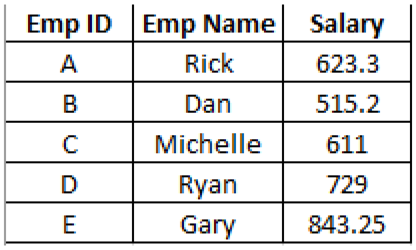
CM 2062 - Statistical Computing with R
Lab Sheet 3

Data Frames

Data Frames are data displayed in a format as a table. In general we have multiple observations for each variable in most situations in data analysis. So we should know how to create data sets.

Data Frames can have different types of data inside it. While the first column can be character, the second and third can be numeric or logical. However, each column should have the same type of data.

Use the **data.frame()** function to create a data frame. Let's see how we create the following data frame in R.



Emp ID	Emp Name	Salary
A	Rick	623.3
B	Dan	515.2
C	Michelle	611
D	Ryan	729
E	Gary	843.25

First we will create the variables in R.

```
> EmpId <- c ("A", "B", "C", "D", "E")
> EmpName <- c("Rick","Dan","Michelle","Ryan","Gary")
> Salary <- c(623.3,515.2,611.0,729.0,843.25)
```

Let's create the data frame calls "EmpData".

```
> EmpData <- data.frame(EmpName, Salary)
> EmpData
   EmpName Salary
1     Rick 623.30
2      Dan 515.20
3 Michelle 611.00
4     Ryan 729.00
5     Gary 843.25
```

Since ID is not a variable which is rather an identifier for the data points, it is better to assign it as row names for the data frame.

```
> row.names(EmpData) <- EmpId
> EmpData
   EmpName Salary
A     Rick 623.30
B      Dan 515.20
C Michelle 611.00
D     Ryan 729.00
E     Gary 843.25
```

Let's check the structure of the data set.

```
> str(EmpData)
'data.frame':   5 obs. of  2 variables:
 $ EmpName: chr  "Rick" "Dan" "Michelle" "Ryan" ...
 $ Salary : num  623 515 611 729 843
```

Let's remove the objects, EmpId, EmpName, Salary.

```
> rm(EmpId, EmpName, Salary)
> EmpId
Error: object 'EmpId' not found
> EmpName
Error: object 'EmpName' not found
> EmpData$EmpName
[1] "Rick"     "Dan"      "Michelle" "Ryan"     "Gary"
```

Extract specific columns from a data frame using column names.

```
> result <- data.frame(EmpData$EmpName,EmpData$Salary)
> result
  EmpData.EmpName EmpData.Salary
1            Rick         623.30
2             Dan         515.20
3        Michelle         611.00
4            Ryan         729.00
5            Gary         843.25
```

Extract first two rows.

```
> result1 <- EmpData[1:2,]
> result1
  EmpName Salary
A    Rick  623.3
B     Dan  515.2
```

Extract 3rd and 5th row with 1st and 2nd column.

```
> result2 <- EmpData[c(3,5),1:2]
> result2
   EmpName Salary
C Michelle 611.00
E     Gary 843.25
```

Expand Data Frame

A data frame can be expanded by adding columns and rows.

Add Column

Just add the column vector using a new column name. Let's add the "Dept" coulmn.

```
> EmpData$Dept <- c("IT","Operations","IT","HR","Finance")
> EmpData
   EmpName Salary       Dept
A     Rick 623.30         IT
B      Dan 515.20 Operations
C Michelle 611.00         IT
D     Ryan 729.00         HR
E     Gary 843.25    Finance
```

Add Row

To add more rows permanently to an existing data frame, we need to bring in the new rows in the same structure as the existing data frame and use the **rbind()** function.

In the example below we create a data frame with new rows and merge it with the existing data frame to create the final data frame.

Let's create a second data frame call "EmpNewData" with following data.

```
> EmpId <- c ("F", "G", "H")
> EmpName <- c("Rasmi","Pranab","Tusar")
> Salary <- c(578.0,722.5,632.8)
> Dept = c("IT","Operations","Finance")
> EmpNewData <- data.frame( EmpName, Salary , Dept)
> EmpNewData
  EmpName Salary       Dept
1   Rasmi  578.0         IT
2  Pranab  722.5 Operations
3   Tusar  632.8   Fianance

> row.names(EmpNewData) <- EmpId
# When the variable names were removed go for second method.
# Always use of second method will be good. After creating the data frame
# always call the variables which are inside the data frame.
> row.names(EmpNewData) <- EmpNewData$EmpId
> EmpNewData
  EmpName Salary       Dept
F   Rasmi  578.0         IT
G  Pranab  722.5 Operations
H   Tusar  632.8   Fianance
```

Let's bind the two data frames to create a data frame with all data call "FinalEmpData".

```
> EmpFinalData <- rbind(EmpData, EmpNewData)
> EmpFinalData
   EmpName Salary       Dept
A     Rick 623.30         IT
B      Dan 515.20 Operations
C Michelle 611.00         IT
D     Ryan 729.00         HR
E     Gary 843.25    Finance
F    Rasmi 578.00         IT
G   Pranab 722.50 Operations
H    Tusar 632.80   Fianance
```

Summary of Data in Data Frame

The statistical summary and nature of the data can be obtained by applying **summary()** function.

```
> summary(EmpFinalData)
   EmpName              Salary            Dept
 Length:8           Min.   :515.2   Length:8
 Class :character   1st Qu.:602.8   Class :character
 Mode  :character   Median :628.0   Mode  :character
                    Mean   :656.9
                    3rd Qu.:724.1
                    Max.   :843.2
```

You can see a spread sheet of the data using **fix()** function.

```
fix(EmpFinalData)
```

You can add variables(columns) or rows to the data frame using spreadsheet.

Exercise

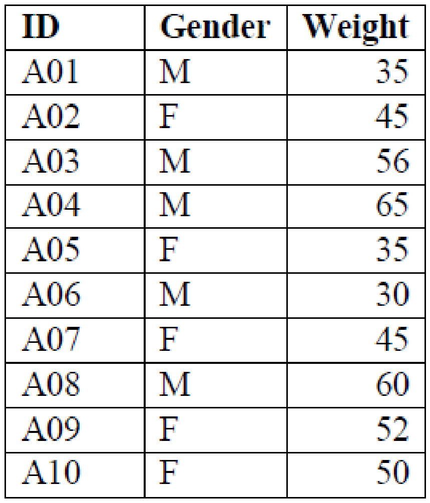
The following table shows 10 students details.

(a) Create a data frame call "Students" for these data and use "ID" as the raw names.

(b) Suppose that for each of the student in the above example, their heights are also measured. The heights (cm) are 86.5,71.8,77.2,84.9,75.4,80.5,92.5,78.5,70.5,85.2. Include this variable also to the data frame.

(c) Suppose that for each of the student in the above example, their age are also noted. The ages in years are 20, 19, 21, 18, 17, 21, 22, 19, 24, 25. Open the data frame in a spread sheet and enter age variable to the data frame.

(d) Obtain the summary of the data.



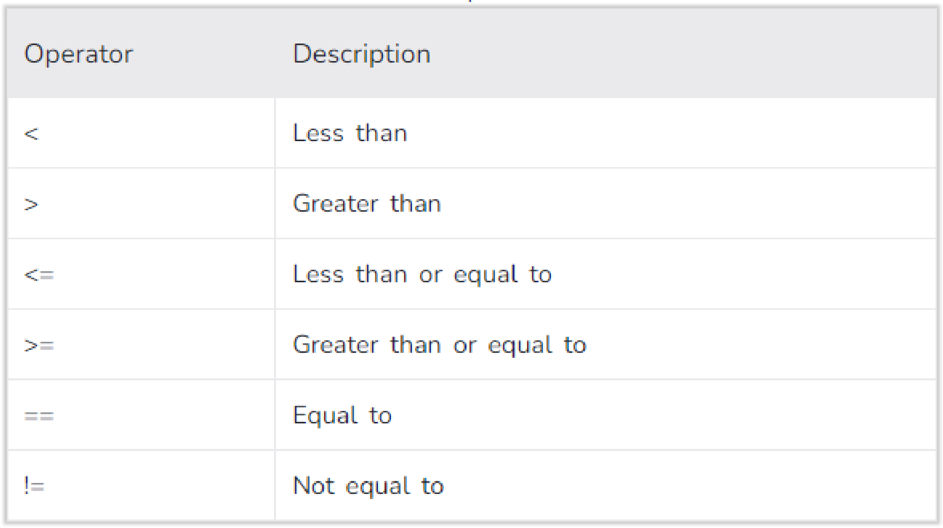
ID	Gender	Weight
A01	M	35
A02	F	45
A03	M	56
A04	M	65
A05	F	35
A06	M	30
A07	F	45
A08	M	60
A09	F	52
A10	F	50

Relational and logical Expressions in R

Relational Operators in R

Relational operators are used to compare between values. Here is a list of relational operators available in R.

Relational Operators in R



Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Examples

```
> x <- 5
> y <- 16
> x < y
[1] TRUE
```

```
>   x > y
[1] FALSE
>   x <= 5
[1] TRUE
>   y >= 20
[1] FALSE
>   y == 16
[1] TRUE
> x != 5
[1] FALSE
```

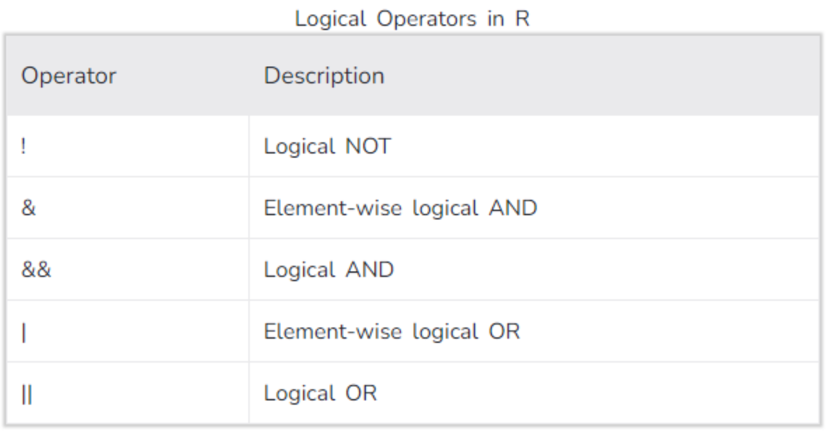
The above mentioned operators work on vectors. All operations are carried out in element-wise fashion.

```
> x <- c(2,8,3)
> y <- c(6,4,1)
> x > y
[1] FALSE  TRUE  TRUE
> x < y
[1]  TRUE FALSE FALSE
```

We can sub set the elements in a vector using relational operators.

```
> X <- c(9, 1, 5, 3, 4, 10, 99)
> X[X>5]
[1]  9 10 99
> X[X>=5]
[1]  9  5 10 99
> X[X<4]
[1] 1 3
```

Logical Operators in R



Logical Operators in R

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

Operators & and | perform element-wise operation producing result having length of the longer operand.

But && and || examines only the first element of the operands resulting into a single length logical vector.

Zero is considered FALSE and non-zero numbers are taken as TRUE.

Examples

```
>  x <- c(TRUE,FALSE,0,6)
>  !x
[1] FALSE  TRUE  TRUE FALSE
>  y <- c(FALSE,TRUE,FALSE,TRUE)
>  x&y
[1] FALSE FALSE FALSE  TRUE
>  x&&y
[1] FALSE
>  x|y
[1]  TRUE  TRUE FALSE  TRUE
>  x||y
[1] TRUE
```

We can sub set the elements in a vector using logical operators.

```
> X <- c(9, 1, 5, 3, 4, 10, 99)
>  X[X > 3 & X < 10]
[1] 9 5 4
>  X[X !=5]
[1]  9  1  3  4 10 99
>  X[X > 10 | X < 5]
[1]  1  3  4 99
```

Example

Recall the data frame call "EmpFinalData".

(a) Add 100 dollar bonus to Salary

(b) Obtain the Salary of the Employees who attached to the Finance department.

(c) Obtain the EmpID of the employees whose salary is above 700 dollars.

(d) Obtain the EmpName of the employees who attached to the IT Department and whose Salary is below 700 dollars.

Answers

```
(a)
> EmpFinalData$Salary <- EmpFinalData$Salary + 100
> EmpFinalData$Salary
[1] 723.30 615.20 711.00 829.00 943.25 678.00 822.50 732.80

(b)
> EmpFinalData$Salary[EmpFinalData$Dept=="Finance"]
[1] 943.25 732.80

(c)
> row.names(EmpFinalData)[EmpFinalData$Salary>700]
[1] "A" "C" "D" "E" "G" "H"

(d)
> EmpFinalData$EmpName[EmpFinalData$Dept=="IT" & EmpFinalData$Salary<700]
[1] "Rasmi"
```

Exercise

Recall the data frame call "Students".

(a) Obtain the wight of the female students.

(b) Obtain the height of the male students.

(c) Obtain the ID of the students whose age is above 20 years.

(d) Obtain the ID of the male students whose weight is below 40 kg.

(e) Obtain the ID of the female student whose height is above 75 cm.

Import Data in to R

There are several ways to import data in to R.

Importing data from a text file

We use **read.table()** function to import data from a text file.

Example
Let's take we have saved "EmpFinalData" in to a text file call "EmpFinalData.txt". Then there are two ways to read this text file to R.

1. Give the full path to access the Text file.

```
EmpFinalData <- read.table("F:/KDU_work/CM 2062/R works/EmpFinalData.txt",
                           header=TRUE)
```

2. Set working directory to text file location and give the text file name only as the path.

```
EmpFinalData <- read.table("EmpFinalData.txt", header=TRUE)
```

After reading the data set in to R you can do any arithmetic operations that you have done to the data fame earlier.

Importing from a CSV file

Enter the "EmpFinalData" in to an excel file and save it as a CSV file.

```
EmpFinalData <- read.csv("F:/KDU_work/CM 2062/R works/EmpFinalData.csv",
                         header=TRUE)

or after setting working directory to the relevant folder

 EmpFinalData <- read.csv("EmpFinalData.csv", header=TRUE)
```

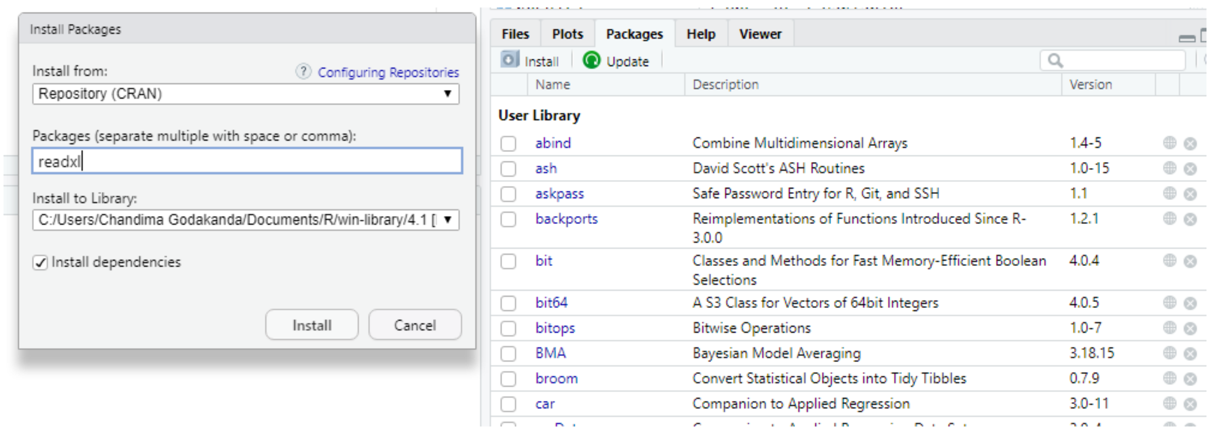
R Packages

One specialty with R is that the system can easily be extended. The system allows you to write new functions and package those functions in a so called ‘R package’ (or ‘R library’). There is a lively R user community and many R packages have been written and made available on CRAN for other users. Just a few examples, there are packages for portfolio optimization, drawing maps, exporting objects to html, time series analysis, spatial statistics and the list goes on and on.

When you download R, already a number (around 30) of packages are downloaded as well. To use a function in an R package, that package has to be attached to the system. When you start R not all of the downloaded packages are attached, only seven packages are attached to the system by default. You can use the function search to see a list of packages that are currently attached to the system, this list is also called the search path.

```
> search()
 [1] ".GlobalEnv"          "package:readr"      "package:readxl"      "tools:rstudio"
 [5] "package:stats"       "package:graphics"   "package:grDevices"   "package:utils"
 [9] "package:datasets"    "package:methods"    "Autoloads"           "package:base"
```

Let’s install "readxl" Package in to R.



Install Packages

Install from: Repository (CRAN)

Packages (separate multiple with space or comma): readxl

Install to Library: C:/Users/Chandima Godakanda/Documents/R/win-library/4.1 [

☑ Install dependencies

Name	Description	Version
User Library		
abind	Combine Multidimensional Arrays	1.4-5
ash	David Scott's ASH Routines	1.0-15
askpass	Safe Password Entry for R, Git, and SSH	1.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1
bit	Classes and Methods for Fast Memory-Efficient Boolean Selections	4.0.4
bit64	A S3 Class for Vectors of 64bit Integers	4.0.5
bitops	Bitwise Operations	1.0-7
BMA	Bayesian Model Averaging	3.18.15
broom	Convert Statistical Objects into Tidy Tibbles	0.7.9
car	Companion to Applied Regression	3.0-11

or use **install.packages()** function in R.

```
> install.packages("readxl")
```

Then we will attach the "readxl" package in to R using **library()** function.

```
> library(readxl)
```

Note: You can download the package manual readxl.pdf from the internet and it contains all the details of the packager including available functions.

Importing data from excel

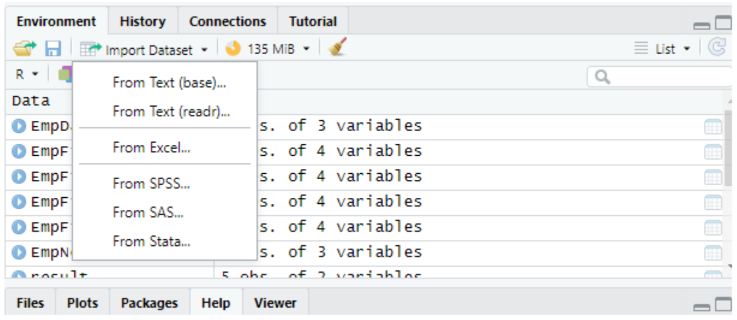
Let's import the data in to R from "EmpFinalData.xlsx" using "**read_excel()**" function in "readxl" package.

```
EmpFinalData <-  read_excel("F:/KDU_work/CM 2062/R works/EmpFinalData.xlsx")

or after setting working directory to the relevant folder

 EmpFinalData <-  read_excel("EmpFinalData.xlsx")
```

Importing Data in to R using "Import Dataset" wizard in R



```
Data
EmpD    s. of 3 variables
EmpF    s. of 4 variables
EmpF    s. of 4 variables
EmpF    s. of 4 variables
EmpF    s. of 4 variables
EmpN    s. of 3 variables
```

RStudio contains a nice feature that makes importing of data more convenient. Essentially, this "Import Dataset" wizard helps us to import data from from different file types.

To import data from SAS,SPSS and Stata, you need to download specific packages to your R program and then you can directly import data in those file formats.

Exercise
Import the "EmpFinalData.txt", "EmpFinalData.csv" and "EmpFinalData.xlsx" files in to R using "Import Dataset" wizard in R.

Exporting data from R

```
# Set working directory to "Export" folder.
setwd("F:/KDU_work/CM 2062/R works/Export")
```

Let's consider the data frame call "EmpFinalData".

Export in to a Text file

```
write.table(EmpFinalData, file = "EmpFinalData.txt", sep = "\t",
            row.names = TRUE, col.names =TRUE)

# If you have specified the raw names in the data frame
then "row.names = TRUE" will help you to export the data frame
in the same way by taking the "EmpID" as the raw names.
```

Export in to a CSV file

```
 write.csv(EmpFinalData, file = "EmpFinalData.csv")
```

Export in to an Excel file

```
install.packages("writexl")
 library(writexl)
 write_xlsx(EmpFinalData, "EmpFinalData.xlsx")
```

Note: Check the differences in the received data set in three file types and decide what is the best file type to export data.

Exercise
Export the "EmpFinalData" and "Students" data frames in to text, excel and csv files by specifying the full path where you should have your data set without setting the working directory.