confidence_intervals_in_R

November 14, 2021

1 Confidence Intervals in R

2 An Exploration of Confidence

Generate a random sample of size 10 from the N(1,2) distribution.
You can do this by typing
`mysamp<-rnorm(10,1,sqrt(2))`
or my generating N(0,1) random variables and "unstandardizing" them by typing
`mysample<-sqrt(2)*rnorm(10)+1`

[7]:
```
mysamp<-rnorm(10,1,sqrt(2))
```

We know that the variance of the distribution this sample came from is 2. Let us suppose that we don't know the mean. Estimate it with the sample mean by typing
`xbar<-mean(mysamp)`

You have found the sample mean and assigned it to a variable called "xbar". View it by typing
`xbar`
in the same cell but on the next line.

[8]:
```
xbar<-mean(mysamp)
xbar
```

1.27917714834265

Let's find the critical values for a 95% confidence interval. We wand to find two values that, when indicated on the x-axis for a standard normal curve, capture area 0.95 between them. This means that we want to find a number that cuts of area 0.95+0.05/2=0.975 to the left and 0.025 to the right. We can get this by typing
`qnorm(0.975)`
Let's call the result "cv" for "critical value".
`cv<-qnorm(0.975)`

[9]:
```
qnorm(0.975)
cv<-qnorm(0.975)
```

1.95996398454005

We are ready to compute the confidence interval!
The lower endpoint is given by
`xbar-cv*sqrt(2/10)`
and the upper endpoint is
`xbar+cv*sqrt(2/10)`

Let's store them in a vector by typing
`myci<-c(xbar-cv*sqrt(2/10),xbar+cv*sqrt(2/10))`
and display it by typing
`myci`

[10]:
```
xbar-cv*sqrt(2/10)
xbar+cv*sqrt(2/10)
myci<-c(xbar-cv*sqrt(2/10),xbar+cv*sqrt(2/10))
myci
```

0.402654607766069

2.15569968891923

1. 0.402654607766069 2. 2.15569968891923

Does your confidence interval contain the true mean of 1 for this sample? It doesn't have to. In fact, 5% of the time it won't! Let's see this in action. Let's look at 100,000 different random samples of size 10. For each sample we will compute a confidence interval and we will keep track of the total number of times the interval contains the true mean of 1.

Begin by initializing a count variable and making a "for loop" by typing
```
count<-0
for(i in 1:100000){
}
```

Just before starting the "for loop", set the appropriate critical value. (It is already set in this jupyter notebook but we will do it again here for completeness of our little piece of code.)
```
count<-0
cv<-qnorm(0.975)
for(i in 1:100000){
}
```

Inside your "for loop", generate a random sample of size 10 from the N(1,2) distribution called "mysamp". Compute the sample mean and call it "xbar".

Check whether or not the resulting confidence interval contains the true mean of 1 and increment your count variable if it does!
`if(xbar-cv*sqrt(2/10)< 1 && xbar+cv*sqrt(2/10)>1){  count<-count+1 }`

[11]:
```
count = 0
cv = qnorm(0.975)
for(i in 1:100000){
    if(xbar-cv*sqrt(2/10)< 1 && xbar+cv*sqrt(2/10)>1){
```

```
        count = count + 1
    }
}
```

Look at the proportion by typing
`count/100000`
What do you see?

[12]:
```
count/100000
```

1

3 Making a Confidence Interval Function

R has built-in functions to make confidence intervals for the mean of a population or the difference in two means. That is, anything confidence interval for a mean or difference of means that requires a t-critical value.In order to get a confidence interval with z-critical values, one would have to load a special package. Instead of doing this, let's work with the base packages in R and write our own function.

In the cell below, type
`normCI<-function(data,variance,level){}`

In **between the braces** (which can be on different lines for clarity) add the lines
`cv<-qnorm(level+(1-level)/2)  xbar<-mean(data)  c(xbar-cv*sqrt(variance/length(data)),xb`

[13]:
```
normCI<-function(data,variance,level){
    cv<-qnorm(level+(1-level)/2)
    xbar<-mean(data)
    c(xbar-cv*sqrt(variance/length(data)),xbar+cv*sqrt(variance/length(data)))

}
```

Now type
`normCI(mysamp,2,0.95)`

Note that you will not get the exact same confidence interval that you originally computed at the beginning of this lab because we have overwritten the vector "mysamp"– many times in fact!

[14]:
```
normCI(mysamp,2,0.95)
```

1. 0.402654607766069 2. 2.15569968891923

4 Built in t-Confidence Intervals in R

Compresive strength of concrete is measured in KN/m^2. A random sample of one type of concrete (cement mixed with pulverized fuel ash) and a random sample of another type of concrete (cement mixed with a new artifical siliceous material produced in a lab) were obtained.

Read in the first random sample from provided data files by typing the following.
`flyash<-read.table("flyash")`
`flyash<-c(unlist(flyash))`
`flyash<-as.vector(flyash)`

Do the same thing for the second sample. The filename for this is 'silicate'.

[15]:
```
flyash<-read.table("flyash")
flyash<-c(unlist(flyash))
flyash<-as.vector(flyash)
silicate<-read.table("silicate")
silicate<-c(unlist(silicate))
silicate<-as.vector(silicate)
```

Assume that the populations are both normally distributed.
Find a 95% confidence interval for the true mean compresive strength of the fly ash mix by typing
`t.test(flyash)`
Can you pick the confidence interval out from this information?

[16]:
```
t.test(flyash)
```

```
One Sample t-test

data:  flyash
t = 81.216, df = 7, p-value = 1.129e-11
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 1355.943 1437.268
sample estimates:
mean of x
 1396.606
```

Suppose that we want to change the default confidence level from 95% to 90%. Type the following
`t.test(flyash',conf.level=0.90)`
Does the width of the resulting confidence interval compare to the width of the previous 95% interval in the way that you expected?

[18]:
```
t.test(flyash,conf.level=0.90)
```

```
One Sample t-test
```

```
data:  flyash
t = 81.216, df = 7, p-value = 1.129e-11
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 1364.026 1429.185
sample estimates:
mean of x
 1396.606
```

Finally, let us do a two-sample t-test to compare the means for both concrete populations by typing
`t.test(flyash,silicate')`

[19]:
```
t.test(flyash,silicate)
```

```
Welch Two Sample t-test

data:  flyash and silicate
t = 2.9765, df = 13.143, p-value = 0.0106
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  17.4014 109.1619
sample estimates:
mean of x mean of y
 1396.606  1333.324
```

Does it appear that the new silcate mix is stronger than the fly ash mix?

You'll notice that the "Welch t-test" was performed. This is the more general test if you can not assume that the populations has equal variances. This is most likely what you will be using in "real life". However, if you would like to perform a "pooled variance test", you would include "var.equal=T" in your last command.

Try this. Is your resulting confidence interval wider or narrower than the Welch confidence interval? Does the relative length make sense to you?

[ ]:
```

```