Stats 102C, Homework 2

Jang Junhyuk

SID: 004 728 134

Problem 1 - Box Muller Transform

Write a function called `my_rnorm()` that will generate samples from a normal distribution. The function must be able to accept optional arguments for mean and sd. The default values are 0 and 1 if the mean and sd are not specified.

Use `runif()` as your source of randomness. You are not allowed to use any of R's normal distribution function (e.g. pnorm, qnorm, dnorm).

Use your function to generate 500 values from the standard normal distribution.

Plot the kernel density estimate of the resulting sample. Plot the theoretic density (you can use dnorm) in another color on top of the kernel density estimate. Comment on the plot.

Create a qqnorm plot. Comment on the plot.

Perform a Shapiro-Wilk test for normality. Be sure to comment on the results.

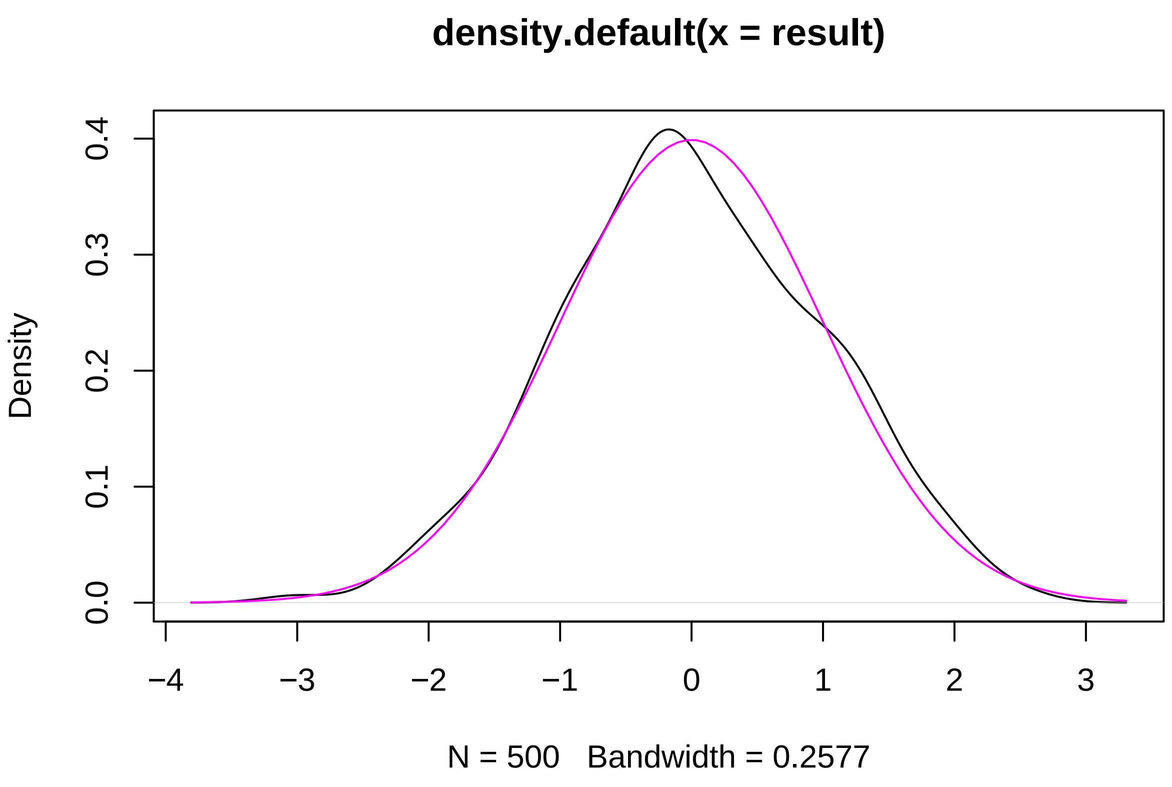
```r
set.seed(1)
my_rnorm <- function(n, mean = 0, sd = 1){
        x <- rep(NA,n)
        y <- rep(NA,n)
        for(i in 1: n){
                th <- runif(1,0,2*pi)
                v <- -2 * log(runif(1))

                x[i] <- sqrt(v) * cos(th)
                y[i] <- sqrt(v) * sin(th)
        }
        return(x)

}

set.seed(1)
rnorm(2)
```

```
## [1] -0.6264538  0.1836433
```

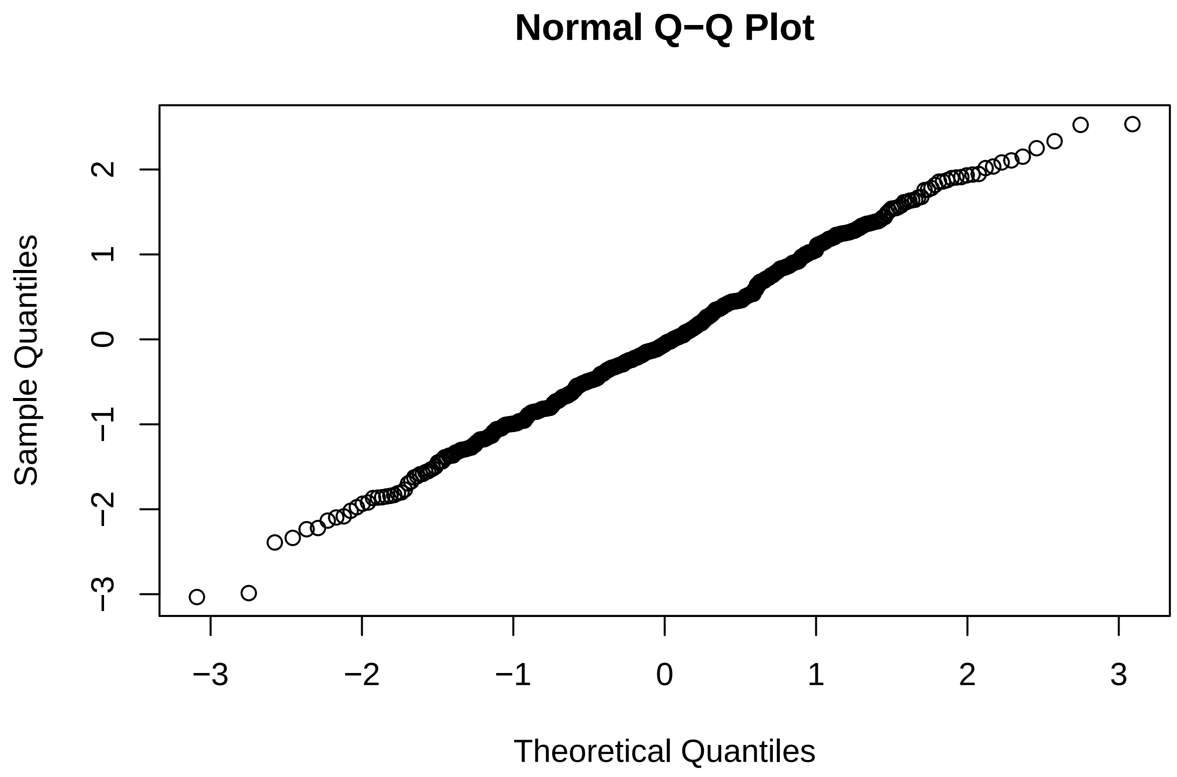
```r
runif(2)
```

```
## [1] 0.2016819 0.8983897
```

```r
result <- my_rnorm(500)

plot(density(result))
curve(dnorm(x, mean=0, sd=1), add=TRUE,log = FALSE, col=6)
```



```r
# we can see the values we have generated are similar to
# the standard normal distribution.

qqnorm(result)
```



```r
# As we can see, points do fall along a straight line indicates that
# our data come from normal distribution.

shapiro.test(result)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  result
## W = 0.99659, p-value = 0.371
```

```r
# Since my p-value is greater than 0.05, there is an insufficient evidence
# to reject the null hypothesis and conclude that our sample drawn from
# the normally distributed population.
```

Problem 2 - RNG based on distribution definitions and convolutions

Using only `runif()` and/or `rnorm()` and the definitions of distributions, generate 500 samples from each of the following distributions. You are not allowed to use any of R's distribution functions for the generation of random values.

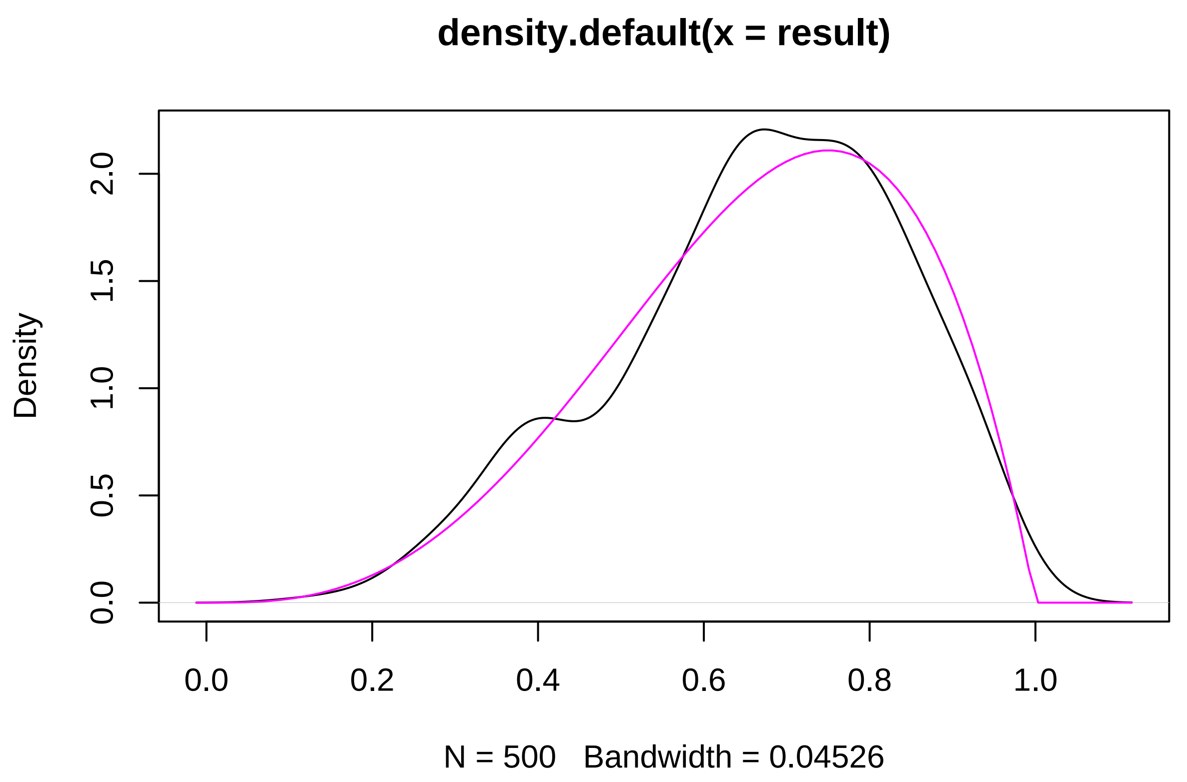
For each distribution:

- After generating your 500 samples, plot the kernel density estimate of the resulting sample. Plot the theoretic density (you can use dchisq, dt, etc.) in another color on top of the kernel density estimate. Comment on the plot.
- Plot the empirical CDF function of your data. Add the theoretic CDF (you can use pchisq, pt, etc.) of the distribution to the same plot (in a different color).
- Use the Kolmogorov-Smirnov test to compare your generated samples to the theoretic distributions. Be sure to print out the resulting p-value and comment on the sample produced by your function.

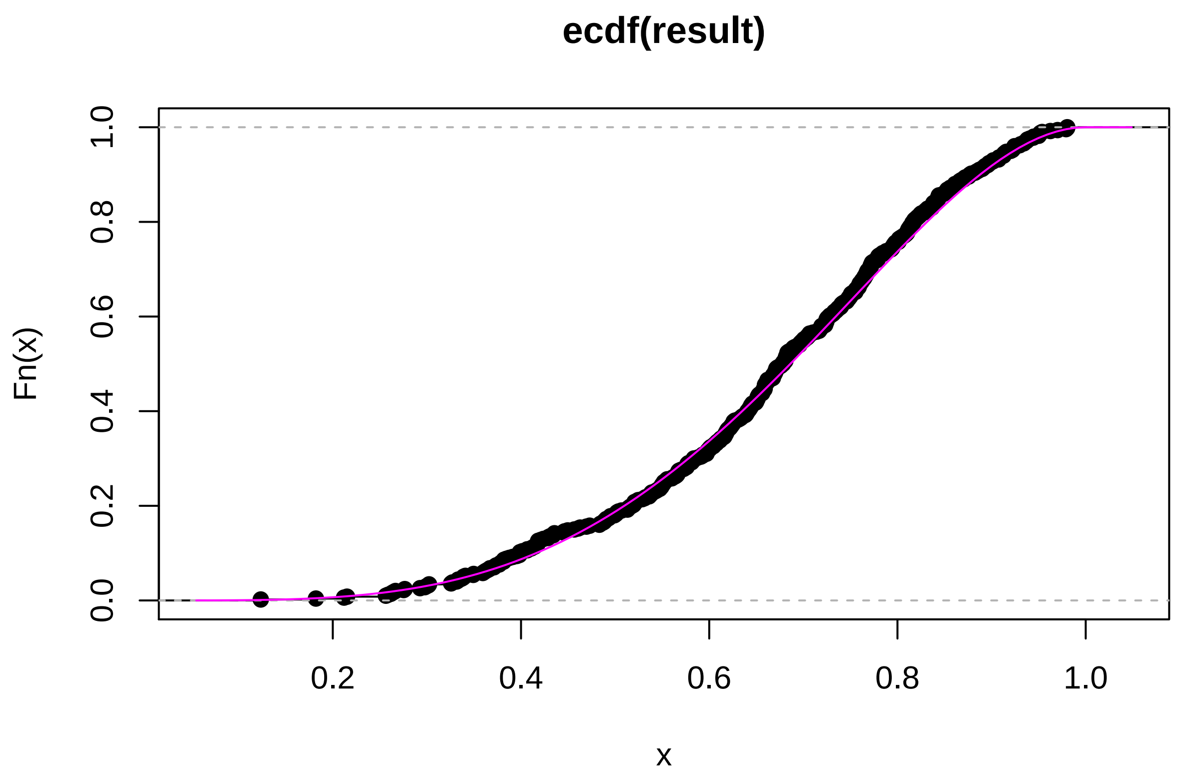
Problem 2a:

- Beta distribution with shape parameters 4 and 2

```r
set.seed(1)
beta <- function(n){
        a <- rep(NA,n)
        b <- rep(NA,n)
        for(i in 1:n){
                uni <- runif(6)
                expo <- -log(uni)
                a[i] <- sum(expo[1:4])
                b[i] <- sum(expo[1:6])
        }
        return(a/b)
}
result <- beta(500)

plot(density(result))
curve(dbeta(x, 4,2), add=TRUE,log = FALSE, col=6)
```



```
# we can see the values we have generated are similar to
# the beta(4,2).

plot(ecdf(result))
curve(pbeta(x,4,2),add = TRUE,log = FALSE, col = 6)
```



```r
ks.test(result,pbeta, shape = 4,shape2 = 2)
```

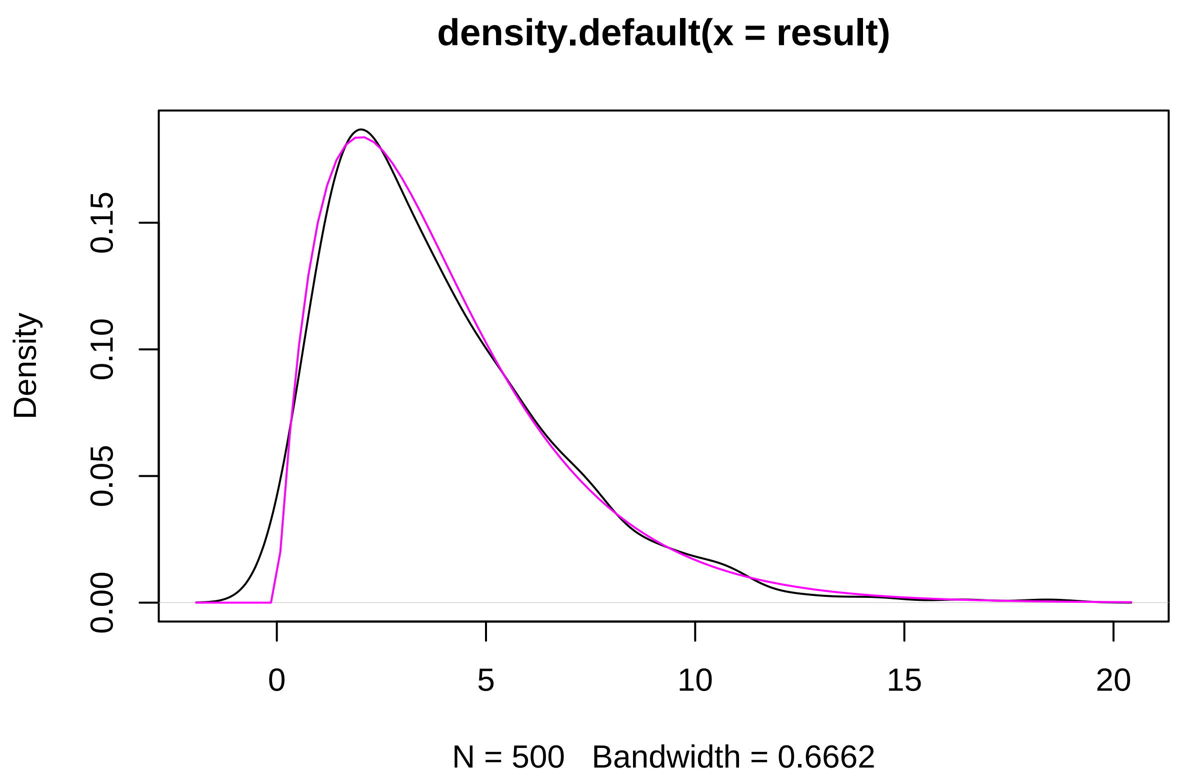
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  result
## D = 0.032859, p-value = 0.6529
## alternative hypothesis: two-sided
```

```r
# p > 0.05
# Large p-value indicates that there is an insufficient evididence to reject the
# null hypothesis and coclude that the two data come from the
# same distribution.
```

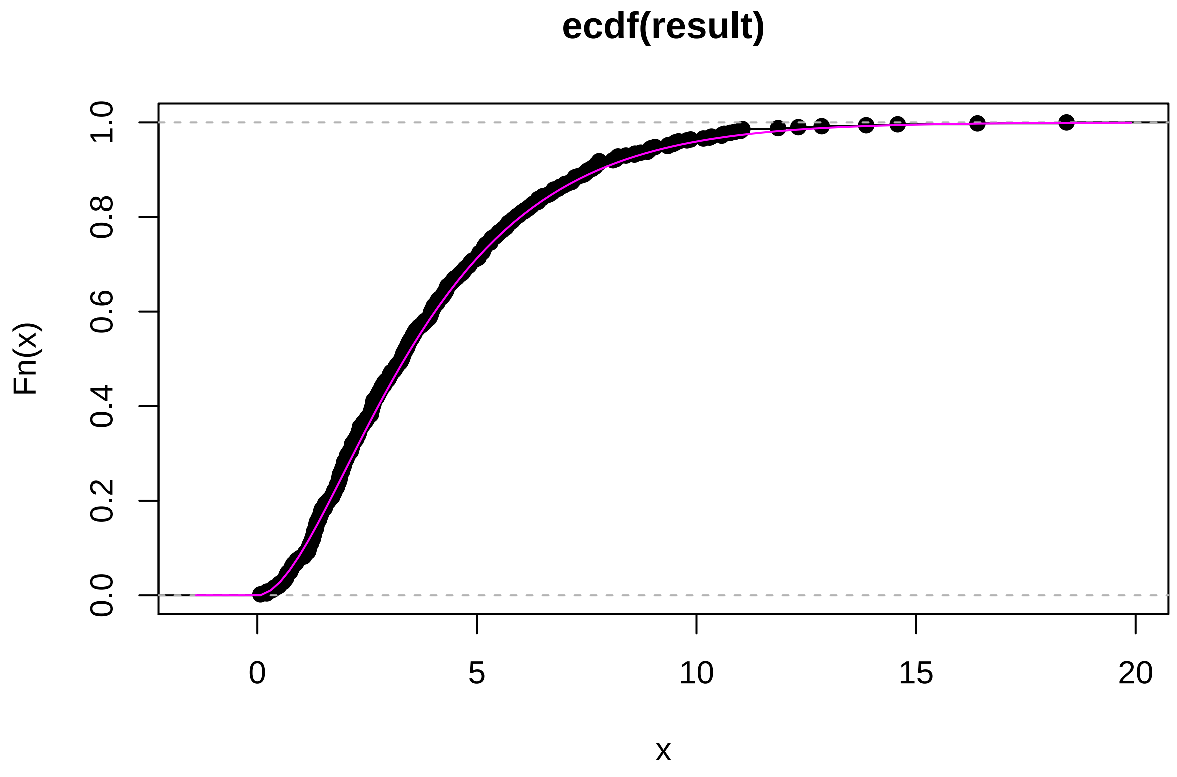
Problem 2b:

- Chi-squared distribution with 4 degrees of freedom

```r
set.seed(1234)
chi <- function(n){
        c <- rep(NA,n)
        for(i in 1:n){
                uni <- rnorm(4)
                c[i] <- sum(uni^2)
        }
        return(c)
}
result <- chi(500)

plot(density(result))
curve(dchisq(x,4), add=TRUE,log = FALSE, col=6)
```



```r
# we can see the values we have generated are similar to
# the chisq(df = 4).

plot(ecdf(result))
curve(pchisq(x,4),add = TRUE,log = FALSE, col = 6)
```



```r
ks.test(result,pchisq,df = 4)
```

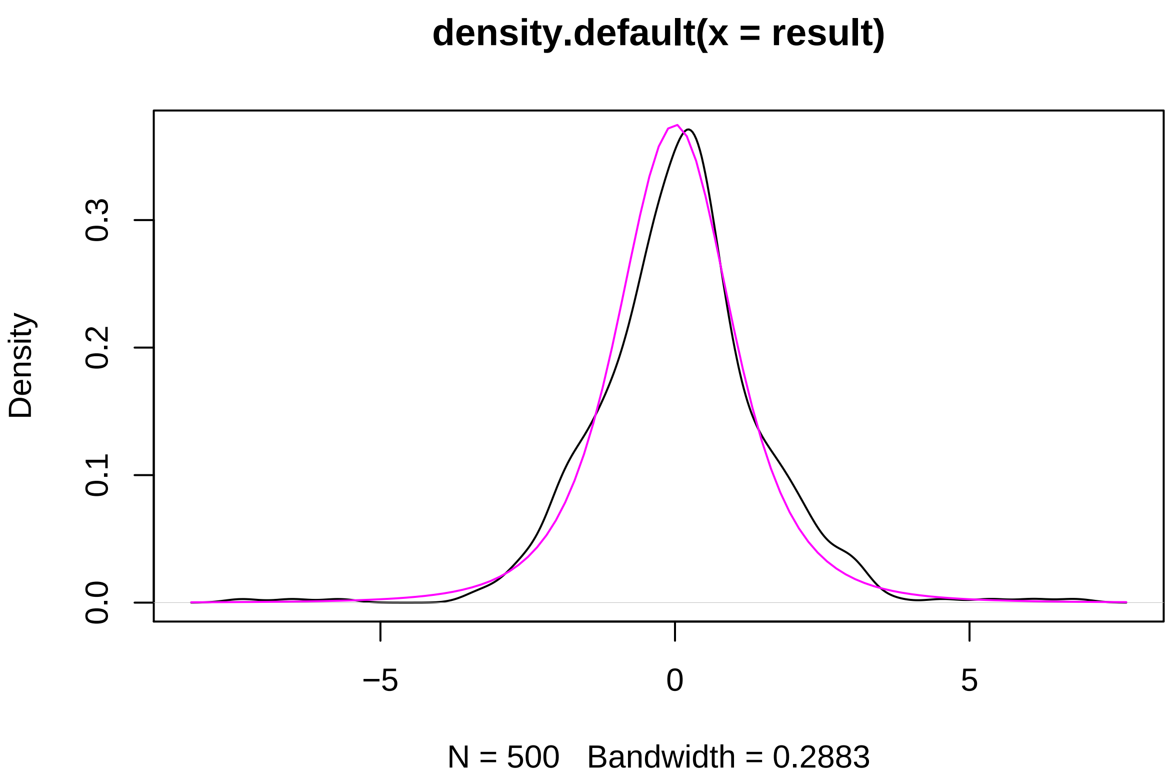
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  result
## D = 0.030518, p-value = 0.7403
## alternative hypothesis: two-sided
```

```r
# p > 0.05
# Large p-value indicates that there is an insufficient evididence to reject the
# null hypothesis and coclude that the two data come from the
# same distribution.
```

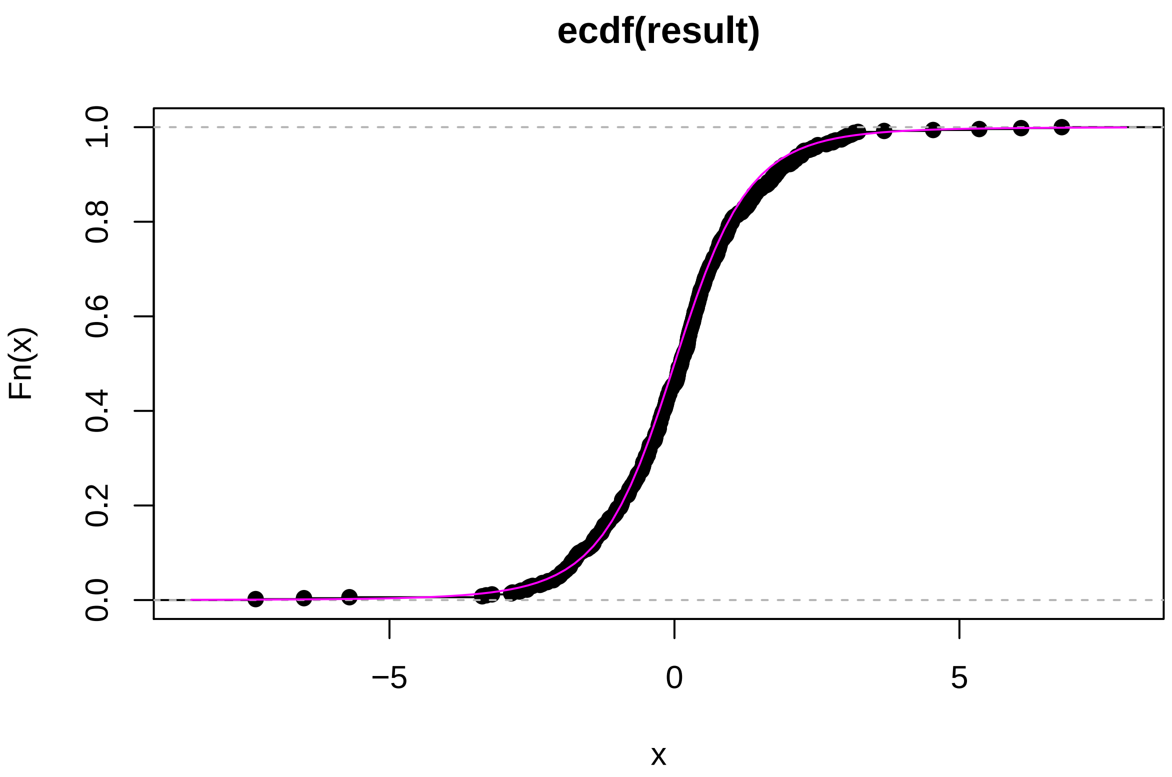
Problem 2c:

- t-distribution with 4 degrees of freedom

```r
set.seed(1234)
chi <- function(n){
        c <- rep(NA,n)
        for(i in 1:n){
                uni <- rnorm(4)
                c[i] <- sum(uni^2)
        }
        return(c)
}

result <- chi(500) # chi with df = 4

tdis <- function(n){
        t <- rep(NA,n)
        for(i in 1:n){
              z <- rnorm(1)
              t[i] <- z / sqrt((result[i])/4)
        }
        return(t)
}
result <- tdis(500)

plot(density(result))
curve(dt(x,4), add=TRUE,log = FALSE, col=6)
```



```r
# we can see the values we have generated are similar to
# the t-distribution with 4 degree of freedom.

plot(ecdf(result))
curve(pt(x,4),add = TRUE,log = FALSE, col = 6)
```



```
ks.test(result,pt,df = 4)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  result
## D = 0.051478, p-value = 0.1413
## alternative hypothesis: two-sided
```

```
# p > 0.05
# Large p-value indicates that there is an insufficient evididence to reject the
# null hypothesis and coclude that the two data come from the
# same distribution.
```

Problem 3 - Bivariate Normal RNG

Use `rnorm()` to generate random values Z from N(0,1).

Identify a lower triangular matrix A and vector b, such that b + AZ, will come from a multivariate normal, with mean c(1,-2) and covariance matrix matrix(c(9, 12, 12, 25), nrow = 2). You can use `chol()` for this, but you could be tested to do the decomposition of a 2x2 matrix by hand.

Use the generated Z values and your matrix A and vector b to produce 500 pairs of values from the bivariate normal with mean c(1,-2) and covariance matrix matrix(c(9, 12, 12, 25), nrow = 2)

Create a scatter plot of your data. Use pch = 19, and cex = 0.4 to make the size the points small. Add contour lines to the plot. See

After generating your pairs of data, run a multivariate Shapiro-Wilk test to test the normality of your generated data. You'll need `library(mvnormtest)`

```
library("mvnormtest")
set.seed(1)
n <- 500
z1 <- rnorm(n)
z2 <- rnorm(n)
Z <- rbind(z1,z2)

sig <- matrix(c(9, 12, 12, 25), nrow = 2)
mu <- c(1,-2)

# Decompse sigma
t11 = sqrt(sig[1,1])
t21 = sig[1,2]/sqrt(sig[1,1])
t22 = sqrt(sig[2,2]-(sig[1,2]/t11)^2)

(A  = matrix(c(t11,t21,0,t22),nrow = 2))
```

```
##      [,1] [,2]
## [1,]    3    0
## [2,]    4    3
```

```
A %*% t(A)
```

```
##      [,1] [,2]
## [1,]    9   12
## [2,]   12   25
```

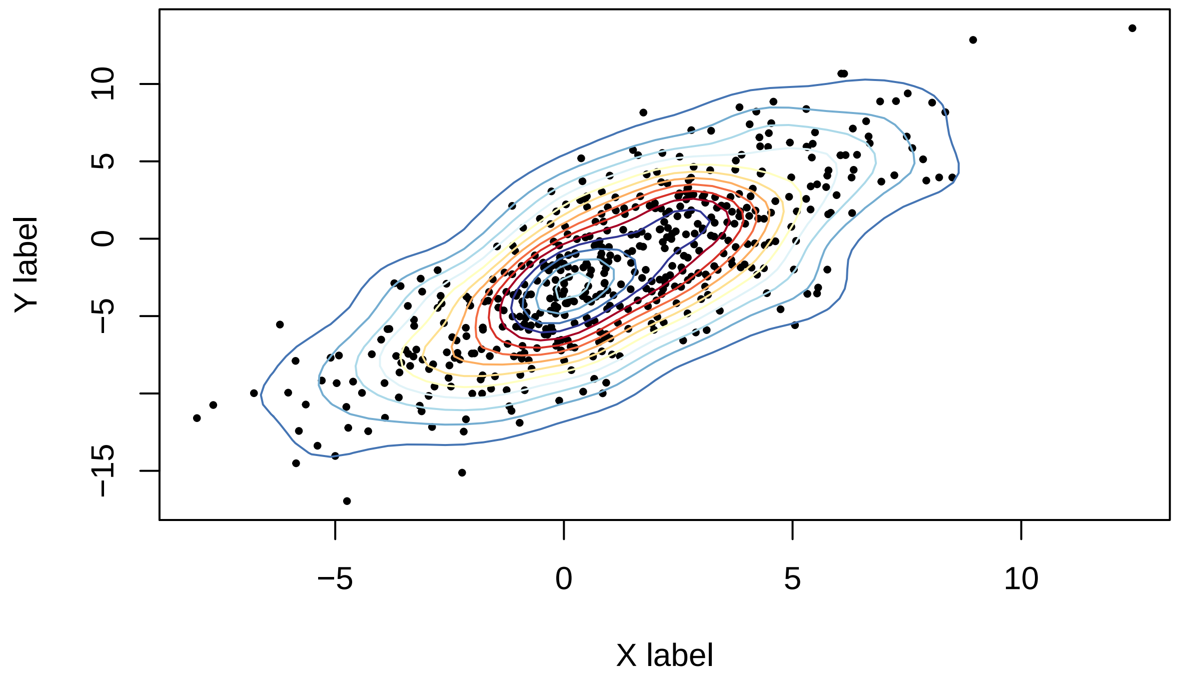
```
X = mu + A %*% Z
X = t(X)
colMeans(X)
```

```
## [1]  1.067932 -2.047245
```

```
cov(X)
```

```
##          [,1]     [,2]
## [1,]  9.21599 11.89098
## [2,] 11.89098 25.38633
```

```
# contour lines to plot
library(MASS)
library(RColorBrewer)
k <- 11
my.cols <- rev(brewer.pal(k, "RdYlBu"))

z <- kde2d(X[,1], X[,2], n=50)
plot(X, xlab="X label", ylab="Y label", pch=19, cex=.4)
contour(z, drawlabels=FALSE, nlevels=k, col=my.cols, add=TRUE)
```



```
# multivariate Shapiro-Wilk test
mshapiro.test(t(X))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.99802, p-value = 0.8362
```

```
# Since my p-value is greater than 0.05, there is an insufficient evidence
# to reject the null hypothesis and conclude that our sample drawn from
# the multivariate normally distributed population.
```

Problem 4 - Monte Carlo Integration

Estimate the following integral by using Monte Carlo estimation.

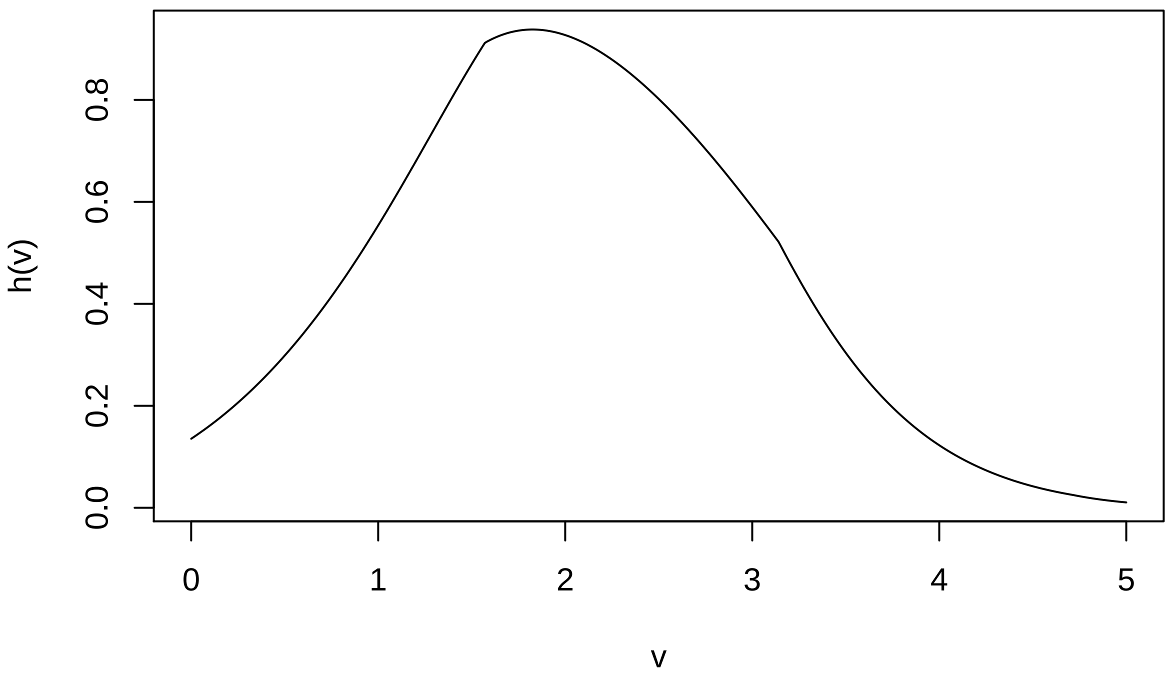
$$\mu = \int_0^5 \exp(-0.5(x-2)^2 - 0.1|\sin(2x)|)dx$$

Generate 5000 samples to estimate $\hat{\mu}$ and compute its standard deviation. Use `runif()` to generate the random uniform values.

Create a plot using the cumulative mean (aka running mean) of the first 1000 values to show how the estimate 'settles' over the course of the samples. Add confidence bands 2 standard errors above and below the running mean. Be sure to use the running standard error for this part. Your plot should resemble figure 3.3 on page 67 of the textbook.

Take note of the limits of the y-axis, in our next HW assignment, you'll compare the performance of classical Monte Carlo integration with importance sampling.

```r
# what the function looks like
h <- function(x) exp(-0.5 * (x - 2) ^ 2 - 0.1 * abs( sin(2*x) ) )
v <- seq(0,5, by = 0.01)
plot(v, h(v), type = "l")
```



```r
set.seed(1)
hx <- rep(NA,5000)
for(i in 1:5000){
        p <- runif(1,0,5)
        hx[i] <- h(p)
}
# monte carlo estimator
mean(hx)
```

```
## [1] 0.4552536
```

```r
# mu_hat
mean(hx) * 5
```

```
## [1] 2.276268
```

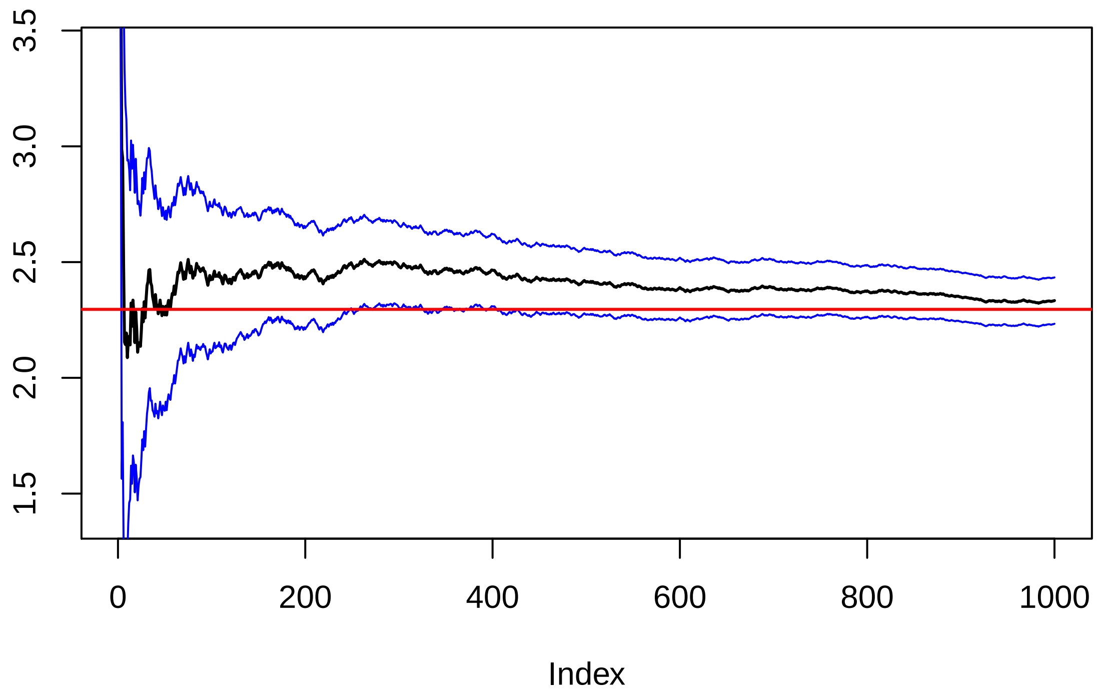
```r
# standard deviation from 5000 samples
sd(hx) *(5/sqrt(length(hx)))
```

```
## [1] 0.02285388
```

```r
cummean <- cumsum(hx) / seq_along(hx)
m <- cummean * 5
x <- hx * 5
# Error estimation
cummean <- cummean[1:1000]
m <- cummean * 5
x <- hx[1:1000] * 5
# standar deviation
error <- sqrt(cumsum((x - m)^2))/(1:1000)
# Plot using the cumulativ mean of the first 1000 values
# Confindence bands 2 standard errors
par(mfrow = c(1,1))
plot(m,type="l",lwd=
               + 1.5,ylim = mean(m)+20*c(-error[10^3],error[10^3]),ylab="")
lines(m + 1.96 * error,col="blue",lwd=1)
lines(m - 1.96 * error,col="blue",lwd=1)
integrate(h,0,5)
```

```
## 2.295823 with absolute error < 0.00017
```

```r
abline(h = 2.295823,col = "red",lwd = 1.5)
```


Index