Lab 1

Sam Burch

2023-02-04

Question 1

```
library(Lahman)
```

```
## Warning: package 'Lahman' was built under R version 4.2.2
```

```
library(broom)
```

```
## Warning: package 'broom' was built under R version 4.2.2
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
Teams = as_tibble(Teams)

df_1 = Teams |>
  select(yearID, W, L, G, AB, H, X2B, X3B, HR, BB, HBP, SF, HA, HRA,
         BBA, SOA, IPouts, FP, R, RA) |>
  filter(yearID >= 1900) |>
  replace_na(list(HBP =0, SF = 0)) |>
  mutate(RD = (R - RA) / G,
         X1B = H - (X2B + X3B + HR)) |>
  mutate(OBP = (H + BB + HBP) / (AB + BB + HBP + SF)) |>
  mutate(SLG = (X1B + 2*X2B + 3*X3B + 4*HR) / AB) |>
  mutate(OPS = OBP + SLG) |>
  mutate(IP = IPouts / 3) |>
  mutate(WHIP = (BBA + HA) / IP) |>
  mutate(FIP = (13*HRA + 3*BBA - 2*SOA) / IP)
df_1
```

```
## # A tibble: 2,610 x 28
##    yearID     W     L     G    AB     H   X2B   X3B    HR    BB   HBP    SF
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
##  1   1900    82    54   141  4860  1423   199    81    26   421    81     0
##  2   1900    66    72   142  4952  1403   163    68    48   395    45     0
##  3   1900    65    75   146  4907  1276   202    51    33   343    65     0
##  4   1900    62    77   144  5026  1335   178    83    33   333    50     0
##  5   1900    60    78   141  4724  1317   177    61    23   369    56     0
##  6   1900    75    63   141  4969  1439   187    82    29   440    72     0
##  7   1900    79    60   140  4817  1312   185   100    26   327    63     0
##  8   1900    65    75   141  4877  1420   141    81    36   406    81     0
##  9   1901    68    65   134  4589  1348   179   111    24   369    52     0
## 10   1901    79    57   138  4866  1353   183   104    37   331    47     0
## # ... with 2,600 more rows, and 16 more variables: HA <int>, HRA <int>,
## #   BBA <int>, SOA <int>, IPouts <int>, FP <dbl>, R <int>, RA <int>, RD <dbl>,
## #   X1B <int>, OBP <dbl>, SLG <dbl>, OPS <dbl>, IP <dbl>, WHIP <dbl>, FIP <dbl>
```

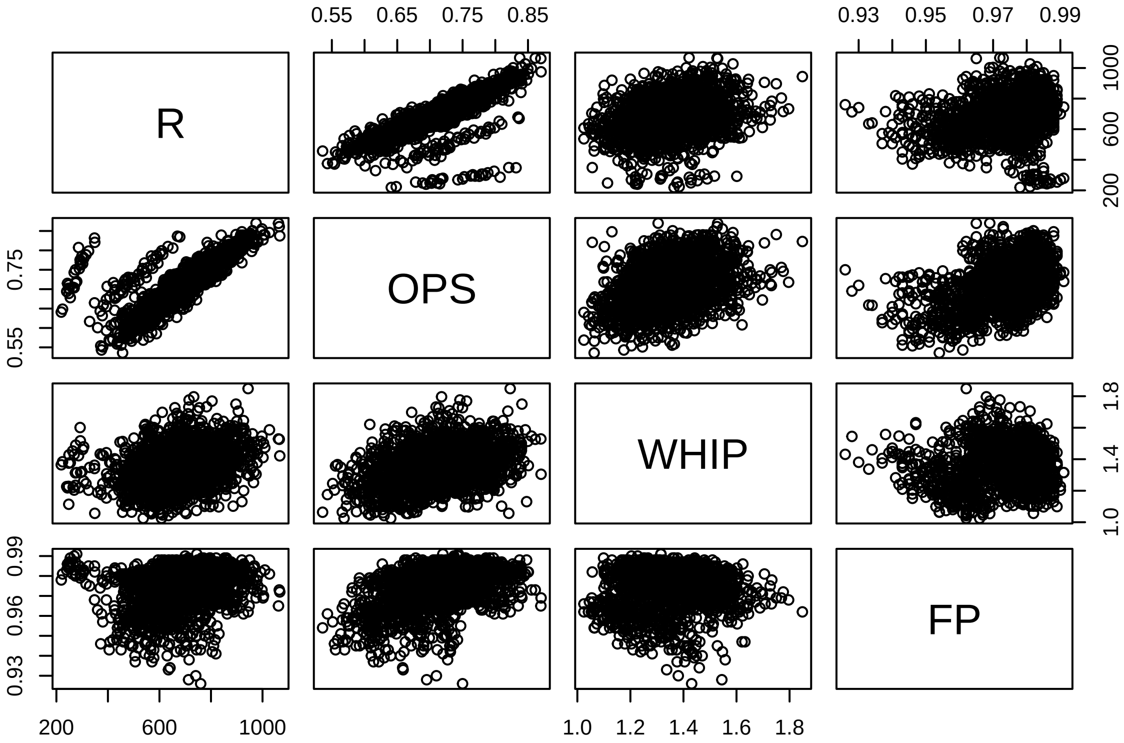
```
mlr_1 = lm(R ~ OPS + WHIP + FP, data = df_1)
summary(mlr_1)
```

```
##
## Call:
## lm(formula = R ~ OPS + WHIP + FP, data = df_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -563.74  -13.16    9.62   29.36  133.98
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1187.52     150.14   7.909 3.79e-15 ***
## OPS          1958.31      29.83  65.659  < 2e-16 ***
## WHIP          -45.19      12.32  -3.668  0.00025 ***
## FP          -1882.77     159.93 -11.772  < 2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 66.43 on 2606 degrees of freedom
## Multiple R-squared:  0.6697, Adjusted R-squared:  0.6693
## F-statistic:  1761 on 3 and 2606 DF,  p-value: < 2.2e-16
```

R^2 here is ok, but let us now test for diagnostics.

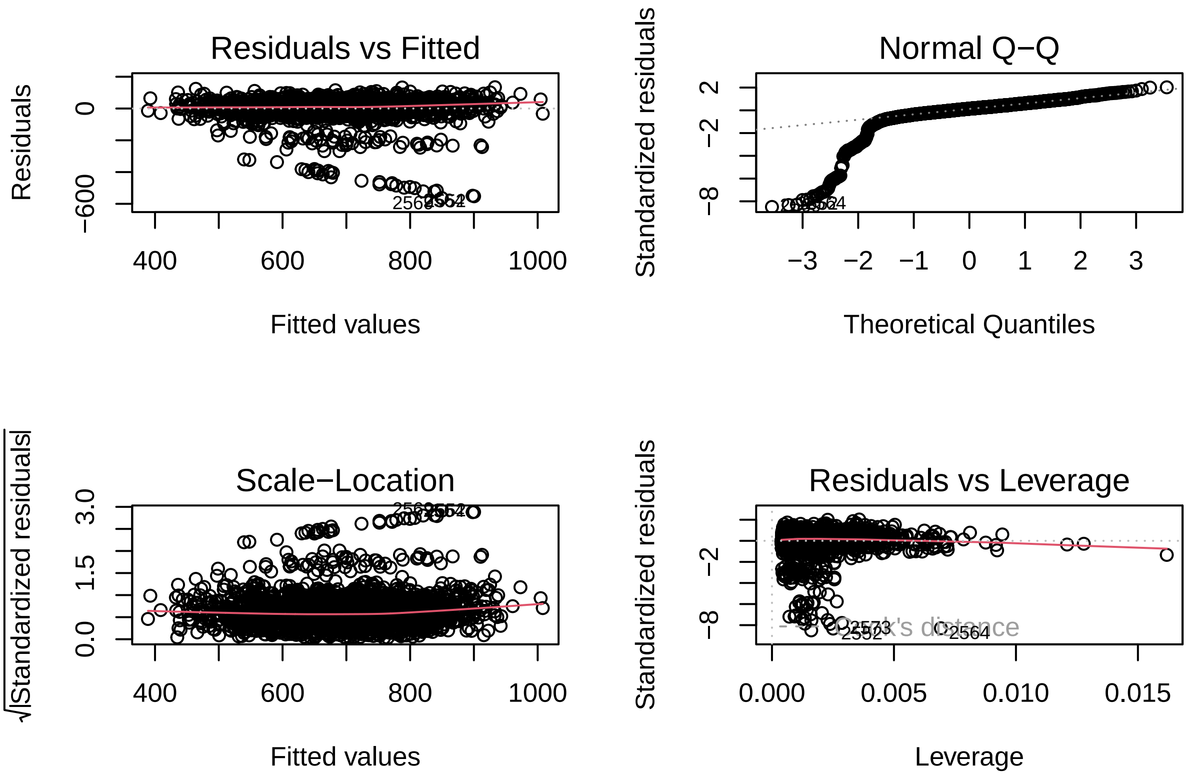
Diagnostics

```
pairs(df_1 |> dplyr::select(R, OPS, WHIP, FP))
```



The linearity assumption holds, as there is clearly a linear relationship between runs and each predictor in the model.

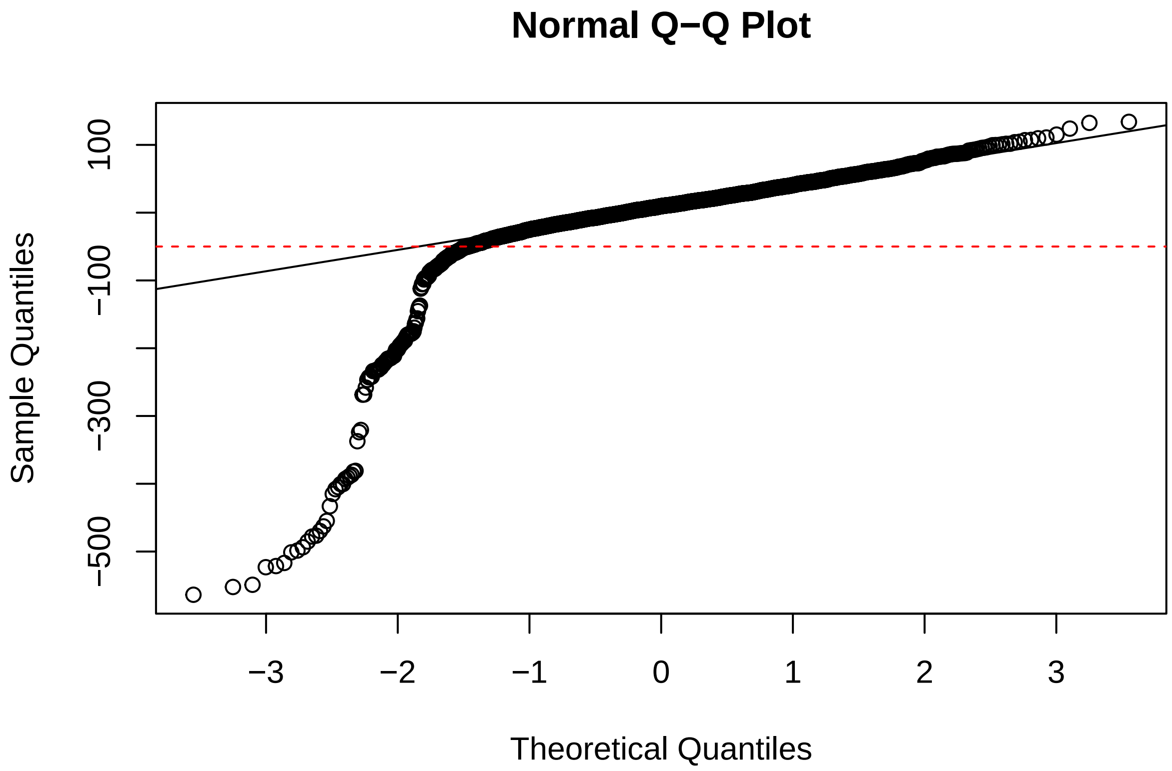
```
par(mfrow = c(2, 2))
plot(mlr_1)
```



As we check that the errors are normally distributed with constant variance and mean 0, only mean 0 seems to suffice. There seems to be deviations from normality and constant variance assumptions. Also, a large right tail in residuals.

Problematic Residuals

```
par(mfrow = c(1, 1))
qqnorm(residuals(mlr_1))
qqline(residuals(mlr_1))
abline(h = -50, lty = 2, col = "red")
```



It seems like certain residuals are causing the normality and constant variance assumptions to fail.

```
df_1 = augment(mlr_1, df_1)

df_1 |>
  filter(.resid < -100) |>
  group_by(yearID) |>
  summarise(n())
```

```
## # A tibble: 6 x 2
##   yearID `n()`
##    <int> <int>
## 1   1918     5
## 2   1919     3
## 3   1921     1
## 4   1981    26
## 5   1994    28
## 6   2020    30
```

```
df_1 |>
  filter(.resid < -50) |>
  group_by(yearID) |>
  summarise(n = n()) |>
  filter(n >= 10)
```

```
## # A tibble: 6 x 2
##   yearID     n
##    <int> <int>
## 1   1918    16
## 2   1919    13
## 3   1981    26
## 4   1994    28
## 5   1995    18
## 6   2020    30
```

These years are where the issues are coming from. Let us try to fix this problem by adding in categorical variables for these years.

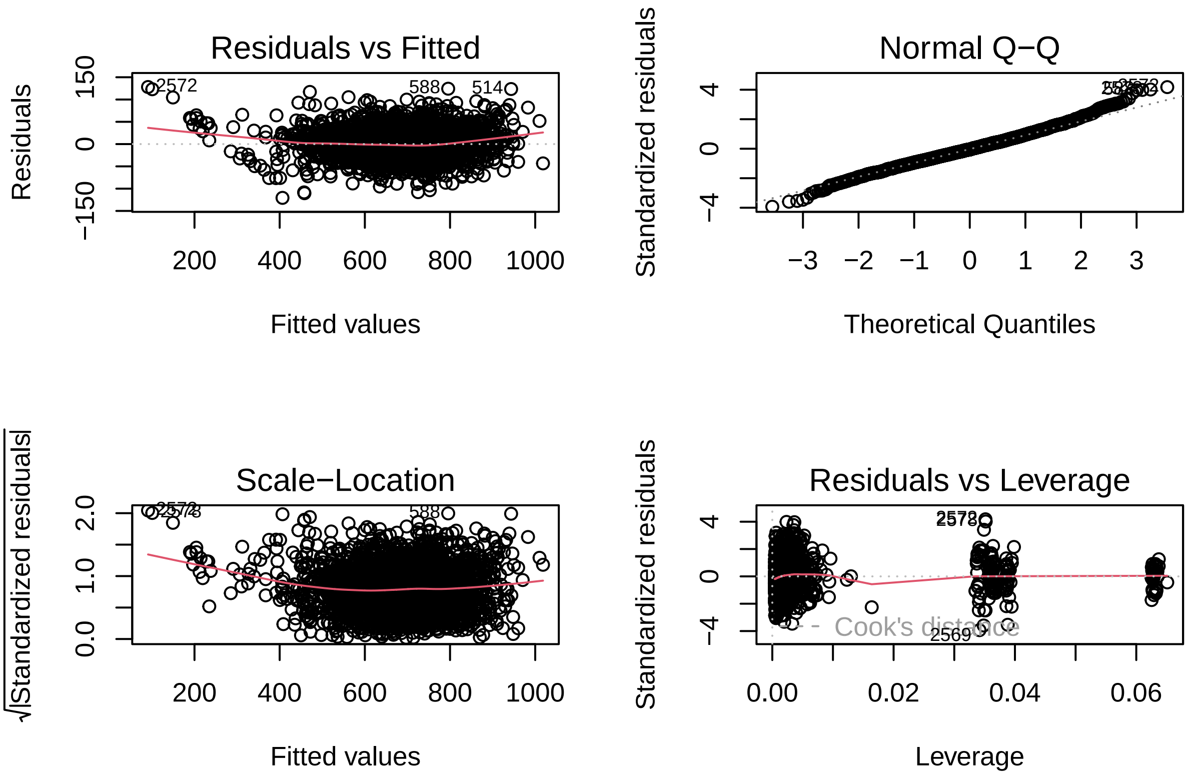
Fixed Model

```
df_1 = df_1 |>
  mutate(year_1918 = ifelse(yearID == 1918, 1, 0),
         year_1919 = ifelse(yearID == 1919, 1, 0),
         year_1981 = ifelse(yearID == 1981, 1, 0),
         year_1994 = ifelse(yearID == 1994, 1, 0),
         year_1995 = ifelse(yearID == 1995, 1, 0),
         year_2020 = ifelse(yearID == 2020, 1, 0))


mlr_2 = lm(R ~ OPS + WHIP + FP + year_1918 + year_1919 + year_1981 + year_1994 + year_1995 + year_2020,
summary(mlr_2)
```

```
##
## Call:
## lm(formula = R ~ OPS + WHIP + FP + year_1918 + year_1919 + year_1981 +
##     year_1994 + year_1995 + year_2020, data = df_1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -120.670  -20.487   -1.574   18.985  128.068
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   625.787     71.396   8.765   <2e-16 ***
## OPS          1970.837     14.110 139.673   <2e-16 ***
## WHIP          -56.189      5.818  -9.658   <2e-16 ***
## FP          -1288.836     75.875 -16.986   <2e-16 ***
## year_1918    -108.151      7.904 -13.683   <2e-16 ***
## year_1919     -79.908      7.872 -10.151   <2e-16 ***
## year_1981    -218.064      6.176 -35.309   <2e-16 ***
## year_1994    -222.023      5.967 -37.206   <2e-16 ***
## year_1995     -70.798      5.960 -11.878   <2e-16 ***
## year_2020    -460.827      5.762 -79.973   <2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 31.22 on 2600 degrees of freedom
## Multiple R-squared:  0.9272, Adjusted R-squared:  0.927
## F-statistic:  3681 on 9 and 2600 DF,  p-value: < 2.2e-16
```

Here, the R^2 is much higher, which is a good sign!

```
par(mfrow = c(2, 2))
plot(mlr_2)
```



All assumptions met now. Thus, this is a much better model than before.

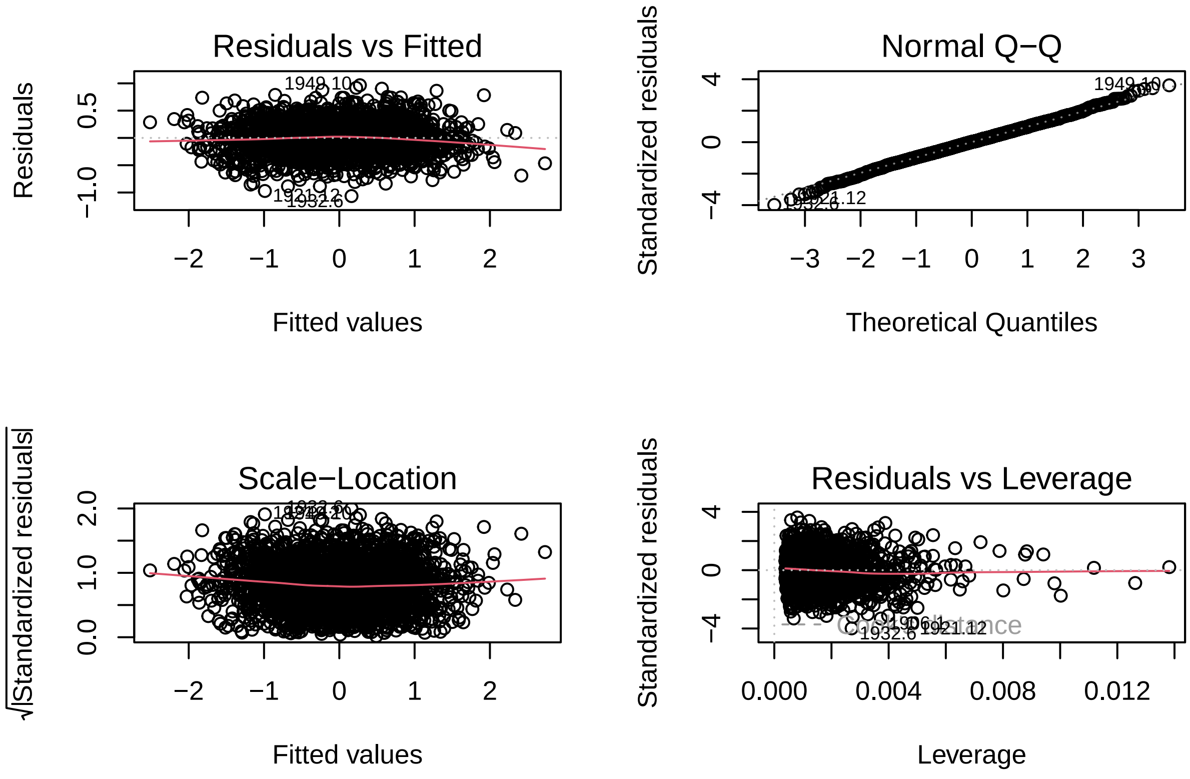
What went wrong is these outlier seasons, due to different circumstances, had drastically different predictor values related to runs. So, after adding in the categorical variables for such years, this problem was solved!

Principled Rescaling / New Model

```
teams_list = Teams |>
  select(yearID, W, L, G, AB, H, X2B, X3B, HR, BB, HBP, SF, HA, HRA,
         BBA, SOA, IPouts, FP, R, RA) |>
  replace_na(list(HBP = 0, SF = 0)) |>
  mutate(RD = (R - RA) / G) |>
  mutate(X1B = H - (X2B + X3B + HR)) |>
  mutate(OBP = (H + BB + HBP) / (AB + BB + HBP + SF)) |>
  mutate(SLG = (X1B + 2*X2B + 3*X3B + 4*HR) / AB) |>
  mutate(OPS = OBP + SLG) |>
  mutate(IP = IPouts / 3) |>
  mutate(WHIP = (BBA + HA) / IP) |>
  split(Teams$yearID) |>
  lapply(avgOBP = (sum(H) + sum(BB) + sum(HBP)) / (sum(AB) + sum(BB) + sum(HBP) + sum(SF)),
         avgSLG = (sum(X1B) + 2*sum(X2B) + 3*sum(X3B) + 4*sum(HR)) / sum(AB),
         avgOPS = avgOBP + avgSLG,
         avgWHIP = (sum(BBA) + sum(HA)) / sum(IP),
         #avgFP here is only rough approximation here
         avgFP = mean(FP),
         mutate) |>
  lapply(OPSscale = OPS / avgOPS,
         WHIPscale = avgWHIP / WHIP,
         FPscale = avgFP / FP,
         mutate)

df_2 = do.call('rbind', teams_list)

df_2 = df_2 |>
  filter(yearID >= 1900)


mlr_3 = lm(RD ~ OPSscale + WHIPscale + FPscale, data = df_2)
summary(mlr_3)
```

```
##
## Call:
## lm(formula = RD ~ OPSscale + WHIPscale + FPscale, data = df_2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.06511 -0.17520  0.00438  0.17697  0.96536
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.62214    1.67890   3.944 8.21e-05 ***
## OPSscale     8.99139    0.10691  84.104  < 2e-16 ***
## WHIPscale    7.02806    0.08821  79.677  < 2e-16 ***
## FPscale    -22.66376    1.61910 -13.998  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2673 on 2606 degrees of freedom
## Multiple R-squared:  0.8754, Adjusted R-squared:  0.8752
## F-statistic:  6102 on 3 and 2606 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
plot(mlr_3)
```



The rescaling allowed for a better model compared to the one in the notes (RD ~ OPS + WHIP + FP) because the Rˆ2 is higher and all the model assumptions are met. This rescaling allowed the predictors to be accounted for their specific years. Meaning, if a year is different from the overall average, this accounts for that difference.

Question 2

Batting

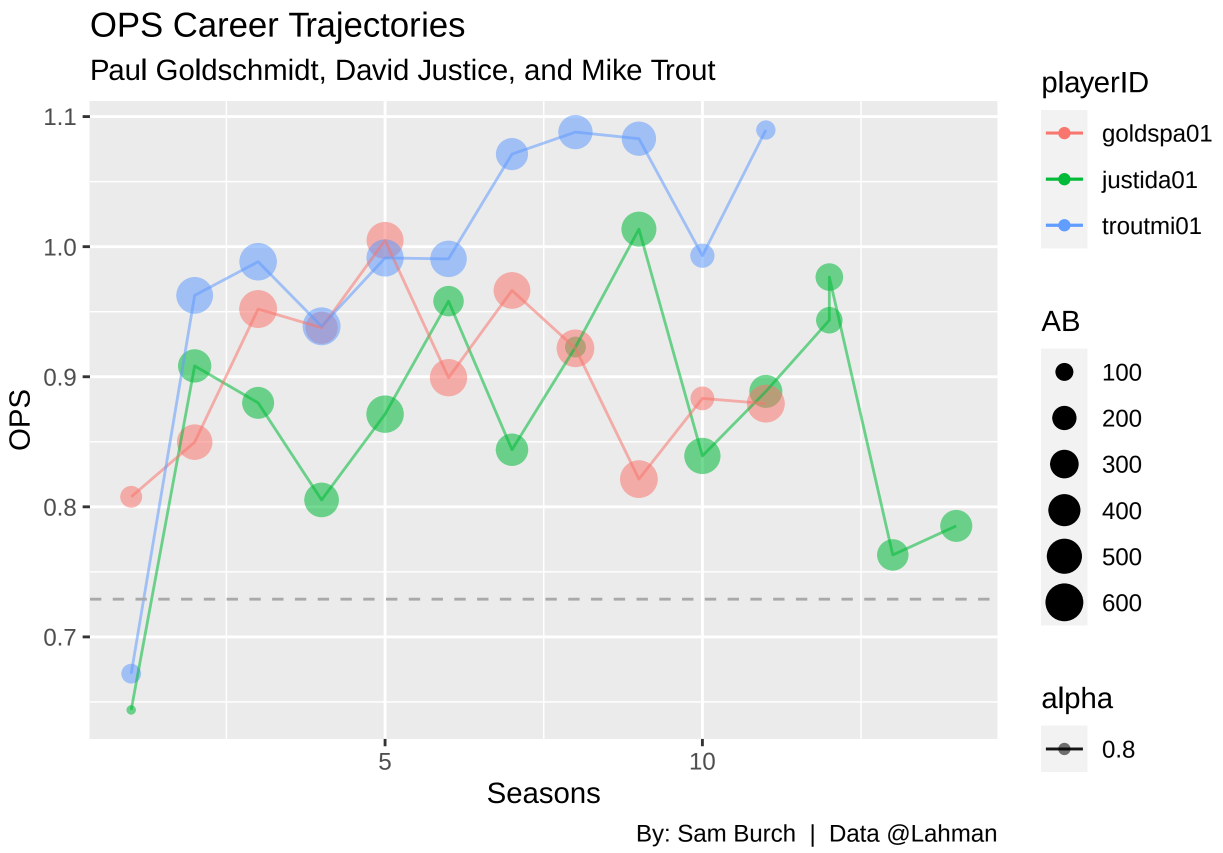
```
Batting = as_tibble(Batting)
People = as_tibble(People)

batters_3 = People |>
  filter(playerID == 'troutmi01' |
         playerID == 'goldspa01' |
         playerID == 'justida01') |>
  dplyr::select(playerID, nameFirst, nameLast) |>
  pull(playerID)

bat_seasons = Batting |>
  filter(playerID %in% batters_3) |>
  group_by(playerID, yearID) |>
  mutate(X1B = H - (X2B + X3B + HR)) |>
  mutate(OBP = (H + BB + HBP) / (AB + BB + HBP + SF)) |>
  mutate(SLG = (X1B + 2*X2B + 3*X3B + 4*HR) / AB) |>
  mutate(OPS = OBP + SLG) |>
  mutate(SB_pct = (SB / (SB + CS))) |>
  mutate(season = case_when((playerID == 'justida01' & yearID == 1989) ~ 1,
                            (playerID == 'justida01' & yearID != 1989) ~ (yearID - 1988),
                            (playerID == 'goldspa01' & yearID == 2011) ~ 1,
                            (playerID == 'goldspa01' & yearID != 2011) ~ (yearID - 2010),
                            (playerID == 'troutmi01' & yearID == 2011) ~ 1,
                            (playerID == 'troutmi01' & yearID != 2011) ~ (yearID - 2010),
  )) |>
  dplyr::select(playerID, yearID, season, G:SO, SB_pct, OBP, SLG, OPS)
bat_seasons
```

```
## # A tibble: 37 x 19
## # Groups:   playerID, yearID [36]
##    playerID  yearID season     G    AB     R     H   X2B   X3B    HR   RBI    SB
##    <chr>      <int>  <dbl> <int> <int> <int> <int> <int> <int> <int> <int> <int>
##  1 justida01   1989      1    16    51     7    12     3     0     1     3     2
##  2 justida01   1990      2   127   439    76   124    23     2    28    78    11
##  3 justida01   1991      3   109   396    67   109    25     1    21    87     8
##  4 justida01   1992      4   144   484    78   124    19     5    21    72     2
##  5 justida01   1993      5   157   585    90   158    15     4    40   120     3
##  6 justida01   1994      6   104   352    61   110    16     2    19    59     2
##  7 justida01   1995      7   120   411    73   104    17     2    24    78     4
##  8 justida01   1996      8    40   140    23    45     9     0     6    25     1
##  9 justida01   1997      9   139   495    84   163    31     1    33   101     3
## 10 justida01   1998     10   146   540    94   151    39     2    21    88     9
## # ... with 27 more rows, and 7 more variables: CS <int>, BB <int>, SO <int>,
## #   SB_pct <dbl>, OBP <dbl>, SLG <dbl>, OPS <dbl>
```

```
Batting |>
  filter(playerID %in% batters_3) |>
  mutate(X1B = H - (X2B + X3B + HR)) |>
  group_by(playerID) |>
  summarise(seasons = n(),
            G = sum(G),
            AB = sum(AB),
            R = sum(R),
            H = sum(H),
            X2B = sum(X2B),
            X3B = sum(X3B),
            HR = sum(HR),
            RBI = sum(RBI),
            SB = sum(SB),
            CS = sum(CS),
            BB = sum(BB),
            SO = sum(SO),

            X1B = sum(X1B),
            HBP = sum(HBP),
            SF = sum(SF)
            ) |>
  mutate(OBP = (H + BB + HBP) / (AB + BB + HBP + SF)) |>
  mutate(SLG = (X1B + 2*X2B + 3*X3B + 4*HR) / AB) |>
  mutate(OPS = OBP + SLG) |>
  mutate(SB_pct = (SB / (SB + CS))) |>
  dplyr::select(playerID, seasons, G:SO, OBP:SB_pct)
```

```
## # A tibble: 3 x 18
##   playerID  seasons     G    AB     R     H   X2B   X3B    HR   RBI    SB    CS
##   <chr>       <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 goldspa01      11  1469  5366   939  1572   341    22   280   927   140    33
## 2 justida01      15  1610  5625   929  1571   280    24   305  1017    53    46
## 3 troutmi01      11  1288  4656   967  1419   268    49   310   816   203    37
## # ... with 6 more variables: BB <int>, SO <int>, OBP <dbl>, SLG <dbl>,
## #   OPS <dbl>, SB_pct <dbl>
```

```
ggplot(bat_seasons, aes(x = season, y = OPS)) +
  labs(
    x = 'Seasons',
    y = 'OPS',
    title = "OPS Career Trajectories",
    subtitle = 'Paul Goldschmidt, David Justice, and Mike Trout',
    caption = 'By: Sam Burch  |  Data @Lahman'
  ) +
  geom_point(aes(size = AB, color = playerID, alpha = .8)) +
  geom_line(aes(color = playerID, alpha = .8), data = bat_seasons |> filter(playerID == 'justida01')) +
  geom_line(aes(color = playerID, alpha = .8), data = bat_seasons |> filter(playerID == 'goldspa01')) +
  geom_line(aes(color = playerID, alpha = .8), data = bat_seasons |> filter(playerID == 'troutmi01')) +
  geom_hline(yintercept =  .729, color = 'darkgrey', linetype = 2)
```



Note: The dotted grey line represents the approximate (modern) average OPS in MLB – .729.

We can see just how great all three batters have been! Basically every year these batters have been in the league, they've had a great OPS – well above average. Trout's even been hovering around 1.1 OPS the last few years, which is absurd. For Justice, this graph shows the fall off (as he got older). Lastly, it only took till their second year for all three to be very good hitters.

Pitching

```
Pitching = as_tibble(Pitching)


pitchers_3 = People |>
  filter(playerID == 'bumgama01' |
           playerID == 'kershcl01' |
           playerID == 'riverma01') |>
  dplyr::select(playerID, nameFirst, nameLast) |>
  pull(playerID)


pitch_seasons = Pitching |>
  filter(playerID %in% pitchers_3) |>
  group_by(playerID, yearID) |>
  mutate(IP = IPouts / 3) |>
  mutate(WHIP = (BB + H) / IP) |>
  mutate(SO_per_9 = SO / (IP * 9)) |>
  mutate(SO_per_BB = SO / BB) |>
  mutate(season = case_when((playerID == 'riverma01' & yearID == 1995) ~ 1,
                            (playerID == 'riverma01' & yearID != 1995) ~ (yearID - 1994),
                            (playerID == 'kershcl01' & yearID == 2008) ~ 1,
                            (playerID == 'kershcl01' & yearID != 2008) ~ (yearID - 2007),
                            (playerID == 'bumgama01' & yearID == 2009) ~ 1,
                            (playerID == 'bumgama01' & yearID != 2009) ~ (yearID - 2008),
                            )) |>
  dplyr::select(playerID:yearID, season, W:L, IPouts:SO, HBP, ERA, WHIP,
                SO_per_9, SO_per_BB, IP)
pitch_seasons
```

```
## # A tibble: 46 x 17
## # Groups:   playerID, yearID [46]
##    playerID yearID season     W     L IPouts     H    ER    HR    BB    SO   HBP
##    <chr>     <int>  <dbl> <int> <int>  <int> <int> <int> <int> <int> <int> <int>
##  1 riverma~   1995      1     5     3    201    71    41    11    30    51     2
##  2 riverma~   1996      2     8     3    323    73    25     1    34   130     2
##  3 riverma~   1997      3     6     4    215    65    15     5    20    68     0
##  4 riverma~   1998      4     3     0    184    48    13     3    17    36     1
##  5 riverma~   1999      5     4     3    207    43    14     2    18    52     3
##  6 riverma~   2000      6     7     4    227    58    24     4    25    58     0
##  7 riverma~   2001      7     4     6    242    61    21     5    12    83     1
##  8 riverma~   2002      8     1     4    138    35    14     3    11    41     2
##  9 riverma~   2003      9     5     2    212    61    13     3    10    63     4
## 10 riverma~   2004     10     4     2    236    65    17     3    20    66     5
## # ... with 36 more rows, and 5 more variables: ERA <dbl>, WHIP <dbl>,
## #   SO_per_9 <dbl>, SO_per_BB <dbl>, IP <dbl>
```

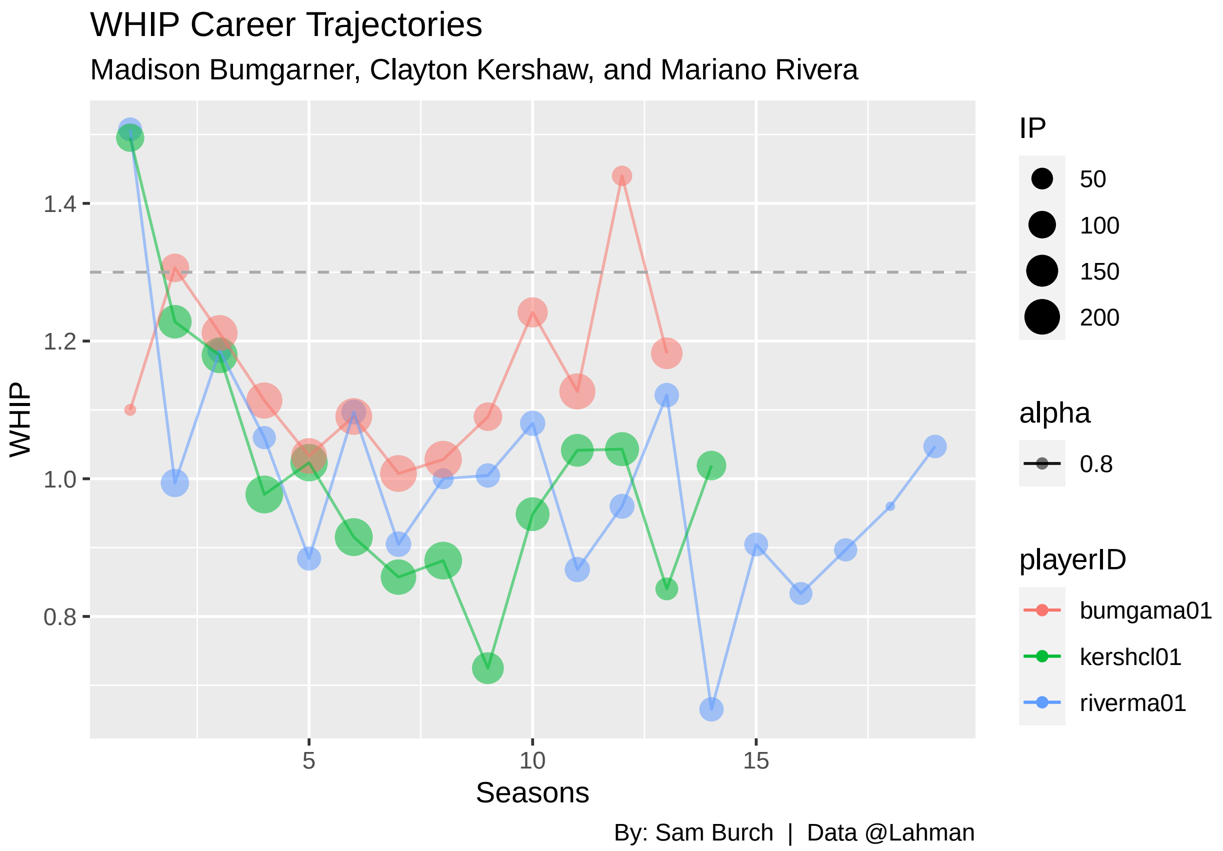
```
Pitching |>
  filter(playerID %in% pitchers_3) |>
  mutate(IP = IPouts / 3) |>
  group_by(playerID) |>
  summarise(
    seasons = n(),
    W = sum(W),
    L = sum(L),
    IPouts = sum(IPouts),
    H = sum(H),
    ER = sum(ER),
    HR = sum(HR),
    BB = sum(BB),
    HBP = sum(HBP),
    SO = sum(SO),

    IP = sum(IP)
  ) |>
  mutate(ERA = (9 * ER) / IP) |>
  mutate(WHIP = (BB + H) / IP) |>
  mutate(SO_per_9 = SO / (IP * 9)) |>
  mutate(SO_per_BB = SO / BB) |>
  dplyr::select(-IP)
```

```
## # A tibble: 3 x 15
##   playerID  seasons     W     L IPouts     H    ER    HR    BB   HBP    SO   ERA
##   <chr>       <int> <int> <int>  <int> <int> <int> <int> <int> <int> <int> <dbl>
## 1 bumgama01      13   127   106   6102  1803   748   229   480    79  1948  3.31
## 2 kershcl01      14   185    84   7364  1859   679   196   606    37  2670  2.49
## 3 riverma01      19    82    60   3851   998   315    71   286    46  1173  2.21
## # ... with 3 more variables: WHIP <dbl>, SO_per_9 <dbl>, SO_per_BB <dbl>
```

```
People |>
  filter(playerID == 'bumgama01' |
           playerID == 'kershcl01' |
           playerID == 'riverma01') |>
  dplyr::select(playerID, nameFirst, nameLast)
```

```
## # A tibble: 3 x 3
##   playerID  nameFirst nameLast
##   <chr>     <chr>     <chr>
## 1 bumgama01 Madison   Bumgarner
## 2 kershcl01 Clayton   Kershaw
## 3 riverma01 Mariano   Rivera
```

```
ggplot(pitch_seasons, aes(season, WHIP)) +
  labs(
    x = 'Seasons',
    y = 'WHIP',
    title = "WHIP Career Trajectories",
    subtitle = 'Madison Bumgarner, Clayton Kershaw, and Mariano Rivera',
    caption = 'By: Sam Burch  |  Data @Lahman'
  ) +
  geom_point(aes(size = IP, color = playerID, alpha = .8)) +
  geom_line(aes(color = playerID, alpha = .8), data = pitch_seasons |> filter(playerID == 'bumgama01'))
  geom_line(aes(color = playerID, alpha = .8), data = pitch_seasons |> filter(playerID == 'riverma01'))
  geom_line(aes(color = playerID, alpha = .8), data = pitch_seasons |> filter(playerID == 'kershcl01'))
  geom_hline(yintercept =  1.30, color = 'darkgrey', linetype = 2)
```



Note: The dotted grey line represents the approximate (modern) average WHIP in MLB – 1.3. Also, the lower the WHIP, the better.

Again, the brilliant play of these guys is clearly seen. Specifically, Kershaw and Rivera have shown elite peaks – with seasons dipping below .8 WHIP. For Bumgarner, his play (while still good) has been more up and down. On top of that, it is interesting how it took a few seasons for these pitchers to become really good. Maybe that suggests pitchers take longer to develop than hitters; however, these graphs don't necessarily prove that.

Question 3 Bob Gibson's 1968 season.

a)

```
Pitching |>
  filter(playerID == 'gibsobo01', yearID == '1968') |>
  select(GS, CG)
```

```
## # A tibble: 1 x 2
##      GS    CG
##   <int> <int>
## 1    34    28
```

```
28/34
```

```
## [1] 0.8235294
```

80% of the 34 games Gibson started (this season) were completed by him.

b)

```
Pitching |>
  filter(playerID == 'gibsobo01', yearID == '1968') |>
  select(BB, SO)
```

```
## # A tibble: 1 x 2
##      BB    SO
##   <int> <int>
## 1    62   268
```

```
268/62
```

```
## [1] 4.322581
```

About 4.3 strikeouts per walk.

c)

```
Pitching |>
  filter(playerID == 'gibsobo01', yearID == '1968') |>
  mutate(IP = IPouts / 3) |>
  select(IP)
```

```
## # A tibble: 1 x 1
##      IP
##   <dbl>
## 1  305.
```

Just over 300 IP.

d)

```
Pitching |>
  filter(playerID == 'gibsobo01', yearID == '1968') |>
  mutate(IP = IPouts / 3) |>
  mutate(WHIP = (BB + H) / IP) |>
  select(WHIP)
```

```
## # A tibble: 1 x 1
##    WHIP
##   <dbl>
## 1 0.853
```

About .85 WHIP.

Question 4 Jim Bunning's perfect game on Father's Day.

```
library(retrosheet)
```

```
## Warning: package 'retrosheet' was built under R version 4.2.2
```

```
##
## For Retrosheet data obtained with this package:
##
## The information used here was obtained free of charge from
## and is copyrighted by Retrosheet. Interested parties may
## contact Retrosheet at "www.retrosheet.org"
```

```
retro_1964 = as_tibble(getRetrosheet(type = 'game', year = 1964))
```

a)

```
retro_1964 |>
  filter(Date == '19640621',
         VisTm == 'PHI',
         HmTm == 'NYN',
         DblHdr == '1') |>
  select(Duration)
```

```
## # A tibble: 1 x 1
##   Duration
##      <int>
## 1      139
```

The game was 2 hours and 19 minutes long.

b)

```
retro_1964 |>
  filter(Date == '19640621',
         VisTm == 'PHI',
         HmTm == 'NYN') |>
  select(Attendance)
```

```
## # A tibble: 2 x 1
##   Attendance
##        <int>
## 1          0
## 2      32026
```

The attendance value is equal to 0 because it is a double header. The attendance for the day is recorded for the second game that day, 32026.

c)

```
retro_1964 |>
  filter(Date == '19640621',
         VisTm == 'PHI',
         HmTm == 'NYN',
         DblHdr == '1') |>
  select(VisD, VisT, VisHR) |>
  sum()
```

```
## [1] 3
```

Three extra base hits by the Phillies.

d)

```
retro_1964 |>
  filter(Date == '19640621',
         VisTm == 'PHI',
         HmTm == 'NYN',
         DblHdr == '1') |>
  mutate(Vis_OBP = (VisH + VisBB + VisHBP) / (VisAB + VisBB + VisHBP + VisSF)) |>
  select(Vis_OBP)
```

```
## # A tibble: 1 x 1
##   Vis_OBP
##     <dbl>
## 1   0.333
```

.333 OBP by the Phillies.