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

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

July 11, 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.

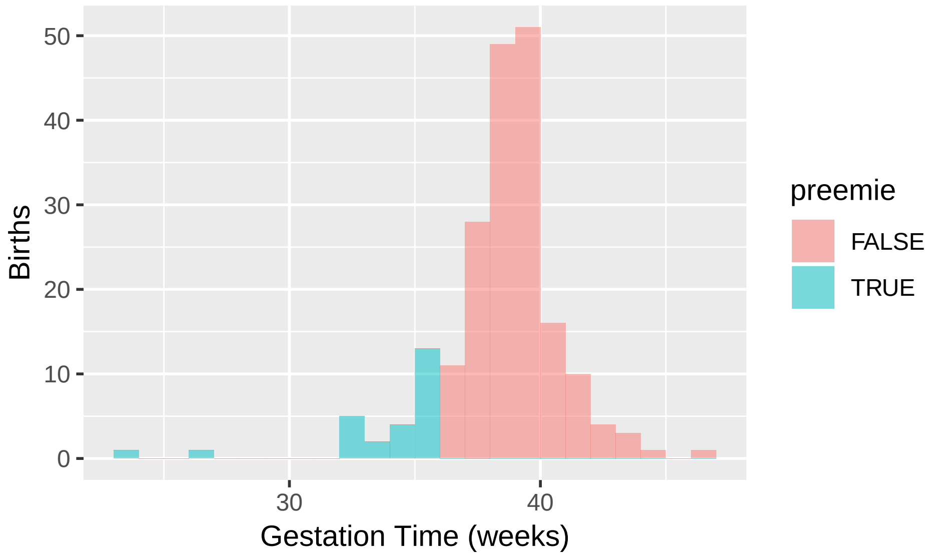
Chapter 13: Sampling Distribution Models and Confidence Intervals for Proportions

```r
# Add page refs
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).

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



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

```
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()
## )
```

```r
set.seed(3245) # For reproducibility
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
```

```r
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
```

```r
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
```

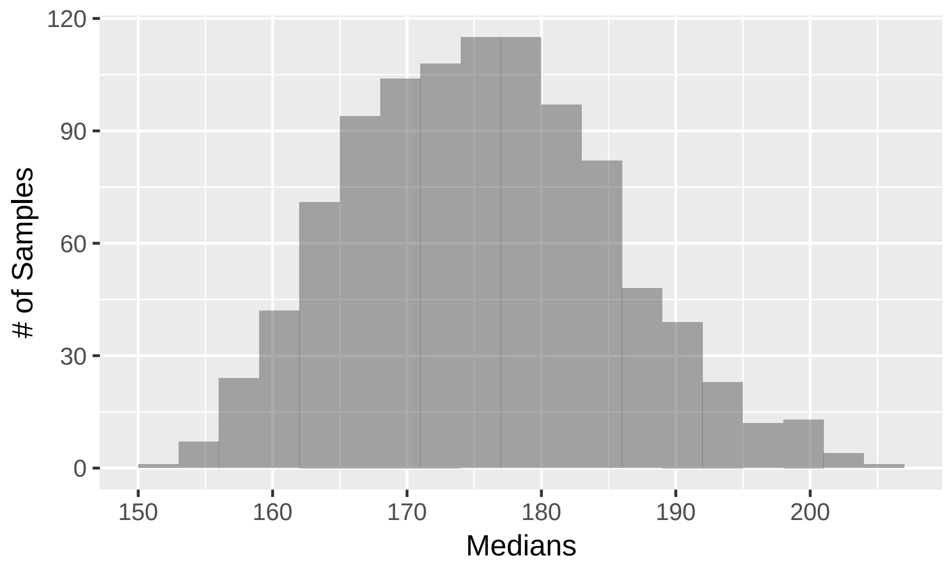
```r
# For the visualization, we need 1,000 favstats
bodyfatsamples <- do(numsim) * favstats(~ weight, data = sample(BodyFat, 10))
```

Here, the `do()` function finds, 1,000 times, the `favstats()` of a random sample of 10 `BodyFat` weights.

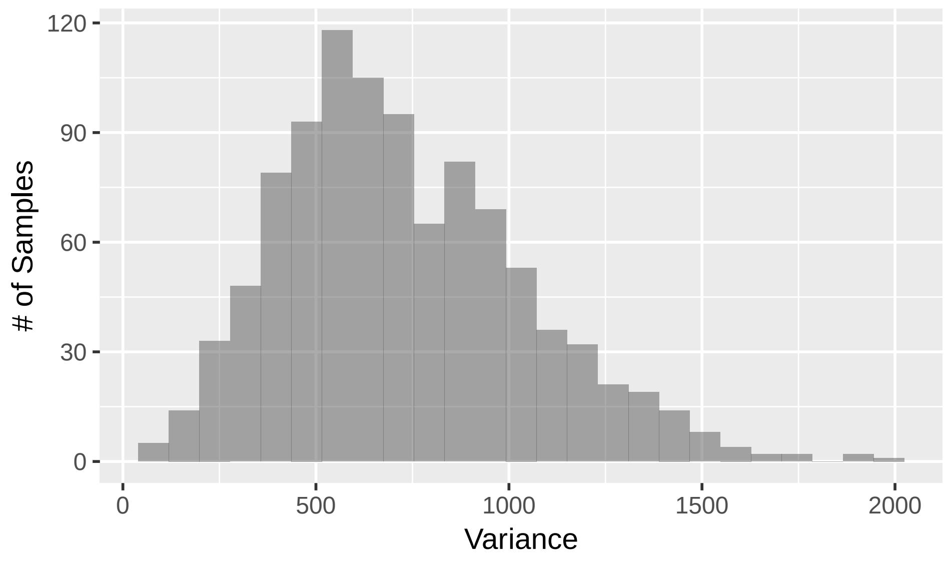
```r
bodyfatsamples <- bodyfatsamples %>%
  clean_names()
names(bodyfatsamples)
```

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

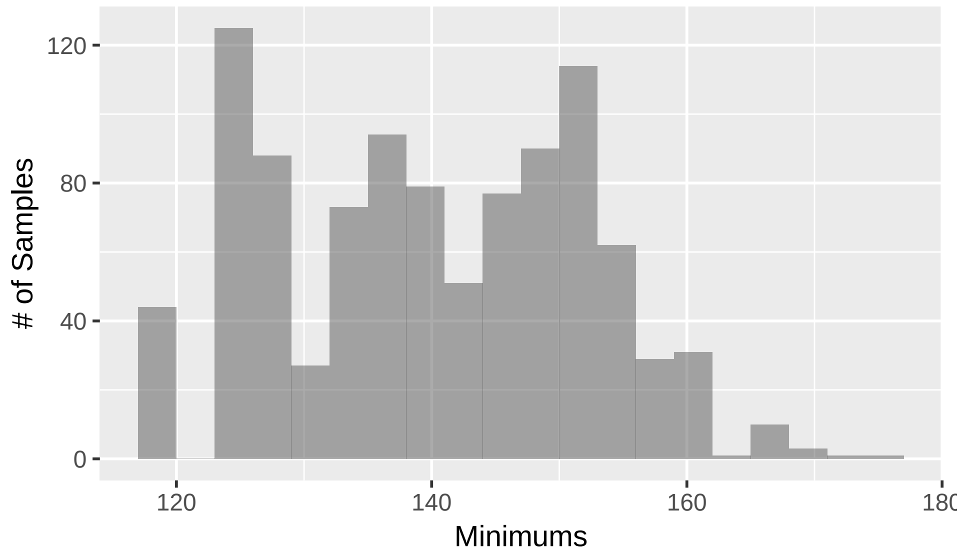
```r
gf_histogram(~ median, data = bodyfatsamples, binwidth = 3, center = 1.5) %>%
  gf_labs(x = "Medians", y = "# of Samples")
```



```
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")
```



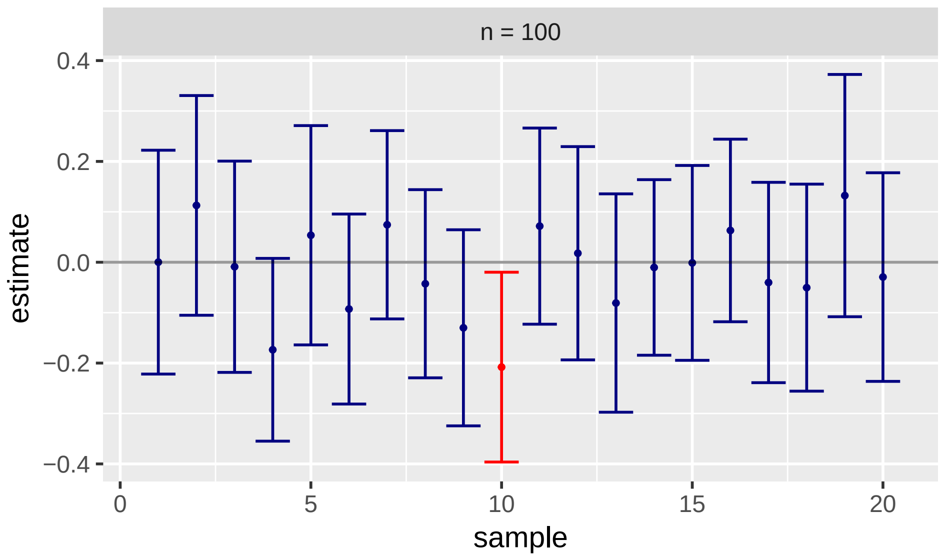
Section 13.3: A Confidence Interval for a Proportion

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

```
# Here is a simulation of Figure 13.9 (page 422)
set.seed(355)
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 0.95 0.00
```



XX Book gets 18/20

```
CIdata <- do(20) * favstats(~ preemie, data = sample(Babies, size = 100))
```

```
## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.

## Warning in FUN(eval(formula[[2]], data, .envir), ...): Auto-converting
## logical to numeric.
```

```
CIdata <- CIdata %>%
  mutate(lowerbound = mean - 1.96 * sd/(100^.5), upperbound = mean + 1.96 * sd/(100^.5)) # use t test f
CIdata %>%
  select(lowerbound, upperbound) # make function that returns center, lower, upper, t test, recreating f
```

```
##    lowerbound upperbound
## 1  0.07164792  0.2083521
## 2  0.03362580  0.1463742
## 3  0.05598667  0.1840133
## 4  0.04836466  0.1716353
## 5  0.03362580  0.1463742
## 6  0.07966142  0.2203386
## 7  0.07966142  0.2203386
## 8  0.05598667  0.1840133
## 9  0.07966142  0.2203386
## 10 0.07164792  0.2083521
## 11 0.07164792  0.2083521
## 12 0.04836466  0.1716353
## 13 0.09600507  0.2439949
## 14 0.09600507  0.2439949
## 15 0.06375246  0.1962475
## 16 0.06375246  0.1962475
## 17 0.07164792  0.2083521
## 18 0.05598667  0.1840133
## 19 0.05598667  0.1840133
## 20 0.06375246  0.1962475
```

```
# CIsim(~ preemie, data = sample(Babies, size = 100), samples = 20)
```

Section 13.5: Margin of Error: Certainty vs. Precision

Section 13.6: Choosing the Sample Size