MONITORING STUDENTS ACADEMIC PERFORMANCE USING R

A project report submitted to
Rashtrasant Tukadoji Maharaj Nagpur University
In partial fulfillment for the award of degree of

Bachelor of Engineering
In
Computer Science and Engineering

BY

HARSHAD CHAUREWAR [BE11CSU805]
and OTHERS

UNDER THE SUPERVISION OF

PROF. AARTI. M. KARANDIKAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR-13

DECEMBER – 2014

MONITORING STUDENTS ACADEMIC PERFORMANCE USING R

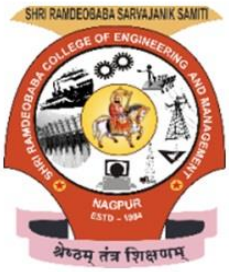
A project report submitted to
Rashtrasant Tukadoji Maharaj Nagpur University
In partial fulfillment for the award of degree of

Bachelor of Engineering
In
Computer Science and Engineering
BY

ABHISHEK DIXIT [BE12CSU707]
AJINKYA MAHAKALKAR [BE12CSU708]
CHANDRASHEKHAR BURADKAR [BE12CSU711]
HARSHAD CHAUREWAR [BE11CSU805]
SAMBHAV CHOPDA [BE12CSU712]

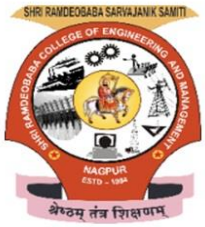
UNDER THE SUPERVISION OF

PROF. AARTI. M. KARANDIKAR



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR-13

DECEMBER – 2015



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR-13

CERTIFICATE

This is to certify that this is a bonafide record of project work entitled

MONITORING STUDENTS ACADEMIC PERFORMANCE USING R

Carried out by

HARSHAD CHAUREWAR [BE11CSU805]

of the Final Year **B. E. Computer Science and Engineering** during the academic year **2014-15** in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Engineering** offered by **RASHTRASANT TUKADOJI MAHARAJ NAGPUR UNIVERSITY, NAGPUR**

UNDER THE SUPERVISION OF

PROF. AARTI. M. KARANDIKAR

HEAD OF THE DEPARTMENT **PRINCIPAL**

(CSE) **(RCOEM)**

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

We extend our sincere thanks to our Project Guide **Prof. Aarti M. Karandikar** who in spite of being busy round the clock, helped, motivated and guided us. She has contributed to a great measure in surmounting all the hurdles faced during the completion of the project.
We also acknowledge the inspiration and encouragement provided by her. Her guidance, noble attitude, humbleness and soft-spoken attitude have taught us so more than just technical education.

We would also like to offer sincere and well devoted thanks to **Dr. M. B. Chandak**, Head of Department, C.S.E. for his continued encouragement and for giving us the opportunity to work on this project.

And finally our deepest gratitude to our parents and well-wishers for their unfailing emotional support of the project. Lastly we owe greatly to the almighty who is always there as the guiding force behind all creations of the world and without whose wish not a leaf can flutter.

Name of the Projectees:

Abhishek Dixit
Ajinkya Mahakalkar
Chandrashekhar Buradkar
Harshad Chaurewar
Sambhav Chopda

Abstract

Emergence of modern techniques for scientific data collection has resulted in large scale accumulation of data pertaining to diverse fields. Conventional database querying methods are inadequate to extract useful information from huge data banks. Cluster analysis is one of the major data analysis methods and the K-means clustering algorithm is widely used for many practical applications.[1]

The ability to monitor the progress of students' academic performance is a critical issue to the academic community of higher learning. A system for analyzing students' results based on cluster analysis and uses standard statistical algorithms to arrange their scores data according to the level their performance is described.

We also implemented K-mean clustering algorithm for students' for analyzing students' result data. This will be a good benchmark to monitoring the progression of academic performance of students in higher Institution for the purpose of making an effective decision by the academic planners.

Keywords: Data Analysis, Clustering, K-means Algorithm, Academic performance.

LIST OF FIGURES

List of Tables

Chapter 1
INTRODUCTION

1. INTRODUCTION

1.1 Definition of Data Analysis

Analysis of data is a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making. Data is collected and analyzed to answer questions, test hypotheses or disprove theories. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains.

Predictive analytics focuses on application of statistical models for predictive forecasting or classification, while text analytics applies statistical, linguistic, and structural techniques to extract and classify information from textual sources, a species of unstructured data. All are varieties of data analysis.[2]

1.2 Phases of Data Science Process

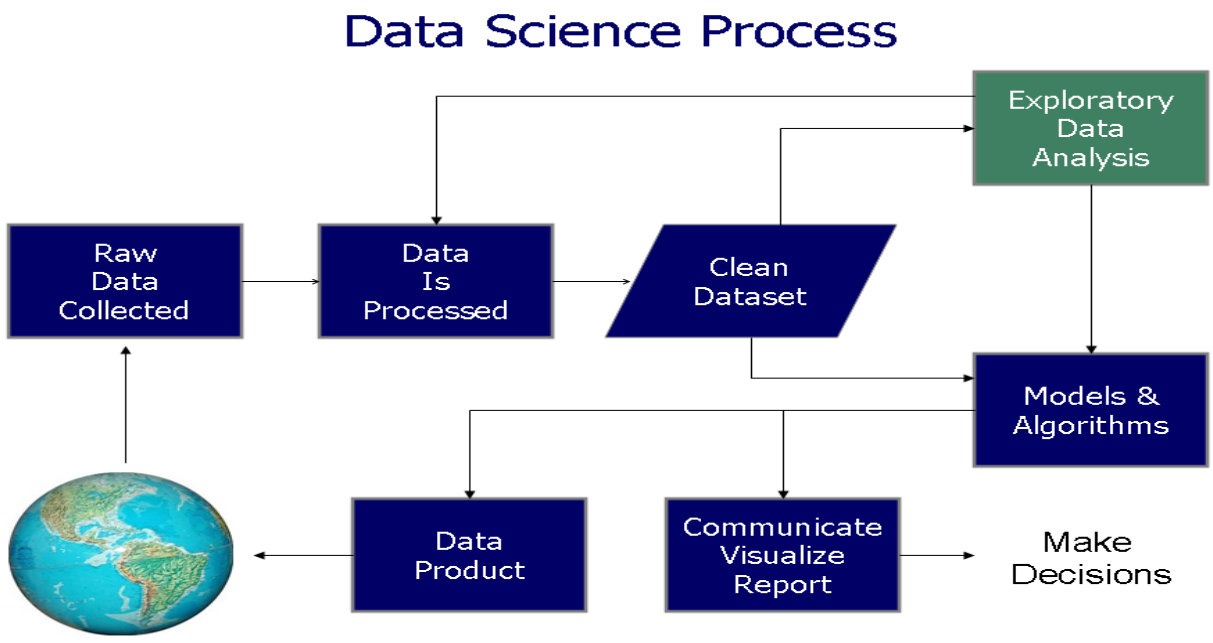


Fig. 1.1

There are several phases that can be distinguished and they are as follows:

1) **Raw Data collected:** This stage includes accessing potential data sources and collecting relevant data to be analyzed.

2) **Data Processing and cleaning:** Raw data often contains semantic errors, missing entries, or inconsistent formatting, so programmers need to reformat and clean data either by writing scripts or manually editing data in spreadsheets.

3) **Exploratory data analysis:** Writing, executing and refining computer programs called data analysis scripts to analyze and obtain insights of data.

4) **Models and algorithms:** Creation of learning models by using the concepts of machine learning and implementing algorithms to train and test the model.

5) **Communicate visualize reports:** Graphical representations of the result with the help of bar graphs, pie charts, histograms etc.

6) **Data Product:** Results in the form of written reports or research publications.

The phases are iterative, in that feedback from later phases may result in additional work in earlier phases.

1.3. PROBLEM DEFINITION

To generate detailed analysis of dataset of various fields which will help in enhancing the decision making and problem solving capabilities.

Data sets are from the fields of:

- Student Data: The Student Academic data with their CGPA, SGPA, name, roll no, semester wise grade of all student from year 2012-2014.

It will use the concept of Data Mining as a base.

1.4 TECHNOLOGY:

SOFTWARE: R- a free, open source statistical programming language available for Windows, Mac and Linux. In the 1990s Ross Ihaka and Robert Gentleman created R and shortly after made it freely available under the GNU General Public License.

1.5 OBJECTIVES

1.5.1 To Explore R:

R can be used for statistical analysis, graphics, and reporting. It can also be used to manipulate data, run statistical analyses such as descriptive statistics, t-tests, regressions, and produce charts.

R can even be used to make maps and play minesweeper.

1.5.2. To create an Application in R to analyze structured data:

As stated earlier, an application which takes as input a structured data in .csv or .xlsx format and outputs various bar graphs(box-plot) and pie charts to graphically depict the input data.

1.5.3. To train the data set used:

Using the concepts of data mining, we will train our dataset to be able to classify the newly entered data into their respective classes. AS a result, which will help in monitoring of student academic performance.

1.6 Organization of Report

In this project, we provided a simple and qualitative methodology to compare the predictive power of clustering algorithm. We demonstrated our technique using k-means clustering algorithm and combined with the deterministic model on a data set of our college results (year 2012-2014) with five courses offered for that semester for each student for total number of 415 students, and produces the numerical interpretation of the results for the performance valuation in form of Box Plot. This model improved on some of the limitations of the existing methods. These models applied fuzzy model to predict students' academic performance on two dataset only (CGPA and SGPA) of RCOEM results. Also the research work provides Data Mining frame work for "Students' academic performance". The research used rough Set theory as a classification approach to analyze student data where the R-studio was used to evaluate the student data to describe different dependencies between the attributes and the student status where the discovered patterns are explained in box plot graphs. Therefore, this clustering algorithm serves as a good benchmark to monitor the progression of students' performance in higher institution on three level (HOD, Staff and Student). It also enhances the decision making by academic planners to monitor the candidates' performance semester by semester by improving on the future academic results in the subsequence academic session.

Chapter 2

REVIEW OF LITERATURE

2. REVIEW OF LITERATURE

In this project we referred to an *(IJCSIS) International Journal of Computer Science and Information Security, Vol. 71,2010* titled "Application of k-Means Clustering algorithm for prediction of Students' Academic Performance". They analyzed the students' results of a private Institution in Nigeria to monitor the progression of academic performance of students in higher Institution for the purpose of making an effective decision by the academic planners.

Cumulative Graded Point Average (CGPA) is a commonly used indicator of = performance. Many Universities set a minimum CGPA that should be maintained in order to continue in the degree program. In some University, the minimum CGPA requirement set for the students is 6.0. Nonetheless, for any graduate program, a CGPA of 7.5 and above is considered an indicator of good academic performance. Therefore, CGPA still remains the most common factor used by the academic planners to evaluate progression in an academic environment [1]. Many factors could act as barriers to students attaining and maintaining a high CGPA that reflects their overall academic performance during their tenure in University. These factors could be targeted by the faculty members in developing strategies to improve student learning and improve their academic performance by way of monitoring the progression of their performance.

Therefore, performance evaluation is one of the bases to monitor the progression of student performance in higher Institution of learning. Base on this critical issue, grouping of students into different categories according to their performance has become a complicated task. So they traditionally grouped of students based on their average scores, it is difficult to obtain a comprehensive view of the state of the students' performance and simultaneously discover important details from their time to time performance.

With the help of data mining methods, such as clustering algorithm, it is possible to discover the key characteristics from the students' performance and possibly use those characteristics for future prediction. There have been some promising results from applying k-means clustering algorithm with the Euclidean distance measure, where the distance is computed by finding the square of the distance between each scores, summing the squares and finding the square root of the sum [6].

That paper presents k-means clustering algorithm as a simple and efficient tool to monitor the progression of students' performance in higher institution. Cluster analysis could be divided into hierarchical clustering and non-hierarchical clustering techniques. Examples of hierarchical techniques are single linkage, complete linkage, average linkage, median, and Ward. Non-hierarchical techniques include k-means, adaptive k-means, k-medoids, and fuzzy clustering. To determine which algorithm is good is a function of the type of data available and the particular purpose of analysis. The problem of selecting the "best" algorithm/parameter setting is a difficult one. A good clustering algorithm ideally should produce groups with distinct non-overlapping boundaries, although a perfect separation cannot typically be achieved in practice. The concept of stability of a clustering algorithm was considered in [3]. The idea behind this validation approach is that an algorithm should be rewarded for consistency. In that paper, they implemented traditional k-means clustering algorithm [6] and Euclidean distance measure of similarity was chosen to be used in the analysis of the students' scores. We also implemented the same in this project by using data set of our college from session 2012-2014. In which we have monitored student data in three levels (HOD, Professor level , Student level).

In that paper, they have provided a simple and qualitative methodology to compare the predictive power of clustering algorithm and the Euclidean distance as a measure of similarity distance. They demonstrated technique using k-means clustering algorithm [6] and combined with the deterministic model in [6] on a data set. This model improved on some of the limitations of the existing methods, such as model developed by [5] and [6]. These models applied fuzzy model to predict students' academic performance on two dataset only (English Language and Mathematics) of Secondary Schools results. Also the research work by [4] only provides Data Mining framework for Students' academic performance. The research by [10] used rough Set theory as a classification approach to analyze student data where the Rosetta toolkit was used to evaluate the student data to describe different dependencies between the attributes and the student status where the discovered patterns are explained in plain English.

Therefore, clustering algorithm serves as a good benchmark to monitor the progression of students' performance in colleges. It enhances the decision making by academic planners to monitor the candidates' performance semester by semester by improving on the future academic results in the subsequence academic session.

Chapter 3

Proposed Approach

&

System Architecture

3.1 Proposed Approach

In this project we initially used data set of our college academic data from year 2012-2014 in the form of excel file. Then we have applied data cleaning technique like removing empty cell and unwanted data on data set. Then imported that data file in using a command read xlsx file in R programming in R-Studio. We have implemented this using three levels i.e. student level where he/she can analyze his/her own performance and can also calculate his/her target SGPA in order to increase his/her CGPA to in upcoming semester. The second level is staff level or professor level were teacher can view individual performance as well as particular class performance in that semester or subject. Then last level is of HOD were he can analysis individual student, particular shift or whole year for that subject or semester. For monitoring the data firstly the user have login using any of the above user's login id. After he/she have to select the data set on which he/she wants check performance. Then we have applied K-mean algorithm on data sets selected by user and generated the output in the form Scattered plot. But the output was not that understandable as many points were overlapping on one another and no result was inferable from that, so we tried to use another data mining technique called as BOX-Plot graph also known as five point summary graph it displayed minimum cgpa, maximum cgpa, quartile1(q1), quartile3(q3) , and median of cgpa in output. The area under the region of q1 and q3 shows the range of students from 25-75% from data set. This graph was easy to understand and interpret the result from it. The output generated also contained the outliers which are nothing but the student's cgpa which doesn't comes under the range. This outliers were mainly the student falling under the cgpa range of below 4.

3.2 SYSTEM ARCHITECTURE

R is a free software programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls and surveys of data miners are showing R's popularity has increased substantially in recent years.

R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. S was created by John Chambers while at Bell Labs. 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*, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S.

R is a GNU project. The source code for the R software environment is written primarily in C, Fortran, and R. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. R uses a command line interface; however, several graphical user interfaces are available for use with R.[3]

3.2.1 STATISTICAL FEATURES

R provides a wide variety of statistical and graphical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and others. R is easily extensible through functions and extensions, and the R community is noted for its active contributions in terms of packages. There are some important differences, but much code written for S runs unaltered. Many of R's standard functions are written in R itself, which makes it easy for users to follow the algorithmic choices made. For computationally intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C, C++, Java, .NET or Python code to manipulate R objects directly.[4]

R is highly extensible through the use of user-submitted packages for specific functions or specific areas of study. Due to its S heritage, R has stronger object-oriented programming facilities than most statistical computing languages. Extending R is also eased by its lexical scoping rules.

Another strength of R is static graphics, which can produce publication-quality graphs, including mathematical symbols. Dynamic and interactive graphics are available through additional packages.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hard copy.

3.2.2 PROGRAMMING FEATURES:

R is an interpreted language that is users typically access it through a command-line interpreter. If a user types "2+2" at the R command prompt and presses enter, the computer replies with "4", as shown below:

```
> 2+2
[1] 4
```

Like other similar languages such as APL and MATLAB, R supports matrix arithmetic. R's data structures include vectors, matrices, arrays, data frames (similar to tables in a relational database) and lists. R's extensible object system includes objects for (among others): regression models, time-series and geo-spatial coordinates. The scalar data type was never a data structure of R. A scalar is represented as a vector with length one in R.

R supports procedural programming with functions and, for some functions, object-oriented programming with generic functions. A generic function acts differently depending on the type of arguments passed to it. In other words, the generic function dispatches the function (method) specific to that type of object. For example, R has a generic print() function that can print almost every type of object in R with a simple "print(objectname)" syntax.

Although used mainly by statisticians and other practitioners requiring an environment for statistical computation and software development, R can also operate as a general matrix calculation toolbox – with performance benchmarks comparable to GNU Octave or MATLAB.

3.2.3 INTERFACES:

3.2.3.1 Graphical user interfaces

- **RStudio** – cross-platform open source IDE (which can also be run on a remote linux server).
- **Deducer** – GUI for menu driven data analysis (similar to SPSS/JMP/Minitab).
- Java GUI for R – cross-platform stand-alone R terminal and editor based on Java (also known as JGR).
- **Rattle GUI** – cross-platform GUI based on RGtk2 and specifically designed for data mining.
- **R Commander** – cross-platform menu-driven GUI based on tcltk (several plug-ins to Rcmdr are also available).
- Revolution Analytics provides a Visual Studio based IDE and has plans for web based point and click interface.
- **RGUI** – comes with the pre-compiled version of R for Microsoft Windows.
- **RKWard** – extensible GUI and IDE for R.

- **RWeka**– allows for the use of the data mining capabilities in Weka and statistical analysis in R. 3.3.2 Editors and IDEs.

3.2.3.2 Text Editors and IDEs:

Text editors and Integrated development environments (IDEs) with some support for R include: ConTEXT, Eclipse (StatET), Emacs (Emacs Speaks Statistics), LyX (modules for knitr and Sweave), Vim, jEdit Kate, Revolution R Enterprise DevelopR (part of Revolution R Enterprise), RStudio, Sublime Text, TextMate, WinEdt (R Package RWinEdt), Tinn-R and Notepad++.

3.2.3.3 Scripting languages

R functionality has been made accessible from several scripting languages such as Python, Perl, Ruby, and F#. PL/R can be used alongside, or instead of, the PL/pgSQL scripting language in the PostgreSQL and Greenplum database management system. Scripting in R itself is possible via littler.

3.2.4 CLASSES Used

The central computation in R is a function call. The call is defined by the function object and the objects supplied as arguments. The result returned by the function in almost all cases is also an object. When a class is defined, an object is stored that contains the information about that class. The object, known as the *metadata* defining the class, is not stored under the name of the class (to allow programmers to write generating functions of that name), but under a specially constructed name. To examine the class definition, call getClass. There are a number of 'basic' classes, corresponding to the ordinary kinds of data occurring in R and the classes those which we used are:

- **Numeric class:**

`"numeric"` is a class corresponding to numeric vectors. The datatype or the object type is double. It can take values of form 1, 0.34, 1e4.
For example

```
> length(x)
[1] 10
> mode(x)
[1] "numeric"
```

- **Integer class:**

"integer" class corresponds to integer values. The data type or the object type is Integer.
For example

```
> length(y)
[1] 20
> typeof(x)
[1] "integer"
```

- **Logical class:**

The "logical" class can only take on two values, TRUE or FALSE**.** For example

```
> x = 1; y = 2    # sample values
> z = x > y       # is x larger than y?
> z               # print the logical value
[1] FALSE
> class(z)        # print the class name of z
[1] "logical"
```

- **Character class:**

The "character" class uses the datatype or the object type 'character'.
A **character** object is used to represent string values in R. We convert objects into character values with the as.character() function.
For example:

```
> x = as.character(3.14)
> x               # print the character string
[1] "3.14"
> class(x)        # print the class name of x
[1] "character"
```

3.2.5 METHODS Used:

Each **R** package will include methods metadata objects corresponding to each generic function for which methods have been defined in that package. When the package is loaded into an R session, the methods for each generic function are *cached*, that is, stored in the environment of the generic function along with the methods from previously loaded packages. This merged table of methods is used to dispatch or select methods from the generic, using class inheritance and possibly group generic functions. The caching computations ensure that only one version of each generic function is visible globally; although different attached packages may contain a copy of the generic function, these behave identically with respect to method selection. R has extensive statistical and graphing capabilities. R provides hundreds of built-in statistical functions as well as its own built-in programming language.

The following functions were used:[5]

-> c(x): A generic function which combines its arguments.
For example

```
> x <- c(1, 2, 3, 4, "Pi")
> x[1] "1"  "2"  "3"  "4"  "Pi"
```

-> cat(x): Prints the arguments and used for concatenating the output.
For example

```
>cat("age", person$age, "\n")
[1] age 45
```

> range(x): Returns the maximum and minimum of x.
For example

```
> range(weights)
[1] 119.5 237.1
```

>seq(x): Used to generate sequence.
For example here a sequence gets generated from 40 to 100 spaced by 5.

```
> breaks < - seq(40, 100, by = 5)
> breaks
 [1]  40  45  50  55  60  65  70  75  80  85  90  95 100
```

> cut(): It uses tables and a right parameter to create a grouped frequency table.
For example

```
> breaks < - seq(40, 100, by = 5)
> breaks
[1]  40  45  50  55  60  65  70  75  80  85  90  95 100
 > wait_time < - cut(waiting, breaks, right = FALSE)
 > table(wait_time)
 wait_time
 [40,45)  [45,50)  [50,55)  [55,60)  [60,65)  [65,70)  [70,75)  [75,80)

      1       20       32       24       17        9       23       54
 [80,85)  [85,90)  [90,95) [95,100)
      57       23       11        1
```

> plot(): Generic functions for ploting of R objects.

> pie(): Used to create a pie chart.

```
 > names(Percent) <-
c("Silver","Black","White","Gray","Red","Blue","Brown", +
"Green","Other")
  > names(Percent)
 [1] "Silver" "Black"  "White"  "Gray"   "Red"    "Blue"   "Brown"
"Green"
 [9] "Other"
 > pie(Percent, col = colors, main = "Pie Graph of Auto Color
Preferences")
```

> barplot(): It is to create a bar plot of the objects.
For example

```
barplot(Percent, col = colors, main = "Bar Chart of Auto Color
Preferences")
```

> hist(): Used to plot a histogram of x. Histograms are similar to bar charts, but because of the underlying continuum on the x axis, there should be no gaps between the bars.

For example after attaching the table data a histogram of waiting is created.
attach(faithful)
> hist(waiting)

3.2.6 EXAMPLES:

3.2.6.1 Example 1:

The following examples illustrate the basic syntax of the language and use of the command-line interface.

In R, the widely preferred assignment operator is an arrow made from two characters "<-", although "=" can be used instead.

```
> x <- c(1,2,3,4,5,6)    # Create ordered collection (vector)
> y <- x^2               # Square the elements of x
> print(y)               # print (vector) y
[1]  1  4  9 16 25 36
> mean(y)                # Calculate average (arithmetic mean) of (vector) y;
result is scalar

[1] 15.16667
> var(y)                 # Calculate sample variance
[1] 178.9667

> lm_1 <- lm(y ~ x)      # Fit a linear regression model "y = f(x)" or "y = B0 +
(B1 * x)"
                         # store the results as lm_1
> print(lm_1)            # Print the model from the (linear model object) lm_1

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     -9.333        7.000

> summary(lm_1)           # Compute and print statistics for the fit
                          # of the (linear model object) lm_1

Call:
lm(formula = y ~ x)

Residuals:
1       2       3       4       5       6
3.3333 -0.6667 -2.6667 -2.6667 -0.6667  3.3333
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -9.3333     2.8441  -3.282 0.030453 *
x             7.0000     0.7303   9.585 0.000662 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Residual standard error: 3.055 on 4 degrees of freedom
Multiple R-squared: 0.9583,    Adjusted R-squared: 0.9478
F-statistic: 91.88 on 1 and 4 DF,  p-value: 0.000662

> par(mfrow=c(2, 2))      # Request 2x2 plot layout
> plot(lm_1)              # Diagnostic plot of regression model
```

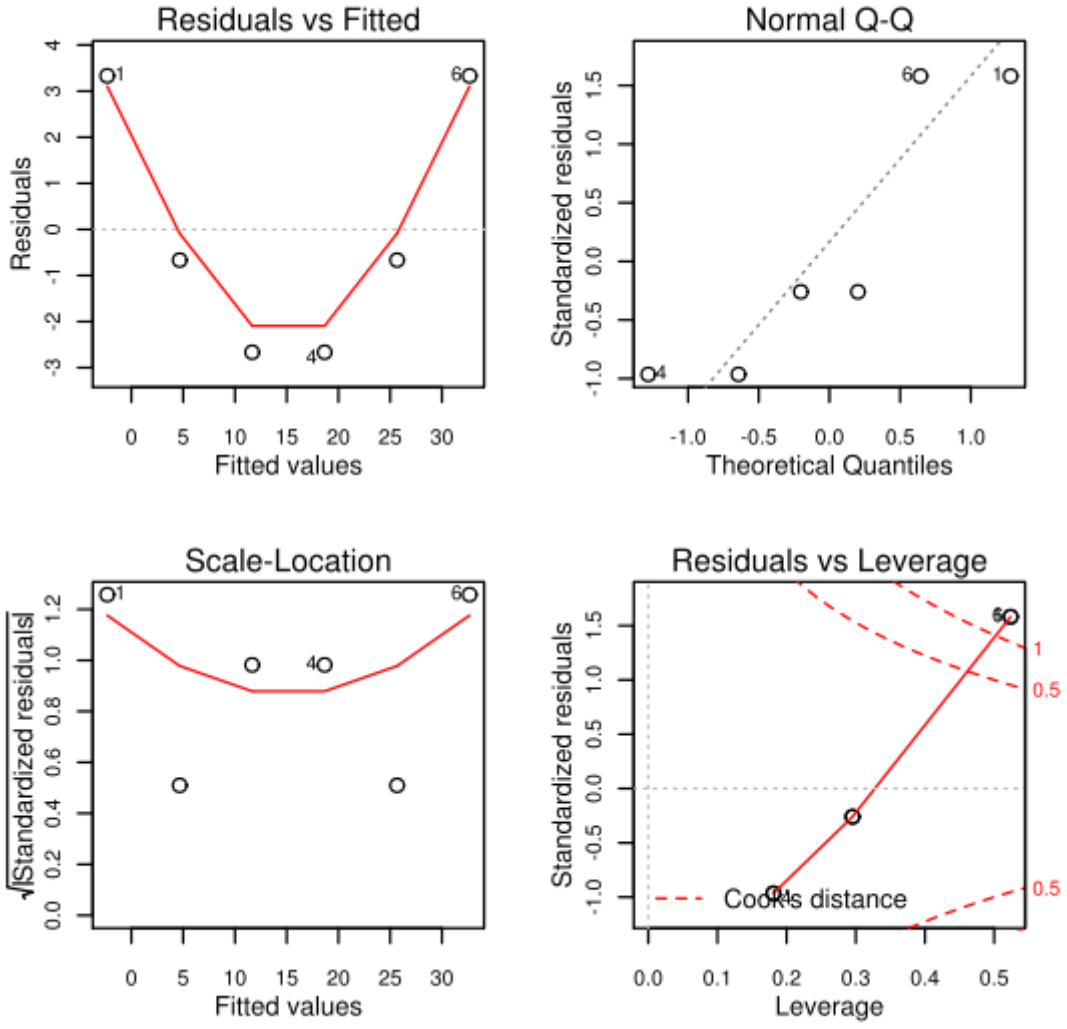


Fig. 2

3.2.6.2 Example 2

The ease of function creation by the user is one of the strengths of using R. Objects remain local to the function, which can be returned as any data type. Below is an example of the structure of a function:[6]

```
functionname <- function(arg1, arg2, ... ){ # declare name of function and
function arguments
statements                                  # declare statements
return(object)                              # declare object data type
}

sumofsquares <- function(x){ # a user-created function
  return(sum(x^2))           # return the sum of squares of the elements of
vector x
}
> sumofsquares(1:3)
```

- `[1] 14`

3.2.7 PACKAGES

The capabilities of R are extended through user-created *packages*, which allow specialized statistical techniques, graphical devices (ggplot2), import/export capabilities, reporting tools (knitr, Sweave), etc. These packages are developed primarily in R, and sometimes in Java, C and Fortran. A core set of packages is included with the installation of R, with more than 5,800 additional packages and 120,000 functions (as of June 2014) available at the Comprehensive R Archive Network (CRAN), Bioconductor, and other repositories.

The "Task Views" page (subject list) on the CRAN website lists a wide range of tasks (in fields such as Finance, Genetics, High Performance Computing, Machine Learning, Medical Imaging, Social Sciences and Spatial Statistics) to which R has been applied and for which packages are available. R has also been identified by the FDA as suitable for interpreting data from clinical research.

Other R package resources include Crantastic, a community site for rating and reviewing all CRAN packages, and R-Forge, a central platform for the collaborative development of R packages, R-related software, and projects. R-Forge also hosts many unpublished beta packages, and development versions of CRAN packages.

The Bioconductor project provides R packages for the analysis of genomic data, such as Affymetrix and cDNA microarray object-oriented data-handling and analysis tools, and has started to provide tools for analysis of data from next-generation high-throughput sequencing methods.

Reproducible research and automated report generation can be accomplished with packages that support execution of R code embedded within LaTeX, OpenDocument format and other markups.

Following are the some packages we used in our projects:

3.2.7.1 RGtk2:

RGtk2 is a binding for R to the GTK2 library and dependent libraries. You can use it to construct arbitrarily complex GUI's from R.

Features

- Virtually complete bindings to GTK, GDK, Pango, Cairo, GdkPixbuf, ATK, and Libglade.
- Full documentation derived from that of the bound libraries.
- Lots of demos to get you started using RGtk2.
- To draw R graphics inside an RGtk2 GUI:

Eg.
```
> win = gtkWindow()
> da = gtkDrawingArea()
> win$add(da)
> asCairoDevice(da)
    • > plot(1:10)
```

3.2.7.2 gWidgets:

The gWidgets API is intended to be a cross platform means within an R session to interact with a graphics toolkit. It uses the EXT JS JavaScript libraries to provide R to web interactivity. Unlike the other implementations, this package is stand alone and does not use gWidgets itself. The API is intended to facilitate the task of writing basic GUIs, as it simplifies many of the steps involved in setting up widgets and packing them into containers. Although not nearly as powerful as any individual toolkit, the gWidgets API is suitable for many tasks or as a rapid prototyping tool for more complicated applications.

E.g. A basic hello world application can be made as follows:

```
library(gWidgets)
options(guiToolkit="RGtk2")                          # avoid question if more than
one is installed
w <- gwindow("Hello world example")                 # top level window
g <- ggroup(cont=w, horizontal=FALSE)               # a box container, added to w
b <- gbutton("Click me for a message", cont=g)      # add button to container g
addHandlerClicked(b, handler=function(h,...) {     # add interactivity through a
handler
      galert("Hello world", parent=h$obj)})
```

3.2.7.3 Png

Plots in PNG and JPEG format can easily be converted to many other bitmap formats, and both can be displayed in modern web browsers. The PNG format is lossless and is best for line diagrams and blocks of color. The JPEG format is lossy, but may be useful for image plots, for example. BMP is a standard format on Windows. TIFF is a meta-format: the default format written by tiff is lossless and stores RGB (and alpha where appropriate) values uncompressed—such files are widely accepted, which is their main virtue over PNG.

png(filename, width=480, height=480)

- **filename** – A character string for the file name, ending in .png.
- **width** – Width of the image, in inches.
- **width** – Height of the image, in inches.
- dev.off()

```
E.g.
      png(file = "myplot.png", bg = "transparent")
      plot(1:10)
      rect(1, 5, 3, 7, col = "white")
      dev.off()
```

3.2.7.4. devtool

Devtools makes package development a breeze: it works with R's existing conventions for code structure, adding efficient tools to support the cycle of package development. With devtools, developing a package becomes so easy that it will be your default layout whenever you're writing a significant amount of code.

The goal of devtools is to make package development as painless as possible. It does this by encapsulating all of the best practices of package development that I've learned over the years. Devtools protects you from many potential mistakes, so you can focus on the problem you're interested in, not on developing a package. Devtools works hand-in-hand with R-Studio, which I believe is the best development environment for most R users.

Eg `install.packages(c("devtools", "roxygen2", "testthat", "knitr"))`

3.2.8 Working with R

We finally address what prompted us to take up R as the medium for our project:

R is a dialect of the S language, and has come to be — by far — the dominant dialect.

3.2.8.1 Difference between R and S

S started as a research project at Bell Labs a few decades ago, it is a language that was developed for data analysis, statistical modeling, simulation and graphics. However, it is a general purpose language with some powerful features — it could (and does) have uses far removed from data analysis.

It should be used for many of the tasks that spreadsheets are currently used for. If a task is non-trivial to do in a spreadsheet, then almost always it would more productively (and safely) be done with R. Spreadsheet Addiction talks about problems with spreadsheets and how R is often a better tool.

3.2.8.2 Characteristics of S Language:

- S is not just a statistics package, it's a language.
- S is designed to operate the way that problems are thought about.
- S is both flexible and powerful.

The Importance of Being a Language

Though the distinction between a package and a language is subtle, that subtle difference has a massive impact. With a package you can perform some set number of tasks — often with some options that can be varied. A language allows you to specify the performance of new tasks.

Your retort may be, "But I won't want to create a new form of regression." Yes, S does allow you to create new forms of regression (and many people have), but S also allows you to easily perform the same sort of standard regression on your 5 datasets (or maybe it is 500 datasets).

The key is abstraction. You easily see that your 5 regressions are really the same — there is merely different data involved with each. In your mind you have abstracted the specific tasks so that they all look similar. Once you've seen the abstraction, it is simple to teach R the abstraction. Languages are all about abstraction.

You can get an idea of the power of language by having a look at the R graphics gallery.

The Way We Think

One of the goals of S, and one that I think has largely been successful, is that the language should mirror the way that people think. A simple example: suppose we think that weight is a function of (dependent on) height and girth. The S formula to express this is:

```
weight ~ height + girth
```

The + is not + as in addition, but + as in "and".

Another feature of S is that it is vector-oriented — meaning that objects are generally treated as a whole — as humans tend to think of the situation — rather than as a collection of individual numbers. Suppose that we want to change the heights from inches to centimeters. In S the command could be:

```
height.cm <- 2.54 * height.inches
```

Here `height.inches` is an object that contains some number — one or millions — of heights. S hides from the user that this is a series of multiplications, but acts more like we think — whatever is in inches multiply by 2.54 to get centimeters.

Experience with C or Fortran can ironically make it harder to use S efficiently. The C-before-S gang tend to translate the problem into "programming" rather than thinking about the problem in the "natural" way.

A Moveable Feast: Flexibility and power abound in S. For instance, it is easy to call C and C++ functionality from R. R does not insist that everything is done in its language, so you can mix tools picking the best tool for each particular task.

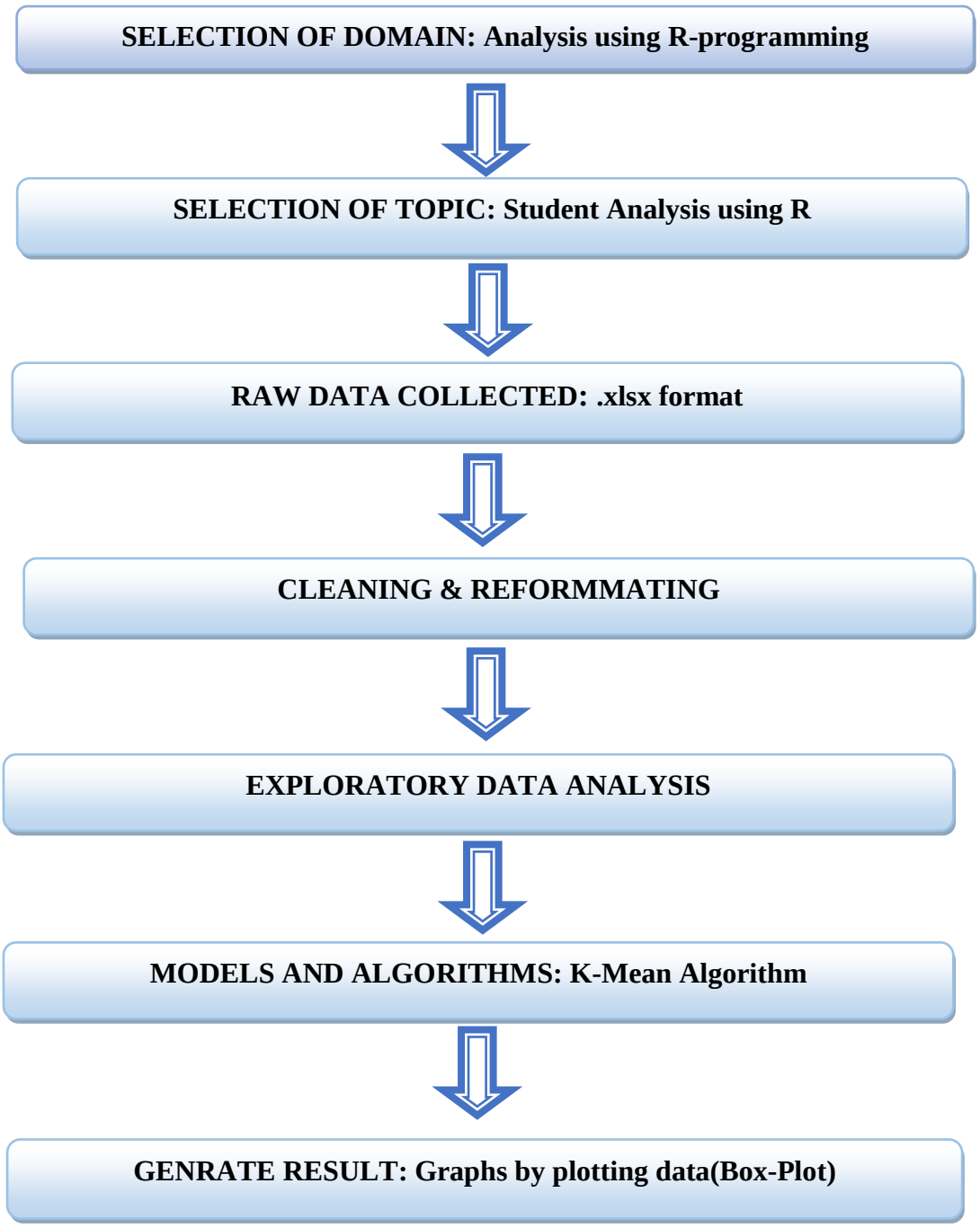
The pieces of code that are written in the S language are always available to the user, so a minor change to the task usually requires only a minor change to the code — a change that can be carried out in a minor amount of time.

3.2.8.3 The Preferred Medium

Given its qualities, R has become the preferred computing environment for a large part of the statistical community. When a new statistical method is invented, chances are it will be implemented first in R.

In March 1999 John Chambers — one of the originators of S at Bell Labs — was presented the ACM Software System Award. It stated, "S has forever altered the way people analyze, visualize, and manipulate data." Previous winners of this award include Unix, TeX and the World-Wide Web. John is now a member of R Core (the group that produces R).

FLOWCHART


SELECTION OF DOMAIN: Analysis using R-programming
SELECTION OF TOPIC: Student Analysis using R
RAW DATA COLLECTED: .xlsx format
CLEANING & REFORMMATING
EXPLORATORY DATA ANALYSIS
MODELS AND ALGORITHMS: K-Mean Algorithm
GENRATE RESULT: Graphs by plotting data(Box-Plot)

Chapter 4

System Description

A. Development of k-mean clustering algorithm

Given a dataset of n data points x_1, x_2, ..., x_n such that each data point is in $\mathbf{R^d}$, the problem of finding the minimum variance clustering of the dataset into k clusters is that of finding k points $\{m_j\}$ (j=1, 2, …, k) in $\mathbf{R^d}$ such that

$$\frac{1}{n}\sum_{i=1}^{n}\left[\min_j d^2(x_i, m_j)\right] \qquad (1)$$

is minimized, where $d(x_i, m_j)$ denotes the Euclidean distance between x_i and m_j. The points $\{m_j\}$ (j=1, 2, …,k) are known as cluster centroids. The problem in Eq.(1) is to find k cluster centroids, such that the average squared Euclidean distance (mean squared error, MSE) between a data point and its nearest cluster centroid is minimized.

The *k*-means algorithm provides an easy method to implement approximate solution to Eq.(1). The reasons for the popularity of *k*-means are ease and simplicity of implementation, scalability, speed of convergence and adaptability to sparse data.

The *k*-means algorithm can be thought of as a gradient descent procedure, which begins at starting cluster centroids, and iteratively updates these centroids to decrease the objective function in Eq.(1). The *k*-means always converge to a local minimum. The particular local minimum found depends on the starting cluster centroids. The problem of finding the global minimum is NP-complete. The *k*-means algorithm updates cluster centroids till local minimum is found. Fig.1 shows the generalized pseudocodes of *k*-means algorithm; and traditional k-means algorithm is presented in fig. 2 respectively.

Before the *k*-means algorithm converges, distance and centroid calculations are done while loops are executed a number of times, say l, where the positive integer l is known as the number of *k*-means iterations. The precise value of l varies depending on the initial starting cluster centroids even on the same dataset. So the computational time complexity of the algorithm is $O(nkl)$, where n is the total number of objects in the dataset, k is the required number of clusters we identified and l is the number of iterations, $k \leq n$, $l \leq n$ [6]

Step 1: Accept the number of clusters to group data into and the dataset to cluster as input values

Step 2: Initialize the first K clusters
- Take first k instances or
- Take Random sampling of k elements

Step 3: Calculate the arithmetic means of each cluster formed in the dataset.

Step 4: K-means assigns each record in the dataset to only one of the initial clusters
- Each record is assigned to the nearest cluster using a measure of distance (e.g Euclidean distance).

Step 5: K-means re-assigns each record in the dataset to the most similar cluster and re-calculates the arithmetic mean of all the clusters in the dataset.

Fig: 4.1: Generalized Pseudo code of Traditional k-means

```
1     MSE = largenumber;
      Select initial
      cluster centroids
2     {mj}j
      K = 1;
3     Do
4       OldMSE = MSE;
5       MSE1 = 0;
6       For j = 1 to k
7         mj = 0; nj = 0;
8       endfor
9       For i = 1 to n
10        For j = 1 to k
11        Compute squared Euclidean distance
          d2(xi, mj);
12        endfor
13      Find the closest centroid mj to xi;
14        mj = mj + xi; nj = nj+1;
15        MSE1=MSE1+ d2(xi, mj);
16      endfor
17      For j = 1 to k
18        nj = max(nj, 1); mj = mj/nj;
19      endfor
20    MSE=MSE1;
    while (MSE<OldMSE)
```

Fig.4.2: Traditional *k*-means algorithm [6]

We applied the model on the data set (academic result from 2012-2014 all semesters) of a university in RTMNU, RCOEM. The result generated is shown in tables 2 respectively. In table 1, it shows the performance index of students

CGPA	***Grade***
9 and above	Excellent
8-8.99	Very Good
7-7.99	Good
6-6.99	Very Fair
5-5.99	Flair
Below 5	Poor

Table 1: Performance Index

Table 5 shows the dimension of the data set (Student's scores) in the form N by M matrices, where N is the rows (# of students) and M is the column (# of courses) offered by each student.

Student's Score	**No. of Students**	**Dimensions(Totalno. of courses)**
Final year	140	7^{th} Semester(Win-14)
Third year	150	5^{th}-6^{th}(13-14) & 5^{th}(Win-14)
Second year	155	3^{rd}-4^{th}(12-14) & 3^{rd}(Win-14)

Table 2: Statistics of the Data used

The overall performance is evaluated by applying deterministic model in Eq. 2 [7] where the group assessment in each of the cluster size is evaluated by summing the average of the individual scores in each cluster.

$$\frac{1}{N}\left(\sum_{j=1}^{n}\left(\frac{1}{n}\sum_{i=1}^{n} Xi\right)\right) \quad \text{.....(2)}$$

Where

N = the total number of students in a cluster and n = the dimension of the data

n= the dimension of the data

We have also implemented the function that if an individual student wants to improve his/her cgpa then then can calculate their target SGPA. For this we have used RODBC package. Package RODBC implements ODBC database connectivity.

Two groups of functions are provided. The mainly internal odbc* commands implement low-level access to the ODBC functions of similar name. The sql* functions operate at a higher level to read, save, copy and manipulate data between data frames and SQL tables. Many connections can be open at once to any combination of DSN/hosts.

sqlQuery is the workhorse function of RODBC. It sends the SQL statement query to the server, using connection channel returned by odbcConnect, and retrieves (some or all of) the results via sqlGetResults. The term 'query' includes any valid SQL statement including table creation, alteration, updates etc as well as 'SELECT's. The sqlQuery command is a convenience wrapper that first calls odbcQuery and then sqlGetResults. If finer-grained control is needed, for example over the number of rows fetched, additional arguments can be passed to sqlQuery or the underlying functions called directly. sqlGetResults is a mid-level function. It is called after a call to sqlQuery or odbcQuery to retrieve waiting results into a data frame. Its main use is with max set to non-zero when it will retrieve the result set in batches with repeated calls. This is useful for very large result sets which can be subjected to intermediate processing.

Ex.

```
library(RODBC)
channel <- odbcConnectExcel("f:/Book1.xls")

sqlTables(channel)
sh1 <- sqlQuery(channel, "select F3,F4,F5,F6,F7 from [Sheet1$] WHERE
F2='BE11CSU805' ")
sh1
result<-kmeans(sh1,1)
boxplot(result)
close(RODBC)
```

Chapter 5
Results
&
Discussions

Results

We used R-studio in our project as compiler for executing r-programs. As we have made three types of user in projects i.e. student, professor and HOD, so they have to login first with the help of their respective login-id as well as passwords.

Initially the user get the following screen after opening the R-Studio as shown in fig 5.1

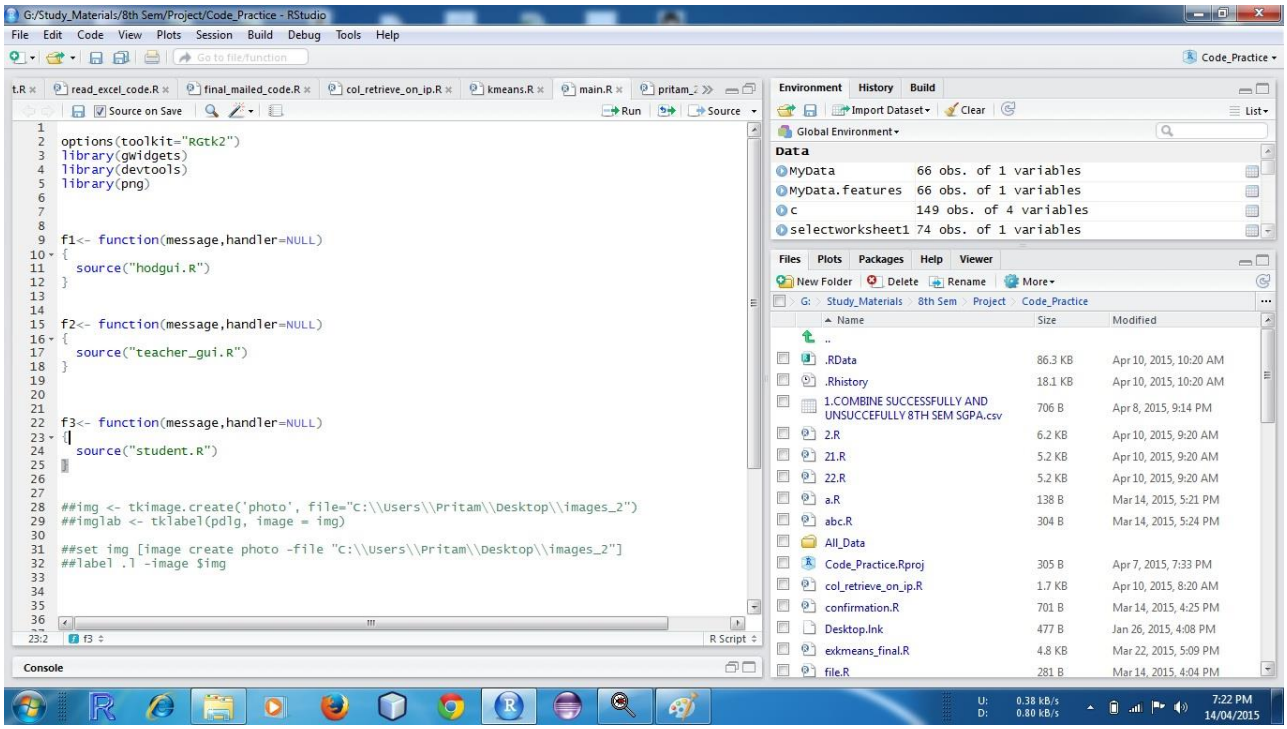


Fig 5.1 :Main page

STEP 1:

After the compiling the main.r file we get the login window which is given in fig 5.2 below
The login page contains the data that user can be of three types

- **HOD**
- **STAFF or Professor**
- **STUDENT**

User will click on particular button to continue…

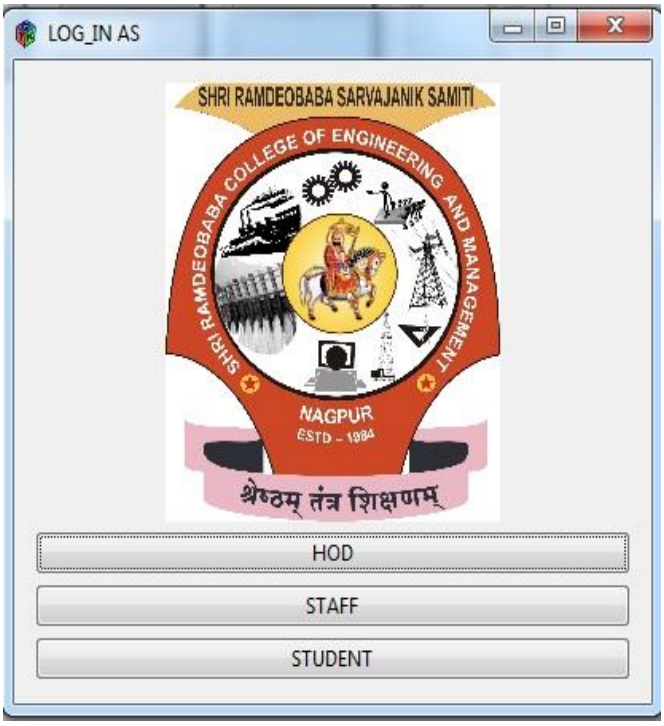


Fig5.2: Types of Users

Source code:

```
options(toolkit="RGtk2")
library(gWidgets)
library(devtools)
library(png)
f1<- function(message,handler=NULL)
{
  source("hodgui.R")
}
f2<- function(message,handler=NULL)
{
  source("teacher_gui.R")
}
f3<- function(message,handler=NULL)
{
  source("student.R")
}
##img <- tkimage.create('photo', file="C:\\ images_2")
##imglab <- tklabel(pdlg, image = img)

##set img [image create photo -file "C:\\images_2"]
##label .l -image $img

win <- gwindow("LOG_IN AS", visible=TRUE)
group<-ggroup(horizontal = FALSE, container=win,
toolkit=guiToolkit())

image<-
gimage(filename="C:\\RCOEM_Logo.png",container=group,size
="button")

button<-
gbutton(text="HOD",border=TRUE,handler=f1,container=group
,toolkit=guiToolkit())

button1<-
gbutton(text="STAFF",border=TRUE,handler=f2,container=gro
up,toolkit=guiToolkit())
button2<-
gbutton(text="STUDENT",border=TRUE,handler=f3,container=g
roup,toolkit=guiToolkit())
```

STEP 2:

After clicking on particular user's button then the login window opens fig 5.3, after putting correctly information new window of that user will be displayed. Suppose the user is HOD Sir, then fig 5.4 will be displayed after correctly inputting the details else an ERROR message will be popup if invalid user or password is given.

Fig 5.3 Login Window

Fig 5.4 HOD_Sir logged in Window

Source code :

```
options(toolkit="RGtk2")
library(gWidgets)
library(devtools)
library(png)

fgoto<-function(message,handler=NULL)
{
  source("col_retrieve_on_ip.R")
}
fvalidate<- function(message,handler=NULL)
{
if(svalue(obj1)=="hod_cs")
  {
    if(svalue(obj3)=="admin")
    {
      dispose(winlogin)
      winlogin1 <- gwindow("Welcome", visible=TRUE)
      group1<-ggroup(horizontal = FALSE, container=winlogin1,
toolkit=guiToolkit())
       obj10 <-
glabel(text=Sys.time(),container=group1,toolkit=guiToolkit())
      obj10 <- glabel(text="Welcome
M.B.Chandak",container=group1,toolkit=guiToolkit())
      image1= gimage(filename = "C:\\hod.png",size
="large_toolbar",container = group1,toolkit=guiToolkit())

      ##image1= gimage(filename = "C:\ \images_2.jpeg",size =
"menu",container = group1,toolkit=guiToolkit())
      ##cal<- gcalendar(container=group1,toolkit=guiToolkit())

      ##time<-
gdkEventGetTime(container=group1,toolkit=guiToolkit())
      f14<-function(message,handler=NULL)
      {
        source("fourth.R")
      }
      f13<-function(message,handler=NULL)
      {
        source("third.R")
      }
      f12<-function(message,handler=NULL)
      {
        source("second.R")
      }
```

```
button11<-gbutton(text="2nd
Year",border=TRUE,handler=f12,container=group1,toolkit=guiToolkit())
      button12<-gbutton(text="3rd
Year",border=TRUE,handler=f13,container=group1,toolkit=guiToolkit())
      button13<-gbutton(text="4th
Year",border=TRUE,handler=f14,container=group1,toolkit=guiToolkit())

       ##button1.setBackground("Blue")
      button3<- gbutton(text="Show
Result",border=TRUE,handler=fgoto,container=group1,toolkit=guiToolkit
())

flogout<-function(message,handler=NULL)
        {
        dispose(winlogin1)
        }

button2<-
gbutton(text="SignOut",border=TRUE,handler=flogout,container=group1,t
oolkit=guiToolkit())

}
    else
    {

      gconfirm("Wrong Id and
Passwors",title="Ok",icon=c("error"),toolkit=guiToolkit())
   }
}else
  {
    gconfirm("Wrong Id and
Password",title="Ok",icon=c("warning"),toolkit=guiToolkit())
   }
}winlogin <- gwindow("Rcoem Nagpur", visible=TRUE)
group<-ggroup(horizontal = FALSE, container=winlogin,
toolkit=guiToolkit())

obj <- glabel(text="Username",container=group,toolkit=guiToolkit())
obj1 <- gedit("", container=group,toolkit=guiToolkit())
obj2 <- glabel(text="Password",container=group,toolkit=guiToolkit())
obj3 <- gedit("", container=group,toolkit=guiToolkit())

visible(obj3)<- FALSE
button1<-
gbutton(text="Login",border=TRUE,handler=fvalidate,container=group,to
olkit=guiToolkit())
```

Step 3:

Following are the all the option or windows sir can browse to perform analysis of data given in fig 5.5,

As HOD Sir only have the privileges that he can analysis any shift any year or particular student. He can analysis fourth year, third year or Second year individually or both shifts simultaneously. The result will be generated based on the data set selected.

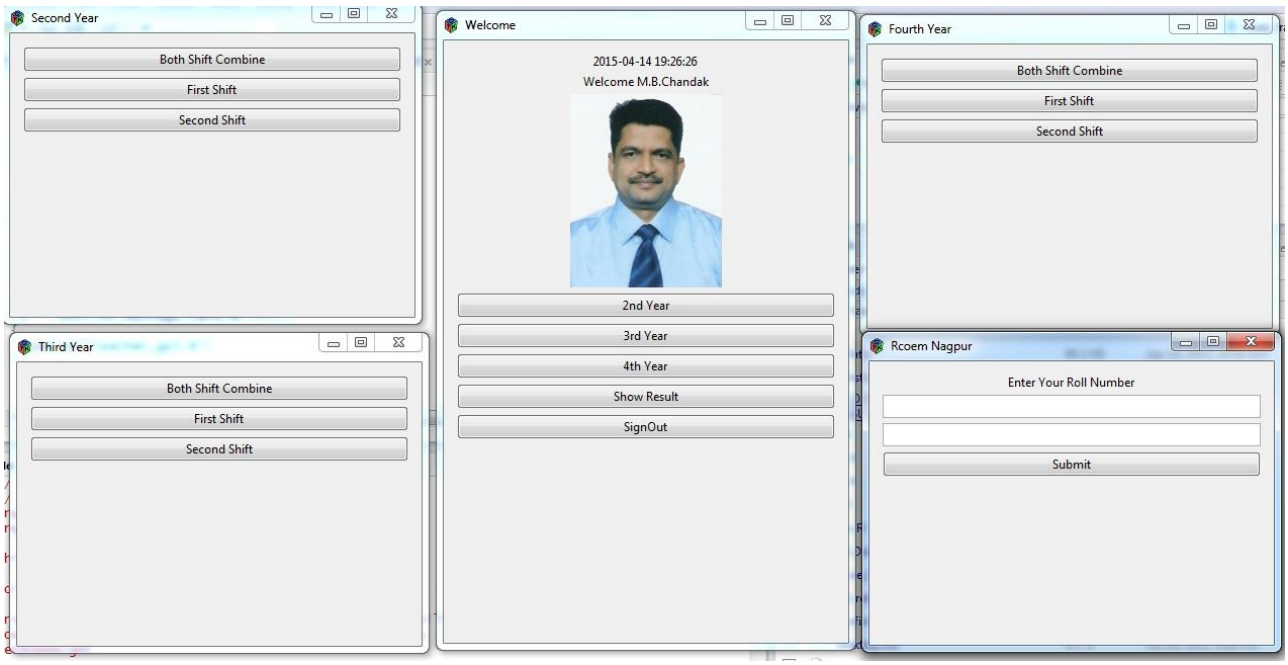


Fig 5.5: HOD Sir all Privileges windows

Step 4:

Sir will get following output on selecting the particular data. Suppose HOD sir wants to see the data analysis of Fourth year CSE and wants to compare the result of both shift, then after clicking on button of "Both shift Combine" then he will get the BOX-Plot graph of both shift of final year. After generating the output. The information is inferred that 1st shift student have box plot region from 7.5-8.5 that means 50% of students comes under this range while 2nd shift Box plot range is from 6.0-7.5 that means 50% of students comes under this range. As well as it is inferred that their is a presence of outliers in 1st shift at 5.90, it means that student doesn't comes under the range of 1st shift so special attention is needed for that student in order to improve his/her performance.

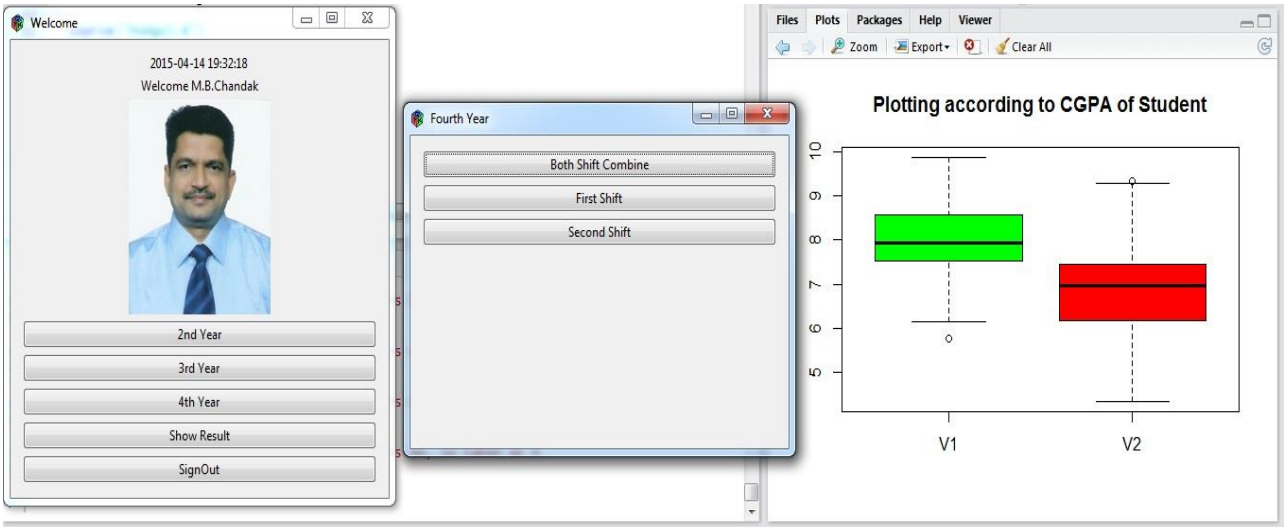


Fig 5.6 Result generated to HOD Sir

Step 5:

Similarly like that of HOD sir any Teaching staff or Professor can use this, for analyzing students performance by loginning it. For example below given are teacher user-id than the output they will be getting after loginning. However professor are given specific privileges to analysis data of a particular class which he/she teaches. Fig 5.7(a) and 5.7(b) are windows that will be displayed after professor level loginning.

Suppose User has logined with USER-id AARTI_KARANDIKAR then it will check which class he/she teaches to then depending on the privileges, he/she can analysis, as here in this case mam teaches to 4th yr 1st shift she can analysis using boxplot that what is range of 50% students and which students are falls under outliers and what is the median of class. This report which is generated if user wants he/she can mailed it to HOD sir by clicking on "MAIL O/P to HOD", and the mail containing the output in .png format will be mailed.

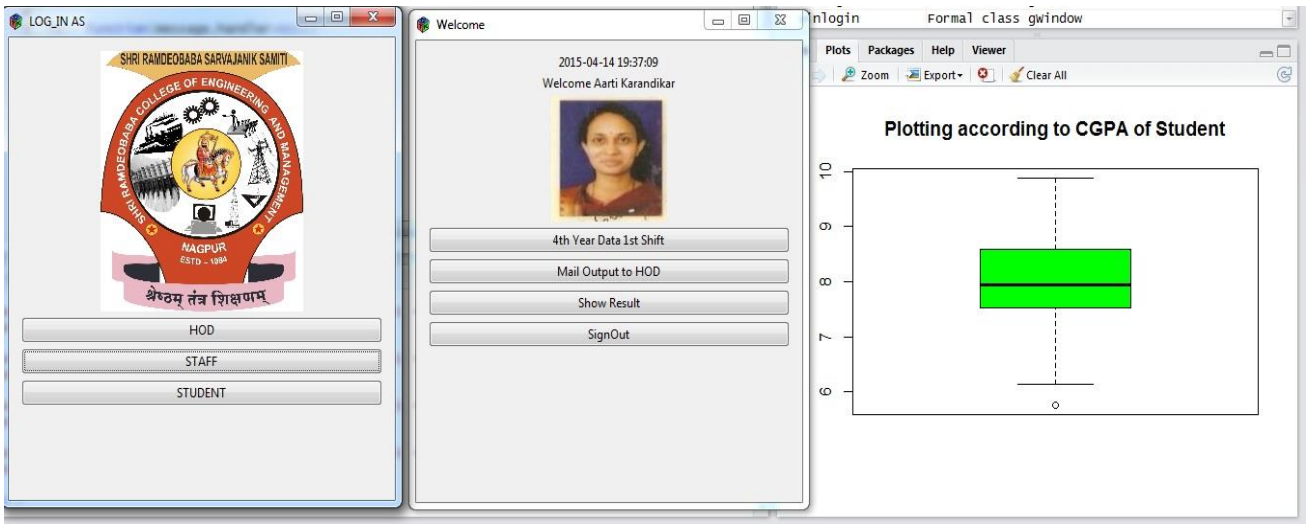


Fig 5.7(a): Output given to professor of Particular subject (Karandikar Mam)

Step 6:

Suppose user logged on using CLASS_TEACHER's ID shown in fig5.7(b) e.g. ABHIJEET RAIPURKAR sir which is currently class teacher of 4th year 2nd Shift, then he can too view performance of individual student as well as performance of whole class can analysis, and similarly all above information like min-max range outliers' median can be analyzed and that can be mailed to HOD sir if needed,

Fig 5.7(b): Output given to professor of particular Class (Abhijeet Raipurkar)

Source Code(mailing)

```
library(mailR)
send.mail(from = "Sambhav.chopda@gmail.com",
to = c("Sambhav.chopda@gmail.com"),
subject = "Project Successful",
body = "Send mail",
smtp = list(host.name = "smtp.gmail.com", port = 465,
user.name = "projectcse440@gmail.com", passwd =
"group3333", ssl = TRUE),
authenticate = TRUE,
##smtp = list(host.name = "aspmx.l.google.com", port =
25),
##authenticate = FALSE,
send = TRUE,
attach.files = c("c:/mygraphs/myplot65.jpg"),debug=TRUE
          )
```

Source code :Teacher gui

```
options(toolkit="RGtk2")
library(gWidgets)
library(devtools)
fgoto<-function(message,handler=NULL)
{    source("col_retrieve_on_ip.R") }

fmail<- function(message,handler=NULL)
{   source("final_mailed_code.R") }

f11<-function(message,handler=NULL)
{    source("Sambhav_42_cgpa.R")   }

fvalidate<- function(message,handler=NULL)
{
  if(svalue(obj1)=="raipurkar")
  {
    if(svalue(obj3)=="cse")
    {
      dispose(winlogin)
      winlogin1 <- gwindow("Welcome", visible=TRUE)
      group1<-ggroup(horizontal = FALSE, container=winlogin1,
toolkit=guiToolkit())
      obj10 <-
glabel(text=Sys.time(),container=group1,toolkit=guiToolkit())
      obj10 <- glabel(text="Welcome
Raipurkar",container=group1,toolkit=guiToolkit())
      image1= gimage(filename =
"C:\\Users\\Sambhav\\Desktop\\abhi.png",size
="large_toolbar",container = group1,toolkit=guiToolkit())


      button1<-gbutton(text="4th Year Data Shift
2",border=TRUE,handler=f11,container=group1,toolkit=guiToolkit())
      button3<- gbutton(text="Show
Result",border=TRUE,handler=fgoto,container=group1,toolkit=guiTool
kit())
      button4<- gbutton(text="Mail Output to
HOD",border=TRUE,handler=fmail,container=group1,toolkit=guiToolkit
())

   flogout<-function(message,handler=NULL)
      {       dispose(winlogin1)}

button2<-
gbutton(text="SignOut",border=TRUE,handler=flogout,container=group
1,toolkit=guiToolkit()) }
```

```
else{
gconfirm("Wrong Id and
Password",title="Ok",icon=c("error"),toolkit=guiToolkit())
} }
else
{
if(svalue(obj1)=="karandikar")
  {
    if(svalue(obj3)=="csecse")
    {
      dispose(winlogin)
      winlogin1 <- gwindow("Welcome", visible=TRUE)
      group1<-ggroup(horizontal = FALSE, container=winlogin1,
toolkit=guiToolkit())
      obj10 <-
glabel(text=Sys.time(),container=group1,toolkit=guiToolkit())
      obj10 <- glabel(text="Welcome Aarti
Karandikar",container=group1,toolkit=guiToolkit())
      image1= gimage(filename =
"C:\\Users\\Sambhav\\Desktop\\karandikar.png",size
="large_toolbar",container = group1,toolkit=guiToolkit())

fk <-function(message,handler=NULL)
      {    source("Sambhav_41_cgpa.R")     }

button1<-gbutton(text="4th Year Data 1st
Shift",border=TRUE,handler=fk,container=group1,toolkit=guiToolkit
())
      button4<- gbutton(text="Mail Output to
HOD",border=TRUE,handler=fmail,container=group1,toolkit=guiToolki
t())
      button3<- gbutton(text="Show
Result",border=TRUE,handler=fgoto,container=group1,toolkit=guiToo
lkit())
      flogout<-function(message,handler=NULL)
      {    dispose(winlogin1)    }
button2<-
gbutton(text="SignOut",border=TRUE,handler=flogout,container=grou
p1,toolkit=guiToolkit())
}
else
  {  gconfirm("Wrong Id and
Password",title="Ok",icon=c("error"),toolkit=guiToolkit()  }

}}}
winlogin <- gwindow("Rcoem Nagpur", visible=TRUE)
group<-ggroup(horizontal = FALSE, container=winlogin,
toolkit=guiToolkit())
obj <-
glabel(text="Username",container=group,toolkit=guiToolkit())
obj1 <- gedit("", container=group,toolkit=guiToolkit())
obj2 <-
glabel(text="Password",container=group,toolkit=guiToolkit())
obj3 <- gedit("", container=group,toolkit=guiToolkit())
```

Step 7:

If a student individually wants to checks his/her academic Performance then he can do that so with same way by logining,, selecting “VIEW MY PERFORMANCE” in from window. Then boxplot using his/her SGPA will be generated in an Output. Fig 5.8 shows the output of loggin through Student user.

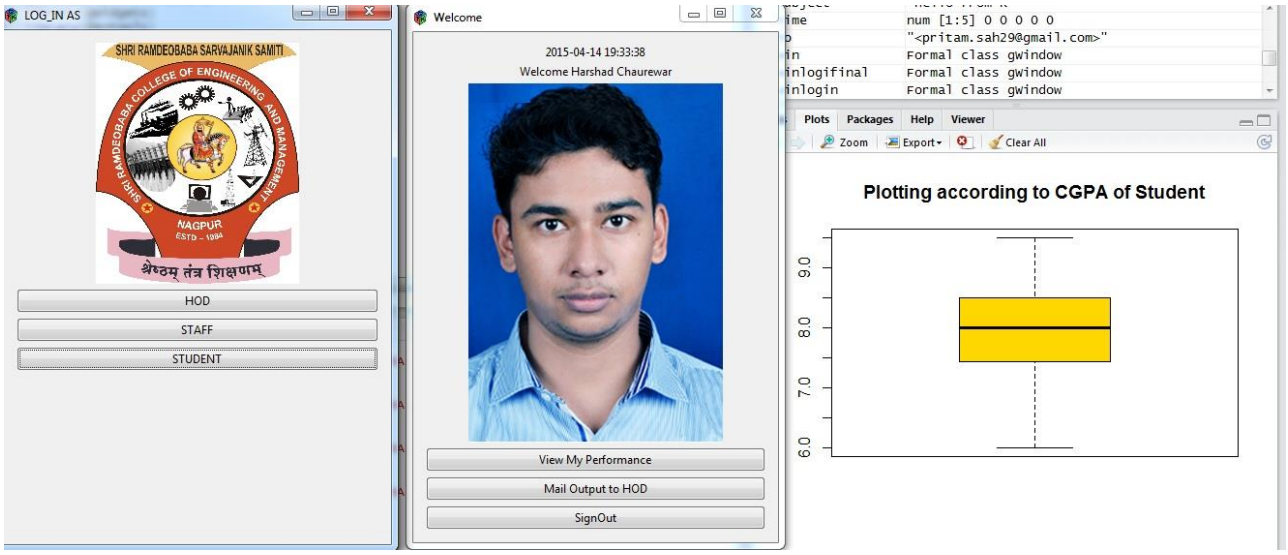


Fig 5.8 : Output given to student

Source Code:

```
options(toolkit="RGtk2")
library(gWidgets)
library(devtools)
library(png)
fgoto<-function(message,handler=NULL)
{     source("col_retrieve_on_ip.R")
      }
f11<-function(message,handler=NULL)
{     source("stud1.R")   }

fmail<-function(message,handler=NULL)
{     source("final_mailed_code.R")    }


fvalidate<- function(message,handler=NULL)
{
  if(svalue(obj1)=="harshad")
  {
    if(svalue(obj3)=="harshad")
    {
       winlogin1 <- gwindow("Welcome", visible=TRUE)
```

```
group1<-ggroup(horizontal = FALSE, container=winlogin1,
toolkit=guiToolkit())

      obj10 <-
glabel(text=Sys.time(),container=group1,toolkit=guiToolkit())
      obj10 <- glabel(text="Welcome Harshad
Chaurewar",container=group1,toolkit=guiToolkit())
      image1= gimage(filename = "C:\\ harshad.png",size
="large_toolbar",container = group1,toolkit=guiToolkit())

button1<-gbutton(text="View My
Performance",border=TRUE,handler=f11,container=group1,toolkit=guiT
oolkit())
      button2<-gbutton(text="Mail Output to
HOD",border=TRUE,handler=fmail,container=group1,toolkit=guiToolkit
())

##button3<- gbutton(text="Show
Result",border=TRUE,handler=fgoto,container=group1,toolkit=guiTool
kit())
      flogout<-function(message,handler=NULL)
      {
        dispose(winlogin1)
      }

button2<-
gbutton(text="SignOut",border=TRUE,handler=flogout,container=group
1,toolkit=guiToolkit())
   }
else
    {gconfirm("Wrong Id and
Passwors",title="Ok",icon=c("error"),toolkit=guiToolkit())  }
  }
else
  {  gconfirm("Wrong Id and
Passwors",title="Ok",icon=c("error"),toolkit=guiToolkit()) }
}
winlogin <- gwindow("Rcoem Nagpur", visible=TRUE)
group<-ggroup(horizontal = FALSE, container=winlogin,
toolkit=guiToolkit())
obj <-
glabel(text="Username",container=group,toolkit=guiToolkit())
obj1 <- gedit("", container=group,toolkit=guiToolkit())
obj2 <-
glabel(text="Password",container=group,toolkit=guiToolkit())
obj3 <- gedit("", container=group,toolkit=guiToolkit())
visible(obj3)<- FALSE

button1<-
gbutton(text="Login",border=TRUE,handler=fvalidate,container=group
,toolkit=guiToolkit())
```

Source Code for K-Mean Algorithm :

```
options(toolkit="RGtk2")
library(gWidgets)
library(devtools)
library(png)


require(XLConnect)
Sambhav <- function (x, centers, iter.max = 10, nstart = 1, algorithm
= c("Hartigan-Wong",

"Lloyd", "Forgy", "MacQueen"), trace = FALSE)
{

  getAnywhere("C_kmns")
  do_one <- function(nmeth) {
    switch(nmeth, {
      isteps.Qtran <- 50 * m
      iTran <- c(as.integer(isteps.Qtran), integer(max(0,
                                                       k - 1)))
      Z <- .Fortran(stats:::C_kmns, x, m, p, centers = centers,
                    as.integer(k), c1 = integer(m), c2 = integer(m),
                    nc = integer(k), double(k), double(k), ncp =
integer(k),
                    D = double(m), iTran = iTran, live = integer(k),
                    iter = iter.max, wss = double(k), ifault =
as.integer(trace))
      switch(Z$ifault, stop("empty cluster: try a better set of
initial centers",
call. = FALSE), Z$iter <- max(Z$iter, iter.max + 1L), stop("number of
cluster centres must lie between 1 and nrow(x)",
call. = FALSE), warning(gettextf("Quick-TRANSfer stage steps exceeded
maximum (= %d)",

isteps.Qtran), call. = FALSE))
    }, {
      Z <- .C(C_kmeans_Lloyd, x, m, p, centers = centers,
              k, c1 = integer(m), iter = iter.max, nc = integer(k),
              wss = double(k))
    }, {
      Z <- .C(C_kmeans_MacQueen, x, m, p, centers =
as.double(centers),
              k, c1 = integer(m), iter = iter.max, nc = integer(k),
              wss = double(k))
    })
    if (m23 <- any(nmeth == c(2L, 3L))) {
      if (any(Z$nc == 0))
```

```
if (m23)
        Z$ifault <- 2L
    }
    if (nmeth %in% c(2L, 3L)) {
      if (any(Z$nc == 0))
        warning("empty cluster: try a better set of initial centers",
                call. = FALSE)
    }
    Z
  }
  x <- as.matrix(x)
  m <- as.integer(nrow(x))
  if (is.na(m))
    stop("invalid nrow(x)")
  p <- as.integer(ncol(x))
  if (is.na(p))
    stop("invalid ncol(x)")
  if (missing(centers))
    stop("'centers' must be a number or a matrix")
  nmeth <- switch(match.arg(algorithm), `Hartigan-Wong` = 1,
                  Lloyd = 2, Forgy = 2, MacQueen = 3)
  storage.mode(x) <- "double"
  if (length(centers) == 1L) {
    if (centers == 1)
      nmeth <- 3
    k <- centers
    if (nstart == 1)
      centers <- x[sample.int(m, k), , drop = FALSE]
    if (nstart >= 2 || any(duplicated(centers))) {
      cn <- unique(x)
      mm <- nrow(cn)
      if (mm < k)
        stop("more cluster centers than distinct data points.")
      centers <- cn[sample.int(mm, k), , drop = FALSE]
    }
  }
  else {
    centers <- as.matrix(centers)
    if (any(duplicated(centers)))
      stop("initial centers are not distinct")
    cn <- NULL
    k <- nrow(centers)
    if (m < k)
      stop("more cluster centers than data points")
  }
  k <- as.integer(k)
  if (is.na(k))
    stop("'invalid value of 'k'")
  iter.max <- as.integer(iter.max)
  if (is.na(iter.max) || iter.max < 1)
    stop("'iter.max' must be positive")
  if (ncol(x) != ncol(centers))
```

```
stop("must have same number of columns in 'x' and 'centers'")
  storage.mode(centers) <- "double"
  Z <- do_one(nmeth)
  best <- sum(Z$wss)
  if (nstart >= 2 && !is.null(cn))
    for (i in 2:nstart) {
      centers <- cn[sample.int(mm, k), , drop = FALSE]
      ZZ <- do_one(nmeth)
      if ((z <- sum(ZZ$wss)) < best) {
        Z <- ZZ
        best <- z
      }
    }
  centers <- matrix(Z$centers, k)
  dimnames(centers) <- list(1L:k, dimnames(x)[[2L]])
  cluster <- Z$c1
  if (!is.null(rn <- rownames(x)))
    names(cluster) <- rn
  totss <- sum(scale(x, scale = FALSE)^2)
  structure(list(cluster = cluster, centers = centers, totss = totss,
                 withinss = Z$wss, tot.withinss = best, betweenss = totss
-
                   best, size = Z$nc, iter = Z$iter, ifault = Z$ifault),
            class = "kmeans")
}

MyData <- read.csv(file="G:\\Study_Materials\\8th
Sem\\Project\\Code_Practice\\Final Year\\1.COMBINE SUCCESSFULLY AND
UNSUCCEFULLY 8TH SEM SGPA.csv",header=FALSE)
print(MyData)

MyData.features = MyData
print(MyData.features)
MyData.features$class <- NULL
print(MyData.features)
MyData <- na.omit(MyData)
results <- Sambhav(MyData.features , 2)
print(results)
boxplot(MyData[c("V1","V2")],col=(c("green","red")),main="Plotting
according to CGPA of Student")
##,col=results$cluster,col=C("gold","darkgreen"))
jpeg("c:/mygraphs/myplot65.jpg")
boxplot(MyData[c("V1","V2")],col=(c("green","red")),main="Plotting
according to CGPA of Student")
dev.off()
```

Step 8:

After applying the above k-mean algorithm, output in the form of Box-plot is generated. Below fig 5.9 shows how output is generated after applying k-means algorithm on data set. Fig 5.9, is the box-plot generated of 2nd year batch (result of winter-14), in this we come to infer that both class majority i.e.50% student falls under the cgpa of 6.5-7.0 to 8.5-9.0 range. As well as are many student whose cgpa doesn't falls under expected range so they comes under outlier lists.

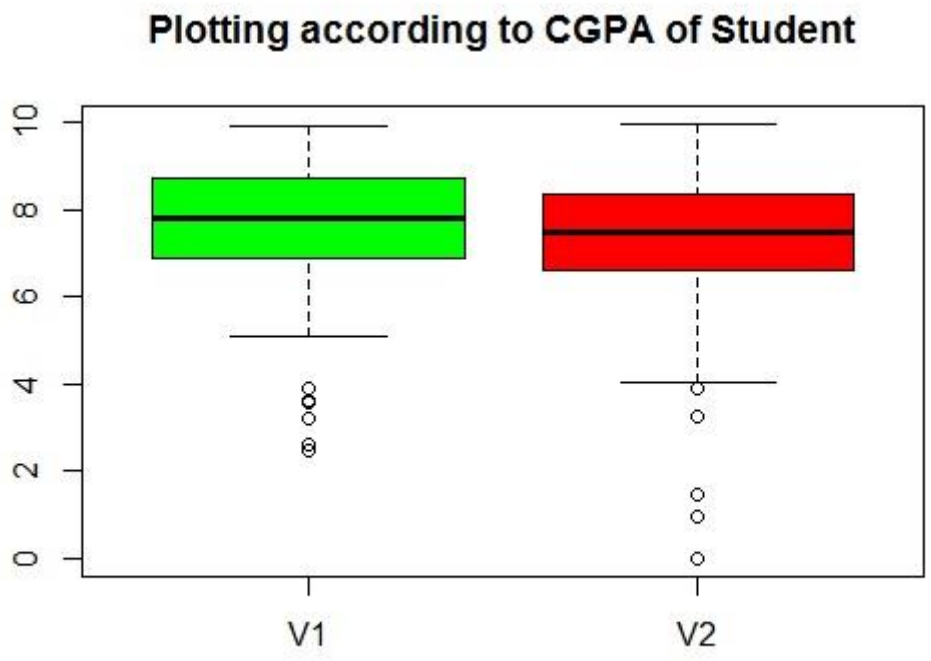


Fig 5.9: Result Generated for both shift of 3rd semester (win-14)

Chapter 6
Conclusion
&
Future Work

6.1 Conclusion

Thus, we have implemented the student monitoring system using R programming. The algorithm which we have used i.e. the k-means clustering algorithm works efficiently. This algorithm creates the clusters based on the student CGPA (Cumulative Grade Point Average). We have used 3 clusters they are:- students below average, average students and students above average. For this, we have used the Box plot representation (also called as five point summary) which is a graphical representation of the data. Also, the box plot representation maps the outliers in the graphs which are the students whose CGPA doesn't fall under the acceptable range of min-max students in the class.

A limitation of this algorithm is that the user needs to give the value of *k*, which is the number of clusters needs to be given as input regardless of the distribution of data points. Also it may contain dead unit problem, that is if the cluster center is wrongly chosen, it may never be updated and thus never represent a class.

In summary, it is implemented correctly in conjunction with the methodologies of data mining which helps the academic planners to plan the academic studies accordingly. Also the students can know about their current CGPA and target SGPA that the student needs to achieve to reach the targeted CGPA in the last semester.

6.2 Future Work

Evolving some statistical methods to compute the value of *k*, depending on the data distribution, is suggested for future research. Methods for refining the computation of initial centroids is worth investigating.

Chapter 7
References

7.0 References

[1] S. Sujit Sansgiry, M. Bhosle, and K. Sail, "Factors that affect academic performance among pharmacy students," American Journal of Pharmaceutical Education, 2006.

[2] Susmita Datta and Somnath Datta, "Comparisons and validation of statistical clustering techniques for microarray gene expression data," Bioinformatics, vol. 19, pp.459–466, 2003.

[3] Rousseeuw P. J, "A graphical aid to the interpretation and validation of cluster analysis," Journal of Computational Appl Math, vol 20, pp. 53– 65, 1987.

[4] Sharmir R. and Sharan R., "Algorithmic approaches to clustering gene expression data," In current Topics in Computational Molecular Biology MIT Press; pp. 53-65, 2002.

[5] Mucha H. J., "Adaptive cluster analysis, classification and multivarite graphics,"Weirstrass Institute for Applied Analysis and Stochastics, 1992.

[6] Fahim A. M., Salem A. M., Torkey F. A. and Ramadan M. A., "An efficient enhanced k-means clustering algorithm," Journal of Zhejiang University Science A., pp. 1626–1633, 2006

[7] J. O. Omolehin, J. O. Oyelade, O. O. Ojeniyi and K. Rauf, "Application of Fuzzy logic in decision making on students' academic performance," Bulletin of Pure and Applied Sciences, vol. 24E(2), pp. 281-187, 2005.

[8] J. O. Omolehin, A. O. Enikuomehin, R. G. Jimoh and K. Rauf, "Profile of conjugate gradient method algorithm on the performance appraisal for a fuzzy system," African Journal of Mathematics and Computer Science Research," vol. 2(3), pp. 030-037, 2009.

[9] N. V. Anand Kumar and G. V. Uma, "Improving Academic Performance of Students by Applying Data Mining Technique," European Journal of Scientific Research, vol. 34(4), 2009.