FOSSEE Summer Fellowship Report

On

FLOSS - R

Submitted by

Digvijay Singh (Noida Institute of Engineering and Technology, Gr. Noida)
Diksha Bothra (Poornima College of Engineering, Jaipur)
Meet Bhatnagar (Vellore Institute of Technology, Vellore)
Neeraj Kumar (Amity University, Rajasthan)

Under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

July 29, 2019

Acknowledgment

We want to express our sincere gratitude to Prof. Kannan M. Moudgalya, Department of Chemical Engineering, IIT Bombay for providing their invaluable guidance, comments, and suggestions throughout the project. We equally respect the contribution of Prof. Pennan Chinnasamy, Centre for Technology Alternatives for Rural Areas (CTARA), IIT Bombay for providing us with 111 years of Indian Meteorological Department's data on rainfall and Prof. K. Narayanan, Department of Humanities and Social Sciences, IIT Bombay along with Dr. Amarendra Singh, Postdoctoral Fellow, Department of Humanities and Social Sciences, IIT Bombay for guiding us through the process of rainfall data analysis. Also, we would like to thank Mrs. Smita Wangikar for her advice and assistance in explaining the statistics required for rainfall data analysis, Mr. Sudhakar Kumar for providing us with an overview of Beamer, LaTeX and version control via Git and Mr. Tushar Bajaj for teaching R Shiny and assistance in the creation of R Shiny applications.

Contents

Chapter 1

Introduction

In this report, we mention our contributions to open-source software (FLOSS) which were made in the duration of the FOSSEE Fellowship, starting from 21st May 2019 to 12th July 2019. Contributions are made using a Free-Libre / Open Source Software (FLOSS) known as 'R' as a part of the FOSSEE Project by IIT Bombay and MHRD, Government of India. FOSSEE project is a part of the National Mission on Education through ICT. The thrust area is the adaptation and deployment of open-source simulation packages equivalent to proprietary software, funded by MHRD, based at the Indian Institute of Technology Bombay (IITB). Our contributions involved Textbook Companion, analysis of the development of various districts in India, analysis followed by forecasting of Rainfall for 594 districts in India, and analysis of financial data from the Ranchi district.

Chapter 2

Textbook Companion

As a part of the selection procedure, each FOSSEE participant was supposed to select a standard textbook related to the domain of Probability, Statistics, and Algebra. Each textbook must contain at least 80 solved examples. Later each participant is supposed to propose his/her selected book on R FOSSEE website. After approval, code all the solved examples in R. Upload all coded examples on the official R FOSSEE web portal for TBC submission. If all of the coded examples are up to the required standard guidelines and working without error then that particular book's codes get published on the official FOSSEE website. In case of a mistake in the code. The participant is supposed to review the code and resubmit it on the same web portal. If this review and resubmit process exceeds beyond five cycles, then all of the codes get rejected, and the proposal gets deleted from the website.

Selected FOSSEE fellows, after submission of their TBCs, were supposed to review the TBCs of other participants. Below is a list of TBCs submitted by fellows and the TBCs reviewed by fellows.

2.1 TBCs submitted by fellows -

S.No	Fellow	Textbook name	Author	Edition
1	**Digvijay Singh**	Probability And Statistics	Degroot and Schervish	4th
2	**Diksha Bothra**	Statistics and Probability Theory	Dr. K.C. Jain and Dr. M.L. Rawat	10th
3	**Meet Bhatnagar**	Fundamentals of Mathematical Statistics	S.C. Gupta and V.K. Kapoor	11th
4	**Neeraj Kumar**	Probability, Random Variables, and Stochastic Processes	A. Papoulis and S. U. Pillai	4th

2.2 TBCs reviewed by fellows[1] -

S.No	Participant	Textbook name	Author	Edition
1	**Ajay Ragh**	Linear Algebra	Jim Hefferon	3rd
2	**Anjana Rajagopal**	Biostatistics - Basic Concepts and Methodology for the Health Sciences	D. W. Wayne and C. L. Cross	10th
3	**Ashiq Mehmood Asharaf**	Introductory Linear Algebra: An Applied First Course	B. Kolman and D. R. Hill	8th
4	**Aswin S**	Matrices And Linear Transformations	C. G. Cullen	2nd
5	**Devdatt**	Fundamentals of Matrix Algebra	Gregory Hartman	3rd
6	**Midhun C. Kachhapalli**	Introduction To Probability	Dimitri P. Bertsekas And John N. Tsitsiklis	2nd
7	**Nivedha R**	Statistics In Education And Psychology	P. C. Dash and Bhabhagrahi Biswal	2nd
8	**Sai Sugun L**	Linear Algebra and Its Applications	David C. Lay	3rd

[1]Diksha Bothra and Meet Bhatnagar reviewed the submitted TBCs.

Chapter 3

District Reports - Data Collection and Analysis

It was designated to each FOSSEE applicant to complete one more task other than TBC. The task was to collect data regarding specific topics like Crime, Electricity, and Education, to name a few, from his or her respective district where he or she resides. These topics are known as variables. Each participant received a list of 23 such variables. The participants were also requested to analyze data on as many variables as they can and later create a report on their findings using Rmarkdown. This data analysis report creation was one of the two selection tasks for the FOSSEE Fellowship. Then the selected fellows[2] were suggested to improve their reports as well as the reports submitted by other participants. Below is a list of all of the districts, variables analyzed, and names of the respective report analyzers.

[2]Neeraj Kumar improved the submitted district reports of non-selected participants.

3.1 List of districts and variables analyzed -

S.No	District	Variables Analyzed	Analyzed by
1	**Delhi**	GDP, Employment and Earnings, Fair Price Shops, Courts, Electricity, Healthcare, Tourism, Malnutrition Data, Industries, Education, Green Coverage, Krishi Vigyan Kendra, Vehicles, Usage of Computing Devices	Meet Bhatnagar
2	**Meerut**	Employment and Earnings, Fair Price Shops, Courts, Electricity, Tourism, Swachh Bharat Mission Implementation, Population and Literacy, Industries, Education, Kisan Credit Card, Loan, Food Processing, Vehicles	Digvijay Singh
3	**Jaipur**	GDP, Employment and Crimes, Fair Price Shops, Courts, Fire Incidents, Electricity, Healthcare, Tourism, Industries, Swachh Bharat Mission Implementation, BPL Households, Own College Data, Green Roof Initiative and Rain Water Harvesting, Population, Education, Green Coverage in Jaipur, Kisan Credit Card, Loans Issued, Krishi Vigyan Kendra, Irrigation and Vehicles	Diksha Bothra and Neeraj Kumar
4	**Coimbatore**	Electricity, Fair Price Shops, Per Capita Income and GDP and Rainfall	Nivedha R
5	**Ahmedabad**	Population, Fair Price Shops, Electricity, Healthcare, Court and Rainfall	Jaini Patel
6	**Tiruvallur**	Irrigation, Electricity and Education	Shankar D K
7	**Varanasi**	Education and Usage of Computing Devices	Ajay Kumar
8	**Virudhunagar**	Education, Fair Price Shops and Crime Rate	Chowmya Rajakumar
9	**Mumbai**	Population and Industry	Lalith Dupathi
10	**Medchal**	Agriculture	Amulya Reddy

Chapter 4

Rainfall Data Analysis

Rainfall data were analyzed, which was made available by the Indian Meteorological Department. Data contained the amount of rainfall in mm for 596 districts in India for 111 years, i.e., from 1901 to 2011. The following flowchart briefly explains the procedure involved in the analysis of the rainfall data.

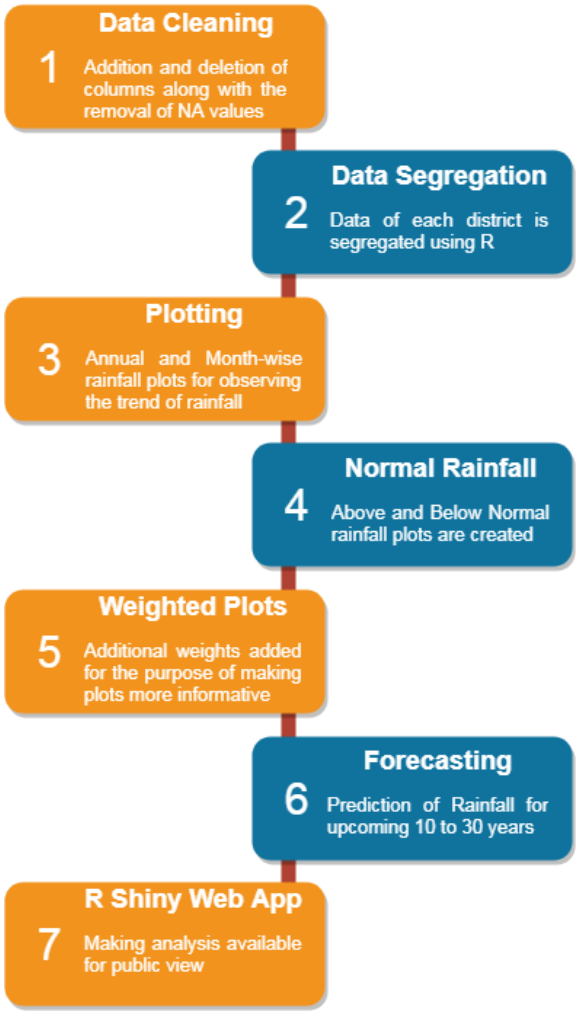


Figure 4.1: Rainfall Data Analysis Flowchart

4.1 Data Cleaning

Original data contained various NA values, which might result in errors in analysis. Hence, it was necessary to remove them. They were removed in R using **`na.omit()`** function. Various district names were also incorrect and even misleading, e.g., the names of two districts were 'West' and 'West District'. These district names provide no information regarding their respective states. Also, there was a column containing the same value throughout, i.e., 'RAIN'. As there was no need for explicitly stating that the data belongs to the amount of rainfall. Hence the column was removed. Later, the introduction of a new column containing values of the area covered by each district made the results from analysis more informative.

4.2 Data Segregation

Excel sheets contained the original data. Hence, it was necessary to read and write into an excel file using R. **`readxl`** and **`openxlsx`** libraries in R performed the required operations of reading and writing into an excel file. After reading data from the excel file, the process of segregation of each district's data into separate data frames took place. Names of these data frames were the same as that of the district whose data they contained. It was done by first extracting all of the district names from the original data using **`unique()`** function which outputs all unique entries in a vector. Later **`assign()`** function is used to assign each data frame containing a particular district's data with the name of their respective district.

Following code chunk was used to perform the operations mentioned above –

```
# Loading library "readxl" for reading in .xlsx file via R.
# Loading library "openxlsx" for writing in .xlsx file via R.
library(readxl)
library(openxlsx)
# Data from file is stored in the data frame named as 'Intermediate'.
Intermediate<-read_excel("Rainfall_Data_(111_years).xlsx")
# Removing NA values from 'Intermediate'.
Intermediate<-na.omit(Intermediate)
# Finding names of each district separately using "unique" function.
Unique<-unique(Intermediate$DIST)
# Creating a folder for storing all .xlsx files having data for each district
    seperately.
dir.create("Intermediate")
# Creating loop for writing an excel file with a particular district data.
for(i in 1:length(Unique))
{
  write.xlsx(Intermediate[which(Intermediate$DIST==Unique[i])[1]:which(Intermediate
      $DIST file=paste0("Intermediate/",Unique[i],".xlsx"),sheetName = Unique[i],
      col.names = TRUE, row.names = TRUE, append = TRUE)
}
# Reading each file and changing its rownames with years also removing all "WC"
    data.
# Storing each district's data in a separate data frame named after the district.
for(j in 1:length(Unique))
{
  Data<-read_excel(paste0("Intermediate/",Unique[j],".xlsx"))
  Data<-as.data.frame(Data)
  row.names(Data)<-Intermediate$Year[1:length(row.names(Data))]
  Data<-Data[,-1]
  assign(Unique[j],Data)
}
```

4.3 Plotting

Time-Series analysis is done to obtain an understanding of the trend of rainfall for each district. The time series analysis involved converting data frames containing rainfall data associated with each district into a time-series format using **`ts()`** function. After a successful conversion, the initial analysis involved month-wise rainfall plotting[3] by taking frequency equals to 12 for the number of months in a year. It resulted into the following –

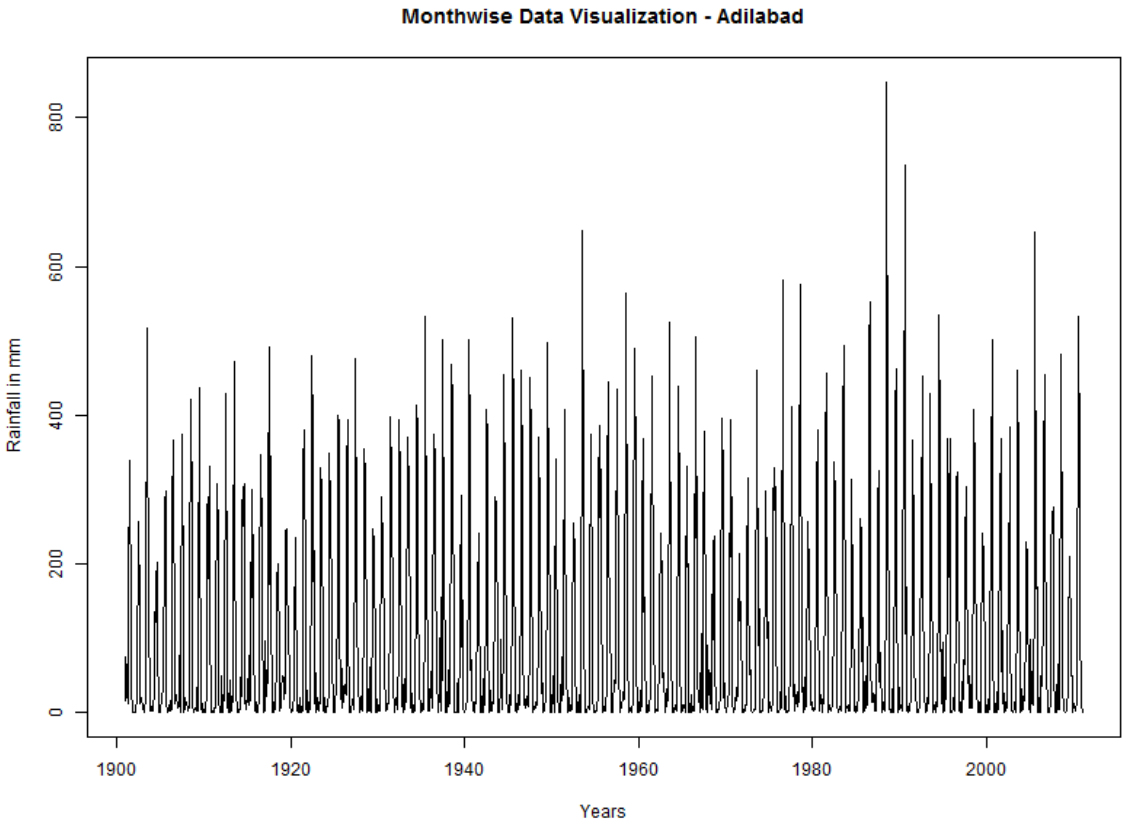


Figure 4.2: Monthwise Rainfall Data Analysis - Adilabad

Figure 4.2 represents the month-wise rainfall data analysis for 'Adilabad' district for 111 years, starting from 1901 to 2011. Due to a large number of data points. A rainfall pattern is not readily visible, and the plot obtained is too crowded. Hence to overcome this issue, instead of month-wise rainfall plotting, the analysis involved annual rainfall plotting by taking frequency equal to 1. Here annual rainfall data of a particular district is considered for obtaining the rainfall pattern in that district.

[3]All plots were created using the **`plot()`** function.

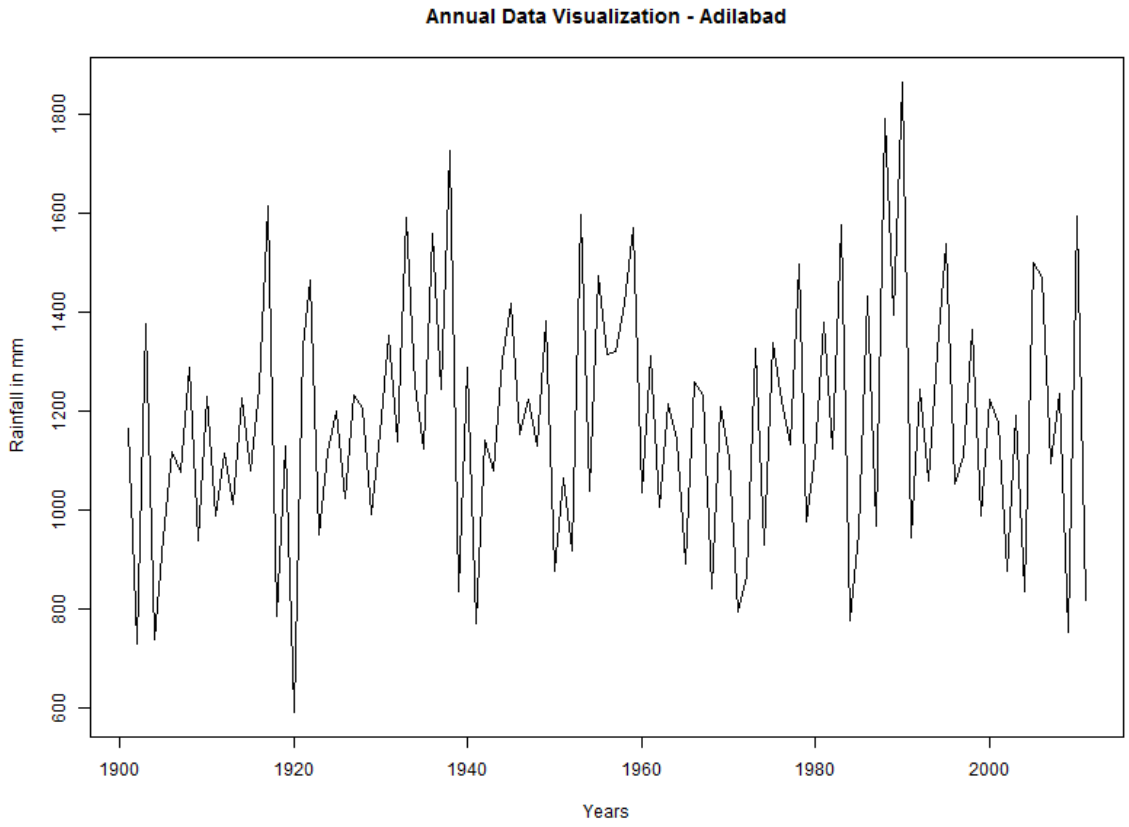


Figure 4.3: Annual Rainfall Data Analysis - Adilabad

In Figure 4.3, annual rainfall analysis is represented graphically. Here the rainfall pattern for 'Adilabad' district for 111 years, i.e. from 1901 to 2011 is more clearly visible.

The moving average analysis was performed to obtain the trend of rainfall. The analysis was performed using **`movavg()`** function from **`pracma`** library in R. The function provides an option to select various types of moving averages to be performed on the time series data. The type was chosen to be **`simple`**, and the length of the backward window was 30 years. Hence, the following output is a product of performing simple moving average analysis –

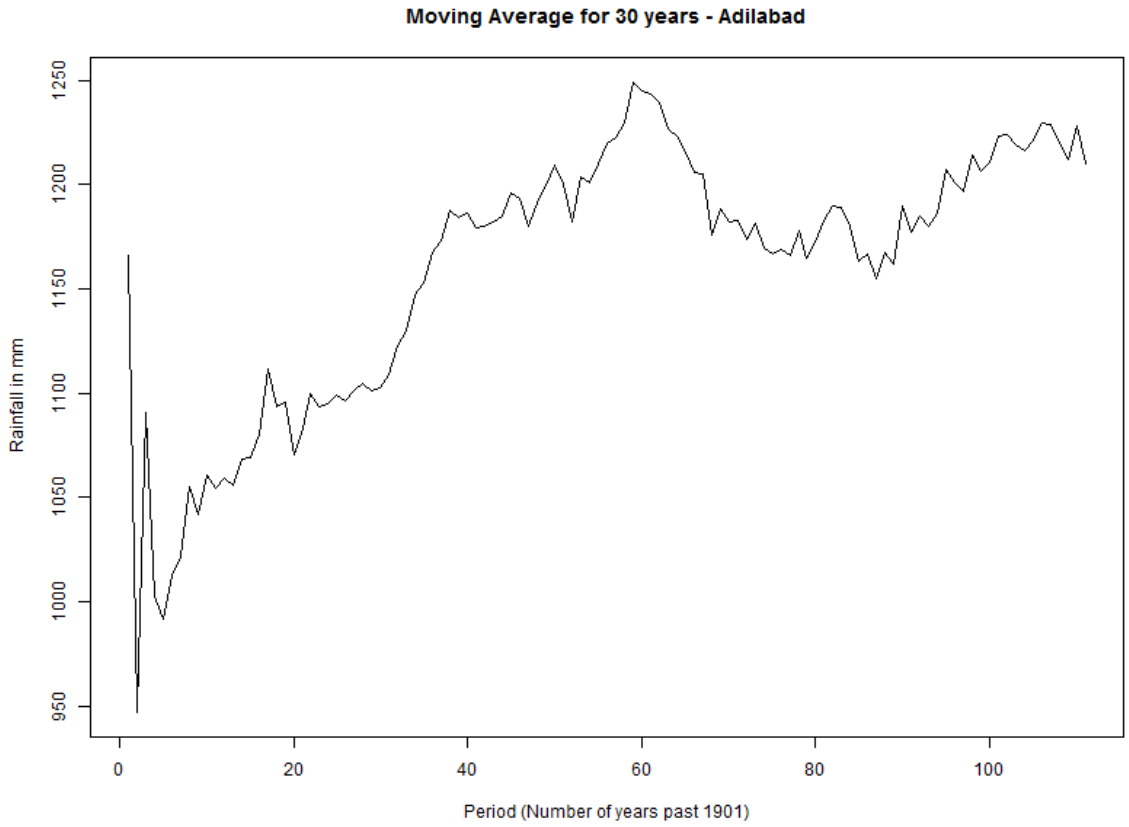


Figure 4.4: Simple Moving Average Analysis for 30 years - Adilabad

In Figure 4.4, a rainfall trend can be observed which was calculated using a simple moving average. We also observed the deviation of actual rainfall from moving average graphically in Figure 4.5 for a better understanding of the rainfall trend.

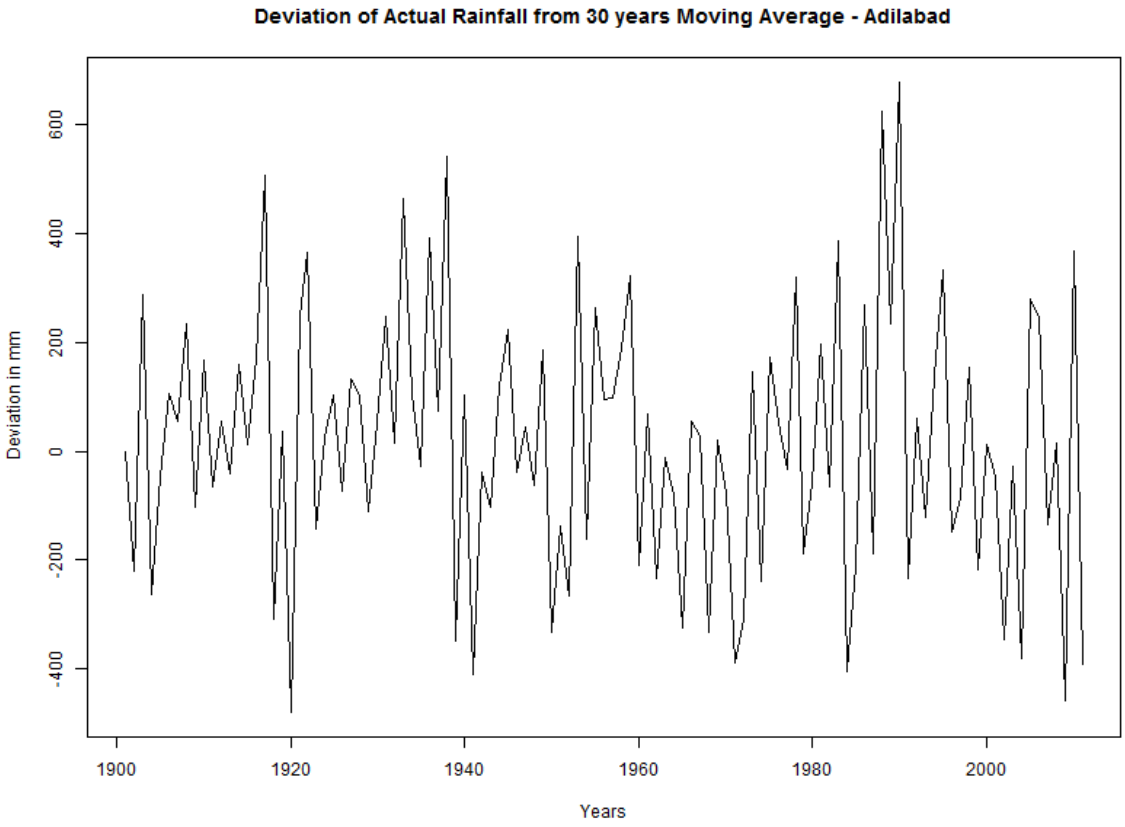


Figure 4.5: Deviation from Simple Moving Average Analysis - Adilabad

4.4 Normal Rainfall

Rainfall data analysis for each district included one more parameter, namely, frequency of above and below normal rainfall. Here, the average of all annual rainfall values for 111 years, i.e. from 1901 to 2011 is the definition of normal rainfall. Above Normal rainfall was defined as the rainfall values equal to or more than 110% of the Normal rainfall and below Normal rainfall as the rainfall values equal to or less than 90% of the Normal rainfall.

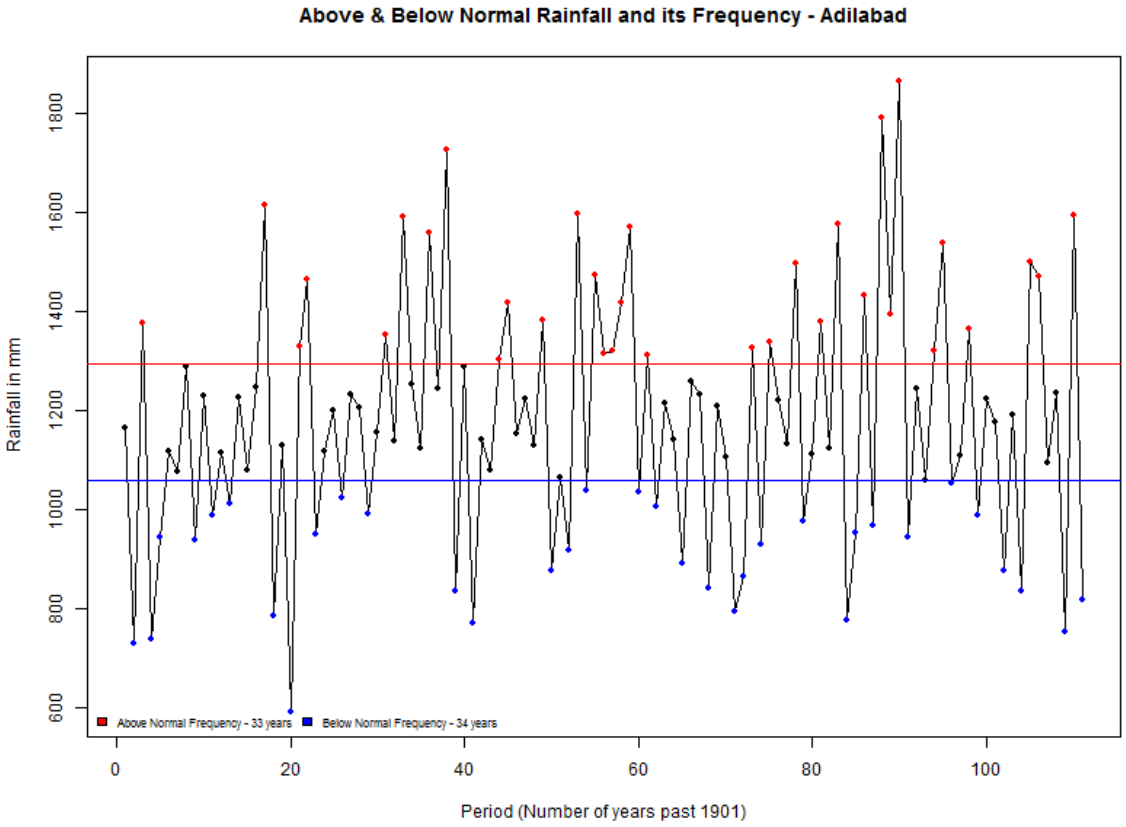


Figure 4.6: Above and Below Normal Rainfall - Adilabad

In Figure 4.6, frequency of above and below Normal rainfall can be observed for the Adilabad district. Here, data points on and above red line belong to the above Normal rainfall section; similarly, data points on and below blue line belong to the below Normal rainfall section.

4.5 Weighted Plots

These plots use different color coding and size variations of scatter points to display more information efficiently.

4.5.1 Area Weighted Plot

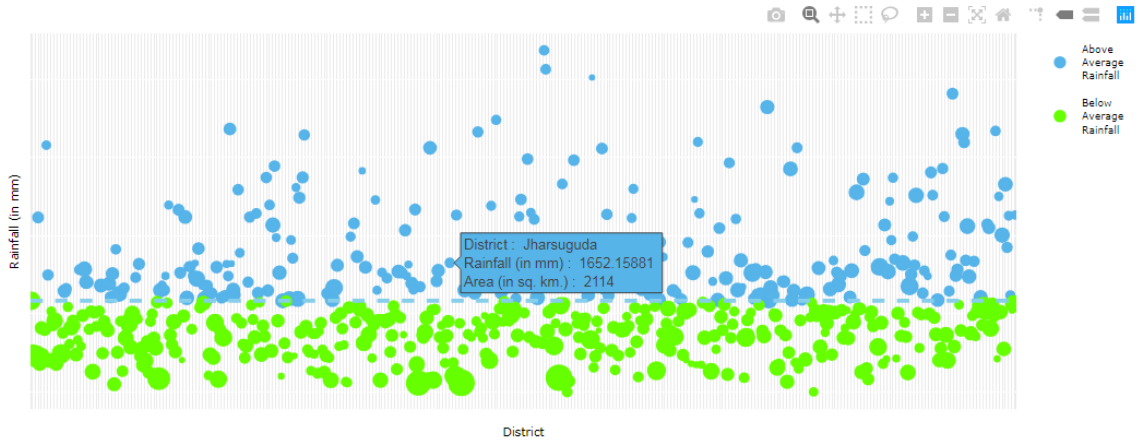


Figure 4.7: Weighted Area Plot

In Figure 4.7, a weighted area plot represents the above and below-average rainfall, district-wise, for the year '1901'. This plot represents information on the amount of rainfall and the area covered by each district.

4.6 Forecasting

The Holt-Winters Method: The method used for the forecast is time series based. This technique uses exponentially decreasing weights as the observations get older. Recent observations are given relatively more weight in forecasting than the earlier observations. Exponential Smoothing is used to generate the smoothed values to obtain estimates. Some time-series data exhibit cyclical or seasonal patterns that cannot be effectively modeled using the polynomial model. Therefore `HoltWinters()` is the best function to deal with that situation.

4.6.1 Steps Involved

For showing the prediction graph, we have taken a sample district named "**Adilabad**". In the same way, we can draw the prediction graphs for the 596 districts of India. Here are the following steps (along with code chunks) which have to be accomplished first before the plotting of the prediction graph:

1. Extracting data related to the amount of rainfall in each district from their individual excel files.

```
# Loading library "openxlsx" for reading xlsx file.
library(openxlsx)
# Reading Adilabad's rainfall data file.
e=read.xlsx("Adilabad.xlsx",sheet = 1)
# Creating data frame.
r=data.frame(e$Jan,e$Feb,e$Mar,e$Apr,e$May,e$Jun,e$Jul,e$Aug,e$Sep,e$Oct,e$
    Nov,e$Dec)
```

2. Converting the data to time series format for the future predication or forecasting using the **`ts()`** function.

```
# Conversion into time-series format.
r<-ts(as.vector(t(as.matrix(r))),frequency = 12,start=1901)
```

3. Plotting the decompose of the data using the `decompose()` and `plot()` from year 1901 to 2011. This gives 4 plots which are as follows:

- Observed
- Trend
- Seasonal
- Random

```
# Decomposing data.
plot(decompose(r))
```

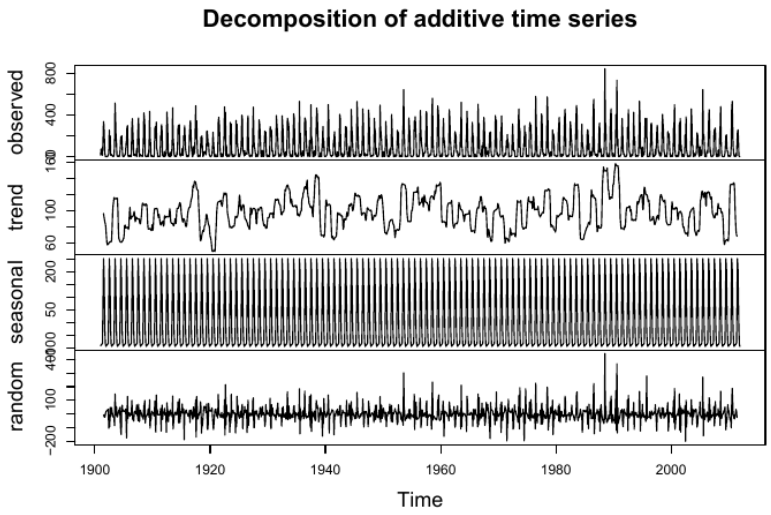


Figure 4.8: Decomposition of additive time series

4. Analysis performed by using Holt-winters function (i.e. `HoltWinters()`) because it does not require any additional packages.

```
# Estimated Holt Winters
hw2<-HoltWinters(r)
```

5. Values of `HoltWinters()` were placed into predict function (i.e. `predict()`) which does the prediction and outputs 3 values:

- Fitted value
- Upper limit
- Lower limit

```
# Calculating prediction.
hw2.pred<-predict(hw2,30*12,prediction.interval = TRUE)
```

6. Now, by inputting predicted values in the plot function of time series format (i.e. `plot.ts()`), the **forecasting** plot can be obtained.

```
# Plotting predicted values.
plot.ts(r,ylab="Rainfall in mm",xlim=c(1950,2020),ylim=c(-1000,1000))
lines(hw2$fitted[,1],lty=2,col="red")
lines(hw2.pred[,1],col="blue")
lines(hw2.pred[,2],lty=2,col="seagreen")
lines(hw2.pred[,3],lty=2,col="seagreen")
```

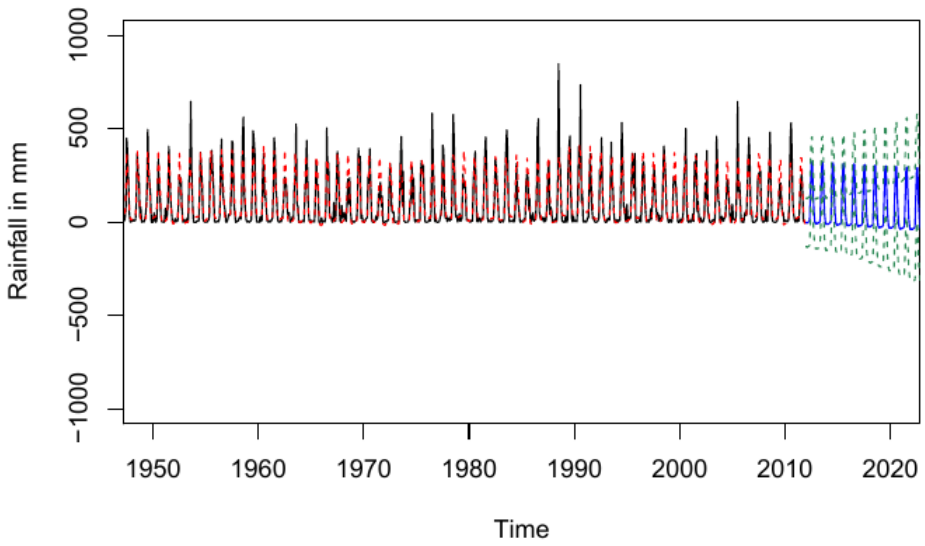


Figure 4.9: Forecasting

4.7 R Shiny Web App

4.7.1 What is R Shiny?

R Shiny is an R package that allows to make an interactive web app using R language. It enables to project all the data analysis and data vitalization capabilities of R language on the web app. It provides with the ability to do real-time data processing capabilities using a web app.

4.7.2 Working of R Shiny

R shiny mainly consists of 3 components:

- The UI component
- the server component
- the calling function

UI Component

The UI component contains all the front end of the shiny web app. It contains all the UI components like button input, slider control, radio button, and the layout of the front end. It also contains advanced customization options like multiple tabs, tabs within tabs, and much more. When this component runs, it creates a corresponding HTML implicitly for the front end that we have designed to provide us a resultant web page.

- General syntax of UI component: `ui <- fluidPage()`
- Inside `fluidPage()` function front end is designed.
- **Note**: For advance front end other functions like `dashboardPage()` in place of `fluidPage()` function can be used.

Server Component

The server component contains all the back end of the shiny web app. It contains all the processing done on data e.g. calculation of districts that received below-average rainfall. It also contains the code chunks to display processed data on the web app e.g. plots, graphs, maps, and much more. When this component runs, it creates the corresponding JavaScript implicitly to provides us the resultant functionality.

- General syntax of UI component: `server <- function(input, output){}`
- Inside curly braces, we define all the functionality of our web app.

Calling Function

The calling function is the function that calls the app. It runs the app.

- General syntax of calling function : `shinyApp(ui = ui, server = server)`

Why R Shiny?

Other platforms also provide ways to make web apps, so why use R Shiny for that? The answer is, it provides two unique features.

First, it gives a way to do real-time data manipulation, visualization, and analysis using the web app. Using its web app, one can give the user the controls to do data visualization. Users can decide the parameters. According to the user's choice, the data processing would happen in the back end.

Second, the ease of creating a web app. It provides built-in functions to create UI components and different ways to interact with them. It then creates the corresponding HTML and JavaScript implicitly to provides the web app. Hence, For rapid web app development, one may use R Shiny.

So, if someone wants to do real-time data analysis using a web app, R Shiny is the way to go.

4.7.3 Components of Shiny Web App

Libraries Used

`library(openxlsx)` : Used to read excel files directly.
`library(shiny)` : Used for implementing the R Shiny framework for web app development.
`library(leaflet)` : Used for implementing interactive maps in the web app.
`library(dplyr)` : Used for fast data manipulation from memory.
`library(magrittr)` : Used for pipelining of instructions.
`library(sf)` : Provides a standardized way to encode spatial vector data. Used for manipulation and projection of geographical data.
`library(raster)` : Used for geographic data analysis and modeling.
`library(sp)` : Provides functions for spatial data.
`library(shinydashboard)` : Used for creating dashboard layout for the web app.
`library(shinydashboardPlus)` : Used for better customizations in the dashboard.
`library(DT)` : Used for projecting table on the web app.
`library(pracma)` : Used for practical numerical math functions like time series.
`library(rgdal)` : Used for binding several packages to allow publishing of the web app.
`library(shinythemes)` : Used for adding custom themes.

UI Components

`selectInput()`: It creates a select list to choose an item from a list of values.

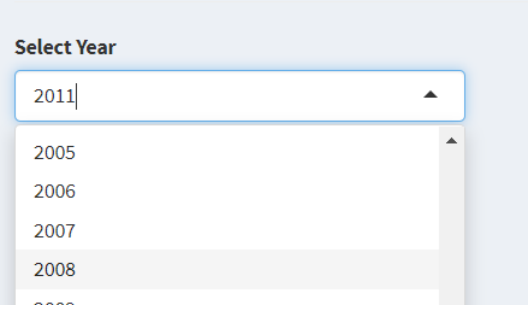


Figure 4.10: `selectInput()`

The 4.10 shows an example of `selectInput()`. This particular `selectInput()` shown in the figure is used to choose a year for which the user wants to see the visualization.

`radioButtons()`: It creates a set of radio buttons for selecting an item from a list.

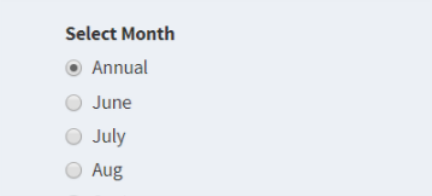


Figure 4.11: `radioButtons()`

The 4.11 shows an example of `radioButtons()`. This particular `radioButtons()` shown in the figure is used to choose a particular time for which the user wants to see the visualization.

`actionButton()`: It is used to create a button which, when clicked by the user, triggers the defined action.

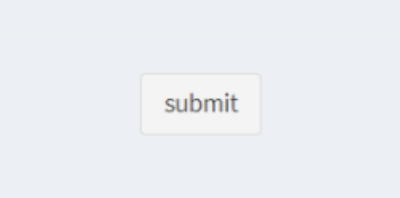


Figure 4.12: `actionButton()`

The 4.12 shows an example of an `actionButton()`. This particular `actionButton()` shown in the figure is used to gather all the parameter set by the user and project a map of India showing amount of rainfall that has occurred customized by the

parameters set by the user.

`sliderInput()`: It is used to constructs a slider to select a numeric value from a range.

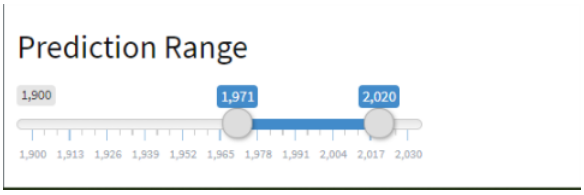


Figure 4.13: `sliderInput()`

The 4.13 shows an example of a `sliderInput()` that we have used in our web app. This particular `sliderInput()` shown in the figure is used to give the user a year range to select. Prediction of rainfall is displayed based on the selected range.

`gradientBox()`: It is a function of shinydashboardPlus. It is used to create an enhanced box.

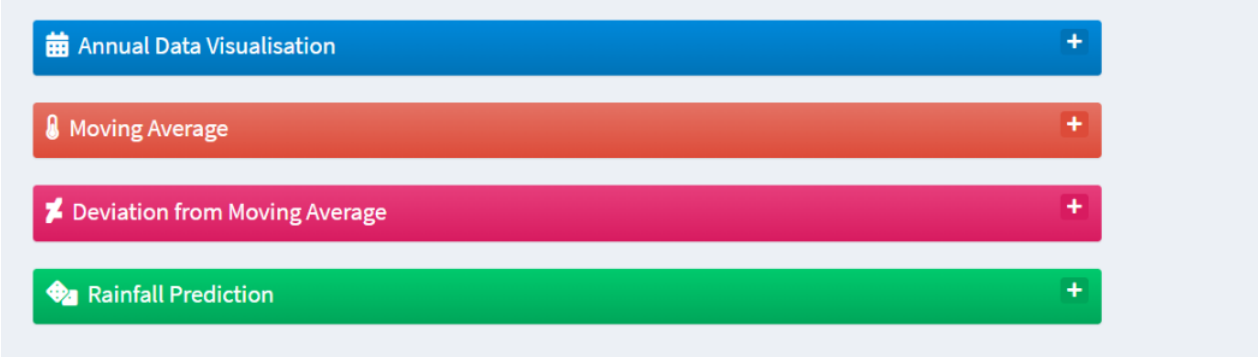


Figure 4.14: `sliderInput()`

The 4.14 shows an example of a `gradientBox()`. 4 gradient boxes are present in the figure. They display 4 different plots when expanded by clicking the plus button on the top right corner.

`tabsetPanel()`: It is used to create two tabs within a page.

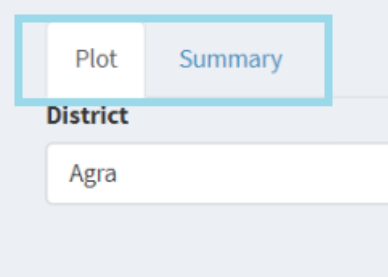


Figure 4.15: `sliderInput()`

The 4.15 figure shows an example of `tabsetPanel()`. This particular `tabsetPanel()` shown in the figure is used to create two tabs, namely Plot and Summary, within the graphs page. The plot tab is used to show all the plots related to rainfall data analysis/visualization. The summary tab is used to show a table with complete rainfall data with sorting options to display particular data.

4.7.4 How to project R analysis on the web app?

The projection of any R analysis on the web app consists of 2 simple steps.

- What to project?

1. Creation of output object: An output object is the first to be created in the server component of the Shiny app. The output object is created using render functions available in the Shiny package. There are different render functions to create outputs for different visualizations.

```
output$plotmov<-renderPlot({

  datatoplot<-read.xlsx(paste("data/NewFolder/",input$districtSelect,".xlsx",sep = ""))
  Annual<-datatoplot


  Annual<-(as.vector(t(as.matrix(Annual[,14]))))
  Annual<-ts(Annual,frequency=1,start=datatoplot$Year[input$slider[1]],
             end=datatoplot$Year[input$slider[2]])

  plot(movavg(Annual,n=30),type="l",ylab="Rainfall in mm",main="Moving Average for 30 years",
       xlab="Period (Number of years past 1901)")

})
```

Figure 4.16: `Example of an Output Object`

The 4.16 shows a code example to create an output object. The example creates an output object for the plot of 'Moving Average for 30 years'. `renderPlot()` is the render function used for the creation of this output object. In the example, the output used for the projection created by the `renderPlot()` is given the name 'plotmov'.

- Where to project?

2. The second step consists of specifying a place in the UI component to display our projection using the created output object.

```
gradientBox(
  title = "Moving Average",
  width = 12,
  icon = "fa fa-thermometer-half",
  gradientColor = "red",
  boxToolSize = "xs",
  closable = FALSE,
  footer = plotOutput("plotmov")
          )
```

Figure 4.17: `Example to show usage of Output Object in UI Component`

The 4.17 shows an example to display plot for 'Moving Average for 30 years'. The plot is displayed using the output object 'plotmov'. Here `plotOutput()` function is used under the 'footer' section of the '`gradientBox()`' to display the 'plotmov' inside the gradient box.

4.7.5 How to use the user's input to make dynamic visualizations?

After the creation of app layout and visualizations, the next step is to make the visualizations dynamic. Any input value given/set by the user using UI components is accessed by `input$componentname`. It can be used anywhere in the server component inside the render functions. It is passed as a parameter inside render functions to make dynamic visualizations. All the render functions are 'reactive' in nature. Whenever the user gives a new input value, the render function processes it again, creating a new output object — later displaying this new output object in the UI component.

```
shaperainfall=merge(shapefile,rainfall %>% filter(Year==input$yearselect),
                    by='DISTRICT',duplicateGeoms = TRUE)}
```

Figure 4.18: `Example of dynamic visualizations from user input`

The 4.18 shows an example of using the user's input to make dynamic visualizations. In the **`merge()`** function, input$yearselect is used to use the year selected by the user to merge the shapefile with rainfall data of the selected year for data analysis.

4.7.6 Final Shiny app

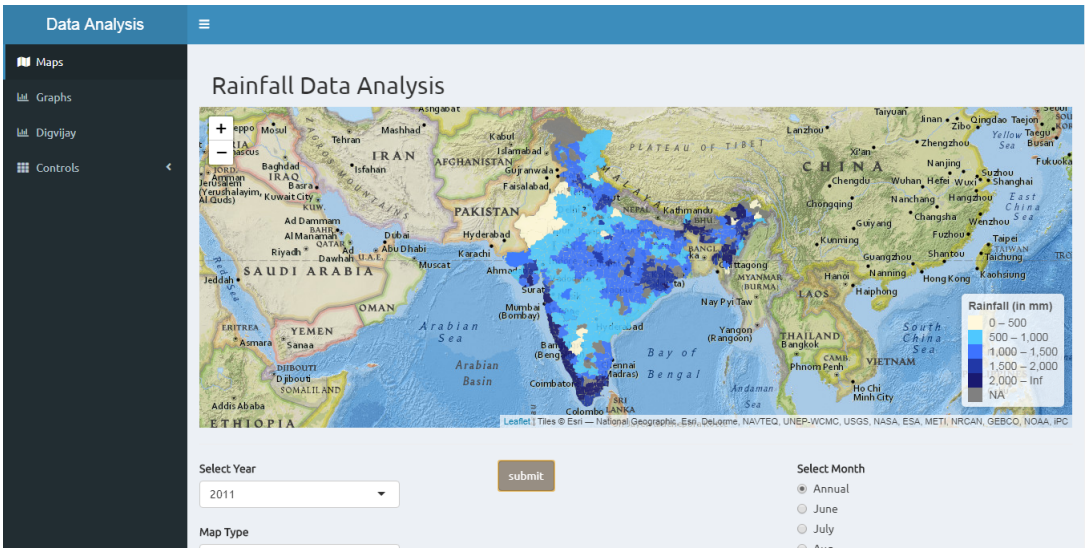
Two Shiny apps were created to represent different aspects of rainfall data analysis. Later merged to form one single app.

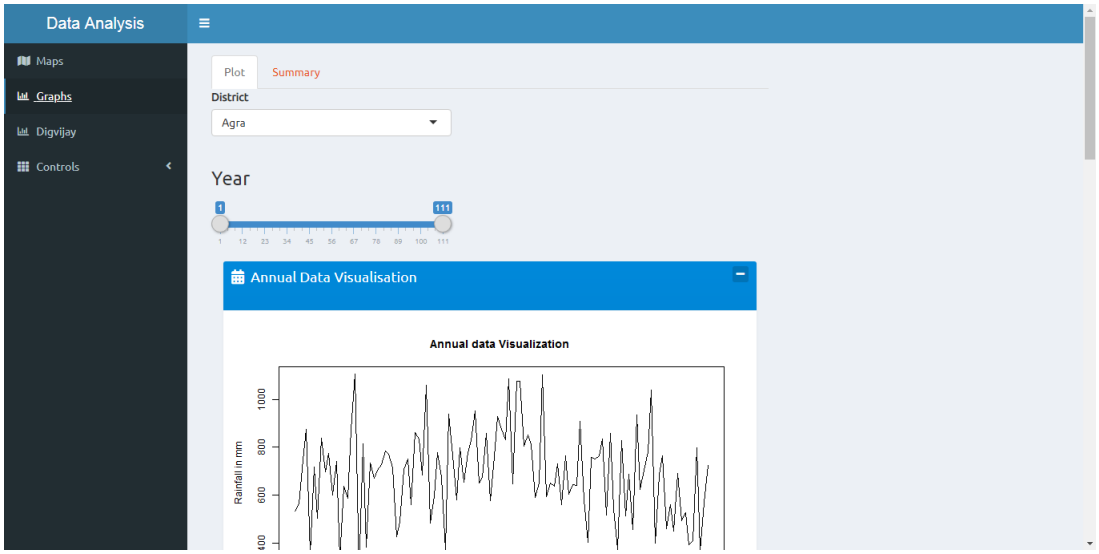
The first shiny app contains 2 tabs, maps, and graphs. The maps tab provides various options to the user to choose from like the year, map type, and time interval. According to the user's selection, a map of India is displayed showing the amount of rainfall in the form of color density for the time interval selected by the user.

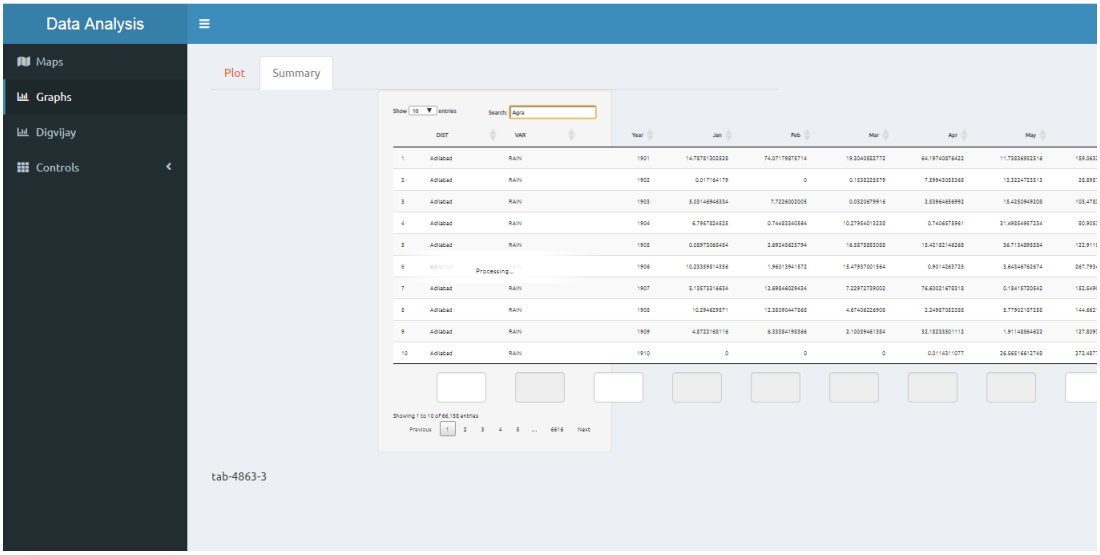
The graph tab contains two subtabs, plot, and summary, an option to choose the district and year range. According to the user's selection, various plots get displayed in plot subtab; namely, Annual Data Visualisation, Moving Average, Deviation from Moving Average, and Rainfall Prediction. In the summary tab, a table is displayed showing rainfall data that we have used along with some sorting options.

The other shiny app contains all of the control options in the sidebar. Depending on the values chosen, above and below Normal rainfall and years variations from annual rainfall are plotted.

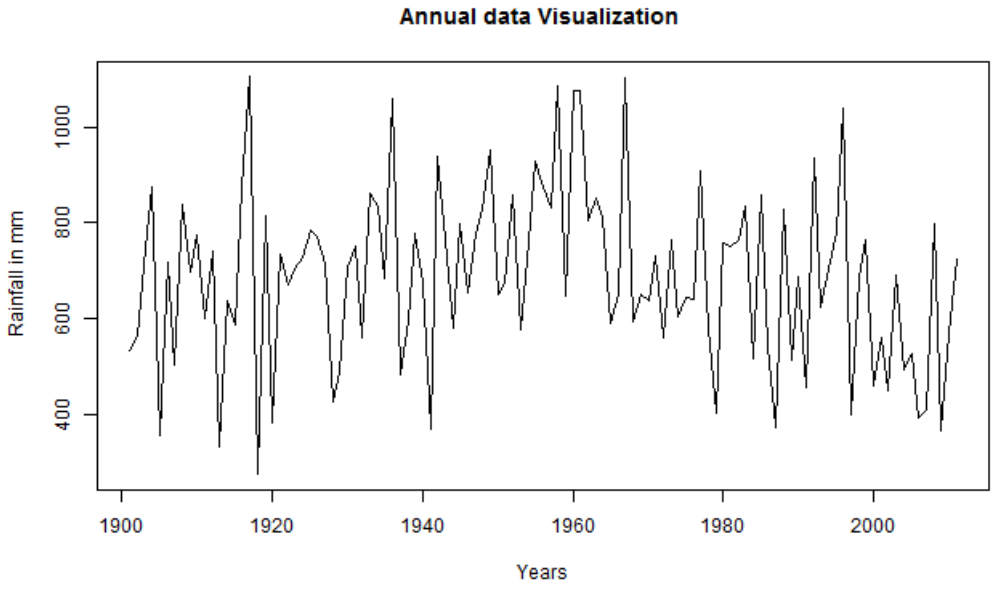
Below are the screenshots of the final Shiny app.



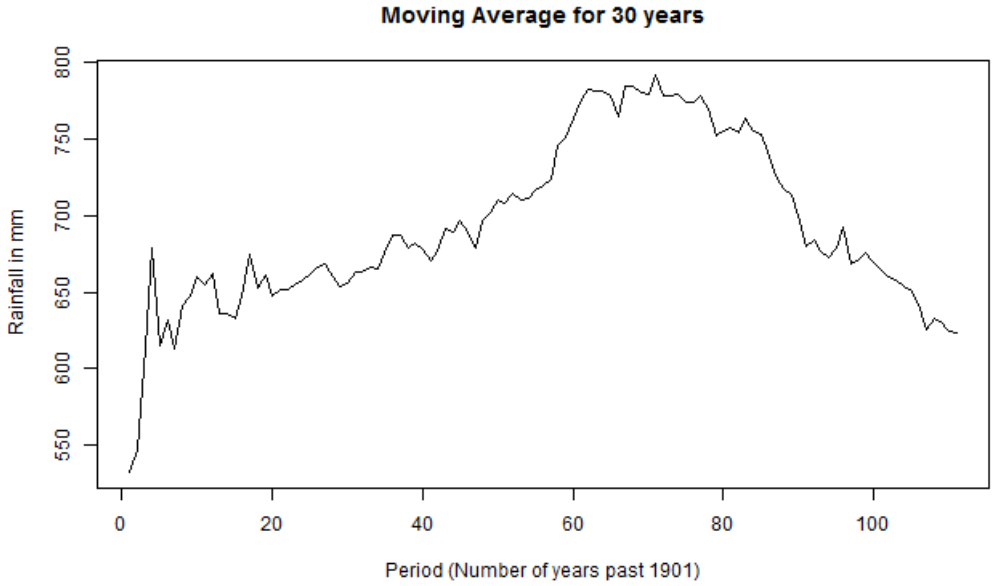

Data Analysis
Maps
Graphs
Digvijay
Controls
Plot
Summary
District
Agra
Year
1
111
1 12 23 34 45 56 67 78 89 100 111
Annual Data Visualisation
Annual data Visualization
Rainfall in mm
1000
800
600
400


Data Analysis
Maps
Graphs
Digvijay
Controls
Plot
Summary
Processing...
tab-4863-3

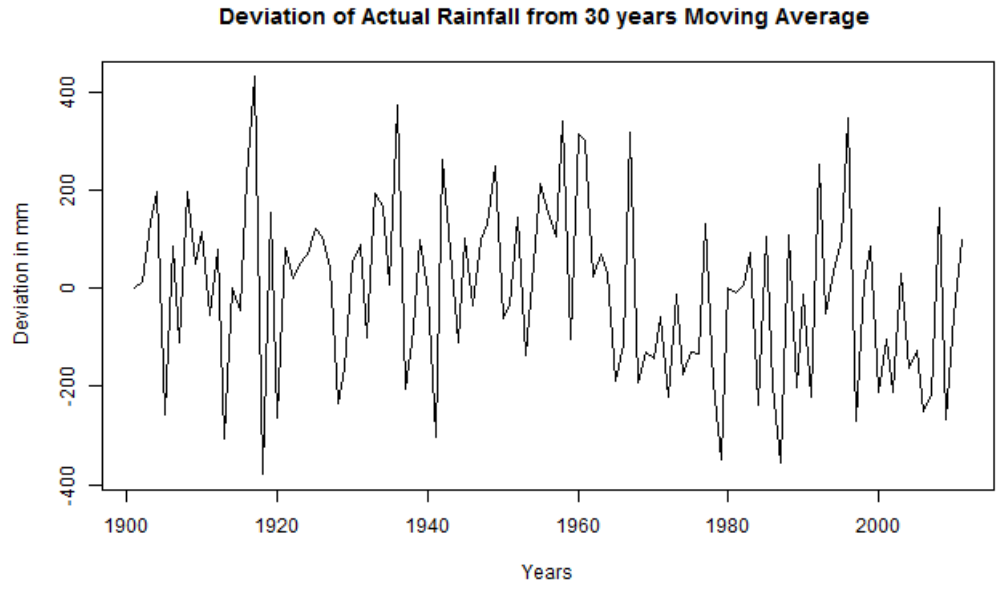
Annual Data Visualisation



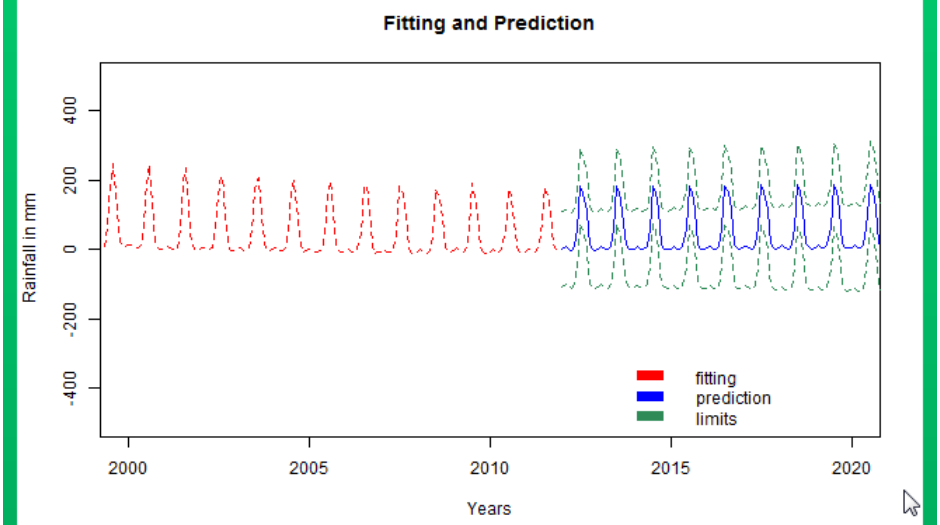
Moving Average



Deviation from Moving Average



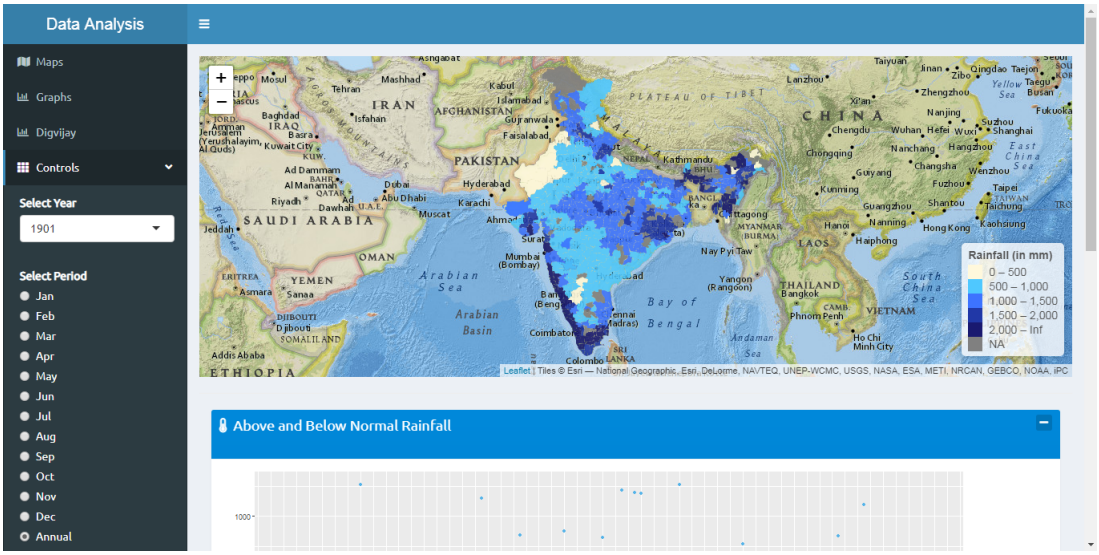
Rainfall Prediction

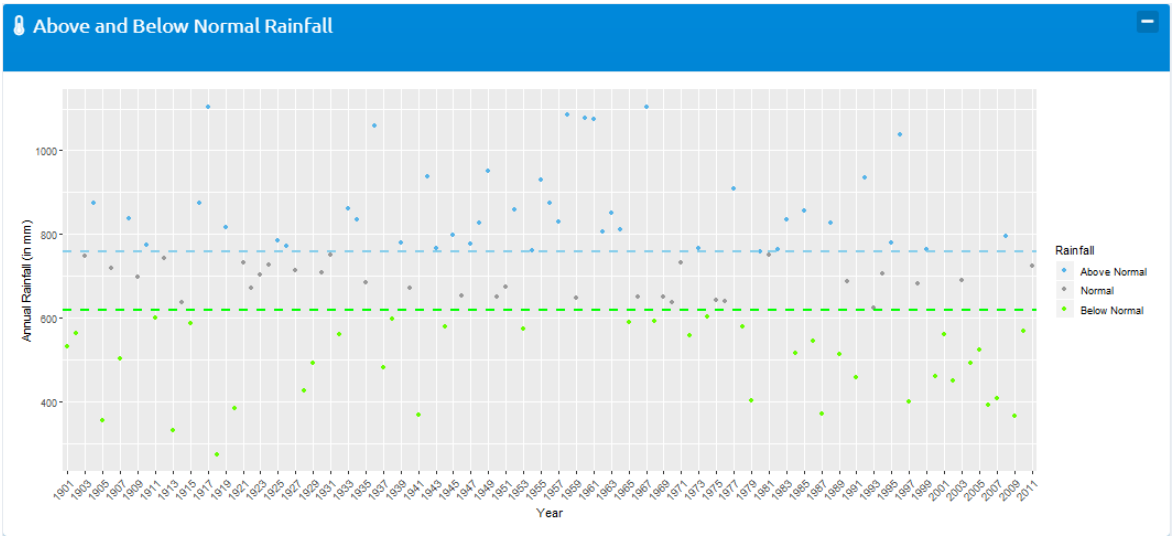


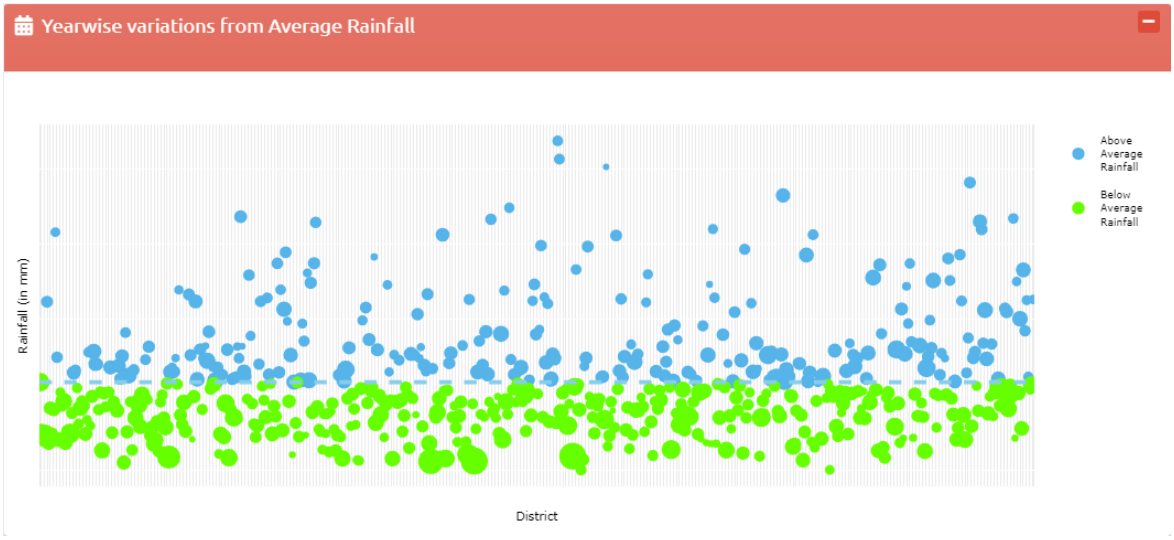
Prediction Range

1,900 2,000 2,020

1,900 1,913 1,926 1,939 1,952 1,965 1,978 1,991 2,004 2,017 2,030


Data Analysis
Maps
Graphs
Digvijay
Controls
Select Year
1901
Select Period
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Annual
Rainfall (in mm)
0 – 500
500 – 1,000
1,000 – 1,500
1,500 – 2,000
2,000 – Inf
NA
Above and Below Normal Rainfall
1000
Leaflet | Tiles © Esri — National Geographic, Esri, DeLorme, NAVTEQ, UNEP-WCMC, USGS, NASA, ESA, METI, NRCAN, GEBCO, NOAA, iPC


Above and Below Normal Rainfall
Annual Rainfall (in mm)
1000
800
600
400
Year
Rainfall
Above Normal
Normal
Below Normal


Yearwise variations from Average Rainfall
Rainfall (in mm)
District
Above Average Rainfall
Below Average Rainfall

Chapter 5

Ranchi District Data Analysis

5.1 14th Finance Commission

The Fourteen Finance Commission (FFC) was constituted by President on 2 January 2013 to give recommendations on a specific aspect of center-state fiscal relations during 2015-2020. FFC was, inter-alia mandated recommended measures needed to augment consolidated funds of the states to supplement the resources of the Panchayats based on the recommendation of the State Finance Commission (SFC).

The funds get transferred to local bodies for providing essential services including water supply, sanitation including septic management, sewage, solid waste management, maintenance of community assets, maintenance of roads, footpaths, streetlighting, and any other necessary services within the functions assigned to them under relevant legislations.

5.2 Ranchi District - Blocks and their respective Panchayats

Ranchi and Bundu subdivisions, which consist of blocks, panchayats, and villages, combine to form the administrative territory of Ranchi. The whole district consists of 18 blocks and 305 panchayats. Under the Ranchi Subdivision, there are 14 blocks, and the Bundu Subdivision consists of 4 blocks. With a geographical area of 5,097 square km, Ranchi district is one of the largest districts in Jharkhand.

Ranchi District has 1311 revenue villages under 305 panchayats. It consists of 18 blocks as per the 2011 census report.

The FFC has released a basic grant of Rs. 2,216,298,916 (221.63 crores approximately) for the period 2015-19 among 18 blocks.

There were 3 main tasks done in Ranchi district analysis. They are as follows-

- Visualization of the Ranchi district data for the amount received and spent under the 14th finance commission.

- Analysis of the data.
- Creation of an interactive dashboard for better visualization of plots.

5.3 Packages Used

Following packages were used :

- `knitr package` : knitr package contains kable function, which gives the advantage of presenting tables elegantly.
- `ggplot2 package` : Various functions of ggplot2 library used are ggplot, ggbar (for making bars), ggtitle (for giving title), labs (for giving xlabel and ylabel explicitly), scale_fill_brewer (to give pallete color to the graph/plot), and much more. ggplot2 is a widely used R package in the field of Data visualization. This package also gives the advantage of an elegant representation of data.
- `plotrix package` : This package can be used to plot 3D graphs/plots.
- `flexdashboard package` : This package is used to make an interactive dashboard. It uses RMarkdown to publish data visualization graphs/plots.
- `shiny package` : This package is used here with a combination of flexdashboard to give dynamic visualizations to the dashboard. This package gives the advantage to use shiny package functions such as selectInput function, renderPlot function, and much more in the dashboard directly.

5.4 Making subsets of data

The available data was an excel spreadsheet having data of each panchayat for every block of Ranchi district. The data contained the amount received and spent for every year from 2015-19. A particular order for the visualization was followed that comprises of following parts :

- First, displaying the table showing the number of panchayats in every block of Ranchi district and the total amount received and spent by each block. This table could be used to verify the results from the bar plots plotted. It also provides a summarized insight into the data. It was displayed using the kable function of the knitr library.
  - Performing a joining operation on 2 excel sheets using the VLOOKUP function. The first sheet contained the number of panchayats for every block.
  - The second sheet contained the total amount received and spent by every block, which was obtained by subsetting the main excel file on a blockwise basis. This subset file contains the amount received and spent for blocks only in the 4 years (2015-19).

```
# Loading library "knitr" for using 'kable' function.
library(knitr)
# Reading '.csv' file.
block<-read.csv("blocks.csv",header = TRUE)
# Creating data frame.
block<-data.frame(block)
# Changing values in columns.
block$Received_in_Crores=round(block$Received_in_Crores/10000000,digits=2)
block$Spent_in_Crores=round(block$Spent_in_Crores/10000000,digits=2)
# Creating table.
kable(block[,c(1:2,4:6)],caption="Blocks in Ranchi District" )
```

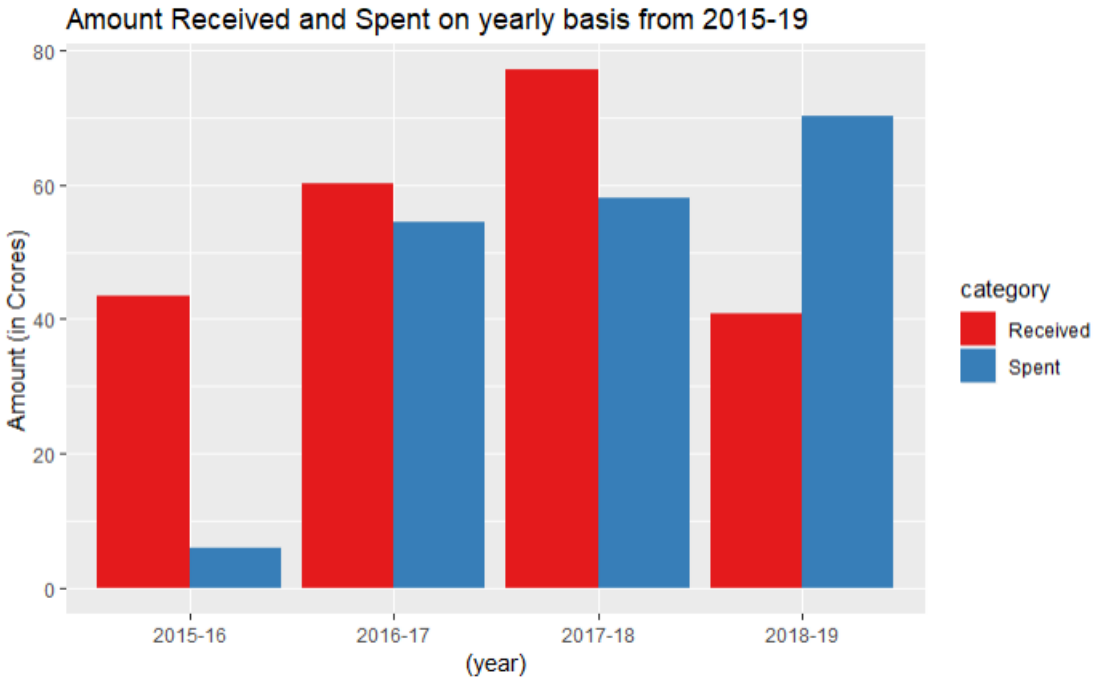
Table 1: Blocks in Ranchi District

S.No.	Block	Number.of.Panchayats	Received_in_Crores	Spent_in_Crores
1	Kanke	21	24.06	20.42
2	Namkum	17	17.97	12.44
3	Tamar	11	16.85	14.05
4	Mandar	14	15.18	13.89
5	Silli	17	13.79	12.63
6	Bero	9	13.75	12.94
7	Angara	32	14.18	11.76
8	Chanho	14	13.02	10.86
9	Ormanjhi	11	11.59	9.60
10	Burmu	19	11.27	9.38
11	Bundu	13	7.81	6.64
12	Khelari	23	9.19	8.54
13	Sonahatu	18	9.77	8.89
14	Ratu	9	13.38	11.01
15	Nagri	20	9.00	7.44
16	Lapung	20	8.26	6.67
17	Rahe	14	6.71	5.90
18	Itki	23	5.86	5.73

- Then barplot was plotted for showing the amount received by Ranchi district for each year from 2015-19. This plot was made using the ggplot2 library.
  - To provide numerical values to y-axis values, which is by default mathematical exponential values, we use options the function with attribute scipen value as 999.
  - The datatype of **`Amount`** vector values is changed to an integer type to perform a mathematical operation on them. It was done using **`as.integer()`** function.

```
# Loading library "readxl" for reading xlsx file.
library(readxl)
# For omitting exponential values in graph.
options(scipen=999)
# Inputting data.
ranchi<-read_excel("ranchi_data.xlsx")
# Categorizing data.
category=c('Received','Spent','Received','Spent','Received','Spent','Received','
    Spent')
```

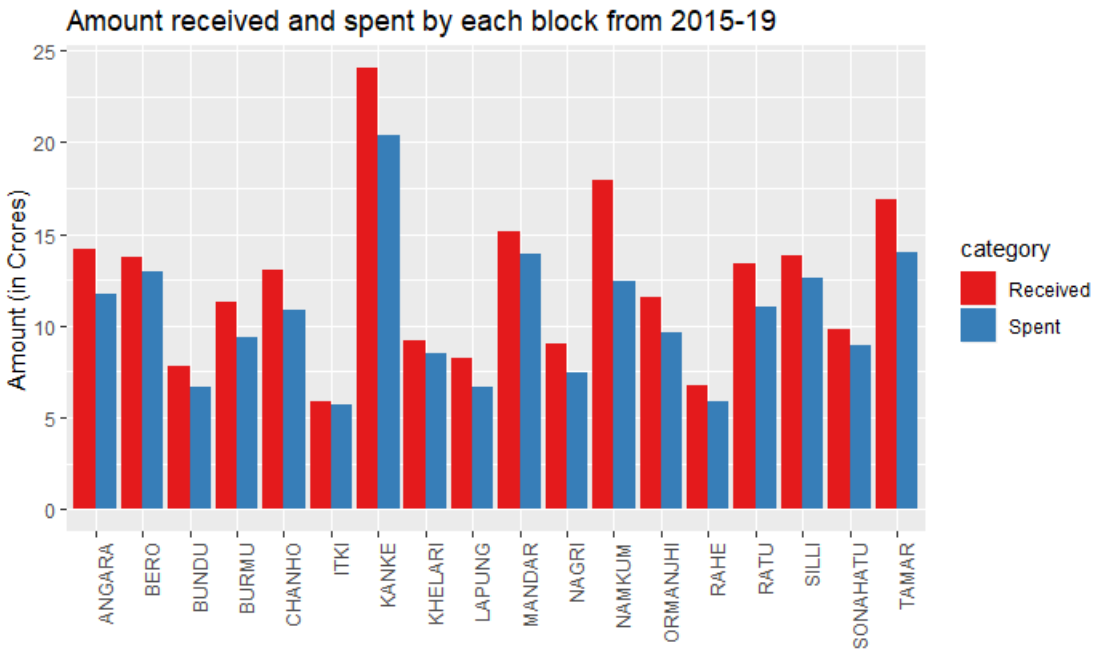
```
year=c("2015-16","2015-16","2016-17","2016-17","2017-18","2017-18","2018-19","
    2018-19")
Amount=c(as.integer(ranchi[327,4])/10000000,as.integer(ranchi[327,11])/10000000,as.
    integer(ranchi[327,5])/10000000,as.integer(ranchi[327,12])/10000000,as.integer(
    ranchi[327,6])/10000000+as.integer(ranchi[327,7])/10000000,as.integer(ranchi
    [327,13])/10000000,as.integer(ranchi[327,9])/10000000,as.integer(ranchi
    [327,14])/10000000)
Finance=data.frame(category,year,Amount)
# Loading library "ggplot" for creating graphs.
library(ggplot2)
ggplot(Finance, aes((year), Amount, fill = category)) + geom_bar(stat="identity",
    position = "dodge") + scale_fill_brewer(palette = "Set1") + ggtitle("Amount
    Received and Spent on yearly basis from 2015-19") + labs(y="Amount (in Crores)"
    )
```



- Further, creating the plot showing the amount received and spent by each block. For this, initially, data was cleaned by removing the NA values by using **`which()`** and **`complete.cases()`** function.
  - Also, the data read from the .xlsx file was by default of list datatype. Hence, unlist and as.numeric functions were used to perform mathematical operations on the data values.

```
# Loading library "readxl" for reading xlsx files.
library(readxl)
# Reading xlsx file containing data related to Chanho.
block=read_excel("CHANHO.xlsx")
# Extracting required data.
pnchyt=c((block[3]))
# Converting data into numeric format.
spent=as.numeric(unlist(c(block[15])))/100000
receive=as.numeric(unlist(c(block[10])))/100000
# Converting into data frame.
```

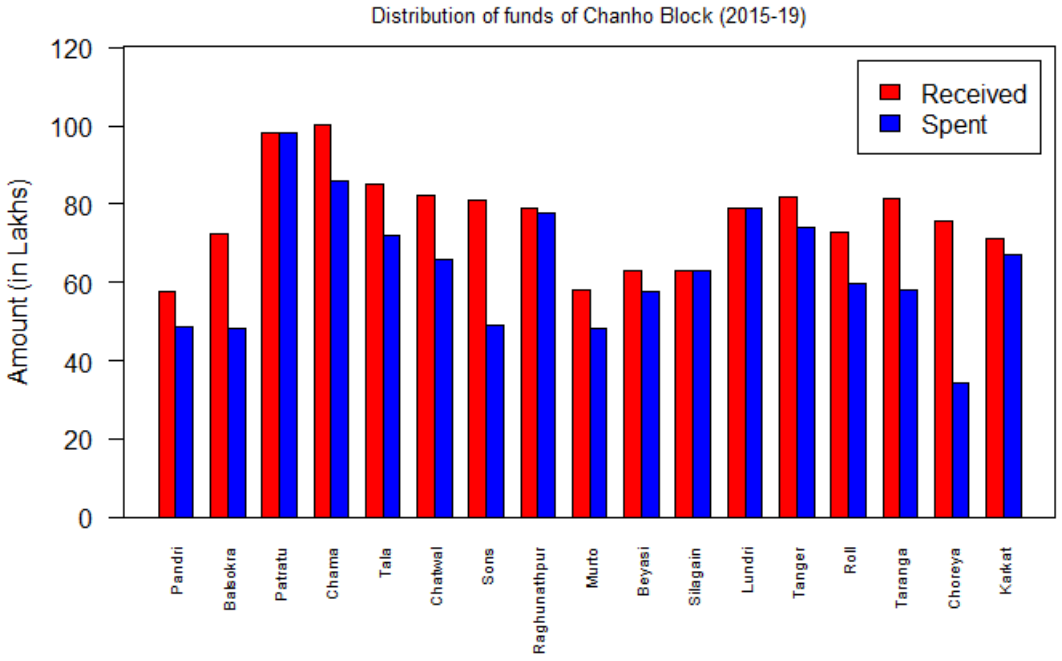
```r
b_p=data.frame(pnchyt,round(receive,digits = 2),round(spent,digits = 2))
names(b_p)=c('pnchayat','receive','spent')
cols=c("red","blue")
# Creating barplot.
barplot(t(b_p[c('receive','spent')]),beside=T,ylim = c(0,20+max(c(receive,spent))),
    col=cols,main ='Distribution of funds of Chanho Block (2015-19)',cex.main=0.8,
    names.arg=b_p$pnchayat,las=2, cex.names = 0.6)
box()
```



- Also, displaying blockwise charts representing the amount received and spent by the various panchayats of every block.
  - For this, every block's data was extracted from the main file and stored in a new spreadsheet for further visualization and analysis of every block.
  - The Chanho block's funds' distribution is displayed below as an example. The barplot function is used here, which is by default, present in the base library.
  - Plotting sidebars by passing the transpose of the matrix whose columns are to plotted. `t()` function was used for taking transpose. Various other parameters were also used, such as 'las=2' and 'cex.names=0.6' for clean and elegant bar plots.

```r
# Loading library "readxl" for reading xlsx files.
library(readxl)
# Reading xlsx file containing data related to Chanho.
block=read_excel("CHANHO.xlsx")
# Extracting required data.
pnchyt=c((block[3]))
```

```
# Converting data into numeric format.
spent=as.numeric(unlist(c(block[15])))/100000
receive=as.numeric(unlist(c(block[10])))/100000
# Converting into data frame.
b_p=data.frame(pnchyt,round(receive,digits = 2),round(spent,digits = 2))
names(b_p)=c('pnchayat','receive','spent')
cols=c("red","blue")
# Creating barplot.
barplot(t(b_p[c('receive','spent')]),beside=T,ylim = c(0,20+max(c(receive,spent))),
    col=cols,main ='Distribution of funds of Chanho Block (2015-19)',cex.main=0.8,
    names.arg=b_p$pnchayat,las=2, cex.names = 0.6)
box()
```



5.5 Dashboard

For dynamic visualizations, an interactive dashboard was made using flexdashboard and shiny. The figure below shows the GUI of the dashboard.

- The dashboard contains 2 rows.
- The first row contains plots of Ranchi district.
- The second row has two tabs.
- Tab 1 shows year-wise expenditure by blocks. In this tab, the user can select for which year and which block the expenditure he or she wants to see.
- Tab 2 shows the overall expenditure of the block in the four years(2015-19). Using this user can select a particular block whose expenditures he or she wants to see.


rstudio
Open in Browser
Republish
Ranchi District : Finance Commision Dashboard
Expenditure by Ranchi District for 14th Finance Commission
Amount Received and Spent on yearly basis from 2015-19 by Ranchi district
Amount received and spent by each block of Ranchi district from 2015-19
Yearwise Expenditure by Blocks
Panchayatwise Expenditure by Blocks
Select Block
BURMU
Select year
2017-18
Analysis of year 2017-18
Amount (in Lakhs)
Received
Spent

Chapter 6

Conclusion

All of the projects completed during fellowship contributed towards the increment in usability and awareness of open-source software, i.e., R. Starting from Textbook Companion, the codes written by participants were made public via an R cloud platform. They act as examples of implementation of functions in R. District data collection and analysis project was initiated to create a social impact. With the motive of assisting district collectors in making informed decisions for the welfare of the citizens of their respective districts. Similarly, rainfall data and its analysis will be made public through a web application built using R Shiny to increase its accessibility among the general public. Analysis of the Ranchi district's data for the amount received and spent under the 14^{th} finance commission was a project done in response to the request by Ranchi's district collector for assistance. The project resulted in successfully providing aid as per the request.

Overall, the FOSSEE fellowship was a great learning experience. Every fellow gained new skills and knowledge. Fellows from different FLOSS get a chance to meet each other and new people at IIT, which helped them in learning by sharing their knowledge with others. The tasks given during the fellowship gave an insight into professional practice. Fellows also learned the different facets of working within an organization. Also, the objective of contributing back to society via open source was a big motivator. In a nutshell, the fellowship taught each fellow work ethics, commitment, and the importance of contributing back to society besides technical skills.