Excercises in Barlow, Chapter 2

Ed Cashin

2016-03-02

This document is rendered in R using the *Rmarkdown* package.

```
> rmarkdown::render('two.Rmd')
```

Exercise 2.1

First, type in the data as x.

```
x <- c(19, 18.7, 19.3, 19.2, 18.9, 19, 20.2, 19.9, 18.6, 19.4, 19.3, 18.8, 19.3, 19.2, 18.7, 18.5, 18.6
```

Calculate the mean manually (somewhat manually), and verify it using the base R function.

```
sum(x)
```

```
## [1] 481.6
```

```
length(x)
```

```
## [1] 25
```

```
sum(x) / length(x)
```

```
## [1] 19.264
```

```
mean(x)
```

```
## [1] 19.264
```

Now calculate the standard deviation and verify with R base *sd*, which uses Barlow's equation 2.11, with $N-1$ on the denominator.

```
(x - mean(x))^2
```

```
##  [1] 0.069696 0.318096 0.001296 0.004096 0.132496 0.069696 0.876096
##  [8] 0.404496 0.440896 0.018496 0.001296 0.215296 0.001296 0.004096
## [15] 0.318096 0.583696 0.440896 0.190096 0.404496 0.541696 0.055696
## [22] 0.018496 0.112896 0.541696 0.132496
```

```
sum((x - mean(x))^2)
```

```
## [1] 5.8976
```

```r
sqrt(sum((x - mean(x))^2)/(length(x)-1))
```

```
## [1] 0.495715
```

```r
sd(x)
```

```
## [1] 0.495715
```

Exercise 2.2

Now add in the instructor's age of 37 to see how that affects the mean and standard deviation.

```r
mean(c(x, 37))
```

```
## [1] 19.94615
```

```r
sd(c(x, 37))
```

```
## [1] 3.512063
```

Exercise 2.3

Below I create a function for the skew.

```r
skew <- function(x) { sum((x - mean(x))^3)/(length(x)*sd(x)^3) }
skew(x)
```

```
## [1] 0.2124908
```

```r
skew(c(x, 37))
```

```
## [1] 4.386403
```

For verifying that, I searched for calculation of skew in R and found a blog about using the "moments" package.

http://www.r-bloggers.com/measures-of-skewness-and-kurtosis/

```r
library(moments)
skewness(x)
```

```
## [1] 0.2259089
```

```r
skew(x)
```

```
## [1] 0.2124908
```

They're a bit different. Subtracting one in the denominator isn't quite enough to make them the same.

```r
skew2 <- function(x) { sum((x - mean(x))^3)/((length(x)-1)*sd(x)^3) }
skew2(x)
```

```
## [1] 0.2213446
```

Looking at the code (shown below), I see that the *moments::skewness* doesn't subtract one from the denominator in the standard deviation calculation.

```r
skewness
```

```
## function (x, na.rm = FALSE)
## {
##     if (is.matrix(x))
##         apply(x, 2, skewness, na.rm = na.rm)
##     else if (is.vector(x)) {
##         if (na.rm)
##             x <- x[!is.na(x)]
##         n <- length(x)
##         (sum((x - mean(x))^3)/n)/(sum((x - mean(x))^2)/n)^(3/2)
##     }
##     else if (is.data.frame(x))
##         sapply(x, skewness, na.rm = na.rm)
##     else skewness(as.vector(x), na.rm = na.rm)
## }
## <environment: namespace:moments>
```

Exercise 2.4

The following contents are in file `grades.dat`.

```
classical quantum
22    63
48    39
76    61
10    30
22    51
4     44
68    74
44    78
10    55
76    58
14    41
56    69
```

The grades are loaded.

```r
df <- data.frame(read.table("grades.dat", header=T))
```

The *dplyr* library can do aggregations, and I want to get the hang of it. I also want to make that data "tidy", so I install *tidyr* and use it.

```r
library(dplyr)
library(tidyr)
```

I want the "classical" or "quantum" course specification to be a categorical column in the data, and *gather* from *tidyr* does that.

```r
df %>% gather(course, grade, classical:quantum)
```

```
##       course grade
## 1  classical    22
## 2  classical    48
## 3  classical    76
## 4  classical    10
## 5  classical    22
## 6  classical     4
## 7  classical    68
## 8  classical    44
## 9  classical    10
## 10 classical    76
## 11 classical    14
## 12 classical    56
## 13   quantum    63
## 14   quantum    39
## 15   quantum    61
## 16   quantum    30
## 17   quantum    51
## 18   quantum    44
## 19   quantum    74
## 20   quantum    78
## 21   quantum    55
## 22   quantum    58
## 23   quantum    41
## 24   quantum    69
```

Then I can group by course and do summary statistics using *dplyr::summarize*.

```r
df %>% gather(course, grade, classical:quantum) %>%
  group_by(course) %>% summarize(mean=mean(grade))
```

```
## Source: local data frame [2 x 2]
##
##      course  mean
##       (chr) (dbl)
## 1 classical 37.50
## 2   quantum 55.25
```

But I cannot think of a fancy way to do the covariance. Base R's *cov* function works on matrices and vectors.

```r
x <- df$classical
y <- df$quantum
cov(x, y)
```

```
## [1] 226.3182
```

By hand, Barlow's equation 2.19c is implemented below.

```
mean(x*y) - (mean(x)*mean(y))
```

```
## [1] 207.4583
```

It's different from the results of `cov`, probably because of the `method` parameter used by default.

```
cov(x, y, method="kendall")
```

```
## [1] 46
```

```
cov(x, y, method="pearson")
```

```
## [1] 226.3182
```

```
cov(x, y, method="spearman")
```

```
## [1] 6.909091
```

I guess not.

Here's the correlation by Barlow's 2.20b, checked with the base R correlation function.

```
(mean(x*y) - (mean(x)*mean(y))) / (sd(x)*sd(y))
```

```
## [1] 0.5180672
```

```
cor(x, y)
```

```
## [1] 0.5651642
```

Exercise 2.6

I tried an old-fashioned stem and leaf plot to get a quick feel for the way the histogram will look.

```
(eighty <- read.table("80.dat"))
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10
## 1 90 90 79 84 78 91 88 90 85  80
## 2 88 75 73 79 78 79 67 83 68  60
## 3 73 79 69 74 76 68 72 72 75  60
## 4 61 66 66 54 71 67 75 49 51  57
## 5 62 64 68 58 56 79 63 68 64  51
## 6 58 53 65 57 59 65 48 54 55  40
## 7 49 42 36 46 40 37 53 48 44  43
## 8 35 39 30 41 41 22 28 36 39  51
```

```r
(eighty <- c(t(eighty)))
```

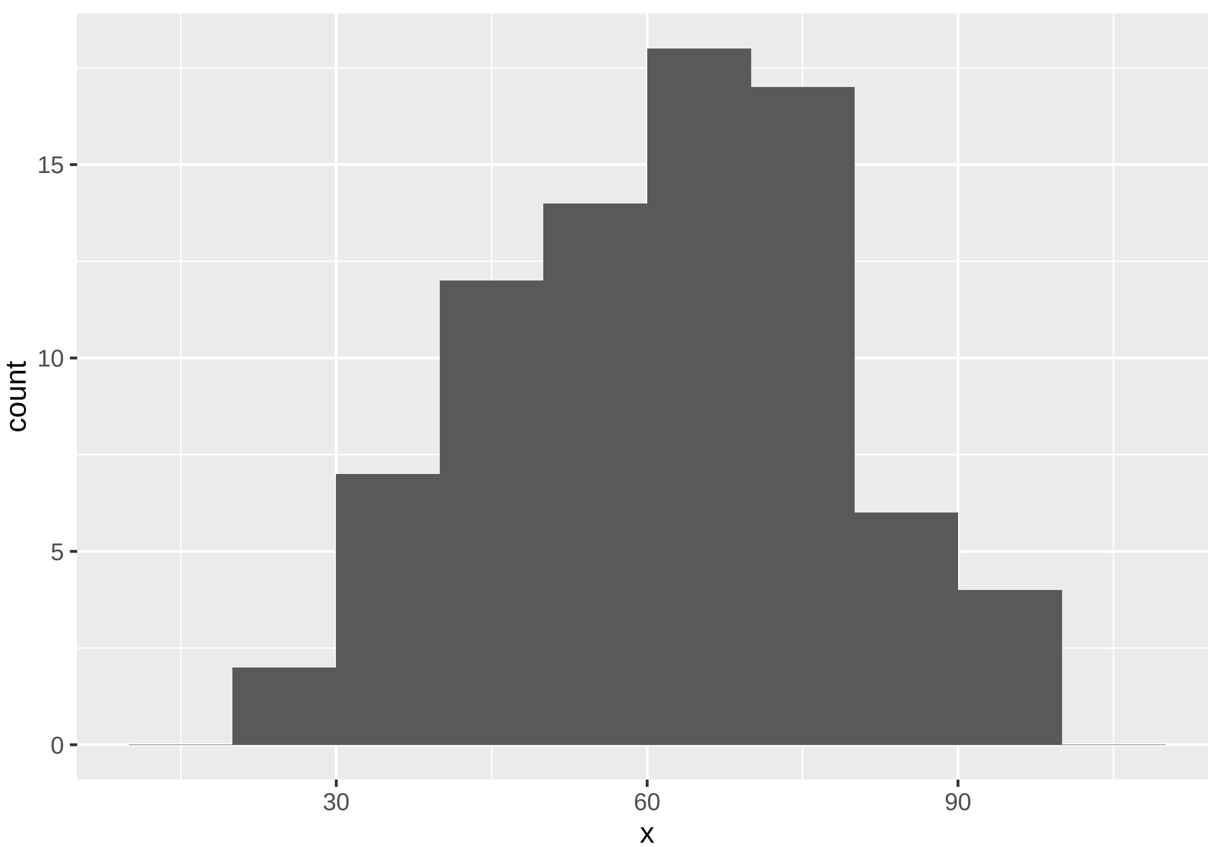
```
##  [1] 90 90 79 84 78 91 88 90 85 80 88 75 73 79 78 79 67 83 68 60 73 79 69
## [24] 74 76 68 72 72 75 60 61 66 66 54 71 67 75 49 51 57 62 64 68 58 56 79
## [47] 63 68 64 51 58 53 65 57 59 65 48 54 55 40 49 42 36 46 40 37 53 48 44
## [70] 43 35 39 30 41 41 22 28 36 39 51
```

```r
stem(eighty)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   2 | 28
##   3 | 0566799
##   4 | 001123468899
##   5 | 11133445677889
##   6 | 001234455667788889
##   7 | 12233455568899999
##   8 | 034588
##   9 | 0001
```

Use histogram in ggplot now.

```r
library(ggplot2)
qplot(x, data=data.frame(x=eighty), binwidth=10)
```



Exercise 2.7

```r
mean(eighty)
```

```
## [1] 61.5875
```

```r
median(eighty)
```

```
## [1] 63.5
```

Using a for loop can find the most popular value. First, create a vector of the distinct values and a corresponding vector to store the counts for each distinct value.

```r
(vals <- sort(unique(eighty)))
```

```
##  [1] 22 28 30 35 36 37 39 40 41 42 43 44 46 48 49 51 53 54 55 56 57 58 59
## [24] 60 61 62 63 64 65 66 67 68 69 71 72 73 74 75 76 78 79 80 83 84 85 88
## [47] 90 91
```

```r
counts <- rep(0, length(vals))
```

Then loop and count, using the boolean indexing of R. The mode is seventy-nine.

```r
for (i in eighty) { counts[vals == i] = counts[vals == i] + 1 }
counts
```

```
##  [1] 1 1 1 1 2 1 2 2 2 1 1 1 1 2 2 3 2 2 1 1 2 2 1 2 1 1 1 2 2 2 2 4 1 1 2
## [36] 2 1 3 1 2 5 1 1 1 1 2 3 1
```

```r
max(counts)
```

```
## [1] 5
```

```r
vals[counts == max(counts)]
```

```
## [1] 79
```

Exercise 2.8

Standard deviation by hand and with base R function follows.

```r
sqrt(sum((eighty - mean(eighty))^2) / length(eighty))
```

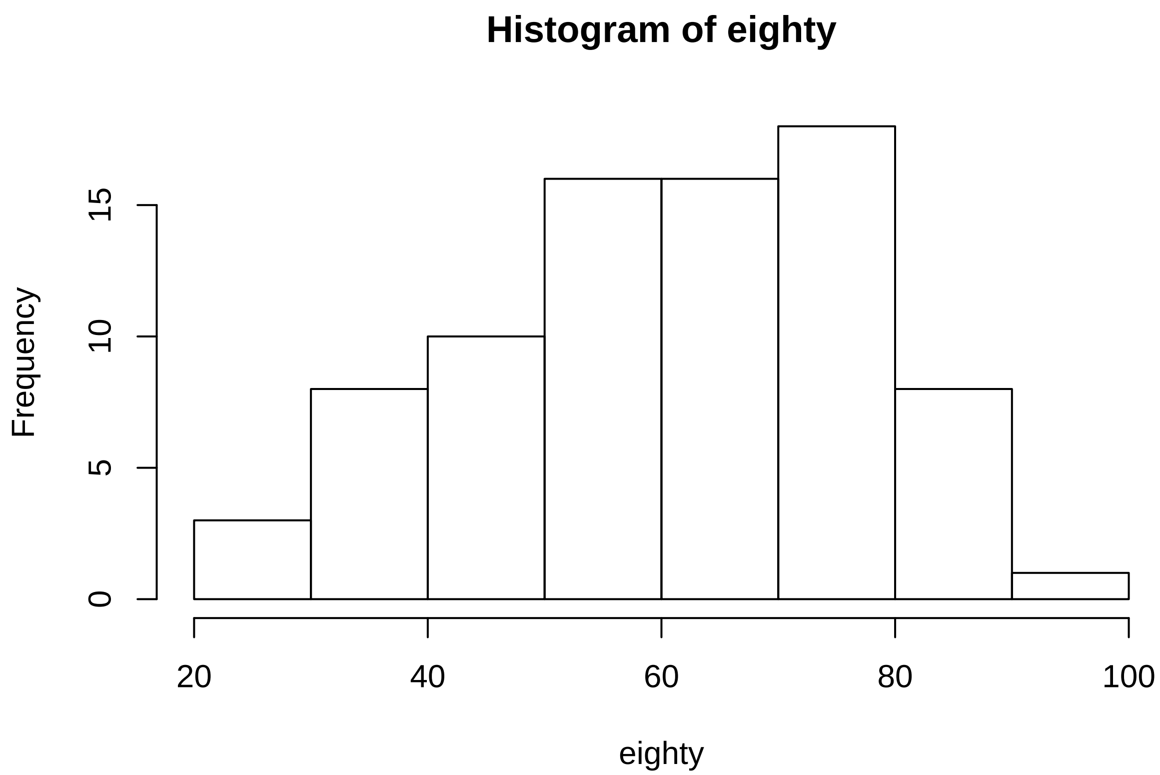
```
## [1] 16.67385
```

```r
sd(eighty)
```

```
## [1] 16.77905
```

For the FWHM, the maximum bin in a histogram is the largest count. The base R *hist* function returns an object with the counts.

```r
hist(eighty)$counts
```




```
## [1]  3  8 10 16 16 18  8  1
```

There are no bins with a height that is exactly half that of the maximum bin, namely nine. From the plot, though, it looks like the histogram is spanning an interval of **forty** horizontal units when measured nine vertical units above the base.

Equation 2.12, for times when a gaussian distribution is an appropriate assumption, would have estimated a FWHM as shown below.

```r
2.35 * sd(eighty)
```

```
## [1] 39.43077
```

So yes, the estimate conforms to the hand-measured FWHM.