DSCI 552 Lab 1

Populations, Sampling, Bootstrapping, and Sampling Distribution

Contents

```
BEGIN ASSIGNMENT
requirements: requirements.R
files:
  - data
generate:
    show_stdout: true
    show_hidden: true
environment: environment.yml
export_cell: false
```

Lab Mechanics

rubric={mechanics:5}

- Paste the URL to your GitHub repo here: **INSERT YOUR GITHUB REPO URL HERE**
- Once you finish the assignment, you must **knit** this `R` markdown to create a `.pdf` file and push everything to your GitHub repo using `git push`. You are responsible for ensuring all the figures, texts, and equations in the `.pdf` file are appropriately rendered.
- You must submit this `.Rmd` **and** the rendered `.pdf` files to Gradescope.

  Heads-up: You need to have a minimum of 3 commits.

Code Quality

rubric={quality:3}

The code that you write for this assignment will be given one overall grade for code quality. Check our **code quality rubric** as a guide to what we are looking for. Also, for this course (and other MDS courses that use `R`), we are trying to follow the `tidyverse` code style. There is a guide you can refer too: http://style.tidyverse.org/

Each code question will also be assessed for code accuracy (i.e., does it do what it is supposed to do?).

Writing

rubric={writing:3}

To get the marks for this writing component, you should:

- Use proper English, spelling, and grammar throughout your submission (the non-coding parts).
- Be succinct. **This means being specific about what you want to communicate, without being superfluous.**

Check our **writing rubric** as a guide to what we are looking for.

Setup

If you fail to load any packages, you can install them and try loading the library again.

```
library(cowplot)
library(infer)
library(knitr)
library(tidyverse)
library(digest)
library(datateachr)
library(testthat)
```

Note you need to install the package `datateachr` via the following:

1. Uncomment the two lines of code below by deleting the `#` at the start of each line.
2. Run the code cell, which will perform the installation.
3. Comment the two lines of code again by adding the `#` back to the start of each line.

```
# options(timeout=9999999)
# devtools::install_github("UBC-MDS/datateachr")
```

Exercise 1: Conceptual Warmup

rubric={autograde:3}

Read the mixed-up table below and assign each object in the code cell below the integer associated with its correct definition.

> **Note:** Some of these terms may have different meanings in other fields different from Statistics, but these are the definitions that we will be using all over this course and subsequent ones.

Terms	Definitions
Point estimate	**1.** The entire set of entities objects of interest.
Population	**2.** A numerical summary value about the population.
Population parameter	**3.** A collected subset of observations from a population.
Sample	**4.** A summary statistic calculated from a random sample that estimates an unknown population parameter of interest.
Observation	**5.** A distribution of point estimates, where each point estimate was calculated from a different random sample coming from the same population.
Sampling distribution	**6.** A quantity or quality (or a set of these) from a single member of a population.

Assign your answers to the objects given below (`point_estimate`, `population`, `population_parameter`, `sample`, `observation`, and `sampling_distribution`). Your answers should each be a numeric vector of length one (e.g., `term <- 9`).

> **Heads-up:** There are hidden tests which are not shown here and will be applied after you submit.

```
BEGIN QUESTION
name: Q1
points:
 - 0
 - 0
 - 0
 - 0
 - 0
 - 0
 - 0.5
 - 0.5
 - 0.5
 - 0.5
 - 0.5
 - 0.5
```

```r
point_estimate <- NULL
population <- NULL
population_parameter <- NULL
sample <- NULL
observation <- NULL
sampling_distribution <- NULL

# BEGIN SOLUTION
point_estimate <- 4
population <- 1
```

```
population_parameter <- 2
sample <- 3
observation <- 6
sampling_distribution <- 5
# END SOLUTION
```

```
## Test ##
testthat::expect_true(exists("point_estimate"))
```

```
## Test ##
testthat::expect_true(exists("population"))
```

```
## Test ##
testthat::expect_true(exists("population_parameter"))
```

```
## Test ##
testthat::expect_true(exists("sample"))
```

```
## Test ##
testthat::expect_true(exists("observation"))
```

```
## Test ##
testthat::expect_true(exists("sampling_distribution"))
```

```
## Hidden Test ##
testthat::expect_equal(
  digest(as.numeric(point_estimate)),
  "dbc09cba9fe2583fb01d63c70e1555a8"
)
```

```
## Hidden Test ##
testthat::expect_equal(
  digest(as.numeric(population)),
  "6717f2823d3202449301145073ab8719"
)
```

```
## Hidden Test ##
testthat::expect_equal(
  digest(as.numeric(population_parameter)),
  "db8e490a925a60e62212cefc7674ca02"
)
```

```
## Hidden Test ##
testthat::expect_equal(
  digest(as.numeric(sample)),
  "e5b57f323c7b3719bbaaf9f96b260d39"
)
```

```
## Hidden Test ##
testthat::expect_equal(
  digest(as.numeric(observation)),
  "0aee9b78301d7ec8998971363be87c03"
)
```

```
## Hidden Test ##
testthat::expect_equal(
  digest(as.numeric(sampling_distribution)),
  "5e338704a8e069ebd8b38ca71991cf94"
)
```

The Dataset

In this lab, we will explore the population and sampling distributions of one of three different populations of trees planted in Vancouver. To do this, we will use the `vancouver_trees` data set, which includes information about the entire population of public trees planted along boulevards in Vancouver, such as their approximate height, diameter, family and species name, and other information describing where and when they were planted.

This data set is originally from the **City of Vancouver's Open Data Porta**l, but we have included it in an R package called **`datateachr`**. The `datateachr` package contains several open source data sets compiled from various sources to make them easily accessible. Let us take a look at the first few rows of the `vancouver_trees` data set.

```
str(vancouver_trees)
```

```
## tibble [146,611 x 20] (S3: tbl_df/tbl/data.frame)
##  $ tree_id           : num [1:146611] 149556 149563 149579 149590 149604 ...
##  $ civic_number      : num [1:146611] 494 450 4994 858 5032 ...
##  $ std_street        : chr [1:146611] "W 58TH AV" "W 58TH AV" "WINDSOR ST" "E 39TH AV" ...
##  $ genus_name        : chr [1:146611] "ULMUS" "ZELKOVA" "STYRAX" "FRAXINUS" ...
##  $ species_name      : chr [1:146611] "AMERICANA" "SERRATA" "JAPONICA" "AMERICANA" ...
##  $ cultivar_name     : chr [1:146611] "BRANDON" NA NA "AUTUMN APPLAUSE" ...
##  $ common_name       : chr [1:146611] "BRANDON ELM" "JAPANESE ZELKOVA" "JAPANESE SNOWBELL" "AUTUMN A
##  $ assigned          : chr [1:146611] "N" "N" "N" "Y" ...
##  $ root_barrier      : chr [1:146611] "N" "N" "N" "N" ...
##  $ plant_area        : chr [1:146611] "N" "N" "4" "4" ...
##  $ on_street_block   : num [1:146611] 400 400 4900 800 5000 500 4900 4900 4900 700 ...
##  $ on_street         : chr [1:146611] "W 58TH AV" "W 58TH AV" "WINDSOR ST" "E 39TH AV" ...
##  $ neighbourhood_name: chr [1:146611] "MARPOLE" "MARPOLE" "KENSINGTON-CEDAR COTTAGE" "KENSINGTON-CED
##  $ street_side_name  : chr [1:146611] "EVEN" "EVEN" "EVEN" "EVEN" ...
##  $ height_range_id   : num [1:146611] 2 4 3 4 2 2 3 3 2 2 ...
##  $ diameter          : num [1:146611] 10 10 4 18 9 5 15 14 16 7.5 ...
##  $ curb              : chr [1:146611] "N" "N" "Y" "Y" ...
##  $ date_planted      : Date[1:146611], format: "1999-01-13" "1996-05-31" ...
##  $ longitude         : num [1:146611] -123 -123 -123 -123 -123 ...
##  $ latitude          : num [1:146611] 49.2 49.2 49.2 49.2 49.2 ...
```

Exercise 2: The Sampling Distribution of the Mean

To begin, we will look at the population of Acer (labelled `ACER` in the data set) trees planted along streets in Vancouver. The Acer genus (or family) of trees are commonly referred to as maple trees, and there are 31 different species currently planted throughout the city. Maple trees are popular along streets in Vancouver, making up around 25% of all planted trees in the city. They are well known for their bright shades of red, orange, and yellow during the fall and for the appearance of a maple leaf on the Canadian flag.

Figure 1: Acer Tree.

For this exercise, we want to estimate the mean diameter of Acer trees, $\hat{\mu}_{\text{diameter}}$, by drawing a sample of $n = 100$ Acer trees from our data set. The diameter in the data set is in **inches**, but we use the metric system in Canada, so make sure to convert the diameter to **meters**.

Q2.1.

rubric={autograde:2}

Filter for `ACER` trees and their diameters. The output **data frame** `answer2_1` should have only 1 column `diameter`. Make sure to convert to the metric system.

```
BEGIN QUESTION
name: Q2.1
points:
 - 2
```

```
answer2_1 <- NULL

# BEGIN SOLUTION
answer2_1 <- vancouver_trees %>%
  filter(genus_name == "ACER") %>%
  select(diameter) %>%
  mutate(diameter = diameter * 0.0254)
# END SOLUTION

mean(answer2_1$diameter)
```

```
## [1] 0.2694092
```

```
## Test ##
testthat::expect_s3_class(answer2_1, "data.frame")
```

```
testthat::expect_equal(nrow(answer2_1), 36062)
testthat::expect_equal(round(sum(answer2_1$diameter), 1), 9715.4)
```

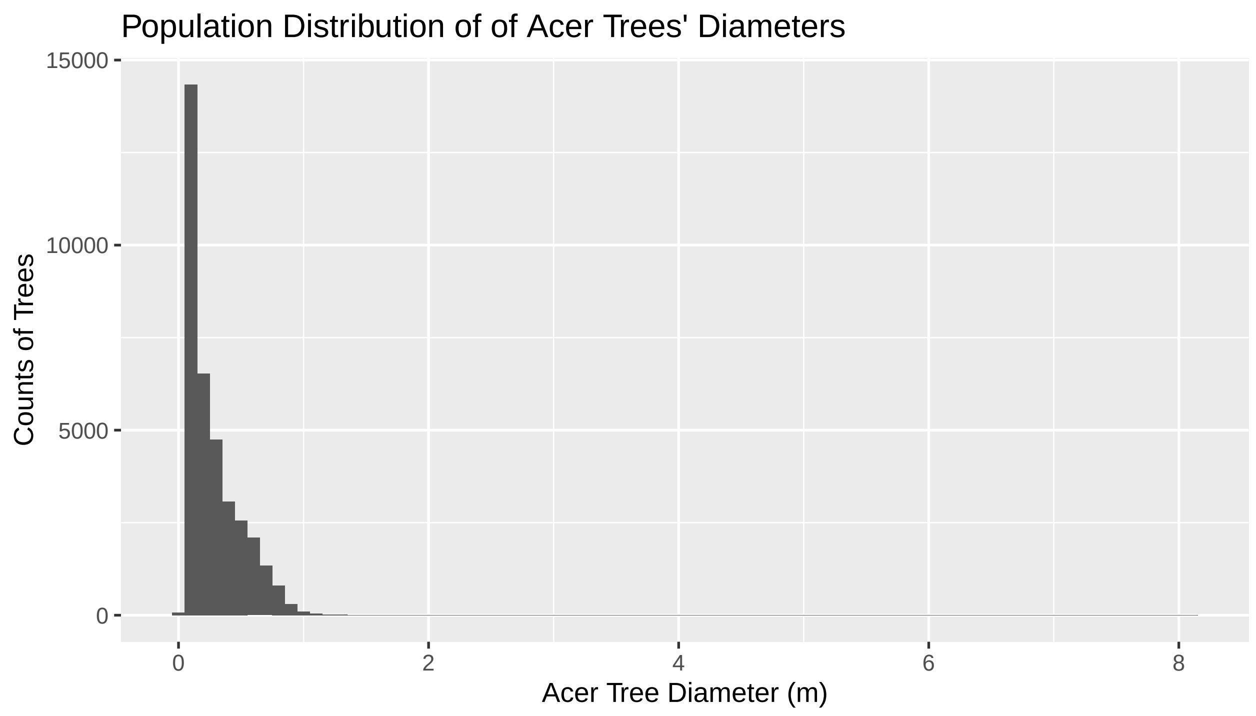
Q2.2.

rubric={autograde:2}

Visualize the population distribution of `ACER` trees' diameters stored in `answer2_1` as a histogram. Ensure that your x and y-axes are human-readable. Moreover, include a title. Assign your plot to an object called `acer_pop_plot`.

```
BEGIN QUESTION
name: Q2.2
points:
 - 0
 - 0.4
 - 0.4
 - 0.4
 - 0.4
 - 0.4
```

```
acer_pop_plot <- NULL

# BEGIN SOLUTION
acer_pop_plot <- answer2_1 %>%
  ggplot(aes(x = diameter)) +
  geom_histogram(binwidth = 0.1) +
  ggtitle("Population Distribution of of Acer Trees' Diameters") +
  xlab("Acer Tree Diameter (m)") +
  ylab("Counts of Trees")
# END SOLUTION

acer_pop_plot
```



Q2.3.

rubric={autograde:3}

Draw a sample of size $n = 100$ from the `ACER` trees population stored in `answer2_1` using appropriate function from the `infer` library. The output data frame `answer2_3` should have two columns:

- `replicate` (which represents the sample number), and
- `diameter` (in meters).

Set seed to `552` so your simulation can be reproducible.

```
BEGIN QUESTION
name: Q2.3
points:
 - 3
```

```
answer2_3 <- NULL

# BEGIN SOLUTION
set.seed(552)

answer2_3 <- rep_sample_n(answer2_1, size = 100)
# END SOLUTION

answer2_3
```

```
## # A tibble: 100 x 2
## # Groups:   replicate [1]
##    replicate diameter
##        <int>    <dbl>
##  1         1   0.318
##  2         1   0.305
##  3         1   0.0762
##  4         1   0.0762
##  5         1   0.0762
##  6         1   0.254
##  7         1   0.178
##  8         1   0.0889
##  9         1   0.190
## 10         1   0.152
## # ... with 90 more rows
```

```
## Test ##
testthat::expect_s3_class(answer2_3, "data.frame")
testthat::expect_equal(nrow(answer2_3), 100)
testthat::expect_equal(round(sum(answer2_3$diameter), 1), 26.5)
```

Q2.4.

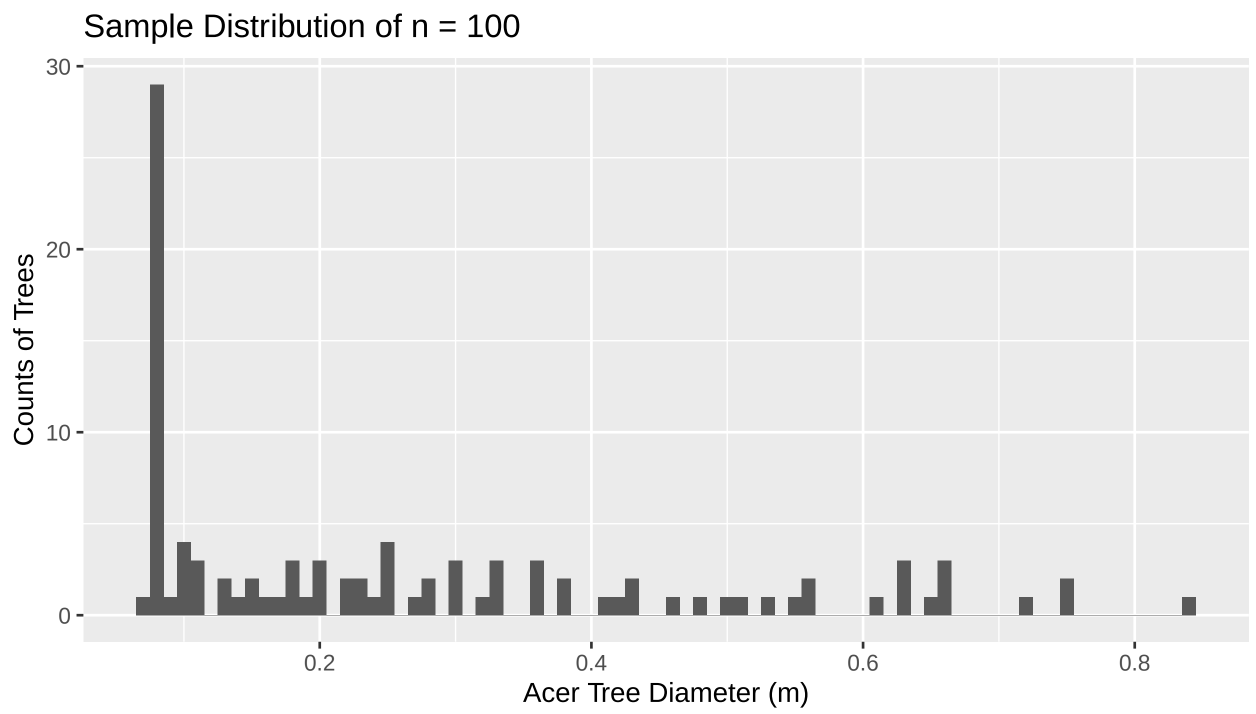
rubric={autograde:2}

Visualize as a histogram the distribution of `ACER` trees' diameters stored in the sample `answer2_3`.

Ensure that your x and y-axes are human-readable. Moreover, include a title. Assign your plot to an object called `single_acer_sample_plot`.

```
BEGIN QUESTION
name: Q2.4
points:
 - 0
 - 0.4
 - 0.4
 - 0.4
 - 0.4
 - 0.4
```

```
single_acer_sample_plot <- NULL

# BEGIN SOLUTION
single_acer_sample_plot <- answer2_3 %>%
  ggplot(aes(x = diameter)) +
  geom_histogram(binwidth = 0.01) +
  ggtitle("Sample Distribution of n = 100") +
  xlab("Acer Tree Diameter (m)")  +
  ylab("Counts of Trees")
# END SOLUTION

single_acer_sample_plot
```



Q2.5.

rubric={autograde:3}

Draw 10,000 samples of size $n = 100$ from the `ACER` trees population stored in `answer2_1`. For each sample, compute the mean diameter. The output data frame `answer2_5` should have two columns:

- `replicate` (which represents the sample number), and
- `mean` (which represents the mean diameter in meters of each sample). Set seed to `552` so your simulation can be reproducible.

```
BEGIN QUESTION
name: Q2.5
points:
 - 3
```

```
answer2_5 <- NULL

# BEGIN SOLUTION
set.seed(552)

answer2_5 <- rep_sample_n(answer2_1, size = 100, reps = 10000) %>%
  group_by(replicate) %>%
  summarise(mean = mean(diameter))
# END SOLUTION

answer2_5
```

```
## # A tibble: 10,000 x 2
##    replicate  mean
##        <int> <dbl>
##  1         1 0.265
##  2         2 0.236
##  3         3 0.238
##  4         4 0.281
##  5         5 0.237
##  6         6 0.267
##  7         7 0.316
##  8         8 0.264
##  9         9 0.279
## 10        10 0.268
## # ... with 9,990 more rows
```

```
## Test ##
testthat::expect_s3_class(answer2_5, "data.frame")
testthat::expect_equal(nrow(answer2_5), 10000)
testthat::expect_equal(round(sum(answer2_5$mean),1), 2693.1)
```

Q2.6.

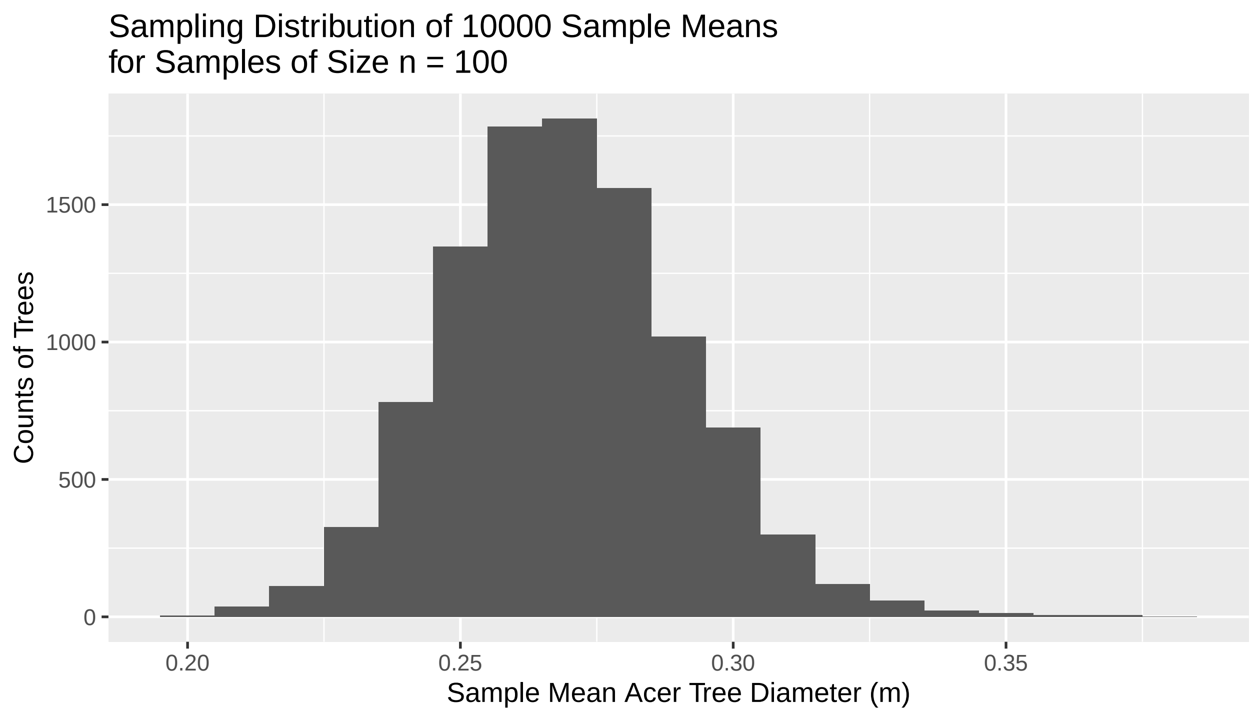
rubric={autograde:2}

Visualize as a histogram the sampling distribution of sample means (stored in `answer2_5`) of `ACER` tree's diameters from 10000 samples of size $n = 100$.

Ensure that your x and y-axes are human-readable. Moreover, include a title. Assign your plot to an object called `sampling_dist_mean_100_plot`.

```
BEGIN QUESTION
name: Q2.6
points:
 - 0
 - 0.5
 - 0.5
 - 0.5
 - 0.5
```

```
sampling_dist_mean_100_plot <- NULL

# BEGIN SOLUTION
sampling_dist_mean_100_plot <- answer2_5 %>%
  ggplot(aes(x = mean)) +
  geom_histogram(binwidth = 0.01) +
  ggtitle("Sampling Distribution of 10000 Sample Means \nfor Samples of Size n = 100") +
  xlab("Sample Mean Acer Tree Diameter (m)") +
  ylab("Counts of Trees")
# END SOLUTION

sampling_dist_mean_100_plot
```

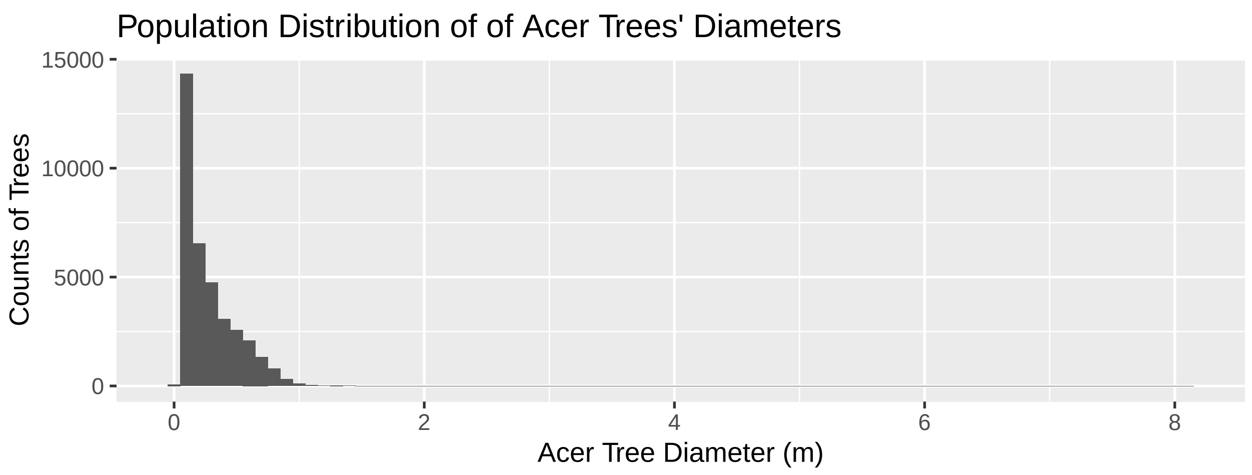


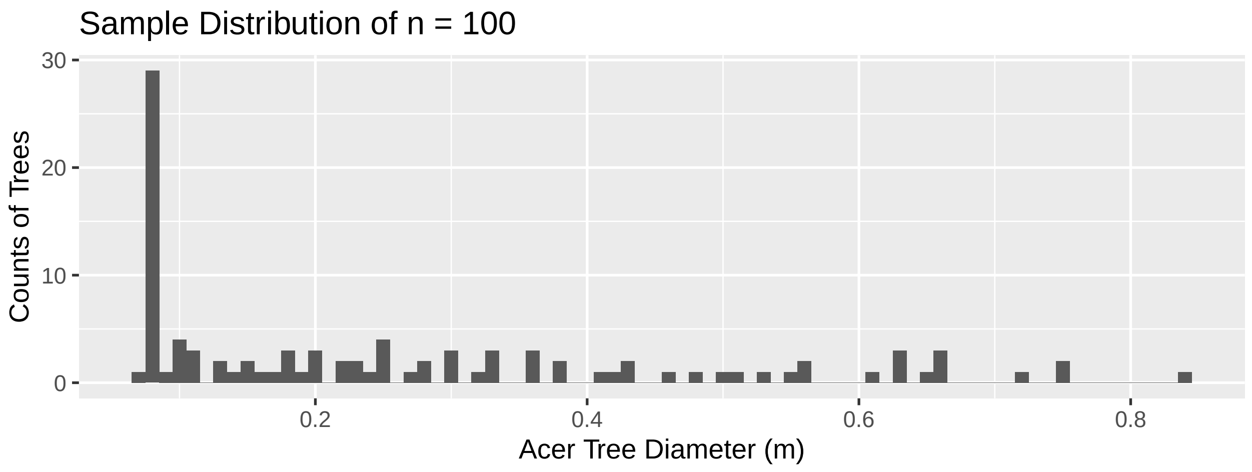
Q2.7.

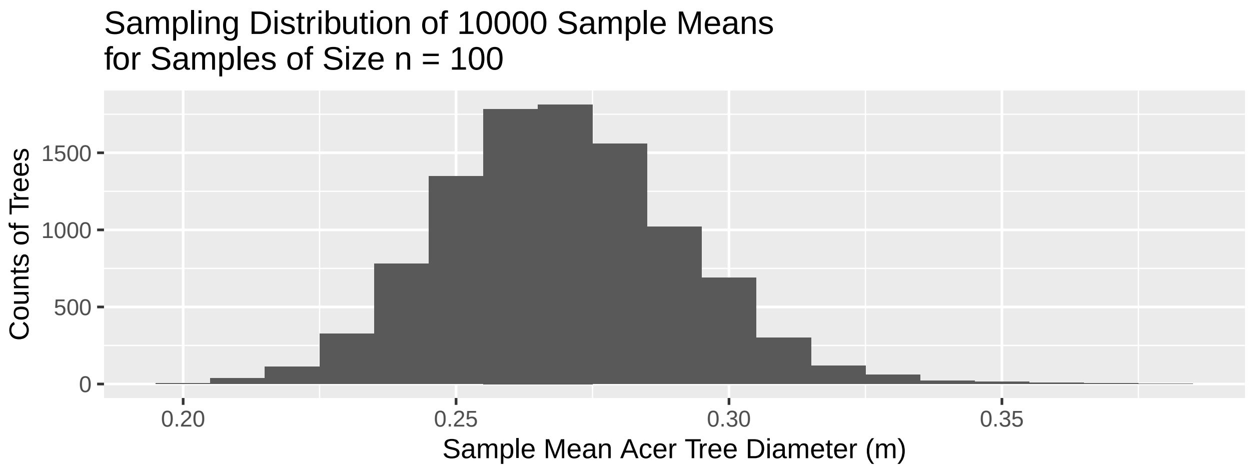
rubric={viz:2}

Combine the three plots above into one graph (you can use `plot_grid()` from package `cowplot`).

```
# BEGIN SOLUTION
plot_grid(acer_pop_plot, single_acer_sample_plot, sampling_dist_mean_100_plot, nrow = 3)
```







```
# END SOLUTION
```

Q2.8.

rubric={reasoning:10}

Describe how these three previous distributions are similar or different and their relationship to each other. Show how the sample mean $\hat{\mu}_{\text{diameter}}$ in `single_acer_sample_plot` behaves in terms of distribution, spread, and center, when compared to the population mean $\mu_{diameter}$ from `acer_pop_plot`. Do the same for the average of the 10000 means from `sampling_dist_mean_100_plot`.

Write between one and three paragraphs.

ANSWER:

The sampling distribution of the 10,000 sample means from `ACER` tree diameters using a size of $n = 100$ is roughly symmetric bell-shaped (actually, it might resemble the form of a normal distribution). However, we can see a slight skew to the right. Its mean is 0.26931.

The population distribution and the distribution of a single random sample of $n = 100$ `ACER` tree diameters are not bell-shaped but heavily skewed to the right. The distribution of a single random sample of 100 `ACER` tree diameters resembles the population distribution to a certain extent. This resemblance between right-skewed distributions might be attributed to the fact that our large size of 100 allows us to obtain a representative enough random sample from the population we are making inference from. The mean of the population distribution is 0.26941, whereas the mean of the single random sample of size $n = 100$ is 0.26542.

As seen in the plots, the population distribution and the distribution of a single random sample of size $n = 100$ are of quite a different shape than the sampling distribution of the 10,000 means coming from samples with a size of $n = 100$. Statistically speaking, the bell-shaped distribution is expected in a set of sample means under a large enough size regardless of the population distribution. However, note the mean of the population distribution is practically the same as the mean of the sampling distribution of the 10,000 means; they are 0.26941 and 0.26931, respectively. The mean of the single random sample of 100 `ACER` tree diameters is not this exact value, but it is close (0.26542). This difference is due to the random noise a mean estimate from a single sample has.

Exercise 3: The Sampling Distribution's Relationship to Sample Size n

We will explore the relationship between sample size n and the shape/spread of the sampling distribution of your estimate. We will use the mean `ACER` tree diameter as our estimate of interest to do this. Using that data, do the following:

Q3.1.

rubric={autograde:5}

Using the population `ACER` diameter data from `answer2_1`, create 3 sampling distributions of the mean by drawing 10,000 samples. Use the sample sizes `10`, `30`, and `100`. For each sampling distribution of the mean, calculate the **mean** and the **standard error** (which is the standard deviation of the sampling distribution of your estimate).

Store the output in a **data frame** called `answer3_1` with the following format:

n	mean	standard_error
10		
30		
100		

You would need to create a function that receives the **population data** and a specific **sample size** as arguments to compute the sample mean and standard deviation **in each one of the 10,000 samples**. Set seed as `552` **as the first line of this function**. Moreover, this function will need to be used **three times** (one per each sample size). Finally, in each of these three times, the function should return the following data frame with 10,000 rows and the following three columns:

- the sample ID number (i.e., `replicate` from `rep_sample_n()`),
- the `mean` computed by sample, and
- the `standard_deviation` computed by sample.

Finally, you can summarize your simulation results in `answer3_1`.

```
BEGIN QUESTION
name: Q3.1
points:
 - 5
```

```r
answer3_1 <- NULL

# BEGIN SOLUTION
#' Create sampling distribution
#'
#' @param data data frame or tibble containing population data
#' @param col unquoted column name containing the population observations
#' @param n numeric vector containing the sample size
#'
#' @return tibble containing three columns: replicate, sample mean, and sample sd
#' @export
#'
#' @examples
#' library(datateachr)
#' create_sampling_dist(apt_buildings, year_built, 50)
```

```r
create_sampling_dist <- function(data, col, n) {
  set.seed(552)
  rep_sample_n(data, n, reps = 10000) %>%
    group_by(replicate) %>%
    summarise(
      mean = mean({{ col }}, na.rm = TRUE),
      standard_deviation = sd({{ col }}, na.rm = TRUE)
    )
}

answer3_1 <- tibble(n = c(10, 30, 100))
answer3_1 <- answer3_1 %>%
  mutate(
    sampling_dist = map(n, ~ create_sampling_dist(answer2_1, diameter, .)),
    mean = map_dbl(sampling_dist, ~ round(mean(.$mean), 5)),
    standard_error = map_dbl(sampling_dist, ~ sd(.$mean))
  )
answer3_1 <- answer3_1 %>%
  select(n, mean , standard_error)
# END SOLUTION

answer3_1
```

```
## # A tibble: 3 x 3
##       n  mean standard_error
##   <dbl> <dbl>          <dbl>
## 1    10 0.270         0.0718
## 2    30 0.269         0.0406
## 3   100 0.269         0.0219
```

```r
## Test ##

# Test data frame shape
testthat::expect_s3_class(answer3_1, "data.frame")
testthat::expect_equal(dim(answer3_1), c(3,3))

# Test means
testthat::expect_equal(
  digest(round((answer3_1 %>%
    filter(n == 10))$mean, 3)),
  "62233ed4e6655a993784e4c0886c4550", 0.01
)
testthat::expect_equal(
  digest(round((answer3_1 %>%
    filter(n == 30))$mean, 3)),
  "2e2181fbd0165bbfda002e3608b01682", 0.01
)
testthat::expect_equal(
  digest(round((answer3_1 %>%
    filter(n == 100))$mean, 3)),
  "2e2181fbd0165bbfda002e3608b01682", 0.01
)

# Test standard errors
```

```r
testthat::expect_equal(
  digest(round((answer3_1 %>%
    filter(n == 10))$standard_error, 3)),
  "c3e33126cea87727bd4c46a59b256ea4", 0.01
)
testthat::expect_equal(
  digest(round((answer3_1 %>%
    filter(n == 30))$standard_error, 3)),
  "e8e37ed54fc14b327440cb6de41e3ac2", 0.01
)
testthat::expect_equal(
  digest(round((answer3_1 %>%
    filter(n == 100))$standard_error, 3)),
  "71b56a9ef95b1d21c0374a8e7efa3070", 0.01
)
```

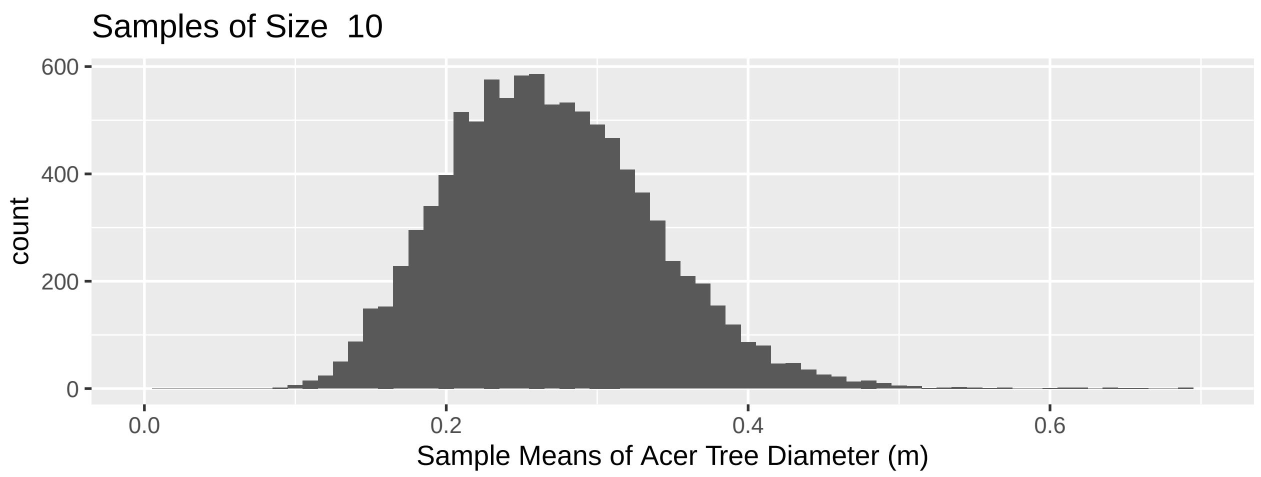
Q3.2.

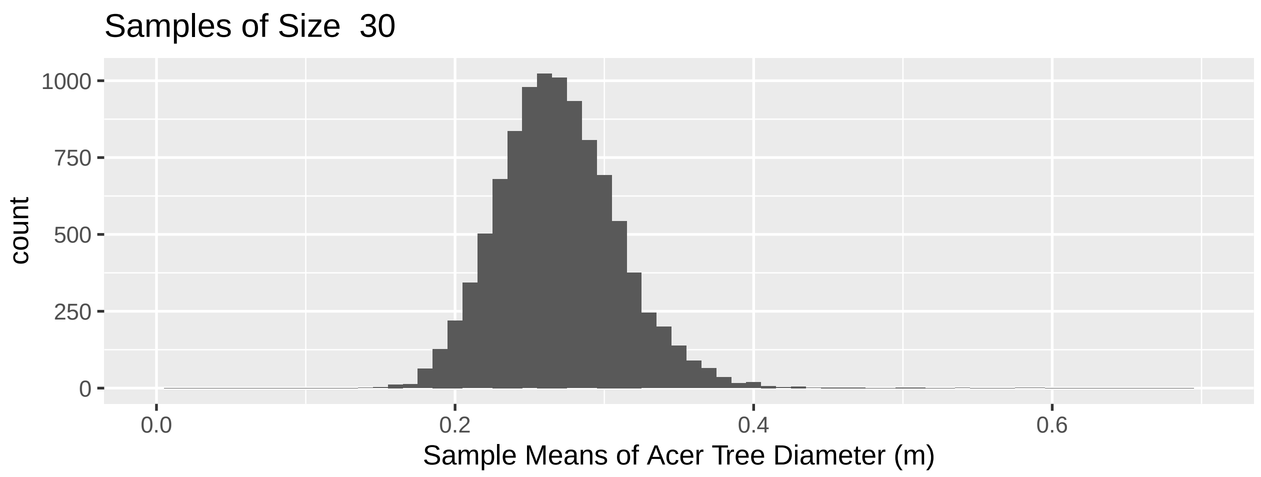
rubric={viz:5}

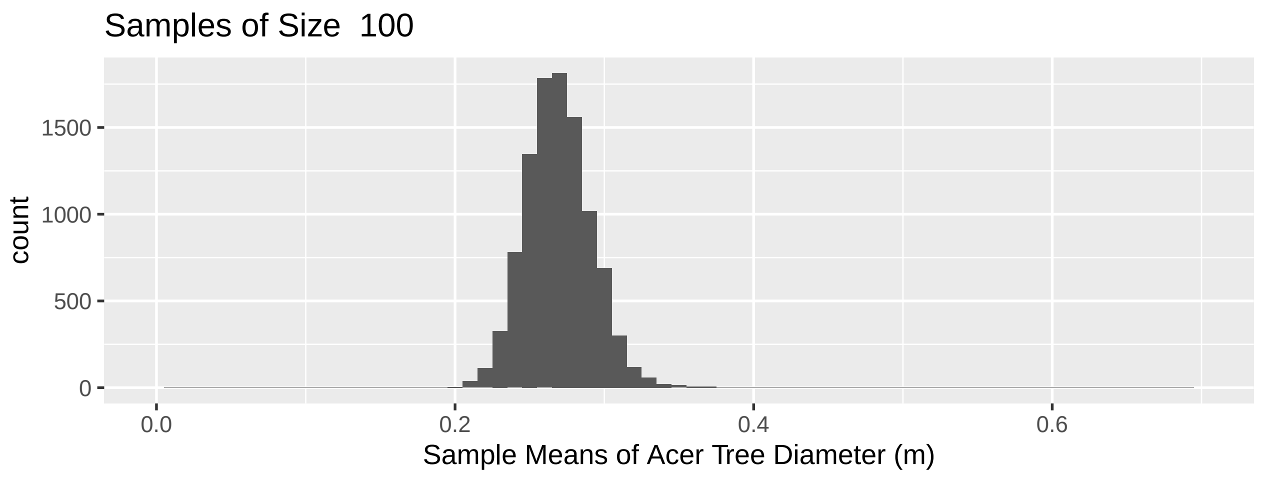
Plot the 3 sampling distributions of the mean one after another (vertically). Store your plots in the variable `sampling_dist_by_n`

Ensure that your x and y-axes are human-readable. Moreover, include a title.

```r
sampling_dist_by_n <- NULL

# BEGIN SOLUTION
#' Create sampling distribution visualization
#'
#' @param data data frame or tibble containing two columns with the column names
#' replicate and estimate
#' @param title character vector to use as the plot's title
#' @param x_label character vector to use as the plot's x label
#' @param bin_width width of bins for histogram, default is 0.01
#'
#' @return ggplot2 object that is a histogram of the sampling distribution of
#' sample estimates
#' @export
#'
#' @examples
#' library(datateachr)
#' create_sampling_dist(apt_buildings, year_built, 50, mean) %>%
#' create_sampling_dist_plot("Sampling distribution \n of
#' sample means \n(n = 50)", "sample means")
create_sampling_dist_plot <- function(data, title, x_label, bin_width = 0.01) {
  data %>%
    ggplot(aes(x = mean)) +
    geom_histogram(binwidth = bin_width) +
    xlim(0, 0.7) +
    ggtitle(title) +
    xlab(x_label)
}

sampling_dist_by_n <- tibble(n = c(10, 30, 100)) %>%
  mutate(
    sampling_dist = map(n, ~ create_sampling_dist(answer2_1, diameter, .)),
    plot = map2(sampling_dist, n, ~ create_sampling_dist_plot(
      .,
      paste("Samples of Size ", .y),
      "Sample Means of Acer Tree Diameter (m)"
    ))
  )

plot_grid(plotlist = sampling_dist_by_n$plot, ncol = 1)
```







```
# END SOLUTION
```

Q3.3.

rubric={accuracy:1,reasoning:5}

Use the `knitr` `kable()` function to create a table to nicely display the sample size, mean, and standard error from `answer3_1`.

```
# BEGIN SOLUTION
kable(select(answer3_1, n, mean, standard_error),
  col.names = c("n", "Mean", "Standard Error")
)
```

n	Mean	Standard Error
10	0.26989	0.0718193
30	0.26926	0.0406214
100	0.26931	0.0218887

```
# END SOLUTION
```

In one paragraph, discuss the impact of changing sample size on the sampling distribution of the mean.

ANSWER:

> *As we increase the sample size n, the spread of the sampling distribution of the mean (as assessed by its standard error) decreases. The table shows us that the empirical expected value (i.e., the mean) of the 10,000 sample means, regardless of the size we are using, is practically equal to the true population mean: 0.26941. In general, each of the three sample sizes has a bell-shaped distribution, but they differ in the magnitude of their respective spread: large for small sample sizes and small for large sample sizes. These sample mean distributions per size allow us to deduce that we are more likely to obtain an estimate closer to the population mean if we have a large enough sample size. In the context of our problem, the smaller the spread in the distribution (as in the sample mean histogram above for the size 100), the more chances we will obtain a single sample mean closer to the true population mean.*

Exercise 4: Distribution of Bootstrap Sample Means

Q4.1.

rubric={autograde:4}

Take your single sample of $n = 100$ `ACER` tree diameters from **Exercise 2** stored in `answer2_3`, and call that your `one_sample` that you collected. Use that `one_sample` for bootstrapping using the corresponding `infer` package function to obtain 10,000 bootstrap samples. Then, calculate the sample means and standard deviations by bootstrap sample.

Store your results in `answer4_1`, which has to be a data frame with 10,000 rows and the following three columns:

- the bootstrap sample ID number (i.e., `replicate` from `rep_sample_n()`),
- the `mean` computed by bootstrap sample, and
- the `standard_deviation` computed by bootstrap sample.

> **Heads-up:** Do not forget to seed your seed to `552`.

```
BEGIN QUESTION
name: Q4.1
points:
 - 4
```

```
answer4_1 <- NULL

# BEGIN SOLUTION
one_sample <- answer2_3 %>%
  ungroup() %>%
  select(diameter)

set.seed(552)
answer4_1 <- one_sample %>%
  rep_sample_n(100, replace = TRUE, reps = 10000) %>%
  group_by(replicate) %>%
  summarise(
    mean = mean(diameter),
    standard_deviation = sd(diameter)
  )
# END SOLUTION

answer4_1
```

```
## # A tibble: 10,000 x 3
##    replicate  mean standard_deviation
##        <int> <dbl>              <dbl>
##  1         1 0.287              0.218
##  2         2 0.258              0.196
##  3         3 0.259              0.240
##  4         4 0.250              0.188
##  5         5 0.279              0.216
##  6         6 0.269              0.193
##  7         7 0.263              0.200
##  8         8 0.287              0.191
##  9         9 0.226              0.178
## 10        10 0.238              0.192
## # ... with 9,990 more rows
## Test ##

# Test data frame shape
testthat::expect_s3_class(answer4_1, "data.frame")
testthat::expect_equal(dim(answer4_1), c(10000, 3))

# Test sample statistics
testthat::expect_equal(
  digest(round(sum(answer4_1$mean), 3)),
  "562bc5ac06cbcbd169e39d62dcaa178a", 0.001
)

testthat::expect_equal(
  digest(round(sum(answer4_1$standard_deviation), 3)),
  "15750fab61fb33ad11f9786e1aa12d4d", 0.001
)
```

Q4.2.

rubric={autograde:2}

Calculate the mean and the standard deviation of the bootstrap distribution stored in `answer4_1`. Bind your results to vectors `bootstrap_mean` and `bootstrap_sd`

```
BEGIN QUESTION
name: Q4.2
points:
 - 2
```

```
bootstrap_mean <- NULL
bootstrap_sd <- NULL

# BEGIN SOLUTION
bootstrap_mean <- round(mean(answer4_1$mean), 3)
bootstrap_sd <- round(sd(answer4_1$mean), 3)
# END SOLUTION

bootstrap_mean
```

```
## [1] 0.265
```

```
bootstrap_sd
```

```
## [1] 0.02
```

```
## Test ##
testthat::expect_equal(
  digest(round(bootstrap_mean, 3)),
  "2ac6c5f62a90852c97faeafdb5af188f", 0.001
)
testthat::expect_equal(
  digest(round(bootstrap_sd, 2)),
  "db643dfa06a42412d2dce382464a83ac", 0.001
)
```

Q4.3.

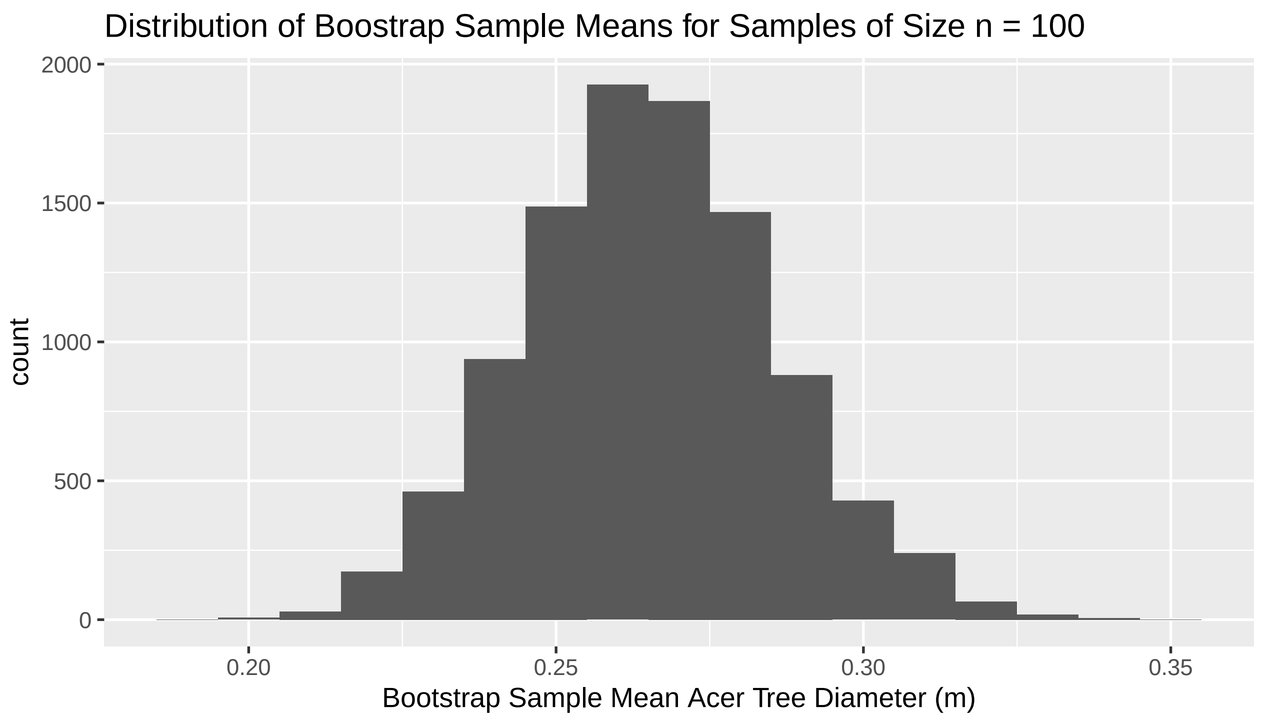
rubric={autograde:2}

Visualize as a histogram the bootstrap distribution of sample means (stored in `answer4_1`) of `ACER` tree's diameters from the 10000 bootstrap samples of size $n = 100$.

Ensure that your x and y-axes are human-readable. Moreover, include a title. Assign your plot to an object called `bootstrap_dist_100_plot`.

```
BEGIN QUESTION
name: Q4.3
points:
 - 0
 - 0.5
 - 0.5
 - 0.5
 - 0.5
```

```
bootstrap_dist_100_plot <- NULL

# BEGIN SOLUTION
 bootstrap_dist_100_plot <- answer4_1 %>%
   ggplot(aes(x = mean)) +
   geom_histogram(binwidth = 0.01) +
   ggtitle("Distribution of Boostrap Sample Means for Samples of Size n = 100") +
   xlab("Bootstrap Sample Mean Acer Tree Diameter (m)")
# END SOLUTION

bootstrap_dist_100_plot
```



Exercise 5: The Relationship Between the Sampling Distribution and the Bootstrapped Sampling Distribution

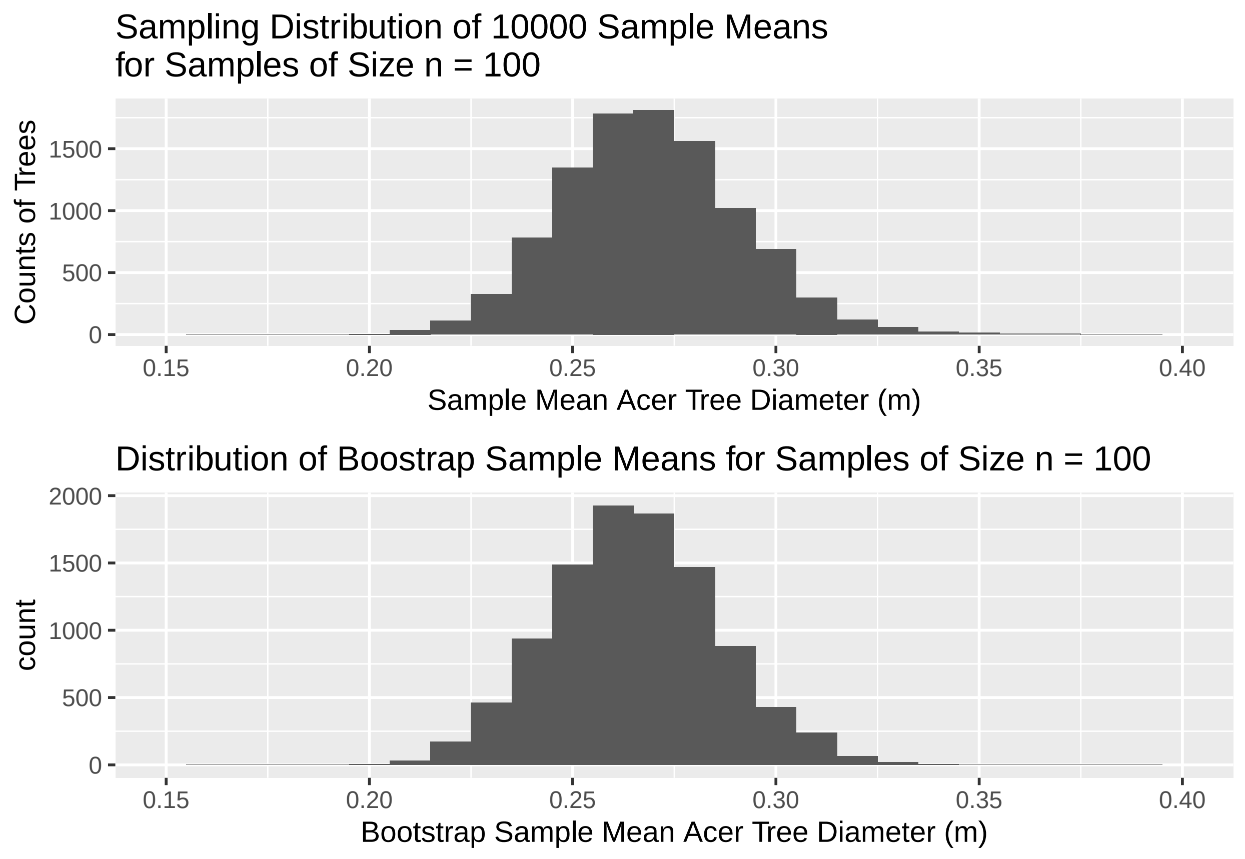
Q5.1.

rubric={accuracy:1,viz:4}

Visualize the distribution of your bootstrap sample mean and the sampling distribution of the mean (from **Exercise 1**) side-by-side. It would be useful to ensure that the x-axis have the same limits in both plots, and display them one-on-top of the other, to clearly see the differences and similarities.

```
# BEGIN SOLUTION
plot_grid(sampling_dist_mean_100_plot + xlim(c(0.15, 0.40)),
  bootstrap_dist_100_plot + xlim(c(0.15, 0.40)),
  ncol = 1
)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
```



```
# END SOLUTION
```

Calculate and report the means and the standard deviations of these two distributions (sampling distribution in `answer2_5` and bootstrap distribution in `answer4_1`) in a table using `knitr`'s `kable()` function.

```
# BEGIN SOLUTION
bootstrap_dist_100 <- answer4_1 %>%
  mutate(distribution = "Distribution of Boostrap Sample Means")

sampling_dist_mean_100 <- answer2_5 %>%
  mutate(distribution = "Sampling Distribution of Sample Means")

full_join(bootstrap_dist_100, sampling_dist_mean_100) %>%
  group_by(distribution) %>%
  summarise(
    dist_mean = mean(mean),
    standard_deviation = sd(mean)
  ) %>%
  kable(col.names = c("Distribution", "Mean", "Standard deviation"))
```

Distribution	Mean	Standard deviation
Distribution of Boostrap Sample Means	0.2653380	0.0204671
Sampling Distribution of Sample Means	0.2693146	0.0218887

```
# END SOLUTION
```

Q5.2.

rubric={reasoning:10}

In two or three paragraphs, discuss the similarities and differences between these distributions. Finally, given that you generally will not know the sampling distribution nor have multiple samples to create the sampling distribution histogram, comment on how a bootstrap distribution might be helpful for estimating a population parameter.

ANSWER:

> *At first glance, both of these distributions look very similar. They are both bell-shaped and have a similar spread. This is because the spread of the bootstrap distribution of bootstrap sample means is similar to the spread of the sampling distribution of the sample means because we used a bootstrap sample size that was the same as our original sample size.*
>
> **(*Not required in the students' solutions, but useful for their understanding when reviewing the solutions:* If the bootstrap sample size were larger than the original sample size, you would underestimate the spread of your sampling distribution. And if you made it smaller than your sample size, you would overestimate it. This is because the empirical sample distribution is an estimate of the population distribution.)**
>
> *The means (bootstrap and sampling distribution) are pretty similar. Moreover, the mean of the sampling distribution differs from the mean of the population 0.26941. In this case, we empirically see that our base sample needs to be representative enough of the population of interest. If not, our bootstrap distribution's accuracy will be as bad as the base sample's.*
>
> **(*Not required in the students' solutions, but useful for their understanding when reviewing the solutions:* As we increase our base sample size from which we are drawing our bootstrap samples with replacement, the bootstrap distribution will center around the base sample mean. And as the base sample mean centers around the population mean we aim to estimate, so will the bootstrap distribution sample mean. Recall that the larger the size is in our single sample, drawn from the population, the more representative it is.)**
>
> *Given that both distributions are very similar in shape and spread, it seems reasonable to use the bootstrap distribution to suggest a plausible range for our estimate when doing estimation in statistical inference (as, in reality, we will never have a sampling distribution and cannot use that). Note that the bootstrap sampling will greatly benefit from a large enough base sample.*

(Challenging) Exercise 6: DRY

Q6.1.

rubric={accuracy:5}

In **Exercise 3**, you might likely violate the DRY programming principle (**D**o not **R**epeat **Y**ourself). Write a function that takes the following parameters:

- `data`: the population data (e.g., the `ACER` tree),
- `col`: the column in the data frame (e.g., the diameters).
- `n`: a number vector specifying different sizes (e.g., `c(10,30,100)`).

This function should return a list with two elements:

1. `plot`: a `ggplot2` object (created using `plot_grid` that combines all the histograms into a panel plot).
2. `df`: a data frame containing the sample size `n`, means `mean` and standard error `standard_error` for each sample size given by the user. The output should look similar to **Q3.1.**

Note this **main function** could use **other auxiliary functions within itself**, if you think that is reasonable.

Include the corresponding docstrings for your function(s).

> **Heads-up:** Remember to `set.seed(552)` in your given sampling function.

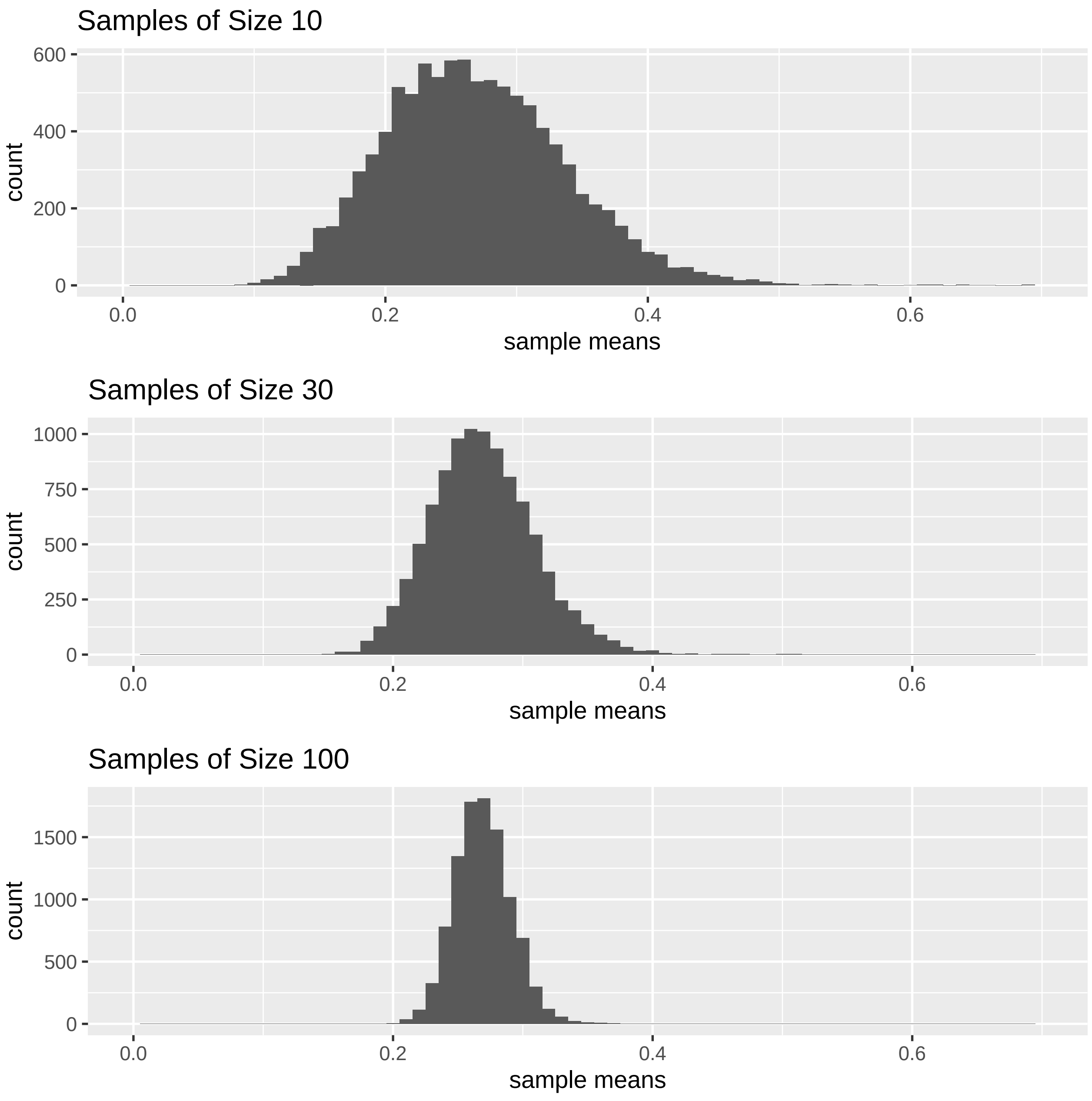
```r
compare_sample_size <- function(data, col, n) {

  # Remember to set.seed(552)
  plot <- NULL
  df <- NULL

  # BEGIN SOLUTION
  #' Create sampling distribution
  #'
  #' @param data data frame or tibble containing population data
  #' @param col unquoted column name containing the population observations
  #' @param n numeric vector containing the sample size
  #'
  #' @return tibble containing two columns, replicate and estimate
  #' @export
  #'
  #' @examples
  #' library(datateachr)
  #' create_sampling_dist(apt_buildings, year_built, 50, mean)
  create_sampling_dist <- function(data, col, n) {
    set.seed(552)
    rep_sample_n(data, n, reps = 10000) %>%
      group_by(replicate) %>%
      summarise(estimate = mean({{ col }}, na.rm = TRUE))
  }

  #' Create sampling distribution visualization
  #'
  #' @param data data frame or tibble containing two columns with the
  #' column names replicate and estimate
  #' @param title character vector to use as the plot's title
  #' @param x_label character vector to use as the plot's x label
  #' @param bin_width width of bins for histogram, default is 0.01
  #'
  #' @return ggplot2 object that is a histogram of the sampling distribution
  #' of sample estimates
  #' @export
  #'
  #' @examples
  #' library(datateachr)
  #' create_sampling_dist(apt_buildings, year_built, 50, mean) %>%
  #'   create_sampling_dist_plot("Sampling
  #'   Distribution \n of Sample Means \n(n = 50)", "sample means")
  create_sampling_dist_plot <- function(data, title, x_label, bin_width = 0.01) {
    data %>%
      ggplot(aes(x = estimate)) +
      geom_histogram(binwidth = bin_width) +
      xlim(0, 0.7) +
      ggtitle(title) +
      xlab(x_label)
  }

  #' Compare sample distributions across sample sizes
  #'
  #' @param data data frame or tibble containing population data
  #' @param col unquoted column name containing the population observations
  #' @param sample_sizes numeric vector containing the sample sizes
  #'
  #' @return list with two elements: 1) a cowlplot object
  #' created using `plot_grid` that contains the three histograms as a panel plot,
  #' and 2) a data frame containing the sample size, means and
  #' standard deviation for each sample size given by the user.
  #' @export
  #'
  #' @examples

  # Create data frame to build up
  sample_size_explore <- tibble(n = n)

  # Create sampling distributions, visualizations and calculate summary stats
  sample_size_explore <- sample_size_explore %>%
    mutate(
      sampling_dist = map(n, ~ create_sampling_dist(data, {{ col }}, .)),
      mean = map_dbl(sampling_dist, ~ round(mean(.$estimate), 5)),
      standard_error = map_dbl(sampling_dist, ~ sd(.$estimate)),
      plot = map2(sampling_dist, n, ~ create_sampling_dist_plot(
        .,
        paste("Samples of Size", .y),
        "sample means"
      ))
    )

  # Create list object to return
  plot <- plot_grid(plotlist = sample_size_explore$plot, ncol = 1)
  df <- select(sample_size_explore, n, mean, standard_error)
```

```
  # END SOLUTION
  return(list(plot, df))
}

# Run to test your function
compare_sample_size(answer2_1, diameter, c(10, 30, 100))
```

```
## Warning: Removed 7 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
```

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



```
##
## [[2]]
## # A tibble: 3 x 3
##       n  mean standard_error
##   <dbl> <dbl>          <dbl>
## 1    10 0.270         0.0718
## 2    30 0.269         0.0406
## 3   100 0.269         0.0219
```

```
# This should return a data frame similar to answer3_1
compare_sample_size(answer2_1, diameter, c(10, 30, 100))[[2]]
```

```
## Warning: Removed 7 rows containing non-finite values (stat_bin).
## Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
```

```
## # A tibble: 3 x 3
##       n  mean standard_error
##   <dbl> <dbl>          <dbl>
## 1    10 0.270         0.0718
## 2    30 0.269         0.0406
## 3   100 0.269         0.0219
```

Q6.2.

rubric={accuracy:3}

Demonstrate that your function works by writing unit tests using testthat. Here are some example test cases that should be written:

1. The object being returned is a list with two elements, the first is a `cowplot` object (i.e., a grid of plots), and the second is a tibble or data frame.
2. The data frame should have 3 columns: `n`, `mean`, `standard_error` and the number of rows should be equal to the length of vector `sample_sizes`.

3. The standard errors for larger `n`'s should get smaller.

4. The standard errors should be smaller than the means.

```
# BEGIN SOLUTION
data <- vancouver_trees %>%
  filter(genus_name == "ACER") %>%
  select(diameter) %>%
  mutate(diameter = diameter * 0.0254)

sample_sizes <- c(10, 30, 100)

answer6_2 <- compare_sample_size(data, diameter, sample_sizes)
```

```
## Warning: Removed 7 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
```

```
# Test if the first element of the output is a ggplot2
testthat::expect_true("ggplot" %in% class(answer6_2[[1]]))
```

```r
# Test if the second element of the output is a data frame
testthat::expect_s3_class(answer6_2[[2]], "data.frame")

# Test if `n`, `mean`, `standard_error` are the column names
testthat::expect_setequal(c("n", "mean", "standard_error"), colnames(answer6_2[[2]]))

# Test if as `n` increases, `standard_error` decreases
sort_df <- answer6_2[[2]][order(answer6_2[[2]]$n), ]
testthat::expect_true(all(diff(sort_df$n) > 0) && all(diff(sort_df$standard_error) < 0))

# Test if standard_error smaller than mean
testthat::expect_false("FALSE" %in% (answer6_2[[2]]$standard_error < answer6_2[[2]]$mean))
# END SOLUTION
```

Submission

Congratulations! You are done the lab!!! Do not forget to:

- Knit the assignment to generate `.pdf` file and push everything to your GitHub repo.
- Submit the `.Rmd` AND the `.pdf` files to Gradescope.