R Code for **Teaching Statistics Using Baseball**, 2nd Ed.

Juju and Oliver

This document contains R code to reproduce the plots, tables, and statistics presented in Jim Albert’s *Teaching Statistics Using Baseball*, 2nd Ed. Some of the data is taken from sources other than the author’s web site or his **tsub** package for **R**.

The following code is usually put in the ***setup*** of the **.RMD** file. It is included here so that you can read it if you don’t have access to the ***.RMD*** for this file.

### Install or load the pacman library  
 if (!require(pacman)){  
 install.packages("pacman", dependencies = TRUE)  
 require(pacman)  
 }  
  
 p\_load(dplyr) ### Used by Albert's tsub package  
   
 ### Install or load the tsub package that ontains a few of Albert's functions and all of the book datasets  
 if (!require(tsub)){  
 p\_load(devtools) ### Install or load the devtools package  
 install\_github("bayesball/tsub") ### Install tsub from github  
 p\_load(tsub) ### Load tsub  
 }  
  
 p\_load(Lahman) ### Contains the Lahman data  
 p\_load(lattice) ### Load the lattice graphics package  
 p\_load(ggplot2) ### Load Hadley's ggplot graphics package  
 p\_load(lubridate) ### Used to modify dates. Things like getting the year from mm-dd-yyyy, etc.  
 p\_load(aplpack) ### Pretty stemplots

## Chapter One

Lahman has a site at <https://www.seanlahman.com/baseball-archive/statistics> that has his data in various formats and describes the tables that make up the data. Documentation for the data can be found at <https://www.seanlahman.com/files/database/readme2021.txt>. The data are also available in the **Lahman** package for **R**.

### Rickey Henderson using Lahman’s data

The first step is to install and load the **Lahman** package.

# p\_load is in the pacman package which is installed above  
 p\_load(Lahman)

The Lahman package contains a number of tables and a few functions that help subset the data. For now we will work with Henderson’s batting data.

### Get the Batting data.frame ###  
 data(Batting)  
 ### Figure out what it looks like ###  
 head(Batting)

## playerID yearID stint teamID lgID G AB R H X2B X3B HR RBI SB CS BB SO  
## 1 abercda01 1871 1 TRO NA 1 4 0 0 0 0 0 0 0 0 0 0  
## 2 addybo01 1871 1 RC1 NA 25 118 30 32 6 0 0 13 8 1 4 0  
## 3 allisar01 1871 1 CL1 NA 29 137 28 40 4 5 0 19 3 1 2 5  
## 4 allisdo01 1871 1 WS3 NA 27 133 28 44 10 2 2 27 1 1 0 2  
## 5 ansonca01 1871 1 RC1 NA 25 120 29 39 11 3 0 16 6 2 2 1  
## 6 armstbo01 1871 1 FW1 NA 12 49 9 11 2 1 0 5 0 1 0 1  
## IBB HBP SH SF GIDP  
## 1 NA NA NA NA 0  
## 2 NA NA NA NA 0  
## 3 NA NA NA NA 1  
## 4 NA NA NA NA 0  
## 5 NA NA NA NA 0  
## 6 NA NA NA NA 0

### Some of the Lahman functions require dplyr ###  
 p\_load(dplyr)

Albert’s text works on Rickey Henderson’s 25 seasons. It would be nice if we could pull this information out of the Lahman data to confirm the values and compute a few statistics.

As a first step, we need to find Henderson’s player information.

data(Master)

## Warning in data(Master): data set 'Master' not found

playerInfo(nameLast = "Henderson")

## playerID nameFirst nameLast  
## 7931 hendebe01 Bernie Henderson  
## 7932 hendebi01 Bill Henderson  
## 7933 hendebi99 Bill Henderson  
## 7934 hendeda01 Dave Henderson  
## 7935 hendeed01 Ed Henderson  
## 7936 hendeha01 Hardie Henderson  
## 7937 hendeji01 Jim Henderson  
## 7938 hendejo01 Joe Henderson  
## 7939 hendeke01 Ken Henderson  
## 7940 henderi01 Rickey Henderson  
## 7941 hendero01 Rod Henderson  
## 7942 hendest01 Steve Henderson

playerInfo(nameFirst = "Rickey")

## playerID nameFirst nameLast  
## 3298 clarkri01 Rickey Clark  
## 3846 cradlri01 Rickey Cradle  
## 7940 henderi01 Rickey Henderson  
## 9487 keetori01 Rickey Keeton

playerInfo("henderi01")

## playerID nameFirst nameLast  
## 7940 henderi01 Rickey Henderson

Knowing that Rickey Henderson is “henderi01” means that we can subset his information.

batting = battingStats()  
 Henderson = batting[Batting$playerID=="henderi01",]  
 head(Henderson)

## playerID yearID stint teamID lgID G AB R H X2B X3B HR RBI SB CS  
## 56777 henderi01 1979 1 OAK AL 89 351 49 96 13 3 1 26 33 11  
## 57733 henderi01 1980 1 OAK AL 158 591 111 179 22 4 9 53 100 26  
## 58676 henderi01 1981 1 OAK AL 108 423 89 135 18 7 6 35 56 22  
## 59632 henderi01 1982 1 OAK AL 149 536 119 143 24 4 10 51 130 42  
## 60637 henderi01 1983 1 OAK AL 145 513 105 150 25 7 9 48 108 19  
## 61632 henderi01 1984 1 OAK AL 142 502 113 147 27 4 16 58 66 18  
## BB SO IBB HBP SH SF GIDP BA PA TB SlugPct OBP OPS BABIP  
## 56777 34 39 0 2 8 3 4 0.274 398 118 0.336 0.338 0.674 0.303  
## 57733 117 54 7 5 6 3 6 0.303 722 236 0.399 0.420 0.819 0.320  
## 58676 64 68 4 2 0 4 7 0.319 493 185 0.437 0.408 0.845 0.365  
## 59632 116 94 1 2 0 2 5 0.267 656 205 0.382 0.398 0.780 0.306  
## 60637 103 80 8 4 1 1 11 0.292 622 216 0.421 0.414 0.835 0.332  
## 61632 86 81 1 5 1 3 7 0.293 597 230 0.458 0.399 0.857 0.321

dim(Henderson)

## [1] 29 29

table(Henderson$yearID)

##   
## 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994   
## 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1   
## 1995 1996 1997 1998 1999 2000 2001 2002 2003   
## 1 1 2 1 1 2 1 1 1

These values do *not* correspond to the yearly values in Table 1.1 of Albert’s book. The years 1989, 1993, 1997, and 2000 include information for two teams. Albert’s table has combined the information.

At this point, we will just import Albert’s Table 1.1 data. Later we will see how to use R to compute yearly information. The first method is to import Albert’s comma seperated value (CSV) file. These files can either be local (I have mine stored in a folder CSV within the same folder as this RMD file) or we can find the file on the web. Both methods are shown below.

### Read the local file  
 Henderson.local <- read.csv("CSV/case\_1\_1.csv")  
 head(Henderson.local, 13)

## Year Age Tm G AB R H X2B X3B HR RBI SB CS BB SO BA OBP SLG  
## 1 1979 20 OAK 89 351 49 96 13 3 1 26 33 11 34 39 0.274 0.338 0.336  
## 2 1980 21 OAK 158 591 111 179 22 4 9 53 100 26 117 54 0.303 0.420 0.399  
## 3 1981 22 OAK 108 423 89 135 18 7 6 35 56 22 64 68 0.319 0.408 0.437  
## 4 1982 23 OAK 149 536 119 143 24 4 10 51 130 42 116 94 0.267 0.398 0.382  
## 5 1983 24 OAK 145 513 105 150 25 7 9 48 108 19 103 80 0.292 0.414 0.421  
## 6 1984 25 OAK 142 502 113 147 27 4 16 58 66 18 86 81 0.293 0.399 0.458  
## 7 1985 26 NYY 143 547 146 172 28 5 24 72 80 10 99 65 0.314 0.419 0.516  
## 8 1986 27 NYY 153 608 130 160 31 5 28 74 87 18 89 81 0.263 0.358 0.469  
## 9 1987 28 NYY 95 358 78 104 17 3 17 37 41 8 80 52 0.291 0.423 0.497  
## 10 1988 29 NYY 140 554 118 169 30 2 6 50 93 13 82 54 0.305 0.394 0.399  
## 11 1989 30 TOT 150 541 113 148 26 3 12 57 77 14 126 68 0.274 0.411 0.399  
## 12 1990 31 OAK 136 489 119 159 33 3 28 61 65 10 97 60 0.325 0.439 0.577  
## 13 1991 32 OAK 134 470 105 126 17 1 18 57 58 18 98 73 0.268 0.400 0.423  
## OPS  
## 1 0.675  
## 2 0.820  
## 3 0.845  
## 4 0.780  
## 5 0.835  
## 6 0.857  
## 7 0.934  
## 8 0.827  
## 9 0.920  
## 10 0.793  
## 11 0.810  
## 12 1.016  
## 13 0.823

tail(Henderson.local, 13)

## Year Age Tm G AB R H X2B X3B HR RBI SB CS BB SO BA OBP SLG  
## 13 1991 32 OAK 134 470 105 126 17 1 18 57 58 18 98 73 0.268 0.400 0.423  
## 14 1992 33 OAK 117 396 77 112 18 3 15 46 48 11 95 56 0.283 0.426 0.457  
## 15 1993 34 TOT 134 481 114 139 22 2 21 59 53 8 120 65 0.289 0.432 0.474  
## 16 1994 35 OAK 87 296 66 77 13 0 6 20 22 7 72 45 0.260 0.411 0.365  
## 17 1995 36 OAK 112 407 67 122 31 1 9 54 32 10 72 66 0.300 0.407 0.447  
## 18 1996 37 SDP 148 465 110 112 17 2 9 29 37 15 125 90 0.241 0.410 0.344  
## 19 1997 38 TOT 120 403 84 100 14 0 8 34 45 8 97 85 0.248 0.400 0.342  
## 20 1998 39 OAK 152 542 101 128 16 1 14 57 66 13 118 114 0.236 0.376 0.347  
## 21 1999 40 NYM 121 438 89 138 30 0 12 42 37 14 82 82 0.315 0.423 0.466  
## 22 2000 41 TOT 123 420 75 98 14 2 4 32 36 11 88 75 0.233 0.368 0.305  
## 23 2001 42 SDP 123 379 70 86 17 3 8 42 25 7 81 84 0.227 0.366 0.351  
## 24 2002 43 BOS 72 179 40 40 6 1 5 16 8 2 38 47 0.223 0.369 0.352  
## 25 2003 44 LAD 30 72 7 15 1 0 2 5 3 0 11 16 0.208 0.321 0.306  
## OPS  
## 13 0.823  
## 14 0.883  
## 15 0.906  
## 16 0.776  
## 17 0.855  
## 18 0.754  
## 19 0.742  
## 20 0.723  
## 21 0.889  
## 22 0.673  
## 23 0.717  
## 24 0.721  
## 25 0.627

dim(Henderson.local)

## [1] 25 19

### Read from the web  
 Henderson.web <- read.csv("http://facweb1.redlands.edu/fac/jim\_bentley/Data/FYS28/TSUB2/CSV/case\_1\_1.csv")  
 head(Henderson.web)

## Year Age Tm G AB R H X2B X3B HR RBI SB CS BB SO BA OBP SLG  
## 1 1979 20 OAK 89 351 49 96 13 3 1 26 33 11 34 39 0.274 0.338 0.336  
## 2 1980 21 OAK 158 591 111 179 22 4 9 53 100 26 117 54 0.303 0.420 0.399  
## 3 1981 22 OAK 108 423 89 135 18 7 6 35 56 22 64 68 0.319 0.408 0.437  
## 4 1982 23 OAK 149 536 119 143 24 4 10 51 130 42 116 94 0.267 0.398 0.382  
## 5 1983 24 OAK 145 513 105 150 25 7 9 48 108 19 103 80 0.292 0.414 0.421  
## 6 1984 25 OAK 142 502 113 147 27 4 16 58 66 18 86 81 0.293 0.399 0.458  
## OPS  
## 1 0.675  
## 2 0.820  
## 3 0.845  
## 4 0.780  
## 5 0.835  
## 6 0.857

dim(Henderson.web)

## [1] 25 19

Albert has made it easy to load all the data that is used in TSUB2. By loading his **TSUB** package, all of the case and exercise data are made available.

p\_load(devtools) ### Install or load the devtools package  
 install\_github("bayesball/tsub") ### Install tsub from github

## Skipping install of 'tsub' from a github remote, the SHA1 (563eded6) has not changed since last install.  
## Use `force = TRUE` to force installation

p\_load(tsub) ### Load tsub

Now that the **tsub** package is installed, we can call the different datasets.

Henderson.tsub <- case\_1\_1  
 head(Henderson.tsub)

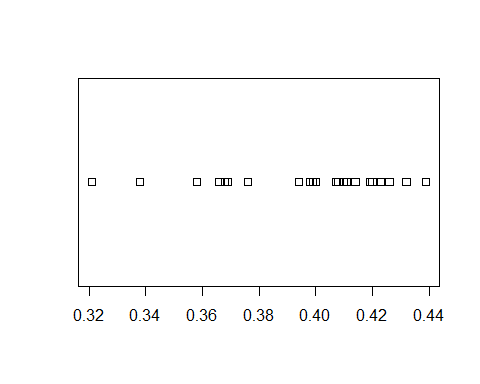
## Year Age Tm G AB R H X2B X3B HR RBI SB CS BB SO BA OBP SLG  
## 1 1979 20 OAK 89 351 49 96 13 3 1 26 33 11 34 39 0.274 0.338 0.336  
## 2 1980 21 OAK 158 591 111 179 22 4 9 53 100 26 117 54 0.303 0.420 0.399  
## 3 1981 22 OAK 108 423 89 135 18 7 6 35 56 22 64 68 0.319 0.408 0.437  
## 4 1982 23 OAK 149 536 119 143 24 4 10 51 130 42 116 94 0.267 0.398 0.382  
## 5 1983 24 OAK 145 513 105 150 25 7 9 48 108 19 103 80 0.292 0.414 0.421  
## 6 1984 25 OAK 142 502 113 147 27 4 16 58 66 18 86 81 0.293 0.399 0.458  
## OPS  
## 1 0.675  
## 2 0.820  
## 3 0.845  
## 4 0.780  
## 5 0.835  
## 6 0.857

These values compare well with those presented in Table 1.1 of the text and the data as imported above.

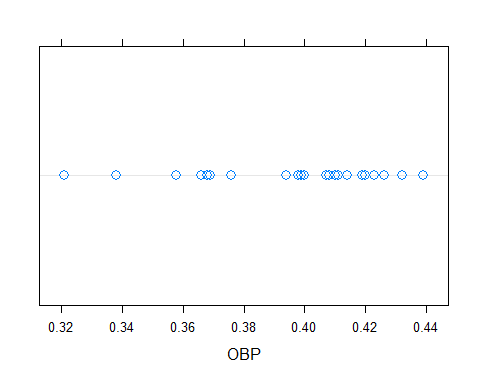
### Henderson’s On Base Percentage

*TSUB2* provides a dotplot of Henderson’s ***OBP*** during his first 23 years. We can create this plot in a few ways.

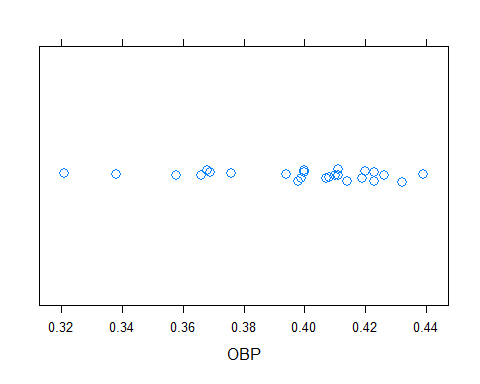
### First use base graphics  
 stripchart(case\_1\_1$OBP)



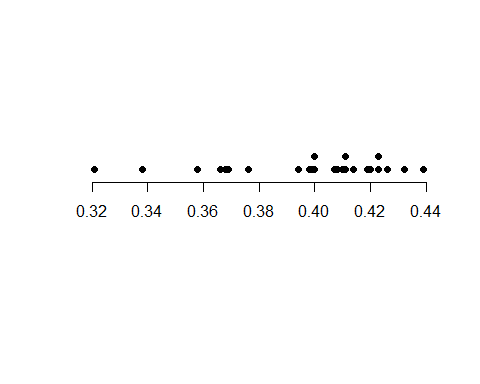
### Use the lattice library. cex is the character size multiplier   
 ### Using pch=1 chooses open circles which better show overlapped data.   
 p\_load(lattice)  
 dotplot(~OBP, data=case\_1\_1, cex=1.25, pch=1)



stripplot(~OBP, data=case\_1\_1, cex=1.25, pch=1, jitter=TRUE, factor=1.5)



### Albert likes the minitab version of the plotrix dotplot  
 p\_load(plotrix)  
 dotplot.mtb(case\_1\_1$OBP)

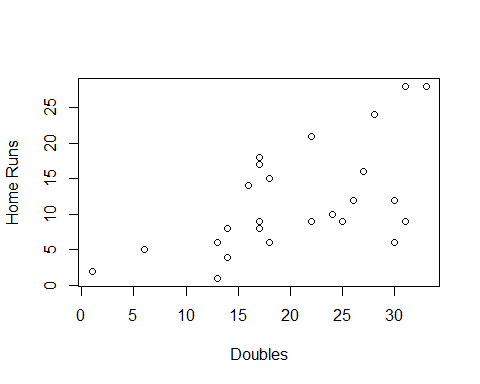


These plots are similar to Figure 1.1 in the text.

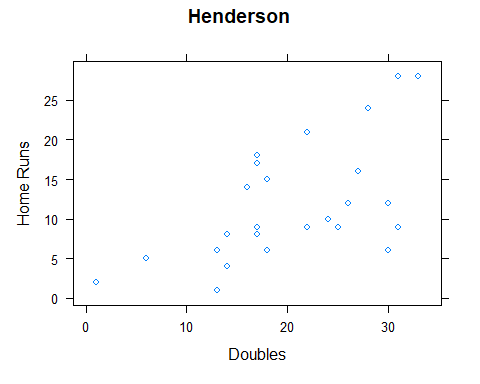
### Henderson’s Doubles versus Home Runs

The text compares Henderson’s home run production (***HR***) against his doubles (***X2B***) using a scatterplot. There are multiple methods for generating scatterplots in R. We will look at three.

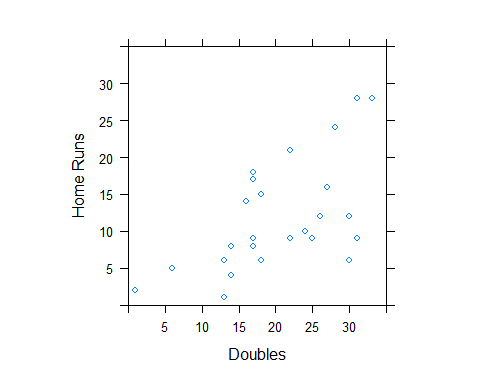
### Using base R we get ###  
 plot(case\_1\_1$X2B, case\_1\_1$HR, xlab="Doubles", ylab="Home Runs")



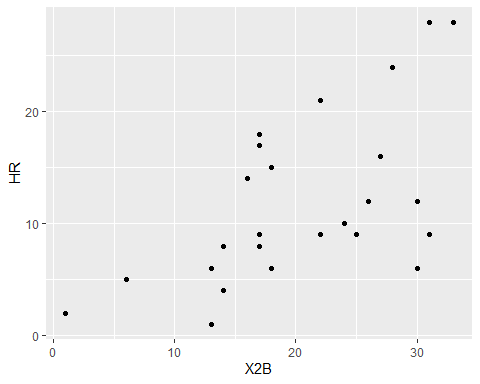
### Using lattice graphics  
 p\_load(lattice)  
 xyplot(HR~X2B, data=case\_1\_1, xlab="Doubles", ylab="Home Runs", main="Henderson")



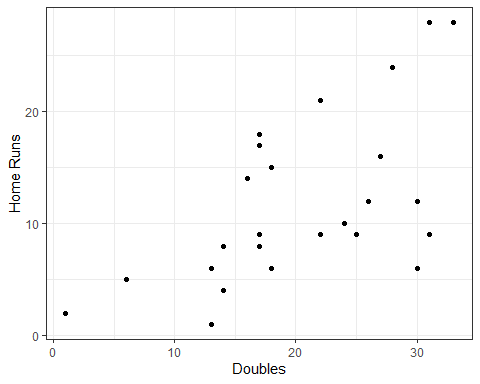
xyplot(HR~X2B, data=case\_1\_1, xlab="Doubles", ylab="Home Runs", xlim=c(0:35), ylim=c(0:35), aspect=1)



### Use ggplot ###  
 p\_load(ggplot2)  
 p = ggplot(case\_1\_1, aes(X2B, HR)) ### Define data.frame and variables  
 p + geom\_point() ### Request a point plot



p + geom\_point() + theme\_bw() + xlab("Doubles") + ylab("Home Runs")



### Simulating Henderson’s Spinner Using R

Hendersons hitting statistics for 1990 were

### Grab Henderon's batting data and get totals by year using dplyr and piping to make it easy  
 p\_load(dplyr)  
 Henderson.total <- Batting %>% ### Get all of the batting data  
 filter(playerID == "henderi01") %>% ### Subset for Henderson  
 group\_by(yearID) %>% ### Work on year subsets from here on out  
 summarize(  
 teams = 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),  
 IBB = sum(IBB),  
 HBP = sum(HBP),  
 SH = sum(SH),  
 SF = sum(SF),  
 GIDP = sum(GIDP)  
 )  
   
 names(Henderson.total)

## [1] "yearID" "teams" "G" "AB" "R" "H" "X2B" "X3B"   
## [9] "HR" "RBI" "SB" "CS" "BB" "SO" "IBB" "HBP"   
## [17] "SH" "SF" "GIDP"

### Now add computed performance statistics for each year... e.g. BA, OBP, etc.  
 Henderson.total <- battingStats(Henderson.total)  
   
 ### Check the data against Albert's Table 1.1  
 Henderson.total %>%   
 filter(teams == 2)

## yearID teams G AB R H X2B X3B HR RBI SB CS BB SO IBB HBP SH SF GIDP  
## 1 1989 2 150 541 113 148 26 3 12 57 77 14 126 68 5 3 0 4 8  
## 2 1993 2 134 481 114 139 22 2 21 59 53 8 120 65 7 4 1 4 9  
## 3 1997 2 120 403 84 100 14 0 8 34 45 8 97 85 2 6 1 2 10  
## 4 2000 2 123 420 75 98 14 2 4 32 36 11 88 75 1 4 3 4 11  
## BA PA TB SlugPct OBP OPS BABIP  
## 1 0.274 674 216 0.399 0.411 0.810 0.292  
## 2 0.289 610 228 0.474 0.432 0.906 0.296  
## 3 0.248 509 138 0.342 0.400 0.742 0.295  
## 4 0.233 519 128 0.305 0.368 0.673 0.272

The values for the four years when Henderson played for two teams match those in Table 1.1.

Albert uses only the 1990 season when creating Henderson’s spinner. We can do the same.

spinner <- Henderson.total %>% ### Grab the total batting data from above  
 filter(yearID == 1990) %>% ### Keep only the 1990 season  
 mutate( ### Create a few variables  
 X1B = H - (X2B + X3B + HR), ### Singles are hits minus doubles, trips, and homers  
 Out = AB - (H + BB), ### Outs are at bats that aren't hits or walks  
 Walk = BB ### Walks are base on balls  
 )  
   
 ### Check the counts  
 spinner

## yearID teams G AB R H X2B X3B HR RBI SB CS BB SO IBB HBP SH SF GIDP  
## 1 1990 1 136 489 119 159 33 3 28 61 65 10 97 60 2 4 2 2 13  
## BA PA TB SlugPct OBP OPS BABIP X1B Out Walk  
## 1 0.325 594 282 0.577 0.439 1.016 0.325 95 233 97

### Probabilities are counts divided by at bats... sort of. We ignore HBP, interference, etc.  
 spinner.prob = spinner[c("X1B","X2B","X3B","HR","Walk","Out")] ### Keep only those outcomes we care about  
 spinner.prob = spinner.prob/sum(spinner.prob)  
 spinner.prob

## X1B X2B X3B HR Walk Out  
## 1 0.194274 0.06748466 0.006134969 0.05725971 0.198364 0.4764826

sum(spinner.prob)

## [1] 1

### Turn the data frame into a vector so that we can plot it  
 dim(spinner.prob)

## [1] 1 6

spinner.prob = as.vector(t(spinner.prob))  
 dim(spinner.prob)

## NULL

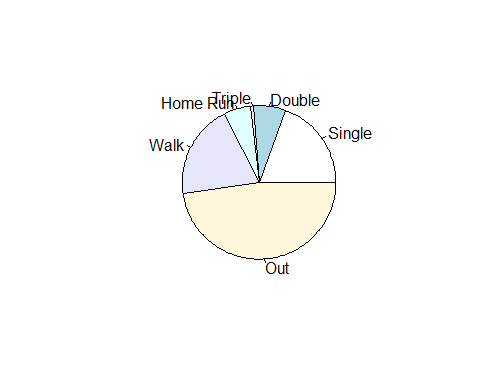
length(spinner.prob)

## [1] 6

spinner.prob

## [1] 0.194274029 0.067484663 0.006134969 0.057259714 0.198364008 0.476482618

### Create Figure 1.3  
 pie(spinner.prob,labels=c("Single", "Double","Triple","Home Run","Walk","Out"))



Now, Albert has created a nice spinner, but he hasn’t used it. Let’s see what we get for Henderson if we let him bat (randomly) 179 times — the number of at bats that he had in 2002).

outcomes = c("X1B","X2B","X3B","HR","Walk","Out") ### Create a list of outcomes with probs in spinner.prob  
 spin2002 = sample(outcomes, 179, replace=TRUE, prob=spinner.prob) ### Sample from outcomes using spinner.probs  
 spin2002 = factor(spin2002,levels=c("X1B","X2B","X3B","HR","Walk","Out")) ### Make the data categorical  
 table(spin2002) ### Compute the number of each type of outcome

## spin2002  
## X1B X2B X3B HR Walk Out   
## 37 10 0 10 29 93

### List the stats for 2002 and 2003  
 Henderson.total %>%   
 filter(yearID >= 2002) %>%  
 mutate(  
 Walk = BB + IBB + HBP,  
 X1B = H - (X2B + X3B + HR),  
 Out = AB - (H + Walk)  
 ) %>%  
 select(yearID, AB, H, X1B, X2B, X3B, HR, Walk, Out)

## yearID AB H X1B X2B X3B HR Walk Out  
## 1 2002 179 40 28 6 1 5 42 97  
## 2 2003 72 15 12 1 0 2 12 45

### Computing Basic Measures of Performance

We have already computed a few measures of a batter’s performance using functions defined by others. In the previous section, we computed a few of these by hand. We can confirm that the functions and raw code provide the same results.

### Slugging percentage  
 Hend7983 = Henderson.total %>%  
 filter(yearID <= 1983) %>% ### Grab Henderson's 1979 to 1983 years  
 mutate( ### Create a few batting stats  
 x1b = H - (X2B + X3B + HR),  
 tb = 1\*x1b + 2\*X2B + 3\*X3B + 4\*HR,  
 slg = round(tb/AB,3)  
 )  
   
 temp <- Henderson.total %>%   
 filter(yearID <= 1983) %>%  
 select(yearID, SlugPct)  
   
 cbind(head(Hend7983[,c(1,6:9,27:29)]), temp$SlugPct)

## yearID H X2B X3B HR x1b tb slg temp$SlugPct  
## 1 1979 96 13 3 1 79 118 0.336 0.336  
## 2 1980 179 22 4 9 144 236 0.399 0.399  
## 3 1981 135 18 7 6 104 185 0.437 0.437  
## 4 1982 143 24 4 10 105 205 0.382 0.382  
## 5 1983 150 25 7 9 109 216 0.421 0.421

Similar computations can be used to compute pitching statistics. We look at Orel Hershiser’s AL years.

### Start a search for Hershier's player ID  
 playerInfo("hers")

## playerID nameFirst nameLast  
## 5420 etherse01 Seth Etherton  
## 8110 hershea01 Earl Hersh  
## 8111 hershfr01 Frank Hershey  
## 8112 hershmi01 Mike Hershberger  
## 8113 hershor01 Orel Hershiser  
## 8114 hershwi01 Willard Hershberger

### Figure out what's in the pitching data.frame  
 names(Pitching)

## [1] "playerID" "yearID" "stint" "teamID" "lgID" "W"   
## [7] "L" "G" "GS" "CG" "SHO" "SV"   
## [13] "IPouts" "H" "ER" "HR" "BB" "SO"   
## [19] "BAOpp" "ERA" "IBB" "WP" "HBP" "BK"   
## [25] "BFP" "GF" "R" "SH" "SF" "GIDP"

### Subset out Hershiser's AL years  
 HershAL = Pitching %>% ### Grab all the pitching data  
 filter(playerID == "hershor01" & lgID == "AL") ### subset hor Orel and AL  
 head(HershAL)

## playerID yearID stint teamID lgID W L G GS CG SHO SV IPouts H ER HR BB  
## 1 hershor01 1995 1 CLE AL 16 6 26 26 1 1 0 502 151 72 21 51  
## 2 hershor01 1996 1 CLE AL 15 9 33 33 1 0 0 618 238 97 21 58  
## 3 hershor01 1997 1 CLE AL 14 6 32 32 1 0 0 586 199 97 26 69  
## SO BAOpp ERA IBB WP HBP BK BFP GF R SH SF GIDP  
## 1 111 0.244 3.87 1 3 5 0 683 0 76 3 4 26  
## 2 125 0.287 4.24 4 11 12 1 908 0 115 5 4 24  
## 3 107 0.272 4.47 2 11 11 0 826 0 105 6 8 31

### Compute his effective "complete"" innings pitched based upon total outs  
 HershAL$ip = HershAL$IPouts/3  
 ### Convert innings to games  
 HershAL$gp = HershAL$ip/9  
 ### As is indicated in TSUB, ERA is earned runs over total games pitched  
 HershAL$era = round(HershAL$ER/HershAL$gp, 4)  
 ### Now compare  
 HershAL[,c("yearID","teamID","IPouts","ER","ip","gp","ERA","era")]

## yearID teamID IPouts ER ip gp ERA era  
## 1 1995 CLE 502 72 167.3333 18.59259 3.87 3.8725  
## 2 1996 CLE 618 97 206.0000 22.88889 4.24 4.2379  
## 3 1997 CLE 586 97 195.3333 21.70370 4.47 4.4693

Not so amazingly, the internal value, **ERA**, is equal to our computed value, **era**.

## Chapter Two

The generation of the graphs and statistics that are presented in the discussion of Case 2.1 is simplified by the use of R. The data from the 2014 MLB season is obtainable from Lahman.

p\_load(Lahman, dplyr)  
   
 MLB2014 = Teams %>% ### Grab the Teams dataframe  
 filter(yearID==2014 & lgID %in% c("AL","NL")) %>% ### Subset 2014 and AL or NL  
 arrange(name) ### Sort by league and wins withing league  
 ### Note: %in% checks if a value is in a list  
   
 head(MLB2014)

## yearID lgID teamID franchID divID Rank G Ghome W L DivWin WCWin LgWin  
## 1 2014 NL ARI ARI W 5 162 81 64 98 N N N  
## 2 2014 NL ATL ATL E 2 162 81 79 83 N N N  
## 3 2014 AL BAL BAL E 1 162 81 96 66 Y N N  
## 4 2014 AL BOS BOS E 5 162 81 71 91 N N N  
## 5 2014 NL CHN CHC C 5 162 81 73 89 N N N  
## 6 2014 AL CHA CHW C 4 162 81 73 89 N N N  
## WSWin R AB H X2B X3B HR BB SO SB CS HBP SF RA ER ERA CG SHO SV  
## 1 N 615 5552 1379 259 47 118 398 1165 86 33 43 36 742 683 4.26 2 4 35  
## 2 N 573 5468 1316 240 22 123 472 1369 95 33 43 27 597 547 3.38 5 13 54  
## 3 N 705 5596 1434 264 16 211 401 1285 44 20 62 36 593 557 3.43 3 13 53  
## 4 N 634 5551 1355 282 20 123 535 1337 63 25 68 52 715 653 4.01 3 7 36  
## 5 N 614 5508 1315 270 31 157 442 1477 65 40 54 41 707 636 3.91 1 11 37  
## 6 N 660 5543 1400 279 32 155 417 1362 85 36 60 38 758 687 4.29 3 6 36  
## IPouts HA HRA BBA SOA E DP FP name  
## 1 4333 1467 154 469 1278 101 147 0.983 Arizona Diamondbacks  
## 2 4365 1369 121 472 1301 85 143 0.986 Atlanta Braves  
## 3 4384 1342 151 472 1174 87 156 0.986 Baltimore Orioles  
## 4 4397 1458 154 482 1213 92 155 0.985 Boston Red Sox  
## 5 4390 1398 115 504 1311 103 137 0.983 Chicago Cubs  
## 6 4323 1468 140 557 1152 107 170 0.982 Chicago White Sox  
## park attendance BPF PPF teamIDBR teamIDlahman45  
## 1 Chase Field 2073730 102 102 ARI ARI  
## 2 Turner Field 2354305 99 99 ATL ATL  
## 3 Oriole Park at Camden Yards 2464473 100 100 BAL BAL  
## 4 Fenway Park II 2956089 102 101 BOS BOS  
## 5 Wrigley Field 2652113 103 104 CHC CHN  
## 6 U.S. Cellular Field 1650821 100 101 CHW CHA  
## teamIDretro  
## 1 ARI  
## 2 ATL  
## 3 BAL  
## 4 BOS  
## 5 CHN  
## 6 CHA

names(MLB2014)

## [1] "yearID" "lgID" "teamID" "franchID"   
## [5] "divID" "Rank" "G" "Ghome"   
## [9] "W" "L" "DivWin" "WCWin"   
## [13] "LgWin" "WSWin" "R" "AB"   
## [17] "H" "X2B" "X3B" "HR"   
## [21] "BB" "SO" "SB" "CS"   
## [25] "HBP" "SF" "RA" "ER"   
## [29] "ERA" "CG" "SHO" "SV"   
## [33] "IPouts" "HA" "HRA" "BBA"   
## [37] "SOA" "E" "DP" "FP"   
## [41] "name" "park" "attendance" "BPF"   
## [45] "PPF" "teamIDBR" "teamIDlahman45" "teamIDretro"

### Case 2.1

While we could compute statistics like team batting average from hits and at bats, we can also import the data from the text. In this case we pull it off of the class website.

### Read the CSV file from the web  
 case.2.1 = read.csv("http://facweb1.redlands.edu/fac/jim\_bentley/Data/FYS28/TSUB2/CSV/case\_2\_1.csv")  
 names(case.2.1) ### Get the dataframe column names

## [1] "Team" "G" "AB" "PA" "H" "X1B" "X2B" "X3B" "HR" "R"   
## [11] "RBI" "BB" "IBB" "SO" "HBP" "SF" "SH" "GDP" "SB" "CS"   
## [21] "AVG"

case.2.1 ### Look at the dataframe

## Team G AB PA H X1B X2B X3B HR R RBI BB IBB SO HBP  
## 1 Tigers 1961 5630 6202 1557 1051 325 26 155 757 731 443 51 1144 44  
## 2 Rockies 2120 5612 6164 1551 1017 307 41 186 755 721 397 39 1281 48  
## 3 Dodgers 2163 5560 6231 1476 1002 302 38 134 718 686 519 30 1246 61  
## 4 Royals 1788 5545 6058 1456 1046 286 29 95 651 604 380 22 985 53  
## 5 Pirates 2220 5536 6224 1436 975 275 30 156 682 659 520 46 1244 78  
## 6 Angels 1929 5652 6284 1464 974 304 31 155 773 729 492 42 1266 60  
## 7 Blue Jays 2060 5549 6167 1435 952 282 24 177 723 690 502 27 1151 41  
## 8 Rangers 1801 5460 6026 1400 1001 260 28 111 637 597 417 37 1162 61  
## 9 Orioles 1901 5596 6130 1434 943 264 16 211 705 681 401 29 1285 62  
## 10 Giants 2140 5523 6087 1407 976 257 42 132 665 636 427 37 1245 43  
## 11 Twins 1785 5567 6233 1412 941 316 27 128 715 675 544 29 1329 53  
## 12 Nationals 1913 5542 6216 1403 959 265 27 152 686 635 517 29 1304 56  
## 13 Indians 1860 5575 6222 1411 962 284 23 142 669 644 504 24 1189 42  
## 14 Cardinals 2069 5426 6086 1371 970 275 21 105 619 585 471 28 1133 86  
## 15 Marlins 1963 5538 6185 1399 987 254 36 122 645 614 501 49 1419 35  
## 16 White Sox 1817 5543 6077 1400 934 279 32 155 660 625 417 33 1362 60  
## 17 Brewers 2008 5462 6065 1366 891 297 28 150 650 617 423 32 1197 73  
## 18 Diamondbacks 1958 5552 6089 1379 955 259 47 118 615 573 398 31 1165 43  
## 19 Rays 1915 5516 6205 1361 957 263 24 117 612 586 527 31 1124 66  
## 20 Yankees 1786 5497 6082 1349 929 247 26 147 633 591 452 16 1133 56  
## 21 Athletics 2003 5545 6245 1354 922 253 33 146 729 686 586 34 1104 49  
## 22 Red Sox 1733 5551 6226 1355 930 282 20 123 634 601 535 36 1337 68  
## 23 Mariners 1928 5450 5977 1328 913 247 32 136 634 600 396 33 1232 60  
## 24 Phillies 1987 5603 6198 1356 953 251 27 125 619 584 443 42 1306 55  
## 25 Astros 1865 5447 6055 1317 895 240 19 163 629 596 495 27 1442 55  
## 26 Braves 1928 5468 6064 1316 931 240 22 123 573 545 472 31 1369 43  
## 27 Cubs 2009 5508 6102 1315 857 270 31 157 614 590 442 29 1477 54  
## 28 Mets 2135 5472 6145 1306 887 275 19 125 629 602 516 42 1264 54  
## 29 Reds 1926 5395 5978 1282 877 254 20 131 595 562 415 22 1252 52  
## 30 Padres 2148 5294 5905 1199 836 224 30 109 535 500 468 27 1294 41  
## SF SH GDP SB CS AVG  
## 1 61 24 137 106 41 0.277  
## 2 48 59 121 85 48 0.276  
## 3 43 47 120 138 50 0.265  
## 4 47 33 131 153 36 0.263  
## 5 35 54 127 104 47 0.259  
## 6 54 26 112 81 39 0.259  
## 7 40 35 128 78 21 0.259  
## 8 45 41 148 105 59 0.256  
## 9 36 35 112 44 20 0.256  
## 10 49 45 113 56 27 0.255  
## 11 44 25 97 99 36 0.254  
## 12 41 60 115 101 23 0.253  
## 13 49 51 126 104 27 0.253  
## 14 39 64 140 57 32 0.253  
## 15 39 71 143 58 21 0.253  
## 16 38 19 127 85 36 0.253  
## 17 37 70 137 102 43 0.250  
## 18 36 56 115 86 33 0.248  
## 19 53 43 135 63 27 0.247  
## 20 47 29 111 112 26 0.245  
## 21 43 19 118 83 20 0.244  
## 22 52 20 138 63 25 0.244  
## 23 34 35 113 96 42 0.244  
## 24 37 59 95 109 26 0.242  
## 25 36 22 122 122 37 0.242  
## 26 27 53 121 95 33 0.241  
## 27 41 57 94 65 40 0.239  
## 28 44 59 112 101 34 0.239  
## 29 37 76 88 122 52 0.238  
## 30 45 56 118 91 34 0.226

case.2.1[order(case.2.1$AVG),c("Team","AVG")] ### Rows are sorted by AVG, columns are only Team and AVG

## Team AVG  
## 30 Padres 0.226  
## 29 Reds 0.238  
## 27 Cubs 0.239  
## 28 Mets 0.239  
## 26 Braves 0.241  
## 24 Phillies 0.242  
## 25 Astros 0.242  
## 21 Athletics 0.244  
## 22 Red Sox 0.244  
## 23 Mariners 0.244  
## 20 Yankees 0.245  
## 19 Rays 0.247  
## 18 Diamondbacks 0.248  
## 17 Brewers 0.250  
## 12 Nationals 0.253  
## 13 Indians 0.253  
## 14 Cardinals 0.253  
## 15 Marlins 0.253  
## 16 White Sox 0.253  
## 11 Twins 0.254  
## 10 Giants 0.255  
## 8 Rangers 0.256  
## 9 Orioles 0.256  
## 5 Pirates 0.259  
## 6 Angels 0.259  
## 7 Blue Jays 0.259  
## 4 Royals 0.263  
## 3 Dodgers 0.265  
## 2 Rockies 0.276  
## 1 Tigers 0.277

### Or, use the MLB2014 data taken from the Lahman Teams data  
 case\_2.1 = MLB2014 ### Make a copy of MLB2014  
 case\_2.1$avg = case\_2.1$H/case\_2.1$AB ### Create a new variable/column avg using H and AB  
 case\_2.1[order(case\_2.1$avg),c("name","avg")] ### Rows are sorted by avg, columns are only name and avg

## name avg  
## 23 San Diego Padres 0.2264828  
## 7 Cincinnati Reds 0.2376274  
## 18 New York Mets 0.2386696  
## 5 Chicago Cubs 0.2387436  
## 2 Atlanta Braves 0.2406730  
## 11 Houston Astros 0.2417845  
## 21 Philadelphia Phillies 0.2420132  
## 25 Seattle Mariners 0.2436697  
## 4 Boston Red Sox 0.2441002  
## 20 Oakland Athletics 0.2441839  
## 19 New York Yankees 0.2454066  
## 27 Tampa Bay Rays 0.2467368  
## 1 Arizona Diamondbacks 0.2483790  
## 16 Milwaukee Brewers 0.2500915  
## 6 Chicago White Sox 0.2525708  
## 15 Miami Marlins 0.2526183  
## 26 St. Louis Cardinals 0.2526723  
## 8 Cleveland Indians 0.2530942  
## 30 Washington Nationals 0.2531577  
## 17 Minnesota Twins 0.2536375  
## 24 San Francisco Giants 0.2547529  
## 3 Baltimore Orioles 0.2562545  
## 28 Texas Rangers 0.2564103  
## 29 Toronto Blue Jays 0.2586052  
## 13 Los Angeles Angels of Anaheim 0.2590234  
## 22 Pittsburgh Pirates 0.2593931  
## 12 Kansas City Royals 0.2625789  
## 14 Los Angeles Dodgers 0.2654676  
## 9 Colorado Rockies 0.2763721  
## 10 Detroit Tigers 0.2765542

### Or, we can use dplyr to do the same thing  
 case\_2.1 = MLB2014 %>% ### Copy the MLB2014 data  
 mutate(avg = H/AB) %>% ### Create a new variable avg using H and AB  
 arrange(avg) ### Sort by avg  
  
 case\_2.1 %>% select(name, avg) ### Display only name and avg

## name avg  
## 1 San Diego Padres 0.2264828  
## 2 Cincinnati Reds 0.2376274  
## 3 New York Mets 0.2386696  
## 4 Chicago Cubs 0.2387436  
## 5 Atlanta Braves 0.2406730  
## 6 Houston Astros 0.2417845  
## 7 Philadelphia Phillies 0.2420132  
## 8 Seattle Mariners 0.2436697  
## 9 Boston Red Sox 0.2441002  
## 10 Oakland Athletics 0.2441839  
## 11 New York Yankees 0.2454066  
## 12 Tampa Bay Rays 0.2467368  
## 13 Arizona Diamondbacks 0.2483790  
## 14 Milwaukee Brewers 0.2500915  
## 15 Chicago White Sox 0.2525708  
## 16 Miami Marlins 0.2526183  
## 17 St. Louis Cardinals 0.2526723  
## 18 Cleveland Indians 0.2530942  
## 19 Washington Nationals 0.2531577  
## 20 Minnesota Twins 0.2536375  
## 21 San Francisco Giants 0.2547529  
## 22 Baltimore Orioles 0.2562545  
## 23 Texas Rangers 0.2564103  
## 24 Toronto Blue Jays 0.2586052  
## 25 Los Angeles Angels of Anaheim 0.2590234  
## 26 Pittsburgh Pirates 0.2593931  
## 27 Kansas City Royals 0.2625789  
## 28 Los Angeles Dodgers 0.2654676  
## 29 Colorado Rockies 0.2763721  
## 30 Detroit Tigers 0.2765542

R makes it very easy to create a stemplot. A few options also make it easy to modify the stemplot so that it is easily interpretable.

### The default stem has too few stems  
 stem(case.2.1$HR)

##   
## The decimal point is 1 digit(s) to the right of the |  
##   
## 8 | 5  
## 10 | 59178  
## 12 | 2335581246  
## 14 | 2670255567  
## 16 | 37  
## 18 | 6  
## 20 | 1

### We can split the stems in half to match the text  
 stem(case.2.1$HR, 2)

##   
## The decimal point is 1 digit(s) to the right of the |  
##   
## 9 | 5  
## 10 | 59  
## 11 | 178  
## 12 | 233558  
## 13 | 1246  
## 14 | 267  
## 15 | 0255567  
## 16 | 3  
## 17 | 7  
## 18 | 6  
## 19 |   
## 20 |   
## 21 | 1

### Further division of stems is overkill  
 stem(case.2.1$HR, 5)

##   
## The decimal point is at the |  
##   
## 94 | 0  
## 96 |   
## 98 |   
## 100 |   
## 102 |   
## 104 | 0  
## 106 |   
## 108 | 0  
## 110 | 0  
## 112 |   
## 114 |   
## 116 | 0  
## 118 | 0  
## 120 |   
## 122 | 000  
## 124 | 00  
## 126 |   
## 128 | 0  
## 130 | 0  
## 132 | 0  
## 134 | 0  
## 136 | 0  
## 138 |   
## 140 |   
## 142 | 0  
## 144 |   
## 146 | 00  
## 148 |   
## 150 | 0  
## 152 | 0  
## 154 | 000  
## 156 | 00  
## 158 |   
## 160 |   
## 162 | 0  
## 164 |   
## 166 |   
## 168 |   
## 170 |   
## 172 |   
## 174 |   
## 176 | 0  
## 178 |   
## 180 |   
## 182 |   
## 184 |   
## 186 | 0  
## 188 |   
## 190 |   
## 192 |   
## 194 |   
## 196 |   
## 198 |   
## 200 |   
## 202 |   
## 204 |   
## 206 |   
## 208 |   
## 210 | 0

If we need a back-to-back stemplot we need to use the ***aplpack*** package.

p\_load(aplpack)  
 ### Create vectors that contain HR for either NL or AL using base R  
 nlhr2014 = MLB2014$HR[MLB2014$lgID=="NL"] ### Use the HR variable and subset only NL  
 alhr2014 = MLB2014$HR[MLB2014$lgID=="AL"] ### Use the HR variable and subset only AL  
 stem.leaf.backback(nlhr2014, alhr2014)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 12, leaf unit: 1   
## nlhr2014 alhr2014  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 9 |5 1   
## 2 95| 10 |   
## 3 8| 11 |17 3   
## 7 5532| 12 |38 5   
## (3) 421| 13 |6 6   
## | 14 |267 (3)   
## 5 7620| 15 |555 6   
## | 16 |3 3   
## | 17 |7 2   
## 1 6| 18 |   
## | 19 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## HI: 211   
## n: 15 15   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Or, use dplyr to do the same thing  
 nlhr2014 = MLB2014 %>% filter(lgID == "NL") %>% select(HR)  
 alhr2014 = MLB2014 %>% filter(lgID == "AL") %>% select(HR)  
 stem.leaf.backback(as.vector(t(alhr2014)), as.vector(t(nlhr2014))) ### Force the dataframes to col vectors

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 12, leaf unit: 1   
## as.vector(t(alhr2014))  
## as.vector(t(nlhr2014))  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 5| 9 |   
## | 10 |59 2   
## 3 71| 11 |8 3   
## 5 83| 12 |2355 7   
## 6 6| 13 |124 (3)   
## (3) 762| 14 |   
## 6 555| 15 |0267 5   
## 3 3| 16 |   
## 2 7| 17 |   
## | 18 |6 1   
## | 19 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## HI: 211   
## n: 15 15   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team on base percentage (OBP) is easily graphed. Computing statistics on the statistic is also simple.

### Add the OBP to the dataframe using dplyr  
 case.2.1 = case.2.1 %>% mutate(OBP = round((H + BB + HBP)/(AB + BB + HBP + SF), 3))  
  
 ### Stem plot sing base R  
 stem(case.2.1$OBP)

##   
## The decimal point is 2 digit(s) to the left of the |  
##   
## 29 | 2  
## 29 | 6  
## 30 | 0022  
## 30 | 5789  
## 31 | 011144  
## 31 | 6777  
## 32 | 001234  
## 32 | 7  
## 33 | 013

### Using aplpack  
 stem.leaf(case.2.1$OBP)

## 1 | 2: represents 0.012  
## leaf unit: 0.001  
## n: 30  
## 1 29\* | 2  
## 2 29. | 6  
## 6 30\* | 0022  
## 10 30. | 5789  
## (6) 31\* | 011144  
## 14 31. | 6777  
## 10 32\* | 001234  
## 4 32. | 7  
## 3 33\* | 013

### While we can read the sorted data from the stem plot, sort the data to check  
 sort(case.2.1$OBP)

## [1] 0.292 0.296 0.300 0.300 0.302 0.302 0.305 0.307 0.308 0.309 0.310 0.311  
## [13] 0.311 0.311 0.314 0.314 0.316 0.317 0.317 0.317 0.320 0.320 0.321 0.322  
## [25] 0.323 0.324 0.327 0.330 0.331 0.333

### How many observations  
 length(case.2.1$OBP)

## [1] 30

### What are the OBPs for the 15th and 16th obs?  
 case.2.1$OBP[15:16] ### Oops, not sorted

## [1] 0.317 0.310

sort(case.2.1$OBP)[15:16] ### That's better

## [1] 0.314 0.314

### Find the median  
 median(case.2.1$OBP)

## [1] 0.314

### What are the OBPs for the 8th and 23rd sorted obs?  
 sort(case.2.1$OBP)[c(8,23)]

## [1] 0.307 0.321

### Find the quartiles  
 quantile(case.2.1$OBP)[c(2,4)]

## 25% 75%   
## 0.30725 0.32075

### Or...  
 summary(case.2.1$OBP)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.2920 0.3073 0.3140 0.3137 0.3207 0.3330

### Case 2.2

The plots and statistics for Derek Jeter that are presented in Case 2.2 are variations on those already presented above. For this case we download the data Albert’s GitHub repository.

### Get the data. Be careful about getting the "raw" CSV file from GitHub  
 case.2.2 <- read.csv("https://raw.githubusercontent.com/bayesball/Teaching-Statistics-Using-Baseball/master/case\_2\_2.csv")  
   
 ### See what's in it  
 head(case.2.2)

## Year Age G PA AB R H X2B X3B HR RBI BB SO BA OBP SLG OPS  
## 1 1995 21 15 51 48 5 12 4 1 0 7 3 11 0.250 0.294 0.375 0.669  
## 2 1996 22 157 654 582 104 183 25 6 10 78 48 102 0.314 0.370 0.430 0.800  
## 3 1997 23 159 748 654 116 190 31 7 10 70 74 125 0.291 0.370 0.405 0.775  
## 4 1998 24 149 694 626 127 203 25 8 19 84 57 119 0.324 0.384 0.481 0.864  
## 5 1999 25 158 739 627 134 219 37 9 24 102 91 116 0.349 0.438 0.552 0.989  
## 6 2000 26 148 679 593 119 201 31 4 15 73 68 99 0.339 0.416 0.481 0.896

### Albert's Table 2.2 contains fewer variables  
 table.2.2 = case.2.2 %>%  
 select(Year, AB:HR, BB:OPS) %>% ### Use ":" to get all variables between those indicated  
 rename(AVG = BA) ### Rename batting average from BA to AVG  
 head(table.2.2)

## Year AB R H X2B X3B HR BB SO AVG OBP SLG OPS  
## 1 1995 48 5 12 4 1 0 3 11 0.250 0.294 0.375 0.669  
## 2 1996 582 104 183 25 6 10 48 102 0.314 0.370 0.430 0.800  
## 3 1997 654 116 190 31 7 10 74 125 0.291 0.370 0.405 0.775  
## 4 1998 626 127 203 25 8 19 57 119 0.324 0.384 0.481 0.864  
## 5 1999 627 134 219 37 9 24 91 116 0.349 0.438 0.552 0.989  
## 6 2000 593 119 201 31 4 15 68 99 0.339 0.416 0.481 0.896

Albert removed Jeter’s rookie year (1995) and his injury year(2013).

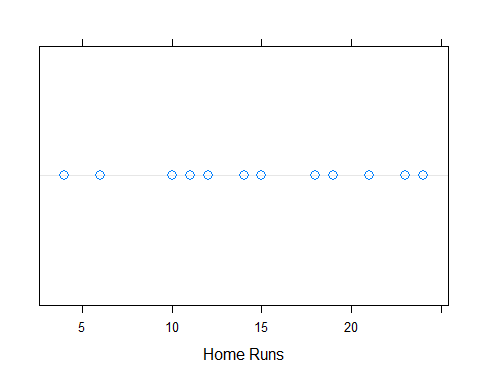
case.2.2$Year ### Look at Jeter's years

## [1] 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009  
## [16] 2010 2011 2012 2013 2014

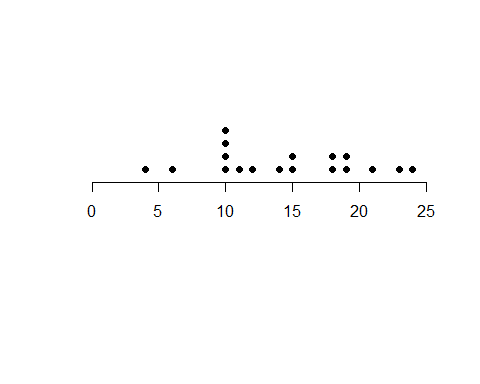
albert.2.2 = case.2.2[-c(1,19), ] ### Remove rows 1 and 19 (negative index), but keep all columns (blank after comma)  
 albert.2.2$Year ### 1995 and 2013 are not listed

## [1] 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010  
## [16] 2011 2012 2014

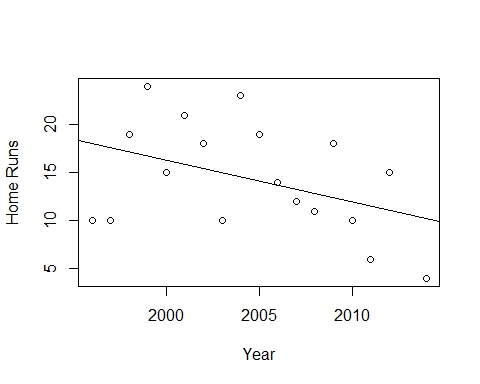
case.2.2 = albert.2.2 ### Update case.2.2   
   
 ### Create Figure 2.5  
 dotplot(~HR, data=case.2.2, cex=1.25, pch=1, xlab="Home Runs")



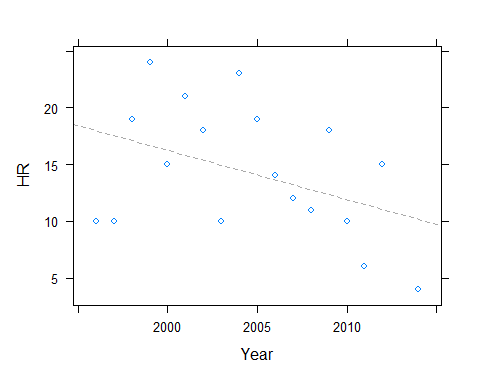
dotplot.mtb(case.2.2$HR)



### Create Figure 2.6  
 plot(case.2.2$Year, case.2.2$HR, xlab="Year", ylab="Home Runs")  
   
 ### Create Figure 2.7  
 plot(case.2.2$Year, case.2.2$HR, xlab="Year", ylab="Home Runs")  
 abline(a=883.68, b=-0.4337) ### Doesn't show in RStudio. Knit the doc.

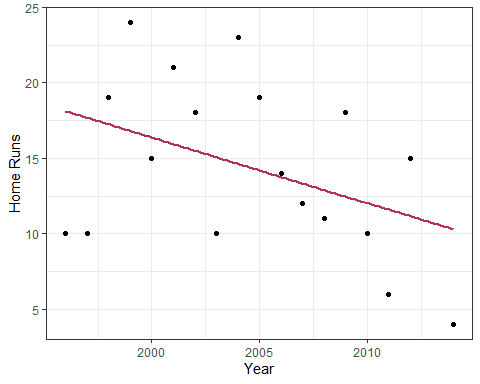


# Lattice approach  
 p\_load(lattice)  
 xyplot(HR~Year, data=case.2.2, pch=1,   
 panel=function(x, y, ...){  
 panel.xyplot(x, y, ...)  
 panel.abline(a=883.68, b=-0.4337, col="darkgray", lty=2)  
 }  
 )



# ggplot2 approach  
 p\_load(ggplot2)  
 ggplot(case.2.2, aes(Year, HR)) + geom\_point() + theme\_bw() + xlab("Year") + ylab("Home Runs") + stat\_smooth(method="lm", se=FALSE, color="maroon")

## `geom\_smooth()` using formula 'y ~ x'



### Create Figure 2.8  
 stem(case.2.2$OPS, 2)

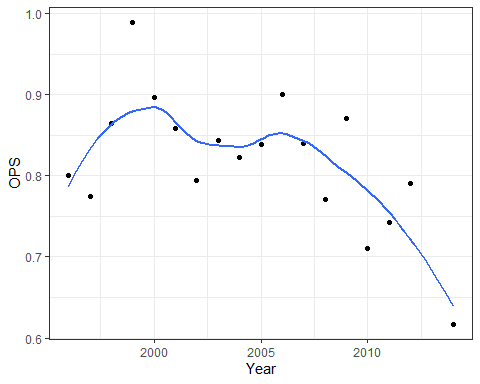
##   
## The decimal point is 1 digit(s) to the left of the |  
##   
## 6 | 2  
## 6 |   
## 7 | 14  
## 7 | 7899  
## 8 | 02444  
## 8 | 667  
## 9 | 00  
## 9 | 9

### Oops, Albert's software truncated to two decimals rather than rounded  
 stem(trunc(100\*case.2.2$OPS)/100, 2)

##   
## The decimal point is 1 digit(s) to the left of the |  
##   
## 6 | 1  
## 6 |   
## 7 | 14  
## 7 | 7799  
## 8 | 02344  
## 8 | 5679  
## 9 | 0  
## 9 | 8

### Create Figure 2.9  
 # ggplot2 approach  
 ggplot(case.2.2, aes(Year, OPS)) + geom\_point() + theme\_bw() + xlab("Year") +   
 geom\_smooth(method = "loess", span = 2/3, se = FALSE)

## `geom\_smooth()` using formula 'y ~ x'



### Case 2.3

This case provides an opportunity to create a new statistic and to use some new graphical features. We will use Randy Johnson’s data from the **tsub** package.

### Get the data  
 case.2.3 = case\_2\_3  
 head(case.2.3)

## yearID W L G IP H R SO BB ERA  
## 1 1988 3 0 4 26.00 23 8 25 7 2.42  
## 2 1989 7 13 29 160.67 147 100 130 96 4.82  
## 3 1990 14 11 33 219.67 174 103 194 120 3.65  
## 4 1991 13 10 33 201.33 151 96 228 152 3.98  
## 5 1992 12 14 31 210.33 154 104 241 144 3.77  
## 6 1993 19 8 35 255.33 185 97 308 99 3.24

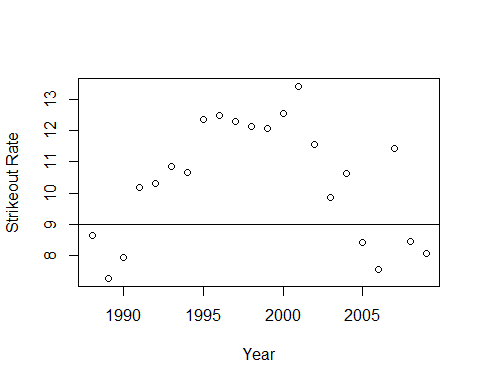
### Compute the magic strikeout rate, SOR  
 case.2.3$sor = (case.2.3$SO/case.2.3$IP) \* 9  
 case.2.3[, c("yearID","sor")]

## yearID sor  
## 1 1988 8.653846  
## 2 1989 7.282007  
## 3 1990 7.948286  
## 4 1991 10.192222  
## 5 1992 10.312366  
## 6 1993 10.856539  
## 7 1994 10.674419  
## 8 1995 12.345449  
## 9 1996 12.473504  
## 10 1997 12.295775  
## 11 1998 12.118856  
## 12 1999 12.058748  
## 13 2000 12.558813  
## 14 2001 13.409701  
## 15 2002 11.561538  
## 16 2003 9.868421  
## 17 2004 10.624008  
## 18 2005 8.414942  
## 19 2006 7.551220  
## 20 2007 11.434621  
## 21 2008 8.461957  
## 22 2009 8.062500

### Create Figure 2.10  
 stem(case.2.3$sor,5)

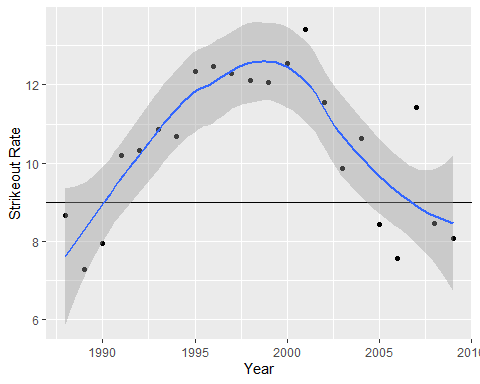
##   
## The decimal point is at the |  
##   
## 7 | 3  
## 7 | 69  
## 8 | 14  
## 8 | 57  
## 9 |   
## 9 | 9  
## 10 | 23  
## 10 | 679  
## 11 | 4  
## 11 | 6  
## 12 | 1133  
## 12 | 56  
## 13 | 4

### Again, the figures differ by rounding versus truncating  
   
 ### Generate Figure 2.11  
 plot(case.2.3$yearID, case.2.3$sor, xlab="Year", ylab="Strikeout Rate")  
 abline(h=9)



# ggplot2 approach  
 p\_load(ggplot2)  
 ggplot(case.2.3, aes(yearID, sor)) + geom\_point() + xlab("Year") + ylab("Strikeout Rate") +   
 geom\_hline(aes(yintercept=9)) + geom\_smooth(method="loess", span = 2/3)

## `geom\_smooth()` using formula 'y ~ x'



Of course, the data for Figures 2.12 and 2.13 are not in case\_2\_3.csv. So, we need to get a little creative.

### We'll use dplyr to pull Johnson's ID from Lahman's People table  
 p\_load(Lahman, dplyr)  
 johnson.id = People %>% ### Grab the People dataframe  
 filter(nameFirst=="Randy" & nameLast=="Johnson") # %>% ### Get Randy Johnson  
 ### There are three Randy Johnson's  
 head(johnson.id)

## playerID birthYear birthMonth birthDay birthCountry birthState birthCity  
## 1 johnsra03 1958 8 15 USA FL Miami  
## 2 johnsra04 1956 6 10 USA CA Escondido  
## 3 johnsra05 1963 9 10 USA CA Walnut Creek  
## deathYear deathMonth deathDay deathCountry deathState deathCity nameFirst  
## 1 NA NA NA <NA> <NA> <NA> Randy  
## 2 NA NA NA <NA> <NA> <NA> Randy  
## 3 NA NA NA <NA> <NA> <NA> Randy  
## nameLast nameGiven weight height bats throws debut finalGame  
## 1 Johnson Randall Stuart 195 74 L L 1980-07-05 1982-10-03  
## 2 Johnson Randall Glenn 190 73 R R 1982-04-27 1984-09-30  
## 3 Johnson Randall David 225 82 R L 1988-09-15 2009-10-04  
## retroID bbrefID deathDate birthDate  
## 1 johnr101 johnsra03 <NA> 1958-08-15  
## 2 johnr001 johnsra04 <NA> 1956-06-10  
## 3 johnr005 johnsra05 <NA> 1963-09-10

### We want the one whose first game was in 1988  
 johnson.id = johnson.id %>%   
 filter(year(debut) == 1988) %>% ### Make sure debut year was 1988  
 select(retroID) ### We need only his ID  
 head(johnson.id)

## retroID  
## 1 johnr005

johnson.id = as.character(johnson.id) ### Make it a value not a dataframe  
   
 ### Starting pitchers and runs can be found in RetroSheet  
 ### RetroSheet can sometimes be hard to install. If p\_load fails,  
 ### uncomment the two lines that follow and run once.  
 ## p\_load(devtools)  
 ## devtools::install\_github("rmscriven/retrosheet")  
 p\_load(retrosheet)  
 game1995 = getRetrosheet("game", 1995)  
   
 ### Use SQL to subset the SEA games and order them by date  
 p\_load(sqldf)  
 sqlsea1995 = sqldf("SELECT \* FROM game1995 WHERE (HmTm='SEA' OR VisTm='SEA') ORDER BY Date")  
 ### Or, use dplyr to get SEA games and order by Date  
 sea1995 = game1995 %>%  
 filter(HmTm=="SEA" | VisTm=="SEA") %>% ### Get home and away games for SEA  
 arrange(Date)  
 ### Check to see if the two dataframes are the same  
 all\_equal(sea1995, sqlsea1995) ### Good news, they are the same

## [1] TRUE

### Now figure out which ones Johnson started in and which were home  
 sea1995$RJstart = sea1995$VisStPchID==johnson.id | sea1995$HmStPchID==johnson.id ### RJ started TRUE/FALSE  
 sea1995$AtHome = sea1995$HmTm=="SEA" ### Home team SEA TRUE/FALSE  
 sea1995[sea1995$AtHome, "runs"] = sea1995[sea1995$AtHome, "HmRuns"] ### Rows: AtHome==TRUE; runs=HmRuns   
 sea1995[!sea1995$AtHome,"runs"] = sea1995[!sea1995$AtHome, "VisRuns"] ### Rows: AtHome==FALSE; runs=VisRuns   
 ### Subset the data for Johnson starts  
 runs.rj.start = sea1995$runs[sea1995$RJstart]  
   
 ### Text "table" of runs when RJ started.   
 runs.rj.start

## [1] 3 15 3 6 4 5 8 11 2 3 1 9 3 5 4 5 3 4 8 2 2 6 6 7 4  
## [26] 7 8 7 6 9

### Create Figure 2.12  
 stem(runs.rj.start, 2)

##   
## The decimal point is at the |  
##   
## 1 | 0  
## 2 | 000  
## 3 | 00000  
## 4 | 0000  
## 5 | 000  
## 6 | 0000  
## 7 | 000  
## 8 | 000  
## 9 | 00  
## 10 |   
## 11 | 0  
## 12 |   
## 13 |   
## 14 |   
## 15 | 0

### While the shape is right, the leaves and stems aren't pretty. Try aplpack.  
 p\_load(aplpack)  
 stem.leaf(runs.rj.start)

## 1 | 2: represents 1.2  
## leaf unit: 0.1  
## n: 30  
## 1 1 | 0  
## 4 2 | 000  
## 9 3 | 00000  
## 13 4 | 0000  
## (3) 5 | 000  
## 14 6 | 0000  
## 10 7 | 000  
## 7 8 | 000  
## 4 9 | 00  
## 10 |   
## 2 11 | 0  
## HI: 15

### Create Figure 2.13  
 runs.rj.nostart = sea1995$runs[!sea1995$RJstart]  
 stem.leaf.backback(runs.rj.start,runs.rj.nostart)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 1.2, leaf unit: 0.1   
## runs.rj.start runs.rj.nostart   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 0 |00 2   
## 1 0| 1 |000000000000 14   
## 4 000| 2 |00000000000000 28   
## 9 00000| 3 |00000000000 39   
## 13 0000| 4 |000000000000 51   
## (3) 000| 5 |000000000000 (12)   
## 14 0000| 6 |0000000000000000 52   
## 10 000| 7 |00000 36   
## 7 000| 8 |00000 31   
## 4 00| 9 |0000000 26   
## | 10 |000000000 19   
## 2 0| 11 |000000 10   
## | 12 |   
## | 13 |   
## | 14 |0 4   
## 1 0| 15 |000 3   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 30 115   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Means and medians  
 mean(runs.rj.start)

## [1] 5.533333

median(runs.rj.start)

## [1] 5

mean(runs.rj.nostart)

## [1] 5.478261

median(runs.rj.nostart)

## [1] 5

### Case 2.4

TSUB2 now looks at attendance. As seen above, this information is included in Lahman’s data. However, we will use Albert’s data for convenience.

### Make a local copy of the tsub dataframe  
 case.2.4 = case\_2\_4  
 head(case.2.4)

## HomeTeam Mean  
## 1 ANA 38221  
## 2 ARI 25602  
## 3 ATL 29065  
## 4 BAL 30426  
## 5 BOS 36495  
## 6 CHA 20381

### The variable names are messed up. We fix them.  
 names(case.2.4) ### Ask for the column names

## [1] "HomeTeam" "Mean"

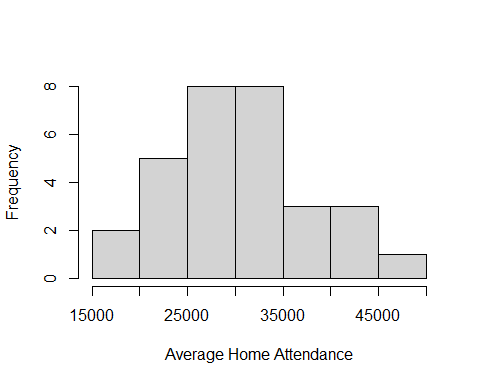
names(case.2.4) <- c("Team","Attendance") ### Set the column names with the new names  
 names(case.2.4) ### Ask for the column names

## [1] "Team" "Attendance"

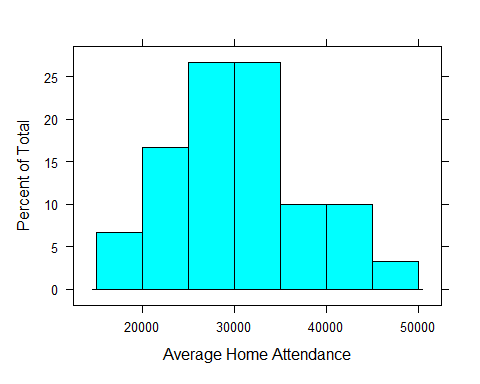
head(case.2.4) ### Look at the dataframe

## Team Attendance  
## 1 ANA 38221  
## 2 ARI 25602  
## 3 ATL 29065  
## 4 BAL 30426  
## 5 BOS 36495  
## 6 CHA 20381

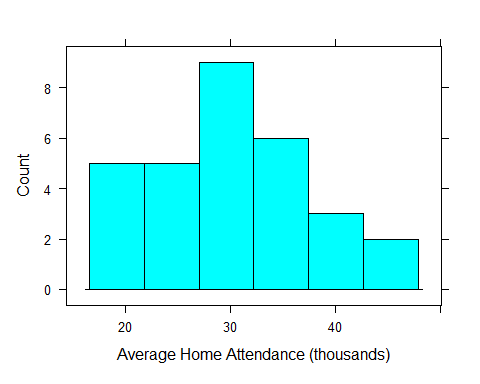
### Plot Figure 2.14 using base R  
 hist(case.2.4$Attendance, xlab="Average Home Attendance", main="")



### and the lattice package  
 histogram(~Attendance, data=case.2.4, xlab="Average Home Attendance", breaks=15000+5000\*(0:7))

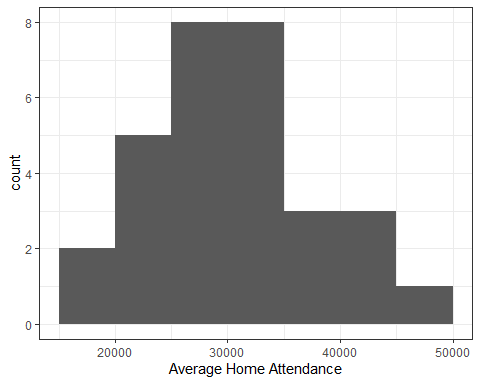


histogram(~Attendance/1000, data=case.2.4, xlab="Average Home Attendance (thousands)", type="count")



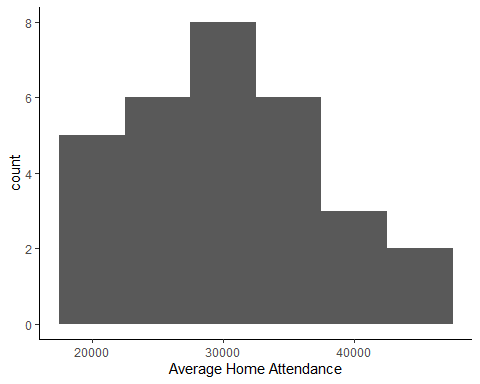
### and using ggplot  
 p\_load(ggplot2)  
 ggplot(case.2.4,aes(Attendance)) + geom\_histogram() + xlab("Average Home Attendance") + stat\_bin(breaks=15000+5000\*(0:7)) + theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggplot(case.2.4,aes(Attendance)) + geom\_histogram() + xlab("Average Home Attendance") + stat\_bin(binwidth=5000) + theme\_classic()

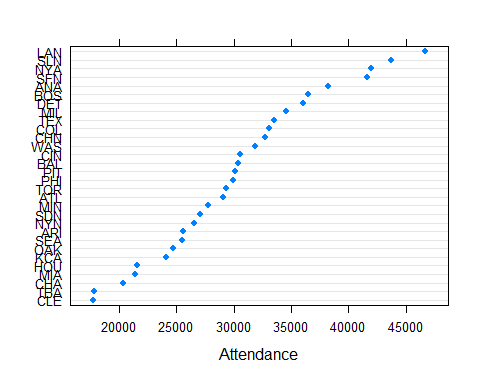
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



### Create Figure 2.15  
 p\_load(aplpack)  
 stem.leaf(case.2.4$Attendance)

## 1 | 2: represents 12000  
## leaf unit: 1000  
## n: 30  
## 2 1. | 77  
## 7 2\* | 01144  
## (8) 2. | 55677999  
## (8) 3\* | 00012334  
## 7 3. | 668  
## 4 4\* | 113  
## 1 4. | 6

### Create Figure 2.16  
 ### We can use the lattice package to create the Bill Cleveland style dotplot of average attendance  
 p\_load(lattice)  
 dotplot(reorder(Team, Attendance)~Attendance, data=case.2.4) ### We order the teams by their mean attendance



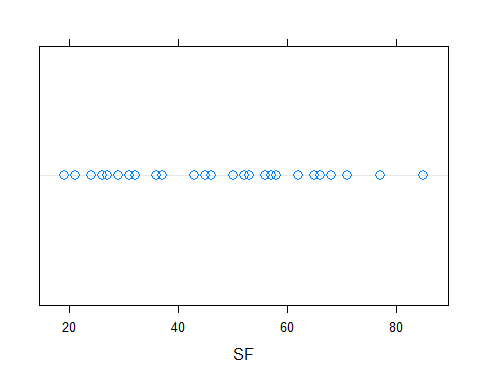
### Case 2.5

Finally, we look at the use of the sacrifice bunt during the 2013 season. The data list ***SF*** which appears to be sac FLIES not bunts. The values are the same as those of Table 2.6. The **tsub** package data do not include ***League***.

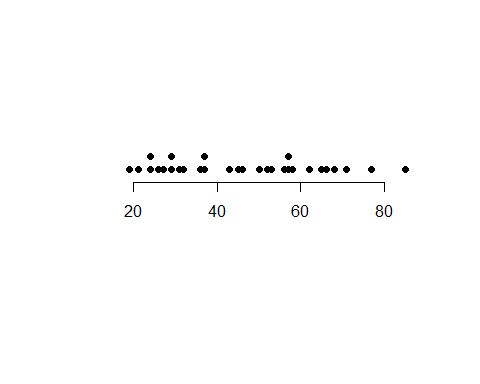
### Get the case\_2\_5 data  
 case.2.5 = read.csv("http://facweb1.redlands.edu/fac/jim\_bentley/Data/FYS28/TSUB2/CSV/case\_2\_5.csv")  
 head(case.2.5)

## BAT\_TEAM SF League  
## 1 ANA 37 AL  
## 2 ARI 50 NL  
## 3 ATL 58 NL  
## 4 BAL 27 AL  
## 5 BOS 24 AL  
## 6 CHA 19 AL

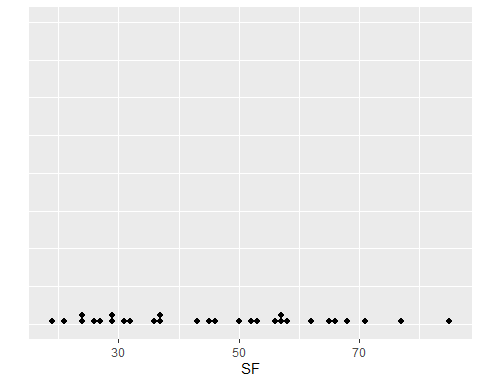
names(case.2.5)[1] <- "Team" ### Replace BAT\_TEAM with Team. Is SH sac bunts?  
   
 ### Figure 2.17   
 p\_load(lattice)  
 dotplot(~SF, data=case.2.5, cex=1.25, pch=1)



dotplot.mtb(case.2.5$SF)



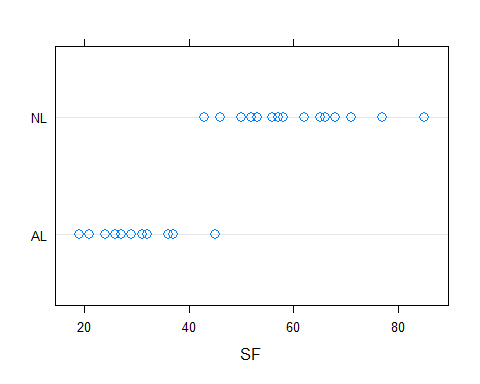
plt = case.2.5 %>%  
 ggplot(aes(x = SF)) +  
 geom\_dotplot(binwidth = 1)  
 plt + theme(axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank()   
 ) + labs(y="")



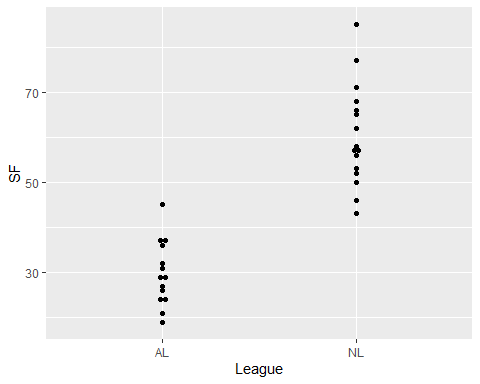
### Compute simple statistics  
 summary(case.2.5$SF)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 19.00 29.50 45.50 46.10 57.75 85.00

### Since we are using the lattice package, Figure 2.18 is an easy fix  
 dotplot(League ~ SF, data=case.2.5, cex=1.25, pch=1)



plt = case.2.5 %>%  
 ggplot(aes(x = League, y = SF)) +  
 geom\_dotplot(binaxis = "y", stackdir = "center", binwidth = 1)  
 plt



### Compute associated statistics  
 by(case.2.5$SF,case.2.5$League,summary)

## case.2.5$League: AL  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 19.00 24.50 29.00 29.79 35.00 45.00   
## ------------------------------------------------------------   
## case.2.5$League: NL  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 43.00 52.75 57.50 60.38 66.50 85.00

## Chapter Three

Chapter 3 moves from describing one characteristic to comparing a couple. R’s developers have made this transition easy.

### Case 3.1

The plots and statistics for comparing Albert Pujols and Manny Ramirez that are presented in Case 3.1 are, for the most part, variations on those already presented above. For this case we download the data from my web site.

### Get the data  
 case.3.1 = read.csv("http://facweb1.redlands.edu/fac/jim\_bentley/Data/FYS28/TSUB2/CSV/case\_3\_1.csv")  
  
 ### Table 3.1 is just the data.frame  
 case.3.1 %>% select(Player, Age, SLG, OBP, OPS)

## Player Age SLG OBP OPS  
## 1 Rameriz 22 0.5206897 0.3571429 0.878  
## 2 Rameriz 23 0.5578512 0.4024605 0.960  
## 3 Rameriz 24 0.5818182 0.3987635 0.981  
## 4 Rameriz 25 0.5383244 0.4147465 0.953  
## 5 Rameriz 26 0.5989492 0.3770739 0.976  
## 6 Rameriz 27 0.6628352 0.4421875 1.105  
## 7 Rameriz 28 0.6970387 0.4567669 1.154  
## 8 Rameriz 29 0.6086957 0.4048387 1.014  
## 9 Rameriz 30 0.6467890 0.4498069 1.097  
## 10 Rameriz 31 0.5869947 0.4270987 1.014  
## 11 Rameriz 32 0.6126761 0.3966817 1.009  
## 12 Rameriz 33 0.5938628 0.3876923 0.982  
## 13 Rameriz 34 0.6191537 0.4390681 1.058  
## 14 Rameriz 35 0.4927536 0.3884007 0.881  
## 15 Rameriz 36 0.6014493 0.4296636 1.031  
## 16 Rameriz 37 0.5312500 0.4176334 0.949  
## 17 Rameriz 38 0.4603774 0.4093750 0.870  
## 18 Pujols 21 0.6101695 0.4029630 1.013  
## 19 Pujols 22 0.5610169 0.3940741 0.955  
## 20 Pujols 23 0.6666667 0.4394161 1.106  
## 21 Pujols 24 0.6570946 0.4147399 1.072  
## 22 Pujols 25 0.6091371 0.4300000 1.039  
## 23 Pujols 26 0.6710280 0.4305994 1.102  
## 24 Pujols 27 0.5681416 0.4285714 0.997  
## 25 Pujols 28 0.6526718 0.4617785 1.114  
## 26 Pujols 29 0.6584507 0.4428571 1.101  
## 27 Pujols 30 0.5962521 0.4142857 1.011  
## 28 Pujols 31 0.5405872 0.3655914 0.906  
## 29 Pujols 32 0.5156507 0.3432836 0.859  
## 30 Pujols 33 0.4373402 0.3295711 0.767  
## 31 Pujols 34 0.4660348 0.3237410 0.790  
## 32 Pujols 35 0.4800664 0.3071104 0.787

names(case.3.1)

## [1] "Player" "yearID" "AB" "BB" "HBP" "H" "X2B" "X3B"   
## [9] "HR" "SF" "Age" "SLG" "OBP" "OPS"

### Albert next creates Figure 3.1, a back-to-back stemplot  
 p\_load(aplpack)  
 (pujols = case.3.1 %>% filter(Player=="Pujols"))

## Player yearID AB BB HBP H X2B X3B HR SF Age SLG OBP OPS  
## 1 Pujols 2001 590 69 9 194 47 4 37 7 21 0.6101695 0.4029630 1.013  
## 2 Pujols 2002 590 72 9 185 40 2 34 4 22 0.5610169 0.3940741 0.955  
## 3 Pujols 2003 591 79 10 212 51 1 43 5 23 0.6666667 0.4394161 1.106  
## 4 Pujols 2004 592 84 7 196 51 2 46 9 24 0.6570946 0.4147399 1.072  
## 5 Pujols 2005 591 97 9 195 38 2 41 3 25 0.6091371 0.4300000 1.039  
## 6 Pujols 2006 535 92 4 177 33 1 49 3 26 0.6710280 0.4305994 1.102  
## 7 Pujols 2007 565 99 7 185 38 1 32 8 27 0.5681416 0.4285714 0.997  
## 8 Pujols 2008 524 104 5 187 44 0 37 8 28 0.6526718 0.4617785 1.114  
## 9 Pujols 2009 568 115 9 186 45 1 47 8 29 0.6584507 0.4428571 1.101  
## 10 Pujols 2010 587 103 4 183 39 1 42 6 30 0.5962521 0.4142857 1.011  
## 11 Pujols 2011 579 61 4 173 29 0 37 7 31 0.5405872 0.3655914 0.906  
## 12 Pujols 2012 607 52 5 173 50 0 30 6 32 0.5156507 0.3432836 0.859  
## 13 Pujols 2013 391 40 5 101 19 0 17 7 33 0.4373402 0.3295711 0.767  
## 14 Pujols 2014 633 48 5 172 37 1 28 9 34 0.4660348 0.3237410 0.790  
## 15 Pujols 2015 602 50 6 147 22 0 40 3 35 0.4800664 0.3071104 0.787

(ramirez = case.3.1 %>% filter(Player=="Ramirez"))

## [1] Player yearID AB BB HBP H X2B X3B HR SF   
## [11] Age SLG OBP OPS   
## <0 rows> (or 0-length row.names)

Wait. There aren’t any Ramirez observations? What’s going on?

table(case.3.1$Player)

##   
## Pujols Rameriz   
## 15 17

It looks like Albert typoed. Let’s fix it and move on.

### Select only those rows with Player == "Rameriz" and replace the Player column value with "Ramirez"  
 case.3.1[case.3.1$Player == "Rameriz", "Player" ] <- "Ramirez"  
 (ramirez = case.3.1 %>% filter(Player=="Ramirez"))

## Player yearID AB BB HBP H X2B X3B HR SF Age SLG OBP OPS  
## 1 Ramirez 1994 290 42 0 78 22 0 17 4 22 0.5206897 0.3571429 0.878  
## 2 Ramirez 1995 484 75 5 149 26 1 31 5 23 0.5578512 0.4024605 0.960  
## 3 Ramirez 1996 550 85 3 170 45 3 33 9 24 0.5818182 0.3987635 0.981  
## 4 Ramirez 1997 561 79 7 184 40 0 26 4 25 0.5383244 0.4147465 0.953  
## 5 Ramirez 1998 571 76 6 168 35 2 45 10 26 0.5989492 0.3770739 0.976  
## 6 Ramirez 1999 522 96 13 174 34 3 44 9 27 0.6628352 0.4421875 1.105  
## 7 Ramirez 2000 439 86 3 154 34 2 38 4 28 0.6970387 0.4567669 1.154  
## 8 Ramirez 2001 529 81 8 162 33 2 41 2 29 0.6086957 0.4048387 1.014  
## 9 Ramirez 2002 436 73 8 152 31 0 33 1 30 0.6467890 0.4498069 1.097  
## 10 Ramirez 2003 569 97 8 185 36 1 37 5 31 0.5869947 0.4270987 1.014  
## 11 Ramirez 2004 568 82 6 175 44 0 43 7 32 0.6126761 0.3966817 1.009  
## 12 Ramirez 2005 554 80 10 162 30 1 45 6 33 0.5938628 0.3876923 0.982  
## 13 Ramirez 2006 449 100 1 144 27 1 35 8 34 0.6191537 0.4390681 1.058  
## 14 Ramirez 2007 483 71 7 143 33 1 20 8 35 0.4927536 0.3884007 0.881  
## 15 Ramirez 2008 552 87 11 183 36 1 37 4 36 0.6014493 0.4296636 1.031  
## 16 Ramirez 2009 352 71 7 102 24 2 19 1 37 0.5312500 0.4176334 0.949  
## 17 Ramirez 2010 265 46 6 79 16 0 9 3 38 0.4603774 0.4093750 0.870

stem.leaf.backback(pujols$OPS, ramirez$OPS)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 0.12, leaf unit: 0.01   
## pujols$OPS ramirez$OPS  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 7\* |   
## 3 986| 7. |   
## | 8\* |   
## 4 5| 8. |778 3   
## 5 0| 9\* |4 4   
## 7 95| 9. |56788 (5)   
## (3) 311| 10\* |0113 8   
## 5 7| 10. |59 4   
## 4 1000| 11\* |0 2   
## | 11. |5 1   
## | 12\* |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 15 17   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

That’s better, but where did Pujols’ 78 come from? How many observations should he have? The online data contains an extra season for Pujols — when he was 35.

### Five number summaries can be computed in a number of ways  
 ### Median  
 (n = nrow(pujols))

## [1] 15

(n+1)/2

## [1] 8

m = sort(pujols$OPS)[(n+1)/2]  
 m

## [1] 1.011

median(pujols$OPS)

## [1] 1.011

### Quartiles  
 sort(pujols$OPS[pujols$OPS < m])

## [1] 0.767 0.787 0.790 0.859 0.906 0.955 0.997

ql = median(pujols$OPS[pujols$OPS < m])  
 ql

## [1] 0.859

qu = median(pujols$OPS[pujols$OPS > m])  
 qu

## [1] 1.101

quantile(pujols$OPS)[c(2,4)]

## 25% 75%   
## 0.8825 1.0865

sort(pujols$OPS)

## [1] 0.767 0.787 0.790 0.859 0.906 0.955 0.997 1.011 1.013 1.039 1.072 1.101  
## [13] 1.102 1.106 1.114

### Min and max are easy  
 min(pujols$OPS)

## [1] 0.767

max(pujols$OPS)

## [1] 1.114

range(pujols$OPS)

## [1] 0.767 1.114

### Together, the statistics from above are the five-number-summary  
 quantile(pujols$OPS)

## 0% 25% 50% 75% 100%   
## 0.7670 0.8825 1.0110 1.0865 1.1140

summary(pujols$OPS) # Ignore the mean for now.

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.7670 0.8825 1.0110 0.9746 1.0865 1.1140

quantile(ramirez$OPS)

## 0% 25% 50% 75% 100%   
## 0.870 0.953 0.982 1.031 1.154

summary(ramirez$OPS)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.8700 0.9530 0.9820 0.9948 1.0310 1.1540

### Albert computes a "step" that is then used to compute the u/l fences  
 ### Note that the difference between steps found using Albert's method and  
 ### the internal function is minimal. We'll use Albert's value.  
 (qu = quantile(ramirez$OPS)[4])

## 75%   
## 1.031

(ql = quantile(ramirez$OPS)[2])

## 25%   
## 0.953

step = 1.5\*(qu-ql)  
 step

## 75%   
## 0.117

1.5 \* c(-1,1) %\*% quantile(ramirez$OPS)[c(2,4)] ### A little different due to different quantiles

## [,1]  
## [1,] 0.117

fu = qu + step  
 fu

## 75%   
## 1.148

fl = ql - step  
 fl

## 25%   
## 0.836

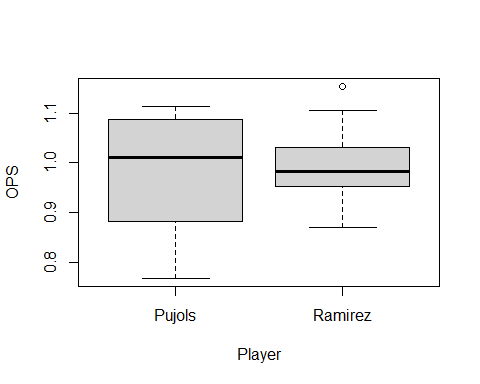
### So values greater than fu and less than fl are possible outliers  
 ramirez$OPS[ramirez$OPS > fu | ramirez$OPS < fl]

## [1] 1.154

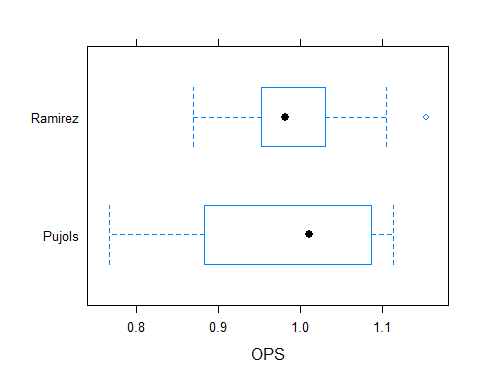
sort(ramirez$OPS)

## [1] 0.870 0.878 0.881 0.949 0.953 0.960 0.976 0.981 0.982 1.009 1.014 1.014  
## [13] 1.031 1.058 1.097 1.105 1.154

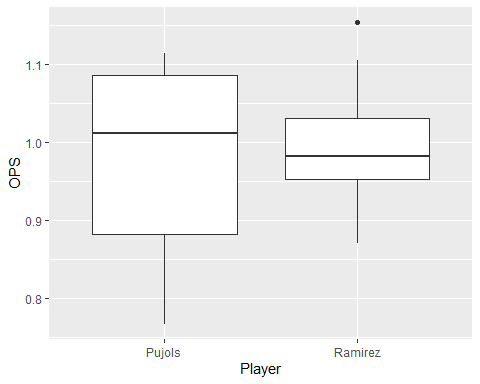
### Now we generate boxplots  
 boxplot(OPS~Player, data=case.3.1, ylab="OPS")



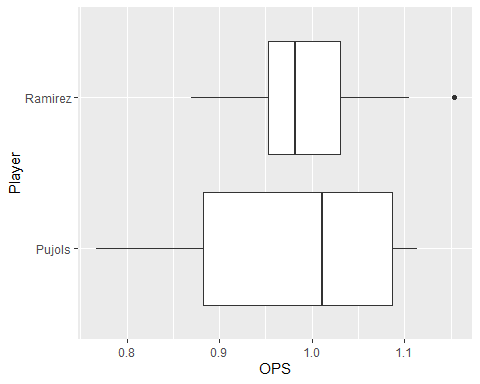
p\_load(lattice)  
 bwplot(Player~OPS, data=case.3.1)



p\_load(ggplot2)  
 ggplot(case.3.1,aes(Player, OPS)) + geom\_boxplot()



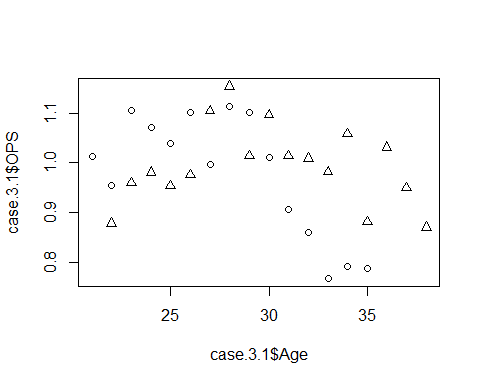
ggplot(case.3.1,aes(Player, OPS)) + geom\_boxplot() + coord\_flip()



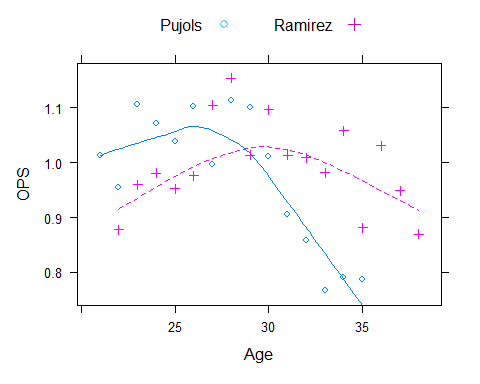
### Compare means  
 median(pujols$OPS)-median(ramirez$OPS)

## [1] 0.029

### Create Figure 3.4  
 plot(case.3.1$Age, case.3.1$OPS, pch=as.numeric(factor(case.3.1$Player)))

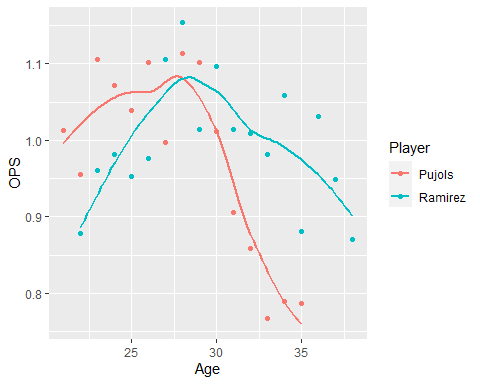


xyplot(OPS~Age, data=case.3.1, group=Player,   
 type=c("p","smooth"), span=0.75, pch=c(1,3), lty=c(1,2),   
 key=list(text=list(c("Pujols", "Ramirez")),   
 points=list(pch=c(1,3), col=c(13,14)),   
 columns=2)  
 )



ggplot(case.3.1, aes(x=Age, y=OPS, color=Player)) +   
 geom\_point() + geom\_smooth(span=0.75, se=FALSE)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



### Case 3.2

The plots and statistics for comparing the pitchers Robin Roberts and Whitey Ford that are presented in Case 3.2 are, for the most part, variations on those already presented above. For this case we again download the data from my web site.

### Get the data  
 case.3.2 = read.csv("http://facweb1.redlands.edu/fac/jim\_bentley/Data/FYS28/TSUB2/CSV/case\_3\_2.csv")  
 ### Albert next creates Figure 3.5, a back-to-back stemplot  
 p\_load(aplpack)  
 roberts = subset(case.3.2, Player=="Roberts")  
 roberts

## Year Player Age Tm W L G IP H ER HR BB SO ERA X.lgERA  
## 1 1948 Roberts 21 PHI 7 9 20 146.7 148 52 10 61 84 3.19 3.96  
## 2 1949 Roberts 22 PHI 15 15 43 226.7 229 93 15 75 95 3.69 3.96  
## 3 1950 Roberts 23 PHI 20 11 40 304.3 282 102 29 77 146 3.02 4.06  
## 4 1951 Roberts 24 PHI 21 15 44 315.0 284 106 20 64 127 3.03 3.84  
## 5 1952 Roberts 25 PHI 28 7 39 330.0 292 95 22 45 148 2.59 3.66  
## 6 1953 Roberts 26 PHI 23 16 44 346.7 324 106 30 61 198 2.75 4.20  
## 7 1954 Roberts 27 PHI 23 15 45 336.7 289 111 35 56 185 2.97 4.03  
## 8 1955 Roberts 28 PHI 23 14 41 305.0 292 111 41 53 160 3.28 3.96  
## 9 1956 Roberts 29 PHI 19 18 43 297.3 328 147 46 40 157 4.45 3.73  
## 10 1957 Roberts 30 PHI 10 22 39 249.7 246 113 40 43 128 4.07 3.80  
## 11 1958 Roberts 31 PHI 17 14 35 269.7 270 97 30 51 130 3.24 3.95  
## 12 1959 Roberts 32 PHI 15 17 35 257.3 267 122 34 35 137 4.27 4.11  
## 13 1960 Roberts 33 PHI 12 16 35 237.3 256 106 31 34 122 4.02 3.88  
## 14 1961 Roberts 34 PHI 1 10 26 117.0 154 76 19 23 54 5.85 4.07  
## 15 1962 Roberts 35 BAL 10 9 27 191.3 176 59 17 41 102 2.78 3.77  
## 16 1963 Roberts 36 BAL 14 13 35 251.3 230 93 35 40 124 3.33 3.52  
## 17 1964 Roberts 37 BAL 13 7 31 204.0 203 66 18 52 109 2.91 3.59  
## 18 1965 Roberts 38 BAL/HOU 10 9 30 190.7 171 59 18 30 97 2.78 3.42  
## 19 1966 Roberts 39 CHC/HOU 5 8 24 112.0 141 60 15 21 54 4.82 3.54  
## X.ERA.  
## 1 124  
## 2 107  
## 3 135  
## 4 127  
## 5 141  
## 6 152  
## 7 136  
## 8 121  
## 9 84  
## 10 93  
## 11 122  
## 12 96  
## 13 96  
## 14 70  
## 15 136  
## 16 106  
## 17 123  
## 18 123  
## 19 73

ford = subset(case.3.2, Player=="Ford")  
 ford

## Year Player Age Tm W L G IP H ER HR BB SO ERA X.lgERA X.ERA.  
## 20 1950 Ford 21 NYY 9 1 20 112.0 87 35 7 52 59 2.81 4.30 153  
## 21 1953 Ford 24 NYY 18 6 32 207.0 187 69 13 110 110 3.00 3.68 123  
## 22 1954 Ford 25 NYY 16 8 34 210.7 170 66 10 101 125 2.82 3.42 121  
## 23 1955 Ford 26 NYY 18 7 39 253.7 188 74 20 113 137 2.63 3.76 143  
## 24 1956 Ford 27 NYY 19 6 31 225.7 187 62 13 84 141 2.47 3.87 156  
## 25 1957 Ford 28 NYY 11 5 24 129.3 114 37 10 53 84 2.57 3.60 140  
## 26 1958 Ford 29 NYY 14 7 30 219.3 174 49 14 62 145 2.01 3.54 176  
## 27 1959 Ford 30 NYY 16 10 35 204.0 194 69 13 89 114 3.04 3.63 119  
## 28 1960 Ford 31 NYY 12 9 33 192.7 168 66 15 65 85 3.08 3.60 117  
## 29 1961 Ford 32 NYY 25 4 39 283.0 242 101 23 92 209 3.21 3.70 115  
## 30 1962 Ford 33 NYY 17 8 38 257.7 243 83 22 69 160 2.90 3.73 129  
## 31 1963 Ford 34 NYY 24 7 38 269.3 240 82 26 56 189 2.74 3.52 128  
## 32 1964 Ford 35 NYY 17 6 39 244.7 212 58 10 57 172 2.13 3.62 170  
## 33 1965 Ford 36 NYY 16 13 37 244.3 241 88 22 50 162 3.24 3.39 105  
## 34 1966 Ford 37 NYY 2 5 22 73.0 79 20 8 24 43 2.47 3.33 135  
## 35 1967 Ford 38 NYY 2 4 7 44.0 40 8 2 9 21 1.64 3.13 191

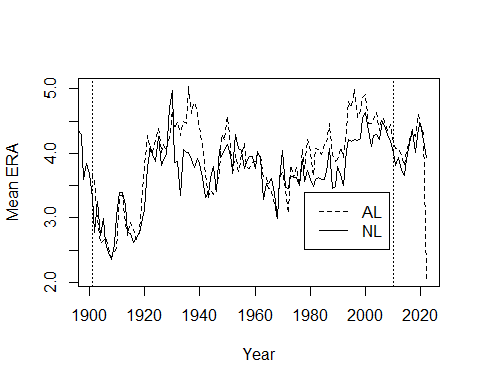
stem.leaf.backback(ford$ERA, roberts$ERA, m=2, Min=1.5, Max=6)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 1.2, leaf unit: 0.1   
## ford$ERA roberts$ERA  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 1\* |   
## 1 6| 1. |   
## 5 4410| 2\* |   
## (6) 988765| 2. |577799 6   
## 5 22000| 3\* |001223 (6)   
## | 3. |6 7   
## | 4\* |0024 6   
## | 4. |8 2   
## | 5\* |   
## | 5. |8 1   
## | 6\* |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 16 19   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The back-to-back stemplot seems to be the same as Figure 3.5.

In the first edition of TSUB, Albert forgot to include league average information. **case\_3\_2.csv** contains this information for ERA which is what we need. However, it might be useful to be able to get this, or similar, information in the future. Code for generating more complete league average data is presented below.

### Make Figure 3.6  
 ### To get league average pitcher information...  
 ### The data can be found at   
 ### http://www.baseball-reference.com/leagues/NL/pitch.shtml#teams\_standard\_pitching::none  
 ### and http://www.baseball-reference.com/leagues/AL/pitch.shtml#teams\_standard\_pitching::none  
 ### Click on CSV and then EXPORT to save them to your machine.  
 ### The files are also available at the URLs given below.  
 nlpitch = read.csv("http://facweb1.redlands.edu/fac/jim\_bentley/Data/FYS04/nlpitchingaverages.csv")  
 alpitch = read.csv("http://facweb1.redlands.edu/fac/jim\_bentley/Data/FYS04/alpitchingaverages.csv")  
 nlpitch$lgID = "NL" ### Identify the data as NL  
 alpitch$lgID = "AL" ### Identify the data as AL  
   
 pitch = rbind(alpitch, nlpitch) ### Stack the data into a single dataframe  
 pitch = pitch %>% filter(Year < 2022) ### Get rid of the incomplete 2022 season  
 plot(alpitch$Year,alpitch$ERA,type="l", lty=2, xlab="Year", ylab="Mean ERA")  
 lines(nlpitch$Year,nlpitch$ERA, lty=1)  
   
 ### Note that Albert only used years where both the AL and NL had data -- 1901 on.  
 ### Albert didn't have data beyond about 2010 but we have through 2021.  
 abline(v=2010,lty=3)  
 abline(v=1901, lty=3)  
 ### Add a legend.  
 legend(1978,3.4,legend=c("AL","NL"),lty=c(2,1))

 We can recreate the plot using actual data scraping rather than downloading.

### Install and load a few support packages.  
 p\_load(rvest) # Scrapes  
 p\_load(stringr) # String tools  
 p\_load(tidyr) # Data frame manipulation  
 ### Get the NL pitching page  
 url = 'http://www.baseball-reference.com/leagues/NL/pitch.shtml#teams\_standard\_pitching::none'   
 webpage = read\_html(url)  
 ### Use the rvest functions html\_nodes and html\_table to extract what we want  
 ### Grab the tables on the page  
 nl\_table = html\_nodes(webpage, 'table')  
 nl\_table

## {xml\_nodeset (2)}  
## [1] <table class="sortable stats\_table" id="teams\_standard\_pitching" data-col ...  
## [2] <table class="sortable stats\_table" id="teams\_standard\_pitching\_totals" d ...

### Two tables are read. Assign the tables to separate data frames  
 nl1 = html\_table(nl\_table)[[1]]  
 nl2 = html\_table(nl\_table)[[2]]  
 head(nl1)

## # A tibble: 6 × 32  
## Year Tms `#P` PAge `R/G` ERA G GF CG SHO tSho SV IP   
## <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>  
## 1 2022 15 377 29.1 4.47 4.14 1292 0.99 0.01 0.00 0.06 0.26 8.86   
## 2 2021 15 497 28.9 4.48 4.20 2428 0.99 0.01 0.01 0.07 0.25 8.76   
## 3 2020 15 383 28.4 4.66 4.47 898 0.98 0.02 0.01 0.06 0.21 8.56   
## 4 2019 15 426 28.3 4.71 4.38 2430 0.99 0.01 0.00 0.06 0.25 8.94   
## 5 2018 15 431 28.2 4.34 4.02 2432 0.99 0.01 0.00 0.07 0.26 8.98   
## 6 2017 15 381 28.4 4.63 4.34 2430 0.99 0.01 0.01 0.06 0.25 8.90   
## # … with 19 more variables: H <chr>, R <chr>, ER <chr>, HR <chr>, BB <chr>,  
## # IBB <chr>, SO <chr>, HBP <chr>, BK <chr>, WP <chr>, BF <chr>, WHIP <chr>,  
## # BAbip <chr>, H9 <chr>, HR9 <chr>, BB9 <chr>, SO9 <chr>, `SO/W` <chr>,  
## # E <chr>

head(nl2)

## # A tibble: 6 × 32  
## Year Tms `#P` PAge `R/G` ERA G GF CG SHO tSho SV IP   
## <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 2022 15 377 29.1 4.47 4.14 1292 1283 9 3 83 331 11451…  
## 2 2021 15 497 28.9 4.48 4.20 2428 2399 29 18 166 596 21265…  
## 3 2020 15 383 28.4 4.66 4.47 898 880 18 7 57 193 7689.1  
## 4 2019 15 426 28.3 4.71 4.38 2430 2412 18 11 143 607 21732…  
## 5 2018 15 431 28.2 4.34 4.02 2432 2415 17 8 169 621 21828…  
## 6 2017 15 381 28.4 4.63 4.34 2430 2403 27 13 135 605 21626…  
## # … with 19 more variables: H <chr>, R <chr>, ER <chr>, HR <chr>, BB <chr>,  
## # IBB <chr>, SO <chr>, HBP <chr>, BK <chr>, WP <chr>, BF <chr>, WHIP <chr>,  
## # BAbip <chr>, H9 <chr>, HR9 <chr>, BB9 <chr>, SO9 <chr>, `SO/W` <chr>,  
## # E <chr>

### Since the first table is averages and the second is totals, get rid of the second  
 nl = nl1  
 rm(nl1, nl2)  
 head(nl)

## # A tibble: 6 × 32  
## Year Tms `#P` PAge `R/G` ERA G GF CG SHO tSho SV IP   
## <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>  
## 1 2022 15 377 29.1 4.47 4.14 1292 0.99 0.01 0.00 0.06 0.26 8.86   
## 2 2021 15 497 28.9 4.48 4.20 2428 0.99 0.01 0.01 0.07 0.25 8.76   
## 3 2020 15 383 28.4 4.66 4.47 898 0.98 0.02 0.01 0.06 0.21 8.56   
## 4 2019 15 426 28.3 4.71 4.38 2430 0.99 0.01 0.00 0.06 0.25 8.94   
## 5 2018 15 431 28.2 4.34 4.02 2432 0.99 0.01 0.00 0.07 0.26 8.98   
## 6 2017 15 381 28.4 4.63 4.34 2430 0.99 0.01 0.01 0.06 0.25 8.90   
## # … with 19 more variables: H <chr>, R <chr>, ER <chr>, HR <chr>, BB <chr>,  
## # IBB <chr>, SO <chr>, HBP <chr>, BK <chr>, WP <chr>, BF <chr>, WHIP <chr>,  
## # BAbip <chr>, H9 <chr>, HR9 <chr>, BB9 <chr>, SO9 <chr>, `SO/W` <chr>,  
## # E <chr>

### Repeat the above for the AL  
 ### Get the AL pitching page  
 url = 'http://www.baseball-reference.com/leagues/AL/pitch.shtml#teams\_standard\_pitching::none'  
 webpage = read\_html(url)  
 ### Use the rvest functions html\_nodes and html\_table to extract what we want  
 ### Grab the tables on the page  
 al\_table = html\_nodes(webpage, 'table')  
 al\_table

## {xml\_nodeset (2)}  
## [1] <table class="sortable stats\_table" id="teams\_standard\_pitching" data-col ...  
## [2] <table class="sortable stats\_table" id="teams\_standard\_pitching\_totals" d ...

### Assign the first table to the data frame  
 al = html\_table(al\_table)[[1]]  
 head(al)

## # A tibble: 6 × 32  
## Year Tms `#P` PAge `R/G` ERA G GF CG SHO tSho SV IP   
## <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>  
## 1 2022 15 374 28.6 4.19 3.86 1284 0.99 0.01 0.00 0.08 0.25 8.86   
## 2 2021 15 471 28.5 4.58 4.32 2430 0.99 0.01 0.00 0.05 0.24 8.79   
## 3 2020 15 375 28.2 4.63 4.42 898 0.99 0.01 0.01 0.05 0.26 8.66   
## 4 2019 15 456 28.4 4.95 4.60 2428 0.99 0.01 0.01 0.05 0.24 8.93   
## 5 2018 15 420 28.6 4.56 4.27 2430 0.99 0.01 0.00 0.06 0.26 8.91   
## 6 2017 15 417 28.6 4.67 4.37 2430 0.99 0.01 0.01 0.05 0.24 8.90   
## # … with 19 more variables: H <chr>, R <chr>, ER <chr>, HR <chr>, BB <chr>,  
## # IBB <chr>, SO <chr>, HBP <chr>, BK <chr>, WP <chr>, BF <chr>, WHIP <chr>,  
## # BAbip <chr>, H9 <chr>, HR9 <chr>, BB9 <chr>, SO9 <chr>, `SO/W` <chr>,  
## # E <chr>

### Did we get the right years for the leagues?  
 range(al$Year)

## [1] "1901" "Year"

table(al$Year)

##   
## 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 Year   
## 1 1 1 1 1 1 1 1 1 1 4

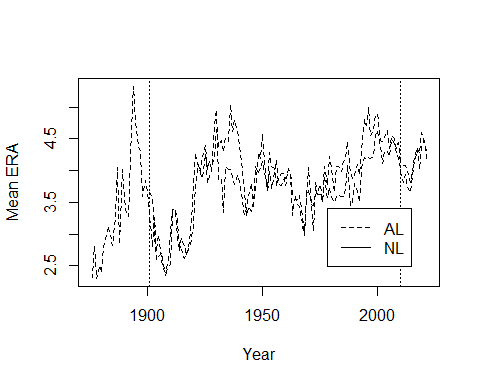
### Oops, al seems to have multiple headers in the data and "Year" appears within Year. We fix that  
 ### by filtering out (!= is not equal) the bad obs and then convert Year to a numeric.  
 al = al %>%  
 filter(Year != "Year") %>%  
 mutate(Year = as.numeric(Year)) %>%  
 filter(Year < 2022) ### As before, we drop 2022 because it's not complete  
 range(al$Year)

## [1] 1901 2021

nl = nl %>%  
 filter(Year != "Year") %>%  
 mutate(Year = as.numeric(Year)) %>%  
 filter(Year < 2022) ### As before, we drop 2022 because it's not complete  
 range(nl$Year)

## [1] 1876 2021

# That's better. Now compare ERAs.  
 plot(nl$Year,nl$ERA,type="l", lty=2, xlab="Year", ylab="Mean ERA")  
 lines(al$Year,al$ERA, lty=2)  
 abline(v=2010,lty=3)  
 abline(v=1901, lty=3)  
 legend(1978,3.4,legend=c("AL","NL"),lty=c(2,1))

 A bit of work, but it’s nice to know that we can get up to date data.

We move on to Figure 3.7.

###  
 ### Create Figure 3.7  
 ### We can use the data already imported for Figure 3.5  
 head(roberts)

## Year Player Age Tm W L G IP H ER HR BB SO ERA X.lgERA X.ERA.  
## 1 1948 Roberts 21 PHI 7 9 20 146.7 148 52 10 61 84 3.19 3.96 124  
## 2 1949 Roberts 22 PHI 15 15 43 226.7 229 93 15 75 95 3.69 3.96 107  
## 3 1950 Roberts 23 PHI 20 11 40 304.3 282 102 29 77 146 3.02 4.06 135  
## 4 1951 Roberts 24 PHI 21 15 44 315.0 284 106 20 64 127 3.03 3.84 127  
## 5 1952 Roberts 25 PHI 28 7 39 330.0 292 95 22 45 148 2.59 3.66 141  
## 6 1953 Roberts 26 PHI 23 16 44 346.7 324 106 30 61 198 2.75 4.20 152

### Note that x.ERA. (\*ERA) is equal to the formula in the book (within rounding)  
 cbind(roberts$X.lgERA/roberts$ERA\*100, roberts$X.ERA.)[1:5,]

## [,1] [,2]  
## [1,] 124.1379 124  
## [2,] 107.3171 107  
## [3,] 134.4371 135  
## [4,] 126.7327 127  
## [5,] 141.3127 141

stem.leaf.backback(ford$X.ERA.,roberts$X.ERA., Max=190)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 12, leaf unit: 1   
## ford$X.ERA. roberts$X.ERA.  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 7 |03 2   
## | 8 |4 3   
## | 9 |366 6   
## 1 5| 10 |67 8   
## 4 975| 11 |   
## (4) 9831| 12 |123347 (6)   
## (1) 5| 13 |566 5   
## 7 30| 14 |1 2   
## 5 63| 15 |2 1   
## | 16 |   
## 3 60| 17 |   
## | 18 |   
## 1 1| 19 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 16 19   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This back-to-back stemplot is similar (within truncation/rounding) to Figure 3.7 in **TSUB2**.

### Case 3.3

Albert now looks at home runs and the rate at which they occur. For this case we download the data from Albert’s GitHub site.

### Get the data. Be careful about getting the "raw" CSV file from GitHub  
 case.3.3 <- read.csv("https://raw.githubusercontent.com/bayesball/Teaching-Statistics-Using-Baseball/master/case\_3\_3.csv")  
   
 ### See what's in it  
 head(case.3.3)

## YEAR HR G RATE  
## 1 1927 54 156 0.35  
## 2 1927 84 153 0.55  
## 3 1927 109 155 0.70  
## 4 1927 74 153 0.48  
## 5 1927 29 153 0.19  
## 6 1927 39 154 0.25

### Get the number of teams in each year to compare to Table 3.4  
 case.3.3 %>%   
 group\_by(YEAR) %>%  
 summarize(nTeams = n())

## # A tibble: 4 × 2  
## YEAR nTeams  
## <int> <int>  
## 1 1927 16  
## 2 1961 18  
## 3 1998 30  
## 4 2001 30

### Or  
 table(case.3.3$YEAR)

##   
## 1927 1961 1998 2001   
## 16 18 30 30

It looks like the data are okay.

*R* doesn’t generate parallel stemplots, so for Figure 3.8 we create multiple back-to-back stemplots

homers.1927 = case.3.3[case.3.3$YEAR==1927,] ### Subset those rows/obs that are YEAR == 1927  
 homers.1961 = case.3.3[case.3.3$YEAR==1961,] ### Blank after the comma means keep all columns  
 homers.1998 = case.3.3[case.3.3$YEAR==1998,]  
 homers.2001 = case.3.3[case.3.3$YEAR==2001,]   
   
 p\_load(aplpack)  
 stem.leaf.backback(homers.1927$RATE,homers.1961$RATE, Min=.1, Max=1.6)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 0.12, leaf unit: 0.01   
## homers.1927$RATE  
## homers.1961$RATE  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 4 9887| 1 |   
## 7 544| 2 |   
## (5) 76553| 3 |   
## 4 8| 4 |   
## 3 5| 5 |6 1   
## | 6 |669 4   
## 2 0| 7 |4 5   
## | 8 |35 7   
## | 9 |13 (2)   
## 1 2| 10 |234 (3)   
## | 11 |0378 6   
## | 12 |1 2   
## | 13 |   
## | 14 |7 1   
## | 15 |   
## | 16 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 16 18   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

stem.leaf.backback(homers.1927$RATE,homers.1998$RATE, Min=.1, Max=1.6)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 0.12, leaf unit: 0.01   
## homers.1927$RATE  
## homers.1998$RATE  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 4 9887| 1 |   
## 7 544| 2 |   
## (5) 76553| 3 |   
## 4 8| 4 |   
## 3 5| 5 |   
## | 6 |68 2   
## 2 0| 7 |017 5   
## | 8 |245 8   
## | 9 |1124889 (7)   
## 1 2| 10 |223 (3)   
## | 11 |3 12   
## | 12 |12478 11   
## | 13 |02367 6   
## | 14 |5 1   
## | 15 |   
## | 16 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 16 30   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

stem.leaf.backback(homers.1927$RATE,homers.2001$RATE, Min=.1, Max=1.6)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 0.12, leaf unit: 0.01   
## homers.1927$RATE homers.2001$RATE  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 4 9887| 1 |   
## 7 544| 2 |   
## (5) 76553| 3 |   
## 4 8| 4 |   
## 3 5| 5 |   
## | 6 |   
## 2 0| 7 |5 1   
## | 8 |146 4   
## | 9 |14899 9   
## 1 2| 10 |112479 (6)   
## | 11 |   
## | 12 |0033367889 (10)   
## | 13 |112 5   
## | 14 |5 2   
## | 15 |2 1   
## | 16 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 16 30   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

stem.leaf.backback(homers.1961$RATE,homers.1998$RATE, Min=.1, Max=1.6)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 0.12, leaf unit: 0.01   
## homers.1961$RATE  
## homers.1998$RATE  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 1 |   
## | 2 |   
## | 3 |   
## | 4 |   
## 1 6| 5 |   
## 4 966| 6 |68 2   
## 5 4| 7 |017 5   
## 7 53| 8 |245 8   
## (2) 31| 9 |1124889 (7)   
## (3) 432| 10 |223 (3)   
## 6 8730| 11 |3 12   
## 2 1| 12 |12478 11   
## | 13 |02367 6   
## 1 7| 14 |5 1   
## | 15 |   
## | 16 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 18 30   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

stem.leaf.backback(homers.1961$RATE,homers.2001$RATE, Min=.1, Max=1.6)

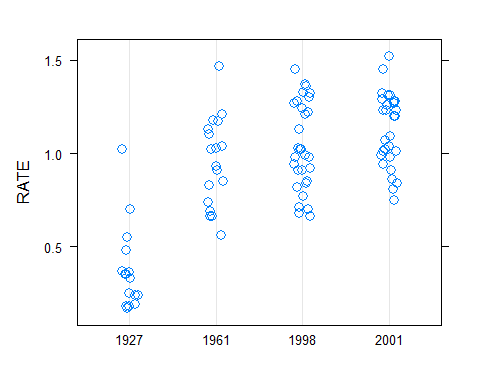
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 0.12, leaf unit: 0.01   
## homers.1961$RATE homers.2001$RATE  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 1 |   
## | 2 |   
## | 3 |   
## | 4 |   
## 1 6| 5 |   
## 4 966| 6 |   
## 5 4| 7 |5 1   
## 7 53| 8 |146 4   
## (2) 31| 9 |14899 9   
## (3) 432| 10 |112479 (6)   
## 6 8730| 11 |   
## 2 1| 12 |0033367889 (10)   
## | 13 |112 5   
## 1 7| 14 |5 2   
## | 15 |2 1   
## | 16 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 18 30   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

stem.leaf.backback(homers.1998$RATE,homers.2001$RATE, Min=.1, Max=1.6)

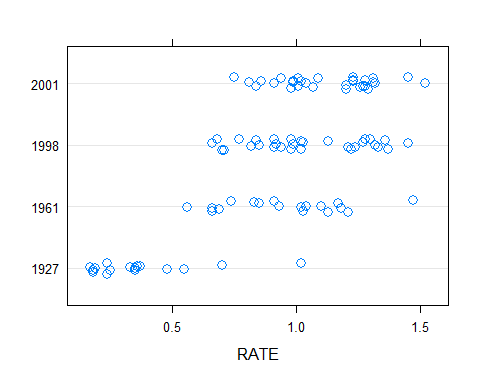
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## 1 | 2: represents 0.12, leaf unit: 0.01   
## homers.1998$RATE homers.2001$RATE  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## | 1 |   
## | 2 |   
## | 3 |   
## | 4 |   
## | 5 |   
## 2 86| 6 |   
## 5 710| 7 |5 1   
## 8 542| 8 |146 4   
## (7) 9884211| 9 |14899 9   
## (3) 322| 10 |112479 (6)   
## 12 3| 11 |   
## 11 87421| 12 |0033367889 (10)   
## 6 76320| 13 |112 5   
## 1 5| 14 |5 2   
## | 15 |2 1   
## | 16 |   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## n: 30 30   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A quick dotplot might help here.

### Figure 3.8 can be generated using the lattice package  
 p\_load(lattice)  
 dotplot(RATE ~ factor(YEAR), data=case.3.3, cex=1.25, pch=1, jitter.x=TRUE)



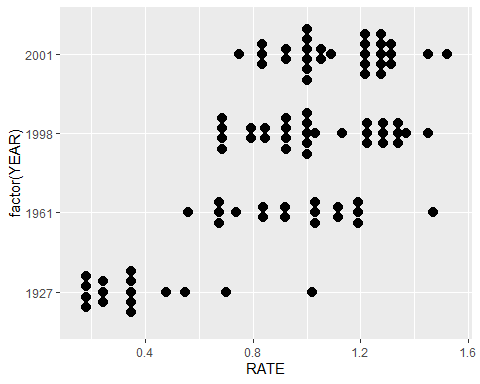
dotplot(factor(YEAR) ~ RATE, data=case.3.3, cex=1.25, pch=1, jitter.y=TRUE)



### or ggplot2  
 plt = case.3.3 %>%  
 ggplot(aes(x = factor(YEAR), y = RATE)) +  
 geom\_dotplot(binaxis = "y", stackdir = "center", binwidth=0.05, dotsize=0.75)  
 plt



plt + coord\_flip()



Tough to interpret? Yep. Maybe a boxplot would be better.

### Create 5-number summaries  
 by(case.3.3[,"RATE"],case.3.3[,"YEAR"], summary)

## case.3.3[, "YEAR"]: 1927  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1700 0.2275 0.3400 0.3725 0.3975 1.0200   
## ------------------------------------------------------------   
## case.3.3[, "YEAR"]: 1961  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.5600 0.7625 0.9750 0.9544 1.1225 1.4700   
## ------------------------------------------------------------   
## case.3.3[, "YEAR"]: 1998  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.660 0.865 1.005 1.040 1.262 1.450   
## ------------------------------------------------------------   
## case.3.3[, "YEAR"]: 2001  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.750 0.990 1.145 1.123 1.278 1.520

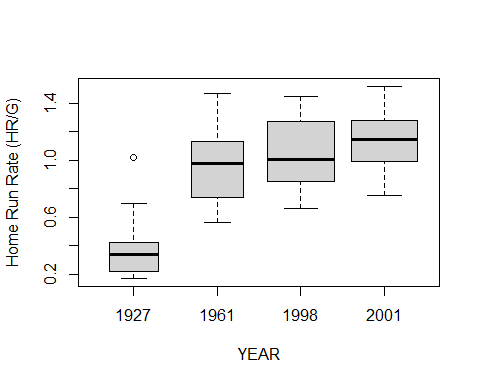
### and figure out how many teams there were in each year  
 cnts = table(case.3.3$YEAR)  
 cnts

##   
## 1927 1961 1998 2001   
## 16 18 30 30

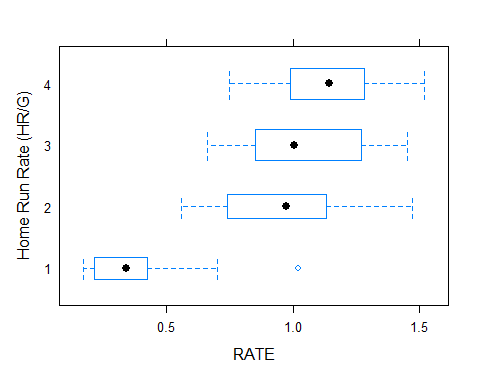
### or  
 case.3.3 %>%  
 group\_by(YEAR) %>%  
 summarise(n = n(),  
 min = fivenum(RATE)[1],  
 Q1 = fivenum(RATE)[2],  
 median = fivenum(RATE)[3],  
 Q3 = fivenum(RATE)[4],  
 max = fivenum(RATE)[5]  
 )

## # A tibble: 4 × 7  
## YEAR n min Q1 median Q3 max  
## <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1927 16 0.17 0.215 0.34 0.425 1.02  
## 2 1961 18 0.56 0.74 0.975 1.13 1.47  
## 3 1998 30 0.66 0.85 1.00 1.27 1.45  
## 4 2001 30 0.75 0.99 1.14 1.28 1.52

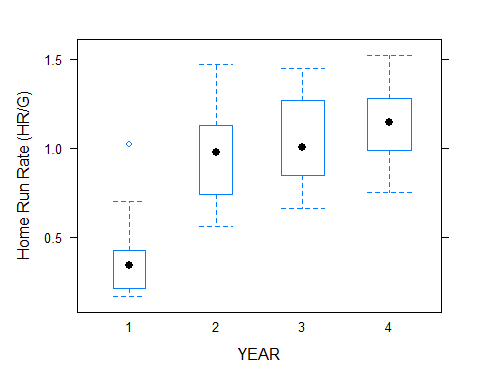
### The Figure 3.9 boxplots look like  
 boxplot(RATE~YEAR, data=case.3.3, ylab="Home Run Rate (HR/G)", varwidth=TRUE)



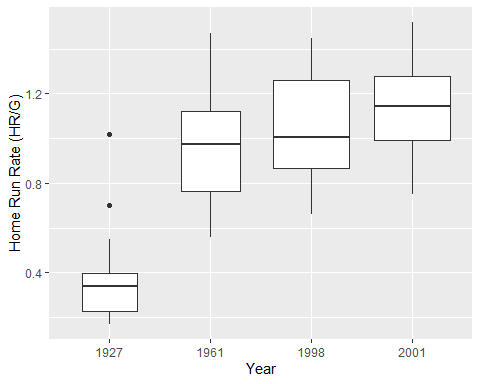
p\_load(lattice)  
 bwplot(YEAR~RATE, data=case.3.3, ylab="Home Run Rate (HR/G)", varwidth=TRUE)



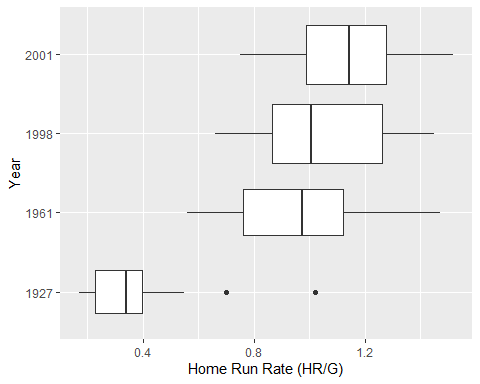
bwplot(RATE~YEAR, data=case.3.3, ylab="Home Run Rate (HR/G)", varwidth=TRUE, horizontal = FALSE)



p\_load(ggplot2)  
 ggplot(case.3.3, aes(factor(YEAR), RATE)) + geom\_boxplot(varwidth=TRUE) + xlab("Year") + ylab("Home Run Rate (HR/G)")



ggplot(case.3.3,aes(factor(YEAR), RATE)) + geom\_boxplot(varwidth=TRUE) + coord\_flip() + xlab("Year") + ylab("Home Run Rate (HR/G)")



### Case 3.4

The text now discusses the idea of the normal distribution. We will suplement the discussion with a couple of “modern” techniques. For this case we download the data from the author’s site. Note that many of the values disagree with those in the text. Please let me know if you figure out why.

### Get the data. Be careful about getting the "raw" CSV file from GitHub  
 case.3.4 <- read.csv("https://raw.githubusercontent.com/bayesball/Teaching-Statistics-Using-Baseball/master/case\_3\_4.csv")  
 head(case.3.4)

## playerID SLG  
## 1 abreujo02 0.5809353  
## 2 ackledu01 0.3984064  
## 3 adamsma01 0.4573055  
## 4 altuvjo01 0.4530303  
## 5 amarial01 0.3144208  
## 6 andruel01 0.3327948

case.3.4$SLG

## [1] 0.5809353 0.3984064 0.4573055 0.4530303 0.3144208 0.3327948 0.3604888  
## [8] 0.5000000 0.3786078 0.5244123 0.3475336 0.4918033 0.4019851 0.4401349  
## [15] 0.3624535 0.3603604 0.5057283 0.4528302 0.3488372 0.3732252 0.3788707  
## [22] 0.4450085 0.3869801 0.4577465 0.5237316 0.4118896 0.4503043 0.2896040  
## [29] 0.4537815 0.3747899 0.4911243 0.3939962 0.3655914 0.4375000 0.4500000  
## [36] 0.4267782 0.3736264 0.3003953 0.3145336 0.3890020 0.3628510 0.5252855  
## [43] 0.4044444 0.4570858 0.4013015 0.3857442 0.4300169 0.5665138 0.3297872  
## [50] 0.4555921 0.4163880 0.4805447 0.4149184 0.4012346 0.4191304 0.5471698  
## [57] 0.3765112 0.4064665 0.3403361 0.3955774 0.3986175 0.4589615 0.4612850  
## [64] 0.3831169 0.4216216 0.4340909 0.4159483 0.5418719 0.4721649 0.4773519  
## [71] 0.4822335 0.4316163 0.3776683 0.3882979 0.3325000 0.3552398 0.3724008  
## [78] 0.4903846 0.3723404 0.3563433 0.3839442 0.3672655 0.4407666 0.3808463  
## [85] 0.3976143 0.3796134 0.4462659 0.3588517 0.3371212 0.3467337 0.3777240  
## [92] 0.3778706 0.3132530 0.3608247 0.4689441 0.4112903 0.3827751 0.5064695  
## [99] 0.3970827 0.4195906 0.3300000 0.3822115 0.4554656 0.3481781 0.3800000  
## [106] 0.4038462 0.3545817 0.4649573 0.3862928 0.4525253 0.5532880 0.3639775  
## [113] 0.5650624 0.3714286 0.4060606 0.5419708 0.3571429 0.3873518 0.3861386  
## [120] 0.3701431 0.4960159 0.4748858 0.4380000 0.3610503 0.3846154 0.4026846  
## [127] 0.4192771 0.3950104 0.5173745 0.4548673 0.3686200 0.3756806 0.4446154  
## [134] 0.4428571 0.4031142 0.3722944 0.4230769 0.4899452 0.4123134 0.4802867  
## [141] 0.4660348 0.4083601 0.4271255 0.4476615 0.4730832 0.3610649 0.3983607  
## [148] 0.3983740 0.5267176 0.3940520 0.4149660 0.4269871 0.4716049 0.3538462  
## [155] 0.4542373 0.3255361 0.3314815 0.4401806 0.3688699 0.4163934 0.5547310  
## [162] 0.3827434 0.3977273 0.5614618 0.3333333 0.4911661 0.4074703 0.4351464  
## [169] 0.3251232 0.4053537 0.4667969 0.4550562 0.3880597 0.3738318 0.4020619  
## [176] 0.3947368 0.4041096

We can create Figure 3.10. Again, rounding creates some minor differences.

### Create Figure 3.10  
 stem(case.3.4$SLG)

##   
## The decimal point is 2 digit(s) to the left of the |  
##   
## 28 | 0  
## 30 | 0345  
## 32 | 560013337  
## 34 | 07889455679  
## 36 | 0011123467990122234456788899  
## 38 | 00123334566677889445567888889  
## 40 | 112233444566781225566699  
## 42 | 023777024588  
## 44 | 00135568003334455567789  
## 46 | 1567922357  
## 48 | 012001126  
## 50 | 0667  
## 52 | 4457  
## 54 | 22735  
## 56 | 157  
## 58 | 1

### or  
 p\_load(aplpack)  
 stem.leaf(case.3.4$SLG)

## 1 | 2: represents 0.12  
## leaf unit: 0.01  
## n: 177  
## 1 2. | 8  
## 5 3\* | 0111  
## 14 t | 222333333  
## 25 f | 44444555555  
## 54 s | 66666666666677777777777777777  
## 82 3. | 8888888888888888999999999999  
## (25) 4\* | 0000000000000011111111111  
## 70 t | 22222333333  
## 59 f | 44444444555555555555555  
## 36 s | 6666677777  
## 26 4. | 888899999  
## 17 5\* | 0001  
## 13 t | 2222  
## 9 f | 44455  
## 4 s | 666  
## 1 5. | 8

The values that Albert presents can be computed fairly easily.

### Compute the statistics that Albert presents  
 n = length(case.3.4$SLG)  
 n # The number of observations

## [1] 177

xbar = mean(case.3.4$SLG)  
 xbar # The sample mean

## [1] 0.4158045

median(case.3.4$SLG)

## [1] 0.4038462

mean(case.3.4$SLG, trim=0.05) # A 10% trimmed mean

## [1] 0.4137492

variance = var(case.3.4$SLG)  
 sd = sqrt(variance)   
 sd # Standard deviation

## [1] 0.05955793

sqrt(variance/n) # Standard error of the mean

## [1] 0.004476648

summary(case.3.4$SLG)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.2896 0.3738 0.4038 0.4158 0.4542 0.5809

### Computing the mean plus/minus k standard deviations is "easy". Use rounded values to match Albert.   
 xbar = .416  
 sd = 0.060  
 xbar + c(-1,1)%\*%t(1:3)\*sd # Compute values for k=1,2,3 at the same time

## [,1] [,2] [,3]  
## [1,] 0.356 0.296 0.236  
## [2,] 0.476 0.536 0.596

# Compare with actual values  
 length(case.3.4$SLG[.356 < case.3.4$SLG & case.3.4$SLG < .476])

## [1] 128

length(case.3.4$SLG[.296 < case.3.4$SLG & case.3.4$SLG < .536])

## [1] 167

length(case.3.4$SLG[.236 < case.3.4$SLG & case.3.4$SLG < .596])

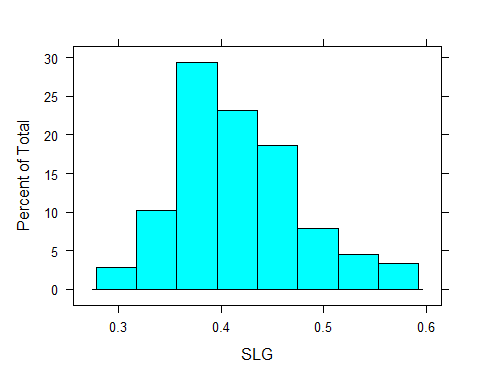
## [1] 177

c(126,167,177)/178 \* 100

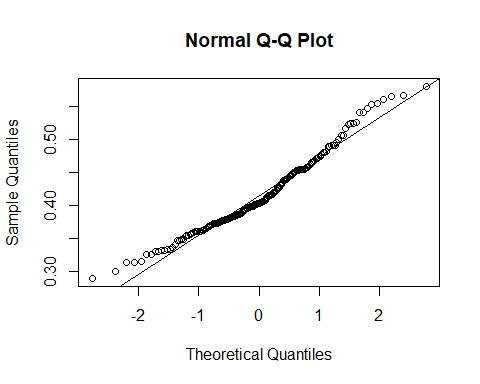
## [1] 70.78652 93.82022 99.43820

Graphical methods are sometimes helpful. A graphical method for checking normality is the normal probability or normal quantile plot. If the data are normally distributed, what we observe and what we expect to see should be the same. Hence, the plotted values should fall on the diagonal line.

histogram(~SLG, data=case.3.4)



qqnorm(case.3.4$SLG)  
 qqline(case.3.4$SLG)

 There are also some “tests” for normality.

p\_load(nortest)  
 ad.test(case.3.4$SLG)

##   
## Anderson-Darling normality test  
##   
## data: case.3.4$SLG  
## A = 1.6986, p-value = 0.0002268

cvm.test(case.3.4$SLG)

##   
## Cramer-von Mises normality test  
##   
## data: case.3.4$SLG  
## W = 0.29369, p-value = 0.0003855

lillie.test(case.3.4$SLG)

##   
## Lilliefors (Kolmogorov-Smirnov) normality test  
##   
## data: case.3.4$SLG  
## D = 0.093368, p-value = 0.0007091

pearson.test(case.3.4$SLG)

##   
## Pearson chi-square normality test  
##   
## data: case.3.4$SLG  
## P = 24.492, p-value = 0.0269

sf.test(case.3.4$SLG)

##   
## Shapiro-Francia normality test  
##   
## data: case.3.4$SLG  
## W = 0.97109, p-value = 0.001516

shapiro.test(case.3.4$SLG)

##   
## Shapiro-Wilk normality test  
##   
## data: case.3.4$SLG  
## W = 0.96939, p-value = 0.0006182

Since all of the p-values are “small” it appears that it is unlikely that the data are truly normal. How different from normal they are is debatable.

### Case 3.5

The last case in this chapter discusses battting averages. For this case we download the data from the author’s site.

### Get the data. Be careful about getting the "raw" CSV file from GitHub  
 case.3.5 <- read.csv("https://raw.githubusercontent.com/bayesball/Teaching-Statistics-Using-Baseball/master/case\_3\_5.csv")  
 head(case.3.5)

## Lastname Firstname Year AB H X2B X3B HR  
## 1 Appling Luke 1941 592 186 26 8 1  
## 2 Benjamin Stan 1941 480 113 20 7 3  
## 3 Berardino Johnny 1941 469 127 30 4 5  
## 4 Bloodworth Jimmy 1941 506 124 24 3 7  
## 5 Boudreau Lou 1941 579 149 45 8 10  
## 6 Bragan Bobby 1941 557 140 19 3 4

dim(case.3.5)

## [1] 98 8

table(case.3.5$Year)

##   
## 1941   
## 98

It appears that the data on GitHub aren’t quite right. We can pull the data from the **TSUB** package.

case.3.5 <- case\_3\_5  
 table(case.3.5$Year)

##   
## 1941 1977 1980 1994   
## 98 168 148 63

head(case.3.5)

## Lastname Firstname Year AB H X2B X3B HR  
## 1 Appling Luke 1941 592 186 26 8 1  
## 2 Benjamin Stan 1941 480 113 20 7 3  
## 3 Berardino Johnny 1941 469 127 30 4 5  
## 4 Bloodworth Jimmy 1941 506 124 24 3 7  
## 5 Boudreau Lou 1941 579 149 45 8 10  
## 6 Bragan Bobby 1941 557 140 19 3 4

Or, we can go the source… Mr. Lahman.

### Unfortunately, the online dataset is for 1941 and not 1977. We go to   
 ### Lahman to get the 1977 data.  
 p\_load(Lahman)  
 data(Batting)  
 head(Batting)

## playerID yearID stint teamID lgID G AB R H X2B X3B HR RBI SB CS BB SO  
## 1 abercda01 1871 1 TRO NA 1 4 0 0 0 0 0 0 0 0 0 0  
## 2 addybo01 1871 1 RC1 NA 25 118 30 32 6 0 0 13 8 1 4 0  
## 3 allisar01 1871 1 CL1 NA 29 137 28 40 4 5 0 19 3 1 2 5  
## 4 allisdo01 1871 1 WS3 NA 27 133 28 44 10 2 2 27 1 1 0 2  
## 5 ansonca01 1871 1 RC1 NA 25 120 29 39 11 3 0 16 6 2 2 1  
## 6 armstbo01 1871 1 FW1 NA 12 49 9 11 2 1 0 5 0 1 0 1  
## IBB HBP SH SF GIDP  
## 1 NA NA NA NA 0  
## 2 NA NA NA NA 0  
## 3 NA NA NA NA 1  
## 4 NA NA NA NA 0  
## 5 NA NA NA NA 0  
## 6 NA NA NA NA 0

batting = battingStats()  
 dim(batting)

## [1] 110495 29

batting = batting[batting$yearID %in% c(1941, 1977, 1980, 1994),]  
 dim(batting)

## [1] 3546 29

batting = batting[batting$lgID %in% c("AL","NL"),]  
 dim(batting)

## [1] 3546 29

batting = batting[batting$AB >= 400, ]  
 dim(batting)

## [1] 477 29

table(batting$yearID)

##   
## 1941 1977 1980 1994   
## 98 168 148 63

### Or, the tidyverse approach  
 p\_load(tidyverse)  
 batting = battingStats() %>% ### Get the supplemented batting data  
 filter(yearID %in% c(1941, 1977, 1980, 1994)) %>% ### Keep the right years  
 filter(lgID %in% c("AL", "NL")) %>% ### Keep only the major leagues  
 filter(AB >= 400) ### Keep people who actually batted  
 table(batting$yearID)

##   
## 1941 1977 1980 1994   
## 98 168 148 63

### Or, the shorter tidyverse approach  
 p\_load(tidyverse)  
 batting = battingStats() %>% ### Get the supplemented batting data  
 filter(yearID %in% c(1941, 1977, 1980, 1994) ### Keep the right years  
 & lgID %in% c("AL", "NL") ### Keep only the major leagues  
 & AB >= 400 ### Keep people who actually batted  
 )   
 table(batting$yearID)

##   
## 1941 1977 1980 1994   
## 98 168 148 63

It’s nice to know that the **TSUB** package data are the same as Lahman’s.

So, onward with the great batters. Albert looks at 1977 and there appear to have been 168 MLB batters with at least 400 appearances in 1977. We can analyze them as Albert did to see if our conclusions are the same.

### Get the 1977 data  
 batting.1977 <- batting %>% filter(yearID == 1977)  
   
 ### Figure 3.11  
 stem(batting.1977$BA)

##   
## The decimal point is 2 digit(s) to the left of the |  
##   
## 20 | 06  
## 21 | 6  
## 22 | 9  
## 23 | 0133559  
## 24 | 001135566778  
## 25 | 00111222344567799  
## 26 | 001111223334444556677899  
## 27 | 001122233445555557788999  
## 28 | 00000122223333444446777788999  
## 29 | 000111122336778899  
## 30 | 00022445778899  
## 31 | 011225788  
## 32 | 002568  
## 33 | 668  
## 34 |   
## 35 |   
## 36 |   
## 37 |   
## 38 | 8

p\_load(aplpack)  
 stem.leaf(batting.1977$BA)

## 1 | 2: represents 0.012  
## leaf unit: 0.001  
## n: 168  
## LO: 0.2 0.206  
## 3 21 | 6  
## 4 22 | 9  
## 11 23 | 0133559  
## 23 24 | 001135566778  
## 40 25 | 00111222344567799  
## 64 26 | 001111223334444556677899  
## (24) 27 | 001122233445555557788999  
## 80 28 | 00000122223333444446777788999  
## 51 29 | 000111122336778899  
## 33 30 | 00022445778899  
## 19 31 | 011225788  
## 10 32 | 002568  
## 4 33 | 668  
## HI: 0.388

Note that Albert says 168 in the text but the plot shows 173. This is strange. We’ll continue computing statistics and see if they differ from the text.

summary(batting.1977$BA)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.2000 0.2607 0.2780 0.2774 0.2923 0.3880

xbar = mean(batting.1977$BA)  
 xbar

## [1] 0.2774226

sd = sqrt(var(batting.1977$BA))  
 sd

## [1] 0.02717155

batting.1977$BA.z = (batting.1977$BA - xbar)/sd  
 stem(batting.1977$BA.z)

##   
## The decimal point is at the |  
##   
## -2 | 86  
## -2 | 3  
## -1 | 8776666  
## -1 | 444333222211100000  
## -0 | 99999988887766666666555555555  
## -0 | 44443333322222221111111100  
## 0 | 0011111111122222222222223444444444  
## 0 | 55555555566777888888899  
## 1 | 000111122222334  
## 1 | 555666889  
## 2 | 222  
## 2 |   
## 3 |   
## 3 |   
## 4 | 1

summary(batting.1977$BA.z)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -2.84940 -0.61361 0.02125 0.00000 0.54570 4.06960

Table 3.6 includes the 1900 season, so we can throw it in too. We compute the means and standard deivations.

p\_load(tidyverse)  
 table.3.6 = battingStats() %>% ### Get the supplemented batting data  
 filter(yearID %in% c(1900, 1941, 1977, 1980, 1994) ### Keep the right years  
 & lgID %in% c("AL", "NL") ### Keep only the major leagues  
 & AB >= 400 ### Keep people who actually batted  
 ) %>%  
 group\_by(yearID) %>% ### Group by year  
 summarize(BA.mean = mean(BA), ### Compute mean BA by year  
 BA.sd = sqrt(var(BA)), ### Compute the sd of BA by year  
 n = n() ### Confirm the number of batters  
 )  
 table.3.6

## # A tibble: 5 × 4  
## yearID BA.mean BA.sd n  
## <int> <dbl> <dbl> <int>  
## 1 1900 0.296 0.0374 45  
## 2 1941 0.281 0.0328 98  
## 3 1977 0.277 0.0272 168  
## 4 1980 0.279 0.0275 148  
## 5 1994 0.293 0.0320 63

## Chapter Four

Chapter 4 moves from looking at a single variable in isolation to looking at how variables behave together. Changing one may cause the other to change, or they may just change together. We discuss both causality and association.

### Case 4.1

In this section, Albert looks at team offensive stats. We will use the data from Albert’s site.

### Get the data. Be careful about getting the "raw" CSV file from GitHub  
 case.4.1 <- read.csv("https://raw.githubusercontent.com/bayesball/Teaching-Statistics-Using-Baseball/master/case\_4\_1.csv")  
 head(case.4.1)

## Tm NBat BatAge R.G G PA AB R H X2B X3B HR RBI SB CS BB SO  
## 1 ARI 52 27.6 3.80 162 6089 5552 615 1379 259 47 118 573 86 33 398 1165  
## 2 ATL 39 26.8 3.54 162 6064 5468 573 1316 240 22 123 545 95 33 472 1369  
## 3 BAL 44 28.3 4.35 162 6130 5596 705 1434 264 16 211 681 44 20 401 1285  
## 4 BOS 55 29.2 3.91 162 6226 5551 634 1355 282 20 123 601 63 25 535 1337  
## 5 CHC 48 26.8 3.79 162 6102 5508 614 1315 270 31 157 590 65 40 442 1477  
## 6 CHW 44 27.7 4.07 162 6077 5543 660 1400 279 32 155 625 85 36 417 1362  
## BA OBP SLG OPS OPS. TB GDP HBP SH SF IBB LOB  
## 1 0.248 0.302 0.376 0.678 88 2086 115 43 56 36 31 1092  
## 2 0.241 0.305 0.360 0.665 87 1969 121 43 53 27 31 1128  
## 3 0.256 0.311 0.422 0.734 106 2363 112 62 35 36 29 1088  
## 4 0.244 0.316 0.369 0.684 92 2046 138 68 20 52 36 1168  
## 5 0.239 0.300 0.385 0.684 88 2118 94 54 57 41 29 1069  
## 6 0.253 0.310 0.398 0.708 100 2208 127 60 19 38 33 1071

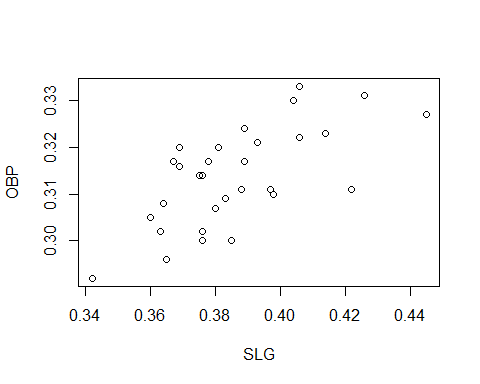
dim(case.4.1)

## [1] 30 29

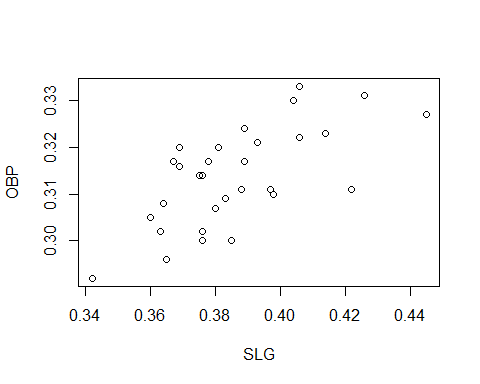
The data appear to be similar to Table 4.1.

Albert now looks at the relationships that exist between these variables using scatter plots.

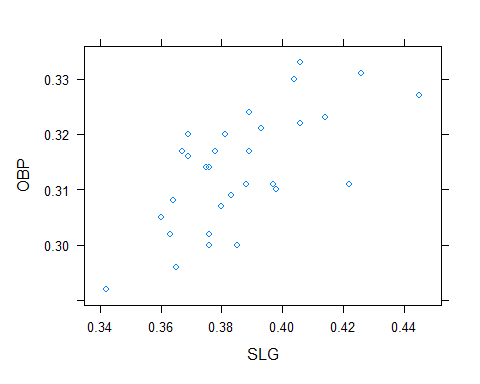
### Figure 4.1 is just a scatter plot  
 plot(case.4.1$SLG, case.4.1$OBP, xlab="SLG", ylab="OBP")



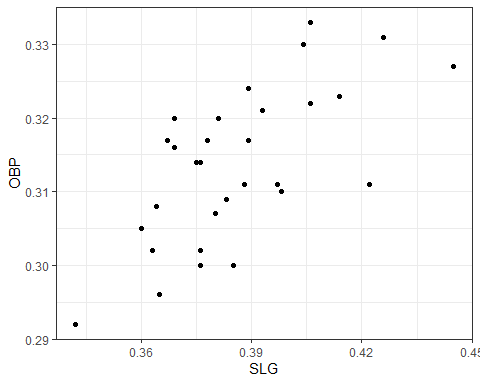
plot(OBP~SLG, data=case.4.1, xlab="SLG", ylab="OBP")



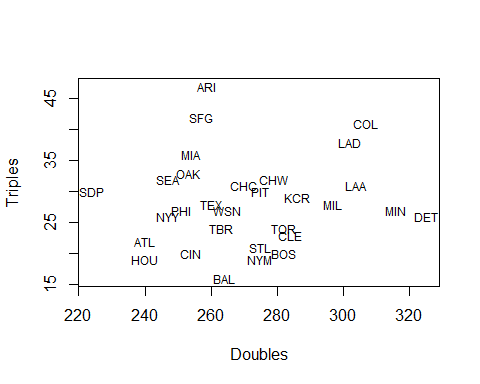
p\_load(lattice)  
 xyplot(OBP~SLG, data=case.4.1, xlab="SLG", ylab="OBP")



p\_load(ggplot2)  
 ggplot(case.4.1, aes(SLG, OBP)) + geom\_point() + xlab("SLG") + ylab("OBP") + theme\_bw()



### Figure 4.2 uses three letter team designators  
 plot(case.4.1$X2B, case.4.1$X3B, ylab="Triples", xlab="Doubles", type="n") ### plot nothing to leave room for the text  
 text(case.4.1$X2B, case.4.1$X3B, case.4.1$Tm, cex=0.75) ### add the text



### Analysis of Table 4.2 data  
 ### The data are somewhat hidden  
 search() ### Look to see which paths are available

## [1] ".GlobalEnv" "package:forcats" "package:purrr"   
## [4] "package:readr" "package:tibble" "package:tidyverse"   
## [7] "package:nortest" "package:tidyr" "package:stringr"   
## [10] "package:rvest" "package:sqldf" "package:RSQLite"   
## [13] "package:gsubfn" "package:proto" "package:retrosheet"  
## [16] "package:devtools" "package:usethis" "package:plotrix"   
## [19] "package:aplpack" "package:lubridate" "package:ggplot2"   
## [22] "package:lattice" "package:Lahman" "package:tsub"   
## [25] "package:dplyr" "package:pacman" "package:stats"   
## [28] "package:graphics" "package:grDevices" "package:utils"   
## [31] "package:datasets" "package:methods" "Autoloads"   
## [34] "package:base"

ls(pos=23) ### On my machine, the tsub package is the 23rd object in the path. We can look at what is in it.

## [1] "AllstarFull" "Appearances" "AwardsManagers"   
## [4] "AwardsPlayers" "AwardsShareManagers" "AwardsSharePlayers"   
## [7] "Batting" "battingLabels" "BattingPost"   
## [10] "battingStats" "CollegePlaying" "Fielding"   
## [13] "fieldingLabels" "FieldingOF" "FieldingOFsplit"   
## [16] "FieldingPost" "HallOfFame" "HomeGames"   
## [19] "Label" "LahmanData" "Managers"   
## [22] "ManagersHalf" "Parks" "People"   
## [25] "Pitching" "pitchingLabels" "PitchingPost"   
## [28] "playerInfo" "Salaries" "Schools"   
## [31] "SeriesPost" "teamInfo" "Teams"   
## [34] "TeamsFranchises" "TeamsHalf"

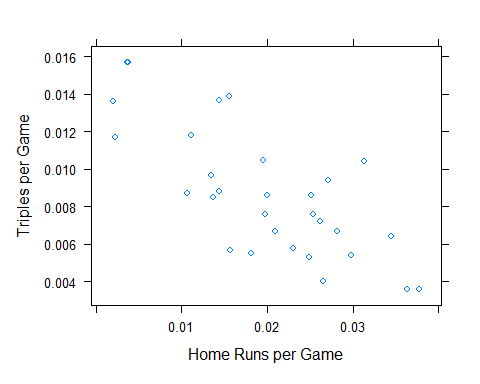
head(case\_4\_1b) ### The case\_4\_1b object contains the proper data.

## teamID yearID HomeRun.Rate Triples.Rate  
## 1 WS1 1916 0.0023 0.0117  
## 2 CHN 1983 0.0254 0.0076  
## 3 ATL 1982 0.0265 0.0040  
## 4 CLE 1983 0.0157 0.0057  
## 5 SDN 2003 0.0231 0.0058  
## 6 SDN 1984 0.0198 0.0076

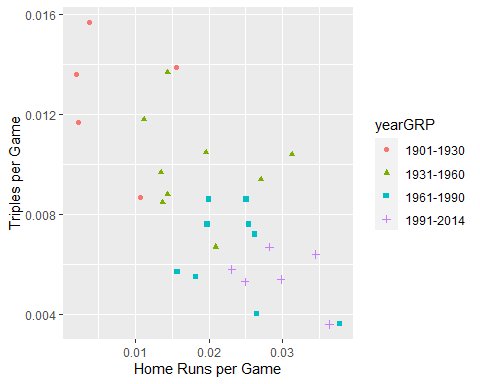
table.4.2 = tsub::case\_4\_1b ### Make a local copy of the case\_4\_1b data  
 names(table.4.2)

## [1] "teamID" "yearID" "HomeRun.Rate" "Triples.Rate"

# It's time to make Figure 4.3  
 p\_load(lattice)  
 xyplot(Triples.Rate~HomeRun.Rate, data=table.4.2, xlab="Home Runs per Game", ylab="Triples per Game")



### Create Figure 4.4  
 # To create the factor variable that contains the year groups we use "cut"  
 table.4.2$yearGRP = cut(table.4.2$yearID,  
 breaks=c(1901,1931,1961,1991,2014),  
 include.lowest=TRUE,  
 labels=c("1901-1930","1931-1960","1961-1990","1991-2014")  
 )  
 p\_load(ggplot2)  
 ggplot(table.4.2, aes(x=HomeRun.Rate, y=Triples.Rate, shape=yearGRP, col=yearGRP)) +   
 geom\_point() + xlab("Home Runs per Game") + ylab("Triples per Game")



### Case 4.2

Albert now turns to looking at offensive statistics that are associated with scoring runs. We will use the data from Albert’s **tsub** package.

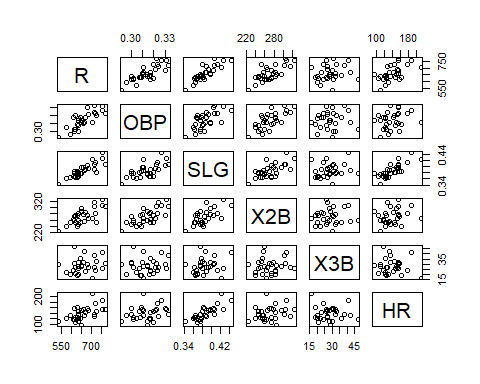
case.4.2 = case\_4\_2 ### Make a local copy  
 head(case.4.2)

## Tm NBat BatAge R.G G PA AB R H X2B X3B HR RBI SB CS BB SO  
## 1 ARI 52 27.6 3.80 162 6089 5552 615 1379 259 47 118 573 86 33 398 1165  
## 2 ATL 39 26.8 3.54 162 6064 5468 573 1316 240 22 123 545 95 33 472 1369  
## 3 BAL 44 28.3 4.35 162 6130 5596 705 1434 264 16 211 681 44 20 401 1285  
## 4 BOS 55 29.2 3.91 162 6226 5551 634 1355 282 20 123 601 63 25 535 1337  
## 5 CHC 48 26.8 3.79 162 6102 5508 614 1315 270 31 157 590 65 40 442 1477  
## 6 CHW 44 27.7 4.07 162 6077 5543 660 1400 279 32 155 625 85 36 417 1362  
## BA OBP SLG OPS OPS. TB GDP HBP SH SF IBB LOB  
## 1 0.248 0.302 0.376 0.678 88 2086 115 43 56 36 31 1092  
## 2 0.241 0.305 0.360 0.665 87 1969 121 43 53 27 31 1128  
## 3 0.256 0.311 0.422 0.734 106 2363 112 62 35 36 29 1088  
## 4 0.244 0.316 0.369 0.684 92 2046 138 68 20 52 36 1168  
## 5 0.239 0.300 0.385 0.684 88 2118 94 54 57 41 29 1069  
## 6 0.253 0.310 0.398 0.708 100 2208 127 60 19 38 33 1071

names(case.4.2)

## [1] "Tm" "NBat" "BatAge" "R.G" "G" "PA" "AB" "R"   
## [9] "H" "X2B" "X3B" "HR" "RBI" "SB" "CS" "BB"   
## [17] "SO" "BA" "OBP" "SLG" "OPS" "OPS." "TB" "GDP"   
## [25] "HBP" "SH" "SF" "IBB" "LOB"

### Make the pairs plot  
 pairs(case.4.2[,c("R","OBP","SLG","X2B","X3B","HR")])



### The correlations presented in Table 4.3 are easily computed  
 case.4.2$TB = case.4.2$AB \* case.4.2$SLG  
 table.4.3 = round(cor(case.4.2[,c("R", "OBP", "SLG", "X2B", "X3B", "HR")]),3) ### The values are rounded to 3 decimal places  
 table.4.3

## R OBP SLG X2B X3B HR  
## R 1.000 0.797 0.857 0.737 0.202 0.578  
## OBP 0.797 1.000 0.668 0.724 0.104 0.260  
## SLG 0.857 0.668 1.000 0.676 0.209 0.788  
## X2B 0.737 0.724 0.676 1.000 0.090 0.238  
## X3B 0.202 0.104 0.209 0.090 1.000 -0.094  
## HR 0.578 0.260 0.788 0.238 -0.094 1.000

### Table 4.4 can be constructed by rearranging the rows of the first column ot Table 4.4  
 as.matrix(table.4.3[c(3,2,4,6,5), 1], ncol=1)

## [,1]  
## SLG 0.857  
## OBP 0.797  
## X2B 0.737  
## HR 0.578  
## X3B 0.202

### Case 4.3

We now look at the most valuable hitting statistics. In this section we get to play with linear regression and prediction. We will use the data from Albert’s package.

### We get the Table 4.4 data from ALbert's tsub package.  
 p\_load(tsub)   
 case.4.3 = case\_4\_3 ### make a local copy of the dataframe  
 names(case.4.3)

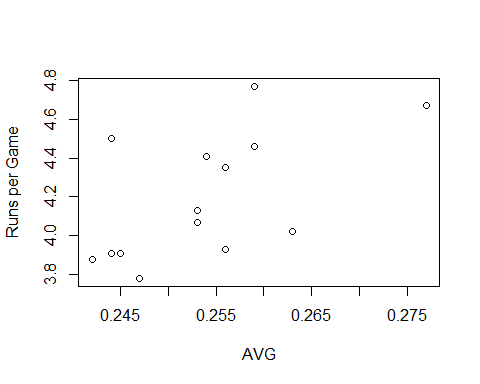
## [1] "teamID" "R" "G" "R.G" "AVG" "OBP" "SLG" "OPS"   
## [9] "RC" "TA" "BRA"

head(case.4.3) ### Table 4.5 similar data

## teamID R G R.G AVG OBP SLG OPS RC TA BRA  
## 1 BAL 705 162 4.35 0.256 0.311 0.422 0.733 723 0.664 0.131  
## 2 BOS 634 162 3.91 0.244 0.316 0.369 0.685 635 0.615 0.117  
## 3 CHA 660 162 4.07 0.253 0.310 0.398 0.708 673 0.634 0.123  
## 4 CLE 669 162 4.13 0.253 0.317 0.389 0.706 683 0.641 0.123  
## 5 DET 757 162 4.67 0.277 0.331 0.426 0.757 790 0.698 0.141  
## 6 HOU 629 162 3.88 0.242 0.309 0.383 0.692 636 0.624 0.118

Figure 4.6 is a plot that is then made into 4.7 by adding a regression line and the residuals.

# Figure 4.6 looks like  
 plot(R.G~AVG, data=case.4.3, ylab="Runs per Game")



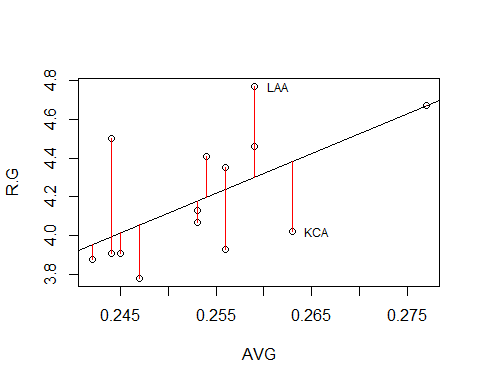
# We compute a linear regression line  
 summary(lm(R.G~AVG, data=case.4.3))

##   
## Call:  
## lm(formula = R.G ~ AVG, data = case.4.3)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.36344 -0.10671 -0.07335 0.13420 0.50569   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.003 1.905 -0.526 0.6075   
## AVG 20.480 7.524 2.722 0.0174 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.264 on 13 degrees of freedom  
## Multiple R-squared: 0.363, Adjusted R-squared: 0.314   
## F-statistic: 7.409 on 1 and 13 DF, p-value: 0.01745

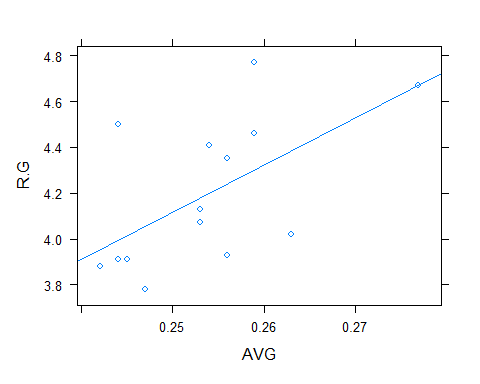
coef(lm(R.G~AVG, data=case.4.3))

## (Intercept) AVG   
## -1.002905 20.480394

# Albert starts looking at goodness of fit. R helps us with this.  
 rpg.lm.avg = lm(R.G~AVG, data=case.4.3) ### Fit and save the regression equation (and some other stuff as well)  
 case.4.3$predrpg = predict(rpg.lm.avg) ### Get the predicted values  
 case.4.3$residrpg = resid(rpg.lm.avg) ### Get the residuals or errors  
   
 # Figure 4.7  
 plot(R.G~AVG, data=case.4.3) ### Plot the points  
 abline(rpg.lm.avg) ### Add the regression line computed above  
 segments(case.4.3$AVG, case.4.3$R.G, case.4.3$AVG, case.4.3$predrpg, col="red") ### Add the vertical errors or residuals  
 text(case.4.3$AVG[c(8,7)]+0.0025, case.4.3$R.G[c(8,7)] , case.4.3$teamID[c(8,7)], cex=0.75) ### Label the interesting points



### Other methods for getting points and the regression line  
 # lattice  
 p\_load(lattice)  
 xyplot(R.G ~ AVG, data=case.4.3, type=c("p","r")) ### Get p=points and r=regression



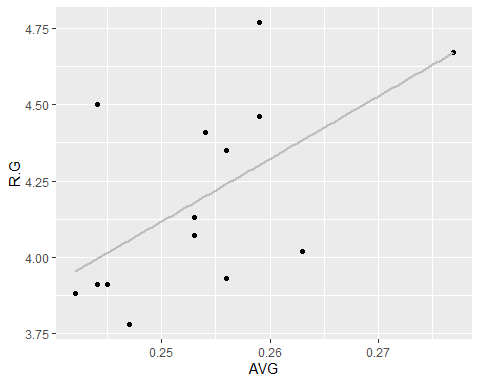
# ggplot2  
 p\_load(ggplot2)  
   
 ggplot(data=case.4.3, aes(x=AVG, y=R.G)) + ### Define the data, set the aesthetics, "+" continues the plot on next line  
 geom\_point() + ### Plot the points, "+" continues the plot on the next line  
 geom\_smooth(method ="lm", se=FALSE, color="maroon") ### Add the regression line without confidence band

## `geom\_smooth()` using formula 'y ~ x'



case.4.3 %>% ### Pipe the data to ggplot using %>%  
 ggplot(aes(x=AVG, y=R.G)) + ### The rest is the same as above except that the data have been piped above  
 geom\_point() +  
 geom\_smooth(method="lm", se=FALSE, color="gray")

## `geom\_smooth()` using formula 'y ~ x'



Mean squared error (MSE) is a measure of the quality of the fit of the regression line. It is the average of the *squared* errors or residuals. The root mean squared error (RMSE) is the almost average error — almost because the square root of a sum is not equal to the summ of the square roots.

# Compute the Mean Squared Error (MSE)  
 case.4.3$sqresidrpg = case.4.3$residrpg^2 ### Square the residuals found above  
 n = nrow(case.4.3) ### Count the number of observations  
 mse = sum(case.4.3$sqresidrpg)/n ### Sum the squared residuals and divide by n. This really should be divide by n-2.  
 mse

## [1] 0.06039262

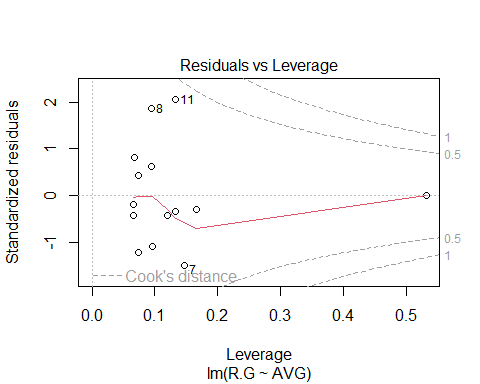
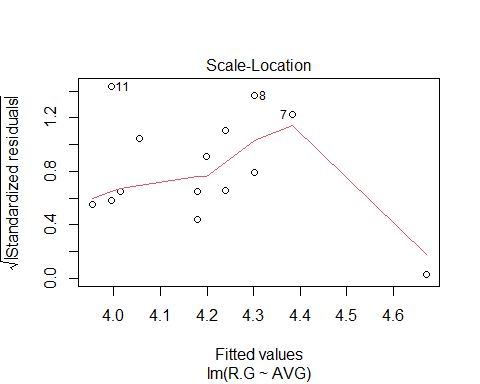
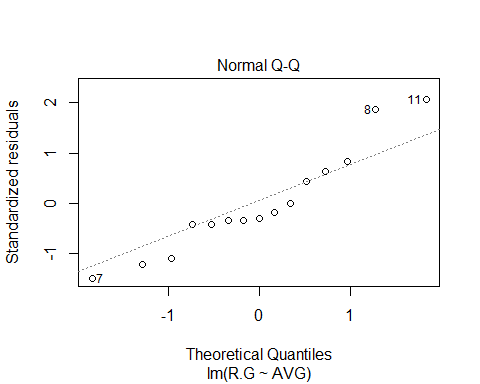
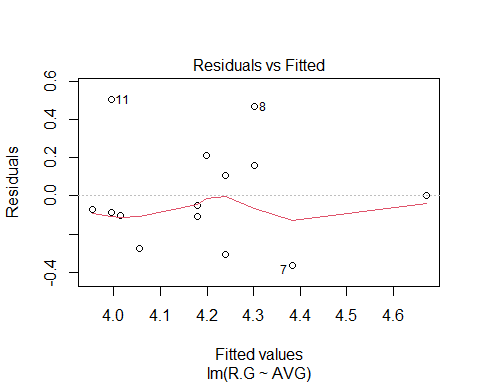
rmse = sqrt(mse)  
 rmse

## [1] 0.2457491

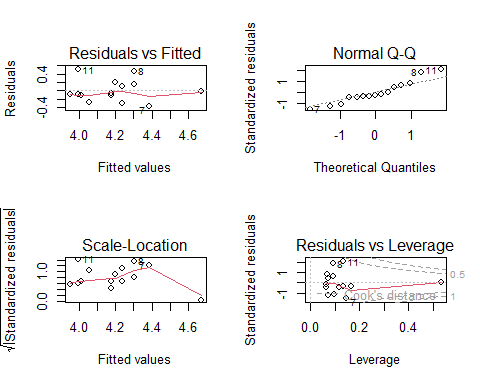
sort(case.4.3$residrpg)

## [1] -0.3634385832 -0.3100758232 -0.2757522747 -0.1086346404 -0.1047914861  
## [6] -0.0843110919 -0.0843110919 -0.0733503033 -0.0486346404 -0.0001641031  
## [11] 0.1099241768 0.1584829939 0.2108849653 0.4684829939 0.5056889081

# Albert suggests that the residuals are normally distributed. R helps us  
 # check this.  
 plot(rpg.lm.avg)



# Or on one page...  
 par(mfrow=c(2,2))  
 plot(rpg.lm.avg)



par(mfrow=c(1,1))  
   
 # Next we try OBP  
 rpg.lm.obp = lm(R.G~OBP, data=case.4.3)  
 coef(rpg.lm.obp)

## (Intercept) OBP   
## -5.155568 29.574132

Tables 4.6 and 4.7 give the values used in computing the RMSE.

### Table 4.7 uses AVG  
 cbind(case.4.3$teamID, round(case.4.3[,c("AVG","R.G")],3), pred=round(predict(rpg.lm.avg),3),   
 res=round(resid(rpg.lm.avg),3), sqrres=round(resid(rpg.lm.avg)^2,3))

## case.4.3$teamID AVG R.G pred res sqrres  
## 1 BAL 0.256 4.35 4.240 0.110 0.012  
## 2 BOS 0.244 3.91 3.994 -0.084 0.007  
## 3 CHA 0.253 4.07 4.179 -0.109 0.012  
## 4 CLE 0.253 4.13 4.179 -0.049 0.002  
## 5 DET 0.277 4.67 4.670 0.000 0.000  
## 6 HOU 0.242 3.88 3.953 -0.073 0.005  
## 7 KCA 0.263 4.02 4.383 -0.363 0.132  
## 8 LAA 0.259 4.77 4.302 0.468 0.219  
## 9 MIN 0.254 4.41 4.199 0.211 0.044  
## 10 NYA 0.245 3.91 4.015 -0.105 0.011  
## 11 OAK 0.244 4.50 3.994 0.506 0.256  
## 12 SEA 0.244 3.91 3.994 -0.084 0.007  
## 13 TBA 0.247 3.78 4.056 -0.276 0.076  
## 14 TEX 0.256 3.93 4.240 -0.310 0.096  
## 15 TOR 0.259 4.46 4.302 0.158 0.025

mse = sum(resid(rpg.lm.avg)^2)/(nrow(case.4.3)-2) ### We divide by n-2 because we used 2 pieces of info to get int and slope  
 rmse = sqrt(mse)  
 c(mse, rmse)

## [1] 0.06968379 0.26397687

anova(rpg.lm.avg)

## Analysis of Variance Table  
##   
## Response: R.G  
## Df Sum Sq Mean Sq F value Pr(>F)   
## AVG 1 0.51631 0.51631 7.4093 0.01745 \*  
## Residuals 13 0.90589 0.06968   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

### Table 4.7 uses OBP  
 cbind(case.4.3$teamID, round(case.4.3[,c("OBP","R.G")],3), pred=round(predict(rpg.lm.obp),3),   
 res=round(resid(rpg.lm.obp),3), sqrres=round(resid(rpg.lm.obp)^2,3))

## case.4.3$teamID OBP R.G pred res sqrres  
## 1 BAL 0.311 4.35 4.042 0.308 0.095  
## 2 BOS 0.316 3.91 4.190 -0.280 0.078  
## 3 CHA 0.310 4.07 4.012 0.058 0.003  
## 4 CLE 0.317 4.13 4.219 -0.089 0.008  
## 5 DET 0.331 4.67 4.633 0.037 0.001  
## 6 HOU 0.309 3.88 3.983 -0.103 0.011  
## 7 KCA 0.314 4.02 4.131 -0.111 0.012  
## 8 LAA 0.322 4.77 4.367 0.403 0.162  
## 9 MIN 0.324 4.41 4.426 -0.016 0.000  
## 10 NYA 0.307 3.91 3.924 -0.014 0.000  
## 11 OAK 0.320 4.50 4.308 0.192 0.037  
## 12 SEA 0.300 3.91 3.717 0.193 0.037  
## 13 TBA 0.317 3.78 4.219 -0.439 0.193  
## 14 TEX 0.314 3.93 4.131 -0.201 0.040  
## 15 TOR 0.323 4.46 4.397 0.063 0.004

mse = sum(resid(rpg.lm.obp)^2)/(nrow(case.4.3)-2) ### We divide by n-2 because we used 2 pieces of info to get int and slope  
 rmse = sqrt(mse)  
 c(mse, rmse)

## [1] 0.05252667 0.22918697

anova(rpg.lm.obp)

## Analysis of Variance Table  
##   
## Response: R.G  
## Df Sum Sq Mean Sq F value Pr(>F)   
## OBP 1 0.73935 0.73935 14.076 0.002419 \*\*  
## Residuals 13 0.68285 0.05253   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

As noted in the comments, the proper formula for mean squared error divides by and not . This is a correction to the amount of information that we have left over after computing the intercept and slope. We call these leftover pieces of information *degrees of freedom.* The correct formula is thus

]

### Case 4.4

We now switch to looking at multiple predictor variables by using multiple linear regression. Albert fits a model using the six batting events (Singles, doubles, triples, home runs, walks, and hit by pitch) to predict runs per game.

### The initial model that Albert fits uses the 2014 data from Case 4.1. This is the same as case\_4\_4  
 MLB2014 = case\_4\_4  
 ### We called this MLB2014  
 head(MLB2014)

## Tm NBat BatAge R.G G PA AB R H X2B X3B HR RBI SB CS BB SO  
## 1 ARI 52 27.6 3.80 162 6089 5552 615 1379 259 47 118 573 86 33 398 1165  
## 2 ATL 39 26.8 3.54 162 6064 5468 573 1316 240 22 123 545 95 33 472 1369  
## 3 BAL 44 28.3 4.35 162 6130 5596 705 1434 264 16 211 681 44 20 401 1285  
## 4 BOS 55 29.2 3.91 162 6226 5551 634 1355 282 20 123 601 63 25 535 1337  
## 5 CHC 48 26.8 3.79 162 6102 5508 614 1315 270 31 157 590 65 40 442 1477  
## 6 CHW 44 27.7 4.07 162 6077 5543 660 1400 279 32 155 625 85 36 417 1362  
## BA OBP SLG OPS OPS. TB GDP HBP SH SF IBB LOB  
## 1 0.248 0.302 0.376 0.678 88 2086 115 43 56 36 31 1092  
## 2 0.241 0.305 0.360 0.665 87 1969 121 43 53 27 31 1128  
## 3 0.256 0.311 0.422 0.734 106 2363 112 62 35 36 29 1088  
## 4 0.244 0.316 0.369 0.684 92 2046 138 68 20 52 36 1168  
## 5 0.239 0.300 0.385 0.684 88 2118 94 54 57 41 29 1069  
## 6 0.253 0.310 0.398 0.708 100 2208 127 60 19 38 33 1071

names(MLB2014)

## [1] "Tm" "NBat" "BatAge" "R.G" "G" "PA" "AB" "R"   
## [9] "H" "X2B" "X3B" "HR" "RBI" "SB" "CS" "BB"   
## [17] "SO" "BA" "OBP" "SLG" "OPS" "OPS." "TB" "GDP"   
## [25] "HBP" "SH" "SF" "IBB" "LOB"

### We can now fit the optimized average  
 MLB2014$X1B = MLB2014$H - (MLB2014$X2B + MLB2014$X3B + MLB2014$HR)  
 rpg.lm.optavg = lm(R.G ~ X1B + X2B + X3B + HR + BB + HBP, data=MLB2014)  
 coef(rpg.lm.optavg)

## (Intercept) X1B X2B X3B HR   
## -1.8352666443 0.0025173114 0.0056001510 0.0076324824 0.0072430677   
## BB HBP   
## 0.0017538445 -0.0009130335

It is likely that the differences that we are seeing here are the result of Albert having rounded his data. The RMSE is

mse = sum(resid(rpg.lm.optavg)^2)/(nrow(MLB2014)-7)  
 rmse = sqrt(mse)  
 c(mse, rmse)

## [1] 0.01730754 0.13155812

anova(rpg.lm.optavg)

## Analysis of Variance Table  
##   
## Response: R.G  
## Df Sum Sq Mean Sq F value Pr(>F)   
## X1B 1 1.15388 1.15388 66.6693 3.016e-08 \*\*\*  
## X2B 1 0.90625 0.90625 52.3614 2.295e-07 \*\*\*  
## X3B 1 0.02087 0.02087 1.2060 0.283496   
## HR 1 0.76495 0.76495 44.1975 8.790e-07 \*\*\*  
## BB 1 0.23774 0.23774 13.7363 0.001163 \*\*   
## HBP 1 0.00278 0.00278 0.1607 0.692214   
## Residuals 23 0.39807 0.01731   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Albert fits other single variable models to fill Table 4.10.

n = nrow(MLB2014)  
 MLB2014$avg = MLB2014$H/MLB2014$AB  
 rpg.lm.avg = lm(R.G ~ avg, data=MLB2014)  
 rmse.avg = sqrt(sum(resid(rpg.lm.avg)^2)/(n-2))  
 MLB2014$tb = 1\*MLB2014$X1B + 2\*MLB2014$X2B + 3\*MLB2014$X3B + 4\*MLB2014$HR  
 MLB2014$slg = MLB2014$tb/MLB2014$AB  
 rpg.lm.slg = lm(R.G ~ slg, data=MLB2014)  
 rmse.slg = sqrt(sum(resid(rpg.lm.slg)^2)/(n-2))  
 MLB2014$obp = (MLB2014$H + MLB2014$BB + MLB2014$HBP)/(MLB2014$AB + MLB2014$BB + MLB2014$HBP + MLB2014$SF)  
 rpg.lm.obp = lm(R.G ~ obp, data=MLB2014)  
 rmse.obp = sqrt(sum(resid(rpg.lm.obp)^2)/(n-2))  
 MLB2014$rc = (MLB2014$H + MLB2014$BB)\*MLB2014$tb/(MLB2014$AB + MLB2014$BB)  
 rpg.lm.rc = lm(R.G ~ rc, data=MLB2014)  
 rmse.rc = sqrt(sum(resid(rpg.lm.rc)^2)/(n-2))  
 MLB2014$ops = MLB2014$obp + MLB2014$slg  
 rpg.lm.ops = lm(R.G ~ ops, data=MLB2014)  
 rmse.ops = sqrt(sum(resid(rpg.lm.ops)^2)/(n-2))  
 rmse.rpg = round(c(rmse.avg, rmse.obp, rmse.slg, rmse.ops, rmse.rc),3)  
 data.frame(Statistic=c("AVG","OBP","SLG","OPS","RC"), rmse.rpg, rmse.n = round(rmse.rpg\*sqrt((n-2)/n),3))

## Statistic rmse.rpg rmse.n  
## 1 AVG 0.212 0.205  
## 2 OBP 0.212 0.205  
## 3 SLG 0.182 0.176  
## 4 OPS 0.150 0.145  
## 5 RC 0.133 0.128

Table 4.11 is easy to create using information from above. Albert takes the coefficients, which he views as weights of the events, and standardizes them by dividing by the “weight” of a single.

t(round(rbind(coef(rpg.lm.optavg)[-1],coef(rpg.lm.optavg)[-1]/coef(rpg.lm.optavg)[2]),5))

## [,1] [,2]  
## X1B 0.00252 1.00000  
## X2B 0.00560 2.22466  
## X3B 0.00763 3.03200  
## HR 0.00724 2.87730  
## BB 0.00175 0.69671  
## HBP -0.00091 -0.36270

### Case 4.5

We now switch to looking at multiple predictor variables by using non-linear (in the coefficients) regression.

### We get the Table 4.11 data from ALbert's tsub package.  
 case.4.5 = case\_4\_5  
 head(case.4.5)

## teamID W L R RA  
## 1 ARI 64 98 615 742  
## 2 ATL 79 83 573 597  
## 3 BAL 96 66 705 593  
## 4 BOS 71 91 634 715  
## 5 CHA 73 89 660 758  
## 6 CHN 73 89 614 707

### Alternatively, we could pull the table from Lahman's data.  
 require(pacman)  
 p\_load(Lahman, sqldf)  
 data(Teams)  
 tbl.4.12 = sqldf("SELECT teamId, W, L, R, RA FROM Teams WHERE (yearID=2014 AND (lgID='AL' OR lgID='NL')) ORDER BY teamID")  
 head(tbl.4.12)

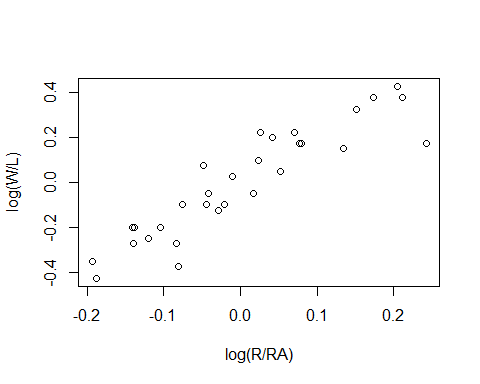
## teamID W L R RA  
## 1 ARI 64 98 615 742  
## 2 ATL 79 83 573 597  
## 3 BAL 96 66 705 593  
## 4 BOS 71 91 634 715  
## 5 CHA 73 89 660 758  
## 6 CHN 73 89 614 707

It’s nice to see that the values are the same. We continue by creating Figure 4.8

### Create Figure 4.8 using Lahman data  
 tbl.4.12$lnwl = log(tbl.4.12$W/tbl.4.12$L)  
 tbl.4.12$lnrra = log(tbl.4.12$R/tbl.4.12$RA)  
 plot(lnwl~lnrra, data=tbl.4.12, xlab="log(R/RA)", ylab="log(W/L)")  
 ### Fit the linearized model through the point (0,0)---i.e. without intercept  
 tbl.4.12.lm = lm(lnwl ~ lnrra - 1, data=tbl.4.12)  
 summary(tbl.4.12.lm)

##   
## Call:  
## lm(formula = lnwl ~ lnrra - 1, data = tbl.4.12)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.26466 -0.05703 0.01246 0.05430 0.17586   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## lnrra 1.8057 0.1509 11.97 9.65e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.09813 on 29 degrees of freedom  
## Multiple R-squared: 0.8316, Adjusted R-squared: 0.8258   
## F-statistic: 143.2 on 1 and 29 DF, p-value: 9.655e-13

### Try again using data from Albert's tsub  
 ### Create Figure 4.8  
 case.4.5$lnwl = log(case.4.5$W/case.4.5$L)  
 case.4.5$lnrra = log(case.4.5$R/case.4.5$RA)  
 plot(lnwl~lnrra, data=case.4.5, xlab="log(R/RA)", ylab="log(W/L)")



# Fit the linearized model through the point (0,0)---i.e. without intercept  
 case.4.5.lm = lm(lnwl ~ lnrra - 1, data=case.4.5)  
 summary(case.4.5.lm)

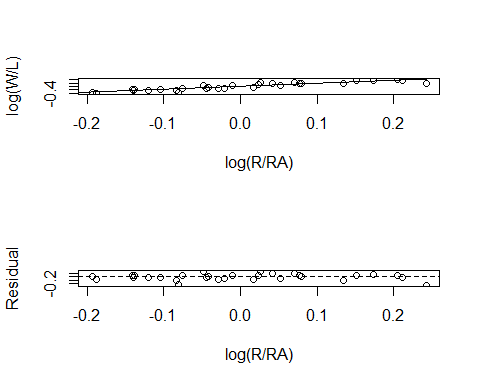
##   
## Call:  
## lm(formula = lnwl ~ lnrra - 1, data = case.4.5)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.26466 -0.05703 0.01246 0.05430 0.17586   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## lnrra 1.8057 0.1509 11.97 9.65e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.09813 on 29 degrees of freedom  
## Multiple R-squared: 0.8316, Adjusted R-squared: 0.8258   
## F-statistic: 143.2 on 1 and 29 DF, p-value: 9.655e-13

coef(tbl.4.12.lm)

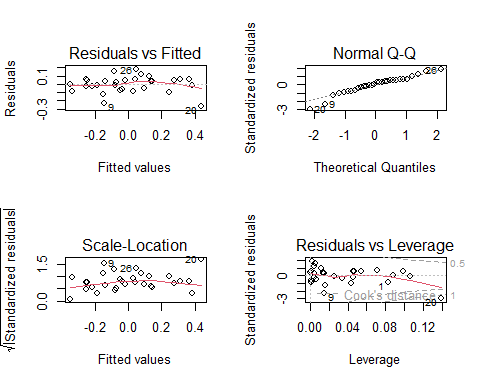
## lnrra   
## 1.805664

As found in the text, the observed coefficient (1.81) is approximately 2.

# Figure 4.9 can be plotted as  
 par(mfrow=c(2,1))  
 plot(lnwl ~ lnrra, data=tbl.4.12, xlab="log(R/RA)", ylab="log(W/L)")  
 abline(tbl.4.12.lm)  
 plot(resid(tbl.4.12.lm)~tbl.4.12$lnrra, xlab="log(R/RA)", ylab="Residual")  
 abline(h=0, lty=2)



par(mfrow=c(1,1))  
 # Note that R has the default diagnostic plots  
 par(mfrow=c(2,2))  
 plot(tbl.4.12.lm)



par(mfrow=c(1,1))

Table 4.13 is generated using R functions “pred” and “resid”.

tbl.4.12$predw = tbl.4.12$L\*(tbl.4.12$R/tbl.4.12$RA)^coef(tbl.4.12.lm) ###coef is 1.805  
 tbl.4.12$residw = tbl.4.12$W - tbl.4.12$predw  
 tbl.4.13 = cbind(tbl.4.12[,c("teamID","W","L")], Predicted=round(tbl.4.12$predw,1), Residual=round(tbl.4.12$residw,3))  
 print(tbl.4.13)

## teamID W L Predicted Residual  
## 1 ARI 64 98 69.8 -5.825  
## 2 ATL 79 83 77.1 1.927  
## 3 BAL 96 66 90.2 5.799  
## 4 BOS 71 91 73.2 -2.241  
## 5 CHA 73 89 69.3 3.686  
## 6 CHN 73 89 69.0 4.009  
## 7 CIN 76 86 81.7 -5.735  
## 8 CLE 85 77 80.4 4.560  
## 9 COL 66 96 83.1 -17.066  
## 10 DET 90 72 81.9 8.127  
## 11 HOU 70 92 71.5 -1.543  
## 12 KCA 89 73 78.8 10.197  
## 13 LAA 98 64 92.6 5.404  
## 14 LAN 94 68 89.4 4.589  
## 15 MIA 77 85 78.5 -1.511  
## 16 MIL 82 80 78.5 3.532  
## 17 MIN 70 92 79.2 -9.173  
## 18 NYA 84 78 71.5 12.451  
## 19 NYN 79 83 85.7 -6.687  
## 20 OAK 88 74 114.7 -26.663  
## 21 PHI 73 89 73.7 -0.732  
## 22 PIT 88 74 85.1 2.851  
## 23 SDN 77 85 74.2 2.843  
## 24 SEA 87 75 95.7 -8.683  
## 25 SFN 88 74 85.5 2.532  
## 26 SLN 90 72 75.5 14.514  
## 27 TBA 77 85 81.8 -4.834  
## 28 TEX 67 95 67.0 0.015  
## 29 TOR 83 79 86.9 -3.860  
## 30 WAS 96 66 96.8 -0.766

These values do not match Albert’s. We can try using the Pythagorean approach of .

tbl.4.12$predw = tbl.4.12$L\*(tbl.4.12$R/tbl.4.12$RA)^2 ### coef is 2  
 tbl.4.12$residw = tbl.4.12$W - tbl.4.12$predw  
 cbind(tbl.4.12[,c("teamID","W","L")], Predicted=round(tbl.4.12$predw,1), Pythag=round(tbl.4.12$residw,3))

## teamID W L Predicted Pythag  
## 1 ARI 64 98 67.3 -3.324  
## 2 ATL 79 83 76.5 2.539  
## 3 BAL 96 66 93.3 2.715  
## 4 BOS 71 91 71.5 -0.550  
## 5 CHA 73 89 67.5 5.526  
## 6 CHN 73 89 67.1 5.874  
## 7 CIN 76 86 81.3 -5.289  
## 8 CLE 85 77 80.8 4.180  
## 9 COL 66 96 81.8 -15.782  
## 10 DET 90 72 83.0 6.987  
## 11 HOU 70 92 69.6 0.367  
## 12 KCA 89 73 79.5 9.546  
## 13 LAA 98 64 96.4 1.649  
## 14 LAN 94 68 92.1 1.915  
## 15 MIA 77 85 77.8 -0.843  
## 16 MIL 82 80 78.3 3.696  
## 17 MIN 70 92 77.9 -7.904  
## 18 NYA 84 78 70.9 13.113  
## 19 NYN 79 83 86.0 -6.981  
## 20 OAK 88 74 120.2 -32.197  
## 21 PHI 73 89 72.3 0.747  
## 22 PIT 88 74 86.4 1.555  
## 23 SDN 77 85 73.1 3.924  
## 24 SEA 87 75 98.2 -11.225  
## 25 SFN 88 74 86.8 1.196  
## 26 SLN 90 72 75.9 14.128  
## 27 TBA 77 85 81.5 -4.501  
## 28 TEX 67 95 64.5 2.488  
## 29 TOR 83 79 87.8 -4.752  
## 30 WAS 96 66 100.8 -4.834

Still not right. Let’s try figuring out what Albert used for .

### Find k using our estimates   
 log(95.8/64)/log(773/630) # Angels

## [1] 1.971945

log(92.0/68)/log(718/617) # Dodgers

## [1] 1.99393

# Both should have been 1.81 which is the "right" answer on average.  
   
 ### Try the same for our predicted values.  
 log(96.4/64)/log(773/630) # Angels

## [1] 2.002467

log(92.1/68)/log(718/617) # Dodgers

## [1] 2.001096

# Both appear to be 2 and not 1.81. Maybe he used the Pythagorean approach.

It looks like Albert used instead of in his estimates and residuals.

Based on our results, Albert’s contention that we hit 80% prediction within 4 runs is a joke. It’s more like 13/30 or 43%.

p\_load(aplpack)  
 stem.leaf(abs(tbl.4.13$Residual))

## 1 | 2: represents 12  
## leaf unit: 1  
## n: 30  
## 6 0\* | 000111  
## 13 t | 2222333  
## (8) f | 44445555  
## 9 s | 6  
## 8 0. | 889  
## 5 1\* | 0  
## 4 t | 2  
## 3 f | 4  
## 2 s | 7  
## HI: 26.663

tbl.4.13 %>%   
 mutate(abs.resid = abs(Residual)) %>%  
 filter(abs.resid <= 4) %>%  
 summarize(n = n())

## n  
## 1 13

table(abs(tbl.4.13$Residual) <= 4)/30\*100

##   
## FALSE TRUE   
## 56.66667 43.33333

quantile(abs(tbl.4.13$Residual))

## 0% 25% 50% 75% 100%   
## 0.01500 2.60975 4.57450 7.76700 26.66300

### Case 4.6

Albert now brings up the concept of regression to the mean — those things that were big will get smaller and those that were small will get bigger. He focuses on slugging (***SLG***), but could have used any baseball stat.

### We get the Table 4.14 data from ALbert's tsub package.  
 case.4.6 = case\_4\_6 ### Make a local copy  
 head(case.4.6)

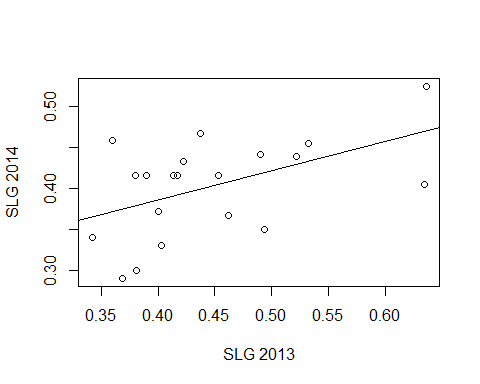
## Name SLG.2013 SLG.2014 Improvement  
## 1 Domonic\_Brown 0.494 0.349 -0.145  
## 2 Melky\_Cabrera 0.360 0.458 0.098  
## 3 Miguel\_Cabrera 0.636 0.524 -0.112  
## 4 Alberto\_Callaspo 0.369 0.290 -0.079  
## 5 Zack\_Cozart 0.381 0.300 -0.081  
## 6 Chris\_Davis 0.634 0.404 -0.230

names(case.4.6)

## [1] "Name" "SLG.2013" "SLG.2014" "Improvement"

Figure 4.11 and the regression line are computed using the methods from above.

plot(SLG.2014 ~ SLG.2013, data=case.4.6, xlab="SLG 2013", ylab="SLG 2014")  
 case.4.6.lm1 = lm(SLG.2014 ~ SLG.2013, case.4.6)  
 abline(case.4.6.lm1)



coef(case.4.6.lm1)

## (Intercept) SLG.2013   
## 0.2421790 0.3580688

summary(case.4.6.lm1)

##   
## Call:  
## lm(formula = SLG.2014 ~ SLG.2013, data = case.4.6)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.08431 -0.04457 0.01647 0.03507 0.08692   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.24218 0.06577 3.682 0.0017 \*\*  
## SLG.2013 0.35807 0.14480 2.473 0.0236 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.05251 on 18 degrees of freedom  
## Multiple R-squared: 0.2536, Adjusted R-squared: 0.2121   
## F-statistic: 6.115 on 1 and 18 DF, p-value: 0.0236

# Compute Davis's predicted 2014 value  
 c(1,.634)%\*%coef(case.4.6.lm1)

## [,1]  
## [1,] 0.4691947

mean(case.4.6$SLG.2013) ### Albert says 0.469 not 0.4469

## [1] 0.4469

c(1,.634)%\*%coef(case.4.6.lm1)-mean(case.4.6$SLG.2014) ### Albert got this one right at .067

## [,1]  
## [1,] 0.06699468

Now look at improvement in Figure 4.11 and the regression model.

plot(Improvement ~ SLG.2013, data=case.4.6, xlab="SLG 2013", ylab="Improvement in SLG")  
 case.4.6.lm2 = lm(Improvement ~ SLG.2013, case.4.6)  
 abline(case.4.6.lm2)



cor(case.4.6[,-1]) ### Remove the Team column

## SLG.2013 SLG.2014 Improvement  
## SLG.2013 1.0000000 0.5035738 -0.7224750  
## SLG.2014 0.5035738 1.0000000 0.2335145  
## Improvement -0.7224750 0.2335145 1.0000000

coef(case.4.6.lm2)

## (Intercept) SLG.2013   
## 0.2421790 -0.6419312

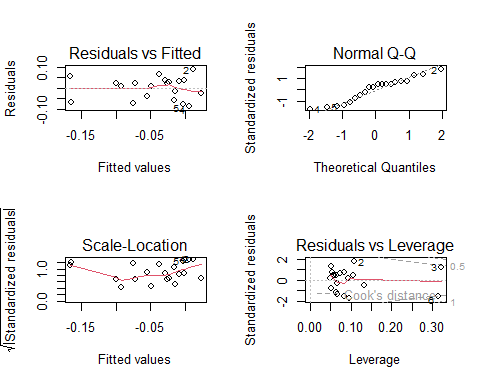
summary(case.4.6.lm2)

##   
## Call:  
## lm(formula = Improvement ~ SLG.2013, data = case.4.6)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.08431 -0.04457 0.01647 0.03507 0.08692   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.24218 0.06577 3.682 0.001704 \*\*   
## SLG.2013 -0.64193 0.14480 -4.433 0.000321 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.05251 on 18 degrees of freedom  
## Multiple R-squared: 0.522, Adjusted R-squared: 0.4954   
## F-statistic: 19.65 on 1 and 18 DF, p-value: 0.0003209

### Albert doesn't look at residuals, but you ought to.  
 par(mfrow=c(2,2))  
 plot(case.4.6.lm1)



plot(case.4.6.lm2)



par(mfrow=c(1,1))

## Note

The above code is intended to show one or two ways of computing some of the statistics that might be used in looking at sports related data. In a few cases, our results differed from Albert’s. While this is unfortunate, the process of identifying possible reasons for these differences is very informative. In the real world it is important to verify results. Having experience in debugging analyses is a good thing. While I am pretty sure that Albert did not intend to include the errors that we found, we should be glad that we had the opportunity to find them.