HW8

2024-11-06

Q1

Write a function which takes 2 arguments n and k which are positive integers. It should return the nXn matrix:

$$\begin{pmatrix} k & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & k & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & k & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & k & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & k & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & k \end{pmatrix}$$

Call the function defined above for n = 6 and k = 5, and provide the matrix you obtain.

One with a for loop, one without.

```
# nkMatFor <- function(n, k) {
#   mat <- matrix(0, n, n)
#
#   diag(mat) <- k
#
#   for (i in 1:(n-1)) {
#     mat[i, i+1] <- 1
#     mat[i+1, i] <- 1
#   }
#
#   return(mat)
# }
#
# n6k5 <- nkMatFor(n = 6,
#                  k = 5)
# n6k5
```

```
nkMat <- function(n, k) {
  mat <- matrix(0, n, n)

  diag(mat) <- k

  mat[row(mat) == col(mat) + 1] <- 1
  mat[row(mat) == col(mat) - 1] <- 1

  return(mat)
```

```r
}

n6k5 <- nkMat(n = 6,
              k = 5)
n6k5
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    1    0    0    0    0
## [2,]    1    5    1    0    0    0
## [3,]    0    1    5    1    0    0
## [4,]    0    0    1    5    1    0
## [5,]    0    0    0    1    5    1
## [6,]    0    0    0    0    1    5
```

Q2

Consider the continuous function

$$f(x) = \begin{cases} x^2 + 2x + 3 & \text{if } x < 0 \\ x + 3 & \text{if } 0 \leq x < 2 \\ x^2 + 4x - 7 & \text{if } 2 \leq x \end{cases}$$

Write a function tmpFn which takes a single argument xVec. The function should return the vector of values of the function f(x) evaluated at the values xVec. Plot the function f(x) for -3 < x < 3.

```
tmpFn <- function(xVec) {
  result <- numeric(length(xVec))

  result[xVec < 0] <- xVec[xVec < 0]^2 + 2 * xVec[xVec < 0] + 3
  result[xVec >= 0 & xVec < 2] <- xVec[xVec >= 0 & xVec < 2] + 3
  result[xVec >= 2] <- xVec[xVec >= 2]^2 + 4 * xVec[xVec >= 2] - 7

  result
}
```

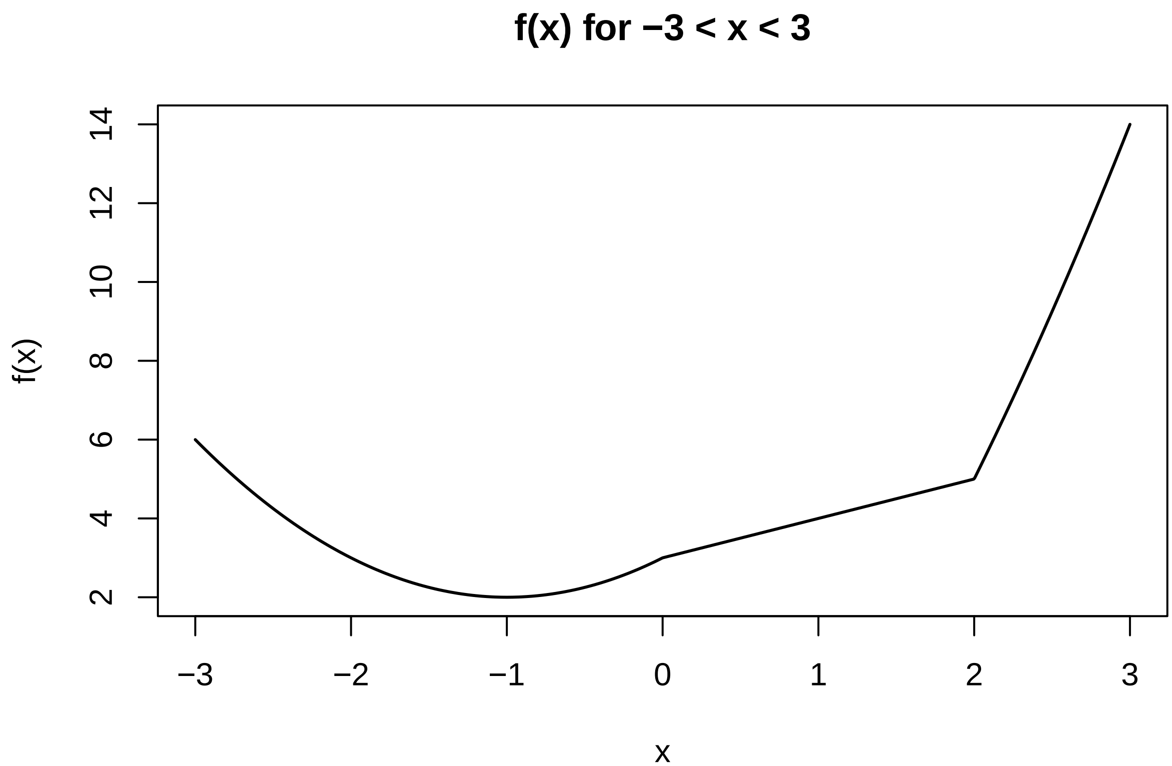
```
xValues <- seq(from = -3,
               to = 3,
               length.out = 1000)

yValues <- tmpFn(xVec = xValues)

length(xValues) == length(yValues)
```

```
## [1] TRUE
```

```
plot(x = xValues, y = yValues,
     type = "l",
     col = "black",
     lwd = 1.5,
     xlab = "x",
     ylab = "f(x)",
     main = "f(x) for -3 < x < 3")
```


f(x) for −3 < x < 3
f(x)
2
4
6
8
10
12
14
−3
−2
−1
0
1
2
3
x

Q3

Greatest common divisor of two integers The greatest common divisor (gcd) of two integers m and n can be calculated using Euclid's Algorithm: Divide m by n. If the remainder is zero, the gcd is n. If not, divide n by the remainder. If the remainder is zero, then the previous remainder is the gcd. If not, continue dividing the remainder into previous remainder until a remainder of zero is obtained. The gcd is the value of the last nonzero remainder. Write a function gcd(m,n) using a while loop to find the gcd of two integers m and n.

```r
gcd <- function(m, n) {
  m <- abs(m)
  n <- abs(n)

  while (n != 0) {
    remainder <- m %% n
    m <- n
    n <- remainder
  }

  m
}
```

```r
# Testing examples
gcd(m = 4*7,
    n = 4*4*4*7)
```

```
## [1] 28
```

```r
gcd(m = 47,
    n = 93)
```

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

Q4

eQTL mapping. (The following problem was suggested by Professor Dan Nettleton.) Write a function order.matrix which takes in a matrix x and returns a matrix containing the row and column indices of the sorted values of x. Test this function on a 4X3 matrix of independent χ_1^2 pseudo-random deviates.

```
order.matrix <- function(mat) {
  sorted.indices <- order(mat)
  result <- arrayInd(sorted.indices, dim(mat))
  colnames(result) <- c("row", "col")
  result
}
```

Had a previous version of this before discovering arrayInd:

```
# order.matrix <- function(mat) {
#   sorted.indices <- order(mat)
#
#   row.indices <- (sorted.indices - 1) %% nrow(mat) + 1
#   col.indices <- (sorted.indices - 1) %/% nrow(mat) + 1
#
#   result <- cbind(row = row.indices,
#                   col = col.indices)
#   result
# }
```

```
set.seed(42)
testMat <- matrix(data = rchisq(4 * 3, df = 1),
                  nrow = 4,
                  ncol = 3)
dim(testMat)
```

```
## [1] 4 3
```

```
testMat
```

```
##          [,1]         [,2]      [,3]
## [1,] 1.521034 4.268462e-01 3.0177684
## [2,] 0.587395 4.222746e-01 0.4038763
## [3,] 3.732687 4.370368e-05 0.5225843
## [4,] 2.963479 1.507955e-04 0.3116673
```

```
testResults <- order.matrix(mat = testMat)
testResults
```

```
##       row col
##  [1,]   3   2
##  [2,]   4   2
##  [3,]   4   3
##  [4,]   2   3
##  [5,]   2   2
```

```
##  [6,]    1    2
##  [7,]    3    3
##  [8,]    2    1
##  [9,]    1    1
## [10,]    4    1
## [11,]    1    3
## [12,]    3    1
```

```r
dim(testResults)
```

```
## [1] 12  2
```

Q5

Polar representation of a number. Let $x \in \mathbb{R}^p$. The polar respresentation of $\mathbf{x} = (x_1, x_2, ..., x_p)$ is given by $(R, \theta_1, \theta_2, ..., \theta_{p-1})$, where:

$$
\begin{aligned}
x_1 &= R\cos\theta_1 \\
x_2 &= R\sin\theta_1\cos\theta_2 \\
x_3 &= R\sin\theta_1\sin\theta_2\cos\theta_3 \\
\dots &= \dots \\
x_{p-1} &= R\prod_{i=1}^{p-2}\sin\theta_i\cos\theta_{p-1} \\
x_p &= R\prod_{i=1}^{p-1}\sin\theta_i,
\end{aligned}
$$

where $0 \leq R < \infty, 0 \leq \theta_1 < 2\pi$ and $0 \leq \theta_i < \pi$ i = 2, 3, . . . , p - 1.

(a)

Write a function polaroid which takes in an arbitrary p-dimensional vector $\mathbf{x}$ and provides its polar representation as a vector, with the first element as R and the remainder being $\theta_1, \theta_2, \ldots, \theta_{p-1}$.

```r
polaroid <- function(x) {
  R <- sqrt(sum(x^2))

  # if radius is length 0, then no angles
  if (R == 0) return(c(R))

  # for an n length input, return length n, but n-1 are thetas
  theta <- numeric(length(x) - 1)

  # atan for arctan
  theta[1] <- atan2(x[2], x[1])
  if (theta[1] < 0) {
    theta[1] <- theta[1] + 2 * pi
  }

  p <- length(x)
  # only need to do function once if input is 2
  # else need to start iterating
  if (p > 2) {
    for (i in 2:(p - 1)) {
      numerator <- sqrt(sum(x[i:p]^2))
      denominator <- sqrt(sum(x[(i-1):p]^2))
      theta[i] <- acos(numerator / denominator)
    }
  }

  dat <- c(R, theta)
  dat
}
```

```r
x <- c(1, 2, 3)
polarRep <- polaroid(x)
polarRep
```

```
## [1] 3.7416574 1.1071487 0.2705498
```

(b)

Write a function normalize which takes in a matrix and returns its normalized form: i.e., the matrix with rows scaled such that the sum of squares of each row is equal to 1.

```r
normalize <- function(x) {
  rowNorm <- sqrt(rowSums(x^2))
  normMat <- x / rowNorm

  normMat[is.nan(normMat)] <- 0
  normMat
}

# here's one using sweep
# normalize <- function(x) {
#   rowNorm <- sqrt(rowSums(x^2))
#   normMat <- sweep(x, 1, row_norms, FUN = "/")
#   normMat[is.nan(normMat)] <- 0
#   normMat
# }
```

```r
set.seed(42)
testMat <- matrix(data = rnorm(12),
                  nrow = 4,
                  ncol = 3)
testMat
```

```
##            [,1]        [,2]       [,3]
## [1,]  1.3709584  0.40426832  2.0184237
## [2,] -0.5646982 -0.10612452 -0.0627141
## [3,]  0.3631284  1.51152200  1.3048697
## [4,]  0.6328626 -0.09465904  2.2866454
```

```r
rowSums(testMat^2)
```

```
## [1] 6.1169942 0.3340795 4.1192458 5.6382226
```

```r
normMatEx <- normalize(x = testMat)
normMatEx
```

```
##            [,1]        [,2]       [,3]
## [1,]  0.5543132  0.16345593  0.8160999
## [2,] -0.9769930 -0.18360767 -0.1085026
## [3,]  0.1789169  0.74474161  0.6429220
## [4,]  0.2665252 -0.03986493  0.9630032
```

```r
rowSums(normMatEx^2)
```

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

(c)

Obtain a 1000X5 matrix **y** of $N(0,1)$ pseudo-random deviates. Use apply and normalize to obtain the normalized values. Call this matrix **z**. We test whether the columns of **z** are uniform on $U(-1,1)$. One may test whether a sample $x \sim U(-1,1)$ using ks.test(x, "punif", min=-1, max=1) where punif represents the cumulative distribution function of the uniform over range $(-1,1)$. Summarize your results.

```r
set.seed(42)

y <- matrix(data = rnorm(1000 * 5),
            nrow = 1000,
            ncol = 5)

z <- apply(X = y,
           MARGIN = 2,
           FUN = function(col) {
             normalize(x = matrix(col, ncol = 1))
             }
           )

ksTests <- apply(X = z,
                 MARGIN = 2,
                 FUN = function(col) {
                   ks.test(col, "punif", min = -1, max = 1)
                   }
                 )

ksTests
```

```
## [[1]]
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  col
## D = 0.515, p-value < 2.2e-16
## alternative hypothesis: two-sided
##
##
## [[2]]
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  col
## D = 0.503, p-value < 2.2e-16
## alternative hypothesis: two-sided
##
##
## [[3]]
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  col
## D = 0.505, p-value < 2.2e-16
## alternative hypothesis: two-sided
##
##
## [[4]]
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  col
## D = 0.52, p-value < 2.2e-16
## alternative hypothesis: two-sided
##
##
## [[5]]
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  col
## D = 0.501, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

The KS tests provide small p-values, which is evidence to support rejecting the null hypothesis that the column vectors are distributed by $U(-1, 1)$, each of the five vectors each column. As we have evidence to suggest the vectors are not distributed by $U(-1, 1)$, then we have evidence that our efforts to normalize the vectors did not effectively transform the data into a uniform distribution over this range (at least when normalizing the sum of squares of each row).

(d)

Obtain polar representations of $\mathbf{y}$ using your function polaroid and test whether $R^2 \sim \chi^2_5$ distribution.

Provide a page of histograms or boxplots of $\theta_1, \theta_2, \theta_3, \theta_4$. Test whether these are from the uniform distributions on their respective ranges, i.e., $[0, 2\pi)$ for θ_1, and $[0, \pi)$ for $\theta_2, \theta_3, \theta_4$.

```r
polarRep <- t(apply(X = y,
                    MARGIN = 1,
                    FUN = polaroid))

rVal <- polarRep[, 1]
thetaVal <- polarRep[, -1]
# thetaVal <- if (ncol(polarRep) > 1) polarRep[, -1] else NA
```

```r
rSq <- rVal^2
csTest <- ks.test(rSq, "pchisq", df = 5)
csTest
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
```

```
## data:  rSq
## D = 0.017123, p-value = 0.9311
## alternative hypothesis: two-sided
```

The large p-value provides evidence in support of not rejecting the null hypothesis that $R^2 \sim \chi^2_5$, such that we have evidence that $R^2 \sim \chi^2_5$.

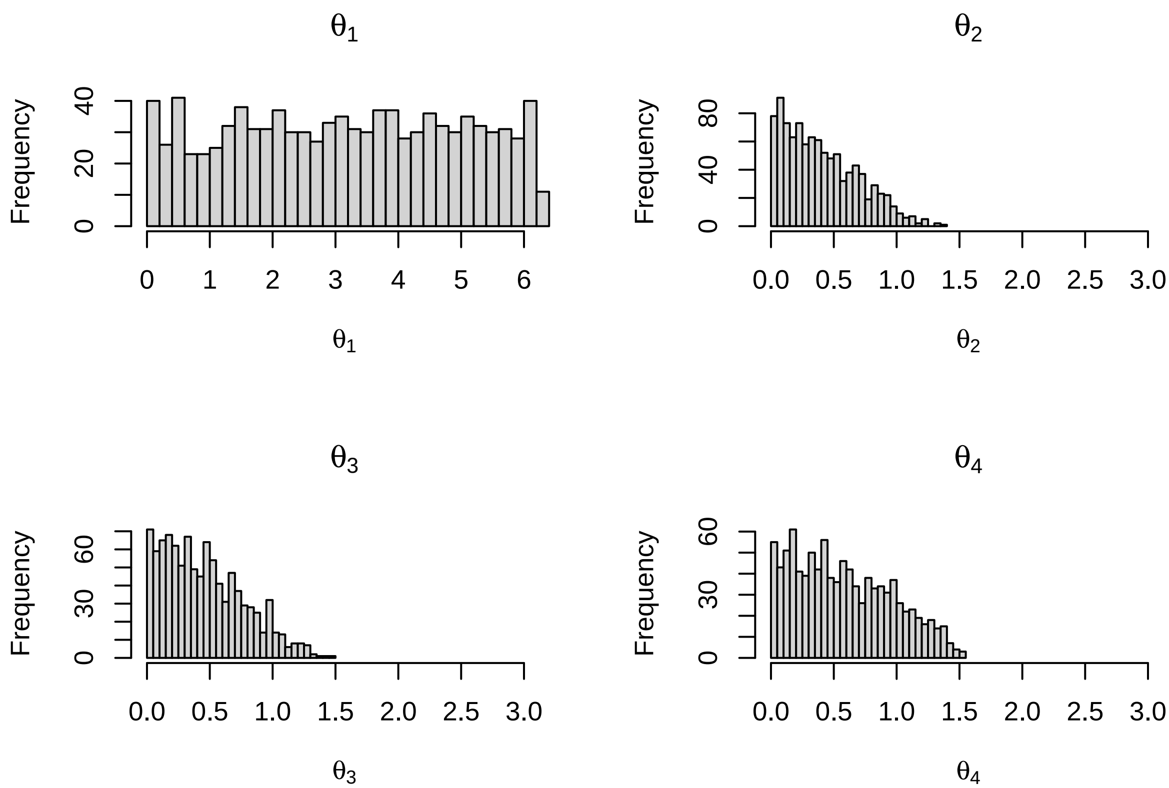
```r
par(mfrow = c(2, 2))

hist(thetaVal[, 1],
     breaks = 30,
     main = expression(theta[1]),
     xlab = expression(theta[1]),
     xlim = c(0, 2 * pi))

hist(thetaVal[, 2],
     breaks = 30,
     main = expression(theta[2]),
     xlab = expression(theta[2]),
     xlim = c(0, pi))

hist(thetaVal[, 3],
     breaks = 30,
     main = expression(theta[3]),
     xlab = expression(theta[3]),
     xlim = c(0, pi))

hist(thetaVal[, 4],
     breaks = 30,
     main = expression(theta[4]),
     xlab = expression(theta[4]),
     xlim = c(0, pi))
```




```r
theta1Test <- ks.test(x = thetaVal[, 1] / (2 * pi),
                      y = "punif",
                      min = 0,
                      max = 1)
theta2Test <- ks.test(x = thetaVal[, 2] / pi,
                      y = "punif",
                      min = 0,
                      max = 1)
theta3Test <- ks.test(x = thetaVal[, 3] / pi,
                      y = "punif",
                      min = 0,
                      max = 1)
theta4Test <- ks.test(x = thetaVal[, 4] / pi,
                      y = "punif",
                      min = 0,
                      max = 1)

list(theta1 = theta1Test,
     theta2 = theta2Test,
     theta3 = theta3Test,
     theta4 = theta4Test)
```

```
## $theta1
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
```

```
## data:  thetaVal[, 1]/(2 * pi)
## D = 0.017778, p-value = 0.9101
## alternative hypothesis: two-sided
##
##
## $theta2
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  thetaVal[, 2]/pi
## D = 0.65547, p-value < 2.2e-16
## alternative hypothesis: two-sided
##
##
## $theta3
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  thetaVal[, 3]/pi
## D = 0.62241, p-value < 2.2e-16
## alternative hypothesis: two-sided
##
##
## $theta4
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  thetaVal[, 4]/pi
## D = 0.54649, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

The above statistical tests may be summarized as follows:

$H_0 : \theta_1 \sim U[0, 2\pi)$

H_0 : individually and respectively $\theta_2, \theta_3, \theta_4 \sim U[0, \pi)$

We do not have evidence in support or rejecting the null hypothesis that $\theta_1 \sim U[0, 2\pi)$ such that we have evidence in support of $\theta_1 \sim U[0, 2\pi)$.

However, we have small p-values in evidence of rejecting the null hypotheses that individually and respectively $\theta_2, \theta_3, \theta_4 \sim U[0, \pi)$, which is evidence in support of $\theta_2, \theta_3,$ and θ_4 not having distributions of $U[0, \pi)$.