STA 6133: Homework 3

Ricardo Cortez & Ben Graf

Due 26 Mar 2021

1.

The beta-binomial distribution with parameters $n \in N$ and $\alpha, \beta > 0$ is a discrete distribution that is often used in Bayesian analysis of count data. Its pmf is given by

$$p(x) = \binom{n}{x} \frac{\Gamma(\alpha+\beta)\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(\alpha)\Gamma(\beta)\Gamma(n+\alpha+\beta)}, \quad for\, x = 0, 1, \ldots, n,$$

(not a good looking one!). As the name suggests, this distribution arises as a beta mixture of binomials: If $X|P \sim bin(n, P)$ and $P \sim beta(\alpha, \beta)$, the the marginal pmf of X is $p(x)$

(a)

Prove the above claim.

$$P(X = x) = \int_0^1 P(X = x, P = p)\, dp$$

$$= \int_0^1 P(X = x|P = p)P(P = p)\, dp \quad \text{by Total Probability}$$

$$= \int_0^1 \binom{n}{x} p^x (1-p)^{n-x} \frac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p)^{\beta-1}\, dp \quad \text{plug in the distributions}$$

$$= \binom{n}{x} \frac{1}{B(\alpha,\beta)} \int_0^1 p^{x+\alpha-1}(1-p)^{n-x+\beta-1}\, dp \quad \text{simplify}$$

$$\text{Use the below beta kernel to simplify even more:}$$
$$beta(\alpha + x, \beta - x + n)$$

$$\text{The earlier equation becomes:}$$
$$= \binom{n}{x} \frac{B(x+\alpha, n-x+\beta)}{B(\alpha,\beta)} \int_0^1 \frac{1}{B(x+\alpha, n-x+\beta)} p^{x+\alpha-1}(1-p)^{n-x+\beta-1}\, dp$$

$$\text{The integral is over a beta distribution, so it equals 1. Simplified and terms rearranged:}$$
$$= \binom{n}{x} \frac{\Gamma(\alpha+\beta)\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(\alpha)\Gamma(\beta)\Gamma(n+\alpha+\beta)} \quad x = 0, 1, .., n$$

(b)

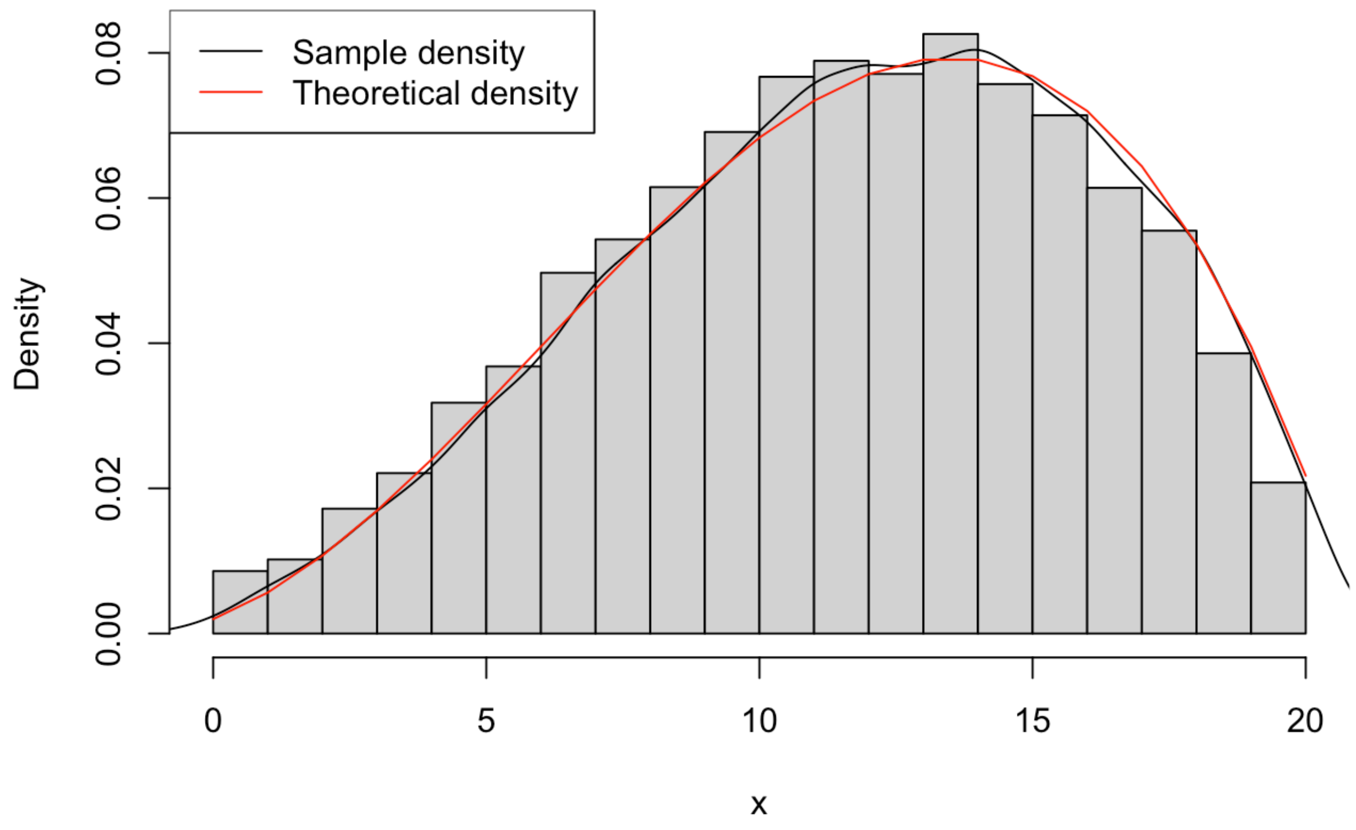
Write an `R` function to simulate from the beta–binomial distribution, having n, α, and β as arguments.

```
# Function to make one draw from a beta-binomial distribution
rbeta_bin <- function(n, alpha, beta) {
  p <- rbeta(1, alpha, beta) #first sample from beta distribution
  x <- sample(c(0,1), size = n, replace = TRUE, prob = c(1-p, p)) #using the draws fr
om the beta draw, draw a binomial sample.
  return(sum(x))
}
```

(c)

For $n = 20, \alpha = 3, \beta = 2$, simulate a random sample of size 10000. Then compare the theoretical pmf, E(X) and var(X) with the respective estimates from the sample.

```
n <- 20
alpha <- 3
beta <- 2
iter <- 10000

#First get Theoretical Values

range <- 0:20
# PMF of beta-binomial with n=20, alpha=3, and beta=2 already known
pmf <- function(x) {
  value <- (x+2) * (x+1) * (21-x) / (2 * 23 * 22 * 21)
  return(value)
}
theo_mean <- n*alpha / (alpha + beta)   # Theoretical mean
theo_var <- n*alpha*beta * (alpha + beta + n) / ((alpha + beta)^2 * (alpha + beta + 1
))   # Theoretical variance


# Make 10000 draws from beta-binomial
samp <- c()
set.seed(1)
for (j in 1:iter) {
  samp[j] <- rbeta_bin(n, alpha, beta)
}

hist(samp, probability = TRUE, main = "", xlab = "x")   # Sample histogram
lines(density(samp))   # Sample density

lines(range, pmf(range), type = "l", col = "red")   # Theoretical PMF
legend("topleft", c("Sample density", "Theoretical density"), lty = 1, col = c("black
", "red"))
```



```r
print(paste0("Theoretical Mean: ", theo_mean))
```

```
## [1] "Theoretical Mean: 12"
```

```r
print(paste0("Sample Mean: ", mean(samp)))  # Sample mean
```

```
## [1] "Sample Mean: 11.9931"
```

```r
print(paste0("Theoretical Variance: ", theo_var))
```

```
## [1] "Theoretical Variance: 20"
```

```r
print(paste0("Sample Variance: ", var(samp)))  # Sample Variance
```

```
## [1] "Sample Variance: 19.7992323132313"
```

It is clear that the simulation is highly accurate. The sample density plot is very close to the pmf, and the sample mean and variance do a great job of approximating the theoretical values.

2.

Directional data are observations of angles (e.g. wind direction). A probability model for such data is the von Mises distribution, a continuous distribution with pdf

$$f_{\boldsymbol{\theta}}(x) = \begin{cases} \frac{1}{2\pi I_0(\theta_1)} e^{\theta_1 cos(x-\theta_2)} & \text{if } 0 \le \text{x} < 2\pi \\ 0 & otherwise \end{cases}, \quad \boldsymbol{\theta} = (\theta_1, \theta_2)$$

where $\theta_1 > 0, 0 \le \theta_2 < 2\pi$ and $I_0(a) = \frac{1}{2\pi}\int_0^{2\pi} e^{acos(x)}dx$ is a Bessel function.

(a)

Assume first that $\theta_2 = 0$. Write down the accept-reject algorithm to sample from this distribution using as proposal density an appropriate uniform distribution. Once the above is done, how can you sample from a von Mises Distribution with $\theta_2 \neq 0$?

First find the max of $f_{\boldsymbol{\theta}}(x) \text{ when } \theta_2 = 0$ by taking the derivative and setting equal to 0:

$$f_{\boldsymbol{\theta}}'(x) = \frac{1}{2\pi I_0(\theta_1)}(-\theta_1 sin(x)e^{\theta_1 cos(x)}) = 0$$
$$= sin(x)e^{\theta_1 cos(x)} = 0$$

This function is zero when $sin(x) = 0$, so $x = 0, \pi, 2\pi$. When $x = 0 \text{ or } 2\pi$:

$$f_{\boldsymbol{\theta}}(0) = \frac{1}{2\pi I_0(\theta_1)} e^{\theta_1 \cos(0)} = \frac{e^{\theta_1}}{2\pi I_0(\theta_1)}$$

Similarly, when $x = \pi$:

$$f_{\boldsymbol{\theta}}(\pi) = \frac{e^{-\theta_1}}{2\pi I_0(\theta_1)}$$

Clearly, the former is a maximum and the latter is a minimum for positive θ_1. Therefore we define:

$$f_{max}(\theta_1) = \frac{e^{\theta_1}}{2\pi I_0(\theta_1)}$$

Using the maximum, we can now specify our algorithm.

1. Generate $Y \sim unif(0, 2\pi), \text{ and } U \sim unif(0, f_{max}(\theta_1))$
2. Set $X = Y$ if $0 < U < f_{\boldsymbol{\theta}}(Y)$
3. Otherwise repeat

In order to sample when $\theta_2 \neq 0$, we can add the value of θ_2 to each kept value of X because this distribution is a location family. If $X \sim f_{\theta_1}(x) \text{ then } Y = X + \theta_2 \sim f_{\boldsymbol{\theta}}(x)$

(b)

Write an R function that implements the algorithm in (a) to sample from $f_{\boldsymbol{\theta}}(x)$. The arguments must be $m \in \mathbb{N}$ and the parameters $\theta_1 > 0,\ 0 \leq \theta_2 < 2\pi$.

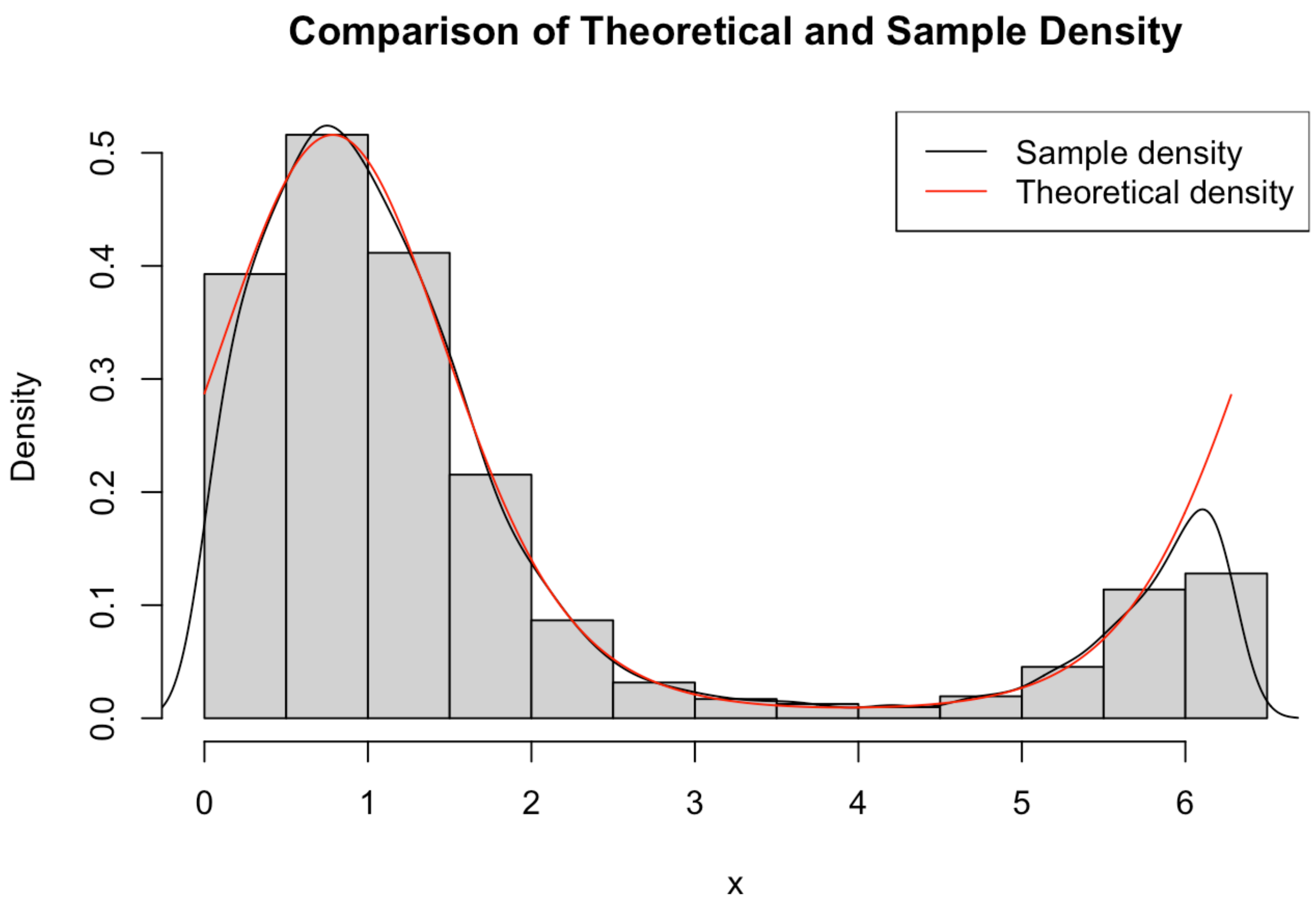
Our function follows:

```
# Calculate von Mises PDF for a given x, theta_1, and theta_2
von_Mises <- function(x, theta_1, theta_2=0) {
  # theta_2 defaults to 0 if no value provided
  bessel <- integrate(function(y) {exp(theta_1 * cos(y)) / (2*pi)}, lower = 0, upper
= 2*pi)
  value <- exp(theta_1 * cos(x-theta_2)) / (2*pi*bessel$value)
  return(value)
}

# Find maximum value of von Mises with theta_2 = 0
fmax <- function(theta_1) {
  bessel <- integrate(function(y) {exp(theta_1 * cos(y)) / (2*pi)}, lower = 0, upper
= 2*pi)
  value <- exp(theta_1) / (2*pi*bessel$value)
  return(value)
}

# Generate n draws from von Mises using accept-reject algorithm
rvon_Mises <- function(n, m, theta_1, theta_2) {
  # n = number of draws
  # m = maximum value the function can take, acts as upper bound for U(0,m)
  output <- c()
  i <- 0
  while(i < n) {   # Keep going until n successful draws
    y <- runif(1, min = 0, max = 2*pi)   # Draw y from unif(0,2*pi)
    u <- runif(1, min = 0, max = m)   # Draw u from unif(0,m)
    vm <- von_Mises(y, theta_1, 0)   # Calculate PDF at y
    if (u < vm) {   # If u < pdf at y, then store that draw in output
      i <- i+1
      output[i] <- y
    }   # Otherwise ignore this draw
  }
  output <- (output + theta_2) %% (2*pi)   # Shift output by theta_2 because location
family
  return(output)
}
```

(c)

Draw a random sample of size 10000 from the von Mises distribution with $\boldsymbol{\theta} = (2, \pi/4)$, and assess the adequacy of the sample.

```
# Make 10000 draws from accept-reject von Mises
theta_1 <- 2
theta_2 <- pi/4
set.seed(1)
samp2 <- rvon_Mises(n = iter, m = fmax(theta_1), theta_1 = theta_1, theta_2 = theta_2
)
hist(samp2, probability = TRUE, main = "Comparison of Theoretical and Sample Density"
, xlab =  "x")   # Sample histogram
lines(density(samp2))   # Sample density
samp_mode <- density(samp2)$x[which(density(samp2)$y == max(density(samp2)$y))]   # S
ample mode, should = theta_2

# Compare with theoretical
range2 <- seq(from = 0, to = 2*pi, by = 0.005)
vM <- von_Mises(x = range2, theta_1 = theta_1, theta_2 = theta_2)
lines(range2, vM, type = "l", col = "red")   # Theoretical PDF
legend("topright", c("Sample density", "Theoretical density"), lty = 1, col = c("blac
k", "red"))
```



```
theo_mode <- range2[which(vM == max(vM))]   # Theoretical mode, should = theta_2
print(paste0("Theoretical Mode: ", theo_mode))
```

```
## [1] "Theoretical Mode: 0.785"
```

```
print(paste0("Sample Mode: ", samp_mode))
```

```
## [1] "Sample Mode: 0.747237850858287"
```

```
print(paste0("Theta_2: ", theta_2))
```

```
## [1] "Theta_2: 0.785398163397448"
```

The histogram matches the theoretical pdf quite well. The plot of the sample density misses a bit at the edges of the graph, but this is due to the periodic nature of the pdf. The sample density plot is trying to smooth its values down to 0 near 0 and 2π because there are no sample results outside these values, nor should there be. The theoretical pdf, on the other hand, is discontinuous at 0 and 2π. Given these caveats, we believe the sample adequately approximates the distribution.
We also compare the theoretical and sample modes, which should equal θ_2. Here again, the sample does fairly well considering the range of the distribution. (We use the mode because, while we can also easily calculate the sample mean and variance, their theoretical counterparts are troublesome to calculate due to the offset period.)

3.

Read the section in the textbook describing the Wishart distribution.

(a)

For $n \geq d + 1 \leq 1$ $(n, d \in N)$ and Σ is a $d \times d$ symmetric positive definite matrix, write an `R` function to simulate from the $W_d(\Sigma, n)$ distribution using the algorithm described in the textbook. The function arguments should be m (= number of required draws), n (= degrees of freedom) and Σ.

The algorithm from the textbook (p.93) is, briefly:
Let $T = (T_{ij})$ be a lower triangular $d \times d$ random matrix with:

- $T_{ij} \overset{iid}{\sim} N(0, 1),\ i > j$
- $T_{ii} \sim \sqrt{\chi^2(n - i + 1)},\ i = 1, \ldots, d$

Then $A = TT^T$ has a $W_d(I_d, n)$ distribution. Obtain Choleski factorization $\Sigma = LL^T$, where L is lower triangular. Then $LAL^T \sim W_d(\Sigma, n)$.

Our `R` implementation follows:

```r
# Generate m draws from Wishart_d(sigma, n) distribution using Johnson algorithm
rwish <- function(m, n, sigma) {
  # m = number of required draws
  # n = degrees of freedom
  # sigma = covariance matrix

  d <- nrow(sigma)

  values <- list()

  for (reps in 1:m) {

    # Calculate values for T matrix in a d^2 length vector
    Tvals <- c()
    for (i in 1:d) {
      for (j in 1:d) {
        index <- (j-1)*d + i
        if (i>j) {
          Tvals[index] <- rnorm(n = 1, mean = 0, sd = 1)
        } else if (i==j) {
          Tvals[index] <- sqrt(rchisq(n = 1, df = n-i+1))
        } else {
          Tvals[index] <- 0
        }
      }
    }

    T <- matrix(data = Tvals, nrow = d, ncol = d, byrow = FALSE)   # Convert vector i
nto dxd matrix

    A <- T %*% t(T)   # TT' is Wishart_d(I_d, n)

    R <- chol(sigma)   # R is upper triangular Choleski decomposition
    L <- t(R)   # L is lower triangular, transpose of R

    values[[reps]] <- L %*% A %*% R   # LAL' is Wishart_d(sigma, n)

  }

  return(values)
}
```

(b)

Get a random sample of size 1000 from the $W_3(\Sigma, 8)$ distribution, where

$$\Sigma = \begin{pmatrix} 1 & -0.5 & 0.5 \\ -0.5 & 1 & -0.5 \\ 0.5 & -0.5 & 1 \end{pmatrix}$$

The code below makes 1000 random draws from the requested distribution:

```
# Make 1000 draws from Wishart
iter3 <- 1000
df <- 8
sigma <- matrix(data = c(1, -0.5, 0.5, -0.5, 1, -0.5, 0.5, -0.5, 1),
                nrow = 3, ncol = 3, byrow = FALSE)
set.seed(1)
samp3 <- rwish(m = iter3, n = df, sigma = sigma)
```

(c)

Look for the formulas of the means and variances of the entries of a random matrix with Wishart distribution (e.g. in a Multivariate Analysis book or Wikipedia). Then for the sample in (b) compare the population means and variances with the respective sample means and variances.

We first compare the theoretical and sample means. From Wikipedia, the theoretical mean of a $W_d(\Sigma, n)$ distribution is $n\Sigma$.

```
# Theoretical mean = df * sigma:
(theo_mean_w <- df*sigma)
```

```
##      [,1] [,2] [,3]
## [1,]    8   -4    4
## [2,]   -4    8   -4
## [3,]    4   -4    8
```

```
# Sample mean:
samp_mean_w <- matrix(data = 0, nrow = 3, ncol = 3)
for (k in 1:iter3) {
  mat <- samp3[[k]]
  samp_mean_w <- samp_mean_w + mat/iter3
}
samp_mean_w
```

```
##           [,1]      [,2]      [,3]
## [1,]  8.145023 -4.064144  4.035614
## [2,] -4.064144  8.064382 -4.013719
## [3,]  4.035614 -4.013719  8.094401
```

Our sample mean does a good job of approximating the theoretical mean. The biggest relative error for any element is 1.8%.

Next we compare the theoretical and sample variances. From Wikipedia, the theoretical variance of a $W_d(\Sigma, n)$ distribution is $n(\sigma_{ij}^2 + \sigma_{ii}\sigma_{jj})$.

```
# Theoretical variance Var_ij = df * (sigma_ij^2 + sigma_ii * sigma_jj):
theo_var_w <- matrix(data = 0, nrow = 3, ncol = 3)
for (i in 1:3) {
  for (j in 1:3) {
    theo_var_w[i,j] <- df * (sigma[i,j]^2 + sigma[i,i] * sigma[j,j])
  }
}
theo_var_w
```

```
##      [,1] [,2] [,3]
## [1,]   16   10   10
## [2,]   10   16   10
## [3,]   10   10   16
```

```
# Sample variance:
samp_var_w <- matrix(data = 0, nrow = 3, ncol = 3)
for (k in 1:iter3) {
  mat <- samp3[[k]]
  samp_var_w <- samp_var_w + (mat - samp_mean_w)^2/(iter3-1)
}
samp_var_w
```

```
##           [,1]      [,2]      [,3]
## [1,] 14.920248  9.158646  9.736066
## [2,]  9.158646 15.875716 10.156006
## [3,]  9.736066 10.156006 15.752757
```

The sample variance is in the right ballpark, but it is not as accurate an estimate of the theoretical variance as we saw for the mean. The largest relative error for any element is 8.4%, not great. Increasing the number of draws can improve this (tested but not shown here).

4.

Let $p = P(X^2 - 3X + 2 > 0)$ where $X \sim N(0, 1)$.

(a)

Find the 'exact' value of p.

We first note that:

$$p = P(X^2 - 3X + 2 > 0) = P((X-2)(X-1) > 0)$$

This product is only greater than 0 if both terms have the same sign. So, either $X - 2 > 0$ and $X - 1 > 0$, which implies $X > 2 \quad$ *or* $\quad X - 2 < 0$ and $X - 1 < 0$, which implies $X < 1$.

Therefore,

$$p = P(X > 2) + P(X < 1) = 1 - P(X < 2) + P(X < 1)$$

This is calculated below:

```
(p <- 1-pnorm(2)+pnorm(1))   # p = P(X^2 - 3X +2 > 0) = P(X>2) + P(X<1)
```

```
## [1] 0.8640949
```

p = 0.8641

(b)

Compute a Monte Carlo approximation of p based on 10^6 simulations, and compute the (likely) upper bound for the estimation error.

We know from Monte Carlo theory that:

$$I(h) = \int_{\mathbb{X}} h(x) f(x) dx = E_f(h(X))$$

can be approximated by:

$$\hat{I}_n(h) = \frac{1}{n} \sum_{i=1}^{n} h(x_i), \quad \text{where } X_1, X_2, \ldots, X_n \overset{iid}{\sim} f(x)$$

In our case, $h(x) = I(X^2 - 3X + 2 > 0)$ and $X \sim N(0, 1)$. The code to carry out this approximation follows:

```
M <- 1e6
set.seed(1)
x <- rnorm(M)
test_x <- ifelse(x^2 -3*x + 2 > 0, 1, 0)
mean(test_x)   # Monte Carlo approximation of p
```

```
## [1] 0.864073
```

```
3*sd(test_x)/sqrt(M)   # (Likely) upper bound for estimation error
```

```
## [1] 0.001028134
```

After one million Monte Carlo draws, we find a highly accurate $\boldsymbol{\hat{p} = 0.86407}$, matching the first 4 digits of p exactly! The (likely) upper bound for the estimation error is **0.001**, also quite good.

(c)

Let δ be a small positive number and $\hat{p}_n$ a Monte Carlo approximation of p based on n simulations. Find the smallest value of n for which the probability that $\hat{p}_n$ will be within δ of the unknown p is about 0.9974. What is n when $\delta = 0.001$?

The problem described above is asking us to solve the following equation for n:

$$P\left(|p - \hat{p}_n \ | < \delta\right) = 0.9974$$

We know $E(\hat{p}_n) = p$ and $Var(\hat{p}_n) = \frac{\hat{\sigma}_n^2}{n}$. Therefore, we can manipulate the equation above to:

$$P\left(\frac{-\delta}{\hat{\sigma}_n/\sqrt{n}} < \frac{|p - \hat{p}_n|}{\hat{\sigma}_n/\sqrt{n}} < \frac{\delta}{\hat{\sigma}_n/\sqrt{n}}\right) = 0.9974$$

The middle term is now a standard normal ($Z \sim N(0,1)$), and symmetric, so we can again rewrite as:

$$2\,P\left(Z < \frac{-\delta}{\hat{\sigma}_n/\sqrt{n}}\right) = 1 - 0.9974$$

$$P\left(Z < \frac{-\delta}{\hat{\sigma}_n/\sqrt{n}}\right) = 0.0013$$

$$\frac{-\delta}{\hat{\sigma}_n/\sqrt{n}} = -3.0115$$

Solving for n:

$$n = \frac{\hat{\sigma}_n^2\,(3.0115)^2}{\delta^2}$$

So this provides the smallest value of n for which the probability that $\hat{p}_n$ will be within δ of the unknown p is about 0.9974. It turns out that $\hat{\sigma}_n^2$ converges as n grows large. We can pull the value from our MC simulation in part (b) above and plug in $\delta = 0.001$ to find n in this scenario:

```
# Calculate n
(var_est <- var(test_x))
```

```
## [1] 0.117451
```

```
delta <- 0.001
(Z <- qnorm(0.0013))
```

```
## [1] -3.011454
```

```
(n <- var_est * (Z / delta)^2)
```

```
## [1] 1065146
```

With $\delta = 0.001$, $\mathbf{n = 1{,}065{,}146}$. We can examine our MC simulation to confirm that the actual n is near the theoretical one, though we have to increase the number of draws, as our simulation in (b) stopped at 1,000,000:

```
# See where the test error crosses the threshold for this set of draws
thresh <- -delta/Z
set.seed(1)
x <- rnorm(2*M)
test_x <- ifelse(x^2 -3*x + 2 > 0, 1, 0)
for (i in 1065001:1066000) {
  test_err <- sd(test_x[1:i])/sqrt(i)
  #print(paste(i,test_err))
  if (test_err < thresh) {break}
}
print(paste0("The test error dropped below the threshold of ",thresh," after ",i," it
erations with a value of ",test_err,"."))
```

```
## [1] "The test error dropped below the threshold of 0.000332065533856369 after 1065
576 iterations with a value of 0.000332065524638254."
```

We see that the actual n is 1,065,576, giving a relative error of 0.04%, very accurate!