# R – Programming Introduction – Part 3

## Agenda of this activity

In this activity, we are going to learn the below aspects of R Programming

1. Dataframes
2. Inbuilt Datasets
3. Read a File
4. Factor Variables

## 1. Dataframes

A data frame has the variables of a dataset as columns and the observations as rows. It's a tabular structure.
Columns can hold different data types but of same dimensions

Example - 1:
Use the 'dataframe' command to create a Dataframe.

```
# create a DataFrame
x <- data.frame(Age = c(25,30,35,40), Working = c(T, T, F, F))
x
```

Structure of Dataframe columns can be observed using 'str' command.

```
# observe the structure of a Data frame with below syntax
str(x)
```

More details about the Dataframe:

```
# use nrows and ncols to see the dimensions of the data
nrow(x)
ncol(x)

# to know the column names, use the names functions
names(x)
```

Referring a particular column of a Dataframe:

To refer to individual columns of the dataframe, use the '$' symbol
x$Age
x$Working

**Sub-setting Dataframes**

Example – 2:

Multiple vectors of same size can be combined to create a Dataframe.

Defining vectors:

```
# Definition of vectors
name <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
type <- c("Terrestrial planet", "Terrestrial planet", "Terrestrial planet",
        "Terrestrial planet", "Gas giant", "Gas giant", "Gas giant", "Gas giant")
diameter <- c(0.382, 0.949, 1, 0.532, 11.209, 9.449, 4.007, 3.883)
rotation <- c(58.64, -243.02, 1, 1.03, 0.41, 0.43, -0.72, 0.67)
rings <- c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE, TRUE)
```

Combining vectors to create dataframe:

```
# Create a data frame from the vectors
planets_df <- data.frame(name,type,diameter,rotation,rings)
planets_df
```

Printing the 3rd column of 1 row:

```
# Print out diameter of Mercury (row 1, column 3)
planets_df[1,3]
```

Printing the entire 4th row

```
# Print out data for Mars (entire fourth row)
planets_df[4,]
```

Instead of using numerics to select elements of a data frame, you can also use the variable names to select columns of a data frame.

Suppose you want to select the first three elements of the type column. One way to do this is

```
planets_df[1:3,2]
```

A possible disadvantage of this approach is that you have to know (or look up) the column number of type, which gets hard if you have a lot of variables.
It is often easier to just make use of the variable name as shown below.

```
planets_df[1:3,"type"]
```

Structure of the dataframe:

```
str(planets_df)
```

Subsetting examples:

```
# Select first 5 values of diameter column
planets_df[1:5,"diameter"]

# Select the entire diameter column (both the lines below means the same)
planets_df[,3]
planets_df[,"diameter"]

# There is a short-cut for above command. If your columns have names, you can use the $ sign to fetch all the values:
planets_df$diameter
```

## 2. In-built Datasets

Let us look at a famous dataset called ‘mtcars’

Loading an in-built dataset as a Data-frame:

```
# to load the dataset, use ‘data’ command
data("mtcars")
```

Data set information (mtcars - Motor Trend Car Road Tests)

The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models).

The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of multivalued discrete and continuous attributes.

A data frame with 32 observations on 11 (numeric) variables

### Description of the Attributes

## - mpg Miles/(US) gallon
## - cyl Number of cylinders
## - disp Displacement (cu.in.)
## - hp Gross horsepower
## - drat Rear axle ratio
## - wt Weight (1000 lbs)
## - qsec 1/4 mile time
## - vs Engine (0 = V-shaped, 1 = straight)
## - am Transmission (0 = automatic, 1 = manual)
## - gear Number of forward gears
## - carb Number of carburetors

Looking at the metadata of the dataset:

```
# dimensions can be observed just like the matrix using dim command
dim(mtcars)

# Investigate the structure of mtcars
str(mtcars)
```

Looking at the top few rows:
Head is an in-built command that can be applied on the dataframe.

```
# First 6 elements of the built-in data frame mtcars
head(mtcars)
```

Top 10 rows:

```
# First 10 elements of the built-in data frame mtcars
my_df <- mtcars[1:10,]
my_df
```

Last 6 rows using the in-built tail command:

```
# Last 6 elements of the built-in data frame mtcars
tail(mtcars)
```

## 3. Read a file from directory

Dataset mtcars is also shared in the form of a .csv file which is needed to be read into the R environment so that any processing can be done on that data

Command to get the current working directory:

```
# Present Working Directory
getwd()
```

Based on the location from where we need to read the dataset(in csv format), we will set the working directory as shown below:

```
## Setting the working directory
setwd("/Users/manojakella/Documents/Academics/Batch78/CSE7312c/Batch80_R_and_Python_Introduction")
```

Once, the working directory is set to right location, the file can be read as a data-frame using read.csv command as shown below:

```
# Reading data from a csv file and Creating a Data Frame

mtcars = read.csv(file = "mtcars.csv",header=TRUE,sep=",")
class(mtcars)
```

Analysing the Data-frame:

```
# Head and Tail functions

# Let's take a look at the top rows of the dataframe

head(mtcars)

# Let's take a look at the bottom rows of the dataframe

tail(mtcars)

# Structure of the R Object
str(mtcars)
```

## 4. Factors

What's a factor and why would you use it?

The term factor refers to a statistical data type used to store categorical variables. The difference between a categorical variable and a continuous variable is that a categorical variable can belong to a limited number of categories. A continuous variable, on the other hand, can correspond to an infinite number of values.

It is important that R knows whether it is dealing with a continuous
or a categorical variable, as the statistical models you will develop in
the future treat both types differently.
(You will see later why this is the case.)

```
x <- factor(c("1", "2", "2", "1","1","2","1","2","2"))
x
```

A good example of a categorical variable is gender.
In many circumstances you can limit the gender categories to
"Male" or "Female". (Sometimes you may need different categories.
For example, you may need to consider chromosomal variation,
hermaphroditic animals, or different cultural norms, but you will
always have a finite number of categories.)

Some Practice problems:

Few exercises from the class

** The following questions were done in the class, please ensure that you solve the question before looking at the solution. Also, it is possible that you solve the same question differently!

* Question 1: Create a vector with elements being 98, 82, 102, 99, 100;
  # a) Extract the first element
  # b) Extract the first and the fifth element from the vector.

* Question 2: Create a vector containing a sequence of numbers in the interval of 40 to 1000, separated by 10. Find the length of the vector.

* Question 3: Create a vector containing a sequence of numbers in the interval of 1 to 2500, separated by 6 and subset the first 50 elements of the vector

* Question 4: Create a vector that has elements: 3, 4, 5, 7 and another vector with elements : 6, 9, 12, 15, 18, 21 ; Use the "+" and "-" operator between the two vectors and report your observation.

* Question 5: Multiply c(34, 43, 22, 43) and c(13, 17)

* Question 6: Create two vectors of sequence 1 to 10 and 11 to 20; column bind these vectors; Now create a matrix containing data from 21 to 40 and column bind that matrix to the resultant matrix of the first operation.

* Question 7:

```
my_list <- list("first_element" = F,
        "second_element" = matrix(data = 1:6, nrow = 2, ncol = 3),
        "third_element" = 20:200)
```

* Access the first element of the list such that
* a) it returns the name and the content as an object of class "list"
* b) returns an object of class "logical"

* Question 8: Create a data frame with two columns using the data.frame() function. First column should contain characters, second column should contain numeric objects. The columns should be assigned valid names.

* Question 9: Create a data frame with one column as a squence of numbers between 1 to 50, the other being from 51 to 100; After creating the data frame change the column names to "column1" and "column2"