MSMS - 106

Ananda Biswas

Practical 03

✎ Consider any non-singular matrix A of order p and find the following results.

(a) $A + A'$ (b) $A - A'$ (c) $A \cdot A'$

(d) $det(A)$ (e) $adj(A)$ (f) A^{-1}

➔ *Creating a matrix*

```
values <- c(1, 1, 0, -2, 3, 0, 1, 4, 3)
A <- matrix(data = values, nrow = 3, ncol = 3, byrow = TRUE)
A

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

➔ *Function for transpose of a matrix*

```
transpose <- function(x){
  temp <- x
  for (i in 1:nrow(x)) {
    for (j in 1:ncol(x)) {
      x[i, j] <- temp[j, i]
    }
  }
  return(x)
}
```

➔ *Function for sum of two matrices*

```
matrix_addition <- function(x, y){
  if(nrow(x) != nrow(y) || ncol(x) != ncol(y)) stop("Dimensions don't match.")

  temp <- matrix(NA, nrow = nrow(x), ncol = ncol(x))

  for (i in 1:nrow(x)) {
    for (j in 1:ncol(x)) {
      temp[i, j] <- x[i, j] + y[i, j]
    }
  }
  return(temp)
}
```

➜ $\mathbf{A} + \mathbf{A}'$

```r
matrix_addition(A, transpose(A))
```

```
##      [,1] [,2] [,3]
## [1,]    2   -1    1
## [2,]   -1    6    4
## [3,]    1    4    6
```

➜ ***Function for subtraction of two matrices***

```r
matrix_subtraction <- function(x, y){
  if(any(dim(x) == dim(y)) == FALSE) stop("Dimensions don't match.")

  temp <- matrix(NA, nrow = nrow(x), ncol = ncol(x))

  for (i in 1:nrow(x)) {
    for (j in 1:ncol(x)) {
      temp[i, j] <- x[i, j] - y[i, j]
    }
  }
  return(temp)
}
```

➜ $\mathbf{A} - \mathbf{A}'$

```r
matrix_subtraction(A, transpose(A))
```

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

➜ ***Function for multiplication of two matrices***

```r
matrix_multiplication <- function(x, y){
  if(ncol(x) != nrow(y)) stop("Dimensions are not compatible for matrix multiplication.")

  temp <- matrix(0, nrow = nrow(x), ncol = ncol(y))

  for (i in 1:nrow(x)) {
    for (j in 1:ncol(y)) {
      for (k in 1:ncol(x)) {
        temp[i, j] <- temp[i, j] + x[i, k] * y[k, j]
      }
    }
  }
  return(temp)
}
```

➜ $\mathbf{A} * \mathbf{A}'$

```
matrix_multiplication(A, transpose(A))
```

```
##      [,1] [,2] [,3]
## [1,]    2    1    5
## [2,]    1   13   10
## [3,]    5   10   26
```

➜ *Calculating determinant*

Here we convert the given matrix to an upper triangular matrix by Gaussian Elimination. Let r be the number of row-swaps in the process. We calculate the determinant of the upper triangular matrix; which is nothing but the product of its diagonal elements. Let it be d. Then determinant of the given matrix is $(-1)^r \times d$.

```
det_function <- function(x){
  if(nrow(x) != ncol(x)) stop("Input must be a real square matrix.")

  n <- nrow(x)

  swap_count <- 0; pivot_count <- 0

  j <- 1; i <- 1

  while (j <= ncol(x)) {
    work_code <- 0; swap_code <- 0

    while (i <= nrow(x)) {
      if(round(x[i, j], digits = 5) != 0){
        if((pivot_count + 1) == i && i != n){

          pivot_count <- pivot_count + 1

          a1 <- as.matrix((x[(i+1):n, j] / x[i, j]))

          a2 <- t(as.matrix(x[i, ]))

          x[(i+1):n, ] <- x[(i+1):n, ] - a1 %*% a2

          work_code <- 1

          break

        } else if((pivot_count + 1) != i) {

          x[c(pivot_count+1, i),] <- x[c(i, pivot_count+1),]

          swap_count <- swap_count + 1

          j <- j - 1
```

```
          swap_code <- 1

          break
        }
      }
      i <- i + 1
    }

    j <- j + 1

    if(work_code == 1) {
      i <- j
    } else {
      i <- 1
    }

    if(swap_code == 1) i <- pivot_count + 1
  }

  product <- 1

  for (i in 1:n) {
    product <- product * x[i, i]
  }

  return((-1)^swap_count * product)
}
```

- $\det(\mathbf{A})$

```
det_function(A)

## [1] 15
```

- ***Function for cofactor of any element of a matrix***

```
cofactor <- function(x, i, j){
  x_sub <- x[-i, -j]
  d <- det_function(x_sub)
  return((-1)^(i+j) * d)
}
```

➔ ***Function for adjoint of a matrix***

```
adjoint <- function(x){
  cofactor_matrix <- matrix(NA, nrow = nrow(x), ncol = ncol(x))

  for (i in 1:nrow(x)) {
    for (j in 1:ncol(x)) {
      cofactor_matrix[i, j] <- cofactor(x, i, j)
    }
  }

  return(transpose(cofactor_matrix))
}
```

➔ **adj(A)**

```
adjoint(A)

##      [,1] [,2] [,3]
## [1,]    9   -3    0
## [2,]    6    3    0
## [3,]  -11   -3    5
```

➔ ***Function for inverse of a matrix***

```
inverse <- function(x){
  if(det_function(x) == 0) stop("Matrix must be invertible.")

  return(adjoint(x) / det_function(x))
}
```

➔ $\mathbf{A}^{-1}$

```
inverse(A)

##            [,1] [,2]      [,3]
## [1,]  0.6000000 -0.2 0.0000000
## [2,]  0.4000000  0.2 0.0000000
## [3,] -0.7333333 -0.2 0.3333333
```