Data manipulations and linear regression

Daniel J. Eck

Background

This lecture is meant to supplement Chapter 2 in your textbook.

We present a brief overview of linear regression.

The `dplyr` package within `tidyverse`

`dplyr` provides comprehensive tools for data manipulations (or wrangling). The main “verbs” are:

- `select()`: choose from a subset of the columns
- `filter()`: choose a subset of the rows based on logical criteria
- `arrange()`: sort the rows based on values of the columns.
- `mutate()`: add or modify the definitions of the column, and create columns that are functions of existing columns.
- `summarize()`: collapse a data frame down to a single row (per group) by aggregating vectors into a single value. Often used in conjunction with `group_by()`
- `left_join()`: add columns from one data set to another, matching observations based on keys.

The pipe operator

The pipe operator `%>%` allows for verbs to be strung in succession so that complicated manipulations can be combined within a single easily digestible sentence.

```
data %>%
    inner_function() %>%
    outer_function()
```

Example: Runs differential regression

```r
library(tidyverse)
library(Lahman)
data(Teams)
head(Teams, 3)
```

```
##   yearID lgID teamID franchID divID Rank  G Ghome  W  L DivWin WCWin LgWin
## 1   1871   NA    BS1      BNA  <NA>    3 31    NA 20 10   <NA>  <NA>     N
## 2   1871   NA    CH1      CNA  <NA>    2 28    NA 19  9   <NA>  <NA>     N
## 3   1871   NA    CL1      CFC  <NA>    8 29    NA 10 19   <NA>  <NA>     N
##   WSWin   R   AB   H X2B X3B HR BB SO SB CS HBP SF  RA  ER  ERA CG SHO SV
## 1  <NA> 401 1372 426  70  37  3 60 19 73 16  NA NA 303 109 3.55 22   1  3
## 2  <NA> 302 1196 323  52  21 10 60 22 69 21  NA NA 241  77 2.76 25   0  1
## 3  <NA> 249 1186 328  35  40  7 26 25 18  8  NA NA 341 116 4.11 23   0  0
##   IPouts  HA HRA BBA SOA   E DP    FP                    name
## 1    828 367   2  42  23 243 24 0.834    Boston Red Stockings
## 2    753 308   6  28  22 229 16 0.829 Chicago White Stockings
## 3    762 346  13  53  34 234 15 0.818  Cleveland Forest Citys
##                           park attendance BPF PPF teamIDBR teamIDlahman45
## 1          South End Grounds I         NA 103  98      BOS            BS1
## 2      Union Base-Ball Grounds         NA 104 102      CHI            CH1
## 3 National Association Grounds         NA  96 100      CLE            CL1
##   teamIDretro
## 1         BS1
## 2         CH1
## 3         CL1
```

```r
Teams %>%
    select(yearID, franchID, 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) %>%
    arrange(desc(RD)) %>%
    head(10)
```

```
##    yearID franchID   W  L   G   AB    H X2B X3B  HR  BB HBP SF   HA HRA BBA
## 1    1939      NYY 106 45 152 5300 1521 259  55 166 701   0  0 1208  85 567
## 2    1927      NYY 110 44 155 5347 1644 291 103 158 635   0  0 1403  42 409
## 3    1902      PIT 103 36 141 4926 1410 189  95  18 372  64  0 1142   4 250
## 4    2020      LAD  43 17  60 2042  523  97   6 118 228  30 12  424  66 145
## 5    1936      NYY 102 51 155 5591 1676 315  83 182 700   0  0 1474  84 663
## 6    1906      CHC 116 36 154 5018 1316 181  71  20 448  45  0 1018  12 446
## 7    2022      LAD 111 51 162 5526 1418 325  31 212 607  56 53 1114 152 407
## 8    1931      NYY  94 59 155 5608 1667 277  78 155 748   0  0 1461  67 543
## 9    1937      NYY 102 52 157 5487 1554 282  73 174 709   0  0 1417  92 506
## 10   1942      NYY 103 51 154 5305 1429 223  57 108 591   0  0 1259  71 431
##     SOA IPouts    FP    R  RA       RD
## 1   565   4044 0.978  967 556 2.703947
## 2   431   4167 0.969  975 599 2.425806
## 3   564   3794 0.958  775 440 2.375887
## 4   517   1616 0.982  349 213 2.266667
## 5   624   4200 0.973 1065 731 2.154839
## 6   702   4165 0.969  704 381 2.097403
## 7  1465   4354 0.986  847 513 2.061728
## 8   686   4230 0.972 1067 760 1.980645
## 9   652   4188 0.972  979 671 1.961783
## 10  558   4125 0.976  801 507 1.909091
```

```r
dat = Teams %>%
    select(yearID, franchID, 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) %>%
    mutate(OBP = (H + BB + HBP)/(AB + BB + HBP + SF)) %>%
    mutate(SLG = (H + X2B + 2*X3B + 3*HR)/AB) %>%
    mutate(OPS = OBP + SLG) %>%
    mutate(WHIP = 3*(HA + BBA)/IPouts)
head(dat, 3)
```

```
##   yearID franchID  W  L   G   AB    H X2B X3B HR  BB HBP SF   HA HRA BBA SOA
## 1   1900      LAD 82 54 141 4860 1423 199  81 26 421  81  0 1370  30 405 300
## 2   1900      ATL 66 72 142 4952 1403 163  68 48 395  45  0 1263  59 463 340
## 3   1900      CHC 65 75 146 4907 1276 202  51 33 343  65  0 1375  21 324 357
##   IPouts    FP   R  RA         RD       OBP       SLG       OPS     WHIP
## 1   3677 0.948 816 722  0.6666667 0.3590078 0.3831276 0.7421354 1.448191
## 2   3721 0.953 778 739  0.2746479 0.3418027 0.3727787 0.7145813 1.391561
## 3   3813 0.933 635 751 -0.7945205 0.3168391 0.3421643 0.6590034 1.336743
```

Note: other packages may contain functions with the same name as those in `dplyr`. For example, the `MASS` package also contains a `select` function.

In the event that you have both `dplyr` and `MASS` loaded in an R session, you can access `dplyr`'s `select` function using `dplyr::select`

```
dat = Teams %>%
    dplyr::select(yearID, franchID, 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(WHIP = 3*(HA + BBA)/IPouts) %>%
    mutate(FIP = 3*(13*HRA + 3*BBA - 2*SOA)/IPouts)
```

Baseball is a game of offense, pitching, and defense. Let's see how well runs differential per game is explained by:

- OPS: on base percentage plus slugging percentage
- WHIP: walks and hits allowed divided by innings pitched
- FP: fielding percentage

using a linear regression model

```
m = lm(RD ~ OPS + WHIP + FP, data = dat)
```

Regression review

Regression model:

$$y = \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon; \qquad \varepsilon \sim N(0, \sigma^2),$$

where we usually specify a model intercept by setting $x_1 = 1$.

Can also write in vector notation:

$$y = \mathbf{x}'\beta + \varepsilon; \qquad \varepsilon \sim N(0, \sigma^2),$$

where $\mathbf{x}, \beta \in \mathbb{R}^p$.

Either way, this model relies on a few assumptions:

- a linear relationship is present
- errors are independent and identically distributed
- errors are normally distributed mean 0 and common variance σ^2

Regression review

Remember that linear regression is about modeling a conditional expectation, the scattering of points is noise. Interest is in

$$E(y|\mathbf{x}) = \mathbf{x}'\beta,$$

where it is important to choose the variables comprising **x** and to be able to defend those choices.

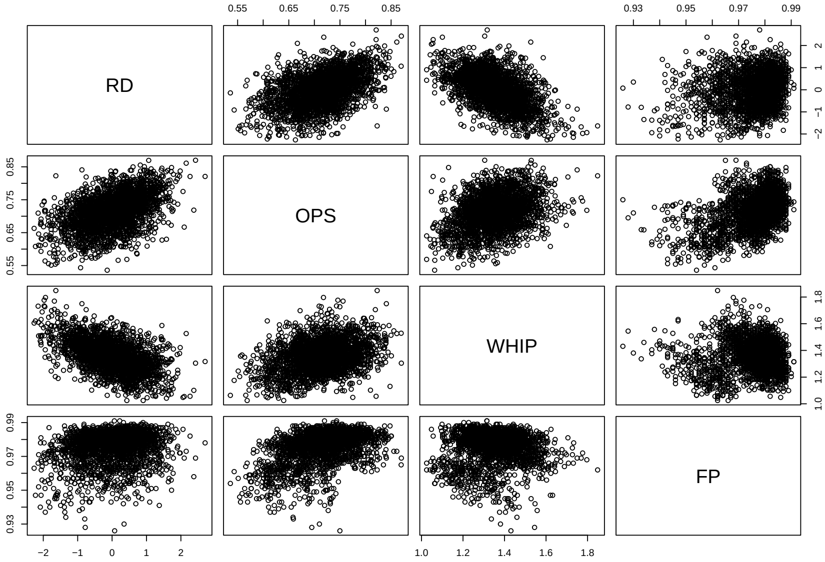
Yes, baseball IS a game of offense, pitching, and defense.

```r
summary(m)
```

```
##
## Call:
## lm(formula = RD ~ OPS + WHIP + FP, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4761 -0.2167 -0.0072  0.2113  1.4780
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14.03489    0.77752   18.05   <2e-16 ***
## OPS          11.80392    0.15453   76.39   <2e-16 ***
## WHIP         -5.39681    0.06357  -84.89   <2e-16 ***
## FP          -15.54174    0.82713  -18.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3484 on 2666 degrees of freedom
## Multiple R-squared:  0.7897, Adjusted R-squared:  0.7895
## F-statistic:  3338 on 3 and 2666 DF,  p-value: < 2.2e-16
```

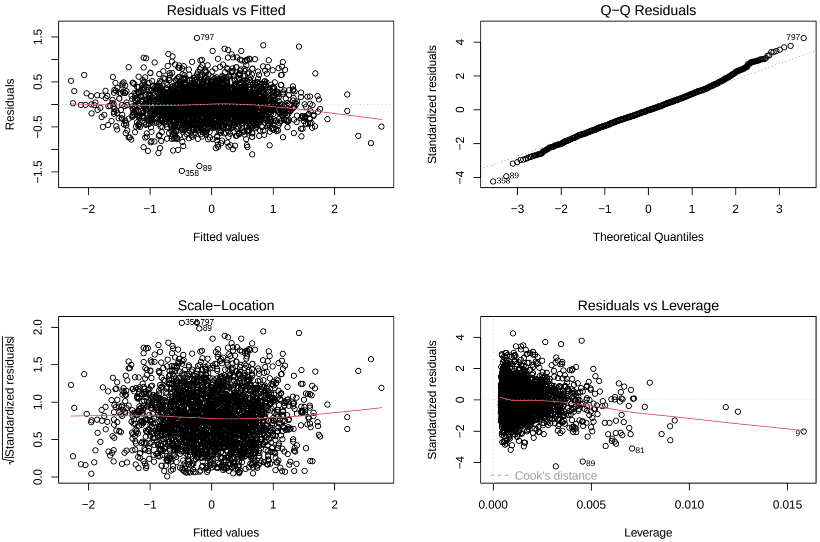
Linearity holds.

```
pairs(dat %>% select(RD, OPS, WHIP, FP))
```



Normality of mean-zero errors with constant variance holds. Although a slightly heavy right tail is observed in the residuals.

```r
par(mfrow = c(2,2))
plot(m)
```



The compiled fractions should be roughly 68% and 95% if errors are truly normal with common variance.

```
library(broom)
dat_aug = augment(m, data = dat)
dat_aug %>%
  mutate(rmse = sqrt((mean(.resid^2)))) %>%
  summarise(N = n(),
            within_1rmse = sum(abs(.resid) < rmse),
            within_2rmse = sum(abs(.resid) < 2 * rmse)) %>%
  mutate(within_1rmse_pct = within_1rmse / N,
         within_2rmse_pct = within_2rmse / N)
```

```
## # A tibble: 1 x 5
##       N within_1rmse within_2rmse within_1rmse_pct within_2rmse_pct
##   <int>        <int>        <int>            <dbl>            <dbl>
## 1  2670         1895         2526            0.710            0.946
```

We will suppose that independence holds, or that any violations of this assumption that may be present in this data do not materially effect our overall conclusions.

A saturated model (one parameter per observation) does not fit the data better than our model with three variables and an intercept.

```r
# likelihood ratio test of fitted model vs a saturated model
m_glm = glm(RD ~ OPS + WHIP + FP, data = dat)
pchisq(m_glm$deviance, m_glm$df.residual, lower = FALSE)
```

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

Thus we have a well-fitting simple and useful model that provides satisfactory dimension reduction.

Investigate large residuals ($y - \hat{y}$)

```
dat_aug %>% filter(abs(.resid) >= 1) %>%
    select(yearID, franchID, W, RD, OPS, WHIP, FP, .resid, .fitted) %>%
    mutate(across(4:9, round, 3)) %>%
    arrange(desc(.resid))
```

```
## # A tibble: 26 x 9
##    yearID franchID     W    RD   OPS  WHIP    FP .resid .fitted
##     <int> <fct>    <int> <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>
##  1   1949 NYY         97 1.24  0.758  1.49 0.977   1.48  -0.239
##  2   1936 NYY        102 2.15  0.861  1.53 0.973   1.32   0.84
##  3   1939 NYY        106 2.70  0.821  1.32 0.978   1.29   1.42
##  4   1950 BOS         94 1.45  0.846  1.59 0.981   1.24   0.211
##  5   1949 BOS         96 1.48  0.8    1.48 0.98    1.21   0.266
##  6   1935 DET         93 1.67  0.798  1.44 0.979   1.19   0.482
##  7   1948 BOS         96 1.21  0.779  1.48 0.981   1.19   0.016
##  8   1943 CIN         87 0.419 0.653  1.34 0.98    1.12  -0.703
##  9   1950 NYY         98 1.44  0.804  1.48 0.979   1.11   0.325
## 10   1938 DET         84 0.432 0.768  1.59 0.976   1.07  -0.635
## # i 16 more rows
```

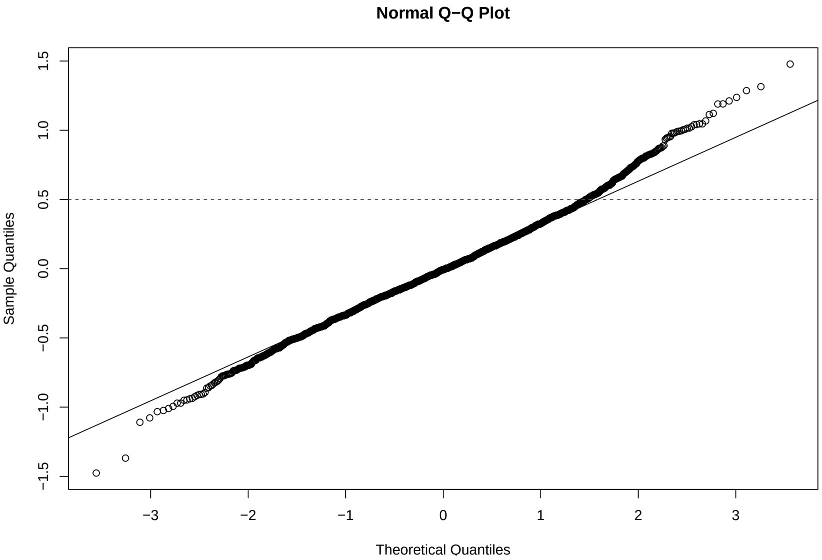
Investigate large fitted values

```
dat_aug %>% filter(.fitted >= 2) %>%
    select(yearID, franchID, W, RD, OPS, WHIP, FP, .resid, .fitted)
```

```
## # A tibble: 5 x 9
##   yearID franchID     W    RD   OPS  WHIP    FP .resid .fitted
##    <int> <fct>    <int> <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>
## 1   1927 NYY        110  2.43 0.870  1.30 0.969  0.221    2.20
## 2   2019 HOU        107  1.73 0.848  1.13 0.988 -0.859    2.59
## 3   2019 LAD        106  1.69 0.810  1.10 0.982 -0.698    2.38
## 4   2020 LAD         43  2.27 0.821  1.06 0.982 -0.493    2.76
## 5   2022 LAD        111  2.06 0.775  1.05 0.986 -0.143    2.20
```

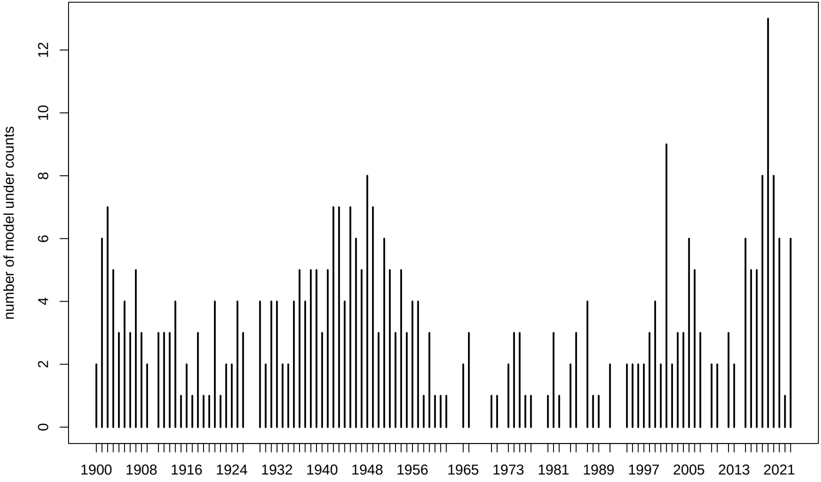
A closer look at problems with fit

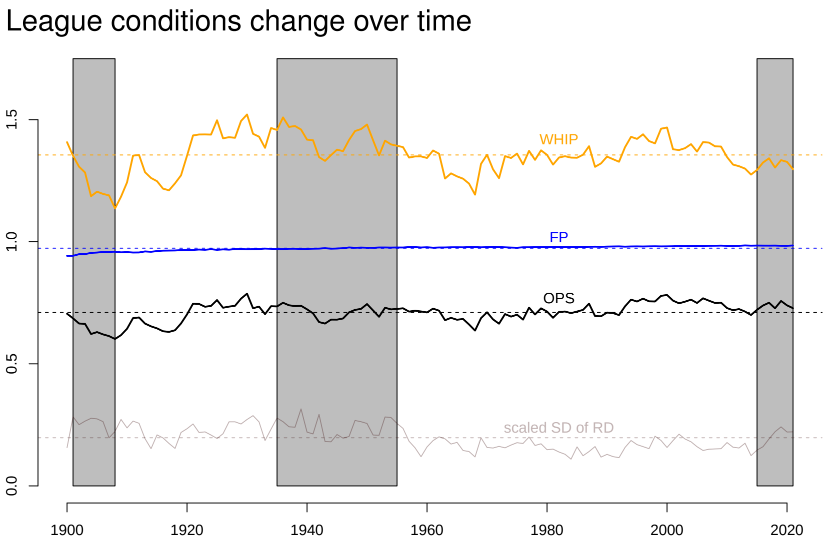
```
qqnorm(resid(m)); qqline(resid(m))
abline(a=0.5, b=0, lty = 2, col = "red")
```



A closer look at problems with fit

```r
plot(table(dat_aug %>% filter(abs(.resid) >= 0.5) %>%
  pull(yearID)), ylab = "number of model under counts")
```





League conditions change over time
WHIP
FP
OPS
scaled SD of RD
0.0
0.5
1.0
1.5
1900
1920
1940
1960
1980
2000
2020