##### Examples with the Quantile Test:

# While R does not have a built-in function to perform the quantile test,

quantile.test<-function(x,xstar=0,quantile=.5,alternative="two.sided"){

 n<-length(x)

 p<-quantile

 T1<-sum(x<=xstar)

 T2<-sum(x< xstar)

 if (alternative=="quantile.less") {

 p.value<-1-pbinom(T2-1,n,p)}

 if (alternative=="quantile.greater"){

 p.value<-pbinom(T1,n,p)}

 if (alternative=="two.sided"){

 p.value<-2\*min(1-pbinom(T2-1,n,p),pbinom(T1,n,p))}

 list(xstar=xstar,alternative=alternative,T1=T1,T2=T2,p.value=p.value)}

# Copy and paste this function into R, and then it can be implemented

# as in the following examples:

## Entrance exam example (p. 139-140 of text):

testscores <- c(189,233,195,160,212,176,231,185,199,213,202,193,174,166,248)

quantile.test(testscores,xstar=193,quantile=0.75,alternative="two.sided")

## House price example:

prices <- c(120, 500, 64, 104, 172, 275, 336, 55, 535, 251, 214, 1250, 402, 27, 109, 17, 334, 205)

quantile.test(prices,xstar=179, quantile=0.5, alternative="quantile.less")

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

# While R does not have a built-in function to perform the quantile CI,

quantile.interval<-function(x,quantile=.5,conf.level=.95){

 n<-length(x)

 p<-quantile

 alpha<-1-conf.level

 rmin1<-qbinom(alpha/2,n,p)-1

 r<-rmin1+1

 alpha1<-pbinom(r-1,n,p)

 smin1<-qbinom(1-alpha/2,n,p)

 s<-smin1+1

 alpha2<-1-pbinom(s-1,n,p)

 clo<-sort(x)[r]

 chi<-sort(x)[s]

 conf.level<-1-alpha1-alpha2

 list(quantile=quantile,conf.level=conf.level,r=r,s=s,interval=c(clo,chi))}

# Copy and paste this function into R, and then it can be implemented

# as in the following examples:

## House price example:

prices <- c(120, 500, 64, 104, 172, 275, 336, 55, 535, 251, 214, 1250, 402, 27, 109, 17, 334, 205)

# 95% CI for median:

quantile.interval(prices, quantile=0.5, conf.level=0.95)

# 95% CI for 0.80 quantile:

quantile.interval(prices, quantile=0.80, conf.level=0.95)

## This function is conservative in that it ensures

## alpha1 is less than or equal to alpha/2 and

## that (1-alpha2) is less than or equal to (1-alpha/2).

## It may be possible to get a shorter exact interval

## that still has confidence level of at least (1-alpha).

###

### Some power calculations:

###

# Normal data:

t.power.norm = function(nsamp=10,nsim=1000,means=0,sds=1){

 lower = qt(.025,df=nsamp - 1)

 upper = qt(.975,df=nsamp - 1)

 ts = replicate(nsim,

 t.test(rnorm(nsamp,mean=means,sd=sds))$statistic)

 myps = replicate(nsim,

 quantile.test(rnorm(nsamp,mean=means,sd=sds))$p.value)

 cbind(sum(ts < lower | ts > upper) / nsim, sum(myps < .05)/nsim )

}

t.power.norm(means=0)

t.power.norm(means=0.5)

t.power.norm(means=0.5,nsamp=20)

t.power.norm(means=0.5,nsamp=100)

# t(df=1), i.e., Cauchy data (HEAVY tails):

t.power.t = function(nsamp=10,nsim=1000,means=0,sds=1){

 lower = qt(.025,df=nsamp - 1)

 upper = qt(.975,df=nsamp - 1)

 ts = replicate(nsim,

 t.test(rt(nsamp,ncp=means,df=1))$statistic)

 myps = replicate(nsim,

 quantile.test(rt(nsamp,ncp=means,df=1))$p.value)

 cbind(sum(ts < lower | ts > upper) / nsim, sum(myps < .05)/nsim )}

t.power.t(means=0)

t.power.t(means=0.5)

t.power.t(means=0.5,nsamp=20)

t.power.t(means=0.5,nsamp=50)