DSCI 552 Lab 2

Introduction to Confidence Intervals and Hypothesis Testing

Contents

Lab Mechanics

rubric={mechanics:5}

- Paste the URL to your GitHub repo here: **https://github.ubc.ca/MDS-2022-23/DSCI_552_lab2_kewang5**
- 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(infer)
library(palmerpenguins)
library(testthat)
library(digest)
library(tidyverse)
library(knitr)
library(cowplot)
library(datateachr)
```

Exercise 1: Median Salary in Vancouver

The `data/salary.csv` data set includes remuneration and expenses from $n = 3003$ employees earning over CAD $75,000/year. This data was retrieved from the **Vancouver Open Data Portal**.

Q1.1.

rubric={reasoning:5}

Imagine you are a data scientist working for the Vancouver City Council. **They want to make an inference on the median salary of all employees in Vancouver who earn over $75,000/year.** Population mean is off the table, given its sensitivity to outliers that are likely to appear in such an unequal city.

The city has collected a small sample (stored in `salary.csv`) of size $n = 3003$. Explain to the Vancouver City Council how you would estimate a bootstrapped 90% confidence interval (CI) of the median salary in Vancouver from this sample.

Write down the steps in written English such as:

Step 1: From the raw data, extract only those rows with `salary > $75,000` and use this dataset as the base sample.

Step 2: Bootstrap from the above base sample data for a large number of times (say `replication = 1000` for example) to get the sampling distribution of the median.

Step 3: From the above sampling distribution, get 3 values `point estimate`, `CI upper bound` and `CI lower bound`, by finding the `50%`, `95%` and `5%` quantile values respectively.

Q1.2.

rubric={autograde:7}

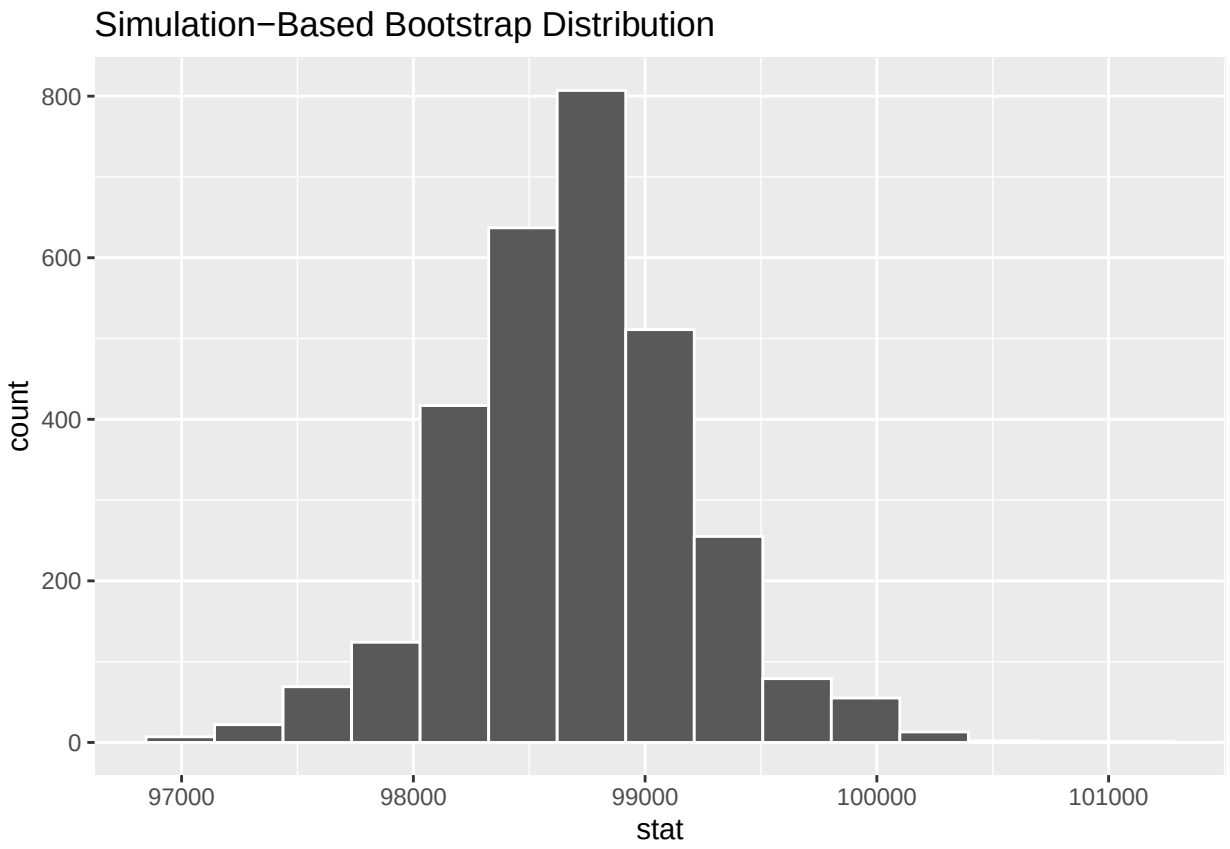
Using the sample `salary`, compute the 90% CI of the median salary of all employees in Vancouver who earn over $75,000/year.

```
salary <- read_csv("data/salary.csv")
salary_group_0 <- salary |>
  mutate(salary_value = Remuneration) |>
  select(salary_value) |>
  filter(salary_value > 75000)
salary_group_0
```

```
## # A tibble: 3,003 x 1
##    salary_value
##           <dbl>
##  1       105216
##  2       102798
##  3        88522
##  4        90366
##  5        84551
##  6       104724
##  7       167451
##  8        79257
##  9        81094
## 10       104033
## # ... with 2,993 more rows
```

```
set.seed(552) # For reproducibility.
bootstrap_salary <- salary_group_0 |>
```

```
  specify(response = salary_value) |>
  generate(reps = 3000, type = "bootstrap") |>
  calculate(stat = "median")
visualise(bootstrap_salary)
```




```
bootstrap_salary
```

```
## Response: salary_value (numeric)
## # A tibble: 3,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1 98691
##  2         2 98616
##  3         3 98457
##  4         4 98172
##  5         5 98557
##  6         6 99980
##  7         7 98858
##  8         8 98648
##  9         9 99012
## 10        10 98557
## # ... with 2,990 more rows
```

You can use the `get_confidence_interval()` function from the `infer` package. Note the following:

- Use seed `552` when appropriate for reproducibility.
- Use 3,000 bootstrap samples.

Store the lower bound in the vector `answer1_2_lower_ci` and the upper bound in `answer1_2_upper_ci`.

```
salary_ci_90 <- bootstrap_salary |>
  get_confidence_interval(level = 0.90, type = "percentile")
salary_ci_90
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1    97836    99510
```

```
answer1_2_lower_ci <- salary_ci_90$lower_ci
answer1_2_upper_ci <- salary_ci_90$upper_ci

answer1_2_lower_ci
```

```
## [1] 97836
```

```
answer1_2_upper_ci
```

```
## [1] 99510
```

```
. = ottr::check("tests/Q1.2.R")
```

```
##
## All tests passed!
```

Q1.3.

rubric={viz:4}

Plot your 3000 bootstrap sample medians as a histogram. Include the 90% bootstrap CI bounds along with the sample median. You could use `ggplot2`'s function `geom_vline()`.

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

Heads-up: This plot does not have auto grading tests.

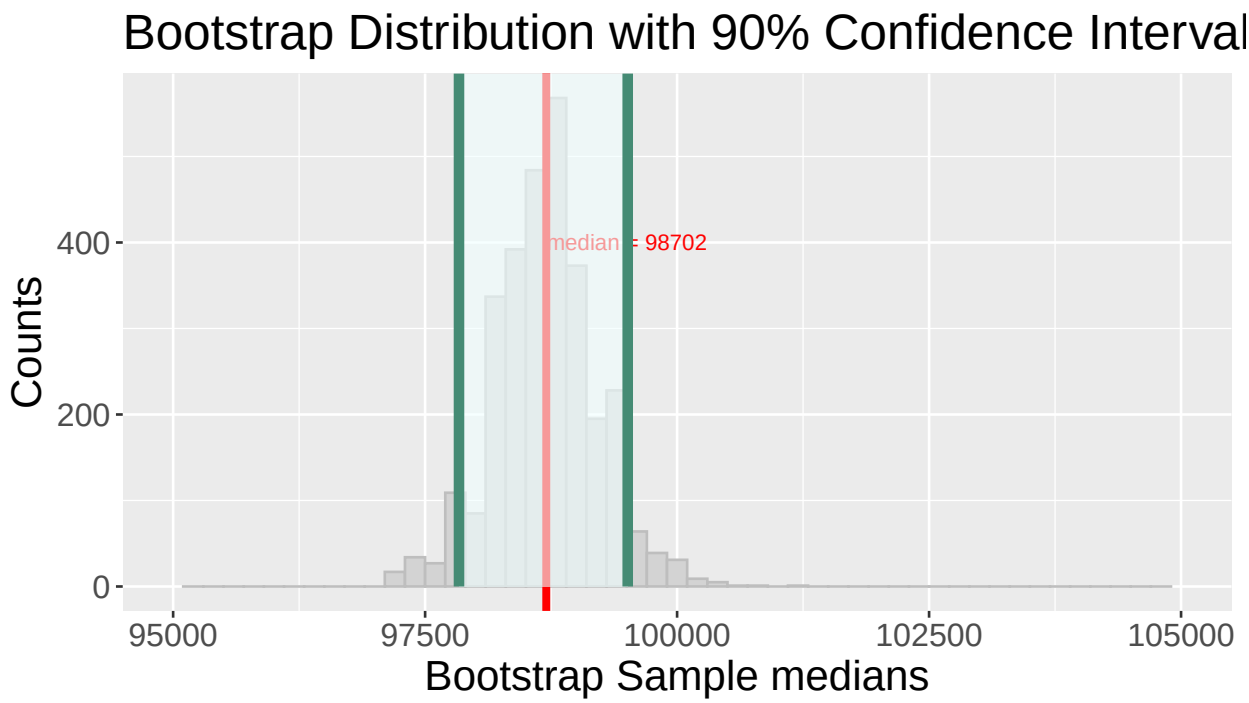
```
median_median_salary <- median(bootstrap_salary$stat)
median_median_salary
```

```
## [1] 98702
```

```
bootstrap_dist_plot_salary <- bootstrap_salary |>
  ggplot(aes(x = stat)) +
  geom_histogram(binwidth = 200, color = "gray", fill = "lightgray") +
  xlim(95000, 105000) +
  labs(x = "Bootstrap Sample medians", y = "Counts") +
  ggtitle("Bootstrap Distribution") +
  theme(text = element_text(size = 16.5)) +
  geom_vline(xintercept = median_median_salary, color = "red", size = 1.5) +
  annotate("text", x = median_median_salary + 800 , y = 400, label = "median = 98702", size = 3, color =
```

```
boot_CI_median_plot <- bootstrap_dist_plot_salary +
  shade_confidence_interval(endpoints = salary_ci_90, color = "aquamarine4", fill = "azure") +
  ggtitle("Bootstrap Distribution with 90% Confidence Interval")
boot_CI_median_plot
```

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


Bootstrap Distribution with 90% Confidence Interval
median = 98702
Counts
400
200
0
95000
97500
100000
102500
105000
Bootstrap Sample medians

Exercise 2: The Canadian 44th Election for Prime Minister

Canada's 44th election for Prime Minister took place last year. Throughout the electoral campaign, we saw many polls carried out to estimate the proportion of votes each party would get **based on a randomly selected sample of eligible voters**. This exercise will use **the September 18th, 2021, poll reported by the Angus Reid Institute (ARI)** that asked $n = 2,042$ people which party they intended to vote for.

> **Note:** The ARI's report states that "*discrepancies in or between totals are due to rounding.*" Hence, the corresponding estimated percentages per party will not add up to 100%.

Let us create a synthetic data set `polls` that resembles the results of the poll by running the code cell below:

```
# Poll's sample size
n <- 2042

# Party vector
party <- c("CPC", "Liberals", "NDP", "Bloc", "PPC", "Green", "Others")

# Setting up sampled voting proportions
# We make an adjustment on Others to 0.0302, instead of 0.01 as in the
# ARI's report, so the other estimated proportions in our computational
# analysis will match.
prop <- c(0.32, 0.3, 0.2, 0.07, 0.05, 0.03, 0.0302)

# Voting counts
votes <- round(n * prop, 0)

# Creating data
polls <- data.frame(party = rep(party, votes))
head(polls)
```

```
##   party
## 1   CPC
## 2   CPC
## 3   CPC
## 4   CPC
## 5   CPC
## 6   CPC
```

```
tail(polls)
```

```
##       party
## 2037 Others
## 2038 Others
## 2039 Others
## 2040 Others
## 2041 Others
## 2042 Others
```

We will be extremely inquisitive with these polling results. Hence, let us check the ARI's methodology:

> *The Angus Reid Institute conducted an online survey from Sept. 15 – 18, 2021 among a representative randomized sample of 2,042 Canadian adults who are members of **Angus** Reid Forum. For comparison purposes only, a probability sample of this size would carry a margin of error of +/- 2.2 percentage points, 19 times out of 20. Discrepancies in or between totals are due to rounding. The survey was self-commissioned and paid for by ARI.*

The summary of this methodology implicates a classical approach to compute the uncertainty of the polling

estimates (i.e., their sampling distribution) using a classical approach via what they call the *margin of error* (**check this helpful resource**). The expression "*19 times out of 20*" implies a 95% confidence (i.e., $\frac{19}{20} \times 100\% = 95\%$).

This classical approach implies **Normal assumptions when** $n \to \infty$ under the frequentist paradigm. Nevertheless, with an online survey of size $n = 2,042$ from a population of millions of voters, **are we entirely sure we can use these classical assumptions?**

Let us try the computational approach!

Q2.1.

rubric={autograde:10}

Your task as an inquisitive Data Scientist in inferential matters is to use the sampled data in `polls` to estimate the voter population proportion parameters per party, p_{party_k}, which is the proportion of Canadian population who intend to vote for each of the k parties in the data set `polls` (**at the time the poll was ran!**). This will yield the following estimate:

$$\hat{p}_{\text{party}_k} = \frac{n_k}{n},$$

where n_k is the number of respondents who expressed their electoral sympathy for the kth party in the sample `polls` and n is the overall sample size.

Furthermore, compute the **95% bootstrap CIs** of the **estimated** voting proportion $\hat{p}_{\text{party}_k}$ of each party, using **15,000 bootstrapped samples**. This will give you the corresponding uncertainty measures **without using margin of errors via Normal approximations**. Use seed `552` when appropriate for reproducibility.

The output data frame `answer2_1` should look like below.

Table 1: Empty `answer2_1`.

party	lower_ci	upper_ci	p_hat
Liberals			
CPC			
NDP			
Bloc			
PPC			
Green			
Others			

Heads-up: Round up all your **final** table results to 2 decimal places.

```
base_sample_props <- polls |>
  group_by(party) |>
  summarise(prop = round(n()/n, 2)) |>
  column_to_rownames("party")
base_sample_props
```

```
##          prop
## Bloc     0.07
## CPC      0.32
## Green    0.03
## Liberals 0.30
## NDP      0.20
## Others   0.03
```

```
## PPC      0.05
```

```
set.seed(552) # For reproducibility.
bootstrap_vote <- polls |>
  specify(response = party) |>
  generate(reps = 15000, type = "bootstrap") |>
  group_by(replicate, party) |>
  summarise(vote_prop = n()/2042) |>
  ungroup() |>
  select(party, vote_prop)
```

```
## `summarise()` has grouped output by 'replicate'. You can override using the
## `.groups` argument.
```

```
bootstrap_vote
```

```
## # A tibble: 105,000 x 2
##    party    vote_prop
##    <fct>        <dbl>
##  1 Bloc        0.0818
##  2 CPC         0.324
##  3 Green       0.0309
##  4 Liberals    0.293
##  5 NDP         0.193
##  6 Others      0.0240
##  7 PPC         0.0534
##  8 Bloc        0.0686
##  9 CPC         0.320
## 10 Green       0.0299
## # ... with 104,990 more rows
```

```
lower_ci <- c(0,0,0,0,0,0,0)
upper_ci <- c(0,0,0,0,0,0,0)
p_hat <- c(0,0,0,0,0,0,0)
result_table <- data.frame(party, lower_ci, upper_ci, p_hat)
rownames(result_table) <- result_table$party
result_table
```

```
##             party lower_ci upper_ci p_hat
## CPC           CPC        0        0     0
## Liberals Liberals        0        0     0
## NDP           NDP        0        0     0
## Bloc         Bloc        0        0     0
## PPC           PPC        0        0     0
## Green       Green        0        0     0
## Others     Others        0        0     0
```

```
get_ci_for_party <- function(data, party_name) {
  this_party_props <- data |>
    filter(party == party_name) |>
    select(vote_prop)
  this_party_props$vote_prop |>
    quantile(probs = c(0.025, 0.975)) |>
    round(2)
}

for(i in 1:length(party)) {
```

```
  party_name <- party[i]
  party_ci_boundaries <- get_ci_for_party(bootstrap_vote, party_name)
  my_lower_bound <- party_ci_boundaries[[1]]
  my_upper_bound <- party_ci_boundaries[[2]]
  my_p_hat <- base_sample_props[party_name, "prop"]
  result_table[party_name, "lower_ci"] <- my_lower_bound
  result_table[party_name, "upper_ci"] <- my_upper_bound
  result_table[party_name, "p_hat"] <- my_p_hat
}
result_table
```

```
##             party lower_ci upper_ci p_hat
## CPC           CPC     0.30     0.34  0.32
## Liberals Liberals     0.28     0.32  0.30
## NDP           NDP     0.18     0.22  0.20
## Bloc         Bloc     0.06     0.08  0.07
## PPC           PPC     0.04     0.06  0.05
## Green       Green     0.02     0.04  0.03
## Others     Others     0.02     0.04  0.03
```

```
answer2_1 <- result_table

# YOUR CODE HERE

answer2_1
```

```
##             party lower_ci upper_ci p_hat
## CPC           CPC     0.30     0.34  0.32
## Liberals Liberals     0.28     0.32  0.30
## NDP           NDP     0.18     0.22  0.20
## Bloc         Bloc     0.06     0.08  0.07
## PPC           PPC     0.04     0.06  0.05
## Green       Green     0.02     0.04  0.03
## Others     Others     0.02     0.04  0.03
```

```
. = ottr::check("tests/Q2.1.R")
```

```
##
## All tests passed!
```

Q2.2.

rubric={viz:6}

Visualize the estimate, $\hat{p}_{party_k}$, and the 95% bootstrap CIs on a plot, using a bar plot with **error bars**. You should map the k parties to the x-axis and the proportion estimates and CIs as **error bars** to the y-axis. The order of the bars has to be decreasing by $\hat{p}_{\text{party}_k}$ from left to right.

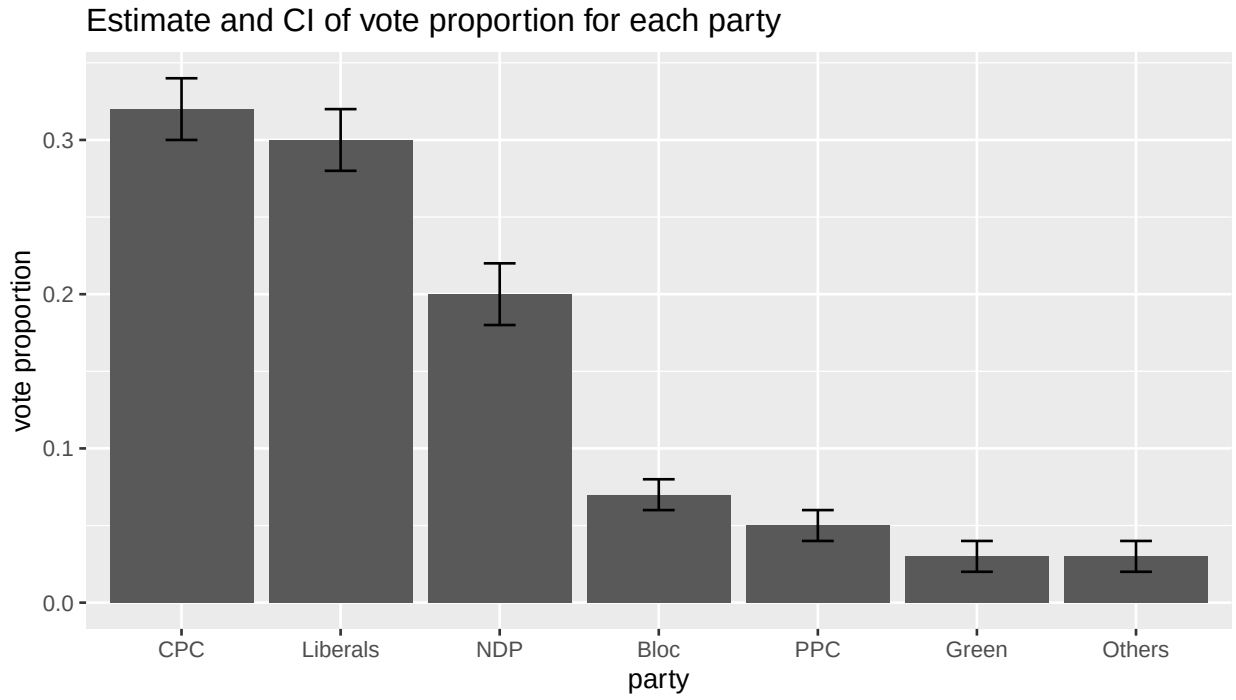
Ensure that your x and y-axes are human-readable. Moreover, include a title. Assign your plot to the variable `party_bar_plot`.

> **Hint:** Use `geom_bar(stat = "identity")` from `ggplot2`.
>
> **Heads-up:** There are no auto-grading test functions for this question. Furthermore, the error bars represent the bounds of your 95% bootstrap CIs per party.

```
party_bar_plot <- ggplot(
  result_table,
```

```
  aes(x = reorder(party, -p_hat), y = p_hat)
) +
geom_bar(stat = "identity") +
geom_errorbar(
  data = result_table,
  aes(
    ymin = lower_ci,
    ymax = upper_ci
  ),
  width = .2,
  position = position_dodge(.9)
) +
ggtitle("Estimate and CI of vote proportion for each party") +
labs(x = "party", y = "vote proportion")

party_bar_plot
```



Q2.3.

rubric={reasoning:5}

In one or two paragraphs, use written English to interpret and report the estimates (and their bootstrap 95% CIs) for each of the k parties. **here** as an example.

According to the poll reported by ARI on Sep 18th 2021, in Canada's 44th election for Prime Minister:

- It is estimated that 32% of the voters would vote for Conservative Party of Canada(CPC) and we are 95% confident that the true proportion of people who will vote for CPC is somewhere between 30% and 34%.
- Similar interpretations for other parties ...

Q2.4 No Attempt

No Attempt to the question so I deleted this part.

Q2.5 No Attempt

No Attempt to the question so I deleted this part.

Exercise 3: Bootstrapping and Penguins

Penguin researchers have hired you to help estimate **the population mean body size in grams (g)** of penguins from three different species: *Pygoscelis papua* (Gentoo), *Pygoscelis adeliae* (Adelie) and *Pygoscelis antarcticus* (Chinstrap). The researchers have told you they are defining the mean body size of penguins by the mean body mass for a given penguin species.

To do your inferential analysis, you will use the `penguins` data set from the `palmerpenguins R` package. Run the code chunk below to work with a copy called `penguins_sample` of **sample size** $n = 342$. Note we discard two penguins with missing data via `drop_na()`.

```
penguins_sample <- penguins |>
  select(species, body_mass_g) |>
  drop_na()
nrow(penguins_sample)
```

```
## [1] 342
```

```
penguins_base_sample_mean <- penguins_sample |>
  group_by(species) |>
  summarise(mean_mass = round(mean(body_mass_g), 2)) |>
  column_to_rownames("species")
penguins_base_sample_mean
```

```
##           mean_mass
## Adelie      3700.66
## Chinstrap   3733.09
## Gentoo      5076.02
```

```
ci_mean <- function(base_sample_data, species_name) {
  set.seed(552)
  penguins_sample |>
    filter(species == species_name) |>
    specify(response = body_mass_g) |>
    generate(reps = 15000, type = "bootstrap") |>
    calculate(stat = "mean") |>
    get_confidence_interval(level = 0.95, type = "percentile") |>
    round(2)
}
```

```
species <- c("Adelie", "Gentoo", "Chinstrap")
lower_ci <- c(0,0,0)
upper_ci <- c(0,0,0)
estimated_mean <- c(0,0,0)
result_table_penguins <- data.frame(species, lower_ci, upper_ci, estimated_mean)
rownames(result_table_penguins) <- result_table_penguins$species
result_table_penguins
```

```
##             species lower_ci upper_ci estimated_mean
## Adelie       Adelie        0        0              0
## Gentoo       Gentoo        0        0              0
## Chinstrap Chinstrap        0        0              0
```

```
for (i in 1:length(species)) {
  species_name <- species[i]
  species_ci <- penguins_sample |>
    ci_mean(species_name)
  result_table_penguins[species_name, "lower_ci"] <-
```

```
    species_ci$lower_ci
  result_table_penguins[species_name, "upper_ci"] <-
    species_ci$upper_ci
  result_table_penguins[species_name, "estimated_mean"] <-
    penguins_base_sample_mean[species_name, "mean_mass"]
}
result_table_penguins
```

```
##             species lower_ci upper_ci estimated_mean
## Adelie       Adelie  3627.48  3774.17        3700.66
## Gentoo       Gentoo  4987.80  5164.44        5076.02
## Chinstrap Chinstrap  3643.01  3825.37        3733.09
```

Your task is to use this data to estimate the population parameter, the μ_{mass_k}, for body mass in grams for each of the k penguin species in the data set.

Q3.1.

rubric={autograde:10}

Compute the estimated mean body mass per penguin, $\hat{\mu}_{\text{mass}_k}$ for the kth species. The estimator will be

$$\hat{\mu}_{\text{mass}_k} = \frac{\sum_{j=1}^{n_k} X_{k,j}}{n_k},$$

where:

- $X_{k,j}$ is the body mass weight for the jth penguin belonging to the kth species in the random sample.
- n_k is the number of penguins belonging to the kth species in the random sample.

Furthermore, compute the **95% bootstrap CIs** of each **estimated** body mass weight $\hat{\mu}_{\text{mass}_k}$ using **15,000 bootstrapped samples**. This will give you the corresponding uncertainty measures.

The output data frame `answer3_1` should look like below.

species	lower_ci	upper_ci	estimated_mean
Adelie			
Gentoo			
Chinstrap			

Heads-up: Round up all your **final** table results to 2 decimal places.

Code a function called `ci_mean()` which computes the **95% bootstrap CI** by species. It should return a tibble with one row and two columns, one for the lower bound and another for the upper bound. Use seed `552` for reproducibility within this function.

```
answer3_1 <- result_table_penguins

# YOUR CODE HERE

answer3_1
```

```
##             species lower_ci upper_ci estimated_mean
## Adelie       Adelie  3627.48  3774.17        3700.66
## Gentoo       Gentoo  4987.80  5164.44        5076.02
## Chinstrap Chinstrap  3643.01  3825.37        3733.09
```

```
. = ottr::check("tests/Q3.1.R")
```

```
##
## All tests passed!
```

Q3.2.

rubric={viz:6}

Visualize the estimate $\hat{\mu}_{\text{mass}_k}$, the 95% bootstrap confidence intervals, and the **sample distribution** on a plot, using a jitter plot, violin plot, or other suitable visualization that shows data distribution (**only pick one class of plot**).

You should map the kth penguin species to the x-axis and the estimate and confidence intervals to the y-axis. Ensure that your x and y-axes are human-readable. Moreover, include a title. Assign your plot to the variable `penguin_sample_plot`.

Heads-up: There are no auto-grading test functions for this question.

```
species_sampling_tibble <- tibble(
  replicate = numeric(),
  body_mass_g = numeric(),
  species = character()
)

for (i in 1:length(species)) {
  species_name <- species[i]
  set.seed(552)
  one_species_sampling <- penguins_sample |>
    filter(species == species_name) |>
    specify(response = body_mass_g) |>
    generate(reps = 15000, type = "bootstrap") |>
    round(2) |>
    mutate(species = species_name)
  species_sampling_tibble <- bind_rows(species_sampling_tibble, one_species_sampling)
}
species_sampling_tibble
```

```
## # A tibble: 5,130,000 x 3
##    replicate body_mass_g species
##        <dbl>       <dbl> <chr>
##  1         1        3600 Adelie
##  2         1        4400 Adelie
##  3         1        3450 Adelie
##  4         1        4250 Adelie
##  5         1        3400 Adelie
##  6         1        3900 Adelie
##  7         1        3425 Adelie
##  8         1        3550 Adelie
##  9         1        3400 Adelie
## 10         1        3750 Adelie
## # ... with 5,129,990 more rows
```

```
species_sampling_nested <- species_sampling_tibble |>
  group_by(species, replicate) |>
  summarise(sample_mean_mass = mean(body_mass_g)) |>
```

```
  group_by(species) |>
  nest()
```

```
## `summarise()` has grouped output by 'species'. You can override using the
## `.groups` argument.
```

```
species_sampling_nested
```

```
## # A tibble: 3 x 2
## # Groups:   species [3]
##   species   data
##   <chr>     <list>
## 1 Adelie    <tibble [15,000 x 2]>
## 2 Chinstrap <tibble [15,000 x 2]>
## 3 Gentoo    <tibble [15,000 x 2]>
```

```
result_table_penguins <- species_sampling_nested |>
  inner_join(result_table_penguins)
```

```
## Joining, by = "species"
```

```
str(result_table_penguins)
```

```
## grouped_df [3 x 5] (S3: grouped_df/tbl_df/tbl/data.frame)
##  $ species       : chr [1:3] "Adelie" "Chinstrap" "Gentoo"
##  $ data          :List of 3
##   ..$ : tibble [15,000 x 2] (S3: tbl_df/tbl/data.frame)
##   .. ..$ replicate       : num [1:15000] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ sample_mean_mass: num [1:15000] 3704 3697 3715 3699 3716 ...
##   ..$ : tibble [15,000 x 2] (S3: tbl_df/tbl/data.frame)
##   .. ..$ replicate       : num [1:15000] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ sample_mean_mass: num [1:15000] 3785 3722 3726 3772 3730 ...
##   ..$ : tibble [15,000 x 2] (S3: tbl_df/tbl/data.frame)
##   .. ..$ replicate       : num [1:15000] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ sample_mean_mass: num [1:15000] 5074 5059 5148 5116 5104 ...
##  $ lower_ci      : num [1:3] 3627 3643 4988
##  $ upper_ci      : num [1:3] 3774 3825 5164
##  $ estimated_mean: num [1:3] 3701 3733 5076
##  - attr(*, "groups")= tibble [3 x 2] (S3: tbl_df/tbl/data.frame)
##   ..$ species: chr [1:3] "Adelie" "Chinstrap" "Gentoo"
##   ..$ .rows  : list<int> [1:3]
##   .. ..$ : int 1
##   .. ..$ : int 2
##   .. ..$ : int 3
##   .. ..@ ptype: int(0)
##   ..- attr(*, ".drop")= logi TRUE
```

```
result_table_penguins
```

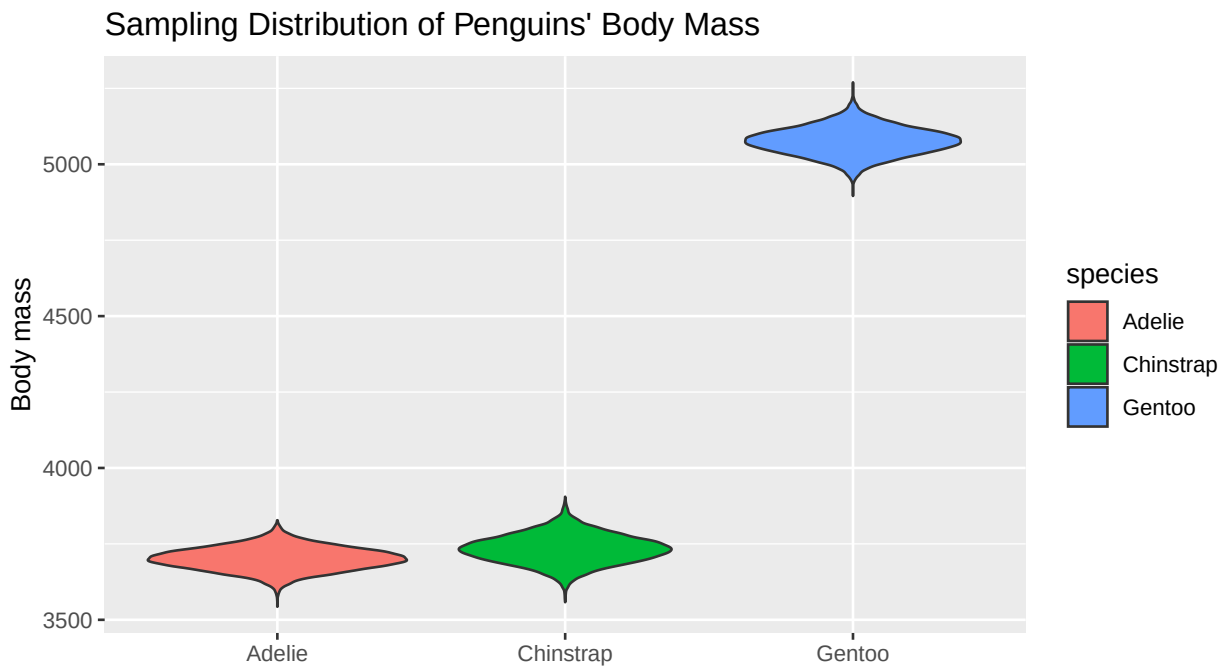
```
## # A tibble: 3 x 5
## # Groups:   species [3]
##   species   data                  lower_ci upper_ci estimated_mean
##   <chr>     <list>                   <dbl>    <dbl>          <dbl>
## 1 Adelie    <tibble [15,000 x 2]>    3627.    3774.          3701.
## 2 Chinstrap <tibble [15,000 x 2]>    3643.    3825.          3733.
## 3 Gentoo    <tibble [15,000 x 2]>    4988.    5164.          5076.
```

```
result_table_penguins_unnested <- result_table_penguins |>
  unnest(cols = data) |>
  select(-replicate)
result_table_penguins_unnested
```

```
## # A tibble: 45,000 x 5
## # Groups:   species [3]
##    species sample_mean_mass lower_ci upper_ci estimated_mean
##    <chr>              <dbl>    <dbl>    <dbl>          <dbl>
##  1 Adelie             3704.    3627.    3774.          3701.
##  2 Adelie             3697.    3627.    3774.          3701.
##  3 Adelie             3715.    3627.    3774.          3701.
##  4 Adelie             3699.    3627.    3774.          3701.
##  5 Adelie             3716.    3627.    3774.          3701.
##  6 Adelie             3710.    3627.    3774.          3701.
##  7 Adelie             3618.    3627.    3774.          3701.
##  8 Adelie             3747.    3627.    3774.          3701.
##  9 Adelie             3663.    3627.    3774.          3701.
## 10 Adelie             3732.    3627.    3774.          3701.
## # ... with 44,990 more rows
```

```
penguin_sample_plot <- ggplot(result_table_penguins_unnested) +
  aes(y = sample_mean_mass, x = species, fill = species) +
  geom_violin(scale = 'count') +
  labs(x = "", y = "Body mass") +
  ggtitle("Sampling Distribution of Penguins' Body Mass")

penguin_sample_plot
```



Q3.3.

rubric={reasoning:5}

In one or two paragraphs, use written English to interpret and report the estimates (and their bootstrap 95% CIs) for each of the k penguin species.

- It is estimated that the mean body mass in of the `Adelie` penguins is `3700.66g` and we are 95% confident that the mean of `Adelie` penguins' body mass is somewhere between `3627.48g` and `3774.17g`.
- It is estimated that the mean body mass in of the `Gentoo` penguins is `5076.02g` and we are 95% confident that the mean of `Gentoo` penguins' body mass is somewhere between `4987.8g` and `5164.44g`.
- It is estimated that the mean body mass in of the `Chinstrap` penguins is `3773.09g` and we are 95% confident that the mean of `Chinstrap` penguins' body mass is somewhere between `3643.01g` and `3825.37g`.

Exercise 4: Hypothesis Test for a Difference in Means

Now, we want to know whether the mean body mass (in grams) of *Pygoscelis antarctica* (Chinstrap) is different from the mean body mass (in grams) of *Pygoscelis adeliae* (Adelie). To answer this question, we will use the data set `penguins_sample` again.

Your task is to answer this question by performing a hypothesis test that uses permutation and the `infer` package (see, for example, **this case**). Let us perform this analysis part by part.

Q4.1.

rubric={reasoning:4}

Using mathematical notation, **along with a proper parameter definition**, clearly specify the null, H_0, and alternative, H_A, hypotheses.

H_0: $\mu_C = \mu_A$

H_A: $\mu_C \neq \mu_A$

Where μ_C is the mean body mass (in grams) of `Chinstrap` penguins and μ_A is the mean body mass(in grams) of `Adelie` penguins.

Q4.2.

rubric={reasoning:2}

Using mathematical notation, **along with a proper estimator definition**, define the test statistic you will use.

Define test statistics $\delta = \mu_C - \mu_A$ Where μ_C and μ_A are the mean body mass (in grams) of `Chinstrap` and `Adelie` penguins. δ is the difference between the 2 mean values.

Q4.3.

rubric={accuracy:10}

Code the hypothesis test using the permutation method provided by the `infer` package with $r = 1000$ permuted data sets. Your code should return your test statistic from **Q4.2** as `chinstrap_adelie_test_stat`, and the p-value `chinstrap_adelie_p_value`. Use `552` as a simulation seed.

```
penguins_sample_AC <- penguins_sample |>
  filter(species == "Adelie" | species == "Chinstrap")
penguins_sample_AC
```

```
## # A tibble: 219 x 2
##    species body_mass_g
##    <fct>         <int>
##  1 Adelie         3750
##  2 Adelie         3800
##  3 Adelie         3250
##  4 Adelie         3450
##  5 Adelie         3650
##  6 Adelie         3625
##  7 Adelie         4675
##  8 Adelie         3475
##  9 Adelie         4250
## 10 Adelie         3300
## # ... with 209 more rows
```

```
set.seed(552) # For reproducibility.
null_distribution_penguins <- penguins_sample_AC |>
  specify(formula = body_mass_g ~ species) |>
  hypothesize(null = "independence") |>
  generate(reps = 1000, type = "permute") |>
  calculate(stat = "diff in means", order = c("Chinstrap", "Adelie"))
```

```
## Dropping unused factor levels Gentoo from the supplied explanatory variable 'species'.
```

```
null_distribution_penguins
```

```
## Response: body_mass_g (numeric)
## Explanatory: species (factor)
## Null Hypothesis: independence
## # A tibble: 1,000 x 2
##    replicate    stat
##        <int>   <dbl>
##  1         1  -22.5
##  2         2   72.9
##  3         3  -49.7
##  4         4  -32.1
##  5         5  -31.0
##  6         6   -1.17
##  7         7 -118.
##  8         8   37.8
##  9         9   19.6
## 10        10   20.7
## # ... with 990 more rows
```

```
obs_diff_means_AC <- penguins_sample_AC |>
  specify(formula = body_mass_g ~ species) |>
  calculate(stat = "diff in means", order = c("Chinstrap", "Adelie"))
```

```
## Dropping unused factor levels Gentoo from the supplied explanatory variable 'species'.
```

```
obs_diff_means_AC
```

```
## Response: body_mass_g (numeric)
## Explanatory: species (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
```

```
## 1  32.4
```

```
chinstrap_adelie_test_stat <- obs_diff_means_AC
chinstrap_adelie_p_value <- null_distribution_penguins |>
  get_p_value(obs_stat = obs_diff_means_AC, direction = "both")

# YOUR CODE HERE

chinstrap_adelie_test_stat
```

```
## Response: body_mass_g (numeric)
## Explanatory: species (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  32.4
```

```
chinstrap_adelie_p_value
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.588
```

Q4.4.

rubric={viz:6}

Visualize the **null distribution** you obtained as a histogram along with an estimated density plot (via `geom_density()`). Next, indicate your **OBSERVED test statistic** δ^* as a solid vertical red line and the corresponding empirical quantile thresholds as blue vertical dashed lines.

Ensure that your x and y-axes are human-readable. Moreover, include a title. Assign your plot to the variable `penguin_null_dist_plot`.

Heads-up: There are no auto-grading test functions for this question.

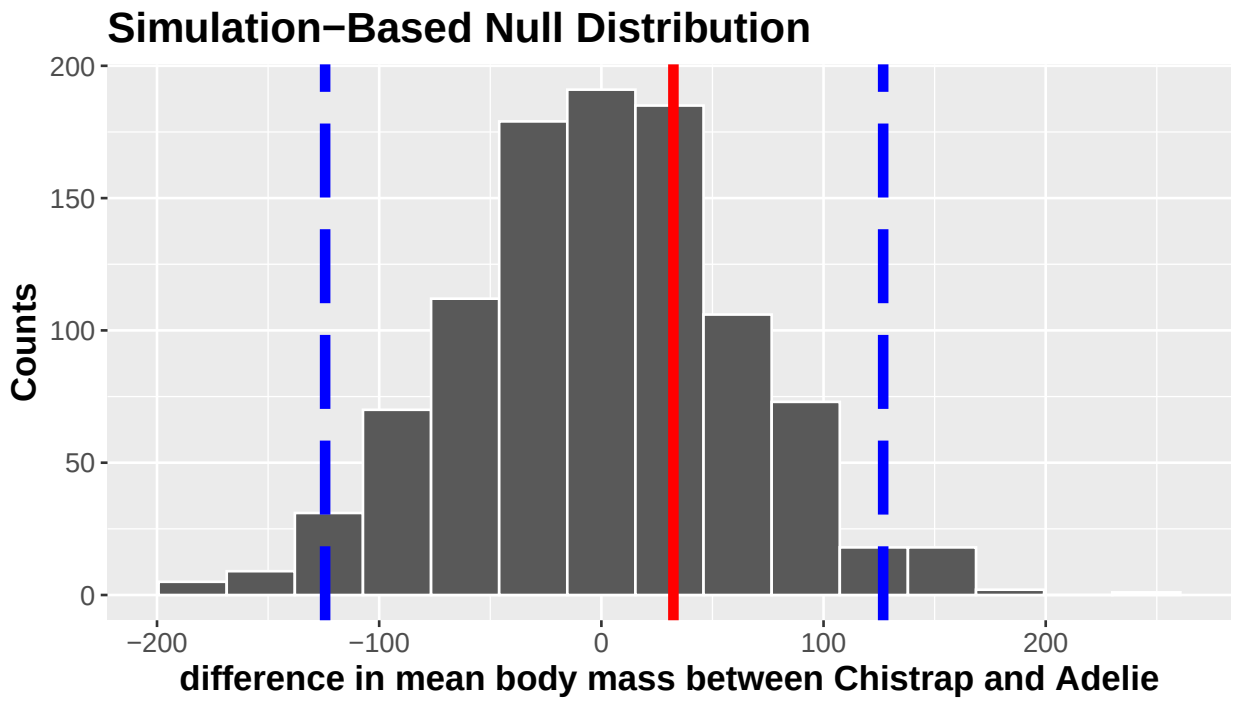
```
alpha_threshold_right <- quantile(null_distribution_penguins$stat, 0.975)
alpha_threshold_left <- quantile(null_distribution_penguins$stat, 0.025)

options(repr.plot.width = 10, repr.plot.height = 10)
h0_dist <- null_distribution_penguins |>
    visualize() +
    theme(text = element_text(size=14)) +
  theme(
    text = element_text(size = 14),
    plot.title = element_text(face = "bold"),
    axis.title = element_text(face = "bold"),
    legend.title = element_text(face = "bold"),
  )

h0_dist <- h0_dist +
  geom_vline(xintercept = alpha_threshold_left, color = "blue", lty = 5, size = 2) +
  geom_vline(xintercept = alpha_threshold_right, color = "blue", lty = 5, size = 2) +
  geom_vline(xintercept = obs_diff_means_AC[[1]], color = "red", size = 2) +
  labs(x = "difference in mean body mass between Chistrap and Adelie", y = "Counts")
```

```
penguin_null_dist_plot <- h0_dist

penguin_null_dist_plot
```



Q4.5.

rubric={reasoning:5}

In **one paragraph**, report the results of your analysis, which should include the **effect size** (i.e., your test statistic δ^*) and the p-value.

Given that our p-value is > 0.05 (our predefined significance shreshold α), we cannot reject the null hypothesis H_0 (the mean body mass of the 2 species are the same). In other words, there is not enough statistical evidence to say the mean body mass of the 2 species (`Chinstrap` and `Adelie`) are different.

Q5 No Attempt

No Attempt to the question so I deleted this part.