# Math 185 - HW2

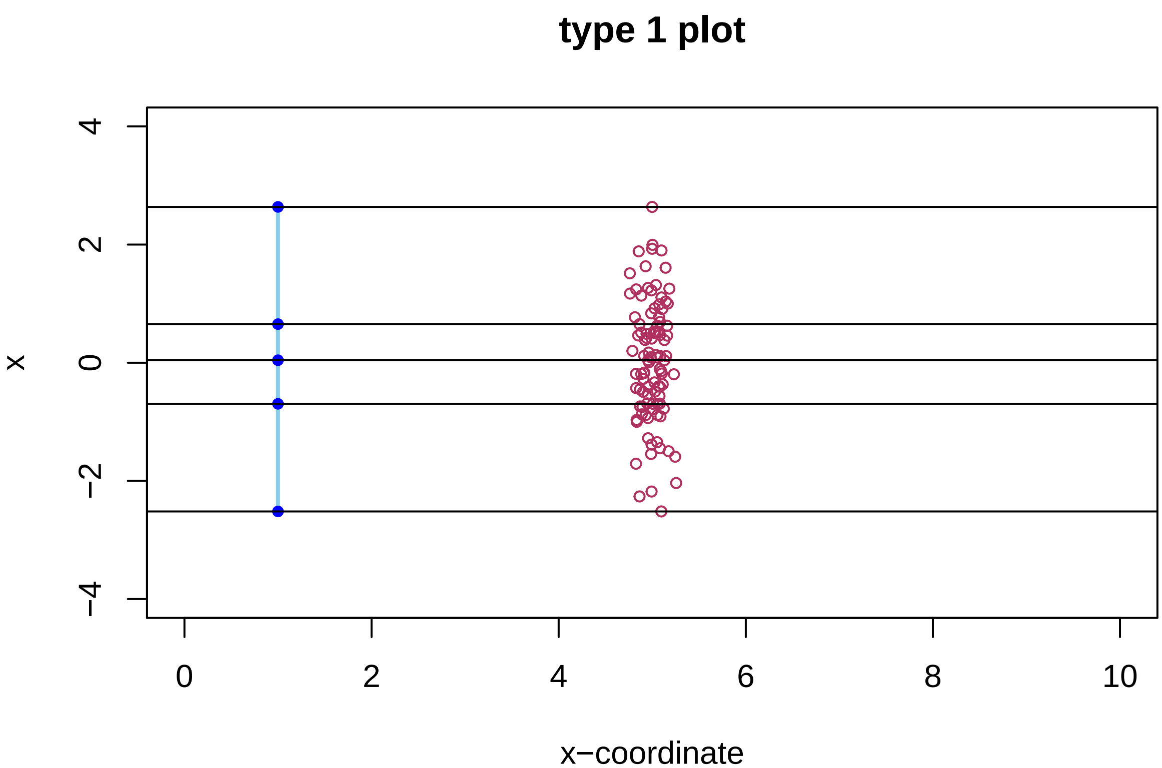
*Dukki Hong / A98058412*

**Problem 1. Produce examples and plots highlighting the differences between the different ways that R offers to compute sample quantiles. There are 9 ways in total that the function quantile offers. Cover as many of them as you can (at least 5).**

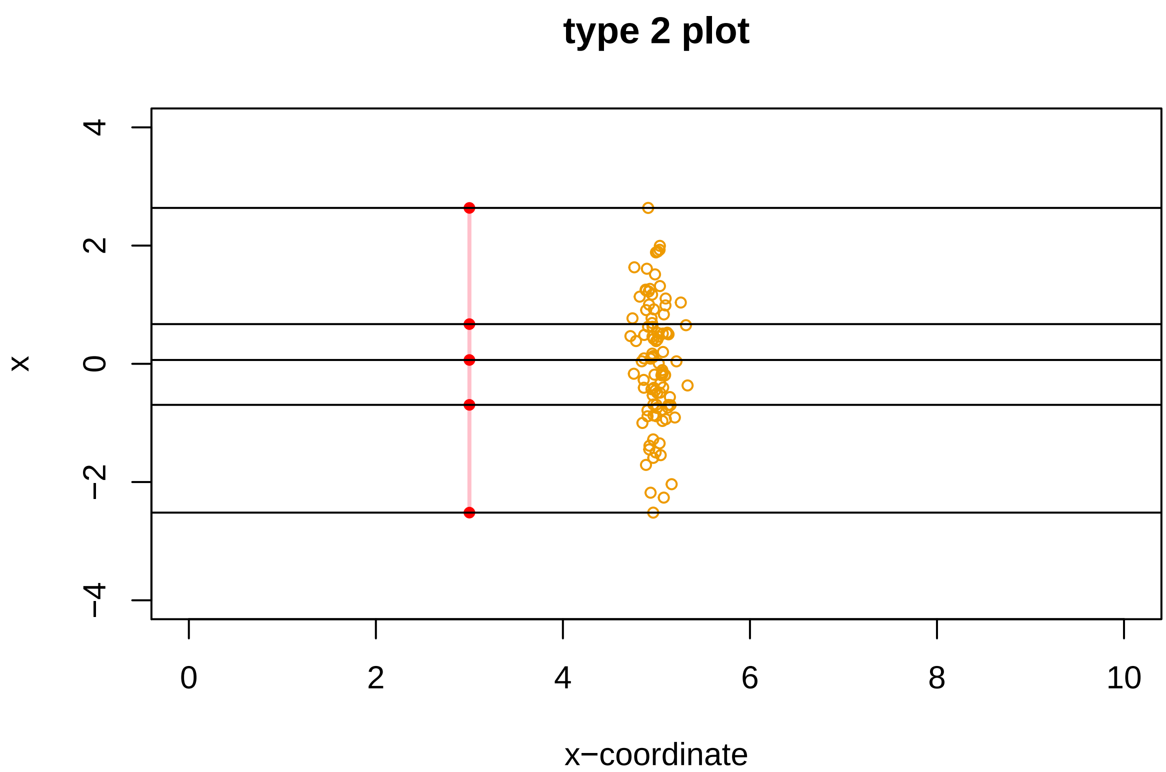
```
#type 1 and 2 quantiles
x <- rnorm(100) #replicate 100 numbers under the standard normal distribution
e1 <- quantile(x, type=1) #1st example
e1
```

```
##          0%         25%         50%         75%        100%
## -2.51721226 -0.69573077  0.04241591  0.65328397  2.63734764
```

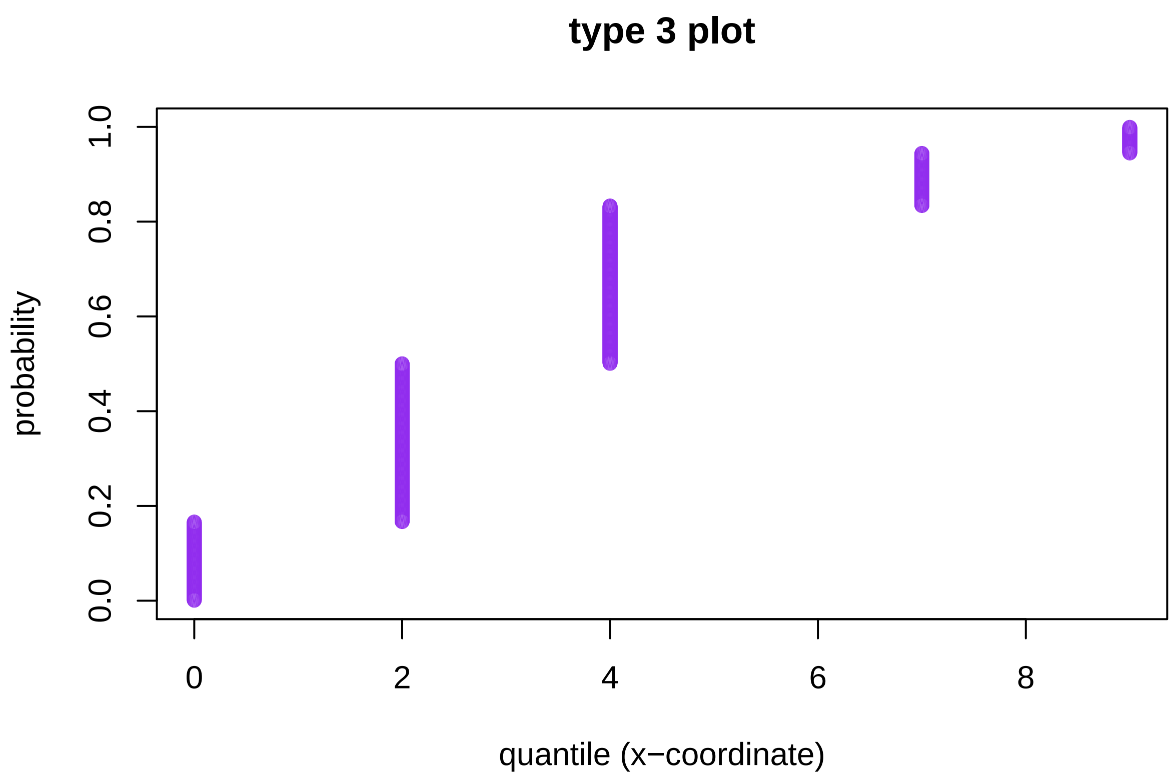
```
#type 1
plot(rep(5,100)+rnorm(100,sd=0.1),x, xlim=c(0,10), ylim=c(-4,4), main='type 1 plot', col='maroon', cex=
segments(1, e1[1], 1, e1[5], lwd=2, col='skyblue') #draw segment
points(rep(1,5), e1, pch=20, col='blue') #plot points
abline(h=quantile(x,type=1))
```



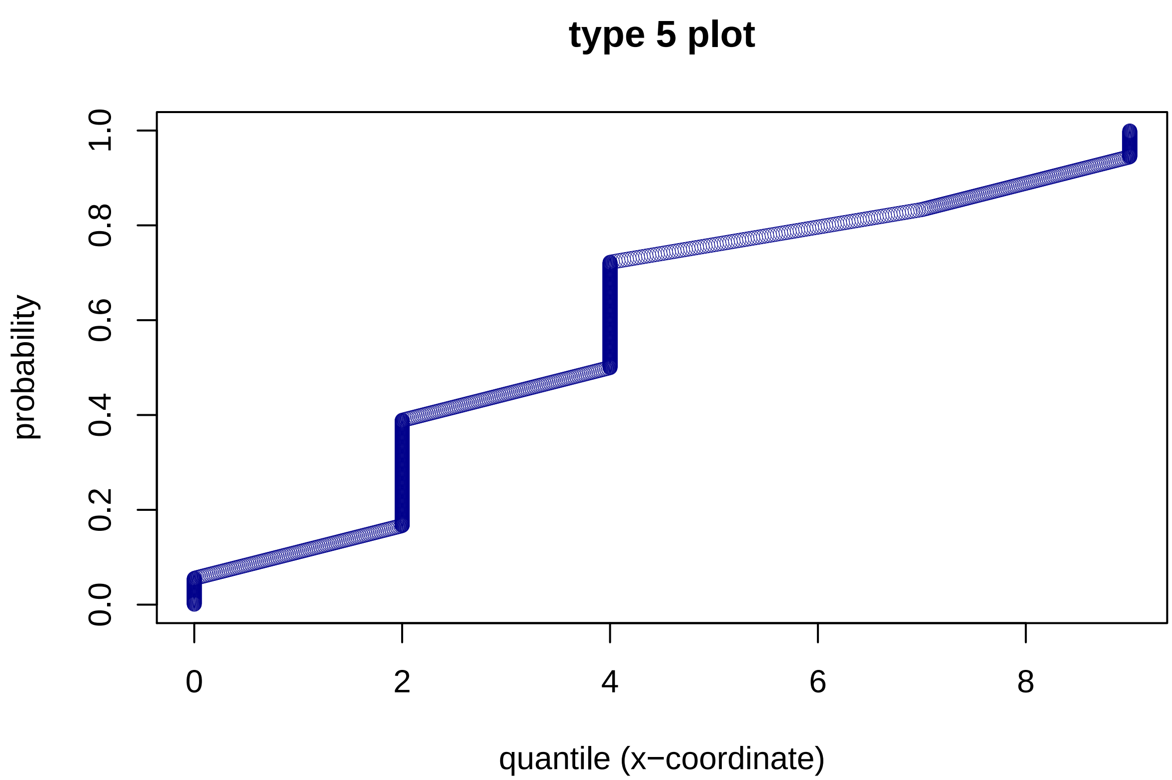

```
#type 2
e2 <- quantile(x, type=2) #2nd example
plot(rep(5,100)+rnorm(100,sd=0.1),x, xlim=c(0,10), ylim=c(-4,4), main='type 2 plot', col='orange2', cex
segments(3, e2[1], 3, e2[5], col='pink', lwd=2) #draw a segment
points(rep(3,5), e2,pch=20, col='red') #plot points
abline(h=quantile(x,type=2))
```



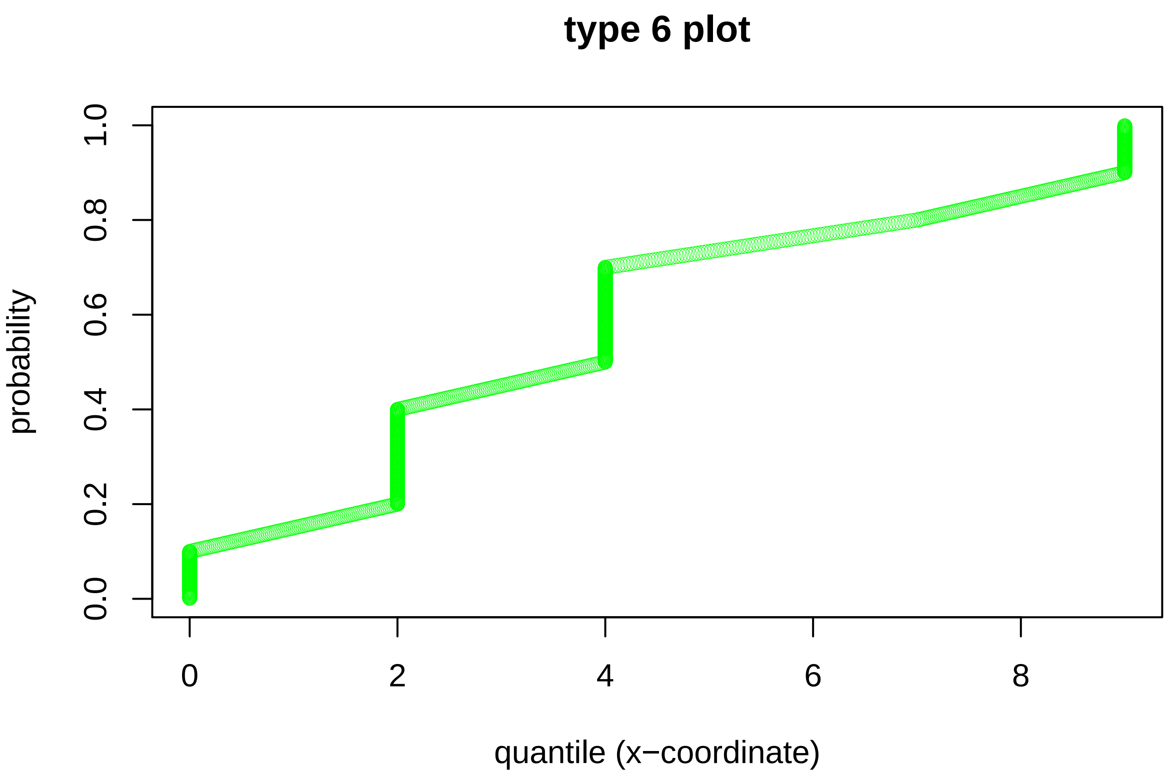

```
#type 3,5,6,7,8 quantiles
x <- c(0,2,2,2,4,4,4,7,9)
p = seq(0.001, 0.999, 0.001)

#type 3
q = quantile(x, probs = p, type=3) #3rd example
plot(q, p, lwd=0.05, col='purple2', main="type 3 plot", ylab="probability", xlab="quantile (x-coordinate
```



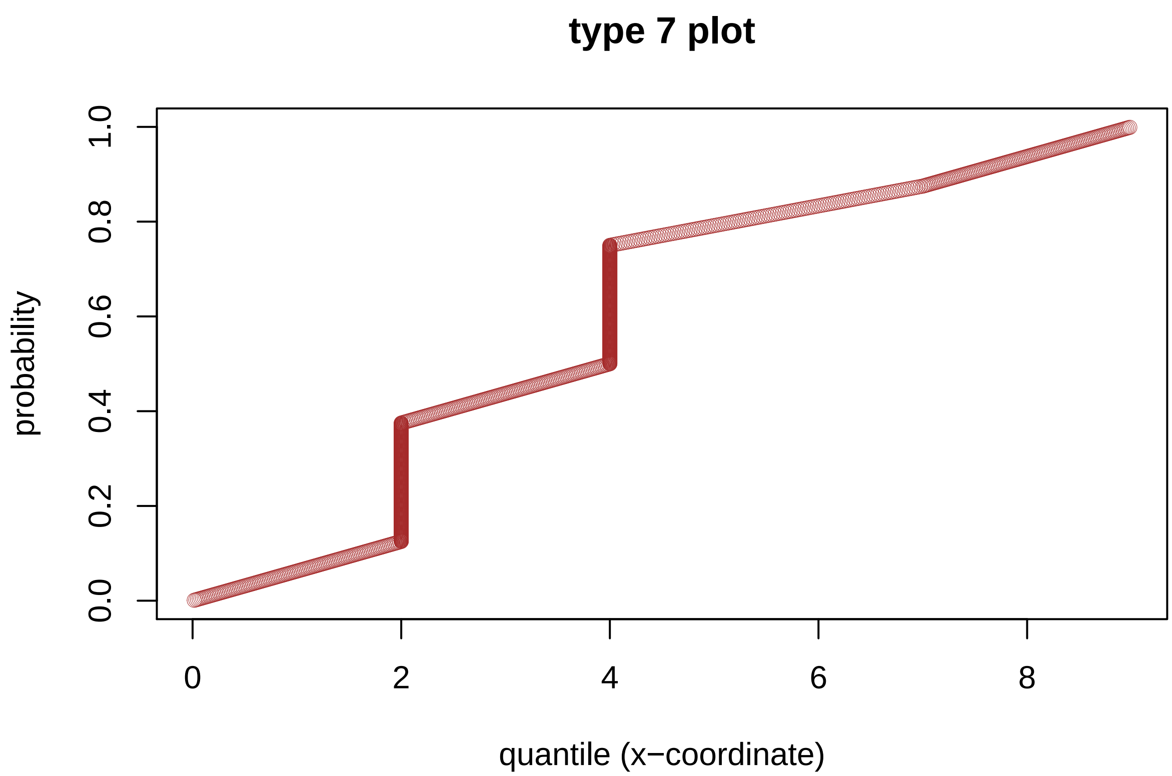

```
#type 5
q = quantile(x, probs = p, type=5) #4th example
plot(q, p, lwd=0.05, col='darkblue', main="type 5 plot", ylab="probability", xlab="quantile (x-coordina
```



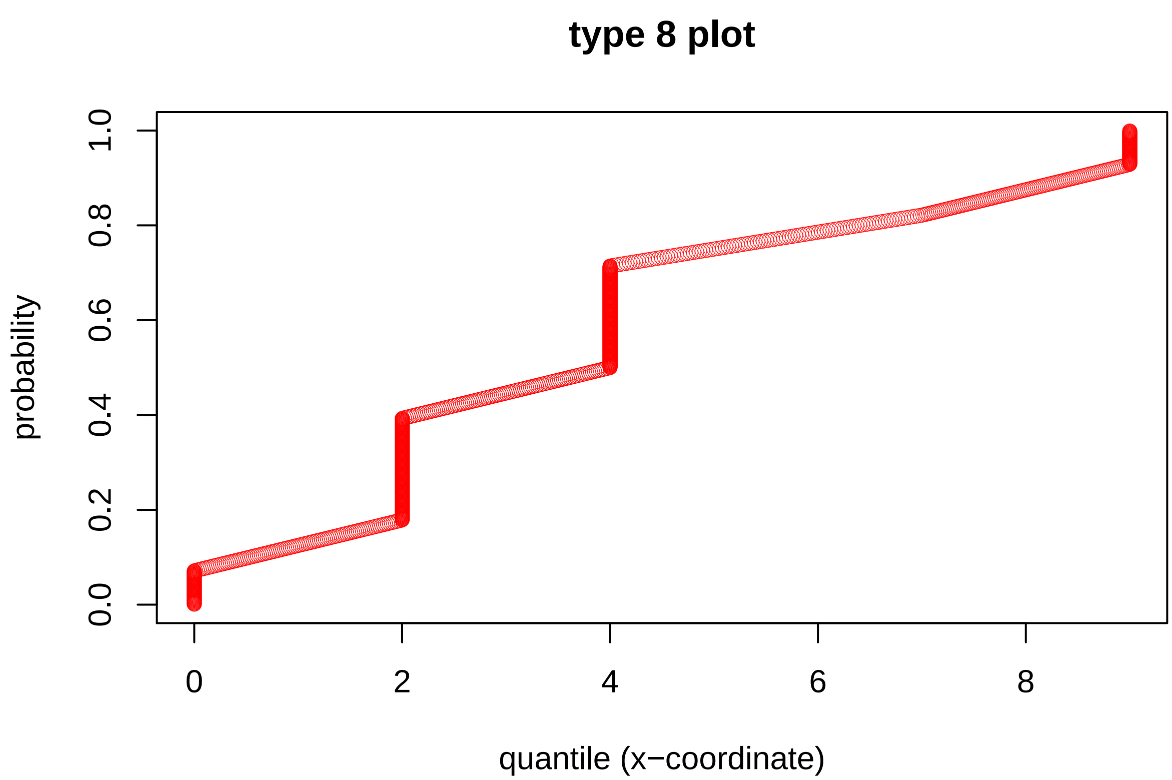
```r
#type 6
q = quantile(x, probs = p, type=6) #5th example
plot(q, p, lwd=0.05, col='green', main="type 6 plot", ylab="probability", xlab="quantile (x-coordinate)
```




```r
#type 7
q = quantile(x, probs = p, type=7) #6th example
plot(q, p, lwd=0.05, col='brown', main="type 7 plot", ylab="probability", xlab="quantile (x-coordinate)
```




```r
#type8
q = quantile(x, probs = p, type=8) #7th example
plot(q, p, lwd=0.05, col='red', main="type 8 plot", ylab="probability", xlab="quantile (x-coordinate)")
```



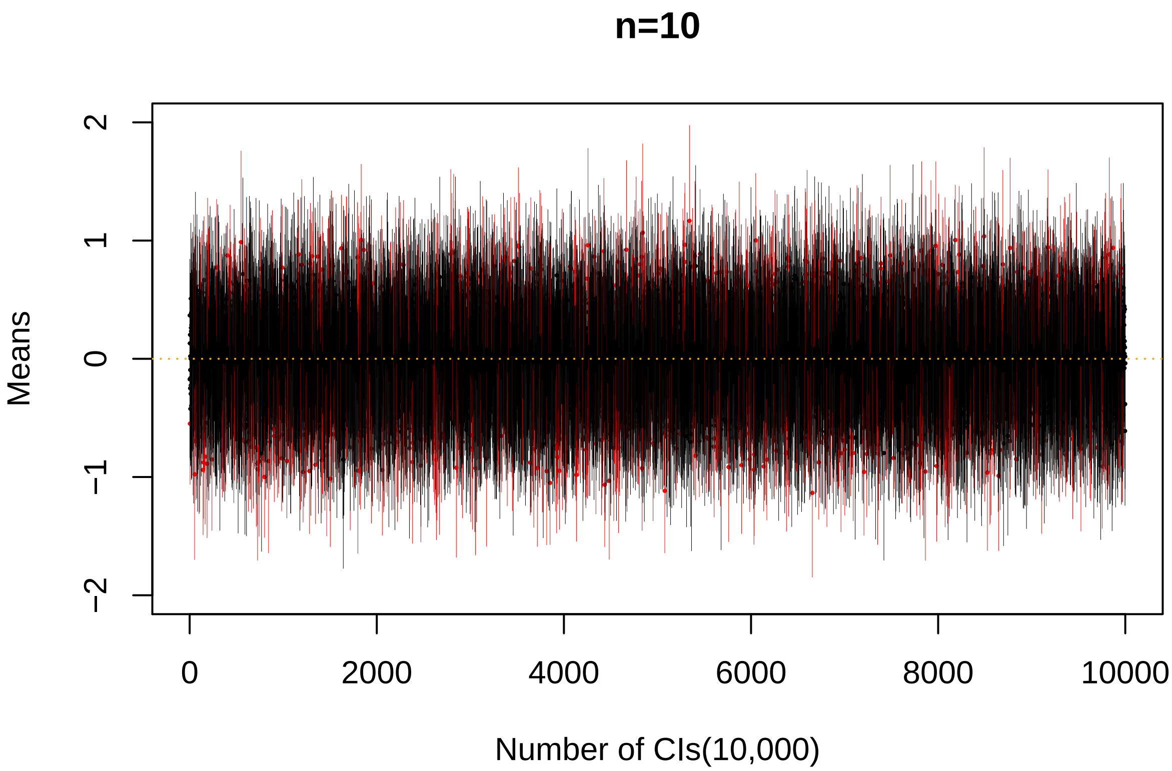
**Problem 2. Perform some simulations to quantify the level of the Student confidence interval for the mean. (Think about why it does not matter what the mean and variance of the underlying distribution are.) Set the level at 90%.**

```
n = c(10,100,1000) #simulate for n=10, 100, 1000
c = c() #create an empty vector

for(i in n){
  x <- rnorm(i)
  y <- t.test(x, conf=0.9)
  lb <- y$conf.int[1]
  ub <- y$conf.int[2]
  c <- c + c(cat(("("),lb ,(","), ub,(")  ")))
}
```

```
## ( -0.6409827 , 0.4384525 )  ( -0.174133 , 0.1751645 )  ( -0.01095139 , 0.09043703 )
```

**A. Generate a standard normal sample of size n and compute the Student CI. Repeat that B = 10,000 times. Compute the fraction of times the interval contains the true mean. Do this for n = 10, 20, . . . , 100. Produce one or several nice plots displaying the result of these simulations.**

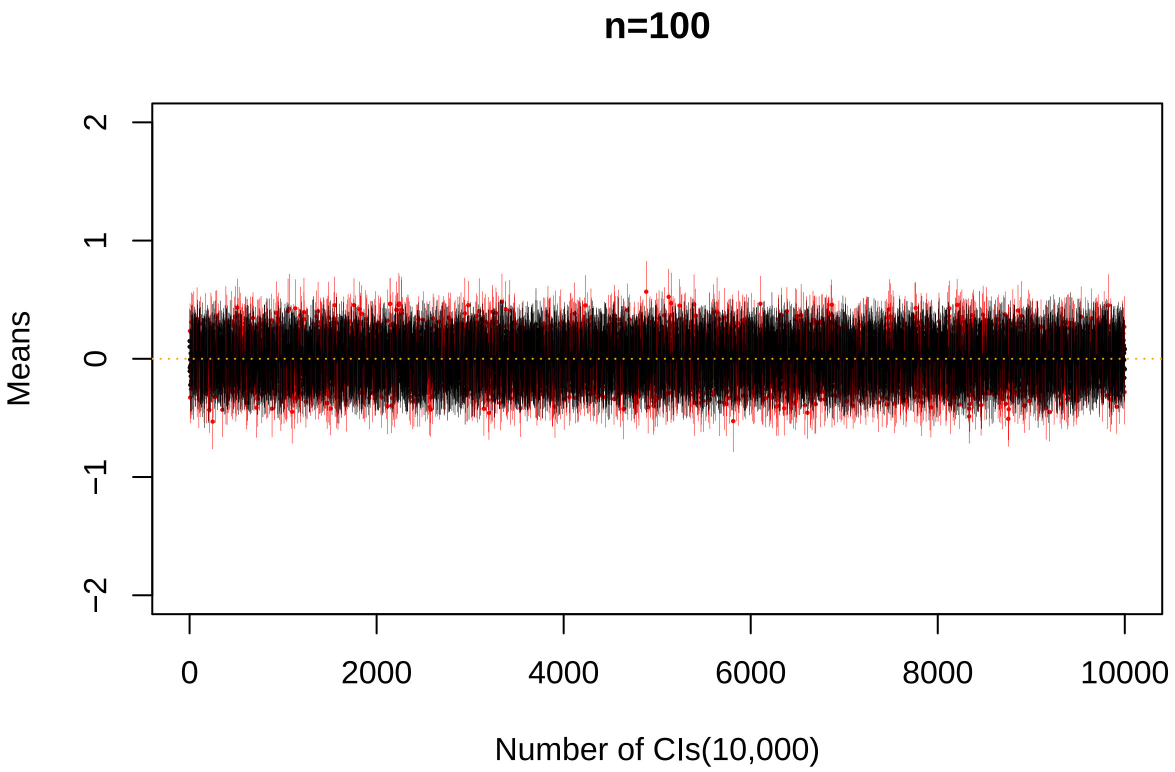
```
#define variables
alpha <- 0.10
mu <- 0
n <- 10
B <- 10000

#setup empty vectors
mean_vec   = numeric(B) #vectors for the mean
ub = numeric(B) #vector for the lb
lb = numeric(B) #vector for the ub

for (i in 1:B) { #forloop to calculate the CI for student's t-dist
  x <- rnorm(n)
  y = t.test(x, conf=1-alpha)
  lb[i] = y$conf.int[1]
  ub[i] = y$conf.int[2]
  mean_vec[i]   = y$estimate
}

#show in red when CI does not include mean
contains_mean = (lb < mu) & (mu < ub)
color_vec     = ifelse(contains_mean == TRUE, 1, 2) #red color when interval does not contain the true 

#plot the graph
plot(1:B, mean_vec, pch=20, ylim = c(-2, 2), col=color_vec, main='n=10', xlab='Number of CIs(10,000)', 
segments(1:B, lb, 1:B, ub, col=color_vec, lwd=0.05)
abline(h=0, lty="dotted", col='orange', lwd=1)
```




```
#calculte for the fraction
frac <- replace(contains_mean, contains_mean[contains_mean == FALSE], 0) #replace TRUE with 1 and FALSE
total.means <- sum(frac[frac == 1]) #sum all the 1's in the vector
fraction <- total.means/length(frac) #calculate for the fraction
cat(('The proportion of times the interval contains the true mean is:'),print(fraction),', which is very
```

```
## [1] 0.9011
## The proportion of times the interval contains the true mean is: 0.9011 , which is very close to the
```

```
################################################################
#When n = 100
n <- 50

for (i in 1:B) {
  x <- rnorm(n)
  y = t.test(x, conf=1-alpha)
  lb[i] = y$conf.int[1]
  ub[i] = y$conf.int[2]
  mean_vec[i]    = y$estimate
}

#show in red when CI does not contain mean
contains_mean = (lb < mu) & (mu < ub)
color_vec     = ifelse(contains_mean == TRUE, 1, 2)

#plot the graph
plot(1:B, mean_vec, pch=20, ylim = c(-2, 2), col=color_vec, main='n=100', xlab='Number of CIs(10,000)',
segments(1:B, lb, 1:B, ub, col=color_vec, lwd=0.05)
abline(h=0, lty="dotted", col='orange', lwd=1)
```




```
#calculate for the fraction
frac1 <- replace(contains_mean, contains_mean[contains_mean == FALSE], 0) #replace TRUE with 1 and FALS
total.means <- sum(frac1[frac1 == 1]) #sum all the 1's in the vector
fraction1 <- total.means/length(frac1) #calculate for the fraction
cat(('The proportion of times the interval contains the true mean is:'),print(fraction1),', which is ve
```

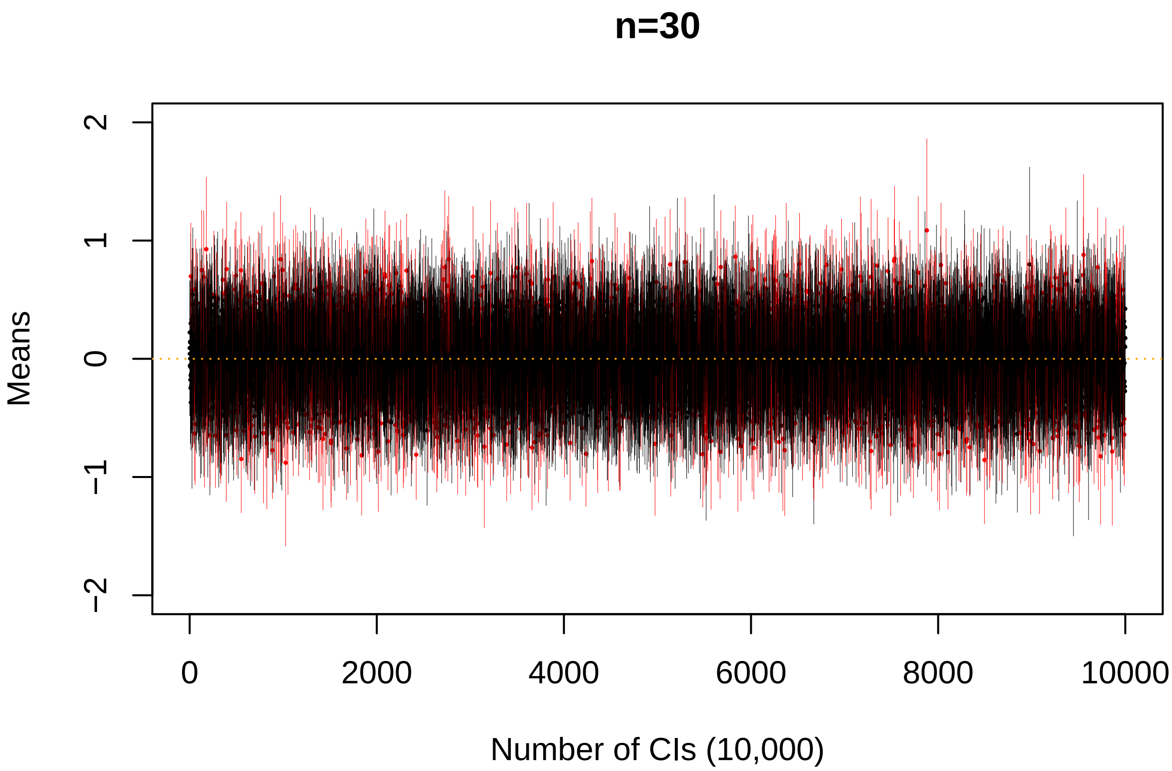
```
## [1] 0.8949
## The proportion of times the interval contains the true mean is: 0.8949 , which is very close to the
```

**B. Same thing, except generate a sample from the double-exponential distribution with rate lambda = sqrt(2), which also has mean 0 and variance 1, and is not in the normal family.**

```
#install.packages("rmutil") #install rmutil
library(rmutil) #import rmutil
```

```
##
## Attaching package: 'rmutil'

## The following object is masked from 'package:stats':
##
##     nobs

## The following objects are masked from 'package:base':
##
##     as.data.frame, units
```

```r
#define variables
alpha <- 0.1 #significance value
n <- 30
mu <- 0 #mu for the CI
par <- 0 #location parameter for the laplace distribution
lambda <- sqrt(2)
#rlaplace(n, m=0, s=1/lambda) #double exponential distribution with rate lambda = sqrt(2)
B <- 10000

#setup empty vectors
mean.vec <- numeric(B) #vectors for the mean
ub <- numeric(B) #vector for the ub for the CI
lb <- numeric(B) #vector for the lb for the CI

for (i in 1:B){ #forloop to compute for the CI 10000 times
  x <-rlaplace(n)
  y <- t.test(x, conf=1-alpha)
  lb[i] <- y$conf.int[1]
  ub[i] <- y$conf.int[2]
  mean.vec[i] <- y$estimate
}

#show in red when CI does not contain the mean
contains_mean <- (lb < mu) & (mu < ub) #set interval
color_vec <- ifelse(contains_mean == TRUE, 1,2) #red color when Ci does not contain the true mean

#plotting the graph
plot(1:B, mean.vec, pch=20, ylim=c(-2,2), col=color_vec, main='n=30', xlab='Number of CIs (10,000)', yla
segments(1:B, lb, 1:B, ub, col=color_vec, lwd=0.05)
abline(h=0, lty="dotted", col='orange', lwd=1)
```



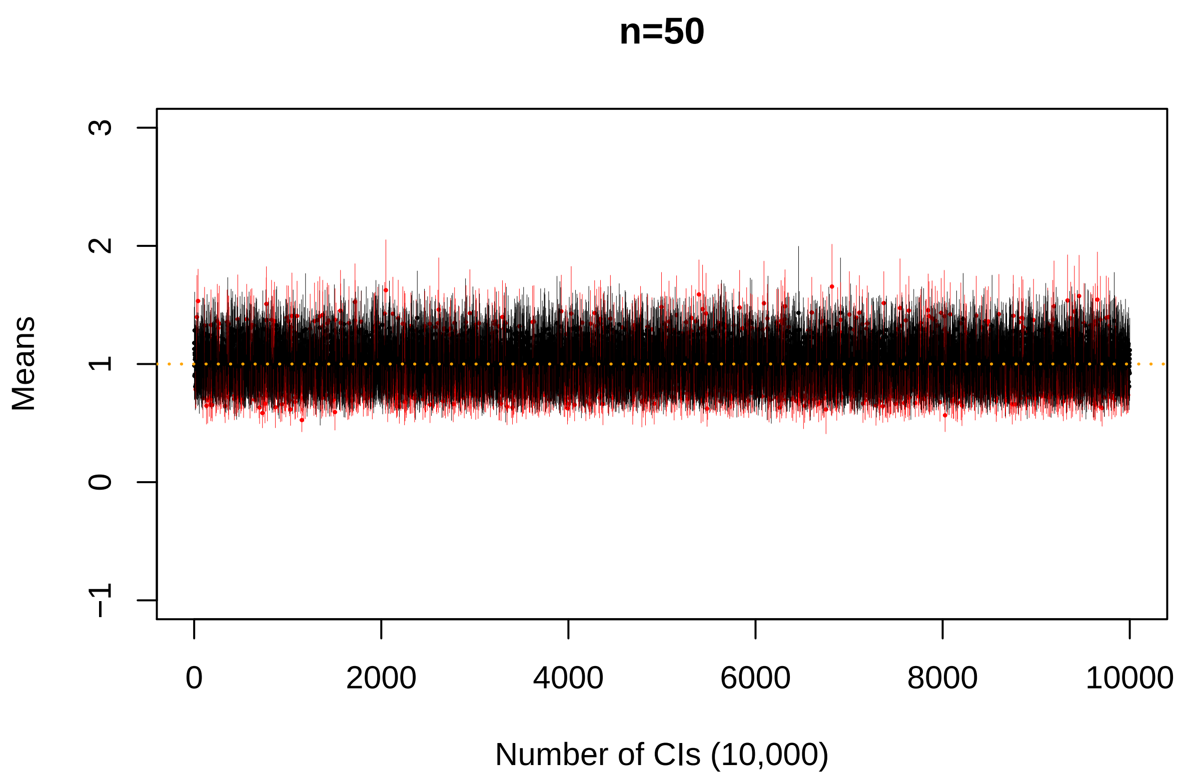

```
#calculate for the fraction
frac2 <- replace(contains_mean, contains_mean[contains_mean == FALSE], 0) #replace TRUE with 1 and FALS
total.means <- sum(frac2[frac2 == 1]) #sum all the 1's in the vector
fraction2 <- total.means/length(frac2) #calculate for the fraction
cat(('The proportion of times the interval contains the true mean is:'),print(fraction2),', which is ve
```

```
## [1] 0.9065
## The proportion of times the interval contains the true mean is: 0.9065 , which is very close to the
```

**C. Same thing, except generate a sample from the exponential distribution with rate lambda = 1, which also has mean 1 and variance 1, and is not in the normal family.**

```
#define variables
alpha <- 0.1 #significance value
n <- 50
mu <- 1 #mu for the CI
lambda <- 1
B <- 10000

#setup vectors
meanvec <- numeric(B) #ectors for the mean
ub <- numeric(B) #vector for the ub for the CI
lb <- numeric(B) #vector for the lb for the CI

for (i in 1:B){ #forloop to compute for the CI 10000 times
  x <-rexp(n, rate=lambda)
  y <- t.test(x, conf=1-alpha)
  lb[i] <- y$conf.int[1]
```

```
  ub[i] <- y$conf.int[2]
  meanvec[i] <- y$estimate
}

#show in red when CI does not contain the mean
contains_mean <- (lb < mu) & (mu < ub) #set interval
color_vec <- ifelse(contains_mean == TRUE, 1,2) #red color when Ci does not contain the true mean

#plotting the graph
plot(1:B, meanvec, pch=20, ylim=c(-1,3), col=color_vec, main='n=50', xlab='Number of CIs (10,000)', ylal
segments(1:B, lb, 1:B, ub, col=color_vec, lwd=0.05)
abline(h=1, lty="dotted", col='orange', lwd=1.5)
```




```
#calculate for the fraction
frac3 <- replace(contains_mean, contains_mean[contains_mean == FALSE], 0) #replace TRUE with 1 and FALS
total.means <- sum(frac3[frac3 == 1]) #sum all the 1's in the vector
fraction3 <- total.means/length(frac3) #calculate for the fraction
cat(('The proportion of times the interval contains the true mean is:'),print(fraction3),'which is clos
```

```
## [1] 0.8877
## The proportion of times the interval contains the true mean is: 0.8877 which is close to the 90% con
```