A Comparative study on Indian Rainfall data using exploratory analysis and evaluating different time series models

Niteshkumar Jha Akshay Jadhav

Contents

List of Tables

List of Figures

1 INTRODUCTION

The variability of the annual rainfall in India and specifically Indian summer monsoon (ISM) affects the agriculture food grain production, industry and generation of hydroelectric power, causing severe strain on the national economy.

Large parts of the country are severely affected due to deficit monsoon rainfall and government of India spends large amount of money on providing relief in affected areas. Importance of ISM to Indian economy and as major global circulation parameter has motivated many scientists to study the variability of Indian monsoon, in the past. In this report,we are understanding histrorical trend for total rainfall and mansoon rainfall based on the data from 1871-2016. Futher, we are also exploring and comparing various forecasting model using machine learing techniques.

The statistical characteristics of the series i.e. Mean, Standard Deviation and variance has been calculated for below different series:

- Monthly Rainfall(All 12 months)
- Rainfall by season(4 seasons viz., Winter (January +February: J+F), Pre-monsoon (March+April+May: MAM), Southwest Monsoon (June to September: JJAS) and Post Monsoon (October +November+December: OND))
- Annual Rainfall

In addition to descriptive statistics, we have also plotted long term trend for above series to understand historical behaviour.

Definition:

The word monsoon (derived from the Arabic “mausam”, meaning “season”), although generally defined as a system of winds characterized by a seasonal reversal of direction, lacks a consistent, detailed definition. Some examples are:

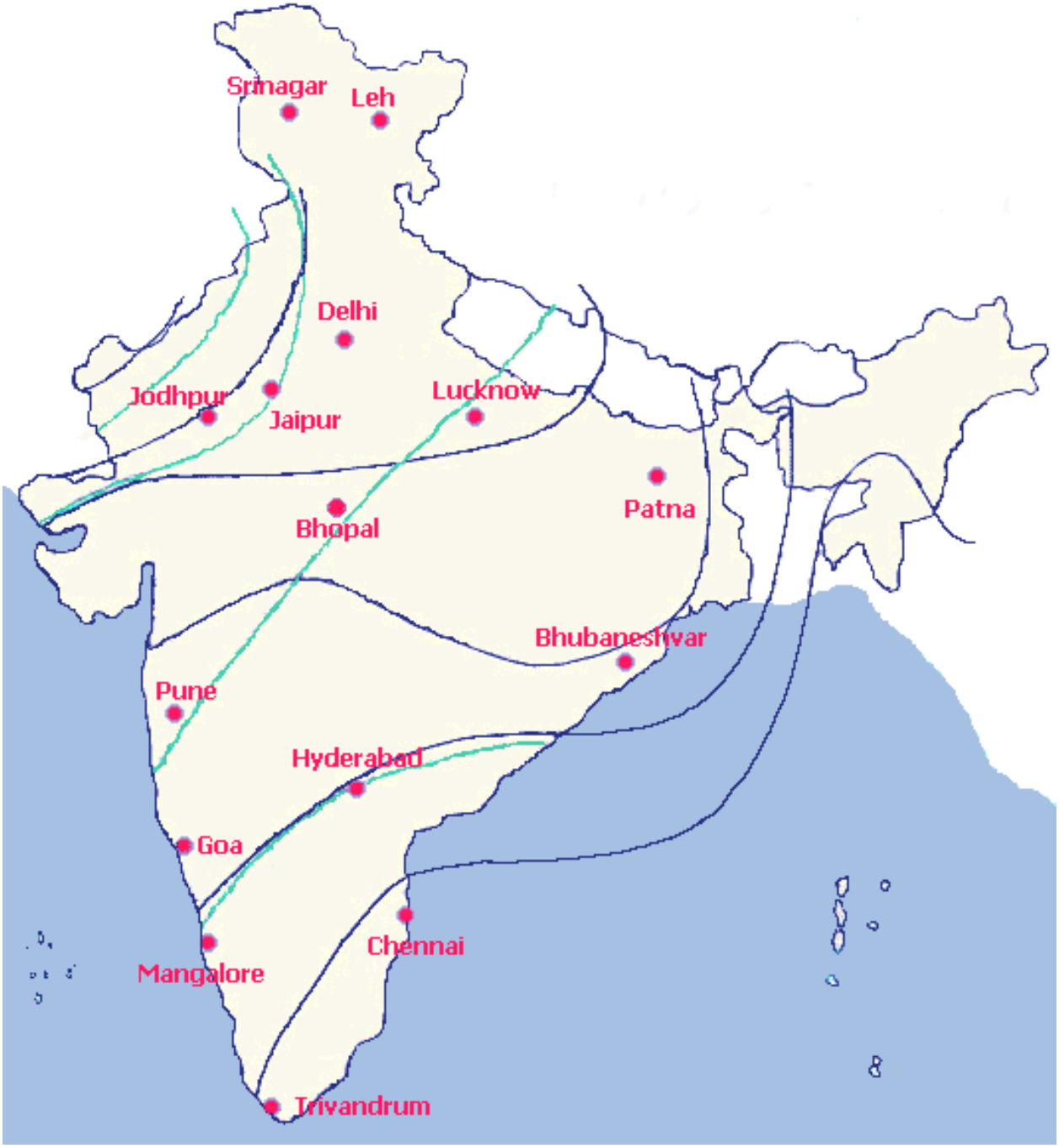


Figure 1.1: Indian Monsoon

- The American Meteorological Society calls it a name for seasonal winds, first applied to the winds blowing over the Arabian Sea from the northeast for six months and from the southwest for six months. The term has since been extended to similar winds in other parts of the world.
- The Intergovernmental Panel on Climate Change (IPCC) describes a monsoon as a tropical and subtropical seasonal reversal in both surface winds and associated precipitation, caused by differential heating between a continental-scale land mass and the adjacent ocean.
- The Indian Meteorological Department defines it as the seasonal reversal of the direction of winds along the shores of the Indian Ocean, especially in the Arabian Sea, which blow from the southwest for half of the year and from the northeast for the other half.
- Colin Stokes Ramage, in Monsoon Meteorology, defines the monsoon as a seasonal reversing wind accompanied by corresponding changes in precipitation.

Background:

Observed initially by sailors in the Arabian Sea traveling between Africa, India, and Southeast Asia, the monsoon can be categorized into two branches based on their spread over the subcontinent:

- Arabian Sea branch
- Bay of Bengal branch

Alternatively, it can be categorized into two segments based on the direction of rain-bearing winds:

- Southwest (SW) monsoon
- Northeast (NE) monsoon

Based on the time of year that these winds bring rain to India, the monsoon can also be categorized into two periods:

- Summer monsoon (May to September)
- Winter monsoon (October to November)

The complexity of the monsoon of South Asia is not completely understood, making it difficult to accurately predict the quantity, timing, and geographic distribution of the accompanying precipitation. These are the most monitored components of the monsoon, and they determine the water availability

in India for any given year.

Process of monsoon creation:

Also known as the thermal theory or the differential heating of sea and land theory, the traditional theory portrays the monsoon as a large-scale sea breeze. It states that during the hot subtropical summers, the massive landmass of the Indian Peninsula heats up at a different rate than the surrounding seas, resulting in a pressure gradient from south to north. This causes the flow of moisture-laden winds from sea to land. On reaching land, these winds rise because of the geographical relief, cooling adiabatically and leading to orographic rains. This is the southwest monsoon. The reverse happens during the winter, when the land is colder than the sea, establishing a pressure gradient from land to sea. This causes the winds to blow over the Indian subcontinent toward the Indian Ocean in a northeasterly direction, causing the northeast monsoon. Because the southwest monsoon flows from sea to land, it carries more moisture, and therefore causes more rain, than the northeast monsoon. Only part of the northeast monsoon passing over the Bay of Bengal picks up moisture, causing rain in Andhra Pradesh and Tamil Nadu during the winter months.

However, many meteorologists argue that the monsoon is not a local phenomenon as explained by the traditional theory, but a general weather phenomenon along the entire tropical zone of Earth. This criticism does not deny the role of differential heating of sea and land in generating monsoon winds, but casts it as one of several factors rather than the only one.

Significance:

The monsoon is the primary delivery mechanism for fresh water in the Indian subcontinent. As such, it affects the environment (and associated flora, fauna, and ecosystems), agriculture, society, hydro-power production, and geography of the subcontinent (like the availability of fresh water in water bodies and the underground water table), with all of these factors cumulatively contributing to the health of the economy of affected countries.

Geographical (wettest spots on Earth):

Mawsynram and Cherrapunji, both in the Indian state of Meghalaya, alternate as the wettest places on

Earth given the quantity of their rainfall, though there are other cities with similar claims. They receive more than 11,000 millimeters of rain each from the monsoon.

Agricultural:

In India, which has historically had a primarily agrarian economy, the services sector recently overtook the farm sector in terms of GDP contribution. However, the agriculture sector still contributes 17-20% of GDP[48] and is the largest employer in the country, with about 60% of Indians dependent on it for employment and livelihood. About 49% of India's land is agricultural; that number rises to 55% if associated wetlands, dryland farming areas, etc., are included. Because more than half of these farmlands are rain-fed, the monsoon is critical to food sufficiency and quality of life.

Despite progress in alternative forms of irrigation, agricultural dependence on the monsoon remains far from insignificant. Therefore, the agricultural calendar of India is governed by the monsoon. Any fluctuations in the time distribution, spatial distribution, or quantity of the monsoon rains may lead to floods or droughts, causing the agricultural sector to suffer. This has a cascading effect on the secondary economic sectors, the overall economy, food inflation, and therefore the general population's quality and cost of living.

Economic:

The economic significance of the monsoon is aptly described by Pranab Mukherjee's remark that the monsoon is the "real finance minister of India". A good monsoon results in better agricultural yields, which brings down prices of essential food commodities and reduces imports, thus reducing food inflation overall. Better rains also result in increased hydroelectric production. All of these factors have positive ripple effects throughout the economy of India.

The down side however is that when monsoon rains are weak, crop production is low leading to higher food prices with limited supply. As a result, the Indian government is actively working with farmers and the nation's meteorological department to produce more drought resistant crops.

Social:

D. Subbarao, former governor of the Reserve Bank of India, emphasized during a quarterly review

of India's monetary policy that the lives of Indians depend on the performance of the monsoon. His own career prospects, his emotional well-being, and the performance of his monetary policy are all "a hostage" to the monsoon, he said, as is the case for most Indians. Additionally, farmers rendered jobless by failed monsoon rains tend to migrate to cities. This crowds city slums and aggravates the infrastructure and sustainability of city life.

Travel:

In the past, Indians usually refrained from traveling during monsoons for practical as well as religious reasons. But with the advent of globalization, such travel is gaining popularity. Places like Kerala and the Western Ghats get a large number of tourists, both local and foreigners, during the monsoon season. Kerala is one of the top destinations for tourists interested in Ayurvedic treatments and massage therapy. One major drawback of traveling during the monsoon is that most wildlife sanctuaries are closed. Also, some mountainous areas, especially in Himalayan regions, get cut off when roads are damaged by landslides and floods during heavy rains.

Environmental:

The monsoon is the primary bearer of fresh water to the area. The peninsular/Deccan rivers of India are mostly rain-fed and non-perennial in nature, depending primarily on the monsoon for water supply. Most of the coastal rivers of Western India are also rain-fed and monsoon-dependent. As such, the flora, fauna, and entire ecosystems of these areas rely heavily on the monsoon.

2 OBJECTIVE

The objective of this study is to understand historical trend of rainfall in India. By understanding past behaviour, we are also trying to build a forecasting model for future rainfall.We are primarily analysing below two variables:

1. Annual Rainfall
2. Masoon Rainfall

Detail exploratory analysis will be performed which will throw insight on nature of rainfall historically and machine learning techniques will be explored for model building. A comparative study will also be perform to check best forecasting model.

3 METHODOLOGY

We will follow below methodology for this project:

1. Importing and understanding the data
2. Exploratory data analysis
3. Building various time series models
4. Comparison of models

5. Conclusion and summary

This entire analysis will be performed using below statistical software:

- **RStudio**(https://www.rstudio.com/)
- **RMarkdown**(https://rmarkdown.rstudio.com/) and
- **Tableau**(https://public.tableau.com/en-us/s/).

BASIC STEPS IN FORECASTING:

Step 1: Problem definition

Often this is the most difficult part of forecasting. Defining the problem carefully requires an understanding of the way the forecasts will be used, who requires the forecasts, and how the forecasting function fits within the organisation requiring the forecasts. A forecaster needs to spend time talking to everyone who will be involved in collecting data, maintaining databases, and using the forecasts for future planning.

Step 2: Gathering information

There are always at least two kinds of information required: (a) statistical data, and (b) the accumulated expertise of the people who collect the data and use the forecasts. Often, it will be difficult to obtain

enough historical data to be able to fit a good statistical model. In that case, the judgmental forecasting methods of Chapter 4 can be used. Occasionally, old data will be less useful due to structural changes in the system being forecast; then we may choose to use only the most recent data. However, remember that good statistical models will handle evolutionary changes in the system; don't throw away good data unnecessarily.

Step 3: Preliminary (exploratory) analysis

Always start by graphing the data. Are there consistent patterns? Is there a significant trend? Is seasonality important? Is there evidence of the presence of business cycles? Are there any outliers in the data that need to be explained by those with expert knowledge? How strong are the relationships among the variables available for analysis? Various tools have been developed to help with this analysis. These are discussed in Chapters 2 and 6.

Step 4: Choosing and fitting models The best model to use depends on the availability of historical data, the strength of relationships between the forecast variable and any explanatory variables, and the way in which the forecasts are to be used. It is common to compare two or three potential models. Each model is itself an artificial construct that is based on a set of assumptions (explicit and implicit) and usually involves one or more parameters which must be estimated using the known historical data. We will discuss regression models, exponential smoothing methods, Box-Jenkins ARIMA models, Dynamic regression models, Hierarchical forecasting, and several advanced methods including neural networks and vector autoregression.

Step 5: Using and evaluating a forecasting model

Once a model has been selected and its parameters estimated, the model is used to make forecasts. The performance of the model can only be properly evaluated after the data for the forecast period have become available. A number of methods have been developed to help in assessing the accuracy of forecasts. There are also organisational issues in using and acting on the forecasts. When using a forecasting model in practice, numerous practical issues arise such as how to handle missing values and outliers, or how to deal with short time series.

4 DATA PREPROCESSING

In this step, we are setting our **R** system by importing and loading libraries require in this project. Once all important libraries will be installed, we are importing and understanding our rainfall input data.

4.1 Checking and loading libraries

Here we are checking **R** libraries and if not found in system then we install it.Libraries which are getting used are as below:

1. **Data Manipulation**: `data.table`,`dplyr`,`sqldf` and `reshape`
2. **Descriptive Statistics**: `pastecs`,`psych`,`tidyquant`,`nortest` and `tseries`
3. **Data Visulization**: `ggthemes`,`ggplot2` and `scales`
4. **Forecasting**: `forecast` and `Metrics`
5. **Reporting**: `kableExtra` and `pander`

```
list.of.packages <-c("data.table","dplyr","plyr","sqldf",
                     "reshape","pastecs","psych","tidyquant",
                     "nortest","tseries","ggthemes","ggplot2",
                     "scales","forecast","Metrics","kableExtra",
                     "pander","tidyverse","lubridate","timeSeries",
                     "magrittr","recipes")

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package

if(length(new.packages)) install.packages(new.packages)
pander::pander("All libraries are updated and installed")
```

All libraries are updated and installed

4.2 Load libraries

Here we load libraries.

```
library(data.table)
library(dplyr)
library(plyr)
library(sqldf)
library(reshape)
library(pastecs)
library(psych)
library(tidyquant)
library(nortest)
library(tseries)
library(ggthemes)
library(ggplot2)
library(scales)
library(forecast)
library(Metrics)
library(kableExtra)
library(pander)
library(tidyverse)
library(lubridate)
library(timeSeries)
library(magrittr)
library(recipes)


pander("All libraries are successfully installed")
```

All libraries are successfully installed

4.3 Data Preprocessing

In this step, we are importing data and understanding it.We have obtained this data by secondary research from this source Indian Institute of Tropical Meteorology.

In this step, we are importing input data and exploring it by some descripitive statistics and graphs.

4.3.1 *Data Import*

Let's start by loading our data set.For importing the data, we are using `read.csv` function from the base **R** packege.

```
setwd("D:/A Notes by 1st sem MSc(Nitesh)/Project/project 2/Final")
rainfall=read.csv('rainfall.csv')


pander("Data loaded successfully")
```

Data loaded successfully

4.3.2 *Structure of data*

In this step, we are checking structure of our **rainfall** dataset.

```
# Stucture of data
str(rainfall)
```

```
## 'data.frame':    146 obs. of  18 variables:
##  $ YEAR: int  1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 ...
##  $ JAN : num  19.6 7.6 3.7 8.6 9.9 0.9 29.1 10.8 2.1 3.9 ...
##  $ FEB : num  10.7 7.5 13.5 15.8 11.4 2.1 23.4 9.1 10.7 16.5 ...
##  $ MAR : num  14.5 7.3 15 10.7 13.1 16 24 10.2 8 15.1 ...
##  $ APR : num  33.9 24 24.3 16.9 23.2 16.5 35.3 36.1 8.2 21.1 ...
```

```
##  $ MAY : num  63.6 43.8 42.8 68.3 50.6 42.5 67.4 66.5 87.7 49.6 ...
##  $ JUN : num  208 189 113 228 193 ...
##  $ JUL : num  278 291 264 307 308 ...
##  $ AUG : num  179 245 214 234 219 ...
##  $ SEP : num  184 188 166 206 210 ...
##  $ OCT : num  36.8 78.5 60.7 93.2 56.6 ...
##  $ NOV : num  32.4 27.6 11.5 18.7 6.3 9.5 18.9 27.3 20.5 52.8 ...
##  $ DEC : num  6.7 19.1 9 4 7.1 2.5 36.5 14 7.1 11.1 ...
##  $ JF  : num  30.3 15.1 17.2 24.4 21.3 3 52.5 19.9 12.7 20.4 ...
##  $ MAM : num  112 75.1 82.1 95.9 86.9 ...
##  $ JJAS: num  849 914 757 974 930 ...
##  $ OND : num  75.8 125.2 81.2 115.9 70 ...
##  $ ANN : num  1067 1129 938 1211 1108 ...
```

Findings:

1. Dataframe consists of 146 observation and 18 variables.

2. All variables are numeric except YEAR variable which is integer.

5 EXPLORATORY DATA ANALYSIS

In this step, we are going to perform below tools of Exploratory data analysis:

- Descriptive statistics
  - Measure of central tendency and dispersion
  - Skewness and kurtasis
  - Normality test
- Histogram
- Box-plot

5.1 Descriptive statistics

In this step, we are checking some descriptive statistics of our **rainfall** dataset. We are calculating below statistics:

Measure of central tendency and dispersion:

A measure of central tendency is a number used to represent the center or middle of a set of data values.In other words, The measures of central tendency describe a distribution in terms of its most "frequent", "typical" or "average" data value. But there are different ways of representing or expressing the idea of "typicality".

Arithmetic Mean

For a given set of observations, Arithmetic Mean is defined as the sum of all the observations divided by the number of observations. Thus, if a variable x assumes n values x_1, x_2, x_3,..... x_n, then **AM** of x, to be denoted by $\bar{x}$, given by,

$$\bar{x} = \frac{x_1 + x_2 + x_3 + ... + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

The Median

The median can be defined as that point in a distribution above which and below which lie 50% of all the cases or observations in the distribution.

Measure of dispersion

Measure of dispersion is defined as lack of uniformity in the sizes or quantities of the items of a group or series.

Range

Range is the difference between the smallest value and the largest value of a series.

It is calculated by below formula:

$$Range = Maximum - Minimum$$

Standard deviation and Variance

Standard deviation is calculated as the square root of average of squared deviations taken from actual mean. It is also called root mean square deviation. The square of standard deviation i.e., σ^2 is called 'variance'. Calculation of standard deviation in case of raw data

Formula for Standard Deviations is as below:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2}$$

Standard error

Standard error is define as standard deviation of sample statistics.Formula for Standard error is as below:

$$SE = \frac{s}{\sqrt{n}}$$

For calculating all above statistics, we are using **stat.desc** from `pastecs`.

```
options("scipen"=100, "digits"=2)
# using stat.desc
kable(stat.desc(rainfall[,2:18],norm = TRUE), "latex",longtable=T,booktabs=T,
      caption="Descriptive statistics") %>%
  kable_styling(latex_options = c("striped", "scale_down"),font_size = 5)
```

Table 5.1: Descriptive statistics

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JF	MAM	JJAS	OND	ANN
nbr.val	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00	146.00
nbr.null	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nbr.na	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
min	0.80	0.60	2.60	8.00	20.50	81.50	121.30	144.10	77.30	14.70	1.80	0.30	3.00	55.20	604.00	50.10	810.90
max	40.50	41.00	52.80	60.50	107.70	241.60	346.00	339.30	267.80	159.50	88.30	56.80	61.10	166.50	1020.20	209.90	1347.00
range	39.70	40.40	50.20	52.50	87.20	160.10	224.70	195.20	190.50	144.80	86.50	56.50	58.10	111.30	416.20	159.80	536.10
sum	1571.60	1817.00	2223.10	3860.10	7699.20	23806.40	39790.10	35366.30	24869.60	11319.50	4504.90	1717.90	3387.80	13781.60	123831.70	17541.20	158542.80
median	9.15	10.65	12.70	24.70	49.80	161.35	278.80	241.65	169.75	72.80	27.10	9.95	21.60	91.90	858.55	120.40	1087.95
mean	10.76	12.45	15.23	26.44	52.73	163.06	272.53	242.23	170.34	77.53	30.86	11.77	23.20	94.39	848.16	120.15	1085.91
SE.mean	0.63	0.73	0.77	0.76	1.32	3.01	3.10	3.12	3.04	2.38	1.51	0.78	0.96	1.70	6.91	2.86	8.39
CI.mean.0.95	1.24	1.45	1.53	1.49	2.61	5.95	6.13	6.17	6.01	4.70	2.99	1.53	1.90	3.36	13.65	5.65	16.58
var	57.64	78.29	87.64	83.36	255.49	1324.89	1404.80	1421.05	1348.12	824.93	334.61	87.99	135.15	422.18	6964.28	1194.45	10275.50
std.dev	7.59	8.85	9.36	9.13	15.98	36.40	37.48	37.70	36.72	28.72	18.29	9.38	11.63	20.55	83.45	34.56	101.37
coef.var	0.71	0.71	0.61	0.35	0.30	0.22	0.14	0.16	0.22	0.37	0.59	0.80	0.50	0.22	0.10	0.29	0.09
skewness	1.09	1.02	1.64	0.94	0.90	-0.06	-1.16	0.02	0.11	0.53	0.85	1.95	0.66	0.69	-0.51	0.35	-0.02
skew.2SE	2.71	2.54	4.09	2.34	2.24	-0.14	-2.88	0.06	0.26	1.32	2.12	4.85	1.64	1.72	-1.26	0.88	-0.05
kurtosis	1.42	0.39	3.17	1.40	0.74	-0.67	2.43	-0.49	-0.58	0.05	0.15	5.57	0.17	0.61	-0.12	-0.31	0.06
kurt.2SE	1.78	0.49	3.97	1.75	0.93	-0.84	3.05	-0.62	-0.73	0.07	0.19	6.99	0.21	0.76	-0.16	-0.39	0.08
normtest.W	0.92	0.90	0.86	0.95	0.95	0.99	0.93	0.99	0.99	0.98	0.93	0.84	0.97	0.97	0.98	0.98	0.99
normtest.p	0.00	0.00	0.00	0.00	0.00	0.28	0.00	0.78	0.40	0.01	0.00	0.00	0.00	0.00	0.02	0.07	0.70

Conclusion:

From the descriptive statistics results in the table 5.1 we can conclude the following:

1. There is no missing value or null value in the dataset as indicated in `nbr.na` and `nbr.null`

respectively.

2. On an average JULY followed by AUGUST received highest rainfall whereas JANUARY received lowest rainfall.
3. On an average, JJASual rainfall in the INDIA is approximate 1085.91mm in which INDIA received lowest rainfall of 810.90mm in the year 1899 whereas maximum rainfall of 1340mm recorded in the year 1917.
4. Skewness value of southwest monsoon(JJAS) is -0.51 which is lies between -0.5 to -1, then we conclude that the distribution is moderately skewed and also distribution is highly skewed because given value is less than -1.
5. Skewness value of annual rainfall (ANN) is -0.02, which is lies between -0.5 to 0.5, and then we conclude that the distribution is approximately skewed, also distribution is highly skewedbecause give value is less than -1.
6. Skew.2SE value of southwest monsoon (JJAS)is -1.26, and then we conclude that skewness is insignificantly different from zero. 7.Kurtosis value of southwest monsoon (JJAS) is -0.12, which is less than -1,and then we conclude that distributions is light tails and is called a platykurtic distribution. 8.Kurtosis value of annual rainfall (ANN) is 0.06, which is lies between 0 to 1,and then we conclude that distributions has heavier tails and is called a leptokurtic distribution. 9.Kurtosis value of annual rainfall (ANN) is 0.06, which is lies between 0 to 1,and then we conclude that distributions has heavier tails and is called a leptokurtic distribution. 10.The value of the shapiro-wilk test of annual rainfall (ANN) is 0.99 which is greater than 0.05, then we say that the data is normal.
7. Most of the variables(month/weather period) are not following normal distribution as shown in shapio-wilk test(`normtest.W` and `normtest.p`) except JUN, AUG, SEP,OND and JJASual variables.(Since p-value > 0.05).

5.2 Checking distribution

In this step, we are checking distribution of each variable by looking at histogram. We are also checking presence of outliers by looking at Box-plot.

Histogram:

A histogram is a bar diagram which is suitable for frequency distributions with continuous classes. The width of all bars is equal to class interval and heights of the bars are in proportion to the frequencies of the respective classes.

Box-plot:

In descriptive statistics, a box plot or box plot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. Outliers may be plotted as individual points. Box plots are non-parametric: they display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution (though Tukey's box plot assumes symmetry for the whiskers and normality for their length). The spacing's between the different parts of the box indicate the degree of dispersion (spread) and skewness in the data, and show outliers. In addition to the points themselves, they allow one to visually estimate various L-estimators, notably the interquartile range, midhinge, range, mid-range, and trim an. Box plots can be drawn either horizontally or vertically. Box plots received their name from the box in the middle.

We are checking distribution of all series by histogram and box-plot. Additionally we are also checking distribution of annual and monsoon seprately.

5.2.1 *All series*

In this step, we are plotting histogram and box-plot of all series.

Histogram

```
multi.hist(rainfall[,2:18])
```

Interpretation:

1. The above diagram shows the histogram of the all the months and we have to check data follos normal distribution or not.

Box-plot

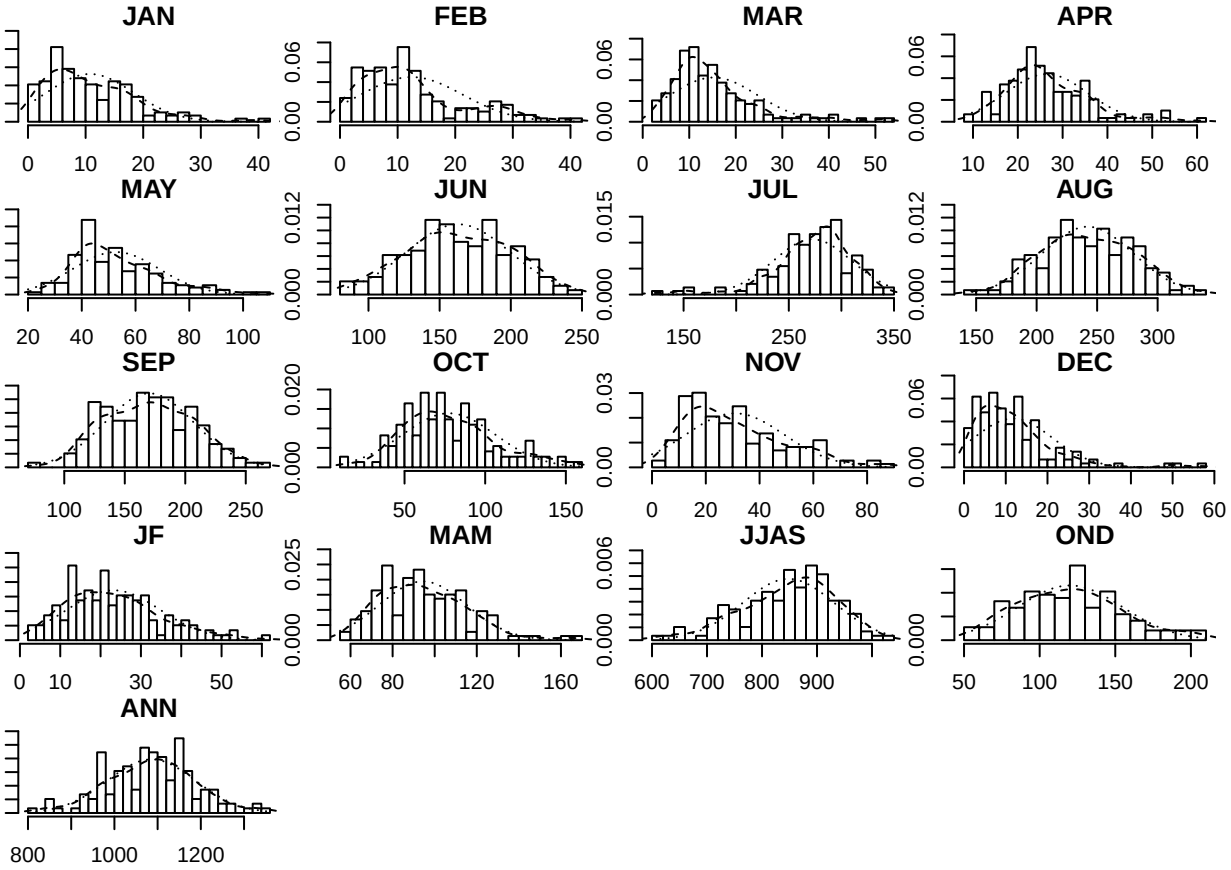


Figure 5.1: Histogram of all series

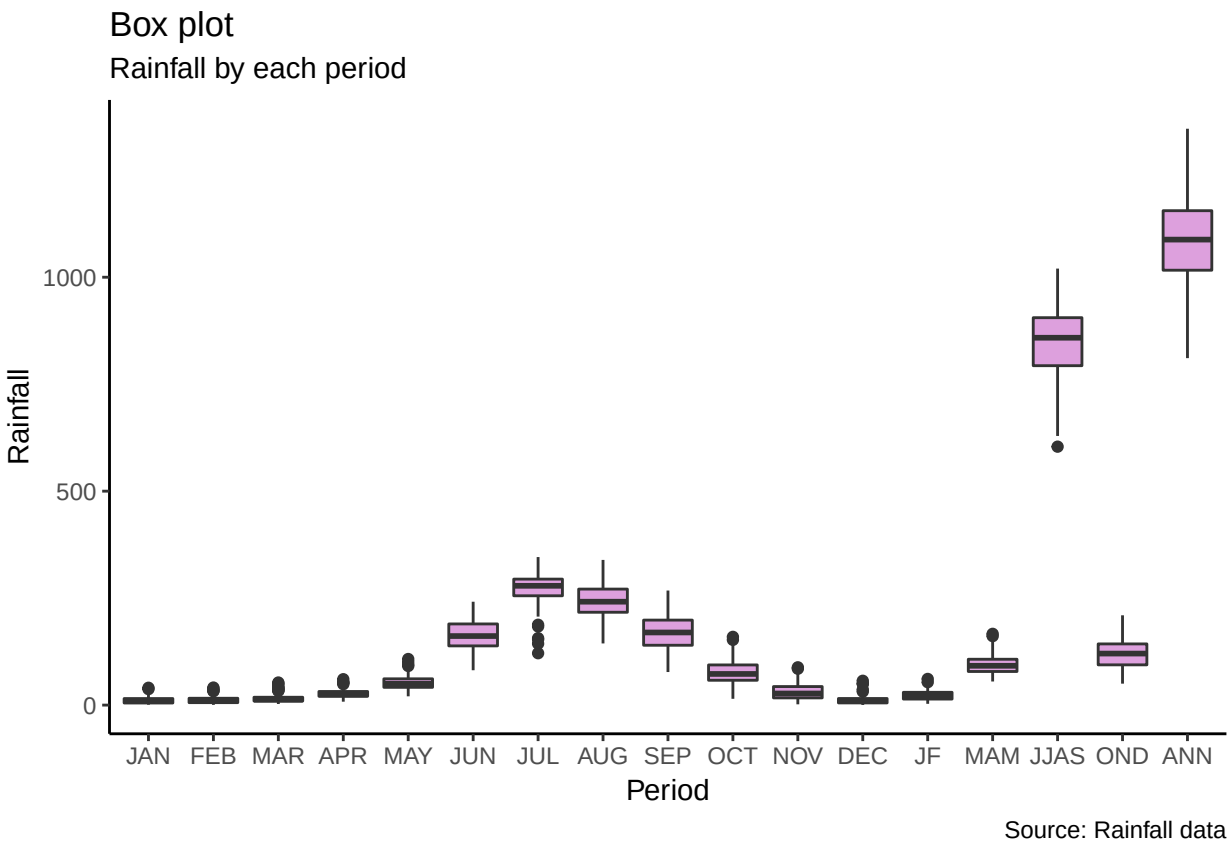


Figure 5.2: Box plot of all series

```
mdata <- melt(rainfall, id="YEAR")


theme_set(theme_classic())


# Plot
g <- ggplot(mdata, aes(variable, value))
g + geom_boxplot(varwidth=T, fill="plum") +
    labs(title="Box plot",
         subtitle="Rainfall by each period",
         caption="Source: Rainfall data",
         x="Period",
         y="Rainfall")
```

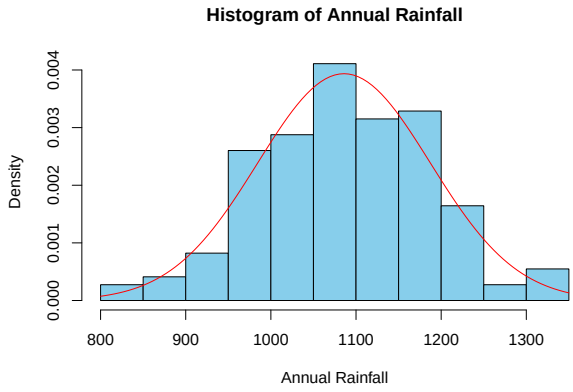


Figure 5.3: Histogram of Annual Rainfall

Interpretation:

1. Annual variable look like following normal distribution which is also confirmed in descriptive statistics section. No outlier has been observed as well. 2.Southewest monsoon(JJAS) which value lies between 1st quartile and 2nd quartile and there is outlier present in the data.

5.2.2 *Annual rainfall*

In this step, we are checking distribution of Annual variable by looking at histogram. We are also checking presence of outliers by looking at Box-plot.

Histogram:

```
# Normal distribution line in histogram
hist(rainfall$ANN, freq=FALSE, col="skyblue", xlab="Annual Rainfall", main= "Histogram o
curve(dnorm(x, mean=mean(rainfall$ANN), sd=sd(rainfall$ANN)), add=TRUE, col="red")
```

Interpretation:

1.The plot shows that the distribution of forecast errors is roughly centred on 1075, and is more or less normally distributed, although it seems to be slightly skewed to the right compared to a normal curve. However, the right skew is relatively small, and so it is plausible that the forecast errors are normally distributed with mean zero.

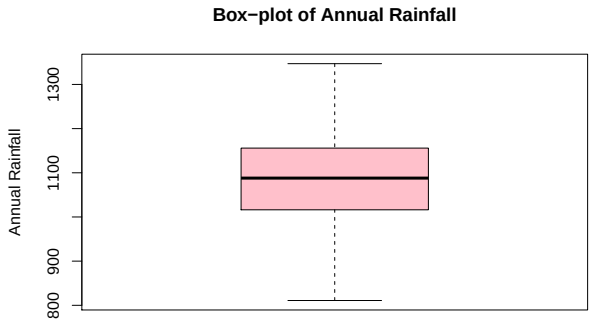


Figure 5.4: Boxplot of Annual Rainfall

Box-plot:

```
boxplot(rainfall$ANN,col = "pink",ylab="Annual Rainfall",
        main= "Box-plot of Annual Rainfall")
```

Interpretation:

1.Annual variable look like following normal distribution which is also confirmed in descriptive statistics section. No outlier has been observed as well.

5.2.3 *Monsoon rainfall*

In this step, we are checking distribution of Monsoon rainfall variable by looking at histogram. We are also checking presence of outliers by looking at Box-plot.

Histogram:

```
# Normal distribution line in histogram
hist(rainfall$JJAS, freq=FALSE, col="skyblue", xlab="Monsoon Rainfall", main= "Histogram
curve(dnorm(x, mean=mean(rainfall$JJAS), sd=sd(rainfall$JJAS)), add=TRUE, col="red")
```

Interpretation:

1.The plot shows that the distribution of forecast errors is roughly centred on 850, and is more or less

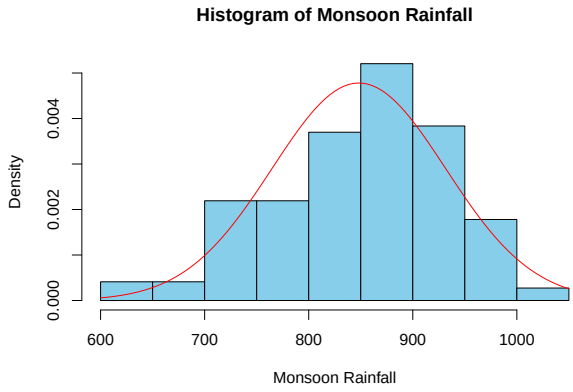


Figure 5.5: Histogram of Monsoon Rainfall

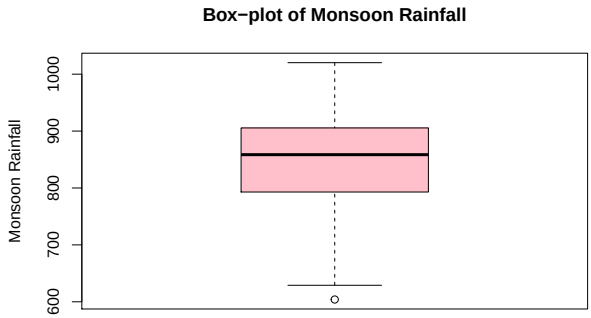


Figure 5.6: Boxplot of Monsoon Rainfall

normally distributed, although it seems to be slightly skewed to the right compared to a normal curve. However, the right skew is relatively small, and so it is plausible that the forecast errors are normally distributed with mean zero.

Box-plot:

```
boxplot(rainfall$JJAS,col = "pink",ylab="Monsoon Rainfall",
        main= "Box-plot of Monsoon Rainfall")
```

Interpretation:

1. Monsoon variable look like following normal distribution which is also confirmed in descriptive statistics section. There is one outlier year(1877) has been observed in which lowest rainfall has

been seen .

Heatmap for Mansoon Rainfall

Let's look at heatmap of monsoon rainfall. This heatmap has been created in **Tableau**.

```
knitr::include_graphics("Rainfall_Heatmap_1.pdf")
```

```
knitr::include_graphics("Rainfall_Heatmap_2.pdf")
```

```
knitr::include_graphics("Rainfall_Heatmap_3.pdf")
```

Interpretation::

1.The above diagram(Heatmap) shows that, the month june, jully, august, september shows the highest rainfall are shown as well as other month sows the lowest rainfall.

2.Then we have to interpret that those four months i.e southewest monsoon(JJAS) are very important for forecastin rainfall data.

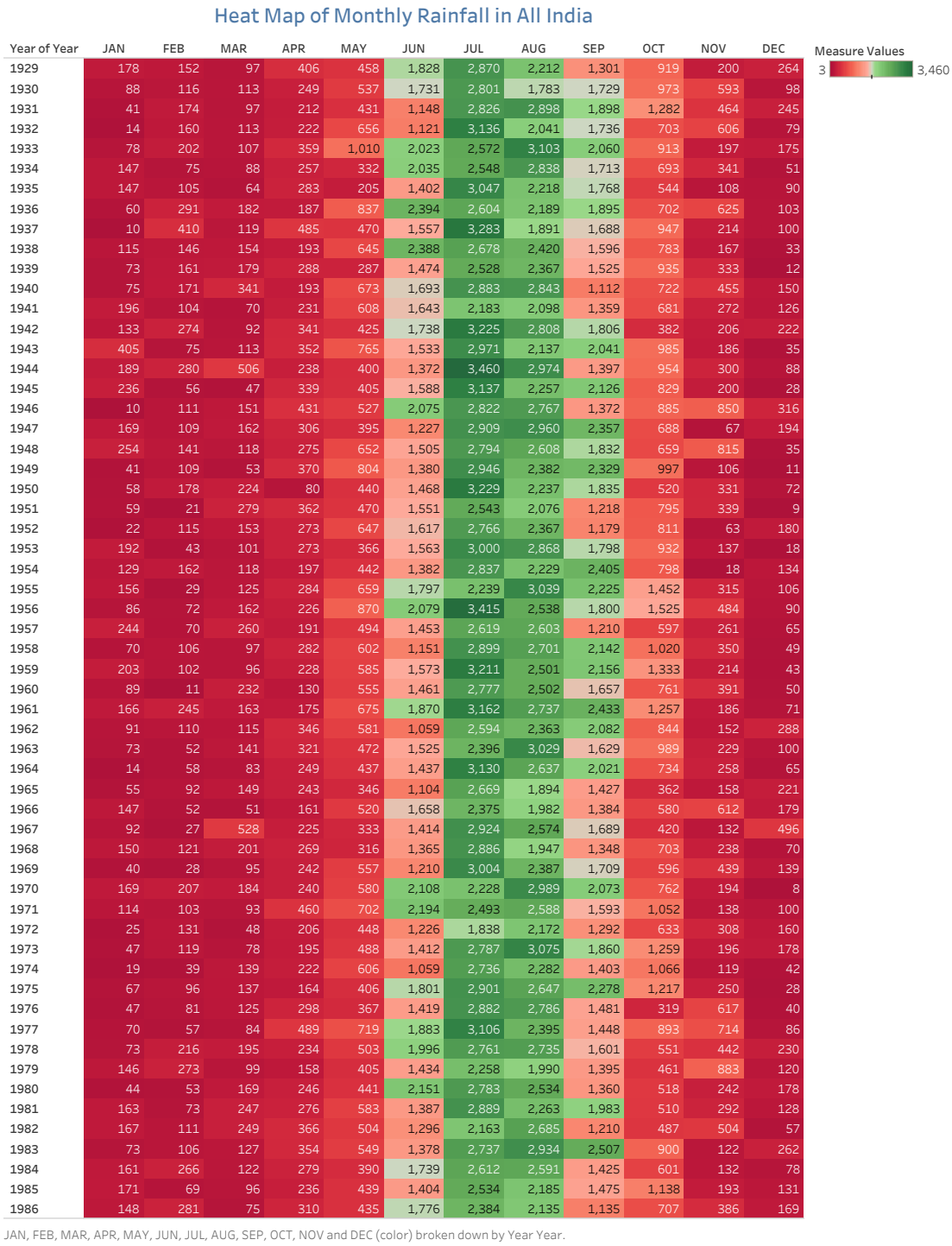
Heat Map of Monthly Rainfall in All India

Year of Year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1871	196	107	145	339	636	2,080	2,778	1,794	1,836	368	324	67
1872	76	75	73	240	438	1,892	2,913	2,452	1,879	785	276	191
1873	37	135	150	243	428	1,130	2,645	2,142	1,656	607	115	90
1874	86	158	107	169	683	2,279	3,069	2,335	2,062	932	187	40
1875	99	114	131	232	506	1,926	3,079	2,187	2,105	566	63	71
1876	9	21	160	165	425	1,233	2,968	1,956	1,620	464	95	25
1877	291	234	240	353	674	1,424	1,564	1,569	1,483	1,078	189	365
1878	108	91	102	361	665	1,304	2,940	3,393	2,124	798	273	140
1879	21	107	80	82	877	1,900	2,241	3,221	1,616	866	205	71
1880	39	165	151	211	496	1,876	2,717	1,768	1,840	877	528	111
1881	13	32	331	237	531	1,607	2,927	2,646	1,433	493	347	62
1882	100	94	90	188	596	2,130	3,311	1,903	1,685	739	596	62
1883	179	36	189	165	621	2,043	2,690	1,879	1,881	830	282	155
1884	71	74	82	203	383	1,593	2,924	2,426	2,385	866	270	305
1885	96	99	136	199	528	1,940	2,728	2,479	1,327	737	321	502
1886	49	12	222	124	711	1,977	3,159	2,218	1,390	1,320	214	171
1887	176	6	148	209	552	1,898	3,013	2,574	1,508	879	411	130
1888	197	136	119	259	531	1,323	2,743	2,800	1,250	382	531	72
1889	106	134	66	243	448	2,035	2,797	2,781	1,693	940	254	68
1890	26	11	173	261	358	2,279	3,009	2,141	1,626	647	351	136
1891	115	134	348	207	595	834	2,571	2,297	2,222	602	173	71
1892	53	106	56	329	541	1,579	3,139	3,063	2,135	993	126	70
1893	224	283	467	252	946	2,416	2,568	2,309	2,256	970	642	22
1894	115	144	181	248	360	2,168	3,114	2,415	2,017	1,388	385	137
1895	109	77	118	376	415	1,982	2,567	2,299	1,388	793	134	119
1896	31	61	68	170	412	2,034	2,845	2,633	773	149	435	175
1897	92	124	208	216	435	1,426	2,599	2,905	1,981	856	130	18
1898	21	307	28	211	375	1,717	2,929	2,208	1,992	655	393	125
1899	58	66	74	522	526	1,951	1,878	1,441	1,020	507	41	26
1900	153	67	103	385	406	1,287	2,521	2,713	2,372	490	104	141
1901	274	337	113	316	391	1,149	2,225	2,590	1,256	540	431	95
1902	67	25	102	354	418	1,038	2,796	1,972	2,112	697	278	269
1903	85	55	102	121	562	1,269	2,789	2,593	1,957	1,288	434	152
1904	59	86	203	311	686	1,795	2,476	1,970	1,262	707	90	99
1905	128	144	286	271	476	908	2,456	2,052	1,748	610	98	25
1906	151	385	220	135	328	1,818	2,875	2,418	1,738	544	199	264
1907	63	324	256	523	264	1,541	2,217	3,014	1,003	205	238	128
1908	156	126	68	156	390	1,278	3,164	2,955	1,573	424	59	18
1909	162	57	30	605	519	2,098	2,961	2,226	1,608	369	101	244
1910	54	42	79	211	357	2,117	2,377	2,827	2,028	1,085	379	3
1911	162	13	241	196	441	1,925	1,534	2,110	1,794	725	415	114
1912	70	187	129	309	362	1,058	3,206	2,579	1,215	660	559	25
1913	12	310	163	204	701	2,151	2,642	1,881	1,170	731	179	208
1914	12	121	147	345	634	1,616	3,208	2,328	1,828	410	206	156
1915	183	256	390	259	631	1,567	2,298	2,131	1,810	887	545	61
1916	12	93	33	233	481	1,991	2,582	2,897	2,036	1,473	508	29
1917	32	346	146	233	714	2,143	2,466	2,755	2,678	1,595	316	46
1918	106	20	145	186	880	1,813	1,435	2,205	1,055	147	472	111
1919	374	122	115	203	512	1,794	2,712	2,808	1,531	707	590	137
1920	207	87	255	228	454	1,426	2,916	1,628	1,220	484	303	5
1921	295	23	110	319	295	1,672	2,666	2,362	1,958	663	191	46
1922	228	47	42	169	371	1,803	2,910	2,028	1,949	518	648	92
1923	107	263	158	207	398	936	2,970	2,604	1,720	608	207	139
1924	141	70	54	226	406	1,108	2,999	2,250	2,269	617	628	106
1925	43	24	94	305	806	1,871	2,858	2,133	1,174	720	434	176
1926	267	50	418	270	518	815	2,876	3,236	2,099	531	109	57
1927	79	232	163	233	441	1,611	3,194	2,218	1,507	613	611	94
1928	99	282	86	229	409	1,541	2,834	1,982	1,320	1,238	169	168

Measure Values
3 3,460

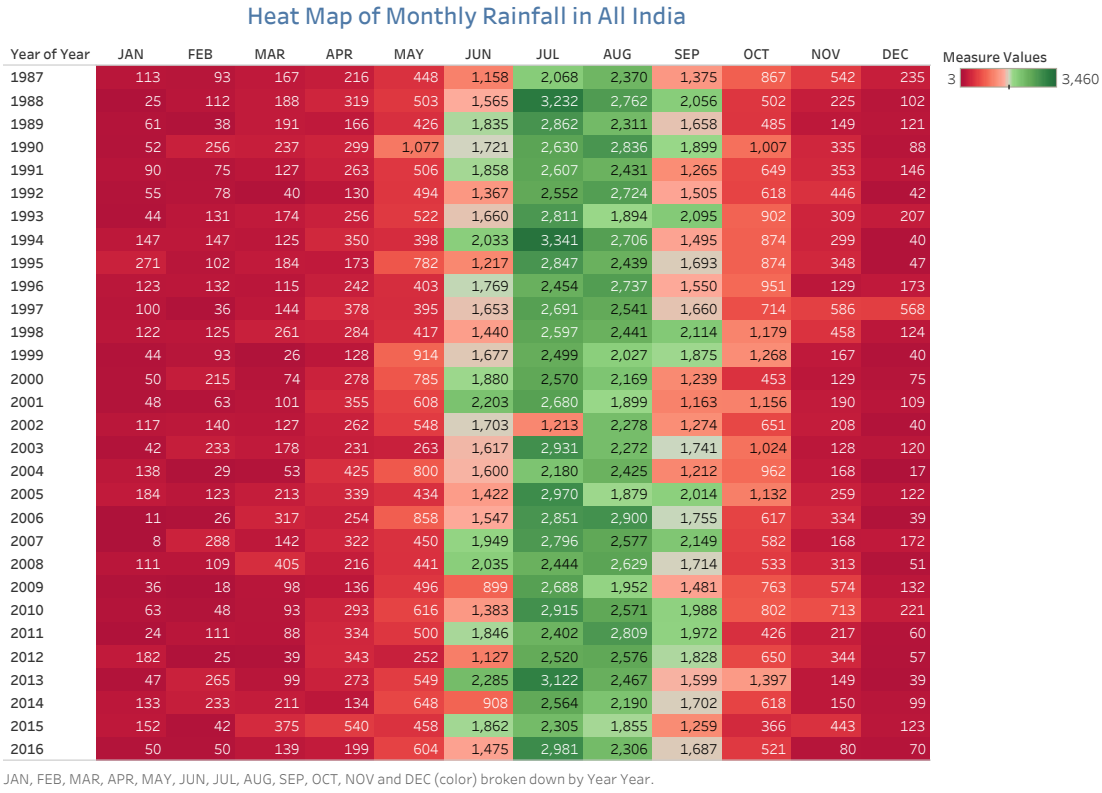
JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV and DEC (color) broken down by Year Year.

Figure 5.7: Heatmap of Monsoon Rainfall Part 1



Year of Year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1929	178	152	97	406	458	1,828	2,870	2,212	1,301	919	200	264
1930	88	116	113	249	537	1,731	2,801	1,783	1,729	973	593	98
1931	41	174	97	212	431	1,148	2,826	2,898	1,898	1,282	464	245
1932	14	160	113	222	656	1,121	3,136	2,041	1,736	703	606	79
1933	78	202	107	359	1,010	2,023	2,572	3,103	2,060	913	197	175
1934	147	75	88	257	332	2,035	2,548	2,838	1,713	693	341	51
1935	147	105	64	283	205	1,402	3,047	2,218	1,768	544	108	90
1936	60	291	182	187	837	2,394	2,604	2,189	1,895	702	625	103
1937	10	410	119	485	470	1,557	3,283	1,891	1,688	947	214	100
1938	115	146	154	193	645	2,388	2,678	2,420	1,596	783	167	33
1939	73	161	179	288	287	1,474	2,528	2,367	1,525	935	333	12
1940	75	171	341	193	673	1,693	2,883	2,843	1,112	722	455	150
1941	196	104	70	231	608	1,643	2,183	2,098	1,359	681	272	126
1942	133	274	92	341	425	1,738	3,225	2,808	1,806	382	206	222
1943	405	75	113	352	765	1,533	2,971	2,137	2,041	985	186	35
1944	189	280	506	238	400	1,372	3,460	2,974	1,397	954	300	88
1945	236	56	47	339	405	1,588	3,137	2,257	2,126	829	200	28
1946	10	111	151	431	527	2,075	2,822	2,767	1,372	885	850	316
1947	169	109	162	306	395	1,227	2,909	2,960	2,357	688	67	194
1948	254	141	118	275	652	1,505	2,794	2,608	1,832	659	815	35
1949	41	109	53	370	804	1,380	2,946	2,382	2,329	997	106	11
1950	58	178	224	80	440	1,468	3,229	2,237	1,835	520	331	72
1951	59	21	279	362	470	1,551	2,543	2,076	1,218	795	339	9
1952	22	115	153	273	647	1,617	2,766	2,367	1,179	811	63	180
1953	192	43	101	273	366	1,563	3,000	2,868	1,798	932	137	18
1954	129	162	118	197	442	1,382	2,837	2,229	2,405	798	18	134
1955	156	29	125	284	659	1,797	2,239	3,039	2,225	1,452	315	106
1956	86	72	162	226	870	2,079	3,415	2,538	1,800	1,525	484	90
1957	244	70	260	191	494	1,453	2,619	2,603	1,210	597	261	65
1958	70	106	97	282	602	1,151	2,899	2,701	2,142	1,020	350	49
1959	203	102	96	228	585	1,573	3,211	2,501	2,156	1,333	214	43
1960	89	11	232	130	555	1,461	2,777	2,502	1,657	761	391	50
1961	166	245	163	175	675	1,870	3,162	2,737	2,433	1,257	186	71
1962	91	110	115	346	581	1,059	2,594	2,363	2,082	844	152	288
1963	73	52	141	321	472	1,525	2,396	3,029	1,629	989	229	100
1964	14	58	83	249	437	1,437	3,130	2,637	2,021	734	258	65
1965	55	92	149	243	346	1,104	2,669	1,894	1,427	362	158	221
1966	147	52	51	161	520	1,658	2,375	1,982	1,384	580	612	179
1967	92	27	528	225	333	1,414	2,924	2,574	1,689	420	132	496
1968	150	121	201	269	316	1,365	2,886	1,947	1,348	703	238	70
1969	40	28	95	242	557	1,210	3,004	2,387	1,709	596	439	139
1970	169	207	184	240	580	2,108	2,228	2,989	2,073	762	194	8
1971	114	103	93	460	702	2,194	2,493	2,588	1,593	1,052	138	100
1972	25	131	48	206	448	1,226	1,838	2,172	1,292	633	308	160
1973	47	119	78	195	488	1,412	2,787	3,075	1,860	1,259	196	178
1974	19	39	139	222	606	1,059	2,736	2,282	1,403	1,066	119	42
1975	67	96	137	164	406	1,801	2,901	2,647	2,278	1,217	250	28
1976	47	81	125	298	367	1,419	2,882	2,786	1,481	319	617	40
1977	70	57	84	489	719	1,883	3,106	2,395	1,448	893	714	86
1978	73	216	195	234	503	1,996	2,761	2,735	1,601	551	442	230
1979	146	273	99	158	405	1,434	2,258	1,990	1,395	461	883	120
1980	44	53	169	246	441	2,151	2,783	2,534	1,360	518	242	178
1981	163	73	247	276	583	1,387	2,889	2,263	1,983	510	292	128
1982	167	111	249	366	504	1,296	2,163	2,685	1,210	487	504	57
1983	73	106	127	354	549	1,378	2,737	2,934	2,507	900	122	262
1984	161	266	122	279	390	1,739	2,612	2,591	1,425	601	132	78
1985	171	69	96	236	439	1,404	2,534	2,185	1,475	1,138	193	131
1986	148	281	75	310	435	1,776	2,384	2,135	1,135	707	386	169

Figure 5.8: Heatmap of Monsoon Rainfall Part 2



Year of Year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1987	113	93	167	216	448	1,158	2,068	2,370	1,375	867	542	235
1988	25	112	188	319	503	1,565	3,232	2,762	2,056	502	225	102
1989	61	38	191	166	426	1,835	2,862	2,311	1,658	485	149	121
1990	52	256	237	299	1,077	1,721	2,630	2,836	1,899	1,007	335	88
1991	90	75	127	263	506	1,858	2,607	2,431	1,265	649	353	146
1992	55	78	40	130	494	1,367	2,552	2,724	1,505	618	446	42
1993	44	131	174	256	522	1,660	2,811	1,894	2,095	902	309	207
1994	147	147	125	350	398	2,033	3,341	2,706	1,495	874	299	40
1995	271	102	184	173	782	1,217	2,847	2,439	1,693	874	348	47
1996	123	132	115	242	403	1,769	2,454	2,737	1,550	951	129	173
1997	100	36	144	378	395	1,653	2,691	2,541	1,660	714	586	568
1998	122	125	261	284	417	1,440	2,597	2,441	2,114	1,179	458	124
1999	44	93	26	128	914	1,677	2,499	2,027	1,875	1,268	167	40
2000	50	215	74	278	785	1,880	2,570	2,169	1,239	453	129	75
2001	48	63	101	355	608	2,203	2,680	1,899	1,163	1,156	190	109
2002	117	140	127	262	548	1,703	1,213	2,278	1,274	651	208	40
2003	42	233	178	231	263	1,617	2,931	2,272	1,741	1,024	128	120
2004	138	29	53	425	800	1,600	2,180	2,425	1,212	962	168	17
2005	184	123	213	339	434	1,422	2,970	1,879	2,014	1,132	259	122
2006	11	26	317	254	858	1,547	2,851	2,900	1,755	617	334	39
2007	8	288	142	322	450	1,949	2,796	2,577	2,149	582	168	172
2008	111	109	405	216	441	2,035	2,444	2,629	1,714	533	313	51
2009	36	18	98	136	496	899	2,688	1,952	1,481	763	574	132
2010	63	48	93	293	616	1,383	2,915	2,571	1,988	802	713	221
2011	24	111	88	334	500	1,846	2,402	2,809	1,972	426	217	60
2012	182	25	39	343	252	1,127	2,520	2,576	1,828	650	344	57
2013	47	265	99	273	549	2,285	3,122	2,467	1,599	1,397	149	39
2014	133	233	211	134	648	908	2,564	2,190	1,702	618	150	99
2015	152	42	375	540	458	1,862	2,305	1,855	1,259	366	443	123
2016	50	50	139	199	604	1,475	2,981	2,306	1,687	521	80	70

JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV and DEC (color) broken down by Year Year.

Figure 5.9: Heatmap of Monsoon Rainfall Part 3

6 FORECASTING ANNUAL RAINFALL

Annual rainfall is the average amount of total rain that a place generally receives.

When we say annual rainfall of India place is x mm, it does not imply that for a particular year the total rainfall my place received was x mm rather it means in general or on an average my place receives x mm of rainfall annually.

Time Series:

A set of ordered observations of a quantitative variables taken at successive points in time is known as 'Time Series'. In other words , arrangement of statistical data in chronological order, i.e., in accordance with occurrence of time , is known as 'Time Series'. Time in terms of years , months, days, or hours, is simply device that enables one to relate all phenomenon to a set of common, stable reference points.

Mathematically, a time series is defined by the functional relationship

$$y_t = f(t)$$

Components of time series:

The various forces at work, affecting the values of a phenomenon in a time series, can be broadly classified into the four categories, commonly known as the components of time series, and they as follows.

- Secular Trend or Long-term movement
- Periodic Changes or Short Term Fluctuations
  - Seasonal variations
  - Cyclic variations
- Random or Irregular Movements

Mathematical Models for Time Series:

The Following are the two models commonly used for the decomposition of a time series into components .

1. Decomposition by Additive Model

$$Y_t = T_t + S_t + C_t + R_t$$

2. Decomposition by Multiplicative Model

$$Y_t = T_t * S_t * C_t * R_t$$

Uses of Time Series:

1. It enables us to study the past behavior of the phenomenon under consideration, i.e., to determine the type and nature of the variations in the data.
2. It enables to predict or estimate or forecast the behavior of the phenomenon in future which is very essential for business planning.
3. It helps us to compare the changes in the values of different phenomenon at different times or places, etc.

Prediction and forecasting:

In statistics, prediction is a part of statistical inference. One particular approach to such inference is known as predictive inference, but the prediction can be undertaken within any of the several approaches to statistical inference. Indeed, one description of statistics is that it provides a means of transferring knowledge about a sample of a population to the whole population, and to other related populations, which is not necessarily the same as prediction over time. When information is transferred across time, often to specific points in time, the process is known as forecasting.

- Fully formed statistical models for stochastic simulation purposes, so as to generate alternative

versions of the time series, representing what might happen over non-specific time-periods in the future

- Simple or fully formed statistical models to describe the likely outcome of the time series in the immediate future, given knowledge of the most recent outcomes (forecasting).
- Forecasting on time series is usually done using automated statistical software packages and programming languages, such as Wolfram Mathematica, R, S, SAS, SPSS, Minitab, pandas (Python) and many others.
- Forecasting on large scale data is done using Spark which has spark-ts as a third party package.

Forecasting:

Forecasting is the process of making predictions of the future based on past and present data and most commonly by analysis of trends. A commonplace example might be estimation of some variable of interest at some specified future date. Prediction is a similar, but more general term. Both might refer to formal statistical methods employing time series, cross-sectional or longitudinal data, or alternatively to less formal judgmental methods. Usage can differ between areas of application: for example, in hydrology the terms "forecast" and "forecasting" are sometimes reserved for estimates of values at certain specific future times, while the term "prediction" is used for more general estimates, such as the number of times floods will occur over a long period. Risk and uncertainty are central to forecasting and prediction; it is generally considered good practice to indicate the degree of uncertainty attaching to forecasts. In any case, the data must be up to date in order for the forecast to be as accurate as possible. In some cases the data used to predict the variable of interest is itself forecasted.

6.1 Building time series model

Let's start building our time series models. At first, we will plot historical trend of annual rainfall using `ggplot2` package.

Plotting Time Series data:

```
rainfall%>%ggplot(aes(YEAR,ANN))+geom_line()+
  geom_point(alpha = 0.5, color = palette_light()[[1]], shape=20,size=2) +
```

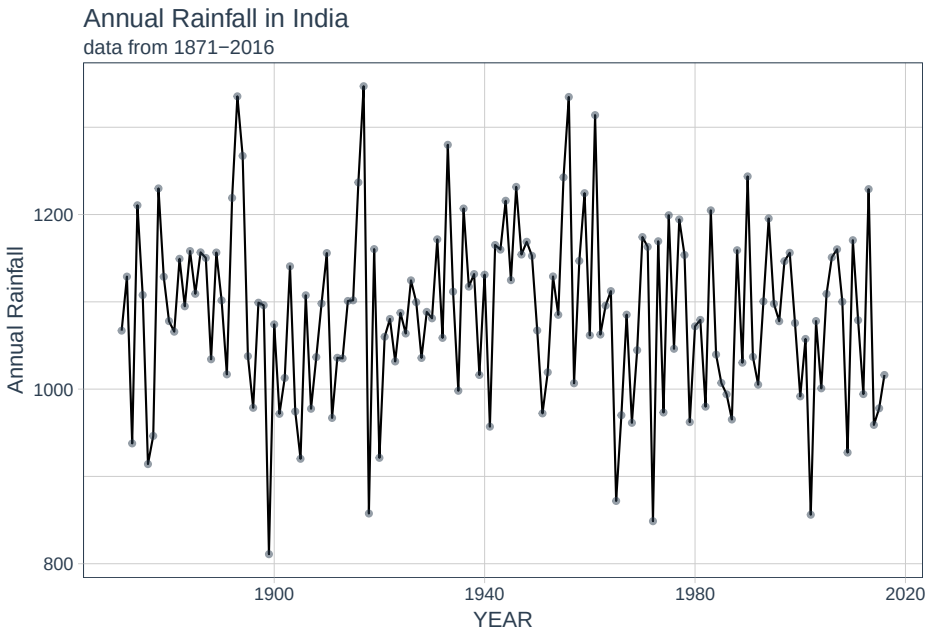


Figure 6.1: Historical trend of annual rainfall

```
  labs(title = "Annual Rainfall in India", x = "YEAR", y = "Annual Rainfall",
       subtitle = "data from 1871-2016") +
   theme_tq()
```

Interpretation:

1. The above figure 6.1 shown the seasonality is present in the data.

Combo-chart for Annual Rainfall:

Let's look at some complex visulization for annual rainfall:

```
#Calculating Mean for ANN
ANN_mean = mean(rainfall$ANN)


#Calculating percentage departure From Mean for ANN
rainfall$Perc_departure_ANN = ((rainfall$ANN-ANN_mean)/ANN_mean)*100


#plotting combo-chart for ANN
```

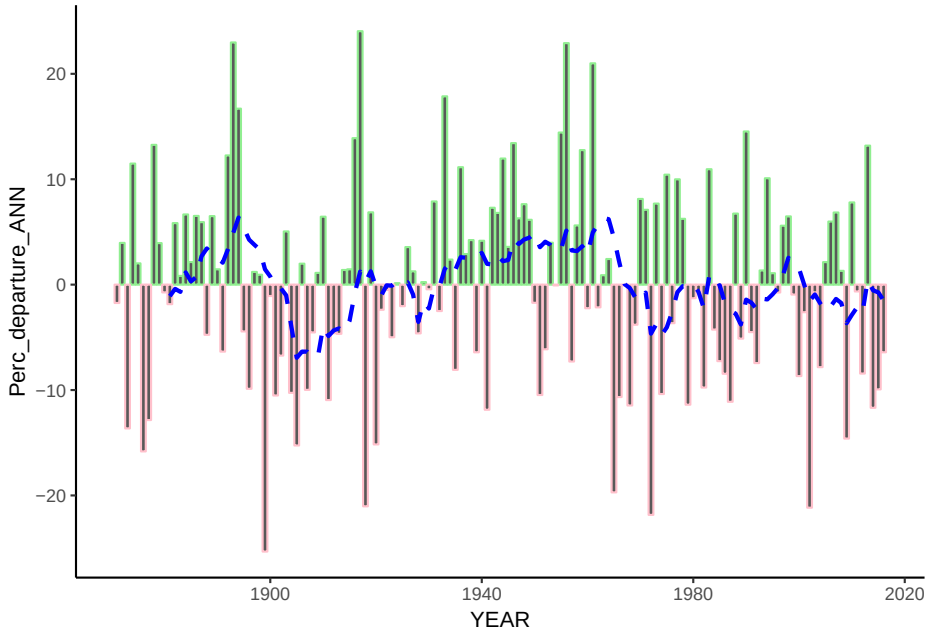


Figure 6.2: Combo chart of annual rainfall

```
rainfall1 <- subset(rainfall, Perc_departure_ANN>= 0)
rainfall2 <- subset(rainfall, Perc_departure_ANN < 0)

ggplot() +
     geom_bar(data = rainfall1, aes(x=YEAR, y=Perc_departure_ANN),
              stat = "identity",colour = "lightgreen") +
     geom_bar(data = rainfall2, aes(x=YEAR, y=Perc_departure_ANN),
              stat = "identity",colour = "pink")  +
  geom_ma(data = rainfall,aes(x= YEAR, y= Perc_departure_ANN),
          ma_fun = SMA, n = 11, color = "blue",size = 1)
```

Interpretatiion:

1. The above fig.6.2 shows the trend line of year 1970 to 2016 are mostly decreasing, which shows the annual rainfall data as compaired to past year data is minimum.

6.2 Model Building

Models for time series data can have many forms and represent different stochastic processes. When modeling variations in the level of a process, broad classes of practical importance are Exponential Smoothing models, the autoregressive (AR) models, the integrated (I) models, and the moving average (MA) models. These three classes depend linearly on previous data points. Combinations of these ideas produce autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) models. The autoregressive fractionally integrated moving average (ARFIMA) model generalizes the former three. Extensions of these classes to deal with vector-valued data are available under the heading of multivariate time-series models and sometimes the preceding acronyms are extended by including an initial "V" for "vector", as in VAR for vector autoregression. An additional set of extensions of these models is available for use where the observed time-series is driven by some "forcing" time-series (which may not have a causal effect on the observed series): the distinction from the multivariate case is that the forcing series may be deterministic or under the experimenter's control. For these models, the acronyms are extended with a final "X" for "exogenous".

Non-linear dependence of the level of a series on previous data points is of interest, partly because of the possibility of producing a chaotic time series. However, more importantly, empirical investigations can indicate the advantage of using predictions derived from non-linear models, over those from linear models, as for example in nonlinear autoregressive exogenous models. Further references on nonlinear time series analysis: (Kantz and Schreiber),and (Abarbanel)

Among other types of non-linear time series models, there are models to represent the changes of variance over time (heteroskedasticity). These models represent autoregressive conditional heteroskedasticity(ARCH) and the collection comprises a wide variety of representation (GARCH, TARCH, EGARCH, FIGARCH, CGARCH, etc.). Here changes in variability are related to, or predicted by, recent past values of the observed series. This is in contrast to other possible representations of locally varying variability, where the variability might be modelled as being driven by a separate time-varying process, as in a doubly stochastic model.

In recent work on model-free analyses, wavelet transform based methods (for example locally

stationary wavelets and wavelet decomposed neural networks) have gained favor. Multiscale (often referred to as multiresolution) techniques decompose a given time series, attempting to illustrate time dependence at multiple scales. See also Markov switching multifractal (MSMF) techniques for modeling volatility evolution.

A Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be considered as the simplest dynamic Bayesian network. HMM models are widely used in speech recognition, for translating a time series of spoken words into text.

For building any forecasting model, below are some key step need to follow:

1. Splitting into train and test:

For validation mechanism, we are spilitting our datasets into training and testing data.

2. Identifying the model performance metrics:

For identification of time series model performance metrics,we are using Scale dependent errors. It is define as forecast errors are on the same scale as the data. Accuracy measures that are based only on error are therefore scale-dependent and cannot be used to make comparisons between series that involve different units.

The two most commonly used scale-dependent measures are based on the absolute errors or squared errors. We are using below accuracy metrics for our models:

- Mean absolute error(MAE)
- Root mean squared error(RMSE)
- Mean absolute percentage error(MAPE)

The formula for Mean absolute error(MAE) is as follow:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(|y_i - \hat{y}_i|)$$

The formula for Root mean squared error(RMSE) is as follow:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$$

When comparing forecast methods applied to a single time series, or to several time series with the same units, the MAE is popular as it is easy to both understand and compute. A forecast method that minimises the MAE will lead to forecasts of the median, while minimising the RMSE will lead to forecasts of the mean. Consequently, the RMSE is also widely used, despite being more difficult to interpret.

However,it is widely seen reporting any error in Percentage form. Percentage errors have the advantage of being unit-free, and so are frequently used to compare forecast performances between data sets. The most commonly used measure is Mean absolute percentage error.

Formula for MAPE is as below:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Actual - Predicted|}{Actual} * 100$$

3. Exploring different time series model

In this step, we are exploring below time series models:

1. Simple Exponential Smoothing
2. Exponential smoothing
3. ARIMA Model
4. Neural Network Modals

6.2.1 *Splitting into train and test*

In this step, we are splitting the data to create the train and test subsets. Commonly, 80/20 split is performed. Unlike regression and classification models in which datasets are typically randomized and put into either a train set or test set, time series are not split in a randomized fashion model is built

on time related event. When separating time series sets the train set is the the older 80% of observations and the test set is the more recent 20% of observations. The reason we do not randomize the data are because we want to know how well our model is going to predict future observations.

- A model which fits the training data well will not necessarily forecast well.
- A perfect fit can always be obtained by using a model with enough parameters.
- Over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data.

```
series.end <- floor(0.8*nrow(rainfall)) #select the first 80% of the data
train <- rainfall[1:series.end,] #assign the first 80% of the data to the train set
test <- rainfall[(series.end+1):nrow(rainfall),] #assign the most recent 20% to the tes
```

6.2.2 *Converting data into time series*

```
# Convert data frame into time serise object


train.ts <- ts(train, start = c(1871), end = c(1986))
test.ts <- ts(test, start = c(1987), end = c(2016))
```

6.2.3 *Test Stationary or Non-stationary*

The null hypothesis assumes that the series is non-stationary (mean does not stay constant). ADF procedure tests whether the change in Y can be explained by lagged value and a linear trend. If contribution of the lagged value to the change in Y is non-significant and there is a presence of a trend component, the series is non-stationary and null hypothesis will not be rejected.

```
pander(adf.test(train.ts[,18], alternative = "stationary"))
```

Table 6.1: Augmented Dickey-Fuller Test: `train.ts[, 18]`

Test statistic	Lag order	P value	Alternative hypothesis
-4.49	4	0.01 * *	stationary

Interpretation:

The annual rainfall data is stationary(p-value = 0.01) by the ADF test hence there is no need of differencing the series. Since p-value is less then 5%, we can reject null hypothesis. This means that our series is stationary.

6.2.4 *Forecasts using Exponential Smoothing*

Exponential smoothing is a common method for making short-term forecasts in time series data.

6.2.4.1 *Simple Exponential Smoothing*

If we have a time series with constant level and no seasonality, we can use simple exponential smoothing for short term forecasts.

The simplest of the exponentially smoothing methods is naturally called simple exponential smoothing (SES). This method is suitable for forecasting data with no clear trend or seasonal pattern

This method is a way of estimating the level at the current time point. Smoothing is controlled by the parameter `alpha` for the estimate of the level at the current time point. The value of alpha lies between 0 and 1. Values of alpha close to 0 mean that little weight is places on the most recent observations when making forecasts of future values.

In **R** we can use the `HoltWinters()` function. For simple exponential smoothing, we need to set the parameters `beta = FALSE` and `gamma = FALSE`.

Key points regading simple exponential smoothing:

1. The simple exponential smoothing (SES) is a short-range forecasting method that assumes a reasonably stable mean in the data with no trend.
2. The exponential smoothing provides an idea that the most recent observations usually give the best guide to the future, therefore we want a weighting scheme with decreasing weights for observations.
3. The choice of the smoothing constant is important in determining the operating characteristics

of exponential smoothing.

4. The smaller the value of alpha , the slower the response. Larger value of alpha cause the smoothed value to react quickly-not only to real changes but also random fluctuations.
5. Simple exponential smoothing model is only good for non-seasonal patterns with approximately zero trend and for short term forecasting because we extend past the next period, the forecasted value for that period has to be used as surrogate for the actual demand for any forecast past the next period.
6. Consequently, there is no ability to add corrective information (the actual demand) and any error grows exponentially.

Key points regading Holt-winters:

1. Holt-Winters smoothing is widely used tool for forecasting business data that contain seasonality, changing trends and seasonal correlation.
2. The holt-winter method, also referred to as double exponential smoothing, is an extension of exponential smoothing designed for trended and seasonal time series.
3. Holt-Winters forecasting is surprisingly powerful despite its simplicity. It can handle many complicated seasonal patterns by simply finding the central value, then adding in the effects of slope and seasonality.

Let's build simple expontional smoothing model.

```
options("scipen"=100, "digits"=2)
# Simple Exponetial Smoothing
SimpleExpSmooth <- HoltWinters(train.ts[,18], beta=FALSE, gamma = FALSE)
SimpleExpSmooth
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = train.ts[, 18], beta = FALSE, gamma = FALSE)
##
```

```
## Smoothing parameters:
##  alpha: 0.018
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##    [,1]
## a 1086
```

```
# Plotting the estimates
# plot(SimpleExpSmooth)


train_forecast = as.data.frame(SimpleExpSmooth$fitted)
train_ANN = train[,c("YEAR", "ANN")]
train_ANN = train_ANN[2:116, ]
train_ANN = cbind(train_ANN, train_forecast)
train_ANN$xhat <- NULL
setnames(train_ANN, "level", "Forecast")


train_ANN$Error = (train_ANN$ANN - train_ANN$Forecast)^2
train_ANN$AbsError = abs(train_ANN$ANN - train_ANN$Forecast)
train_ANN$APE = abs(train_ANN$ANN - train_ANN$Forecast)/train_ANN$ANN



SampleSize = nrow(train_ANN)
MSE = sum(train_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_ANN$AbsError)/SampleSize
MAPE = (sum(train_ANN$APE)/SampleSize)*100


statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
```

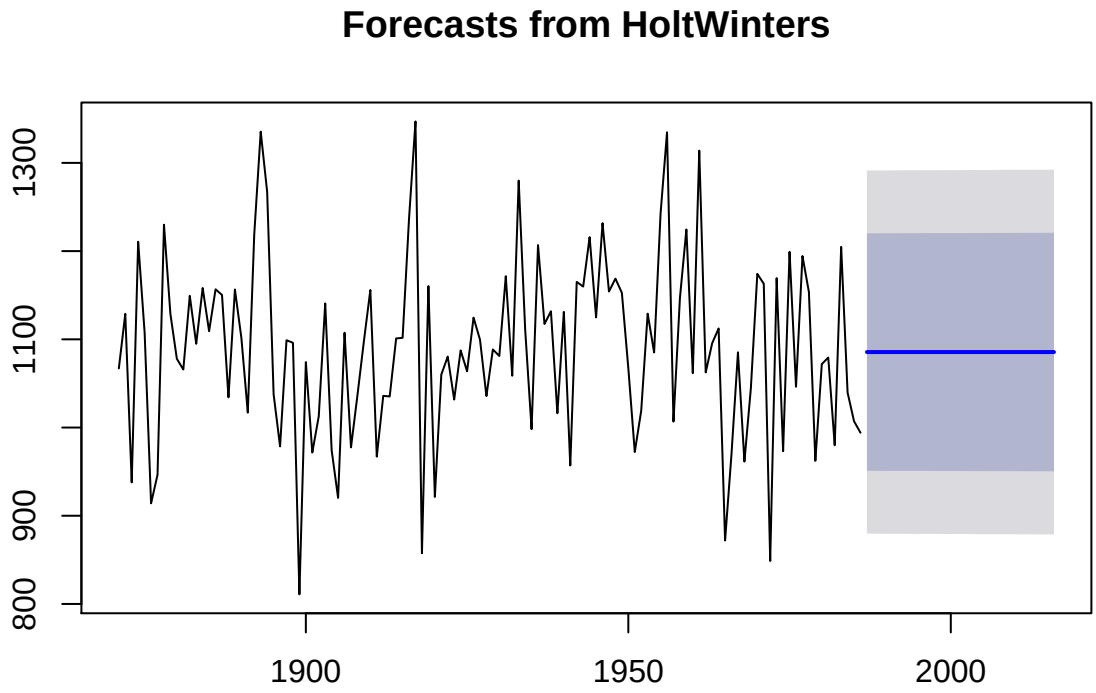


Figure 6.3: Forecasted value using simple expontional smoothing for Annual Rainfall

```
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)


train_statistics <- data.frame(cbind(statistics, Value))
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
rownames(train_statistics) <-  NULL



library(forecast)
prediction_SES <- forecast(SimpleExpSmooth, h=30)
plot(prediction_SES)
```

```
Forecast = as.data.frame(prediction_SES$mean)


test_ANN = test[,c("YEAR", "ANN")]
test_ANN = cbind(test_ANN, Forecast)
test_ANN = data.table(test_ANN)
setnames(test_ANN, 'x', 'Forecast')


test_ANN$Error = (test_ANN$ANN - test_ANN$Forecast)^2
test_ANN$AbsError = abs(test_ANN$ANN - test_ANN$Forecast)
test_ANN$APE = abs(test_ANN$ANN - test_ANN$Forecast)/test_ANN$ANN


#kable(test_ANN, "latex",longtable=T,booktabs=T,caption="Simple Exponential Smoothing
#  kable_styling(latex_options = c("striped", "scale_down"))


SampleSize = nrow(test_ANN)
MSE = sum(test_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_ANN$AbsError)/SampleSize
MAPE = (sum(test_ANN$APE)/SampleSize)*100


statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)


test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL


Finalresult_SES = merge(train_statistics, test_statistics, by ="statistics")
```

```
kable(Finalresult_SES, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Annua
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 6.2: Accuracy Metrics of Annual Rainfall using Simple Exponential Smoothing

statistics	training_set	testing_set
MAE	83.1545447979645	74.4177828512167
MAPE	7.73522961725787	7.15903856937843
MSE	11016.121643753	8394.97129760281
RMSE	104.957713598158	91.6240759713451
SampleSize	115	30

Interpretation:

1.To make forecasts using simple exponential smoothing in R, we can fit a simple exponential smoothing predictive model using the "HoltWinters()" function in R. To use HoltWinters() for simple exponential smoothing, we need to set the parameters beta=FALSE and gamma=FALSE in the HoltWinters() function (the beta and gamma parameters are used for Holt's exponential smoothing, or Holt-Winters exponential smoothing

2.By default, HoltWinters() just makes forecasts for the same time period covered by our original time series. In this case, our original time series included rainfall for All India from 1871-1986, so the forecasts are also for 1871-1986.

3.The output of HoltWinters() tells us that the estimated value of the alpha parameter is about 0.018. This is very close to zero, telling us that the forecasts are based on both recent and less recent observations (although somewhat more weight is placed on recent observations).

4.Here the forecasts for 1987-2016 are plotted as a blue line, the 80% prediction interval as an sky blue shaded area, and the 95% prediction interval as a grey shaded area.

5.From graph, The plot shows the original time series in black, and the forecasts as a blue line. The time series of forecasts is much smoother than the time series of the original data here.

6. Overall model accuraacy is 92.85% (MAPE = 7.15%) indecating good fit of the model. This model also does not exhibit over-fitting as training and test error are almost same.

6.2.4.2 *Exponential Smoothing*

Exponential smoothing is a rule of thumb technique for smoothing time series data using the exponential window function. Whereas in the simple moving average the past observations are weighted equally, exponential functions are used to assign exponentially decreasing weights over time. It is an easily learned and easily applied procedure for making some determination based on prior assumptions by the user, such as seasonality. Exponential smoothing is often used for analysis of time-series data. Exponential smoothing is one of many window functions commonly applied to smooth data in signal processing, acting as low-pass filters to remove high frequency noise. This method is preceded by Poisson's use of recursive exponential window functions in convolutions from the 19th century, as well as Kolmogorov and Zurbenko's use of recursive moving averages from their studies of turbulence in the 1940s.

The raw data sequence is often represented by beginning at time $t = 0$, and the output of the exponential smoothing algorithm is commonly written as s_t, which may be regarded as a best estimate of what the next value of x will be. When the sequence of observations begins at time $t = 0$, the simplest form of exponential smoothing is given by the formulas.

$$s_0 = x_0$$

$$s_t = \alpha x_t + (1 - \alpha) s_{t-1}, t > 0$$

where α is the smoothing factor, and $0 < \alpha < 1$

This simple form of exponential smoothing is also known as an exponentially weighted moving

average (EWMA). Technically it can also be classified as an autoregressive integrated moving average (ARIMA) (0,1,1) model with no constant term.

```
options("scipen"=100, "digits"=2)
# Exponetial Smoothing
ExpSmooth <- HoltWinters(train.ts[,18], gamma = FALSE)
ExpSmooth
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = train.ts[, 18], gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.28
##  beta : 0.16
##  gamma: FALSE
##
## Coefficients:
##     [,1]
## a 1039.8
## b   -5.6
```

```
# Plotting the estimates
#plot(ExpSmooth)


train_forecast = as.data.frame(ExpSmooth$fitted)
train_ANN = train[,c("YEAR", "ANN")]
train_ANN = train_ANN[3:116, ]
train_ANN = cbind(train_ANN, train_forecast)
train_ANN$xhat <- NULL
setnames(train_ANN, "level", "Forecast")
```

```
train_ANN$Error = (train_ANN$ANN - train_ANN$Forecast)^2
train_ANN$AbsError = abs(train_ANN$ANN - train_ANN$Forecast)
train_ANN$APE = abs(train_ANN$ANN - train_ANN$Forecast)/train_ANN$ANN



SampleSize = nrow(train_ANN)
MSE = sum(train_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_ANN$AbsError)/SampleSize
MAPE = (sum(train_ANN$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

train_statistics <- data.frame(cbind(statistics, Value))
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
rownames(train_statistics) <-  NULL

library(forecast)
prediction_SES <- forecast(ExpSmooth, h=30)
plot(prediction_SES)
```

```
Forecast = as.data.frame(prediction_SES$mean)


test_ANN = test[,c("YEAR", "ANN")]
test_ANN = cbind(test_ANN, Forecast)
test_ANN = data.table(test_ANN)
setnames(test_ANN, 'x', 'Forecast')
```

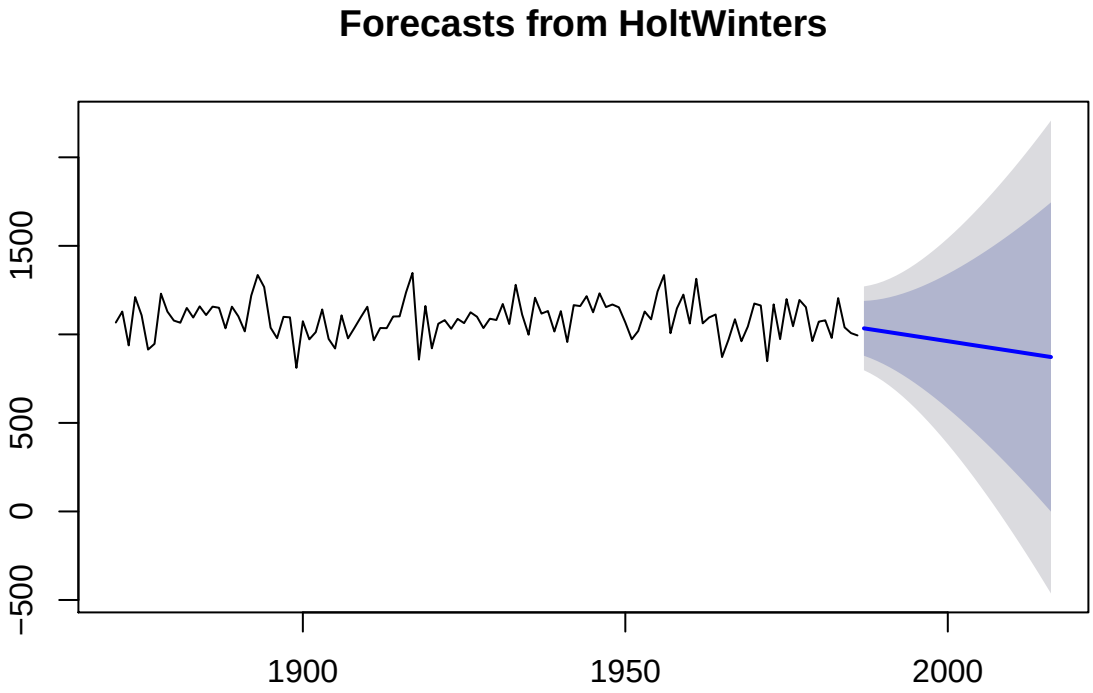


Figure 6.4: Forecasted value using expontional smoothing for Annual Rainfall

```
test_ANN$Error = (test_ANN$ANN - test_ANN$Forecast)^2
test_ANN$AbsError = abs(test_ANN$ANN - test_ANN$Forecast)
test_ANN$APE = abs(test_ANN$ANN - test_ANN$Forecast)/test_ANN$ANN
#kable(test_ANN, "latex",longtable=T,booktabs=T,caption="Exponential Smoothing forecas
#  kable_styling(latex_options = c("striped", "scale_down"))


SampleSize = nrow(test_ANN)
MSE = sum(test_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_ANN$AbsError)/SampleSize
MAPE = (sum(test_ANN$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)


test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL

Finalresult_ES = merge(train_statistics, test_statistics, by ="statistics")

kable(Finalresult_ES, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Annual
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 6.3: Accuracy Metrics of Annual Rainfall using Exponential Smoothing

statistics	training_set	testing_set
MAE	89.7097800576399	129.444131352738
MAPE	8.4867087668793	11.683074250111
MSE	13398.2129214487	23178.8936147882
RMSE	115.750649766853	152.246161248119
SampleSize	114	30

Interpretation:

1.To make forecasts using exponential smoothing in R, we can fit a simple exponential smoothing predictive model using the "HoltWinters()" function in R. To use HoltWinters() for simple exponential smoothing, we need to set the parameter gamma=FALSE in the HoltWinters() function (the beta and gamma parameters are used for Holt's exponential smoothing, or Holt-Winters exponential smoothing

2.By default, HoltWinters() just makes forecasts for the same time period covered by our original time series. In this case, our original time series included rainfall for All India from 1871-1986, so the forecasts are also for 1871-1986.

3.The output of HoltWinters() tells us that the estimated value of the alpha parameter is about 0.028 and gamma parametre is about 0.16. This is very close to zero, telling us that the forecasts are based on both recent and less recent observations (although somewhat more weight is placed on recent observations).

4.Here the forecasts for 1987-2016 are plotted as a blue line, the 80% prediction interval as an sky blue shaded area, and the 95% prediction interval as a grey shaded area.

5.From graph, The plot shows the original time series in black, and the forecasts as a blue line. The time series of forecasts is much smoother than the time series of the original data here 6. Overall model accuraacy is 88.32% (MAPE = 11.68%) indecating not a very good fit of the model. This model also exhibit some over-fitting as training and test error are little bit different.

6.2.4.3 *Forecasts using ARIMA Model*

ARIMA Model:

ARIMA stands for Autoregressive Integrated Moving Average models. Univariate (single vector) ARIMA is a forecasting technique that projects the future values of a series based entirely on its own inertia. Its main application is in the area of short term forecasting requiring at least 40 historical data points. It works best when your data exhibits a stable or consistent pattern over time with a minimum amount of outliers. Sometimes called Box-Jenkins (after the original authors), ARIMA is usually superior to exponential smoothing techniques when the data is reasonably long and the correlation between past observations is stable. If the data is short or highly volatile, then some smoothing method may perform better. If you do not have at least 38 data points, you should consider some other method than ARIMA.

Basic Concepts:

The first step in applying ARIMA methodology is to check for stationarity. "Stationarity" implies that the series remains at a fairly constant level over time. If a trend exists, as in most economic or business applications, then your data is NOT stationary. The data should also show a constant variance in its fluctuations over time. This is easily seen with a series that is heavily seasonal and growing at a faster rate. In such a case, the ups and downs in the seasonality will become more dramatic over time. Without these stationarity conditions being met, many of the calculations associated with the process cannot be computed.

Differencing:

If a graphical plot of the data indicates nonstationarity, then you should "difference" the series. Differencing is an excellent way of transforming a nonstationary series to a stationary one. This is done by subtracting the observation in the current period from the previous one. If this transformation is done only once to a series, you say that the data has been "first differenced". This process essentially eliminates the trend if your series is growing at a fairly constant rate. If it is growing at an increasing rate, you can apply the same procedure and difference the data again. Your data would then be "second differenced".

Autocorrelations:

"Autocorrelations" are numerical values that indicate how a data series is related to itself over time. More precisely, it measures how strongly data values at a specified number of periods apart are correlated to each other over time. The number of periods apart is usually called the "lag". For example, an autocorrelation at lag 1 measures how values 1 period apart are correlated to one another throughout the series. An autocorrelation at lag 2 measures how the data two periods apart are correlated throughout the series. Autocorrelations may range from +1 to -1. A value close to +1 indicates a high positive correlation while a value close to -1 implies a high negative correlation. These measures are most often evaluated through graphical plots called "correlagrams". A correlagram plots the auto- correlation values for a given series at different lags. This is referred to as the "autocorrelation function" and is very important in the ARIMA method.

Let's start building our ARIMA models. For this, let's plot ACF and PACF plot to identify order of ARIMA models.

ACF and PACF Plot

After a time series has been stationarized by differencing, the next step in fitting an ARIMA model is to determine whether AR or MA terms are needed to correct any autocorrelation that remains in the differenced series.By looking at the autocorrelation function (ACF) and partial autocorrelation (PACF) plots of the differenced series, we can tentatively identify the numbers of AR and/or MA terms that are needed. ACF plot is merely a bar chart of the coefficients of correlation between a time series and lags of itself. The PACF plot is a plot of the partial correlation coefficients between the series and lags of itself.

In general, the "partial" correlation between two variables is the amount of correlation between them which is not explained by their mutual correlations with a specified set of other variables. For example, if we are regressing a variable Y on other variables X_1, X_2, and X_3, the partial correlation between Y and X_3 is the amount of correlation between Y and X_3 that is not explained by their common correlations with X_1 and X_2. This partial correlation can be computed as the square root of the reduction in variance that is achieved by adding X_3 to the regression of Y on X_1 and X_2.

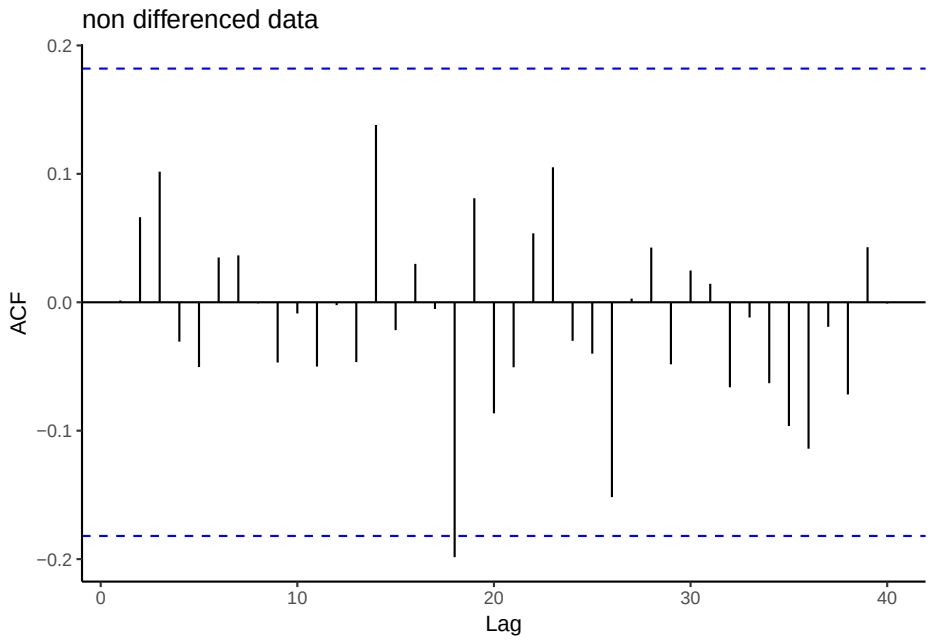


Figure 6.5: ACF plot of Annual Rainfall

A partial autocorrelation is the amount of correlation between a variable and a lag of itself that is not explained by correlations at all lower-order-lags. The autocorrelation of a time series Y at lag 1 is the coefficient of correlation between Y_t and Y_{t-1}, which is presumably also the correlation between $Yt-1$ and Y_{t-2}. But if Y_t is correlated with Y_{t-1}, and Y_{t-1} is equally correlated with Y_{t-2}, then we should also expect to find correlation between Y_t and Y_{t-2}. In fact, the amount of correlation we should expect at lag 2 is precisely the square of the lag-1 correlation. Thus, the correlation at lag 1 "propagates" to lag 2 and presumably to higher-order lags. The partial autocorrelation at lag 2 is therefore the difference between the actual correlation at lag 2 and the expected correlation due to the propagation of correlation at lag 1.

```
ggAcf(train.ts[,18],main="non differenced data",lag=40)
```

```
ggPacf(train.ts[,18],main="non differenced data",lag=40)
```

** Interpretation:**

From ACF and PACF plot, 1. The partial autocorrelations at all lags can be computed by fitting a succession of autoregressive models with increasing numbers of lags. 2. In particular, the partial

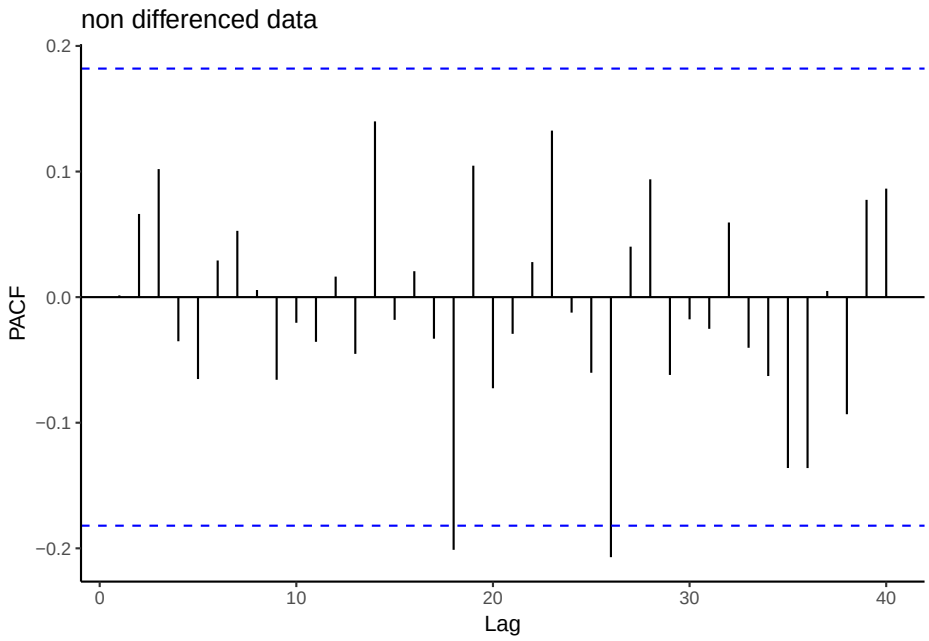


Figure 6.6: PACF plot of Annual Rainfall

autocorrelation at lag k is equal to the estimated AR(k) coefficient in an autoregressive model with k terms–i.e., a multiple regression model in which Y is regressed on LAG(Y,1), LAG(Y,2), etc., up to LAG(Y,k). 3. Thus, by mere inspection of the PACF you can determine how many AR terms you need to use to explain the autocorrelation pattern in a time series: if the partial autocorrelation is significant at lag k and not significant at any higher order lags–i.e., if the PACF "cuts off" at lag k–then this suggests that you should try fitting an autoregressive model of order k.

Short note on AR model:

Autoregrssive model:

The notation $AR(p)$ refers to the autoregressive model of order p. The $AR(p)$ model is written as

$$X_t = c + \sum_{i}^{p} [m_X]_{t-1} + \epsilon_t$$

where m1,m2,….mp are parameters, c is a constant, and the random variable ϵ_t is white noise.

Some constraints are necessary on the values of the parameters so that the model remains stationary.

For example, processes in the $AR(1)$ model with $|m_1| = 1$ are not stationary.

Fitting AR Model

Let's fit AR model with order 3. For this, we are using `arima` function.

```
options("scipen"=100, "digits"=2)


# fitting the model
ar3 <- arima(train.ts[,18], order=c(3, 0, 0))
ar3
```

```
##
## Call:
## arima(x = train.ts[, 18], order = c(3, 0, 0))
##
## Coefficients:
##          ar1    ar2    ar3  intercept
##       -0.005  0.066  0.103       1089
## s.e.   0.092  0.092  0.093         11
##
## sigma^2 estimated as 10495:  log likelihood = -702,  aic = 1413
```

```
# calculate train accuracy
train_forecast = as.data.frame(fitted(ar3))
train_ANN = train[,c("YEAR", "ANN")]
#train_ANN = train_ANN[3:116, ]
train_ANN = cbind(train_ANN, train_forecast)
setnames(train_ANN, "x", "Forecast")


train_ANN$Error = (train_ANN$ANN - train_ANN$Forecast)^2
train_ANN$AbsError = abs(train_ANN$ANN - train_ANN$Forecast)
train_ANN$APE = abs(train_ANN$ANN - train_ANN$Forecast)/train_ANN$ANN
```

```
SampleSize = nrow(train_ANN)
MSE = sum(train_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_ANN$AbsError)/SampleSize
MAPE = (sum(train_ANN$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

train_statistics <- data.frame(cbind(statistics, Value))
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
rownames(train_statistics) <-  NULL

# forecasting for test data
ar3_forecast <- forecast(ar3,h = 30)

# plotting the forecasted series
plot(ar3_forecast)
```

```
# calculate test accuracy
Forecast = as.data.frame(ar3_forecast$mean)
test_ANN = test[,c("YEAR", "ANN")]
test_ANN = cbind(test_ANN, Forecast)
test_ANN = data.table(test_ANN)
setnames(test_ANN, 'x', 'Forecast')

test_ANN$Error = (test_ANN$ANN - test_ANN$Forecast)^2
```

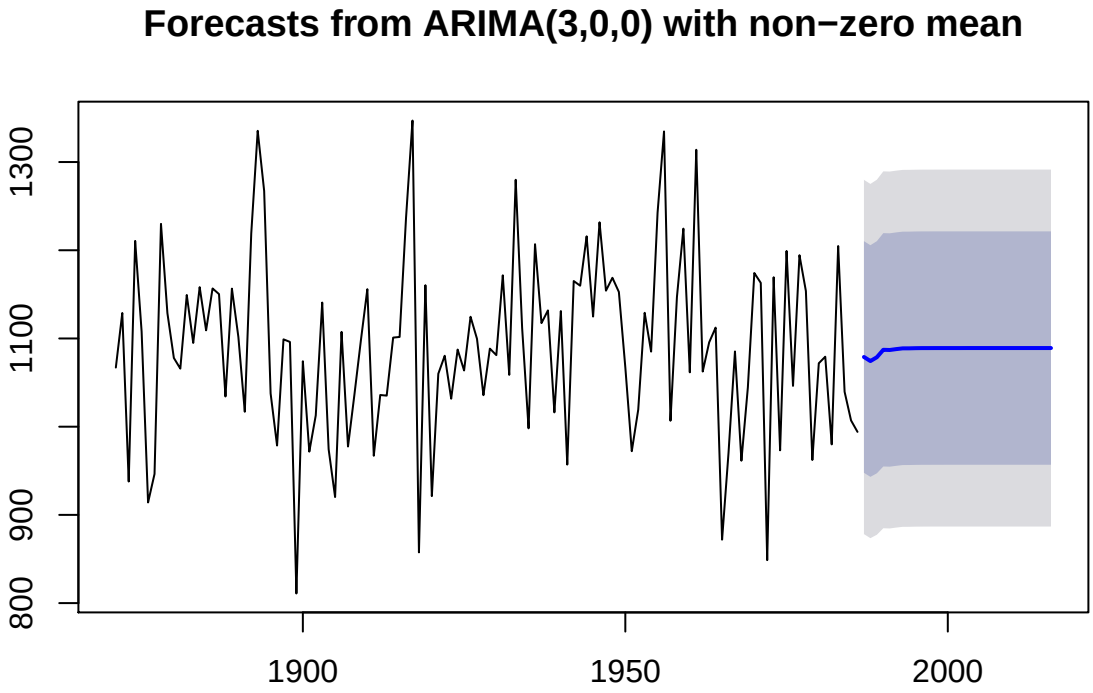


Figure 6.7: Forecasted value using AR model for Annual Rainfall

```
test_ANN$AbsError = abs(test_ANN$ANN - test_ANN$Forecast)
test_ANN$APE = abs(test_ANN$ANN - test_ANN$Forecast)/test_ANN$ANN
#kable(test_ANN, "latex",longtable=T,booktabs=T,caption="ARIMA Model of forecasting")
#  kable_styling(latex_options = c("striped", "scale_down"))

SampleSize = nrow(test_ANN)
MSE = sum(test_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_ANN$AbsError)/SampleSize
MAPE = (sum(test_ANN$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL

# summarising the model results
Finalresult_ar3 = merge(train_statistics, test_statistics, by ="statistics")

kable(Finalresult_ar3, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Annua
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 6.4: Accuracy Metrics of Annual Rainfall using AR Model

statistics	training_set	testing_set
MAE	79.5272908037679	74.678392225741

MAPE	7.45444051321105	7.19660077964034
MSE	10494.6574654656	8474.68080698905
RMSE	102.443435443495	92.05802956282
SampleSize	116	30

Interpretation:

1. Overall model accuraacy is 92.81% (MAPE = 7.19%) indecating good fit of the model. This model also does not exhibit over-fitting as training and test error are almost same.

6.2.5 *Forecasts using Neural Network Modals*

Reading some forecasting papers we saw that many technicians use Artificial Neural Networks. We realized that deepening this powerful method with our knowledge would take more time than we had. We decided to choose two known statistics methods, which still gave good results, to be compared later with a simple Neural Networks. This tool has proven to be more difficult than we thought, and the results have not been as good as expected.

Artificial neural networks are forecasting methods that are based on simple mathematical models of the brain. They allow complex nonlinear relationships between the response variable and its predictors.

```
options("scipen"=100, "digits"=2)


# fitting the model
NN <- nnetar(train.ts[,18])
NN
```

```
## Series: train.ts[, 18]
## Model:  NNAR(1,1)
## Call:   nnetar(y = train.ts[, 18])
##
## Average of 20 networks, each of which is
```

```
## a 1-1-1 network with 4 weights
## options were - linear output units
##
## sigma^2 estimated as 10578
```

```
# calculate train accuracy
train_forecast = as.data.frame(NN$fitted)
train_ANN = train[,c("YEAR", "ANN")]
#train_ANN = train_ANN[2:116, ]
train_ANN = cbind(train_ANN, train_forecast)
setnames(train_ANN, "x", "Forecast")

train_ANN = train_ANN[2:116,]
train_ANN$Error = (train_ANN$ANN - train_ANN$Forecast)^2
train_ANN$AbsError = abs(train_ANN$ANN - train_ANN$Forecast)
train_ANN$APE = abs(train_ANN$ANN - train_ANN$Forecast)/train_ANN$ANN

SampleSize = nrow(train_ANN)
MSE = sum(train_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_ANN$AbsError)/SampleSize
MAPE = (sum(train_ANN$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

train_statistics <- data.frame(cbind(statistics, Value))
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
rownames(train_statistics) <-  NULL
```

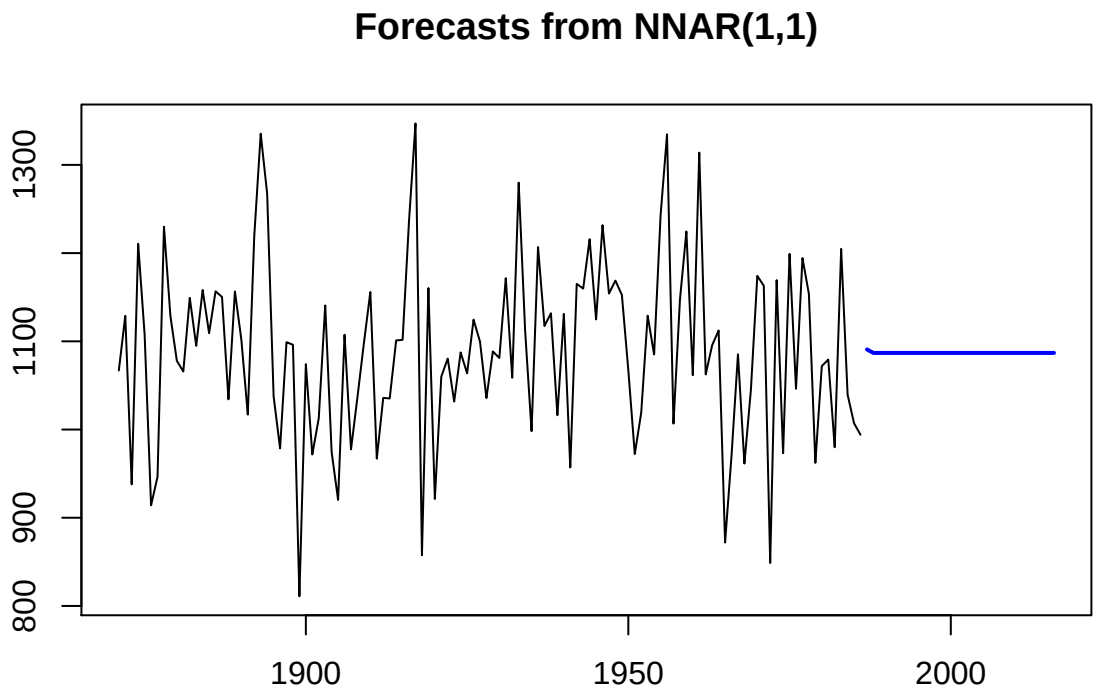


Figure 6.8: Forecasted value using Neural network for Annual Rainfall

```
# forecasting for test data
NN_forecast <- forecast(NN,h = 30)


# plotting the forecasted series
plot(NN_forecast)
```

```
# calculate test accuracy
Forecast = as.data.frame(NN_forecast$mean)
test_ANN = test[,c("YEAR", "ANN")]
test_ANN = cbind(test_ANN, Forecast)
test_ANN = data.table(test_ANN)
setnames(test_ANN, 'x', 'Forecast')
```

```
test_ANN$Error = (test_ANN$ANN - test_ANN$Forecast)^2
test_ANN$AbsError = abs(test_ANN$ANN - test_ANN$Forecast)
test_ANN$APE = abs(test_ANN$ANN - test_ANN$Forecast)/test_ANN$ANN
#kable(test_ANN, "latex",longtable=T,booktabs=T,caption="Neural Network Modals of fore
#  kable_styling(latex_options = c("striped", "scale_down"))

SampleSize = nrow(test_ANN)
MSE = sum(test_ANN$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_ANN$AbsError)/SampleSize
MAPE = (sum(test_ANN$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL

# summarising the model results
Finalresult_NN = merge(train_statistics, test_statistics, by ="statistics")

kable(Finalresult_NN, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Annual
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 6.5: Accuracy Metrics of Annual Rainfall using Neural Network Modals

statistics	training_set	testing_set

MAE	80.5158907749423	74.7213201172777
MAPE	7.54911278360722	7.19697474650496
MSE	10578.1985435779	8465.16244415081
RMSE	102.85036968129	92.0063174143537
SampleSize	115	30

Interpretation:

1. Overall model accuraacy is 92.80% (MAPE = 7.20%) indecating good fit of the model. This model also does not exhibit over-fitting as training and test error are almost same.

7 FORECASTING MONSOON RAINFALL

Let's start building our time series models. At first, we will plot historical trend of monsoon rainfall using ggplot2 package.

Plotting Time Series data:

```
rainfall%>%ggplot(aes(YEAR,JJAS))+geom_line()+
  geom_point(alpha = 0.5, color = palette_light()[[1]], shape=20,size=2) +
   labs(title = "JJAS Rainfall in India", x = "YEAR", y = "JJAS Rainfall",
       subtitle = "data from 1871-2016") +
    theme_tq()
```

Combo-chart for Annual Rainfall:

Let's look at some complex visulization for annual rainfall:

```
#Calculating Mean for JJAS
JJAS_mean = mean(rainfall$JJAS)

#Calculating percentage departure From Mean for JJAS
rainfall$Perc_departure_JJAS = ((rainfall$JJAS-JJAS_mean)/JJAS_mean)*100

#plotting combo-chart for ANN
rainfall1 <- subset(rainfall, Perc_departure_JJAS>= 0)
rainfall2 <- subset(rainfall, Perc_departure_JJAS < 0)

ggplot() +
    geom_bar(data = rainfall1, aes(x=YEAR, y=Perc_departure_JJAS),
```

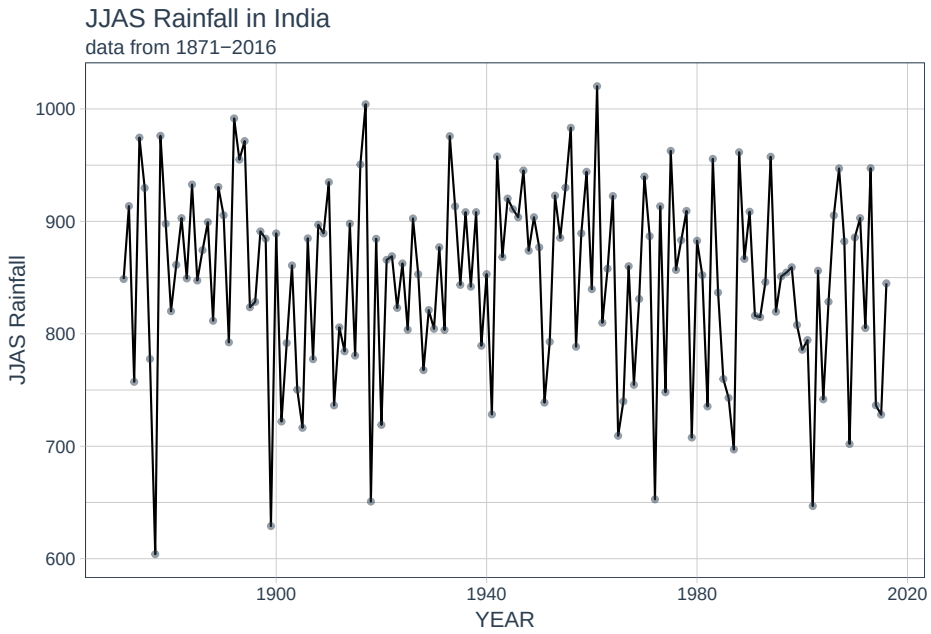


Figure 7.1: Historical trend of monsoon rainfall

```
            stat = "identity",colour = "lightgreen") +
    geom_bar(data = rainfall2, aes(x=YEAR, y=Perc_departure_JJAS),
            stat = "identity",colour = "pink")  +
  geom_ma(data = rainfall,aes(x= YEAR, y= Perc_departure_JJAS),
          ma_fun = SMA, n = 11, color = "blue",size = 1)
```

We can get similar chart using **Tableau** with more effective visulization as shown in below image:

7.1 Model Building

Let's start model building. We are following same process as mentioned in Annual Rainfall chapter.

7.1.1 *Splitting into train and test*

```
series.end <- floor(0.8*nrow(rainfall)) #select the first 80% of the data
train <- rainfall[1:series.end,] #assign the first 80% of the data to the train set
test <- rainfall[(series.end+1):nrow(rainfall),] #assign the most recent 20% to the tes
```

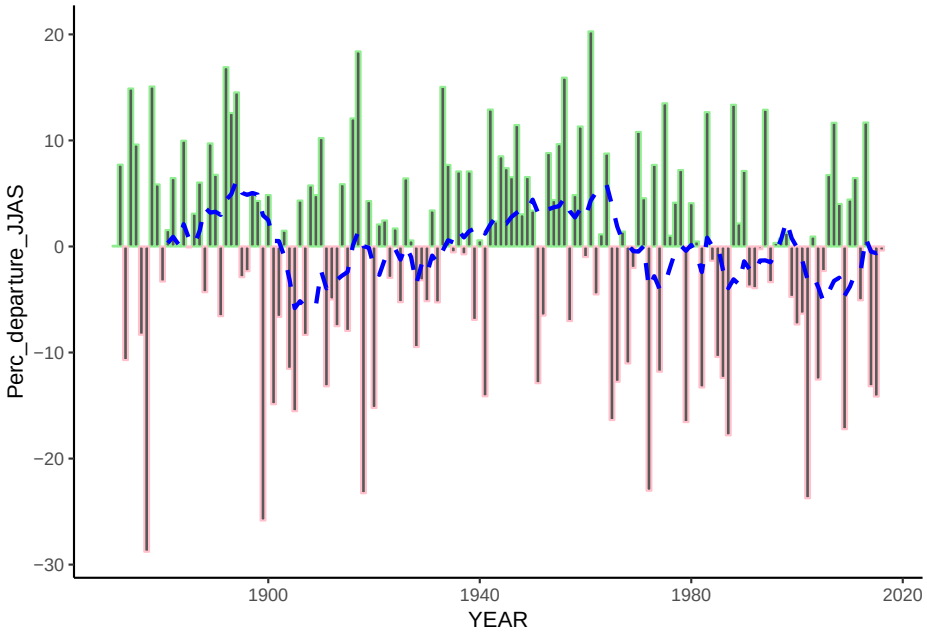


Figure 7.2: Combo chart of monsoon rainfall

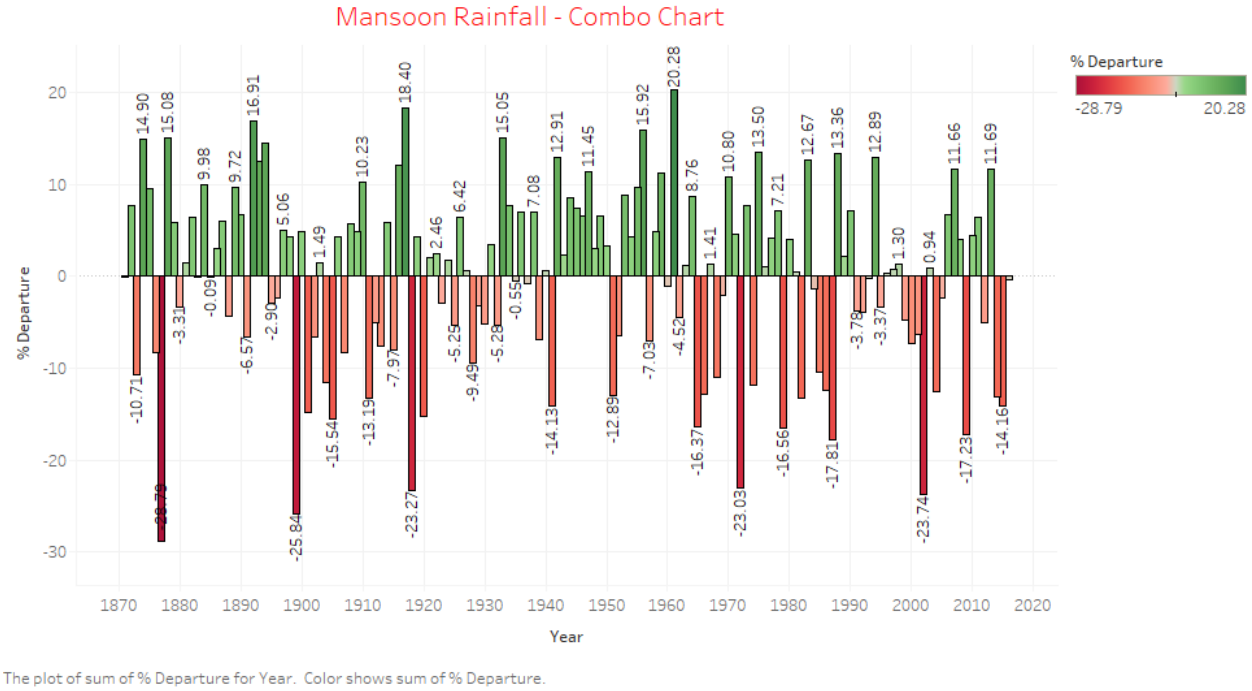


Figure 7.3: Mansoon rainfall-combochart

7.1.2 *Converting data into time series*

```
# Convert data frame into time serise object


train.ts <- ts(train, start = c(1871), end = c(1986))
test.ts <- ts(test, start = c(1987), end = c(2016))
```

7.1.3 *Test Stationary or Non-stationary*

The null hypothesis assumes that the series is non-stationary (mean does not stay constant). ADF procedure tests whether the change in Y can be explained by lagged value and a linear trend. If contribution of the lagged value to the change in Y is non-significant and there is a presence of a trend component, the series is non-stationary and null hypothesis will not be rejected.

```
pander(adf.test(train.ts[, 16], alternative = "stationary"))
```

Table 7.1: Augmented Dickey-Fuller Test: `train.ts[, 16]`

Test statistic	Lag order	P value	Alternative hypothesis
-4.59	4	0.01 * *	stationary

Interpretation:

The monsoon rainfall data is stationary(p-value = 0.01) by the ADF test hence there is no need of differencing the series. Since p-value is less then 5%, we can reject null hypothesis. This means that our series is stationary.

7.1.4 *Forecasts using Exponential Smoothing*

Exponential smoothing is a common method for making short-term forecasts in time series data.

7.1.4.1 Simple Exponential Smoothing

If we have a time series with constant level and no seasonality, we can use simple exponential smoothing for short term forecasts.

This method is a way of estimating the level at the current time point. Smoothing is controlled by the parameter `alpha` for the estimate of the level at the current time point. The value of alpha lies between 0 and 1. Values of alpha close to 0 mean that little weight is places on the most recent observations when making forecasts of future values.

In **R** we can use the `HoltWinters()` function. For simple exponential smoothing, we need to set the parameters `beta = FALSE` and `gamma = FALSE`.

```
options("scipen"=100, "digits"=2)
# Simple Exponetial Smoothing
SimpleExpSmooth <- HoltWinters(train.ts[, 16], beta=FALSE, gamma = FALSE)
SimpleExpSmooth
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = train.ts[, 16], beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.000066
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##    [,1]
## a  849
```

```
# Plotting the estimates
#plot(SimpleExpSmooth)

train_forecast = as.data.frame(SimpleExpSmooth$fitted)
train_JJAS = train[,c("YEAR", "JJAS")]
train_JJAS = train_JJAS[2:116, ]
train_JJAS = cbind(train_JJAS, train_forecast)
train_JJAS$xhat <- NULL
setnames(train_JJAS, "level", "Forecast")

train_JJAS$Error = (train_JJAS$JJAS - train_JJAS$Forecast)^2
train_JJAS$AbsError = abs(train_JJAS$JJAS - train_JJAS$Forecast)
train_JJAS$APE = abs(train_JJAS$JJAS - train_JJAS$Forecast)/train_JJAS$JJAS

SampleSize = nrow(train_JJAS)
MSE = sum(train_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_JJAS$AbsError)/SampleSize
MAPE = (sum(train_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

train_statistics <- data.frame(cbind(statistics, Value))
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
rownames(train_statistics) <-  NULL

library(forecast)
prediction_SES <- forecast(SimpleExpSmooth, h=30)
```

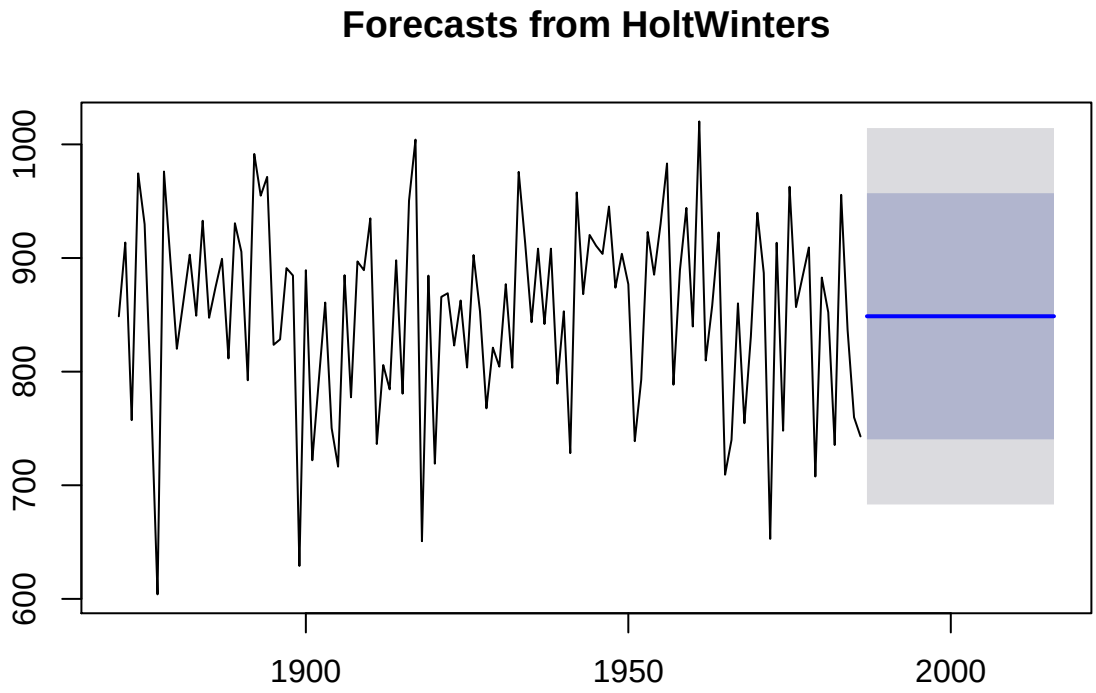


Figure 7.4: Forecasted value using simple expontional smoothing for Monsoon Rainfall

```
plot(prediction_SES)
```

```
Forecast = as.data.frame(prediction_SES$mean)


test_JJAS = test[,c("YEAR", "JJAS")]
test_JJAS = cbind(test_JJAS, Forecast)
test_JJAS = data.table(test_JJAS)
setnames(test_JJAS, 'x', 'Forecast')


test_JJAS$Error = (test_JJAS$JJAS - test_JJAS$Forecast)^2
test_JJAS$AbsError = abs(test_JJAS$JJAS - test_JJAS$Forecast)
test_JJAS$APE = abs(test_JJAS$JJAS - test_JJAS$Forecast)/test_JJAS$JJAS
```

```
#kable(test_JJAS, "latex",longtable=T,booktabs=T,caption="Simple Exponential Smoothing
#  kable_styling(latex_options = c("striped", "scale_down"))

SampleSize = nrow(test_JJAS)
MSE = sum(test_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_JJAS$AbsError)/SampleSize
MAPE = (sum(test_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL

Finalresult_SES = merge(train_statistics, test_statistics, by ="statistics")

kable(Finalresult_SES, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Monso
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 7.2: Accuracy Metrics of Monsoon Rainfall using Simple Exponential Smoothing

statistics	training_set	testing_set
MAE	68.5108200070982	62.2616730080843
MAPE	8.35245730210993	7.87199814840844
MSE	7090.07147747071	6485.75554935378
RMSE	84.2025621787764	80.5341886986749

SampleSize	115	30

Interpretation:

1.To make forecasts using simple exponential smoothing in R, we can fit a simple exponential smoothing predictive model using the "HoltWinters()" function in R. To use HoltWinters() for simple exponential smoothing, we need to set the parameters beta=FALSE and gamma=FALSE in the HoltWinters() function (the beta and gamma parameters are used for Holt's exponential smoothing, or Holt-Winters exponential smoothing

2.By default, HoltWinters() just makes forecasts for the same time period covered by our original time series. In this case, our original time series included rainfall for All India from 1871-1986, so the forecasts are also for 1871-1986.

3.The output of HoltWinters() tells us that the estimated value of the alpha parameter is about 0.018. This is very close to zero, telling us that the forecasts are based on both recent and less recent observations (although somewhat more weight is placed on recent observations).

4.Here the forecasts for 1987-2016 are plotted as a blue line, the 80% prediction interval as an sky blue shaded area, and the 95% prediction interval as a grey shaded area.

5.From graph, The plot shows the original time series in black, and the forecasts as a blue line. The time series of forecasts is much smoother than the time series of the original data here.

6. Overall model accuraacy is 92.13% (MAPE = 7.87%) indecating good fit of the model. This model also does not exhibit over-fitting as training and test error are almost same.

7.1.4.2 Exponential Smoothing

Let's look at performance of Exponential Smoothing Model.

```
options("scipen"=100, "digits"=2)
# Exponetial Smoothing
```

```r
ExpSmooth <- HoltWinters(train.ts[, 16], gamma = FALSE)
ExpSmooth
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = train.ts[, 16], gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.29
##  beta : 0.21
##  gamma: FALSE
##
## Coefficients:
##    [,1]
## a 799.1
## b  -8.7
```

```r
# Exponetial Smoothing
ExpSmooth <- ets(train.ts[, 16])
ExpSmooth
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = train.ts[, 16])
##
##   Smoothing parameters:
##     alpha = 0.0001
##
##   Initial states:
```

```
##     l = 852.0116
##
##   sigma:  0.099
##
##  AIC AICc  BIC
## 1585 1585 1593
```

```r
# Plotting the estimates
#plot(ExpSmooth)

train_forecast = as.data.frame(ExpSmooth$fitted)
train_JJAS = train[,c("YEAR", "JJAS")]
train_JJAS = train_JJAS[1:116, ]
train_JJAS = cbind(train_JJAS, train_forecast)
train_JJAS$xhat <- NULL
setnames(train_JJAS, "x", "Forecast")

train_JJAS$Error = (train_JJAS$JJAS - train_JJAS$Forecast)^2
train_JJAS$AbsError = abs(train_JJAS$JJAS - train_JJAS$Forecast)
train_JJAS$APE = abs(train_JJAS$JJAS - train_JJAS$Forecast)/train_JJAS$JJAS

SampleSize = nrow(train_JJAS)
MSE = sum(train_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_JJAS$AbsError)/SampleSize
MAPE = (sum(train_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

train_statistics <- data.frame(cbind(statistics, Value))
```

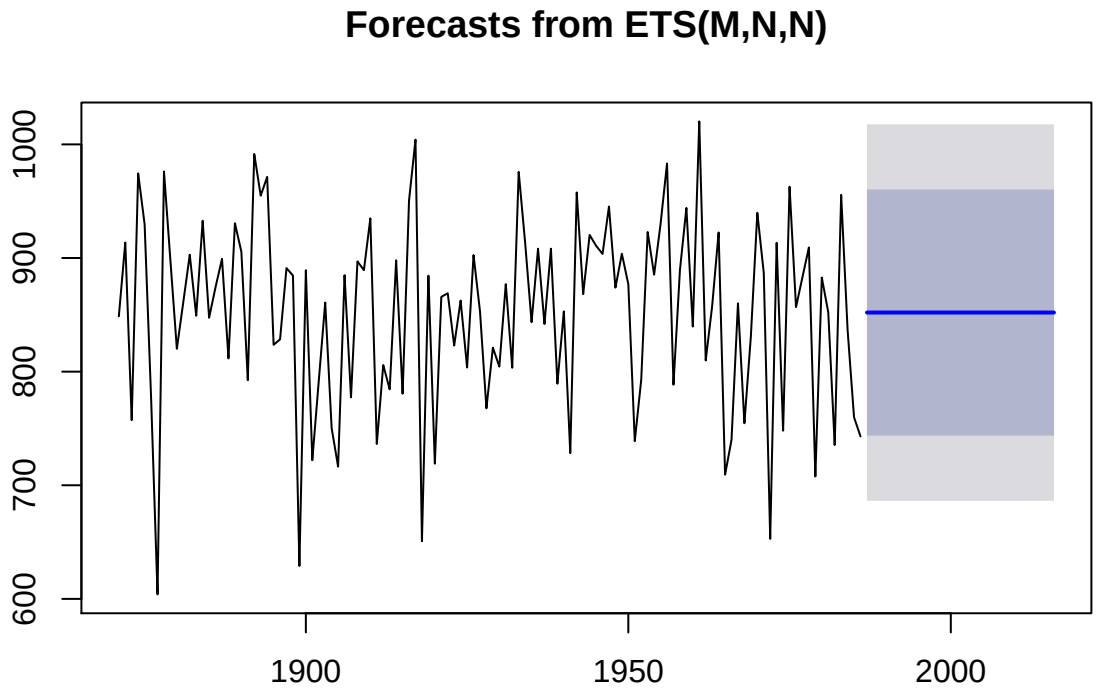


Figure 7.5: Forecasted value using expontional smoothing for Monsoon Rainfall

```
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
rownames(train_statistics) <-  NULL

library(forecast)
prediction_SES <- forecast(ExpSmooth, h=30)
plot(prediction_SES)
```

```
Forecast = as.data.frame(prediction_SES$mean)

test_JJAS = test[,c("YEAR", "JJAS")]
test_JJAS = cbind(test_JJAS, Forecast)
```

```
test_JJAS = data.table(test_JJAS)
setnames(test_JJAS, 'x', 'Forecast')

test_JJAS$Error = (test_JJAS$JJAS - test_JJAS$Forecast)^2
test_JJAS$AbsError = abs(test_JJAS$JJAS - test_JJAS$Forecast)
test_JJAS$APE = abs(test_JJAS$JJAS - test_JJAS$Forecast)/test_JJAS$JJAS
#kable(test_JJAS, "latex",longtable=T,booktabs=T,caption="Exponential Smoothing foreca
#  kable_styling(latex_options = c("striped", "scale_down"))

SampleSize = nrow(test_JJAS)
MSE = sum(test_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_JJAS$AbsError)/SampleSize
MAPE = (sum(test_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL

Finalresult_ES = merge(train_statistics, test_statistics, by ="statistics")

kable(Finalresult_ES, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Monsoo
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 7.3: Accuracy Metrics of Monsoon Rainfall using Exponential Smoothing

statistics	training_set	testing_set
MAE	67.4003605848218	62.5481730481038
MAPE	8.25143437177435	7.93621324399969
MSE	7018.43297432342	6597.5841291963
RMSE	83.776088320734	81.2255140285138
SampleSize	116	30

Interpretation:

1. Overall model accuraacy is 92.07% (MAPE = 7.93%) indecating not a very good fit of the model. This model also does not exhibit over-fitting as training and test error are almost same.

7.1.5 Forecast using ARIMA Model

Let's start building our ARIMA models. For this, let's plot ACF and PACF plot to identify order of ARIMA models.

ACF and PACF Plot

```
ggAcf(train.ts[,16],main="non differenced data",lag=40)
```

```
ggPacf(train.ts[,16],main="non differenced data",lag=40)
```

Fitting AR Model

Let's fit AR model with order 1. For this, we are using `arima` function.

```
options("scipen"=100, "digits"=4)
# fitting the model
ar3 <- arima(train.ts[,16], order=c(1, 0, 0))
```

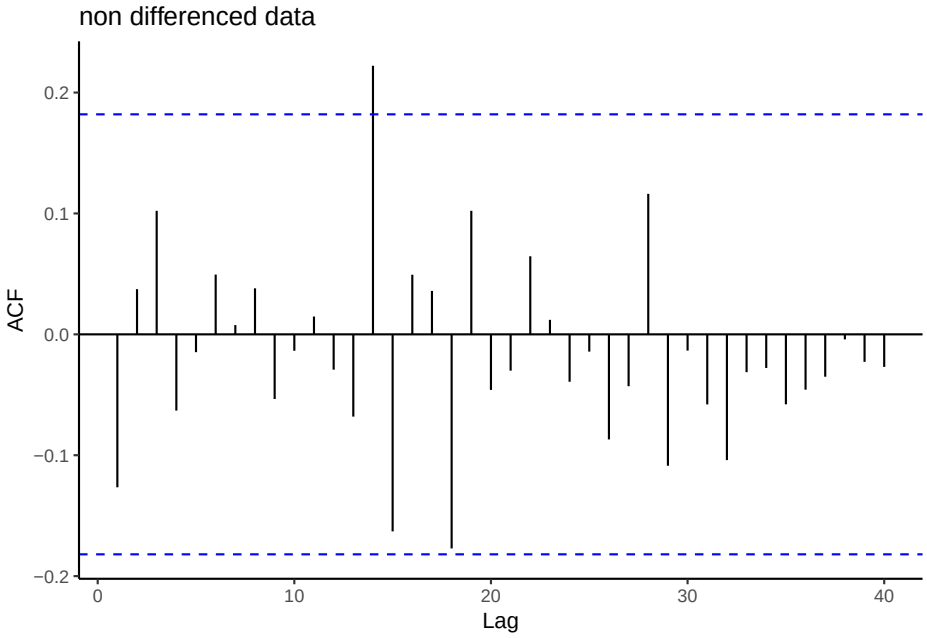


Figure 7.6: ACF plot of Monsoon Rainfall

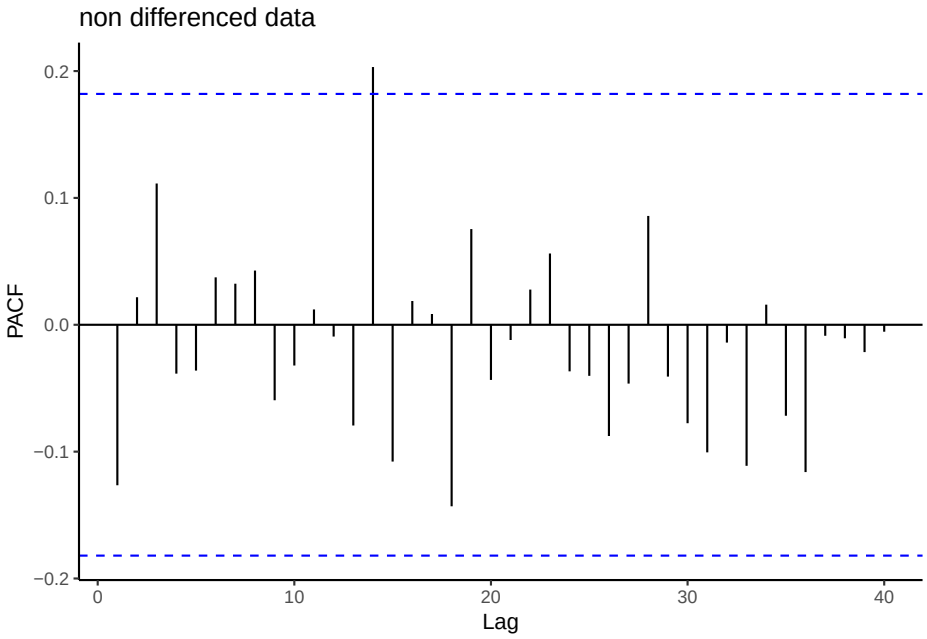


Figure 7.7: PACF plot of Monsoon Rainfall

```
ar3
```

```
##
## Call:
## arima(x = train.ts[, 16], order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       -0.127      852.1
## s.e.   0.092        6.9
##
## sigma^2 estimated as 6904:  log likelihood = -677,  aic = 1361
```

```
# calculate train accuracy
train_forecast = as.data.frame(fitted(ar3))
train_JJAS = train[,c("YEAR", "JJAS")]
#train_JJAS = train_JJAS[3:116, ]
train_JJAS = cbind(train_JJAS, train_forecast)
setnames(train_JJAS, "x", "Forecast")

train_JJAS$Error = (train_JJAS$JJAS - train_JJAS$Forecast)^2
train_JJAS$AbsError = abs(train_JJAS$JJAS - train_JJAS$Forecast)
train_JJAS$APE = abs(train_JJAS$JJAS - train_JJAS$Forecast)/train_JJAS$JJAS

SampleSize = nrow(train_JJAS)
MSE = sum(train_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_JJAS$AbsError)/SampleSize
MAPE = (sum(train_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
```

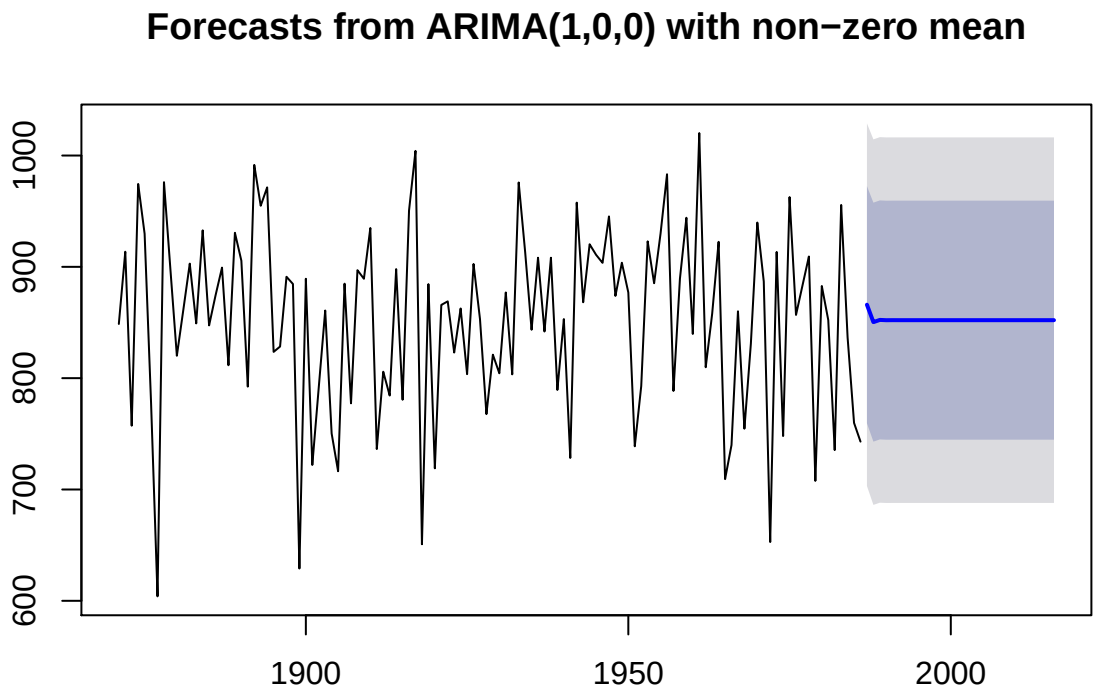


Figure 7.8: Forecasted value using AR model for Monsoon Rainfall

```
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)


train_statistics <- data.frame(cbind(statistics, Value))
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
rownames(train_statistics) <-  NULL


# forecasting for test data
ar3_forecast <- forecast(ar3,h = 30)


# plotting the forecasted series
plot(ar3_forecast)
```

```
# calculate test accuracy
Forecast = as.data.frame(ar3_forecast$mean)
test_JJAS = test[,c("YEAR", "JJAS")]
test_JJAS = cbind(test_JJAS, Forecast)
test_JJAS = data.table(test_JJAS)
setnames(test_JJAS, 'x', 'Forecast')

test_JJAS$Error = (test_JJAS$JJAS - test_JJAS$Forecast)^2
test_JJAS$AbsError = abs(test_JJAS$JJAS - test_JJAS$Forecast)
test_JJAS$APE = abs(test_JJAS$JJAS - test_JJAS$Forecast)/test_JJAS$JJAS
#kable(test_JJAS, "latex",longtable=T,booktabs=T,caption="ARIMA Model of forecasting")
#  kable_styling(latex_options = c("striped", "scale_down"))

SampleSize = nrow(test_JJAS)
MSE = sum(test_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_JJAS$AbsError)/SampleSize
MAPE = (sum(test_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL

# summarising the model results
Finalresult_ar3 = merge(train_statistics, test_statistics, by ="statistics")
```

```
kable(Finalresult_ar3, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Monsc
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 7.4: Accuracy Metrics of Monsoon Rainfall using AR Model

statistics	training_set	testing_set
MAE	66.0328500456671	63.0756403971191
MAPE	8.08789983094549	8.01026085336179
MSE	6903.75153148502	6763.79835873689
RMSE	83.0888171265244	82.2423148916474
SampleSize	116	30

Interpretation:

1. Overall model accuraacy is 91.99% (MAPE = 8.01%) indecating good fit of the model. This model also does not exhibit over-fitting as training and test error are almost same.

7.1.6 *Forecasts using Neural Network Modals*

Artificial neural networks are forecasting methods that are based on simple mathematical models of the brain. They allow complex nonlinear relationships between the response variable and its predictors.

Let's build our Artificial neural networks model.

```
options("scipen"=100, "digits"=2)
# fitting the model
NN <- nnetar(train.ts[,16], order=c(1, 0, 0))
NN
```

```
## Series: train.ts[, 16]
## Model:  NNAR(1,1)
```

```
## Call:   nnetar(y = train.ts[, 16], order = c(1, 0, 0))
##
## Average of 20 networks, each of which is
## a 1-1-1 network with 4 weights
## options were - linear output units
##
## sigma^2 estimated as 6875
```

```
# calculate train accuracy
train_forecast = as.data.frame(fitted(NN))
train_JJAS = train[,c("YEAR", "JJAS")]
train_JJAS = cbind(train_JJAS, train_forecast)
setnames(train_JJAS, "x", "Forecast")

train_JJAS = train_JJAS[2:116, ]
train_JJAS$Error = (train_JJAS$JJAS - train_JJAS$Forecast)^2
train_JJAS$AbsError = abs(train_JJAS$JJAS - train_JJAS$Forecast)
train_JJAS$APE = abs(train_JJAS$JJAS - train_JJAS$Forecast)/train_JJAS$JJAS

SampleSize = nrow(train_JJAS)
MSE = sum(train_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(train_JJAS$AbsError)/SampleSize
MAPE = (sum(train_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

train_statistics <- data.frame(cbind(statistics, Value))
train_statistics$training_set <- train_statistics$V2
train_statistics$V2 <- NULL
```

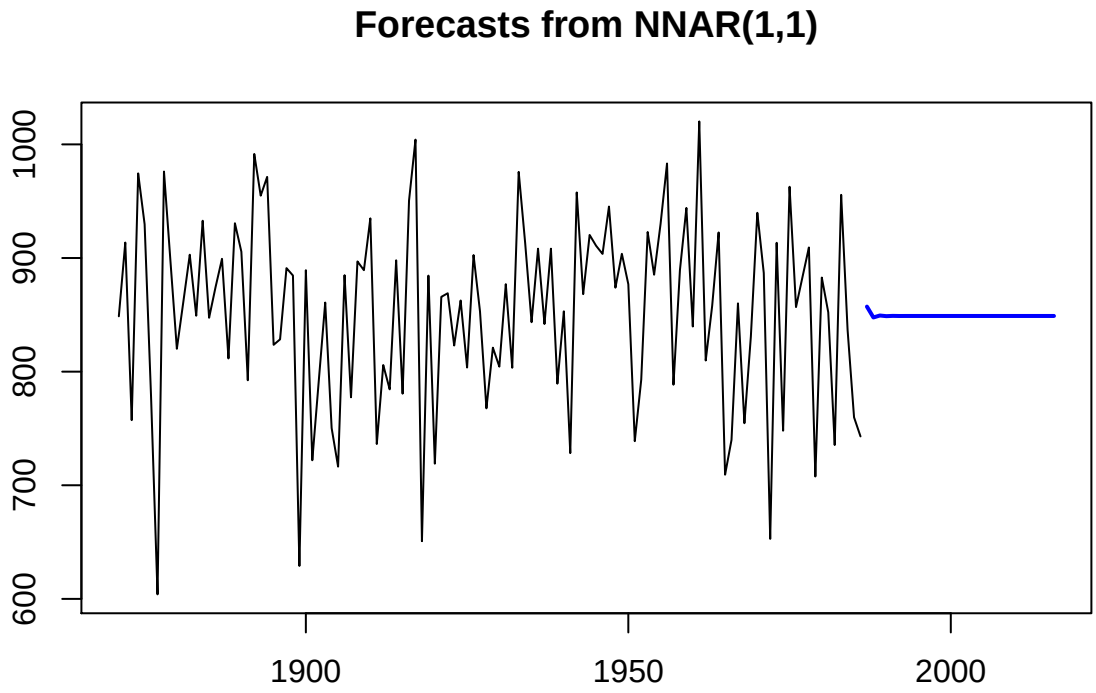


Figure 7.9: Forecasted value using Neural network for Monsoon Rainfall

```
rownames(train_statistics) <-  NULL


# forecasting for test data
NN_forecast <- forecast(NN,h = 30)


# plotting the forecasted series
plot(NN_forecast)
```

```
# calculate test accuracy
Forecast = as.data.frame(NN_forecast$mean)
test_JJAS = test[,c("YEAR", "JJAS")]
test_JJAS = cbind(test_JJAS, Forecast)
```

```
test_JJAS = data.table(test_JJAS)
setnames(test_JJAS, 'x', 'Forecast')

test_JJAS$Error = (test_JJAS$JJAS - test_JJAS$Forecast)^2
test_JJAS$AbsError = abs(test_JJAS$JJAS - test_JJAS$Forecast)
test_JJAS$APE = abs(test_JJAS$JJAS - test_JJAS$Forecast)/test_JJAS$JJAS
#kable(test_JJAS, "latex",longtable=T,booktabs=T,caption="Neural Network Modals of for
#  kable_styling(latex_options = c("striped", "scale_down"))

SampleSize = nrow(test_JJAS)
MSE = sum(test_JJAS$Error)/SampleSize
RMSE = sqrt(MSE)
MAE = sum(test_JJAS$AbsError)/SampleSize
MAPE = (sum(test_JJAS$APE)/SampleSize)*100

statistics = c("SampleSize", "MSE", "RMSE", "MAE", "MAPE")
Value = rbind(SampleSize, MSE, RMSE, MAE, MAPE)

test_statistics <- data.frame(cbind(statistics, Value))
test_statistics$testing_set <- test_statistics$V2
test_statistics$V2 <- NULL
rownames(test_statistics) <-  NULL

# summarising the model results
Finalresult_NN = merge(train_statistics, test_statistics, by ="statistics")

kable(Finalresult_NN, "latex",longtable=T,booktabs=T,caption="Accuracy Metrics of Monsoo
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 7.5: Accuracy Metrics of Monsoon Rainfall using Neural Network Modals

statistics	training_set	testing_set
MAE	66.4283895248038	62.5920244115478
MAPE	8.13487283953853	7.91988254007018
MSE	6875.26918115977	6590.16283081785
RMSE	82.9172429664649	81.1798178786936
SampleSize	115	30

Interpretation:

1. Overall model accuraacy is 92.o5% (MAPE = 7.95%) indecating good fit of the model. This model also does not exhibit over-fitting as training and test error are almost same.

8 CONCLUSION

Comparison of time series model

From the chapter 6 and 7,we can conclued the following:

1. For the collect data of annual rainfall data, Simple Exponential Smoothing Model is the best model among outperform the other 3 model(Exponential Smoothing Model, ARIMA Model and Neural Network Modals) based on the MAPE statistics.
2. For the collect data of monsoon rainfall data, Simple Exponential Smoothing Model is the best model among outperform the other 3 model(Exponential Smoothing Model, ARIMA Model and Neural Network Modals) based on the MAPE statistics.

   There is further scope of improvement can be possible by trying other time series model(Long Short Term Memory networks(LSTM), Time Bely Neural Network(TBLN), etc.) or by introducing independent variables like temperature, pollution, wind speed etc. in ARIMAX OR Regression Models.

Disriptive statistics using monsoon southwest(JJAS) and annuall rainfall(ANN)

1. Skewness value of southwest monsoon(JJAS) is -0.51 which is lies between -0.5 to -1, then we conclude that the distribution is moderately skewed and also distribution is highly skewed because given value is less than -1.
2. Skewness value of annual rainfall (ANN) is -0.02, which is lies between -0.5 to 0.5, and then we conclude that the distribution is approximately skewed, also distribution is highly skewedbecause give value is less than -1.
3. Skew.2SE value of southwest monsoon (JJAS)is -1.26, and then we conclude that skewness is insignificantly different from zero. 4.Kurtosis value of southwest monsoon (JJAS) is -0.12, which is less than -1,and then we conclude that distributions is light tails and is called

a platykurtic distribution. 5.Kurtosis value of annual rainfall (ANN) is 0.06, which is lies between 0 to 1,and then we conclude that distributions has heavier tails and is called a leptokurtic distribution. 6.Kurtosis value of annual rainfall (ANN) is 0.06, which is lies between 0 to 1,and then we conclude that distributions has heavier tails and is called a leptokurtic distribution. 7.The value of the shapiro-wilk test of annual rainfall (ANN) is 0.99 which is greater than 0.05, then we say that the data is normal.

4. Most of the variables(month/weather period) are not following normal distribution as shown in shapio-wilk test(`normtest.W` and `normtest.p`) except JUN, AUG, SEP,OND and JJASual variables.(Since p-value > 0.05). 8.The above diagram(Heatmap) shows that, the month june, jully, august, september shows the highest rainfall are shown as well as other month sows the lowest rainfall. 9.Then we have to interpret that those four months i.e southewest monsoon(JJAS) are very important for forecastin rainfall data.
5. Monsoon variable look like following normal distribution which is also confirmed in descriptive statistics section. There is one outlier year(1877) has been observed in which lowest rainfall has been seen.

9 SESSION INFO

```
devtools::session_info()
```

```
## - Session info ---------------------------------------------------------
##  setting  value
##  version  R version 3.5.3 (2019-03-11)
##  os       Windows 7 x64 SP 1
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United States.1252
##  ctype    English_United States.1252
##  tz       Asia/Calcutta
##  date     2019-04-07
##
## - Packages -------------------------------------------------------------
##  package              * version    date       lib source
##  assertthat             0.2.1      2019-03-21 [1] CRAN (R 3.5.3)
##  backports              1.1.3      2018-12-14 [1] CRAN (R 3.5.2)
##  bit                    1.1-14     2018-05-29 [1] CRAN (R 3.5.2)
##  bit64                  0.9-7      2017-05-08 [1] CRAN (R 3.5.2)
##  blob                   1.1.1      2018-03-25 [1] CRAN (R 3.5.3)
##  bookdown               0.9        2018-12-21 [1] CRAN (R 3.5.3)
##  boot                   1.3-20     2017-08-06 [1] CRAN (R 3.5.3)
##  broom                  0.5.1      2018-12-05 [1] CRAN (R 3.5.3)
##  callr                  3.2.0      2019-03-15 [1] CRAN (R 3.5.3)
```

```
##  cellranger            1.1.0      2016-07-27 [1] CRAN (R 3.5.3)
##  chron                 2.3-53     2018-09-09 [1] CRAN (R 3.5.3)
##  class                 7.3-15     2019-01-01 [1] CRAN (R 3.5.3)
##  cli                   1.1.0      2019-03-19 [1] CRAN (R 3.5.3)
##  colorspace            1.4-1      2019-03-18 [1] CRAN (R 3.5.3)
##  crayon                1.3.4      2017-09-16 [1] CRAN (R 3.5.3)
##  curl                  3.3        2019-01-10 [1] CRAN (R 3.5.3)
##  data.table          * 1.12.0     2019-01-13 [1] CRAN (R 3.5.3)
##  DBI                   1.0.0      2018-05-02 [1] CRAN (R 3.5.3)
##  desc                  1.2.0      2018-05-01 [1] CRAN (R 3.5.3)
##  devtools              2.0.1      2018-10-26 [1] CRAN (R 3.5.3)
##  digest                0.6.18     2018-10-10 [1] CRAN (R 3.5.3)
##  dplyr               * 0.8.0.1    2019-02-15 [1] CRAN (R 3.5.3)
##  evaluate              0.13       2019-02-12 [1] CRAN (R 3.5.3)
##  forcats             * 0.4.0      2019-02-17 [1] CRAN (R 3.5.3)
##  forecast            * 8.5        2019-01-18 [1] CRAN (R 3.5.3)
##  foreign               0.8-71     2018-07-20 [1] CRAN (R 3.5.3)
##  fracdiff              1.4-2      2012-12-02 [1] CRAN (R 3.5.3)
##  fs                    1.2.7      2019-03-19 [1] CRAN (R 3.5.3)
##  generics              0.0.2      2018-11-29 [1] CRAN (R 3.5.3)
##  ggplot2             * 3.1.0      2018-10-25 [1] CRAN (R 3.5.3)
##  ggthemes            * 4.1.0      2019-02-19 [1] CRAN (R 3.5.3)
##  glue                  1.3.1      2019-03-12 [1] CRAN (R 3.5.3)
##  gower                 0.2.0      2019-03-07 [1] CRAN (R 3.5.3)
##  gsubfn              * 0.7        2018-03-16 [1] CRAN (R 3.5.3)
##  gtable                0.3.0      2019-03-25 [1] CRAN (R 3.5.3)
##  haven                 2.1.0      2019-02-19 [1] CRAN (R 3.5.3)
##  highr                 0.8        2019-03-20 [1] CRAN (R 3.5.3)
##  hms                   0.4.2      2018-03-10 [1] CRAN (R 3.5.3)
##  htmltools             0.3.6      2017-04-28 [1] CRAN (R 3.5.3)
```

```
##  httr                    1.4.0      2018-12-11 [1] CRAN (R 3.5.3)
##  ipred                   0.9-8      2018-11-05 [1] CRAN (R 3.5.3)
##  jsonlite                1.6        2018-12-07 [1] CRAN (R 3.5.3)
##  kableExtra            * 1.1.0      2019-03-16 [1] CRAN (R 3.5.3)
##  knitr                   1.22       2019-03-08 [1] CRAN (R 3.5.3)
##  labeling                0.3        2014-08-23 [1] CRAN (R 3.5.2)
##  lattice                 0.20-38    2018-11-04 [1] CRAN (R 3.5.3)
##  lava                    1.6.5      2019-02-12 [1] CRAN (R 3.5.3)
##  lazyeval                0.2.2      2019-03-15 [1] CRAN (R 3.5.3)
##  lmtest                  0.9-36     2018-04-04 [1] CRAN (R 3.5.3)
##  lubridate             * 1.7.4      2018-04-11 [1] CRAN (R 3.5.3)
##  magrittr              * 1.5        2014-11-22 [1] CRAN (R 3.5.3)
##  MASS                    7.3-51.1   2018-11-01 [1] CRAN (R 3.5.3)
##  Matrix                  1.2-15     2018-11-01 [1] CRAN (R 3.5.3)
##  memoise                 1.1.0      2017-04-21 [1] CRAN (R 3.5.3)
##  Metrics               * 0.1.4      2018-07-09 [1] CRAN (R 3.5.3)
##  mnormt                  1.5-5      2016-10-15 [1] CRAN (R 3.5.2)
##  modelr                  0.1.4      2019-02-18 [1] CRAN (R 3.5.3)
##  munsell                 0.5.0      2018-06-12 [1] CRAN (R 3.5.3)
##  nlme                    3.1-137    2018-04-07 [1] CRAN (R 3.5.3)
##  nnet                    7.3-12     2016-02-02 [1] CRAN (R 3.5.3)
##  nortest               * 1.0-4      2015-07-30 [1] CRAN (R 3.5.2)
##  pander                * 0.6.3      2018-11-06 [1] CRAN (R 3.5.3)
##  pastecs               * 1.3.21     2018-03-15 [1] CRAN (R 3.5.3)
##  PerformanceAnalytics * 1.5.2       2018-03-02 [1] CRAN (R 3.5.3)
##  pillar                  1.3.1      2018-12-15 [1] CRAN (R 3.5.3)
##  pkgbuild                1.0.3      2019-03-20 [1] CRAN (R 3.5.3)
##  pkgconfig               2.0.2      2018-08-16 [1] CRAN (R 3.5.3)
##  pkgload                 1.0.2      2018-10-29 [1] CRAN (R 3.5.3)
##  plyr                  * 1.8.4      2016-06-08 [1] CRAN (R 3.5.3)
```

```
##  prettyunits           1.0.2      2015-07-13 [1] CRAN (R 3.5.3)
##  processx              3.3.0      2019-03-10 [1] CRAN (R 3.5.3)
##  prodlim               2018.04.18 2018-04-18 [1] CRAN (R 3.5.3)
##  proto               * 1.0.0      2016-10-29 [1] CRAN (R 3.5.3)
##  ps                    1.3.0      2018-12-21 [1] CRAN (R 3.5.3)
##  psych               * 1.8.12     2019-01-12 [1] CRAN (R 3.5.3)
##  purrr               * 0.3.2      2019-03-15 [1] CRAN (R 3.5.3)
##  quadprog              1.5-5      2013-04-17 [1] CRAN (R 3.5.2)
##  Quandl                2.9.1      2018-08-14 [1] CRAN (R 3.5.3)
##  quantmod            * 0.4-14     2019-03-24 [1] CRAN (R 3.5.3)
##  R6                    2.4.0      2019-02-14 [1] CRAN (R 3.5.3)
##  Rcpp                  1.0.1      2019-03-17 [1] CRAN (R 3.5.3)
##  readr               * 1.3.1      2018-12-21 [1] CRAN (R 3.5.3)
##  readxl                1.3.1      2019-03-13 [1] CRAN (R 3.5.3)
##  recipes             * 0.1.5      2019-03-21 [1] CRAN (R 3.5.3)
##  remotes               2.0.2      2018-10-30 [1] CRAN (R 3.5.3)
##  reshape             * 0.8.8      2018-10-23 [1] CRAN (R 3.5.3)
##  rlang                 0.3.3      2019-03-29 [1] CRAN (R 3.5.3)
##  rmarkdown             1.12       2019-03-14 [1] CRAN (R 3.5.3)
##  rpart                 4.1-13     2018-02-23 [1] CRAN (R 3.5.3)
##  rprojroot             1.3-2      2018-01-03 [1] CRAN (R 3.5.3)
##  RSQLite             * 2.1.1      2018-05-06 [1] CRAN (R 3.5.3)
##  rstudioapi            0.10       2019-03-19 [1] CRAN (R 3.5.3)
##  rvest                 0.3.2      2016-06-17 [1] CRAN (R 3.5.3)
##  scales              * 1.0.0      2018-08-09 [1] CRAN (R 3.5.3)
##  sessioninfo           1.1.1      2018-11-05 [1] CRAN (R 3.5.3)
##  sqldf               * 0.4-11     2017-06-28 [1] CRAN (R 3.5.3)
##  stringi               1.4.3      2019-03-12 [1] CRAN (R 3.5.3)
##  stringr             * 1.4.0      2019-02-10 [1] CRAN (R 3.5.3)
##  survival              2.43-3     2018-11-26 [1] CRAN (R 3.5.3)
```

```
##  tibble              * 2.1.1      2019-03-16 [1] CRAN (R 3.5.3)
##  tidyquant           * 0.5.5      2018-05-09 [1] CRAN (R 3.5.3)
##  tidyr               * 0.8.3      2019-03-01 [1] CRAN (R 3.5.3)
##  tidyselect            0.2.5      2018-10-11 [1] CRAN (R 3.5.3)
##  tidyverse           * 1.2.1      2017-11-14 [1] CRAN (R 3.5.3)
##  timeDate            * 3043.102   2018-02-21 [1] CRAN (R 3.5.2)
##  timeSeries          * 3042.102   2017-11-17 [1] CRAN (R 3.5.3)
##  tseries             * 0.10-46    2018-11-19 [1] CRAN (R 3.5.3)
##  TTR                 * 0.23-4     2018-09-20 [1] CRAN (R 3.5.3)
##  urca                  1.3-0      2016-09-06 [1] CRAN (R 3.5.3)
##  usethis               1.4.0      2018-08-14 [1] CRAN (R 3.5.3)
##  viridisLite           0.3.0      2018-02-01 [1] CRAN (R 3.5.3)
##  webshot               0.5.1      2018-09-28 [1] CRAN (R 3.5.3)
##  withr                 2.1.2      2018-03-15 [1] CRAN (R 3.5.3)
##  xfun                  0.6        2019-04-02 [1] CRAN (R 3.5.3)
##  xml2                  1.2.0      2018-01-24 [1] CRAN (R 3.5.3)
##  xts                 * 0.11-2     2018-11-05 [1] CRAN (R 3.5.3)
##  yaml                  2.2.0      2018-07-25 [1] CRAN (R 3.5.2)
##  zoo                 * 1.8-5      2019-03-21 [1] CRAN (R 3.5.3)
##
## [1] C:/Program Files/R/R-3.5.3/library
```

10 APPENDEX

data.table

Data manipulation operations such as subset, group, update, join etc., are all inherently related. Keeping these related operations together allows for: 1. concise and consistent syntax irrespective of the set of operations you would like to perform to achieve your end goal. 2. performing analysis fluidly without the cognitive burden of having to map each operation to a particular function from a potentially huge set of functions available before performing the analysis. 3. Automatically optimizing operations internally, and very effectively, by knowing precisely the data required for each operation, leading to very fast and memory efficient code. Briefly, if you are interested in reducing programming and compute time tremendously, then this package is for you. The philosophy that data. Table adheres to makes this possible. Our goal is to illustrate it through this series of vignettes.

dplyr

Describe what the dplyr package in R is used for. Apply common dplyr functions to manipulate data in R. Employ the 'pipe' operator to link together a sequence of functions. Employ the 'mutate' function to apply other chosen functions to existing columns and create new columns of data.

plyr

plyr is a set of tools for a common set of problems: you need to split up a big data structure into homogeneous pieces, apply a function to each piece and then combine all the results back together. For example, you might want to: • fit the same model each patient subsets of a data frame • quickly calculate summary statistics for each group • perform group-wise transformations like scaling or standardizing

It's already possible to do this with base R functions (like split and the apply family of functions), but plyr makes it all a bit easier with: • totally consistent names, arguments and outputs • convenient parallelization through the for each package • input from and output to data. frames, matrices and lists

• progress bars to keep track of long running operations • built-in error recovery, and informative error messages • labels that are maintained across all transformations

Considerable effort has been put into making plyr fast and memory efficient, and in many cases plyr is as fast as, or faster than, the built-in equivalents. A detailed introduction to plyr has been published in JSS: “The Split-Apply-Combine Strategy for Data Analysis”

Sqldf

sqldf is an R package for running SQL statements on R data frames, optimized for convenience. The user simply specifies an SQL statement in R using data frame names in place of table names and a database with appropriate table layouts/schema is automatically created, the data frames are automatically loaded into the database, the specified SQL statement is performed, the result is read back into R and the database is deleted all automatically behind the scenes making the database’s existence transparent to the user who only specifies the SQL statement. Surprisingly this can at times be even faster than the corresponding pure R calculation (although the purpose of the project is convenience and not speed). This link suggests that for aggregations over highly granular columns that sqldf is faster than another alternative tried. sqldf is free software published under the GNU General Public License that can be downloaded from CRAN.

Printing nested tables in R – bridging between the {reshape} and {tables} packages This post shows how to print a prettier nested pivot table, created using the {reshape} package (similar to what you would get with Microsoft Excel), so you could print it either in the R terminal or as a LaTeX table. This task is done by bridging between the cast_df object produced by the {reshape} package, […]

pastecs v1.3.21

Regularization, decomposition and analysis of space-time series. The pastecs R package is a PNEC-Art4 and IFREMER (Benoit Belief Benoit.Beliaeff@ifremer.fr) initiative to bring PASSTEC 2000 functionalities to R.

Name Description

AutoD2:- AutoD2, CrossD2 or CenterD2 analysis of a multiple time-series

deccensus:-Time decomposition using the CENSUS II method

disjoin:-Complete disjoined coded data (binary coding)

marbio:-Several zooplankton taxa measured across a transect

decloess:-Time series decomposition by the LOESS method

decmedian:-Time series decomposition using a running median

marphy:-Physico-chemical records at the same stations as for marbio

reglin:-Regulation of a series using a linear interpolation

extract:-Extract a subset of the original dataset

regconst:-Regulate a series using the constant value method

first:-Get the first element of a vector

daystoyears:-Convert time units from “days” to “years” or back

stat.desc:-Descriptive statistics on a data frame or time series

buysbal:-Buys-Ballot table

regspline:-Regulation of a time series using splines

stat.pen:-Pennington statistics on a data frame or time series

vario:-Compute and plot a semi-variorum

match.tol:-Determine matching observation with a tolerance in time-scale

regul:-Regulation of one or several time series using various methods

pennington:-Calculate Pennington statistics

stat.slide:-Sliding statistics

trend.test:-Test if an increasing or decreasing trend exists in a time series

tseries:-Convert a ‘regul’ or a ‘tsd’ object into a time series

tsd:-Decomposition of one or several regular time series using various methods

decdiff:-Time series decomposition using differences (trend elimination)

decevf:-Time series decomposition using eigenvector filtering (EVF)

gleissberg.table:-Table of probabilities according to the Gleissberg distribution

is.tseries:-Is this object a time series?

pgleissberg:-Gleissberg distribution probability

regarea:-Regulate a series using the area method

regul.adj:-Adjust regulation parameters

regul.screen:-Test various regulation parameters

abund:-Sort variables by abundance

bnr:-A data frame of 163 benthic species measured across a transect

disto:-Compute and plot a distogram

escouf:-Choose variables using the Escoufier’s equivalent vectors method

last:-Get the last element of a vector

local.trend:-Calculate local trends using cumsum

releve:-A data frame of six phytoplankton taxa followed in time at one station

specs:-Collect parameters (“specifications”) from one object to use them in another analysis

turnogram:-Calculate and plot a turnogram for a regular time series

turnpoints:-Analyze turning points (peaks or pits)

GetUnitText:-Format a nice time units for labels in graphs

decaverage:-Time series decomposition using a moving average

decreg:-Time series decomposition using a regression model

psych

Procedures for Psychological, Psychometric, and Personality Research A general purpose toolbox for personality, psychometric theory and experimental psychology. Functions are primarily for multivariate analysis and scale construction using factor analysis, principal component analysis, cluster analysis and reliability analysis, although others provide basic descriptive statistics. Item Response Theory is done using factor analysis of tetra choric and poly choric correlations. Functions for analyzing data at multiple levels include within and between group statistics, including correlations and factor analysis. Functions for simulating and testing particular item and test structures are included. Several functions serve as a useful front end for structural equation modeling. Graphical displays of path diagrams, factor analysis and structural equation models are created using basic graphics. Some of the functions are written to support a book on psychometric theory as well as publications in personality research

tidy quant

Bringing financial analysis to the ‘tidyverse’. The ‘tidyquant’ package provides a convenient wrapper to various ‘xts’, ‘zoo’, ‘quantmod’, ‘TTR’ and ‘PerformanceAnalytics’ package functions and returns the objects in the tidy ‘tibble’ format. The main advantage is being able to use quantitative functions with the ‘tidyverse’ functions including ‘purrr’, ‘dplyr’, ‘tidyr’, ‘ggplot2’, ‘lubridate’, etc. See the ‘tidyquant’ website for more information, documentation and examples. nortest: Tests for Normality Five omnibus tests for testing the composite hypothesis of normality.

tseries

Time series analysis and computational finance.

ggthemes Extra Themes, Scales and Geoms for 'ggplot2'

Some extra themes, geoms, and scales for 'ggplot2'. Provides 'ggplot2' themes and scales that replicate the look of plots by Edward Tufte, Stephen Few, 'Fivethirtyeight', 'The Economist', 'Stata', 'Excel', and 'The Wall Street Journal', among others. Provides 'geoms' for Tufte's box plot and range frame.

ggplot2

Create Elegant Data Visualisations Using the Grammar of Graphics A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

scales

Scale Functions for Visualization

Graphical scales map data to aesthetics, and provide methods for automatically determining breaks and labels for axes and legends.

Forecast

Forecasting Functions for Time Series and Linear Models

Methods and tools for displaying and analysing univariate time series forecasts including exponential smoothing via state space models and automatic ARIMA modelling.

Metrics

Evaluation Metrics for Machine Learning

An implementation of evaluation metrics in R that are commonly used in supervised machine learning. It implements metrics for regression, time series, binary classification, classification, and information retrieval problems. It has zero dependencies and a consistent, simple interface for all functions.

kableExtra

Construct Complex Table with 'kable' and Pipe Syntax

Build complex HTML or 'LaTeX' tables using 'kable()' from 'knitr' and the piping syntax from 'magrittr'. Function 'kable()' is a light weight table generator coming from 'knitr'. This package simplifies the way to manipulate the HTML or 'LaTeX' codes generated by 'kable()' and allows users to construct complex tables and customize styles using a readable syntax.

pander

An R 'Pandoc' Writer

Contains some functions catching all messages, 'stdout' and other useful information while evaluating R code and other helpers to return user specified text elements (like: header, paragraph, table, image, lists etc.) in 'pandoc' markdown or several type of R objects similarly automatically transformed to markdown format. Also capable of exporting/converting (the resulting) complex 'pandoc' documents to e.g. HTML, 'PDF', 'docx' or 'odt'. This latter reporting feature is supported in brew syntax or with a custom reference class with a smarty caching 'backend'.

tidyverse

Easily Install and Load the 'Tidy verse'

The 'tidyverse' is a set of packages that work in harmony because they share common data representations and 'API' design. This package is designed to make it easy to install and load multiple 'tidyverse' packages in a single step. Learn more about the 'tidyverse' at.

lubridate

Make Dealing with Dates a Little Easier

Functions to work with date-times and time-spans: fast and user friendly parsing of date-time data, extraction and updating of components of a date-time (years, months, days, hours, minutes, and seconds), algebraic manipulation on date-time and time-span objects. The 'lubridate' package has a consistent and memorable syntax that makes working with dates easy and fun. Parts of the 'CCTZ' source code, released under the Apache 2.0 License, are included in this package. See.

timeSeries

Rmetrics - Financial Time Series Objects

Provides a class and various tools for financial time series. This includes basic functions such as scaling and sorting, sub setting, mathematical operations and statistical functions.

magrittr

A Forward-Pipe Operator for R

Provides a mechanism for chaining commands with a new forward-pipe operator, %>%. This operator will forward a value, or the result of an expression, into the next function call/expression. There is flexible support for the type of right-hand side expressions. For more information, see package vignette. To quote Rene Magritte, “Cenci nest pas un pipe.”

recipes

Preprocessing Tools to Create Design Matrices

An extensible framework to create and preprocess design matrices. Recipes consist of one or more data manipulation and analysis “steps”. Statistical parameters for the steps can be estimated from an initial data set and then applied to other data sets. The resulting design matrices can then be used as inputs into statistical or machine learning models.

REFERENCES

1. People.duke.edu. (2019). Identifying the orders of AR and MA terms in an ARIMA model. [online] Available at: https://people.duke.edu/~rnau/411arim3.htm [Accessed 6 Apr. 2019].

2. En.wikipedia.org. (2019). Monsoon of South Asia. [online] Available at: https://en.wikipedia.org/wiki/Monsoon_of_South_Asia [Accessed 6 Apr. 2019].

3. George E. P. Box,Gwilym M. Jenkins,Gregory C. Reinsel,Greta M. Ljung (2015). Time Series Analysis: Forecasting and Control (Wiley Series in Probability and Statistics) 5th Edition.

4. Tsai, E. (2019). RPubs - Time Series Analysis in R. [online] Rpubs.com. Available at: http://www.rpubs.com/edwardtsai/timeseriesanalysis [Accessed 6 Apr. 2019].

5. T.M.J.A. Cooray. Applied Time Series : Analysis and Forecasting

6. B.L.Agrawal (2017) Programed Statistics 3th Edition.