R programming for sample size calculations

Cori Mar
Center for Studies in Demography and Ecology

I. Loops in R

Example Loop

```
total <- 0
for (i in 1:5)  {
  total <- total + i
}

total
```

```
## [1] 15
```

Deconstructing the loop

```
total <- 0
for (i in 1:5)  {
  cat("i = ", i, "\n" )
  total <- total + i
  cat("     total = ", total, "\n\n")
}
```

```
## i =  1
##      total =  1
##
## i =  2
##      total =  3
##
## i =  3
##      total =  6
##
## i =  4
##      total =  10
##
## i =  5
##      total =  15
```

Exercise 1

*Write a loop that calculates 1*2*3*4*5*

Loop to add the odd integers 1, 3, . . . , 17

```
total <- 0
for (i in seq(from=1, by=2, to=17))  {
  #cat("i = ", i, "\n" )
  total <- total + i
  #cat("     total = ", total, "\n\n")
}
```

```
total
```

```
## [1] 81
```

Deconstructing Loop to add the odd integers 1, 3, . . . , 17

```
seq(from=1, by=2, to=17)
```

```
## [1]  1  3  5  7  9 11 13 15 17
```

```
total <- 0
for (i in seq(from=1, by=2, to=17))  {
  cat("i = ", i, "\n" )
  total <- total + i
  cat("     total = ", total, "\n\n")
}
```

```
## i =  1
##      total =  1
##
## i =  3
##      total =  4
##
## i =  5
##      total =  9
##
## i =  7
##      total =  16
##
## i =  9
##      total =  25
##
## i =  11
##      total =  36
##
## i =  13
##      total =  49
##
## i =  15
##      total =  64
##
## i =  17
##      total =  81
```

Exercise 2

Use the cat() function to figure out what this loop does

total <- 0
for (i in 1:5) {
total <- total + 2*i
}

II. Random sampling in R

The sample() function
see help(sample)

What's the difference between replace=FALSE and replace=TRUE?

```
set.seed(7)

### replace=FALSE
sample(x=1:20, size=10, replace=FALSE)
```

```
##  [1] 20  8  3  2  4 12  5 13 17  6
```

```
sample(x=1:20, size=10, replace=FALSE)
```

```
##  [1]  4  5 14  2  8 17 16  1 12 20
```

```
sample(x=1:20, size=10, replace=FALSE)
```

```
##  [1] 13  6 18 16 17  1  9  7 12  4
```

```
### replace=TRUE
sample(x=1:20, size=10, replace=TRUE)
```

```
##  [1] 14  6  4  4  8 17 10 16 17 10
```

```
sample(x=1:20, size=10, replace=TRUE)
```

```
##  [1] 16  8 16  9 19  7  2 17 18 20
```

```
sample(x=1:20, size=10, replace=TRUE)
```

```
##  [1] 12 15 16 13 15  8  4  4  8  6
```

In 10 samples with replace=FALSE, how many samples would you expect to contain "1"?

```
set.seed(11)

count <- 0
for (i in 1:10) {
  test <- sample(x=1:20, size=10, replace=FALSE)
  cat("  sample:",test,"\n")
  count <- count + sum(test==1)
  cat("count =",count,"\n\n")
}
```

```
##   sample: 6 1 10 19 2 15 16 4 11 14
## count = 1
##
##   sample: 4 9 17 15 12 19 7 5 2 6
## count = 1
##
##   sample: 5 13 7 6 1 8 17 16 2 20
## count = 2
##
##   sample: 11 7 8 4 14 10 17 2 16 1
## count = 3
##
##   sample: 5 20 9 12 6 13 18 1 7 10
```

```
## count = 4
##
##   sample: 11 12 8 6 4 7 17 20 2 15
## count = 4
##
##   sample: 3 15 4 10 13 20 1 7 19 16
## count = 5
##
##   sample: 14 9 8 4 16 19 17 5 13 3
## count = 5
##
##   sample: 2 10 19 5 14 7 3 9 18 16
## count = 5
##
##   sample: 8 15 1 9 13 4 11 3 20 16
## count = 6
```

```
count
```

```
## [1] 6
```

In 100 samples with replace=FALSE, how many samples would you expect to contain "1"?

```
set.seed(17)

count <- 0
for (i in 1:100) {
  test <- sample(x=1:20, size=10, replace=FALSE)
  count <- count + sum(test==1)
}
count
```

```
## [1] 56
```

```
count <- 0
for (i in 1:100) {
  test <- sample(x=1:20, size=10, replace=FALSE)
  count <- count + sum(test==1)
}
count
```

```
## [1] 48
```

```
count <- 0
for (i in 1:100) {
  test <- sample(x=1:20, size=10, replace=FALSE)
  count <- count + sum(test==1)
}
count
```

```
## [1] 56
```

```
count <- 0
for (i in 1:100) {
  test <- sample(x=1:20, size=10, replace=FALSE)
  count <- count + sum(test==1)
}
count
```

```
## [1] 50
```

Nesting loops

```
set.seed(22)

count.vec <- NULL
for (j in 1:10) {
  count <- 0
  for (i in 1:100) {
    test <- sample(x=1:20, size=10, replace=FALSE)
    count <- count + sum(test==1)
  }

count.vec <- c(count.vec, count)
cat("j =",j, " ;  count.vec =", count.vec,"\n")
}
```

```
## j = 1  ;  count.vec = 50
## j = 2  ;  count.vec = 50 57
## j = 3  ;  count.vec = 50 57 54
## j = 4  ;  count.vec = 50 57 54 50
## j = 5  ;  count.vec = 50 57 54 50 41
## j = 6  ;  count.vec = 50 57 54 50 41 50
## j = 7  ;  count.vec = 50 57 54 50 41 50 57
## j = 8  ;  count.vec = 50 57 54 50 41 50 57 48
## j = 9  ;  count.vec = 50 57 54 50 41 50 57 48 46
## j = 10  ;  count.vec = 50 57 54 50 41 50 57 48 46 56
```

```
count.vec
```

```
##  [1] 50 57 54 50 41 50 57 48 46 56
```

```
mean(count.vec)
```

```
## [1] 50.9
```

```
sd(count.vec)
```

```
## [1] 5.195083
```

III. Writing a function

```
get.sample.count <- function() {
   count <- 0
  for (i in 1:100) {
    test <- sample(x=1:20, size=10, replace=FALSE)
    count <- count + sum(test==1)
  }
return(count)
}
```

Calling the function

```
set.seed(481)

get.sample.count()
```

```
## [1] 58
```

```
get.sample.count()
```

```
## [1] 48
```

```
get.sample.count()
```

```
## [1] 52
```

```
set.seed(481)

get.count <- get.sample.count()
get.count
```

```
## [1] 58
```

Passing values to the function

```
get.sample.count2 <- function(number) {
   count <- 0
  for (i in 1:100) {
    test <- sample(x=1:20, size=10, replace=FALSE)
    count <- count + sum(test==number)
  }
return(count)
}
```

Count the number of "1"s

```
set.seed(481)

get.sample.count2(number=1)
```

```
## [1] 58
```

```
get.sample.count2(number=1)
```

```
## [1] 48
```

```
get.sample.count2(number=1)
```

```
## [1] 52
```

Count the number of "2"s

```
set.seed(481)

get.sample.count2(number=2)
```

```
## [1] 44
```

```
get.sample.count2(number=2)
```

```
## [1] 41
```

```
get.sample.count2(number=2)
```

```
## [1] 60
```

Exercise 3

A. Using the same seed (481), what is the number of times "19" occurs in the hundred samples

B. Using the same seed (481), what is the number of times "7" occurs in the hundred samples

C. Using the same seed (481), what is the number of times each of 1:20 in the hundred samples

Passing and returning multiple values to a function

This function returns a vector of counts: the number of times each of 1:20 (integers from 1 to 20) occurs in the sample.

```
get.sample.count3 <- function(sample.size=10, num.samples=100) {

  count.vec <- rep(0,20)
  for (i in 1:num.samples) {

    test <- sample(x=1:20, size=sample.size, replace=FALSE)

    for (j in 1:20) {
      count.vec[j] <- count.vec[j] + sum(test==j)
    }
  }

  return(count.vec)
}
```

Testing the function with the defaults (same as before)

```
set.seed(481)

get.sample.count3()
```

```
##  [1] 58 44 50 55 42 55 45 57 43 46 47 52 51 57 52 50 45 47 48 56
```

Increasing the number of samples to 1000

```
set.seed(481)

get.sample.count3(num.samples=1000)
```

```
##  [1] 511 497 509 522 495 487 496 518 496 491 467 494 518 507 500 481 494
## [18] 490 493 534
```

Increasing the number of samples to 1000 and decreasing the sample size to 5

What is the new expected value for the count for each integer from 1 to 20?

```r
set.seed(481)

get.sample.count3(sample.size=5, num.samples=1000)
```

```
##  [1] 268 245 243 261 249 245 263 251 239 248 259 251 246 255 217 252 247
## [18] 252 258 251
```

IV. Calculating power and sample size

Classic problem: Do two samples come from the same population or two different populations? Another way to put it: Assuming that the two samples come from two different populations, how big must the sample size be to detect this? Often this is done just for a single populations statistic, commonly the mean, with some assumption about the shape of the population distribution, e.g., that it's normal. Or you don't care about the population distribution, only its mean, so you take advantage of the Central Limit Theorem which enables you to assume that the distribution of the sample means is normal, even if the population values are not, given that the sample size is large enough, although how large is large enough is often unknown.

Intuitively, you need a smaller sample size to detect a large difference between two populations and a larger sample size if the two populations are very similar.

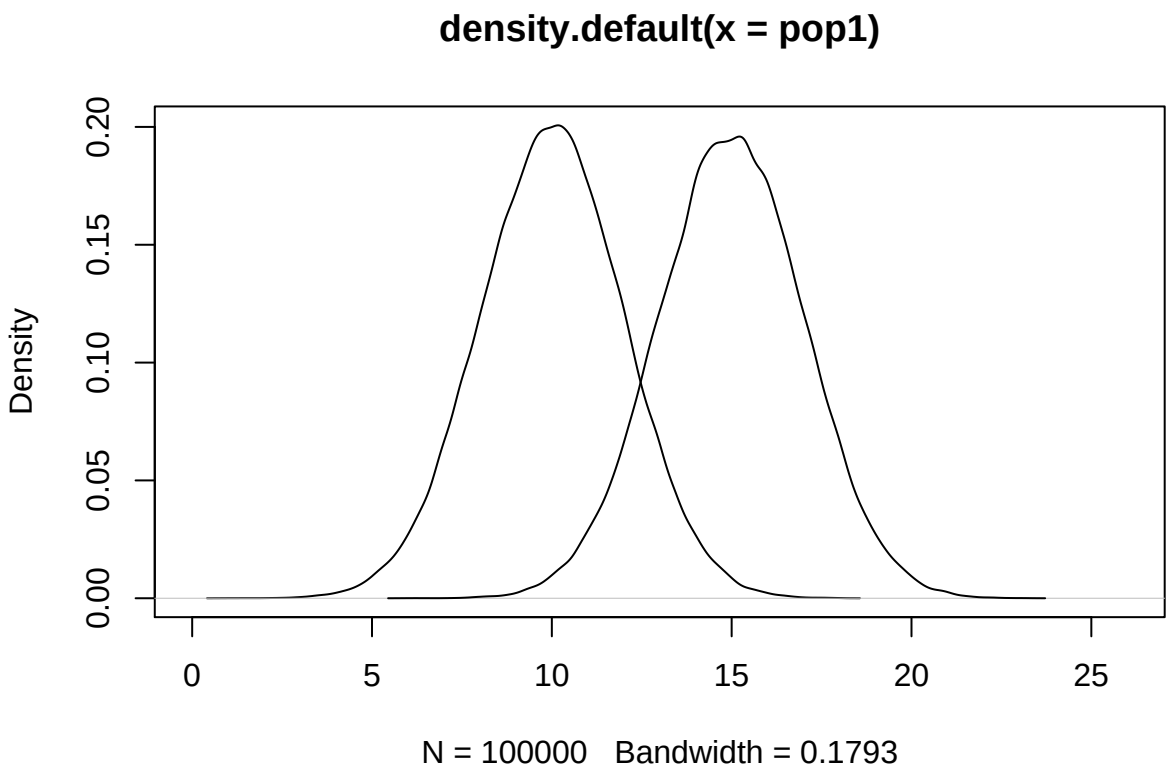
Let's start with a demonstration of the above using R's function that samples from a normal distribution of specified mean and standard deviation: see help(rnorm)

First: two populations very different from each other.

```r
set.seed(314)

pop1 <- rnorm(n=100000, mean=10, sd=2)
pop2 <- rnorm(n=100000, mean=15, sd=2)

plot(density(pop1), xlim=c(0,26))
lines(density(pop2))
```

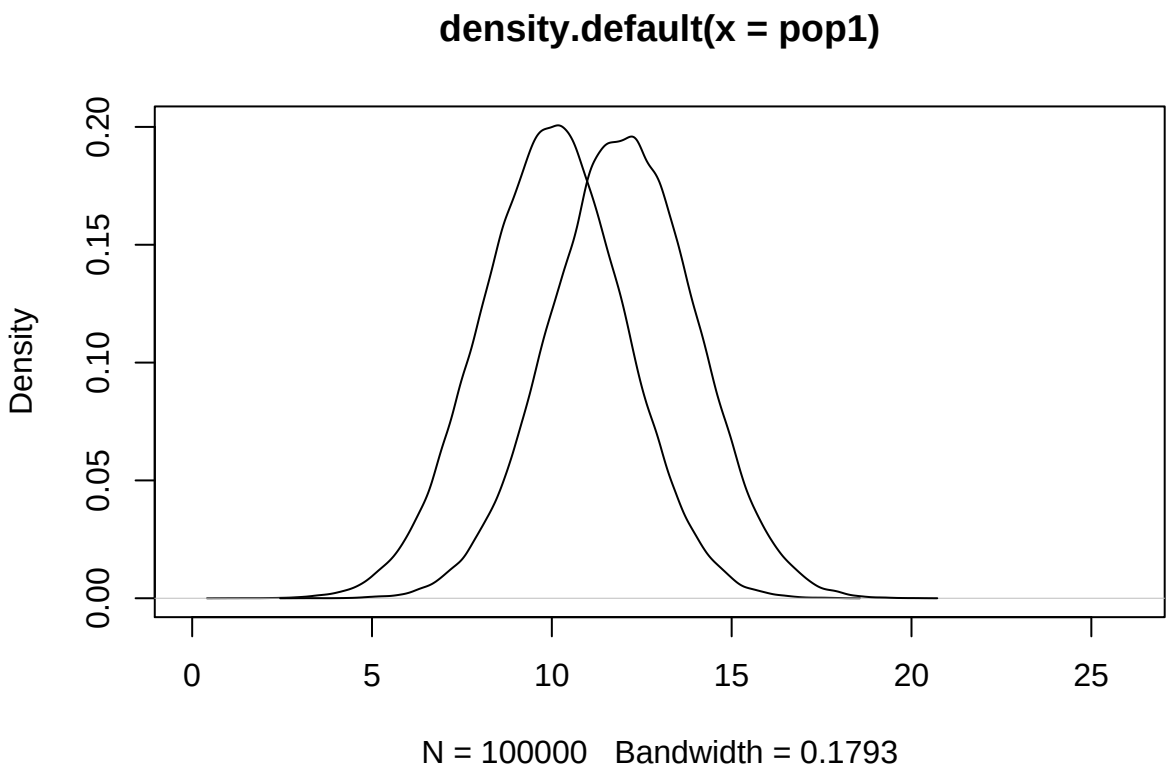


Second: two populations much closer to each other.

```
set.seed(314)

pop1 <- rnorm(n=100000, mean=10, sd=2)
pop2 <- rnorm(n=100000, mean=12, sd=2)

plot(density(pop1), xlim=c(0,26))
lines(density(pop2))
```



Recall the simplest two-sample t-test: independent samples, equal sample sizes, equal variance.
(see, for example, https://en.wikipedia.org/wiki/Student%27s_t-test)

null hypothesis: mean(population1) == mean(population2) alternative hypothesis: mean(population1) != mean(population2)
therefore two-tailed test

t <- (mean(sample1) - mean(sample2)) / (pooled.sd * sqrt(2/n))

df <- 2*n - 2

where

n = sample size

pooled.sd <- sqrt((sd(sample1)^2 + sd(sample2)^2)/n)

Outline of function

1. Pass in two populations, sample size, number of samples, alpha level
2. for each sample

- sample1, sample2
- calculate t
- compare to critical value (two-tailed)
- reject or not reject null

3. return number of rejections

```
simple.t <- function(pop1, pop2, n, num.samples, alpha) {

  t.crit <- qt(p=1-alpha/2, df=2*n-2)

  reject.count <- 0
```

```
  for (i in 1:num.samples) {

    sample1 <- sample(pop1, size=n, replace=TRUE)

    sample2 <- sample(pop2, size=n, replace=TRUE)

    pooled.sd <- sqrt( ( sd(sample1)^2 + sd(sample2)^2 )/n )

    t.value <- ( mean(sample1) - mean(sample2) ) / ( pooled.sd * sqrt(2/n) )

    if (abs(t.value) > t.crit) reject.count <- reject.count + 1
  }

  return(reject.count)
}
```

Running the code for populations with large difference, num.samples = 1000, alpha=.05.

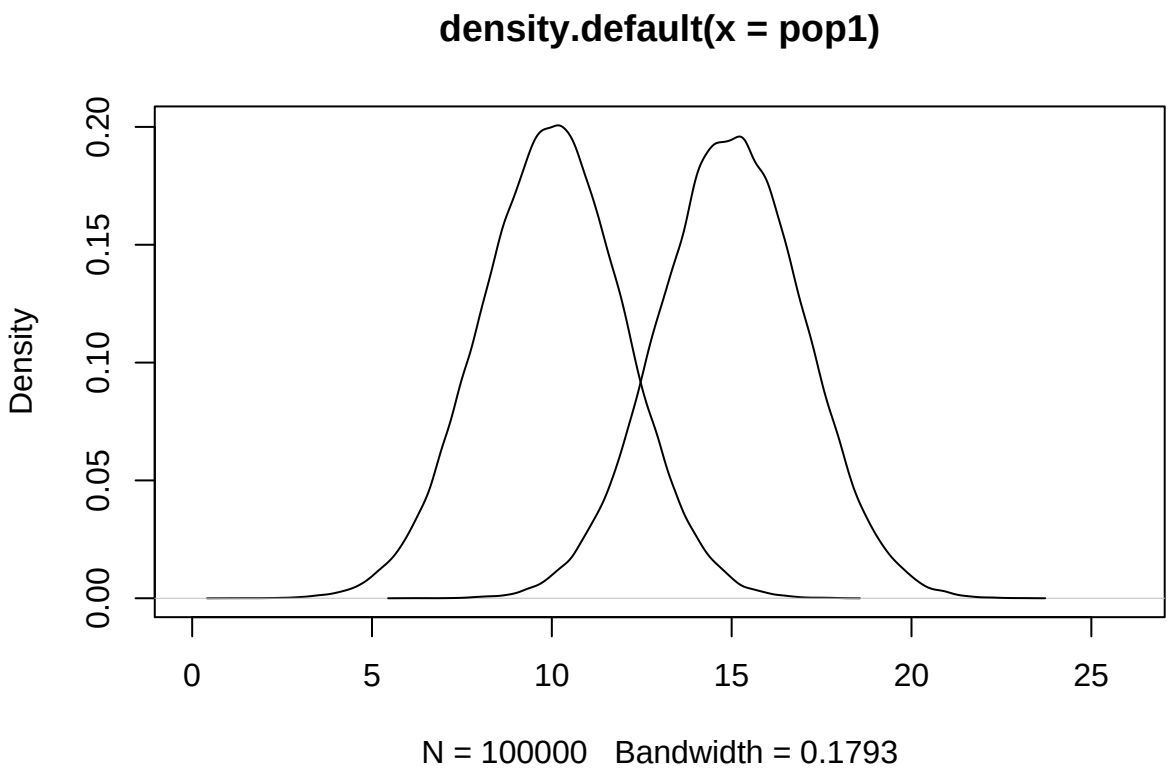
Very small sample size necessary.

```
source("function.code.R")

num.samples <- 1000
alpha <- .05

set.seed(314)
pop1 <- rnorm(n=100000, mean=10, sd=2)
pop2 <- rnorm(n=100000, mean=15, sd=2)

plot(density(pop1), xlim=c(0,26))
lines(density(pop2))
```




```
### very small sample
n <- 5
simple.t(pop1, pop2, n, num.samples, alpha)
```

```
## [1] 993
```

Running the code for populations with smaller difference, num.samples = 1000, alpha=.05.

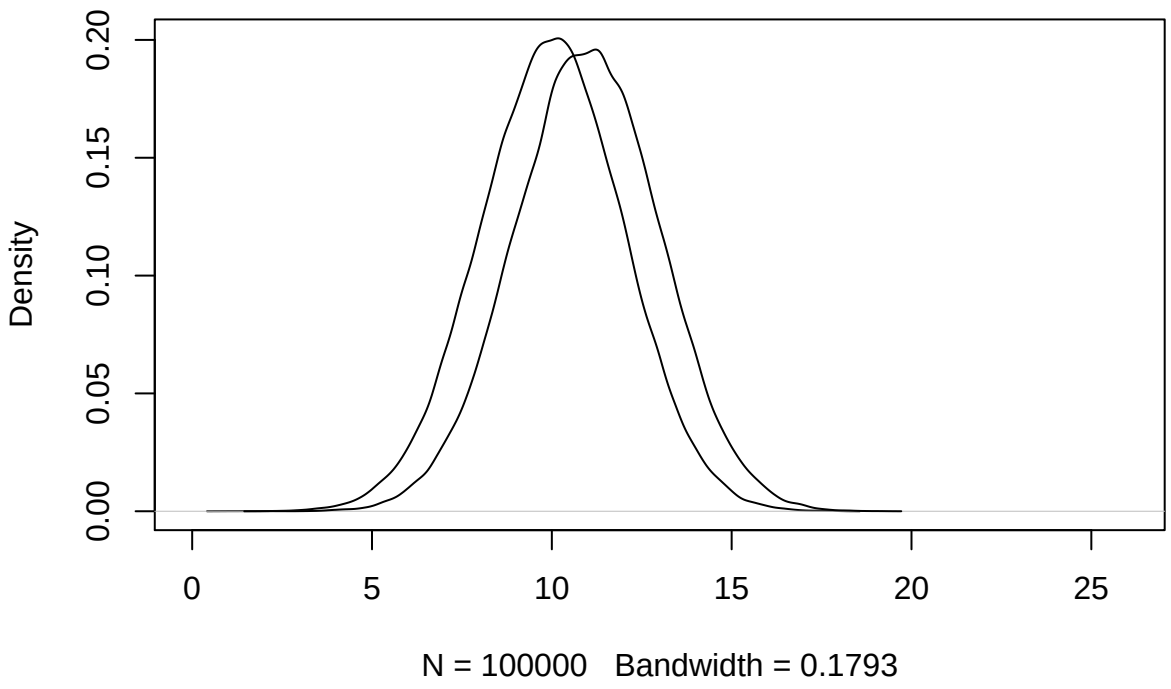
Bigger sample size necessary.

```
source("function.code.R")

alpha <- .05
num.samples <- 1000

set.seed(314)
pop1 <- rnorm(n=100000, mean=10, sd=2)
pop2 <- rnorm(n=100000, mean=11, sd=2)

plot(density(pop1), xlim=c(0,26))
lines(density(pop2))
```




```
### very small sample
n <- 5
simple.t(pop1, pop2, n, num.samples, alpha)
```

```
## [1] 316
```

```
### slightly bigger sample
n <- 10
simple.t(pop1, pop2, n, num.samples, alpha)
```

```
## [1] 593
```

```
### still bigger sample
n <- 15
simple.t(pop1, pop2, n, num.samples, alpha)
```

```
## [1] 738
```

```
### sample with >80% power
n <- 20
simple.t(pop1, pop2, n, num.samples, alpha)
```

```
## [1] 846
```

Running the code for populations with even smaller difference, num.samples = 1000, alpha=.05.

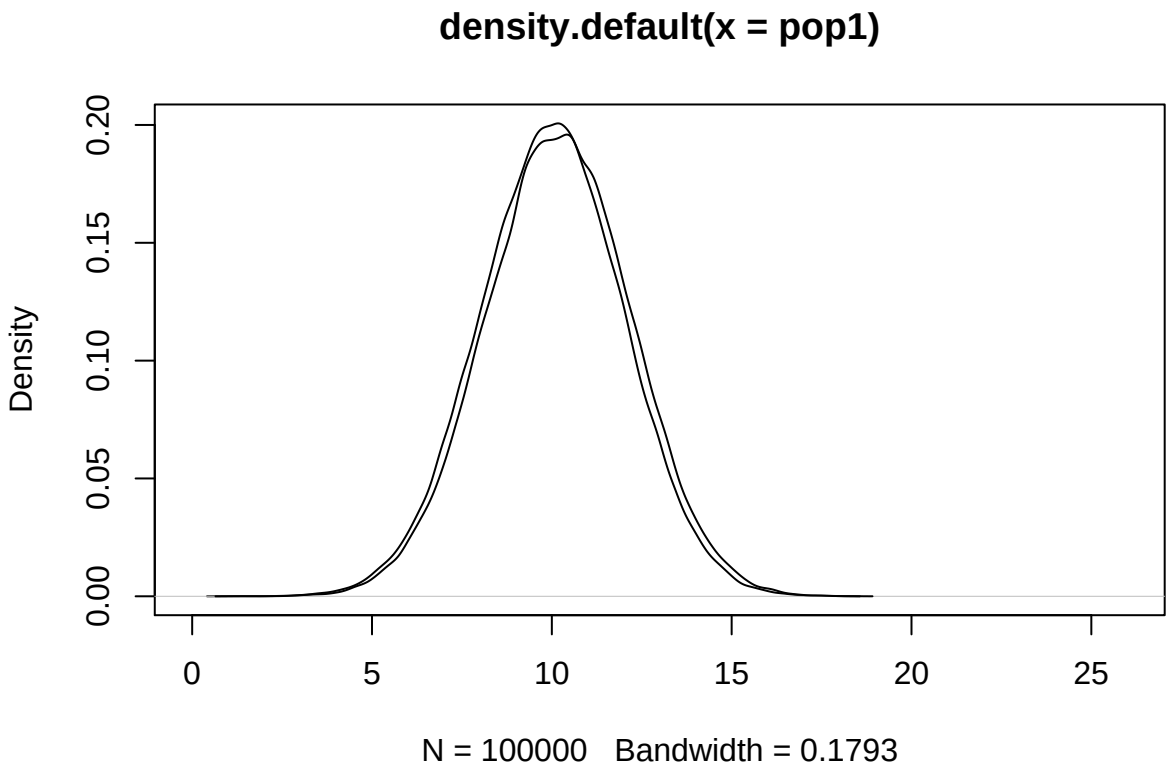
```
source("function.code.R")

alpha <- .05
num.samples <- 1000

set.seed(314)
```

```r
pop1 <- rnorm(n=100000, mean=10, sd=2)
pop2 <- rnorm(n=100000, mean=10.2, sd=2)

plot(density(pop1), xlim=c(0,26))
lines(density(pop2))
```

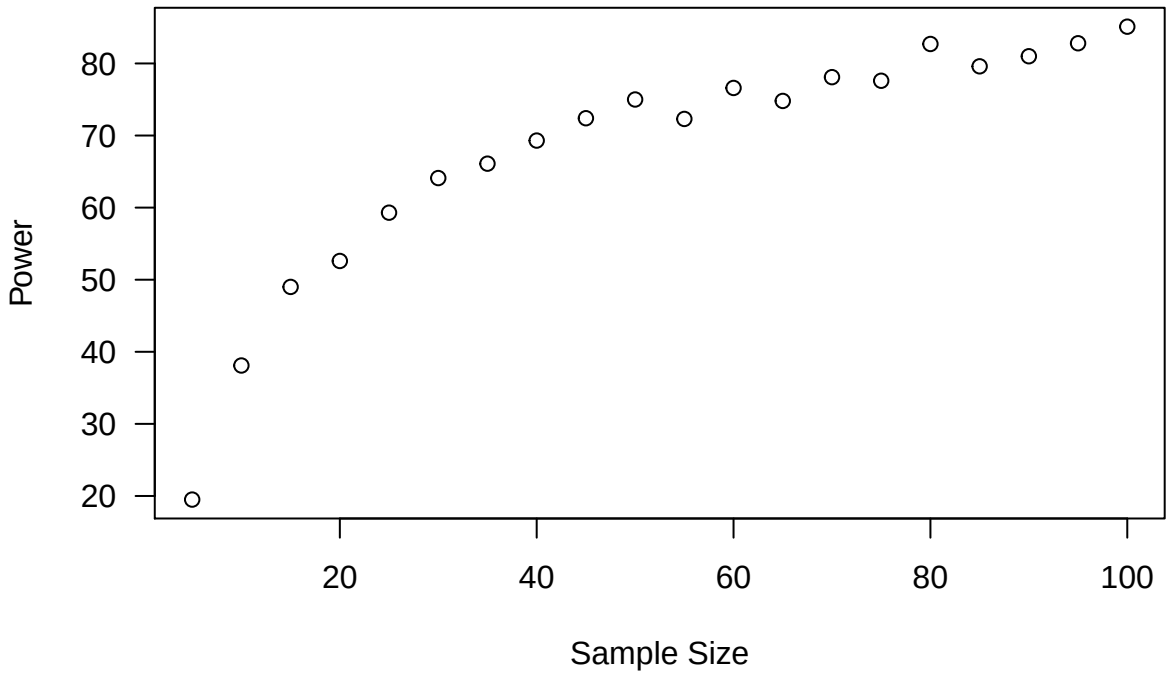



```r
reject.vec <- NULL
for (n in seq(5,100,by=5)) {

  num.reject <- simple.t(pop1, pop2, n, num.samples, alpha)

  reject.vec <- c(reject.vec, num.reject)
}

plot(seq(5,100,by=5), (reject.vec/1000)*100, xlab="Sample Size",ylab="Power",las=1)
```



Exercise 4

What do you need to do to smooth out this plot?

About what size sample gives 80% power?