**Please, upload your answers in Canvas: assignment PC LAB – April 16**

Name: Vu The Doan

Student nr: 12918687

PC3b

**1.** Write an R-***function*** defined as

**geometric <- function(r,n)**

that calculates the sum of the following geometric series *without using loops*:

Example:

**> geometric(1.03,10)**

**[1] 12.8078**

**2.** The mathematician Euler proved the following:

a. Write an R-functions

Euler <- function(n)

that returns the sum of the series *without using loops* (only vectors!).

b. Next, extend the script to display the following output:

pi^2/6 is 1.6449

The sum of the first 50 terms is 1.6251

The user should be able to specify the value of n in the script.

**3**. Generate a *3 x 5* matrix using

set.seed(321)

x<-matrix(sample(1:30,3\*5,replace=T),3,5); x

Write a R-function

maxmat <- function(mat)

that returns a list with vectors containing the max. of each column, max. of each row and the max. of the whole matrix. For instance,

**results <- maxmat(x)**

should contain:

* the maximum of each column (results$maxcol)
* the maximum of each row (**result$maxrow**)
* the maximum of the whole matrix (**results$maxtot**).

Code the function using for loops (so don’t use the build-in function max).

Sample output:

**> results <- maxmat(x)**

**> results**

**$maxcol**

**[1] 29 25 30 17 28**

**$maxrow**

**[1] 22 26 30**

**$maxtot**

**[1] 30**

**4.** [using a subfunction] The distance between two points and is given by

The area of a triangle is

where *a*, *b* and *c* are the lengths of the sides of the triangle and

Write an R-function

areatri <- function(x1,y1,x2,y2,x3,y3)

that calculates the area of a triangle. This R-function should also contain the ***sub***function (so included inside the areatri function):

dist <- function(x1,y1,x2,y2)

Hint:

**a = dist(x1,y1,x2,y2)**

To determine b and c, you should again use the subfunction dist with the other coordinates.

What is the area of a triangle defined by the coordinates , (, ?

Area: 6

**5**. We consider the Taylor approximation in the point 0 for:

(1)

The main idea is to calculate the Taylor approximation for a given value of and .

a. Write a function

Approx <- function(x,M)

that returns the right-hand side of formula (1) for a vector x and scalar M.

For instance,

> Approx(-1:1,4)

[1] -3.352381 0.000000 3.352381

b. The script should show the function and its Taylor approximation based on terms in a graph. Use ggplot() to create a graph that shows the function as a solid blue line and its Taylor approximation as red asterisks for .

Look at the end of the document for a graph when equals 5.

c. Next, the script should show the function values and its approximation in 6 decimals for the first 5 values of , i.e. for

Sample output for :

x log((1+x)/(1-x)) Taylor

--------------------------------------

-0.99 -5.293305 -3.476570

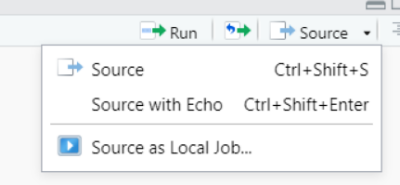
-0.96 -3.891820 -3.204569

-0.93 -3.316780 -2.962075

-0.90 -2.944439 -2.744946

-0.87 -2.666159 -2.549612

Note that by default, the source code is shown when running a program. To get a clean table, you should select “Source” (so *without* Echo).



d. Finally, the script should show the summation symbolically. For instance,

Note: the program must work for arbitrary values of *M*.

Sample output for :

log((1+x)/(1-x))~2(x^1/1+x^3/3+x^5/5+x^7/7+x^9/9)



getwd()

setwd("D:/UvA/Year 1/Block 5/Programming and Numerical Analysis/Week 3")

rm(list = ls())

#1

geometric <- function(r,n) {

print((1-r^(n+1))/(1-r))

}

geometric(1.03,10)

#2

Euler <- function(n){

i <- 1

x <- NULL

while(i<=n) {

x <- c(x, 1/(i^2))

i <- i+1

}

return(sum(x))

}

n <- as.numeric(readline("Enter value for n: "))

cat("pi^2/6 is 1.6449\n")

cat((sprintf("The sum of the first %s terms is: %.4f\n",n, Euler(n))))

#3

set.seed(321)

x<-matrix(sample(1:30,3\*5,replace=T),3,5); x

maxtotal <- function(x) {

max <- -Inf

for (i in 1:nrow(x)) {

for (j in 1:ncol(x)) {

if (x[i,j]>max)

max <- x[i,j]

}

}

return(max)

}

maxtotal(x)

maxvec <- function(n) {

max <- -Inf

for (i in 1:length(n)) {

if (n[i]>max)

max <- n[i]

}

return(max)

}

maxvec(x[,1])

maxmat <- function(x) {

maxcol <- c(maxvec(x[,1]), maxvec(x[,2]), maxvec(x[,3]), maxvec(x[,4]), maxvec(x[,5]))

maxrow <- c(maxvec(x[1,]), maxvec(x[2,]), maxvec(x[3,]))

maxtot <- maxtotal(x)

results <- list(maxcol, maxrow, maxtot)

names(results) <- c("maxcol", "maxrow", "maxtot")

return(results)

}

maxmat(x)

#4

dist <- function(x1,y1,x2,y2) {

return(sqrt((x1-x2)^2+(y1-y2)^2))

}

areatri <- function(x1,y1,x2,y2,x3,y3) {

a <- dist(x1,y1,x2,y2)

b <- dist(x2,y2,x3,y3)

c <- dist(x3,y3,x1,y1)

s <- (a+b+c)/2

area <- sqrt(s\*(s-a)\*(s-b)\*(s-c))

return(area)

}

areatri(0,0,4,0,4,3)

#5

x<- -1:1; M <- 4

approx <- function(x, M) {

a <- NULL

for(i in 1:M) {

a <- c(a, (x^(2\*i-1))/(2\*i-1))

}

a

sum(a)

return(2\*sum(a))

}

approx(-1, 4); approx(1,4)

Approx <- function(x, M) {

re <- NULL

for (i in 1:length(x)) {

re <- c(re, approx(x[i],M))

}

return(re)

}

app <- Approx(seq(-0.99, 0.99, 0.03),4)