

**PES UNIVERSITY EC Campus**

**Electronic City, Hosur Road,**

**Bangalore-100**

A Project Report on

**EDA ON LANDSLIDES IN AMERICA**

**(2007-2016)**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

For the Academic year 2020

by

**Sohan Naidu**

**(PES2UG19CS398)**

**Haritha Madana**

**(PES2UG19CS141)**

**Serena A. Gomez**

**(PES2UG19CS372)**

**Netra Shaligram**

**(PES2UG19CS253)**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Table of Contents

[CHAPTER-1 6](#_Toc58694878)

[INTRODUCTION 6](#_Toc58694879)

[System requirements and configuration 6](#_Toc58694880)

[R installation procedure 7](#_Toc58694881)

[The Dataset 11](#_Toc58694882)

[Libraries Used. 11](#_Toc58694883)

[Explain what type of analysis is done in the work. 11](#_Toc58694884)

[CHAPTER-2 12](#_Toc58694885)

[Data Structure 12](#_Toc58694886)

[Vectors 12](#_Toc58694887)

[Lists 13](#_Toc58694888)

[Data frames 14](#_Toc58694889)

[Matrices 15](#_Toc58694890)

[Arrays 17](#_Toc58694891)

[Factors 18](#_Toc58694892)

[CHAPTER-3 20](#_Toc58694893)

[Operations on Dates 20](#_Toc58694894)

[Reading the date column from the excel file and converting character to date: 20](#_Toc58694895)

[Making dates shorter for demonstrating operations and displaying what day of the week it is on that date: 20](#_Toc58694896)

[Displaying the month of the dates: 20](#_Toc58694897)

[Displaying which quarter of the year the dates fall in: 20](#_Toc58694898)

[CHAPTER-4 21](#_Toc58694899)

[Conditional statements on Vector, data frame, array, matrix 21](#_Toc58694900)

[Conditional statement on a vector: 21](#_Toc58694901)

[Conditional statement on data frames: 21](#_Toc58694902)

[Conditional statement on an array : 22](#_Toc58694903)

[Conditional statement on a matrix 22](#_Toc58694904)

[CHAPTER 5 23](#_Toc58694905)

[Loops 23](#_Toc58694906)

[Loop on a vector 23](#_Toc58694907)

[Loop on a dataframe 23](#_Toc58694908)

[Loop on a matrix 24](#_Toc58694909)

[Loop on an array: 24](#_Toc58694910)

[CHAPTER - 6 25](#_Toc58694911)

[Strings 25](#_Toc58694912)

[CHAPTER – 7 27](#_Toc58694913)

[Function to show recursion: 27](#_Toc58694914)

[CHAPTER – 8 28](#_Toc58694915)

[Usage of functions : 28](#_Toc58694916)

[All: 28](#_Toc58694917)

[Any: 28](#_Toc58694918)

[Apply: 28](#_Toc58694919)

[Which : 28](#_Toc58694920)

[Match : 28](#_Toc58694921)

[Order: 28](#_Toc58694922)

[Rank: 28](#_Toc58694923)

[Sub 29](#_Toc58694924)

[Aggregate: 29](#_Toc58694925)

[CHAPTER-9 31](#_Toc58694926)

[Data Visualization using basic R plots 31](#_Toc58694927)

[Data Preparation 31](#_Toc58694928)

[What size of Landslides are most likely to occur? 32](#_Toc58694929)

[Occurrence of different types of triggers of landslides: 33](#_Toc58694930)

[Number of fatalities and injuries that occurred vs the distance travelled by the landslide: 34](#_Toc58694931)

[Fatalities in US over the years 35](#_Toc58694932)

[CHAPTER-10 36](#_Toc58694933)

[Data visualization using ggplot 36](#_Toc58694934)

[Distribution of landslide per type and country: 36](#_Toc58694935)

[Distribution of landslide type: 37](#_Toc58694936)

[Distribution of landslides per trigger per year 38](#_Toc58694937)

[To check if there is a seasonality of occurrence of landslides: 39](#_Toc58694938)

[To analyze the region of occurrences of various types of landslides and the number of fatalities they caused: 41](#_Toc58694939)

[CHAPTER-11 42](#_Toc58694940)

[Statistical Analysis of landslides in 2015 42](#_Toc58694941)

[Analysis of fatalities, injuries and distance traveled by landslides 42](#_Toc58694942)

[Finding levels of landslide type, landslide size and triggers: 42](#_Toc58694943)

[Statistical Analysis of landslides in USA 43](#_Toc58694944)

[Analysis of fatalities 43](#_Toc58694945)

[Analysis of injuries 44](#_Toc58694946)

[Analysis of distance traveled by landslide 44](#_Toc58694947)

[Finding levels of year, landslide type, landslide size and triggers: 45](#_Toc58694948)

[CHAPTER-12 46](#_Toc58694949)

[Probability Distribution Function: Normal Distribution 46](#_Toc58694950)

[Plotting Normal curves for population in the dataset. 46](#_Toc58694951)

[CHAPTER-13 48](#_Toc58694952)

[Hypothesis Testing 48](#_Toc58694953)

[CHAPTER-14 49](#_Toc58694954)

[CONCLUSION 49](#_Toc58694955)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

**LIST OF FIGURES**

[Figure 1: CRAN 7](#_Toc58694956)

[Figure 2: RStudio Download 7](#_Toc58694957)

[Figure 3: RStudio 8](#_Toc58694958)

[Figure 4: Install R packages 9](#_Toc58694959)

[Figure 5: Install R packages 10](#_Toc58694960)

[Figure 6: Install R packages 10](#_Toc58694961)

[Figure 7: Landslide Size Occurrences 32](#_Toc58694962)

[Figure 8: Occurrence of landslides due to different types of triggers 33](#_Toc58694963)

[Figure 9: Fatalities vs Distance traveled by landslide 34](#_Toc58694964)

[Figure 10: Injuries vs Distance traveled by landslide 34](#_Toc58694965)

[Figure 11: Fatalities in USA from 2007 - 2016 35](#_Toc58694966)

[Figure 12: Distribution of landslide per type and country 36](#_Toc58694967)

[Figure 13: Distribution of landslide type 37](#_Toc58694968)

[Figure 14: Distribution of landslides per trigger per year 38](#_Toc58694969)

[Figure 15: is there seasonality of occurrence of landslides? (heatmap) 39](#_Toc58694970)

[Figure 16: is there seasonality of occurrence of landslides? (boxplot) 40](#_Toc58694971)

[Figure 17: World map - landslide type and fatalities 41](#_Toc58694972)

[Figure 18: Fatalities in USA 43](#_Toc58694973)

[Figure 19: Injuries in USA 44](#_Toc58694974)

[Figure 20: Distance travelled by landslides in USA 45](#_Toc58694975)

[Figure 21: Normal Distribution 46](#_Toc58694976)

[Figure 22: Normal Distribution 47](#_Toc58694977)

[Figure 23: Normal Distribution 47](#_Toc58694978)

CHAPTER-1

# INTRODUCTION

R is an open-source programming language and software environment for statistical computing and graphics.

R is a GNU package and is available freely under the GNU General Public License. This means that R is available with source code, and you are free to use R, but you must adhere to the license.

R is a HLL because it shares many similarities to human languages. The R language is widely used among statisticians and data miners for developing statistical analysis and data analysis.

## System requirements and configuration

* An Intel-compatible platform running Windows 10 /8.1/8 /7 /Vista /XP /2000 Windows Server 2019 /2016 /2012 /2008 /2003
* At least 256 MB of RAM, a mouse, and enough disk space for recovered files, image files, etc.
* The administrative privileges are required to install and run R‑Studio utilities.
* A network connection for data recovering over network.

## R installation procedure

Step 1 – Install R

* Download the R installer from <https://cran.r-project.org/>

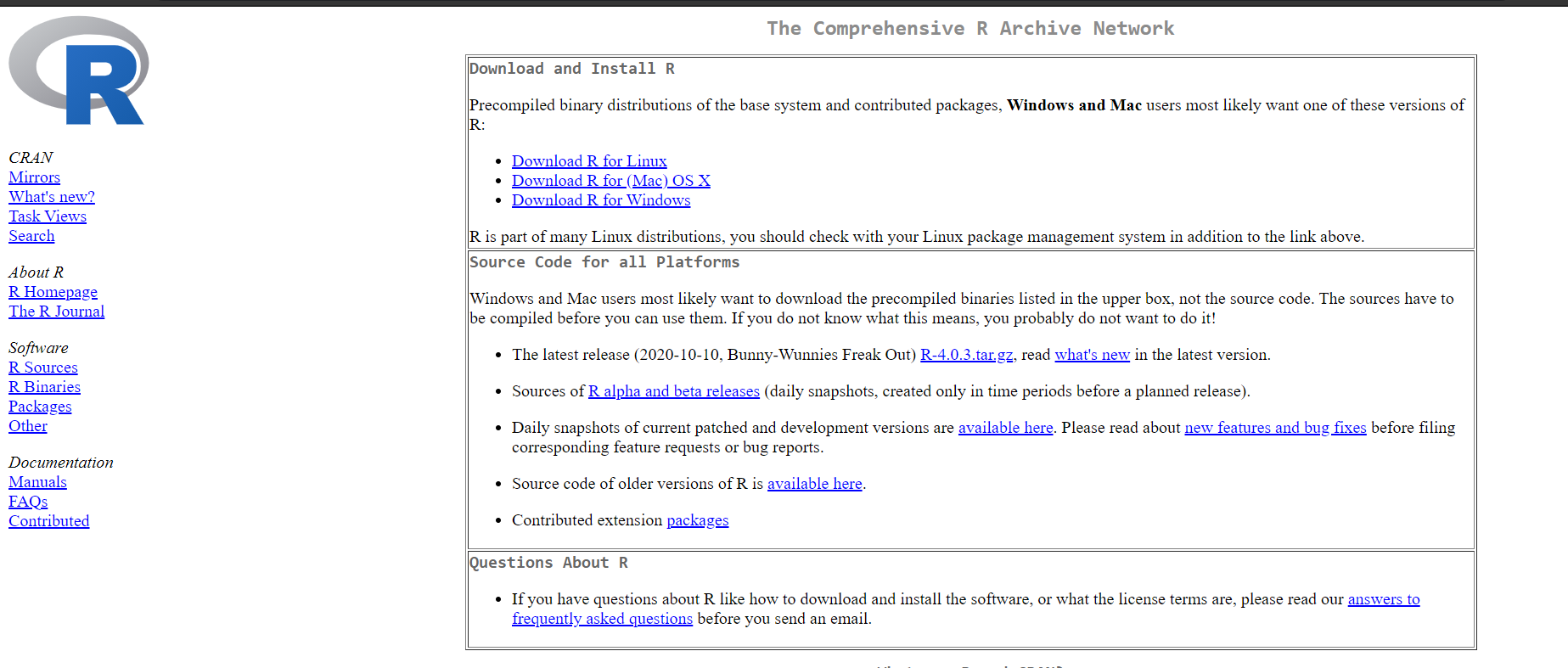


Figure 1: CRAN

* Run the installer. Default settings are fine. If you do not have admin rights on your laptop, then ask your local IT support.

Step 2 – Install RStudio

* Download RStudio: <https://rstudio.com/products/rstudio/download/>

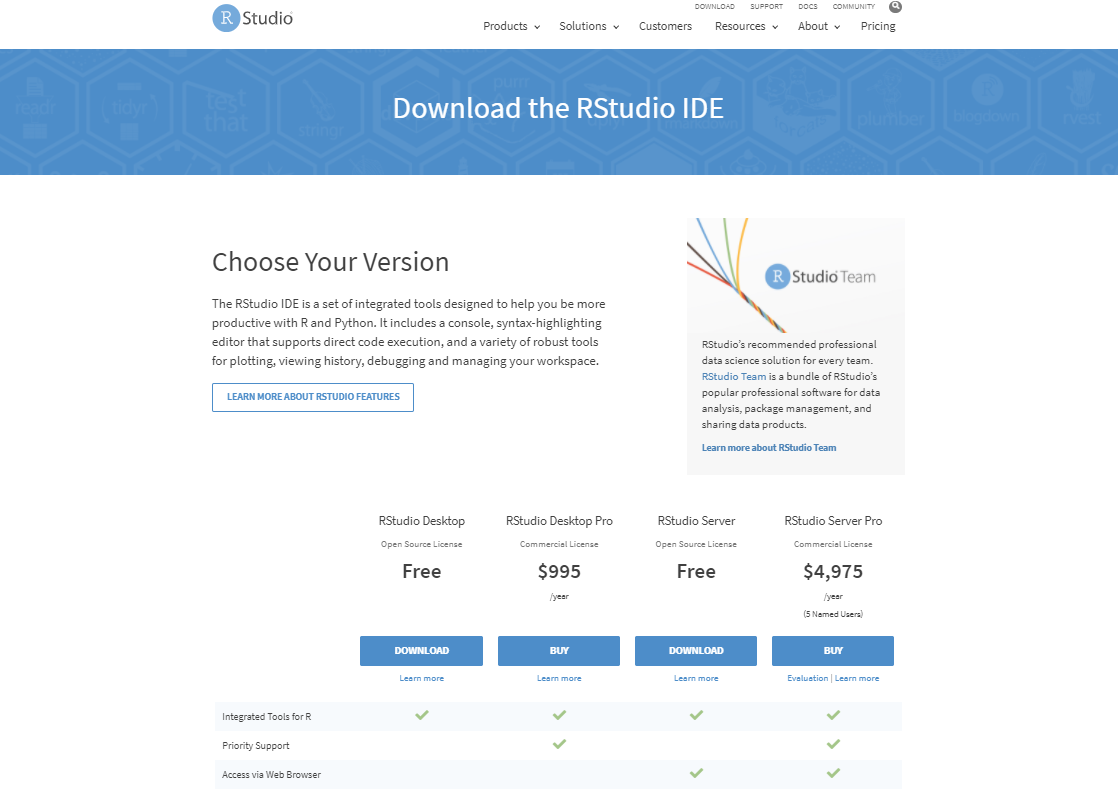


Figure 2: RStudio Download

* Once the installation of R has completed successfully (and not before), run the RStudio installer.
* If you do not have administrative rights on your laptop, step 2 may fail. Ask your IT Support or download a pre-built zip archive of RStudio which doesn’t need installing.
  + Download the appropriate archive for your system (Windows/Linux only – the Mac version can be installed into your personal “Applications” folder without admin rights).
  + Double clicking on the zip archive should automatically unpack it on most Windows machines.

Step 3 – Check that R and RStudio are working

* Open RStudio.
* In the left-hand window, by the ‘>’sign, type ‘4+5’(without the quotes) and hit enter. An output line reading ‘[1] 9’ should appear.
* This means that R and RStudio are working.
* If this is not successful, contact us or your local IT support for further advice

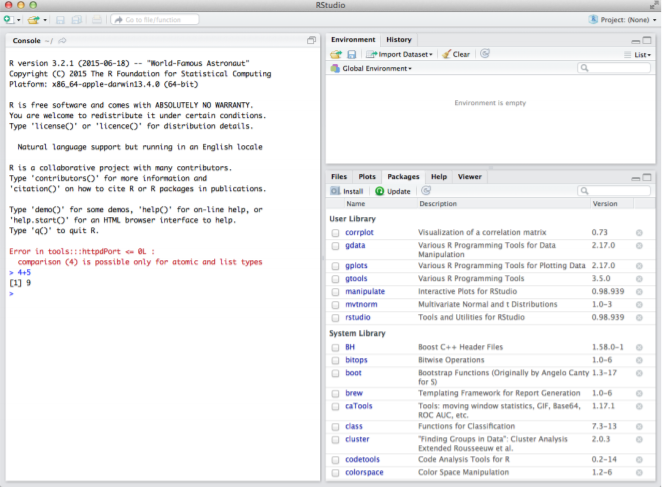


Figure 3: RStudio

Step 4 – Install R packages required for the workshop

* Click on the tab ‘ Packages’ then ‘Install’ as shown in Image 4. Or Tools -> Install packages.
* Install the following packages: mixOmics version 6.1.0, mvtnorm, RColorBrewer, corrplot, igraph (see Image 4). For apple mac users, if you are unable to install the mixOmics imported library rgl, you will need to install the XQuartz software first <https://www.xquartz.org/>
* Check that the packages are installed by typing ‘library(mixOmics)’ (without the quotes) in the prompt and press enter (see Image 5).
* Then type ‘sessionInfo()’ and check that mixOmics version 6.1.0 has been installed

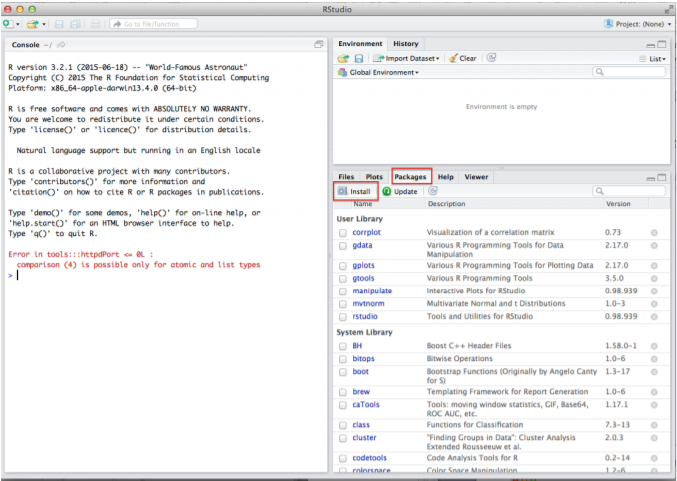


Figure 4: Install R packages

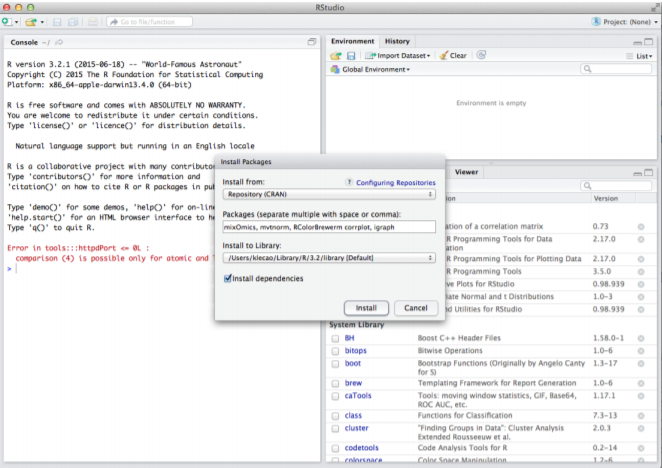


Figure 5: Install R packages

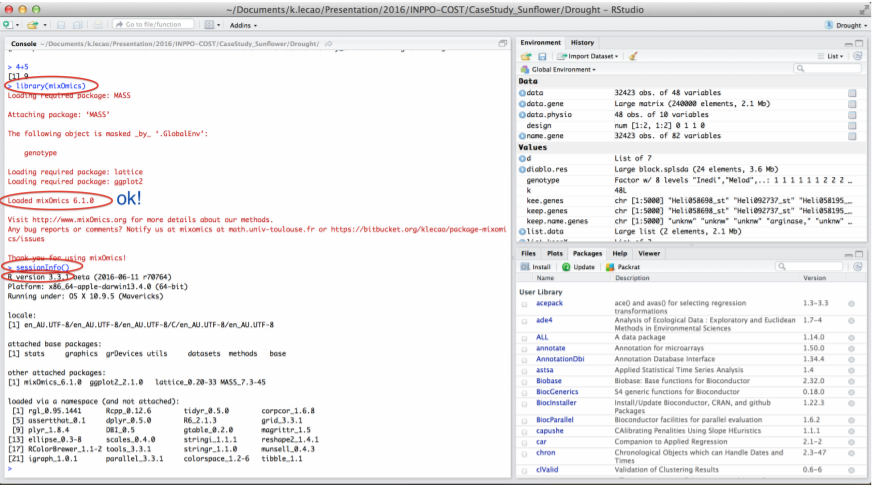


Figure 6: Install R packages

## The Dataset

Name: Landslides after rainfall (2007- 2016)

Source: Kaggle

The dataset consists of 23 columns (features or variables) and 1694 rows(recordings).

Data types and variables:

Numeric variables: id, population, distance, latitude, longitude, fatalities, injuries.

Categorical variables: continent\_code, country\_name, state/province, city/town, location\_description, hazard\_type, landslide\_type, landslide\_size, trigger, storm\_name, source\_name, source\_link.

Temporal data: date, time.

Spatial data: latitude, longitude, geolocation.

## Libraries Used.

ggplot2, dplyr, grid, gridExtra, tidyr, glue, scales

## Explain what type of analysis is done in the work.

* Installed R and Rstudio.
* Used the dataset Landslides after rainfall (2007 – 2016) to perform the following:
  + Exploratory data analysis using various functions.
  + Data visualization using basic r plots and ggplot.
  + Statistical analysis using basic functions.
  + Hypothesis testing.

CHAPTER-2

# Data Structure

(Some data structure variables have been shortened in the code because it’s easier to display the output)

## Vectors

A vector is the simplest type of data structure in R. Simply put, a vector is a sequence of data elements of the same basic type. Members of a vector are called Components.

Creating vectors:

data<-read.csv("C:\\Users\\Haritha\_M\\Desktop\\SEM3\\R\\PROJECT\\LANDSLIDES.csv")

id\_vector<-data[["id"]]

population\_vector<-data$population

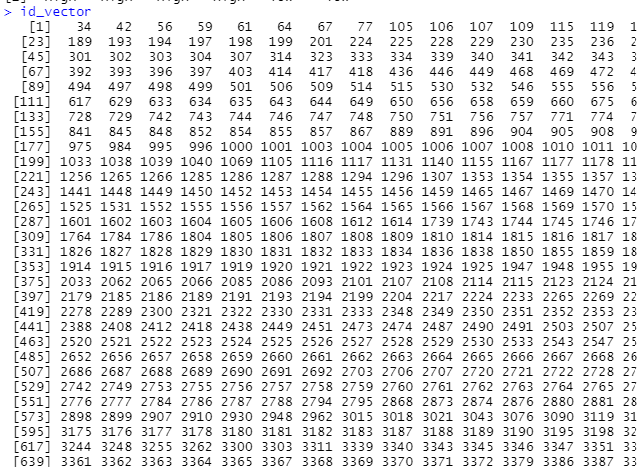
state\_vector<-data$state.province

distance\_vector<-data$distance

latitude\_vector<-data$latitude

longitude\_vector<-data$longitude

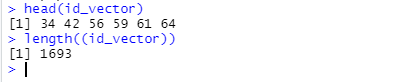
Fig: Displaying the id vector



Showing the 1st 6 elements of id\_vector and its length:

head(id\_vector))

length(id\_vector))



Accessing the first ten ids in the dataset:

cat("The first 10 ids in the data set are:",id\_vector[1:10])

Showing the 50th element of population vector:

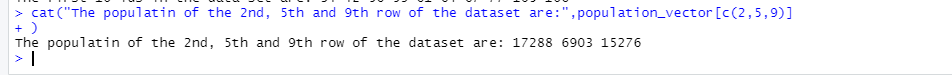
cat("The population of the 50th value is:",population\_vector[50])



Getting the population of the 2nd, 5th, 9th row:

(using a vector)

cat("The populatin of the 2nd, 5th and 9th row of the dataset are:",population\_vector[c(2,5,9)])



## Lists

A listis a generic vector containing other objects. Lists are data objects of R that contain various types of elements.

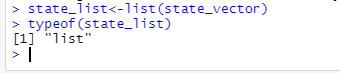
Creating a list:

Using list() on a pre-made vector:

state\_list<-list(state\_vector)

id\_list<-list(id\_vector)

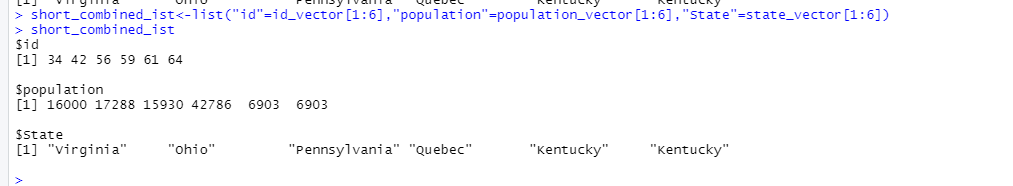
population\_list<-list(population\_vector)



Combining multiple vectors:

combinedlist<-list(id\_vector,population\_vector,state\_vector)

short\_combined\_ist<-list("id"=id\_vector[1:6],"population"=population\_vector[1:6],"State"=state\_vector[1:6])



Accessing different elements of the list:

short\_combined\_ist["id"]

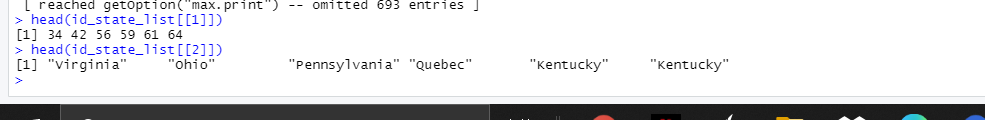
short\_combined\_ist[c(1,2)]

short\_combined\_ist$population



Merging two lists:

id\_state\_list<-c(id\_list,state\_list)



## Data frames

A data frame is a 2-dimensional data structure wherein each column consists of the value of one variable and each row consists of a value set from each column.

Reading an existing data frame:

data<-read.csv("C:\\Users\\Haritha\_M\\Desktop\\SEM3\\R\\PROJECT\\LANDSLIDES.csv")

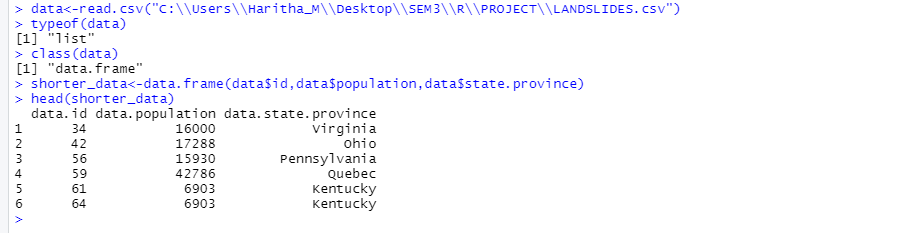
typeof(data)

class(data)

Creating a dataframe from id, population, state, province columns:

shorter\_data<-data.frame(data$id,data$population,data$state.province)

Fig. Displaying the dataframes



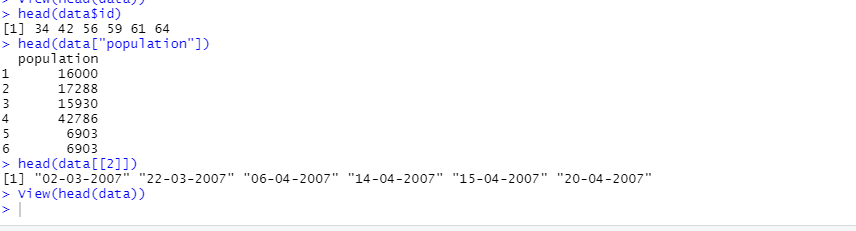
Accessing different parts of the data frame:

head(data$id)

head(data["population"])

head(data[[2]])

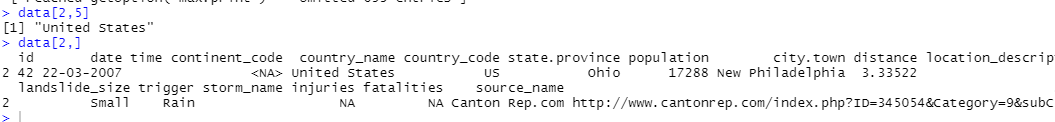
View(head(data))



Displaying the i) cell at 2nd row, 5th col. ii) second row

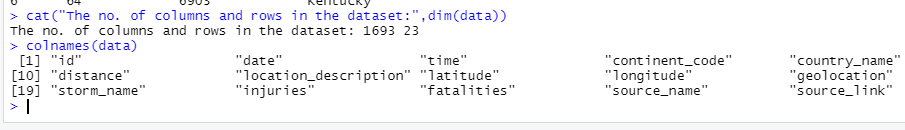
data[2,5]

data[2,]

Displaying information about the data frame:

cat("The no. of columns and rows in the dataset:",dim(data))

cat("Information of the data frame:",summary(data),sep="\n")



(

## Matrices

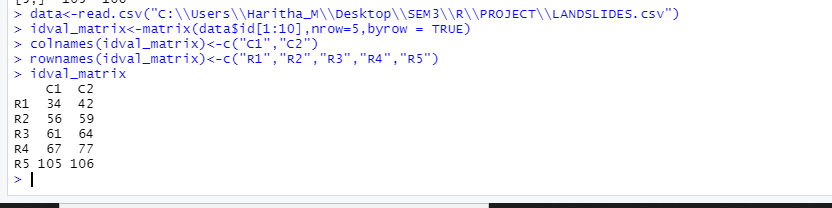
Matrices are used to arrange elements in the two-dimensional layout. They contain elements of the same data type.

Creating matrices from first 10 values of data frame's ‘id’ column:

idval\_matrix<-matrix(data$id[1:10],nrow=5, byrow = TRUE)

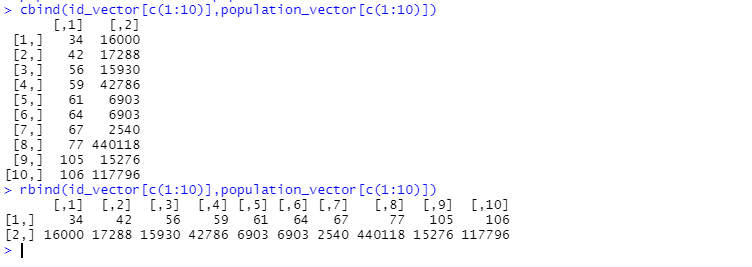
colnames(idval\_matrix)<-c("C1","C2")

rownames(idval\_matrix)<-c("R1","R2","R3","R4","R5")

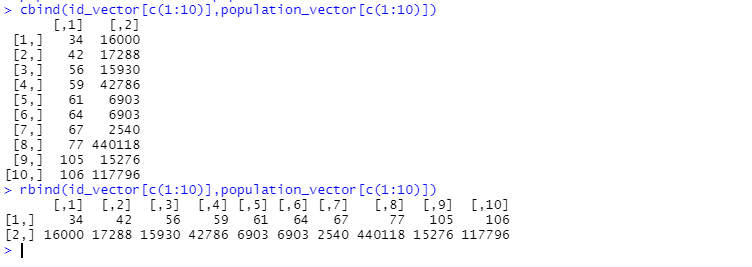


Creating using cbind() and rbind():

cbind(data$id,data$population)



rbind(id\_vector[c(1:10)],population\_vector[c(1:10)])



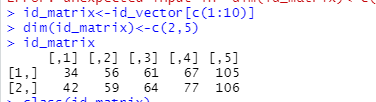
Creating using dim():

(id\_vector has been shortened to display the output easily)

id\_matrix<-id\_vector

id\_matrix<-id\_vector[c(1:10)]

dim(id\_matrix)<-c(2,5)

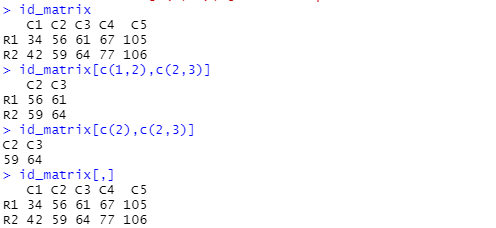


Accessing different elements of the matrix:

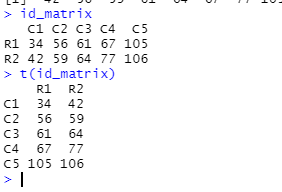
id\_matrix[c(1,2),c(2,3)]

id\_matrix[c(2),c(2,3)]

id\_matrix[,]



Transpose of a matrix:



## Arrays

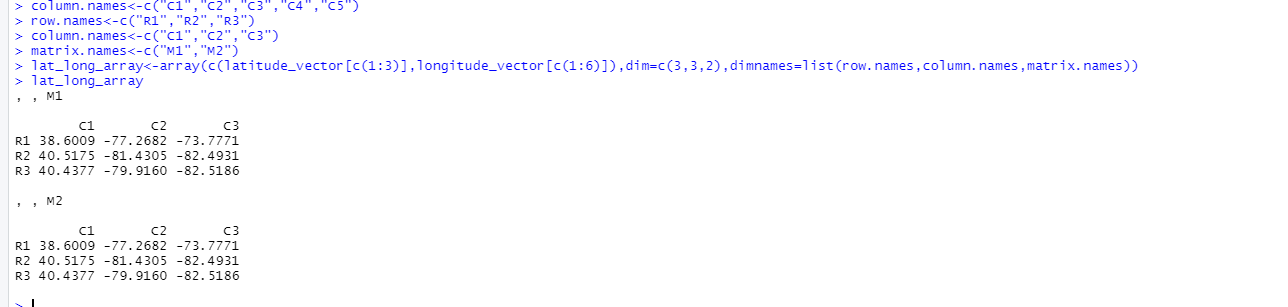
Creating an array and naming the rows, columns and matrices in it:

column.names<-c("C1","C2","C3")

matrix.names<-c("M1","M2")

row.names<-c("R1","R2","R3")

lat\_long\_array<-array(c(latitude\_vector[c(1:3)],longitude\_vector[c(1:6)]),dim=c(3,3,2),dimnames=list(row.names,column.names,matrix.names))



Accessing different parts of the array:

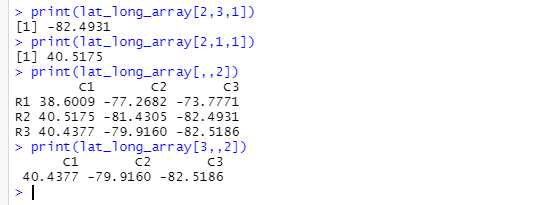
Syntax: array\_name[rownum,colnum,matrixnum]

print(lat\_long\_array[2,3,1])

print(lat\_long\_array[2,1,1])

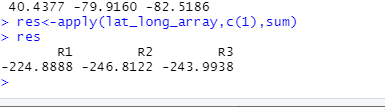
print(lat\_long\_array[,,2])

print(lat\_long\_array[3,,2])



apply() function:

res<-apply(lat\_long\_array,c(1),sum)



In the above code, we are calculating the sum of rows of both matrices in the array.

## Factors

Factors are data objects that are used in order to categorize and store data as levels.

countrynames<-data$country\_name

is.factor(countrynames)

countrynames<-factor(countrynames)

is.factor(countrynames)

nlevels(countrynames)

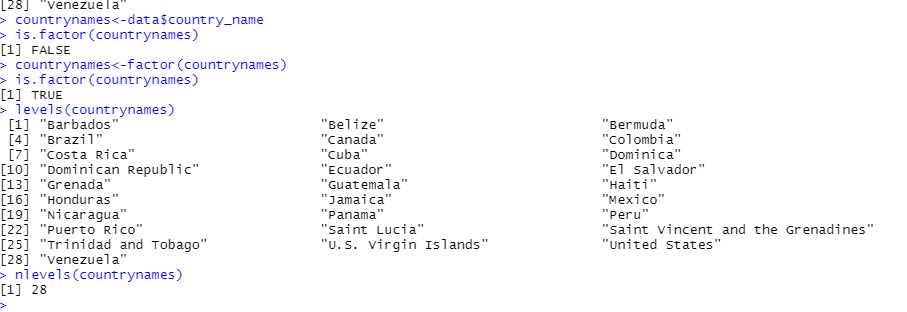
typeoflandslide<-factor(data$landslide\_type)

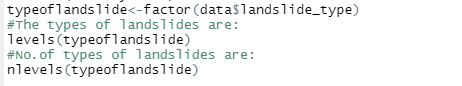
#The types of landslides are:

levels(typeoflandslide)

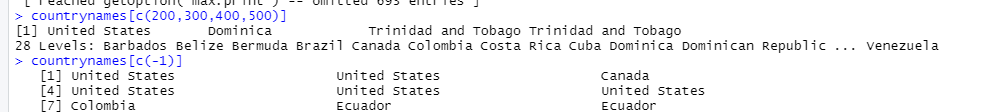
#No.of types of landslides are:

nlevels(typeoflandslide)





Accessing components:



CHAPTER-3

# Operations on Dates

## Reading the date column from the excel file and converting character to date:

dframe<-read.csv("C:\\Users\\Haritha\_M\\Desktop\\SEM3\\R\\PROJECT\\LANDSLIDES.csv")

typeof(dframe$date)

#to be converted to date

dates<-as.Date(dframe$date,"%d-%m-%Y")

typeof(dates)

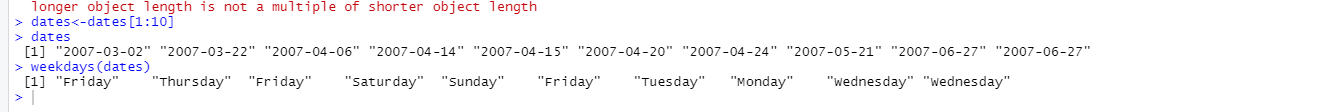
#dates are stored as double in R

class(dates)

****

## Making dates shorter for demonstrating operations and displaying what day of the week it is on that date:

weekdays(dates)



## Displaying the month of the dates:

months(dates)



## Displaying which quarter of the year the dates fall in:

quarters(dates)



CHAPTER-4

# Conditional statements on Vector, data frame, array, matrix

Decision making is an important part of programming. This can be achieved in R programming using the conditional if...else statement.

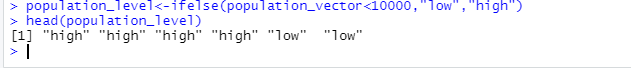
Syntax of ifelse() function:

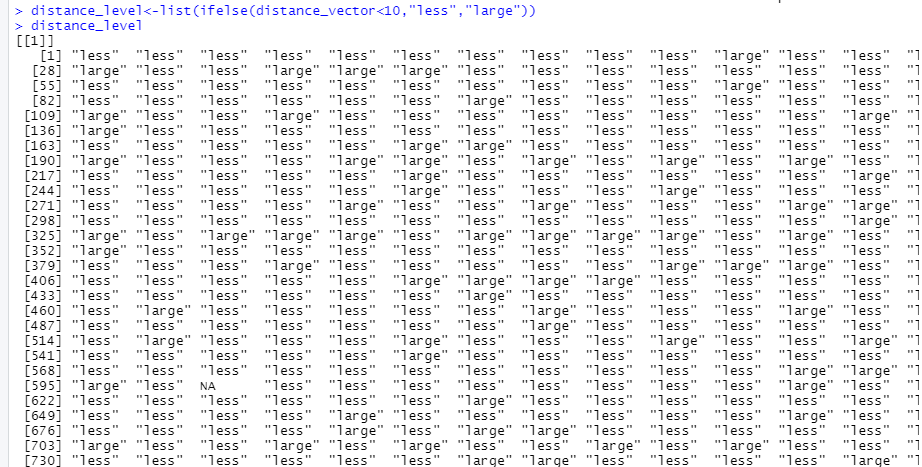
ifelse(test\_expression, x, y)

## Conditional statement on a vector:

population\_level<-ifelse(population\_vector<10000,"low","high")

distance\_level<-list(ifelse(distance\_vector<10,"less","large"))





## Conditional statement on data frames:

data$populationlevel<-ifelse(population\_vector<10000,"low","high")

head(data$populationlevel)

data$distancelevel<-ifelse(distance\_vector<10,"low","high")

head(data$distancelevel)

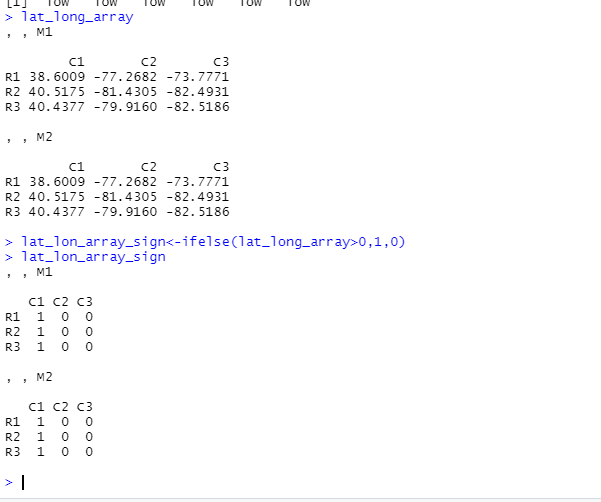


## Conditional statement on an array :

In the below code segment, values<0 are replaced by 0, else replaced by 1.

lat\_lon\_array\_sign<-ifelse(lat\_long\_array>0,1,0)

lat\_lon\_array\_sign



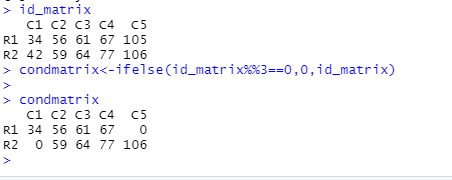
## Conditional statement on a matrix

(if value is divisible by 3, it gets replaced by 0)

id\_matrix

condmatrix<-ifelse(id\_matrix%%3==0,0,id\_matrix)

condmatrix



CHAPTER 5

# Loops

## Loop on a vector

Created a dataframe with onlt the data of USA:

dataUSA <- subset(data,data$country\_name=="United States")

Created a vector which contains the fatalities of USA:

vec =dataUSA$fatalities

vec\_fatalities <- [vec!is.na(vec)]

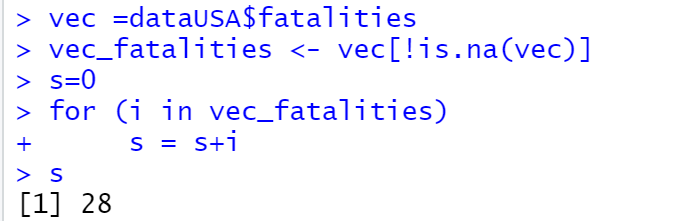
Calculated the total number of fatalites in USA:

s=0

for (i in vec\_fatalities)

s = s+i

OUTPUT



## Loop on a dataframe

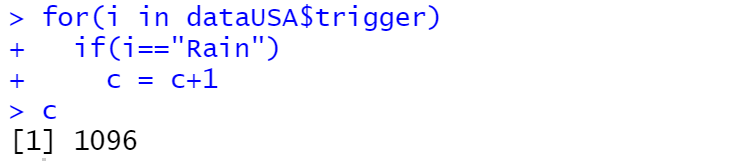
Using the above created dataframe, the total number of times lanslides have occurred in the US due to rain is calculated.

for(i in dataUSA$trigger)

if(i=="Rain")

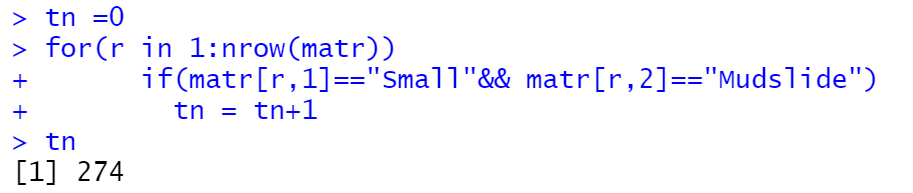
c = c+1

OUTPUT:



## Loop on a matrix

Using dataUSA, two vectors with the landslide size and landslide type is created and put into a matrix. The number of times there was small and mudslide landslide is calculated.



## Loop on an array:

The province which has the greatest distance in the year 2007 is calculated.

vecn = c(dataUSA$state.province[1:23]) //vector contains the provinces of USA

vecd = c (dataUSA$distance[1:23])// vector contains the distance of the lanslides of the province in USA (2007)

arr = array(c(vecn,vecd),dim =c(23,2,1))

gn=0

ign=0

for(i in 1:nrow(arr)) {

number = as.double(arr[i,2,1])

if(number > gn) {

gn = number

ign = i

}

}

vecn[ign]

OUTPUT

> vecn[ign]

[1] "Wyoming"

CHAPTER - 6

# Strings

Simple functions like sprintf, grep, union, setdiff are used.

Code :

dataset = read.csv("catalog.csv", header = TRUE)

for(i in nrow(dataset))

{

# Storing all unique countries in a vector

listOfCountries = unique(c(dataset$country\_name))

}

totalCountries = sprintf("There are %d different countries where landslides occurred in this dataset.", length(listOfCountries))

# Certain string operations

grep1 = grep("it", listOfCountries, value = TRUE)

grep2 = grep("z", listOfCountries, value = TRUE)

grep3 = grep("a", listOfCountries, value = TRUE)

union1 = union(grep1, grep2)

diff = setdiff(grep3, union1)

# Searching for an element in the list.

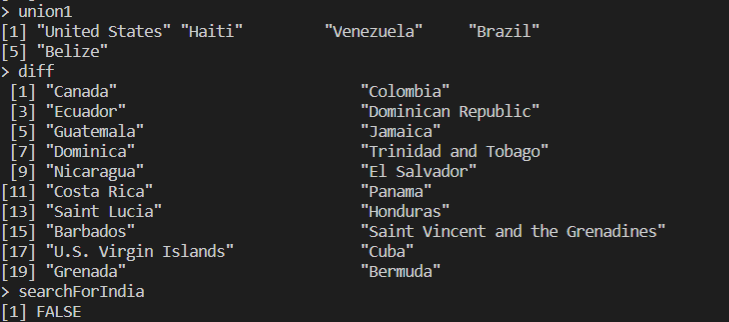
searchForIndia = is.element("India",listOfCountries)

# Sorting the elements in the list.

sortedListOfCountries = sort(listOfCountries)

Output :







CHAPTER – 7

# Function to show recursion:

To add the total number of fatalities in USA:

In the code vec\_fatalities is a vector which contains the fatalities of USA.

s=0

rec\_sum <- function(sum,n) {

if(n<=0) {

print("FinalSum")

print(sum)

return (sum)

}

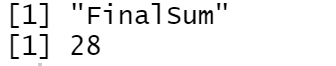
sum = sum + vec\_fatalities[n]

rec\_sum(sum,n-1)

}

finalSum = rec\_sum(s, length(vec\_fatalities))

OUTPUT:



CHAPTER – 8

# Usage of functions :

## All:

here vec2 contains fatalities

all(vec2>1)//checks if all the fatalities are greater than 1

## Any:

any(vec2>10)//checks if any of the fatalites are greater than 10

## Apply:

ma = cbind(fm)// fm is a vector which contains fatalities of Mexico

ma1 = apply(ma,2,sum)//adds the values present in the column

OUTPUT

fm

284

## Which :

To get the column number of the column specifies.

which(names(data)=="fatalities")

OUTPUT

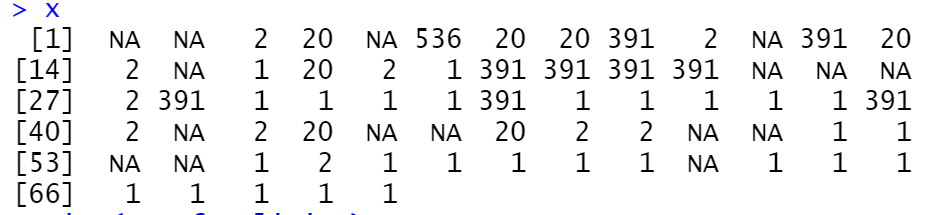
[1] 21

Match : vecm\_fatalities contains the fatalities of Mexico

vec\_fatalities contains the fatalities of USA

x <- match(vecm\_fatalities,vec\_fatalities)

OUTPUT



## Order:

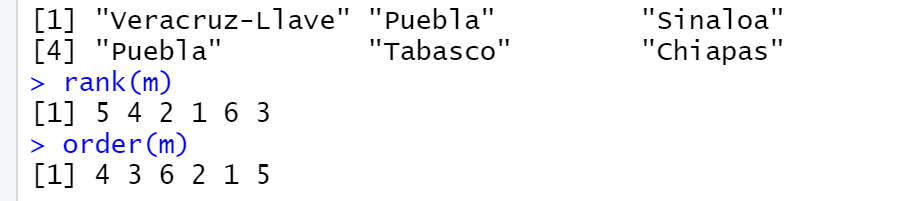
order(m)// m contains the fatalities of Mexica

## Rank:

rank(m)

OUTPUT

Ranks the province in Mexico based on fatalities(only 2007)



Sub:

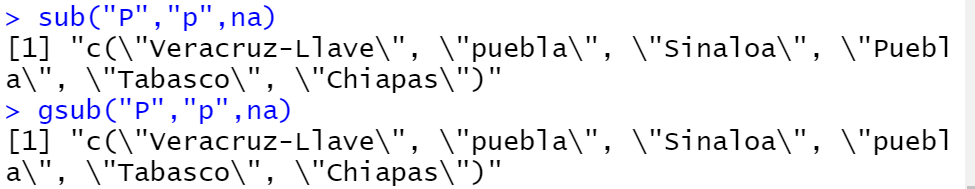
na=data.frame(head(dataMEX$state.province,6))

sub("P","p",na)

Gsub:

gsub("P","p",na)

OUTPUT



## Aggregate:

To calculate the average fatalities of provinces in Mexico

p = c(dataMEX$state.province)

f = c(dataMEX$fatalities)

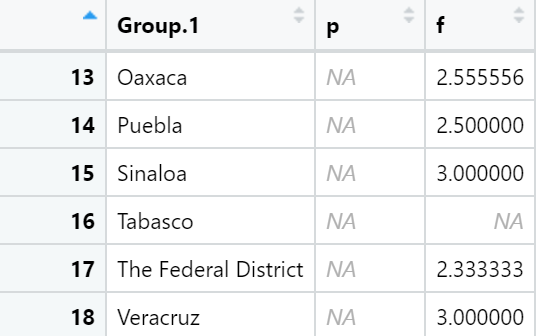
na= data.frame(p,f)

agg = aggregate(na,by=list(na$p),FUN=mean)

OUTPUT









CHAPTER-9

# Data Visualization using basic R plots

## Data Preparation

data<-read.csv("C:/users/seren/OneDrive/Desktop/2ndYear/R/R project/catalog.csv")

New year and month columns created:

data$year<-sapply(data$date,function(x) as.numeric(strsplit(x,'/')[[1]][3])) + 2000

data$month<-sapply(data$date,function(x) as.numeric(strsplit(x,'/')[[1]][1]))

New column of abbreviated months (MonthAbb) was created followed by a column of the ordered months (month\_ordered):

myMonths <- c("January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December")

data$MonthAbb <- myMonths[ data$month ]

data$month\_ordered <- factor(data$MonthAbb, levels = month.name)

Values in landslide type, landslide size, trigger converted to lower case:

data$landslide\_type <- tolower(data$landslide\_type)

data$landslide\_size <- tolower(data$landslide\_size)

data$trigger <- tolower(data$trigger)

Removing all null values from landslide type, landslide size, year and trigger:

data<-data[!is.na(data$year),]

data<-data[!(data$landslide\_type == ""),]

data<-data[!(data$landslide\_size == ""),]

data<-data[!(data$trigger == ""),]

Forming a new dataset with recordings of the United States:

dataUSA = subset(data, data$country\_name=="United States")

dataUSA<-dataUSA[!is.na(dataUSA$fatalities),]

dataUSA<-dataUSA[!is.na(dataUSA$injuries),]

dataUSA<-dataUSA[order(dataUSA$year),]

## What size of Landslides are most likely to occur?

Code:

size = length(levels(as.factor(data$landslide\_size)))

pie(table(data$landslide\_size), labels = levels(as.factor(data$landslide\_size)), main = "Landslide Size Occurrences", col = terrain.colors(size), cex=0.8)

Plot:

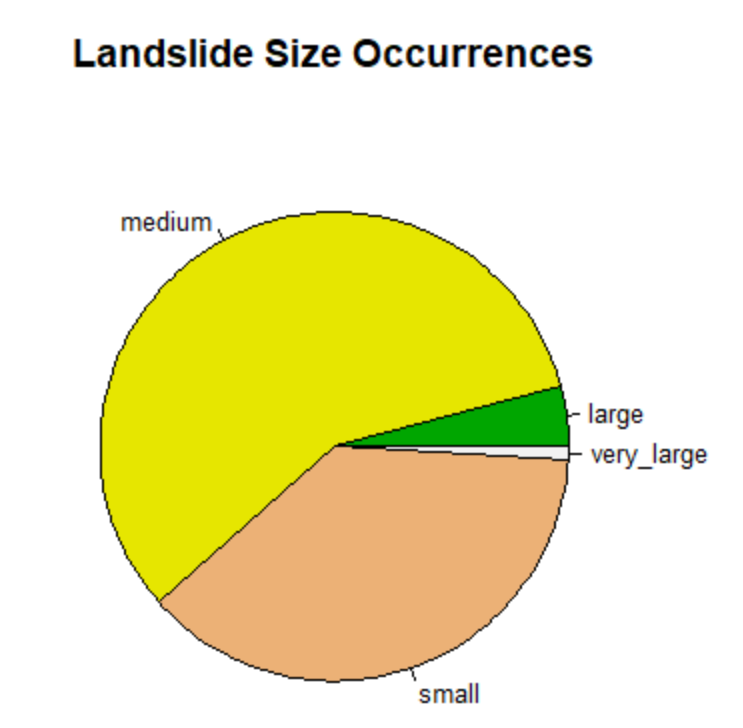


Figure 7: Landslide Size Occurrences

Inference:

From the above chart we can infer that medium sized landslides are most likely to occur and very large landslides are least likely to occur.

## Occurrence of different types of triggers of landslides:

Code:

barplot(table(data$trigger), col=brewer.pal(5,"Set2"), xlab="Frequency", main="Occurrences of different types of triggers", horiz=T, las=2, cex.names=0.5)

Plot:

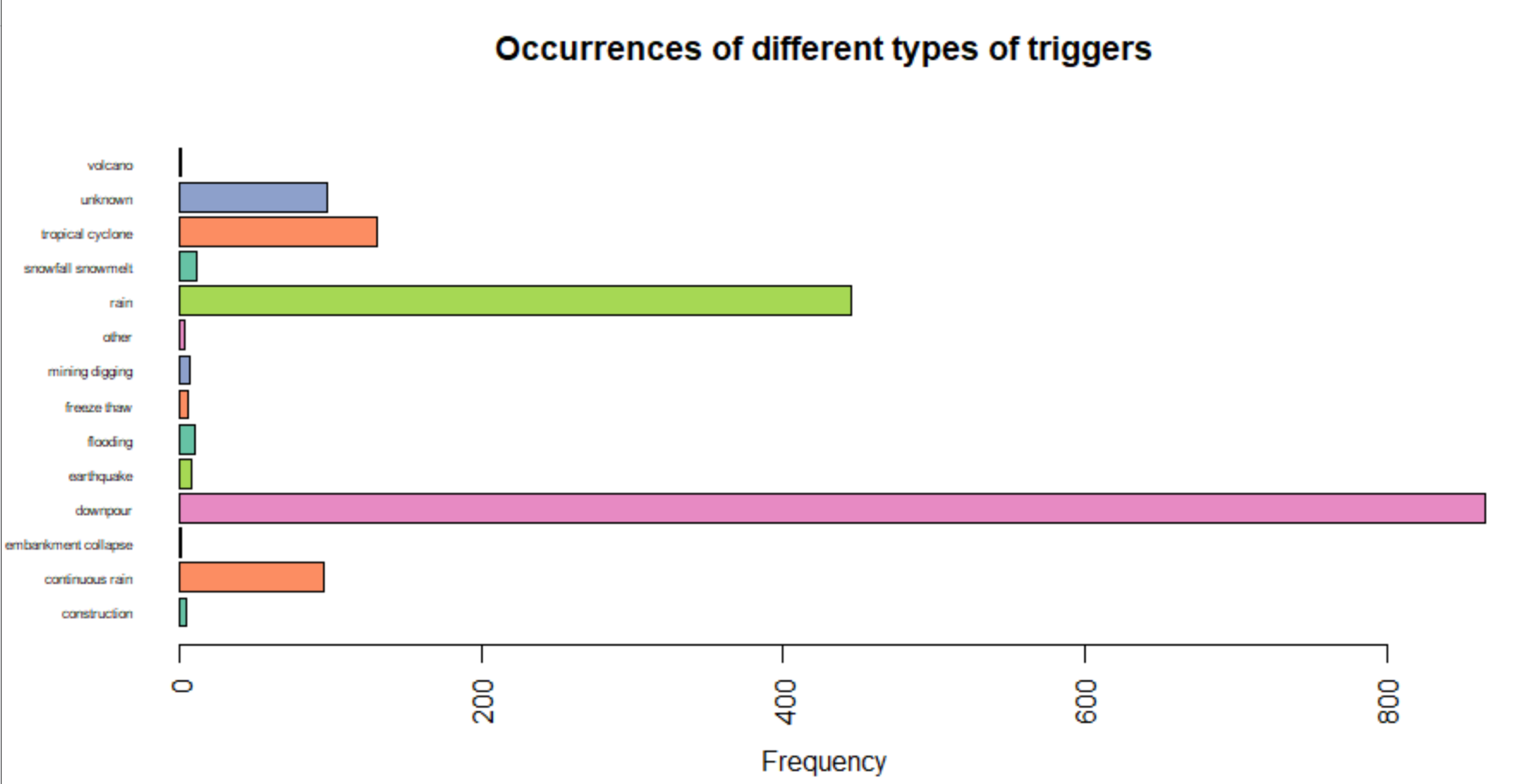


Figure 8: Occurrence of landslides due to different types of triggers

Inference:

From the above plot we can infer that landslides are most likely to be caused by downpour followed by rain, and it is least likely to be caused by a volcano and embarkment collapse.

## Number of fatalities and injuries that occurred vs the distance travelled by the landslide:

Code:

plot(y=data$fatalities, x=data$distance, type="p", xlab="distance", ylab="fatalities", main = "Fatalities vs Distance traveled by landslide")

plot(y=data$injuries, x=data$distance, type="p", xlab="distance", ylab="injuries", main="Injuries vs Distance traveled by landslide")

Plots:

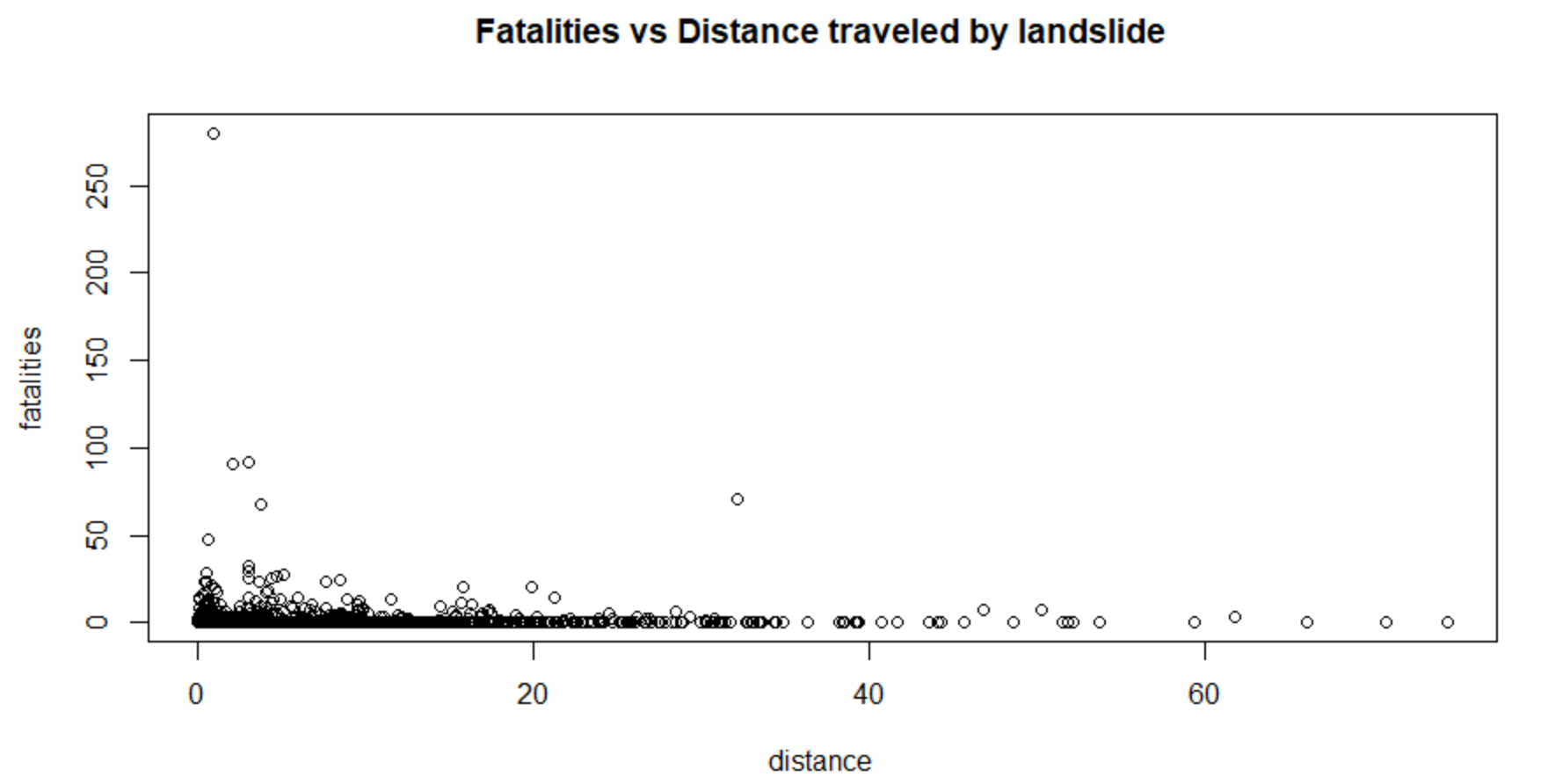


Figure 9: Fatalities vs Distance traveled by landslide

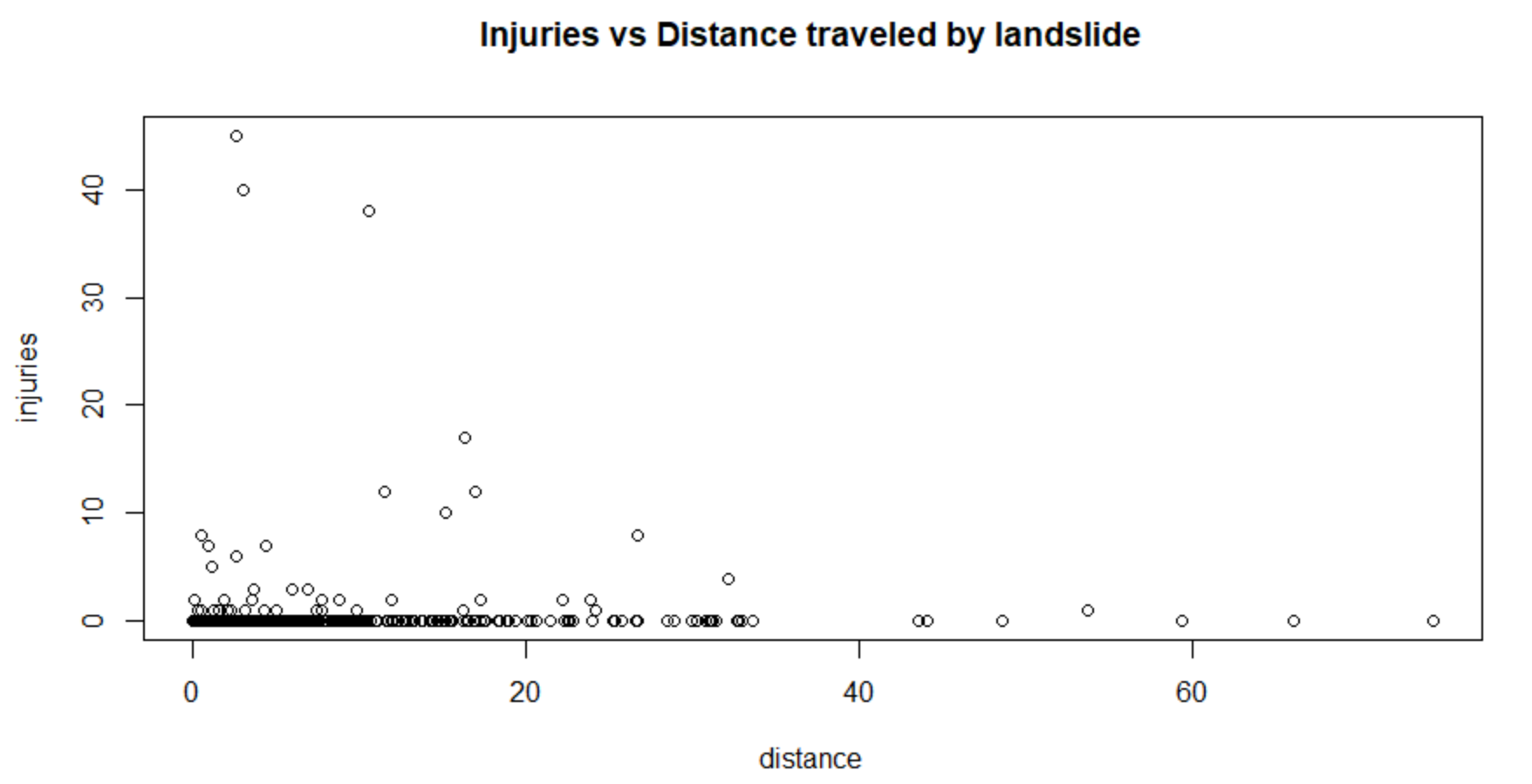


Figure 10: Injuries vs Distance traveled by landslide

Inference:

From the above plots we can infer that landslides that happen over a shorter distance are more fatal and injurious than the ones that have been recorded over longer distances.

## Fatalities in US over the years

Code:

sum = aggregate(x=dataUSA$fatalities, by=list(dataUSA$year), FUN=sum)

plot(x=sum$Group.1, y=sum$x, type="l", xlab="year", ylab="total fatalities", col="red", main="Fatalities in USA over the years")

Plot:

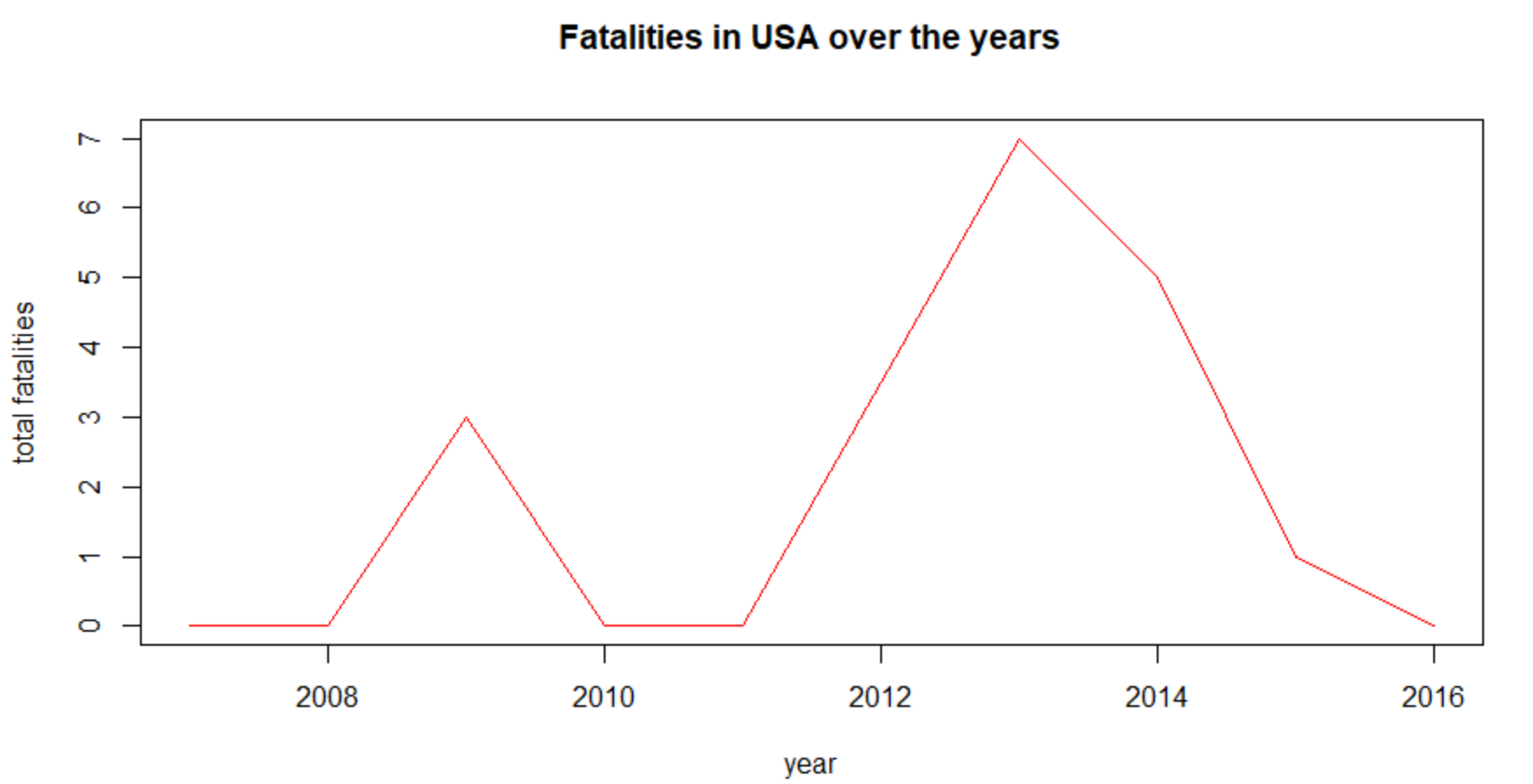


Figure 11: Fatalities in USA from 2007 - 2016

Inference:

From the above plot we can infer that the greatest number of fatalities that occurred in USA due to landslides was around the year 2013.

CHAPTER-10

# Data visualization using ggplot

## Distribution of landslide per type and country:

Steps:

* Group the columns by country\_name and landslide\_type
* Summarize the content and calculate the count
* Analyze the data using ggplot2 with country name along the x axis and frequency of occurrence of different landslide types along the y axis.

Code:

data %>%

group\_by(country\_name, landslide\_type) %>%

summarize(count = n()) %>%

ggplot(aes(x = reorder(country\_name, count), y = count, fill = landslide\_type)) +

xlab('Country') + ylab('Frequency') +

ggtitle("Distribution of landslide per type and country")+

coord\_flip() +

geom\_bar(stat = 'identity')

Plot:

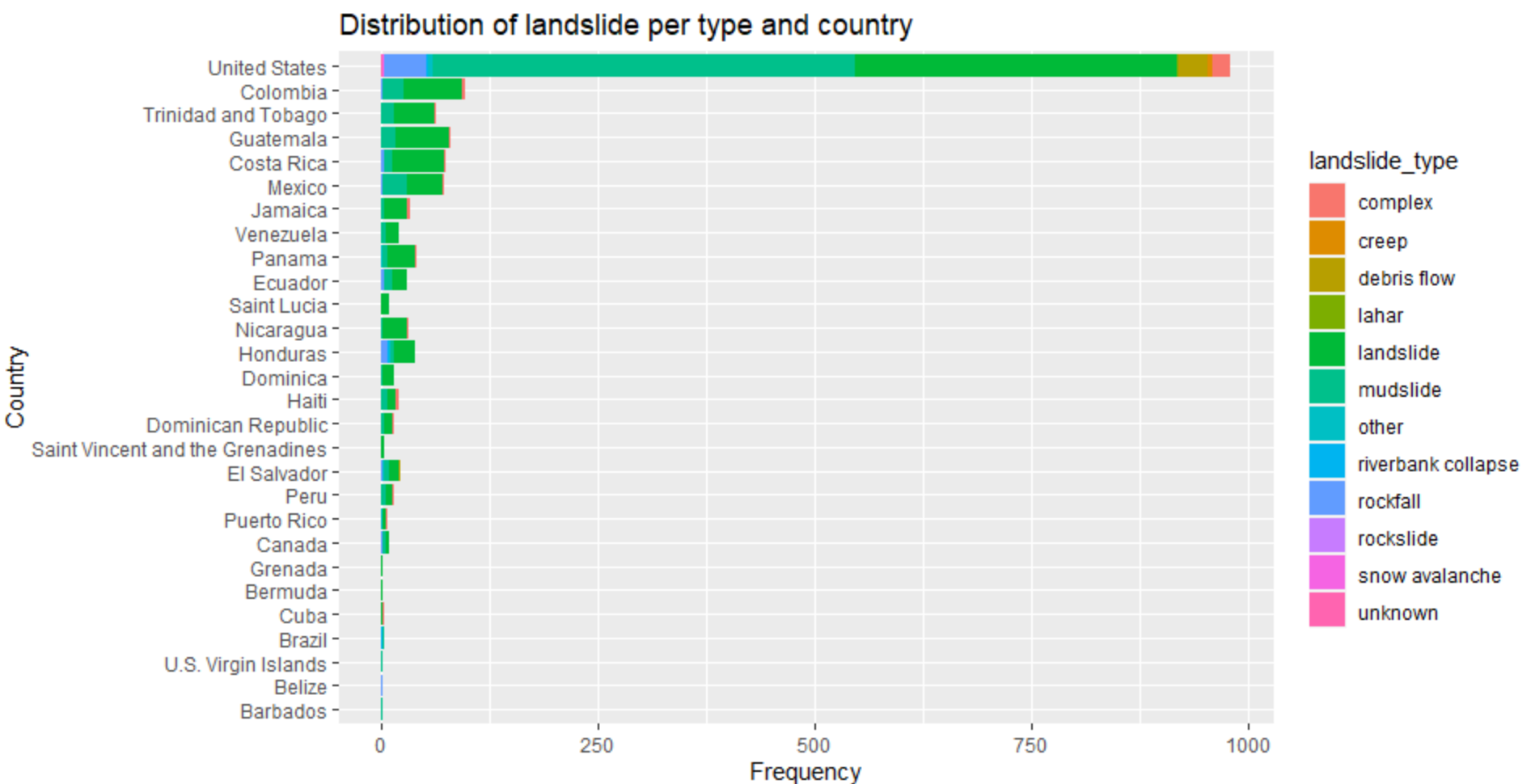


Figure 12: Distribution of landslide per type and country

Inference:

From the above plot we can infer:

* Landslides most commonly occur in the United States.
* Landslides and mudslides are the types of landslides that are most likely to occur.

## Distribution of landslide type:

Steps:

* Group the columns by year landslide\_type
* Summarize the content and calculate the count
* Analyze the data using ggplot2

Code:

data%>%

group\_by(landslide\_type) %>%

summarize(count = n()) %>%

ggplot(aes(x='', y=count, fill=landslide\_type))+

ggtitle("Distribution of landslide type")+

geom\_bar(stat = "identity")+

theme\_void()+

coord\_polar("y")+

geom\_text((aes(label=percent(count/sum(count),accuracy = 0.01))), position = position\_stack(vjust = 0.5), size=3)

Plots:

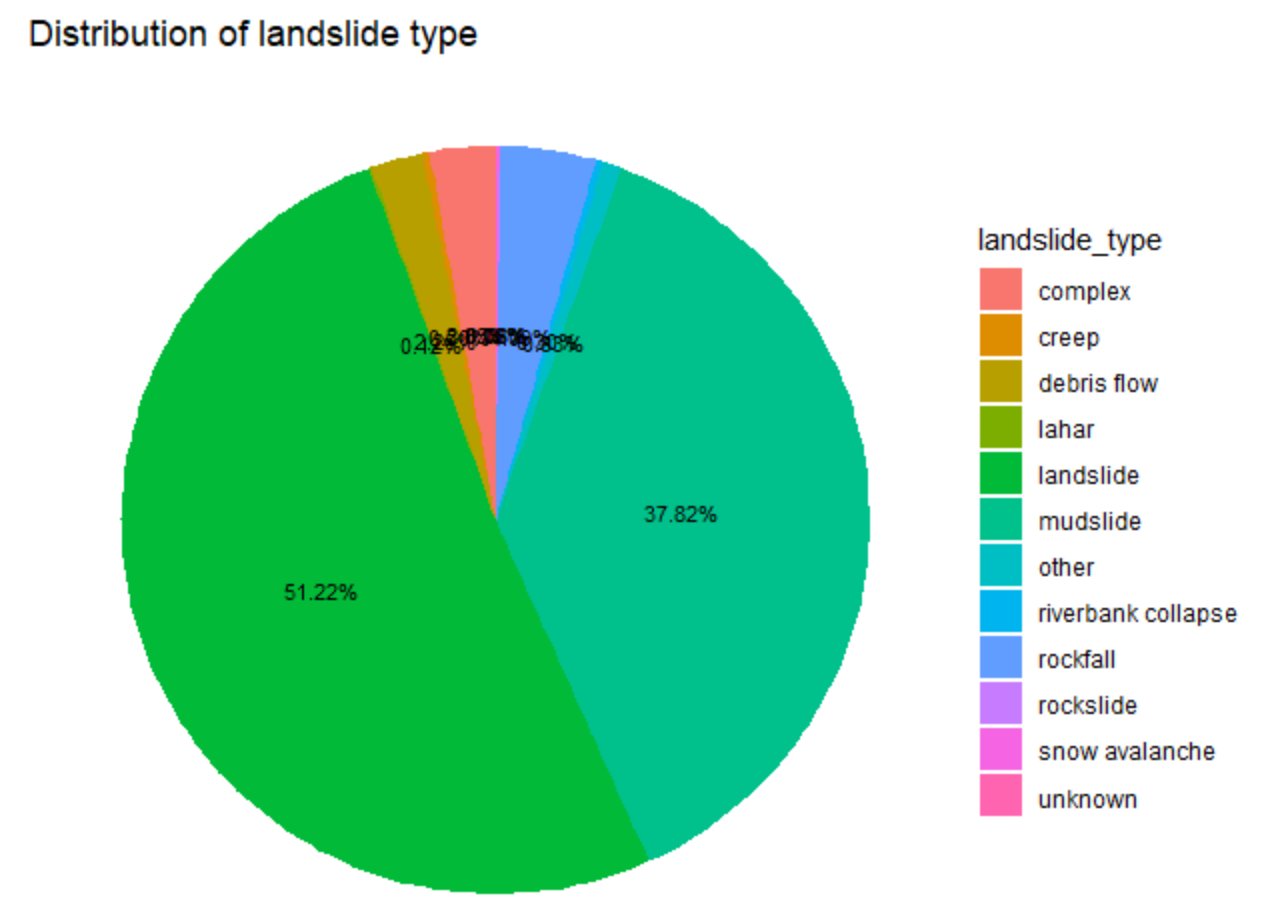


Figure 13: Distribution of landslide type

Inference:

From the above plot we can infer that landslides are the most common type of landslide to occur followed by mudslides.

## Distribution of landslides per trigger per year

Steps:

* Group the columns by year and trigger
* Summarize the content and calculate the count
* Analyze the data using ggplot2 with year along the x axis and frequency of occurrence of different triggers along the y axis.

Code:

data %>%

group\_by(year, trigger) %>%

summarize(count = n()) %>%

ggplot() +

ggtitle("Distribution of landslide per trigger per year")+

geom\_line(aes(x = year, y = count, color = trigger), size = 2)

Plot:

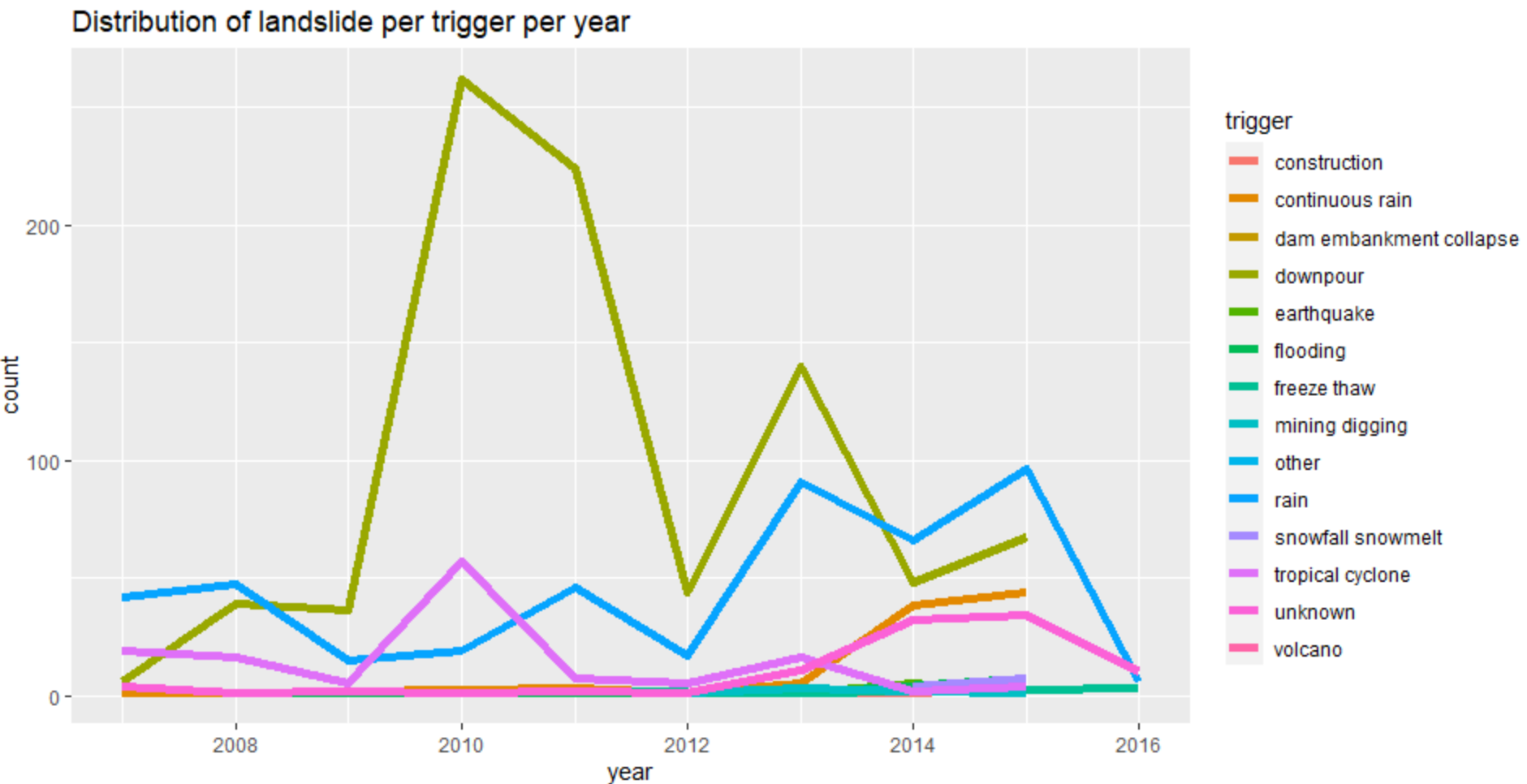


Figure 14: Distribution of landslides per trigger per year

Inference:

From the above plot we can infer that landslides are most commonly triggered by downpour followed by rain.

## To check if there is a seasonality of occurrence of landslides:

Steps:

* Group the columns by year and month\_ordered
* Summarize the content and calculate the count
* Analyze the data using ggplot2 with month\_ordered along the x axis and frequency of occurrence of different year along the y axis.

Code:

data %>%

group\_by(year, month\_ordered) %>%

summarize(count = n()) %>%

ggplot(aes(x = year, y = month\_ordered)) +

ggtitle("Seasonality of occurence of landslides") +

geom\_tile(aes(fill = count), colour = "white")

data %>% select(month\_ordered, year) %>%

group\_by(month\_ordered, year) %>%

summarize(count = n()) %>%

ggplot(aes(x = month\_ordered, y = count)) +

xlab('Month') + ylab('Frequency') +

ggtitle("Seasonality of occurence of landslides") +

geom\_boxplot(colour = 'black') +

geom\_smooth(method = 'lm', formula = y ~ poly(x, 2), color = 'blue', lty = 2, aes(group = 1))

Plots:

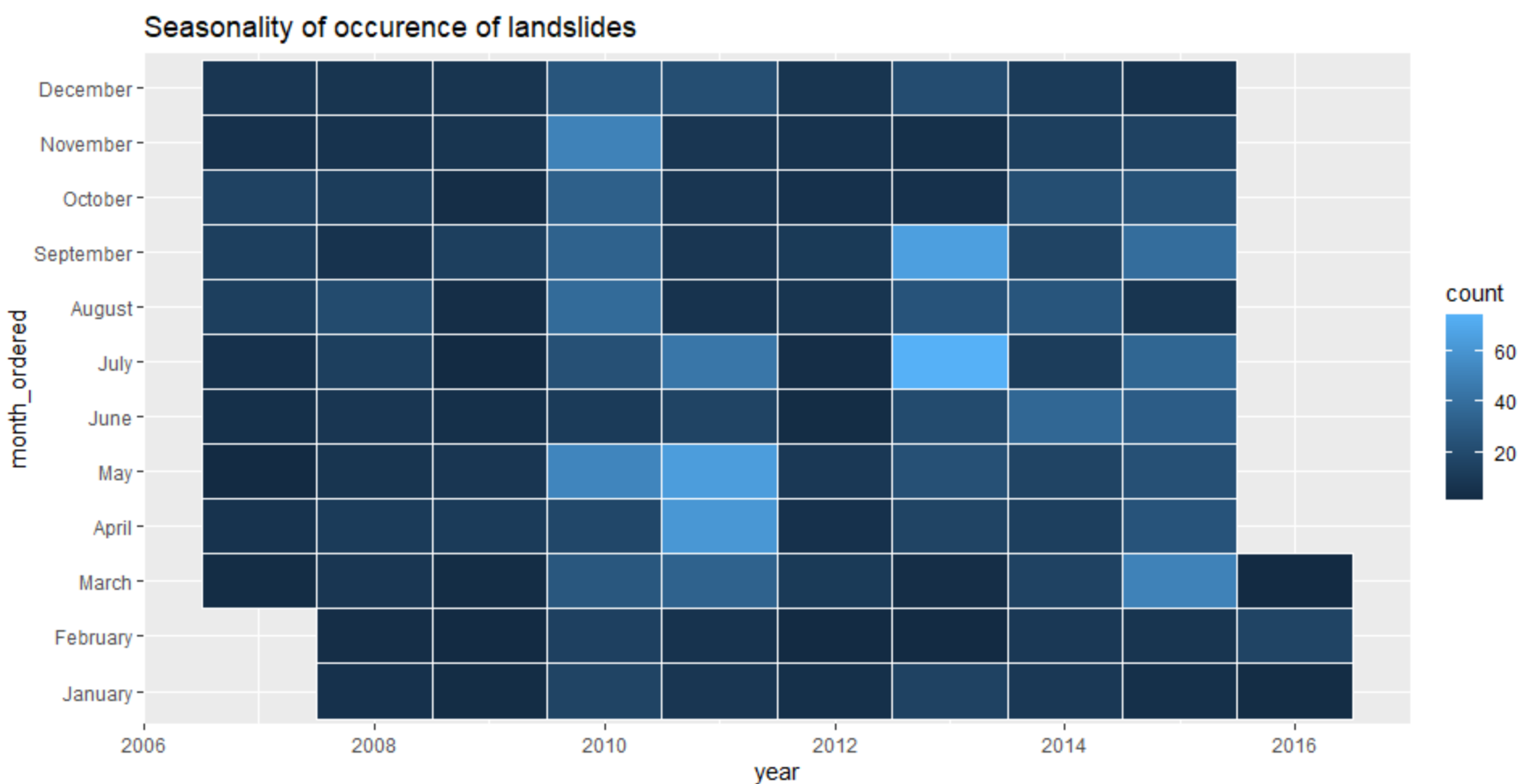


Figure 15: is there seasonality of occurrence of landslides? (heatmap)

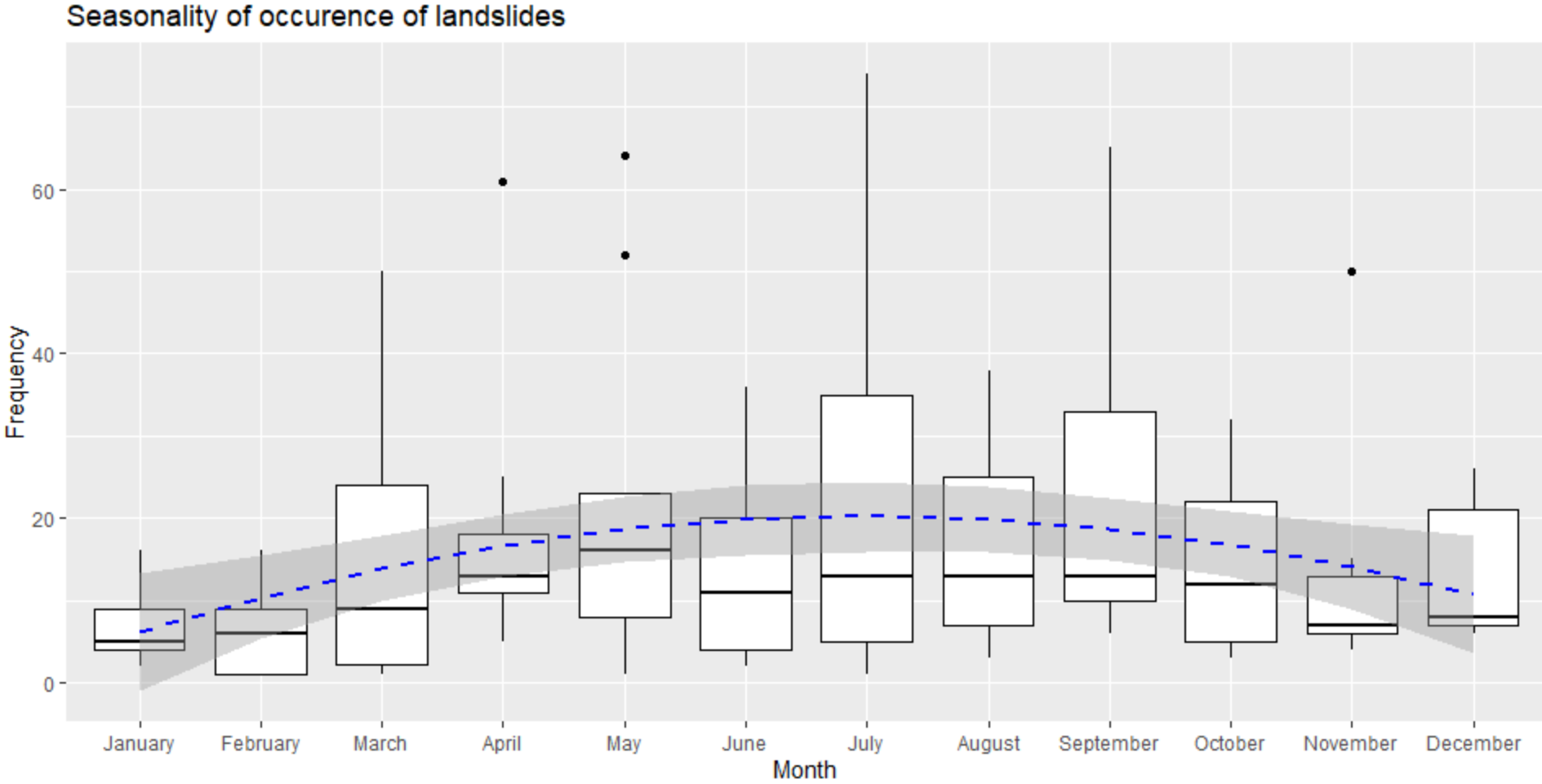


Figure 16: is there seasonality of occurrence of landslides? (boxplot)

Inference:

There is no clear seasonality that can be visualized in the heatmap. However, in the boxplots there is a seasonality where in the beginning and end of the year there are less landslides than during the mid of the year.

## To analyze the region of occurrences of various types of landslides and the number of fatalities they caused:

Code:

worldMap <- fortify(map\_data("world"), region = "region")

map <- ggplot() +

geom\_map(data = worldMap, map = worldMap, aes(x = long, y = lat, map\_id = region)) +

ggtitle("Frequency of occurrence of fatalities due to different landslide types on World Map") +

xlim(-160,-30) + ylim(-60,60)

map + geom\_point(data = data, aes(x = longitude, y = latitude, color = landslide\_type, size = fatalities))

Plot:

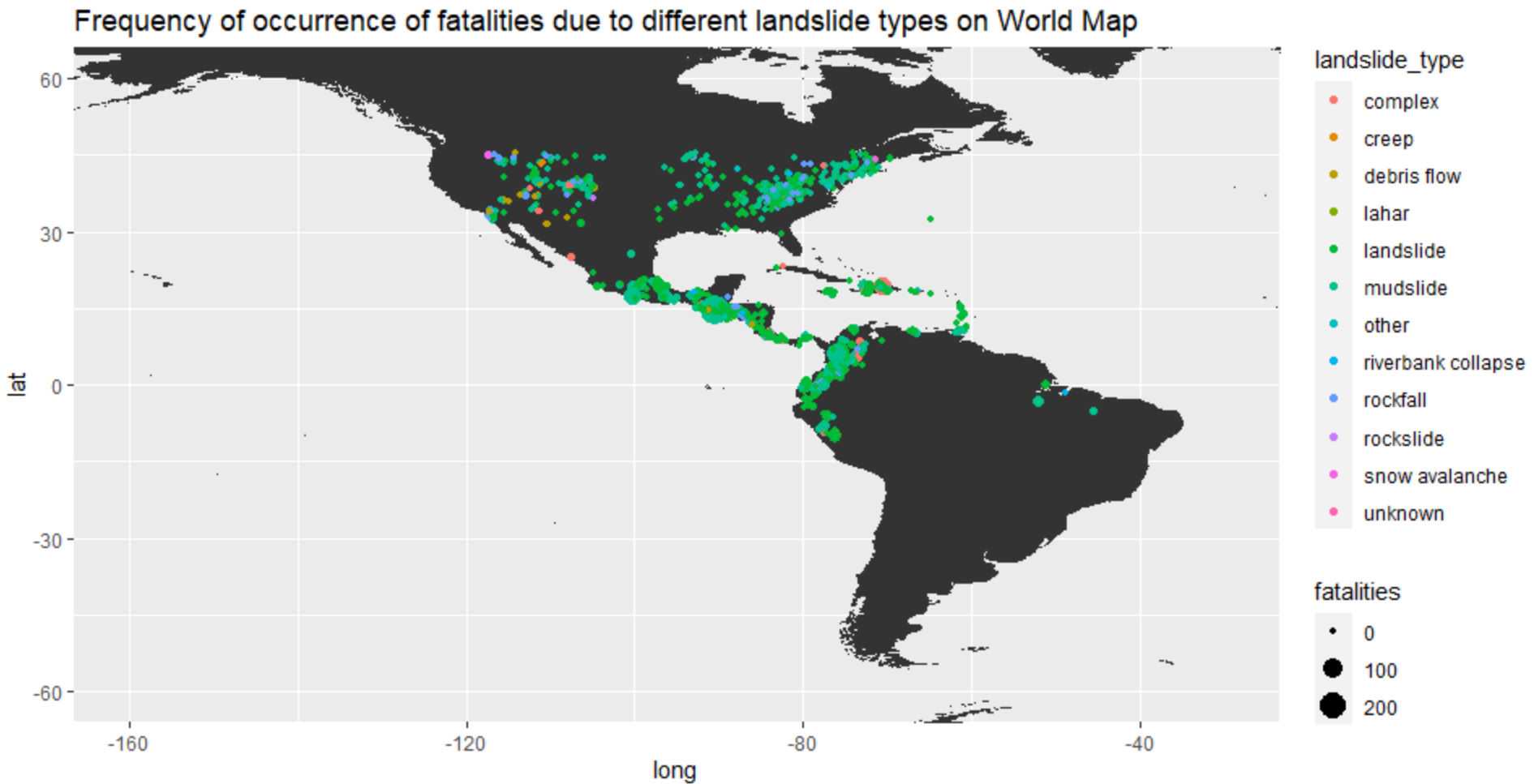


Figure 17: World map - landslide type and fatalities

Inference:

From the above plot we can infer:

* Most common landslide types are landslides and mudslides.
* Most number of fatalities are caused by mudslides.
* Landslides are most likely to occur in regions closer to waterbodies.

CHAPTER-11

# Statistical Analysis of landslides in 2015

## Analysis of fatalities, injuries and distance traveled by landslides

Code:

data2015 = subset(data, data$year=="2015")

data2015<-data2015[!is.na(data2015$fatalities),]

data2015<-data2015[!is.na(data2015$injuries),]

cols <- c("distance", "fatalities", "injuries");

summary(data2015[cols])

Output:

distance fatalities injuries

Min. : 0.02568 Min. : 0.000 Min. : 0.0000

1st Qu.: 1.84109 1st Qu.: 0.000 1st Qu.: 0.0000

Median : 3.89837 Median : 0.000 Median : 0.0000

Mean : 6.64711 Mean : 1.483 Mean : 0.5132

3rd Qu.: 8.38116 3rd Qu.: 0.000 3rd Qu.: 0.0000

Max. :74.46097 Max. :280.000 Max. :45.0000

## Finding levels of landslide type, landslide size and triggers:

Code:

levels(factor(data2015$landslide\_type))

levels(factor(data2015$landslide\_size))

levels(factor(data2015$trigger))

Output:

* Landslide type:   
  "creep" "debris flow" "landslide" "mudslide" "other" "riverbank collapse" "rockfall" "rockslide” "unknown"
* Landslide size:  
  "large" "medium" "small"
* Trigger:   
  "construction" "continuous rain" "downpour" "earthquake" "flooding" "mining digging" "rain" "snowfall snowmelt" "tropical cyclone" "unknown" “volcano”

# Statistical Analysis of landslides in USA

## Analysis of fatalities

Steps:

* Find the mean, median and standard deviation of fatalities, injuries and distance.
* Plot histogram/boxplot

Code:

mean(dataUSA$fatalities)

median(dataUSA$fatalities)

sd(dataUSA$fatalities)

hist(dataUSA$fatalities)

Output:

mean: 0.05947955

median: 0

standard deviation: 0.4364388

Plot:

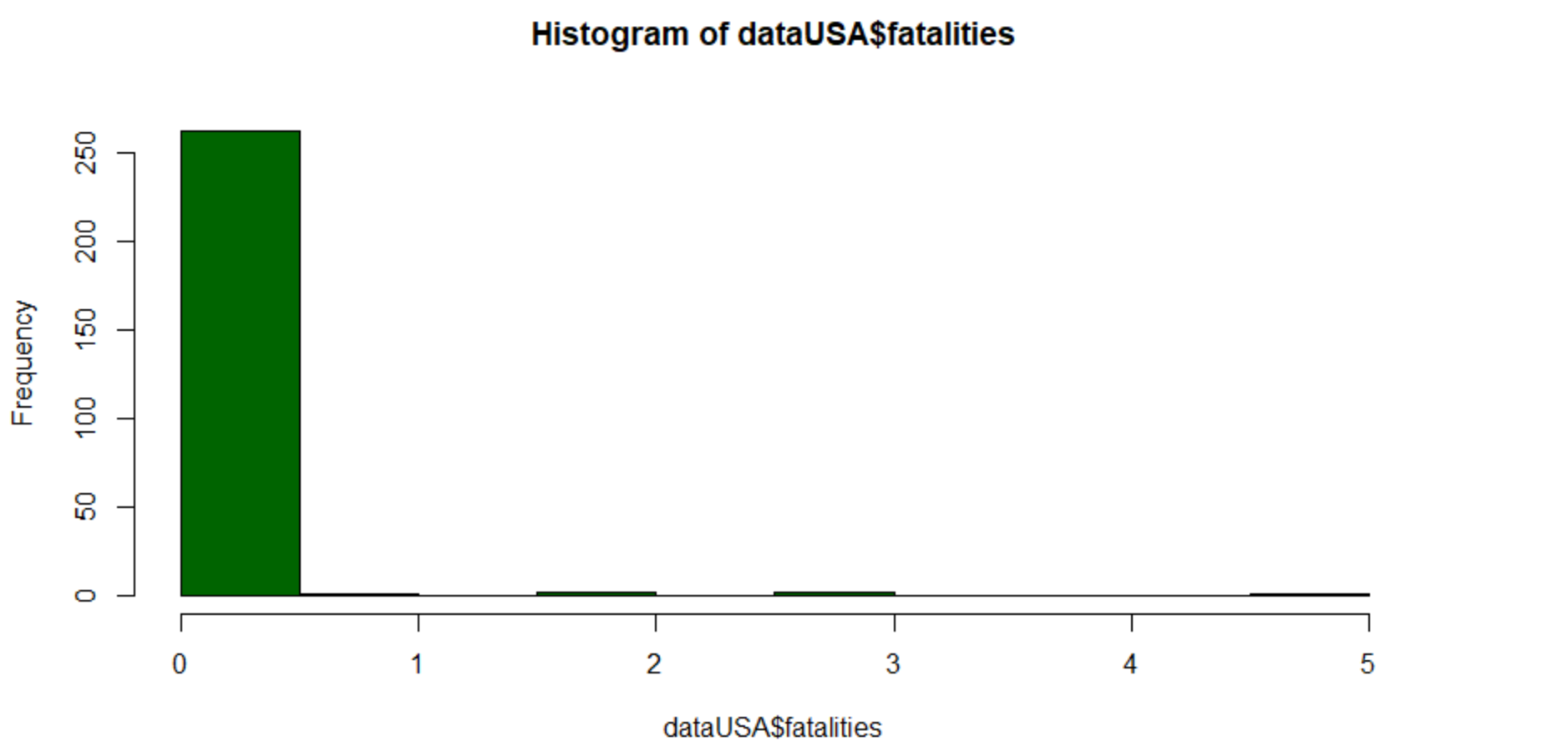


Figure 18: Fatalities in USA

## Analysis of injuries

Code:

mean(dataUSA$injuries)

median(dataUSA$injuries)

sd(dataUSA$injuries)

hist(dataUSA$injuries, col = "darkgreen")

Output:

mean: 0.04832714

median: 0

standard deviation: 0.261823

Plot:

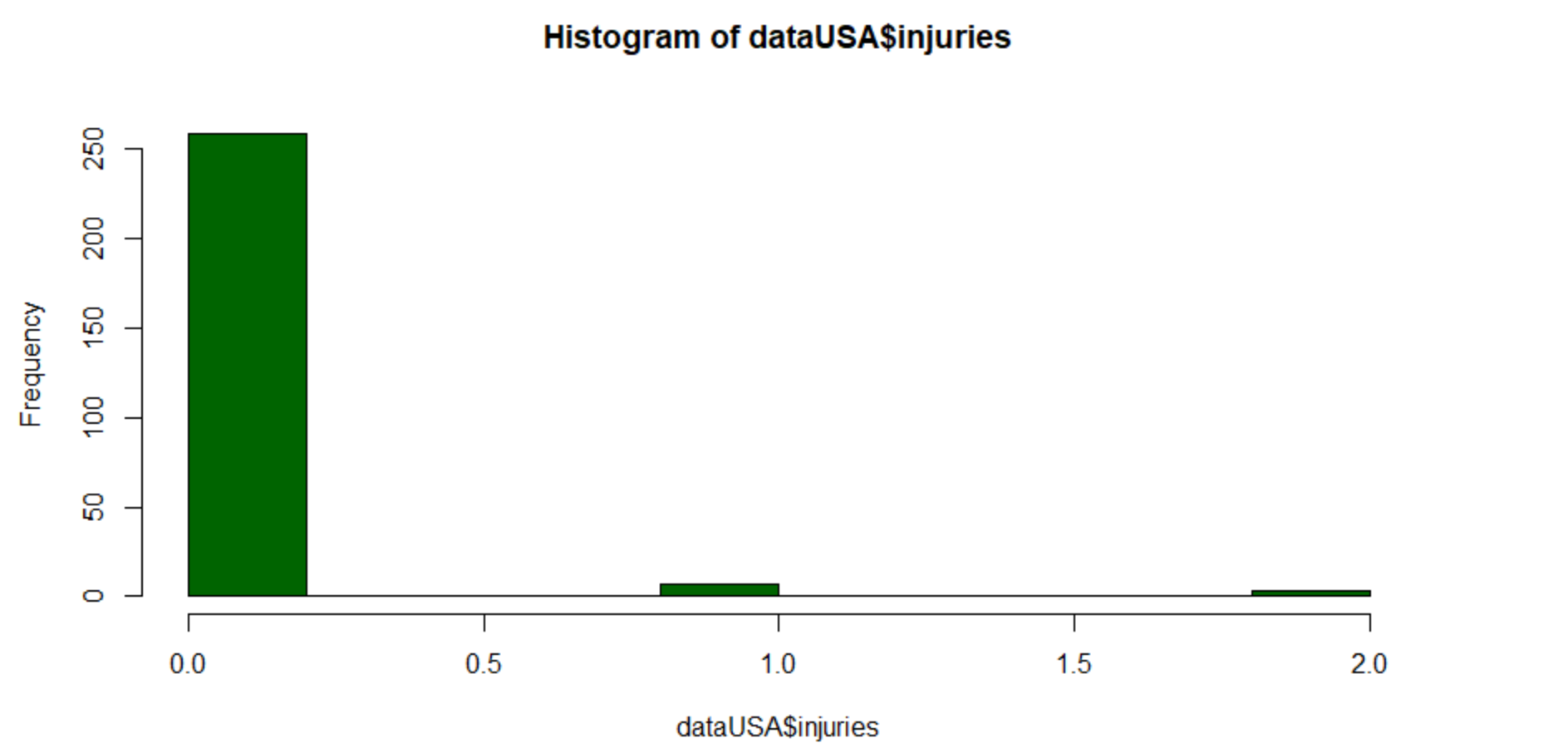


Figure 19: Injuries in USA

## Analysis of distance traveled by landslide

Code:

mean(dataUSA$distance)

median(dataUSA$distance)

sd(dataUSA$distance)

boxplot(dataUSA$distance, col="lightgreen")

Output:

mean: 7.961526

median: 5.07093

standard deviation: 10.032

Plot:

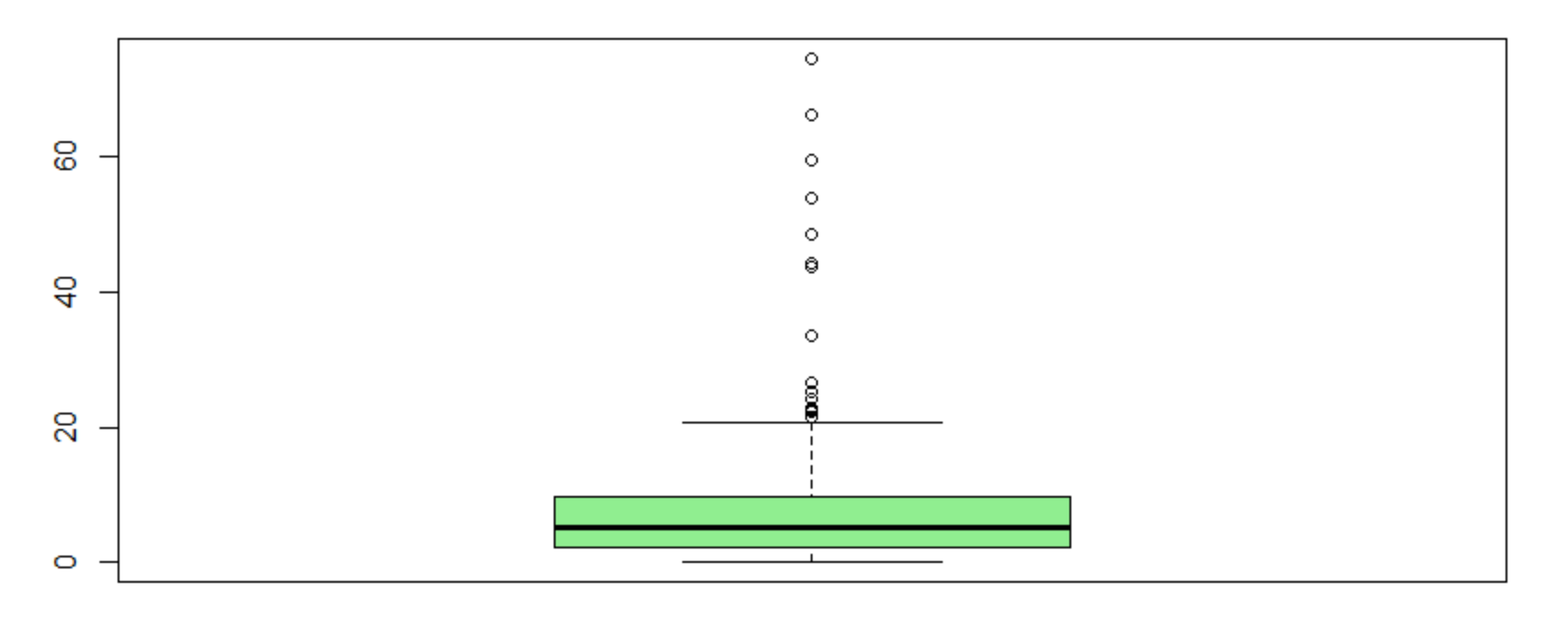


Figure 20: Distance travelled by landslides in USA

## Finding levels of year, landslide type, landslide size and triggers:

Code:

levels(factor(dataUSA$year))

levels(factor(dataUSA$landslide\_type))

levels(factor(dataUSA$landslide\_size))

levels(factor(dataUSA$trigger))

Output:

* Year:   
  "2007" "2008" "2009" "2010" "2011" "2013" "2014" "2015" "2016"
* Landslide type:   
  "complex" "creep" "debris flow" "landslide" "mudslide" "other" "riverbank collapse" "rockfall" "rockslide" "snow avalanche" "unknown"
* Landslide size:  
  "large" "medium" "small" "very\_large"
* Trigger:   
  "construction" "continuous rain" "downpour" "earthquake" "flooding" "freeze thaw" "mining digging" "other" "rain" "snowfall snowmelt" "tropical cyclone" "unknown"

CHAPTER-12

# Probability Distribution Function: Normal Distribution

## Plotting Normal curves for population in the dataset.

dataset = read.csv("catalog.csv", header = TRUE)

population = dataset$population

meanPopulation = mean(population)

sdPopulation = sd(population)

# Plotting Normal Distribution

xrange = seq(-2500000, 2500000, length = 400)

yrange = dnorm(xrange, meanPopulation, sdPopulation)

plot(xrange, yrange, type = "l", lwd = 2, col = "red")

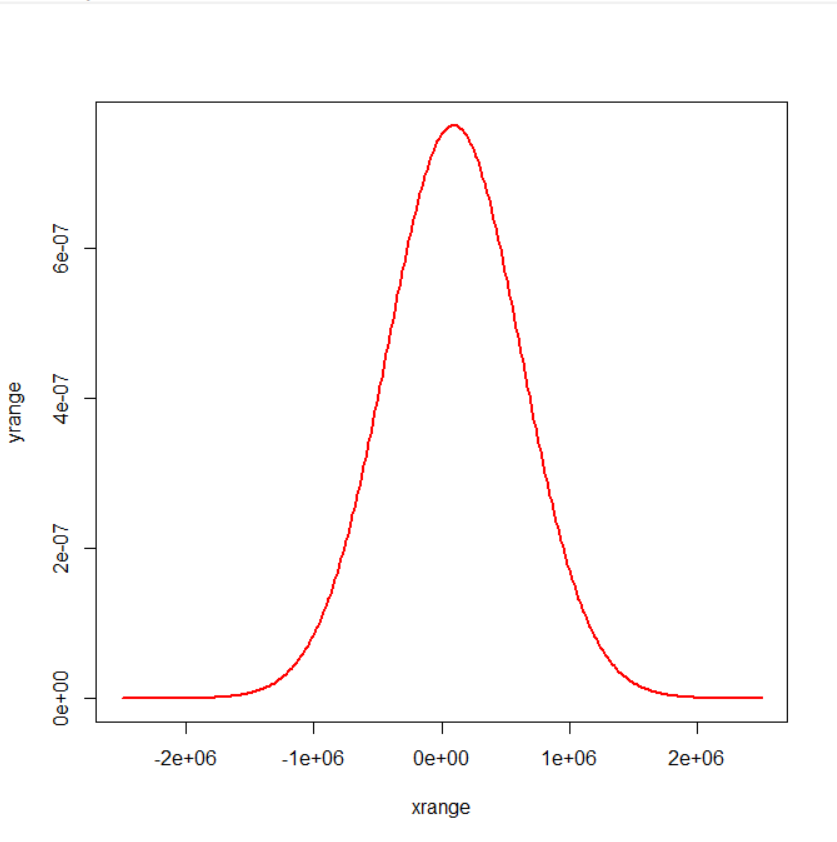


Figure 21: Normal Distribution

xrange = seq(-2500000, 2500000, length = 400)

yrange = pnorm(xrange, meanPopulation, sdPopulation)

plot(xrange, yrange, type = "l", lwd = 2, col = "red")

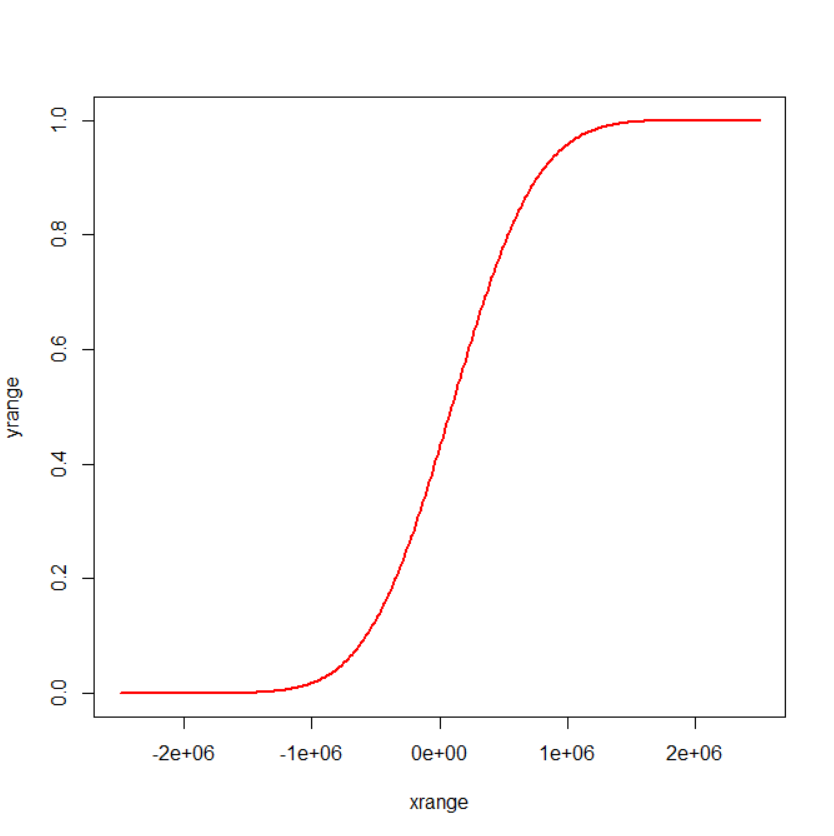


Figure 22: Normal Distribution

yrange = rnorm(nrows(dataset), meanPopulation, sdPopulation)

hist(yrange,type="l",lwd=2,col="red")

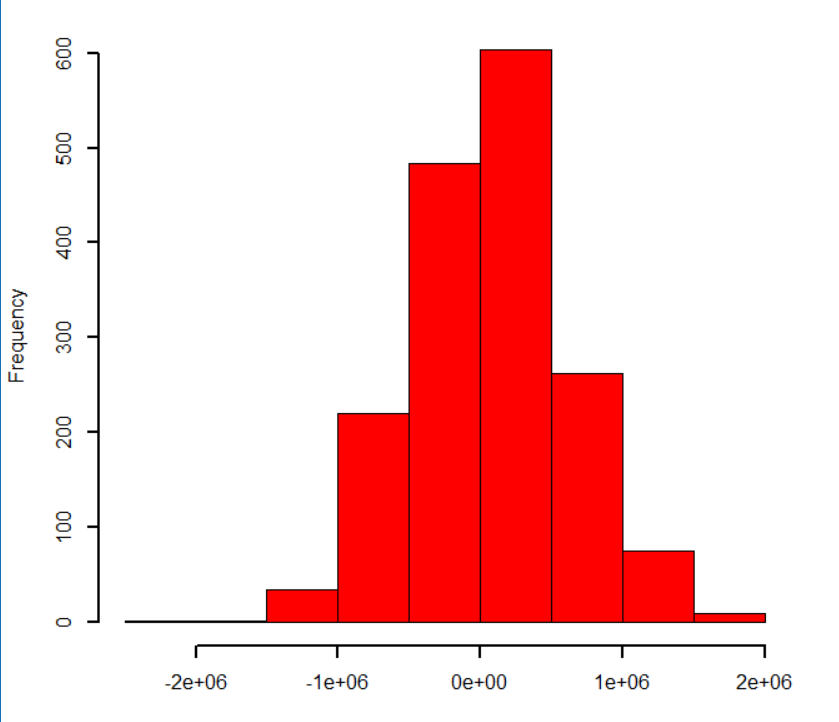


Figure 23: Normal Distribution

CHAPTER-13

# Hypothesis Testing

# Assume a news channel claims that, there is an average of 2 fatalities when landslides occur. At significance level 0.05,

# in a sample of 100 people, the mean fatalities turned out to be 0.7, and the standard deviation is 3.27.

# Is it possible to reject this claim, and tell that the average fatalities is less than 2?

# Solution :

# Null hypothesis : H0 = 2

# Alternate hypothesis : H1 < 2

dataset = read.csv("catalog.csv", header = TRUE)

fatalities = dataset$fatalities

claimedMean = 2

alpha = 0.05

n = 100

sd = 3.27

sampleMean = 0.7

# Calculating z-score

z = (mean - 2)/(sd/sqrt(n))

# Calculating p-value

p = pnorm(z, mean, sd, lower.tail = TRUE)

# Checking condition

if(p < alpha){

print("Rejected")

} else{

print("Accepted")

}

Output :



CHAPTER-14

# CONCLUSION

In this report we:

* Installed R and Rstudio.
* Used the dataset, Landslides after rainfall (2007 – 2016) to perform the following:
  + Exploratory data analysis using various functions.
  + Data visualization using basic r plots and ggplot.
  + Statistical analysis using basic functions.
  + Hypothesis testing.
* Observations and inferences were recorded.