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

Nicholas Horton (nhorton@amherst.edu)

2025-01-15

Table of contents

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. This file as well as the associated Quarto 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 (https://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.

We begin by loading packages that will be required for our analyses.

```r
library(mosaic)
library(tidyverse)
```

Chapter 13: Sampling Distribution Models and Confidence Intervals for Proportions

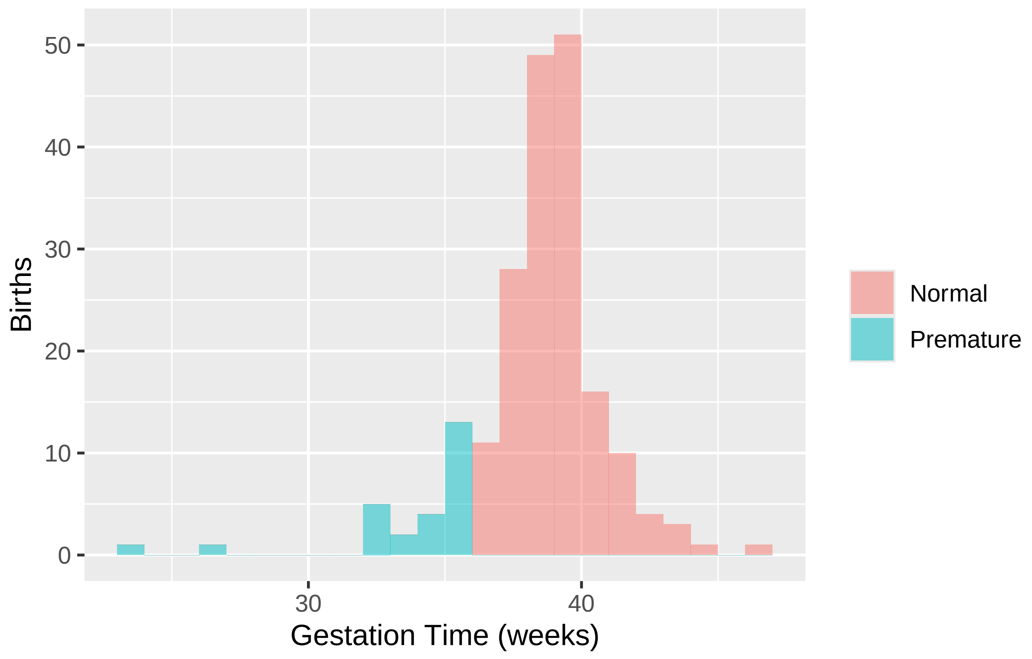
```r
Babies <- read_csv("http://nhorton.people.amherst.edu/is5/data/Babysamp_98.csv") |>
  janitor::clean_names() |>
  mutate(status = ifelse(preemie, "Premature", "Normal"))
glimpse(Babies)
```

```
Rows: 200
Columns: 13
$ mom_age        <dbl> 35, 22, 35, 23, 23, 26, 25, 32, 41, 22, 25, 24, 20, 19, ~
$ dad_age        <dbl> 35, 21, 42, NA, 28, 31, 37, 38, 39, 24, NA, 29, 22, 20, ~
$ mom_educ       <dbl> 17, 12, 15, 6, 13, 12, 16, 16, 14, 12, 2, 14, 12, 13, 12~
$ mom_marital    <dbl> 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1,~
$ numlive        <dbl> 2, 1, 0, 2, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,~
$ dobmm          <dbl> 2, 3, 6, 8, 9, 10, 7, 12, 11, 2, 2, 12, 11, 7, 4, 4, 9, ~
$ gestation      <dbl> 39, 42, 39, 40, 42, 39, 38, 38, 36, 40, 40, 40, 41, 36, ~
$ sex            <chr> "F", "F", "F", "F", "F", "M", "F", "M", "M", "M", "F", "~
$ weight         <dbl> 3175, 3884, 3030, 3629, 3481, 3374, 2693, 4338, 2834, 29~
$ prenatalstart <dbl> 1, 2, 2, 1, 2, 4, 1, 1, 2, 1, 1, NA, 2, 8, 2, 2, 1, 2, 3~
$ orig_id        <dbl> 1047483, 1468100, 2260016, 3583052, 795674, 3544316, 372~
$ preemie        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, ~
$ status         <chr> "Normal", "Normal", "Normal", "Normal", "Normal", "Norma~
```

By default, `read_csv()` prints the variable names. These messages have been 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).

The `mutate()` function is used in conjunction with the `ifelse()` function to create a new variable `status`.

```r
# Figure 13.1, page 411
gf_histogram(~ gestation, binwidth = 1, center = .5, fill = ~ status, 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") |>
  janitor::clean_names()
set.seed(3245) # To ensure we get the same values when we run it multiple times
num_sim <- 1000 # Number of samples
```

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

```
  response    min       Q1 median       Q3    max mean       sd  n missing
1   weight 140.25 160.5625 170.75 182.5625 262.75  177 33.57496 10       0
```

```r
df_stats(~ weight, data = sample(BodyFat, 10)) # df_stats of another random sample
```

```
  response    min       Q1  median      Q3   max  mean       sd  n missing
1   weight 133.25 161.6875 180.375 196.625 218.5 178.2 27.27428 10       0
```

The `do()` function in the `mosaic` package is used in combination with the `*` operator to run functions repeatedly. We will use these functions to sample in different ways.

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

```
  response    min       Q1 median       Q3    max    mean       sd  n missing
1   weight 145.25 149.7500 167.00 179.7500 195.75 167.575 18.44137 10       0
2   weight 126.50 163.4375 174.75 202.0625 241.25 182.975 36.24156 10       0
  .row .index
1    1      1
2    1      2
```

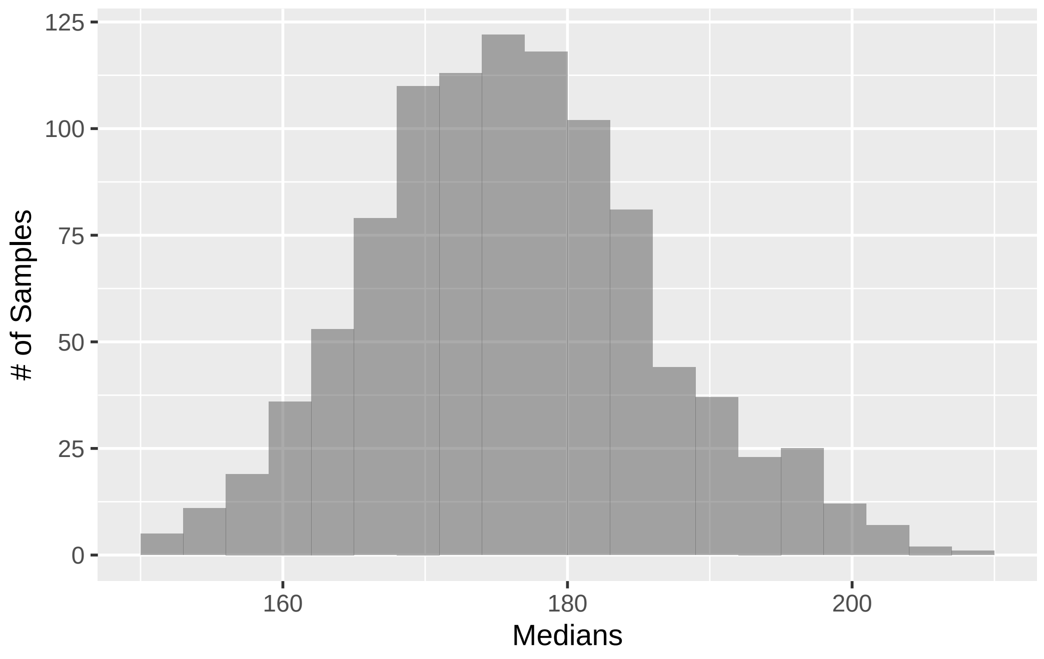
```r
# For the visualization, we need num_sim = 1,000 df_stats
bodyfatsamples <- do(num_sim) * df_stats(~ weight, data = sample(BodyFat, 10))
```

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

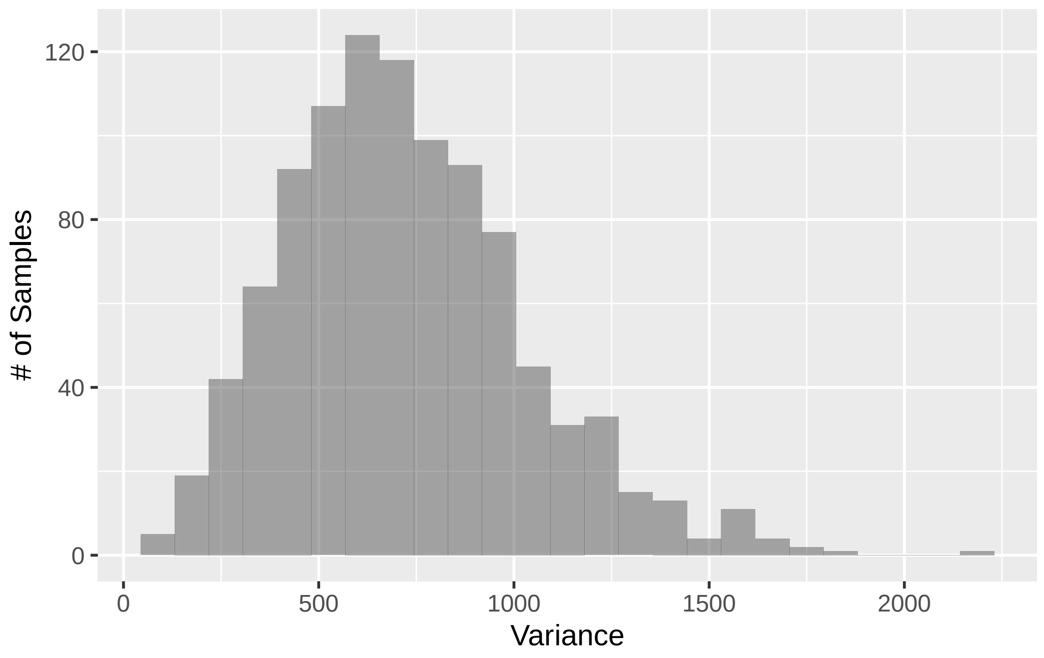
```r
bodyfatsamples <- bodyfatsamples |>
  janitor::clean_names()
names(bodyfatsamples)
```

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

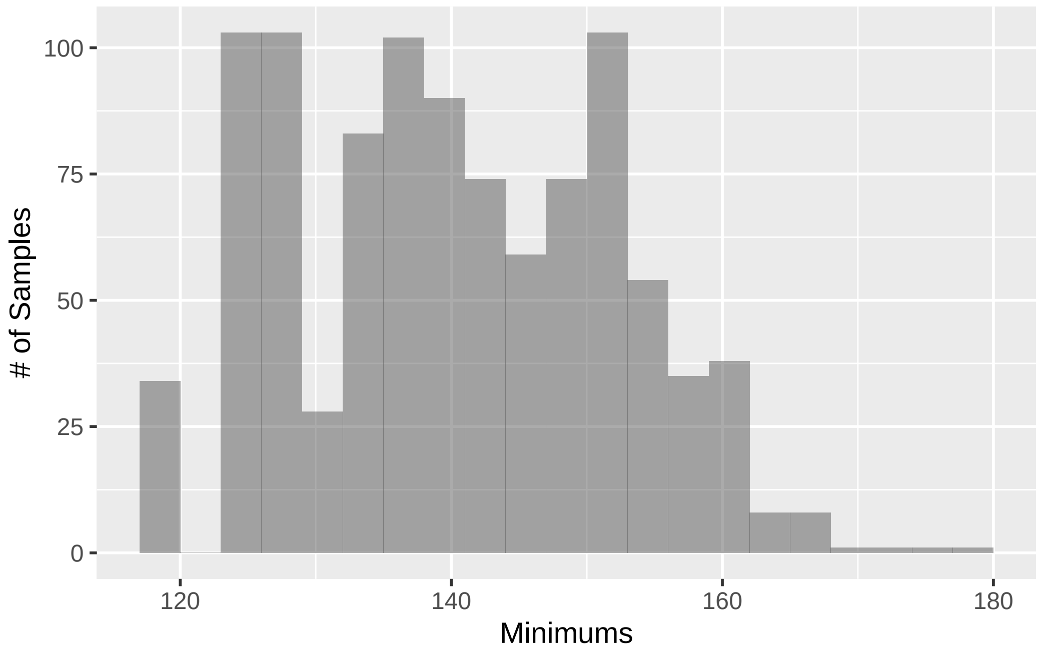
```r
gf_histogram(~ median, data = bodyfatsamples, binwidth = 3, center = 1.5) |>
  gf_labs(x = "Medians", y = "# of Samples")
```



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



```r
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.

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

```
[1] 0.6802632
```

```r
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 1520
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
```

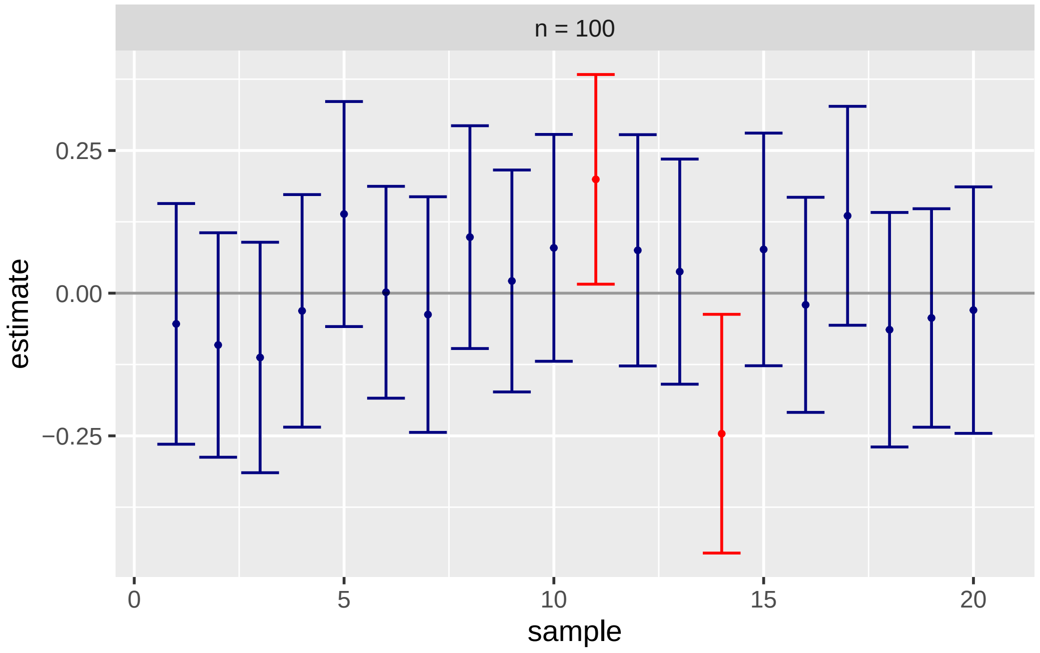
The intervals are almost exactly the same (not surprising, given how large a sample size we have).

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

```r
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
```



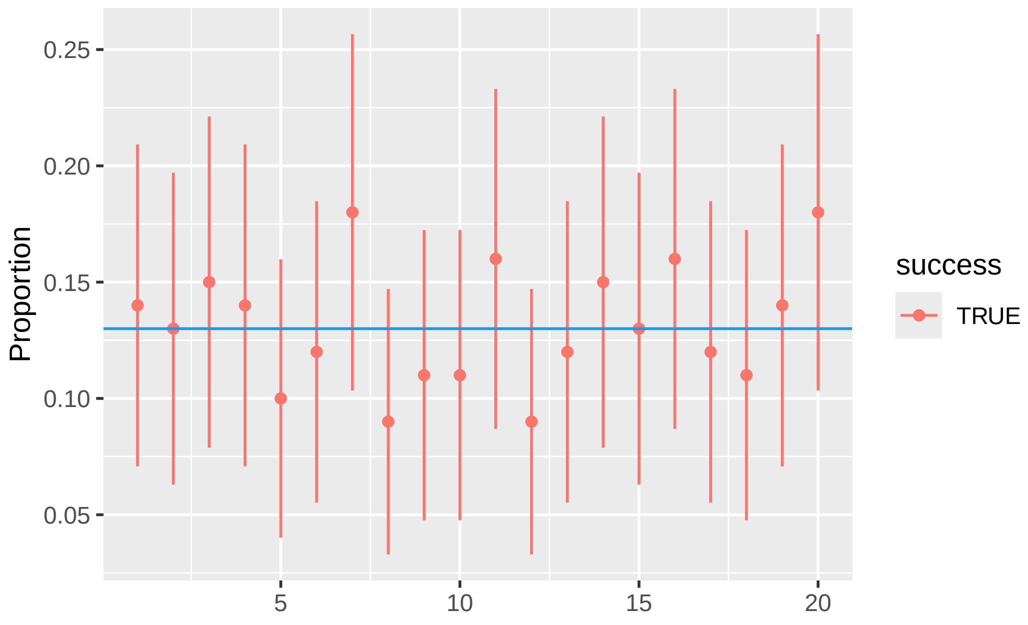
We expect 19 of the 20 intervals to cover the true mean, but since only 20 samples are drawn, there is more variability. Only 18 out of the 20 intervals cover the true mean in this example.

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

```r
set.seed(234)
findingpoints <- function(sampsize) {
  CItest <- do(1) * t.test(~preemie, data = sample(Babies, size = sampsize))
  # Using do() so that CItest can run as a data frame
  CItest <- CItest |>
    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:numsamp), data = ConfData, color = ~ success) |>
  gf_segment(upper + lower ~ (1:numsamp) + (1:numsamp), 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