# midterm 2008, Q1

# part b (i)

# methodology :

# 1. Generate the random variate form given distribution

# 2. Calculate the test statistic

# 3. Count no of times test statistic is > 4.605. This count/ total no of simulations gives the power of the test.

# no. of simulations

nsim <-1000

# sample size given

n<-50

#double exponential random variate generation

#mean =0 , and stdev <- sqrt(2)

mn <- 0

stdev <- sqrt(2)

# Initializing count variable

count<- 0

# Running the simulation

for (j in 1:nsim) {

# generating double exponential variate through prob integral transformation.

#suniform <- runif(n, min=-0.5, max=0.5)

#s<- mn-((abs(suniform)/suniform)\*(log(1-2\*(abs(suniform)))))

s <- rexp(n,1)-rexp(n,1)

# hist(s)

s\_mn <- mean(s)

s\_err <- (s-s\_mn)

s\_err2 <- (sum(s\_err^2))/n

s\_err3 <- (sum(s\_err^3))/n

s\_err4 <- (sum(s\_err^4))/n

s\_g1 <- s\_err3/(s\_err2^1.5)

s\_g2 <- (s\_err4/(s\_err2^2))-3

test\_statistic <- (n/6)\*((s\_g1^2) + ((s\_g2^2)/4))

#cat (test\_statistic,"\n")

if (test\_statistic > 4.605) { count <- count+1 }

}

power\_of\_test <- (count/nsim)\*100

power\_of\_test

**RESULT : POWER IS 58%. Since the double exponential is a little close normal distribution, the test is not able fully reject the DE(0,1) 99% of the times.**

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

# part b (ii)

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

# no. of simulations

nsim <-1000

# sample size given

n<-50

# Initializing count variable

count<- 0

# Running the simulation

for (j in 1:nsim) {

# generating t-variate with 30 degrees of freedom

s<- rt(n,30)

s\_mn <- mean(s)

s\_err <- (s-s\_mn)

s\_err2 <- (sum(s\_err^2))/n

s\_err3 <- (sum(s\_err^3))/n

s\_err4 <- (sum(s\_err^4))/n

s\_g1 <- s\_err3/(s\_err2^1.5)

s\_g2 <- (s\_err4/(s\_err2^2))-3

test\_statistic <- (n/6)\*((s\_g1^2) + ((s\_g2^2)/4))

if (test\_statistic > 4.605) { count <- count+1 }

}

power\_of\_test <- (count/nsim)\*100

power\_of\_test

**RESULT : POWER IS 10.4%. Since The power is still lower, given the fact that t-distribution is very much similar to normal at 30 deg of freedom. So the test is not able reject with high confidence.**

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

# part c

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

# no. of simulations

nsim <-1000

# sample size given

n<-100

# for plotting power of test again p values.

p<- seq(0,0.5,by=.05)

power\_of\_test <- seq(1:length(p))

for (i in length(p)){

# Initializing count variable

count<- 0

# Running the simulation

for (j in 1:nsim) {

# generating p\*N(0,1)+(1-p)\*N(2,3)

s<- p[i]\*rnorm(n,mean=0, sd=1) + (1-p[i])\*rnorm(n,mean=2, sd=sqrt(3))

# hist(s)

s\_mn <- mean(s)

s\_err <- (s-s\_mn)

s\_err2 <- (sum(s\_err^2))/n

s\_err3 <- (sum(s\_err^3))/n

s\_err4 <- (sum(s\_err^4))/n

s\_g1 <- s\_err3/(s\_err2^1.5)

s\_g2 <- (s\_err4/(s\_err2^2))-3

test\_statistic <- (n/6)\*((s\_g1^2) + ((s\_g2^2)/4))

if (test\_statistic > 4.605) { count <- count+1 }

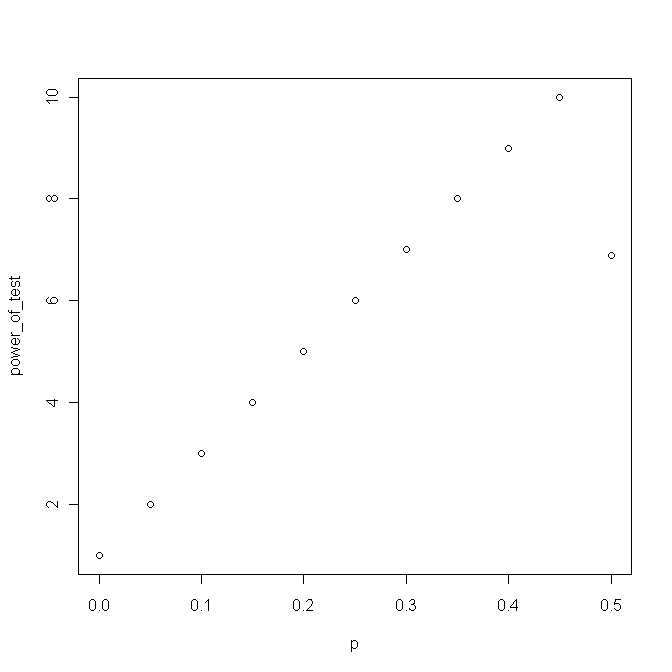
}

power\_of\_test[i] <- (count/nsim)\*100

}

plot(p,power\_of\_test)

**RESULT: The graph shows it is a monotonously increasing function as a function of p , with a discontinuity at p=0.5**

****

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

# part d

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

# no. of simulations

nsim <-1000

# for plotting power of test again n values.

n<-c(200,300,400,500,750,1000)

power\_of\_test <- seq(1:length(n))

for (i in length(n)){

# Initializing count variable

count<- 0

# Running the simulation

for (j in 1:nsim) {

# generating p\*N(0,1)+(1-p)\*N(2,3) ; p is fixed at 0.01

p<-0.01

s<- p\*rnorm(n[i],mean=0, sd=1) + (1-p)\*rnorm(n[i],mean=2, sd=sqrt(3))

# hist(s)

s\_mn <- mean(s)

s\_err <- (s-s\_mn)

s\_err2 <- (sum(s\_err^2))/n[i]

s\_err3 <- (sum(s\_err^3))/n[i]

s\_err4 <- (sum(s\_err^4))/n[i]

s\_g1 <- s\_err3/(s\_err2^1.5)

s\_g2 <- (s\_err4/(s\_err2^2))-3

test\_statistic <- (n[i]/6)\*((s\_g1^2) + ((s\_g2^2)/4))

if (test\_statistic > 4.605) { count <- count+1 }

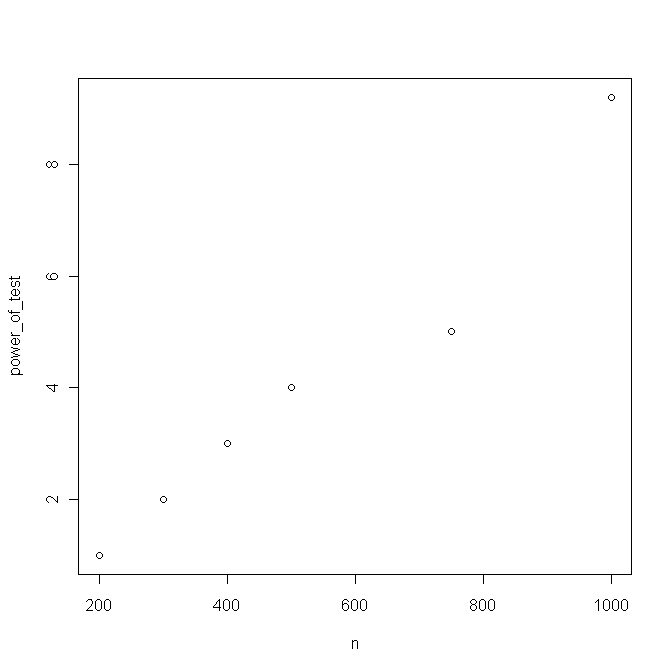
}

power\_of\_test[i] <- (count/nsim)\*100

}

plot(n,power\_of\_test)

**RESULT : Ability of test to reject when Ha is true goes up with size of sample, as it gets more information in the sample.**

****

**Key observations & findings:**

1. **Ability of Jarque-bera to reject distributions like DE and t is low due to similarality of those distributions with normal distribution.**
2. **But as sample size increases the power to reject goes high.**
3. **Even for a mixed normal as p increases which means X has a single normal N(0,1) , power to reject otherwise increases.**

# midterm 2008, Q2

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

# part a

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

**# From analytical derivation 2\*(2n<x> - n(ln(2\*e\*<x>)) is chi-square(1)**

# simulation

nsim <- 1000

# sample size

n<-40

# test\_statistic initialization

test\_statistic <- seq(1:nsim)

for (i in 1:nsim) {

# generating exp variate with lamba=2

s<- rexp(n,2)

s\_mean <- mean(s)

test\_statistic[i] <- 2\*(2\*n\*s\_mean - n\*(log(2\*exp(1)\*s\_mean)))

}

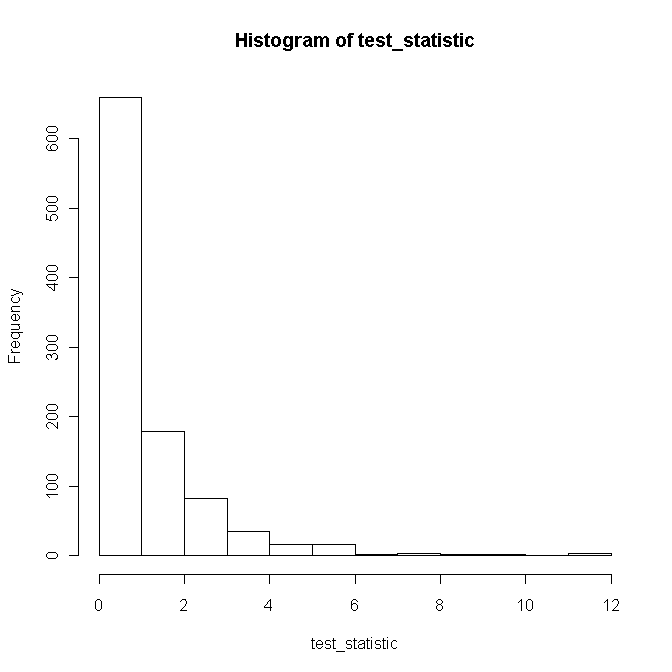
hist(test\_statistic)

# getting the 95% quantile as the cutoff

cutoff <- quantile(test\_statistic,0.95)

cutoff

**RESULT : The test\_statistic is as follows . A 95% cutoff is 3.86**

****

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

# part b

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

# simulation

nsim <- 1000

# sample size

n<-40

# count initialization

count <- 0

# test\_statistic initialization

test\_statistic <- seq(1:nsim)

for (i in 1:nsim) {

# generating exp variate with lamba=2

s<- rexp(n,1)

s\_mean <- mean(s)

test\_statistic <- 2\*(2\*n\*s\_mean - n\*(log(2\*exp(1)\*s\_mean)))

# using cutoff of "3.86" obtained from part(a)

if (test\_statistic > 3.86) { count <- count+1 }

}

power\_of\_test <- (count/nsim)\*100

power\_of\_test

**RESULT : power is 99.3%. Since the test is specifically designed for distinguishing lamba differences it is performing well.**

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

# part c

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

# sample size

n<- 40

nsim<-1000

# sample of exp(2)

s<-rexp(n,2)

s\_mean <- mean(s)

# parametric bootstrap

s\_bt\_lambda <- seq(1:nsim)

for (i in 1:nsim) {

# lamba = 1/s\_mean

s\_bootstrap <- rexp(n,(1/s\_mean))

s\_bt\_lambda[i] <- 1/mean(s\_bootstrap) }

hist(s\_bt\_lambda)

se\_lamba\_lower\_bound <- quantile(s\_bt\_lambda,0.025)

se\_lamba\_upper\_bound <- quantile(s\_bt\_lambda,0.975)

#stdev\_s\_bt\_lamba <- sd(s\_bt\_lamba)

#se\_lamba\_lower\_bound <- (1/s\_mean)-1.96\*stdev\_s\_bt\_lamba

#se\_lamba\_upper\_bound <- (1/s\_mean)+1.96\*stdev\_s\_bt\_lamba

cat("paramteric bootstrap : lower bound:",se\_lamba\_lower\_bound,"upper bound:",se\_lamba\_upper\_bound," \n")

# Non- parametric bootstrap

s\_bt\_lambda<- rep(0,nsim)

for (i in 1:nsim) {

# lamba = 1/s\_mean

s\_bootstrap <- sample(s,n, replace=TRUE)

s\_bt\_lambda[i] <- 1/mean(s\_bootstrap) }

hist(s\_bt\_lambda)

se\_lamba\_lower\_bound <- quantile(s\_bt\_lambda,0.025)

se\_lamba\_upper\_bound <- quantile(s\_bt\_lambda,0.975)

cat("paramteric bootstrap : lower bound:",se\_lamba\_lower\_bound,"upper bound:",se\_lamba\_upper\_bound," \n")

**RESULT : paramteric bootstrap : lower bound: 1.580313 upper bound: 2.493695**

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

#part e(i)

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

# Data actually comes from gamma(2,4)

# simulation

nsim <- 1000

# sample size

n<-40

# count initialization

count <- 0

# test\_statistic initialization

test\_statistic <- seq(1:nsim)

for (i in 1:nsim) {

# generating gamma variate with shape 2 and rate =2

s<- rgamma(n,2,4)

s\_mean <- mean(s)

test\_statistic <- 2\*(2\*n\*s\_mean - n\*(log(2\*exp(1)\*s\_mean)))

# using cutoff of "3.86" obtained from part(a)

if (test\_statistic > 3.86) { count <- count+1 } }

power\_of\_test <- (count/nsim)\*100

power\_of\_test

# Data actually comes from gamma(2,2)

# simulation

nsim <- 1000

# sample size

n<-40

# count initialization

count <- 0

# test\_statistic initialization

test\_statistic <- seq(1:nsim)

for (i in 1:nsim) {

# generating gamma variate with shape 2 and rate =2

s<- rgamma(n,2,2)

s\_mean <- mean(s)

test\_statistic <- 2\*(2\*n\*s\_mean - n\*(log(2\*exp(1)\*s\_mean)))

# using cutoff of "3.86" obtained from part(a)

if (test\_statistic > 3.86) { count <- count+1 } }

power\_of\_test <- (count/nsim)\*100

power\_of\_test

**RESULT: power of the test is 100%.**

**# Sample data for part c is**

0.35464726 0.84054386 0.33029537 0.05806520 0.57206834 0.33109086 0.79583994 0.55643561 0.29413124 0.09706847 0.37849162 0.07955111 0.12420339 0.01378694 0.90147302 0.36579466 1.04815769 0.05268678 0.71912030 0.24658099 0.65729199 0.30321275 0.16597101 1.02850789 1.12175549 0.35368603 0.27197910 1.27648600 0.68338176 0.29712345 0.62046640 1.41061306 0.50614943 0.19416457 0.20478059 0.83077741 1.07550163 0.36572701 0.83366178 0.19505689

# midterm 2008, Q3

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

# part a

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

males<- c(905.8,1012.9,1202.8,1161.6,1022.2,926.5,768.4,648.0,651.4,622.9,579.0,472.1,417.1,318.2,228.6,142.3,83.9,53.5)

females<-c (860.0,966.5,1149.5,1115.5,1010.5,929.8,737.3,627.0,622.5,621.5,600.1,495.3,437.4,354.6,273.1,201.0,128.5,99.4)

total\_males <- sum(males)

total\_females <- sum(females)

total\_sample\_size <- total\_males+ total\_females

#sample proportion

s\_p <- total\_males/total\_sample\_size

s\_p

# sample error

se <- sqrt((s\_p \* (1-s\_p))/total\_sample\_size)

# applying wald test

p\_value =2\* pnorm( -abs(s\_p-0.5)/se)

p\_value

**RESULT : it is 0.93, so H0 is not rejected at 5% level of significance ; Also we can use pearson chi square test as well.**

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

# part b

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

# using bonferroni method

# At 5% Level of significance for overall test

alpha\_individual <- (.05/length(males))

# counting no of rejections of individual hypothesis

count <- 0

# initializing p-values

p\_value <- rep(1,length(males))

for (i in 1: length(males)) {

total\_males <- males[i]

total\_females <- females[i]

total\_sample\_size <- total\_males+ total\_females

#sample proportion

s\_p <- total\_males/total\_sample\_size

s\_p

# sample error

se <- sqrt((s\_p \* (1-s\_p))/total\_sample\_size)

# applying wald test

p\_value[i] =2\* pnorm( -abs(s\_p-0.5)/se)

if (p\_value[i] < alpha\_individual) {

cat(males[i],"\n")

count<- count+1 }}

count

**RESULT : Reject H0 if count >0; In this case count =3 , so H0 is rejected.**

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

# part c

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

# using B-H method

# Getting the ordered p-values from part(b)

ordered\_pvalues <- sort(p\_value, decreasing=FALSE)

# assuming all the age groups are independent and so are the corresponding p\_values

# FDR (alpha value) =5% or 0.05

p\_i <- seq(1:length(males))\*(0.05/length(males))

for (i in 1: length(males)) {

if (ordered\_pvalues[i] > p\_i[i]) {

cat ("i value:",i,"\n")

break}}

# cut-off p\_value

cutoff\_p\_value <- ordered\_pvalues[i-1]

cat ("cut-off p\_value :",cutoff\_p\_value,"\n")

# printing age ranges whose hypotheses are rejected.

for (i in 1 : length(males) ) {

if (p\_value[i] <= cutoff\_p\_value) {

cat ("Group no :",i,"rejected\n")}}

**RESULT : Group no : 16 rejected; Group no : 17 rejected; Group no : 18 rejected**

# midterm 2008, Q4

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

# part a

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

# using Jarque-bera test , JB which follows chi-square with 2 deg of freedom, so 1, part (a) cutoof of 4.605 is valid

sec\_x<- c(75,94,50,68,75,72,59,84,84,78,57,71,80,54,59,73,71,54,64,69,76,49,76,75,98,74,47,85,53,87,72,82,67,58,94,68,83,48,63,51,55,87,54,95,47,85,83,66,91,81,73,93,91,84,66,62,60,58,45,76,86,72)

sec\_y<- c(49,76,99,62,87,54,78,69,52,65,66,60,59,57,75,75,42,61,93,77,77,74,76,96,86,71,68,71,81,69,42,74,71,82,74,73,66,84,100,86,56,89,68,73,67,69,81,52,83,80,84,45,42,68,85,92,56,50,90,57,88)

# testing normality using shapiro.test

shapiro.test(sec\_x)

shapiro.test(sec\_y)

**RESULT : Since both p\_values are greater than .01, We can say that both the data are normal.**

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

# part b

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

# H0: ux-uy =0 vs H1 : ux-uy =/ 0

# test\_statistic : Xbar - Ybar

# se = sqrt( sx^2/nx + sy^2/ny)

# applying wald test

ux<- mean(sec\_x)

uy<- mean(sec\_y)

sx2<- var(sec\_x)

sy2<- var(sec\_y)

se<- sqrt((sx2^2/length(sec\_x))+(sy2^2/length(sec\_y)))

# walds test

p\_value <- 2\* pnorm(-abs(ux-uy)/se)

p\_value

**RESULT : since p\_value is 0.99, we cant reject H0, and say that means of two sections are close**

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

# part c

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

# For Inter quartile range we need phi(0.75)-phi(0.25) = u+0.675s-(u-0.675s) = 1.35s, since s is MLE for sigma.

iqr\_x <- 1.35\* sqrt(var(sec\_x))

iqr\_x

iqr\_y <- 1.35\* sqrt(var(sec\_y))

iqr\_y

# directly based on given samples

iqr2\_x <- quantile(sec\_x,0.75) - quantile(sec\_x,0.25)

iqr2\_x

iqr2\_y <- quantile(sec\_y,0.75) - quantile(sec\_y,0.25)

iqr2\_x

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

# part d

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

# no of simulations

nsim <-1000

# let d^ = iqrx^ -iqry^

# Now Ho: d^= 0 vs Ha : d^ =/0

# se^ will be obtained through bootstrap.

# apply wald test statistic d^/se^ to find p\_value and decide based on given LOS =5%

iqr\_x <- 1.35\* sqrt(var(sec\_x))

iqr\_x

iqr\_y <- 1.35\* sqrt(var(sec\_y))

iqr\_y

# to calculate se^ using bootstrap

btstrp\_iqr\_d <- seq(1:nsim)

for (i in 1: nsim) {

btstrp\_sec\_x <- sample(sec\_x,length(sec\_x), replace=TRUE)

btstrp\_iqr\_x <- 1.35\* sqrt(var(btstrp\_sec\_x))

btstrp\_sec\_y <- sample(sec\_y,length(sec\_y), replace=TRUE)

btstrp\_iqr\_y <- 1.35\* sqrt(var(btstrp\_sec\_y))

btstrp\_iqr\_d[i] <- btstrp\_iqr\_x - btstrp\_iqr\_y }

# finding stdev of bootstrapped differences in iqr

se\_d <- sd (btstrp\_iqr\_d)

# applying wald test

p\_value <- 2\* pnorm(-abs(iqr2\_x-iqr2\_y)/se\_d)

p\_value

**RESULT : since p\_value is ~ 0.13, we dont reject Ho and so Inter quartile difference between the two section is not significant**

**In above case I am testing actual sample value against MLE based se^. Instead of MLE based se^ estimation, we can use PERMUTATION TESTS as well for parts d, e,f**

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

# part e

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

# no of simulations

nsim <-1000

# let d^ = px^ -py^ (px, py indicate proporation of students scoring more than 80)

# d^ = phi((80-uy)/sigma\_y)- phi((80-ux)/sigma\_x)

# Now Ho: d^= 0 vs Ha : d^ =/0

# se^ will be obtained through bootstrap.

# apply wald test statistic d^/se^ to find p\_value and decide based on given LOS =5%

# calculating sample proporations > 80

count <-0

for (i in 1 : length(sec\_x)) {

if ( sec\_x[i] > 80 ) { count <- count+1 }}

sec\_x\_p <- count/length(sec\_x)

count <-0

for (i in 1 : length(sec\_y)) {

if ( sec\_y[i] > 80 ) { count <- count+1 }}

sec\_y\_p <- count/length(sec\_y)

# For given section samples, difference in proportion is

s\_p <- sec\_y\_p - sec\_x\_p

# to calculate se^ using bootstrap

btstrp\_p\_d <- seq(1:nsim)

for (i in 1: nsim) {

btstrp\_sec\_x <- sample(sec\_x,length(sec\_x), replace=TRUE)

btstrp\_p\_x <- pnorm((80-mean(btstrp\_sec\_x))/sd(btstrp\_sec\_x))

btstrp\_sec\_y <- sample(sec\_y,length(sec\_y), replace=TRUE)

btstrp\_p\_y <- pnorm((80-mean(btstrp\_sec\_y))/sd(btstrp\_sec\_y))

btstrp\_p\_d[i] <- btstrp\_p\_y - btstrp\_iqr\_x }

# finding stdev of bootstrapped differences in iqr

se\_d <- sd (btstrp\_p\_d)

# applying wald test

p\_value <- 2\* pnorm(-abs(s\_p)/se\_d)

p\_value

**RESULT : since p\_value is ~ 0.79 we dont reject Ho and so Inter quartile difference between the two section is not significant**

# In above case I am testing actual sample value against MLE based se^, we can also use actual proportions of samples in bootstrap to caluclate se^ and use that std error for wald's test.

########################## part f#########################

# no of simulations

# Ho1 : p1x-p1y =0 vs Ha1 : p1x-p1y /= 0 and so on.

# px-py is estimated using px^-py^ and it has se^ of sqrt( px^(1-px^)/nx + py^(1-py^)/ny)

# we will use borrenheins multi testing.

# so alpha ^ = 0.05/3 = 0.0167

px<- rep(0,3)

for (j in 1 : length(sec\_x)) {

if (sec\_x[j] < 50 ) {

px[1] <- px[1]+1

} else if (sec\_x[j] <= 80 ) {

px[2] <-px[2]+1}

else { px[3] <- px[3] +1 }}

px[1] <- px[1]/length(sec\_x)

px[2] <- px[2]/length(sec\_x)

px[3] <- px[3]/length(sec\_x)

px

py<-rep(0,3)

for (j in 1 : length(sec\_x)) {

if (sec\_x[j] < 50 ) {

py[1] <- py[1]+1

} else if (sec\_x[j] <= 80 ) {

py[2] <-py[2]+1}

else { py[3] <- py[3] +1 }}

py[1] <- py[1]/length(sec\_y)

py[2] <- py[2]/length(sec\_y)

py[3] <- py[3]/length(sec\_y)

py

for (i in 1:3) {

p\_value = 2\* pnorm ( -abs(px[i]-py[i])/(sqrt(((px[i]\*(1-px[i]))/length(sec\_x))+((py[i]\*(1-py[i]))/length(sec\_y)))))

if (p\_value < (0.05/3)) { cat ( p\_value,i," the H0 rejected\n") }}

**RESULT : since p\_values are 0.9 , none of the individual hypothesis is rejection. Since borrenhoni passed B-H which is a less restrictive test, will also pass.**

# midterm 2008, Q5

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

# part a

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

nsim <- 1000

ca<-c(0.3,0.3,0.32,0.43,0.44,0.47,0.52,0.59,0.7,0.77,0.79,0.81,0.95,1.33,1.43,1.54)

cb<-c(0.04,0.14,0.19,0.21,0.25,0.26,0.32,0.35,0.4,0.4,0.42,0.53,0.53,0.56,0.57,0.68)

cab <- c(ca,cb)

# sample absoulute value of mean

s\_mean <- abs( mean(ca) -mean(cb))

# permuting

# initializing count

count <-0

#perm\_mean <- rep(0,nsim)

for (i in 1:nsim) {

perm <- sample(cab,replace=FALSE)

perm\_mean <- abs(mean(perm[1:length(ca)])-mean(perm[(length(ca)+1):length(perm)]))

if ( perm\_mean > s\_mean) { count <- count +1 }}

p\_value <- count/nsim

cat ("Difference in mean for given samples:",s\_mean, "p\_value:", p\_value,"\n")

**RESULT : sor for p\_value of 0.003, we can reject Ho and so the holding capabilities of both the cannulae are different.**

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

# part b

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

nsim <- 1000

s\_quantile\_diff <- abs(quantile(ca,0.25) -quantile(cb,0.25))

# permuting

# initializing count

count <-0

perm\_quant\_diff <- rep(0,nsim)

for (i in 1:nsim) {

perm <- sample(cab,replace=FALSE)

perm\_quant\_diff[i] <- abs(quantile(perm[1:length(ca)],0.25)-quantile(perm[(length(ca)+1):length(perm)],0.25))

if ( perm\_quant\_diff[i] > s\_quantile\_diff ) { count <- count +1 }}

p\_value <- count/nsim

cat ("Difference in quantile diff for given samples:",s\_quantile\_diff, "p\_value:", p\_value,"\n")

**RESULT: extracting confidence interval from the histogram of permuted quantile diff values.**

q\_diff\_lower\_bound <- quantile (perm\_quant\_diff,.025)

q\_diff\_upper\_bound <- quantile (perm\_quant\_diff,.975)

cat ("Lower Bound:",q\_diff\_lower\_bound,"Upper Bound:",q\_diff\_upper\_bound,"\n")

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

# part c

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

nsim <- 1000

# mad ratio for the sample

mad\_ratio <- mad(ca)/mad(cb)

perm\_mad\_ratio <- rep(0, nsim)

# bootstrapping

for (i in 1:nsim) {

perm <- sample(cab,replace=TRUE)

perm\_mad\_ratio [i] <- mad(perm[1:length(ca)])/mad(perm[(length(ca)+1):length(perm)])}

# extracting confidence interval from the histogram of permuted quantile diff values.

q\_diff\_lower\_bound <- quantile (perm\_mad\_ratio,.025)

q\_diff\_upper\_bound <- quantile (perm\_mad\_ratio,.975)

cat ("Lower Bound:",q\_diff\_lower\_bound,"Upper Bound:",q\_diff\_upper\_bound,"\n")

**RESULT : bounds are : Lower Bound :0.3912778 Upper Bound: 2.625164**

# midterm 2008, Q6

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

# part a ############################

nsim <- 1000

rf<-c(146.8,383,90.9,178.1,267.5,95.5,156.5,180,90.9,139.7,200.2,171.7,187.2,184.9,70.1,58,84.1,55.6,133.1,271.8,135.9,71.9,99.4,110.6,47.5,97.8,122.7,58.4,154.4,173.7,118.8,88,84.6,171.5,254.3,185.9,137.2,138.9,96.2,85,45.2,74.7,264.9,113.8,133.4,68.1,156.4)

# calculating hubers estimator by taking K=1.28

# cutoff to determine outlier |x-median|> ((mad/.6745)\*1.28)

cutoff <- ((mad(rf)/.6745)\*1.28)

# counts to determine if outliers are less than or greater than median

l<-0

u<-0

rf\_median <- median(rf)

sample\_sum <- 0

for (i in 1: length(rf)) {

if (abs(rf[i] - rf\_median) > cutoff) {

if (rf[i]> rf\_median) { u <- u+1 } else { l <- l+1}

}else {

sample\_sum <- sample\_sum + rf[i] }}

rf\_huber <- ((cutoff\* (u-l))+sample\_sum)/(length(rf)-l-u)

**RESULT : HUBER ESTIMATOR IS [1] 133.8148**

# Alternatively load MASS package and use huber function

library(MASS)

huber(rf, k=1.28)

# bootstrap based estimate

bt\_huber <- rep(0,nsim)

for ( i in 1: nsim) {

s\_bt <- sample (rf, replace=TRUE)

cutoff <- ((mad(s\_bt)/.6745)\*1.28)

# counts to determine if outliers are less than or greater than median

l<-0

u<-0

rf\_median <- median(s\_bt)

sample\_sum <- 0

for (i in 1: length(s\_bt)) {

if (abs(s\_bt[i] - rf\_median) > cutoff) {

if (s\_bt[i]> rf\_median) { u <- u+1 } else { l <- l+1}

}else {

sample\_sum <- sample\_sum + s\_bt[i] }}

bt\_huber[i] <- ((cutoff\* (u-l))+sample\_sum)/(length(s\_bt)-l-u)}

huber\_lb <- quantile(bt\_huber,0.025)

huber\_ub <- quantile(bt\_huber,0.975)

cat ("huber est LowerB:",huber\_lb,"huber est UpperB:",huber\_ub,"\n")

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

# part c

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

# prob of rainfall < 50mm p^

count <- 0

for (i in 1 : length(rf)) {

if (rf[i] <50) {

count <- count +1 }}

prob50 <- count/length(rf)

prob50

bt\_prob <- rep (0, nsim)

for ( i in 1 : nsim) {

s\_bt <- sample ( rf, replace=TRUE)

count <- 0

for (i in 1 : length(s\_bt)) {

if (s\_bt[i] <50) {

count <- count +1 }}

bt\_prob[i] <- count/length(s\_bt)}

cat ("prob50 lower bound",quantile(bt\_prob50,.025),"prob 50 upper bound", quantile(bt\_prob50,.975),"\n")

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

# part d

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

# prob of rainfall < 50mm p^

count <- 0

for (i in 1 : length(rf)) {

if (rf[i] >250) {

count <- count +1 }}

prob250 <- count/length(rf)

prob250

bt\_prob250 <- rep (0, nsim)

for ( i in 1 : nsim) {

s\_bt <- sample ( rf, replace=TRUE)

count <- 0

for (i in 1 : length(s\_bt)) {

if (s\_bt[i] >250) {

count <- count +1 }}

bt\_prob250[i] <- count/length(s\_bt)}

cat ("prob250 lower bound",quantile(bt\_prob250,.025),"prob 250 upper bound", quantile(bt\_prob250,.975),"\n)