

IS5 in R: Confidence Intervals for Means (Chapter 14)

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

July 16, 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 14: Confidence Intervals for Means

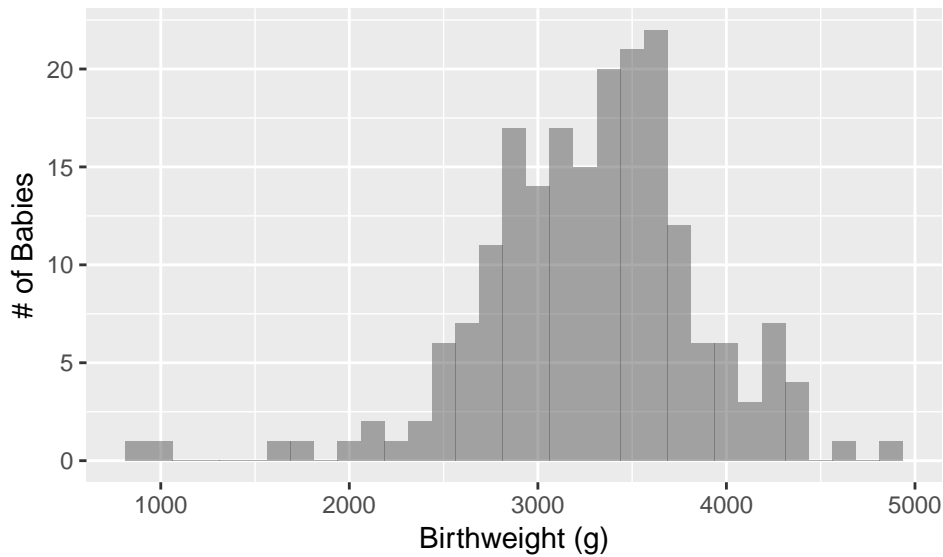
```
library(mosaic)
library(readr)
library(janitor)
Babies <- read_csv("http://nhorton.people.amherst.edu/is5/data/Babysamp_98.csv") %>%
  clean_names()
```

```
## 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 14.1, page 441
gf_histogram(~ weight, data = Babies, binwidth = 125) %>%
  gf_labs(x = "Birthweight (g)", y = "# of Babies")
```



```
# Figure 14.2
set.seed(12346) # To ensure we get the same values when we run it multiple times
numsim <- 10000 # Number of simulations

# What does do() do?
mean(~ weight, data = sample(Babies, size = 100)) # Mean of a random sample of 100

## [1] 3214.53

mean(~ weight, data = sample(Babies, size = 100)) # Mean of another random sample

## [1] 3290.34

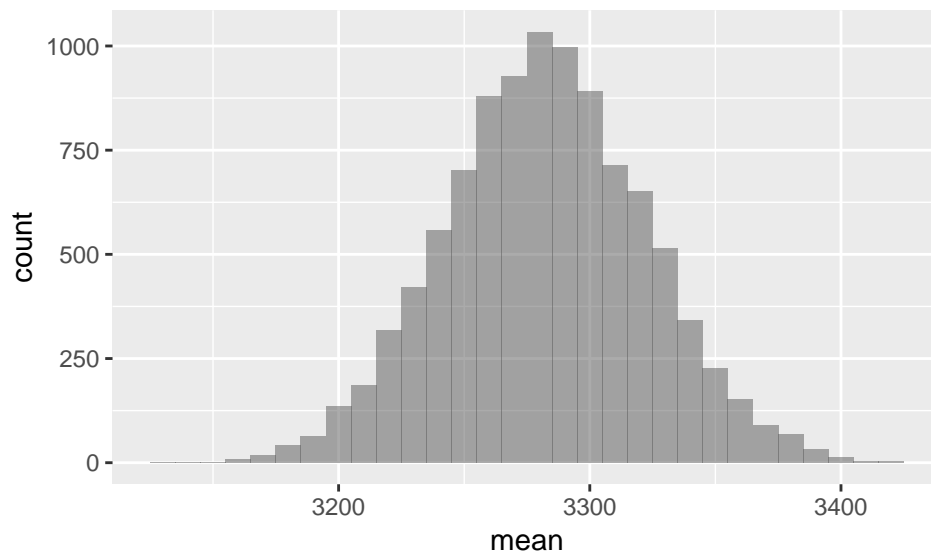
do(2) * mean(~ weight, data = sample(Babies, size = 100)) # Calculates the mean twice

##      mean
## 1 3297.84
## 2 3286.70

# For the visualization, we need 10,000 means
WeightMeans <- do(numsim) * mean(~ weight, data = sample(Babies, size = 100))
```

The `do()` function repeatedly calculates the mean of a random sample of 100 baby weights.

```
gf_histogram(~ mean, data = WeightMeans, binwidth = 10)
```



```
favstats(~ mean, data = WeightMeans)
```

```
##      min      Q1  median      Q3      max      mean      sd      n missing
## 3132.17 3255.54 3282.14 3309.163 3422.09 3282.299 40.01813 10000      0
```

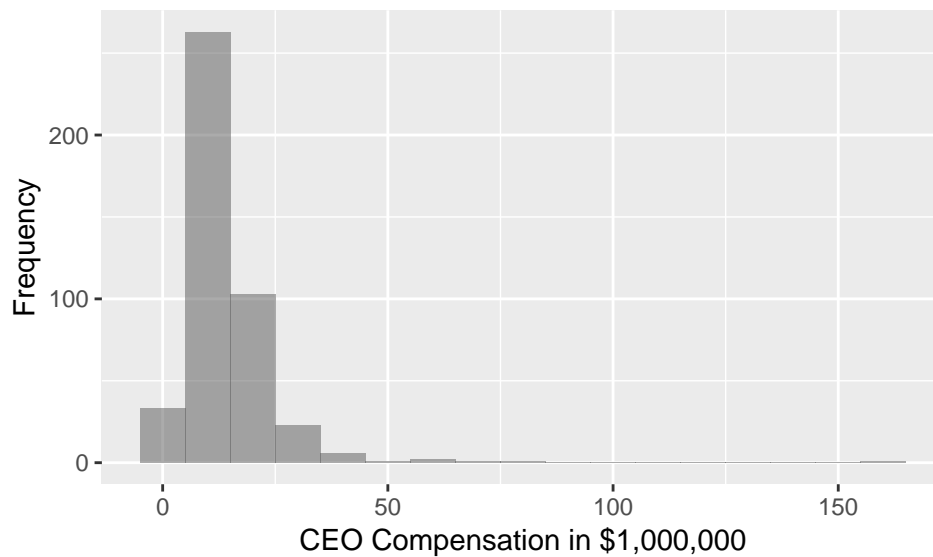
Section 14.1: The Central Limit Theorem

```
CEOCComp <- read_csv("http://nhorton.people.amherst.edu/is5/data/CEO_Compensation_2014.csv") %>%
  clean_names()
```

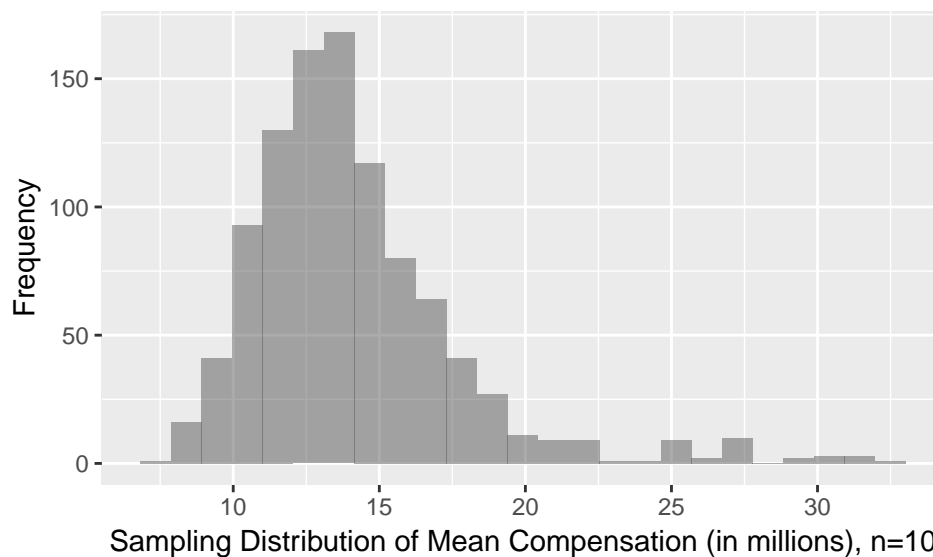
```
## Parsed with column specification:
## cols(
##   Employer = col_character(),
##   CEO = col_character(),
##   CEO_Compensation = col_integer(),
##   Median_Worker_Comp = col_integer(),
##   Ratio = col_integer(),
##   Company_Rating = col_double(),
##   `CEO_Compensation_($M)` = col_double()
## )
```

Figure 14.3

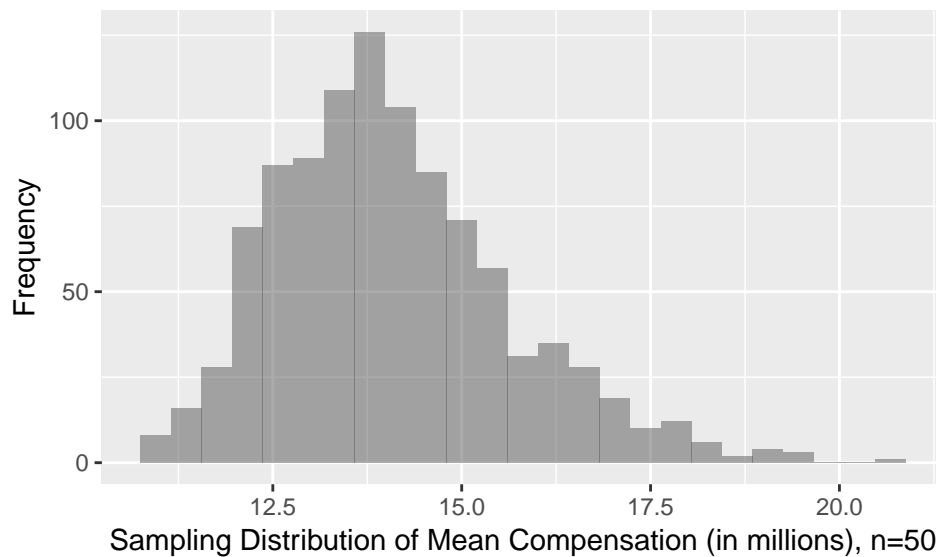
```
gf_histogram(~ ceo_compensation_m, data = CEOComp, binwidth = 10) %>%
  gf_labs(x = "CEO Compensation in $1,000,000", y = "Frequency")
```



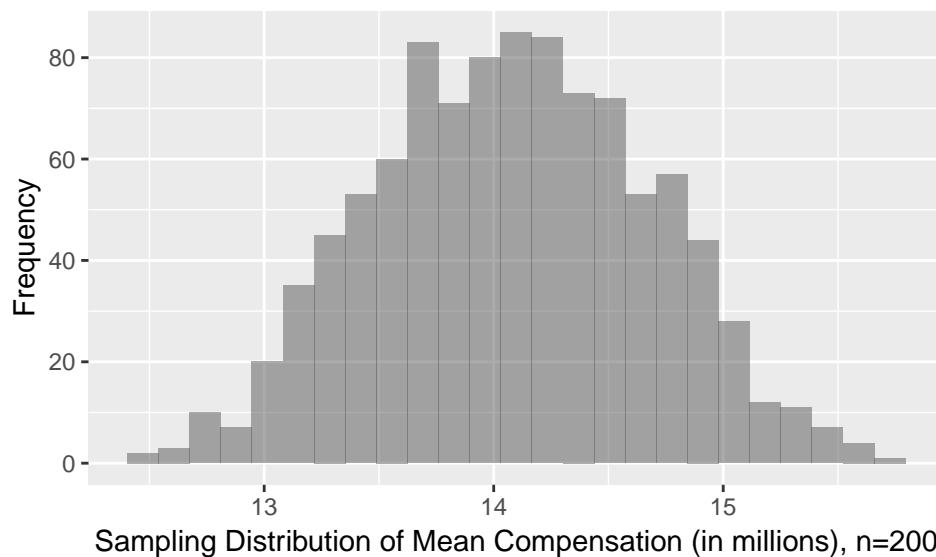
```
# Figure 14.4
set.seed(14354)
numsim <- 1000 # Here the number of simulations is 1,000
CEOMeans <- do(numsim) * mean(~ ceo_compensation_m, data = sample(CEOComp, size = 10))
gf_histogram(~ mean, data = CEOMeans) %>%
  gf_labs(x = "Sampling Distribution of Mean Compensation (in millions), n=10", y = "Frequency")
```



```
# Figure 14.5
CEOMeans2 <- do(numsim) * mean(~ ceo_compensation_m, data = sample(CEOComp, size = 50))
gf_histogram(~ mean, data = CEOMeans2) %>%
  gf_labs(x = "Sampling Distribution of Mean Compensation (in millions), n=50", y = "Frequency")
```



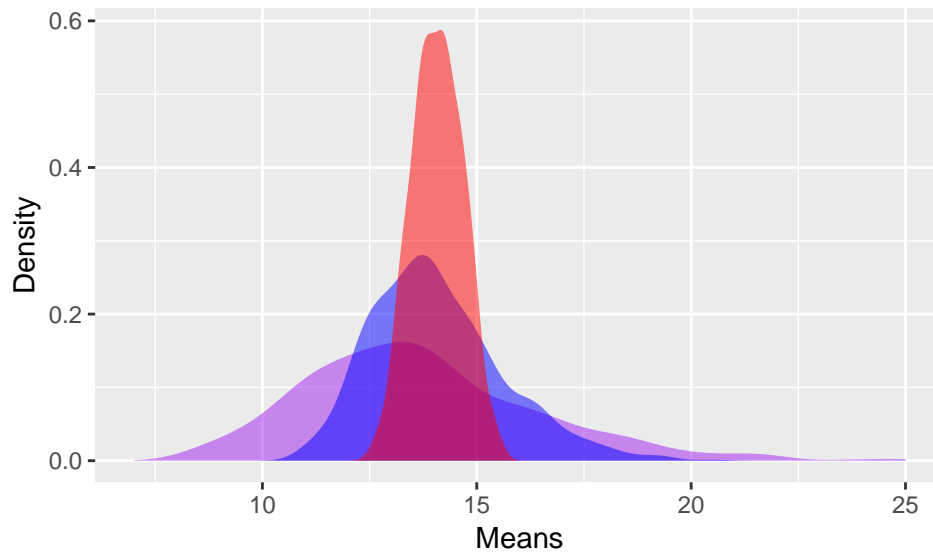
```
# Figure 14.7 (skipped 14.6 because it's similar)
CEOMeans3 <- do(numsim) * mean(~ ceo_compensation_m, data = sample(CEOComp, size = 200))
gf_histogram(~ mean, data = CEOMeans3) %>%
  gf_labs(x = "Sampling Distribution of Mean Compensation (in millions), n=200", y = "Frequency")
```



For each example sample size, the `do()` function calculates the mean of random samples of that specified size.

```
# Samples as overlaid density plots
gf_density(~ mean, data = CEOMeans, fill = "purple") %>%
  gf_density(~ mean, data = CEOMeans2, fill = "blue") %>%
  gf_density(~ mean, data = CEOMeans3, fill = "red") %>%
  gf_labs(y = "Density", x = "Means") +
  xlim(7, 25)
```

```
## Warning: Removed 27 rows containing non-finite values (stat_density).
```

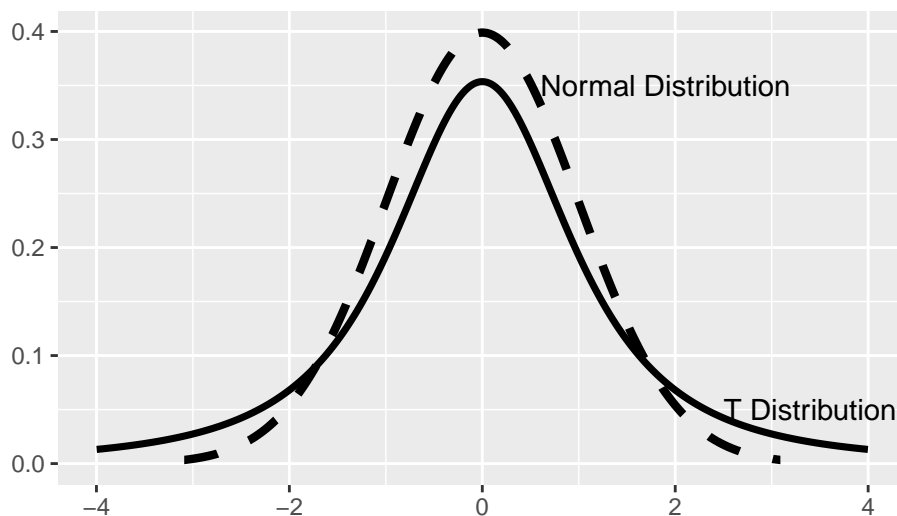


The purple density reflects the distribution of the means from random samples of size 10. The blue density reflects the distribution of the means from random samples of size 50. The red density reflects the distribution of the means from random samples of size 200.

Section 14.2: A Confidence Interval for the Mean

```
# Figure 14.9, page 446
gf_dist(dist = "norm", linetype = 2, lwd = 1.5) %>%
  gf_refine(annotate(geom = "text", x = 1.9, y = .35, label = "Normal Distribution")) %>%
  gf_labs(x = "", y = "") %>%
  gf_dist(dist = "t", df = 2, lwd = 1.25) %>%
  gf_refine(annotate(geom = "text", x = 3.4, y = .05, label = "T Distribution")) +
  xlim(-4, 4)
```

Warning: Removed 4104 rows containing missing values (geom_path).



Example 14.1: A One-Sample t -Interval for the Mean

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

```
## Parsed with column specification:
## cols(
##   Kind = col_character(),
##   Location = col_character(),
##   Mirex = col_double(),
##   Hexachlorobenzene = col_double(),
##   HCH_gamma = col_double(),
##   `Heptachlor Epoxide` = col_double(),
##   Dieldrin = col_double(),
##   Endrin = col_double(),
##   `Total Chlordane` = col_double(),
##   `Total DDT` = col_double(),
##   Dioxin = col_double(),
##   `Total Pesticides` = col_double(),
##   `Total PCBs` = col_integer()
## )
```

```
Salmon <- Salmon %>%
  filter(mirex != "NA")
favstats(~ mirex, data = Salmon)
```

```
##   min    Q1 median      Q3   max   mean      sd  n missing
##    0 0.056  0.079 0.13475 0.194 0.09134 0.04952388 150      0
```

```
t.test(~ mirex, data = Salmon)
```

```
##
## One Sample t-test
##
## data: mirex
## t = 22.589, df = 149, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.08334978 0.09933022
## sample estimates:
## mean of x
##  0.09134
```

```
salmonlm <- lm(mirex ~ 1, data = Salmon) # equiv to t.test(~ mirex)
# replication of interval from page 448
confint(salmonlm, data = Salmon)
```

```
##              2.5 %      97.5 %
## (Intercept) 0.08334978 0.09933022
```

```
tstats <- xqt(df = 149, p = c(.025, .975), plot = FALSE) # t statistics
sey <- sd(~ mirex, data = Salmon)/(150^(1/2)) # standard error
mean(~ mirex, data = Salmon) + sey * tstats # calculations match those from confint
```

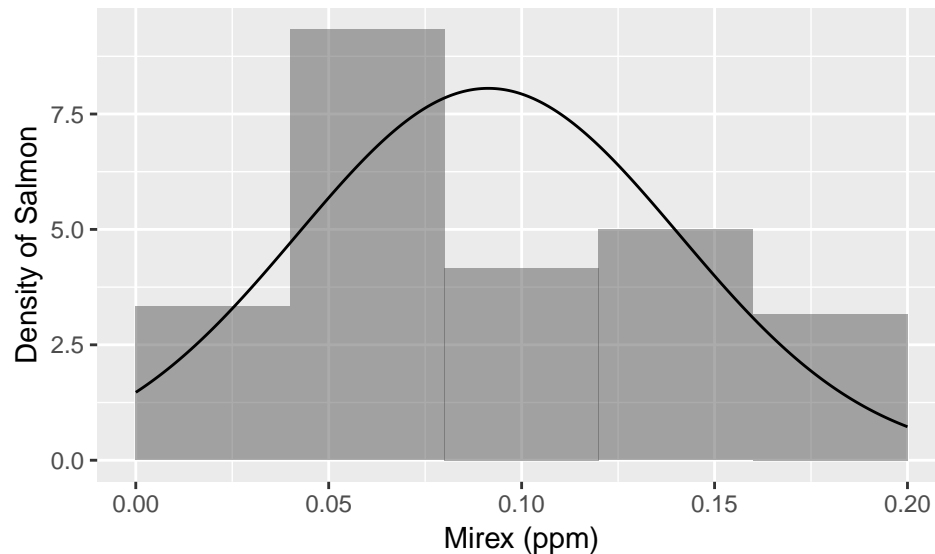
```
## [1] 0.08334978 0.09933022
```

The `confint()` function takes an object, in this case a linear regression model, as an argument.

Example 14.2: Checking Assumptions and Conditions for Student's t

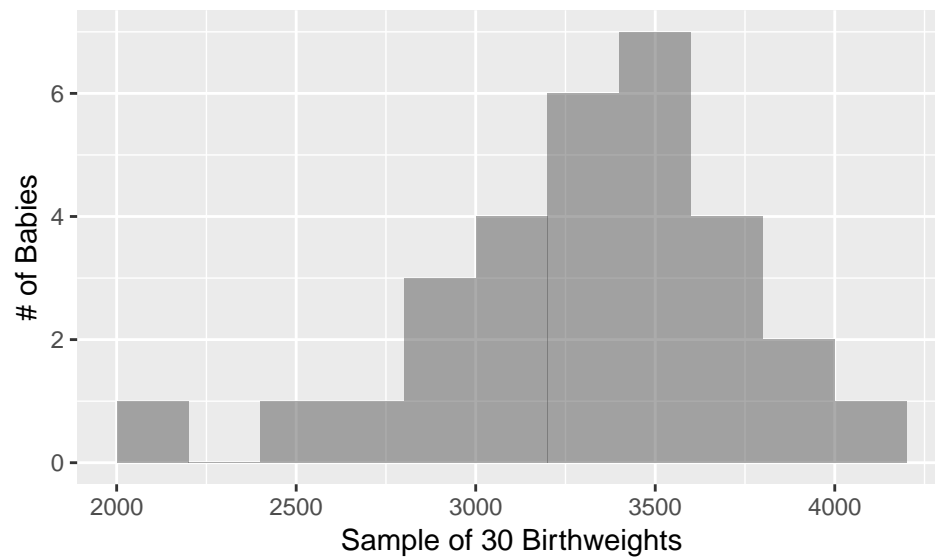
```
# With a normal distribution (page 450)
gf_dhistogram(~ mirex, data = Salmon, binwidth = 0.04, center = 0.02) %>%
  gf_dist(dist = "norm", mean = mean(~ mirex, data = Salmon), sd = sd(~ mirex, data = Salmon)) %>%
  gf_labs(x = "Mirex (ppm)", y = "Density of Salmon") +
  xlim(0, .2)
```

```
## Warning: Removed 1733 rows containing missing values (geom_path).
```



Step-By-Step Example: A One-Sample t -Interval for the Mean

```
set.seed(34)
BabiesSample <- sample(Babies, size = 30)
gf_histogram(~ weight, data = BabiesSample, binwidth = 200, center = 100) %>%
  gf_labs(x = "Sample of 30 Birthweights", y = "# of Babies")
```



```
favstats(~ weight, data = BabiesSample)
```

```
##   min      Q1 median      Q3     max     mean      sd  n missing
```



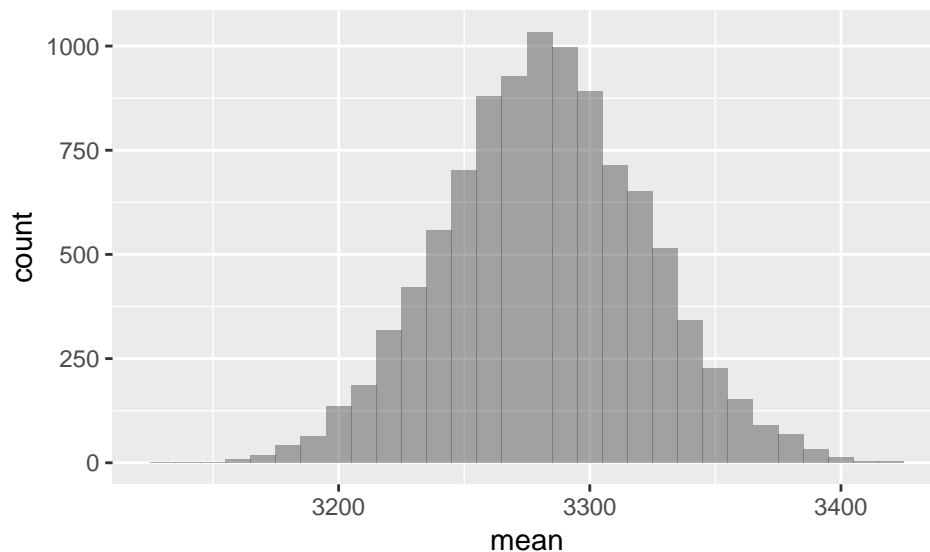
```
## 2182 3056.25 3382.5 3572 4054 3316.6 423.2446 30 0
babieslm <- lm(weight ~ 1, data = BabiesSample)
confint(babieslm, level = 0.90)

##           5 %      95 %
## (Intercept) 3185.302 3447.898
```

Section 14.3: Interpreting Confidence Intervals

Section 14.4: Picking Our Interval up by Our Bootstraps

```
gf_histogram(~ mean, data = WeightMeans, binwidth = 10)
```

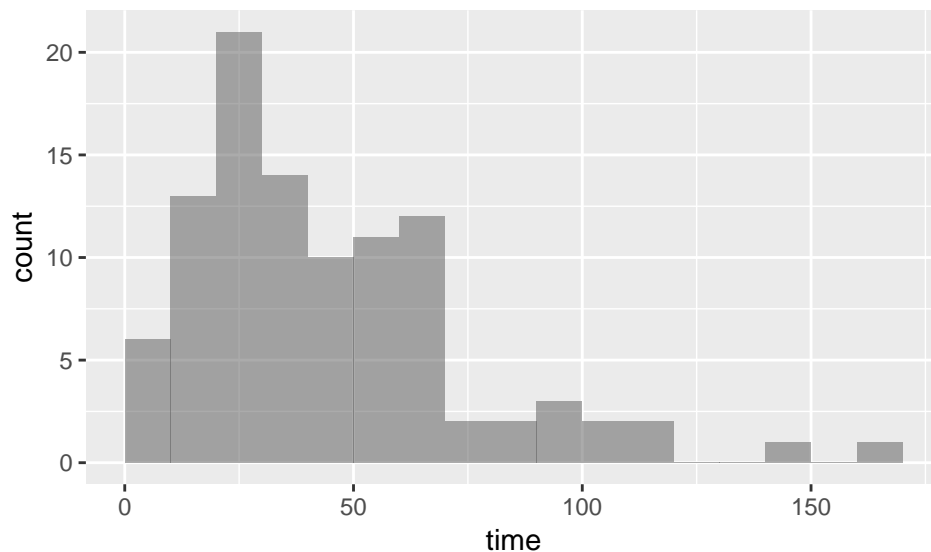


Step-By-Step Example: A Bootstrap Confidence Interval for the Mean

```
CommuteSample <- read_csv("http://nhorton.people.amherst.edu/is5/data/Commuter_sample.csv")

## Parsed with column specification:
## cols(
##   time = col_integer()
## )

gf_histogram(~ time, data = CommuteSample, binwidth = 10, center = 5)
```



```
# Bootstrap
set.seed(134) # To ensure we get the same values when we run it multiple times
numsim <- 10000
commutebootstrap <- do(numsim) * mean(~ time, data = resample(CommuteSample))
```

The `resample()` function samples of that data set size with replacement.

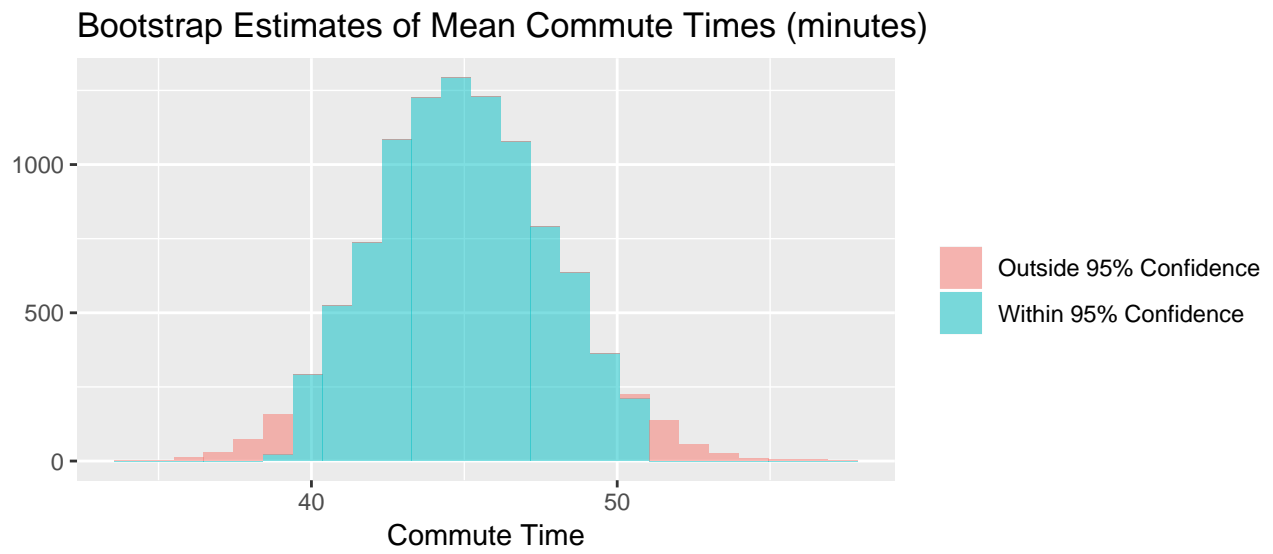
```
qdata(~ mean, p = c(.025, .975), data = commutebootstrap) # grab the percentiles with qdata
```

```
##      quantile      p
## 2.5%      39.28 0.025
## 97.5%      50.98 0.975
```

```
confint(commutebootstrap, method = "quantile") # an equivalent quantile approach
```

```
##   name lower upper level      method estimate
## 1 mean 39.28 50.98  0.95 percentile      44.98
```

```
commutebootstrap <- commutebootstrap %>%
  mutate(interval = ifelse(mean > 39.28 & mean < 50.98, "Within 95% Confidence",
                           "Outside 95% Confidence")) # for fill
gf_histogram(~ mean, fill = ~ interval, data = commutebootstrap,
             title = "Bootstrap Estimates of Mean Commute Times (minutes)") %>%
  gf_labs(x = "Commute Time", y = "", fill = "")
```



Section 14.5: Thoughts About Confidence Intervals