STA2023 R Labs

Lab 6

Confidence Intervals

```
> #Conf Interval for means, sigma known.
> # xbar plus minus se
> sigma=1.1;n=20; alpha=0.05;xbar=10
> se<-qnorm(1-alpha/2)*sigma/sqrt(n)
> xbar-se;xbar+se
[1] 9.517913
[1] 10.48209
```

Same procedure implementing a function in r:

```
> z.conf.int<-function(xbar,sigma,n,cl){
+  se<-qnorm((1-cl)/2)*sigma/sqrt(n)
+  l=xbar+se
+  r=xbar-se
+  ci=c(l,r)
+  return(ci)
+ }
> z.conf.int(10,1.1,20,.95) # function values (xbar,sigma,n,cl), where cl is
conf.level
[1]  9.517913 10.482087
```

```
# You may use the function z.conf.int(xbar,sigma,n,conf.level) or xbar   plus
minus se
```

```
> # t  conf interval
> s=1.1;n=20; alpha=0.05;xbar=10
> se<-qt(1-alpha/2,n-1)*s/sqrt(n)
> xbar-se;xbar+se
[1] 9.485184
[1] 10.51482
```

Implementing a function:

```
> t.conf.int<-function(xbar,s,n,cl){
+  se<-qt((1-cl)/2, n-1)*s/sqrt(n)
+  l=xbar+se
+  r=xbar-se
+  ci=c(l,r)
+  return(ci)
+ }
> t.conf.int(10,1.1,20,.95)# function values (xbar,s,n,cl), where cl is
conf.level

[1]  9.485184 10.514816
```

```
> x<-c(9.2,9.9,11.1,10.3,13,12.3,9.2,11,10.8) # using sample data.
> t.test(x, mu=10, conf.level = .90) # by default, mu=0 and conf.int =.95 You
may not enter a mu value, since, for now, we are only concerned with the
confidence interval.

        One Sample t-test

data:  x
t = 1.7525, df = 8, p-value = 0.1178
alternative hypothesis: true mean is not equal to 10
90 percent confidence interval:
  9.953868 11.557243
sample estimates:
mean of x
 10.75556

> # 90% conf interval is (9.953868, 11.557243)


 > # one prop conf interval
 > install.packages("epitools")
 > require(epitools)

 > binom.approx(25,200,conf.level = .90)
    x   n proportion      lower     upper conf.level
 1 25 200      0.125 0.08653451 0.1634655        0.9


 > # interval is (0.08653451 0.16346550) This is the normal apprx interval.



 > prop.test(25,200,conf.level = .90)

        1-sample proportions test with continuity correction

 data:  25 out of 200, null probability 0.5
 X-squared = 111, df = 1, p-value < 2.2e-16
 alternative hypothesis: true p is not equal to 0.5
 90 percent confidence interval:
  0.08932699 0.17132419
 sample estimates:
     p
 0.125


> # interval is (0.08932699, 0.17132419) This is Wilson method which is
preferred in the general practice of statistical analisys. In STA2023 we are
using the normal apprx method via binom.approx from “epitools” package.

Two proportions

>#prop.test(x=c(x1,x2), n=c(n1,n2), p = NULL, conf.level = 0.95, correct = F)
# set continuity correction to FALSE.
```

Example:

```
> prop.test(x=c(10,12), n=c(50,48), conf.level = 0.95, correct = F)

2-sample test for equality of proportions without continuity correction

data:  c(10, 12) out of c(50, 48)
X-squared = 0.35167, df = 1, p-value = 0.5532
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.2152222  0.1152222
sample estimates:
prop 1 prop 2
  0.20   0.25

># The 95% conf interval is -0.2152222, 0.1152222
```

Two samples z.test and t.test with summary

BSDA: Basic Statistics and Data Analysis

```
> install.packages("BSDA")
> require(BSDA)

> #zsum.test(mean1,s1,n1,mean2,s2,n2,alt="two.sided", mu = 0, var.equal = F,
conf.level=.95) # Two samples z test with summary statistics.
```

In the practice of statistical analysis, t tests are commonly used. As a rule, we don't know the population mean and standard deviation. In this class the two-samples tests and intervals will be t.tests, no z.tests.

```
> #tsum.test(mean1,s1,n1,mean2,s2,n2,alt="two.sided", mu = 0, var.equal = F,
conf.level=.95) # Two samples t test with summary statistics. Example:

> tsum.test(10,1.1,50,11,.99, 48, alt="two.sided", conf.level=.95)# pool off
since var.equal = F by default.


        Welch Modified Two-Sample t-Test

data:  Summarized x and y
t = -4.7341, df = 95.609, p-value = 7.617e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.4193131 -0.5806869
sample estimates:
mean of x mean of y
       10        11
```

2 samples t.test with data:

```
>#t.test(x,y, conf.level = ?) where x is a vector of values, sample 1:
x<-c(a,b,c etc); y<-c(d,e,f,etc) another vector of values, sample2.
```

Sample size.
In designing experiments, we need to determine how large should the sample size b
e. In R, we use Power Analysis in order to calculate sample size; see
https://www.statmethods.net/stats/power.html

In STA2023 we will implement the following formulas in R:

```
 > #n<-(z_critical*sigma/E)^2  # for means


 For proportions:

 > #n<-(z_critical^2 * phat * qhat/E^2)# qhat=1-phat; If phat unknown, enter 0.5


> #z critical values for common conf intervals:
> # qnorm(1-0.alpha/2) # where alpha is 1-conf. level; example, for 95%, alph
a=0.05


> qnorm(1-0.05/2) #  95%, alpha=0.05
[1] 1.959964
> qnorm(1-0.01/2) #  99%, alpha=0.01
[1] 2.575829
> qnorm(1-0.02/2) #  98%, alpha=0.02
[1] 2.326348
> qnorm(1-0.10/2) #  90%, alpha=0.10
[1] 1.644854
```

Implement a function in R: sample size formula for Estimating a Single Mean

```
> n.sample.mean<-function(conf.level,sigma,error){
+     n=qnorm((1-conf.level)/2)*sigma/error
+     return(ceiling(n^2))
+ }


>#  n.sample.mean(conf.level,sigma,error) # example:

> n.sample.mean(.95,575,134) # sample size for 95% ci, sigma = 575, Error=134
[1] 71
```

Implement a function in R: sample size formula for Estimating a Single Proportion

```
 > n.sample.prop<-function(conf.level,phat,error){
 +     n=(qnorm((1-conf.level)/2))^2*phat*(1-phat)/error^2
 +     return(ceiling(n))
 + }


>#  n.sample.prop(conf.level,phat,error) # example:

> n.sample.prop(.99,0.14,0.04)# sample size for 99% ci, phat=0.14, Error=0.04

[1] 500
```