Run Expectancy (Chapter 5)

Introduction

Prepare the Data

Runs Scored

The data for the 2011 season is housed in the `all2011.csv` file in the `data` folder. The field names are stored in a file called `fields.csv` in the same folder. Let's load them in and set the header with the `fields` file.

```
data2011 <- read.csv("../data/all2011.csv", header=FALSE)
fields <- read.csv("../data/fields.csv")
names(data2011) <- fields[, "Header"]
```

The loaded dataset has 97 attributes and 191864 observations.

What we're after is the average number of runs scored in the rest of the inning with each given combination of outs and runners. That's 3 outs times 2^3 base combinations, for 24 base/out states. We want to make a new variable

$$\text{RUNS.ROI} = \text{Total Runs Scored in Inning} - \text{Current Runs Scored}$$

And average that variable across each base/out state.

```
data2011$RUNS <- data2011$AWAY_SCORE_CT + data2011$HOME_SCORE_CT
```

Defines the current total runs scored at each plate appearance. We'll want to uniquely identify each half inning using a new variable `HALF.INNING`

```
data2011$HALF.INNING <- paste(data2011$GAME_ID, data2011$INN_CT, data2011$BAT_HOME_ID)
```

Any time a runner or batters destination id is greater than 3, a run scores, these are described in the variables `BAT_DEST_ID` and `RUN1_DEST_ID`, with `RUN1` being the runner on first, `RUN2` being on second, etc.

```
data2011$RUNS.SCORED <- (data2011$BAT_DEST_ID > 3) +
                        (data2011$RUN1_DEST_ID > 3) +
                        (data2011$RUN2_DEST_ID > 3) +
                        (data2011$RUN3_DEST_ID > 3)
```

This is next part is incredibly slick. We run two aggregations, one aggregating total runs scored across the innings, i.e. summing all the values of `RUNS.SCORED` across plate appearance, grouped by `HALF.INNING`. Next we do an aggregation using the "[" function, which is the subset function, passing it a value of 1 for its first argument. In other words, evaluate `x[1, ]` for each `x`, grouped by `HALF.INNING`.

```
RUNS.SCORED.INNING <- aggregate(data2011$RUNS.SCORED, list(HALF.INNING=data2011$HALF.INNING), sum)
RUNS.SCORED.START <- aggregate(data2011$RUNS, list(HALF.INNING=data2011$HALF.INNING), "[", 1)
```

Now that we have these two vectors, we can calculate the max runs in a given half inning

```
MAX <- data.frame(HALF.INNING=RUNS.SCORED.START$HALF.INNING)
MAX$x <- RUNS.SCORED.INNING$x + RUNS.SCORED.START$x
data2011 <- merge(data2011, MAX)
N <- ncol(data2011)
names(data2011)[N] <- "MAX.RUNS"

data2011$RUNS.ROI <- data2011$MAX.RUNS - data2011$RUNS
```

And there you have it, total runs in the inning minus current runs, just as we defined it in the beginning.

Base/Out State

To deterimine the base/out states, we first determine the bases

```
RUNNER1 <- ifelse(as.character(data2011[, "BASE1_RUN_ID"]) == "", 0, 1)
RUNNER2 <- ifelse(as.character(data2011[, "BASE2_RUN_ID"]) == "", 0, 1)
RUNNER3 <- ifelse(as.character(data2011[, "BASE3_RUN_ID"]) == "", 0, 1)
```

The above vectors are the same length as our dataframe, with a 0 if the respective base was empty for the plate appearance, and 1 if there was a runner on. We then write a function to combine those three vectors with the outs

```
get.state <- function(runner1, runner2, runner3, outs){
  runners <- paste(runner1, runner2, runner3, sep="")
  paste(runners, outs)
}

data2011$STATE <- get.state(RUNNER1, RUNNER2, RUNNER3, data2011$OUTS_CT)
```

We only care about plays were there is a change in state, so using the variables available to us we'll calculate the state after the play.

```
NRUNNER1 <- as.numeric(data2011$RUN1_DEST_ID == 1 |
                       data2011$BAT_DEST_ID == 1)

NRUNNER2 <- as.numeric(data2011$RUN1_DEST_ID == 2 |
                       data2011$RUN2_DEST_ID == 2 |
                       data2011$BAT_DEST_ID == 2)

NRUNNER3 <- as.numeric(data2011$RUN1_DEST_ID == 3 |
                       data2011$RUN2_DEST_ID == 3 |
                       data2011$RUN3_DEST_ID == 3 |
                       data2011$BAT_DEST_ID == 3)

NOUTS <- data2011$OUTS_CT + data2011$EVENT_OUTS_CT

data2011$NEW.STATE <- get.state(NRUNNER1, NRUNNER2, NRUNNER3, NOUTS)
```

We throw away the plays where there was no change in runs or state – I looked into it, they're mostly errors on foul flies, and errors on picked off runners, where they end up back at the same base.

```
data2011 <- subset(data2011, (STATE != NEW.STATE) | (RUNS.SCORED > 0))
```

Last thing to take care of is walkoffs, where a run is scored but the game ends without getting 3 outs. We can take care of those with `ddply` in the `plyr` library. We create a new dataframe called `data.outs` which is the total outs in each `HALF.INNING`. Then, we merge that back with our main data frame and filter off any that are less than 3.

```
library(plyr)

data.outs <- ddply(data2011, .(HALF.INNING), summarize, Outs.Inning=sum(EVENT_OUTS_CT))
data2011 <- merge(data2011, data.outs)
data2011C <- subset(data2011, Outs.Inning == 3)
```

Expected Runs

At last we can compute the expected runs for each base/out state. Note we parse the grouping variable (state) to get the number of outs and order by outs.

```
RUNS <- aggregate(data2011C$RUNS.ROI, list(data2011C$STATE), mean)
RUNS$Outs <- substr(RUNS$Group, 5, 5)
RUNS <- RUNS[order(RUNS$Outs), ]
```

We can now make a matrix

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

The Matrix

The table is given in Table 1

	0 outs	1 out	2 outs
000	0.47	0.25	0.10
001	1.45	0.94	0.32
010	1.06	0.65	0.31
011	1.93	1.34	0.54
100	0.84	0.50	0.22
101	1.75	1.15	0.49
110	1.41	0.87	0.42
111	2.17	1.47	0.76

Table 1: Run Expectancy Matrix for 2011