VISUALCOVID DASHBOARD

Data Science & Big Data Analytics (15IT423E) REPORT

Submitted by

AYUSHMAN GUPTA (RA1711003010903)

SAUNDARYA PATHAK (RA1711003010855)

BANGARU MOHNISH (RA1711003011397)

ANINDA GHOSH (RA1711003010853)

Under the guidance of

Dr. R. S. Ponmagal

(Associate Professor, Department of Computer Science & Engineering)

in partial fulfilment for the aware of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

S.R.M. Nagar, Kattankulathur, Kanchipuram District

MAY 2020

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1965)

BONAFIDE CERTIFICATE

Certified that this project titled **VISUALCOVID DASHBOARD** is the bonafide work of **AYUSHMAN GUPTA (RA1711003010903), SAUNDARYA PATHAK (RA1711003010855), BANGARU MOHNISH (RA1711003011397), ANINDA GHOSH (RA1711003010853),** who carried out the work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other report.

SIGNATURE

Dr. R. S. Ponmagal
GUIDE
Associate Professor
Dept. of Computer Science & Engineering

SIGNATURE

Dr. B. AMUTHA
HEAD OF THE DEPARTMENT
Dept. of Computer Science &Engineering

Signature of the Internal Examiner

Signature of the External Examiner

ABSTRACT

The purpose behind the research involved in this project was that we wanted to create a simulator to visualize how the pandemic has spread across the world, crippling day-to-day activities, the economy, and the livelihoods of the people. A dashboard is hence the best way to visualize all the data and package it in a neat and tidy web application. Since the implementation had to be in R, we used the various tools available to us completely deploy a web application in R, from the frontend UI design to the backend functions responsible for the processing in the background and providing the data that the website would show.

The web application would thus provide key figures such as total confirmed cases, the number of estimated recoveries, the number of deceased cases, as well as the number of affected countries. It would be updated regularly to provide accuracy in accordance to the actual number of cases in the world.

It would also show tabular data for all the countries with various columns of data. It would also show a map with the countries marked which are affected with clear tooltips to visualize each country’s data from afar. The map will also have a simulation feature to visualize a simulation of the cases evolving in the various countries.

There will be various plots which will represent data such as evolution of the cases, new cases, cases per country.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. Sandeep Sancheti**, Vice Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. C. Muthamizhchelvan**, Director, Faculty of Engineering and Technology, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. B. Amutha**, Professor & Head, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for her valuable suggestions and encouragement throughout the period of the project work.

Our inexpressible respect and thanks to our guide and Data Science and Big Data Analytics faculty, **Dr. R.S.Ponmagal**, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for providing me an opportunity to pursue my project under her mentorship. She provided us the freedom and support to explore the research topics of my interest. Her passion for solving the real problems and making a difference in the world has always been inspiring.

We sincerely thank staff and students of the Computer Science and Engineering Department, SRM Institute of Science and Technology, for their help. Finally, we would like to thank my parents, our family members and our friends for their unconditional love, constant support and encouragement.

TABLE OF CONTENTS

S. No	Contents	Page No.
1.	Introduction	1
2.	Overview of Data Science and R	2
3.	R Packaged Used	11
4.	Implementation	14
5.	Screenshots	16
6.	Appendix	20

Chapter 1

INTRODUCTION

Data science is an exciting discipline that allows you to turn raw data into understanding, insight, and knowledge.

What is data science?

Data science is the study of data. It involves developing methods of recording, storing, and analysing data to effectively extract useful information. The goal of data science is to gain insights and knowledge from any type of data — both structured and unstructured.
Data science is related to computer science, but is a separate field. Computer science involves creating programs and algorithms to record and process data, while data science covers any type of data analysis, which may or may not use computers. Data science is more closely related to the mathematics field of Statistics, which includes the collection, organization, analysis, and presentation of data.

Because of the large amounts of data modern companies and organizations maintain, data science has become an integral part of IT. For example, a company that has petabytes of user data may use data science to develop effective ways to store, manage, and analyse the data. The company may use the scientific method to run tests and extract results that can provide meaningful insights about their users.

Chapter 2

OVERVIEW OF DATA SCIENCE AND R

What is R?

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing software and data analysis. Polls, data mining surveys, and studies of scholarly literature databases show substantial increases in popularity; as of February 2020, R ranks 13th in the TIOBE index, a measure of popularity of programming languages.

A GNU package, the official R software environment is written primarily in C, Fortran, and R itself (thus, it is partially self-hosting) and is freely available under the GNU General Public License. Pre-compiled executables are provided for various operating systems. Although R has a command line interface, there are several third-party graphical user interfaces, such as RStudio, an integrated development environment, and Jupyter, a notebook interface.

R is an implementation of the S programming language combined with lexical scoping semantics, inspired by Scheme. S was created by John Chambers in 1976, while at Bell Labs. There are some important differences, but much of the code written for S runs unaltered.

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is developed by the R Development Core Team (of which, as of August 2018, Chambers was a member). R is named partly after the first names of the first two R authors and partly as a play on the name of S. The project was conceived in 1992, with an initial version released in 1995 and a stable beta version (v1.0) on 29th February 2000.

COVID-19 Pandemic

The **COVID-19 pandemic**, also known as the **coronavirus pandemic**, is an ongoing pandemic of coronavirus disease 2019 (COVID-19), caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The outbreak was first identified in Wuhan, China, in December 2019. The World Health Organization declared the outbreak a Public Health Emergency of International Concern on 30 January, and a pandemic on 11 March. As of 21 May 2020, more than 5.07 million cases of COVID-19 have been reported in more than 188 countries and territories, resulting in more than 332,000 deaths. More than 1.93 million people have recovered from the virus.

The virus is primarily spread between people during close contact, most often via small droplets produced by coughing, sneezing, and talking. The droplets usually fall to the ground or onto surfaces rather than travelling through air over long distances. Less commonly, people may become infected by touching a contaminated surface and then touching their face. It is most contagious during the first three days after the onset of symptoms, although spread is possible before symptoms appear, and from people who do not show symptoms.

Common symptoms include fever, cough, fatigue, shortness of breath, and loss of smell. Complications may include pneumonia and acute respiratory distress syndrome. The time from exposure to onset of symptoms is typically around five days but may range from two to fourteen days. There is no known vaccine or specific antiviral treatment. Primary treatment is symptomatic and supportive therapy.

Recommended preventive measures include hand washing, covering one's mouth when coughing, maintaining distance from other people, wearing a face mask in public settings, and monitoring and self-isolation for people who suspect they are infected. Authorities worldwide have responded by implementing travel restrictions, lockdowns, workplace hazard controls, and facility closures. Many places have also worked to increase testing capacity and trace contacts of infected persons.

The pandemic has caused global social and economic disruption, including the largest global recession since the Great Depression. It has led to the postponement or cancellation of sporting, religious, political, and cultural events, widespread supply shortages exacerbated by panic buying, and decreased emissions of pollutants and greenhouse gases. Schools, universities, and

colleges are currently closed either on a nationwide or local basis in 185 countries, affecting approximately 98.5 percent of the world's student population. Misinformation about the virus has spread online, and there have been incidents of xenophobia and discrimination against Chinese people and against those perceived as being Chinese or as being from areas with high infection rates.

R Studio

RStudio is an integrated development environment (IDE) for R, a programming language for statistical computing and graphics. It is available in two formats: RStudio Desktop is a regular desktop application while RStudio Server runs on a remote server and allows accessing RStudio using a web browser. RStudio is a free, open source IDE (integrated development environment) for R. (You must install R before you can install RStudio.) Its interface is organized so that the user can clearly view graphs, data tables, R code, and output all at the same time. It also offers an Import-Wizard-like feature that allows users to import CSV, Excel, SAS (*.sas7bdat), SPSS (*.sav), and Stata (*.dta) files into R without having to write the code to do so.

The RStudio IDE is partly written in the C++ programming language and uses the Qt framework for its graphical user interface. The bigger percentage of the code is written in Java. JavaScript is also amongst the languages used.

Work on the RStudio IDE started around December 2010, and the first public beta version (v0.92) was officially announced in February 2011. Version 1.0 was released on 1 November 2016. Version 1.1 was released on 9 October 2017.

In April 2018, RStudio PBC (at the time RStudio, Inc.) announced that it will provide operational and infrastructure support to Ursa Labs in support of the Labs focus on building a new data science runtime powered by Apache Arrow.

In April 2019, RStudio PBC (at the time RStudio, Inc.) released a new product, the RStudio Job Launcher. The Job Launcher is an adjunct to RStudio Server. The launcher provides the ability to start processes within various batch processing systems (e.g. Slurm) and container

orchestration platforms (e.g. Kubernetes). This function is only available in RStudio Server Pro (fee-based application).

Types of plots in R

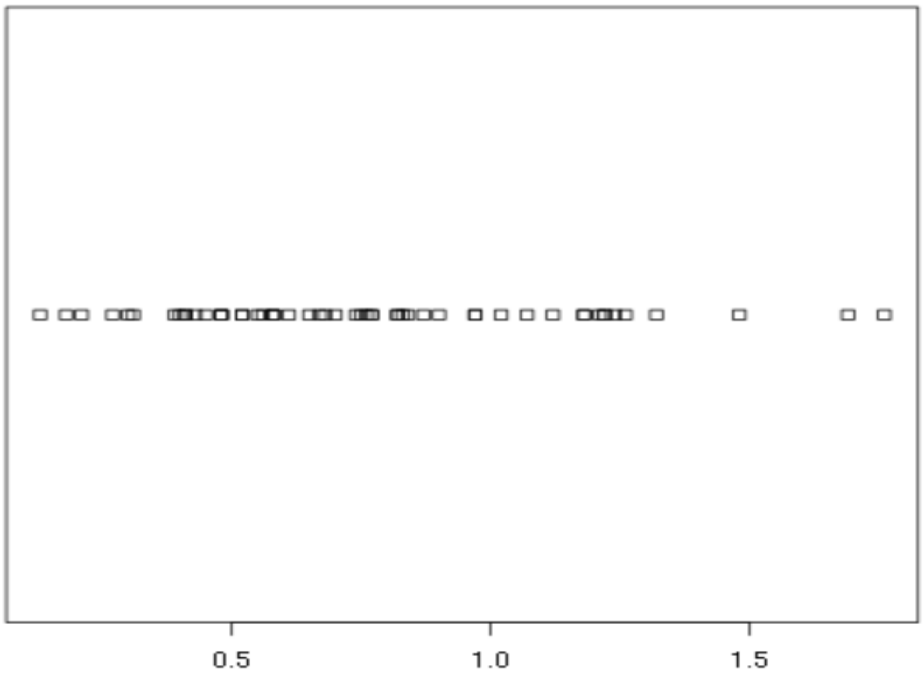
1. Strip Chart

A strip chart is the most basic type of plot available. It plots the data in order along a line with each data point represented as a box. Here we provide examples using the *w1* data frame mentioned at the top of this page, and the one column of the data is *w1$vals*.

To create a strip chart of this data, use the stripchart command:

> help(stripchart)

> stripchart(w1$vals)



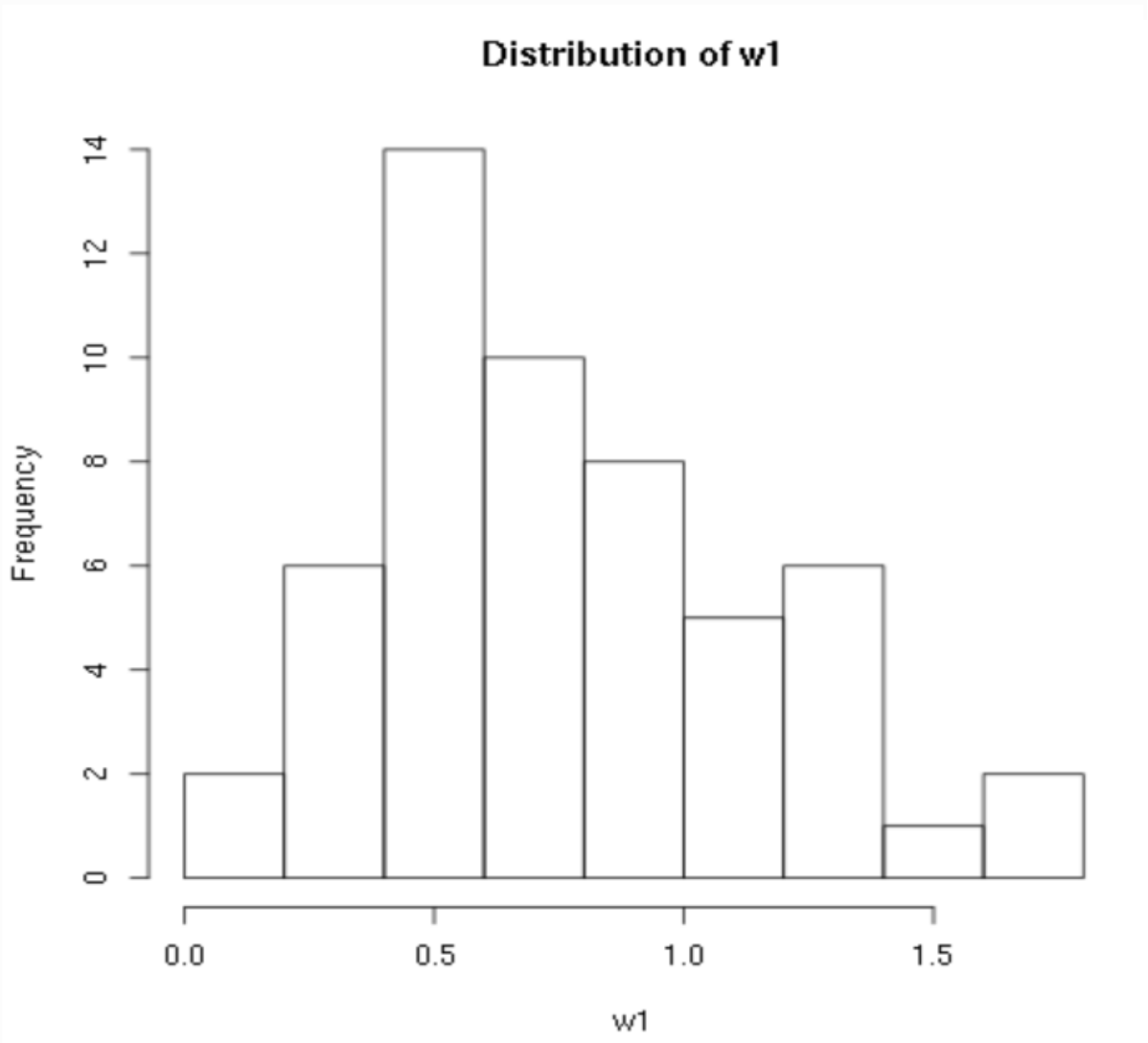
2. Histograms

A histogram is very common plot. It plots the frequencies that data appears within certain ranges. Here we provide examples using the *w1* data frame mentioned at the top of this page, and the one column of data is *w1$vals*.

To plot a histogram of the data, use the "hist" command:

> hist(w1$vals)

> hist (w1$vals, main="Distribution of w1", xlab="w1")

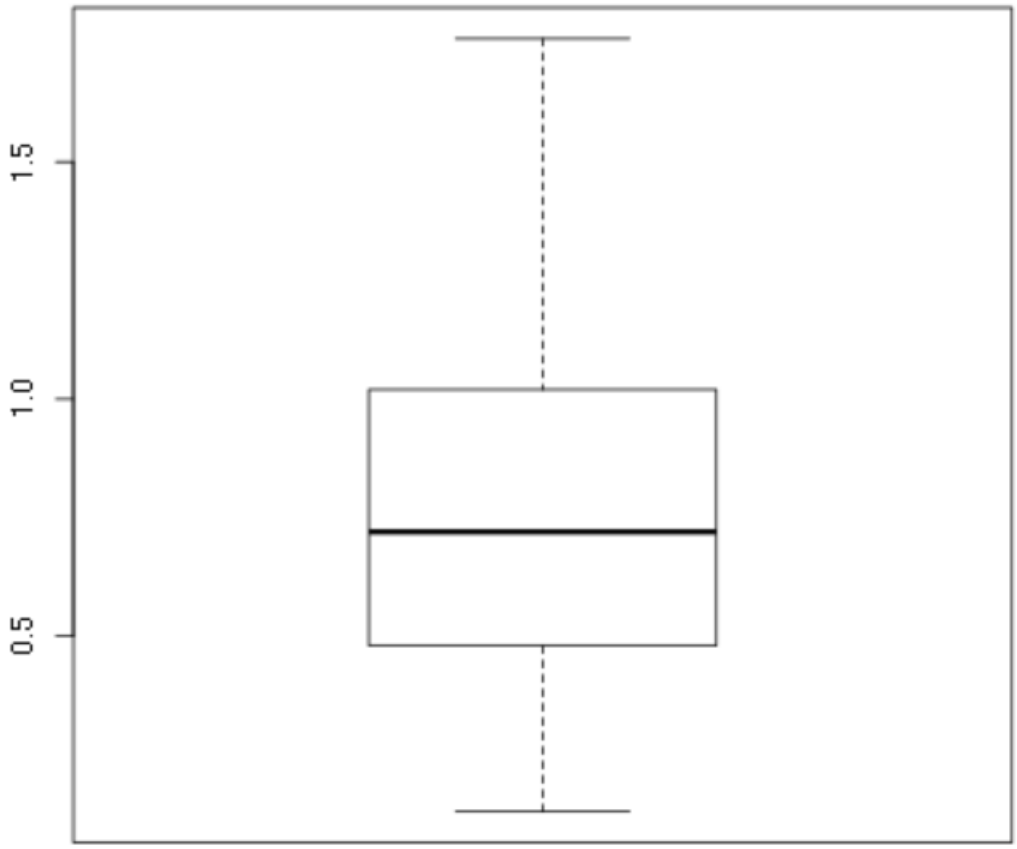


3. Box plots

A boxplot provides a graphical view of the median, quartiles, maximum, and minimum of a data set. Here we provide examples using two different data sets. The first is the *w1* data frame mentioned at the top of this page, and the one column of data is *w1$vals*.

We first use the *w1* data set and look at the boxplot of this data set:

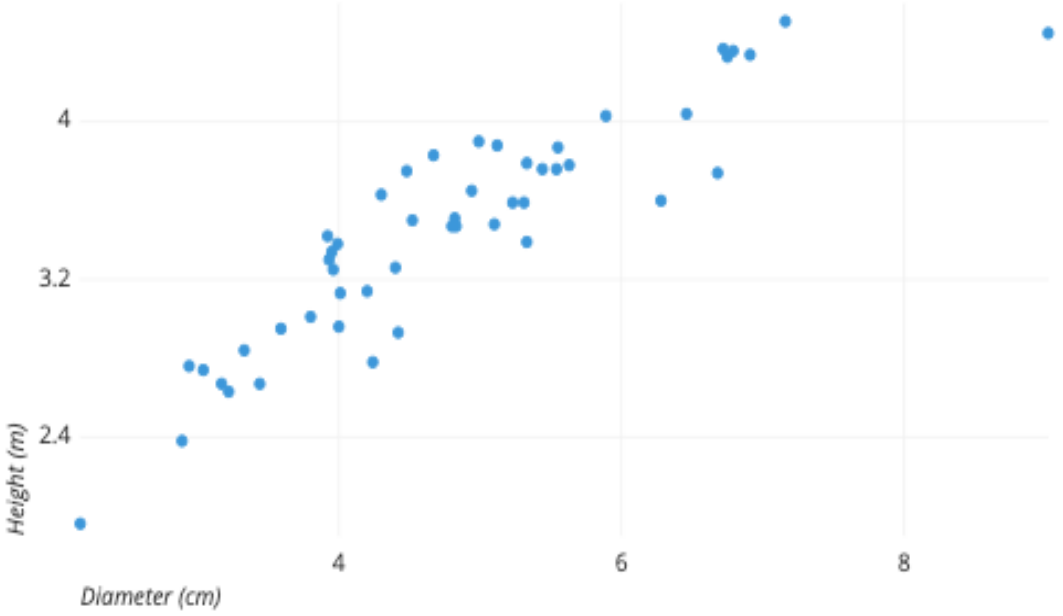
> boxplot(w1$vals)



4. Scatter Plots

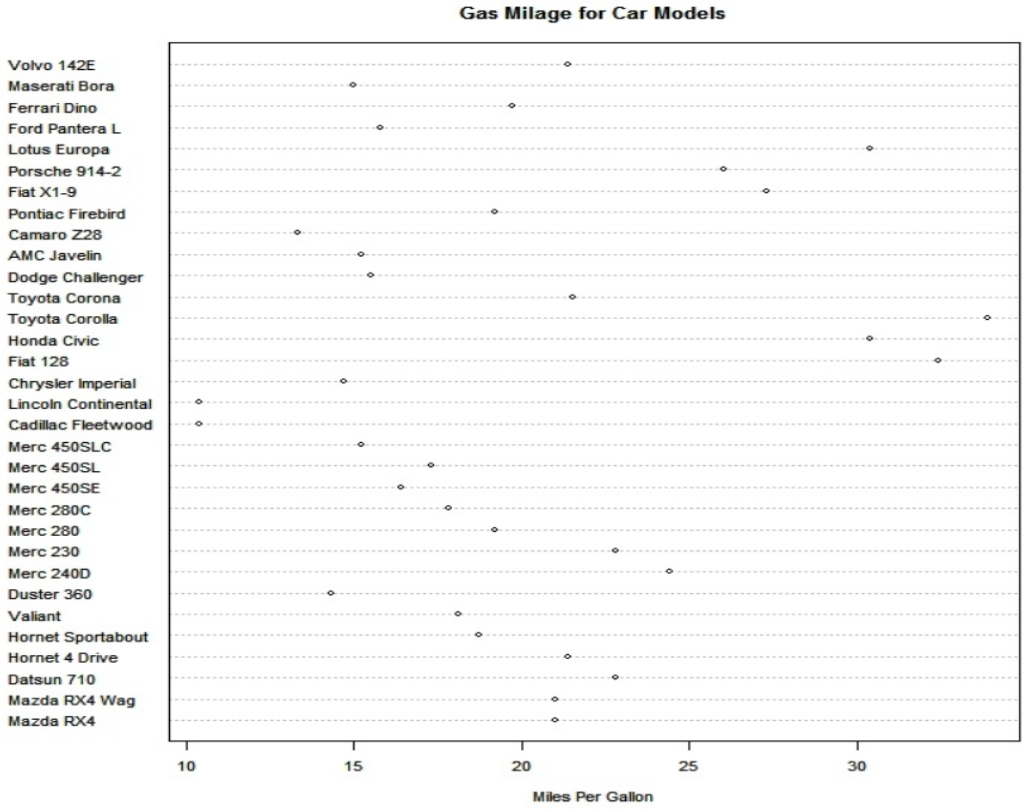
A scatter plot provides a graphical view of the relationship between two sets of numbers.

The command to plot each pair of points as an x-coordinate and a y-coordinate is “plot:”



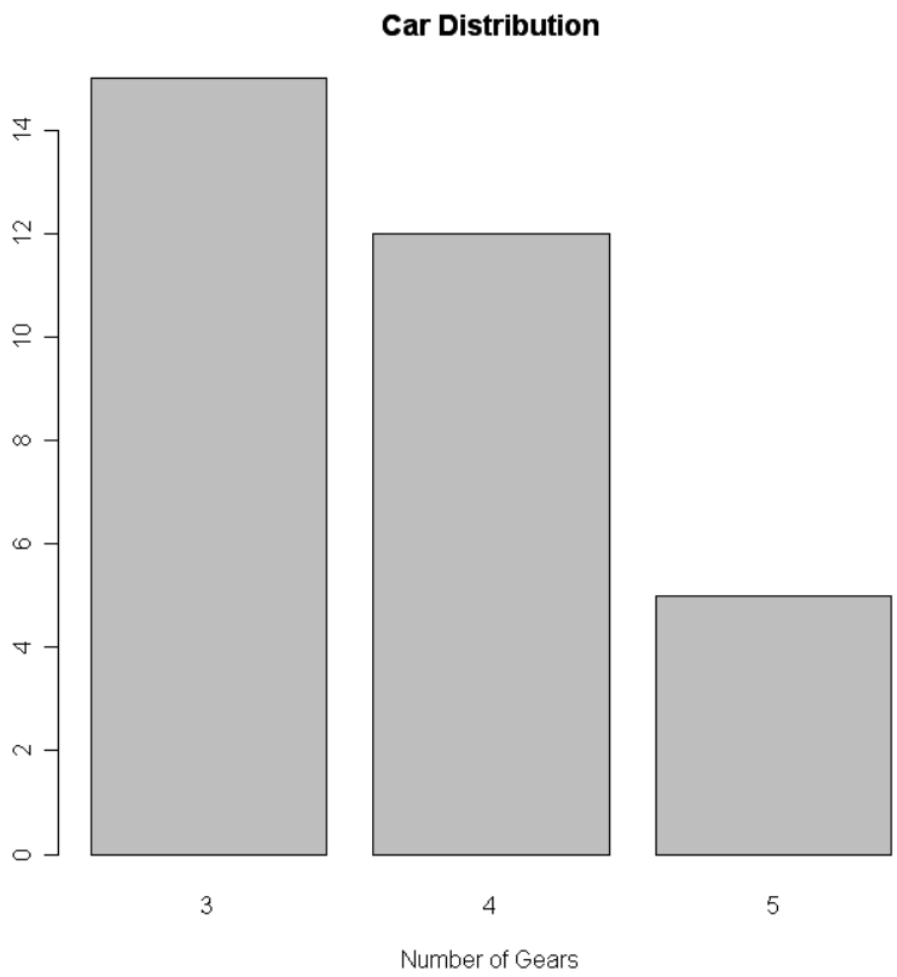
5. Dot Plots

Create dotplots with the **dotchart(*x*, labels=)** function, where *x* is a numeric vector and **labels** is a vector of labels for each point. You can add a **groups=** option to designate a factor specifying how the elements of *x* are grouped. If so, the option **gcolor=** controls the color of the groups label. **cex** controls the size of the labels.



6. Bar Plots

Create barplots with the **barplot(***height***)** function, where *height* is a vector or matrix. If **height is a vector**, the values determine the heights of the bars in the plot. If **height is a matrix** and the option **beside=FALSE** then each bar of the plot corresponds to a column of height, with the values in the column giving the heights of stacked "sub-bars". If **height is a matrix** and **beside=TRUE**, then the values in each column are juxtaposed rather than stacked. Include option **names.arg= (***character vector***)** to label the bars. The option **horiz=TRUE** to create a horizontal bar plot.


Car Distribution
0
2
4
6
8
10
12
14
3
4
5
Number of Gears

Chapter 3

R PACKAGES USED

Shiny

Shiny is an R package that makes it easy to build interactive web apps straight from R. You can host standalone apps on a webpage or embed them in R Markdown documents or build dashboards. You can also extend your Shiny apps with CSS themes, HTML widgets, and JavaScript actions. Shiny combines the computational power of R with the interactivity of the modern web.

Shiny Dashboard

R Shiny is an excellent tool for interactive data visualizations. However, fitting a large number of graphs onto just one Shiny page may prove to be a challenge. In our experience, virtually all projects with new KPIs being introduced along the way result in inadequate and not readable final reports.

Dashboards provide a solution. They allow the developer to intuitively structure their reports by breaking them down into sections, panels and tabs. This makes it much easier for the final user to navigate through the data.

shinydashboard is one available solution and provides decent features.

Leaflet

Leaflet is one of the most popular open-source JavaScript libraries for interactive maps. It is used by websites ranging from The New York Times and The Washington Post to GitHub and Flickr, as well as GIS specialists like OpenStreetMap, Mapbox, and CartoDB.

This R package makes it easy to integrate and control Leaflet maps in R.

Features

- Interactive panning/zooming
- Compose maps using arbitrary combinations of:
    - Map tiles
    - Markers
    - Polygons
    - Lines
    - Popups
    - GeoJSON
- Create maps right from the R console or RStudio
- Embed maps in knitr/R Markdown documents and Shiny apps
- Easily render spatial objects from the sp or sf packages, or data frames with latitude/longitude columns
- Use map bounds and mouse events to drive Shiny logic
- Display maps in non-spherical mercator projections
- Augment map features using chosen plugins from leaflet plugins repository

WBStats

The World Bank is a tremendous source of global socio-economic data; spanning several decades and dozens of topics, it has the potential to shed light on numerous global issues. To help provide access to this rich source of information, The World Bank themselves, provide a well-structured RESTful API. While this API is extremely useful for integration into web services and other high-level applications, it becomes quickly overwhelming for researchers who have neither the time nor the expertise to develop software to interface with the API. This leaves the researcher to rely on manual bulk downloads of spreadsheets of the data they are interested in. This too is can quickly become overwhelming, as the work is manual, time consuming, and not easily reproducible. The goal of the wbstats R-package is to provide a bridge between these alternatives and allow researchers to focus on their research questions and not the question of accessing the data.

Highlighted features of the **wbstats** R-package:

- Uses version 2 of the World Bank API that provides access to more indicators and metadata than the previous API version
- Access to all annual, quarterly, and monthly data available in the API
- Support for searching and downloading data in multiple languages
- Access to the World Bank Data Catalog Metadata, providing among other information; update schedules and supported languages

Plotly

Plotly's R graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, and 3D (WebGL based) charts.

Any graph made with the plotly R package is powered by the JavaScript library plotly.js. The plot_ly() function provides a 'direct' interface to plotly.js with some additional abstractions to help reduce typing. These abstractions, inspired by the Grammar of Graphics and ggplot2, make it much faster to iterate from one graphic to another, making it easier to discover interesting features in the data.

Chapter 4

Implementation

The implementation of the COVID-19 analysis application called **VisualCovid Dashboard** has been carried out keeping in mind the concepts of modularity and minimalist code, the modules have been split into categories of functionality such as UI and the Sections.

The User interface has been programmed in CSS using the R programming language. This is made possible with the Shiny Dashboard R package which provides the functionality to build HTML elements along with complete styling attributes in R itself without having to create a frontend in some other scripting language.

There are 4 modules for the UI:

1. UI (main)
2. UI_overview
3. UI_fulltable
4. UI_plots

- UI (main) is the main user interface file of the web app which binds all the other modules together for seamless functioning.
- UI_overview is the design of the User Interface on the overview screen which further contains sections for the backend implementation in R.
- UI_fulltable is the user interface created solely to display the table data for the COVID-19 pandemic
- UI_plots is the UI design to display the plots of the data from the table which further contains the section for the backend implementation in R.

The global.R has the backend functions. They pull the covid stats that are updated daily from a github repository that has updated data from John Hopkins University (https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data). There is also an update data function that pulls the latest data from the repository and updates the data tables in the project.

The backend of the web application has been coded purely in R making use of the mentioned libraries and the following modules:

1. Content_overview
2. Content_plots

The Content_overview model is the backend implementation of the UI_overview module and is split into further modules for all the individual features of the overview page:

1. keyFigures
2. Maps
3. Summary table

The key figures module is used to display the key values for information of the user without having to look any further such as deaths, recoveries, Infection count etc. We further used Shiny Dashboard to make the card layouts to display all the information. The key figures are dynamically updated at runtime.

The Maps module is the graphical representation of the data in the table sorted by geographical preference of the user which can be seen as a time series data as well using the slider and the play button on the overview page. The slider functionality is provided from leaflet itself making it easy for us to showcase how the cases have progressed in the world.

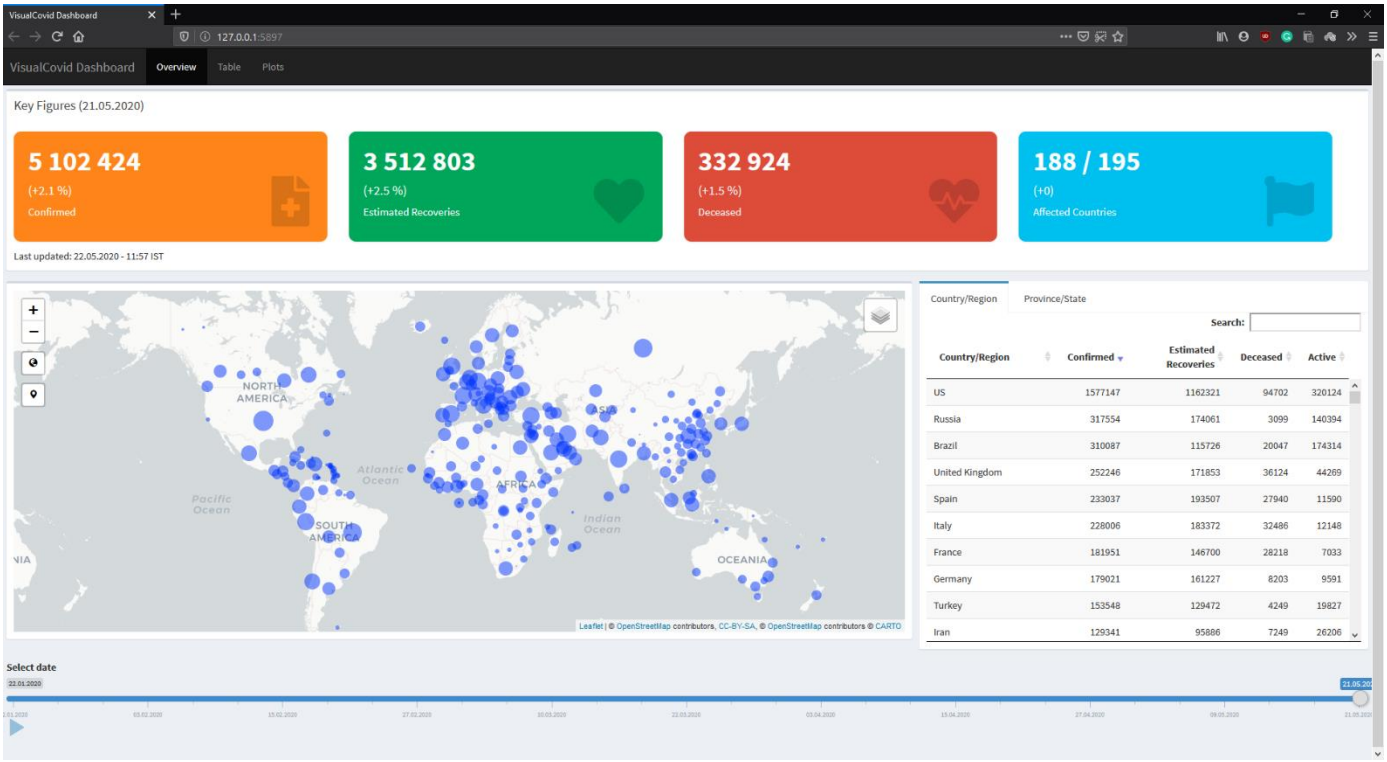
Summary table module is responsible for displaying the summary table of the data of infections, deaths, recoveries etc according to the geographical preference of the user.

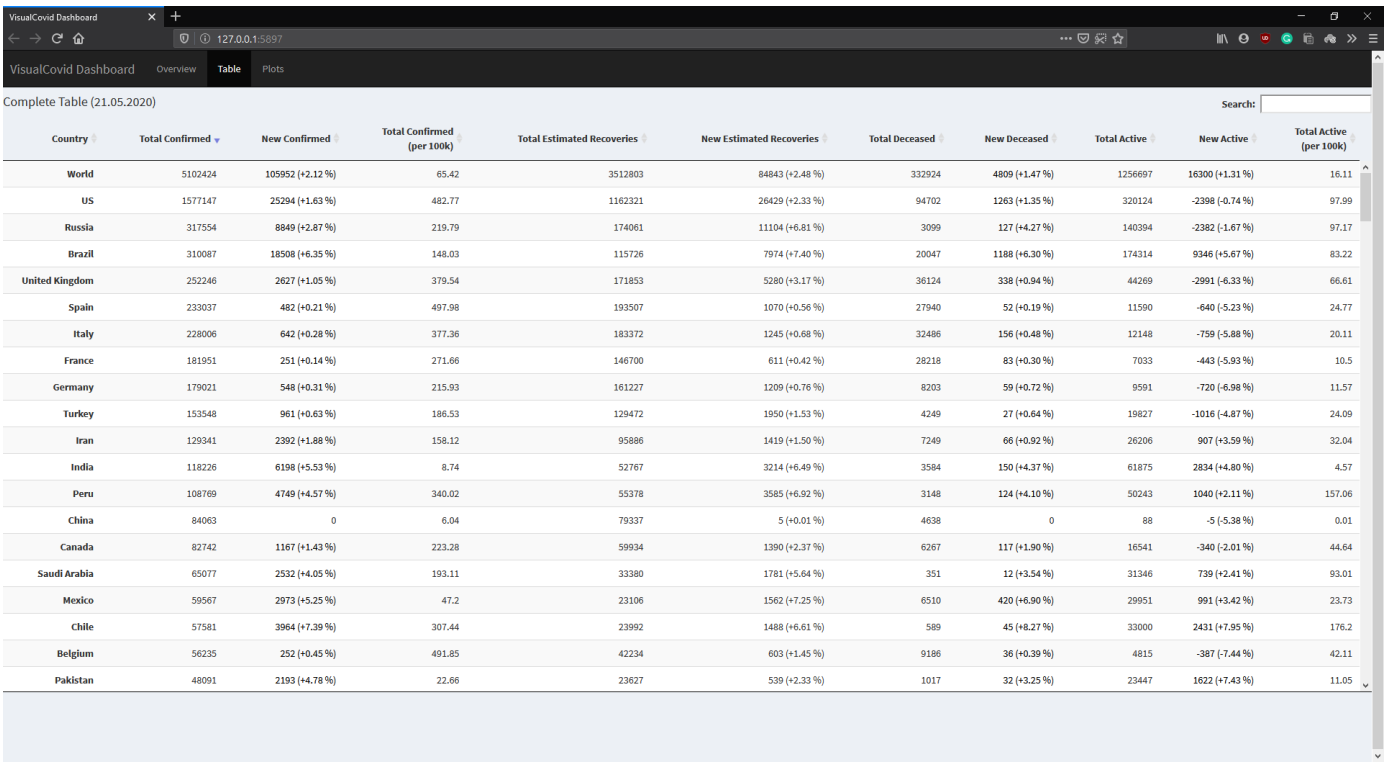
The Content_plots module is the backend implementation of the UI_plots module purely coded in R it is used for plotting the data onto graphs by using the scatter-pot technique by the means of lines and making the data easier to analyse. It also shows the time series data in the form of line charts for evolution of cases since outbreak, the new cases, etc.

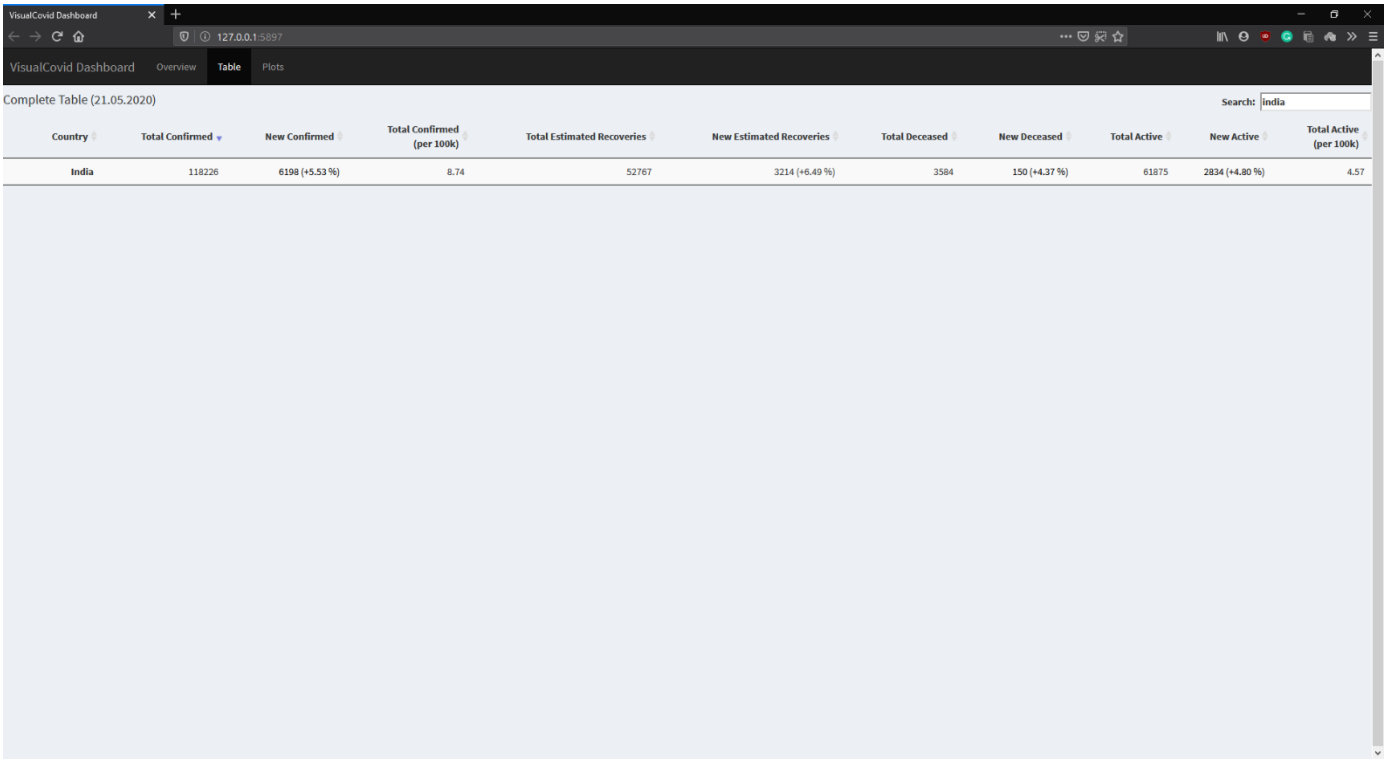
The time series data for each country further allows us to predict how the cases will increase or decrease moving forward provided that the time series is stationary and would be predicted using models such as ARIMA.

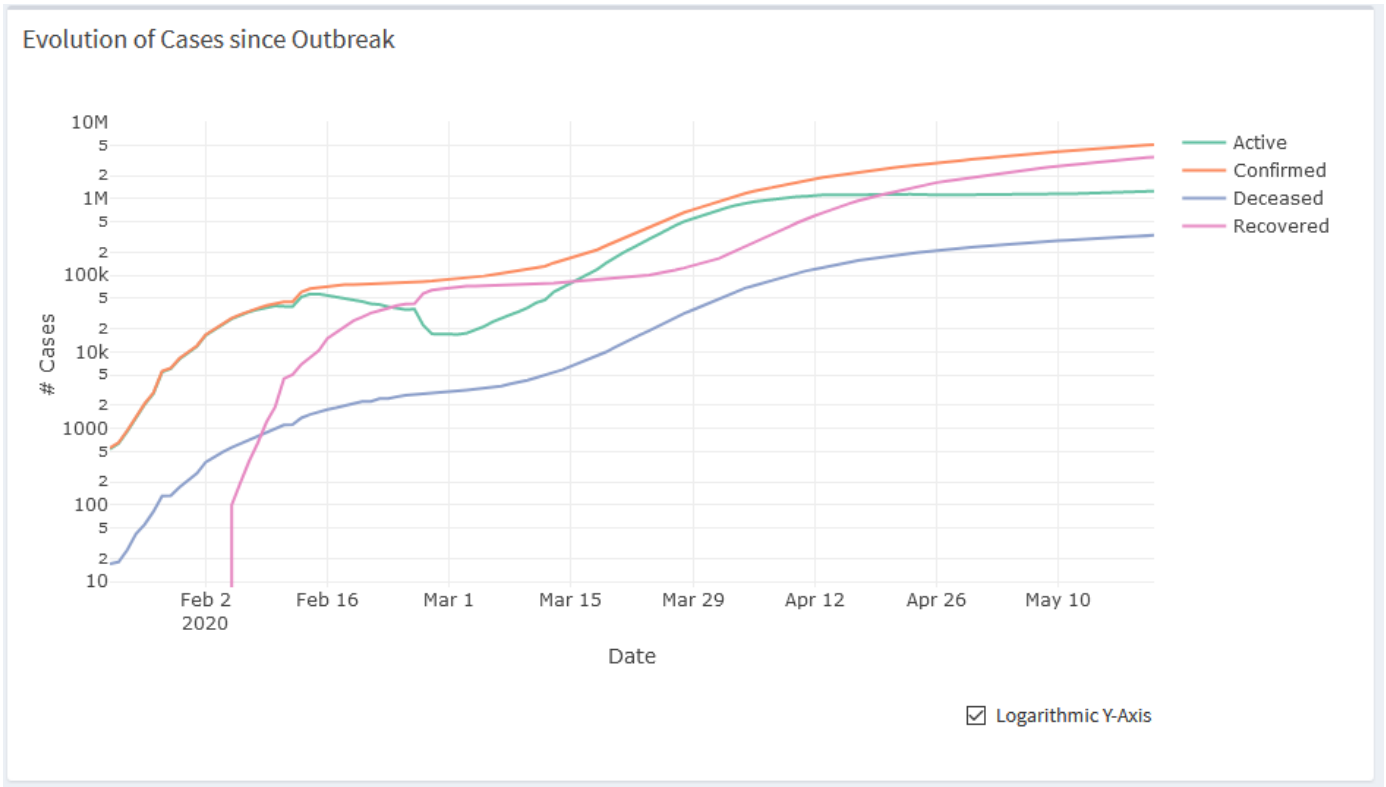
Chapter 5

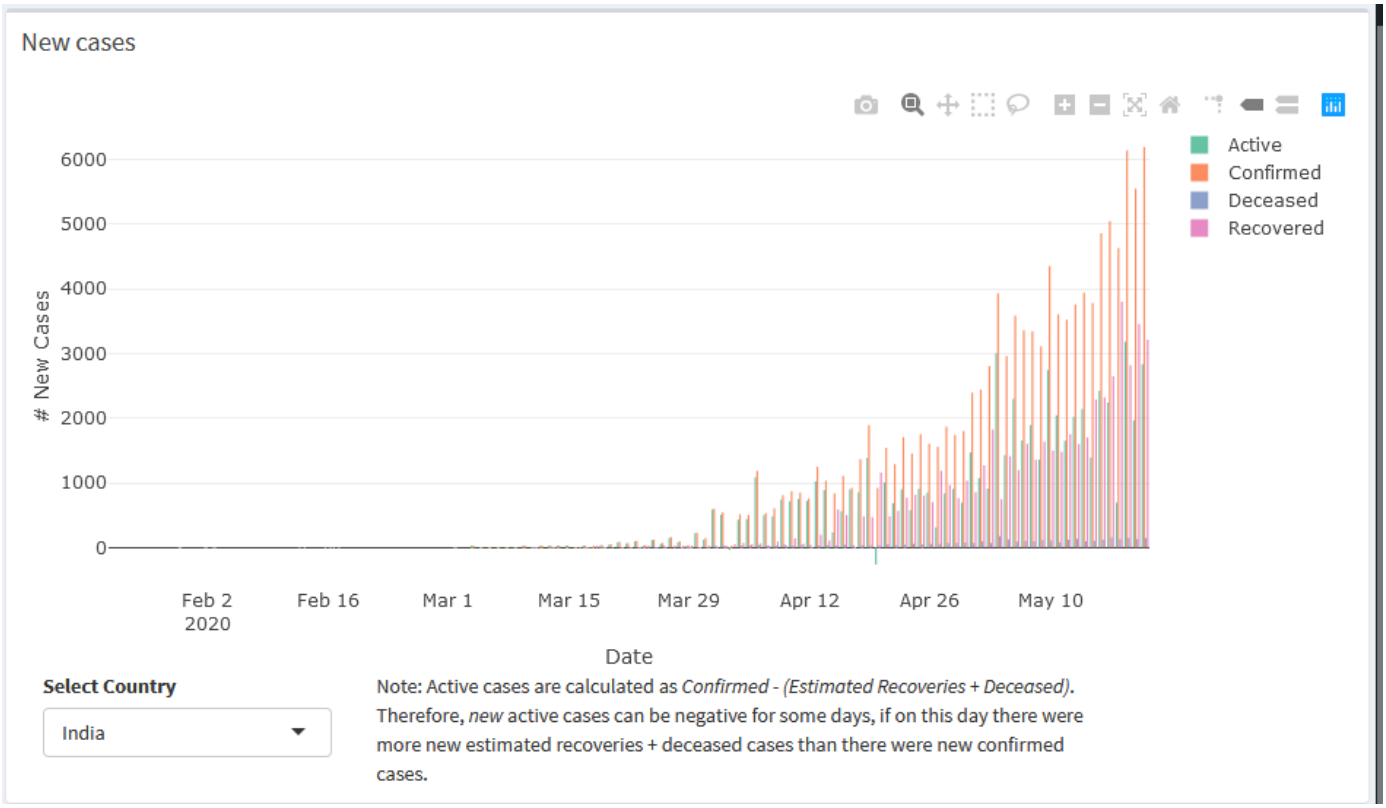
Screenshots



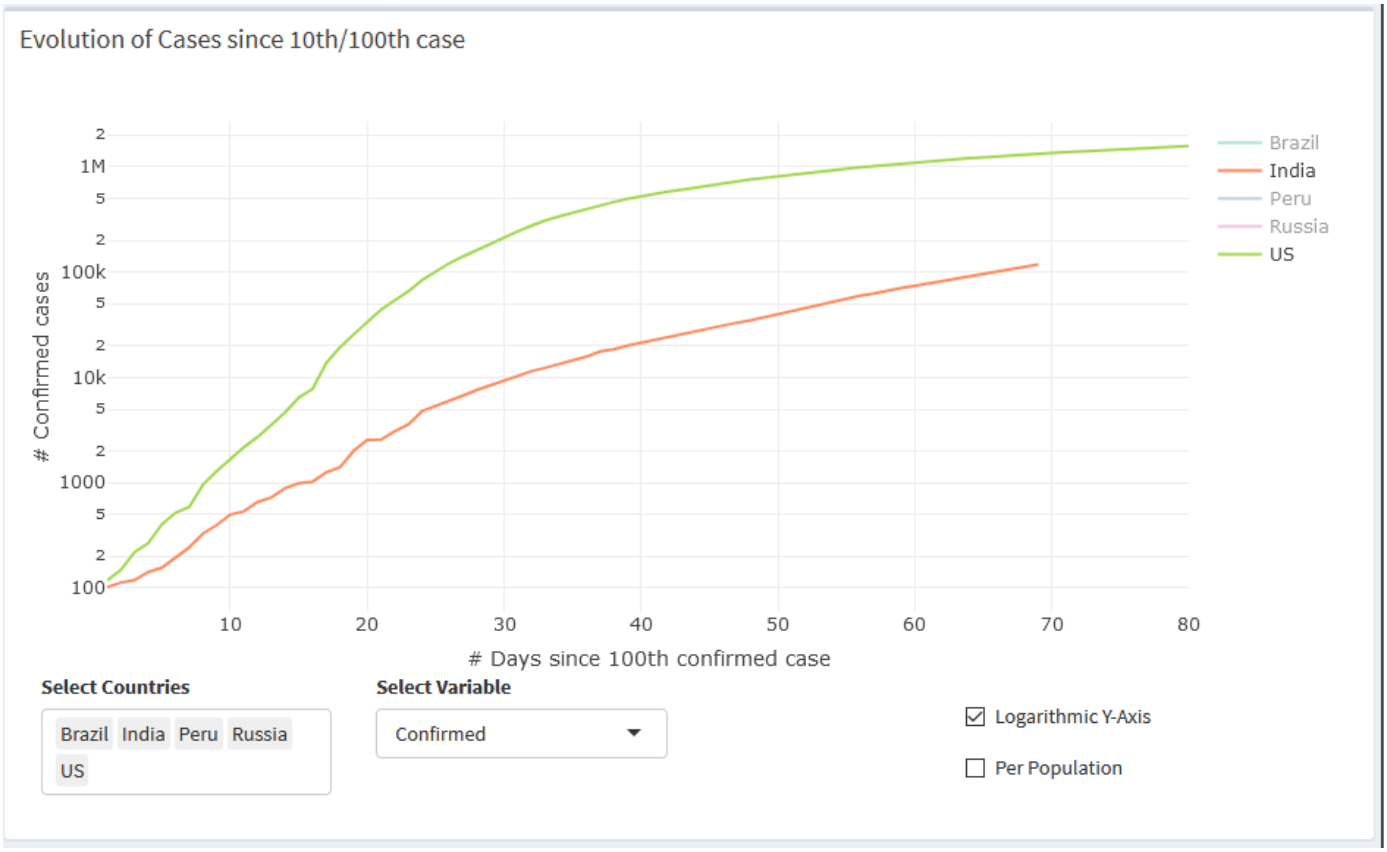



VisualCovid Dashboard
127.0.0.1:5897
VisualCovid Dashboard
Overview
Table
Plots
Complete Table (21.05.2020)
Search: india
Country
Total Confirmed
New Confirmed
Total Confirmed (per 100k)
Total Estimated Recoveries
New Estimated Recoveries
Total Deceased
New Deceased
Total Active
New Active
Total Active (per 100k)
India
118226
6198 (+5.53 %)
8.74
52767
3214 (+6.49 %)
3584
150 (+4.37 %)
61875
2834 (+4.80 %)
4.57


Evolution of Cases since Outbreak
Cases
10M
5
2
1M
5
2
100k
5
2
10k
5
2
1000
5
2
100
5
2
10
Feb 2
2020
Feb 16
Mar 1
Mar 15
Mar 29
Apr 12
Apr 26
May 10
Date
Active
Confirmed
Deceased
Recovered
Logarithmic Y-Axis


New cases
Active
Confirmed
Deceased
Recovered
New Cases
6000
5000
4000
3000
2000
1000
0
Feb 2
2020
Feb 16
Mar 1
Mar 15
Mar 29
Apr 12
Apr 26
May 10
Date
Select Country
India
Note: Active cases are calculated as *Confirmed - (Estimated Recoveries + Deceased)*. Therefore, *new* active cases can be negative for some days, if on this day there were more new estimated recoveries + deceased cases than there were new confirmed cases.


Cases per Country
Cases
1M
100k
10k
1000
100
10
1
Feb 2
2020
Feb 16
Mar 1
Mar 15
Mar 29
Apr 12
Apr 26
May 10
Date
Brazil
India
Peru
Russia
US
Confirmed
Estimated Recoveries
Deceased
Select Countries
Brazil
India
Peru
Russia
US
Logarithmic Y-Axis
Per Population


Evolution of Cases since 10th/100th case
Confirmed cases
2
1M
5
2
100k
5
2
10k
5
2
1000
5
2
100
10
20
30
40
50
60
70
80
Days since 100th confirmed case
Brazil
India
Peru
Russia
US
Select Countries
Brazil India Peru Russia US
Select Variable
Confirmed
Logarithmic Y-Axis
Per Population

Appendix

global.R

```
# Importing the necessary R packages

library(shiny)

library(shinydashboard)

library(DT)

library(fs)

library(wbstats)

library(leaflet)

library(plotly)

library(tidyverse)

source("utils.R", local = T)

# Reading the COVID data

downloadGithubData <- function() {

  download.file(

    url      = "https://github.com/CSSEGISandData/COVID-19/archive/master.zip",

    destfile = "data/covid19_data.zip"

  )

  data_path <- "COVID-19-
master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_"
```

```
  unzip(
    zipfile   = "data/covid19_data.zip",
    files     = paste0(data_path, c("confirmed_global.csv", "deaths_global.csv", "recovered_global.csv")),
    exdir     = "data",
    junkpaths = T
  )
}

# Download data from Johns Hopkins (https://github.com/CSSEGISandData/COVID-19) if the data is older than 1.5h
updateData <- function() {
  if (!dir_exists("data")) {
    dir.create('data')
    downloadGithubData()
  } else if ((!file.exists("data/covid19_data.zip")) || (as.double(Sys.time() - file_info("data/covid19_data.zip")$modification_time, units = "hours") > 0.5)) {
    downloadGithubData()
  }
}

# Update with start of app
updateData()
```

```
# Loading the datasets

data_confirmed <- read_csv("data/time_series_covid19_confirmed_global.csv")

data_recovered <- read_csv("data/time_series_covid19_recovered_global.csv")

data_deceased  <- read_csv("data/time_series_covid19_deaths_global.csv")

# Get latest data

current_date <- as.Date(names(data_confirmed)[ncol(data_confirmed)], format =
"%m/%d/%y")

changed_date <- file_info("data/covid19_data.zip")$change_time

# Evolution data by country

data_confirmed_sub <- data_confirmed %>%

                pivot_longer(names_to = 'date', cols = 5:ncol(data_confirmed)) %>%

                group_by(`Province/State`, `Country/Region`, date, Lat, Long) %>%

                summarise('confirmed' = sum(value, na.rm = T))

data_recovered_sub <- data_recovered %>%

                pivot_longer(names_to = "date", cols = 5:ncol(data_recovered)) %>%

                group_by(`Province/State`, `Country/Region`, date, Lat, Long) %>%

                summarise("recovered" = sum(value, na.rm = T))

data_deceased_sub <- data_deceased %>%

               pivot_longer(names_to = "date", cols = 5:ncol(data_deceased)) %>%

               group_by(`Province/State`, `Country/Region`, date, Lat, Long) %>%

               summarise("deceased" = sum(value, na.rm = T))
```

```
data_evolution <- data_confirmed_sub %>%
  full_join(data_deceased_sub) %>%
  ungroup() %>%
  mutate(date = as.Date(date, "%m/%d/%y")) %>%
  arrange(date) %>%
  group_by(`Province/State`, `Country/Region`, Lat, Long) %>%
  mutate(
    recovered = lag(confirmed, 14, default = 0) - deceased,
    recovered = ifelse(recovered > 0, recovered, 0),
    active = confirmed - recovered - deceased
  ) %>%
  pivot_longer(names_to = "var", cols = c(confirmed, recovered, deceased, active)) %>%
  ungroup()
# Calculating new cases
data_evolution <- data_evolution %>%
  group_by(`Province/State`, `Country/Region`) %>%
  mutate(value_new = value - lag(value, 4, default = 0)) %>%
  ungroup()
rm(data_confirmed, data_confirmed_sub, data_recovered, data_recovered_sub,
data_deceased, data_deceased_sub)
# Download population data from World Bank API in R
```

```
population <- wb(country = "countries_only", indicator = "SP.POP.TOTL", startdate = 2018, enddate = 2020) %>%

  select(country, value) %>%

  rename(population = value)

countryNamesPop <- c("Brunei Darussalam", "Congo, Dem. Rep.", "Congo, Rep.", "Czech Republic",

                     "Egypt, Arab Rep.", "Iran, Islamic Rep.", "Korea, Rep.", "St. Lucia", "West Bank and Gaza", "Russian Federation",

                     "Slovak Republic", "United States", "St. Vincent and the Grenadines", "Venezuela, RB")

countryNamesDat <- c("Brunei", "Congo (Kinshasa)", "Congo (Brazzaville)", "Czechia", "Egypt", "Iran", "Korea, South",

                      "Saint Lucia", "occupied Palestinian territory", "Russia", "Slovakia", "US", "Saint Vincent and the Grenadines", "Venezuela")

population[which(population$country %in% countryNamesPop), "country"] <- countryNamesDat

# Data from wikipedia

noDataCountries <- data.frame(

  country    = c("Cruise Ship", "Guadeloupe", "Guernsey", "Holy See", "Jersey", "Martinique", "Reunion", "Taiwan*"),

  population = c(3700, 395700, 63026, 800, 106800, 376480, 859959, 23780452)

)

population      <- bind_rows(population, noDataCountries)
```

```
data_evolution <- data_evolution %>%
  left_join(population, by = c("Country/Region" = "country"))
rm(population, countryNamesPop, countryNamesDat, noDataCountries)
# Up to date data
data_atDate <- function(inputDate) {
  data_evolution[which(data_evolution$date == inputDate),] %>%
    distinct() %>%
    pivot_wider(id_cols = c("Province/State", "Country/Region", "date", "Lat", "Long",
"population"), names_from = var, values_from = value) %>%
    filter(confirmed > 0 |
             recovered > 0 |
             deceased > 0 |
             active > 0)
}
data_latest <- data_atDate(max(data_evolution$date))
# Top 5 Country data based on active cases
top5_countries <- data_evolution %>%
  filter(var == "active", date == current_date) %>%
  group_by(`Country/Region`) %>%
  summarise(value = sum(value, na.rm = T)) %>%
  arrange(desc(value)) %>%
```

```
  top_n(5) %>%
  select(`Country/Region`) %>%
  pull()
```

ui.R

```
source("UI/ui_overview.R", local = TRUE)
source("UI/ui_plots.R", local = TRUE)
source("UI/ui_fullTable.R", local = TRUE)
ui <- fluidPage(
  title = "VisualCovid Dashboard",
  tags$head(
    tags$link(rel = "shortcut icon", type = "image/png", href = "logo.png")
  ),
  tags$style(type = "text/css", ".container-fluid {padding-left: 0px; padding-right: 0px
!important;}"),
  tags$style(type = "text/css", ".navbar { background-color: #212121; margin-bottom: 0px;
}"),
  tags$style(type = "text/css", ".content {padding: 0px;}"),
  tags$style(type = "text/css", ".row {margin-left: 0px; margin-right: 0px;}"),
  tags$style(HTML(".col-sm-12 { padding: 5px; margin-bottom: -15px; }")),
  tags$style(HTML(".col-sm-6 { padding: 5px; margin-bottom: -15px; }")),
  tags$style(HTML(".small-box { padding: 10px; border-radius: 10px; }")),
```

```
  tags$style(HTML(".small-box .icon-large { margin-right: 20px; }")),
  navbarPage(
    title       = div("VisualCovid Dashboard", style = "padding-left: 10px"),
    collapsible = TRUE,
    fluid       = TRUE,
    inverse     = TRUE,
    tabPanel("Overview", page_overview, value = "page-overview"),
    tabPanel("Table", page_fullTable, value = "page-fullTable"),
    tabPanel("Plots", page_plots, value = "page-plots"),
    tags$script(HTML("var header = $('.navbar > .container-fluid');")
    )
  )
)
```

server.R

```
server <- function(input, output) {
  sourceDirectory("sections", recursive = TRUE)
  showNotification("Welcome to the VisualCovid Stats Simulator",
                   duration = NULL, type = "warning")
  # Trigger once an hour
  dataLoadingTrigger <- reactiveTimer(3600000)
```

```
  observeEvent(dataLoadingTrigger, {
    updateData()
  })
  observe({
    data <- data_atDate(input$timeSlider)
  })
}
```

utils.R

```
capFirst <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(toupper(substring(s, 1, 1)), substring(s, 2),
        sep = "", collapse = " ")
}
# This function helps to source multiple files which are in the same directory. Just provide it with a path and all .R files in the directory it is
# pointed to will be sourced. Can be done recursively or not.
sourceDirectory <- function(path, recursive = FALSE, local = TRUE) {
  if (!dir.exists(path)) {
    warning(paste(path, "is not a valid directory!"))
    return(NULL)
  }
```

```
  # Source it where function is called (local)
  if (is.logical(local) && local) { env <- parent.frame() }
  # Source it in global environment
  else if (is.logical(local) && !local) { env <- globalenv() }
  # Source it in defined environment
  else if (is.environment(local)) { env <- local }
  else { stop("'local' must be TRUE, FALSE or an environment") }
  files <- list.files(path = path, pattern = ".*\\.R", all.files = F, full.names = TRUE, recursive =
recursive)
  for (fileToSource in files) {
    tryCatch(
      {
        source(fileToSource, local = env)
        cat(fileToSource, "sourced.\n")
      },
      error = function(cond) {
        message("Loading of file \"", fileToSource, "\" failed.")
        message(cond)
      }
    )
  }}
```