1.1. **Overview.** We covered two methods of construction confidence intervals means of normal populations. First, in situations where the population standard deviation is known, or we have a large enough sample that the sample standard deviation can be considered reliable (one rule of thumb: $n > 30$; opinions vary a bit on this), the quantity

$$\frac{\overline{x} - \mu}{\sigma}$$

has a standard or $N(0, 1)$ distribution which leads to a $100(1 - \alpha)$ percent confidence interval for μ of:

$$\overline{x} \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

where σ is the (known) population standard deviation, n is the sample size, and $z_{\alpha/2}$ is the number for which

$$P(z \leq z_{\alpha/2}) = 1 - \frac{\alpha}{2}$$

(this is called the *large sample* or σ *known* situation)

If we have neither a large sample nor a reliable estimate of σ, it is better to use the fact that the quantity

$$\frac{\overline{x} - \mu}{s}$$

has a t distribution with $n - 1$ degrees of freedom, which leads to a $100(1 - \alpha)$ percent confidence interval for μ of:

$$\overline{x} \pm t_{\alpha/2}\frac{s}{\sqrt{n}}$$

where s is the sample standard deviation, n is the sample size, and $t_{\alpha/2,n-1}$ is the number for which

$$P(t_{n-1} \leq t_{\alpha/2,n-1}) = 1 - \frac{\alpha}{2}$$

(this is called the *small sample* or σ *unknown* situation)

In this exercise, we will use R to generate simulated samples from a $N(\mu, \sigma)$ population with a specified mean μ and standard deviation

σ. Using these samples, we will construct both the "σ-known" and "σ-unknown" confidence intervals, and observe the proportion of time that the generated intervals contain the true population mean μ. Finally, we will construct confidence intervals (incorrectly) using the sample standard deviation s in place of σ in the "σ-known" variation, to see how far off the proportion of intervals that contain μ is from $1-\alpha$. In the third case, the confidence intervals will be

$$\overline{x} \pm z_{\alpha/2}\frac{s}{\sqrt{n}}$$

Here we treat the sample standard deviation s as if it were the true population standard deviation σ.

1.2. **The R code.** One of the objectives of this assignment is to gain experience using R.

Like many programming languages, R gives you the ability to define your own functions. This feature is tailor made for and exercise like this one and we define a function called "z_versus_t". The function is written with the following arguments (values which are supplied at run time):

- `mu` is the population mean
- `sigma` is the population standard deviation
- `ssize` is the size of an IID sample from a $N(\mu, \sigma)$ population
- `reps` is the number of samples to simulate
- `alpha` is the alpha level

The first line of the code that defines the function is:

```
z_versus_t<-function(mu=0,sigma=1,ssize=10,alpha=0.05,reps=10000)
```

If you invoke the function as

```
z_versus_t()
```

the parameter values used are the ones supplied in the function definition (i.e., these are the default values of none are supplied)

You can override some or all of the defaults when you call the function:

```
z_versus_t(ssize=20,mu=4,sigma=5)
```

would generate samples of size 20 from a $N(4, 5)$ distribution. The `reps` and `alpha` parameters would take their default values of 10,000 and 0.05.

Before you can run the `z_versus_t` function, you need to load it into your R session with the `source()` command.

If you are connected to the internet, you can load it directly from a URL with

```
source("http://www.sandgquinn.org/stonehill/MTH396/Spring2011/z_versus_t.r")
```

Alternatively, you can download the file z_versus_t.r from the website, copy it to the directory where R is running, and use

```
source("z_versus_t.r")
```

To check if the function z_versus_t is loaded, display the contents of the R workspace by typing

```
ls()
```

If the function is in the workspace, it should appear as

```
"z_versus_t"
```

You can remove it from the workspace by typing

```
rm(z_versus_t)
```

1.3. **The Assignment. Question 1** Explain the role of the R function call `qnorm(1-alpha/2,1,0)` in the program

Question 2 Explain the role of the R function call `qt(1-alpha/2,ssize-1)` in the program

Question 3 Explain the role of the R function call `rnorm(ssize,mu,sigma)` in the program

Question 4 Use the `z_versus_t` function to generate 10,000 samples of size 10 from a $N(4, 5)$ population and compute confidence intervals.

- How many of the confidence intervals constructed using the "σ-known" method contained μ?
- How many of the confidence intervals constructed using the "σ-unknown" method contained μ?
- How many of the confidence intervals constructed treating s as σ" contained μ?

Question 5 Repeat question 4 using sample sizes of 5, 20, 30, 40, and 50. If you were designing a "rule of thumb" stating that it is OK to use s in place of σ for samples of k or larger, what value of k would you pick?

1.4. **The R Code for z_versus_t.**

```
#############################################################################
# z_versus_t.r
#
# This simulation program compares confidence intervals computed two ways:
#   1) Using the large-sample or sigma-known approach (normal distributon)
#   2) Using the small-sample or sigma-unknown approach (t distribution)
#   3) Incorrectly using s in place of sigma in the large sample approach
#############################################################################
# Constants:
#  mu    is the population mean
#  sigma is the population standard deviation
#  ssize is the size of an IID sample from a Normal(mu,sigma) population
#  reps  is the number of samples to simulate
#  alpha is the alpha level
#############################################################################
z_versus_t<-function(mu=0,sigma=1,ssize=10,alpha=0.05,reps=10000) {

cat("t versus z simulation: mu=",mu," sigma=",sigma," reps=",reps," alpha=",alpha," ssize=",ssize,"\n")
in_ci_z=0   #number of samples for which CI based on normal contains mu
in_ci_t=0   #number of samples for which CI based on t contains mu
in_ci_w=0   #number of samples for which CI using s in place of sigma contains mu
#
# Generate (reps) samples of size (ssize) from a normal(mu,sigma) population
#
for(i in c(1:reps)){
 x<-rnorm(ssize,mu,sigma)                          #generate sample of size ssize
```

```
 xbar<-mean(x)                                      #sample mean xbar
 s<-sd(x)                                           #sample standard deviation s
 LLz<-xbar-qnorm(1-alpha/2,0,1)*sigma/sqrt(ssize) #Upper limit of CI based on normal distribution, known s
 ULz<-xbar+qnorm(1-alpha/2,0,1)*sigma/sqrt(ssize) #Lower limit of CI based on normal distribution, known s
 if((mu > LLz) && (mu < ULz)) in_ci_z<-1+in_ci_z  #add 1 to counter if CI contains mu
 LLt<-xbar-qt(1-alpha/2,ssize-1)*s/sqrt(ssize)    #lower limit for CI based on t distribution with ssize-1
 ULt<-xbar+qt(1-alpha/2,ssize-1)*s/sqrt(ssize)    #upper limit for CI based on t distribution with ssize-1
 if((mu > LLt) && (mu < ULt)) in_ci_t<-1+in_ci_t  #add 1 to counter if CI contains mu
 LLw<-xbar-qnorm(1-alpha/2,0,1)*s/sqrt(ssize)     #Upper limit of CI based on normal distribution, known s
 ULw<-xbar+qnorm(1-alpha/2,0,1)*s/sqrt(ssize)     #Lower limit of CI based on normal distribution, known s
 if((mu > LLw) && (mu < ULw)) in_ci_w<-1+in_ci_w  #add 1 to counter if CI contains mu
 }
cat("Proportion of CIs containing mu - sigma known (normal distribution)",in_ci_z/reps,"\n")
cat("Proportion of CIs containing mu - sigma unknown (t distribution)",in_ci_t/reps,"\n")
cat("Proportion of CIs containing mu - treating s as known sigma",in_ci_w/reps,"\n")
}
```