Lab 10: Proportions and Inference

Your name and Student ID

Today's date

```
BEGIN ASSIGNMENT
requirements: requirements.R
generate: true
files:
 - data
 - turn_in.py
 - src
```

Instructions

- Due date: Friday, April 30th (Friday before RRR week)
- Late penalty: 50% late penalty if submitted within 24 hours of due date, no marks for assignments submitted thereafter.
- This assignment is graded on **correct completion**, all or nothing. You must pass all public tests and submit the assignment for credit.
- Submission process: Follow the submission instructions on the final page. Make sure you do not remove any `\newpage` tags or rename this file, as this will break the submission.

Boston Data on Median Household Value and Distance to Employment Centers

We are examining a data set used to predict housing prices in the area around Boston (Harrison, D. and Rubinfeld, 1978). We wish to examine specifically the association of the measure of housing price (`medv`, median value of owner-occupied homes in the $1000s) and a measure of adjacency to employment (a weighted distance, roughly in miles). The data frame (Boston) is contained in another package (MASS), which we load below.

```r
### NOTE: All of the code is to get you started on the lab. You do not need to
### understand any functions below that you have not seen before.

# Load library with data
library(MASS)

### NOTE: This package has a function `select()` that can be confused with
### dplyr's select. To overcome this, we first import the data we need and then
### detach the library before loading dplyr.

# List variables
boston_housing <-
  read.csv("data/Boston.csv")

# Variable definition - take a quick look at the variables in the data frame
```

```r
#help(Boston)
detach(package:MASS)
library(testthat)
library(broom)
library(dplyr)
library(ggplot2)
library(tidyr)
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.0.5
```

```r
### Normally, when we are doing inference, we take a random sample from the
### entire population so we can see how well we can make inference when we only
### have 50 rows of data. If you have after the lab, try taking the following
### random sample from the data, and see if your results change.
```

Section 1

1. [1 mark] Perform and summarize the results of a linear regression of `medv` (median value of owner-occupied homes in $1000s) and `dis` (weighted mean of distances to five Boston employment centres) using Boston data.

Assign the linear model to an object called `p1`.

Be careful about which variable is explanatory and which is response!

```
BEGIN QUESTION
name: p1
manual: false
points: 1
```

```
p1 <- lm(medv ~ dis, data = boston_housing) # SOLUTION
```

```
## Test ##
test_that("p1a", {
  expect_true(class(p1) %in% c("lm", "glm"))
  print("Checking: p1 a linear model")
})
```

```
## [1] "Checking: p1 a linear model"
## Test passed
```

```
## Test ##
test_that("p1b", {
  expect_true(round(p1$coefficients[2],2) == 1.09)
  print("Checking: explanatory and response variables")
})
```

```
## [1] "Checking: explanatory and response variables"
## Test passed
```

2. [1 mark] Interpret the slope parameter and p-value from the table. What null and alternative hypotheses does this p-value refer to? Store the slope parameter, ROUNDED to 2 decimal places, to the object `p2`.

```
BEGIN QUESTION
name: p2
manual: false
points: 1
```

```r
p2 <- "YOUR ANSWER HERE"

# BEGIN SOLUTION
p2 <- 1.09
# END SOLUTION
```

```r
## Test ##
test_that("p2", {
  expect_true(all.equal(p2, 1.092, tol = 0.01))
  print("Checking: value of slope to 2 decimals")
})
```

```
## [1] "Checking: value of slope to 2 decimals"
## Test passed
```

We find that the weighted mean of distances to five Boston employment centres has a positive and significant impact on neighborhood median home value, since the p-value on `dis` is close to zero. Let β_d represent the slope coefficient for `dis`. The hypotheses are:

$$H_0 : \beta_d = 0$$
$$H_A : \beta_d \neq 0$$

3. [1 mark] Derive a 95% CI for this slope parameter and assign the object `p8` to a vector of the lower bound followed by the upper bound of the interval, to AT LEAST one decimal place of precision. In your opinion, would you expect the direction of this relationship to hold if the data were collected today?

```
BEGIN QUESTION
name: p3
manual: true
```

```r
#Put your code here
p3 <- "YOUR ANSWER HERE"

# BEGIN SOLUTION NO PROMPT
#note to future GSIs - make this an AG question in future iterations.

upper_ci <-
  1.091613 + 0.1883784 * qt(p=0.975, df = (506-2))
lower_ci <-
  1.091613 - 0.1883784 * qt(p=0.975, df = (506-2))
p3 <- c(lower_ci, upper_ci)
# END SOLUTION
```

Most answers are fine. I believe the relationship may reverse if data were collected today due to gentrification / general increase in popularity of living in the city.

4. [1 mark] Use a function to look at the r-squared value for this model. Assign the r-squared ROUNDED to 2 decimal places, to the object `p4`. Does `dis` explain alot of the variance in median household value? Would you expect it to?

```
BEGIN QUESTION
name: p4
manual: false
points: 1
```

```
p4 <- "YOUR ANSWER HERE"

# BEGIN SOLUTION NO PROMPT
glance(p1)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    0.0625        0.0606  8.91      33.6 1.21e-8     1 -1824. 3654. 3667.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
p4 <- 0.06
# END SOLUTION
```

```
## Test ##
test_that("p4", {
  expect_true(all.equal(p4, 0.06246437, tol = 0.01))
  print("Checking: value of r_squared to two decimals")
})
```

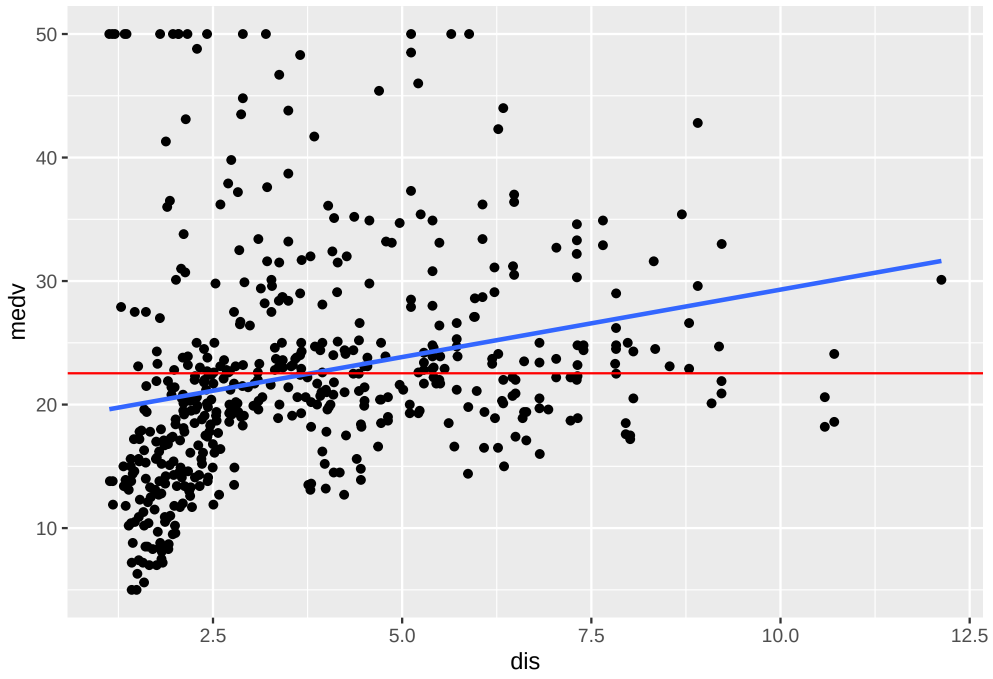
```
## [1] "Checking: value of r_squared to two decimals"
## -- Failure (<text>:3:3): p4 ----------------------------------------------------
## all.equal(p4, 0.06246437, tol = 0.01) is not TRUE
##
## ‘actual‘ is a character vector (’Mean relative difference: 0.04107283’)
## ‘expected‘ is a logical vector (TRUE)
```

The distance variable does not explain very much variance in the data. I would say this is expected, as there are many other factors that might impact the price of a neighborhood.

5. [2 marks] Back to the fit of the model of `medv` vs. `dis`. Make a plot with the raw data points, the fitted line from the simple linear regression model (only containing `medv` and `dis`) and also add a line with a slope of 0. You can have that line cross the y axis at the average value of `medv` to vertically bisect the data points. Store your plot as the object `p5`.

```
BEGIN QUESTION
name: p5
manual: false
points: 2
```

```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
## Test ##
test_that("p5a", {
  expect_true("ggplot" %in% class(p5))
  print("Checking: ggplot defined")
})
```

```
## [1] "Checking: ggplot defined"
## Test passed
```

```
## Test ##
test_that("p5b", {
  expect_true(identical(p5$data, boston_housing))
  print("Checking: boston_housing data used")
})
```

```
## [1] "Checking: boston_housing data used"
## Test passed
## Test ##
test_that("p5c", {
  expect_true(rlang::quo_get_expr(p5$mapping$x) == "dis")
  print("Checking: dis on x axis")
})
```

```
## [1] "Checking: dis on x axis"
## Test passed
```

```
## Test ##
test_that("p5d", {
  expect_true(rlang::quo_get_expr(p5$mapping$y) == "medv")
  print("Checking: medv on y axis")
})
```

```
## [1] "Checking: medv on y axis"
## Test passed
```

```
## Test ##
test_that("p5e", {
  expect_true("GeomPoint" %in% class(p5$layers[[1]]$geom) | "GeomSmooth" %in% class(p5$layers[[1]]$geom
  print("Checking: points added or lines drawn")
})
```

```
## [1] "Checking: points added or lines drawn"
## Test passed
```

```
## Test ##
test_that("p5f", {
  expect_true("GeomPoint" %in% class(p5$layers[[2]]$geom) | "GeomSmooth" %in% class(p5$layers[[2]]$geom
  print("Checking: points added or lines drawn")
})
```

```
## [1] "Checking: points added or lines drawn"
## Test passed
```

```
## Test ##
test_that("p5g", {
  expect_true("GeomPoint" %in% class(p5$layers[[3]]$geom) | "GeomSmooth" %in% class(p5$layers[[3]]$geom
  print("Checking: points added or lines drawn")
})
```

```
## [1] "Checking: points added or lines drawn"
## Test passed
```

6. [2 marks] For you, does the plot raise any concerns about the assumptions of the linear regression you just performed? What other plots might you do to explore the fit? One helpful plot would compare the distribution of model residuals to a theoretical normal distribution. Assign the object `p6` to the FIRST TWO LETTERS of the name of this plot. ## STOP: REMOVE EVAL= F FROM THIS CODE CHUNK BEFORE CONTINUING ON

```
BEGIN QUESTION
name: p6
manual: false
points: 2
```

```r
p6 <- "YOUR ANSWER HERE"

# BEGIN SOLUTION
p6 <- "QQ"
# END SOLUTION

## Go over this code with your GSI and make sure you understand it.


### below are some plots for model diagnostics that we've gone over in lecture.
data_with_predictions <-
  augment(p1)
ggplot(data_with_predictions, aes(dis, medv)) +
  geom_smooth(method = "lm", se = F) +
  geom_point(alpha = 0.5) +
  geom_segment(aes(xend = dis, yend = .fitted), lty = 2, alpha = 0.5) +
  theme_minimal(base_size = 15) +
  labs(title = "(a) Scatter plot")
ggplot(data_with_predictions, aes(sample = .resid)) +
  geom_qq() +
  geom_qq_line() +
  theme_minimal(base_size = 15) +
  labs(y = "Residuals", x = "Theoretical quantiles", title = "(b) QQplot")
ggplot(data_with_predictions, aes(y = .resid, x = .fitted)) +
  geom_point() +
  theme_minimal(base_size = 15) +
  geom_hline(aes(yintercept = 0)) +
  labs(y = "Residuals", x = "Fitted values", title = "(c) Fitted vs. residuals")
predictions_reshaped <-
  data_with_predictions %>%
  select(medv, .resid) %>%
  gather(key = "type", value = "value", medv, .resid)
ggplot(predictions_reshaped, aes(y = value)) +
  geom_boxplot(aes(fill = type)) +
  theme_minimal(base_size = 15) +
  labs(title = "(d) Amount explained")
```

```r
## Test ##
test_that("p6", {
  expect_true(toupper(p6) == "QQ")
  print("Checking: answer for plot type")
})
```

```
## -- Error (<text>:3:3): p6 ----------------------------------------------------
## Error: object 'p6' not found
## Backtrace:
##  1. testthat::expect_true(toupper(p6) == "QQ")
##  4. base::toupper(p6)
```

This plot does raise concerns about the assumptions of a linear model. We see that the data pattern for lower values does not match the data pattern for larger values of the independent variable. This raises questions around the linearity of the data, the normality of the data, and equal variance of the residuals. Overall, we would be very concerned with using results of this model for interpretation or prediction.

Regardless of your answer, we go forward using the model to make inferences about the points on the line.

Pointwise Confidence Intervals and Multiple Testing

As you learned in lecture, there are two types of confidence intervals applicable to estimating a point on the plot which are related to whether one is predicting the population average among individuals with $X = x$ (**mean response**) or whether one is predicting the actual Y for a particular individual (** single observation**). For this assignment, we will concentrate on the confidence interval for the mean response. We do so because it is rare to use statistical models in public health as forecasting models (predicting an individual's health in the future) and more common to use them to estimate population-level changes (how does the mean health change in a population as we change exposure). However, as precision medicine becomes more of a reality and the models accurately predict health (i.e., have high R^2's), then statistical forecasting may become more common in our field.

7. [2 marks] Calculate four 95% confidence intervals for the mean response, one at each `dis` value: 2.5, 5.0, 7.5, and 10.0 miles. Store the lower bounds for each confidence interval, ROUNDED to two decimal places, in a vector called `p7`.

Hint: Use the predict function, and be sure to specify interval = "confidence"

OPTIONAL: If time allows, add the four CIs to a scatter plot of the data (along with the line of best fit).

```
BEGIN QUESTION
name: p7
manual: false
points: 2
```

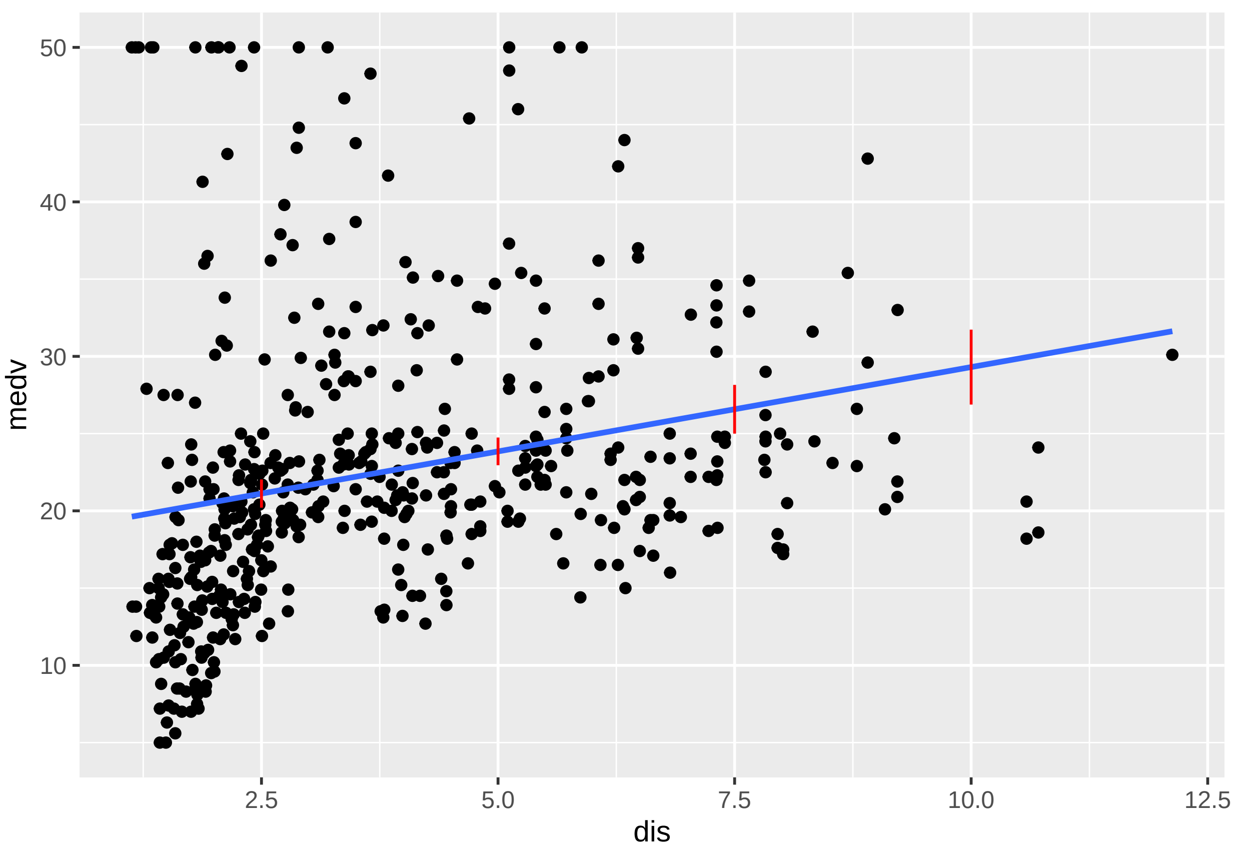
```
#Put your code here
### Helpful Data Frame:
ci_dataframe <-
  data.frame(dis = c(2.5, 5.0, 7.5, 10))
p7 <- "YOUR CODE AND ANSWER HERE"

# BEGIN SOLUTION NO PROMPT
cis <- predict(p1, newdata = ci_dataframe, interval = "confidence") %>%
  cbind(ci_dataframe)
cis
```

```
##        fit      lwr      upr  dis
## 1 21.11912 20.20485 22.03339  2.5
## 2 23.84815 22.95092 24.74539  5.0
## 3 26.57719 25.00035 28.15402  7.5
## 4 29.30622 26.88135 31.73108 10.0
```

```
p7 <- c(20.20, 22.95, 25.00, 26.88)
boston_housing %>%
  ggplot(aes(x=dis, y=medv)) + geom_point() +
  ## line of best fit:
  geom_smooth(method = "lm", se = FALSE) +
  ## cis
  geom_segment(data=cis, aes(x=dis, xend=dis, y=lwr, yend = upr),
               color = "red")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
# END SOLUTION
```

```
## Test ##
test_that("p7a", {
  expect_true(class(p7) == "numeric" &
               length(p7) == 4)
  print("Checking: p7 is a vector with 4 numbers")
})
```

```
## [1] "Checking: p7 is a vector with 4 numbers"
## Test passed
```

```
## Test ##
test_that("p7b", {
  expect_true(round(p7[1],2) == 20.20)
  print("Checking: lowerbound for dis = 2.5 to two decimal places")
})
```

```
## [1] "Checking: lowerbound for dis = 2.5 to two decimal places"
## Test passed
```

```
## Test ##
test_that("p7c", {
  expect_true(round(p7[2],2) == 22.95)
  print("Checking: lowerbound for dis = 5 to two decimal places")
})
```

```
## [1] "Checking: lowerbound for dis = 5 to two decimal places"
## Test passed
```

```
## Test ##
test_that("p7d", {
  expect_true(round(p7[3],2) == 25.00)
  print("Checking: lowerbound for dis = 7.5 to two decimal places")
})
```

```
## [1] "Checking: lowerbound for dis = 7.5 to two decimal places"
## Test passed
```

```
## Test ##
test_that("p7e", {
  expect_true(round(p7[4],2) == 26.88)
  print("Checking: lowerbound for dis = 10 to two decimal places")
})
```

```
## [1] "Checking: lowerbound for dis = 10 to two decimal places"
## Test passed
```

8. [1 mark] Interpret the pointwise 95% confidence interval of the median house price when distance = 10.

```
BEGIN QUESTION
name: p8
manual: true
```

With 95% confidence we believe the average neighborhood median home value for Boston neighborhoods with a distance value of 10 is somewhere between 26.88 and 31.73.

9. [1 mark] Do the CI's differ in length for different values of `dis`? Why or why not?

```
BEGIN QUESTION
name: p9
manual: true
```

The CI's do differ in length for different values of distance. The intervals will be the narrowest near the mean of `dis`.

It is difficult to understand the changing lengths of these intervals. If you are interested, a thorough explanation is here: https://stats.stackexchange.com/questions/85560/shape-of-confidence-interval-for-predicted-values-in-linear-regression

Section 2

The rest of the assignment is for practice. You are responsible for the material but it will not be part of your grade.

Tests of Changes in Sex Ratios Based on a Single Sample

There is a long literature studying changes in sex-ratios of births due to stressful events, such as 9/11. In today's lab, we consider a relatively small study that recorded biomarkers of stress on pregnancy. In the group of subjects that had the highest markers of stress (based on cortisol), there were 14 births to males out of a total of 38.

In this lab, we will compare the four methods we learned to calculate CIs for proportions. Recall that two of these methods involved hand calculations (though we can treat R as if it were a calculator) and two of the methods used built-in R functions.

Store your answers to questions that ask for confidence intervals like so:

```
pX <- c(lowerbound, upperbound)

# For example, if lowerbound = 10, upperbound = 20:
pX <- c(10, 20)
```

10. [1 mark] Use the Normal approximation to construct a 95% (using z = 1.96) confidence interval in this high stress group. Store your answer to `p10` using the following format: `p10 <- c(lowerbound, upperbound)`. We also call this specific method of constructing the CI the "large sample method".

```
BEGIN QUESTION
name: p10
manual: false
points: 0
```

```
# YOUR CODE HERE

# Replace "lowerbound" and "upperbound" with your answer
# If your answer is a number, make sure it doesn't have quotes around it
p10 <- c("lowerbound", "upperbound")
p10
```

```
## [1] "lowerbound" "upperbound"
```

```
# BEGIN SOLUTION
p.hat <- 14/38 # estimate proportion
se <- sqrt(p.hat*(1-p.hat)/38) # standard error
p10 <- c(p.hat - 1.96*se, p.hat + 1.96*se) # CI
p10
```

```
## [1] 0.2150476 0.5217945
```

```
# END SOLUTION
```

```r
## Test ##
test_that("p11a", {
  expect_true(all.equal(p10[1],0.2150476, tol = 0.001))
  print("Checking: lowerbound to 3 decimal places")
})
```

```
## [1] "Checking: lowerbound to 3 decimal places"
## Test passed
```

```r
## Test ##
test_that("p11b", {
  expect_true(all.equal(p10[2],0.5217945, tol = 0.001))
  print("Checking: upperbound to 3 decimal places")
})
```

```
## [1] "Checking: upperbound to 3 decimal places"
## Test passed
```

The 95% CI goes from 21.5% to 52.2% using the large sample method.

11. [1 mark] Create the 95% CI again, this time using the R function that implements the Wilson Score method with a continuity correction. Then, assign the lowerbound and upperbound to `p11`.

```
BEGIN QUESTION
name: p11
manual: false
points: 0
```

```
# YOUR CODE HERE

# Replace "lowerbound" and "upperbound" with your answer
# If your answer is a number, make sure it doesn't have quotes around it
p11 <- c("lowerbound", "upperbound")
p11
```

```
## [1] "lowerbound" "upperbound"
```

```
# BEGIN SOLUTION
prop.test(14, 38, p=0.5)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  14 out of 38, null probability 0.5
## X-squared = 2.1316, df = 1, p-value = 0.1443
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.2229295 0.5400424
## sample estimates:
##         p
## 0.3684211
```

```
p11 <- c(0.2229295, 0.5400424)
# END SOLUTION
```

```
## Test ##
test_that("p11a", {
  expect_true(all.equal(p11[1], 0.2229295, tol = 0.001))
  print("Checking: lowerbound to 3 decimal places")
})
```

```
## [1] "Checking: lowerbound to 3 decimal places"
## Test passed
```

```
## Test ##
test_that("p11b", {
  expect_true(all.equal(p11[2],0.5400424, tol = 0.001))
  print("Checking: upperbound to 3 decimal places")
})
```

```
## [1] "Checking: upperbound to 3 decimal places"
## Test passed
```

The 95% CI goes from 22.3% to 54.0% using the Wilson Score method with continuity correction.

12. [1 mark] Create the 95% CI (using z = 1.96) again, this time using the Plus 4 method. Store your answer to `p12` using the following format: `p12 <- c(lowerbound, upperbound)`.

```
BEGIN QUESTION
name: p12
manual: false
points: 0
```

```r
# YOUR CODE HERE

# Replace "lowerbound" and "upperbound" with your answer
# If your answer is a number, make sure it doesn't have quotes around it
p12 <- c("lowerbound", "upperbound")
p12
```

```
## [1] "lowerbound" "upperbound"
```

```r
# BEGIN SOLUTION
p.tilde <- (14 + 2)/(38 + 4)
se <- sqrt(p.tilde*(1 - p.tilde)/42) # standard error
p12 <- c(p.tilde - 1.96*se, p.tilde + 1.96*se) # CI
p12
```

```
## [1] 0.2340838 0.5278209
```

```r
# END SOLUTION
```

```r
## Test ##
test_that("p12a", {
  expect_true(all.equal(p12[1], 0.2340838, tol = 0.001))
  print("Checking: lowerbound to 3 decimal places")
})
```

```
## [1] "Checking: lowerbound to 3 decimal places"
## Test passed
```

```r
## Test ##
test_that("p12b", {
  expect_true(all.equal(p12[2],0.5278209, tol = 0.001))
  print("Checking: upperbound to 3 decimal places")
})
```

```
## [1] "Checking: upperbound to 3 decimal places"
## Test passed
```

The 95% CI goes from 23.4% to 52.8% using the Plus 4 method.

13. [1 mark] Create the 95% CI again, this time using the R function that implements the Clopper Pearson (Exact) method. Then, assign the lowerbound and upperbound to `p13`.

```
BEGIN QUESTION
name: p13
manual: false
points: 0
```

```r
# YOUR CODE HERE

# Replace "lowerbound" and "upperbound" with your answer
# If your answer is a number, make sure it doesn't have quotes around it
p13 <- c("lowerbound", "upperbound")
p13
```

```
## [1] "lowerbound" "upperbound"
```

```r
# BEGIN SOLUTION
binom.test(14, 38, p=0.5)
```

```
##
##  Exact binomial test
##
## data:  14 and 38
## number of successes = 14, number of trials = 38, p-value = 0.1433
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.2181250 0.5400572
## sample estimates:
## probability of success
##              0.3684211
```

```r
p13 <- c(0.2181250, 0.5400572)
# END SOLUTION
```

```r
## Test ##
test_that("p13a", {
  expect_true(all.equal(p13[1], 0.2181250, tol = 0.001))
  print("Checking: lowerbound to 3 decimal places")
})
```

```
## [1] "Checking: lowerbound to 3 decimal places"
## Test passed
```

```r
## Test ##
test_that("p13b", {
  expect_true(all.equal(p13[2], 0.5400572, tol = 0.001))
  print("Checking: upperbound to 3 decimal places")
})
```

```
## [1] "Checking: upperbound to 3 decimal places"
## Test passed
```

The 95% CI goes from 21.8% to 54.0% using the Exact method.

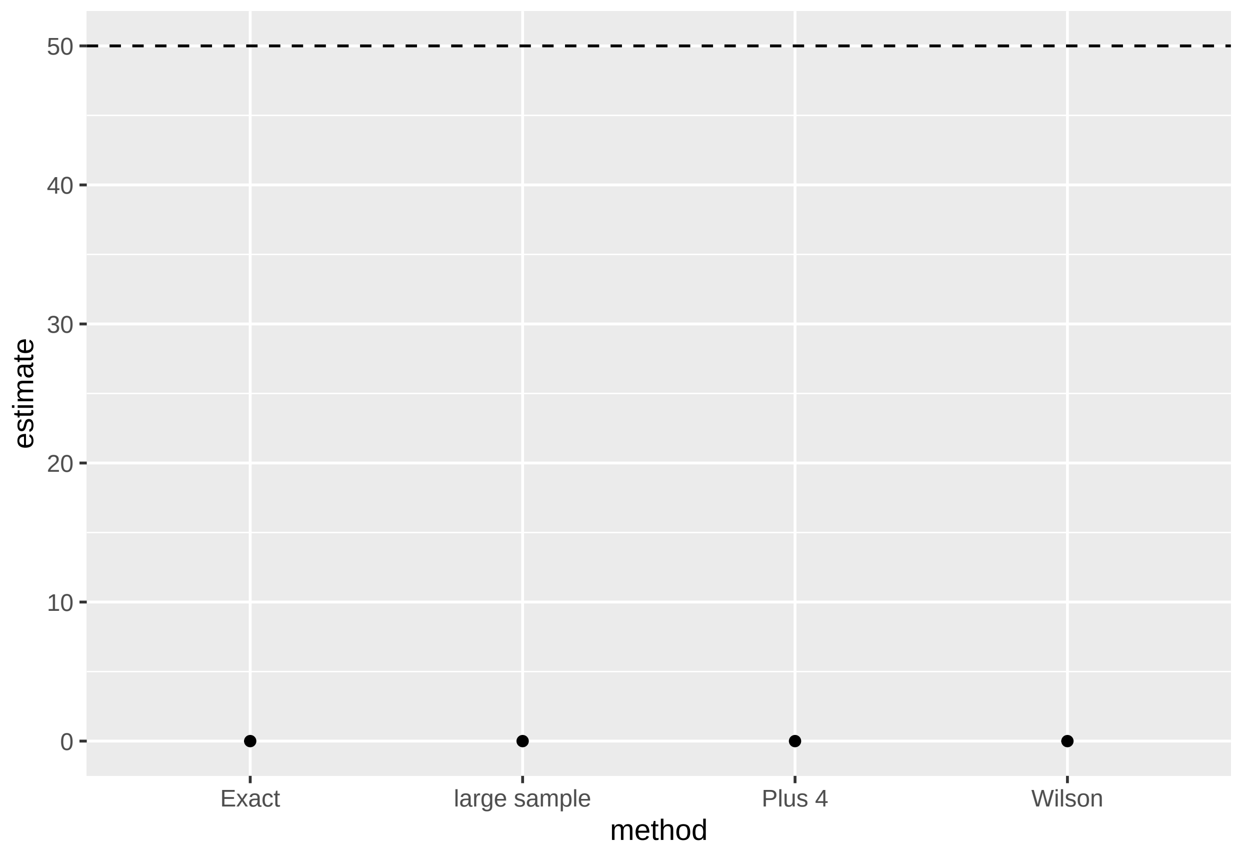
14. [2 marks] Here is a code template to help you to graphically present these estimates. Graphical presentations of estimates and their CIs is very useful for assessing whether the CIs overlap the null hypothesized value and tends to be better than presenting tables of estimates to readers of your research. You can fill in the values accordingly.

```
BEGIN QUESTION
name: p14
manual: true
```

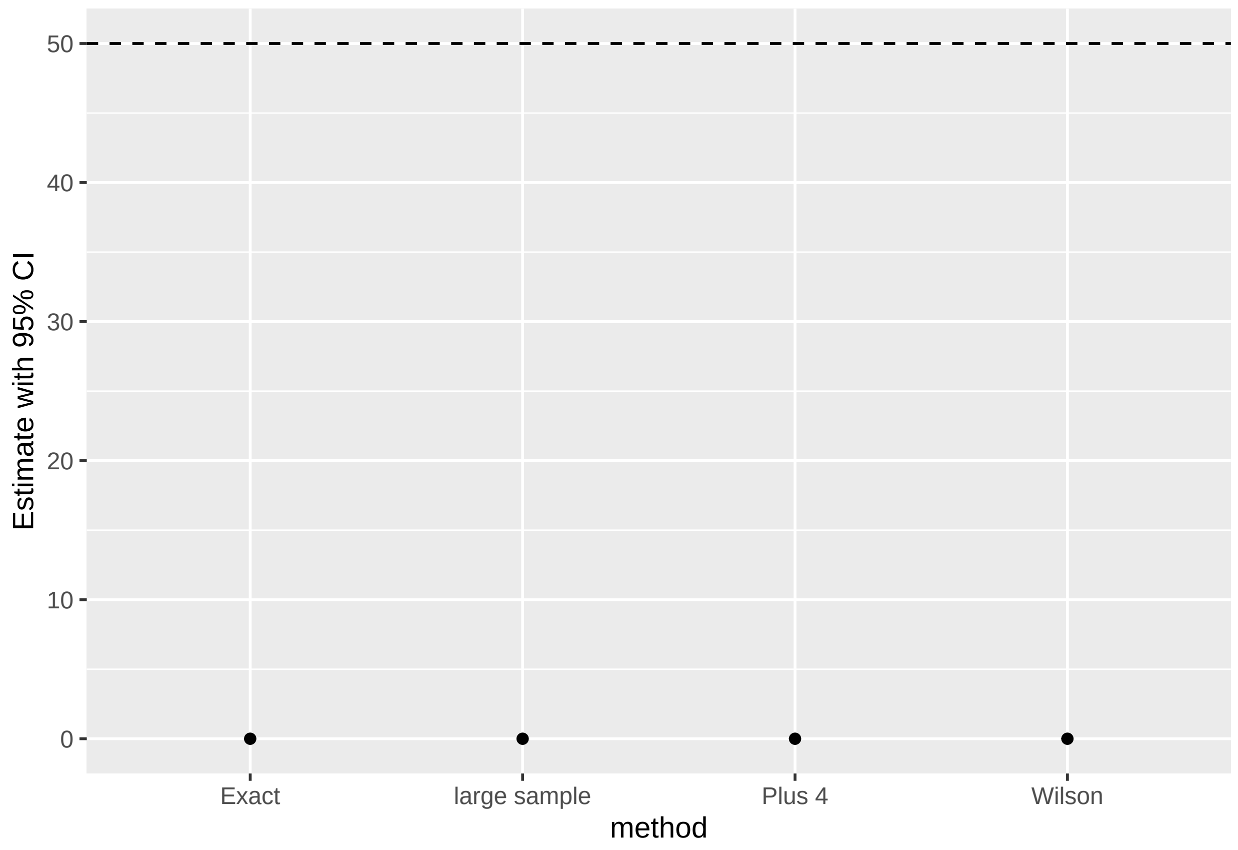
```
# First make a tibble (an easy way to make a data frame) with the data about
# each confidence interval. To do this, replace each instance of 0.00 with the
# estimate from your calculations above.

library(ggplot2)
library(tibble)
sex_CIs <- tibble(method   = c("large sample", "Exact", "Wilson", "Plus 4"),
                  lower_CI = c(0.0           , 0.0    ,  0.0    ,  0.0),
                  upper_CI = c(0.0           , 0.0    ,  0.0    ,  0.0),
                  estimate = c(0.0           , 0.0    ,  0.0    ,  0.0)
                  )

# Build the ggplot incrementally, to understand how it works.
# Step 1: (qu: why do we put a horizontal line at 50?)
ggplot(data = sex_CIs, aes(x = method, y = estimate)) +
  geom_point() +
  geom_hline(aes(yintercept = 50), lty = 2)
```

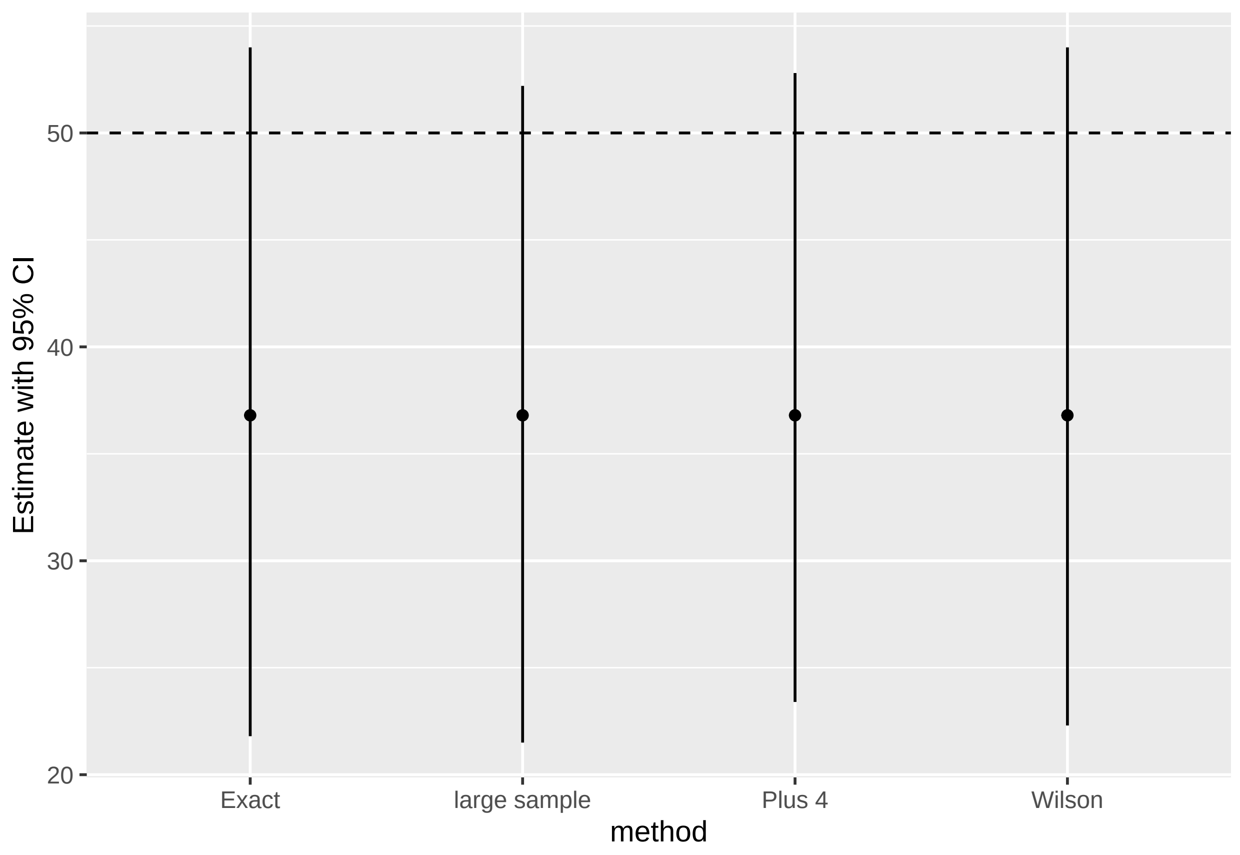


```
# Step 2:
ggplot(data = sex_CIs, aes(x = method, y = estimate)) +
  geom_point() +
  geom_hline(aes(yintercept = 50), lty = 2) +
  geom_segment(aes(xend = method, y = lower_CI, yend = upper_CI)) +
  labs(y = "Estimate with 95% CI")
```



```
# BEGIN SOLUTION
sex_CIs <- tibble(method   = c("large sample", "Exact", "Wilson", "Plus 4"),
                  lower_CI = c(21.5          , 21.8   ,  22.3   ,  23.4),
                  upper_CI = c(52.2          , 54.0   ,  54.0   ,  52.8),
                  estimate = c(36.8          , 36.8   ,  36.8   ,  36.8)
                  )

ggplot(data = sex_CIs, aes(x = method, y = estimate)) +
  geom_point() +
  geom_hline(aes(yintercept = 50), lty = 2) +
  geom_segment(aes(xend = method, y = lower_CI, yend = upper_CI)) +
  labs(y = "Estimate with 95% CI")
```



```
# END SOLUTION

# No autograder is provided for this question
```

What does `geom_segment()` do? In particular, what do `x`, `xend`, `y` and `yend` specify in this case?

`geom_segment()` creates a segment of a line that goes between the coordinate denoted by (`x`, `y`) the the one denoted by (`xend`, `yend`).

15. [1 mark] Based on this plot, what can you say about the confidence intervals for the sex ratio in the high stress group?

```
BEGIN QUESTION
name: p15
manual: true
```

The estimated proportion of male births is 36.8%. Because all the CIs overlap the null value of 50%, the null hypothesis of $p_0 = 50\%$ is not rejected at a 5% level. However, most of the values in the CI are <50%, and might suggest that with more data the null hypothesis could be rejected (as the CIs become narrower as sample size increases.)

Submission

For assignments in this class, you'll be submitting using the **Terminal** tab in the pane below. In order for the submission to work properly, make sure that:

1. Any image files you add that are needed to knit the file are in the `src` folder and file paths are specified accordingly.
2. You **have not changed the file name** of the assignment.
3. The file is saved (the file name in the tab should be **black**, not red with an asterisk).
4. The file knits properly.

Once you have checked these items, you can proceed to submit your assignment.

1. Click on the **Terminal** tab in the pane below.
2. Copy-paste the following line of code into the terminal and press enter.

```
cd; cd ph142-sp21/lab/lab10; python3 turn_in.py
```

3. Follow the prompts to enter your Gradescope username and password. When entering your password, you won't see anything come up on the screen–don't worry! This is just for security purposes–just keep typing and hit enter.
4. If the submission is successful, you should see "Submission successful!" appear as output.
5. If the submission fails, try to diagnose the issue using the error messages–if you have problems, post on Piazza.

The late policy will be strictly enforced, **no matter the reason**, including submission issues, so be sure to submit early enough to have time to diagnose issues if problems arise.