Discussion 8

Case study: confidence interval

Jingwei Xiong

UCD statistics

2023-02-27

Case study: confidence interval

$100(1-\alpha)\%$ CI for sample mean μ when σ is unknown

A confidence interval is an interval that contains the population parameter with probability $1-\alpha$. A confidence interval takes on the form:

$$\bar{X} - t_{\alpha/2,N-1}\frac{s}{\sqrt{N}}, \bar{X} + t_{\alpha/2,N-1}\frac{s}{\sqrt{N}}$$

Where $t_{\alpha/2,N-1}$ is the value needed to generate an area of $\alpha/2$ in each tail of a t-distribution with n-1 degrees of freedom.

Interpretation: We are $(1-\alpha)100\%$ confident that the true mean is between the confidence interval.

The required assumptions of the confidence intervals are:

> 1. A random sample was taken from population with unkonwn mean μ and standard deviation σ
>
> 2. Sample mean $\bar{X}$ must be normally distributed.

The second assumption can be satisfied by either:

> sample size $N \geq 30$
>
> or population is normally distributed.

So much for the review, now let's begin the case study.

To start with, we need to first get a population. In this case study, we assume the population follows an exponential distribution with rate $\lambda = 2$.

From our course knowledge we know the exponential distribution has mean $\mu = \frac{1}{\lambda} = 1/2$, standard deviation $\sigma = \frac{1}{\lambda} = 1/2$.

We have already known that using `rexp` we can draw exponential samples.

```
my_sample = rexp(n = 30, rate = 2)
my_sample
```

```
 [1] 0.07427674 0.11551644 0.44610798 0.15497129 0.17133099 0.49432589
 [7] 0.12195484 0.40227292 0.31643956 0.29100271 0.16692751 0.28526091
[13] 0.74585378 0.13317268 0.15485652 1.80018091 0.56629893 0.37514366
[19] 0.61663932 0.31141212 0.02938152 1.20752473 0.16713365 0.66785010
[25] 0.82632120 0.20457522 0.42252089 0.09751961 0.21769481 0.47502838
```

Now let's calculate the confidence interval based on the sample by hand (not really, we use R to calculate).

To calculate a confidence interval, we need the following steps

1. Calculate the mean, standard error of the mean
2. Find the t-score that corresponds to the confidence level
3. Calculate the margin of error and construct the confidence interval

Step 1:

```
sample_mean = mean(my_sample)
sample_n = length(my_sample)
sample_sd = sd(my_sample)
sample_se = sample_sd/sqrt(sample_n)
```

Step 2: Find the t-score that corresponds to the confidence level

We need to have $\alpha/2$ probability in the lower and upper tails, we divide by two because there are two tails.

The `qt()` command will calculate the t-score, $t_{\alpha/2,N-1}$

```
alpha = 0.05
t.score = qt(p=alpha/2, df=sample_n - 1, lower.tail=F)
print(t.score)
```

```
[1] 2.04523
```

Step 3: Calculate the margin of error and construct the confidence interval

The margin of error is $t_{\alpha/2,N-1}\frac{s}{\sqrt{N}}$, which is `t.score` multiplies `sample_se`.

```
margin_error = t.score * sample_se
```

So know we can construct the confidence interval using the mean +/- the margin of error:

```
lower_bound = sample_mean - margin_error
upper_bound = sample_mean + margin_error
print(c(lower_bound, upper_bound))
```

```
[1] 0.2627047 0.5412617
```

Now, we finish all of this calculation. This is very similar with we calculate it in actual exam.

To put them together, you can have a function as:

```
one_sample_mean_CI = function(my_sample, alpha = 0.05){
  sample_mean = mean(my_sample)
  sample_n = length(my_sample)
  sample_sd = sd(my_sample)
  sample_se = sample_sd/sqrt(sample_n)
  t.score = qt(p=alpha/2, df=sample_n - 1, lower.tail=F)
  margin_error = t.score * sample_se
  return(c(sample_mean - margin_error, sample_mean + margin_error))
}
one_sample_mean_CI(my_sample, alpha = 0.05)
```

```
[1] 0.2627047 0.5412617
```

R has already written the function for us:

```
confint( lm(my_sample ~ 1), level = 0.95 )
```

```
                2.5 %    97.5 %
(Intercept) 0.2627047 0.5412617
```

Or you can use `t.test`:

```
t.test(my_sample, level = 0.95)$conf.int
```

```
[1] 0.2627047 0.5412617
attr(,"conf.level")
[1] 0.95
```

Now we have the 95% confidence interval for `my_sample` is 0.263, 0.541.

We know we are 95% confident that the true population mean (actual 0.5) is between 0.263, 0.541. But what does it really means?

Now we want to use R simulation to show you that this means:

> A confidence interval is an interval that contains the population parameter with probability $1 - \alpha$

Or more specifically, if we draw 100 samples from the population, and construct the confidence interval, on average 95 confidence intervals will **cover** that true population mean.

Now we construct 100 confidence intervals and stored in data frame `conf_intervals`.

```
library(tidyverse)
set.seed(114)
n = 100 # simulate 100 times
N = 30 # sample size 30
rate = 2

conf_intervals <- data.frame(index = 1:n, lower = numeric(n), upper =

for (i in 1:n) {
  sample_data <- rexp(n = N, rate = rate)
  conf_interval <- confint( lm(sample_data ~ 1), level = 0.95)
  conf_intervals[i, 2:3] <- conf_interval
}
head(conf_intervals, 3)
```

```
  index     lower     upper
1     1 0.2906168 0.7795004
2     2 0.3464676 0.7167644
3     3 0.3477496 0.6614161
```

We can then identify if confidence interval captured true $\mu = 0.5$

```
conf_intervals = conf_intervals %>%
  mutate(capture = (lower < 0.5) & (upper > 0.5) )
head(conf_intervals, 3)
```

```
  index     lower     upper capture
1     1 0.2906168 0.7795004    TRUE
2     2 0.3464676 0.7167644    TRUE
3     3 0.3477496 0.6614161    TRUE
```

Now how many confidence interval captured the true mean?

```
sum(conf_intervals$capture)
```

```
[1] 94
```

It's very close to 95, right?

Now let's draw the result of whether the confidence intervals captured the true population mean:

```
ggplot(conf_intervals) +
  geom_segment(aes(
    y = index, yend = index, x = lower, xend = upper,
    alpha = factor(capture, levels = c("TRUE", "FALSE"))
  )) +
  geom_point(aes(x = (lower+upper)/2, y = index, color = capture)) +
  labs(
    x = expression("sample mean"),
    y = "Confidence interval number",
    alpha = "Captured"
  ) +
  geom_vline(xintercept = 0.5, color = "red") +
  theme_light() +
  theme(
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.minor.x = element_blank()
  )
```

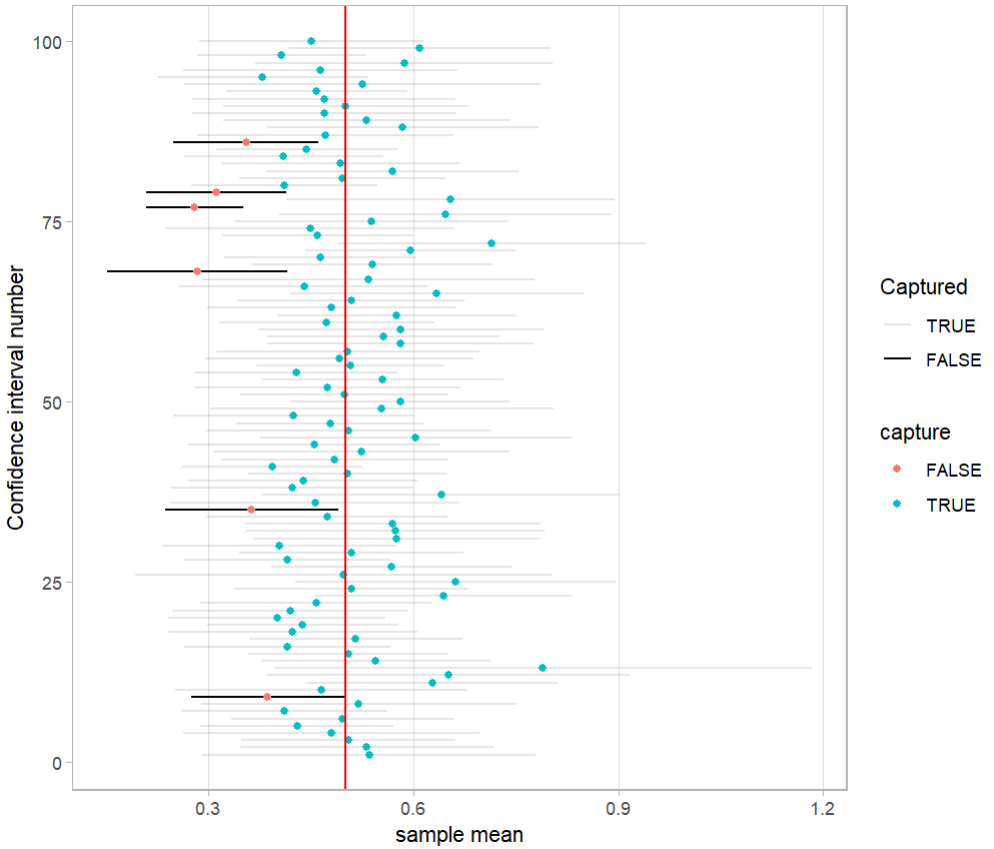
- `ggplot(conf_intervals)` initializes the plot using the conf_intervals data.frame as the data source.

- `geom_segment(aes(y = index, yend = index, x = lower, xend = upper, alpha = factor(capture, levels = c("TRUE", "FALSE"))))` adds the horizontal line segments representing the confidence intervals. The `aes()` function maps the y aesthetic to the index column of the data.frame (which corresponds to the confidence interval number), the `yend` aesthetic to the same index column, the `x` aesthetic to the lower bounds of the intervals, the `xend` aesthetic to the upper bounds of the intervals, and the `alpha` aesthetic (transparency) to the capture column of the data.frame (which indicates whether or not the interval captures the true population mean).

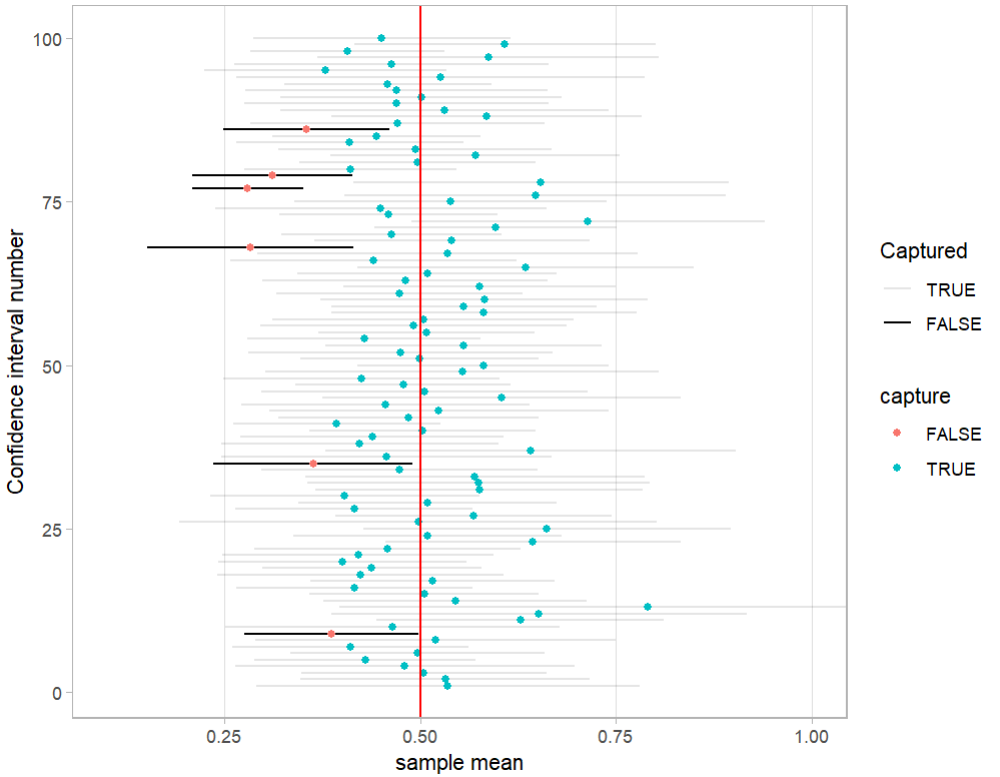
- `geom_point(aes(x = (lower+upper)/2, y = index, color = capture))` adds the colored points representing the midpoints of the confidence intervals.

- `geom_vline(xintercept = 0.5, color = "red")` adds a vertical line at x = 0.5 to represent the true population mean. The xintercept and color arguments specify the position and color of the line, respectively.

- `theme_light()` sets the plot theme to a light background.
- `theme(panel.grid.major.y = element_blank(), panel.grid.minor.y = element_blank(), panel.grid.minor.x = element_blank())` removes the grid lines from the plot.



This is very similar to the picture in lecture 19 note correct? Now you can do this.



This is where the "95% confidence level" comes into play: for every 100 95% confidence intervals, we expect that 95 of them will capture and that five of them won't.

Note that "expect" is a probabilistic statement referring to a long-run average. In other words, for every 100 confidence intervals, we will observe about 95 confidence intervals that capture, but not necessarily exactly 95.

To further accentuate our point about confidence levels, let's generate a figure similar, but this time constructing 75% confidence intervals instead. Let's visualize the results with the scale on the x-axis being the same as previous figure to make comparison easy.

First wrap the code to generate the CIs as the first function:

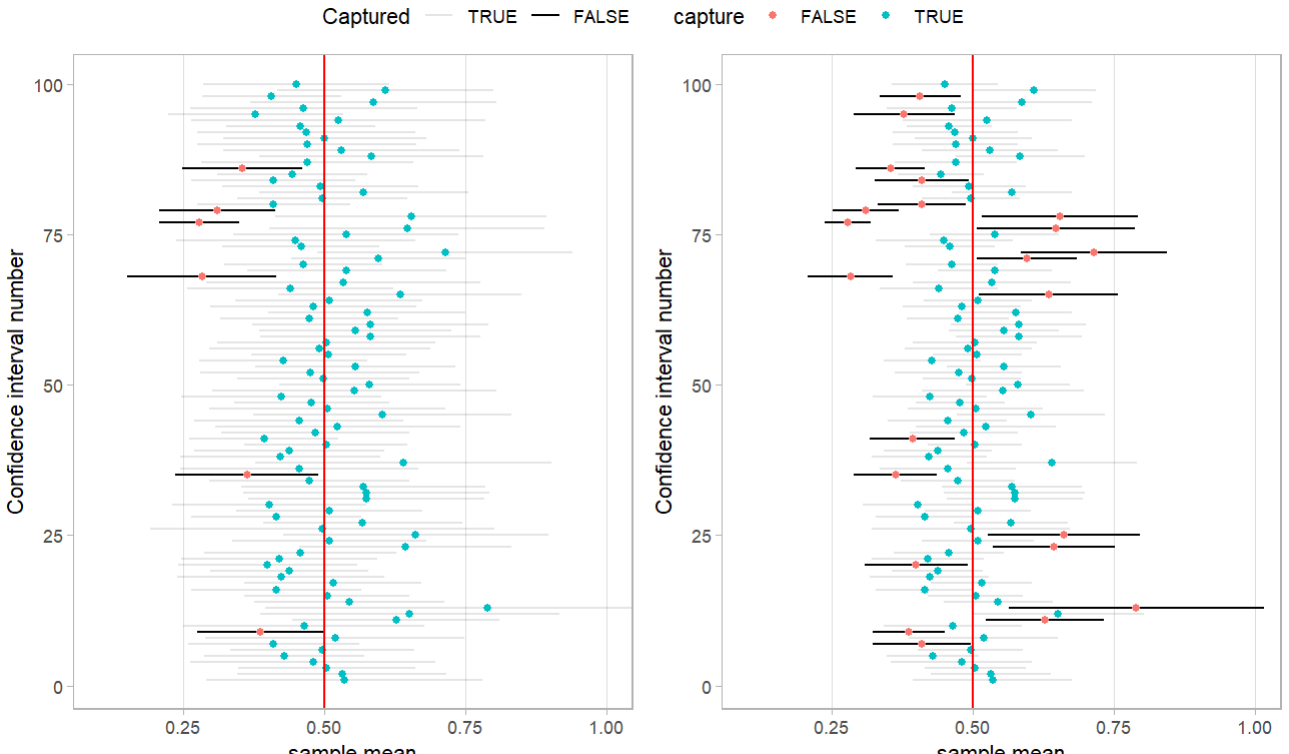
```
generate_CIs = function(seed = 114, n = 100, N = 30, rate = 2, alpha
  set.seed(seed)
  conf_intervals <- data.frame(index = 1:n, lower = numeric(n), upper
  for (i in 1:n) {
    sample_data <- rexp(n = N, rate = rate)
    conf_interval <- confint( lm(sample_data ~ 1), level = alpha)
    conf_intervals[i, 2:3] <- conf_interval
  }
  conf_intervals = conf_intervals %>%
    mutate(capture = (lower < 0.5) & (upper > 0.5) )
  return(conf_intervals)
}
df1 = generate_CIs(seed = 114, n = 100, N = 30, rate = 2, alpha = 0.9

df2 = generate_CIs(seed = 114, n = 100, N = 30, rate = 2, alpha = 0.7
```

Then wrap the plot code into another function

```
plot_function = function(dataframe, population_mean = 0.5, xlim = c(0
  fig = ggplot(dataframe) +
  geom_segment(aes(
    y = index, yend = index, x = lower, xend = upper,
    alpha = factor(capture, levels = c("TRUE", "FALSE"))
  )) +
  geom_point(aes(x = (lower+upper)/2, y = index, color = capture)) +
  labs(
    x = expression("sample mean"),
    y = "Confidence interval number",
    alpha = "Captured"
  ) +
  geom_vline(xintercept = population_mean, color = "red") +
  theme_light() + coord_cartesian(xlim = xlim) +
  theme(
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.minor.x = element_blank()
  )
  return(fig)
}
```

Now we can generate two plots and compare them:

```
library(ggpubr)
df1 = generate_CIs(seed = 114, n = 100, N = 30, rate = 2, alpha = 0.9
df2 = generate_CIs(seed = 114, n = 100, N = 30, rate = 2, alpha = 0.7
fig1 = plot_function(df1)
fig2 = plot_function(df2)
ggarrange(fig1, fig2, ncol = 2, common.legend = TRUE)
```



Width of confidence intervals

Now we want to go over some factors that determine their width.

1. Impact of confidence level

The quantification of the confidence level should match what many expect of the word “confident.” In order to be more confident in our best guess of a range of values, we need to widen the range of values.

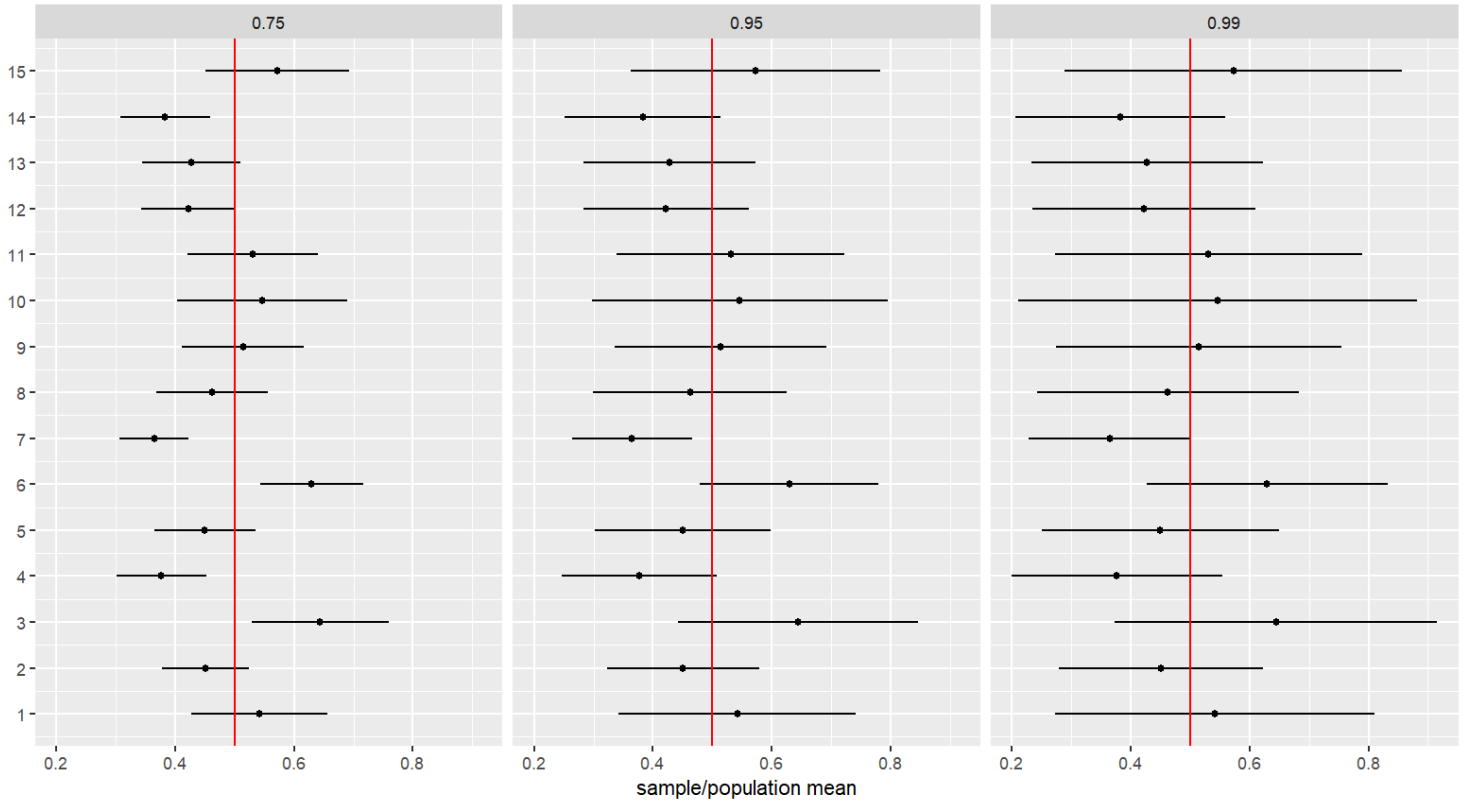
Higher confidence levels tend to produce wider confidence intervals.

1. Impact of sample size

Larger sample sizes tend to produce narrower confidence intervals. Because as the sample size increases, the standard error decreases.

Impact of confidence level

```
df1 = generate_CIs(seed = 1, n = 15, N = 30, rate = 2, alpha = 0.99)
  mutate(confidence_level = 0.99)
df2 = generate_CIs(seed = 1, n = 15, N = 30, rate = 2, alpha = 0.95)
  mutate(confidence_level = 0.95)
df3 = generate_CIs(seed = 1, n = 15, N = 30, rate = 2, alpha = 0.75)
  mutate(confidence_level = 0.75)
# bind the 3 data frames:
df = bind_rows(df1, df2, df3)

ggplot(df) +
  geom_point(aes(x = (lower+upper)/2, y = index)) +
  geom_segment(aes(y = index , yend = index , x = lower, xend = upper
  labs(x = expression("sample/population mean"), y = "") +
  scale_y_continuous(breaks = 1:15) +
  facet_wrap(~confidence_level) +
  geom_vline(xintercept = 0.5, color = "red")
```


0.75
0.95
0.99
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0.2
0.4
0.6
0.8
sample/population mean

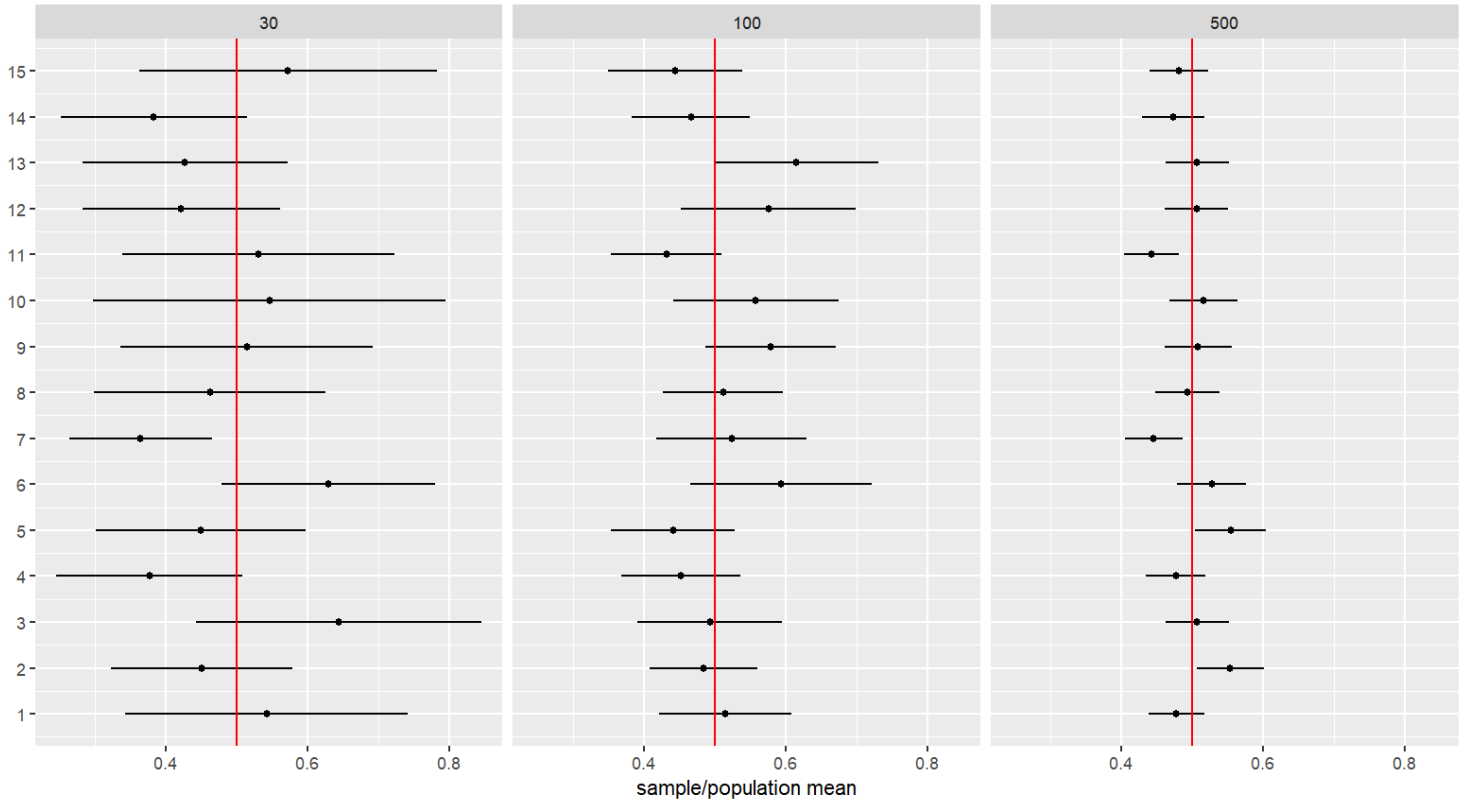
```
df %>%
  mutate(width = upper - lower) %>%
  group_by(confidence_level) %>%
  summarize(`Mean width` = mean(width)) %>%
  rename(`Confidence level` = confidence_level) %>%
  knitr::kable(
    digits = 3,
    caption = "Average width of 75, 95, and 99% confidence intervals'
    booktabs = TRUE,
    longtable = TRUE,
    linesep = ""
  )
```

Table: Average width of 75, 95, and 99% confidence intervals

Confidence level	Mean width
0.75	0.189
0.95	0.329
0.99	0.443

Impact of sample size

```
df1 = generate_CIs(seed = 1, n = 15, N = 30, rate = 2, alpha = 0.95)
  mutate(sample_size = 30)
df2 = generate_CIs(seed = 1, n = 15, N = 100, rate = 2, alpha = 0.95)
  mutate(sample_size = 100)
df3 = generate_CIs(seed = 1, n = 15, N = 500, rate = 2, alpha = 0.95)
  mutate(sample_size = 500)
# bind the 3 data frames:
df = bind_rows(df1, df2, df3)

ggplot(df) +
  geom_point(aes(x = (lower+upper)/2, y = index)) +
  geom_segment(aes(y = index , yend = index , x = lower, xend = upper
  labs(x = expression("sample/population mean"), y = "") +
  scale_y_continuous(breaks = 1:15) +
  facet_wrap(~sample_size) +
  geom_vline(xintercept = 0.5, color = "red")
```


30
100
500
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0.4
0.6
0.8
sample/population mean

```
df %>%
  mutate(width = upper - lower) %>%
  group_by(sample_size) %>%
  summarize(`Mean width` = mean(width)) %>%
  rename(`Sample size` = sample_size) %>%
  knitr::kable(
    digits = 3,
    caption = "Average width of 95% confidence intervals based on $n
    booktabs = TRUE,
    longtable = TRUE,
    linesep = ""
  )
```

Table: Average width of 95% confidence intervals based on $n = 30$, 100, and 500

Sample size	Mean width
30	0.329
100	0.195
500	0.089

Your turn: 1 sample proportion confidence interval

Here is the code of how we calculate the 1 sample proportion confidence interval. Can you make similar plots as we do today?

```
set.seed(123)
n = 100 # simulate 100 times
N = 120 # sample size 120
true_prob = 0.35
alpha = 0.05

conf_intervals <- data.frame(index = 1:n, lower = numeric(n), upper =

for (i in 1:n) {
  sample_data <- rbinom(n = N, size = 1, prob = true_prob)
  sample_prop <- mean(sample_data)
  se_prop <- sqrt((sample_prop*(1 - sample_prop))/N)
  conf_interval <- sample_prop + qnorm(c(alpha/2, 1-alpha/2)) * se_pr
  conf_intervals[i, 2:3] <- conf_interval
}
head(conf_intervals, 3)
```

```
  index     lower     upper
1     1 0.2804464 0.4528869
```