Introduction to R

Exercise 01

Solve the following $Ax = b$ linear system using the 'solve' command:

$$A = \begin{bmatrix} 12 & -1 & -5 & 0 \\ -1 & 7 & -2 & -1 \\ -5 & 2 & 10 & 1 \\ 0 & -1 & 1 & 3 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

```r
a <- matrix(c(12,-1, -5, 0, -1, 7, 2, -1, -5, 2, 10, 1, 0, -1, 1, 3), nrow = 4)
b <- matrix(c(1, 2, 3, 4), nrow = 4)
c <- solve(a, b)
c
```

```
##           [,1]
## [1,] 0.1873874
## [2,] 0.4738739
## [3,] 0.1549550
## [4,] 1.4396396
```

Exercise 02

A symmetric matrix $A \in \mathbb{R}^{n \times n}$ (i.e., $A^t = A$) is positive definite if, for all nonzero vectors $x \in \mathbb{R}^n$, the following condition holds:

$$x^t A x > 0$$

A criterion used to verify if a symmetric matrix is positive definite is called the "Sylvester's criterion," which states:

Sylvester's Criterion: $A \in \mathbb{R}^{n \times n}$ is positive definite if and only if A has a positive determinant, and all the submatrices of A listed below have a positive determinant:

- The 1×1 submatrix formed by the first row and the first column;
- The 2×2 submatrix formed by the first two rows and columns;
- …
- The $(n-1) \times (n-1)$ submatrix formed by the first $(n-1)$ rows and columns.

Write an R function that takes a matrix A as an argument and returns TRUE or FALSE depending on whether it is positive definite or not. Test your function on the matrix A from the previous exercise, which is positive definite.

```r
posMatrix <- function(mat) {
  n = nrow(mat)
  m = ncol(mat)

  if (n != m) {
    print("The number of rows and columns should be the same!")
  } else {
    if (!all.equal(t(mat), mat)) {
      print("The given matrix is not symmetric")
    } else {
      positive = T
      for (i in 1:n) {
        b = as.matrix(mat[1:i, 1:i])
        determinant = det(b)
        if (determinant < 0) {
          positive = F
          break
        }
      }
      print(positive)
    }
  }
}

a <- matrix(c(12,-1, -5, 0, -1, 7, 2, -1, -5, 2, 10, 1, 0, -1, 1, 3), nrow = 4)
posMatrix(a)
```

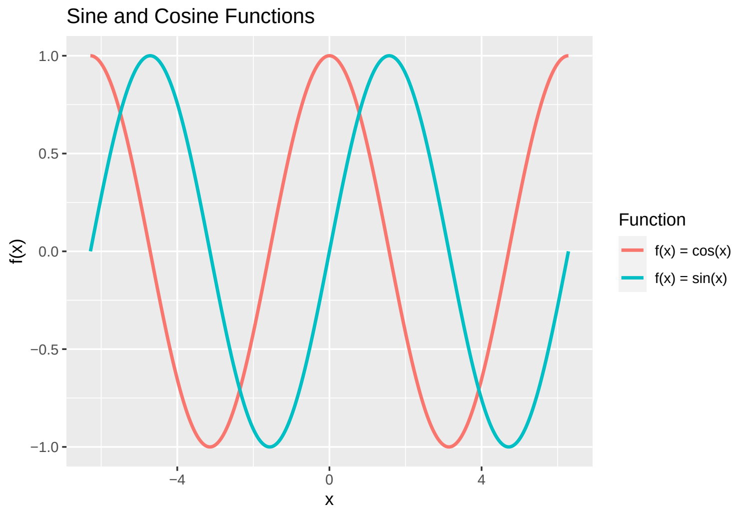
```
## [1] TRUE
```

Exercise 03

Using ggplot2, create plots of the sine and cosine functions for angles ranging from -2π to 2π. Display both functions in the same figure.

```r
library(ggplot2)
library(tidyverse)

df1 <- tibble(column_x = seq(-2*pi, 2*pi, length.out = 1000),
            column_y = sin(column_x),
            fun_type = "f(x) = sin(x)")

df2 <- df1 %>%
bind_rows(tibble(column_x = seq(-2*pi, 2*pi, length.out = 1000),
            column_y = cos(column_x),
            fun_type = "f(x) = cos(x)"))


ggplot(df2) +
  aes(x = column_x, y = column_y, color = fun_type) +
  geom_line(size = 1) +
  labs(x = "x",
  y = "f(x)",
```

```
  title = "Sine and Cosine Functions",
  color = "Function")
```



Exercise 04

We know that solutions to linear systems obtained on a computer, as in Exercise 01, are approximated, since real numbers are represented in an approximate manner on a computer. When the matrix of a linear system is ill-conditioned, the solutions of the linear systems obtained by the computer can have greatly amplified errors. We can quantify the ill-conditioning of an invertible matrix A using the condition number, which is defined as:

$$\kappa(A) = \|A\|\|A^{-1}\|$$

where $\|\cdot\|$ represents the matrix norm, and A^{-1} is the inverse of A. The larger the value of $\kappa(A)$, the more ill-conditioned the matrix is. In other words, solutions of linear systems calculated by computers generally present more numerical errors. You can calculate the condition number in R using the kappa function.

It is possible to improve the conditioning of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ by increasing its diagonal elements. This process is sometimes used in machine learning, for instance, in the regularization process of Ridge Regression. In this exercise, we aim to create a graph illustrating the condition number of A as we increase the elements of its diagonal.

To do this, you can use a symmetric matrix A generated randomly through the following process:

```
set.seed(1)
X = matrix(runif(100,-1,1),nrow=10)
A = t(X) %*% X
```

In this process, we first generate a matrix $X \in \mathbb{R}^{10\times 10}$ with random values within the range $[-1, 1]$. Then, we create $A := X^T X$, which results in a symmetric matrix. The set.seed function initializes the seed of the random number generator, ensuring consistent results.

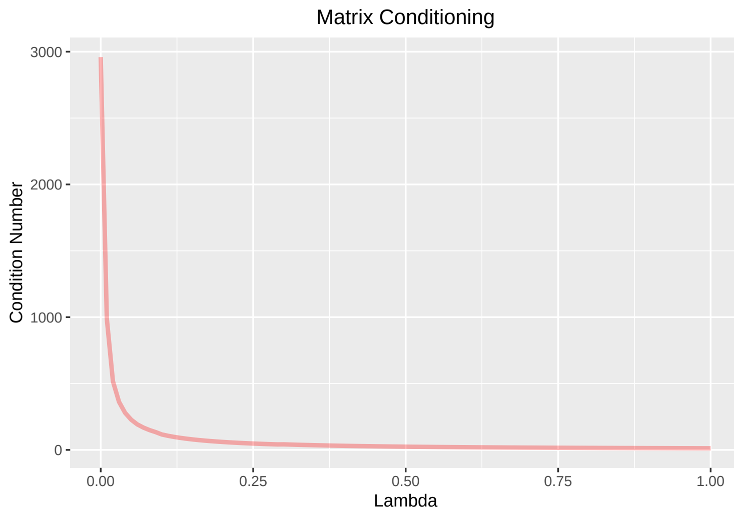
Tips: you can use the **diag** function to generate an identity matrix. Afterward, calculate the condition number of A with diagonal elements increased by λ as follows:

$$A' = A + \lambda I$$

where $\lambda \geq 0$, and $I \in \mathbb{R}^{d \times d}$ is the identity matrix.

```
library(tidyverse)

cond <- function(mat) {
  I = diag(nrow(mat))
  .data <- tibble(lmbd = numeric(), k = numeric())

  for (i in seq(0,1,0.01)) {
    B = A + i*I
    K = kappa(B)
    .data <- add_row(.data, lmbd = i, k = K)
  }

  print(.data)
  ggplot(.data) +
    aes(x = lmbd, y = k) +
    geom_line(alpha = 0.3, size = 1.3, color = "red") +
    labs(x = "Lambda",
    y = "Condition Number",
    title = "Matrix Conditioning") +
    theme(plot.title = element_text(hjust = 0.5))
}

set.seed(1)
X = matrix(runif(100,-1,1),nrow=10)
A = t(X) %*% X
cond(A)
```

```
## # A tibble: 101 x 2
##     lmbd     k
##    <dbl> <dbl>
##  1  0    2959.
##  2  0.01  989.
##  3  0.02  518.
##  4  0.03  362.
##  5  0.04  279.
##  6  0.05  228.
##  7  0.06  192.
##  8  0.07  168.
##  9  0.08  149.
## 10  0.09  134.
## # i 91 more rows
```



Matrix Conditioning
3000
2000
1000
0
Condition Number
0.00
0.25
0.50
0.75
1.00
Lambda