**Confidence Intervals for Means of a Single Population**

**Ex. 1:** Recall from class that confidence intervals give us a technique that supposedly allows us to “trap” the parameter of interest  of the time. As an example, consider 95% confidence intervals for the mean. Then, in other words, if we calculate 100 such confidence intervals, we would expect 95 of them to cover the true mean, while 5 should miss. The first order of business today is to visualize this.

Let’s start out supposing that our data comes from a normal distribution with some mean and some standard deviation. Let’s also suppose for now that we know what the standard deviation is (an unrealistic assumption in the real world). If these facts are true, we know that a  confidence interval for the mean is:



So if I get 100 different random samples of size n, and get 100 different CIs, I can use the following code to visualize this:

mu = 0 # Add the initial settings.

sigma = 5

n = 100

alpha = 0.05

####Initialize the (Blank) Plot##########

plot(x=c(1,125),y=c(-1.5,1.5),type="n",xlab="Different Samples",ylab="Estimated CI for the Mean")

legend("topright",legend=c("Hits","Misses"),col=c("blue","red"),lty=1)

abline(h=0)

##########################################

for(i in 1:100){ # Indicate how many times the loop will run.

x = rnorm(n, mu, sigma) # Generate new data each time.

upper = mean(x) + qnorm(1-alpha/2, 0, 1)\*sigma/sqrt(n) # Find the bounds.

lower = mean(x) - qnorm(1-alpha/2, 0, 1)\*sigma/sqrt(n)

points(x=c(i,i),y=c(lower, upper),type="l",col=ifelse(lower<0 & upper>0,"blue","red"))

}

The figure this generates shows several confidence intervals that are created when random data is generated from a distribution with mean 0. As you can see, most of the intervals contain the value 0 (blue hits), but not all (some are red misses).

**Ex. 2:** OK, now we’ve seen that, in general, we get very close to the 95/5 split for hits/misses. But there were several assumptions/settings in this scenario that we explored: we knew the data were Normal, we had a large sample size (n=100), and we had the unrealistic assumption that we knew sigma. Recall that as long as n is large (>30), it doesn’t matter what the distribution of the data is, nor does it matter if we think we know sigma or simply estimate it with s. But when n is small, we need to assume the data are Normal, and we need to then use the t-distribution to construct our CIs. Let’s explore that now:

############

#

#Case 1(a)

#

#Using Z multipliers, assuming we know sigma

#

############

mu = 10 # Add the initial settings.

sigma = 5

n = 10 #Note the small sample size

alpha = 0.05

successes = 0

for(i in 1:1000){ # Indicate how many times the loop will run.

x = rnorm(n, mu, sigma) # Generate new data each time.

upper = mean(x) + qnorm(1-alpha/2, 0, 1)\*sigma/sqrt(n) # Find the bounds.

lower = mean(x) - qnorm(1-alpha/2, 0, 1)\*sigma/sqrt(n)

if(upper > mu && lower < mu){ # Check to see if we’ve trapped the mean.

successes = successes + 1} # If so, score another success.

}

print(successes)

############

#

#Case 1(b)

#

#Using t multipliers, assuming we know sigma

#

############

mu = 10 # Add the initial settings.

sigma = 5

n = 10 #Note the small sample size

alpha = 0.05

successes = 0

for(i in 1:1000){ # Indicate how many times the loop will run.

x = rnorm(n, mu, sigma) # Generate new data each time.

upper = mean(x) + qt(1-alpha/2, n-1)\*sigma/sqrt(n) # Find the bounds.

lower = mean(x) - qt(1-alpha/2, n-1)\*sigma/sqrt(n)

if(upper > mu && lower < mu){ # Check to see if we’ve trapped the mean.

successes = successes + 1} # If so, score another success.

}

print(successes)

############

#

#Case 2(a)

#

#Using Z multipliers, \*not\* assuming we know sigma

#

############

mu = 10 # Add the initial settings.

sigma = 5

n = 10 #Note the small sample size

alpha = 0.05

successes = 0

for(i in 1:1000){ # Indicate how many times the loop will run.

x = rnorm(n, mu, sigma) # Generate new data each time.

upper = mean(x) + qnorm(1-alpha/2, 0, 1)\*sd(x)/sqrt(n) # Find the bounds.

lower = mean(x) - qnorm(1-alpha/2, 0, 1)\*sd(x)/sqrt(n)

if(upper > mu && lower < mu){ # Check to see if we’ve trapped the mean.

successes = successes + 1} # If so, score another success.

}

print(successes)

############

#

#Case 2(b)

#

#Using t multipliers, \*not\* assuming we know sigma

#

############

mu = 10 # Add the initial settings.

sigma = 5

n = 10 #Note the small sample size

alpha = 0.05

successes = 0

for(i in 1:1000){ # Indicate how many times the loop will run.

x = rnorm(n, mu, sigma) # Generate new data each time.

upper = mean(x) + qt(1-alpha/2, n-1)\*sd(x)/sqrt(n) # Find the bounds.

lower = mean(x) - qt(1-alpha/2, n-1)\*sd(x)/sqrt(n)

if(upper > mu && lower < mu){ # Check to see if we’ve trapped the mean.

successes = successes + 1} # If so, score another success.

}

print(successes)

After running the four code chunks above, you’ll get a number of successes (CI covered the true parameter), which I want you to paste into the table below.

|  |  |  |
| --- | --- | --- |
|  | Using known sigma | Using sample s.d. |
| Using z-multipliers | Paste # from Case 1(a) here | Paste # from Case 2(a) here |
| Using t-multipliers | Paste # from Case 1(b) here | Paste # from Case 2(b) here |

Now, based on these numbers you’ve just generated, discuss what is going on. Perhaps you’ll want to refer back to our conversations in lectures about why it would be necessary to use t rather than z (t accounts for some additional uncertainty/variability – from what? How does that help explain what you’ve seen here?)

**Ex. 3:** For the final simulation study (before we start looking at real data), we’re going to change the distribution from which our data is drawn. Instead of using the normal distribution, we’ll use the exponential distribution – a distribution which is often used to describe the amount of time between events. For our simulation, we’ll use an exponential distribution with mean 1. One interesting property of the exponential distribution is that the mean is always equal to the standard deviation. Thus, we only need to specify one parameter value when we generate our random data. For the exponential distribution, this parameter is defined as 1/mean. Thus, we’ll need to make a few changes to the code.

mu = 1 # Add the initial settings.

sigma = mu

n = 10 #Note the small sample size

alpha = 0.05

successes = 0

for(i in 1:1000){ # Indicate how many times the loop will run.

x = rexp(n, 1/mu) # Generate new data each time.

upper = mean(x) + qt(1-alpha/2, n-1)\*sd(x)/sqrt(n) # Find the bounds.

lower = mean(x) - qt(1-alpha/2, n-1)\*sd(x)/sqrt(n)

if(upper > mu && lower < mu){ # Check to see if we’ve trapped the mean.

successes = successes + 1} # If so, score another success.

}

print(successes)

We’ll now repeat this several times (varying n) to study the effect of sample size using the exponential distribution as opposed to the normal distribution.

Change the sample size to the following:

|  |  |
| --- | --- |
| New value for sample size | Actual number of successes |
| n = 10 |  |
| n = 20 |  |
| n = 30 |  |
| n = 50 |  |

Comment on the results. Why does the confidence interval appear to work as advertised in some cases and not in others? What theorem justifies these results?

**Ex. 4:** Now let’s see what it looks like with real data, when all of the calculations are done internally by R. As I mentioned in class, using z- instead of t-multipliers is technically not correct when we do not know sigma (which, in reality, we never do). As a result, R does not have a built in z-multiplier CI, but always uses t in the function t.test(). So, if you load the ex1\_review.R (upload to RStudio, and remember to use load(file.choose())), you’ll get a dataset called review1, which we’ve seen several times before. There is a variable called Height, and I want you to calculate the default 95% CI by simply using the code:

with(review1, t.test(Height))

This will produce a lot of output, but in the middle you’ll see something labeled the “95 percent confidence interval:”. Report the values here, and write a sentence interpreting what this means. The interpretation sentence will always be of the form “I am 95% confident that \_\_\_\_\_describe\_parameter\_\_\_\_\_ is between \_\_\_lower\_bound\_\_\_ and \_\_\_upper\_bound\_\_\_”.

ALSO, without writing an interpretation, also calculate the 90% CI and the 99% CI by using

with(review1, t.test(Height,conf.level=.9))

with(review1, t.test(Height,conf.level=.99))

Comment on the widths of these intervals relative to the 95% CI.