


#Creating non-Statcast dataset
set = Pitching %>%
  filter(yearID >= 2017 & yearID <= 2021 & yearID != 2020) %>%
  group_by(teamID, yearID) %>%
  summarise(IPouts = sum(IPouts),
            HR = sum(HR),
            H = sum(H),
            ER = sum(ER)) %>%
  mutate(ERA = round((ER*27)/IPouts, 3),
         HAp9 = round(H/(IPouts/27),3),
         HRAp9 = round(HR/(IPouts/27),3)) %>%
  select(yearID, teamID, ERA, HRAp9, HAp9) %>%
  mutate(teamID = as.character(teamID)) %>%
  mutate(teamID = replace(teamID, teamID == "CHA", "CHW")) %>%
  mutate(teamID = replace(teamID, teamID == "CHN", "CHC")) %>%
  mutate(teamID = replace(teamID, teamID == "KCA", "KCR")) %>%
  mutate(teamID = replace(teamID, teamID == "LAN", "LAD")) %>%
  mutate(teamID = replace(teamID, teamID == "NYN", "NYM")) %>%
  mutate(teamID = replace(teamID, teamID == "NYA", "NYY")) %>%
  mutate(teamID = replace(teamID, teamID == "SDN", "SDP")) %>%
  mutate(teamID = replace(teamID, teamID == "SFN", "SFG")) %>%
  mutate(teamID = replace(teamID, teamID == "SLN", "STL")) %>%
  mutate(teamID = replace(teamID, teamID == "TBA", "TBR")) %>%
  mutate(teamID = replace(teamID, teamID == "WAS", "WSN"))


set = bwar_pit %>%
  left_join(set)

road = sc %>%
```

```
  mutate(yearID = year(game_date)) %>%
  select(pitch_type, inning_topbot, home_team, away_team, yearID) %>%
  filter(inning_topbot == "Bot") %>%
  select(pitch_type, away_team, yearID) %>%
  mutate(junk = as.integer(pitch_type == "SL" | pitch_type == "SI" | pitch_type == "FS" | pitch_type ==
  mutate(team = away_team) %>%
  group_by(team, yearID) %>%
  summarise(junk = sum(junk, na.rm = T), N = n())

home = sc %>%
  mutate(yearID = year(game_date)) %>%
  select(pitch_type, inning_topbot, home_team, away_team, yearID) %>%
  filter(inning_topbot == "Top") %>%
  select(pitch_type, home_team, yearID) %>%
  mutate(junk = as.integer(pitch_type == "SL" | pitch_type == "SI" | pitch_type == "FS" | pitch_type ==
  mutate(team = home_team) %>%
  group_by(team) %>%
  summarise(junk1 = sum(junk, na.rm = T), N1 = n())

junk = home %>%
  left_join(road) %>%
  mutate(frac_junk = (junk1 + junk) / (N1+N),
         teamID = team) %>%
  filter(yearID != 2022) %>%
  select(teamID, yearID, frac_junk) %>%
  mutate(teamID = replace(teamID, teamID == "AZ", "ARI")) %>%
  mutate(teamID = replace(teamID, teamID == "CWS", "CHW")) %>%
  mutate(teamID = replace(teamID, teamID == "KC", "KCR")) %>%
  mutate(teamID = replace(teamID, teamID == "SD", "SDP")) %>%
  mutate(teamID = replace(teamID, teamID == "SF", "SFG")) %>%
  mutate(teamID = replace(teamID, teamID == "TB", "TBR")) %>%
  mutate(teamID = replace(teamID, teamID == "WSH", "WSN"))

q3_data = set %>%
  left_join(junk) %>%
  select(yearID, teamID, frac_junk, ERA, HAp9, HRAp9, WAR)
```

Use the data set that you created to study the 2021 pitching season. Were the 2021 Giants successful? Did they perform similarly to other teams that throw a lot of junk balls where junk is defined as sinkers, splitters, sliders, and change ups? Elaborate.

```
frac_SFG = q3_data %>%
  filter(yearID == 2021 & teamID == "SFG") %>%
  pull(frac_junk)

frac_sd = sqrt(var(q3_data$frac_junk))

new = q3_data %>%
  mutate(close = abs(frac_junk - frac_SFG)) %>%
  arrange(close) %>%
  head(15) %>%
  select(-close) %>%
  arrange(desc(WAR))
```

```r
new
```

```
## # A tibble: 15 x 7
## # Groups:   teamID [8]
##    yearID teamID frac_junk   ERA  HAp9 HRAp9   WAR
##     <dbl> <chr>      <dbl> <dbl> <dbl> <dbl> <dbl>
##  1   2021 SFG        0.492  3.25  7.76 0.934 26.7
##  2   2021 TOR        0.501  3.91  8.05 1.34  21.4
##  3   2019 CIN        0.485  4.18  7.95 1.34  21.2
##  4   2017 KCR        0.494  4.63  9.26 1.23  18.5
##  5   2021 MIA        0.492  3.96  8.15 1.03  15.3
##  6   2018 TEX        0.482  4.92  9.54 1.40  14.6
##  7   2017 TOR        0.499  4.42  8.97 1.25  14.2
##  8   2018 LAA        0.498  4.15  8.47 1.28  14.0
##  9   2021 LAA        0.494  4.69  8.69 1.19  13.8
## 10   2017 DET        0.501  5.36 10.1   1.38  12.9
## 11   2021 KCR        0.485  4.64  8.73 1.2    9.56
## 12   2019 LAA        0.487  5.12  8.84 1.67   8.98
## 13   2018 CIN        0.501  4.65  9.31 1.42   8.06
## 14   2018 KCR        0.485  4.95  9.69 1.29   7.94
## 15   2019 TOR        0.492  4.79  9.06 1.42   7.77
```

```r
sqrt(var(new$WAR))
```

```
## [1] 5.644536
```

Yes, the 2021 Giants were very successful. The number I included above is the standard deviation for pitching WAR among the 14 teams closest to the 2021 Giants in terms of `frac_junk`. The 2021 Giants were almost one full standard deviation better in WAR than the next closest team. In every single variable measured above, the 2021 Giants rank the best. I would say they were very successful compared to the other teams that threw a similar fraction of "junk."

Question 4 For each pitch type thrown by Kevin Gausman, Logan Webb, Anthony DeSclafani, and Alex Wood, compute annual averages of `release_spin_rate`, `effective_speed`, `plate_x`, `plate_z`, `pfx_x`, `pfx_z`, `release_x`, and `release_z`, and compute the annual pitch type percentages for each of these pitchers. Now display a graphic showing how these annual averages change over time for each of these pitchers. It is best to display all nine plots for each pitcher in a single grid of plots rather than printing off nine separate plots. This can be achieved using the `grid.arrange` function in the `gridExtra` package. Comment on how the approach of these pitchers changed over time with an emphasis on any changes made in 2021. Comment on any commonalities or differences between these pitchers. What are some of the reasons for the pitching success of 2021 San Francisco Giants pitchers?

```r
library(gridExtra)
percentages = sc %>%
  filter(pitcher_name == "Kevin Gausman" | pitcher_name == "Logan Webb" | pitcher_name == "Anthony DeScl
  mutate(Name = pitcher_name,
         Year = year(game_date)) %>%
  select(Name, Year, pitch_type) %>%
  mutate(SL = as.integer(pitch_type == "SL"),
         FF = as.integer(pitch_type == "FF"),
         FS = as.integer(pitch_type == "FS"),
         CH = as.integer(pitch_type == "CH"),
```

```r
        SI = as.integer(pitch_type == "SI"),
        KC = as.integer(pitch_type == "KC"),
        PO = as.integer(pitch_type == "PO"),
        CU = as.integer(pitch_type == "CU"),
        FC = as.integer(pitch_type == "FC"),
        Unknown = as.integer(is.na(pitch_type))) %>%
  group_by(Name, Year) %>%
  summarise(SL = sum(SL, na.rm = T),
            FF = sum(FF, na.rm = T),
            FS = sum(FS, na.rm = T),
            CH = sum(CH, na.rm = T),
            SI = sum(SI, na.rm = T),
            KC = sum(KC, na.rm = T),
            PO = sum(PO, na.rm = T),
            CU = sum(CU, na.rm = T),
            FC = sum(FC, na.rm = T),
            Unknown = sum(Unknown, na.rm = T),
            N = n()) %>%
  mutate(SLpct = round(SL/N,2),
         FFpct = round(FF/N,2),
         FSpct = round(FS/N,2),
         CHpct = round(CH/N,2),
         SIpct = round(SI/N,2),
         KCpct = round(KC/N,2),
         POpct = round(PO/N,2),
         CUpct = round(CU/N,2),
         FCpct = round(FC/N,2),
         NApct = round(Unknown/N, 2)) %>%
  select(Year, Name, SLpct:NApct)


averages = sc %>%
  filter(pitcher_name == "Kevin Gausman" | pitcher_name == "Logan Webb" | pitcher_name == "Anthony DeScl
  mutate(Name = pitcher_name,
         Year = year(game_date)) %>%
  select(Year, Name, release_spin_rate, effective_speed, plate_x, plate_z, pfx_x, pfx_z, release_pos_x,
  group_by(Name, Year) %>%
  summarise(release_spin_rate = mean(release_spin_rate, na.rm = T),
            effective_speed = mean(effective_speed, na.rm = T),
            plate_x = mean(plate_x, na.rm = T),
            plate_z = mean(plate_z, na.rm = T),
            pfx_x = mean(pfx_x, na.rm = T),
            pfx_z = mean(pfx_z, na.rm = T),
            release_pos_x = mean(release_pos_x, na.rm = T),
            release_pos_z = mean(release_pos_z, na.rm = T))
averages %>%
  left_join(percentages)
```

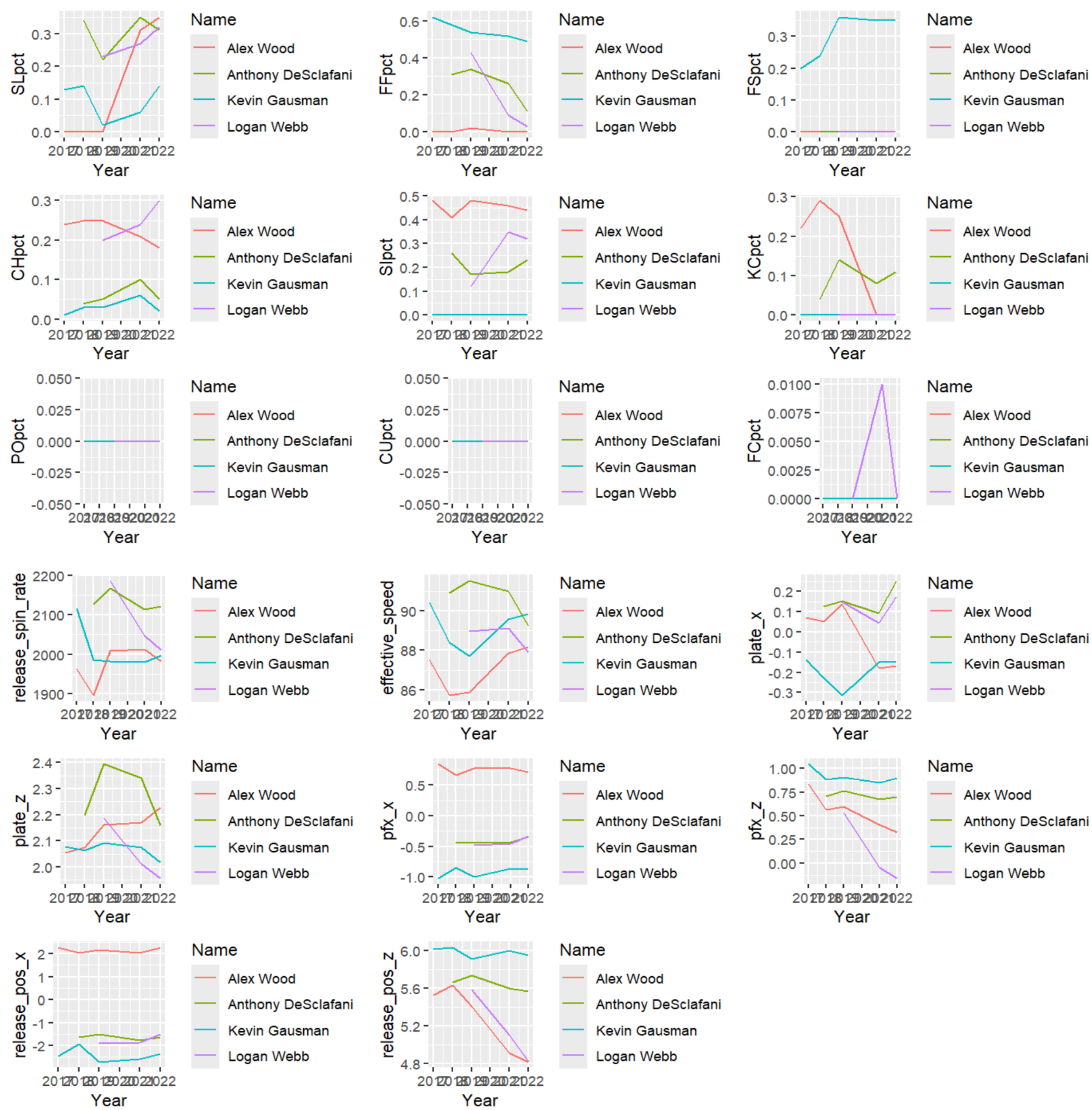
```
## # A tibble: 17 x 20
## # Groups:   Name [4]
##    Name    Year release_spin_rate effective_speed plate_x plate_z  pfx_x   pfx_z
##    <chr>  <dbl>             <dbl>           <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
##  1 Alex ~  2017             1962.            87.5  0.0711    2.05  0.842  0.837
```

```
##   2 Alex ~  2018             1896.             85.7  0.0513    2.07  0.659  0.567
##   3 Alex ~  2019             2008.             85.9  0.139     2.16  0.770  0.600
##   4 Alex ~  2021             2012.             87.9 -0.178     2.17  0.773  0.405
##   5 Alex ~  2022             1981.             88.2 -0.168     2.23  0.716  0.323
##   6 Antho~  2018             2127.             90.9  0.127     2.20 -0.430  0.711
##   7 Antho~  2019             2167.             91.5  0.154     2.39 -0.449  0.762
##   8 Antho~  2021             2113.             91.0  0.0934    2.34 -0.427  0.680
##   9 Antho~  2022             2122.             89.2  0.253     2.16 -0.342  0.698
##  10 Kevin~  2017             2115.             90.4 -0.138     2.08 -1.02   1.05
##  11 Kevin~  2018             1986.             88.4 -0.227     2.06 -0.845  0.884
##  12 Kevin~  2019             1981.             87.7 -0.313     2.09 -0.991  0.904
##  13 Kevin~  2021             1980.             89.6 -0.151     2.07 -0.867  0.849
##  14 Kevin~  2022             1996.             89.8 -0.145     2.02 -0.878  0.897
##  15 Logan~  2019             2186.             89.0  0.151     2.19 -0.485  0.530
##  16 Logan~  2021             2048.             89.1  0.0443    2.01 -0.456 -0.0524
##  17 Logan~  2022             2011.             87.9  0.177     1.96 -0.338 -0.163
## # i 12 more variables: release_pos_x <dbl>, release_pos_z <dbl>, SLpct <dbl>,
## #   FFpct <dbl>, FSpct <dbl>, CHpct <dbl>, SIpct <dbl>, KCpct <dbl>,
## #   POpct <dbl>, CUpct <dbl>, FCpct <dbl>, NApct <dbl>
```

```r
SL_graph = ggplot(percentages, aes(x = Year, y = SLpct, color = Name))+
  geom_line()
FF_graph = ggplot(percentages, aes(x = Year, y = FFpct, color = Name))+
  geom_line()
FS_graph = ggplot(percentages, aes(x = Year, y = FSpct, color = Name))+
  geom_line()
CH_graph = ggplot(percentages, aes(x = Year, y = CHpct, color = Name))+
  geom_line()
SI_graph = ggplot(percentages, aes(x = Year, y = SIpct, color = Name))+
  geom_line()
KC_graph = ggplot(percentages, aes(x = Year, y = KCpct, color = Name))+
  geom_line()
PO_graph = ggplot(percentages, aes(x = Year, y = POpct, color = Name))+
  geom_line()
CU_graph = ggplot(percentages, aes(x = Year, y = CUpct, color = Name))+
  geom_line()
FC_graph = ggplot(percentages, aes(x = Year, y = FCpct, color = Name))+
  geom_line()

spin = ggplot(averages, aes(x = Year, y = release_spin_rate, color = Name))+
  geom_line()
ef_speed = ggplot(averages, aes(x = Year, y = effective_speed, color = Name))+
  geom_line()
p_x = ggplot(averages, aes(x = Year, y = plate_x, color = Name))+
  geom_line()
p_z = ggplot(averages, aes(x = Year, y = plate_z, color = Name))+
  geom_line()
pfx_x = ggplot(averages, aes(x = Year, y = pfx_x, color = Name))+
  geom_line()
pfx_z = ggplot(averages, aes(x = Year, y = pfx_z, color = Name))+
  geom_line()
r_x = ggplot(averages, aes(x = Year, y = release_pos_x, color = Name))+
  geom_line()
r_z = ggplot(averages, aes(x = Year, y = release_pos_z, color = Name))+
```

```r
geom_line()
```



Looking at these graphs (I had to include screenshots due to the R code not displaying in a visible way), the first thing that is very obvious is that Logan Webb developed a cutter (FC) that he used in 2021. However, that is used at such a low percentage that the graph is misleading. Across the four pitchers, they all used thier sliders (SL) more frequently than in the past, significantly more for Wood and DeSclafani. They all also threw less four-seam fastballs (FF) and knuckle-curveballs (KC).

It seems like the biggest change from the average stats is that release spin rate is a little down for Webb and DeSclafani. It looks like the Giants were trying to maybe get all of their pitchers to a certain range of the `plate_x` variable with a couple pitchers going down in 2021, and a couple went up

I think a big reason for the pitching success is that these pitchers changed their arsenal of pitches, and hitters weren't ready for it. They went very slider heavy as a staff, and it appears they found four good sliders.

The biggest visual change among the other graph is the spin rate graph. I think that the change in the spin rate for the 2021 season confused hitters.