STAT 2450 Assignment 2

Tasneem Hoque (B00841761)

20-05-2022

Notes

Personal information

Please fill up the header information (banner id, first and last name, and date of completion).

This entire assignment (A2) is marked on a maximum of 20 points.

[Note that regardless of this total. each assignment (i.e. A1 to A7) will have the same weight in the final mark.]

This assignment has 5 questions each worth 4 points.

Question 1 (*4 points*)

Use a “for” loop to evaluate the following sum, when x=.2.

$$y = 1 - x + x^2 - x^3 \ldots - x^{29} + x^{30}$$

Hint: You will need to initialize the variable y, increment the value of y in the body of your for loop, then print the value of y when the loop is finished.

Note that you can use a loop index (i) in the calculation of the new term to add to y. The generic term to add takes the form:

$$(-1)^i x^i$$

and i must start with the value 0 and end with the value 30.

Your code:

```
y = 0;
for (i in 0:30){
  y = y + ((-1)^i*(0.2)^i)
}

y
```

```
## [1] 0.8333333
```

Question 2 (*4 points*)

You will write a function called listdivisors that take a single argument variable, n and print a list of the divisors of n.

When called with an integer argument, listdivisors(n) should return the vector v of divisors of the number n.

For example, the integer numbers that are divisors of the integer n=6 (i.e. that divide 6 with no remainder), are 1, 2, 3 and 6. Therefore, listdivisors(6) should print

```
1 2 3 6
```

To write the body of your function, you will create a variable v which you can initialize as an empty vector. Then use a for loop to scan all integers i smaller or equal to n. If the loop index divides n (use the modulo operator to check this), then add this value to the vector v. Otherwise dont do anything. Finally, when the loop is terminated, return the vector v.

Apply this function to list the divisors of 40

```
listdivisors <- function(x){
  v = c()
  for (i in 1:x){
    if (x%%i == 0)
      v = append(v,i)
  }
  return (v)
}


listdivisors(40)
```

```
## [1]  1  2  4  5  8 10 20 40
```

Question 3 (*4 points*)

We use the function sample to generate a random vector “x” of 100 integers sampled with replacement in the interval [15,132].

```
set.seed(250)
x=sample(15:132,100,replace=T)
```

We want to use a ‘for’ loop and the function ‘ifelse’ to count the number of even integers and the number of odd integers in x.

Initialize a counter variable named neven.

Use a for loop to iterate over the elements of a vector x.

Increment the value of neven by 1 if the current value of the loop index is even (or else increment the value by 0). Do not use if(), but use the ifelse function for this. The body of your loop should only contain one line of code that updates neven.

When the loop is finished, print the number of even elements and the number of odd elements of x.

Use the syntax : paste(“message”,variablename) to print the message:

The number of even elements is:

followed by the value of the counter variable neven.

After this, compute the number of odd elements (use a substraction) in a variable called nodd, and print it. Use the message:

The number of odd elements is:

followed by the value of the variable nodd.

```
# initialize the value of the variable neven
neven = 0
# use a for loop to loop over the elements of vector x
#for...{
  # in the body of the for loop, use the function ifelse to add 1 to neven when i is even, and
  # add zero to neven if i is odd. The body of the loop should contain a single line of code.
for (i in x){
  ifelse(i%%2==0,neven<-neven+1,neven<-neven-0)
}

#}
# print the first message and the value of neven
paste("The number of even elements is:", neven)
```

```
## [1] "The number of even elements is: 53"
```

```
# use the length of vector x and neven to calculate the number (nodd) of odd elements
nodd= length(x)-neven
# print the second message and the value of nodd
paste("The number of odd elements is:", nodd)
```

```
## [1] "The number of odd elements is: 47"
```

Question 4 (*4 points*)

We generate a random 5x5 matrix whose entries are the numbers 1,2, … 25, but in random positions, using the following code:

```
set.seed(133) #set the seed for the random number generator
x=matrix(sample(1:25), byrow=T,ncol=5) #
x
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   25    6    4   17   13
## [2,]   22    9    2   21    8
## [3,]   23   18   24    1   12
## [4,]   11   19   15    7    3
## [5,]   16   20   14    5   10
```

Complete the following code

```
# we can store the dimension of the matrix in a vector dx of length 2:
dx=dim(x)

# use dx to print the number of rows of the matrix x:
# hint: this is the first element of dx
dx[1]
```

```
## [1] 5
```

```
# use dx to print the number of columns of the matrix x:
# hint: this is the second element of dx
dx[2]
```

```
## [1] 5
```

Then, use two nested for loops (one, with index i, for the rows, one, with index j, for the columns) to loop over each position i,j in the matrix x, and, if the associated element x[i,j] is even, replace this matrix element by its square root, otherwise, if it is odd, replace it by its square.

Note that you can use the elseif function to do this, something like:

x[i,j]=ifelse(…)

For the loop bounds (i.e. the number of rows and the number of columns), use the two elements of vector dx.

```
for (i in 1:dx[1]) {
  for (j in 1:dx[2]) {
      x[i, j] <- ifelse(x[i, j]%%2 == 0, sqrt(x[i,j]), x[i,j]^2)
  }
}

x
```

```
##            [,1]       [,2]       [,3] [,4]       [,5]
## [1,] 625.000000   2.449490   2.000000  289 169.000000
## [2,]   4.690416  81.000000   1.414214  441   2.828427
## [3,] 529.000000   4.242641   4.898979    1   3.464102
## [4,] 121.000000 361.000000 225.000000   49   9.000000
## [5,]   4.000000   4.472136   3.741657   25   3.162278
```

Question 5 (*4 points*)

The goal of this problem is to plot the graph of the probability density function of the normal distribution of the normal distribution with mean 1 and standard deviation 0.5.

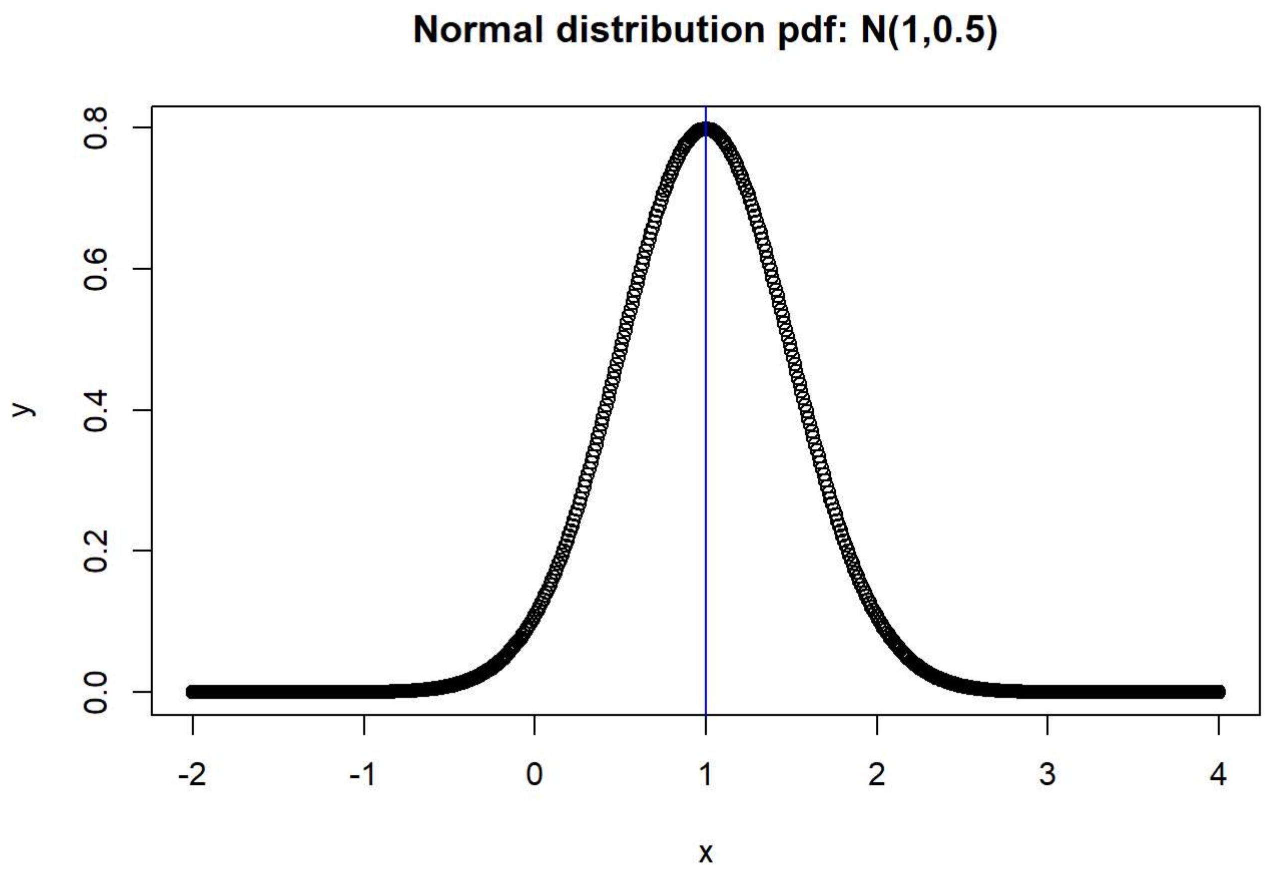
To write this code, use the seq function to define a vector x of regularly spaced values between -2 and 4 by steps of 0.01.

Then apply the function dnorm to x with appropriate arguments to calculate a vector y.

Call the plot function on x and y. Use a line type and the title ‘Normal distribution pdf: N(1,0.5)’

Finally use abline to overlay a vertical blue line that goes through x=1.

```
x = seq(-2, 4, by = 0.01)
y = dnorm(x , mean=1, sd= 0.5 )
plot(x,y, main = "Normal distribution pdf: N(1,0.5)" )
abline(v=1, col='blue')
```


Normal distribution pdf: N(1,0.5)
y
x