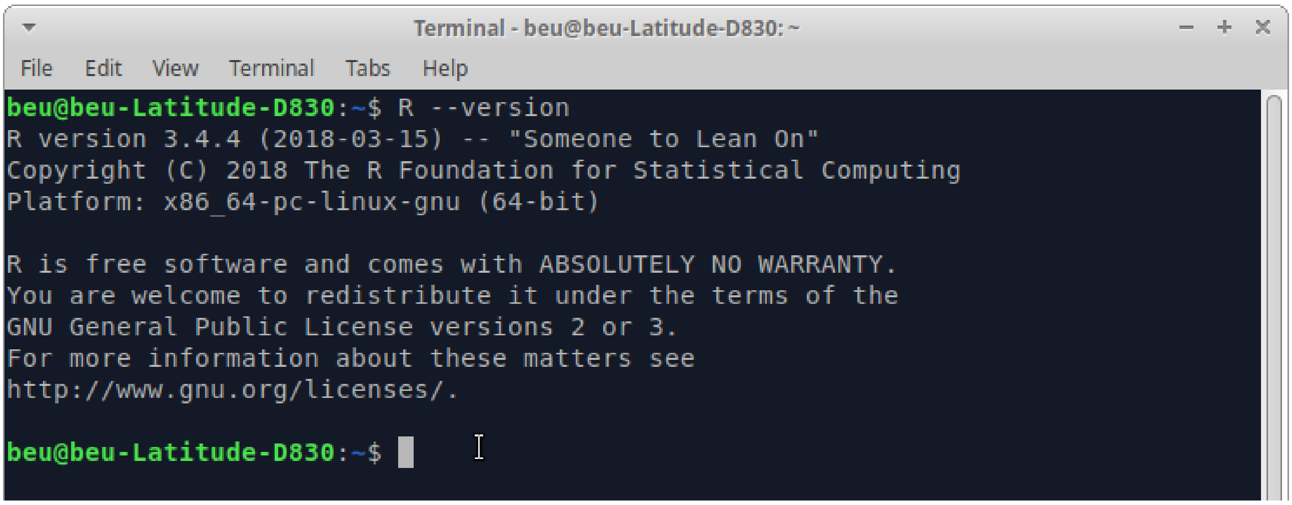
LABWORK - 1

Installation of R:



Terminal - beu@beu-Latitude-D830: ~
File Edit View Terminal Tabs Help

```
beu@beu-Latitude-D830:~$ R --version
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under the terms of the
GNU General Public License versions 2 or 3.
For more information about these matters see
http://www.gnu.org/licenses/.

beu@beu-Latitude-D830:~$
```

Done installation !

Interacting with R by doing some arithmetic operations

FUNCTION	R EXPRESSION	WORKING EXAMPLE
Addition	+	> 27 + 43 [1] 70
Subtraction	-	> 10 - 1 [1] 9
Multiplication	*	> 10 * 10 [1] 100
Division	/	> 90 / 9 [1] 10 > 3 / 5 [1] 0.6
Modulus	%%	> 20 %% 3 [1] 2
Exponent	^	> 4 ^ 2 [1] 16 > 2 ^ -3 [1] 0.125

Square Root	sqrt ()	> sqrt (100) [1] 10
Natural logarithm	log ()	> log (1) [1] 0
Absolute value	abs ()	> abs (-1) [1] 1 > abs (11) [1] 11
Equality	==	> x = 7 > x == 7 [1] TRUE > x == 3 [1] FALSE
Inequality	!=	> y = -3 > y != -3 [1] FALSE > y != 4 [1] TRUE
Less than	<	> 5 >2 [1] TRUE > 5 > 5 [1] FALSE > 5 > 10 [1] FALSE
Greater than	>	> 5 < 2 [1] FALSE > 5 < 5 [1] FALSE > 5 < 10 [1] TRUE
Less than or equal to	<=	> a = 10 > b = 5 > a <= b [1] FALSE
Greater than or equal to	>=	> a = 1 > b = 0 > a >= b [1] TRUE

Introduction to R Data Types

VECTORS:

1. **Creating a vector of numbers for 1 to 5**

```
> x <- 1:5                                # Creating a vector using : operator
> x
[1] 1 2 3 4 5
```

2. **Creating a vector using c ( ) function**

```
> x = c(99, 100, 101)                     # Creating a vector using c ( ) operator
> x
[1]  99 100 101
```

3. **Some arithmetic operations on vectors**

```
> myVec <- c (1, 3, 5, 2, 4)              # Creating a vector using c ( ) function
> myVec + 10                              # Addition operation
[1] 11 13 15 12 14
> myVec * 10                              # Multiplication operation
[1] 10 30 50 20 40
> myVec / 5                               # Division operation
[1] 0.2 0.6 1.0 0.4 0.8
> myVec %% 5                              # Modulo division
[1] 1 3 0 2 4
> myVec ^ 2                               #  Exponentiation
[1]  1  9 25  4 16
> sqrt (myVec)                            # Squaring all elements in myVec
[1] 1.000000 1.732051 2.236068 1.414214 2.000000
> log (myVec)                             # Logarthmic function applied on all the
                                            elements in myVec
[1] 0.0000000 1.0986123 1.6094379 0.6931472 1.3862944
> abs (myVec)                             # Absolute value function
[1] 1 3 5 2 4
> a <- c (1, 2, 3, 4, 5)
> b <- c (5, 4, 3, 2, 1)
> a + b
[1] 6 6 6 6 6
> a * b                                   # Elements-wise operations
[1] 5 8 9 8 5
> a / b
[1] 0.2 0.5 1.0 2.0 5.0
> a %% b
[1] 1 2 0 0 0
> a ^ b
[1]  1 16 27 16  5
```

4. Working with indexes

```
> myVec1 <- 1:10                        # Creating a vector using : operator
> myVec1
 [1]  1  2  3  4  5  6  7  8  9 10
> myVec1 [1]                            # Accessing the 1st element in the vector
                                          myVec1
[1] 1
> myVec1 [5]                            # Accessing the 5th element in the vector
                                          myVec1
[1] 5
> myVec1 [-1]                           # Accessing all the elements except the 1st
                                          element in the vector myVec1
[1]  2  3  4  5  6  7  8  9 10
> myVec1 [-7]                           # Accessing all the elements except the 1st
                                          element in the vector myVec1
[1]  1  2  3  4  5  6  8  9 10
> myVec1 [3] <- 11                      # Modifying the 3rd elements
> myVec1
 [1]  1  2 11  4  5  6  7  8  9 10
> myVec1[2:4] <- c(100, 200,300)        # Modifying the 2nd , 3rd  and 4th elements
> myVec1
 [1]   1 100 200 300   5   6   7   8   9  10
> myVec1[3 : 8]                         # Slicing the vector
[1] 11  4  5  6  7  8
names(myVec1)<- c("A","B","C","D","E","F","G","H","I","J")
                                        # Giving names to the elements of the vector
> myVec1
A    B   C    D    E    F    G    H    I     J
1    2    3    4    5    6    7    8    9    10
```

5. Modifying the data type of the vector

```
> x <- 1:10                             # Creating a vector using : operator
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x[11] <- "A"                          # Adding a character element to a vector
> x
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "A"
                                        # Coercion of data to a character vector
> str(x)
 chr [1:11] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "A"
```

```
> is.character (x)                # Checking whether the data type of x is
                                    character
[1] TRUE
> is.integer (x)                  # Checking whether the data type of x is
                                    integer
[1] FALSE
> x <- x[-11]
> x <- as.integer (x)             # Coercing the vector back to integer
> is.integer (x)                  # Checking the data type of x
[1] TRUE
> is.character (x)                # Checking the data type
[1] FALSE
```

6. Deleting a vector

```
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x <- NULL                       # Assigning a NULL to x
> x
NULL
> x[3]
NULL
```

7. Subsetting vectors

```
> favBooks <- c ("Discrete_Mathematics", "Linear_Algebra", "Graph_Theory",
"Mathematical_Statistics", "Abstract_Algebra", "Algorithm_Design",
"Computer_Networking")            # Creating a vector favBooks
> favBooks[2]                     # Accessing the 2nd element
[1] "Linear_Algebra"
> favBooks[6]                     # Accessing the 6th element
[1] "Algorithm_Design"
> favBooks[c (7,4,3)]             # Accessing the 7th , 4th and 3rd elements
[1] "Computer_Networking"     "Mathematical_Statistics"
[3] "Graph_Theory"
> favBooks[c (1,2)]               # Accessing the 1st and 2nd elements
[1] "Discrete_Mathematics" "Linear_Algebra"
> myShelf <- favBooks[c(1,2,3,7,6,5,4,3,2,1,2,3,4,6,5,4,3,6,2,7,1)]
                                  # Repeating indices to create an object with
> myShelf                           more elements than the original one
 [1] "Discrete_Mathematics"    "Linear_Algebra"
 [3] "Graph_Theory"            "Computer_Networking"
 [5] "Algorithm_Design"       "Abstract_Algebra"
 [7] "Mathematical_Statistics" "Graph_Theory"
 [9] "Linear_Algebra"         "Discrete_Mathematics"
[11] "Linear_Algebra"         "Graph_Theory"
[13] "Mathematical_Statistics" "Algorithm_Design"
[15] "Abstract_Algebra"       "Mathematical_Statistics"
[17] "Graph_Theory"           "Algorithm_Design"
[19] "Linear_Algebra"         "Computer_Networking"
[21] "Discrete_Mathematics"
# Conditional Subsetting
# TRUE will select the element with the same index and FALSE will omitt it
> favBooks[c (TRUE, FALSE, FALSE, TRUE, FALSE, TRUE, TRUE)]
[1] "Discrete_Mathematics"    "Mathematical_Statistics"
[3] "Algorithm_Design"        "Computer_Networking"
> a <- c(20,31,42,53,64,75)
> a >= 50                           # Will return logicals with TRUE for the
                                      indices that meet the condition
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
> a[a >= 50]                        # Selecting the variables which are above 50
[1] 53 64 75
```

8. Loops

```
> v <- c(11,22,33,44,55)                # Creating a vector
> for (i in v) {                        # Iterating through the vector
        print (i)
  }
[1] 11
[1] 22
[1] 33
[1] 44
[1] 55
> sum <- 0                              # Initializing sum variable
> for (i in v) {
        sum <- sum + i
  }
> sum                                   # Printing the sum of all elements in
                                          vector v
[1] 165                                 # 11 + 22 + 33 + 44 + 55 = 165
```

MATRICES:

1. **Creating a Matrix from a vector**

```
> myMatrix <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)           # Creating a vector
> dim (myMatrix) <- c (3,3)                           # Turning the vector into a matrix of
> myMatrix                                               dimension 3 X 3
      [,1]  [,2]  [,3]
[1,]   1     4     7
[2,]   2     5     8
[3,]   3     6     9
```

2. **Creating a Matrix using matrix ( ) function**

```
> matrix(1:12, nrow = 4, ncol = 3 )                  # Creating a matrix of dim 4X3
      [,1]  [,2]  [,3]
[1,]   1     5     9
[2,]   2     6    10
[3,]   3     7    11
[4,]   4     8    12
> matrix(1:12, nrow = 4, ncol = 3, byrow = TRUE )     # Filling matrix row-wise
      [,1]  [,2]  [,3]
[1,]   1     2     3
[2,]   4     5     6
[3,]   7     8     9
[4,]  10    11    12
> myMatrix <- matrix (1:12, nrow = 4, ncol = 3, dimnames = list (c ("X","Y","Z",
"W"), c("A","B","C")))              # Naming the  rows and columns of the matrix
> myMatrix
       A     B     C
 X     1     5     9
 Y     2     6    10
 Z     3     7    11
 W     4     8    12
> colnames(myMatrix)                # Accessing the column names
[1] "A" "B" "C"
> rownames(myMatrix)                # Accessing the row names
[1] "X" "Y" "Z" "W"
```

```
> colnames(myMatrix) <- c("C1","C2", "C3")   # Modifying the column names
> rownames(myMatrix) <- c("R1","R2", "R3", "R4")  # Modifying the row names
> myMatrix
      C1   C2   C3
 R1    1    5    9
 R2    2    6   10
 R3    3    7   11
 R4    4    8   12
```

3. Creating a Matrix using cbind ( ) and rbind ( ) functions

```
> cbind (c (1,2,3), c (4,5,6), c (7,8,9))
      [,1]  [,2]  [,3]
[1,]    1     4     7
[2,]    2     5     8
[3,]    3     6     9
> rbind (c (1,2,3), c (7,8,9))
      [,1]  [,2]  [,3]
[1,]    1     2     3
[2,]    7     8     9
```

4. Working with indices

```
> myMatrix
      C1   C2   C3
 R1    1    5    9
 R2    2    6   10
 R3    3    7   11
 R4    4    8   12
> myMatrix [c (1,3), c (2,3)]            # Selecting 1st & 3rd rows and 2nd&3rd columns
         C2      C3
  R1      5       9
  R3      7      11
> myMatrix [c (3,1), ]                     # Selecting 3rd & 1st rows and entire columns
      C1   C2   C3
 R3    3    7   11
 R1    1    5    9
```

```
> myMatrix [, c (3,1)]          # Selecting all of the rows and 3rd & 1st columns
         C3     C1
   R1     9      1
   R2    10      2
   R3    11      3
   R4    12      4
> myMatrix [-2,-1]                 # Selecting  all of the rows and columns
                                     except 2nd row and 1st column
         C2     C3
   R1     5      9
   R3     7     11
   R4     8     12
# Indexing a matrix with single vector
# Here the matrix acts like a vector formed by stacking columns of the matrix
> myMatrix [3:9]
[1] 3 4 5 6 7 8 9
> myMatrix [c (9,7,2)]
[1] 9 7 2
```

5. Matrix Operators in R

OPERATOR	R EXPRESSION	WORKING EXAMPLE
Transposition	t	> myMatrix C1 C2 C3 R1 1 5 9 R2 2 6 10 R3 3 7 11 R4 4 8 12 > t (myMatrix) R1 R2 R3 R4 C1 1 2 3 4 C2 5 6 7 8 C3 9 10 11 12

Inversion	solve ( )	> A <- matrix (c (5, 3, 4, 1, -1, 0, 0, 2, -1), nrow = 3) > A [,1] [,2] [,3] [1,] 5 1 0 [2,] 3 -1 2 [3,] 4 0 -1 > Ainv <- solve(A) > Ainv [,1] [,2] [,3] [1,] 0.0625 0.0625 0.125 [2,] 0.6875 -0.3125 -0.625 [3,] 0.2500 0.2500 -0.500
Matrix Multiplication	**%*%**	> A [,1] [,2] [,3] [1,] 5 1 0 [2,] 3 -1 2 [3,] 4 0 -1 > Ainv [,1] [,2] [,3] [1,] 0.0625 0.0625 0.125 [2,] 0.6875 -0.3125 -0.625 [3,] 0.2500 0.2500 -0.500 > AProduct <- Ainv %*% A > Aproduct [,1] [,2] [,3] [1,] 1 0 0 [2,] 0 1 0 [3,] 0 0 1

ARRAYS:

1. **Creating an array using array ( ) function**

```
> myArray <- array (1:20, dim = c(2,5,2))      # Creating an array with 2 rows
> myArray                                         5 columns and 2 tables
, , 1
        [,1]  [,2]  [,3]  [,4]  [,5]
 [1,]    1     3     5     7     9
 [2,]    2     4     6     8    10
, , 2
        [,1]  [,2]  [,3]  [,4]  [,5]
 [1,]   11    13    15    17    19
 [2,]   12    14    16    18    20

# This array has three dimensions
```

2. **Creating an array using array ( ) function**

```
> vector1 <- c(1, 3, 5)
> vector2 <- c (11, 22, 33, 44, 55, 66, 77, 88, 99)
> vArray <- array (c (vector1, vector2), dim = c (3, 3, 3))     # Creating a 3-D array
> vArray
, , 1
        [,1]  [,2]  [,3]
 [1,]    1    11    44
 [2,]    3    22    55
 [3,]    5    33    66
, , 2
        [,1]  [,2]  [,3]
 [1,]   77     1    11
 [2,]   88     3    22
 [3,]   99     5    33
, , 3
        [,1]  [,2]  [,3]
 [1,]   44    77     1
 [2,]   55    88     3
 [3,]   66    99     5
```

3. Accessing the array elements

```
> print (vArray [1, 3, 1])          # Printing the element in 1st row and 3rd
[1] 44                                 column of the 1st matrix

> print (vArray [3, , 1])           # Printing the 3rd row of the 1st matrix of
[1]  5 33 66                          the vArray

> print (vArray [, , 3])            # Printing the 3rd matrix
          [,1]   [,2]   [,3]
  [1,]     44     77     1
  [2,]     55     88     3
  [3,]     66     99     5
```