

STES'
SINHGAD COLLEGE OF ENGINEERING, PUNE
DEPARTMENT OF COMPUTER ENGINEERING



LAB MANUAL
310246: SKILL DEVELOPMENT LAB

(2015 COURSE)

TE COMPUTER

Suggested List of Laboratory Assignments on PYTHON

1. Getting Started with Python (Example Word count exercise)
2. Build the Hangman Game using Python.
3. Write python code loads the any dataset (example Game_medal.csv), and plot the graph.
4. Write python code loads the any dataset (example Game_medal.csv), and does some basic datacleaning. Add component on data set.

Suggested List of Laboratory Assignments on DATA Science with R

1. Getting Started with R installation, R objects and basic statistics.
2. Using R for data preprocessing, exploratory analysis, visualization.
3. Using R for correlation and regression analysis.
4. Data analysis case study using R for readily available data set using any one machine learning algorithm.

Suggested Mini Project on PYTHON and DATA Science with R

1. Implementing a simple Recommender System based on user buying pattern.
2. Twitter Sentiment Analysis in Python
3. Applying linear regression model to a real world problem.

Subject Teacher/s:

Prof. G. G. Chiddarwar

Prof. S. H. Shaikh

Prof. N. A. Mhetre

Experiment No	1
Title	Word count exercise
Roll No.	
Date	
Sign of staff	

Problem Statement:

Write a python program that reads a text file, scramble the words in the file as per the rules given below and write the output to the new text file named by appending the word 'scrabble' to input filename.

RULES:

1. Words less than or equal to 3 characters need not be scrambled.
2. Don't scramble first and last character.
3. Special characters are to be maintained as it is. Special characters like (. , ; , ? , , , !)
4. Maintain sequence of line of text file.
5. Use functions, modules as per requirement.
6. It should be based on functionality, coding standards, modularity.
7. Functionality like number of lines, number of words, vowels in the text file.

Aim:

To scramble the words in the text file according to rules mentioned in problem statement.

Objective:

- To make user familiar with python variables, data types, control structure etc.
- To use file access methods
- To use inbuilt library functions and
- To create module and understand importance of it.

Input – Prompt user to enter input text file name.

Output– Scrambled text file as per rules mentioned and input text file named by appending the word scramble.

Theory:

1. Introduction to Python

- Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991.
- Python Features
 - Python is object-oriented
 - Structure supports such concepts as polymorphism, operation overloading, and multiple inheritance
 - It's free (open source)
 - Downloading and installing Python is free and easy
 - Source code is easily accessible
 - Free doesn't mean unsupported! Online Python community is huge
 - It's portable
 - Python runs virtually every major platform used today

- As long as you have a compatible Python interpreter installed, Python programs will run in exactly the same manner, irrespective of platform
- It's powerful
 - Dynamic typing
 - Built-in types and tools
 - Library utilities
 - Third party utilities (e.g. Numeric, NumPy, SciPy)
 - Automatic memory management
- It's mixable
 - Python can be linked to components written in other languages easily
 - Linking to fast, compiled code is useful to computationally intensive problems
 - Python is good for code steering and for merging multiple programs in otherwise conflicting languages
- Python/C integration is quite common
- WARP is implemented in a mixture of Python and Fortran
- It's easy to use
 - Rapid turnaround: no intermediate compile and link steps as in C or C++
 - Python programs are compiled automatically to an intermediate form called *bytecode*, which the interpreter then reads
 - This gives Python the development speed of an interpreter without the performance loss inherent in purely interpreted languages
- It's easy to learn
 - Structure and syntax are pretty intuitive and easy to grasp

2. Python functions

- Built-in functions: these functions pre-defined and are always available.
- Functions defined in modules: these functions are pre-defined in particular modules and can only be used when the corresponding module is imported.
- User defined functions: these are defined by the programmer.
- Python documentation lists 80 built-in functions at:
<http://docs.python.org/library/functions.html>
- The modules example are: sys, math, time
- There are 100s of “standard” modules in Python:
 - Generation of random numbers and probability distributions
 - Accessing files and directories Y Web development
 - Network programming
 - Graphics, etc.
- A module is simply a file (just like the files that you have been creating your programs in) that contains related Python statements and function definitions.
- Programmers can define their own modules.
- There are 1000s of third-party modules available for Python.

2. Creating Modules, advantages of using modules.

- Modules allow logical organization of code
- Grouping related code into modules make it easier to understand and use.
- Modules are files containing Python definitions, statements and runnable code (ex. *name.py*)

Syntax : `import module1[,module2[...moduleN]`

- A module's definitions can be imported into other modules by using "import *name*"
- The module's name is available as a global variable value
- To access a module's functions, type "*name.function()*"
- When the interpreter encounters the import statement , it imports the module if it is present.
- Modules can contain executable statements along with function definitions
- Each module has its own private symbol table used as the global symbol table by all functions in the module
- Modules can import other modules
- Executing Modules
 - `python name.py <arguments>`
 - Runs code as if it was imported
 - Setting `__name__ == "__main__"` *the file can be used as a script and an importable module*
 - By adding following code at the end of your module
 - `if __name__ == "__main__":`
 - `import sys` `fib(int(sys.argv[1]))`
 - `>>>python fibo.py 50`
 - If the module is imported, the code is not run:
 - `>>>import fibo`
- Advantages of Modules -Minimization of dependencies is the goal, grouping logical code into modules make it easy to understand and use .
- It's possible to put several modules into a Package. A directory of Python code is said to be a package. A package is imported like a "normal" module. Each directory named within the path of a package must also contain a file named `__init__.py`, or else your package import will fail.

3. Functions used

-
- Splitlines Method - The method `splitlines()` returns a list with all the lines in string, optionally including the line breaks (if num is supplied and is true)
Syntax – `str.splitlines(num=string.count("\n"))`
- Split Method- The method `split()` returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified), optionally

- limiting the number of splits to num. Syntax
 – `str.split(str=" ", num=string.count(str))`
- Join Method-The method `join()` returns a string in which the string elements of sequence have been joined by str separator. Syntax
 –`str.join(sequences)`
- Len Method -The method `len()` returns the length of the string.
 Syntax – `len(str)`
- Find Method - It determines if string str occurs in string, or in a substring of string if starting index beg and ending index end are given. Syntax-
`str.find(str, beg=0, end=len(string))`
- Open Function - Before reading or writing a file, you have to open it using Python's built-in `open()` function. This function creates a file object, which would be utilized to call other support methods associated with it.
 Syntax – `fileobject=open(filename,[access mode],[buffering])`
- Close Function- The `close()` method of a file object flushes any unwritten information and closes the file object, after which no more writing can be done. Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the `close()` method to close a file. Syntax –
`fileobject.close()`
- Write Method - The `write()` method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.
 Syntax – `fileobject.write(string)`
- Read Method - The `read()` method reads a string from an open file. It is important to note that Python strings can have binary data apart from text data.
 Syntax – `fileobject.read([count])`

Algorithm:

1. Start
2. Accept the file name at command prompt (as command line argument)
3. Open file specified at command prompt to read the data
4. Create new file
5. named by appending the word 'scrabble' to input filename to write the data after scrabble
6. For each line in file
 - a. Read line and split it into words with blank spaces
 - b. Read each word and apply conditions
 - i. Words less than or equal to 3 characters need not be scrabbled.
 - ii. Don't scrabble first and last character.
 - iii. Special characters are to be maintained as it is. Special characters like (. , ; , ? , , , !)
 - c. Merge the words after applying conditions and make a sentence
 - d. Write the sentence in the new file

7. Close both the files.

UML Diagram – Draw UML diagram relevant to design.

Conclusion: Thus, we have successfully studied basics, file handling functions, library functions etc.

Experiment No	2
Title	Build the Hangman Game using Python.
Roll No.	
Date	
Sign of staff	

Problem Statement:

Write a python program for building Hangman Game.

Aim:

Create a Hangman Game using Python.

Objective:

- To make user familiar with Hangman Game.
- To use python dictionary or list.
- To import standard modules and use inbuilt functions.

Input – Prompt user to enter single character for guessing word.

Output– Identification of Correct/Wrong word i.e. to win/loss the game.

Theory:**3. Concept of Hangman Game.**

- The origins of Hangman are obscure meaning not discovered, but it seems to have arisen in Victorian times, says Tony Augarde, author of *The Oxford Guide to Word Games*. The game is mentioned in Alice Bertha Gomme's "Traditional Games" in 1894 under the name "Birds, Beasts and Fishes." The rules are simple; a player writes down the first and last letters of a word and another player guesses the letters in between. In other sources, the game is called "Gallows", "The Game of Hangin", or "Hanger".
- Hangman is a paper and pencil guessing game for two or more players. One player thinks of a word, phrase or sentence and the other tries to guess it by suggesting letters or numbers, within a certain number of guesses.


Rules:


- The word to guess is represented by a row of dashes, representing each letter of the word.
- In most variants, proper nouns, such as names, places, and brands, are not allowed. Slang words, sometimes referred to as informal or shortened words, are also not allowed.
- If the guessing player suggests a letter which occurs in the word, the other player writes it in all its correct positions.
- If the suggested letter or number does not occur in the word, the other player draws one element of a hanged man stick figure as a tally mark.
- The player guessing the word may, at any time, attempt to guess the whole word. If the word is correct, the game is over and the guesser wins.


- Otherwise, the other player may choose to penalize the guesser by adding an element to the diagram. On the other hand, if the other player makes enough incorrect guesses to allow his opponent to complete the diagram, the game is also over, this time with the guesser losing.
- However, the guesser can also win by guessing all the letters or numbers that appears in the word, thereby completing the word, before the diagram is completed.


4. Hangman Game Example:

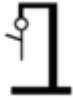
The following example illustrates a player trying to guess the word hangman using a strategy based solely on letter frequency.

0 
 Word: _ _ _ _ _
 Guess: E
 Misses:

1 
 Word: _ _ _ _ _
 Guess: T
 Misses: E

2 
 Word: _ _ _ _ _
 Guess: A
 Misses: e,t

3 
 Word: _ A _ _ _ A _
 Guess: O
 Misses: e,t



4

Word: _ **A** _ _ _ **A** _

Guess: I

Misses: e,o,t



5

Word: _ **A** _ _ _ **A** _

Guess: S

Misses: e,i,o,t



6

Word: _ **A** _ _ _ **A** _

Guess: N

Misses: e,i,o,s,t

7



8

Word: **H A N** _ _ **A N**

Guess: R

Misses: e,i,o,s,t



9

Word: **H A N** _ _ **A N**

Guess:

Misses: e,i,o,r,s,t

The guessing player has lost this game as the diagram had been completed before all the letters were guessed.

Algorithm:

8. Start
9. Identify different words with different levels.
10. Store the words in dictionary (Here, you can use different level of dictionaries such as, easy, medium and hard).
11. Use random.choice function for fetching any random word from these three dictionaries and then user can guess that word.
12. If wrong guesses are entered then draw hanging man using different symbols.
13. If correct choices are entered then You won the game (You can give 10 choices for guessing the letters).

UML Diagram – Draw UML diagrams according to program.

Conclusion: Thus, we have successfully studied python basics, functions and implemented hangman game.

Experiment No.	3
Title	Write a python code that loads any dataset(.csv) and plot the graph
Roll No.	
Date	
Sign of staff	

Problem Statement:

Write a python program that Read the given CSV file & perform following operations.

1. Draw the Pie Chart of overall result analysis for Distinction, First, Higher Second, Second, Pass and Fail class.
2. Draw the Subject wise Bar Chart. (Distinction, First, Higher Second , Second, Pass and Fail class)
3. Tabular representation of subject wise Marks. (Sub, Max, Min, Avg. & Fail)
4. Tabular representation of top five students of the Department.(Seat No., Name, SGPA)
5. Tabular representation of Subject wise Result. (Sr. No., Name of the subject, No. of student appeared, No. of students passed and passing percentage)

Aim:

To read the CSV file and plot the charts mentioned in problem statement.

Objective:

- To make user familiar with pythons NUMPY package.
- To make user familiar with pythons PANDAS package.
- To make user familiar with pythons MATPLOTLIB package.
- To perform mathematical operations over data using inbuilt library functions.
- To use file access methods
- To create module and understand importance of it.

Input :

Student result data in CSV format.

Output :

Visualize different charts, graphs and tables of result analysis.

Theory:**1. Numpy:**

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

How to import it:

import numpy as np

2. matplotlib.pyplot:

It is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that “axes” here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis).

How to import it:

```
importmatplotlib.pyplot as plt
```

3.matplotlib.pyplot.xlabel:

This sets the x axis label of the current axis.

Syntax:

```
matplotlib.pyplot.xlabel(s, *args, **kwargs)
```

4.matplotlib.pyplot.ylabel:

This sets the y axis label of the current axis.

Syntax:

```
matplotlib.pyplot.ylabel(s, *args, **kwargs)
```

5. matplotlib.pyplot.title:

This sets a title of the current axes. Set one of the three available axes titles. The available titles are positioned above the axes in the center, flush with the left edge, and flush with the right edge.

Syntax:

```
matplotlib.pyplot.title(s, *args, **kwargs)
```

6.matplotlib.pyplot.show:

Display a figure. When running in ipython with its pylab mode, display all figures and return to the ipython prompt.

In non-interactive mode, display all figures and block until the figures have been closed; in interactive mode it has no effect unless figures were created prior to a change from non-interactive to interactive mode (not recommended). In that case it displays the figures but does not block.

A single experimental keyword argument, `block`, may be set to `True` or `False` to override the blocking behavior described above.

Syntax:

```
matplotlib.pyplot.show(*args, **kw)
```

Algorithm:

14. Start
15. Import pandas as pd
16. Import csv
17. Import matplotlib.pyplot as plt(I.e. importing function for plotting of graph)
18. Import numpy
19. Open .csv file in read mode
20. Plot x,y labels for graph
21. Specify title of the graph to be displayed
22. Show output to user
23. End

Conclusion:

Thus, we have successfully performed plotting of graph for given dataset.

Experiment No	4
Title	Python code that loads any dataset for doing some basic data cleaning
Roll No.	
Date	
Sign of staff	

Problem Statement:

Write a Python code that loads any dataset and does some basic data cleaning, Add component on dataset.

RULES:

1. Handling NaN values using Numpy/Dataframe
2. Removing unwanted symbols like , , @, etc from data.
3. Renaming the column header with suitable name using dataframe method
4. Add/Map student information with roll numbers using dataframes.
5. Visualize following graphs using Matplotlib tables
 - i. Visualize top 5 students from each subject.
 - ii. Division wise result analysis (Overall)
 - iii. Division wise result analysis (Subject wise)

Aim:

To do some basic data cleaning in .csv file according to rules mentioned in problem statement.

Objective:

- To clean up a .csv file and adding a component to the dataset.
- Performing Statistical operation on python's NUMPY array.
- Adding and updating and modifying components on python's PANDAS Dataframe.
- To make user familiar with python's MATPLOTLIB package.
- To represent the analytical statistics in tabular representation.
- To perform mathematical operations over data using inbuilt library functions.
- To use file access methods
- To create module and understand importance of it.

Input – Prompt user to enter .csv file name that needs to be cleaned.

Output– An .csv file which has cleaned up.

Theory:

The following theory explains the whole procedure in step by step manner,

- Add a specific table column to one output file.
- Use subclass for drawing the graphs which takes parameters on row, column and position.
- Remove the commas from the specified column of the table in the database.
- The tuples with no value or NaN (Not a Number) values must be cleared.
- The specified columns must be copied and pasted through the code. Use the code for this operation.

- The new file created must contain the necessary rows and columns with values.
- This includes
 - data cleaning and adding a component to the database
 - handling of NaN (Not a Number) values using NumPy
 - removing unwanted words like (and, or, like, the, in, etc)
 - removing unwanted symbols like (. , , ! , @ , # , & , etc)
 - removing the column header with suitable name using DF
- Add roll_no column and map with student name.
- Visualize groups:
 - Visualize top 5 students of each division
 - Division wise overall result analysis (subplot)
 - Division wise subject analysis (subplot)
 - Subclade name, grade points, card points, marks, credits

Algorithm:

24. Start
25. Import CSV
26. Open the .csv file in read mode
27. Open the output.csv file in write mode
28. Skip the header rows of input file
29. Shift the cursor to the next row
30. Replace the other rows with empty values
31. Write this line/row in the output.csv file
32. Close input and output files
33. Open the output file in read mode
34. Read the contents
35. Stop

Conclusion: Thus, python code for loading the .csv dataset and adding a component as well as cleaning the input is successfully implemented and written.

Experiment No	5
Title	R basics (Study of all R objects)
Roll No.	
Date	
Sign of staff	

Problem Statement:

- Started with R installation.
- Create different R Objects and perform basic statistics on it.

Aim:

Getting start with R installation, R objects and basic statistics

Objective:

- To access the R website, download and install R
- To Expand R by installing R packages
- Master the use of the R interactive environment
- To make user familiar with R syntax
- understand, manipulate and explore different types of R objects such as vectors, matrices and data frames

Theory:

5. Introduction to R

- R is a system for statistical computation and graphics. It provides, among other things, a programming language, high level graphics, interfaces to other languages and debugging facilities.
- The R language is a dialect of S which was designed in the 1980s and has been in widespread use in the statistical community since. Its principal designer, John M. Chambers, was awarded the 1998 ACM Software Systems Award for S.
- The language syntax has a superficial similarity with C, but the semantics are of the FPL (functional programming language) variety with stronger affinities with Lisp and APL. In particular, it allows “computing on the language”, which in turn makes it possible to write functions that take expressions as input, something that is often useful for statistical modeling and graphics.
- The design of the language contains a number of fine points and common pitfalls which may surprise the user. Most of these are due to consistency considerations at a deeper level. There are also a number of useful shortcuts and idioms, which allow the user to express quite complicated operations succinctly. Many of these become natural once one is familiar with the underlying concepts.
- **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.

4. R objects

Like C and java in R, the variables are not declared as some data type. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable. There are many types of R-objects. The frequently used ones are:

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

a) Vector

The simplest of these objects is the vector object and there are six data types of these atomic vectors, also termed as six classes of vectors. The other R-Objects are built upon the atomic vectors.

- i) **character:** "a", "swc"
- ii) **numeric:** 2, 15.5
- iii) **integer:** 2L (the L tells R to store this as an integer)
- iv) **logical:** TRUE, FALSE
- v) **complex:** 1+4i (complex numbers with real and imaginary parts)

When you want to create vector with more than one element, you should use c() function which means to combine the elements into a vector.

```
# Create a vector.
apple<- c('red','green',"yellow")
print (apple)
```

```
# Get the class of the vector.
print(class(apple))
```

When we execute the above code, it produces the following result:
[1] "red" "green" "yellow" [1] "character"

b) List

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
# Create a list.
list1 <- list(c(2,5,3),21.3,sin)
```

```
# Print the list.  
print(list1)
```

When we execute the above code, it produces the following result:

```
[[1]]  
[1] 2 5 3  
[[2]]  
[1] 21.3 [[3]]  
function(x) .Primitive("sin")
```

c) Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.  
M = matrix( c('a','a','b','c','b','a'), nrow=2,ncol=3,byrow = TRUE)  
print(M)
```

When we execute the above code, it produces the following result:

```
[,1] [,2] [,3]  
[1,] "a" "a" "b" [2,] "c" "b" "a"
```

d) Arrays

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimension. In the below example we create an array with two elements which are 3x3 matrices each.

```
# Create an array.  
a <- array(c('green','yellow'),dim=c(3,3,2))  
print(a)
```

When we execute the above code, it produces the following result:

```
,, 1  
[,1] [,2] [,3]  
[1,] "green" "yellow" "green"  
[2,] "yellow" "green" "yellow"  
[3,] "green" "yellow" "green"  
,, 2  
[,1] [,2] [,3]  
[1,] "yellow" "green" "yellow"  
[2,] "green" "yellow" "green"  
[3,] "yellow" "green" "yellow"
```

e) Factors

Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling. Factors are created using the factor() function. Then levels functions gives the count of levels.

```
# Create a vector.
apple_colors<- c('green','green','yellow','red','red','red','green')

# Create a factor object.
factor_apple<- factor(apple_colors)

# Print the factor.
print(factor_apple)
print(nlevels(factor_apple))
```

When we execute the above code, it produces the following result:

```
[1] greengreen yellow red redred yellow green
Levels: green red yellow
```

```
# applying the nlevels function we can know the number of distinct values
[1] 3
```

f) Data Frames

Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length. Data Frames are created using the data.frame() function.

```
# Create the data frame.
BMI <- data.frame(

  gender = c("Male", "Male", "Female"),
  height = c(152, 171.5, 165),
  weight = c(81, 93, 78), Age = c(42, 38, 26)

)

print(BMI)
```

When we execute the above code, it produces the following result:

	gender	height	weight	Age
1	Male	152.0	81	42
2	Male	171.5	93	38

3 Female 165.0 78 26

5. Basic Statistics

a) Mean

In R, a mean can be calculated on an isolated variable via the `mean(VAR)` command, where `VAR` is the name of the variable whose mean you wish to compute. Alternatively, a mean can be calculated for each of the variables in a dataset by using the `mean(DATAVAR)` command, where `DATAVAR` is the name of the variable containing the data. The code sample below demonstrates both uses of the mean function.

```
1. > #calculate the mean of a variable with mean(VAR)
2. > #what is the mean Age in the sample?
3. > mean(Age)
4. [1] 32.3
5. > #calculate the mean of all variables in a dataset with mean(DATAVAR)
6. > #what is the mean of each variable in the dataset?
7. > mean(dataset)
8. Age..... Income
9. 32.3..... 34000.0
```

b) Standard Deviation

Within R, standard deviations are calculated in the same way as means. The standard deviation of a single variable can be computed with the `sd(VAR)` command, where `VAR` is the name of the variable whose standard deviation you wish to retrieve. Similarly, a standard deviation can be calculated for each of the variables in a dataset by using the `sd(DATAVAR)` command, where `DATAVAR` is the name of the variable containing the data. The code sample below demonstrates both uses of the standard deviation function.

```
1. > #calculate the standard deviation of a variable with sd(VAR)
2. > #what is the standard deviation of Age in the sample?
3. > sd(Age)
4. [1] 19.45602
5. > #calculate the standard deviation of all variables in a dataset with sd(DATAVAR)
6. > #what is the standard deviation of each variable in the dataset?
7. > sd(dataset)
8. Age..... Income
9. 19.45602.... 32306.10175
```

c) Range

Minimum and Maximum

Keeping with the pattern, a minimum can be computed on a single variable using the `min(VAR)` command. The maximum, via `max(VAR)`, operates identically. However, in contrast to the mean and standard deviation functions, `min(DATAVAR)` or `max(DATAVAR)` will retrieve the minimum or maximum value from the entire dataset, *not from each individual variable*. Therefore, it is recommended that minimums

and maximums be calculated on individual variables, rather than entire datasets, in order to produce more useful information. The sample code below demonstrates the use of the min and max functions.

1. `> #calculate the min of a variable with min(VAR)`
2. `> #what is the minimum age found in the sample?`
3. `> min(Age)`
4. `[1] 5`
5. `> #calculate the max of a variable with max(VAR)`
6. `> #what is the maximum age found in the sample?`
7. `> max(Age)`
8. `[1] 70`

Range

The range of a particular variable, that is, its maximum and minimum, can be retrieved using the `range(VAR)` command. As with the min and max functions, using `range(DATAVAR)` is not very useful, since it considers the entire dataset, rather than each individual variable. Consequently, it is recommended that ranges also be computed on individual variables. This operation is demonstrated in the following code sample.

1. `> #calculate the range of a variable with range(VAR)`
2. `> #what range of age values are found in the sample?`
3. `> range(Age)`
4. `[1] 5....70`

d) Percentiles

Values from Percentiles (Quantiles)

Given a dataset and a desired percentile, a corresponding value can be found using the `quantile(VAR, c(PROB1, PROB2,...))` command. Here, VAR refers to the variable name and PROB1, PROB2, etc., relate to probability values. The probabilities must be between 0 and 1, therefore making them equivalent to decimal versions of the desired percentiles (i.e. 50% = 0.5). The following example shows how this function can be used to find the data value that corresponds to a desired percentile.

1. `> #calculate desired percentile values using quantile(VAR, c(PROB1, PROB2,...))`
2. `> #what are the 25th and 75th percentiles for age in the sample?`
3. `> quantile(Age, c(0.25, 0.75))`
4. `25%..... 75%`
5. `17.75..... 44.25`

Note that `quantile(VAR)` command can also be used. When probabilities are not specified, the function will default to computing the 0, 25, 50, 75, and 100 percentile values, as shown in the following example.

1. `> #calculate the default percentile values using quantile(VAR)`

2. `> #what are the 0, 25, 50, 75, and 100 percentiles for age in the sample?`
3. `> quantile(Age)`
4. 0%..... 25%..... 50%..... 75%..... 100%
5. 5.00... 17.75..... 30.00... 44.25..... 70.00

Percentiles from Values (Percentile Rank)

In the opposite situation, where a percentile rank corresponding to a given value is needed, one has to devise a custom method. To begin, consider the steps involved in calculating a percentile rank.

1. count the number of data points that are at or below the given value
2. divide by the total number of data points
3. multiply by 100

From the preceding steps, the formula for calculating a percentile rank can be derived: $\text{percentile rank} = \text{length}(\text{VAR}[\text{VAR} \leq \text{VAL}]) / \text{length}(\text{VAR}) * 100$, where VAR is the name of the variable and VAL is the given value. This formula makes use of the length function in two variations. The first, $\text{length}(\text{VAR}[\text{VAR} \leq \text{VAL}])$, counts the number of data points in a variable that are below the given value. Note that the " \leq " operator can be replaced with other combinations of the $<$, $>$, and $=$ operators, supposing that the function were to be applied to different scenarios. The second, $\text{length}(\text{VAR})$, counts the total number of data points in the variable. Together, they accomplish steps one and two of the percentile rank computation process. The final step is to multiply the result of the division by 100 to transform the decimal value into a percentage. A sample percentile rank calculation is demonstrated below.

1. `> #calculate the percentile rank for a given value using the custom formula:`
`length(VAR[VAR <= VAL]) / length(VAR) * 100`
2. `> #in the sample, an age of 45 is at what percentile rank?`
3. `> length(Age[Age <= 45]) / length(Age) * 100`
4. `[1] 75`

e) Summary

A very useful multipurpose function in R is `summary(X)`, where X can be one of any number of objects, including datasets, variables, and linear models, just to name a few. When used, the command provides summary data related to the individual object that was fed into it. Thus, the summary function has different outputs depending on what kind of object it takes as an argument. Besides being widely applicable, this method is valuable because it often provides exactly what is needed in terms of summary statistics. A couple examples of how `summary(X)` can be used are displayed in the following code sample. I encourage you to use the summary command often when exploring ways to analyze your data in R. This function will be revisited throughout the R Tutorial Series.

1. `> #summarize a variable with summary(VAR)`
2. `> summary(Age)`

The output of the preceding summary is pictured below.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
5.00	17.75	30.00	32.30	44.25	70.00

1. > #summarize a dataset with summary(DATAVAR)
2. > summary(dataset)

The output of the preceding summary is pictured below.

Age		Income	
Min.	: 5.00	Min.	: 0
1st Qu.:	17.75	1st Qu.:	0
Median :	30.00	Median :	27500
Mean :	32.30	Mean :	34000
3rd Qu.:	44.25	3rd Qu.:	56250
Max.	:70.00	Max.	:100000

Conclusion: Thus, we have successfully studied R installation, R objects, basic statistics etc.

Experiment No	6
Title	Using R for data Preprocessing, exploratory analysis, visualization.
Roll No.	
Date	
Sign of staff	

Problem Statement:

Write a python program for data preprocessing, exploratory analysis and Visualization.

Aim:

To apply various pre-processing functions on any dataset and analyze or visualize that data by using various functions of R.

Objective:

- To make user familiar with data preprocessing functions.
- To use different R functions for exploratory analysis and visualization.

Input – Any dataset

Output– Visualize the output by using functions like histogram, barplotetc

Theory:

Data Preprocessing

- Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.
- Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.
- Data preprocessing is used database-driven applications such as customer relationship management and rule-based applications (like neural networks).

Data goes through a series of steps during preprocessing:

- Data Cleaning: Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.
- Data Integration: Data with different representations are put together and conflicts within the data are resolved.
- Data Transformation: Data is normalized, aggregated and generalized.
- Data Reduction: This step aims to present a reduced representation of the data in a data warehouse.
- Data Discretization: Involves the reduction of a number of values of a continuous attribute by dividing the range of attribute intervals.

Summary of Transform Methods:

Below is a quick summary of all of the transform methods supported in the method argument of the *preProcess()* function in caret.

- “zv”: remove attributes with a zero variance (all the same value).
- “nzv”: remove attributes with a near zero variance (close to the same value).
- “center”: subtract mean from values.
- “scale”: divide values by standard deviation.
- “range”: normalize values.
- “pca”: transform data to the principal components.
- “ica”: transform data to the independent components.
- “spatialSign”: project data onto a unit circle.

The following sections will demonstrate some of the more popular methods.

Scale:

The scale transform calculates the standard deviation for an attribute and divides each value by that standard deviation.

```
1 #load libraries
2 library(caret)
3 #load the dataset
4 data(iris)
5 #summarize data
6 summary(iris[,1:4])
7 #calculate the pre-process parameters from the dataset
8 preprocessParams <- preProcess(iris[,1:4], method=c("scale"))
9 #summarize transform parameters
10 print(preprocessParams)
11 #transform the dataset using the parameters
12 transformed <- predict(preprocessParams, iris[,1:4])
13 #summarize the transformed dataset
14 summary(transformed)
```

Center:

The center transform calculates the mean for an attribute and subtracts it from each value

```
1 #load libraries
2 library(caret)
3 #load the dataset
4 data(iris)
5 #summarize data
6 summary(iris[,1:4])
7 #calculate the pre-process parameters from the dataset
8 preprocessParams <- preProcess(iris[,1:4], method=c("center"))
9 #summarize transform parameters
10 print(preprocessParams)
11 #transform the dataset using the parameters
12 transformed <- predict(preprocessParams, iris[,1:4])
13 #summarize the transformed dataset
14 summary(transformed)
```

Standardize:

Combining the scale and center transforms will standardize your data. Attributes will have a mean value of 0 and a standard deviation of 1.

```
1 #load libraries
2 library(caret)
3 #load the dataset
4 data(iris)
5 #summarize data
6 summary(iris[,1:4])
7 #calculate the preprocess parameters from the dataset
8 preprocessParams <- preProcess(iris[,1:4], method=c("center", "scale"))
9 #summarize transform parameters
10 print(preprocessParams)
11 #transform the dataset using the parameters
12 transformed <- predict(preprocessParams, iris[,1:4])
13 #summarize the transformed dataset
14 summary(transformed)
```

Exploratory Analysis

Why do we use exploratory graphs in data analysis?

- Understand data properties
- Find patterns in data
- Suggest modeling strategies
- “Debug” analyses
- When we are dealing with a single data point, let’s say temperature or, wind speed, or age, the following techniques are used for the initial exploratory data analysis.
- **Five-number summary**- This essentially provides information about the minimum value, 1st quartile, median, 3rd quartile and the maximum.

Functions used for plotting data:

- Boxplot(): Gives you boxplot
- Barplot(): It plot bars according to your data.
- Hist(): It plots histogram.
- Pie(): plots pie chart
- Plot(): generic x-y plotting

Visualization

We can Visualize data by using above R graphics functions.

Algorithm:

36. Start
37. Load any standard dataset such as, iris, weather etc.
38. Display the contents of dataset.
39. Apply preProcess function on columns with different attributes.
40. Analyze and visualize the dataset using different functions such as, histogram, barplot etc.
41. End

UML Diagram – Draw UML diagrams according to program.

Conclusion: Thus, we have successfully studied data pre-processing, exploratory analysis and visualization.

Experiment No.	7
Title	Using R for Correlation and regression analysis.
Roll No.	
Date	
Sign of staff	

Problem Statement:

1. To perform Correlation analysis Read the R inbuilt dataset called mtcars dataset and store it in dataframes.
2. Read two variables from dataset mpg and wt and find correlation between them.
3. Plot a scattered graph for two variable data.
4. Perform correlation and analyse the relation between them.
5. To perform regression analysis Read the R inbuilt dataset called airquality dataset and store it in dataframes.
6. Apply Linear Regression and construct the model.
7. Predict the Temperature.

Aim: Using R for Correlation and regression analysis.

Objective:

- Computes the correlation between mpg and wt variables in mtcars data set.
- To predict the temperature for a particular month using solar radiation, ozone and wind data in *airquality data set*

Input : .CSV File for Correlation(*mtcars*) and Regression(MASS - *airquality*) Analysis.

Output :

1. Correlation coefficient Between Variables mpg and wt (1,-1, or 0).
2. Predicted temperature Value.

Theory:

Correlation Coefficient

The **correlation coefficient** of two variables in a data set equals to their [covariance](#) divided by the product of their individual [standard deviations](#). It is a normalized measurement of how the two are linearly related.

$$r_{xy} = \frac{S_{xy}}{S_x S_y}$$

Formally, the **sample correlation coefficient** is defined by the following formula, where s_x and s_y are the sample standard deviations, and s_{xy} is the sample covariance.

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

If the correlation coefficient is close to 1, it would indicate that the variables are positively linearly related and the [scatter plot](#) falls almost along a straight line with positive slope. For -1, it indicates that the variables are negatively linearly related and the scatter plot almost falls along a straight line with negative slope. And for zero, it would indicate a weak linear relationship between the variables.

- Use the function **cor.test(x,y)** to analyze the correlation coefficient between two variables and to get significance level of the correlation.
- Three possible correlation methods using the function **cor.test(x,y)**: pearson, kendall, spearman.

How to Perform Correlation

R can perform correlation with the `cor()` function. Built-in to the base distribution of the program are three routines, for Pearson, Kendal and Spearman Rank correlations.

Step 1: Arrange your data in a .CSV file

Use a column for each variable and give it a meaningful name. Don't forget that variable names in R can contain letters and numbers but the only punctuation allowed is a period.

Step 2 :-

Read your data file into memory and give it a sensible name.

Step 3:-

Attach your data set so that the individual variables are read into memory.

To get the correlation coefficient you type

```
>cor.test(var1, var2, method = "method")
```

The default method is "pearson"	cor.p	= cor.test(var1, var2)
If you specify "spearman" you will get the spearman correlation coefficient	cor.s	= cor.test(var1, var2, method = "spearman")

To see a summary of your correlation test

```
>cor.s
```

Correlation Step by Step

First create your data file. Use a spreadsheet and make each column a variable. Each row is a replicate. The first row should contain the variable names. Save this as a .CSV file

Read the data into R and save as some name	your.data	= read.csv(file.choose())
Allow the factors within the data to be accessible to R		attach(your.data)
Decide on the method, run the correlation and assign the result to a new variable. Methods are "pearson" (default), "kendal" and "spearman"	your.cor	= cor(var1, var2, method = "pearson")
Have a look at the resulting correlation coefficient		your.cor
Perform a pairwise correlation on all the variables in the data set. Decide on the method ("pearson" (default), "kendal" and "spearman")	cor.mat	= cor(your.data, method = "pearson")
have a look at the resulting correlation matrix		cor.mat
To evaluate the statistical significance of your correlation decide on the appropriate method (pearson is the default, see above), assign a variable and run the test	your.cor	cor.test(var1, var2, method="spearman")
Have a look at the result of your significance test		your.cor
Plot a graph of the two variables from your correlation. pch=21 plots an open circle, pch=19 plots a solid circle. Try other values.		plot(x.var, y.var, xlab="x-label", ylab="y-label", pch=21))
Add a line of best fit (if appropriate)		abline(lm(y.var ~ x.var))

Regression :-

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

Linear Regression

Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph.

$$y = ax + b$$

Following is the description of the parameters used –

- **y** is the response variable.
- **x** is the predictor variable.
- **a** and **b** are constants which are called the coefficients.

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is –

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the **lm()** functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
- To predict the weight of new persons, use the **predict()** function in R.

Algorithm:

1. Correlation :

Step 1 : Import your data into R

1. **Prepare your data**
2. **Save your data** in an external .txt or .csv files
3. **Import your data into R** as follow:

Step 2 : Visualize your data using scatter plots

Step 3 : Perform Preliminary test to check the test assumptions

1. **Is the covariation linear?** Yes, form the plot above, the relationship is linear. In the situation where the scatter plots show curved patterns, we are dealing with nonlinear association between the two variables.
2. **Are the data from each of the 2 variables (x, y) follow a normal distribution?**
 - Use Shapiro-Wilk normality test → R function: **shapiro.test()**
 - and look at the normality plot → R function: **ggpubr::ggqqplot()**

Step 4 : Performing Pearson correlation test

Step 5 : Interpretation of the result

Step 6 : Access to the values returned by cor.test() function

The function **cor.test()** returns a list containing the following components:

- **p.value**: the p-value of the test
- **estimate**: the correlation coefficient

Step 7 : Interpret correlation coefficient

Correlation coefficient is comprised between **-1** and **1**:

2. Regression :

Step 1 : Carry out the experiment of gathering a sample of observed values of height and corresponding weight.

Step 2 : Create a relationship model using the **lm()** functions in R.

Step 3 : Find the coefficients from the model created and create the mathematical equation using these

Step 4 : Get a summary of the relationship model to know the average error in prediction.
Also called **residuals**.

Step 5 : To predict the weight of new persons, use the **predict()** function in R.

Conclusion:

- **-1** indicates a strong **negative correlation** : this means that every time **x increases, y decreases** (left panel figure)
- **0** means that there is no **association** between the two variables (x and y) (middle panel figure)
- **1** indicates a strong **positive correlation**

The regression algorithm assumes that the data is normally distributed and there is a linear relation between dependent and independent variables. It is a good mathematical model to analyze relationship and significance of various variables.

Experiment No	8
Title	Data analysis case study using R for readily available data set using any one machine learning algorithm
Roll No.	
Date	
Sign of staff	

Problem Statement:

Data analysis case study using R for readily available data set using any one machine learning algorithm

Aim:

To read any dataset, analyse it and then apply any suitable machine learning algorithm.

Objective:

- To understand Machine learning.
- To study different machine learning algorithms.
- To study the suitable algorithm for selected dataset.
- To virtualize suitable output.

Input – Dataset

Output– Virtualize the output as per the dataset and algorithm selected.

Theory:

1. Introduction to Machine Learning

Machine learning is a branch in computer science that studies design of algorithms that can learn. Typical machine learning tasks are concept of function learning and find predictive patterns. These tasks are learned through available data that were observed through experience or instruction. The ultimate goal is to improve the learning in such way that it becomes automatic, so that human like ourselves don't need to interface any more. Simple and efficient tool for data mining and data analysis. Accessible to everybody and reusable in various context. Built on numpy, scipy, matplotlib. open source commercially usable, BSD license.

a) Machine learning types

- supervised learning**, which trains a model on known input and output data so that it can predict future outputs, and
- unsupervised learning**, which finds hidden patterns or intrinsic structures in input data.

b) Supervised machine learning

The aim of supervised machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

- Classification** techniques predict categorical responses, for example, whether an email is genuine or spam, or whether a tumor is cancerous or benign. Classification models

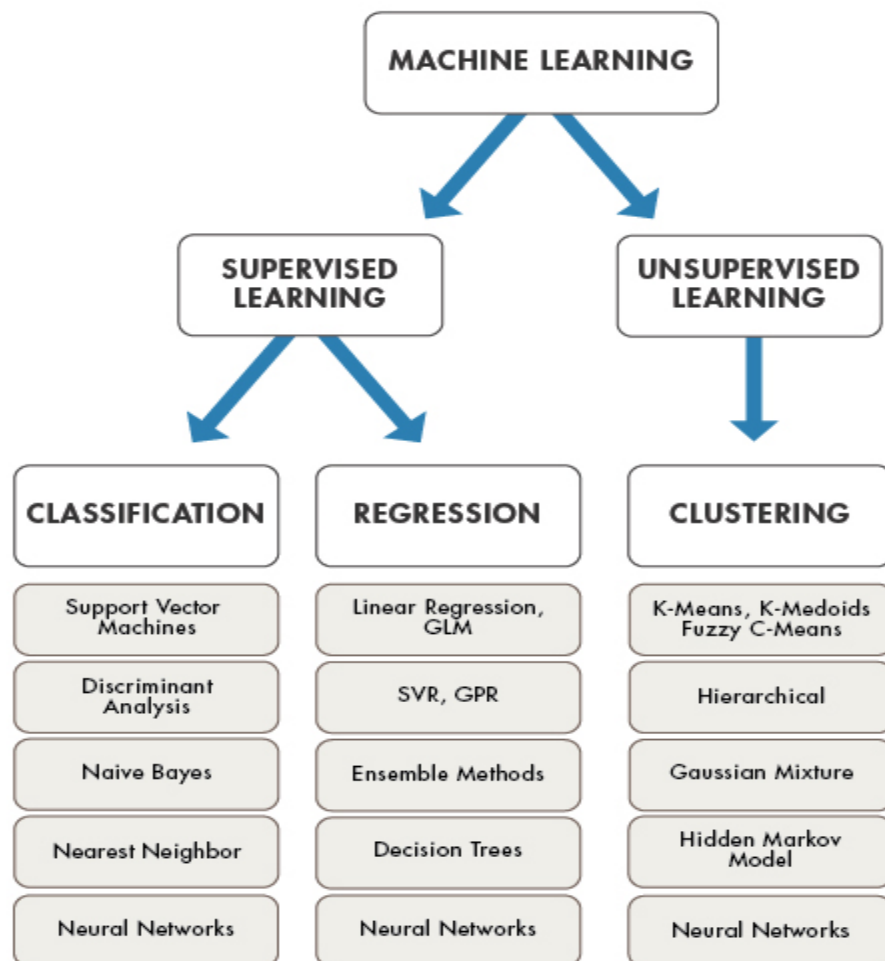
classify input data into categories. Typical applications include medical imaging, image and speech recognition, and credit scoring.

- ii. **Regression** techniques predict continuous responses, for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.

c) Unsupervised learning

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labelled responses.

- i. **Clustering** is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data. Applications for clustering include gene sequence analysis, market research, and object recognition.
- ii. **Association** is a rule learning problem where you want to discover rules that describe large portions of your data, such as people that buy x tend to buy y.



Algorithm in Machine Learning

2. K-means clustering

k-means clustering is a method of vector quantization originally from signal processing that is popular for cluster analysis in data mining. K-means is one of the most widely used clustering algorithms since partitioning algorithms are preferred more in pattern recognition due to

the available data. The main reasons for its popularity are ease in implementation, efficiency, simplicity and empirical success.

K-means clustering aims to partition 'n' observations into k cluster in which each observation belongs to the clusters with the nearest mean, serving as a prototype of the cluster.

Let us consider a set of n d-dimensional points $X = \{x_i\}$, $i = 1, \dots, n$ that needs to be clustered into a set of k clusters, $C = \{c_k, k=1, \dots, K\}$; K-means algorithm minimizes the squared error between the empirical mean of a cluster and the points in the cluster. Let the cluster c_k has the average u_k , then squared error between c_k and u_k is defined as

$$J(c_k) = \sum_{x_i \in c_k} \|x_i - u_k\|^2$$

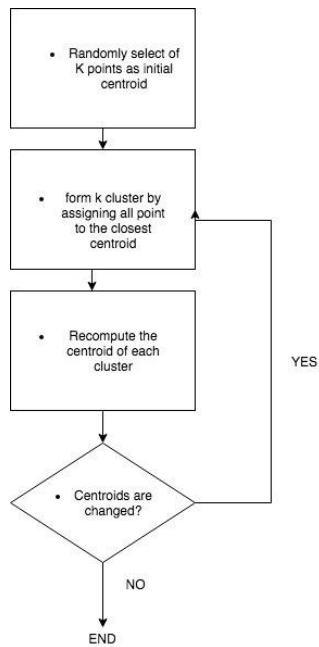
K-means minimizes the sum of the squared error over all K clusters,

$$J(c_k) = \sum_{k=1}^k \sum_{x_i \in c_k} \|x_i - u_k\|^2$$

K-means Algorithm

Suppose that we have n sample feature vectors x_1, x_2, \dots, x_n all from the same class, and we know that they fall into k compact clusters, $k < n$. Let m_i be the mean of the vectors in cluster i. If the clusters are well separated, we can use a minimum-distance classifier to separate them. That is, we can say that x is in cluster i if $\|x - m_i\|$ is the minimum of all the k distances. This suggests the following procedure for finding the k means:

- Make initial guesses for the means m_1, m_2, \dots, m_k
- Until there are no changes in any mean
 - Use the estimated means to classify the samples into clusters
 - For i from 1 to k
 - Replace m_i with the mean of all of the samples for cluster i
 - end_for
- end_until



Flowgraph of K-means Clustering

Algorithm:

42. Start
43. Read .csv file data.
44. Apply k-means function, Specify number of clusters.
45. Plot clusters as returned by the function.
46. Stop.

UML Diagram – Draw UML diagram relevant to design.

Conclusion: Thus, we have successfully studied machine learning concept, and applied K-means clustering algorithm and plotted the clusters.