
# Sinhgad Institutes

**Name of the Student:**_____**Roll no:**_____

**CLASS:- T.E.[I.T]** **Division: A**

**Course: - 2019**

**Subject: 314457: Data Science and Big Data Analytics Laboratory**

**PART_ B _Assignment No. 01**

**Marks: _____/10**

**Date of Performance: ___/____/_____** **Sign with Date: _____**

## Part- B

## ASSIGNMENT NO: 01

**TITLE:**

To Perform the data cleaning operations using Python on the Facebook data sets.

**AIM:**

To Perform the following operations using Python on the Facebook metrics data sets

1. Create data subsets
2. Merge Data
3. Sort Data
4. Transposing Data
5. Shape and reshape Data

**OBJECTIVE:**

1. To understand and apply the Analytical concept of Big data using Python.
2. To apply the Analytical concept of Big data using R/Python.

**Software used:** IDLE Shell 3.9.6 (Python 3.9.6)

**THEORY:**

**R programming Language:**

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac. R is free software distributed under a GNU-style copy left, and an official part of the GNU project called GNU S. R was initially written by Ross Ihaka and Robert Gentleman at the

Department of Statistics of the University of Auckland in Auckland, New Zealand. R made its first appearance in 1993.

- A large group of individuals has contributed to R by sending code and bug reports.
- Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive.

**Features of R**

As stated earlier, R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R:

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

**Installing R**

To download R:

1. Go to http://cran:r-project:org/mirrors:html.
2. Select your closest local mirror.
3. Select your operating system (Linux, OS X, or Windows).
4. Depending on your OS, download the appropriate _le, along with any required packages.
5. When the download is complete, unzip the _le and install.

**Basic Commands in R**

- To read File (csv) use:
- read.csv("/Path/to/file")

- Get library locations containing R packages
- .libPaths()

- Get the list of all the packages installed
- library()

- Get all packages currently loaded in the R environment
- search()

- Install package manually
- install.packages(file_name_with_path, repos = NULL, type = "source")
- # Read the first worksheet in the file input.xlsx.
- data <- read.xlsx("input.xlsx", sheetIndex = 1) • print(data)

**Assignment and Expression Operations**

R commands are either assignments or expressions Commands are separated either by a semicolon ; or newline An expression command is evaluated and (normally) printed An assignment command evaluates an expression and passes the value to a variable but the result is not printed

**Expression Operations**

> 1 + 1

[1] 2

**Assignment Operations**

> res <- 1 + 1

“<-” is the assignment operator in R

a series of commands in a file (script) can be executed with the command: source(“myfile.R”)

**Sample session**

> 1:5

[1] 1 2 3 4 5

> powers.of.2 <- 2^(1:5)

> powers.of.2 [1] 2 4 8 16 32

> class(powers.of.2)

[1] "numeric"

> ls()

[1] "powers.of.2" "res"

> rm(powers.of.2)

**Subsetting in R**

R has powerful indexing features for accessing object elements. These features can be used to select and exclude variables and observations. The following code snippets demonstrate ways to keep or delete variables and observations and to take random samples from a dataset.

**Selecting (Keeping) Variables**

```
# select variables v1, v2, v3 myvars
<- c("v1", "v2", "v3")
newdata <- mydata[myvars]

# another method myvars <- paste("v", 1:3, sep="")
newdata <- mydata[myvars]

# select 1st and 5th thru 10th variables newdata
<- mydata[c(1,5:10)]
```

**Excluding (DROPPING) Variables**

```
# exclude variables v1, v2, v3
myvars <- names(mydata) %in% c("v1", "v2", "v3") newdata
<- mydata[!myvars]
# exclude 3rd and 5th variable newdata <- mydata[c(-3,-5)]
# delete variables v3 and v5 mydata$v3
<- mydata$v5 <- NULL
```

**Selecting Observations**

```
# first 5 observations newdata
<- mydata[1:5,]

# based on variable values
newdata <- mydata[ which(mydata$gender=='F' & mydata$age > 65), ]

# or attach(mydata)
```

newdata <- mydata[ which(gender=='F' & age > 65),] detach(mydata)

**Selection using the Subset Function**

The subset ( ) function is the easiest way to select variables and observations. In the following example, we select all rows that have a value of age greater than or equal to 20 or age less then 10. We keep the ID and Weight columns.

**# using subset function**

newdata <- subset(mydata, age >= 20 | age < 10, select=c(ID, Weight))

In the next example, we select all men over the age of 25 and we keep variables weight through income (weight, income and all columns between them).

**# using subset function (part 2)**

newdata <- subset(mydata, sex=="m" & age > 25, select=weight:income)

**Merging Data**

**Adding Columns**

To merge two data frames (datasets) horizontally, use the merge function. In most cases, you join two data frames by one or more common key variables (i.e., an inner join).

# merge two data frames by ID total <- merge(data frameA,data frameB,by="ID") # merge two data frames by ID and Country

total <- merge(data frameA,data frameB,by=c("ID","Country"))

**Adding Rows**

To join two data frames (datasets) vertically, use the rbind function. The two data frames must

have the same variables, but they do not have to be in the same order. total <- rbind(data frameA, data frameB)

If data frameA has variables that data frameB does not, then either:

1. Delete the extra variables in data frameA or
2. Create the additional variables in data frameB and set them to NA (missing)

**Sorting Data**

To sort a data frame in R, use the order( ) function. By default, sorting is ASCENDING.

Prepend the sorting variable by a minus sign to indicate DESCENDING order. Here are some examples.

# sorting examples using the mtcars dataset attach(mtcars)

# sort by mpg

newdata <- mtcars[order(mpg),]

# sort by mpg and cyl

newdata <- mtcars[order(mpg, cyl),]

#sort by mpg (ascending) and cyl (descending) newdata

<- mtcars[order(mpg, -cyl),] detach(mtcars)

Reshaping Data

R provides a variety of methods for reshaping data prior to analysis.

**Transpose**

Use the t() function to transpose a matrix or a data frame. In the later case, row names become variable (column) names.

# example using built-in dataset mtcars t(mtcars)

**The Reshape2 Package**

Hadley Wickham has created a comprehensive package called reshape2 to massage data. Basically, you "melt" data so that each row is a unique id-variable combination. Then you "cast" the melted data into any shape you would like. Here is a very simple example.

| **mydata Id** | **time** | **x1** | **x2** |
|---|---|---|---|
| 1 | 1 | 5 | 6 |
| 1 | 2 | 3 | 5 |
| 2 | 1 | 6 | 1 |
| 2 | 2 | 2 | 4 |

# Example of melt function

library (reshape)

mdata <- melt (mydata, id=c ("id","time"))

| newdata | Id | time | variable | value |
|---|---|---|---|---|
| 1 | 1 | x1 | | 5 |
| 1 | 2 | x1 | | 3 |
| 2 | 1 | x1 | | 6 |
| 2 | 2 | x1 | | 2 |
| 1 | 1 | x2 | | 6 |
| 1 | 2 | x2 | | 5 |
| 2 | 1 | x2 | | 1 |
| 2 | 2 | x2 | | 4 |

**# cast the melted data**

# syntax of cast (data, formula, function) subjmeans

<- cast (mdata, id~variable, mean) timemeans <- cast (mdata, time~variable, mean)

Subjmeans

| Id | x1 | x2 |
|---|---|---|
| 1 | 4 | 5.5 |
| 2 | 4 | 2.5 |

timemeans

| time | x1 | x2 |
|---|---|---|
| 1 | 5.5 | 3.5 |
| 3 | 2.5 | 4.5 |

**CONCLUSION:**

After the study of this assignment we are familiar with the data cleaning operations using Python.

**Write Short Answers for Following Questions**

1. What is R?
2. Explain about data import in R language.
3. Two vectors X and Y are defined as follows – X <- c(3, 2, 4) and Y <- c(1, 2). What will be output of vector Z that is defined as Z <- X*Y.
4. How missing values and impossible values are represented in R language?
5. How many data structures does R language have?
6. What is the value of f (2) for the following R code?

```
b <- 4
 f <- function (a)
{
b <- 3
b^3 + g (a)
}
g <- function (a)
{
a*b }
```

**Viva Questions**

1. Explain about the significance of transpose in R language.
2. What are the different types of sorting algorithms available in R language?
3. How can you add datasets in R?
4. What is the memory limit in R?
5. How will you read a .csv file in R language?
6. How do you write R commands?
7. Which function helps you perform sorting in R language?
8. How will you merge two dataframes in R programming language?
9. Write the R programming code for an array of words so that the output is displayed in decreasing frequency order.
10. Explain the usage of which() function in R language with the help of example.
11. Explain the usage of subset() function in R with the help of example.

## Python Program

**##Datafreame**

```
import pandas as pd
#data = {"Roll-num": [10,20,30,40,50,60,70], "Age":[12,14,13,12,14,13,15], "NAME":['John','Camili','Rheana','Joseph','Amanti','Alexa','Siri']}
data = pd.read_csv('dataset_Facebook.csv')
block = pd.DataFrame(data)
print("Original Data frame:\n")
print(block)
print(block.loc[[0,1,3]])
print(block.loc[0:3])
#print(block.loc[0:2,['Age','NAME']])
print(block.loc[0:2,['Type','like']])
print(block.iloc[[0,1,3,6],[0,2]])
```

**##Merging**

```
import pandas as pd
d1 = {'Name': ['Pankaj', 'Meghna', 'Lisa'], 'Country': ['India', 'India', 'USA'], 'Role': ['CEO', 'CTO', 'CTO']}
df1 = pd.DataFrame(d1)
print('DataFrame 1:\n', df1)
df2 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Pankaj', 'Anupam', 'Amit']})
print('DataFrame 2:\n', df2)
df_merged = df1.merge(df2)
print('Result Inner Join:\n', df_merged)
print('Result Left Join:\n', df1.merge(df2, how='left'))
print('Result Right Join:\n', df1.merge(df2, how='right'))
print('Result Outer Join:\n', df1.merge(df2, how='outer'))
```

**#Sorting**

```
# importing pandas package
import pandas as pd
```

```
# assign dataset
csvData = pd.read_csv("records.csv")
# displaying unsorted data frame
print("\nBefore sorting:")
print(csvData)
# sort data frame
#csvData.sort_values(["Salary"], axis=0, ascending=[False], inplace=True)
csvData.sort_values(["Salary"],axis=0,        ascending=[True], inplace=True)
# displaying sorted data frame
print("\nAfter sorting:")
print(csvData)
```

**#Transpose**

```
import pandas as pd
df = pd.read_csv('records.csv')
print(df)
print(df.T)
print(df.shape)
print(df.size)
```

**#Shap**

```
import pandas as pd
df = pd.read_csv('records.csv')
print(df)
print(df.shape)
print(df.size)
```