**Sampling Distributions**

In this lab, we will begin by reviewing what we’ve already learned about the sampling distribution for a sample proportion (), and extend those ideas to explore the sampling distribution for a sample mean ().

**Sample Proportions**

Recall that when we learned about the Normal approximation to the Binomial distribution, we said that for X~Bin(n,p), if np and n(1-p) were both large enough (some say they both need to be at least 5, others say 10), then XN(mean=np, variance=np(1-p)), and therefore that:

N(mean=p, variance=p(1-p)/n)

Note that the only thing that is approximate here is the Normality – those formulas for the mean and variance are exact. So let’s explore this with a simple example. Think of the students in the classroom as a population, for which we can look around and determine the true population proportion (p) of females. Do that now, and create some data in R that records whether each person is a female:

gender=c(rep(1, enter number of females here), rep(0, enter number of males here))

mean(gender) ##this tells you p!!

5/mean(gender) ##solving for n if we want np=5

5/(1-mean(gender)) ##solving for n if we want n(1-p)=5

Now suppose I couldn’t observe the entire population (as is usually the case in reality) and I randomly sample n people from the population. We’ve talked in class about the idea that I could look at all possible sets of n (=2 or 3 or any other positive integer) and obtain lots of different values of and that therefore has a probability distribution (which we call a sampling distribution). Let’s see this actually happen. The following code will generate all possible sets of size n=2, 3, 4, 5 and 10.

####

#Get all possible samples of size 2

####

n=length(gender)

out2=numeric(choose(n,2))

count=1

for (i in 1:(n-1)){

for (j in (i+1):n){

out2[count]=mean(gender[c(i,j)])

count=count+1

}}

####

#Get all possible samples of size 3

####

n=length(gender)

out3=numeric(choose(n,3))

count=1

for (i in 1:(n-2)){

for (j in (i+1):(n-1)){

for (k in (j+1):n){

out3[count]=mean(gender[c(i,j,k)])

count=count+1

}}}

####

#Get all possible samples of size 4

####

n=length(gender)

out4=numeric(choose(n,4))

count=1

for (i in 1:(n-3)){

for (j in (i+1):(n-2)){

for (k in (j+1):(n-1)){

for (l in (k+1):n){

out4[count]=mean(gender[c(i,j,k,l)])

count=count+1

}}}}

####

#Get all possible samples of size 5

####

n=length(gender)

out5=numeric(choose(n,5))

count=1

for (i in 1:(n-4)){

for (j in (i+1):(n-3)){

for (k in (j+1):(n-2)){

for (l in (k+1):(n-1)){

for (m in (l+1):n){

out5[count]=mean(gender[c(i,j,k,l,m)])

count=count+1

}}}}}

####

#Get all possible samples of size 10

####

n=length(gender)

out10=numeric(choose(n,10))

count=1

for (i in 1:(n-9)){

for (j in (i+1):(n-8)){

for (k in (j+1):(n-7)){

for (l in (k+1):(n-6)){

for (m in (l+1):(n-5)){

for (o in (m+1):(n-4)){

for (p in (o+1):(n-3)){

for (q in (p+1):(n-2)){

for (r in (q+1):(n-1)){

for (t in (r+1):n){

out10[count]=mean(gender[c(i,j,k,l,m,o,p,q,r,t)])

count=count+1

}}}}}}}}}}

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

Now that we’ve drawn all possible samples of size n=2, 3, 4, 5 and 10, let’s explore the Normality, and see how it changes as a function of the size of the sample we’re drawing from the population:

par(mfrow=c(3,2))

hist(out2)

hist(out3)

hist(out4)

hist(out5)

hist(out10)

What do you notice?

What about the mean and variance? I said above that these are exact formulas. Let’s see!!!

mean(out2) ##compare to mean(gender)

var(out2) ##compare to p(1-p)/n, given below 🡪 why don’t they match?

mean(gender)\*(1-mean(gender))/2

mean(out3) ##compare to mean(gender)

var(out3) ##compare to p(1-p)/n, given below 🡪 why don’t they match?

mean(gender)\*(1-mean(gender))/3

mean(out4) ##compare to mean(gender)

var(out4) ##compare to p(1-p)/n, given below 🡪 why don’t they match?

mean(gender)\*(1-mean(gender))/4

mean(out5) ##compare to mean(gender)

var(out5) ##compare to p(1-p)/n, given below 🡪 why don’t they match?

mean(gender)\*(1-mean(gender))/5

mean(out10) ##compare to mean(gender)

var(out10) ##compare to p(1-p)/n, given below 🡪 why don’t they match?

mean(gender)\*(1-mean(gender))/10

What do you notice about the means? Why do the variances not match? Recall that when we derived this formula for the variance, we required that all of the data be independent with the same probability distribution. When we sample from a (small) finite population, the observations no longer have the same probability distribution. If we wanted a bunch of independent observations, we could do something like:

ind=rbinom(100000,2,mean(gender))/2

mean(ind)

var(ind)

**Sample Means**

Now let’s replicate some of these concepts for sample means. Let’s go around the room and each student can report his/her height, and we’ll record them:

hts=c(70,75,73,64,63,69,68,73,70,70,72,66,65,73,68,65,72,72,66)

hts=c(66,74,77,66,64,70,69,70,76,70,74,69,76,64,65,69,68)

n=length(hts)

####

#Get all possible samples of size 2

####

out2=numeric(choose(n,2))

count=1

for (i in 1:(n-1)){

for (j in (i+1):n){

out2[count]=mean(hts[c(i,j)])

count=count+1

}}

####

#Get all possible samples of size 3

####

out3=numeric(choose(n,3))

count=1

for (i in 1:(n-2)){

for (j in (i+1):(n-1)){

for (k in (j+1):n){

out3[count]=mean(hts[c(i,j,k)])

count=count+1

}}}

####

#Get all possible samples of size 4

####

out4=numeric(choose(n,4))

count=1

for (i in 1:(n-3)){

for (j in (i+1):(n-2)){

for (k in (j+1):(n-1)){

for (l in (k+1):n){

out4[count]=mean(hts[c(i,j,k,l)])

count=count+1

}}}}

####

#Get all possible samples of size 5

####

out5=numeric(choose(n,5))

count=1

for (i in 1:(n-4)){

for (j in (i+1):(n-3)){

for (k in (j+1):(n-2)){

for (l in (k+1):(n-1)){

for (m in (l+1):n){

out5[count]=mean(hts[c(i,j,k,l,m)])

count=count+1

}}}}}

####

#Get all possible samples of size 10

####

out10=numeric(choose(n,10))

count=1

for (i in 1:(n-9)){

for (j in (i+1):(n-8)){

for (k in (j+1):(n-7)){

for (l in (k+1):(n-6)){

for (m in (l+1):(n-5)){

for (o in (m+1):(n-4)){

for (p in (o+1):(n-3)){

for (q in (p+1):(n-2)){

for (r in (q+1):(n-1)){

for (t in (r+1):n){

out10[count]=mean(hts[c(i,j,k,l,m,o,p,q,r,t)])

count=count+1

}}}}}}}}}}

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

Now that we’ve drawn all possible samples of size n=2, 3, 4, 5 and 10, let’s explore the Normality, and see how it changes as a function of the size of the sample we’re drawing from the population:

par(mfrow=c(3,2))

hist(out2)

hist(out3)

hist(out4)

hist(out5)

hist(out10)

What do you notice?

What about the mean and variance? Let’s see!!!

truemean=mean(hts)

trueSD=sqrt((n-1)\*var(hts)/n)

mean(out2)

sd(out2)

trueSD/sqrt(2) ##--> WHY??

mean(out3)

sd(out3)

trueSD/sqrt(3)

mean(out4)

sd(out4)

trueSD/sqrt(4)

mean(out5)

sd(out5)

trueSD/sqrt(5)

mean(out10)

sd(out10)

trueSD/sqrt(10)

**Time Permitting: Bootstrap Approximation to Sampling Distribution**

Recall that we interpret probabilities as long-term relative frequencies. So relatively speaking, that one largest possible mean height shouldn’t happen often. So if I randomly sample n observations from the dataset, and do it a lot of times, I’ll see roughly the same proportion of those large number of repetitions having a certain characteristic (e.g. that one largest possible mean height) as there are in the total set of possible samples of size n. This is the idea behind the bootstrap – I can get a really good approximation to the sampling distribution of a statistic in a simple way: just resample the observed data a lot of times (often 1000, 5000 or 10000) with replacement (so everything is independent and identically distributed) and calculate the statistic of interest on each resampled dataset. Collect them, and you’ve got a lot of observations from the distribution of the statistic of interest.

##Non-Parametric Bootstrap

outB=numeric(5000)

for (i in 1:5000){

tmp=sample(hts,10,replace=TRUE)

outB[i]=mean(tmp)

}

dev.new()

par(mfrow=c(1,2))

hist(out10)

hist(outB)

mean(outB)

sd(outB)

trueSD/sqrt(10)