# Remember to save it to your T drive

Due date is next Tuesday

# set directory to your own for saving plots

Q1a: Run this code from the class10-supplement:

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

# Sampling from a Normal Model and comparison with Population via qqnorm

#pdf("Rplots.pdf")

par(mfrow=c(2,2))

set.seed(1)

# Population normal with mean mu and standard deviation .5

mu=1.5 ; sd = .5

u =seq(-3,3,.01) ; x=mu+sd\*u ; fx = dnorm(x,mu,sd)

plot(x,fx,main="Normal Population mu=1.5,sd=.5",xlab="x", ylab="Density")

# Sample of n=200 data points from population

n=200

y = rnorm(n,mu,sd)

hist(y,freq=FALSE,main="Sample Data")

o=hist(y,freq=F)

hist(y,freq=FALSE,main=" Population + Sample",ylim=range(fx,o$density))

lines(x,fx,col=2)

# See how well it fits

# qq-plot Comparison between Population Density and Histogram

# Data quantiles y\_(1)< y\_(2) < .... , y\_(n) [ordered y-data values ]

# correspond to 100(i+.5)/(n+1) for i=1,2, .... , n quantiles in population

#

squant = sort(y)

ps = ((1:n)+.5)/(n+1)

pquant=qnorm(ps,mu,sd) ; pquant01=qnorm(ps,0,1)

#plot 1:

plot(pquant01,pquant,ylab="Quantiles Normal: mu,sd ",

xlab="Quantiles Normal: 0, 1"); abline(a=mu,b=sd,col=4,lwd=2)

#plot 2:

plot(pquant01,squant,xlab="Population",ylab="Data",main="Direct Normal QQ Plot")

#plot 3:

qqnorm(y); abline(lm(squant~pquant01)$coef,col=4)

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

Q1b: In the plot with the title 'Population + Sample' what is the red curve plotting?

Q1c: In the plot title 'Normal Q-Q plot' what are we checking for?

Q1d: In the plot title 'Normal Q-Q plot' why did we show the blue line?

Q1e: What are we trying to demonstrate in plot 1?

Q1f: Run this code but (at the beginning) change mu to 2 and sd to 1.

Q2a: # Run this SAMPLING DISTRIBUTION code

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

# If n values are sampled from a population data = { y\_1, ... , y\_n}

# and we compute a statistic - e.g. the sample average y\_bar

# How will y\_bar vary if a different set of n values were sampled?

# Case 1: Data coming from a Normal population

# Case 2: Data coming from a non-Normal Population

set.seed(1)

par(mfrow=c(2,3))

# Case 1 Illustration : Data coming from a Normal population

mu=3 ; sd=.5

yy=rnorm(1000000,mu,sd) ; popmean=mean(yy) ; popsd=sd(yy) # pop mean and sd

hist(yy,main="Population")

print(round(c(mu,popmean,sd,popsd),2))

ns=500 ; n =20 ; mu=popmean ; sd=popsd

ybars=NULL

for(s in (1:ns) ) { y=rnorm(n,mu,sd) ; ybar=mean(y)

ybars = c(ybars,ybar)}

hist(ybars,freq=F)

sy=sort(ybars) ; lines(sy,dnorm(sy,mu,sd/sqrt(n))) # Theory: ybar N: mu, sd/sqrt(n)

qqnorm(ybars)

#### Theoretical Result:

#### sample mean is normal with same mean and standard deviation = sd/sqrt(n)

# Case 2 Illustration: Here data are coming from a non-Normal Population Uniform

set.seed(1)

a=6; b=14 ; yy=runif(100000,a,b) ; popmean=mean(yy) ; popsd=sd(yy) # pop mean and sd

hist(yy,main='Population')

print(round(c((a+b)/2,popmean,(b-a)/sqrt(12),popsd),2))

ns=500 ; n =20 ; mu=popmean ; sd=popsd

ybars=NULL

for(s in (1:ns) ) { y=runif(n,a,b) ; ybar=mean(y)

ybars = c(ybars,ybar)}

hist(ybars,freq=F)

sy=sort(ybars) ; lines(sy,dnorm(sy,popmean,popsd/sqrt(n))) # Theory: ybar N: mu, sd/sqrt(n)

qqnorm(ybars)

#### Theoretical Result:

#### sample mean is approximately normal

#### with same mean and standard deviation = sd/sqrt(n)

#### CENTRAL LIMIT THEORM (CLT) : Averages will have an approximate Normal

#### Sampling Distribution

### for more detail a good explanation is in

# http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704\_Probability/BS704\_Probability12.html

#dev.off()

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

Q2b: If you used an exponential distribution with parameter lambda=6 instead of a uniform in Case 2 what do you think the mean and sd values would be in the histogram? (Hint: if the rate of an exponential is 1/lambda then the mean is lambda and the sd is also lambda).

Q2c: What is the theorem that tells you the answer to Q2b?

Q2d: Try it for the exponential instead of the uniform for Case 2. Set lambda to 6. For the exponential you need the function rexp with 'rate=1/lambda'). Then to print the means use

print(round(c(lambda,popmean,lambda,popsd),2))

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

Q3a: Run this code which demonstrates the bootstrap method:

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

# If the population structure is unknown, it is possible to use the Bootstrap (Efron)

# device to approximate the sampling distribution of statistics.

# The idea is to treat the sample data as the population distribution (histogram) and

# sample from that 'bootstrapped population' to simulate sampling from the true population.

# Illustation: Use of the Bootstrap to Approximate the sampling distribution of

# the sample mean and standard deviation.

setwd('~/Desktop')

#pdf('RplotsB.pdf')

set.seed(1)

par(mfrow=c(1,2))

# Step 1:

# Calculate a very large sample which is like a 'population':

# Suppose the true is exponential [eg waiting time characteristic]

yy = rexp(100000,rate=1/7) ; syy=sort(yy)

hist(yy,freq=F,main="Population");lines(syy,dexp(syy,1/7))

# Step 2:

# Calculate a small sample which is like the type of sample you might get in an experiment:

# A sample of size 100 is taken and the mean and standard deviation

# is calculated.

n=100 ; y = sample(yy,n,replace=T) ;hist(y,main="Sample")

ymean=mean(y) ; ysd=sd(y)

# Step 3:

# Now we first look at the type of samples we might find from the true population:

# A. TRUE SAMPLING DISTRIBUTION OF sample mean and sd from exponential

NS=10000 ; ymeansT =NULL ; ysdsT=NULL

for( s in 1:NS) { ys = sample(yy,n,replace=T)

ymeansT=c(ymeansT,mean(ys)) ; ysdsT=c(ysdsT,sd(ys))}

# Step 4:

# then we first look at the type of samples we might find if we use our smaller dataset.

# B. Bootstrap SAMPLING DISTRIBUTION OF sample mean and sd from exponential

NB=700 ; ymeansB =NULL ; ysdsB=NULL

for( b in 1:NB) { yb = sample(y,n,replace=T)

ymeansB=c(ymeansB,mean(yb)) ; ysdsB=c(ysdsB,sd(yb))}

par(mfrow=c(2,3))

hist(ymeansT,xlab="mean",main="True")

print(round(c(mean(ymeansT),sd(ymeansT)),3))

hist(ymeansB,xlab="mean",main="Bootstrap")

print(round(c(mean(ymeansB),sd(ymeansB)),3))

qqplot(ymeansT,ymeansB,main="qq-plot",xlab="TRUE-quantiles",ylab="Boot-quantiles")

hist(ysdsT,xlab="sd",main="");hist(ysdsB,xlab="sd",main="")

qqplot(ysdsT,ysdsB,main="qq-plot",xlab="TRUE-quantiles",ylab="Boot-quantiles")

#dev.off()

# NB:

# We see that even though we started with a small sample (from Step 2) to calculate the

# 'sampling distribution' (in Step 4) we end up with something reasonably close to

# the sampling distribution in Step 3 (which is based on the 'true' population.

# Bootstrap procedure works well if the sample data is representative

# of the true population [works better with larger samples]

Q3b: Re-run this with n=1000 in Step 2.