Namburu_Setu

Test Items

Read each question carefully and address each element. Do not print contents of vectors or data frames unless requested.

(1) (4 points) This problem deals with vector manipulations.

(1)(a) Create a vector that contains the following, in this order, and print the contents. Do not round off any values unless requested. * A sequence of integers from 0 to 6, inclusive. * Two repetitions of the vector c(2, -5.1, -23). * The sum of 7/42 and 3

```
a <- c(seq(0,6),rep(c(2,-5.1,-23),2),((7/42)+3))
print(a)
```

```
##  [1]   0.000000   1.000000   2.000000   3.000000   4.000000   5.000000
##  [7]   6.000000   2.000000  -5.100000 -23.000000   2.000000  -5.100000
## [13] -23.000000   3.166667
```

(1)(b) Determine the length of the vector created in (1)(a) and denote as L. Print L. Generate a sequence starting with 1 and ending with L and add to the vector from (1)(a). This is vector addition, not vector combination. Print the contents. Do not round off any values.

```
L <- length(a)
cat("L = ", L,"\n")
```

```
## L =  14
```

```
b<-seq(1, L)
df <- a+b
print(df)
```

```
##  [1]   1.00000   3.00000   5.00000   7.00000   9.00000  11.00000  13.00000
##  [8]  10.00000   3.90000 -13.00000  13.00000   6.90000 -10.00000  17.16667
```

(1)(c) Extract the first and last elements of the vector you have created in (1)(b) to form another vector using the extracted elements. Form a third vector from the elements not extracted. Print these vectors.

```
df1<-df[c(1,14)]
print(df1)
```

```
## [1]  1.00000 17.16667
```

```
df2<-df[c(2:13)]
print(df2)
```

```
##  [1]   3.0   5.0   7.0   9.0  11.0  13.0  10.0   3.9 -13.0  13.0   6.9
## [12] -10.0
```

(1)(d) Use the vectors from (c) to reconstruct the vector in (b). Print this vector. Sum the elements and round to two decimal places.

```
df3<-append(df1,df2,after=1)
df3
```

```
##  [1]   1.00000   3.00000   5.00000   7.00000   9.00000  11.00000  13.00000
##  [8]  10.00000   3.90000 -13.00000  13.00000   6.90000 -10.00000  17.16667
```

```
round(sum(df3),2)
```

```
## [1] 76.97
```

4 points

(2) (5 points) The expression y = sin(x) - cos(x) is a trigonometric function.

(2)(a) Using the trigonometric function above, write a function as defined by R in the proper format that accepts values for x and returns a value for y.

```
trig.fn<-function(x){
  y<-(sin(x) - cos(x))
  return(y)
}
```

(2)(b) Create a vector, x, of 4001 equally-spaced values from -2 to 2, inclusive. Compute values for y using the vector x and your function in (a). **Do not print x or y.** Find the value in the vector x that corresponds to the minimum value in the vector y. In other words, restrict attention to only the values of x and y you have computed. Round to 3 decimal places and print the value of x you find and the corresponding minimum value for y.

Finding the two desired values can be accomplished in as few as two lines of code. Do not use packages or programs you may find on the internet or elsewhere. Do not print the elements of the vectors x and y. Use coding methods shown in the *Quick Start Guide for R*.

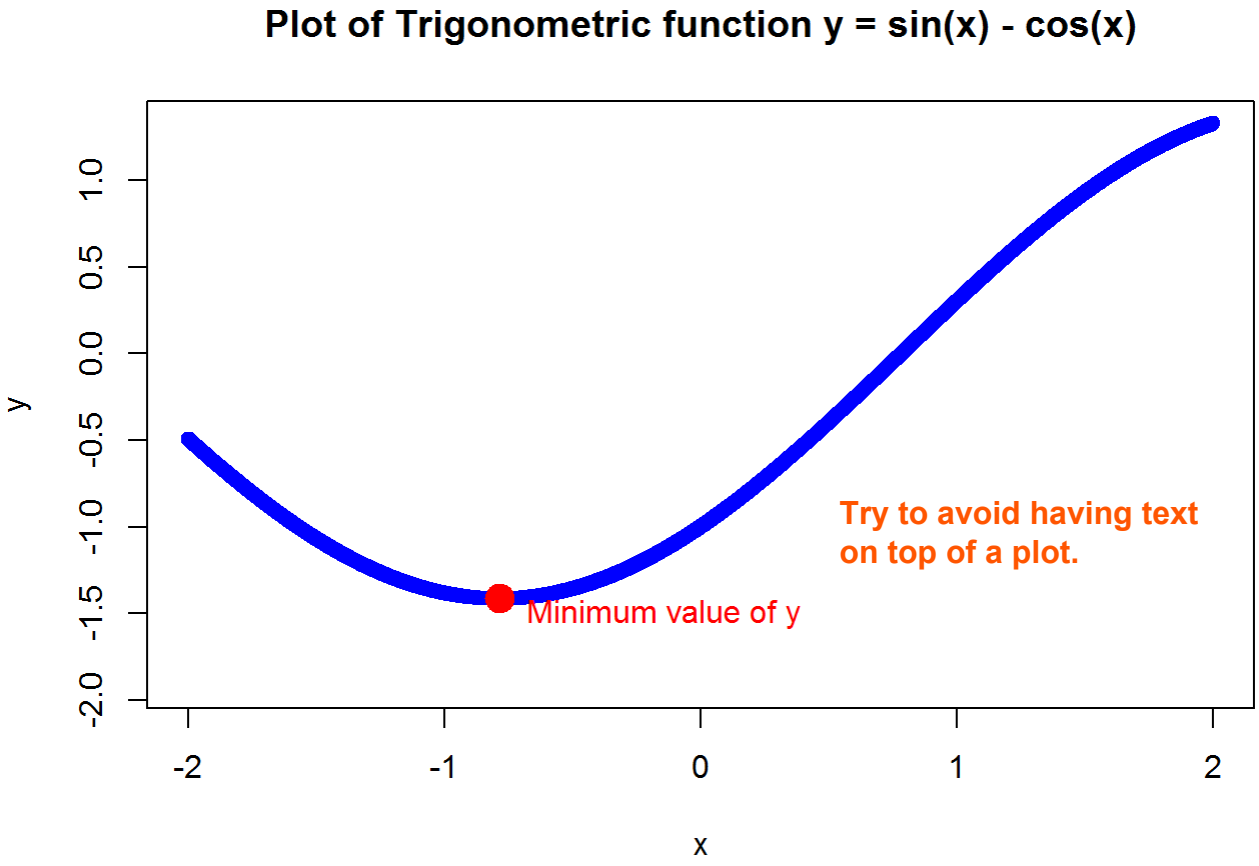
```
x<-seq(-2,2,length=4001)
y<-trig.fn(x)

cat("min(y) = ",round(min(y),3), "and the corresponding value of x = ", round(x[which.min(y)],
3))
```

```
## min(y) =  -1.414 and the corresponding value of x =  -0.785
```

(2)(c) Plot y versus x in color, with x on the horizontal axis. Show the location of the minimum value of y determined in 2(b). Add a title and other features such as text annotations. Text annotations may be added via *text()*.

```
plot(x,y,col="blue", main="Plot of Trigonometric function y = sin(x) - cos(x)",ylim=c(min(y)-0
```

```
.5,max(y)))
points(x[which.min(y)],min(y),cex=2,pch=19,col="red")
text(x[which.min(y)],min(y),"Minimum value of y", adj=c(-0.1,1),col="red")
```



5 points

(3) (8 points) Use the “trees” dataset for the following items. This dataset has three variables (Girth, Height, Volume) on 31 trees.

(3)(a) Use *data(trees)* to load the file. Check the structure with *str()*. Use *apply()* to return the median values for the three variables in “trees.” Using R and logicals, print the row number and the three measurements: Girth, Height and Volume, of the tree with Volume equal to median Volume.

```
##Loading dat
data("trees")

##structure of the data
str(trees)
```

```
## 'data.frame':    31 obs. of  3 variables:
##  $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

```
##Median of the three variables
apply(trees,2,median)
```

```
##  Girth Height Volume
##   12.9   76.0   24.2
```

```
##row number where volume equal to median volume
ind<-which(trees$Volume==median(trees$Volume))
ind
```

```
## [1] 11
```

```
##row number and the three measurements where volume is equal to median volume
trees[ind,]
```

```
##    Girth Height Volume
## 11  11.3     79   24.2
```

(3)(b) Girth is defined as the diameter of a tree taken at 4 feet 6 inches from the ground. Convert each diameter to a radius, r. Calculate the cross-sectional area of each tree using pi times the squared radius. Sort r ascending and print. Present the stem-and-leaf plot of the radii, and a histogram of the radii in color.

```
r<-trees$Girth/2 ###diameter or Girth=2*radius

area<-pi*(r**2) ## cross-sectional area

##Sort radii and print
sort(r)
```

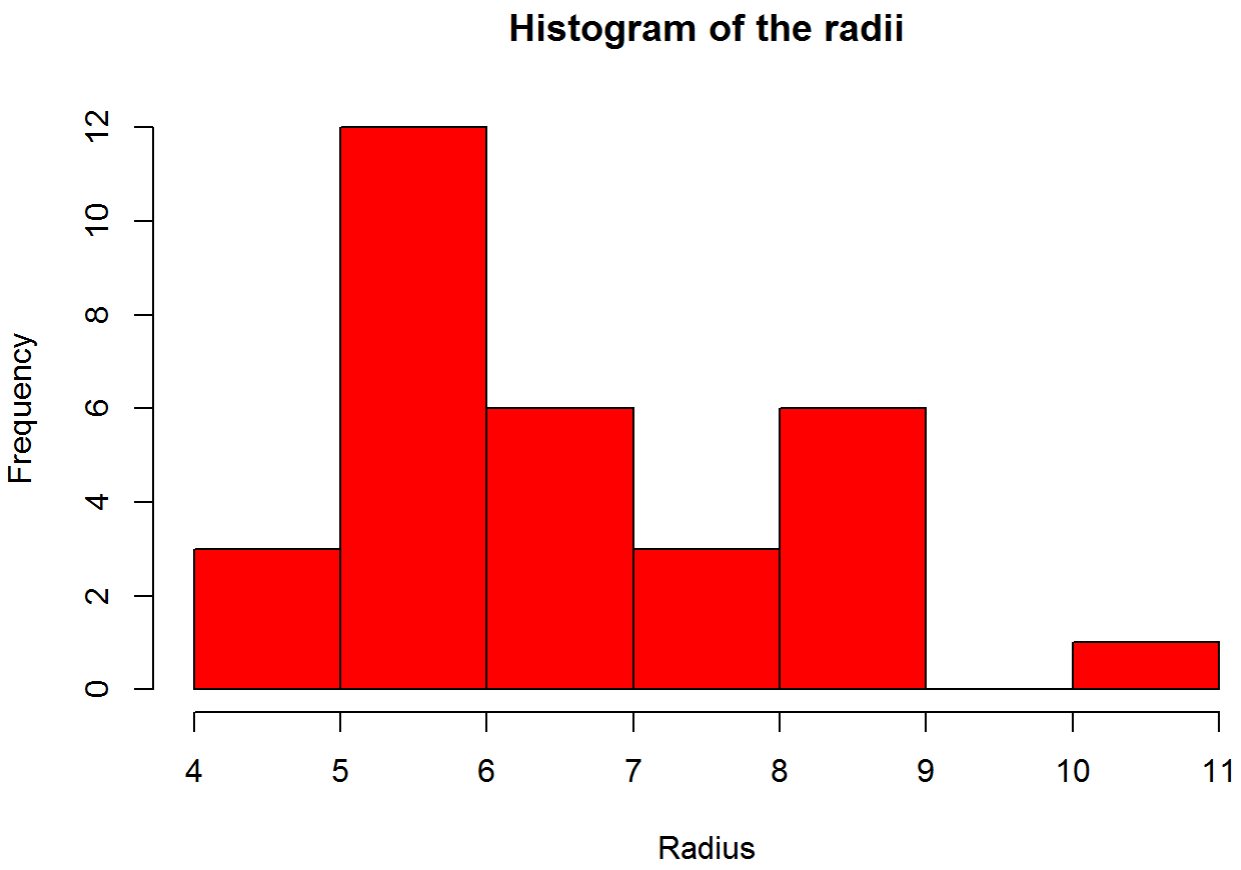
```
##  [1]  4.15  4.30  4.40  5.25  5.35  5.40  5.50  5.50  5.55  5.60  5.65
## [12]  5.70  5.70  5.85  6.00  6.45  6.45  6.65  6.85  6.90  7.00  7.10
## [23]  7.25  8.00  8.15  8.65  8.75  8.95  9.00  9.00 10.30
```

```
##stem and leaf plot of the radii
stem(r)
```

```
##
##   The decimal point is at the |
##
##    4 | 234
##    5 | 34455667779
##    6 | 055799
##    7 | 013
##    8 | 0278
##    9 | 000
##   10 | 3
```

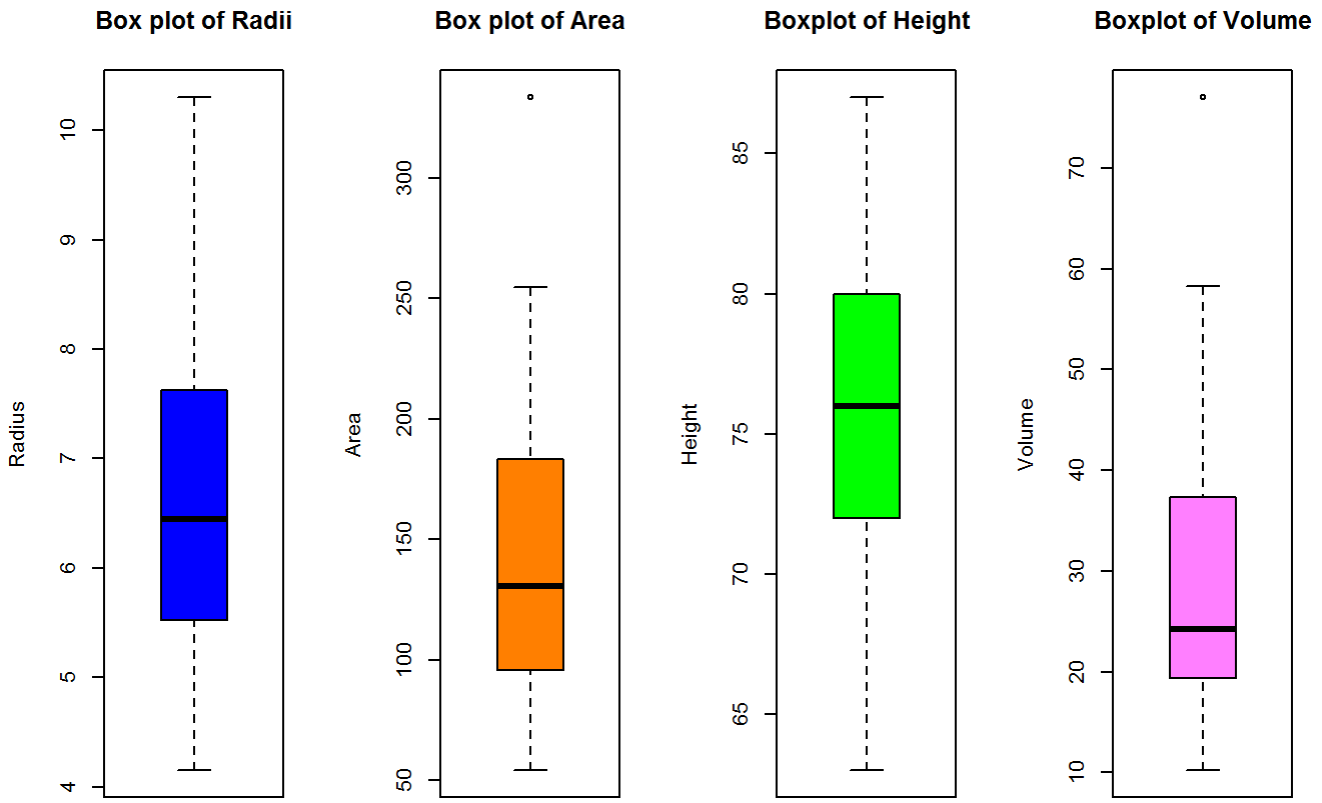
```
##Histogram of radii in color
```

```
hist(r,col=rgb(1,0,0,1), main="Histogram of the radii", xlab = "Radius")
```



(3)(c) Use *par(mfrow = c(1, 4))* and present colored boxplots of the radii and areas calculated in (b) along with Height and Volume. Label each accordingly.

```
par(mfrow=c(1,4))

boxplot(r,col="blue",main="Box plot of Radii",ylab="Radius")

new_orange = rgb(255, 127, 0, maxColorValue = 255)
boxplot(area,col=new_orange,main="Box plot of Area",ylab="Area")

boxplot(trees$Height,col="green",main="Boxplot of Height",ylab="Height")

boxplot(trees$Volume,col=rgb(1,0,1,0.5),main="Boxplot of Volume",ylab="Volume")
```




```
par(mfrow=c(1,1))
```

(3)(d) Demonstrate that the outlier revealed in the boxplot of Volume is not an extreme outlier. It is possible to do this with one line of code using *boxplot.stats* or logicals.

```
##boxplot.stats with coef=1.5 shows outliers, coef=3 shows extreme outliers, so using coef=3 h
ere
boxplot.stats(trees$Volume,coef=3)$out
```

```
## numeric(0)
```

8 points

(4) (2 points) Use matrix operations shown in the “Quick Start Guide” to solve the following system of linear equations. Display the R script and the numerical solutions for x, y and z. Use matrix operations with your solution to reproduce the values 1, 1, 3 as a means of checking if your solution is correct. This last demonstration can be accomplished with matrix multiplication in one line of code. Print your result.

x - y + z = 1

x + y + z = 1

x + y - z = 3

```
##System of equations to matrices
A<-matrix(c(1,-1,1,1,1,1,1,1,-1),nr=3,byrow=T)  ##lhs
```

```
B<-c(1,1,3) ###rhs

###Solve for x,y,z
solve(A,B)
```

```
## [1]  2  0 -1
```

```
###Verify if the solution is correct

A%*%solve(A,B)
```

```
##      [,1]
## [1,]    1
## [2,]    1
## [3,]    3
```

2 points

(5) (6 points) The Cauchy distribution is an example of a “heavy-tailed” distribution in that it will have (more) outliers in both tails. This problem involves comparing it with a normal distribution which typically has very few outliers.

5(a) Use *set.seed(124)* and *rcauchy()* with n = 100, location = 0 and scale = 0.1 to generate a random sample designated as y. Generate a second random sample designated as x with *set.seed(127)* and *rnorm()* using n = 100, mean = 0 and sd = 0.15.

Generate a new object using *cbind(x, y)*. Do not print this object. Use *apply()* with this object to compute the inter-quartile range or IQR for each column: x and y. Round the results to four decimal places and present. (The point of this exercise is to demonstrate the similarity of the IQR values.)

For information about rcauchy(), use help in R (?rcauchy). **Do not print x and y.**

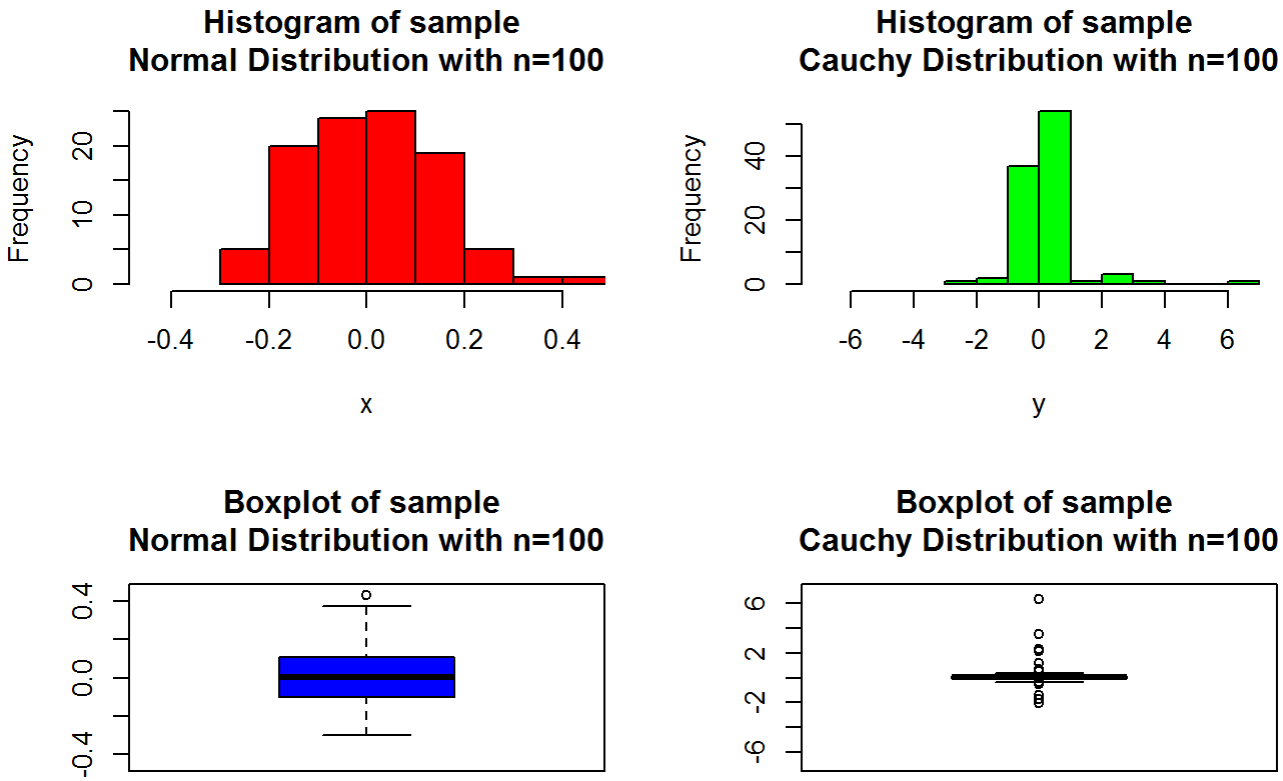
```
##Cauchy distribution sample
set.seed(124)
y<-rcauchy(100,location = 0,scale = 0.1)

#Normal distribution sample
set.seed(127)
x<-rnorm(100,mean=0,sd=0.15)

z<-cbind(x,y)

round(apply(z, 2, FUN=IQR), 4)
```

```
##      x      y
## 0.2041 0.2141
```

(5)(b) This item will illustrate the difference between a heavy-tailed distribution and one which does not have heavy tails. Use *par(mfrow = c(2, 2))* to generate a display with four diagrams. On the first row, present two histograms in color, one for x and the second for y. Set *xlim = c(-7, 7)* for the Cauchy results, and *xlim = c(-0.45, 0.45)* for the Normal results. Do not specify ylim for the histograms. On the second, show vertical boxplots for x and y. Use the xlim interval values

respectively for the ylim of the boxplots.

```
par(mfrow = c(2,2))

hist(x, col = rgb(1,0,0,1), xlim = c(-0.45, 0.45),main = "Histogram of sample \nNormal Distrib
ution with n=100")
hist(y, col = rgb(0,1,0,1), xlim = c(-7, 7), main = "Histogram of sample \nCauchy Distribution
 with n=100")

boxplot(x, col = "blue", ylim = c(-0.45, 0.45), main = "Boxplot of sample \nNormal Distributio
n with n=100")
boxplot(y, col = "bisque", ylim = c(-7, 7), main = "Boxplot of sample \nCauchy Distribution wi
th n=100")
```

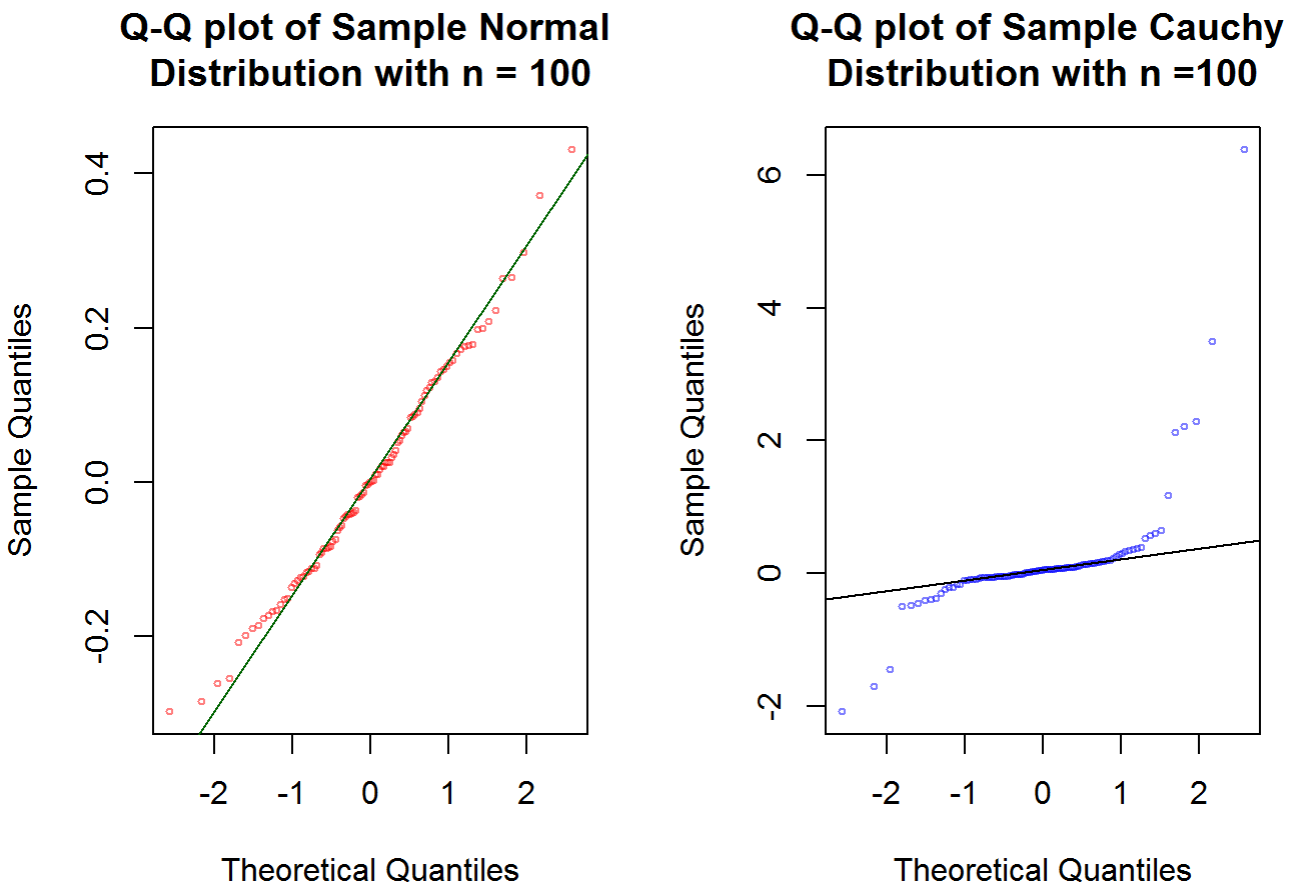



```
par(mfrow = c(1,1))
```

(5)(c) QQ plots are useful for detecting the presence of heavy-tailed distributions. Use *par(mfrow = c(1, 2))* and present side-by-side plots, one for each sample, using *qqnorm()* and *qqline()*. Add color and titles. Use cex = 0.5 to control the size of the plotted data points.

```
par(mfrow = c(1,2))

qqnorm(x, main = "Q-Q plot of Sample Normal \nDistribution with n = 100", col = rgb(1,0,0,0.5)
, cex = 0.5)
qqline(x, col = "darkgreen")
```

```
qqnorm(y, main = "Q-Q plot of Sample Cauchy \nDistribution with n =100", col = rgb(0,0,1,0.5),
 cex = 0.5)
qqline(y, col = "black")
```




```
par(mfrow = c(1,1))
```

6 points

25 points