Introduction to the Practice of Statistics using R: Chapter 3

Nicholas J. Horton* Ben Baumer

March 10, 2013

Contents

Introduction

This document is intended to help describe how to undertake analyses introduced as examples in the Sixth Edition of *Introduction to the Practice of Statistics* (2009) by David Moore, George McCabe and Bruce Craig. More information about the book can be found at `http://bcs.whfreeman.com/ips6e/`. This file as well as the associated `knitr` reproducible analysis source file can be found at `http://www.math.smith.edu/~nhorton/ips6e`.

This work leverages initiatives undertaken by Project MOSAIC (`http://www.mosaic-web.org`), an NSF-funded effort to improve the teaching of statistics, calculus, science and computing in the undergraduate curriculum. In particular, we utilize the `mosaic` package, which was written to simplify the use of R for introductory statistics courses. A short summary of the R needed to teach introductory statistics can be found in the mosaic package vignette (`http://cran.r-project.org/web/packages/mosaic/vignettes/MinimalR.pdf`).

To use a package within R, it must be installed (one time), and loaded (each session). The package can be installed using the following command:

```
> install.packages('mosaic')                # note the quotation marks
```

*Department of Mathematics and Statistics, Smith College, nhorton@smith.edu

The # character is a comment in R, and all text after that on the current line is ignored.
Once the package is installed (one time only), it can be loaded by running the command:

```
> require(mosaic)
```

This needs to be done once per session.
We also set some options to improve legibility of graphs and output.

```
> trellis.par.set(theme=col.mosaic())  # get a better color scheme for lattice
> options(digits=3)
```

The specific goal of this document is to demonstrate how to replicate the analysis described in Chapter 3: Producing Data.

1 Design of experiments

1.1 Randomizing subjects

It's straightforward to randomly divide 40 students into two groups of 20 students each (as described in Example 3.11 on page 185).

```
> students = 1:40    # equivalent to seq(from=1, to=40, by=1)
> group1 = sample(students, size=20)
> sort(group1)

 [1]  1  2  3  6  8 12 14 15 16 18 19 21 23 29 30 31 33 34 36 38

> group2 = students[-group1]    # all but those values are included
> sort(group2)

 [1]  4  5  7  9 10 11 13 17 20 22 24 25 26 27 28 32 35 37 39 40
```

2 Sampling design

2.1 Simple random samples

We reproduce a random sampling of resorts (from Figure 3.8, page 202).

```
> resorts = c("Aloha Kai", "Captiva", "Palm Tree", "Sea Shell", "Anchor Down",
   "Casa del Mar", "Radisson", "Silver Beach")
> # generate a SRS of size 3
> sampled = sample(resorts, size=3)
> sampled

[1] "Silver Beach" "Anchor Down"  "Sea Shell"
```

3 Toward statistical inference

3.1 Simulate a random sample

It's straightforward to use R to generate simple random samples. Example 3.32 (page 214) describes how this is done by using a table of random digits. It's more generalizable to do this with a set of possible options each with specified probabilities (probability frustrated=0.6, probability not-frustrated=0.4):

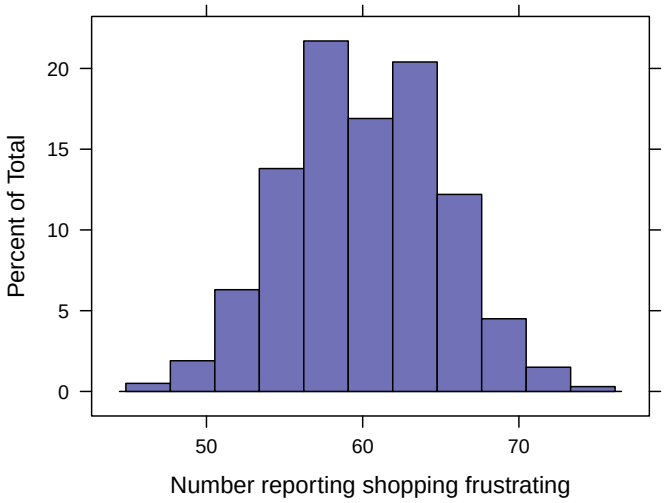
```
> srs1 = sample(c("Frustrating", "Not-frustrating"), size=100, prob=c(0.6, 0.4),
   replace=TRUE)
> tally(srs1)


    Frustrating Not-frustrating           Total
             62              38             100
```

We can repeat the process, which will (generally) give different answers.

```
> n = 100
> n

[1] 100

> tally(sample(c("Frustrating", "Not-frustrating"), size=n, prob=c(0.6, 0.4),
   replace=TRUE))


    Frustrating Not-frustrating           Total
             62              38             100

> tally(sample(c("Frustrating", "Not-frustrating"), size=n, prob=c(0.6, 0.4),
   replace=TRUE))


    Frustrating Not-frustrating           Total
             61              39             100

> tally(sample(c("Frustrating", "Not-frustrating"), size=n, prob=c(0.6, 0.4),
   replace=TRUE))


    Frustrating Not-frustrating           Total
             63              37             100
```

We can repeat the process many times using the `do()` function, which saves the results.

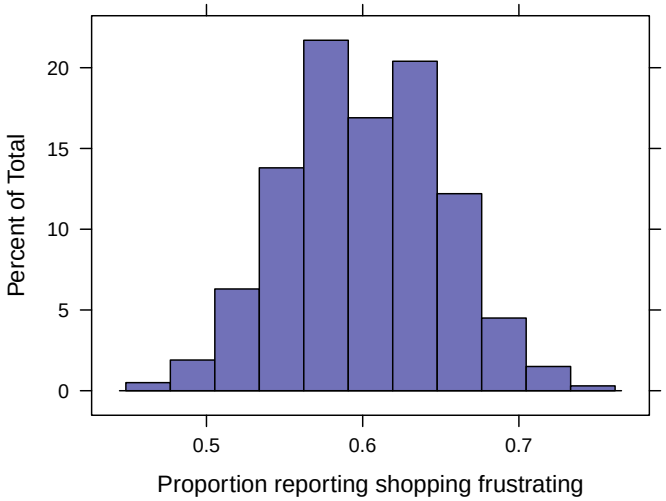
```
> res = do(1000) * tally(sample(c("Frustrating", "Not-frustrating"), size=n,
   prob=c(0.6, 0.4), replace=TRUE))
> histogram(~ Frustrating, xlab="Number reporting shopping frustrating", data=res)
```



We see that the sampling distribution for the number reporting *Frustrating* in m=1000 simple random samples each of size n=100 is centered at the value of around 60, which we would expect since the true probability of being Frustrating is in fact $p = 0.60$.

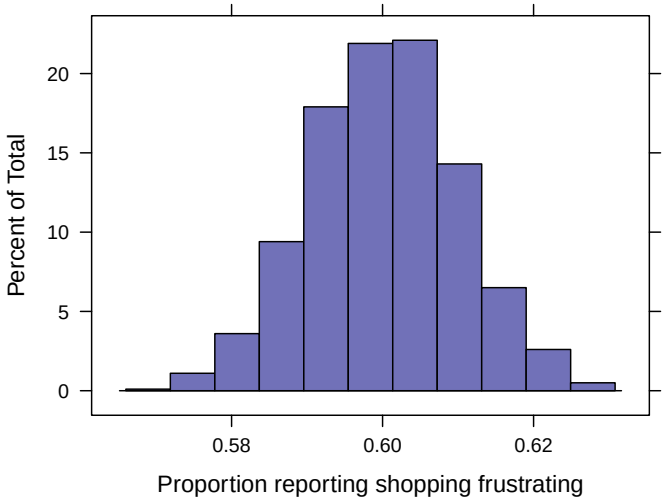
The results are equivalent if rescaled as a proportion (by dividing by the sample size).

```
> sd(~ Frustrating/n, data=res)

[1] 0.0493

> histogram(~ Frustrating/n, xlab="Proportion reporting shopping frustrating", data=res)
```



What happens if we take samples of size n=2500 (as displayed in Example 3.33, on pages 214–215).

```
> n=2500
> n

[1] 2500

> res = do(1000) * tally(sample(c("Frustrating", "Not-frustrating"), size=n,
   prob=c(0.6, 0.4), replace=TRUE))
> sd(~ Frustrating/n, data=res)

[1] 0.00985

> histogram(~ Frustrating/n, xlab="Proportion reporting shopping frustrating",
   data=res)
```



The sampling distribution is much narrower, given the much larger sample size.

3.2 Capture-recapture sampling

R can be used as a calculator, as for the calculations in Example 3.34 (page 220).

```
> 200*120/12

[1] 2000
```