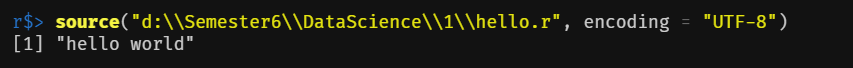
***1. Write a program that prints ‘Hello World’ to the screen.***

print("hello world")

***OUTPUT:***

******

***2. Write a program that asks the user for a number n and prints the sum of the numbers 1 to n***

n <- as.integer(readline("Enter the limit of numbers that you want to sum : "))

sum <- 0

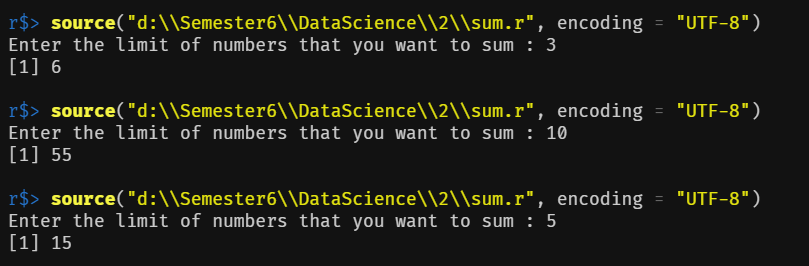
for (i in 1:n) {

sum <- sum + i

}

print(sum)

***OUTPUT:***

******

***3. Write a program that prints a multiplication table for numbers up to 12.***

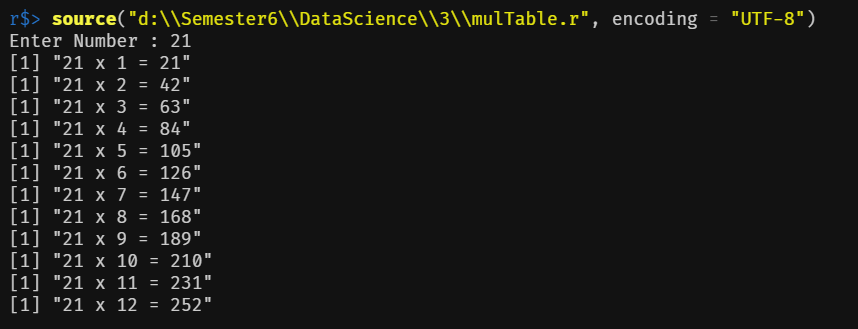
num <- as.integer(readline("Enter Number : "))

for (i in 1:12) {

print(paste(num, "x", i, "=", num \* i))

}

***OUTPUT:***



***4. Write a function that returns the largest element in a list.***

lmax <- function(a) {

maxim <- a[1]

for (i in a) {

if (i > maxim) {

maxim <- i

}

}

return(maxim)

}

size <- as.numeric(readline("Enter the size of the list : "))

x <- list()

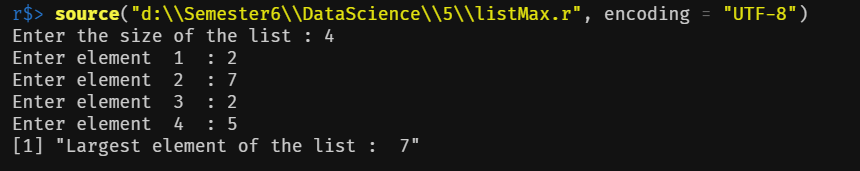
for (i in 1:size) {

x[i] <- as.numeric(readline(paste("Enter element ", i, " : ")))

}

print(paste("Largest element of the list : ", lmax(x)))

***OUTPUT:***

******

***5. Write a function that computes the total of a list.***

size <- as.numeric(readline("Enter the size of the vector : "))

x <- c()

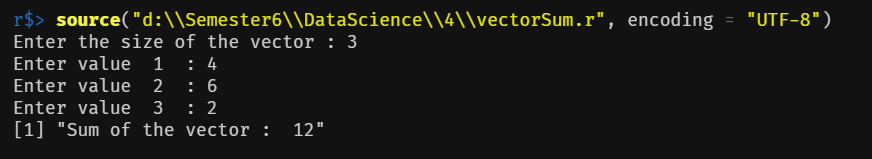
for (i in 1:size) {

x[i] <- as.numeric(readline(paste("Enter element ", i, " : ")))

}

print(paste("Sum of the vector : ", sum(x)))

***OUTPUT:***

******

***6. Write a function that tests whether a string is a palindrome.***

palindrome <- function(string, i, j) {

if (i == j) {

return(TRUE)

}

if (substring(string, i, i) == substring(string, j, j)) {

return(palindrome(string, i + 1, j - 1))

}

else {

return(FALSE)

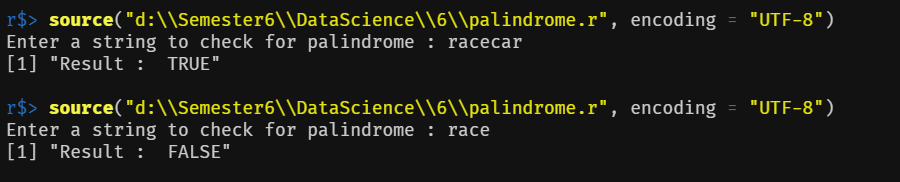
}

}

string <- readline("Enter a string to check for palindrome : ")

print(paste("Result : ", palindrome(string, 1, nchar(string))))

***OUTPUT:***

******

***7. Implement linear search.***

linear\_search <- function(vec) {

flag <- FALSE

num <- as.integer(readline("Enter element to search : "))

j <- 1

for (i in vec) {

if (i == num) {

print(paste("Element found at position", j))

flag <- TRUE

}

j <- j + 1

}

if (flag == FALSE) {

print("Element not found")

}

}

n <- as.integer(readline("Enter number of elements:"))

x <- c()

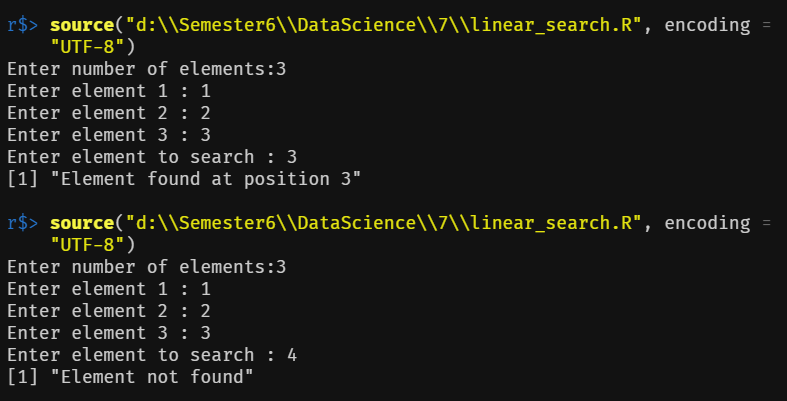
for (i in 1:n) {

x[i] <- as.integer(readline(paste("Enter element", i, ": ")))

}

linear\_search(x)

***OUTPUT:***

******

***8. Implement binary search.***

binary\_search <- function(vec, s, e, num) {

if (e >= s) {

mid <- (e + s) %/% 2

if (vec[mid] == num) {

return(mid)

} else if (vec[mid] > num) {

return(binary\_search(vec, s, (mid - 1), num))

}

else {

return(binary\_search(vec, (mid + 1), e, num))

}

}

else {

return(-1)

}

}

n <- as.integer(readline("Enter number of elements :"))

x <- c()

for (i in 1:n) {

x[i] <- as.integer(readline(paste("Enter element", i, ": ")))

}

num <- as.integer(readline("Enter element to search : "))

sort(x)

if (binary\_search(x, 1, length(x), num) == -1) {

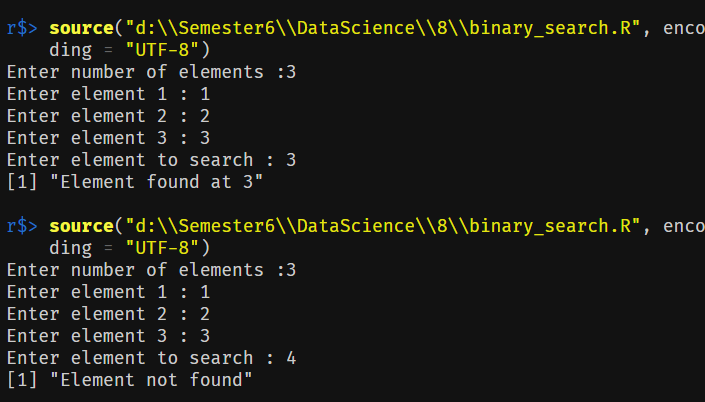
print("Element not found")

} else {

print(paste("Element found at", binary\_search(x, 1, length(x), num)))

}

***OUTPUT:***

******

***9. Implement matrices addition,***

***subtraction, Multiplication and other operations.***

m <- matrix(1:6, nrow = 2, ncol = 3)

dimnames(m) <- list(c("X", "Y"), c("A", "B", "C"))

print(paste("Rows : ", nrow(m), " Columns : ", ncol(m)))

print(paste("Element [1][2] : ", m[1, 2]))

print("Full Matrix : ")

print(m)

print("Row 1 : ")

print(m[1, ])

print("Column 2 : ")

print(m[, 2])

m[1, 2] <- 9

print(paste("Element [1][2] after change : ", m[1, 2]))

m <- cbind(m, c(7:8))

colnames(m)[4] <- "D"

print("After adding column : ")

print(m)

m <- rbind(m, c(0))

rownames(m)[3] <- "Z"

print("After adding row : ")

print(m)

names <- list(c("X", "Y", "Z"), c("A", "B", "C", "D"))

n <- matrix(1:12, nrow = 3, ncol = 4, dimnames = names, byrow = TRUE)

print("Row Major Matrix : ")

print(n)

print("Addition : ")

print(m + n)

print("Subtraction : ")

print(m - n)

print("Multiplication : ")

print(m \* n)

print("Division : ")

print(m / n)

str <- "NOT A MATRIX"

print("Check whether an object is a matrix : ")

print(is.matrix(str))

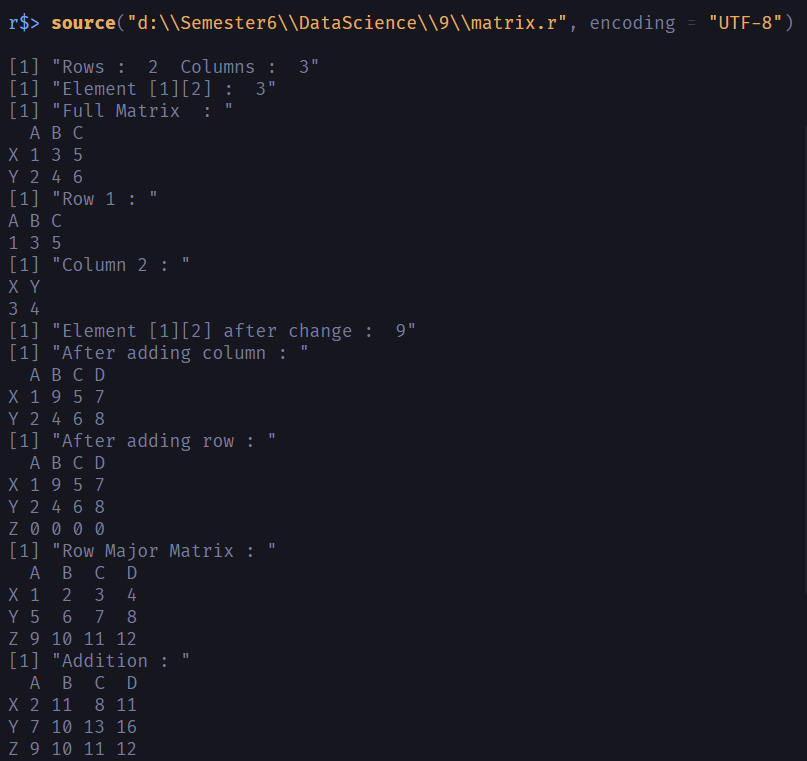
print(is.matrix(m))

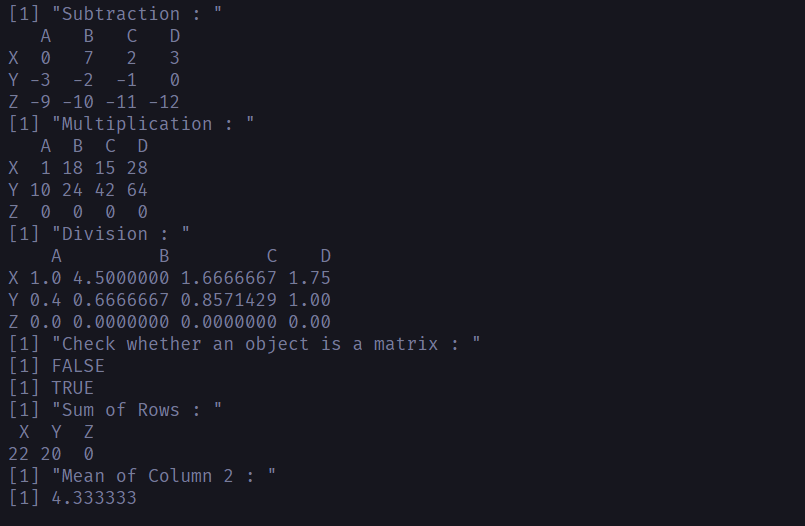
print("Sum of Rows : ")

print(rowSums(m))

print("Mean of Column 2 : ")

print(mean(m[, 2]))

***OUTPUT:***

******

***10. Fifteen students were enrolled in a course. There ages were:  
20 20 20 20 20 21 21 21 22 22 22 22 23 23 23  
i. Find the median age of all students.  
ii. Find the median age of all students under 22 years.  
iii. Find the mean age of all students.  
iv. Find the modal age for all students v.  
v. Two more students enter the class. The age of both students is 23. What is now mean,  
mode and median?***

getmode <- function(v) {

t <- table(v)

return(names(which(t == max(t))))

}

students <- c(20, 20, 20, 20, 20, 21, 21, 21, 22, 22, 22, 22, 23, 23, 23)

print("Age of all the students in class : ")

print(students)

print("Median age of students under 22 : ")

print(median(students[students < 22]))

print("Median age of all students : ")

print(median(students))

print("Mean age of all students : ")

print(mean(students))

print("Modal age of all students : ")

print(getmode(students))

students <- c(students, c(23, 23))

print("Two new students, both of age 23 join the class : ")

print(students)

print("Median age of all students : ")

print(median(students))

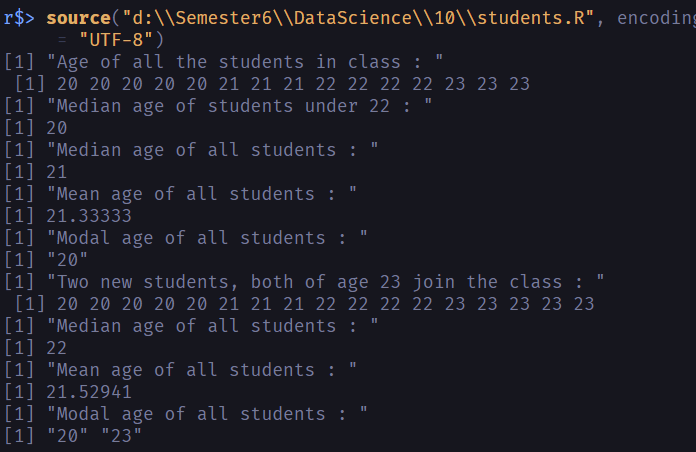
print("Mean age of all students : ")

print(mean(students))

print("Modal age of all students : ")

print(getmode(students))

***OUTPUT:***

***11. Following table gives a frequency distribution of systolic blood pressure. Compute all the  
measures of dispersion:  
Midpoint n  
95.5 5  
105.5 8  
115.5 22  
125.5 27  
135.5 17  
145.5 9  
155.5 5  
165.5 5  
175.5 2***

dispersion <- function() {

bp <- c(rep(95.5, 5), rep(105.5, 8), rep(115.5, 22), rep(125.5, 27), rep(135.5, 17), rep(145.5, 9), rep(155.5, 5), rep(165.5, 5), rep(175.5, 2))

print(mean(bp))

print(range(bp))

print(var(bp))

print(sd(bp))

print(unique(bp))

print(unique(mean(bp) - bp))

print(quantile(bp))

print(summary(bp))

out <- boxplot(bp)$out

print(out)

a <- which(bp %in% out)

print(a)

print(mode(bp))

}

dispersion()

mode <- function(vec) {

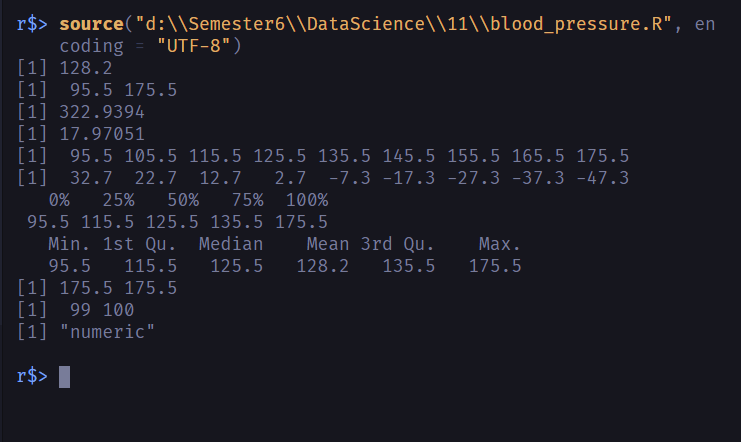
t <- table(vec)

n <- names(which(t == max(t)))

return(n)

}

***OUTPUT:***

******

***12. Randomly generate 30 numbers in the range of 1 to 40 and do the following  
i) Generate box plot  
ii) Identify outliers, if any  
iii) Display 5 point summary of data distribution***

#U=c(floor(rnorm(30)%%40+1))

#U=sample(1:40,size = 30)

U=c(1,2,2,3,2,1,3,2,4,2,5,4,100,-100)

print(U)

boxplot(U)

outlier <- boxplot.stats(U)

print(outlier)

print(summary(U))

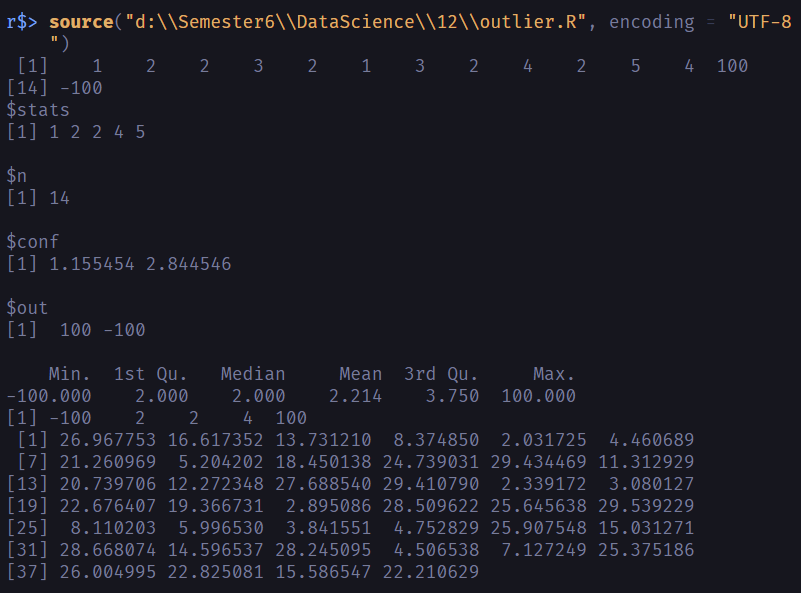
print(fivenum(U))

V=runif(40,min=1,max=30)

print(V)

floor(V)

***OUTPUT:***

******

