Basics of R
Practice Problems and Solutions

Viraj
dataTasteMaker

April 13, 2016

Contents

Introduction

This document has some questions and solutions which should help understand the basiscs of R

The document will be kept updating with my new learnings, along with separate modules on my github repository

Before starting we need to ensure, we clean up the environment, and set the working directory. This helps is avoiding unncessary hiccups while executing the script(s), command(s).

ASSIGNMENTS, DATA TYPES AND DATA STRUCTURES

1. Assign first five prime numbers to an object named *'prime'*.

As it is just 5 numbers, we would simply type in the numbers which we are aware of

```
prime <- c(2,3,5,7,11)
```

However, as an alternate approach, we can also use the **Primes** function from the ***numbers*** package to get the list of prime numbers.

```r
library(numbers)
```

```
##
## Attaching package: 'numbers'

## The following object is masked from 'package:psych':
##
##     omega
```

```r
prime <- Primes(20)[1:5]
prime
```

```
## [1]  2  3  5  7 11
```

---

2. Coerce object *'prime'* to character data type and assign the output to *'character'* and then check its class.

```r
cat("The class of the \'prime\' object is -- ", class(prime))
```

```
## The class of the 'prime' object is --  numeric
```

```r
character <- as.character(prime)
class(character)
```

```
## [1] "character"
```

---

3. Check if elements in *'prime'* are $>$ 5 and save the output in *'logical'*.

```
prime
```

```
## [1]  2  3  5  7 11
```

```
logical <- (prime > 5)
logical
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE
```

---

4. Create an object *'inflation'* containing 'RBI predicts the inflation rate to reduce in the coming quarter'. Then replace *reduce* with *moderate*.

Here we can use the base function called `gsub`, or alternatively we can use the string functions from the ***stringr*** package. Both the approaches are shown below

```
### Using the gsub function

inflation <- 'RBI predicts the inflation rate to reduce in the coming quarter'
inflation <- gsub('reduce','moderate',inflation)
inflation
```

```
## [1] "RBI predicts the inflation rate to moderate in the coming quarter"
```

Now using *stringr* package

```
inflation <- 'RBI predicts the inflation rate to reduce in the coming quarter'
library(stringr)
inflation <- str_replace(inflation, "reduce", "moderate")
inflation
```

```
## [1] "RBI predicts the inflation rate to moderate in the coming quarter"
```

The `str_replace()` function replaces the first instance of the search string, if we want to replace all the occurances, we can use `str_replace_all()`

---

5. Vector named *'vowels'* containing all the vowels in English language

```
vowels <- c("a","e","i","o","u")
vowels
```

```
## [1] "a" "e" "i" "o" "u"
```

and just incase, you know the position of Vowels in our alphabets

```
vowels <-  LETTERS[c(1,5,9,15,21)]
vowels
```

```
## [1] "A" "E" "I" "O" "U"
```

---

6. Create a vector *'numbers'* of numbers 1 and 2, 10 times each.

```
# all 1's and 2's together
numbers <- rep(1:2,each = 10)
numbers
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
```

```
# 1's and 2's, in alternate pattern
numbers <- rep(1:2, times = 10)
numbers
```

```
##  [1] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
```

```
# If we know the length post the repeatition we can specify the lenght
numbers <- rep(1:2, length = 20)
numbers
```

```
##  [1] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
```

---

7. Matrix of dimension 7 x 8, elements being numbers 1-7.

Again there are multiple ways this can be created, a standard one is shown below, and alternate method using ***mapply***

```
mat1 <- matrix(1:7, nrow = 7, ncol = 8)
mat1
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    1    1    1    1    1    1    1
## [2,]    2    2    2    2    2    2    2    2
## [3,]    3    3    3    3    3    3    3    3
## [4,]    4    4    4    4    4    4    4    4
## [5,]    5    5    5    5    5    5    5    5
## [6,]    6    6    6    6    6    6    6    6
## [7,]    7    7    7    7    7    7    7    7
```

```
mat2 = mapply(rep, 1:8,7)
mat2
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    2    3    4    5    6    7    8
## [2,]    1    2    3    4    5    6    7    8
## [3,]    1    2    3    4    5    6    7    8
## [4,]    1    2    3    4    5    6    7    8
## [5,]    1    2    3    4    5    6    7    8
## [6,]    1    2    3    4    5    6    7    8
## [7,]    1    2    3    4    5    6    7    8
```

---

8. Create an array with dimensions 2 x 3 x 4 and elements 1-15.

```
arr <- array(1:15, dim = c(2,3,4))

dim(arr)
```

```
## [1] 2 3 4
```

```
arr
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]    7    9   11
## [2,]    8   10   12
##
## , , 3
##
##      [,1] [,2] [,3]
## [1,]   13   15    2
## [2,]   14    1    3
##
## , , 4
##
##      [,1] [,2] [,3]
## [1,]    4    6    8
## [2,]    5    7    9
```

To subset such an array we can use the below syntax

Here we are extracting the element(s) located in - 1st row - 3rd Column for all the 4 levels

```
arr[1,3,]
```

```
## [1]  5 11  2  8
```

LOADING DATA INTO R

10. Create a data frame of 4 rows consisting of four vectors

- a. Customer.Id
- b. Names
- c. Age
- d. Default.prob

```
# c <- as.integer((runif(1000)[1:4])*100)
Customer.Id <- sample(1:10,4,replace = F) # some random numbers
Names <- c("Jon","Robb","Brann","Arya")
Age <- sample(18:99,4,replace = F)  # random age values from from 18:99
Default.prob <- rnorm(4)  # random probabilties

df <- data.frame(Customer.Id,Names,Age,Default.prob)
df
```

```
##   Customer.Id Names Age Default.prob
## 1           2   Jon  32    1.0783346
## 2           6  Robb  48   -0.6153747
## 3           1 Brann  27   -0.1147759
## 4           3  Arya  24    1.3479764
```

---

11. Download data from file "LungCapData.csv" using read.table (.) argument and save it as data1

```
data1 <- read.table("../datasets/LungCapData.csv",header = T, sep = ",")
#View(data1)
head(data1)
```

```
##   LungCap Age Height Smoke Gender Caesarean
## 1   6.475   6   62.1    no   male        no
## 2  10.125  18   74.7   yes female        no
## 3   9.550  16   69.7    no female       yes
## 4  11.125  14   71.0    no   male        no
## 5   4.800   5   56.9    no   male        no
## 6   6.225  11   58.7    no female        no
```

---

12. Download data using read.clipboard (.) and save it as data2

Load in the *psych* library
Also, before using the `read.clipboard()` function, we need to go and explicitly copy the contents from CSV, Excel file.

This is usefull when we are working with vectors, and have just a row or a column data to be copied and brought into *R*. With the tabular data, we have to be extra carefull in seletion of columns, rows, the headers, etc.
On my personal note, I would not use this function that often
Below, is the code which can be used.

```
library(psych)
data2 <- read.clipboard()
head(data2)
```

SIMPLE MANIPULATION, VECTORS and MATRICES

13. Create a vector 'vec1' and 'vec2' with elements 1 to 15 and 115 to 101.

```
vec1 <- seq(1,15)
vec2 <- seq(115,101,-1)
vec1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
vec2
```

```
##  [1] 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101
```

Alternate way is to add 100 to the elements of '*vec1*' and sort it in decreasing order

```
vec2 <- vec1 + 100
vec2 <- sort(vec2, decreasing = T)
vec2
```

```
##  [1] 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101
```

---

14. Create a vector 'vec3' of the sum of the log values of vec1 and vec2 in one argument and print the result.

Solution 1 One way to interpret the requirement is to have a vector of same length as 'vec1' and 'vec2', with the summation of log values of each of the elements

```
vec3 <- log(vec1) + log(vec2)
vec3
```

```
##  [1] 4.744932 5.429346 5.826000 6.104793 6.318968 6.492240 6.637258
##  [8] 6.761573 6.870053 6.966024 7.051856 7.129298 7.199678 7.264030
## [15] 7.323171
```

Solution 2 The other way to interpret the requirement is to have a single element vector with total of all the log values of 'vec1' and 'vec2'

```
vec3 <- sum(log(vec1), log(vec2))
vec3
```

```
## [1] 98.11922
```

---

15. Print the 7th element in vec2.

The indexing of vectors starts from 1, so we can simply use the actual number for the position we need to extract the value of.

```
vec2[7]
```

```
## [1] 109
```

---

16. Create matrix 'mat1' by combining the vec1 and vec2 column wise.

As the dimensions of the matrix is not specified, we can create the matrix in two ways -
- a 2 x 15 matrix OR - a 15 x 2 matrix

```
mat1 <- matrix(data = c(vec1,vec2), nrow = length(vec1), byrow = F)
dim(mat1)
```

```
## [1] 15  2
```

```
mat1
```

```
##       [,1] [,2]
##  [1,]    1  115
##  [2,]    2  114
##  [3,]    3  113
##  [4,]    4  112
##  [5,]    5  111
##  [6,]    6  110
##  [7,]    7  109
##  [8,]    8  108
##  [9,]    9  107
## [10,]   10  106
## [11,]   11  105
## [12,]   12  104
## [13,]   13  103
## [14,]   14  102
## [15,]   15  101
```

```
mat1 <- matrix(data = c(vec1,vec2), ncol = length(vec1), byrow = F)
dim(mat1)
```

```
## [1]  2 15
```

```
mat1
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    1    3    5    7    9   11   13   15  114   112   110   108   106
## [2,]    2    4    6    8   10   12   14  115  113   111   109   107   105
##      [,14] [,15]
## [1,]   104   102
## [2,]   103   101
```

---

17. Change the dimensions of mat1 to 5 x 6 and print mat1.

```
## The current dimensions of the matrix is  2 15
```

```r
dim(mat1) <- c(5,6)
mat1
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    6   11  115  110  105
## [2,]    2    7   12  114  109  104
## [3,]    3    8   13  113  108  103
## [4,]    4    9   14  112  107  102
## [5,]    5   10   15  111  106  101
```

---

18. Generate a 5 x 5 matrix 'mat2' with elements 1:5 in the diagonal and other elements being 0.

Here we are going to use the *diag* function to generate the diagonal matrix.

```r
mat2 <- diag(x = seq(1,5),nrow = 5, ncol = 5)
mat2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    2    0    0    0
## [3,]    0    0    3    0    0
## [4,]    0    0    0    4    0
## [5,]    0    0    0    0    5
```

---

19. Add another column of elements 6:10 in mat2, making it 5 x 6 matrix.

As we have to add a column, we are going to use the *cbind()* function, with the new set of values.

```r
mat2 <- cbind(mat2,as.vector(seq(6,10)))
mat2
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    6
## [2,]    0    2    0    0    0    7
## [3,]    0    0    3    0    0    8
## [4,]    0    0    0    4    0    9
## [5,]    0    0    0    0    5   10
```

---

20. Print the values of 4th column of mat2.

Here, again the indexing starts with 1, so we can use the number directly to extract the values.
The format of accessing a element in a matrix is *matrix[row_number, col_number]*. If we do not specify either of the two, it indiates all the values to be extracted from either the row or column.

```
mat2[,4]
```

```
## [1] 0 0 0 4 0
```

---

21. Find which elements in mat2 are greater than or equal to 5.

```
mat2[mat2 >= 5]
```

```
## [1]  5  6  7  8  9 10
```

DATA MANIPULATION

23. Download Retail Score data and save it as 'RSC'.

We will be using the *psych* package for next few steps. So do install and/or load that library. [`install.packages("psych")`]

```
RSC <- read.table("../datasets/RetailScoreData.csv",header = T, sep = ",")
```

---

24. Run describe() function on the data.

The ***describe()*** function provides, the most frequently used and required summary statistics, e.g. mean, std. dev., median, range, skew, kurtosis, etc.

```
describe(RSC)
```

```
##          vars    n      mean        sd    median  trimmed      mad   min
## branch      1 1500     52.20     27.93     64.00    53.67    19.27     3
## ncust       2 1500   3478.07    864.71   3491.00  3499.38  1235.01  1919
## customer    3 1500 257714.55 139555.15 315991.50 265110.14 96778.20 10012
## age         4 1500     34.17     13.14     31.00    32.61    11.86    18
## ed          5 1500      2.64      1.14      2.00     2.60     1.48     1
## employ      6 1500      6.95      8.98      4.00     5.14     5.93     0
## address     7 1500      6.31      6.05      5.00     5.39     5.93     0
## income      8 1500     59.59     67.13     40.00    46.09    23.72    12
## debtinc     9 1500      9.93      6.67      8.50     9.19     6.23     0
## creddebt   10 1500      1.93      2.97      0.99     1.33     1.01     0
## othdebt    11 1500      3.84      5.33      2.21     2.81     2.08     0
## default    12 1500      0.37      0.48      0.00     0.33     0.00     0
##                max     range  skew kurtosis      se
## branch       91.00     88.00 -0.49    -1.30    0.72
## ncust      4809.00   2890.00 -0.11    -1.07   22.33
## customer 453777.00 443765.00 -0.49    -1.30 3603.30
## age          79.00     61.00  0.95     0.32    0.34
## ed            5.00      4.00  0.30    -0.84    0.03
## employ       63.00     63.00  1.97     4.56    0.23
## address      34.00     34.00  1.32     1.64    0.16
## income     1079.00   1067.00  5.53    51.70    1.73
## debtinc      40.70     40.70  1.08     1.24    0.17
## creddebt     35.97     35.97  4.87    36.11    0.08
## othdebt      63.47     63.47  4.81    34.28    0.14
## default       1.00      1.00  0.56    -1.69    0.01
```

However, this works best for the numeric fields/attributes/variables. For a categorical variable, it does calculate the mean, median however it would not make sense to read such type of data. In this case, the ***summary()*** function from the base package works nice.

It shows the summary based on the type of the variables. E.g. in the **iris**, dataset, the *Species* variable is a categorical (factor), and the others are numeric, the `summary` function identifies this and displays the summary statistics accordingly.

```r
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##                 
##                 
##                 
```

25. Take a subset of the *creddebt* and *othedebt* column of the data and assign the values to 'credit.debt' and 'other.debt'.

There are multiple ways to do this, I am listing down few which I generally use, feel free to suggest any other techniques

- Using *sample* function to extract some random values of the mentioned columns. To ensure, I get a consistent output, for further processing I will fix the seed value.

```
set.seed(7)
```

```
credit.debt <- sample(RSC$creddebt,20)
other.debt <- sample(RSC$othdebt,20)
credit.debt
```

```
##  [1] 0.64 1.96 0.44 4.28 0.50 0.53 2.21 8.10 0.18 2.30 0.01 0.09 8.32 2.10
## [15] 1.71 0.37 4.35 9.52 0.47 6.60
```

```
other.debt
```

```
##  [1]  2.67  0.81  0.32  2.22  3.19  5.34  0.49  1.87  1.27  2.90 26.92
## [12] 27.54  1.45  0.94  5.93  0.44  0.45  4.50  1.19  7.92
```

- We can use the regular subsetting method as below, where we are not randomly pickup the values. Instead we are referrencing the observation position to extract.

```
credit.debt <- RSC$creddebt[1400:1450]
other.debt <- RSC$othdebt[1400:1450]
```

- Using the *dplyr* package

```
credit.debt <- RSC %>% select(creddebt) %>%
  slice(1401:1450) %>%
  collect %>% .[["creddebt"]]

other.debt <- RSC %>% select(othdebt) %>%
  slice(1401:1450) %>%
  collect %>% .[["othdebt"]]
```

---

26. Find the mean and median values of 'credit.debt' and 'other.debt'.

```
## credit.debt =

##  [1]  2.05  1.74  0.18  1.18  2.15  1.00  2.00  0.46  0.53  0.13  0.99
## [12]  0.91  0.14  0.05  0.13  1.68  0.27  3.40  3.57  0.34  0.69  3.04
## [23]  2.12  1.39  0.39  0.10  1.62  0.32  1.37  4.82  0.71 35.52  0.96
## [34]  1.56  0.36  5.47  0.46  0.86  0.07  4.76  0.15  0.05  0.41  0.20
## [45]  1.19  3.85  0.78  1.47  0.37  0.03

##  Mean (credit.debt) =  1.9598
##  Median (credit.debt) 0.885

## other.debt =

##  [1]  3.72  5.22  2.27  2.25  2.77  4.06  3.43  1.57  1.06  0.36  1.83
## [12]  1.85  0.88  0.78  0.70  4.03  2.15  5.95 11.19  0.23  0.87  2.10
## [23]  5.45  4.83  1.23  0.63  8.94  4.32  3.30  9.27  0.66 40.70  1.23
## [34]  1.56  1.26  8.21  1.17  1.32  0.73  5.04  0.41  0.34  1.08  0.80
## [45]  1.29  9.67  1.04  1.59  1.27  0.12

##  Mean (other.debt) =  3.5346
##  Median (other.debt) 1.58
```

---

27. Create a vector 'total.debt' by adding element to element of the two vectors, 'credit.debt' and 'other.debt'.

```
total.debt <- credit.debt + other.debt
total.debt
```

```
##  [1]  5.77  6.96  2.45  3.43  4.92  5.06  5.43  2.03  1.59  0.49  2.82
## [12]  2.76  1.02  0.83  0.83  5.71  2.42  9.35 14.76  0.57  1.56  5.14
## [23]  7.57  6.22  1.62  0.73 10.56  4.64  4.67 14.09  1.37 76.22  2.19
## [34]  3.12  1.62 13.68  1.63  2.18  0.80  9.80  0.56  0.39  1.49  1.00
## [45]  2.48 13.52  1.82  3.06  1.64  0.15
```

---

28. Round of the elements in vector 'total.debt' in multiples of tens.

```
total.debt <- round(total.debt,1)
```

---

29. Paste the elements of the two vectors, 'credit.debt' and 'other.debt' using separator ",".

```
debts <- paste(credit.debt, other.debt, sep = ", ")
debts
```

```
##  [1] "2.05, 3.72"  "1.74, 5.22"  "0.18, 2.27"  "1.18, 2.25"  "2.15, 2.77"
##  [6] "1, 4.06"     "2, 3.43"     "0.46, 1.57"  "0.53, 1.06"  "0.13, 0.36"
## [11] "0.99, 1.83"  "0.91, 1.85"  "0.14, 0.88"  "0.05, 0.78"  "0.13, 0.7"
## [16] "1.68, 4.03"  "0.27, 2.15"  "3.4, 5.95"   "3.57, 11.19" "0.34, 0.23"
## [21] "0.69, 0.87"  "3.04, 2.1"   "2.12, 5.45"  "1.39, 4.83"  "0.39, 1.23"
## [26] "0.1, 0.63"   "1.62, 8.94"  "0.32, 4.32"  "1.37, 3.3"   "4.82, 9.27"
## [31] "0.71, 0.66"  "35.52, 40.7" "0.96, 1.23"  "1.56, 1.56"  "0.36, 1.26"
## [36] "5.47, 8.21"  "0.46, 1.17"  "0.86, 1.32"  "0.07, 0.73"  "4.76, 5.04"
## [41] "0.15, 0.41"  "0.05, 0.34"  "0.41, 1.08"  "0.2, 0.8"    "1.19, 1.29"
## [46] "3.85, 9.67"  "0.78, 1.04"  "1.47, 1.59"  "0.37, 1.27"  "0.03, 0.12"
```

---

30. Create a vector 'Names' whose elements will be "Andrie de Vries" and "Joris Meys" using authors <- c("Andrie","Joris") lastnames <- c("de Vries","Meys")

```
authors <- c("Andrie", "Joris")
lastnames <- c("de Vries", "Meys")

Names <- paste(authors, lastnames, sep = " ")
Names
```

```
## [1] "Andrie de Vries" "Joris Meys"
```

---

31. Create a vector 'NAMES' whose elements will have "Jonas" added to all the elements of first names.

```
firstnames <- c("Joris", "Carolien", "Koen")
lastname <- "Meys"
names <- paste("Jonas", firstnames, lastname)
names
```

```
## [1] "Jonas Joris Meys"    "Jonas Carolien Meys" "Jonas Koen Meys"
```

---

32. Load the RetailScoreData file as 'Retail.data' and Create a data.frame 'Retail.3779 with all the observations where ncust is 3779.

```
Retail.data <- read.table("../datasets/RetailScoreData.csv",header = T, sep = ",")
```

Again here we can use any of the subsetting methods to achieve the output. I am going to use the *dplyr* package and its functions.

```
table(Retail.data$ncust)
```

```
##
## 1919 2251 2600 2658 3017 3080 3388 3491 3572 3779 4098 4358 4501 4650 4809
##  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100
```

```
Retail.3779 <- Retail.data %>%
  filter(ncust == 3779)
```

---

33. Sort the data.frame 'Retail.3779' in the decreasing order of variable 'age' and assign it to Retail.3779.sort.

```
Retail.3779.sort <- Retail.3779 %>%
  arrange(desc(age))
```

Another way to sort is like

```
Retail.3779.sort <- Retail.3779[order(-Retail.3779$age),]
```

---

34. See how many observations in 'Retail.3779' are employed for more than 10 years.

```
Retail.3779 %>%
  filter(employ > 10) %>%
  summarise(employed_more_than_10_yrs = n())
```

```
##   employed_more_than_10_yrs
## 1                        19
```

Another way to sort is like

```
sum(Retail.3779$employ > 10)
```

---

35. Find the mean of all observations in 'Retail.data' in variables 'creddebt' and 'othdebt' grouped by 'ncust'.

```
mean_by_ncust <- Retail.data %>%
  group_by(ncust) %>%
  summarise(creddebt_mean = mean(creddebt, na.rm = T),
            othdebt_mean = mean(othdebt, na.rm = T))
mean_by_ncust
```

```
## Source: local data frame [15 x 3]
##
##    ncust creddebt_mean othdebt_mean
##    (int)         (dbl)        (dbl)
## 1   1919        1.6179       3.4213
## 2   2251        1.5734       2.8963
## 3   2600        2.1402       4.5531
## 4   2658        1.5674       3.1970
## 5   3017        1.6331       3.6244
## 6   3080        1.8170       3.9222
## 7   3388        1.9411       3.7843
## 8   3491        2.4720       4.0947
## 9   3572        1.6511       3.2608
## 10  3779        1.7692       3.2160
## 11  4098        2.5523       5.3292
## 12  4358        2.0555       3.7208
## 13  4501        2.1407       4.0505
## 14  4650        1.9969       4.0650
## 15  4809        2.0959       4.5293
```

- Well, we can do it in many other ways, like one below using *describeBy* function from *psych* package

```
d <- cbind("creddebt" = describeBy(Retail.data$creddebt, Retail.data$ncust, mat = T)[,c(2,5)] , "othdeb
describeBy(Retail.data$othdebt, Retail.data$ncust, mat = T)[,c(2,5)])
d <- d[-3]
d
```

```
##     creddebt.group1 creddebt.mean othdebt.mean
## 11             1919        1.6179       3.4213
## 12             2251        1.5734       2.8963
## 13             2600        2.1402       4.5531
## 14             2658        1.5674       3.1970
## 15             3017        1.6331       3.6244
## 16             3080        1.8170       3.9222
## 17             3388        1.9411       3.7843
## 18             3491        2.4720       4.0947
## 19             3572        1.6511       3.2608
## 110            3779        1.7692       3.2160
## 111            4098        2.5523       5.3292
## 112            4358        2.0555       3.7208
## 113            4501        2.1407       4.0505
## 114            4650        1.9969       4.0650
## 115            4809        2.0959       4.5293
```

- OR by using the *aggregate* or *aggregate.data.frame* functions from the base package

```
creddebt <- aggregate(x = Retail.data$creddebt, by = list(Retail.data$ncust), FUN = mean)

othdebt <- aggregate(x = Retail.data$othdebt, by = list(Retail.data$ncust), FUN = mean)

aggregate.data.frame(Retail.data, by = list(Retail.data$ncust), FUN = mean)[,c(3,11,12)]
```

```
##    ncust creddebt othdebt
## 1   1919   1.6179  3.4213
## 2   2251   1.5734  2.8963
## 3   2600   2.1402  4.5531
## 4   2658   1.5674  3.1970
## 5   3017   1.6331  3.6244
## 6   3080   1.8170  3.9222
## 7   3388   1.9411  3.7843
## 8   3491   2.4720  4.0947
## 9   3572   1.6511  3.2608
## 10  3779   1.7692  3.2160
## 11  4098   2.5523  5.3292
## 12  4358   2.0555  3.7208
## 13  4501   2.1407  4.0505
## 14  4650   1.9969  4.0650
## 15  4809   2.0959  4.5293
```

---

36. Split the 'Retail.data' using the split functions and assign the 5th data.frame (sublist 5 – [[5]]) to 'Retail.3017'. The split is to be done on 'ncust'

```
splitted_Retail.data <- split(Retail.data,Retail.data$ncust)
Retail.3017 <- splitted_Retail.data[5]
# Retail.3017 is a list of data frame, so to extract the values of the data frame we need to use the be
head(Retail.3017$`3017`)
```

```
##   branch ncust customer age ed employ address income debtinc creddebt
## 1      3  3017    10012  28  2      7       2     44    17.7     2.99
## 2      3  3017    10017  64  5     34      17    116    14.7     5.05
## 3      3  3017    10030  40  1     20      12     61     4.8     1.04
## 4      3  3017    10039  30  1     11       3     27    34.5     1.75
## 5      3  3017    10069  25  1      2       2     30    22.4     0.76
## 6      3  3017    10071  35  1      2       9     38    10.9     1.46
##   othdebt default
## 1    4.80       0
## 2   12.00       0
## 3    1.89       0
## 4    7.56       0
## 5    5.96       1
## 6    2.68       1
```

Use the 'airquality' data from the data stream given in R to perform the following analysis.

37. Find summary statistics of the data

Before stating with the data, make it point to understand the structure of the data. If the dataset is pre-loaded in R, or if it comes with some package you can view the description about the data using the help command `?airquality`

```
New York Air Quality Measurements.
Daily air quality measurements in New York, May to September 1973.

A data frame with 154 observations on 6 variables.

- [,1]  Ozone    numeric     Ozone (ppb)

- [,2]  Solar.R  numeric     Solar R (lang)

- [,3]  Wind     numeric     Wind (mph)

- [,4]  Temp     numeric     Temperature (degrees F)

- [,5]  Month    numeric     Month (1--12)

- [,6]  Day  numeric     Day of month (1--31)
```

For more information do visit the documentation site at **airquality**

```r
ipdata <- airquality
summary(ipdata)
```

```
##      Ozone           Solar.R           Wind             Temp      
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00  
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88  
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00  
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
##  NA's   :37       NA's   :7                                       
##      Month            Day      
##  Min.   :5.000   Min.   : 1.0  
##  1st Qu.:6.000   1st Qu.: 8.0  
##  Median :7.000   Median :16.0  
##  Mean   :6.993   Mean   :15.8  
##  3rd Qu.:8.000   3rd Qu.:23.0  
##  Max.   :9.000   Max.   :31.0  
## 
```

Again you can use different ways to view the summary statistics. The *describe* function from *psych* package too is helpful.

```
describe(ipdata, na.rm = T)
```

```
##         vars   n   mean    sd median trimmed   mad  min   max range  skew
## Ozone      1 116  42.13 32.99   31.5   37.80 25.95  1.0 168.0   167  1.21
## Solar.R    2 146 185.93 90.06  205.0  190.34 98.59  7.0 334.0   327 -0.42
## Wind       3 153   9.96  3.52    9.7    9.87  3.41  1.7  20.7    19  0.34
## Temp       4 153  77.88  9.47   79.0   78.28  8.90 56.0  97.0    41 -0.37
## Month      5 153   6.99  1.42    7.0    6.99  1.48  5.0   9.0     4  0.00
## Day        6 153  15.80  8.86   16.0   15.80 11.86  1.0  31.0    30  0.00
##         kurtosis   se
## Ozone       1.11 3.06
## Solar.R    -1.00 7.45
## Wind        0.03 0.28
## Temp       -0.46 0.77
## Month      -1.32 0.11
## Day        -1.22 0.72
```

---

38. Find the following, for the 'airquality' dataset

- a. Skewness
- b. Kurtosis

Here we can use the *moments* package and the two functions from the package named *skewness* and *kurtosis*

```
library(moments)
skewness(ipdata, na.rm = T)
```

```
##        Ozone      Solar.R         Wind         Temp        Month
##  1.225680663 -0.423634197  0.344398467 -0.374169579 -0.002367988
##          Day
##  0.002625783
```

```
kurtosis(ipdata, na.rm = T)
```

```
##    Ozone  Solar.R     Wind     Temp    Month      Day
## 4.184071 2.023567 3.068849 2.570600 1.705474 1.801025
```

*As there are NA's in the dataset, we have used the **na.rm=T** to exluce the NAs*

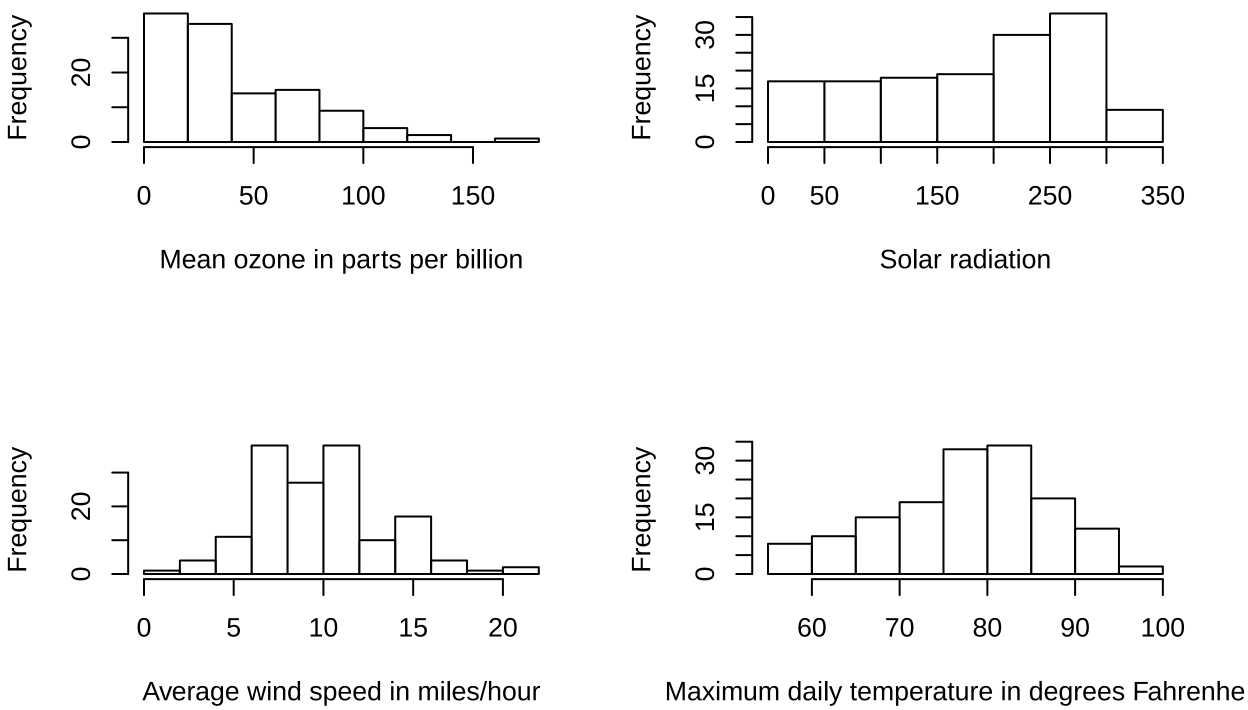
If you notice, there is some difference in skewness values when we use the *describe* function, and when we use the *skewness* function. Do understand the difference please refer to the **type=** parameter of *describe* function in the help files **describe**

---

39. Draw a histogram of the following data

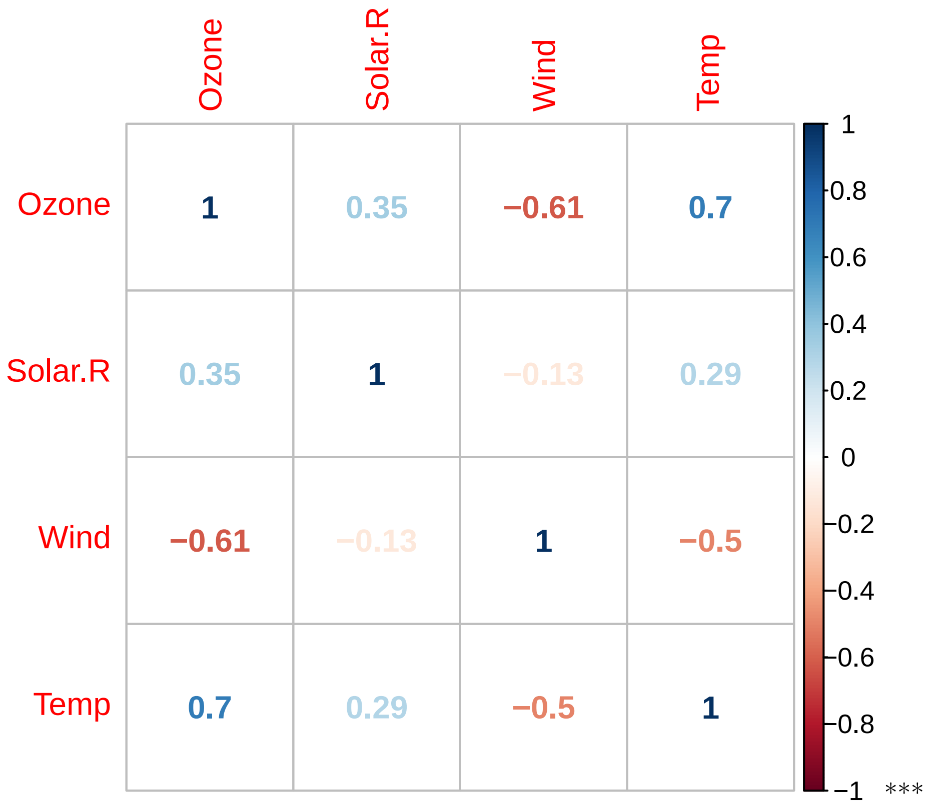
- a. Ozone
- b. Solar.R
- c. Wind
- d. Temp

```
par(mfrow = c(2,2))
?airquality
hist(ipdata$Ozone, xlab = "Mean ozone in parts per billion", main = " ")
hist(ipdata$Solar.R, xlab = "Solar radiation", main = " ")
hist(ipdata$Wind, xlab = "Average wind speed in miles/hour", main = " ")
hist(ipdata$Temp, xlab = "Maximum daily temperature in degrees Fahrenheit ", main = " ")
```



par(mfrow = c(2,2)) is used to format the output so that the plots are aligned in 2 x 2 format. *** ### 40. Find correlation and covariance matrix among the following variables Ozone, Solar.R, Wind & Temp

```
library(corrplot)
cor_matrix <- cor(ipdata[,-c(5,6)],use = "complete.obs")
corrplot(cor_matrix,method = "number")
```



Need to work on covariance matrix. Stay Tune for more. . .