Value of plays using run expectancy

Daniel J. Eck

Background

This lecture is meant to supplement Chapter 5 in your textbook.

We will now study the value of baseball events.

Introduction to the run expectancy matrix

The run expectancy matrix is the average number of runs scored for each combination of outs and runners on base.

There are 8 possible arrangements of runners on the three bases, and the number of possible outs equals 3. Thus there are a total of 24 possible arrangements of outs and runners which form the run expectancy matrix.

The run expectancy matrix establishes a baseline value for baseball events in a context-free environment.

Data

We will calculate the run expectancy matrix. We first load in relevant software packages

```
library(Lahman)
library(tidyverse)
library(retrosheet)
```

and then we load in play-by-play data from the 2016 season

```
fields = read_csv("../../fields.csv")
dat2016 = read_csv("../../all2016.csv",
                    col_names = pull(fields, Header),
                    na = character())
```

Data manipulations

We compute a runs scored in the remainder of the inning variable and add it to `dat2016`.

```
dat2016 = dat2016 %>%
  mutate(RUNS = AWAY_SCORE_CT + HOME_SCORE_CT,
         HALF.INNING = paste(GAME_ID, INN_CT, BAT_HOME_ID),
         RUNS.SCORED = (BAT_DEST_ID > 3) + (RUN1_DEST_ID > 3) +
                 (RUN2_DEST_ID > 3) + (RUN3_DEST_ID > 3))

half_innings = dat2016 %>%
  group_by(HALF.INNING) %>%
  summarise(Outs.Inning = sum(EVENT_OUTS_CT),
            Runs.Inning = sum(RUNS.SCORED),
            Runs.Start = first(RUNS),
            MAX.RUNS = Runs.Inning + Runs.Start)

dat2016 = dat2016 %>%
  inner_join(half_innings, by = "HALF.INNING") %>%
  mutate(RUNS.ROI = MAX.RUNS - RUNS)
```

Creating the matrix

Now that the runs scored in the remainder of the inning variable have been computed for each plate appearance, it is straightforward to compute the run expectancy matrix.

We create a `BASES` variable which indicates the base runner state (eg, 100 corresponds to a runner on first), and a `STATE` variable which adds the number of outs to `BASES`.

```
dat2016 =
  dat2016 %>% mutate(BASES = paste(ifelse(BASE1_RUN_ID != "",1,0),
                                   ifelse(BASE2_RUN_ID != "",1,0),
                                   ifelse(BASE3_RUN_ID != "",1,0), sep = ""),
                     STATE = paste(BASES, OUTS_CT))
```

Creating the matrix (continued)

We now trim `dat2016` to only include plays in which the state of the game changed and a half inning reached 3 outs.

```
dat2016 = dat2016 %>%
  mutate(NRUNNER1 = as.numeric(RUN1_DEST_ID == 1 | BAT_DEST_ID == 1),
         NRUNNER2 = as.numeric(RUN1_DEST_ID == 2 | RUN2_DEST_ID == 2 | BAT_DEST_ID == 2),
         NRUNNER3 = as.numeric(RUN1_DEST_ID == 3 | RUN2_DEST_ID == 3 |
           RUN3_DEST_ID == 3 | BAT_DEST_ID == 3),
         NOUTS = OUTS_CT + EVENT_OUTS_CT,
         NEW.BASES = paste(NRUNNER1, NRUNNER2, NRUNNER3, sep = ""),
         NEW.STATE = paste(NEW.BASES, NOUTS)) %>%
  filter((STATE != NEW.STATE) | (RUNS.SCORED > 0)) %>%
  filter(Outs.Inning == 3)
```

Creating the matrix (continued)

We now create the run expectancy matrix `RUNS_out`

```
RUNS = dat2016 %>%
  group_by(STATE) %>%
  summarize(Mean = mean(RUNS.ROI)) %>%
  mutate(Outs = substr(STATE, 5, 5)) %>%
  arrange(Outs)

RUNS_out = matrix(round(RUNS$Mean, 2), 8, 3)
dimnames(RUNS_out)[[1]] = c("000","001","010","011",
                             "100","101","110","111")
dimnames(RUNS_out)[[2]] = c("0 outs", "1 out", "2 outs")
RUNS_out
```

```
##     0 outs 1 out 2 outs
## 000   0.50  0.27   0.11
## 001   1.35  0.94   0.37
## 010   1.13  0.67   0.31
## 011   1.93  1.36   0.55
## 100   0.86  0.51   0.22
## 101   1.72  1.20   0.48
## 110   1.44  0.92   0.41
## 111   2.11  1.54   0.70
```

Measuring the success of a batting play

When a player comes to bat with a particular runners out situation, the run expectancy matrix tells us the number of runs a team is expected to score in the remainder of the half inning:

$$\text{Run Value} = (\text{Runs}_{\text{new state}} - \text{Runs}_{\text{old state}}) + \text{Runs}_{\text{scored on play}}$$

```
dat2016 = dat2016 %>%
  left_join(select(RUNS, - Outs), by = "STATE") %>%
  rename(Runs.State = Mean) %>%
  left_join(select(RUNS, -Outs), by = c("NEW.STATE" = "STATE")) %>%
  rename(Runs.New.State = Mean) %>%
  replace_na(list(Runs.New.State = 0)) %>%
  mutate(run_value = Runs.New.State - Runs.State + RUNS.SCORED)
```

Example: Jose Altuve

We will now study Jose Altuve's 2016 season.

The code below isolates the run value for each of Altuve's batting events and displays his first 3 batting events.

```
data('People')
altuve.id = People %>% filter(nameFirst == "Jose", nameLast == "Altuve") %>% pull(retroID)
# BAT_EVENT_FL == TRUE distinguishes batting events from non batting events like steals
altuve = dat2016 %>% filter(BAT_ID == altuve.id,
                            BAT_EVENT_FL == TRUE)
altuve %>% select(STATE, NEW.STATE, run_value) %>%
  slice(1:3)
```

```
## # A tibble: 3 x 3
##   STATE NEW.STATE run_value
##   <chr> <chr>         <dbl>
## 1 000 1 000 2        -0.162
## 2 000 1 100 1         0.244
## 3 000 1 000 2        -0.162
```

We can see that Jose Altuve was 13th in total RE24 value.

```r
dat2016 %>% inner_join(People %>% select(nameFirst, nameLast, retroID),
             by = c("BAT_ID" = "retroID")) %>%
  filter( BAT_EVENT_FL == TRUE) %>%
  group_by(BAT_ID) %>%
  summarise(nameFirst = unique(nameFirst),
            nameLast = unique(nameLast),
            RE24 = sum(run_value)) %>%
  arrange(desc(RE24)) %>% as.data.frame() %>% head(20)
```

```
##      BAT_ID nameFirst    nameLast     RE24
## 1  troum001      Mike       Trout 65.21086
## 2  ortid001     David       Ortiz 58.75680
## 3  freef001   Freddie     Freeman 46.19011
## 4  donaj001      Josh   Donaldson 46.10413
## 5  vottj001      Joey       Votto 45.74782
## 6  bryak001      Kris      Bryant 45.68620
## 7  arenn001     Nolan     Arenado 45.34225
## 8  murpd006    Daniel      Murphy 44.57236
## 9  rizza001   Anthony       Rizzo 40.19773
## 10 bettm001    Mookie       Betts 35.65014
## 11 cabrm001    Miguel     Cabrera 34.27806
## 12 goldp001      Paul Goldschmidt 34.27474
## 13 altuj001      Jose      Altuve 33.16617
## 14 encae001     Edwin Encarnacion 32.03130
## 15 blacc001   Charlie    Blackmon 31.86506
## 16 ramih003    Hanley     Ramirez 31.39952
## 17 machm001     Manny     Machado 29.48961
## 18 belta001    Adrian      Beltre 29.11069
## 19 gonzc001    Carlos    Gonzalez 28.29142
## 20 cruzn002    Nelson        Cruz 28.18916
```

We can see the number of opportunities Jose Altuve had in each base out state.

```r
altuve %>% group_by(STATE) %>%
  summarise(N = n(), avg_run_value = mean(run_value),
                      total_run_value = sum(run_value), se_run_value = sd(run_value)/ sqrt(N)) %
    as.data.frame()
```

```
##    STATE   N avg_run_value total_run_value se_run_value
## 1  000 0 185   0.036051610       6.6695479   0.02707034
## 2  000 1 104   0.030673539       3.1900481   0.02786826
## 3  000 2 126  -0.002899510      -0.3653383   0.01736461
## 4  001 0   3  -0.103059751      -0.3091793   0.30703088
## 5  001 1   8   0.382615911       3.0609273   0.15351209
## 6  001 2  12   0.100203614       1.2024434   0.14924366
## 7  010 0  15  -0.078676867      -1.1801530   0.15080076
## 8  010 1  22   0.016366766       0.3600688   0.13704177
## 9  010 2  22   0.049357251       1.0858595   0.10129068
## 10 011 0   5   0.317988565       1.5899428   0.37330473
## 11 011 1   5  -0.096378018      -0.4818901   0.18380432
## 12 011 2   8   0.290682347       2.3254588   0.34111391
## 13 100 0  31   0.162685083       5.0432376   0.13171672
## 14 100 1  58   0.085061807       4.9335848   0.07587254
## 15 100 2  39   0.005568296       0.2171636   0.06065280
## 16 101 0   8  -0.316187264      -2.5294981   0.20184533
## 17 101 1   7   0.187174415       1.3102209   0.21187277
## 18 101 2   6   0.363020367       2.1781222   0.23086490
## 19 110 0   6   0.684712857       4.1082771   0.50690064
## 20 110 1  14  -0.060673082      -0.8494231   0.22048071
## 21 110 2  19   0.089659990       1.7035398   0.19236237
## 22 111 0   2  -0.239618713      -0.4792374   0.32927105
## 23 111 1   5   0.215543377       1.0777169   0.52456637
## 24 111 2   1  -0.695272354      -0.6952724           NA
```

Cleaner presentation of sample sizes for each out base state:

```
altuve_RE = altuve %>% group_by(STATE) %>%
  summarize(N = n(), avg_run_value = mean(run_value)) %>%
  mutate(Outs = substr(STATE, 5, 5)) %>%
  arrange(Outs)

altuve_N_mat = matrix(round(altuve_RE$N, 4), 8, 3)
dimnames(altuve_N_mat)[[1]] = c("000","001","010","011",
                               "100","101","110","111")
dimnames(altuve_N_mat)[[2]] = c("0 outs", "1 out", "2 outs")
altuve_N_mat
```

```
##     0 outs 1 out 2 outs
## 000    185   104    126
## 001      3     8     12
## 010     15    22     22
## 011      5     5      8
## 100     31    58     39
## 101      8     7      6
## 110      6    14     19
## 111      2     5      1
```

```
colSums(altuve_N_mat)
```

```
## 0 outs  1 out 2 outs
##    255    223    233
```

```
rowSums(altuve_N_mat)
```

```
## 000 001 010 011 100 101 110 111
## 415  23  59  18 128  21  39   8
```

Cleaner presentation of run value for each out base state:

```r
altuve_RE_mat = matrix(round(altuve_RE$avg_run_value, 4), 8, 3)
dimnames(altuve_RE_mat)[[1]] = c("000","001","010","011",
                                 "100","101","110","111")
dimnames(altuve_RE_mat)[[2]] = c("0 outs", "1 out", "2 outs")

colMeans(altuve_RE_mat)
```

```
##    0 outs     1 out    2 outs
## 0.0579875 0.0950500 0.0250500
```

```r
rowMeans(altuve_RE_mat)
```

```
##         000         001         010         011         100         101
##  0.02130000  0.12656667 -0.00430000  0.17076667  0.08446667  0.07800000
##         110         111
##  0.23790000 -0.23980000
```

We detect statistically significant differences for Jose Altuve's performance across base out states using an anova test. However, a close look reveals that these differences are not intuitive and our detected statistical significance may be just noise.

```r
# performance in different states
summary(aov(run_value ~ -1 + STATE, data = altuve))
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## STATE       24   9.97  0.4153   1.873 0.00716 **
## Residuals  687 152.32  0.2217
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
round(coef(summary(lm(run_value ~ -1 + STATE, data = altuve))), 3)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## STATE000 0      0.036      0.035   1.041    0.298
## STATE000 1      0.031      0.046   0.664    0.507
## STATE000 2     -0.003      0.042  -0.069    0.945
## STATE001 0     -0.103      0.272  -0.379    0.705
## STATE001 1      0.383      0.166   2.298    0.022
## STATE001 2      0.100      0.136   0.737    0.461
## STATE010 0     -0.079      0.122  -0.647    0.518
## STATE010 1      0.016      0.100   0.163    0.871
## STATE010 2      0.049      0.100   0.492    0.623
## STATE011 0      0.318      0.211   1.510    0.131
## STATE011 1     -0.096      0.211  -0.458    0.647
## STATE011 2      0.291      0.166   1.746    0.081
## STATE100 0      0.163      0.085   1.924    0.055
## STATE100 1      0.085      0.062   1.376    0.169
## STATE100 2      0.006      0.075   0.074    0.941
## STATE101 0     -0.316      0.166  -1.899    0.058
## STATE101 1      0.187      0.178   1.052    0.293
## STATE101 2      0.363      0.192   1.888    0.059
## STATE110 0      0.685      0.192   3.562    0.000
## STATE110 1     -0.061      0.126  -0.482    0.630
## STATE110 2      0.090      0.108   0.830    0.407
## STATE111 0     -0.240      0.333  -0.720    0.472
## STATE111 1      0.216      0.211   1.024    0.306
## STATE111 2     -0.695      0.471  -1.477    0.140
```

Two-way anova tests do not reveal statistical significance.

```
summary(aov(run_value ~ -1 + BASES + Outs, data = altuve %>%
              mutate(Outs = substr(STATE, 5, 5))))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## BASES         8   3.04  0.3802   1.675  0.101
## Outs          2   0.15  0.0774   0.341  0.711
## Residuals 701 159.10  0.2270
```

```
summary(aov(run_value ~ -1 + BASES + Outs, data = altuve %>%
              filter(!(BASES %in% c("111", "011"))) %>%
               mutate(Outs = substr(STATE, 5, 5))))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## BASES         6   2.39  0.3975   1.896 0.0792 .
## Outs          2   0.16  0.0816   0.389 0.6777
## Residuals 677 141.99  0.2097
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A dichotomy between RISP and no RISP reveals statistical significance.

```
# performance with RISP
altuve = altuve %>% mutate(RISP = ifelse(!BASES %in% c("100","000"),1,0))
summary(aov(run_value ~ -1 + RISP + Outs, data = altuve %>%
              mutate(Outs = substr(STATE, 5, 5))))
```

```
##            Df Sum Sq Mean Sq F value Pr(>F)
## RISP        1   1.08  1.0813   4.766 0.0294 *
## Outs        3   0.81  0.2687   1.184 0.3147
## Residuals 707 160.41  0.2269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
altuve %>% group_by(RISP) %>% summarise(N = n(), avg_run_value = mean(run_value),
                    total_run_value = sum(run_value), se_run_value = sd(run_value)/ sqrt(N))
```

```
## # A tibble: 2 x 5
##    RISP     N avg_run_value total_run_value se_run_value
##   <dbl> <int>         <dbl>           <dbl>        <dbl>
## 1     0   543        0.0363            19.7       0.0164
## 2     1   168        0.0802            13.5       0.0537
```

However, Jose Altuve ranked 75th among roughly 200 full time players in the difference in run value with RISP and with runners not in RISP.

We have mixed results. Can check out bref for a complete breakdown

```
dat2016 %>% inner_join(People %>% select(nameFirst, nameLast, retroID),
             by = c("BAT_ID" = "retroID")) %>%
  filter( BAT_EVENT_FL == TRUE) %>%
  mutate(RISP = ifelse(!BASES %in% c("100","000"),1,0)) %>%
  group_by(BAT_ID) %>%
  summarise(nameFirst = unique(nameFirst),
            nameLast = unique(nameLast),
            N = n(),
            diff = mean(run_value[which(RISP == 1)]) - mean(run_value[which(RISP == 0)])) %>%
  filter(N >= 400) %>% arrange(desc(diff)) %>% slice(73:77)
```

```
## # A tibble: 5 x 5
##   BAT_ID   nameFirst nameLast      N   diff
##   <chr>    <chr>     <chr>     <int>  <dbl>
## 1 martv001 Victor    Martinez    608 0.0454
## 2 keplm001 Max       Kepler      445 0.0448
## 3 altuj001 Jose      Altuve      711 0.0440
## 4 lemad001 DJ        LeMahieu    634 0.0427
## 5 ortid001 David     Ortiz       623 0.0415
```

Altuve's situational OPS

The calculation on Altuve's situational OPS will involve knowledge of the retrosheet event codes. These codes are included as a comment in this code chunk in the accompanying .Rmd file

```
altuve %>%
  select(BASES, EVENT_CD, NEW.BASES, NOUTS, Outs.Inning, OUTS_CT) %>%
  filter(EVENT_CD %in% c(2,3,14,15,16,18,19,20,21,22,23)) %>%
  mutate(RISP = ifelse(!BASES %in% c("100","000"),1,0)) %>%
  group_by(RISP) %>%
  summarise(n = n(),
            AB = n - sum(EVENT_CD == 14) -
              sum(EVENT_CD == 15) -
              sum(EVENT_CD == 16),
            H = sum(EVENT_CD %in% 20:23),
            OBP_noSF = (H + sum(EVENT_CD %in% 14:16)) /
              (AB + sum(EVENT_CD %in% 14:16)),
            SLG = (sum(EVENT_CD == 20) +
              2 * sum(EVENT_CD == 21) +
              3 * sum(EVENT_CD == 22) +
              4 * sum(EVENT_CD == 23))/AB,
            OPS_noSF = OBP_noSF + SLG)
```

```
## # A tibble: 2 x 7
##    RISP     n    AB     H OBP_noSF   SLG OPS_noSF
##   <dbl> <int> <int> <int>    <dbl> <dbl>    <dbl>
## 1     0   543   503   163    0.374 0.511    0.885
## 2     1   168   143    50    0.446 0.559    1.01
```

We now compute the situational OPS split (OPS when RISP minus OPS when runners not in scoring position) for every player, and find Altuve's rank by this metric. First, some data transformation.

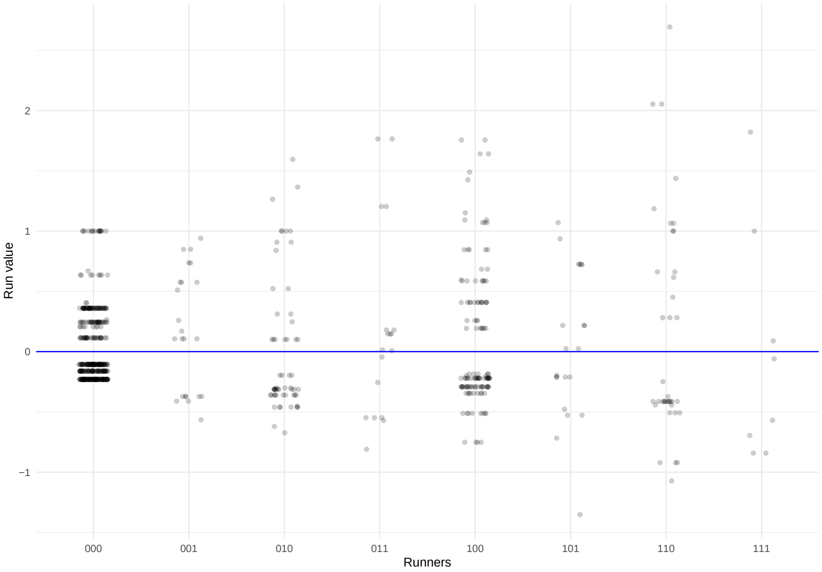
```
dat2016_OPS = dat2016 %>%
  inner_join(People %>% select(nameFirst, nameLast, retroID),
             by = c("BAT_ID" = "retroID")) %>%
  filter( BAT_EVENT_FL == TRUE) %>%
  mutate(RISP = ifelse(!BASES %in% c("100","000"),1,0)) %>%
  group_by(BAT_ID, RISP) %>%
  summarise(nameFirst = unique(nameFirst),
            nameLast = unique(nameLast),
            n = n(),
            AB = n - sum(EVENT_CD == 14) -
              sum(EVENT_CD == 15) -
              sum(EVENT_CD == 16),
            H = sum(EVENT_CD %in% 20:23),
            OBP_noSF = (H + sum(EVENT_CD %in% 14:16)) /
              (AB + sum(EVENT_CD %in% 14:16)),
            SLG = (sum(EVENT_CD == 20) +
              2 * sum(EVENT_CD == 21) +
              3 * sum(EVENT_CD == 22) +
              4 * sum(EVENT_CD == 23))/AB,
            OPS_noSF = OBP_noSF + SLG) %>%
  filter(n_distinct(RISP) == 2)
```

Jose Altuve's situational OPS split difference ranks 44th (among batters with at least 400 batting events).

```r
dat2016_OPS %>%
  mutate(OPS_noSF_diff = OPS_noSF[RISP == 1] - OPS_noSF[RISP == 0]) %>%
  summarise(nameFirst = unique(nameFirst),
            nameLast = unique(nameLast),
            n = sum(n),
            OPS_noSF_diff = unique(OPS_noSF_diff)) %>%
  filter(n >= 400) %>%
  arrange(desc(OPS_noSF_diff)) %>%
  slice(42:46)
```

```
## # A tibble: 5 x 5
##   BAT_ID   nameFirst nameLast     n OPS_noSF_diff
##   <chr>    <chr>     <chr>    <int>         <dbl>
## 1 solay001 Yangervis Solarte    441         0.124
## 2 franm004 Maikel    Franco     629         0.123
## 3 altuj001 Jose      Altuve     711         0.121
## 4 forsl001 Logan     Forsythe   565         0.120
## 5 aybae001 Erick     Aybar      458         0.120
```

```
ggplot(altuve, aes(BASES, run_value)) +
  geom_jitter(width = 0.15, alpha = 0.20) +
  geom_hline(yintercept = 0, color = "blue") +
  xlab("Runners") + ylab("Run value") +
  theme_minimal()
```



Example: All batters

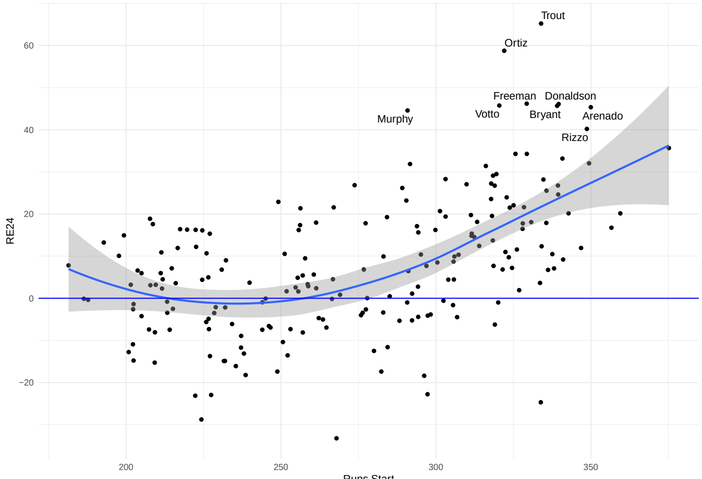
We create a new variable: total starting runs potential `Runs.Start`. This variable sums the run values of all base-out states at the start of a batter's plate appearance.

```
runs = dat2016 %>%
  filter(BAT_EVENT_FL == TRUE) %>%
  inner_join(People, by = c("BAT_ID" = "retroID")) %>%
  group_by(BAT_ID) %>%
  summarise(RE24 = sum(run_value),
            PA = length(run_value),
            Runs.Start = sum(Runs.State),
            nameLast = unique(nameLast)) %>%
  filter(PA >= 400)

head(runs)
```

```
## # A tibble: 6 x 5
##   BAT_ID     RE24    PA Runs.Start nameLast
##   <chr>     <dbl> <int>      <dbl> <chr>
## 1 abrej003  12.3    693       334. Abreu
## 2 alony001  -6.94   528       247. Alonso
## 3 altuj001  33.2    711       341. Altuve
## 4 andet001 -10.9    428       202. Anderson
## 5 andre001  17.4    567       256. Andrus
## 6 aokin001  -2.12   466       229. Aoki
```

Batters with larger values of `Runs.Start` tend to have larger runs contributions. Batters with at least 40 RE24 are labeled.

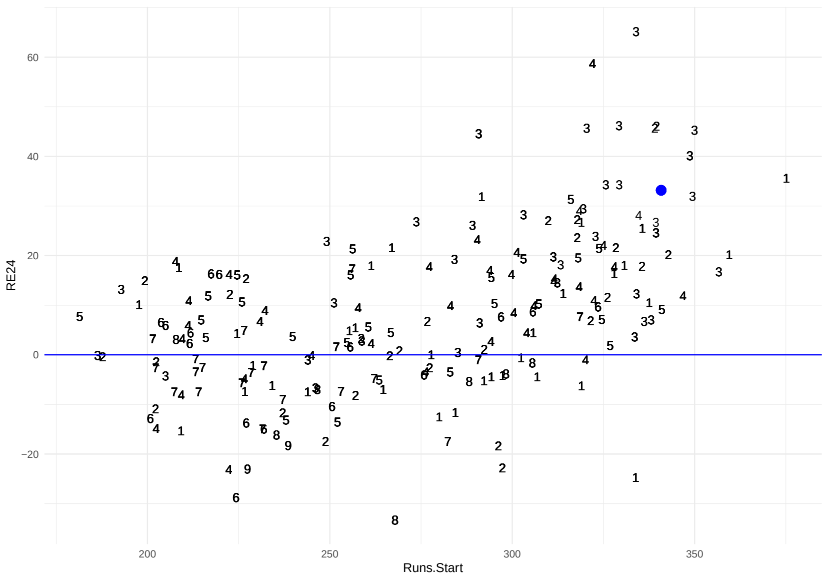
```
library(ggrepel)
ggplot(runs, aes(Runs.Start, RE24)) +
  geom_point() + geom_smooth() +
  geom_hline(yintercept = 0, color = "blue") +
  geom_text_repel(data = filter(runs, RE24 >= 40), aes(label = nameLast)) +
  theme_minimal()
```



Simple lineup analysis

Managers like to put their best hitters near the middle third of the lineup.

```
regulars = dat2016 %>% inner_join(runs, by = "BAT_ID")
positions = regulars %>% group_by(BAT_ID, BAT_LINEUP_ID) %>%
  summarise(N = n()) %>% arrange(desc(N)) %>%
  mutate(Position = first(BAT_LINEUP_ID))
runs = runs %>% inner_join(positions, by = "BAT_ID")

ggplot(runs, aes(Runs.Start, RE24, label = Position)) +
  geom_text() +
  geom_hline(yintercept = 0, color = "blue") +
  geom_point(data = filter(runs, BAT_ID == altuve.id),
             size = 4, shape = 16, color = "blue") +
  theme_minimal()
```



RE24
60
40
20
0
−20
200
250
300
350
Runs.Start

Value of home runs

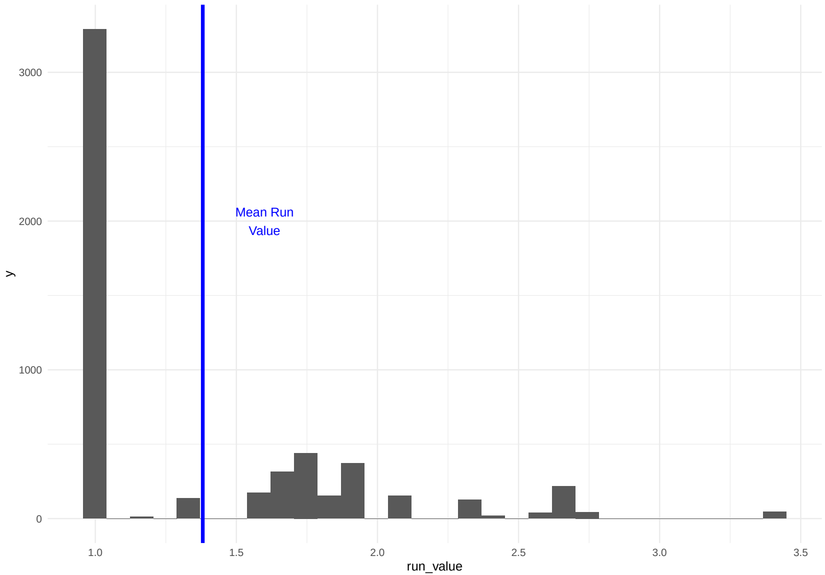
```
## get home runs
home_runs = dat2016 %>% filter(EVENT_CD == 23)

home_runs_N = home_runs %>% group_by(STATE) %>%
  mutate(Outs = substr(STATE, 5, 5)) %>%
  arrange(Outs) %>%
  summarise(Outs = unique(Outs), N = n()) %>%
  arrange(Outs)

## frequency table of home runs
home_runs_N_mat = matrix(round(home_runs_N$N / sum(home_runs_N$N), 3), 8, 3)
dimnames(home_runs_N_mat)[[1]] = c("000","001","010","011",
                                   "100","101","110","111")
dimnames(home_runs_N_mat)[[2]] = c("0 outs", "1 out", "2 outs")
home_runs_N_mat
```

```
##     0 outs 1 out 2 outs
## 000  0.272 0.172  0.150
## 001  0.002 0.007  0.011
## 010  0.018 0.027  0.028
## 011  0.004 0.007  0.007
## 100  0.057 0.064  0.061
## 101  0.005 0.013  0.011
## 110  0.015 0.023  0.028
## 111  0.003 0.008  0.008
```

```
avg_hr = home_runs %>% summarise(avg_run_value = mean(run_value))
avg_hr
```

```
## # A tibble: 1 x 1
##   avg_run_value
##           <dbl>
## 1          1.38
```

```r
ggplot(home_runs, aes(run_value)) +
  geom_histogram() +
  geom_vline(data = avg_hr, aes(xintercept = avg_run_value),
             color = "blue", size = 1.5) +
  annotate("text", 1.6, 2000, label = "Mean Run\nValue", color = "blue") +
  theme_minimal()
```



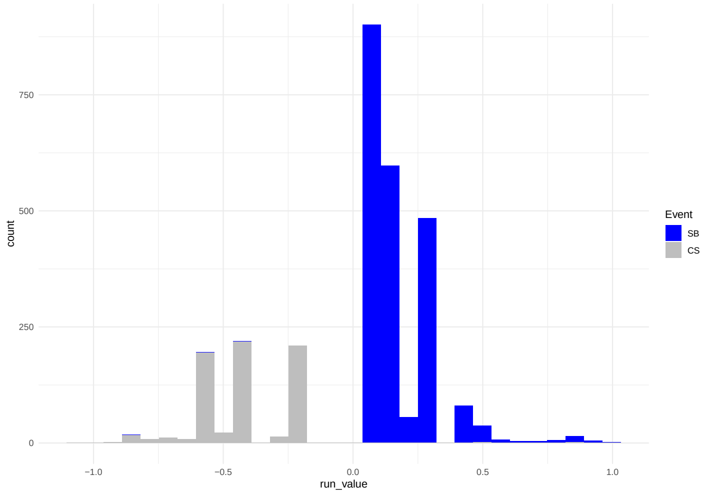
Value of base stealing

```
stealing = dat2016 %>% filter(EVENT_CD %in% c(4,6))
stealing %>% group_by(EVENT_CD) %>% summarise(N = n(),
  avg_run_value = mean(run_value)) %>%
    mutate(pct = N/sum(N))
```

```
## # A tibble: 2 x 4
##   EVENT_CD     N avg_run_value   pct
##      <dbl> <int>         <dbl> <dbl>
## 1        4  2199         0.180 0.756
## 2        6   710        -0.416 0.244
```

Histogram of the run values of all steal attempts during the 2016 season.

```
ggplot(stealing, aes(run_value, fill = factor(EVENT_CD))) +
    geom_histogram() +
    scale_fill_manual(name = "Event", values = c("blue", "grey"),
                                               labels = c("SB", "CS")) +
  theme_minimal()
```



We can compute the marginal break-even success rate needed to justify a stolen base attempt across the 2016 season

$$a * \mathrm{SB}_{\text{avg value}} + (1 - a) * \mathrm{CS}_{\text{avg value}} = 0$$

which implies that

$$a = -\frac{\mathrm{CS}_{\text{avg value}}}{\mathrm{SB}_{\text{avg value}} - \mathrm{CS}_{\text{avg value}}}.$$

From a previous slide we compute

```
a = 0.416 / ( 0.180 + 0.416)
a
```

```
## [1] 0.6979866
```