

IS5 in R: Sampling Distribution Models and Confidence Intervals for Proportions (Chapter 13)

Nicholas Horton (nhorton@amherst.edu) and Margaret Chien

October 28, 2018

Introduction and background

This document is intended to help describe how to undertake analyses introduced as examples in the Fifth Edition of *Intro Stats* (2018) by De Veaux, Velleman, and Bock. More information about the book can be found at http://wps.aw.com/aw_deveaux_stats_series. This file as well as the associated R Markdown reproducible analysis source file used to create it can be found at <http://nhorton.people.amherst.edu/is5>.

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 vignettes (<http://cran.r-project.org/web/packages/mosaic>). A paper describing the `mosaic` approach was published in the *R Journal*: <https://journal.r-project.org/archive/2017/RJ-2017-024>.

Chapter 13: Sampling Distribution Models and Confidence Intervals for Proportions

```
library(mosaic)
library(readr)
library(janitor)
Babies <- read_csv("http://nhorton.people.amherst.edu/is5/data/Babysamp_98.csv") %>%
  clean_names()
```

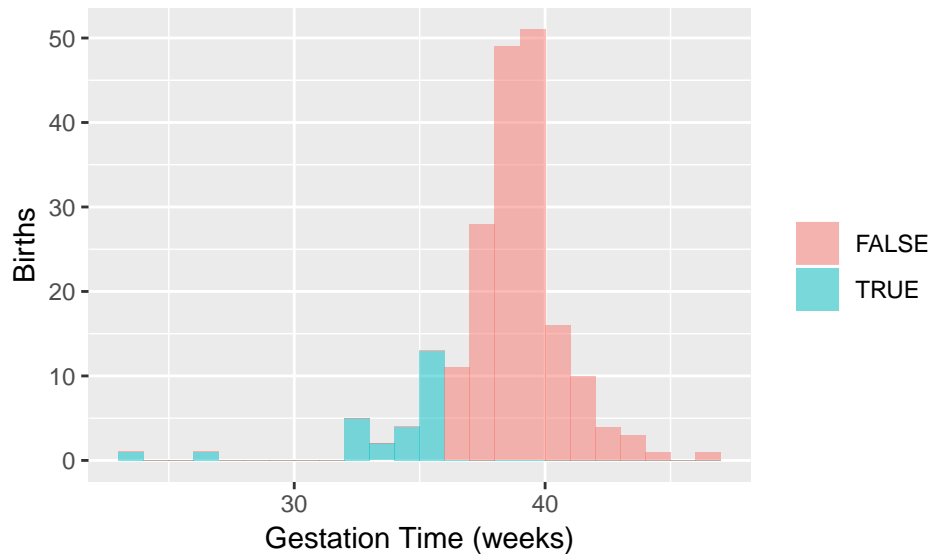
```
## Parsed with column specification:
## cols(
##   MomAge = col_integer(),
##   DadAge = col_integer(),
##   MomEduc = col_integer(),
##   MomMarital = col_integer(),
##   numlive = col_integer(),
##   dobmm = col_integer(),
##   gestation = col_integer(),
##   sex = col_character(),
##   weight = col_integer(),
##   prenatalstart = col_integer(),
##   orig.id = col_integer(),
##   preemie = col_logical()
## )
```

By default, `read_csv()` prints the variable names. These messages can be suppressed using the `message=FALSE` code chunk option to save space and improve readability.

Here we use the `clean_names()` function from the `janitor` package to sanitize the names of the columns (which would otherwise contain special characters or whitespace).

Figure 13.1, page 411

```
gf_histogram(~ gestation, binwidth = 1, center = .5, fill = ~ preemie, data = Babies) %>%
  gf_labs(x = "Gestation Time (weeks)", y = "Births", fill = "")
```



Section 13.1: The Sampling Distribution Model for a Proportion

The Normal Model

Section 13.2: When Does the Normal Model Work? Assumptions and Conditions

Random Matters: Does the Normal Model Always Work? Sampling Distributions for Other Statistics

page 418

```
BodyFat <- read_csv("http://nhorton.people.amherst.edu/is5/data/Bodyfat.csv") %>%
  clean_names()
```

```
## Parsed with column specification:
```

```
## cols(
```

```
##   Density = col_double(),
```

```
##   Pct.BF = col_double(),
```

```
##   Age = col_integer(),
```

```
##   Weight = col_double(),
```

```
##   Height = col_double(),
```

```
##   Neck = col_double(),
```

```
##   Chest = col_double(),
```

```
##   Abdomen = col_double(),
```

```
##   Waist = col_double(),
```

```
##   Hip = col_double(),
```

```
##   Thigh = col_double(),
```

```
##   Knee = col_double(),
```

```
##   Ankle = col_double(),
```

```
##   Bicep = col_double(),
```

```
##   Forearm = col_double(),
```

```
##   Wrist = col_double()
```

```
## )

set.seed(3245) # To ensure we get the same values when we run it multiple times
numsim <- 1000 # Number of samples

# What does do() do?
favstats(~ weight, data = sample(BodyFat, 10)) # favstats of one random sample of 10

##      min      Q1 median      Q3      max      mean      sd n missing
## 148.25 166.25 176.5 192.125 247.25 183.725 29.88925 10      0

favstats(~ weight, data = sample(BodyFat, 10)) # favstats of another random sample

##      min      Q1 median      Q3      max      mean      sd n missing
## 127.5 154.9375 161 189.4375 216.25 168.725 27.3563 10      0

do(2) * favstats(~ weight, data = sample(BodyFat, 10)) # finds favstats twice

##      min      Q1 median      Q3      max      mean      sd n missing .row
## 1 125.00 168.7500 188.875 208.8750 241.75 186.400 33.96367 10      0      1
## 2 156.75 167.4375 179.875 189.5625 224.50 182.625 21.03610 10      0      1
##      .index
## 1          1
## 2          2

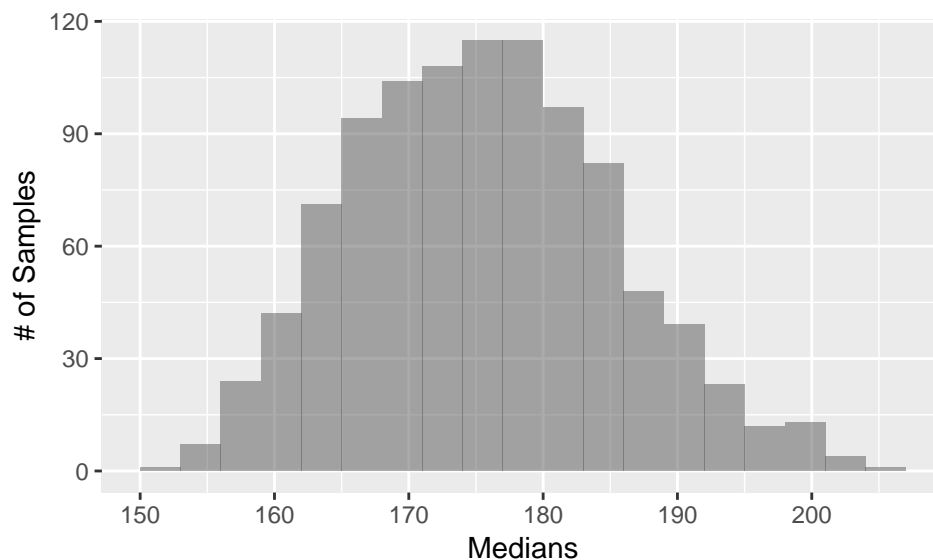
# For the visualization, we need 1,000 favstats
bodyfatsamples <- do(numsim) * favstats(~ weight, data = sample(BodyFat, 10))
```

Here the do() function repeatedly calculates the summary statistics for a random sample of 10 weights.

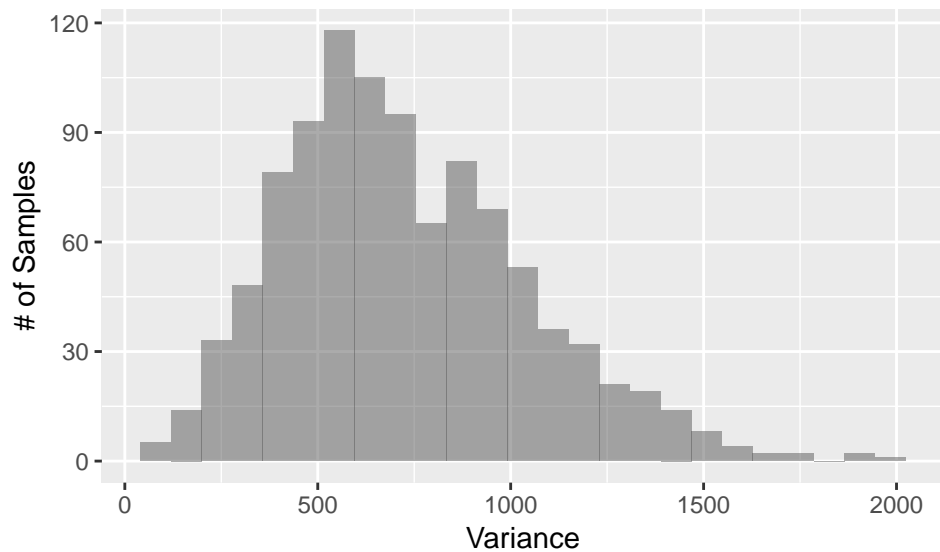
```
bodyfatsamples <- bodyfatsamples %>%
  clean_names()
names(bodyfatsamples)

## [1] "min"      "q1"      "median"   "q3"      "max"      "mean"     "sd"
## [8] "n"        "missing" "row"      "index"

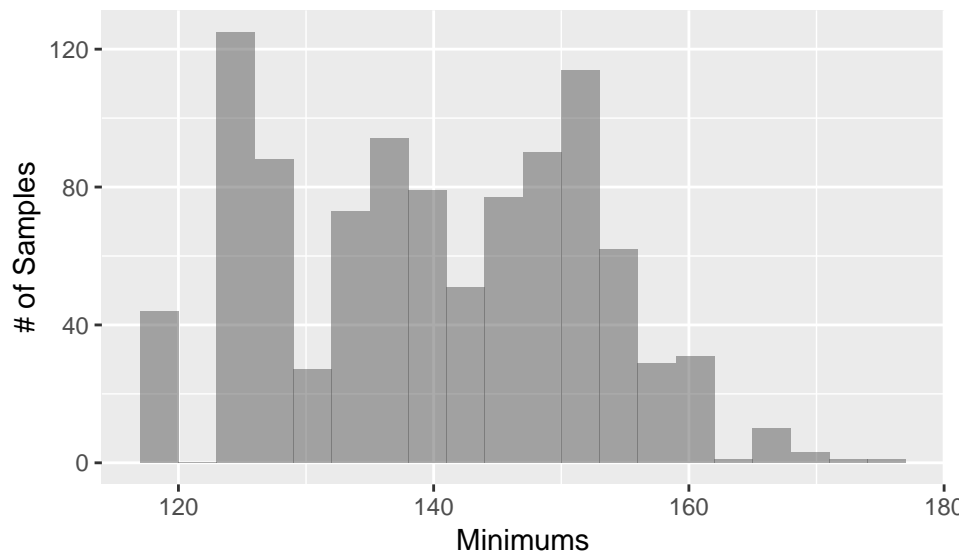
gf_histogram(~ median, data = bodyfatsamples, binwidth = 3, center = 1.5) %>%
  gf_labs(x = "Medians", y = "# of Samples")
```



```
gf_histogram(~ sd^2, data = bodyfatsamples) %>%
  gf_labs(x = "Variance", y = "# of Samples")
```



```
gf_histogram(~ min, data = bodyfatsamples, binwidth = 3, center = 1.5) %>%
  gf_labs(x = "Minimums", y = "# of Samples")
```



Section 13.3: A Confidence Interval for a Proportion

Section 13.4: Interpreting Confidence Intervals: What Does 95% Confidence Really Mean?

First we can replicate the example on pages 423-424.

```
y <- 1034
n <- 1520
phat <- y/n
phat
```

```
## [1] 0.6802632
```

```
sephat <- sqrt(phat * (1 - phat) / n)
sephat
```

```
## [1] 0.01196225
```

```
phat + c(-2, 2) * sephat # matches interval on the bottom of page 423
```

```
## [1] 0.6563386 0.7041877
```

Note that we should actually use 1.96 rather than 2 as the multipliers.

We can also use the `prop.test()` and `binom.test()` functions to calculate the interval for us.

```
prop.test(y, n, correct = FALSE) # large sample methods
```

```
##
## 1-sample proportions test without continuity correction
##
## data: y out of n
## X-squared = 197.57, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
## 0.6563883 0.7032292
## sample estimates:
## p
## 0.6802632
```

```
binom.test(y, n) # exact methods
```

```
##
##
##
## data: y out of n
## number of successes = 1034, number of trials = 1520, p-value <
## 2.2e-16
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
## 0.6561572 0.7036706
## sample estimates:
## probability of success
## 0.6802632
```

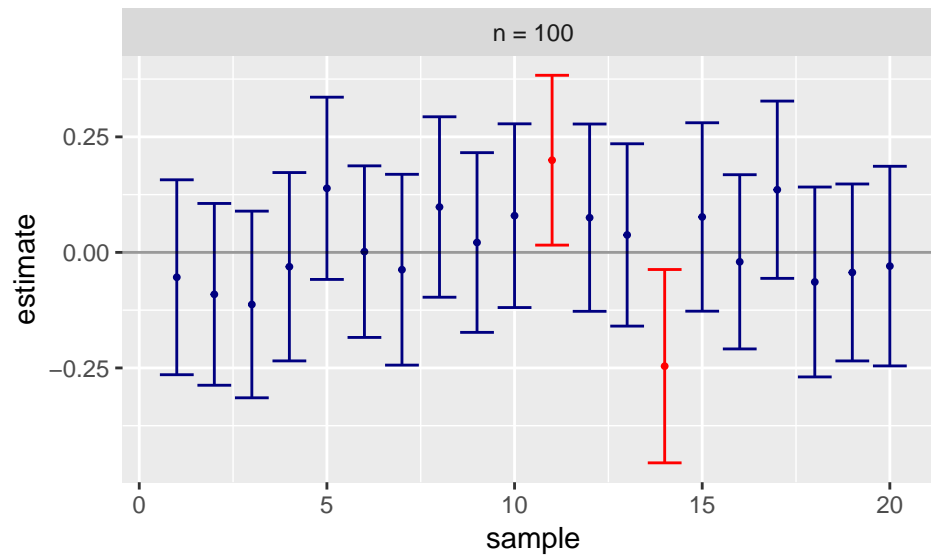
The intervals are almost exactly the same (not surprising, given how large a sample size we have).

Next, we can recreate the simulation displayed in Figure 13.9 (page 422)

```
set.seed(118)
CIsim(n = 100, samples = 20) # We expect 19/20 intervals to cover the true mean
```

```
## Interval coverage:
```

```
##      cover
## n      Low  Yes High
## 100 0.05 0.90 0.05
```



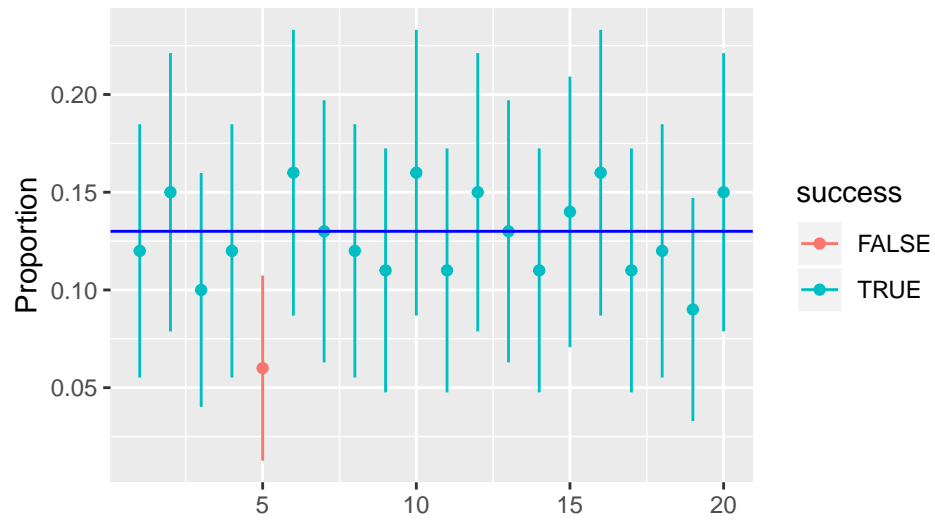
We expect 19 of the 20 intervals to cover the true mean, but since only 20 samples are drawn, there is more variability. Only 18 out of the 20 intervals cover the true mean in this example.

To get the actual plot, the code is more complicated.

```
set.seed(234)
findingpoints <- function(sampsize) {
  CIttest <- do(1) * t.test(~ preemie, data = sample(Babies, size = sampsize))
  # Using do() so that CIttest can run as a data frame
  CIttest <- CIttest %>%
    select(lower, upper) %>%
    mutate(mean = (upper + lower)/2,
           success = ifelse(lower <= .11 & upper >= .11, TRUE, FALSE))
}

numsamp <- 20
ConfData <- do(numsamp) * findingpoints(sampsize = 100)

gf_point(mean ~ (1:20), data = ConfData, color = ~ success) %>%
  gf_segment(upper + lower ~ (1:20) + (1:20), data = ConfData) %>%
  gf_hline(yintercept = ~ mean(preemie), data = Babies, color = 4) %>%
  gf_labs(x = "", y = "Proportion")
```



Section 13.5: Margin of Error: Certainty vs. Precision

Section 13.6: Choosing the Sample Size