Configuration Manual

Global Warming and Natural Disasters to Global Peace Index

Wakako O'Sullivan
Student ID: 17143951

School of Computing
National College of Ireland

Supervisor: Dr Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Wakako O'Sullivan
Student ID:	17143951
Programme:	Data Analytics **Year:** 2020
Module:	MSc Research Project (Top Up)
Lecturer:	Dr Catherine Mulwa
Submission Due Date:	28th September 2020
Project Title:	Global Warming and Natural Disasters to Global Peace Index
Word Count:	6143 **Page Count:** 77

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Wakako O'Sullivan

Date: 25th September 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Global Warming and Natural Disasters to Global Peace Index

Wakako O'Sullivan
Student ID: 17143951

1 Introduction

This document presents as follows: chapter 2. hardware specifications, chapter 3. software installations, chapter 4. programming for implementation of data collection, preparation and implementation, and chapter 5. result evaluations.

2 Hardware Specification

Figure 1 presents hardware Specifications.

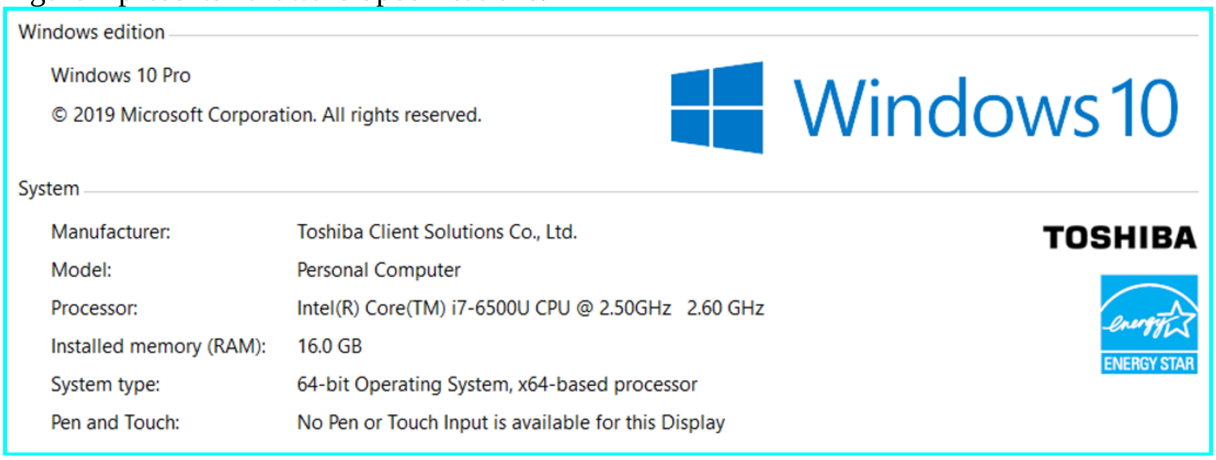


Figure 1: Hardware Specifications

3 Software Information and Installation

This section presents the common software and installation part of software which were used for implementation and result evaluation. The installation of R, RStudio and MySQL are presented in subsection 3.2 and 3.3.

3.1 Common Software

Microsoft Word was used for writing all documents. Microsoft Excel was used for data checking, handling data and creation of process flow diagram (Technical report Fig4). And Tableau used for creating the chart for presentation (Technical report Fig 20 & 21, and Fig 181 to 184).

3.2 R, R studio and Packages

<u>R version 3.5.1 software:</u>

Figure 2 presents the download site of R version 3.5.1.

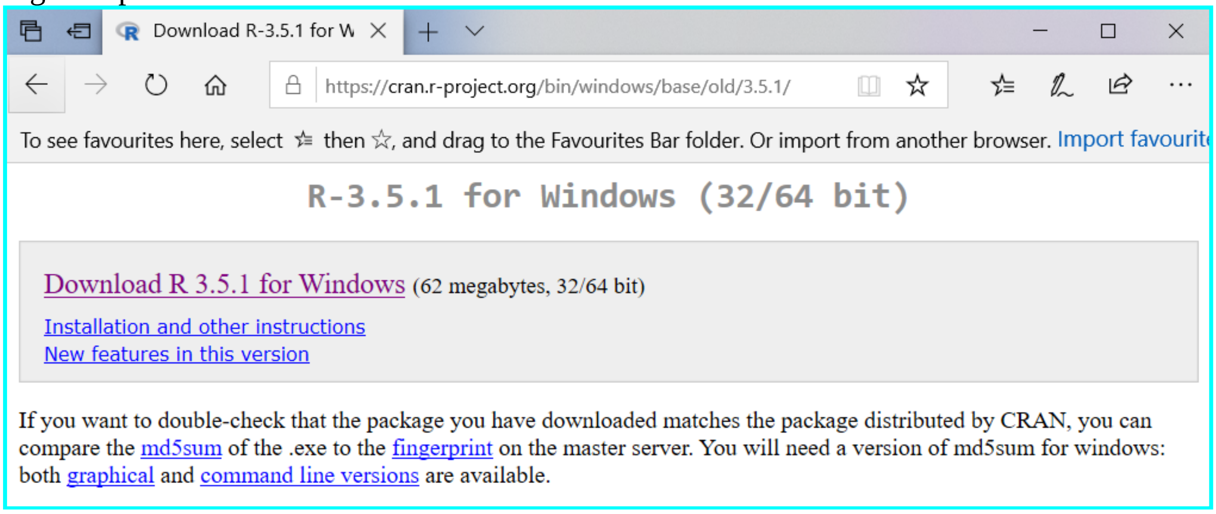


Figure 2: R version 3.5.1 download site

Figure 3 presents the language settings. English is selected.

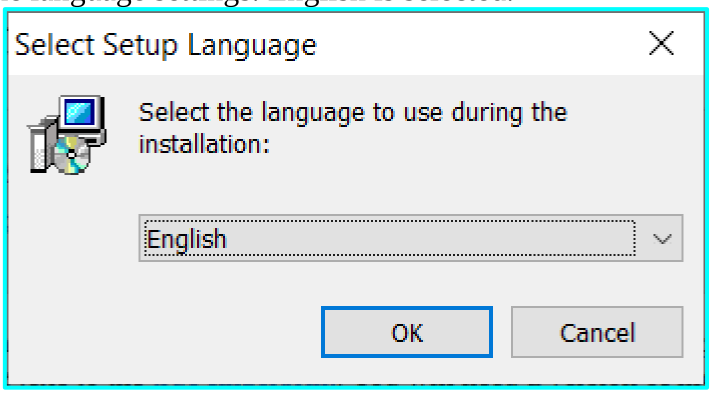


Figure 3: R 3.5.1 Installation language settings

Figure 4 to 7 present from starting installation until complete.

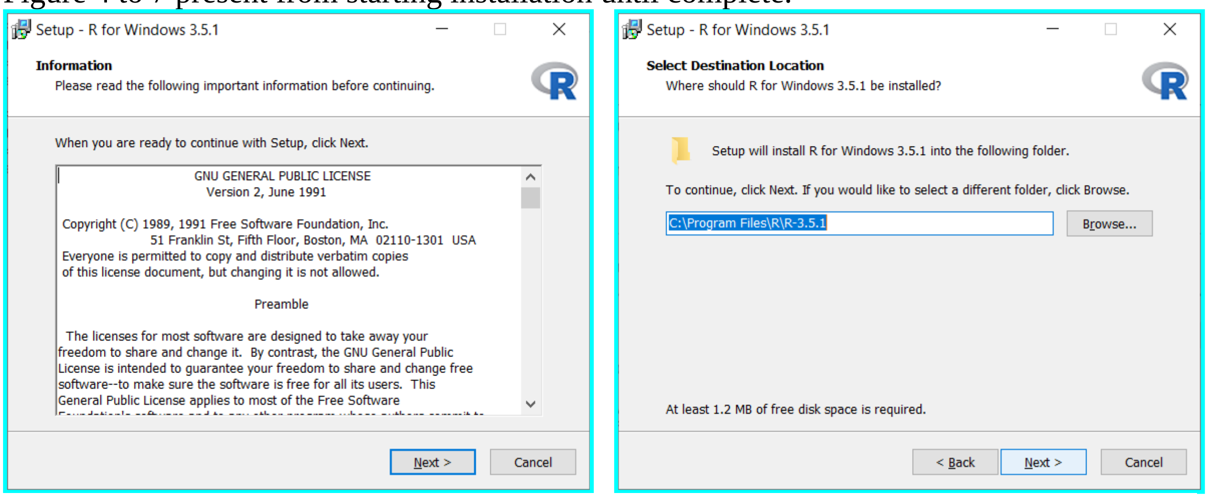


Figure 4: Stat Installation and set installation path

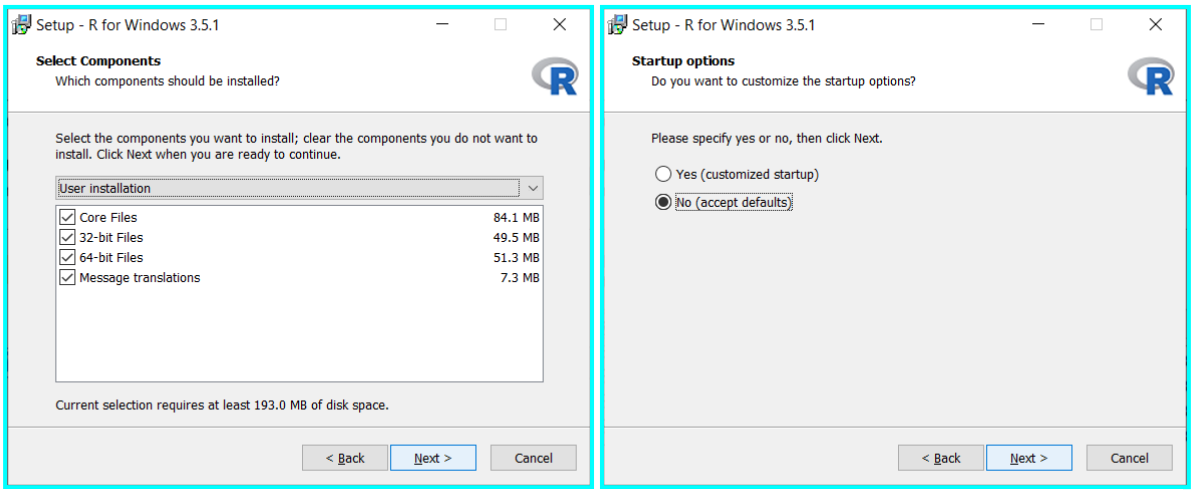


Figure 5: Select component and set default option

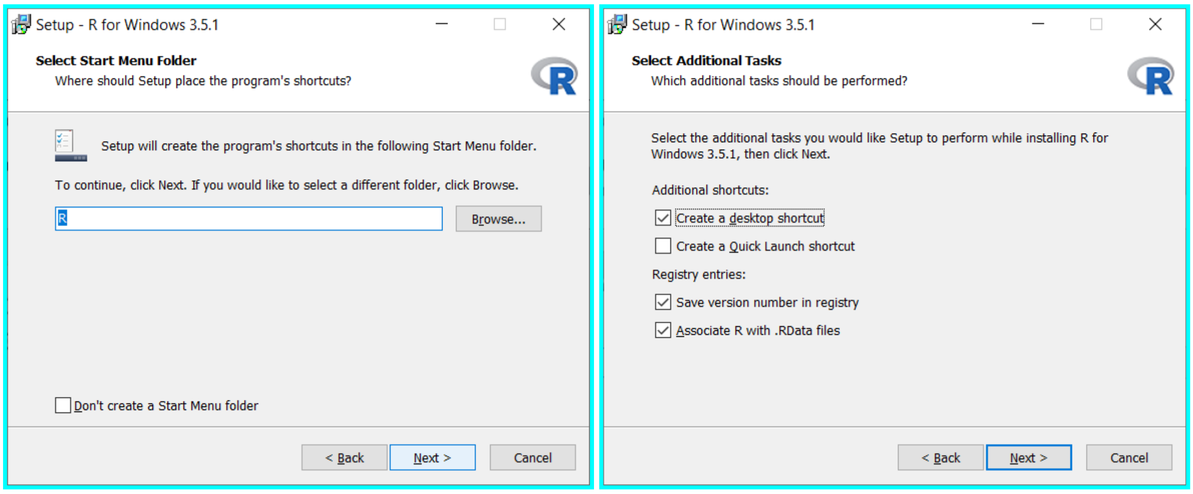


Figure 6: Set Start Menu Name and Select additional tasks

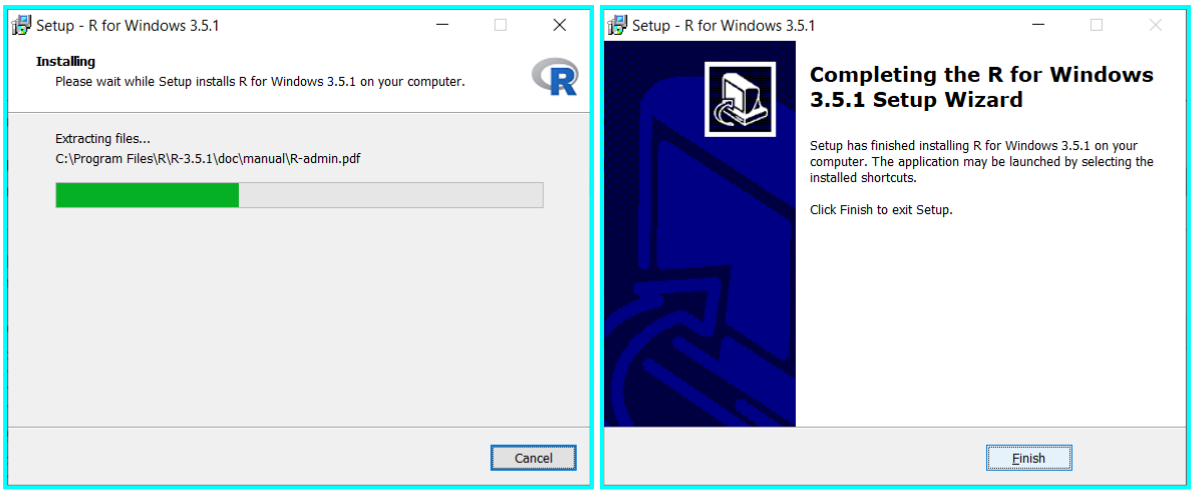


Figure 7: Start Installation and Complete

RStudio:

Figure 8 and 9 present download sites of RStudio version 1.3.959.

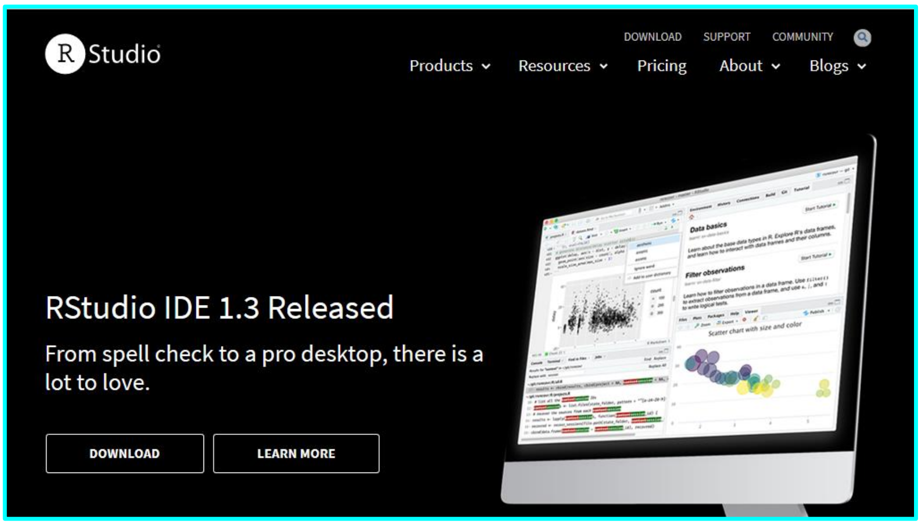


Figure 8: RStudio Download site

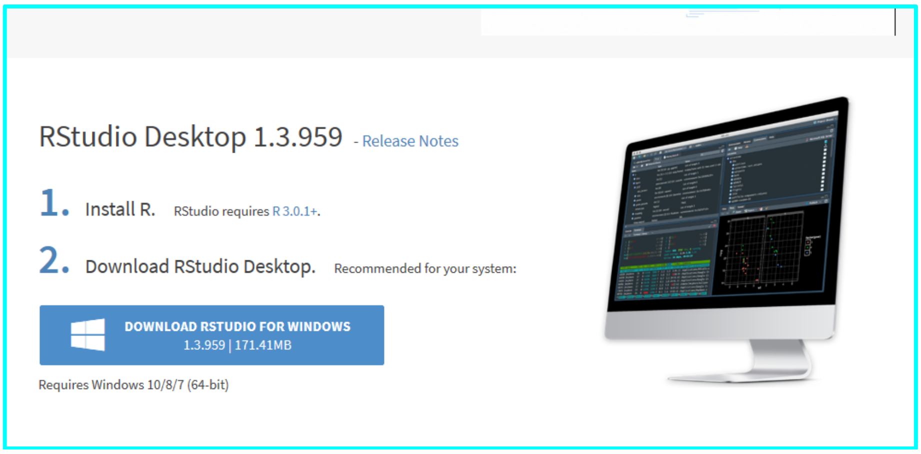


Figure 9: Version 1.3.959 Download site

Figure 10 to 13 present installation of RStudio.

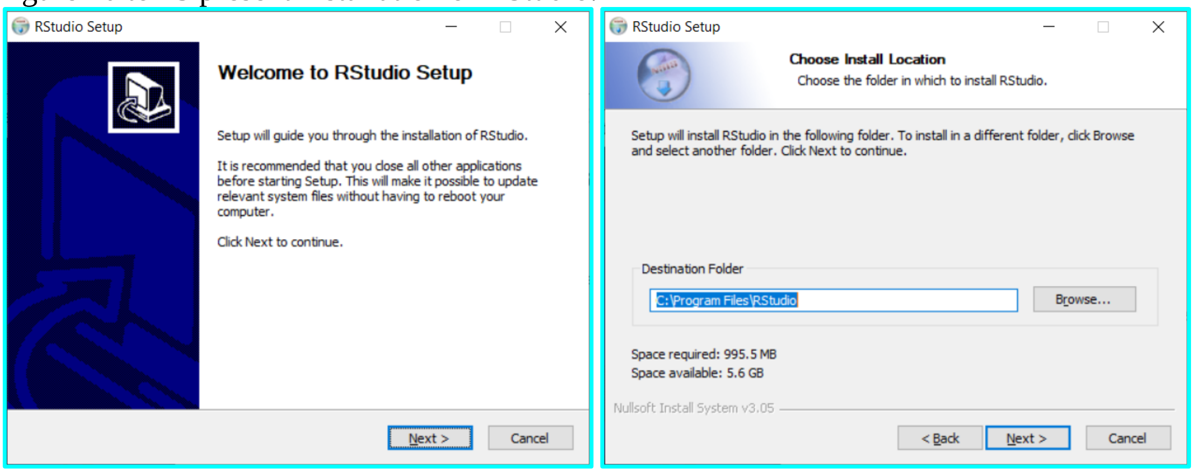


Figure 10: RStudio Start Downloading and Set Installation Path

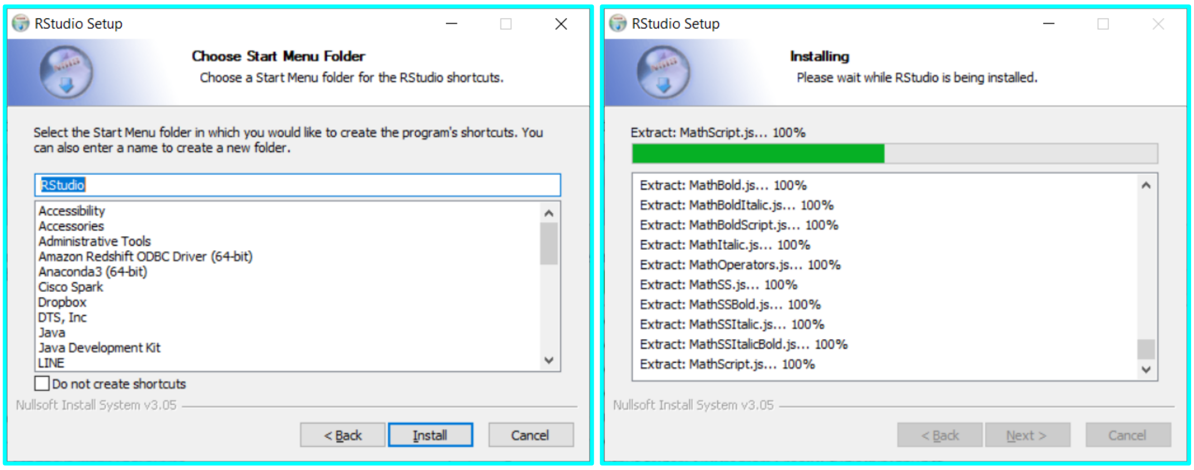


Figure 11: Set Menu Name and Start Installation

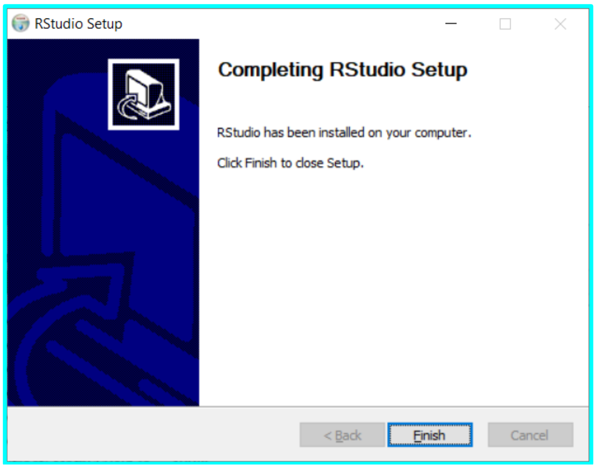


Figure 12: Installation Complete

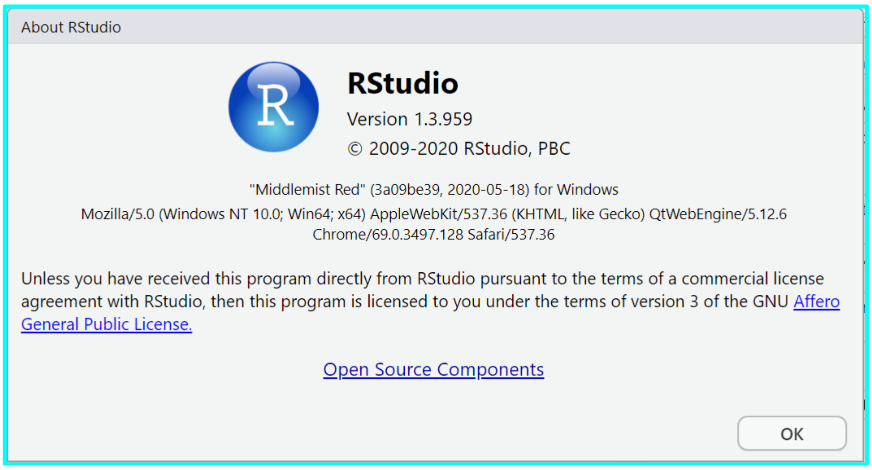


Figure 13: Check the version of RStudio

R Packages:

Table 1 presents name of R packages, function names and usage. The package is downloaded when the necessary function is used, and it is used by calling it in the program.

Table 1: Package used in R

Package name	Function	Usage
tabulizer	extract tables	Scrape pdf data from site
dplyr	rbind_all	Bind row data
	aggregate	Obtain average
xml2	read_html	Read html
rvest	html_nodes	Read html node
stringr	str_subset	Read subset in html
69+.table	transpose	Transpose data frame
mice	mice	Random Forest
outliers	rm.outlier	Set median on the outliers
stats	shapiro.test	Shapiro-Wilk Normality Test
	lm	Fitting Linear Models
	step	Choose a model by AIC in a Stepwise Algorithm
	predict	Model predictions
GGally	ggpairs	generalized pairs plot – Correlation check
caret	createDataPartition	Data splitting
	train	Model prediction over different parameters
	confusionMatrix	Create a confusion mat
e1071		Used with caret package
ggplot2		Graphics
rlang		Used by ggplot2
recipes		For design metrics
base	prop.table	Express table entries as a percentage of peripheral entries
	sapply	Missing data check
kernlab		For Support Vector Machine
randomForest		For Random Forest
mboost		For Boosting
klaR		For Naïve Bayes
party		For Conditional Inference Tree
pROC	roc	ROC curve visualisation

3.3 MySQL

Figure 14 presents MySQL Community Downloads site.

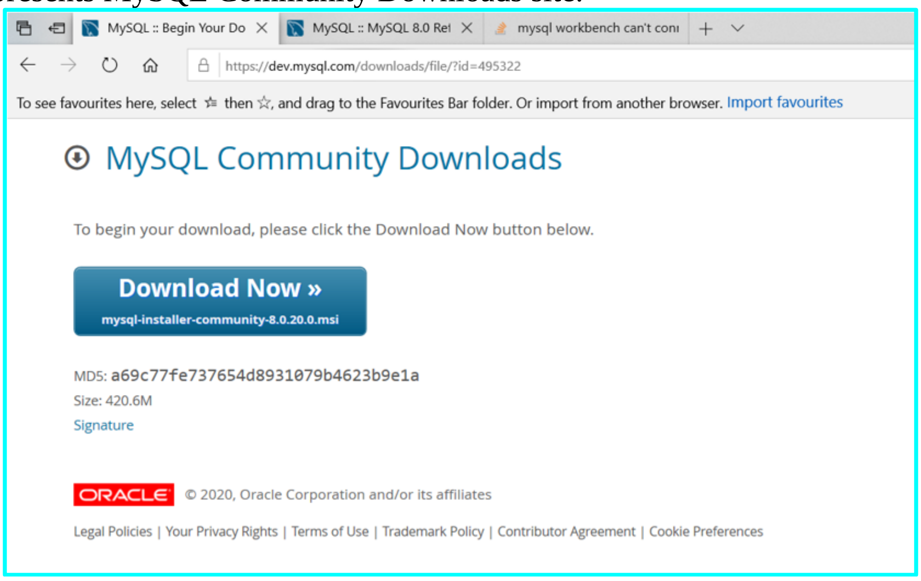


Figure 14: MySQL download site

Figure 15 presents, installation start.

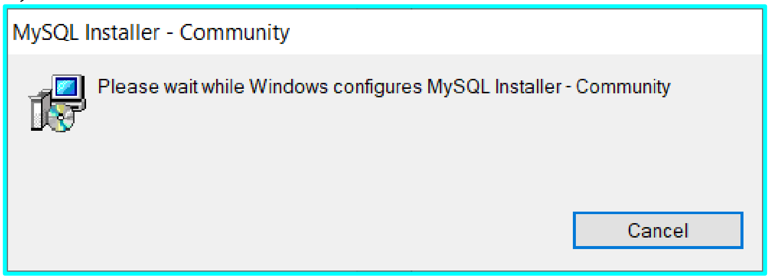


Figure 15: Start Installation pop up

Figure 16 presents Installer closure settings.

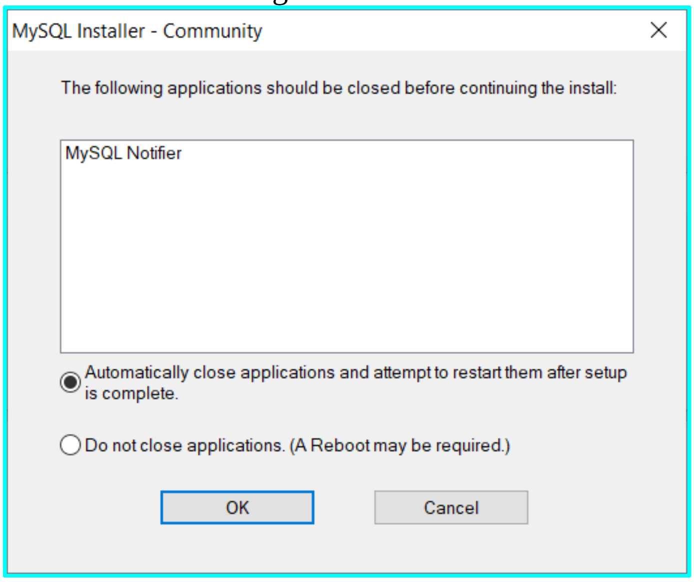


Figure 16: Closure Settings

From Figure 17 to 29 present all process of MySQL version 5.7 installation.

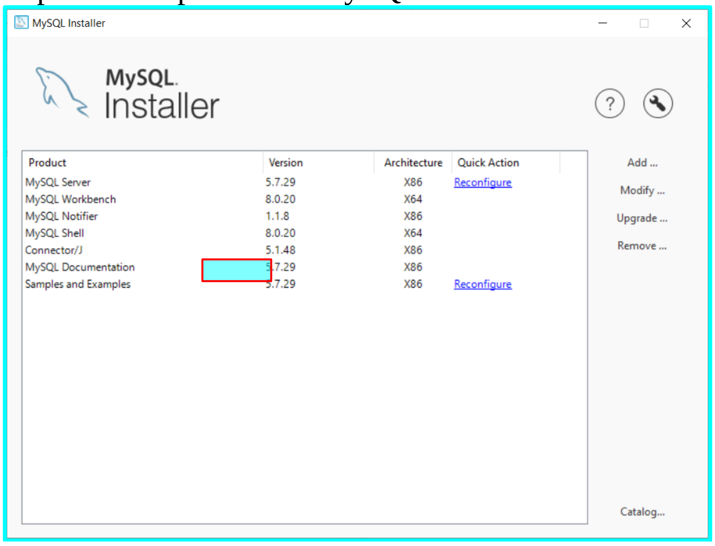


Figure 17: Installation of MySQL 5.7

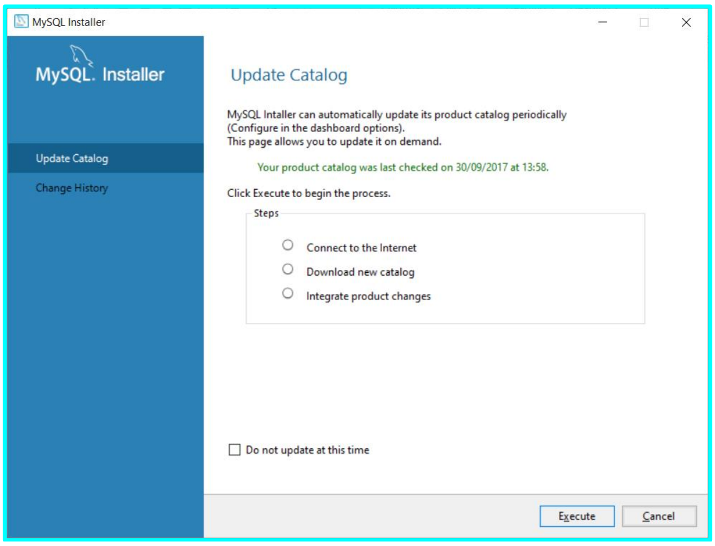


Figure 18: Installation of MySQL 5.7

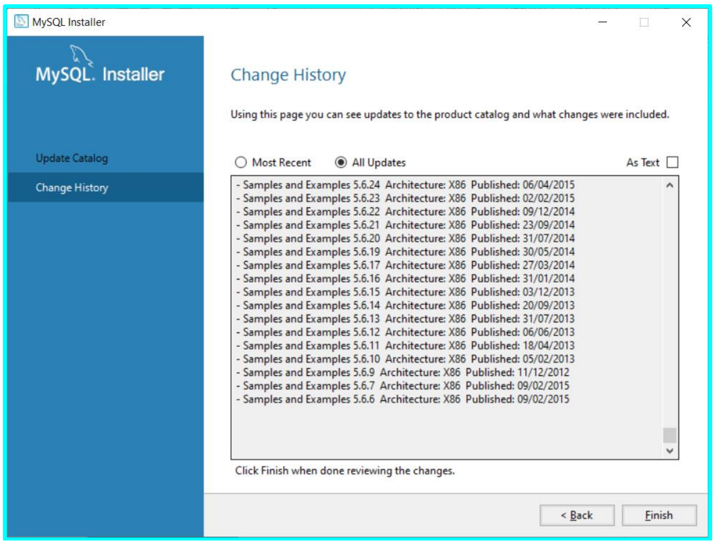


Figure 19: Installation of MySQL 5.7

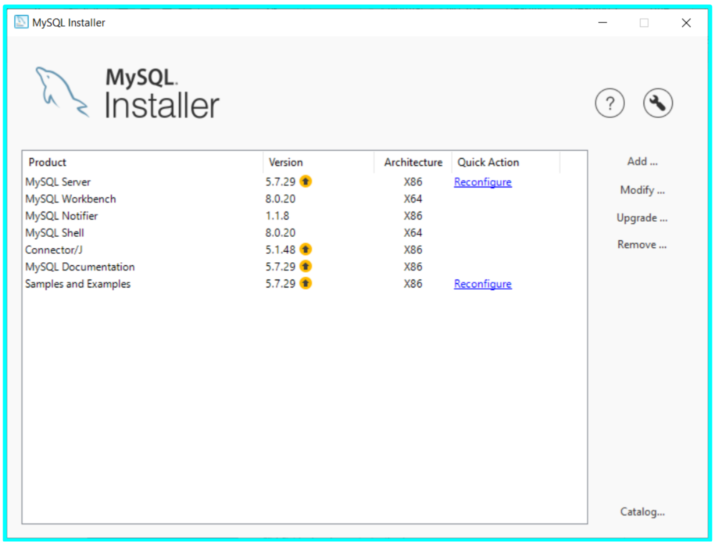


Figure 20: Installation of MySQL 5.7

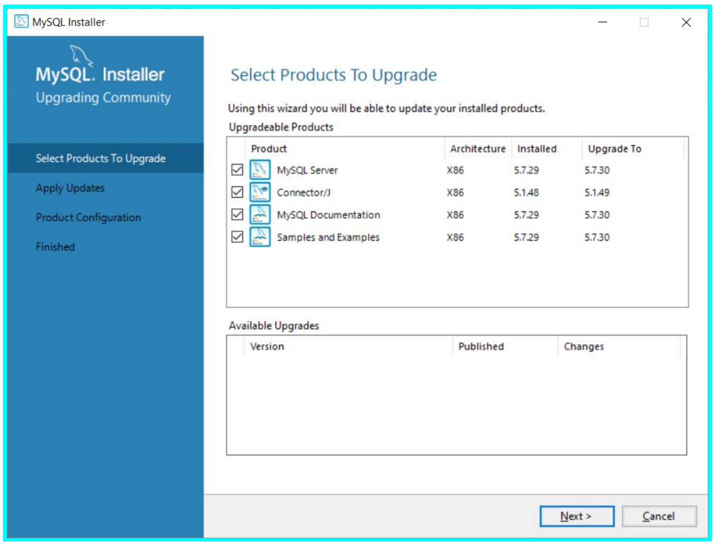


Figure 21: Installation of MySQL 5.7

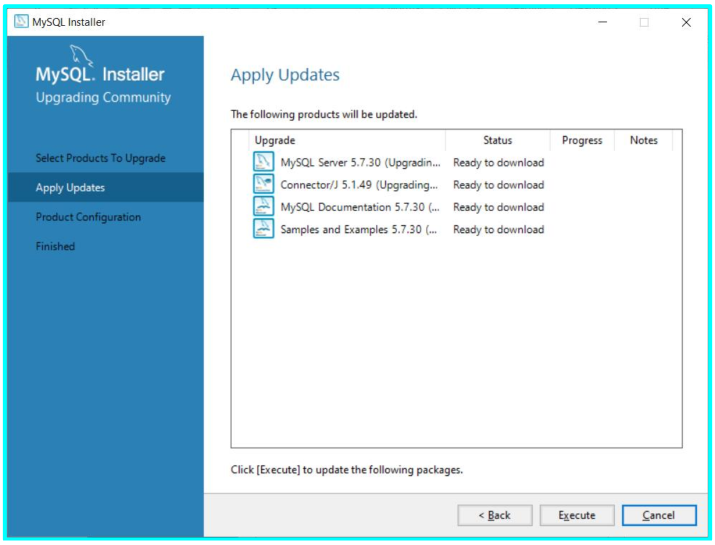


Figure 22: Installation of MySQL 5.7

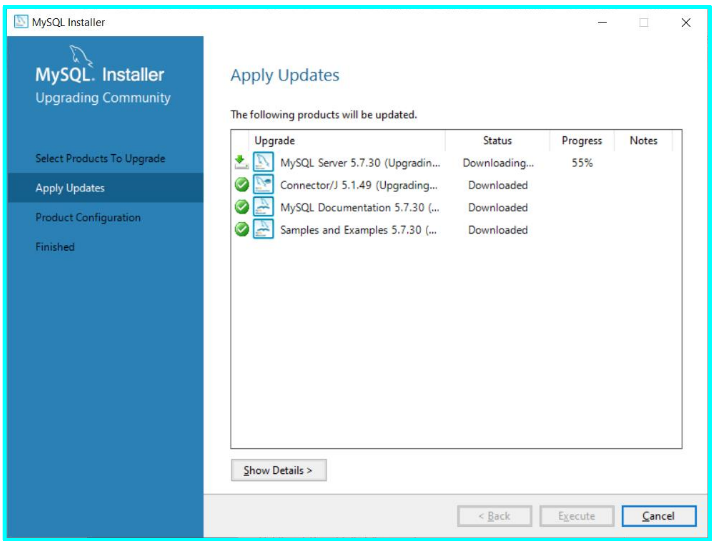


Figure 23: Installation of MySQL 5.7

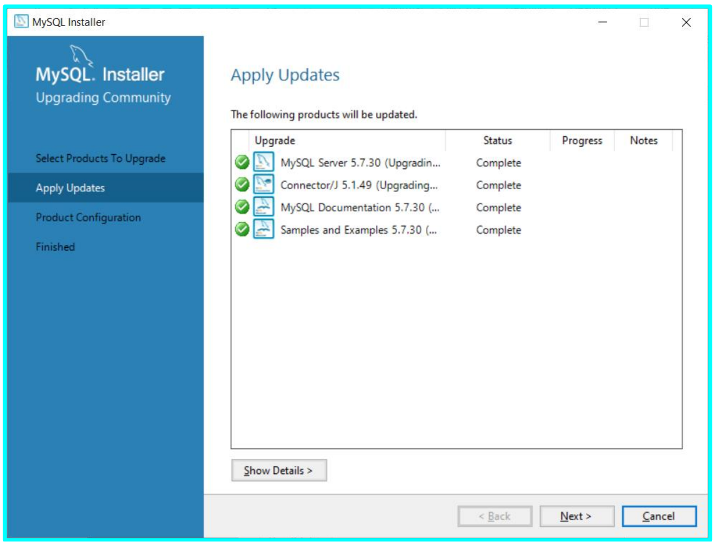


Figure 24: Installation of MySQL 5.7

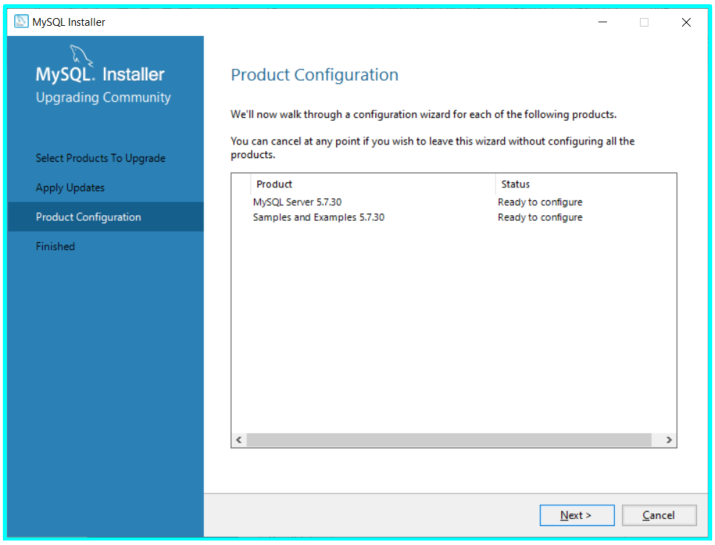


Figure 25: Installation of MySQL 5.7

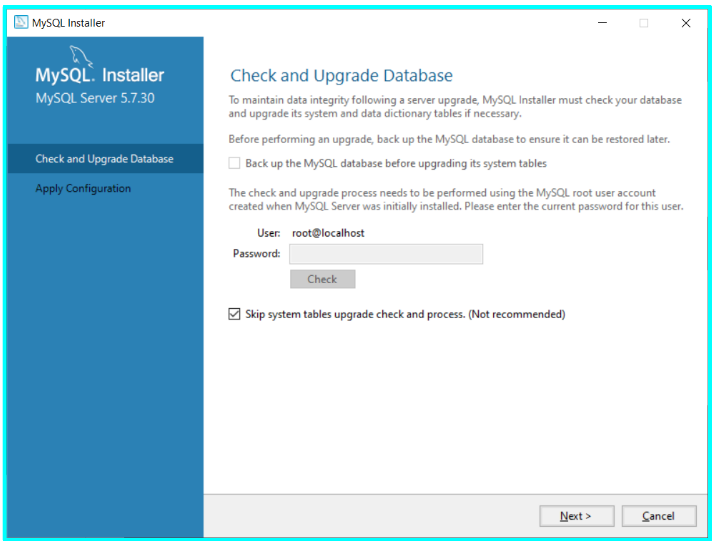


Figure 26: Installation of MySQL 5.7

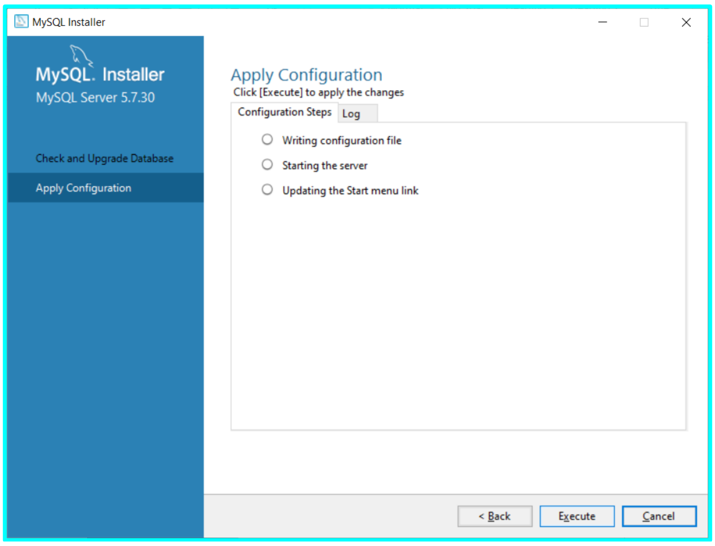


Figure 27: Installation of MySQL 5.7

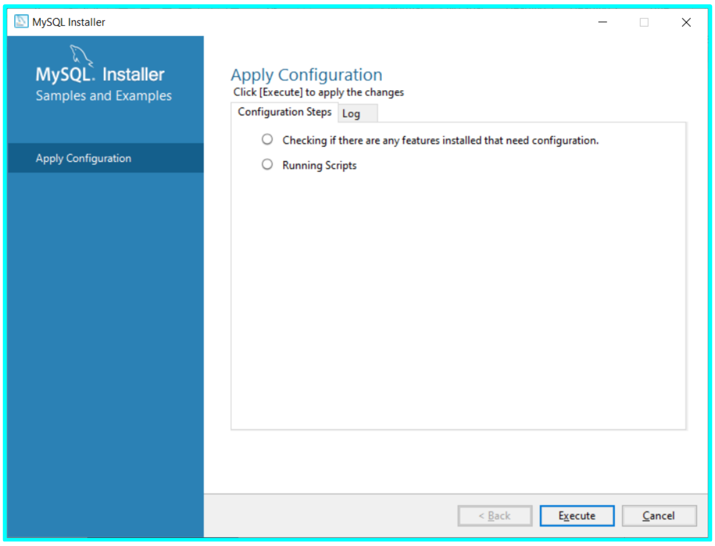


Figure 28: Installation of MySQL 5.7

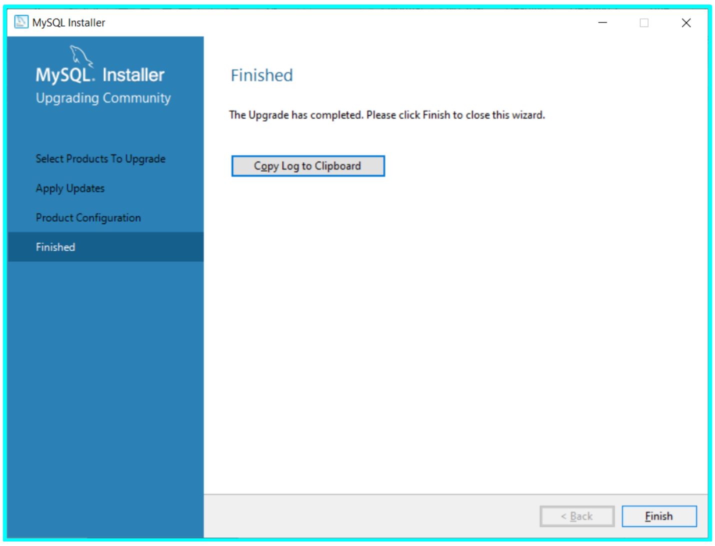


Figure 29: Installation of MySQL 5.7

4 Programming Code

This section presents programming code of data collection, data preparation, pre-processing, and implementation of all five datasets with R and MySQL.

4.1 Data Collection for Data 1, Data 2, and Data 3 by R

4.1.1 Data 1: GPI Score

Data 1 was scraped from a PDF in the website. The image of PDF is presented in Figure 30.

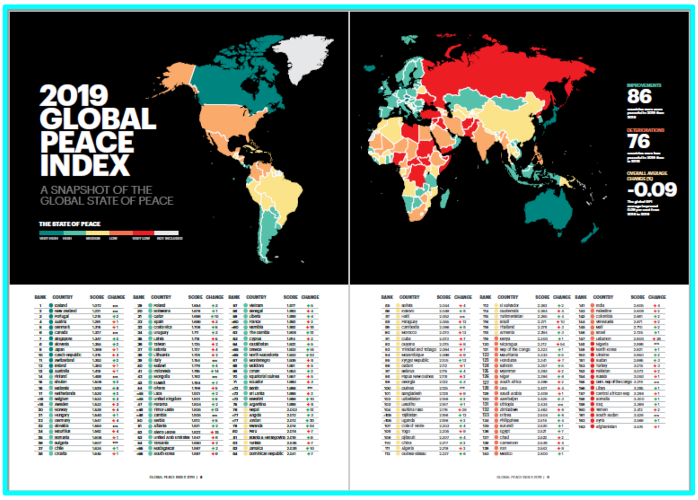


Figure 30: Image of source data of GPI Ranking 2019

Figure 31 presents how to scrape the GPI score from the pdf in the site.

```r
# Collect GPI 2019 ranking and scores from 2 pages.
library(tabulizer)
GPI2019 <- 'http://visionofhumanity.org/app/uploads/2019/07/GPI-2019web.pdf'
GPI2019_country1 <- as.data.frame(extract_tables(GPI2019, pages = 10))
GPI2019_country2 <- as.data.frame(extract_tables(GPI2019, pages = 11))

# set header : to bind the data it requires to set the same header
header <- c("Rank", "Country", "Score")
Contry1 <- GPI2019_country1[,c(1:3)]
Contry2 <- GPI2019_country1[,c(5:7)]
Contry3 <- GPI2019_country1[,c(9:11)]
Contry4 <- GPI2019_country2[,c(1:3)]
Contry5 <- GPI2019_country2[,c(5:7)]
Contry6 <- GPI2019_country2[,c(9:11)]
colnames(Contry1) = header
colnames(Contry2) = header
colnames(Contry3) = header
colnames(Contry4) = header
colnames(Contry5) = header
colnames(Contry6) = header

library(dplyr)
GPI2019_country <- rbind_all(list(Contry1, Contry2, Contry3,
                                  Contry4, Contry5, Contry6))
```

Figure 31: Scrap GPI score data from the site

The difference of country name was checked against GPI country name in excel file (Technical report 4.2.1 Data 2). Figure 32 presents the adjustment of country names. (Original Source: Garrett Grolemund[1])

```
# convert country name (set the same country name as the URL in GPI2019_country)
GPI2019_country$Country[GPI2019_country$Country == "New Zealand"]          <- "New_Zealand"
GPI2019_country$Country[GPI2019_country$Country == "Czech Republic"]       <- "Czech_Republic"
GPI2019_country$Country[GPI2019_country$Country == "Costa Rica"]           <- "Costa_Rica"
GPI2019_country$Country[GPI2019_country$Country == "United Kingdom"]       <- "United_Kingdom"
GPI2019_country$Country[GPI2019_country$Country == "Sierra Leone"]         <- "Sierra_Leone"
GPI2019_country$Country[GPI2019_country$Country == "United Arab Emirates"] <- "United_Arab_Emirates"
GPI2019_country$Country[GPI2019_country$Country == "The Gambia"]           <- "Gambia"
GPI2019_country$Country[GPI2019_country$Country == "North Macedonia"]      <- "Macedonia"
GPI2019_country$Country[GPI2019_country$Country == "Moldova"]              <- "Moldavia"
GPI2019_country$Country[GPI2019_country$Country == "Equatorial Guinea"]    <- "Equatorial_Guinea"
GPI2019_country$Country[GPI2019_country$Country == "Eswatini"]             <- "Swaziland"
GPI2019_country$Country[GPI2019_country$Country == "Bosnia & Herzegovina"] <- "Bosnia-Herzegovinia"
GPI2019_country$Country[GPI2019_country$Country == "Dominican Republic"]   <- "Dominican_Republic"
GPI2019_country$Country[GPI2019_country$Country == "Trinidad and Tobago"]  <- "Trinidad_and_Tobago"
GPI2019_country$Country[GPI2019_country$Country == "Kyrgyz Republic"]      <- "Kyrgyzstan"
GPI2019_country$Country[GPI2019_country$Country == "Papua New Guinea"]     <- "Papua_New_Guinea"
GPI2019_country$Country[GPI2019_country$Country == "Burkina Faso"]         <- "Burkina_Faso"
GPI2019_country$Country[GPI2019_country$Country == "Cote d' Ivoire"]       <- "Ivory_Coast"
GPI2019_country$Country[GPI2019_country$Country == "El Salvador"]          <- "El_Salvador"
GPI2019_country$Country[GPI2019_country$Country == "Rep of the Congo"]     <- "Congo"
GPI2019_country$Country[GPI2019_country$Country == "South Africa"]         <- "South_Africa"
GPI2019_country$Country[GPI2019_country$Country == "Saudi Arabia"]         <- "Saudi_Arabia"
GPI2019_country$Country[GPI2019_country$Country == "North Korea"]          <- "North_Korea"
GPI2019_country$Country[GPI2019_country$Country == "Central African Rep"]  <- "Central_African_Rep"
GPI2019_country$Country[GPI2019_country$Country == "Sri Lanka"]            <- "Sri_Lanka"
GPI2019_country$Country[GPI2019_country$Country == "South Korea"]          <- "South_Korea"
GPI2019_country$Country[GPI2019_country$Country == "South Sudan"]          <- "South_Sudan"
GPI2019_country$Country[GPI2019_country$Country == "Dem. Rep of the Congo"] <- "Rep_Congo"
```

Figure 32: Update country names

Figure 33 presents how to remove countries from GPI data which are not in the weather data site. (If the county does not exist in the search list, program recognize as an error and programming stops, therefore those countries are removed in advance for the automation and will be recovered when data 2 and data 3 are joining)

```
# EXTRACT TARGET COUNTRIES 163
write.csv(GPI2019_country163, "GPI2019_country163.csv")

# remove no data country
#   (because in the scrape automation, if those counties are in the list,
#    it returns error message and program stops)
GPI2019_country$Country[GPI2019_country$Country == "South_Sudan" |
                        GPI2019_country$Country == "Palestine"   |
                        GPI2019_country$Country == "Taiwan"      |
                        GPI2019_country$Country == "Kosovo"      |
                        GPI2019_country$Country == "Timor-Leste" |
                        GPI2019_country$Country == "Serbia"      |
                        GPI2019_country$Country == "Montenegro"  |
                        GPI2019_country$Country == "Rep_Congo"] <- NA
GPI2019_country <- na.omit(GPI2019_country)

# get 155 country
# EXTRACT TARGET COUNTRIES 155
write.csv(GPI2019_country, "GPI2019_country.csv")
```

Figure 33: Temporary removed countries

[1] https://rstudio-education.github.io/hopr/modify.html

4.1.2 Data 2: Global Warming Time-Series Weather information

Figure 34 presents the site which has 10 factors and its parameter information in Table 2 below.

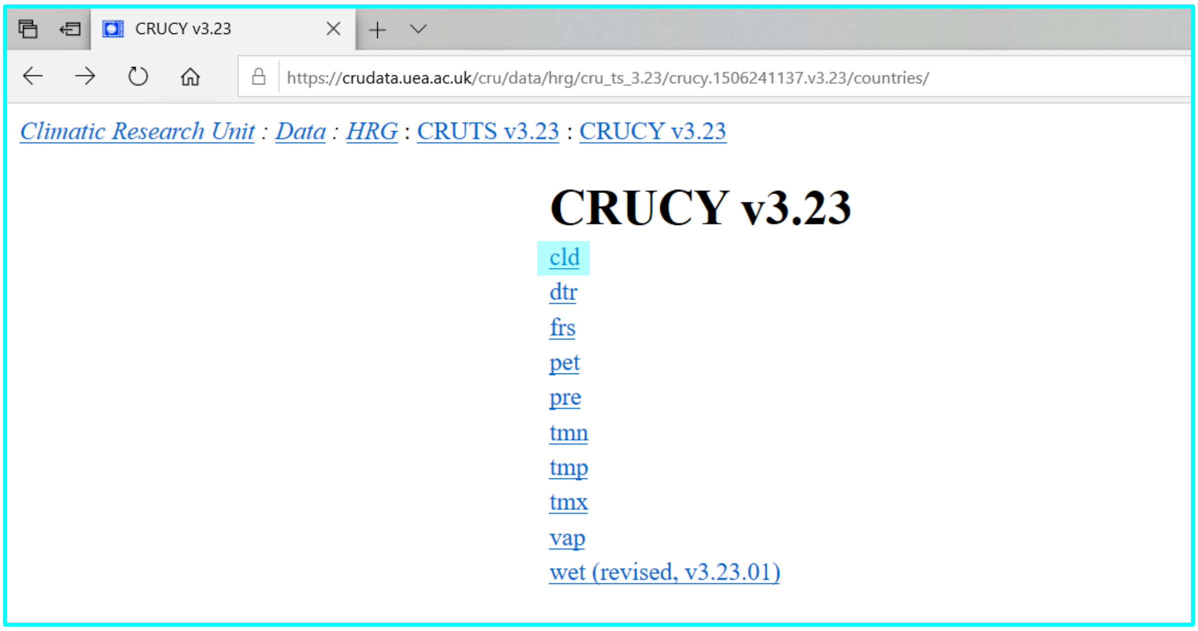


Figure 34: Root site of the weather data

Table 2: Parameter full name in Figure 34

web	Parameter name	web	Parameter name	web	Parameter name
cld	Cloud cover	pre	Precipitation	vap	Vapour Pressure
dtr	Diurnal Temperature Range	tmn	Minimum Temperature	wet	Rain Days
frs	Ground Frost Frequency	tmp	Mean Temperature		
pet	Potential Evapotranspiration	tmx	Maximum Temperature		

Figure 35 presents a sample from cld at the top of the link in Figure 34 above. This site shows links of 289 countries. The dada of 163 countries which matches with GPI data need to be scrapped.

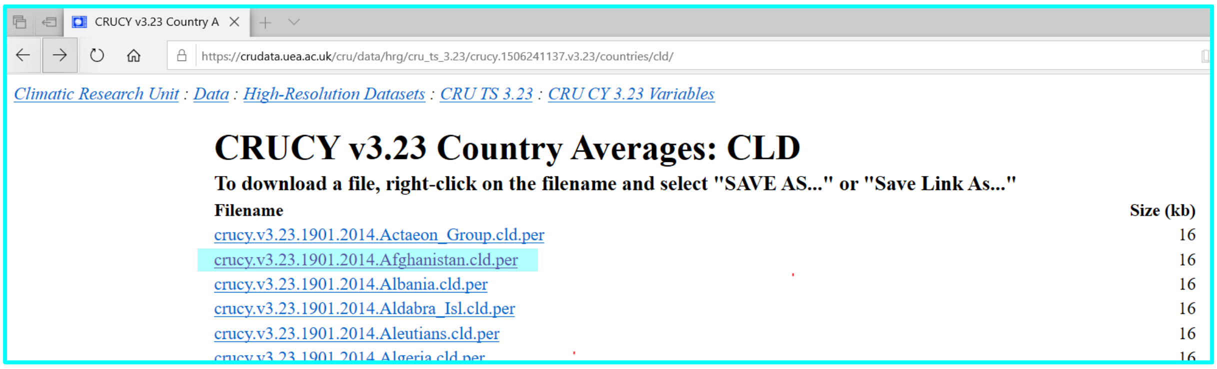
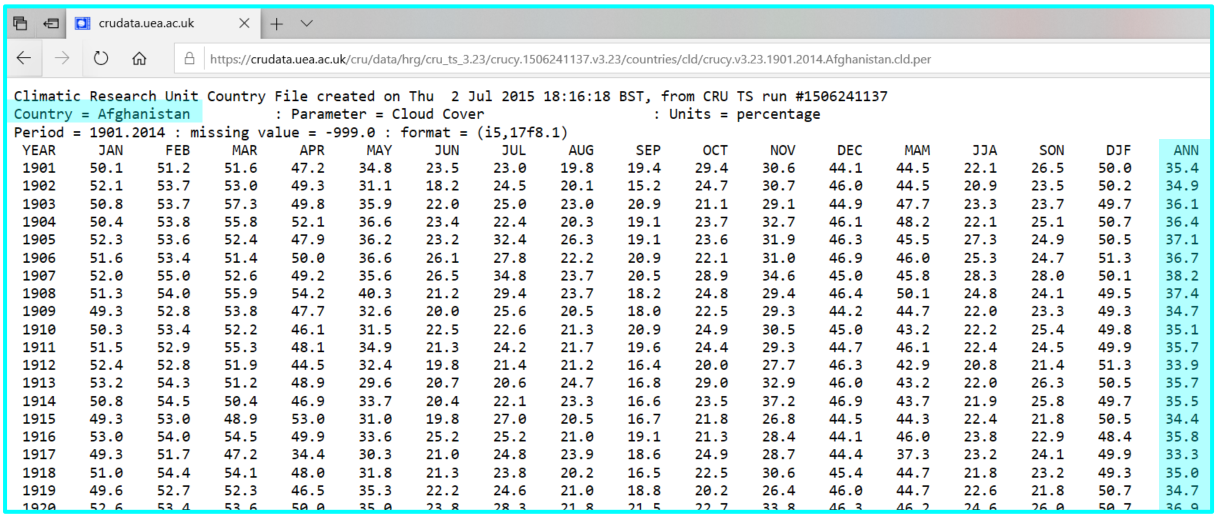


Figure 35: Inside of the site of figure 34 above

Figure 36 presents a data sample country Afghanistan in Figure 35 above. ANN (annual average) is the target date to be collected.



Climatic Research Unit Country File created on Thu 2 Jul 2015 18:16:18 BST, from CRU TS run #1506241137
Country = Afghanistan : Parameter = Cloud Cover : Units = percentage
Period = 1901.2014 : missing value = -999.0 : format = (i5,17f8.1)

YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	MAM	JJA	SON	DJF	ANN
1901	50.1	51.2	51.6	47.2	34.8	23.5	23.0	19.8	19.4	29.4	30.6	44.1	44.5	22.1	26.5	50.0	35.4
1902	52.1	53.7	53.0	49.3	31.1	18.2	24.5	20.1	15.2	24.7	30.7	46.0	44.5	20.9	23.5	50.2	34.9
1903	50.8	53.7	57.3	49.8	35.9	22.0	25.0	23.0	20.9	21.1	29.1	44.9	47.7	23.3	23.7	49.7	36.1
1904	50.4	53.8	55.8	52.1	36.6	23.4	22.4	20.3	19.1	23.7	32.7	46.1	48.2	22.1	25.1	50.7	36.4
1905	52.3	53.6	52.4	47.9	36.2	23.2	32.4	26.3	19.1	23.6	31.9	46.3	45.5	27.3	24.9	50.5	37.1
1906	51.6	53.4	51.4	50.0	36.6	26.1	27.8	22.2	20.9	22.1	31.0	46.9	46.0	25.3	24.7	51.3	36.7
1907	52.0	55.0	52.6	49.2	35.6	26.5	34.8	23.7	20.5	28.9	34.6	45.0	45.8	28.3	28.0	50.1	38.2
1908	51.3	54.0	55.9	54.2	40.3	21.2	29.4	23.7	18.2	24.8	29.4	46.4	50.1	24.8	24.1	49.5	37.4
1909	49.3	52.8	53.8	47.7	32.6	20.0	25.6	20.5	18.0	22.5	29.3	44.2	44.7	22.0	23.3	49.3	34.7
1910	50.3	53.4	52.2	46.1	31.5	22.5	22.6	21.3	20.9	24.9	30.5	45.0	43.2	22.2	25.4	49.8	35.1
1911	51.5	52.9	55.3	48.1	34.9	21.3	24.2	21.7	19.6	24.4	29.3	44.7	46.1	22.4	24.5	49.9	35.7
1912	52.4	52.8	51.9	44.5	32.4	19.8	21.4	21.2	16.4	20.0	27.7	46.3	42.9	20.8	21.4	51.3	33.9
1913	53.2	54.3	51.2	48.9	29.6	20.7	20.6	24.7	16.8	29.0	32.9	46.0	43.2	22.0	26.3	50.5	35.7
1914	50.8	54.5	50.4	46.9	33.7	20.4	22.1	23.3	16.6	23.5	37.2	46.9	43.7	21.9	25.8	49.7	35.5
1915	49.3	53.0	48.9	53.0	31.0	19.8	27.0	20.5	16.7	21.8	26.8	44.5	44.3	22.4	21.8	50.5	34.4
1916	53.0	54.0	54.5	49.9	33.6	25.2	25.2	21.0	19.1	21.3	28.4	44.1	46.0	23.8	22.9	48.4	35.8
1917	49.3	51.7	47.2	34.4	30.3	21.0	24.8	23.9	18.6	24.9	28.7	44.4	37.3	23.2	24.1	49.9	33.3
1918	51.0	54.4	54.1	48.0	31.8	21.3	23.8	20.2	16.5	22.5	30.6	45.4	44.7	21.8	23.2	49.3	35.0
1919	49.6	52.7	52.3	46.5	35.3	22.2	24.6	21.0	18.8	20.2	26.4	46.0	44.7	22.6	21.8	50.7	34.7

Figure 36: Inside of the site of Figure 35 above

Figure 37 presents the programming how to create a link table for the automation and create download path.

```
# create download data link in a frame
data_contents <- c("Cloud cover",
                   "Diurnal Temperature Range",
                   "Ground Frost Frequency",
                   "Potential Evapotranspiration",
                   "Precipitation",
                   "Minimum Temperature",
                   "Mean Temperature",
                   "Maximum Temperature",
                   "Vapour Pressure",
                   "Rain Days")

# Get key URL address
data_link <- c("https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/cld/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/dtr/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/frs/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/pet/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/pre/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/tmn/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/tmp/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/tmx/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/vap/",
               "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_3.23/crucy.1506241137.v3.23/countries/wet/"
)

download_link  <- data.frame(data_contents, data_link) # use for output file name
download_count <- download_link[,2]                    # use for counter length

# create download path
dir_path <- "C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Download_data"
dir.create(file.path("Download_data"), recursive = TRUE)
```

Figure 37: Programming Code of Creating link table

Figure 38 to Figure 43 present data scraping automation to obtain URL from 10 sites for 163 countries in **for** loop. (Original Source: Arvid Kingl[2], 2018)

```
# Download Automation
download_loop = 1:length(download_count)
for (i in download_loop) {
  # refresh from previous loop
  page <- NULL

  # GET Full address of target file for downloading.
  library(xml2)
  page <- read_html(as.character(download_count[i]))

  library(rvest)
  library(stringr)
  zip_url = page %>%
    html_nodes("a") %>%
    html_attr("href") %>%
    str_subset("\\.per")
  zip_url

  zip_url_frame <- data.frame(zip_url)
  zip_url_frame$adress <- download_link[i,2]
  full_url <- paste(zip_url_frame$adress, zip_url_frame$zip_url, sep="")
  full_url <- data.frame(full_url)
```

Figure 38: Scraping Automation 1 – to be continued to Automation 2

Figure 39 presents how to pick country name from URL which are matching with GPI country, and create path which use temporary during the downloading, and clear the data before the next country information (this is avoiding data duplication)

```
  # pickup target 163 countries from GPI2019 countries
  # remove 110 letters from the beginning and 9 letters from the end
  if(i==10) { # the last one (10th) has different number of character in the http address
    full_url$Country <- str_sub(full_url[,1], start = 114, end = -9)
  }
  else {
    full_url$Country <- str_sub(full_url[,1], start = 111, end = -9)
  }

  # right join to check country and data between GPI2019 and data existance
  full_url_merge_R <- right_join(full_url, GPI2019_country, by="Country")
  #write.csv(full_url_merge_R, "merge_data.csv")

  # eventually we get 155 countries data location URL data
  Total_Country <- full_url_merge_R[,2:4]
  Total_url <- full_url_merge_R[,1]

  # Automatically download per files

  # ---  remove temp files from previous action
  delete_tempfiles <- dir("C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Download_data/DownloadData/DownloadData_per",
                          recursive=T,full.names=T)
  file.remove(delete_tempfiles)
  # ---

  tmp_dir = file.path(dir_path, "DownloadData")
  per_dir <- file.path(tmp_dir, "DownloadData_per")

  dir.create(file.path("Download_data","DownloadData_per"), recursive = TRUE)
```

Figure 39: Scraping Automation 2 -- to be continued to Automation 3

[2] https://www.datacamp.com/community/tutorials/r-web-scraping-rvest

Figure 40 presents downloading data per URL.

```
# Pull out each url using a for loop
zip_loop = 1:length(Total_url)
for (j in zip_loop) {
  tmp_url = as.character(Total_url[j])
  # per_file <- str_sub(tmp_url, start = 111, end = -9)  ## get country name
  # per_file <- paste("Country_", i, ".csv", sep = "")
  per_file <- j  # file name is number
  per_file = file.path(per_dir, per_file)                ## per file name as file path
  download.file(tmp_url, destfile = per_file)            ## download per file
  Sys.sleep(1)
}

# read data to join them all
files <- list.files(path = "C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Download_data/DownloadData/DownloadData_per",
                    full.names = T)
```

Figure 40: Scraping Automation 3 -- to be continued to Automation 4

Figure 41 presents scraping country name and rest of data. The data was collected with flat shape therefore it was transformed to the original shape and jointed the country which are previously obtained.

```
all_Country <- NULL
for (k in zip_loop) {
  inputdata <- read.delim(files[k], head=F, sep="," ,na.strings = c("", "NA"))

  # remove the 1st and the 3rd rows
  Year_data <- inputdata[c(-3,-1),]

  # get country name
  Country_org <- str_sub(Year_data[1,1], start = 11)
  Country_org <- gsub(" ", "", strsplit(Country_org,split=":")[[1]][1])  # remove space

  # remove 1st row
  Year_data <- Year_data[-1,]

  # set in data frame
  frame_data <- strsplit(as.character(Year_data[,1]), " +")

  # convert to data frame
  Counrty_clean <- as.data.frame(frame_data,drop=FALSE)
  Counrty_clean_update <- Counrty_clean[-1,]

  # back to original form
  library(data.table)
  Country_trans <- transpose(Counrty_clean_update)

  # set header
  header <- c("Year", "JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP",
              "OCT", "NOV", "DEC", "MAM", "JJA", "SON", "DJF", "ANN")
  colnames(Country_trans) = header

  # remove header duplication
  Country_trans <- Country_trans[-1,]

  # set country name
  Country_trans$Country <- Country_org
  all_Country <- rbind(all_Country, Country_trans)
}
```

Figure 41: Scraping Automation 4 -- to be continued to Automation 5

Figure 42 presents tidying the data sort by country and year and creating output file with factor name which are in contents name (top in Fig 37). All data was extracted for data checking.

```
# set file export location
setwd("C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Output")

# switch the order of all data.
all_Country <- all_Country[,c(19,1:18)]

# sort data by country year
Sort_all_Country <- all_Country[order(all_Country$Country, all_Country$Year),]

# extract data
file_name <- paste(download_link[i,1], ".","csv", sep="")
file_name

# file export
write.csv(Sort_all_Country, file_name)
```

Figure 42: Scraping Automation 5 -- to be continued to Automation 6

Figure 43 presents creating final output. From the scraped data country name, Year and ANN (annual mean) were extracted and joined for 163 countries.

```
  # ------------------------------------------
  # set only annual average data --> header = "ANN"
  ave_data <- Sort_all_Country[,c(1,2,19)]

  # extract data
  file_name_ave <- paste(download_link[i,1], "_","ave",".","csv", sep="")
  file_name_ave

  # file export
  write.csv(ave_data, file_name_ave)

  # ------------------------------------------
  # gather all ave data
  # set header from file name
  Title_name <- paste(download_link[i,1])

  if (i==1){
    header_ave <- c("Counrty", "Year", Title_name)
    colnames(ave_data) = header_ave
    GW_ave_all <- NULL
    GW_ave_all <- ave_data
  }
  else {
    header_ave <- c("Counrty1", "Year1", Title_name)
    colnames(ave_data) = header_ave
    GW_ave_all <- cbind(GW_ave_all, ave_data)
    GW_ave_all <- subset(GW_ave_all, select = -c(Counrty1, Year1))

  }
  # go back to original work location
  setwd("C:/Wakako/NCI/2020_Data_Analytics_MSc/Program")
}
```

Figure 43: Scraping Automation 6 -- Automation end here

Figure 44 presents to set shorten the name for header.

```
# ---  remove temp files
delete_tempfiles <- dir("C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Download_data/DownloadData/DownloadData_per",
                        recursive=T,full.names=T)
file.remove(delete_tempfiles)
# ---

# Set header in the final collected data
Final_header <- c("Country",
                  "Year",
                  "Cld_cv",    # "Cloud cover",
                  "Temp_day", # "Diurnal Temperature Range",
                  "Gnd_Fr",    # "Ground Frost Frequency",
                  "Pot_Eva",   # "Potential Evapotranspiration",
                  "Prcp",      # "Precipitation",
                  "Min_Tmp",   # "Minimum Temperature",
                  "Mean_Tmp", # "Mean Temperature",
                  "Max_Tmp",   # "Maximum Temperature",
                  "Vap_prs",   # "Vapour Pressure",
                  "Rn_day")    # "Rain Days"

colnames(GW_ave_all) = Final_header
dim(GW_ave_all)
str(GW_ave_all)
```

Figure 44: Set short name header

Figure 45 to 47 present how to obtain the oldest and the newest 10 years average of mean temperature and its increase rate.

```
# ---------------------------------------------------------------------------------- #
# Obtain increased mean temperature between the oldest 10 years (1901-1910) and
#                                              the newest 10 years (2005-2014)
# then find increased temperature and increased rate
# ---------------------------------------------------------------------------------- #
# obtain country, year and mean temp
temp_check <- GW_ave_all[,c(1,2,9)]
temp_check$Year <- as.numeric(temp_check$Year)
dim(temp_check)
str(temp_check)

# remain year between 1901-1910 and 2005-2014.
temp_check$Year[temp_check$Year > 1910 & temp_check$Year < 2005 ] <- NA
temp_check <- na.omit(temp_check)
dim(temp_check)
str(temp_check)

# Transpose the data country vs year
library(data.table)
setDT(temp_check)
Year10 <- dcast(temp_check, Country ~ Year, value.var = "Mean_Tmp")
str(Year10)

# get mean of oldest and newest 10 years
Year10_header <- c("Country", "Y1901", "Y1902", "Y1903", "Y1904", "Y1905",
                   "Y1906", "Y1907", "Y1908", "Y1909", "Y1910",
                   "Y2005", "Y2006", "Y2007", "Y2008", "Y2009",
                   "Y2010", "Y2011", "Y2012", "Y2013", "Y2014")

colnames(Year10) = Year10_header
str(Year10)
dim(Year10)
```

Figure 45: Obtaining the older and the newest 10 years mean temperature

```
# Set data as numeric type
Year10$Y1901 <- as.numeric(Year10$Y1901)
Year10$Y1902 <- as.numeric(Year10$Y1902)
Year10$Y1903 <- as.numeric(Year10$Y1903)
Year10$Y1904 <- as.numeric(Year10$Y1904)
Year10$Y1905 <- as.numeric(Year10$Y1905)
Year10$Y1906 <- as.numeric(Year10$Y1906)
Year10$Y1907 <- as.numeric(Year10$Y1907)
Year10$Y1908 <- as.numeric(Year10$Y1908)
Year10$Y1909 <- as.numeric(Year10$Y1909)
Year10$Y1910 <- as.numeric(Year10$Y1910)
Year10$Y2005 <- as.numeric(Year10$Y2005)
Year10$Y2006 <- as.numeric(Year10$Y2006)
Year10$Y2007 <- as.numeric(Year10$Y2007)
Year10$Y2008 <- as.numeric(Year10$Y2008)
Year10$Y2009 <- as.numeric(Year10$Y2009)
Year10$Y2010 <- as.numeric(Year10$Y2010)
Year10$Y2011 <- as.numeric(Year10$Y2011)
Year10$Y2012 <- as.numeric(Year10$Y2012)
Year10$Y2013 <- as.numeric(Year10$Y2013)
Year10$Y2014 <- as.numeric(Year10$Y2014)
str(Year10)

Year10$Old10ave <- (Year10$Y1901+Year10$Y1902+Year10$Y1903+Year10$Y1904+Year10$Y1905+
                      Year10$Y1906+Year10$Y1907+Year10$Y1908+Year10$Y1909+Year10$Y1910)/10
Year10$New10ave <- (Year10$Y2005+Year10$Y2006+Year10$Y2007+Year10$Y2008+Year10$Y2009+
                      Year10$Y2010+Year10$Y2011+Year10$Y2012+Year10$Y2013+Year10$Y2014)/10
str(Year10)
dim(Year10)
```

Figure 46: Calculating average of old and new mean temperature

```
# obtain difference between oldest and newest, and increase rates
Temp_Inc <- Year10[,c(1,22,23)]
Temp_Inc$Tp_diff <- Temp_Inc$New10ave - Temp_Inc$Old10ave
Temp_Inc$Tp_IncR <- Temp_Inc$New10ave/Temp_Inc$Old10ave
Temp_Inc <- Temp_Inc[,c(1,4,5)]
str(Temp_Inc)
```

Figure 47: Calculating the increase rate

Figure 48 presents to get average data of all data set by country and joint the dataset from Figure 47 and complete preparation of data 2.

```
# ------------------------------------------------------------------------------- #
# Change data type to numeric
GW_ave_all$Cld_cv   <- as.numeric(GW_ave_all$Cld_cv)
GW_ave_all$Temp_day <- as.numeric(GW_ave_all$Temp_day)
GW_ave_all$Gnd_Fr   <- as.numeric(GW_ave_all$Gnd_Fr)
GW_ave_all$Pot_Eva  <- as.numeric(GW_ave_all$Pot_Eva)
GW_ave_all$Prcp     <- as.numeric(GW_ave_all$Prcp)
GW_ave_all$Min_Tmp  <- as.numeric(GW_ave_all$Min_Tmp)
GW_ave_all$Mean_Tmp <- as.numeric(GW_ave_all$Mean_Tmp)
GW_ave_all$Max_Tmp  <- as.numeric(GW_ave_all$Max_Tmp)
GW_ave_all$Vap_prs  <- as.numeric(GW_ave_all$Vap_prs)
GW_ave_all$Rn_day   <- as.numeric(GW_ave_all$Rn_day)

# calculate average data by country in the range of 1900 to 2014
GW_agg_ave_all <- aggregate(GW_ave_all[,c(3:12)], by = list(GW_ave_all$Country), FUN = mean)
str(GW_agg_ave_all)

# update the title
names(GW_agg_ave_all)[names(GW_agg_ave_all)=="Group.1"] <- "Country"
str(GW_agg_ave_all)

# join temperature difference and temperature increase rate on GW_agg_ave_all
GW_agg_ave_allT <- merge(x=GW_agg_ave_all, y=Temp_Inc, by="Country", all.x=TRUE)
str(GW_agg_ave_allT)

write.csv(GW_agg_ave_allT, "GW_agg_ave_allT.csv")

GW_agg_ave_all <- GW_agg_ave_allT
```

Figure 48: Data 2 preparation is complete here

4.1.3 Data 3: CO2 Emission data

Figure 49 presents loading fossil CO2 data and select country name and CO2 emission data of 1970 and 2017.

```
# ============================================================================ #
# Data3 from EU Open Data Portal
# Obtain CO2 Emissions rate of 1970, 2017 and increase rate between 1970 and 2017
# ============================================================================ #
# Load data
setwd("C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Output")
CO2_EM <- read.csv("fossil_CO2_totals_by_country.csv", header=T,
                   na.strings=c(""), stringsAsFactors = T)

str(CO2_EM)
dim(CO2_EM)

# Get Country name and 1970 and 2017 data
header <- c("Country", "CO2_1970", "CO2_2017")
CO2_INC <- CO2_EM[,c(1,2,49)]
colnames(CO2_INC) = header

CO2_INC$Country <- as.character(CO2_INC$Country )
str(CO2_INC)
```

Figure 49: Collection of CO2 Emission data

The difference of country name was checked against GPI country name in excel file (Technical report 4.2.1 Data 2). Figure 50 to 53 present correcting country name and dealing with missing data.

```
# Set the same country name as the GPI data
CO2_INC$Country[CO2_INC$Country == "Bosnia and Herzegovina"]    <- "Bosnia-Herzegovinia"
CO2_INC$Country[CO2_INC$Country == "Burkina Faso"]              <- "Burkina_Faso"
CO2_INC$Country[CO2_INC$Country == "Central African Republic"]  <- "Central_African_Rep"
CO2_INC$Country[CO2_INC$Country == "Czechia"]                   <- "Czech_Republic"
CO2_INC$Country[CO2_INC$Country == "Hong Kong" |
                CO2_INC$Country == "Macao"]                     <- "China"
CO2_INC$Country[CO2_INC$Country == "Costa Rica"]                <- "Costa_Rica"
CO2_INC$Country[CO2_INC$Country == "Faroes" |
                CO2_INC$Country == "Greenland" ]                <- "Denmark"
CO2_INC$Country[CO2_INC$Country == "Dominican Republic"]        <- "Dominican_Republic"
CO2_INC$Country[CO2_INC$Country == "El Salvador"]               <- "El_Salvador"
CO2_INC$Country[CO2_INC$Country == "Equatorial Guinea"]         <- "Equatorial_Guinea"
CO2_INC$Country[CO2_INC$Country == "France and Monaco"|
                CO2_INC$Country == "French Guiana"     |
                CO2_INC$Country == "French Polynesia"  |
                CO2_INC$Country == "Guadeloupe"        |
                CO2_INC$Country == "Martinique"        |
                CO2_INC$Country == "New Caledonia"     |
                CO2_INC$Country == "Reunion"           |
                CO2_INC$Country == "Saint Pierre and Miquelon"] <- "France"
CO2_INC$Country[CO2_INC$Country == "The Gambia"]                <- "Gambia"
CO2_INC$Country[CO2_INC$Country == "Italy, San Marino and the Holy See"] <- "Italy"
CO2_INC$Country[CO2_INC$Country == "Israel and Palestine, State of"] <- "Israel"
CO2_INC$Country[CO2_INC$Country == "Ivory Coast"]               <- "Ivory_Coast"
CO2_INC$Country[CO2_INC$Country == "former Yugoslav Republic of Macedonia, the"] <- "Macedonia"
CO2_INC$Country[CO2_INC$Country == "Moldova"]                   <- "Moldavia"
CO2_INC$Country[CO2_INC$Country == "Myanmar/Burma"]             <- "Myanmar"
```

Figure 50: Country name adjustment – part 1

```
CO2_INC$Country[CO2_INC$Country == "Curacao"]                   <- "Netherlands"
CO2_INC$Country[CO2_INC$Country == "New Zealand"]               <- "New_Zealand"
CO2_INC$Country[CO2_INC$Country == "North Korea"]               <- "North_Korea"
CO2_INC$Country[CO2_INC$Country == "Papua New Guinea"]          <- "Papua_New_Guinea"
CO2_INC$Country[CO2_INC$Country == "Saudi Arabia"]              <- "Saudi_Arabia"
CO2_INC$Country[CO2_INC$Country == "Sierra Leone"]              <- "Sierra_Leone"
CO2_INC$Country[CO2_INC$Country == "South Africa"]              <- "South_Africa"
CO2_INC$Country[CO2_INC$Country == "South Korea"]               <- "South_Korea"
CO2_INC$Country[CO2_INC$Country == "Spain and Andorra"]         <- "Spain"
CO2_INC$Country[CO2_INC$Country == "Sri Lanka"]                 <- "Sri_Lanka"
CO2_INC$Country[CO2_INC$Country == "Sudan and South Sudan"]     <- "Sudan"
CO2_INC$Country[CO2_INC$Country == "Switzerland and Liechtenstein"] <- "Switzerland"
CO2_INC$Country[CO2_INC$Country == "Trinidad and Tobago"]       <- "Trinidad_and_Tobago"
CO2_INC$Country[CO2_INC$Country == "United Arab Emirates"]      <- "United_Arab_Emirates"
CO2_INC$Country[CO2_INC$Country == "Democratic Republic of the Congo"] <- "Rep_Congo"
CO2_INC$Country[CO2_INC$Country == "Serbia and Montenegro"]     <- "Serbia"
CO2_INC$Country[CO2_INC$Country == "Bermuda"                |
                CO2_INC$Country == "British Virgin Islands" |
                CO2_INC$Country == "Cayman Islands"         |
                CO2_INC$Country == "Falkland Islands"       |
                CO2_INC$Country == "Gibraltar"              |
                CO2_INC$Country == "Saint Helena, Ascension and Tristan da Cunha" |
                CO2_INC$Country == "Turks and Caicos Islands" |
                CO2_INC$Country == "United Kingdom"]            <- "United_Kingdom"
```

Figure 51: Country name adjustment – part 2

```
CO2_INC$Country[CO2_INC$Country == "United States"]              <- "USA"
CO2_INC$Country[CO2_INC$Country == "Anguilla"                   |
                CO2_INC$Country == "Antigua and Barbuda"        |
                CO2_INC$Country == "Aruba"                      |
                CO2_INC$Country == "Bahamas"                    |
                CO2_INC$Country == "Barbados"                   |
                CO2_INC$Country == "Belize"                     |
                CO2_INC$Country == "Brunei"                     |
                CO2_INC$Country == "Cape Verde"                 |
                CO2_INC$Country == "Comoros"                    |
                CO2_INC$Country == "Cook Islands"               |
                CO2_INC$Country == "Dominica"                   |
                CO2_INC$Country == "Fiji"                       |
                CO2_INC$Country == "Grenada"                    |
                CO2_INC$Country == "Kiribati"                   |
                CO2_INC$Country == "Luxembourg"                 |
                CO2_INC$Country == "Maldives"                   |
                CO2_INC$Country == "Malta"                      |
                CO2_INC$Country == "Palau"                      |
                CO2_INC$Country == "Puerto Rico"                |
                CO2_INC$Country == "Saint Kitts and Nevis"      |
                CO2_INC$Country == "Saint Lucia"                |
                CO2_INC$Country == "Saint Vincent and the Grenadines" |
                CO2_INC$Country == "Samoa"                      |
                CO2_INC$Country == "Serbia and Montenegro"      |
                CO2_INC$Country == "Seychelles"                 |
                CO2_INC$Country == "Solomon Islands"            |
```

Figure 52: Country name adjustment – part 3

At the section of add missing country in Figure 53, data for Montenegro was separated from Serbia and Montenegro, therefore the same data as Serbia. South Sudan use the same data as Sudan.

```
                CO2_INC$Country == "Suriname"                   |
                CO2_INC$Country == "Tonga"                      |
                CO2_INC$Country == "Vanuatu"                    |
                CO2_INC$Country == "Western Sahara"             |
                CO2_INC$Country == "International Aviation"     |
                CO2_INC$Country == "International Shipping"     |
                CO2_INC$Country == "EU28"                       |
                CO2_INC$Country == "GLOBAL TOTAL"]              <- NA
# remove no data rows
CO2_INC <- na.omit(CO2_INC)

# add missing country rows from data 2.
df <- data.frame(
  Country = c("Palestine", "Kosovo", "Palestine", "Montenegro",
              "Macedonia", "South_Sudan", "Gambia"),
  CO2_1970 = c(0,0,0,0.64,0,1.57,0),
  CO2_2017 = c(0,0,0,0.93,0,14.65,0),
  stringsAsFactors = FALSE
)

df
CO2_INC <- rbind(CO2_INC, df)
dim(CO2_INC)
```

Figure 53: Country name adjustment – part 4 and deal with missing data

By adjusting country name inf Fig 50 to 53, it turned out that some countries have a value of two or more. Based on this, Figure 54 presents adding each data country to fit in 163 countries, and it joined to the data prepared in data 2.

```
# Aggregate each country and calculate increasing rate
library(dplyr)
CO2_Agg <- aggregate(CO2_INC[,c(2:3)],  by=list(CO2_INC$Country), FUN=sum)
CO2_Agg$Inc_Rate <- round(CO2_Agg$CO2_2017/CO2_Agg$CO2_1970, digits = 5)
dim(CO2_Agg)
str(CO2_Agg)

# update column name to "Country"
names(CO2_Agg)[names(CO2_Agg)=="Group.1"] <- "Country"
write.csv(CO2_Agg, "CO2_Agg.csv")

#rm(GW_ALL)
# Global Warming final data
# Join increase rate of CO2 emission to the other Global Warming data above
GW_ALL <- merge(x=CO2_Agg, y=GW_agg_ave_all, by="Country", all.x=TRUE)

# Join GPI ranking score as GPI_Score with numeric data type
GW_ALL <- merge(x=GPI2019_country163, y=GW_ALL, by="Country", all.x=TRUE)
names(GW_ALL)[names(GW_ALL)=="Score"] <- "GPI_Score"

# set as numeric format
GW_ALL$GPI_Score <- as.numeric(GW_ALL$GPI_Score)
str(GW_ALL)
summary(GW_ALL)
```

Figure 54: Aggregate data by country and join to data 2

Figure 55 presents removing unused factors. Rank was removed because it is not analytical data, and there is Mean Temperature therefore Minimum Temperature and Maximum Temperature were removed.

```
# Remove Rank, Min_Tmp and Max_Tmp --> Mean_Tmp can tell
GW_ALL$Rank    <- NULL
GW_ALL$Min_Tmp <- NULL
GW_ALL$Max_Tmp <- NULL
str(GW_ALL)
```

Figure 55: Removing unused data

Figure 56 presents dealing for missing data by random forest. (Origianl Source: NCI ADM Class Text)

```
# some country doesn't have full data
# missing data is filled by Random forest
# Random Forest
    library(mice)
    # use 4 factors to judge the missing data, (those data no observation missing)
    mice_mod <- mice(GW_ALL[, !names(GW_ALL) %in%
                                  c(2:5)], m = 3, maxit = 3, method ='rf')
    mice_output <- complete(mice_mod)

    # Replace missing data in GW_ALL main data
    GW_ALL$Inc_Rate <- mice_output$Inc_Rate
    GW_ALL$Cld_cv   <- mice_output$Cld_cv
    GW_ALL$Temp_day <- mice_output$Temp_day
    GW_ALL$Gnd_Fr   <- mice_output$Gnd_Fr
    GW_ALL$Pot_Eva  <- mice_output$Pot_Eva
    GW_ALL$Prcp     <- mice_output$Prcp
    GW_ALL$Mean_Tmp <- mice_output$Mean_Tmp
    GW_ALL$Vap_prs  <- mice_output$Vap_prs
    GW_ALL$Rn_day   <- mice_output$Rn_day
    GW_ALL$Tp_diff  <- mice_output$Tp_diff
    GW_ALL$Tp_IncR  <- mice_output$Tp_IncR

    str(GW_ALL)
    dim(GW_ALL)
    summary(GW_ALL)
```

Figure 56: Random Forest was conducted for missing data

Figure 57 presents the final data was sorted and extracted as Global Warming data.

```
# update data order
GW_ALL <- GW_ALL[,c(1,6:15,3:5,2)]
str(GW_ALL)
summary(GW_ALL)
write.csv(GW_ALL, "GW_ALL.csv")        # Final Global Warming data
```

Figure 57: Global Warming final data compete

4.2 Data Collection for Data 4 and Data 5 by MySQL

The program was created in a text format. The tests were run in the MySQL shell, and final program was run by batch from the Windows command prompt console which is presented in figure 58.

```
Command Prompt
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\wakak>cd C:\Program Files (x86)\MySQL\MySQL Server 5.7\bin

C:\Program Files (x86)\MySQL\MySQL Server 5.7\bin>C:\Program Files (x86)\MySQL\MySQL Server 5.7\bi
n>mysql -u root -p < C:\Wakako\NCI\2020_Data_Analytics_MSc\Program\Output2\MySQL_Natural_Disasters
_Program4.txt
```

Figure 58: MySQL programming run by batch command

4.2.1 Data 4: Natural Disasters information

On the MySQL shell screen, the program code is hard to see, therefore it is displayed by Syntax Highlight[3] site which are shown from Figure 59 to 73.

Figure 59 presents how to create a table for natural disaster and load natural disasters data.

```
/* remove existing data */
Drop database ND3;
create database ND3;
use ND3;

/* Create table for Natural Disasters data */
create table Natural_Disasters (
    Dis_No varchar(13) NOT NULL, Year int NOT NULL, Seq int NOT NULL,
    Disaster_Group varchar(7) NOT NULL, Disaster_Subgroup varchar(17) NOT NULL,

    Disaster_Type varchar(19) NOT NULL, Disaster_Subtype varchar(32), Disaster_Subsubtype varchar(23),
    Event_Name varchar(76), Entry_Criteria varchar(10), Country varchar(58) NOT NULL, ISO varchar(3),
    Region varchar(25), Continent varchar(8), Location varchar(344), Origin varchar(118),
    Associated_Dis varchar(29), Associated_Dis2 varchar(29), OFDA_Response varchar(3),
    Appeal varchar(3), Declaration varchar(3), Aid_Contribution int, Dis_Mag_Value int,
    Dis_Mag_Scale varchar(10), Latitude varchar(8), Longitude varchar(8), Local_Time int,
    River_Basin varchar(320), Start_Year int, Start_Month int, Start_Day int, End_Year int,
    End_Month int, End_Day int, No_Deaths int, No_Missing int, Total_Deaths int, No_Injured int,
    No_Affected int, No_Homeless int, Total_Affected int, Reconstruction_Costs_US int,
    Insured_Damages_US int, Total_Damages_US int, CPI float, primary key (Dis_No));

/* Load data */
load data local infile
'C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Output2/emdat_public_2020_04_29_query_uid-0ZbMRD.csv'
into table ND3.Natural_Disasters fields terminated by ',' enclosed by '"' ignore 1 lines;
```

Figure 59: Table creation in MySQL

[3] http://www.planetb.ca/syntax-highlight-word

Figure 60 presents preparation for country name comparison between GPI and weather information datasets.

```
/* Need to adjust the country name with GPI 2019 data sheet*/
/* Create table for GPI 2019 Country Ranking sheet */
create table GPI(
    ID int NOT NULL auto_increment,
    Rank int NOT NULL,
    Country varchar(20) NOT NULL,
    score float NOT NULL,
    primary key (ID));

load data
local infile 'C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/GPI2019_country163.csv'
into table ND3.GPI fields terminated by ',' enclosed by '"' LINES TERMINATED BY '\r\n' ignore 1 lines;

/* Create country name table from Natural_Disasters to adjust the same name as GPI rank table */
/* create table country_ND(select Country from Natural_disasters group by Country); */
create table country_ND(
    select distinct Country
    from Natural_Disasters);

/* find different naming of coutries */
create table country_adj1(
    select a.Country as ND_C,
           b.Country as GPI_C
    from country_ND a left join GPI b on a.Country = b.Country);
create table country_adj2(
    select a.Country as ND_C,
           b.Country as GPI_C
    from country_ND a right join GPI b on a.Country = b.Country);
```

Figure 60: Create tables to compare country name

Figure 61 presents exporting the files for double check in excel and pick adjusted name.

```
/* Export for Data check */
select * from country_adj1;
select * from country_adj2;

select * from country_adj1
    into outfile 'C:\\Wakako\\NCI\\2020_Data_Analytics_MSc\\Program\\Output2\\country_adj1.csv'
    fields terminated by ',' enclosed by '"' LINES TERMINATED BY '\n';
select * from country_adj2
    into outfile 'C:\\Wakako\\NCI\\2020_Data_Analytics_MSc\\Program\\Output2\\country_adj2.csv'
    fields terminated by ',' enclosed by '"' LINES TERMINATED BY '\n';
```

Figure 61: Export for check

Figure 62 to 65 present setting the adjusted country name were.

```
/* Update Country name in Natural_Disasters data -- should be the same as the gpi*/
update Natural_Disasters set country = 'Rep_Congo'
                where country = 'Congo (the Democratic Republic of the)';
update Natural_Disasters set country = 'Congo' where country = 'Congo (the)';
update Natural_Disasters set country = 'Palestine' where country = 'Palestine, State of';
update Natural_Disasters set country = 'Taiwan' where country = 'Taiwan (Province of China)';

update Natural_Disasters set country = 'Bolivia' where country = 'Bolivia (Plurinational State of)';
update Natural_Disasters set country = 'Bosnia-Herzegovinia' where country = 'Bosnia and Herzegovina';
update Natural_Disasters set country = 'Burkina_Faso' where country = 'Burkina Faso';

update Natural_Disasters set country = 'Central_African_Rep' where country = 'Central African Republic';
update Natural_Disasters set country = 'China' where country = 'Macao';
update Natural_Disasters set country = 'Congo'
                where country = 'Congo (the Democratic Republic of the)'
                or country = 'Congo (the)';
update Natural_Disasters set country = 'Costa_Rica' where country = 'Costa Rica';
update Natural_Disasters set country = 'Czech_Republic' where country = 'Czech Republic (the)';

update Natural_Disasters set country = 'Dominican_Republic' where country = 'Dominican Republic (the)'
                or country = 'Dominica';
update Natural_Disasters set country = 'El_Salvador' where country = 'El Salvador';
update Natural_Disasters set country = 'Equatorial_Guinea' where country = 'Equatorial Guinea';
```

Figure 62: Modifying country names part1 – to be continued

```sql
update Natural_Disasters set country = 'France' where country = 'Martinique' or country = 'Guadeloupe'
                        or country = 'New Caledonia' or country = 'Réunion'
                        or country = 'French Polynesia' or country = 'Wallis and Futuna'
                        or country = 'French Guiana' or country = 'Saint Barthélemy'
                        or country = 'Saint Martin (French Part)';
update Natural_Disasters set country = 'Germany' where country = 'Germany Fed Rep'
                        or country = 'Germany Dem Rep';
update Natural_Disasters set country = 'Gambia' where country = 'Gambia (the)';
update Natural_Disasters set country = 'Iran' where country = 'Iran (Islamic Republic of)';
update Natural_Disasters set country = 'Ivory_Coast' where country = 'Ivory Coast';
update Natural_Disasters set country = 'Slovakia' where country = 'Czechoslovakia';
update Natural_Disasters set country = 'Laos' where country like 'Lao %';
update Natural_Disasters set country = 'Macedonia'
                        where country = 'Macedonia (the former Yugoslav Republic of)';
update Natural_Disasters set country = 'Moldavia' where country = 'Moldova (the Republic of)';
update Natural_Disasters set country = 'Netherlands' where country = 'Netherlands (the)'
                        or country = 'Netherlands Antilles'
                        or country = 'Sint Maarten (Dutch part)';
update Natural_Disasters set country = 'New_Zealand' where country = 'Tokelau'
                        or country = 'New Zealand' or country = 'Cook Islands (the)';
update Natural_Disasters set country = 'Niger' where country = 'Niger (the)';
update Natural_Disasters set country = 'South_Korea' where country = 'Korea (the Republic of)';
update Natural_Disasters set country = 'North_Korea' where country like 'Korea (%';
```

Figure 63: Modifying country names part2 – to be continued

```sql
update Natural_Disasters set country = 'No_data' where country = 'Cabo Verde'
                        or country = 'Saint Vincent and the Grenadines'
                        or country = 'Comoros (the)' or country = 'Hong Kong'
                        or country = 'Puerto Rico' or country = 'Anguilla'
                        or country = 'Bahamas (the)' or country = 'Saint Kitts and Nevis'
                        or country = 'Fiji' or country = 'Belize' or country = 'Solomon Islands'
                        or country = 'Vanuatu' or country = 'Yugoslavia' or country = 'Tonga'
                        or country = 'Antigua and Barbuda' or country = 'Barbados'
                        or country = 'Niue' or country = 'Saint Lucia' or country = 'Grenada'
                        or country = 'Suriname' or country = 'Kiribati' or country = 'Tuvalu'
                        or country = 'Maldives' or country = 'Luxembourg'
                        or country = 'Sao Tome and Principe'
                        or country = 'Micronesia (Federated States of)'
                        or country = 'Marshall Islands (the)' or country = 'Seychelles'
                        or country = 'Brunei Darussalam'
                        or country = 'Northern Mariana Islands (the)' or country = 'Palau';
update Natural_Disasters set country = 'Papua_New_Guinea' where country = 'Papua New Guinea';
update Natural_Disasters set country = 'Philippines' where country = 'Philippines (the)';
update Natural_Disasters set country = 'Portugal' where country = 'Azores Islands';
update Natural_Disasters set country = 'Russia' where country = 'Soviet Union'
                        or country = 'Russian Federation (the)';
update Natural_Disasters set country = 'Saudi_Arabia' where country = 'Saudi Arabia';
update Natural_Disasters set country = 'Sierra_Leone' where country = 'Sierra Leone';
update Natural_Disasters set country = 'Saudi_Arabia' where country = 'Saudi Arabia';
update Natural_Disasters set country = 'South_Africa' where country = 'South Africa';
update Natural_Disasters set country = 'Spain' where country = 'Canary Is';
update Natural_Disasters set country = 'Sri_Lanka' where country = 'Sri Lanka';
update Natural_Disasters set country = 'Sudan' where country = 'Sudan (the)';
update Natural_Disasters set country = 'South_Sudan' where country = 'South Sudan';
```

Figure 64: Modifying country names part3 – to be continued

```sql
update Natural_Disasters set country = 'Syria' where country = 'Syrian Arab Republic';
update Natural_Disasters set country = 'Tanzania' where country = 'Tanzania, United Republic of';
update Natural_Disasters set country = 'Trinidad_and_Tobago' where country = 'Trinidad and Tobago';
update Natural_Disasters set country = 'United_Arab_Emirates'
                where country = 'United Arab Emirates (the)';
update Natural_Disasters set country = 'United_Kingdom' where country = 'Montserrat'
                or country = 'Bermuda'
                or country = 'United Kingdom of Great Britain and Northern Ireland (the)'
                or country = 'Turks and Caicos Islands (the)'
                or country = 'Virgin Island (British)'
                or country = 'Saint Helena, Ascension and Tristan da Cunha'
                or country = 'Cayman Islands (the)';
update Natural_Disasters set country = 'USA' where country = 'United States of America (the)'
                or country = 'Guam' or country = 'Samoa' or country = 'American Samoa'
                or country = 'Virgin Island (U.S.)';

update Natural_Disasters set country = 'Venezuela' where country = 'Venezuela (Bolivarian Republic of)';
update Natural_Disasters set country = 'Vietnam' where country = 'Viet Nam';
update Natural_Disasters set country = 'Yemen' where country = 'Yemen Arab Rep'
                or country = 'Yemen P Dem Rep';
update Natural_Disasters set country = 'Serbia' where country = 'Serbia Montenegro';
```

Figure 65: Modifying country names part4

Figure 66 presents the data of natural disasters is cleaned and exported for checking.

```sql
/* Note: missing country should be added
Kosovo
*/

select count(*) from Natural_Disasters;

/* Remove row if the country has NULL */
delete from Natural_Disasters where country = 'No_data';

select count(*) from Natural_Disasters;

/* Check counry name should be 162 -- Kosovo will added later */
select distinct(country) from natural_disasters order by country;
select * from natural_disasters
    into outfile 'C:\\Wakako\\NCI\\2020_Data_Analytics_MSc\\Program\\Output2\\ND3_check.csv'
    fields terminated by ',' enclosed by '"' LINES TERMINATED BY '\n';
```

Figure 66: Data of Natural Disasters is cleaned and exported

4.2.2 Data 5: Covid 19 information

The key purpose in data 5 is to obtain country, total case, and total death (both per million people) and population on 08/06/2020 which as the latest information at the point of time.

Figure 67 presents how to create Covid19 table and load the data set.

```
/* create COV19 table */
create table COV19 (
    iso_code varchar(3) NOT NULL, country varchar(32) NOT NULL, date date NOT NULL, total_cases int,
    new_cases int, total_deaths int, new_deaths int, total_cases_per_million int,
    new_cases_per_million int, total_deaths_per_million int, new_deaths_per_million int,
    total_tests int, new_tests int, total_tests_per_thousand int, new_tests_per_thousand int,

    tests_units varchar(43), population int, population_density int, median_age int, aged_65_older int,
    aged_70_older int, gdp_per_capita int, extreme_poverty int, cvd_death_rate int,
    diabetes_prevalence int, female_smokers int, male_smokers int, handwashing_facilities int,
    hospital_beds_per_100k int);

/* import COVID-19 data */
load data local infile 'C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/Output2/owid-covid-data.csv'
        into table ND3.COV19 fields terminated by ',' enclosed by '"' ignore 1 lines;

/* check number of observation */
select count(*) from COV19;

/* keep only latest date information which is 2020/05/17*/
create table COV19_0517 as (
    select country,
           total_cases_per_million,
           total_deaths_per_million,
           population
    from COV19 where date = '2020/06/08' order by country);
```

Figure 67: Create table and load the data for Covid 19

Figure 69 presents checking country names and missing country.

```
/* Set the same country name with GPI data */
/* find different naming of coutries */
create table country_adj3(
    select a.Country as COV_C,
           b.Country as GPI_C
    from COV19_0517 a left join GPI b on a.Country = b.Country);

create table country_adj4(
    select a.Country as COV_C,
               b.Country as GPI_C
    from COV19_0517 a right join GPI b on a.Country = b.Country);

/* export data for contry name and missing country checking */
select 'COV_Country', 'GPI_Country'
union all
select * from country_adj3
    into outfile 'C:\\Wakako\\NCI\\2020_Data_Analytics_MSc\\Program\\Output2\\Missing_check_GPI.csv'
    fields terminated by ',' enclosed by '"' LINES TERMINATED BY '\n';

select 'COV_Country', 'GPI_Country'
union all
select * from country_adj4
    into outfile 'C:\\Wakako\\NCI\\2020_Data_Analytics_MSc\\Program\\Output2\\Missing_check_COV.csv'
    fields terminated by ',' enclosed by '"' LINES TERMINATED BY '\n';
```

Figure 68: Country name, missing country check

Figure 69 and 70 present adjusting the country name, remove countries if those are not in the GPI list and add dummy country record for known missing country from GPI list.

```
/* Update Country name in COV19_0517 data -- should be the same country name as the GPI*/
update COV19_0517 set country = 'Bosnia-Herzegovinia' where country = 'Bosnia and Herzegovina';
update COV19_0517 set country = 'Burkina_Faso' where country = 'Burkina Faso';
update COV19_0517 set country = 'Central_African_Rep' where country = 'Central African Republic';
update COV19_0517 set country = 'Costa_Rica' where country = 'Costa Rica';
update COV19_0517 set country = 'Czech_Republic' where country = 'Czech Republic';
update COV19_0517 set country = 'Dominican_Republic' where country = 'Dominican Republic';
update COV19_0517 set country = 'El_Salvador' where country = 'El Salvador';
update COV19_0517 set country = 'Equatorial_Guinea' where country = 'Equatorial Guinea';
update COV19_0517 set country = 'Ivory_Coast' where country = 'Ivory Coast';
update COV19_0517 set country = 'Moldavia' where country = 'Moldova';
update COV19_0517 set country = 'Morocco' where country = 'Western Sahara';
update COV19_0517 set country = 'New_Zealand' where country = 'New Zealand';
update COV19_0517 set country = 'Papua_New_Guinea' where country = 'Papua New Guinea';
update COV19_0517 set country = 'Saudi_Arabia' where country = 'Saudi Arabia';
update COV19_0517 set country = 'Sierra_Leone' where country = 'Sierra Leone';
update COV19_0517 set country = 'South_Africa' where country = 'South Africa';
update COV19_0517 set country = 'South_Korea' where country = 'South Korea';
update COV19_0517 set country = 'Sri_Lanka' where country = 'Sri Lanka';
update COV19_0517 set country = 'Rep_Congo' where country = 'Democratic Republic of Congo';
update COV19_0517 set country = 'South_Sudan' where country = 'South Sudan';
update COV19_0517 set country = 'Timor-Leste' where country = 'Timor';
update COV19_0517 set country = 'Trinidad_and_Tobago' where country = 'Trinidad and Tobago';
update COV19_0517 set country = 'United_Arab_Emirates' where country = 'United Arab Emirates';
update COV19_0517 set country = 'Denmark' where country = 'Faeroe Islands' or country = 'Greenland';
update COV19_0517 set country = 'France' where country = 'French Polynesia' or country = 'New Caledonia';
update COV19_0517 set country = 'Netherlands' where country = 'Bonaire Sint Eustatius and Saba'
        or country = 'Sint Maarten (Dutch part)';
update COV19_0517 set country = 'USA' where country = 'Guam' or country = 'Northern Mariana Islands'
        or country = 'United States' or country = 'United States Virgin Islands';
```

Figure 69: Adjusting country name

```
update COV19_0517 set country = 'United_Kingdom' where country = 'Anguilla' or country = 'Bermuda'
        or country = 'British Virgin Islands' or country = 'Cayman Islands'
        or country = 'Falkland Islands' or country = 'Gibraltar' or country = 'Guernsey'
        or country = 'Isle of Man' or country = 'Jersey' or country = 'Montserrat'
        or country = 'Turks and Caicos Islands' or country = 'United Kingdom';
update COV19_0517 set country = 'No_data' where country = 'Andorra' or country = 'Antigua and Barbuda'
        or country = 'Aruba' or country = 'Bahamas' or country = 'Barbados'
        or country = 'Belize' or country = 'Brunei' or country = 'Cape Verde'
        or country = 'Comoros' or country = 'Curacao' or country = 'Dominica' or country = 'Fiji'
        or country = 'Grenada' or country = 'Liechtenstein' or country = 'Luxembourg'
        or country = 'Maldives' or country = 'Malta' or country = 'Monaco'
        or country = 'Puerto Rico' or country = 'Saint Kitts and Nevis'
        or country = 'Saint Lucia' or country = 'Saint Vincent and the Grenadines'
        or country = 'San Marino' or country = 'Sao Tome and Principe' or country = 'Seychelles'
        or country = 'Suriname' or country = 'Vatican' or country = 'World';

/* Remove row if the country has NULL */
delete from COV19_0517 where country = 'No_data';

/* Add dummy 0 data for North_Korea and Turkmenistan */
insert into COV19_0517 values('North_Korea', 0, 0, 0);
insert into COV19_0517 values('Turkmenistan', 0, 0, 0);

/* Check country -- should be 163 */
select distinct(country) from COV19_0517 order by country;
```

Figure 70: Adjusting country name and dealing missing data

Figure 71 obtaining count of death per million people and insert dummy record for missing country.

```
/* Death per million by Natural Disasters  -- populiration data is in COV19_0517 table */
create table ND_DPM as (
    select a.Country,
           a.Disaster_Type,
           round(sum(a.No_Deaths) * 1000000 / b.population, 1) as ND_DPM
    from natural_disasters as a,
         COV19_0517 as b
    where a.Country = b.Country
    group by a.Country, a.Disaster_Type);

/* Insert dummy record for Kosovo */
insert into ND_DPM values('Kosovo', 'Storm', 0);
```

Figure 71: Converting to per million and set dummy data for missing

Figure 72 presents transposing data set by disaster type from data 4 and merge with Covid selected data which are number of new infected cases, number of death and death rate. (Original source: OMG Ponies[4])

```
/* join COVID case and death per million and number of death by Natural Disasters per million  */
create table ND_DPM_F as(
select a.Country,
  max(case when a.Disaster_Type = 'Animal accident' then a.ND_DPM ELSE 0 END) as Animal,
  max(case when a.Disaster_Type = 'Drought' then a.ND_DPM ELSE 0 END) as Drought,
  max(case when a.Disaster_Type = 'Earthquake' then a.ND_DPM ELSE 0 END) as EarthQ,
  max(case when a.Disaster_Type = 'Epidemic' then a.ND_DPM ELSE 0 END) as Epidemic,
  max(case when a.Disaster_Type = 'Extreme temperature' then a.ND_DPM ELSE 0 END) as Ex_temp,
  max(case when a.Disaster_Type = 'Flood' then a.ND_DPM ELSE 0 END) as Flood,
  max(case when a.Disaster_Type = 'Fog' then a.ND_DPM ELSE 0 END) as Fog,
  max(case when a.Disaster_Type = 'Impact' then a.ND_DPM ELSE 0 END) as Impact,
  max(case when a.Disaster_Type = 'Insect infestation' then a.ND_DPM ELSE 0 END) as Insect,
  max(case when a.Disaster_Type = 'Landslide' then a.ND_DPM ELSE 0 END) as LandS,
  max(case when a.Disaster_Type = 'Mass movement (dry)' then a.ND_DPM ELSE 0 END) as Mass_move,
  max(case when a.Disaster_Type = 'Storm' then a.ND_DPM ELSE 0 END) as Storm,
  max(case when a.Disaster_Type = 'Volcanic activity' then a.ND_DPM ELSE 0 END) as Volcanic,
  max(case when a.Disaster_Type = 'Wildfire' then a.ND_DPM ELSE 0 END) as Wildfire,
  b.COV_CASE_PM,
  b.COV_DEATH_PM,
  b.COV_DEATH_PM/b.COV_CASE_PM as COV_Death_Rate
from
  ND_DPM a left join COVID_DATA b on a.Country = b.Country
group by
  a.Country);
```

Figure 72: Transpose data set

[4] https://stackoverflow.com/questions/3392956/sql-how-to-transpose

In Figure 73, as for the death rate calculation in Figure 72 above, its rate cannot have correct value when the denominator Covid new cases are 0 value. Therefore, when Covid new cases is 0, set 0 in death rate at the top of the programming in Figure 73. The total disaster number per country was counted and join with GPI score to the main data. Set the final headers and export as the final natural disasters data ND3_ALL.csv. This ends the program of MySQL.

```
/* set 0 in Cov Death rate if Denominator(Cov Case) is 0 */
update ND_DPM_F set COV_Death_Rate= 0 where COV_CASE_PM = 0;

/* Join Total disaster count per country in the final data and create as ND_ALL (final data) */
create table total_cnt as(
    select Country,
           count(Disaster_Type) as Total_Disaster
    from Natural_disasters group by Country);

create table ND_TCNT as(
    select a.*,
           b.Total_Disaster
    from ND_DPM_F as a left join total_cnt as b on a.Country = B.Country group by a.Country);

/* Join GPI ranking score from GPI data */
create table ND_ALL as(
    select a.*,
           b.score
    from ND_TCNT as a left join GPI as b on a.Country = B.Country group by a.Country);


/* Extract Natural Disaster final data to CSV with header */
select 'Country', 'Animal', 'Drought', 'EarthQ', 'Epidemic', 'Ex_temp', 'Flood', 'Fog', 'Impact',
       'Insect', 'LandS', 'Mass_move', 'Storm', 'Volcanic', 'Wildfire', 'COV_CASE', 'COV_DEATH',
       'COV_D_Rate', 'Total_Disaster', 'GPI_Score'
union all
select * from ND_ALL
    into outfile 'C:\\Wakako\\NCI\\2020_Data_Analytics_MSc\\Program\\ND3_All.csv'
    fields terminated by ',' enclosed by '"' LINES TERMINATED BY '\n';
```

Figure 73: Finalize the MySQL programming and export Natural Disasters data

To follow up the missing data, the operation was moved to R from MySQL. Figure 74 presents the preparation for running the random forest.

```
# import Natural Disaster
ND_ALL <- read.csv(file="ND3_ALL.csv", head=T, sep=",", na.strings = c("..", "NA"))
str(ND_ALL)

# remove 99% 0 data column (Animal, Fog, Impact and Insect)
ND_ALL <- ND_ALL[,c(-2, -8:-10)]
ND_ALL$Country <- as.character(ND_ALL$Country)
ND_ALL <- ND_ALL[,c(1:11,15,12:14,16)]
str(ND_ALL)
summary(ND_ALL)

# Kosovo didn't have Natural Disasters data
# --> predict by Random forest based on Covid and GPI score

    # set NA
    ND_ALL$Drought[ND_ALL$Country     == "Kosovo" & ND_ALL$Drought     == 0 ] <- NA
    ND_ALL$EarthQ[ND_ALL$Country      == "Kosovo" & ND_ALL$EarthQ      == 0 ] <- NA
    ND_ALL$Epidemic[ND_ALL$Country    == "Kosovo" & ND_ALL$Epidemic    == 0 ] <- NA
    ND_ALL$Ex_temp[ND_ALL$Country     == "Kosovo" & ND_ALL$Ex_temp     == 0 ] <- NA
    ND_ALL$Flood[ND_ALL$Country       == "Kosovo" & ND_ALL$Flood       == 0 ] <- NA
    ND_ALL$LandS[ND_ALL$Country       == "Kosovo" & ND_ALL$LandS       == 0 ] <- NA
    ND_ALL$Mass_move[ND_ALL$Country   == "Kosovo" & ND_ALL$Mass_move   == 0 ] <- NA
    ND_ALL$Storm[ND_ALL$Country       == "Kosovo" & ND_ALL$Storm       == 0 ] <- NA
    ND_ALL$Volcanic[ND_ALL$Country    == "Kosovo" & ND_ALL$Volcanic    == 0 ] <- NA
    ND_ALL$Wildfire[ND_ALL$Country    == "Kosovo" & ND_ALL$Wildfire    == 0 ] <- NA
    ND_ALL$Total_Disaster[ND_ALL$Country  == "Kosovo" & ND_ALL$Total_Disaster == 0 ] <- NA
```

Figure 74: Natural Disasters data was imported in R and preparing for missing data

Figure 75 presents conducting random forest to fill up the missing data. The mice function will predict and fill up if it found NA in dataset. And complete the data preparation for natural disasters.

```
# use 4 factors to judge the missing data, (those data no observation missing)
library(mice)
mice_mod_nd <- mice(ND_ALL[, !names(ND_ALL) %in%
                         c(12:16)], m = 3, maxit = 3, method ='rf')
mice_output_nd <- complete(mice_mod_nd)

# Replace missing data in GW_ALL main data
ND_ALL$Drought   <- mice_output_nd$Drought
ND_ALL$EarthQ    <- mice_output_nd$EarthQ
ND_ALL$Epidemic  <- mice_output_nd$Epidemic
ND_ALL$Ex_temp   <- mice_output_nd$Ex_temp
ND_ALL$Flood     <- mice_output_nd$Flood
ND_ALL$LandS     <- mice_output_nd$LandS
ND_ALL$Mass_move <- mice_output_nd$Mass_move
ND_ALL$Storm     <- mice_output_nd$Storm
ND_ALL$Volcanic  <- mice_output_nd$Volcanic
ND_ALL$Wildfire  <- mice_output_nd$Wildfire
ND_ALL$Total_Disaster <- mice_output_nd$Total_Disaster

write.csv(ND_ALL, "ND_ALL.csv")  #Final Natural Disaster data
```

Figure 75: conducting random forest for missing data and complete data preparation

4.3 Data Preparation Programming and Results by R

Figure 76 presents the programming and the results of missing data check. It is confirmed that there is no missing data in both GW_ALL and ND_ALL data sets. (Original Source: NCI ADM Class Text)

```
> # missing data check
> GW_MC <- sapply(GW_ALL, function(x) sum(is.na(x)))
> GW_MC
  Country    Cld_cv  Temp_day    Gnd_Fr   Pot_Eva      Prcp  Mean_Tmp   Vap_prs    Rn_day
        0         0         0         0         0         0         0         0         0
  Tp_diff   Tp_IncR  CO2_1970  CO2_2017  Inc_Rate GPI_Score
        0         0         0         0         0         0
> ND_MC <- sapply(ND_ALL, function(x) sum(is.na(x)))
> ND_MC
       Country        Drought         EarthQ       Epidemic        Ex_temp          Flood
             0              0              0              0              0              0
         LandS      Mass_move          Storm       Volcanic       Wildfire Total_Disaster
             0              0              0              0              0              0
      COV_CASE      COV_DEATH     COV_D_Rate      GPI_Score
             0              0              0              0
```

Figure 76: Missing data check

4.3.1 Programming and Results of Data Distribution Check By histogram

Figure 77 presents programming of GW_ALL histogram. 13 graphs were created with blue.

```
# Global Warming data
par(mfrow=c(3,5))
hist(GW_ALL$Cld_cv,   main = "Cloud Cover", col = "blue")
hist(GW_ALL$Temp_day, main = "Diurnal Temperature Range", col = "blue")
hist(GW_ALL$Gnd_Fr,   main = "Ground Frost Frequency", col = "blue")
hist(GW_ALL$Pot_Eva,  main = "Potential Evapotranspiration", col = "blue")
hist(GW_ALL$Prcp,     main = "Precipitation", col = "blue")
hist(GW_ALL$Mean_Tmp, main = "Mean Temperature", col = "blue")
hist(GW_ALL$Vap_prs,  main = "Vapour Pressure", col = "blue")
hist(GW_ALL$Rn_day,   main = "Rain Days", col = "blue")
hist(GW_ALL$Tp_diff,  main = "Temp difference 1910's vs 2000's", col = "blue")
hist(GW_ALL$Tp_IncR,  main = "Temp Increase rate", col = "blue")
hist(GW_ALL$CO2_1970, main = "CO2 Emissions in 1970", col = "blue")
hist(GW_ALL$CO2_2017, main = "CO2 Emissions in 2017", col = "blue")
hist(GW_ALL$Inc_Rate, main = "CO2 Inc Rate 1970/2017", col = "blue")
```

Figure 77: Histogram programming of GW_ALL

Figure 78 presents the histogram of global warming factors. Cloud cover, diurnal temperature range, vapour pressure, rain days, temp difference 1910's vs 2000's and temp increase rate are showing good distribution in the shape. Ground forest frequency, precipitation, CO2 emissions in 1970, CO2 emissions in 2017 and CO2 increase rate 1970/2017 are positively skewed distribution. Potential evapotranspiration and mean temperature are negatively skewed distribution. Potential evapotranspiration, temp increase rate, CO2 emissions in 2017 and CO2 increase rate 1970/2017 are showing positive kurtosis which are showing over 100 in y-axis scale.

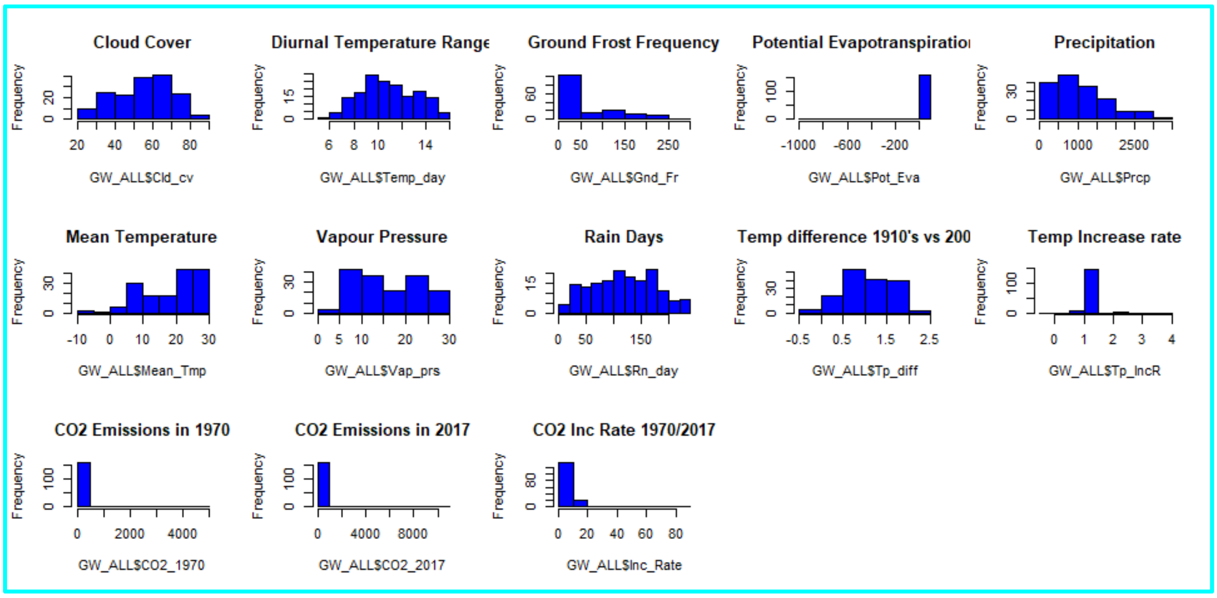


Figure 78: Histogram of Global Warming data sets

Figure 79 presents programming of ND_ALL histogram. 14 graphs were created with orange.

```
# Natural Disaster data
par(mfrow=c(3,5))
hist(ND_ALL$Drought,    main = "Drought", col = "Orange")
hist(ND_ALL$EarthQ,     main = "Earthquake", col = "Orange")
hist(ND_ALL$Epidemic,   main = "Epidemic", col = "Orange")
hist(ND_ALL$Ex_temp,    main = "Ex_temp", col = "Orange")
hist(ND_ALL$Flood,      main = "Flood", col = "Orange")
hist(ND_ALL$LandS,      main = "Land Slide", col = "Orange")
hist(ND_ALL$Mass_move,  main = "Mass move", col = "Orange")
hist(ND_ALL$Storm,      main = "Storm", col = "Orange")
hist(ND_ALL$Volcanic,   main = "Volcanic", col = "Orange")
hist(ND_ALL$Wildfire,   main = "Wildfire", col = "Orange")
hist(ND_ALL$COV_CASE,   main = "COVID CASE", col = "Orange")
hist(ND_ALL$COV_DEATH,  main = "COVID DEATH", col = "Orange")
hist(ND_ALL$COV_D_Rate, main = "COV Death Rate", col = "Orange")
hist(ND_ALL$Total_Dis,  main = "Total Disaster", col = "Orange")
```

Figure 79: Histogram programming of ND_ALL

Figure 80 presents the histogram of natural disasters factors. All data sets are positively skewed. Apart from Covid death rate and total disaster, data sets are showing positive kurtosis which are showing over 100 in y-axis scale. The data distribution in this dataset does not seem very good.

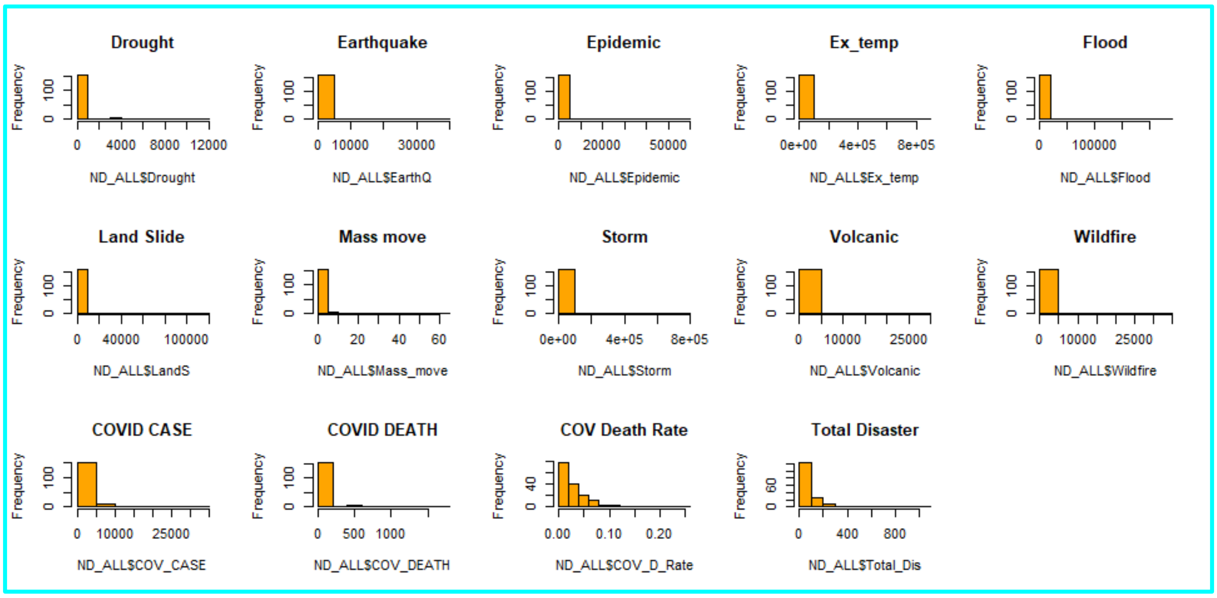


Figure 80: Histogram of Natural Disasters data sets

4.3.2 Outlier check

Figure 81 presents Q-Q plot for outlier checking in global warming data set.

```
# outlier check by Q-Q plot -----------------------------------
# Global Warming data
par(mfrow=c(3,5))
qqnorm(GW_ALL$Cld_cv,   main = "Cloud Cover")
qqline(GW_ALL$Cld_cv,   col = "blue")
qqnorm(GW_ALL$Temp_day, main = "Diurnal Temperature Range")
qqline(GW_ALL$Temp_day, col = "blue")
qqnorm(GW_ALL$Gnd_Fr,   main = "Ground Frost Frequency")
qqline(GW_ALL$Gnd_Fr,   col = "blue")
qqnorm(GW_ALL$Pot_Eva,  main = "Potential Evapotranspiration")
qqline(GW_ALL$Pot_Eva,  col = "blue")
qqnorm(GW_ALL$Prcp,     main = "Precipitation")
qqline(GW_ALL$Prcp,     col = "blue")
qqnorm(GW_ALL$Mean_Tmp, main = "Mean Temperature")
qqline(GW_ALL$Mean_Tmp, col = "blue")
qqnorm(GW_ALL$Vap_prs,  main = "Vapour Pressure")
qqline(GW_ALL$Vap_prs,  col = "blue")
qqnorm(GW_ALL$Rn_day,   main = "Rain Days")
qqline(GW_ALL$Rn_day,   col = "blue")
qqnorm(GW_ALL$Tp_diff, main = "Temp Diff 1910's vs 2000's")
qqline(GW_ALL$Tp_diff, col = "blue")
qqnorm(GW_ALL$Tp_IncR, main = "Temp Increase rate")
qqline(GW_ALL$Tp_IncR, col = "blue")
qqnorm(GW_ALL$CO2_1970, main = "CO2 Emissions in 1970")
qqline(GW_ALL$CO2_1970, col = "blue")
qqnorm(GW_ALL$CO2_2017, main = "CO2 Emissions in 2017")
qqline(GW_ALL$CO2_2017, col = "blue")
qqnorm(GW_ALL$Inc_Rate, main = "Increase Rate 1970/2017")
qqline(GW_ALL$Inc_Rate, col = "blue")
```

Figure 81: Programming of Q-Q plot for Global Warming

Figure 82 presents Q-Q plots of global warming data. cloud cover and diurnal temperature range, potential evapotranspiration, precipitation, and rain days are good fit in the line and no outliers. Ground forest frequency, mean temperature, vapour pressure are temp diff 1910's vs 2000's are less good fit in the line as it can been seen the plots don't keep straight line from the start to the end, otherwise there is no significant outlier. Temp increase rate, CO2 emissions in 1970, CO2 emission in 2017 and Increase rate 1970/2017 are showing that the dots leave the line very far toward the end, and outlier can be found in each chart.

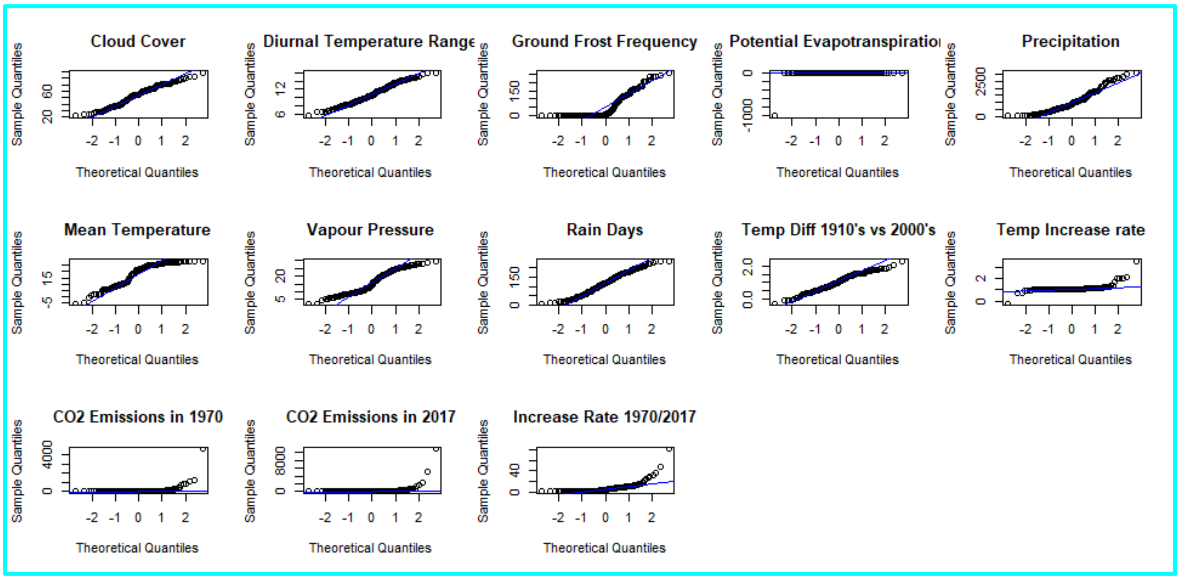


Figure 82: Q-Q Plot of Global Warming data set

Figure 83 presents Q-Q plot for outlier checking in natural disasters data set.

```
# Natural Disaster data
par(mfrow=c(3,5))
qqnorm(ND_ALL$Drought,   main = "Drought")
qqline(ND_ALL$Drought,   col = "Orange")
qqnorm(ND_ALL$EarthQ,    main = "Earthquake")
qqline(ND_ALL$EarthQ,    col = "Orange")
qqnorm(ND_ALL$Epidemic,  main = "Epidemic")
qqline(ND_ALL$Epidemic,  col = "Orange")
qqnorm(ND_ALL$Ex_temp,   main = "Ex_temp")
qqline(ND_ALL$Ex_temp,   col = "Orange")
qqnorm(ND_ALL$Flood,     main = "Flood")
qqline(ND_ALL$Flood,     col = "Orange")
qqnorm(ND_ALL$LandS,     main = "Land Slide")
qqline(ND_ALL$LandS,     col = "Orange")
qqnorm(ND_ALL$Mass_move, main = "Mass move")
qqline(ND_ALL$Mass_move, col = "Orange")
qqnorm(ND_ALL$Storm,     main = "Storm")
qqline(ND_ALL$Storm,     col = "Orange")
qqnorm(ND_ALL$Volcanic,  main = "Volcanic")
qqline(ND_ALL$Volcanic,  col = "Orange")
qqnorm(ND_ALL$Wildfire,  main = "Wildfire")
qqline(ND_ALL$Wildfire,  col = "Orange")
qqnorm(ND_ALL$COV_CASE,  main = "COVID CASE")
qqline(ND_ALL$COV_CASE,  col = "Orange")
qqnorm(ND_ALL$COV_DEATH, main = "COVID DEATH")
qqline(ND_ALL$COV_DEATH, col = "Orange")
qqnorm(ND_ALL$COV_D_Rate, main = "COV Death RATE")
qqline(ND_ALL$COV_D_Rate, col = "Orange")
qqnorm(ND_ALL$Total_Disaster, main = "Total Disaster")
qqline(ND_ALL$Total_Disaster, col = "Orange")
```

Figure 83 Programming of Q-Q plot for Natural Disasters

Figure 84 presents Q-Q plot of natural disasters data set. Drought, earthquake, epidemic, extreme temp, flood, land slide, mass move, storm, volcanic and wildfire have similar shape and plots are almost following straight line up to 2 in x-axis, and each one has one or more significant outlier(s). Covid case, Covid death, Covid death rate and total disaster have similar shape which the plots are following the straight line up to 1, and start increasing after 1 until toward end in x-axis, and those has one are more significant outlier(s).

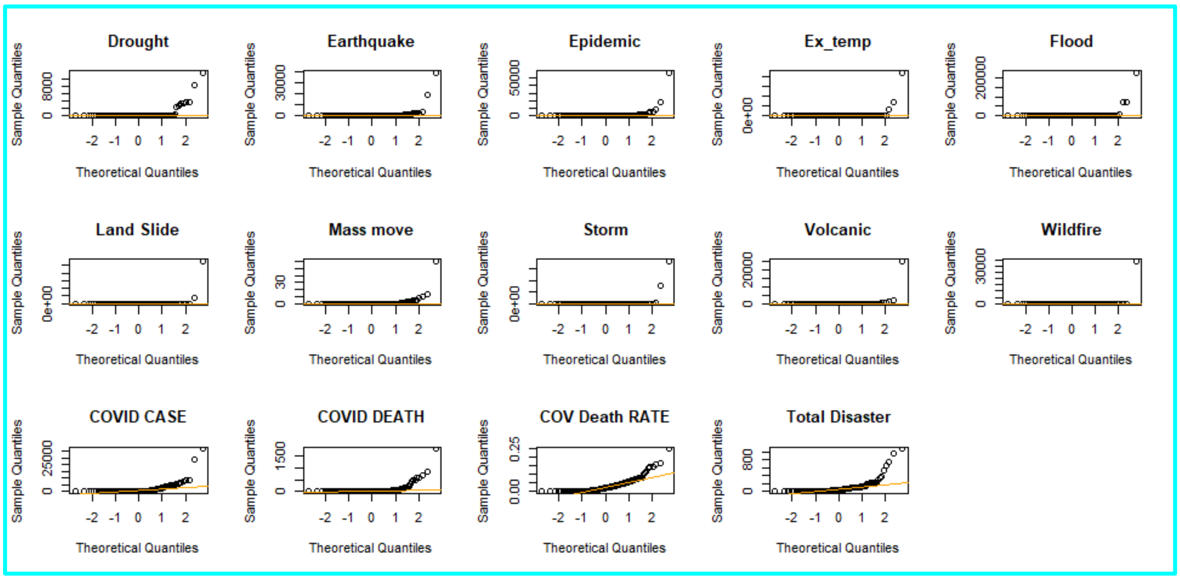


Figure 84: Q-Q Plot of Natural Disasters data set

Figure 85 presents the programming of replacing outlier with median for global warming data.

```
# Remove outliers -- median is used instead of mean in outlier replacement.
library(outliers)
# Median
GW_OUT <- GW_ALL
str(GW_ALL)
GW_OUT$Cld_cv   <- rm.outlier(GW_ALL$Cld_cv,   fill = TRUE, median = TRUE)
GW_OUT$Temp_day <- rm.outlier(GW_ALL$Temp_day, fill = TRUE, median = TRUE)
GW_OUT$Gnd_Fr   <- rm.outlier(GW_ALL$Gnd_Fr,   fill = TRUE, median = TRUE)
GW_OUT$Pot_Eva  <- rm.outlier(GW_ALL$Pot_Eva,  fill = TRUE, median = TRUE)
GW_OUT$Prcp     <- rm.outlier(GW_ALL$Prcp,     fill = TRUE, median = TRUE)
GW_OUT$Mean_Tmp <- rm.outlier(GW_ALL$Mean_Tmp, fill = TRUE, median = TRUE)
GW_OUT$Vap_prs  <- rm.outlier(GW_ALL$Vap_prs,  fill = TRUE, median = TRUE)
GW_OUT$Rn_day   <- rm.outlier(GW_ALL$Rn_day,   fill = TRUE, median = TRUE)
GW_OUT$Tp_diff  <- rm.outlier(GW_ALL$Tp_diff,  fill = TRUE, median = TRUE)
GW_OUT$Tp_IncR  <- rm.outlier(GW_ALL$Tp_IncR,  fill = TRUE, median = TRUE)
GW_OUT$CO2_1970 <- rm.outlier(GW_ALL$CO2_1970, fill = TRUE, median = TRUE)
GW_OUT$CO2_2017 <- rm.outlier(GW_ALL$CO2_2017, fill = TRUE, median = TRUE)
GW_OUT$Inc_Rate <- rm.outlier(GW_ALL$Inc_Rate, fill = TRUE, median = TRUE)

Final_GW_ALL <- GW_OUT # Outlier removed
write.csv(Final_GW_ALL, "Final_GW_ALL.csv")
```

Figure 85: Programming of outlier treatment for Global Warming data set

Figure 86 presents the programming of replacing outlier with median for natural disaster data,

```
# Median
ND_OUT <- ND_ALL
str(ND_ALL)
ND_OUT$Drought    <- rm.outlier(ND_ALL$Drought,    fill = TRUE, median = TRUE)
ND_OUT$EarthQ     <- rm.outlier(ND_ALL$EarthQ,     fill = TRUE, median = TRUE)
ND_OUT$Epidemic   <- rm.outlier(ND_ALL$Epidemic,   fill = TRUE, median = TRUE)
ND_OUT$Ex_temp    <- rm.outlier(ND_ALL$Ex_temp,    fill = TRUE, median = TRUE)
ND_OUT$Flood      <- rm.outlier(ND_ALL$Flood,      fill = TRUE, median = TRUE)
ND_OUT$LandS      <- rm.outlier(ND_ALL$LandS,      fill = TRUE, median = TRUE)
ND_OUT$Mass_move  <- rm.outlier(ND_ALL$Mass_move,  fill = TRUE, median = TRUE)
ND_OUT$Storm      <- rm.outlier(ND_ALL$Storm,      fill = TRUE, median = TRUE)
ND_OUT$Volcanic   <- rm.outlier(ND_ALL$Volcanic,   fill = TRUE, median = TRUE)
ND_OUT$Wildfire   <- rm.outlier(ND_ALL$Wildfire,   fill = TRUE, median = TRUE)
ND_OUT$COV_CASE   <- rm.outlier(ND_ALL$COV_CASE,   fill = TRUE, median = TRUE)
ND_OUT$COV_DEATH  <- rm.outlier(ND_ALL$COV_DEATH,  fill = TRUE, median = TRUE)
ND_OUT$COV_D_Rate <- rm.outlier(ND_ALL$COV_D_Rate, fill = TRUE, median = TRUE)
ND_OUT$Total_Disaster <- rm.outlier(ND_ALL$Total_Disaster, fill = TRUE, median


Final_ND_ALL <- ND_OUT # Outlier removed
write.csv(Final_ND_ALL, "Final_ND_ALL.csv")
```

Figure 86: Programming of outlier treatment for Natural Disaster data set

4.3.3 Normality Test

Figure 87 presents programming of Shapiro-Wilk normality test for global warming data.

```
#   Shapiro-Wilk works for sample size of 3 to 5000
# Global Warming -----
S_Cld_cv   <- shapiro.test(GW_OUT$Cld_cv)
S_Temp_day <- shapiro.test(GW_OUT$Temp_day)
S_Gnd_Fr   <- shapiro.test(GW_OUT$Gnd_Fr)
S_Pot_Eva  <- shapiro.test(GW_OUT$Pot_Eva)
S_Prcp     <- shapiro.test(GW_OUT$Prcp)
S_Mean_Tmp <- shapiro.test(GW_OUT$Mean_Tmp)
S_Vap_prs  <- shapiro.test(GW_OUT$Vap_prs)
S_Rn_day   <- shapiro.test(GW_OUT$Rn_day)
S_Tp_diff  <- shapiro.test(GW_OUT$Tp_diff)
S_Tp_IncR  <- shapiro.test(GW_OUT$Tp_IncR)
S_CO2_1970 <- shapiro.test(GW_OUT$CO2_1970)
S_CO2_2017 <- shapiro.test(GW_OUT$CO2_2017)
S_Inc_Rate <- shapiro.test(GW_OUT$Inc_Rate)

Norm_Pvalue <- c(S_Cld_cv$p.value, S_Temp_day$p.value, S_Gnd_Fr$p.value,
                 S_Pot_Eva$p.value, S_Prcp$p.value, S_Mean_Tmp$p.value,
                 S_Vap_prs$p.value, S_Rn_day$p.value,
                 S_Tp_diff$p.value, S_Tp_IncR$p.value,
                 S_CO2_1970$p.value, S_CO2_2017$p.value, S_Inc_Rate$p.value)

Attribute <-   c('Cloud Cover', 'Diurnal Temperature Range',
                 'Ground Frost Frequency', 'Potential Evapotranspiration',
                 'Precipitation', 'Mean Temperature', 'Vapour Pressure',
                 'Rain Days', 'Temp Difference', 'Temp Inc Rate',
                 'CO2 1970', 'CO2 2017', 'Increase Rate 1970/2017')

GW_Norm_result <- cbind.data.frame(Attribute,Norm_Pvalue)
GW_Norm_result
```

Figure 87: Programming for Normality test of Global Warming

Figure 88 presents programming of Shapiro-Wilk normality test for natural disasters data

```
# Natural Disaster -----
S_Drought    <- shapiro.test(ND_OUT$Drought)
S_EarthQ     <- shapiro.test(ND_OUT$EarthQ)
S_Epidemic   <- shapiro.test(ND_OUT$Epidemic)
S_Ex_temp    <- shapiro.test(ND_OUT$Ex_temp)
S_Flood      <- shapiro.test(ND_OUT$Flood)
S_LandS      <- shapiro.test(ND_OUT$LandS)
S_Mass_move  <- shapiro.test(ND_OUT$Mass_move)
S_Storm      <- shapiro.test(ND_OUT$Storm)
S_Volcanic   <- shapiro.test(ND_OUT$Volcanic)
S_Wildfire   <- shapiro.test(ND_OUT$Wildfire)
S_COV_CASE   <- shapiro.test(ND_OUT$COV_CASE)
S_COV_DEATH  <- shapiro.test(ND_OUT$COV_DEATH)
S_COV_D_Rate <- shapiro.test(ND_OUT$COV_D_Rate)
S_Total_Disaster <- shapiro.test(ND_OUT$Total_Disaster)


# remove Drought, Mass_move and Volcanic ---> 0 input
Norm_Pvalue <- c(S_Drought$p.value, S_EarthQ$p.value, S_Epidemic$p.value,
                 S_Ex_temp$p.value, S_Flood$p.value, S_LandS$p.value,
                 S_Mass_move$p.value, S_Storm$p.value, S_Volcanic$p.value,
                 S_Wildfire$p.value, S_COV_CASE$p.value, S_COV_DEATH$p.value,
                 S_COV_D_Rate$p.value, S_Total_Disaster$p.value)

Attribute <-   c('Drought','Earthquake', 'Epidemic', 'Ex_temp', 'Flood',
                 'Land Slide', 'Mass_move', 'Storm', 'Volcanic', 'Wildfire',
                 'Covid Case pm', 'Covid Death pm', 'Cov Death Rate',
                 'Total Disaster')

ND_Norm_result <- cbind.data.frame(Attribute,Norm_Pvalue)
ND_Norm_result
```

Figure 88: Programming for Normality test of Natural Disasters

Figure 89 presents the results of Shapiro-Wilk normality test from global warming and natural disasters. None of them had a P value of 0.05 or more, and the data distribution status was not shown to be normal in both datasets.

```
> GW_Norm_result
                     Attribute  Norm_Pvalue
1                  Cloud Cover 1.640013e-03
2     Diurnal Temperature Range 1.166132e-02
3        Ground Frost Frequency 4.245524e-15
4  Potential Evapotranspiration 3.365806e-02
5                 Precipitation 5.397832e-07
6              Mean Temperature 1.385559e-09
7               Vapour Pressure 1.211062e-06
8                     Rain Days 8.720414e-03
9               Temp Difference 1.620008e-02
10                Temp Inc Rate 2.061860e-20
11                     CO2 1970 1.964711e-23
12                     CO2 2017 1.283017e-24
13       Increase Rate 1970/2017 3.116158e-16
```

```
> ND_Norm_result
        Attribute  Norm_Pvalue
1         Drought 8.476282e-26
2      Earthquake 1.398374e-26
3        Epidemic 8.843834e-26
4         Ex_temp 1.763095e-27
5           Flood 2.611712e-27
6      Land Slide 9.348602e-28
7       Mass_move 9.699538e-25
8           Storm 8.895404e-28
9        Volcanic 1.020312e-26
10       Wildfire 3.655533e-25
11  Covid Case pm 4.486983e-21
12 Covid Death pm 9.169961e-23
13 Cov Death Rate 3.304107e-13
14 Total Disaster 5.322328e-20
```

Figure 89: Result of Normality Test for Global Warming and Natural Disasters

4.3.4 Correlation check

Figure 90 presents programming of correlation check.

```
# Correlation check --------------
library(GGally)
# Global Warming data
str(GW_OUT)
dataG <- GW_OUT[,2:14]
ggpairs(dataG, title="Correlation of Global Warming Data")

# Natural Disaster data
str(ND_OUT)
dataN <- ND_OUT[,2:15]
ggpairs(dataN, title="Correlation of Natural Disasters Data")
```

Figure 90: Program of correlation check for Global Warming and Natural Disasters

In Figure 91, those with a correlation of 60% are indicate in blue and Cld_Cv, Gnd_Fr, Pot_Eva, Vap_prs, Rn_day were removed in case1.

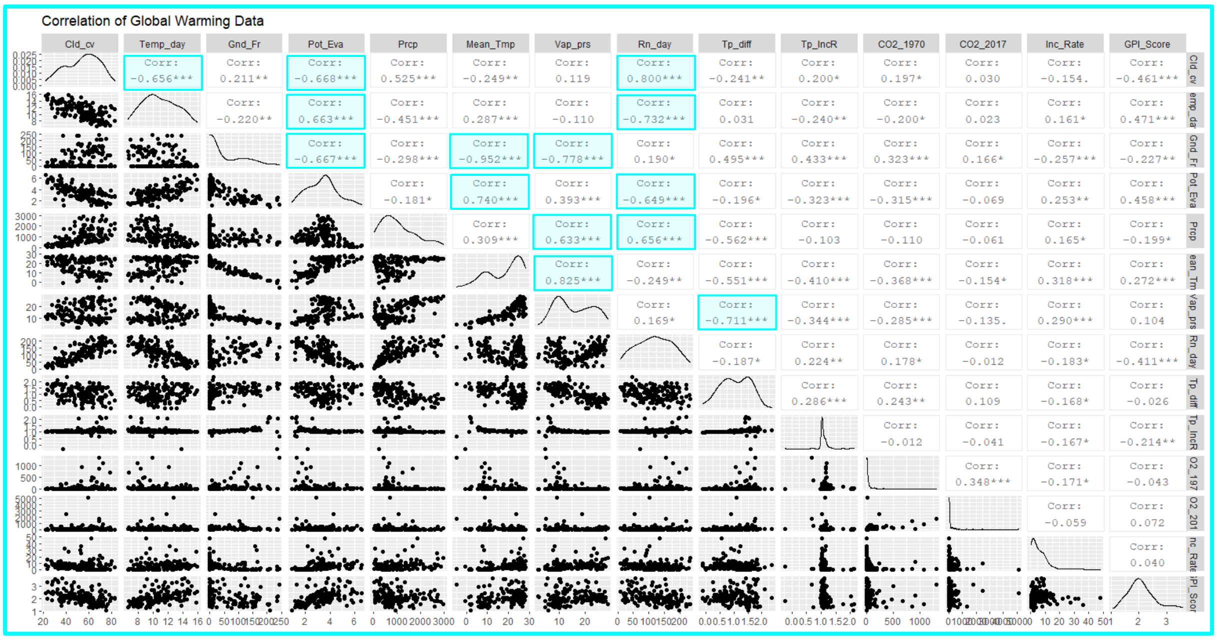


Figure 91: Pair matrix of Global Warming

In Figure 92, those with a correlation of 60% are indicate in blue and Epidemic, Flood, LandS, Storm, COV_D_Rate were removed for case1.

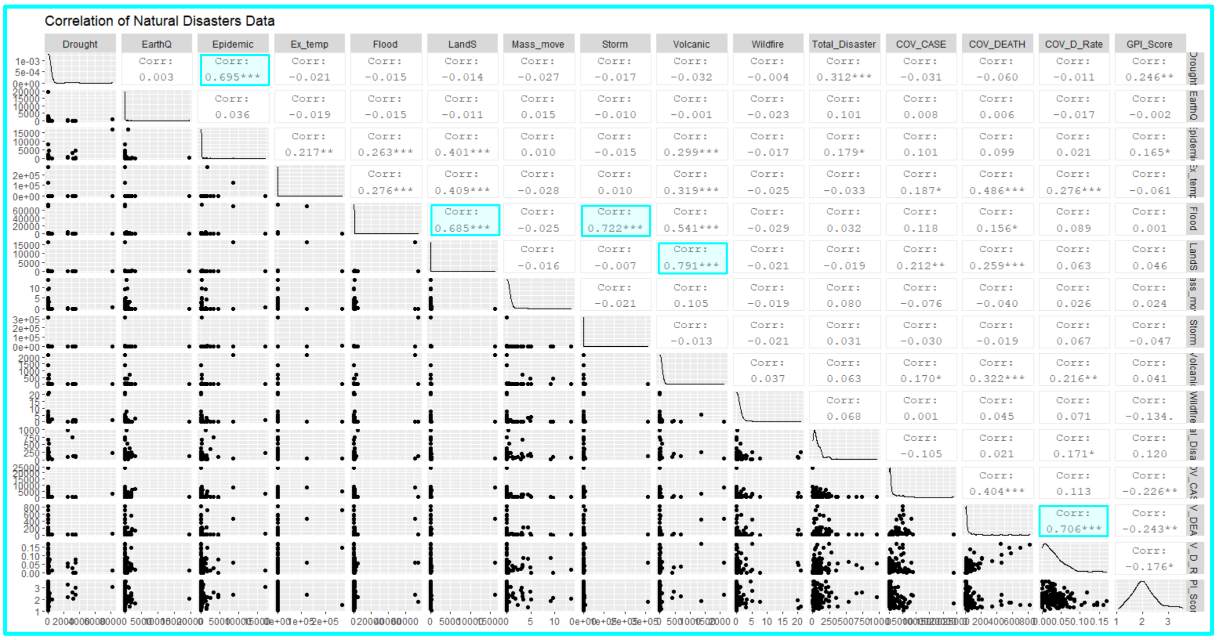


Figure 92: Pair matrix of Natural Disasters

4.3.5 Model selection by AIC

Case 1: Figure 93 presents the programming of case 1. (Original Source: R Documentation[5])

```
# =======
# case 1: remove highly correlated data manually
# ---> remove over 60% correlation Cld_Cv, Gnd_Fr, Pot_Eva, Vap_prs, Rn_day
GW_update <- GW_OUT[,c(-2,-4,-5,-8,-9)]
str(GW_update)

# ---> remove Epidemic, Flood, LandS, Storm, COV_D_Rate
ND_update <- ND_OUT[,c(-4,-6,-7,-9,-15)]
str(ND_update)

# Multivariate regression (by AIC) --------------
library(stats) # Lib for AIC
# select factors from GW
model.lmG1 <- lm(GPI_Score ~., data = GW_update[c(2:9)])
summary(model.lmG1)
model.lmG2<-step(model.lmG1)

# select factors from ND
model.lmN1 <- lm(GPI_Score ~., data = ND_update[c(2:11)])
summary(model.lmN1)
model.lmN2<-step(model.lmN1)

# joint selected data
GW1 <- GW_update[,c(1,2,4,7)]
ND1 <- ND_update[,c(1,2,6,7,9:11)]
Final1 <- merge(x=GW1, y=ND1, by="Country", all.x=TRUE)
str(Final1)

model.lmF1 <- lm(GPI_Score ~., data = Final1[c(2:10)])
summary(model.lmF1)
par(mfrow=c(2,2),oma = c(1,1,2,1),mar = c(4, 4, 2, 1))
plot(model.lmF1,pch=21,bg=1,col=1,cex=1.0)

# run final model
model.lmF1_2 <- step(model.lmF1)
```

Figure 93: Programming of Case 1

[5] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/step.html

Figure 94 shows the results of case 1 that R^2 is 0.3018, and Temp_day, Drought, wildfire and COV_CASE are effective at P-value < 0.05.

```
> summary(model.lmF1)

Call:
lm(formula = GPI_Score ~ ., data = Final1[c(2:10)])

Residuals:
     Min       1Q   Median       3Q      Max 
-0.90819 -0.25939 -0.03848  0.13617  1.64225 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.046e+00  1.929e-01   5.422 2.24e-07 ***
Temp_day     8.476e-02  1.626e-02   5.212 5.91e-07 ***
Mean_Tmp     9.068e-03  4.649e-03   1.951   0.0529 .  
CO2_1970     5.031e-05  2.265e-04   0.222   0.8245    
Drought      1.115e-04  4.338e-05   2.571   0.0111 *  
Volcanic     2.731e-04  1.702e-04   1.605   0.1105    
Wildfire    -2.372e-02  1.171e-02  -2.026   0.0445 *  
COV_CASE    -2.865e-05  1.438e-05  -1.992   0.0481 *  
COV_DEATH   -3.000e-04  3.236e-04  -0.927   0.3553    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4272 on 154 degrees of freedom
Multiple R-squared:  0.3363,	Adjusted R-squared:  0.3018 
F-statistic: 9.752 on 8 and 154 DF,  p-value: 6.633e-11
```

Figure 94: Results of Case 1

Regarding the result plot Figure 95, 99,103,115,121 and 125, there are four plots in each figure. This section explains how to interpret each plot.

Residuals vs Fitted plot tells weather the residuals have a non-linear pattern. If dots are on these 0 lines, it means 0 residual, above this line means positive residuals and below are negative residuals. The red line shows the pattern of residual movement (Deo, 2016). The ideal plots would have a symmetrical pattern around the 0 line with no clear pattern in the data, no well-defined shape, no clear outliers, and no large residuals. The good indication is non-linear relationships. (Hagerman, 2017).

Normal Q-Q plot checks if the distribution of residual is normal or not. It is considered as normally distributes when the dots follow to the line closely (Hagerman, 2017).

Scale-Location plot shows the spread of points over the range of predicted values and simplifies homoscedasticity analysis, which is one of the regression assumptions. If the graph shows horizontal line, it is considered as data is homoscedastic and not horizontal it is considered as data is heteroscedastic (Deo, 2016; Kim, 2015). The ideal plots would be that the red line is almost horizontal and the plots spread around the red line shouldn’t be vary with the fitted values (ALEX, 2019).

Residuals vs Leverage plot shows the cases that impact. Cook's distance is a good measure to consider in terms of how much the prediction score changes when an observation is excluded. Cook's distance is indicated by the red dotted lines in the graph, and the areas of interest are the top right and bottom right corners outside the dotted line and excluding points in that area can affect the model. (Kim, 2015; Deo, 2016).

Figure 95 presents plotted results of case 1. Residuals vs Fitted plot shows positive residuals are more than negative residuals, there is not characteristic pattern, relatively shapeless, plot 134 might be considered as an outlier, overall, this can be considered as a good pattern. Normal Q-Q, up to 1 on the x-axis shows a good straight line but after that plot start leaving the line. The plot 134 might be a potential issue. Scale-Location plot tells slight horizontal line, plots are well spread around the line. Residuals vs Leverage shows that there are no dots outside of Cook's distance. It might be considered that there is no influential case.

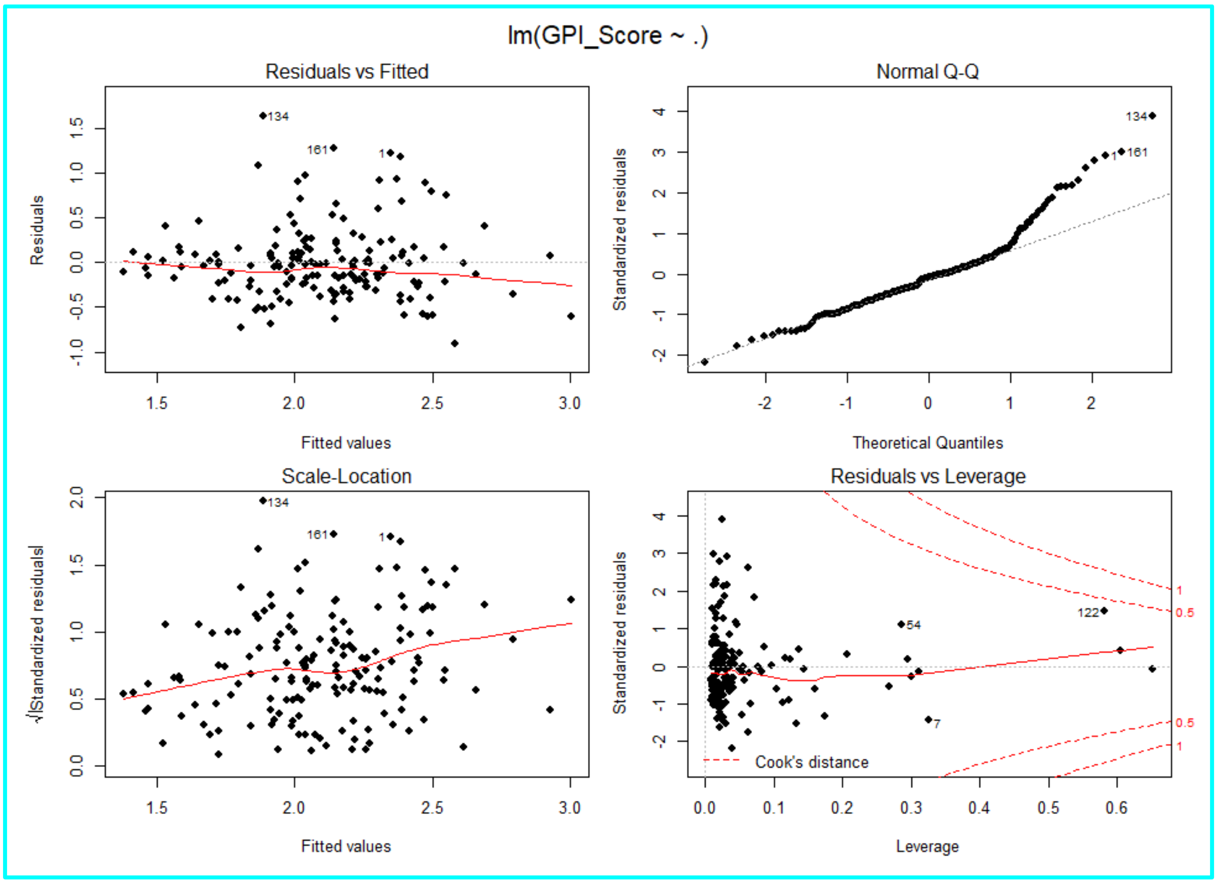


Figure 95: Plotted results of Case 1

Figure 96 presents the result of AIC = -271.6 and selected model by step function in R.

```
Step:  AIC=-271.6
GPI_Score ~ Temp_day + Mean_Tmp + Drought + Volcanic + Wildfire +
    COV_CASE

           Df Sum of Sq    RSS     AIC
<none>                  28.265 -271.60
- Volcanic  1    0.3644 28.629 -271.51
- Wildfire  1    0.7645 29.029 -269.25
- Mean_Tmp  1    0.8924 29.157 -268.53
- COV_CASE  1    1.1153 29.380 -267.29
- Drought   1    1.7505 30.015 -263.80
- Temp_day  1    5.4688 33.734 -244.77
```

Figure 96: Result model of Case 1

Case 2: Figure 97 presents the programming.

```
# =======
# case 2: select by model
# select factors from GW
model.lmG1_ALL <- lm(GPI_Score ~., data = GW_OUT[c(2:15)])
summary(model.lmG1_ALL)
model.lmG2_ALL <- step(model.lmG1_ALL)

# select factors from ND
model.lmN1_ALL <- lm(GPI_Score ~., data = ND_OUT[c(2:16)])
summary(model.lmN1_ALL)
model.lmN2_ALL <- step(model.lmN1_ALL)

# joint selected data
# Build Final data --- merge the result from GW and ND
GW2 <- GW_OUT[,c(1:3,8,12,14)]
ND2 <- ND_OUT[,c(1,2,10,11,13,14,16)]
Final2 <- merge(x=GW2, y=ND2, by="Country", all.x=TRUE)
str(Final2)

model.lmF2 <- lm(GPI_Score ~., data = Final2[c(2:12)])
summary(model.lmF2)
par(mfrow=c(2,2),oma = c(1,1,2,1),mar = c(4, 4, 2, 1))
plot(model.lmF2,pch=21,bg=1,col=1,cex=1.0)
model.lmF2_2 <- step(model.lmF2)
```

Figure 97: Programming of Case 2

Figure 98 shows the results of case 2 that R^2 is 0.365, and Cld_cv, Temp_day, Vap_prs, Drought, wildfire and COV_CASE are effective at P-value < 0.05.

```
> summary(model.lmF2)

Call:
lm(formula = GPI_Score ~ ., data = Final2[c(2:12)])

Residuals:
     Min       1Q   Median       3Q      Max 
-0.70897 -0.25430 -0.06911  0.17762  1.41427 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.078e+00  3.582e-01   5.801 3.70e-08 ***
Cld_cv      -1.224e-02  2.994e-03  -4.088 7.02e-05 ***
Temp_day     5.311e-02  1.976e-02   2.688   0.0080 ** 
Vap_prs      1.271e-02  5.328e-03   2.385   0.0183 *  
CO2_1970     1.193e-04  2.184e-04   0.546   0.5857    
Inc_Rate    -8.568e-03  4.956e-03  -1.729   0.0859 .  
Drought      9.965e-05  4.157e-05   2.397   0.0177 *  
Volcanic     2.886e-04  1.635e-04   1.765   0.0796 .  
Wildfire    -2.951e-02  1.132e-02  -2.607   0.0100 *  
COV_CASE    -3.319e-05  1.390e-05  -2.387   0.0182 *  
COV_DEATH   -2.064e-04  3.137e-04  -0.658   0.5115    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4074 on 152 degrees of freedom
Multiple R-squared:  0.4042,   Adjusted R-squared:   0.365 
F-statistic: 10.31 on 10 and 152 DF,  p-value: 3.704e-13
```

Figure 98: Result of Case 2

Figure 99 presents plotted results of case 2. The shape of the plots is very similar to case 1. Residuals vs Fitted plot shows positive residuals are more than negative residuals, no characteristic pattern, relatively shapeless, plot 134 might be an outlier, overall, this can be a good pattern. Normal Q-Q, up to 1 on the x-axis shows a good straight line but after that plot start leaving the line. The plot 134 might be a potential issue. Scale-Location plot tells slight horizontal line, plots are well spread around the line. Residuals vs Leverage shows that there are no dots outside of Cook's distance. It might be considered that there is no influential case.

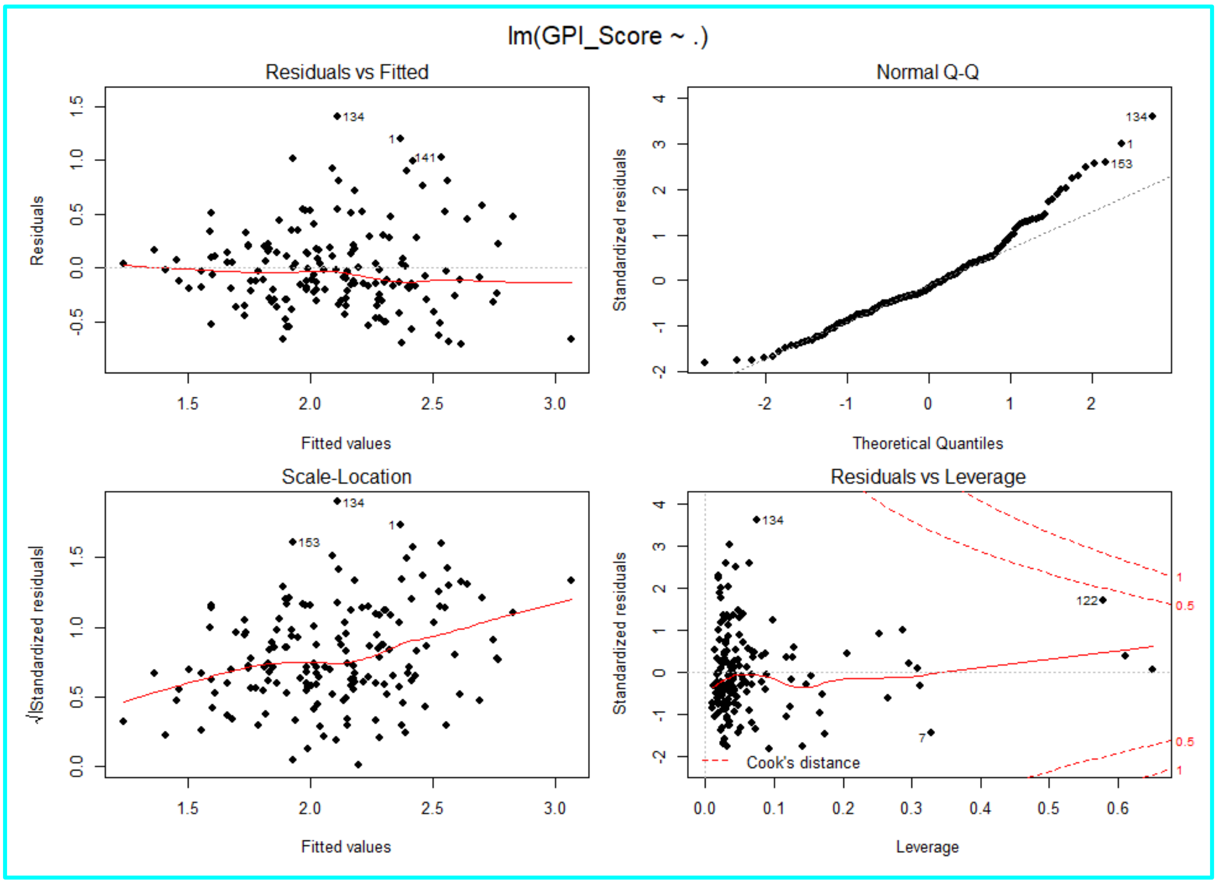


Figure 99: Plotted results of Case 2

Figure 100 presents the result of AIC = -285.37 and selected model by step function in R.

```
Step:  AIC=-285.37
GPI_Score ~ Cld_cv + Temp_day + Vap_prs + Inc_Rate + Drought +
    Volcanic + Wildfire + COV_CASE

           Df Sum of Sq    RSS     AIC
<none>                  25.344 -285.37
- Volcanic  1   0.47015 25.814 -284.38
- Inc_Rate  1   0.47593 25.820 -284.34
- Vap_prs   1   1.04686 26.391 -280.78
- Wildfire  1   1.13221 26.477 -280.25
- Temp_day  1   1.22308 26.567 -279.69
- COV_CASE  1   1.28765 26.632 -279.30
- Drought   1   1.59351 26.938 -277.44
- Cld_cv    1   2.86882 28.213 -269.89
```

Figure 100: Result model of Case 2

Case 3: Figure 101 presents the programming.

```
# =======
# case 3:
# Join both GW and ND data before AIC run
# total all
GW3 <- GW_OUT[,c(-15)]
ND3 <- ND_OUT
Final3 <- merge(x=GW3, y=ND3, by="Country", all.x=TRUE)
str(Final3)

model.lmF3 <- lm(GPI_Score ~., data = Final3[c(2:29)])
summary(model.lmF3)
par(mfrow=c(2,2),oma = c(1,1,2,1),mar = c(4, 4, 2, 1))
plot(model.lmF3,pch=21,bg=1,col=1,cex=1.0)
model.lmF3_2 <- step(model.lmF3)
```

Figure 101: Programming of Case 3

Figure 102 presents the results of case 3 that R^2 is 0.3187, and Cld_cv, wildfire and COV_CASE are effective at P-value < 0.05.

```
> summary(model.lmF3)

Call:
lm(formula = GPI_Score ~ ., data = Final3[c(2:29)])

Residuals:
     Min       1Q   Median       3Q      Max 
-0.68173 -0.24201 -0.05649  0.18635  1.43569 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)   
(Intercept)     1.993e+00  6.477e-01   3.078  0.00253 **
Cld_cv         -9.151e-03  4.467e-03  -2.049  0.04243 * 
Temp_day        3.844e-02  2.816e-02   1.365  0.17447   
Gnd_Fr         -1.475e-06  1.952e-03  -0.001  0.99940   
Pot_Eva         3.723e-02  7.169e-02   0.519  0.60441   
Prcp           -7.227e-05  9.789e-05  -0.738  0.46163   
Mean_Tmp        4.342e-04  2.302e-02   0.019  0.98498   
Vap_prs         1.417e-02  1.521e-02   0.932  0.35322   
Rn_day         -1.199e-04  1.444e-03  -0.083  0.93397   
Tp_diff         1.040e-01  1.050e-01   0.990  0.32373   
Tp_IncR        -7.205e-02  1.961e-01  -0.367  0.71383   
CO2_1970        3.155e-05  3.234e-04   0.098  0.92243   
CO2_2017       -1.713e-04  1.842e-04  -0.930  0.35398   
Inc_Rate       -1.035e-02  5.537e-03  -1.869  0.06379 . 
Drought         6.051e-05  6.317e-05   0.958  0.33984   
EarthQ         -6.499e-06  2.143e-05  -0.303  0.76216   
Epidemic        3.205e-05  4.112e-05   0.780  0.43705   
Ex_temp        -1.247e-07  1.811e-06  -0.069  0.94519   
Flood          -1.040e-04  9.561e-05  -1.087  0.27880   
LandS           4.969e-04  4.582e-04   1.085  0.28005   
Mass_move       1.185e-02  2.097e-02   0.565  0.57294   
Storm           2.580e-05  2.298e-05   1.122  0.26366   
Volcanic        3.245e-04  2.844e-04   1.141  0.25577   
Wildfire       -3.107e-02  1.249e-02  -2.487  0.01409 * 
Total_Disaster  7.671e-04  4.078e-04   1.881  0.06210 . 
COV_CASE       -4.195e-05  1.653e-05  -2.537  0.01232 * 
COV_DEATH       6.796e-05  5.194e-04   0.131  0.89610   
COV_D_Rate     -1.874e+00  1.706e+00  -1.098  0.27395   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.422 on 135 degrees of freedom
Multiple R-squared:  0.4323,	Adjusted R-squared:  0.3187 
F-statistic: 3.807 on 27 and 135 DF,  p-value: 1.289e-07
```

Figure 102: Result of Case 3

In Figure 103 Residuals vs Fitted plot shows positive residuals are more than negative residuals, no characteristic pattern, relatively shapeless, plot 134 might be an outlier, overall, this can be a good pattern. Normal Q-Q, up to 1 on the x-axis shows a good straight line but after that plot start leaving the line. The plot 134 might be a potential issue. Scale-Location plot tells slight horizontal line, plots are well spread around the line. Residuals vs Leverage shows that there is a dot outside of 1 Cook's distance on the right but very close to the line. It might be considered that there is not significant influential case.

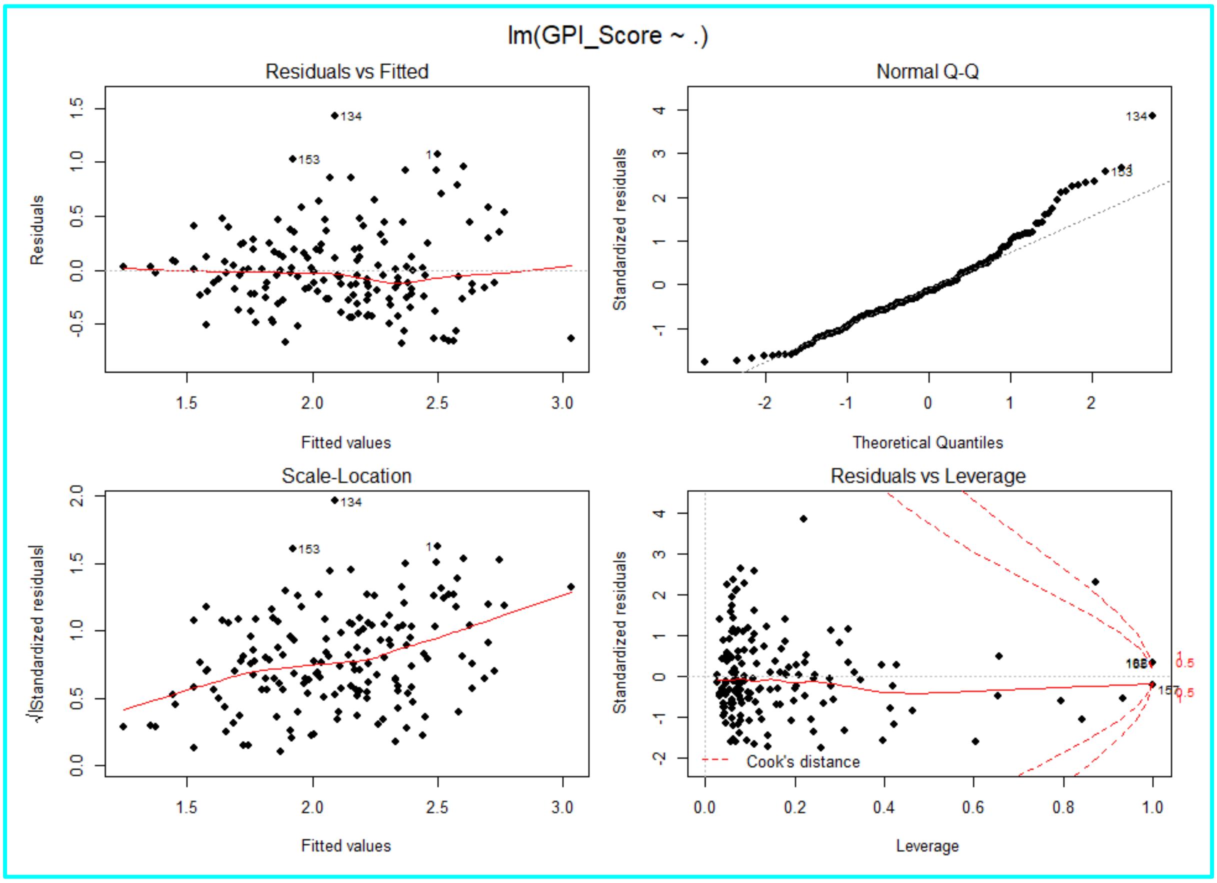


Figure 103: Plotted results of Case 3

Figure 104 presents the result of AIC=-285.37 and selected model by step function in R.

```
Step:  AIC=-285.37
GPI_Score ~ Cld_cv + Temp_day + Vap_prs + Inc_Rate + Drought +
    Volcanic + Wildfire + COV_CASE

           Df Sum of Sq    RSS     AIC
<none>                  25.344 -285.37
- Volcanic  1   0.47015 25.814 -284.38
- Inc_Rate  1   0.47593 25.820 -284.34
- Vap_prs   1   1.04686 26.391 -280.78
- Wildfire  1   1.13221 26.477 -280.25
- Temp_day  1   1.22308 26.567 -279.69
- COV_CASE  1   1.28765 26.632 -279.30
- Drought   1   1.59351 26.938 -277.44
- Cld_cv    1   2.86882 28.213 -269.89
```

Figure 104: Result model of Case 3

Case 4.1: Figure 105 presents the programming of case 4.1 natural logarithm transformation.

```
# =======
# case 4: Transform data
ALL <- merge(x=GW_OUT[,c(-15)], y=ND_OUT, by="Country", all.x=TRUE)
# ---------------------------------------------
# case 4.1: Natural Log Transformation
# ---------------------------------------------
LOG <- log(ALL[,c(2:29)])
```

Figure 105: Programming of Case 4.1

Figure 106 presents calculation error of case 4.1. This is invalid and no further test.

```
> LOG <- log(ALL[,c(2:29)])
Warning messages:
1: In FUN(X[[i]], ...) : NaNs produced
2: In FUN(X[[i]], ...) : NaNs produced
3: In FUN(X[[i]], ...) : NaNs produced
```

Figure 106: Result of Case 4.1

Case 4.2: Figure 107 presents the programming of natural logarithm 10 transformation.

```
# ---------------------------------------------
# case 4.2: Natural Log10 Transformation
# ---------------------------------------------
LOG10 <- log10(ALL[,c(2:29)])
```

Figure 107: Programming of Case 4.2

Figure 108 presents calculation error of case 4.2. This is invalid and no further test.

```
> LOG10 <- log10(ALL[,c(2:29)])
Warning messages:
1: In lapply(X = x, FUN = .Generic, ...) : NaNs produced
2: In lapply(X = x, FUN = .Generic, ...) : NaNs produced
3: In lapply(X = x, FUN = .Generic, ...) : NaNs produced
```

Figure 108: Result of Case 4.2

Case 4.3: Figure 109 presents the programming of square root transformation.

```
# ---------------------------------------------
# case 4.3: Square root Transformation
# ---------------------------------------------
SQRT <- sqrt(ALL[,c(2:29)])
```

Figure 109: Programming of Case 4.3

Figure 110 presents calculation error of case 4.3. This is invalid and no further test.

```
> SQRT <- sqrt(ALL[,c(2:29)])
Warning messages:
1: In FUN(X[[i]], ...) : NaNs produced
2: In FUN(X[[i]], ...) : NaNs produced
3: In FUN(X[[i]], ...) : NaNs produced
```

Figure 110: Result of Case 4.3

Case 4.4: Figure 111 presents the programming of 1/x transformation.

```
# ---------------------------------------------
# case 4.4: Reciprocal Transformation (1/x)
# ---------------------------------------------
D1 <- 1/ALL[,c(2:29)]
```

Figure 111: Programming of Case 4.4

Figure 112 presents the part of appearance of infinity. This is invalid and no further test.

```
> D1 <- 1/ALL[,c(2:29)]
> D1
       Cld_cv   Temp_day       Gnd_Fr   Pot_Eva         Prcp   Mean_Tmp    Vap_prs
1  0.02715967 0.07039210 8.149377e-03 0.2601552 0.0030899082 0.07905138 0.17506143
2  0.01783814 0.09901850 1.148117e-02 0.4035398 0.0009978031 0.08597933 0.10161333
3  0.02735190 0.06997299 6.702334e-02 0.2123696 0.0112511473 0.04426497 0.10773011
4  0.01586440 0.07418011 5.990541e-01 0.2960270 0.0010171542 0.04621559 0.05774783
5  0.02145438 0.07603041 1.463095e-02 0.2751629 0.0017909435 0.06733609 0.09093810
6  0.01877223 0.08720263 6.091696e-03 0.3828073 0.0017792567 0.14046328 0.16393443
7  0.02771363 0.07250064 9.064885e-02 0.1864878 0.0019452232 0.04631887 0.07686084
8  0.01554765 0.12577229 6.686923e-03 0.5943691 0.0008814824 0.15603613 0.12043102
9  0.01757415 0.09771987 9.734521e-03 0.3485173 0.0023569497 0.08367587 0.10234312
10 0.03042272 0.07607608 8.382353e+00 0.2914483 0.0122328097 0.03693743 0.06206108
11 0.01889482 0.11059371 5.428571e+01 0.3035952 0.0003713307 0.03999018 0.04086607
12 0.01443952 0.12099342 6.567653e-03 0.5903677 0.0015940823 0.15908457 0.11722365
13 0.01401507 0.11738056 1.130414e-02 0.5571848 0.0012022171 0.10186757 0.09746922
14 0.01442837 0.08445070          Inf 0.2416790 0.0009470436 0.03618818 0.04502903
```

Figure 112: Result of Case 4.4

Case 4.5: Figure 113 presents the programming of case 4.5.

```
# --------------------------------------------
# case 4.5: Power of 3
# --------------------------------------------
P3 <- (ALL[,c(2:29)])^3

model.lmP3 <- lm(GPI_Score ~., data = data.frame(P3[,c(1:28)]))
summary(model.lmP3)
par(mfrow=c(2,2),oma = c(1,1,2,1),mar = c(4, 4, 2, 1))
plot(model.lmP3,pch=21,bg=1,col=1,cex=1.0)
model.lmp3_2 <- step(model.lmP3)
```

Figure 113: Programming of Case 4.5

Figure 114 shows the results of case 4.5 that R^2 is 0.1577, and Temp_day is effective at P-value < 0.05.

```
> summary(model.lmP3)

Call:
lm(formula = GPI_Score ~ ., data = data.frame(P3[, c(1:28)]))

Residuals:
    Min      1Q  Median      3Q     Max 
-13.622  -4.466  -1.712   1.805  31.457 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)   
(Intercept)     1.079e+01  3.383e+00   3.189  0.00177 **
Cld_cv         -1.519e-05  8.018e-06  -1.894  0.06030 . 
Temp_day        2.478e-03  1.241e-03   1.997  0.04785 * 
Gnd_Fr         -5.390e-07  4.183e-07  -1.289  0.19973   
Pot_Eva         1.097e-02  2.285e-02   0.480  0.63206   
Prcp           -1.329e-10  1.860e-10  -0.714  0.47634   
Mean_Tmp       -2.327e-04  2.356e-04  -0.987  0.32525   
Vap_prs         3.513e-04  2.643e-04   1.329  0.18595   
Rn_day         -1.511e-07  3.698e-07  -0.409  0.68342   
Tp_diff         2.505e-01  4.048e-01   0.619  0.53697   
Tp_IncR         2.311e-01  7.456e-01   0.310  0.75706   
CO2_1970       -1.244e-09  6.093e-09  -0.204  0.83848   
CO2_2017       -5.110e-10  8.935e-10  -0.572  0.56838   
Inc_Rate       -8.358e-05  6.593e-05  -1.268  0.20710   
Drought         6.811e-11  1.131e-10   0.602  0.54799   
EarthQ         -4.056e-13  1.118e-12  -0.363  0.71732   
Epidemic       -9.134e-13  1.285e-11  -0.071  0.94341   
Ex_temp         1.052e-16  4.761e-16   0.221  0.82545   
Flood          -1.962e-10  2.520e-10  -0.778  0.43765   
LandS           1.717e-08  2.206e-08   0.778  0.43785   
Mass_move      -5.300e-04  3.109e-03  -0.170  0.86490   
Storm           2.768e-12  3.556e-12   0.779  0.43763   
Volcanic        3.947e-09  4.676e-09   0.844  0.40010   
Wildfire       -1.105e-03  5.859e-04  -1.886  0.06143 . 
Total_Disaster  2.205e-08  2.561e-08   0.861  0.39075   
COV_CASE       -8.391e-13  6.463e-13  -1.298  0.19637   
COV_DEATH       1.145e-08  2.278e-08   0.503  0.61600   
COV_D_Rate     -2.687e+03  2.300e+03  -1.168  0.24470   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.207 on 135 degrees of freedom
Multiple R-squared:  0.2981,	Adjusted R-squared:  0.1577 
F-statistic: 2.124 on 27 and 135 DF,  p-value: 0.002647
```

Figure 114: Result of Case 4.5

In Figure 115 Residuals vs Fitted plot shows positive residuals are more than negative residuals, there is a pattern as the plots are fitted near red line around 0 to 20 on x-axis and relatively linear, overall, this can be said as heteroskedasticity. Normal Q-Q, up to 1 on the x-axis shows a good straight line but after that plot start leaving the line. The plot 134 might be a potential issue. Scale-Location plot tells the plots spreading up to 20 on x-axis, and the red line is diagonal. Residuals vs Leverage shows that there are three plots outside of Cook's distance on the right middle and 5 plots on the right edge of red line. Those are appearing not on in the top and bottom of the right corner but there are quite a few plots. It can be considered that there might be some influential cases.

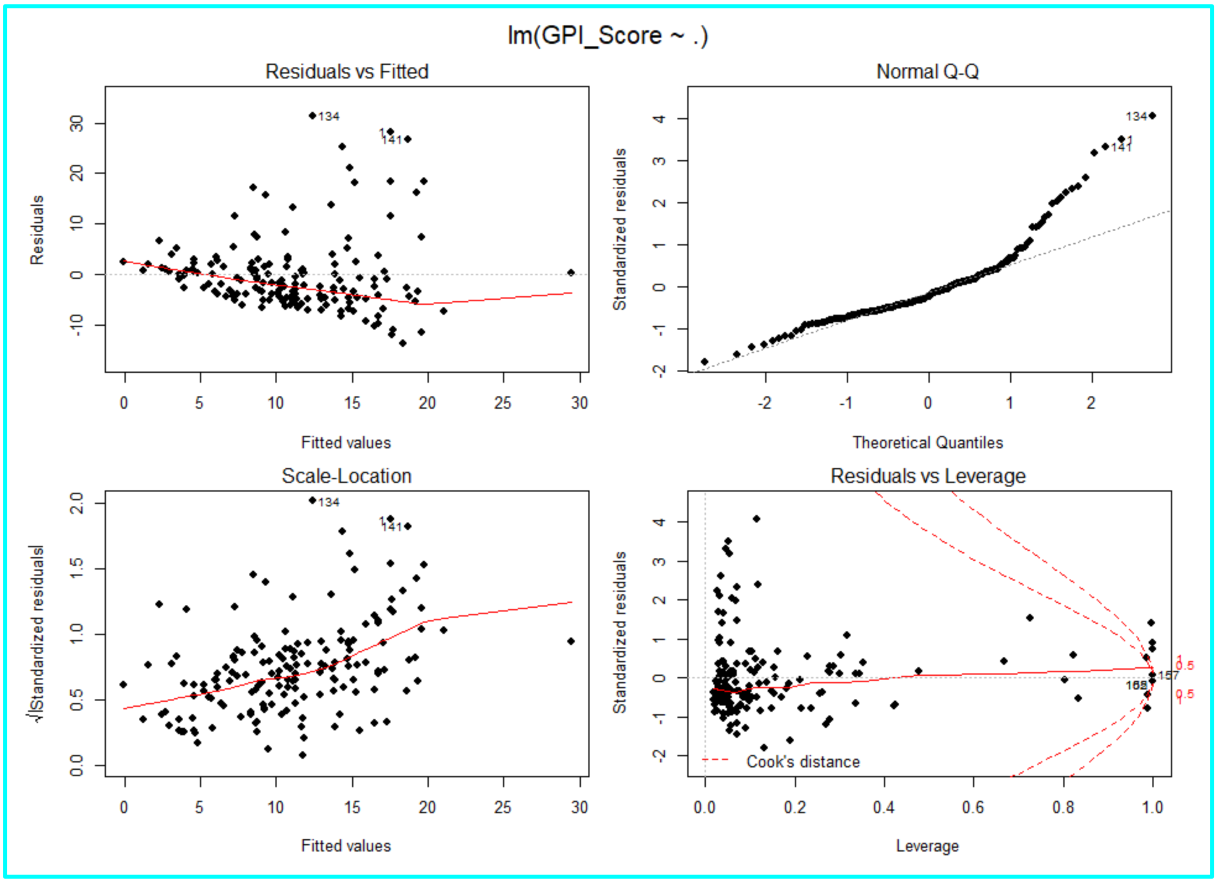


Figure 115: Plotted results of Case 4.5

Figure 116 presents the result of AIC= 677.26 and selected model by step function in R.

```
Step:  AIC=677.26
GPI_Score ~ Cld_cv + Temp_day + Gnd_Fr + Drought + Wildfire

           Df Sum of Sq     RSS    AIC
<none>                   9653.7 677.26
- Gnd_Fr    1    160.41  9814.1 677.95
- Wildfire  1    236.84  9890.5 679.21
- Temp_day  1    391.81 10045.5 681.74
- Cld_cv    1    595.79 10249.5 685.02
- Drought   1    657.65 10311.4 686.00
```

Figure 116: Result model of Case 4.5

Case 4.6: Figure 117 presents the programming of case 4.6.

```
# ----------------------------------------------
# case 4.6: exp
# ----------------------------------------------
ex <- exp(ALL[,c(2:29)])
```

Figure 117: Programming of Case 4.6

Figure 118 presents the part of appearance of infinity. This is invalid and no further test.

```
> ex <- exp(ALL[,c(2:29)])
> ex
        Cld_cv    Temp_day       Gnd_Fr   Pot_Eva          Prcp     Mean_Tmp
1 9.781784e+15 1477911.2789 1.957684e+53 46.705393 3.569043e+140 3.117634e+05
2 2.220221e+24   24321.6653 6.709164e+37 11.918242           Inf 1.124992e+05
```

Figure 118: Result of Case 4.6

Case 4.7: Figure 119 presents the programming of case 4.7.

```
# ----------------------------------------------
# case 4.7: sin
# ----------------------------------------------
sin <- sin(ALL[,c(2:29)])

model.lmsin <- lm(GPI_Score ~., data = data.frame(sin[,c(1:28)]))
summary(model.lmsin)
par(mfrow=c(2,2),oma = c(1,1,2,1),mar = c(4, 4, 2, 1))
plot(model.lmsin,pch=21,bg=1,col=1,cex=1.0)
model.lmsin_2 <- step(model.lmsin)
```

Figure 119 : Programming of Case 4.7

Figure 120 shows the results of case 4.7 that R^2 is 0.1318, and Pot_Eva, Tp_diff and LandS are effective at P-value < 0.05.

```
> summary(model.lmsin)

Call:
lm(formula = GPI_Score ~ ., data = data.frame(sin[, c(1:28)]))

Residuals:
     Min       1Q   Median       3Q      Max 
-1.07988 -0.09086  0.05214  0.14821  0.52564 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)     0.965459   0.263382   3.666 0.000354 ***
Cld_cv         -0.039922   0.037458  -1.066 0.288424    
Temp_day        0.011455   0.036780   0.311 0.755940    
Gnd_Fr         -0.057325   0.050110  -1.144 0.254659    
Pot_Eva         0.175172   0.040250   4.352 2.64e-05 ***
Prcp            0.058065   0.035507   1.635 0.104309    
Mean_Tmp       -0.007012   0.036841  -0.190 0.849328    
Vap_prs         0.050719   0.038491   1.318 0.189844    
Rn_day         -0.034626   0.035437  -0.977 0.330250    
Tp_diff        -0.267848   0.101161  -2.648 0.009068 ** 
Tp_IncR         0.019653   0.296425   0.066 0.947238    
CO2_1970       -0.039699   0.039496  -1.005 0.316619    
CO2_2017        0.023859   0.038091   0.626 0.532123    
Inc_Rate       -0.032065   0.037408  -0.857 0.392867    
Drought        -0.104254   0.089592  -1.164 0.246618    
EarthQ         -0.046609   0.055813  -0.835 0.405136    
Epidemic       -0.019187   0.042622  -0.450 0.653312    
Ex_temp        -0.084387   0.056823  -1.485 0.139853    
Flood          -0.021650   0.039602  -0.547 0.585490    
LandS           0.114141   0.052422   2.177 0.031192 *  
Mass_move      -0.078785   0.111501  -0.707 0.481040    
Storm          -0.070606   0.045965  -1.536 0.126859    
Volcanic       -0.020711   0.118981  -0.174 0.862071    
Wildfire        0.114431   0.078601   1.456 0.147758    
Total_Disaster  0.017128   0.036295   0.472 0.637759    
COV_CASE       -0.026106   0.036699  -0.711 0.478096    
COV_DEATH      -0.030622   0.040636  -0.754 0.452430    
COV_D_Rate      0.856158   0.830395   1.031 0.304373    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2996 on 135 degrees of freedom
Multiple R-squared:  0.2762,	Adjusted R-squared:  0.1314 
F-statistic: 1.908 on 27 and 135 DF,  p-value: 0.008722
```

Figure 120: Result of Case 4.7

In Figure 121 Residuals vs Fitted plot shows negative residuals are more than positive residuals, there is a pattern as the plots are fitted near red line around 0.6 to 1.0 on x-axis and relatively linear, overall, this can be said as heteroskedasticity. Normal Q-Q shows the plots start to -1 is away from the line and from 1 to the end on the x-axis shows a good straight line. The plot 134 might be a potential issue. Scale-Location plot tells the plots gathering 0.5 to 1.0 on x-axis, and the red line is diagonally going down toward 1.0. Residuals vs Leverage shows that a plot 95 is located outside of 0.5 Cook's distance on the corner. It can be considered that 95 might be some influential cases.

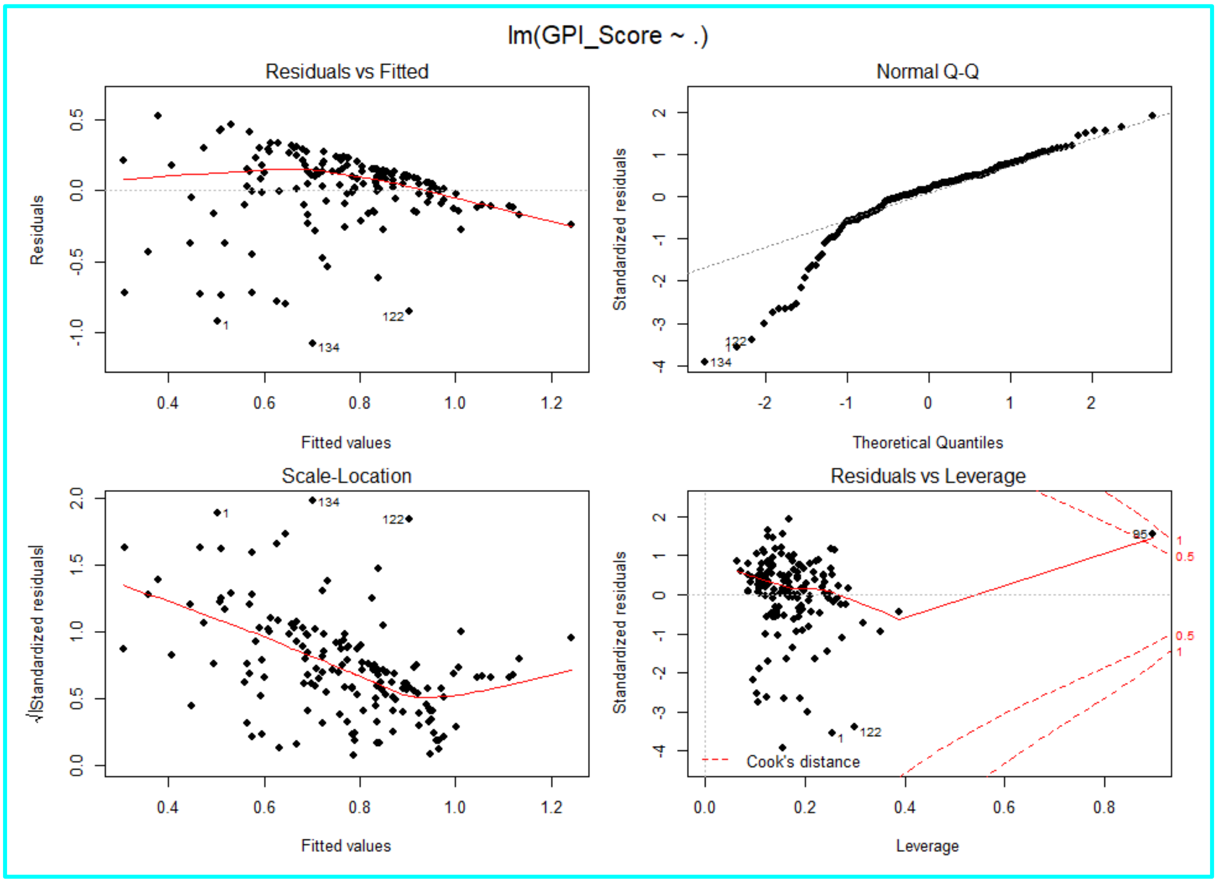


Figure 121: Plotted results of Case 4.7

Figure 122 presents the result of AIC = -394.38 and selected model by step function in R.

```
Step:  AIC=-394.38
GPI_Score ~ Pot_Eva + Prcp + Vap_prs + Tp_diff + LandS + Storm +
    Wildfire

           Df Sum of Sq    RSS     AIC
<none>                  13.146 -394.38
- Vap_prs   1   0.17320 13.319 -394.24
- Storm     1   0.24379 13.390 -393.38
- Prcp      1   0.24531 13.391 -393.36
- Wildfire  1   0.31405 13.460 -392.53
- LandS     1   0.72367 13.870 -387.64
- Tp_diff   1   0.73704 13.883 -387.48
- Pot_Eva   1   2.30538 15.451 -370.04
```

Figure 122: Result model of Case 4.7

Case 4.8: Figure 123 presents the programming of case 4.8.

```
# -----------------------------------------------
# case 4.8: abs
# -----------------------------------------------
abs <- abs(ALL[,c(2:29)])

model.lmabs <- lm(GPI_Score ~., data = data.frame(abs[,c(1:28)]))
summary(model.lmabs)
par(mfrow=c(2,2),oma = c(1,1,2,1),mar = c(4, 4, 2, 1))
plot(model.lmabs,pch=21,bg=1,col=1,cex=1.0)
model.lmabs_2 <- step(model.lmabs)
```

Figure 123: Programming of Case 4.8

Figure 124 shows the results of case 4.8 that R^2 is 0.3225, and Cld_cv, wildfire and COV_CASE are effective at P-value < 0.05.

```
> summary(model.lmabs)

Call:
lm(formula = GPI_Score ~ ., data = data.frame(abs[, c(1:28)]))

Residuals:
     Min       1Q   Median       3Q      Max 
-0.68477 -0.23602 -0.05377  0.17903  1.41650 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)   
(Intercept)     1.936e+00  6.466e-01   2.994  0.00327 **
Cld_cv         -9.206e-03  4.452e-03  -2.068  0.04056 * 
Temp_day        3.582e-02  2.807e-02   1.276  0.20402   
Gnd_Fr          7.625e-04  1.920e-03   0.397  0.69186   
Pot_Eva         2.942e-02  7.168e-02   0.410  0.68218   
Prcp           -7.866e-05  9.770e-05  -0.805  0.42215   
Mean_Tmp        1.016e-02  2.332e-02   0.436  0.66363   
Vap_prs         1.110e-02  1.553e-02   0.715  0.47602   
Rn_day          6.494e-05  1.447e-03   0.045  0.96426   
Tp_diff         1.213e-01  1.074e-01   1.129  0.26089   
Tp_IncR        -1.458e-01  2.245e-01  -0.650  0.51696   
CO2_1970        1.044e-05  3.237e-04   0.032  0.97433   
CO2_2017       -1.801e-04  1.839e-04  -0.979  0.32919   
Inc_Rate       -1.061e-02  5.525e-03  -1.921  0.05684 . 
Drought         6.102e-05  6.304e-05   0.968  0.33480   
EarthQ         -6.399e-06  2.139e-05  -0.299  0.76524   
Epidemic        2.704e-05  4.171e-05   0.648  0.51787   
Ex_temp        -1.185e-07  1.807e-06  -0.066  0.94782   
Flood          -1.053e-04  9.537e-05  -1.104  0.27166   
LandS           5.064e-04  4.572e-04   1.108  0.26999   
Mass_move       1.107e-02  2.094e-02   0.529  0.59801   
Storm           2.621e-05  2.292e-05   1.144  0.25481   
Volcanic        3.333e-04  2.836e-04   1.176  0.24182   
Wildfire       -3.051e-02  1.235e-02  -2.470  0.01476 * 
Total_Disaster  7.995e-04  4.093e-04   1.953  0.05288 . 
COV_CASE       -4.349e-05  1.656e-05  -2.626  0.00964 **
COV_DEATH       1.206e-04  5.173e-04   0.233  0.81594   
COV_D_Rate     -2.037e+00  1.697e+00  -1.201  0.23201   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4211 on 135 degrees of freedom
Multiple R-squared:  0.4347,	Adjusted R-squared:  0.3217 
F-statistic: 3.845 on 27 and 135 DF,  p-value: 1.033e-07
```

Figure 124: Result of Case 4.8

In Figure125 presents similar plots as Figure 105. Residuals vs Fitted plot shows positive residuals are more than negative residuals, no characteristic pattern, relatively shapeless, plot 134 might be an outlier, overall, this can be a good pattern. Normal Q-Q, up to 1 on the x-axis shows a good straight line but after that plot start leaving the line. The plot 134 might be a potential issue. Scale-Location plot tells slight horizontal line, plots are well spread around the line. Residuals vs Leverage shows that there is a dot outside of Cook's distance on the right but very close to the line. It might be considered that there is not significant influential case.

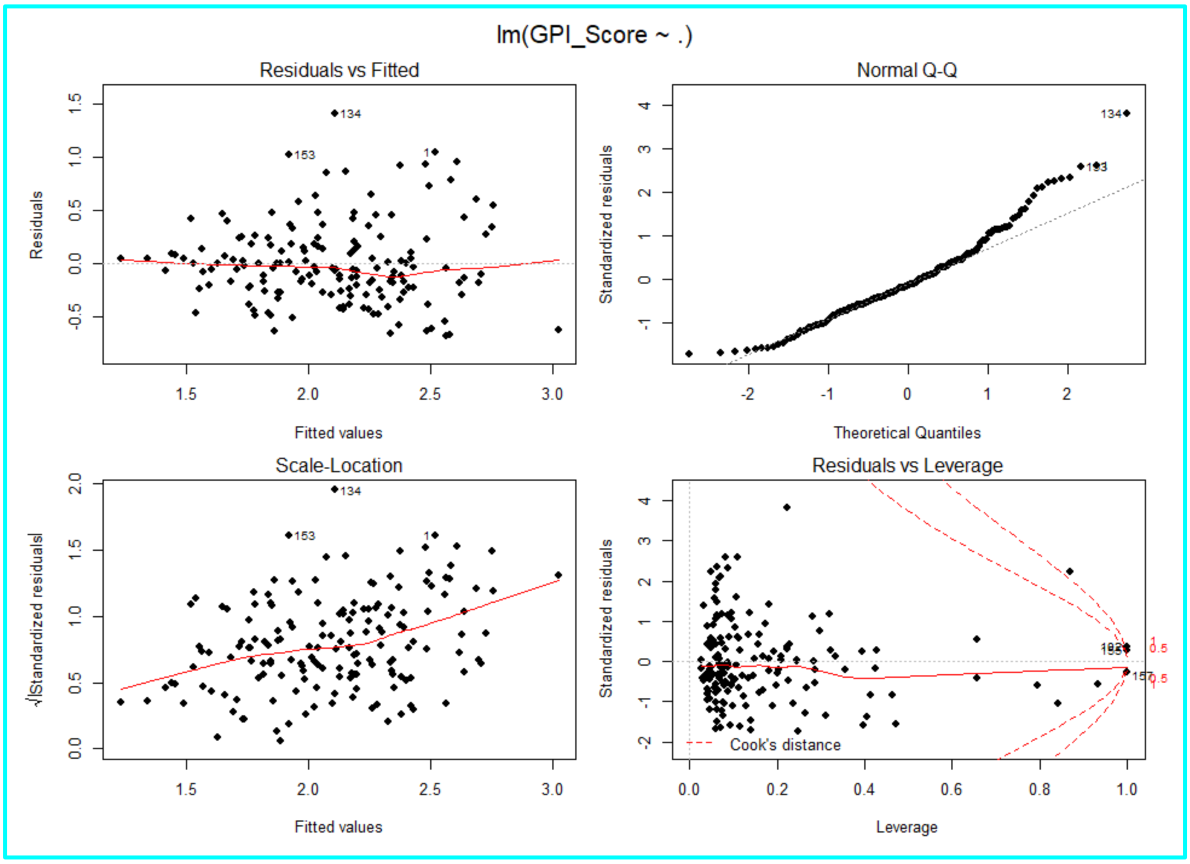


Figure 125: Plotted results of Case 4.8

Figure 126 presents the result of AIC and selected model by step function in R.

```
Step:  AIC=-285.37
GPI_Score ~ Cld_cv + Temp_day + Vap_prs + Inc_Rate + Drought +
    Volcanic + Wildfire + COV_CASE

           Df Sum of Sq    RSS     AIC
<none>                  25.344 -285.37
- Volcanic  1   0.47015 25.814 -284.38
- Inc_Rate  1   0.47593 25.820 -284.34
- Vap_prs   1   1.04686 26.391 -280.78
- Wildfire  1   1.13221 26.477 -280.25
- Temp_day  1   1.22308 26.567 -279.69
- COV_CASE  1   1.28765 26.632 -279.30
- Drought   1   1.59351 26.938 -277.44
- Cld_cv    1   2.86882 28.213 -269.89
```

Figure 126: Result model of Case 4.8

4.3.6 Data Validation

Figure 127 to 140 present programming of linear regression and result plots for 28 explanatory variables.

```
# ------------------------------------------------------- #
# 4.3.3 Additional verification: LINEAR REGRESSION  #
# ------------------------------------------------------- #
      str(Final_ALL)
      Original_OUT <- Final_ALL

      # response variable
      GPI_Score   <- Original_OUT$GPI_Score

      # explanatory variable
      Cld_cv      <- Original_OUT$Cld_cv
      Temp_day    <- Original_OUT$Temp_day
      Gnd_Fr      <- Original_OUT$Gnd_Fr
      Pot_Eva     <- Original_OUT$Pot_Eva
      Prcp        <- Original_OUT$Prcp
      Mean_Tmp    <- Original_OUT$Mean_Tmp
      Vap_prs     <- Original_OUT$Vap_prs
      Rn_day      <- Original_OUT$Rn_day
      Tp_diff     <- Original_OUT$Tp_diff
      Tp_IncR     <- Original_OUT$Tp_IncR
      CO2_1970    <- Original_OUT$CO2_1970
      CO2_2017    <- Original_OUT$CO2_2017
      Inc_Rate    <- Original_OUT$Inc_Rate
      Drought     <- Original_OUT$Drought
      EarthQ      <- Original_OUT$EarthQ
      Epidemic    <- Original_OUT$Epidemic
      Ex_temp     <- Original_OUT$Ex_temp
      Flood       <- Original_OUT$Flood
      LandS       <- Original_OUT$LandS
      Mass_move   <- Original_OUT$Mass_move
      Storm       <- Original_OUT$Storm
      Volcanic    <- Original_OUT$Vap_prs
      Wildfire    <- Original_OUT$Wildfire
      Total_Disaster <- Original_OUT$Total_Disaster
      COV_CASE    <- Original_OUT$COV_CASE
      COV_DEATH   <- Original_OUT$COV_DEATH
      COV_D_Rate  <- Original_OUT$COV_D_Rate
```

Figure 127: Programming of Linear Regression and result plot – part 1

At the top of Figure 128 presents 27 graphs are set to appear on one screen. (Original Source: Celia Rowland[6])

```
# ------------------------------------------------------
par(mfrow=c(3,9))
model_Cld_cv <- lm(GPI_Score ~ Cld_cv)
yfit_Cld_cv <- model_Cld_cv$fitted.values
plot(Cld_cv, GPI_Score, main = "Cloud cover")
lines(Cld_cv, yfit_Cld_cv, col = 'blue')
summary(model_Cld_cv)

# ------
model_Temp_day <- lm(GPI_Score ~ Temp_day)
yfit_Temp_day <- model_Temp_day$fitted.values
plot(Temp_day, GPI_Score, main = "Day time temp")
lines(Temp_day, yfit_Temp_day, col = 'blue')
summary(model_Temp_day)
```

Figure 128: Programming of Linear Regression and result plot – part 2

[6] https://kenanfellows.org/kfp-cp-sites/cp19/cp19/lesson-1-least-squares-linear-regression-r/index.html

```r
# -----
model_Gnd_Fr <- lm(GPI_Score ~ Gnd_Fr)
yfit_Gnd_Fr <- model_Gnd_Fr$fitted.values
plot(Gnd_Fr, GPI_Score, main = "Grand frost")
lines(Gnd_Fr, yfit_Gnd_Fr, col = 'blue')
summary(model_Gnd_Fr)

# -----
model_Pot_Eva <- lm(GPI_Score ~ Pot_Eva)
yfit_Pot_Eva <- model_Pot_Eva$fitted.values
plot(Pot_Eva, GPI_Score, main = "Evaporation")
lines(Pot_Eva, yfit_Pot_Eva, col = 'blue')
summary(model_Pot_Eva)
```

Figure 129: Programming of Linear Regression and result plot – part 3

```r
# -----
model_Prcp <- lm(GPI_Score ~ Prcp)
yfit_Prcp <- model_Prcp$fitted.values
plot(Prcp, GPI_Score, main = "Precipitation")
lines(Prcp, yfit_Prcp, col = 'blue')
summary(model_Prcp)

# -----
model_Mean_Tmp <- lm(GPI_Score ~ Mean_Tmp)
yfit_Mean_Tmp <- model_Mean_Tmp$fitted.values
plot(Mean_Tmp, GPI_Score, main = "Mean Temp")
lines(Mean_Tmp, yfit_Mean_Tmp, col = 'blue')
summary(model_Mean_Tmp)
```

Figure 130: Programming of Linear Regression and result plot – part 4

```r
# -----
model_Vap_prs <- lm(GPI_Score ~ Vap_prs)
yfit_Vap_prs <- model_Vap_prs$fitted.values
plot(Vap_prs, GPI_Score, main = "Vapour Pre")
lines(Vap_prs, yfit_Vap_prs, col = 'blue')
summary(model_Vap_prs)

# -----
model_Rn_day <- lm(GPI_Score ~ Rn_day)
yfit_Rn_day <- model_Rn_day$fitted.values
plot(Rn_day, GPI_Score, main = "Rainy days")
lines(Rn_day, yfit_Rn_day, col = 'blue')
summary(model_Rn_day)
```

Figure 131: Programming of Linear Regression and result plot – part 5

```r
# -----
model_Tp_diff <- lm(GPI_Score ~ Tp_diff)
yfit_Tp_diff <- model_Tp_diff$fitted.values
plot(Tp_diff, GPI_Score, main = "Temp Difference")
lines(Tp_diff, yfit_Tp_diff, col = 'blue')
summary(model_Tp_diff)

# -----
model_Tp_IncR <- lm(GPI_Score ~ Tp_IncR)
yfit_Tp_IncR <- model_Tp_IncR$fitted.values
plot(Tp_IncR, GPI_Score, main = "Temp inc rate")
lines(Tp_IncR, yfit_Tp_IncR, col = 'blue')
summary(model_Tp_IncR)
```

Figure 132: Programming of Linear Regression and result plot – part 6

```
# -----
model_CO2_1970  <- lm(GPI_Score ~ CO2_1970 )
yfit_CO2_1970 <- model_CO2_1970$fitted.values
plot(CO2_1970 , GPI_Score, main = "CO2 1970")
lines(CO2_1970 , yfit_CO2_1970, col = 'blue')
summary(model_CO2_1970)

# -----
model_CO2_2017 <- lm(GPI_Score ~ CO2_2017)
yfit_CO2_2017 <- model_CO2_2017$fitted.values
plot(CO2_2017, GPI_Score, main = "CO2 2017")
lines(CO2_2017, yfit_CO2_2017, col = 'blue')
summary(model_CO2_2017)
```

Figure 133: Programming of Linear Regression and result plot – part 7

```
# -----
model_Inc_Rate <- lm(GPI_Score ~ Inc_Rate)
yfit_Inc_Rate <- model_Inc_Rate$fitted.values
plot(Inc_Rate, GPI_Score, main = "CO2 inc rate")
lines(Inc_Rate, yfit_Inc_Rate, col = 'blue')
summary(model_Inc_Rate)

# -----
model_Drought <- lm(GPI_Score ~ Drought)
yfit_Drought <- model_Drought$fitted.values
plot(Drought, GPI_Score, main = "Drought")
lines(Drought, yfit_Drought, col = 'purple')
summary(model_Drought)
```

Figure 134: Programming of Linear Regression and result plot – part 8

```
# -----
model_EarthQ <- lm(GPI_Score ~ EarthQ)
yfit_EarthQ <- model_EarthQ$fitted.values
plot(EarthQ, GPI_Score, main = "Earthquake")
lines(EarthQ, yfit_EarthQ, col = 'purple')
summary(model_EarthQ)

# -----
model_Epidemic <- lm(GPI_Score ~ Epidemic)
yfit_Epidemic <- model_Epidemic$fitted.values
plot(Epidemic, GPI_Score, main = "Epidemic")
lines(Epidemic, yfit_Epidemic, col = 'purple')
summary(model_Epidemic)
```

Figure 135: Programming of Linear Regression and result plot – part 9

```
# -----
model_Ex_temp <- lm(GPI_Score ~ Ex_temp)
yfit_Ex_temp <- model_Ex_temp$fitted.values
plot(Ex_temp, GPI_Score, main = "Extreme Temp")
lines(Ex_temp, yfit_Ex_temp, col = 'purple')
summary(model_Ex_temp)

# -----
model_Flood <- lm(GPI_Score ~ Flood)
yfit_Flood <- model_Flood$fitted.values
plot(Flood, GPI_Score, main = "Flood")
lines(Flood, yfit_Flood, col = 'purple')
summary(model_Flood)
```

Figure 136: Programming of Linear Regression and result plot – part 10

```r
# -----
model_LandS <- lm(GPI_Score ~ LandS)
yfit_LandS <- model_LandS$fitted.values
plot(LandS, GPI_Score, main = "Land Slide")
lines(LandS, yfit_LandS, col = 'purple')
summary(model_LandS)

# -----
model_Mass_move <- lm(GPI_Score ~ Mass_move)
yfit_Mass_move <- model_Mass_move$fitted.values
plot(Mass_move, GPI_Score, main = "Mass movement")
lines(Mass_move, yfit_Mass_move, col = 'purple')
summary(model_Mass_move)
```

Figure 137: Programming of Linear Regression and result plot – part 11

```r
# -----
model_Storm <- lm(GPI_Score ~ Storm)
yfit_Storm <- model_Storm$fitted.values
plot(Storm, GPI_Score, main = "Storm")
lines(Storm, yfit_Storm, col = 'purple')
summary(model_Storm)

# -----
model_Volcanic <- lm(GPI_Score ~ Volcanic)
yfit_Volcanic <- model_Volcanic$fitted.values
plot(Volcanic, GPI_Score, main = "Volcanic")
lines(Volcanic, yfit_Volcanic, col = 'purple')
summary(model_Volcanic)
```

Figure 138: Programming of Linear Regression and result plot – part 12

```r
# -----
model_Wildfire <- lm(GPI_Score ~ Wildfire)
yfit_Wildfire <- model_Wildfire$fitted.values
plot(Wildfire, GPI_Score, main = "Wildfire")
lines(Wildfire, yfit_Wildfire, col = 'purple')
summary(model_Wildfire)

# -----
model_Total_Disaster <- lm(GPI_Score ~ Total_Disaster)
yfit_Total_Disaster <- model_Total_Disaster$fitted.values
plot(Total_Disaster, GPI_Score, main = "Total Disaster")
lines(Total_Disaster, yfit_Total_Disaster, col = 'purple')
summary(model_Total_Disaster)
```

Figure 139: Programming of Linear Regression and result plot – part 13

```r
# -----
model_COV_CASE <- lm(GPI_Score ~ COV_CASE)
yfit_COV_CASE <- model_COV_CASE$fitted.values
plot(COV_CASE, GPI_Score, main = "Cov19 Case")
lines(COV_CASE, yfit_COV_CASE, col = 'purple')
summary(model_COV_CASE)

# -----
model_COV_DEATH <- lm(GPI_Score ~ COV_DEATH)
yfit_COV_DEATH <- model_COV_DEATH$fitted.values
plot(COV_DEATH, GPI_Score, main = "Cov19 Death")
lines(COV_DEATH, yfit_COV_DEATH, col = 'purple')
summary(model_COV_DEATH)

# -----
model_COV_D_Rate <- lm(GPI_Score ~ COV_D_Rate)
yfit_COV_D_Rate <- model_COV_D_Rate$fitted.values
plot(COV_D_Rate, GPI_Score, main = "Cov19 Death Rate")
lines(COV_D_Rate, yfit_COV_D_Rate, col = 'purple')
summary(model_COV_D_Rate)
```

Figure 140: Programming of Linear Regression and result plot – part 14

5 Result Evaluations

5.1 Data Preparation for Run Models by R

Figure 141 presents how to set response variable. 1 is set when GPI score is higher than the mean, and 0 is set when it is lower or equal to the mean.

```r
# ========================================================================= #
# STAGE3: Machine Learning ================================================= #
# Independent variables to use are as follows:                              #
#   GPI_Score ~ Cld_cv + Temp_day + Vap_prs + Inc_Rate + Drought + Volcanic +   #
#              Wildfire + COV_CASE                                          #
# ========================================================================= #
# Set Risk_Rank

str(Final_ALL)
SELECTED_LR1 <- Final_ALL[,c(2,3,8,14,15,23,24,26,29 )]
str(SELECTED_LR1)
# Set if GPI Score is Higher than mean set 1 (high risk), else 0 (low risk)
SELECTED_LR1$Risk_Rank[SELECTED_LR1$GPI_Score >  mean(SELECTED_LR1$GPI_Score)] <- 1
SELECTED_LR1$Risk_Rank[SELECTED_LR1$GPI_Score <= mean(SELECTED_LR1$GPI_Score)] <- 0
SELECTED_LR1$Risk_Rank <- as.factor(SELECTED_LR1$Risk_Rank)
SELECTED_LR1$GPI_Score <- NULL
str(SELECTED_LR1)
```

Figure 141: Set response variable

Figure 142 presents the variables and data type selected.

```
> str(SELECTED_LR1)
'data.frame':	163 obs. of  9 variables:
 $ Cld_cv   : num  36.8 56.1 36.6 63 46.6 ...
 $ Temp_day : num  14.2 10.1 14.3 13.5 13.2 ...
 $ Vap_prs  : num  5.71 9.84 9.28 17.32 11 ...
 $ Inc_Rate : num  7.77 1.15 9.31 8.67 2.31 ...
 $ Drought  : num  1 0 0 1.8 0.2 0 23.5 0 0 0 ...
 $ Volcanic : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Wildfire : num  0 0 0.7 0 0.7 0 20.9 0 0 0 ...
 $ COV_CASE : num  523 433 232 3 504 ...
 $ Risk_Rank: Factor w/ 2 levels "0","1": 2 1 2 1 1 2 1 1 2 2 ...
```

Figure 142: Test model data

Figure 143 presents the risk rank split 0 and 1, and percentage of 0 = 57.7% and 1 = 42.3%.

```
> levels(SELECTED_LR1$Risk_Rank)
[1] "0" "1"
> library(base)
> percentage <- prop.table(table(SELECTED_LR1$Risk_Rank)) * 100
> cbind(freq=table(SELECTED_LR1$Risk_Rank), percentage=percentage)
  freq percentage
0   94   57.66871
1   69   42.33129
```

Figure 143: Rank Split

Figure 144 presents splitting model 70:30, and check percentage of split (Original Source: jasminecaur[7]).

```
library(caret)
library(ggplot2)
library(rlang)
library(recipes)

# Set Train 70% and Test data 30%
seed_split <- 13
set.seed(seed_split)
indexes=createDataPartition(SELECTED_LR1$Risk_Rank, p=.7, list = F)
train1 = SELECTED_LR1[indexes, ]
test1 = SELECTED_LR1[-indexes, ]
```

Figure 144: Split data Train and Test

Figure 145 presents common setting for train function.

```
# Run algorithms using 10-fold cross validation
control  <- trainControl(method="cv", number=10)
metric  <- "Accuracy"
seed_model <- 7
```

Figure 145: Train function common settings

5.2 Research of Confusion Matrix and AUC-ROC Curve

Figure 146 presents a confusion matrix which is a table that summarises the results of classification and its formula for Sensitivity(TPR), Specificity (SPC), Precision (PPV), Negative Predictive Value (NPV) and Accuracy (Medicine, 2018).

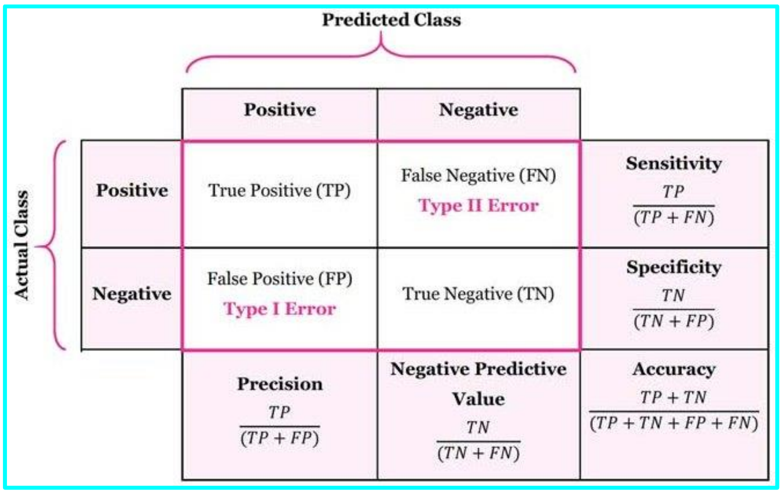


Figure 146: Confusion Matrix (SIRSAT, 2019)

[7] https://community.rstudio.com/t/createdatapartition/6780/3

5.2.1 Sensitivity

It is also called True Positive Rate (TPR) or Recall. It is the percentage that it does not be missed the problem. The higher the index, the better the model.

5.2.2 Specificity

It is also called SPC or True Negative Rate (TNR). It is an indicator that it does not suspected that there is no problem unnecessarily. The higher the index, the better the model.

5.2.3 Precision

It is also called Positive Predictive Value (PPV). It shows how many of the positive ones have problems. Or it can be said that it represents the credibility of the positive judgment.

5.2.4 Accuracy

This is the overall correct answer rate, regardless of the type of wrong answer and used to check the degree of learning in the middle of learning with machine learning. The formula divides the number of correct answers (true positive/true negative) by the total number. The higher numbers indicate better learning.

5.2.5 AUC-ROC Curve

AUC-ROC curve is well known evaluation metrics visualisation tool to measure the performance and check a classification problem. It shows the how much the model can distinguish between sensitivity and specificity. Figure 147 presents ROC curve. It shows sensitivity on y-axis and specificity on x-axis. AUC tells the area size of under the curve and it is considered as the excellent model of AUC is 0.9 to 1, 0.8 to 0.9 is very good, 0.7 to 0.8 is good, 0.6 to 0.7 is satisfactory, 0.5 to 0.6 is unsatisfactory and under 0.5 can be considered as failed (Narkhede, 2018). It shows that blue is satisfactory model, yellow is good model, and red, green, and black are very good model in this chart.

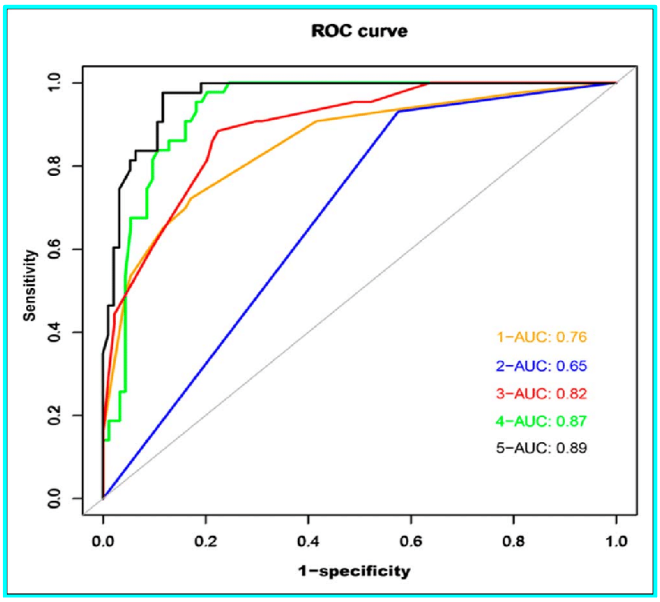


Figure 147: AUC Curve (Chen, et al., 2019)

5.3 Programming of Result Output and Visualisation

Figure 148 to 159 present programming of machine learning model (Original Source: Model Selection - Jason Brownlee[89])

Figure 148 presents the programming of generalized linear model, confusion matrix.

```
# 5.2 Generalized Linear Model (GLM)
    set.seed(seed_model)
    fit.glm <- train(Risk_Rank ~., data=train1, method="glm",
                                         metric=metric, trControl=control)
    pre_glm <- predict(fit.glm, test1)
    cf_glm <- caret::confusionMatrix(pre_glm, test1$Risk_Rank)
    cf_glm
```

Figure 148: Programming of GLM

Figure 149 presents the programming of linear discriminant analysis and confusion matrix.

```
 # 5.3  Linear Discriminant Analysis (LDA)
    set.seed(seed_model)
    fit.lda <- train(Risk_Rank ~., data=train1, method="lda",
                                         metric=metric, trControl=control)
    pre_lda <- predict(fit.lda, test1)
    cf_lda <- caret::confusionMatrix(pre_lda, test1$Risk_Rank)
    cf_lda
```

Figure 149: Programming of LDA

Figure 150 presents the programming of CART and confusion matrix.

```
# 5.4 Classification And Regression Tree (CART)
    set.seed(seed_model)
    fit.cart <- train(Risk_Rank~., data=train1, method="rpart",
                                         metric=metric, trControl=control)
    pre_cart <- predict(fit.cart, test1)
    cf_cart <- caret::confusionMatrix(pre_cart, test1$Risk_Rank)
    cf_cart
```

Figure 150: Programming of CART

Figure 151 presents the programming of kNN and confusion matrix.

```
# 5.5 k-Nearest Neighbors (KNN)
    set.seed(seed_model)
    fit.knn <- train(Risk_Rank~., data=train1, method="knn",
                                        metric=metric, trControl=control)
    pre_knn <- predict(fit.knn, test1)
    cf_knn <- caret::confusionMatrix(pre_knn, test1$Risk_Rank)
    cf_knn
```

Figure 151: Programming of kNN

Figure 152 presents the programming of SVM linear and confusion matrix.

```
# 5.6 Support Vector Machines (SVM) with Linear Kernel
    library(kernlab)
    set.seed(seed_model)
    fit.svml <- train(Risk_Rank~., data=train1, method="svmLinear",
                                         metric=metric, trControl=control)
    pre_svml <- predict(fit.svml, test1)
    cf_svml <- caret::confusionMatrix(pre_svml, test1$Risk_Rank)
    cf_svml
```

Figure 152: Programming of SVM Linear

[8] https://machinelearningmastery.com/machine-learning-in-r-step-by-step/
[9] https://machinelearningmastery.com/evaluate-machine-learning-algorithms-with-r/

Figure 153 presents the programming of SVM RBF and confusion matrix.

```
# 5.7 Support Vector Machines (SVM) with Radial Basis Function (RBF) Kernel
    set.seed(seed_model)
    fit.svm <- train(Risk_Rank~., data=train1, method="svmRadial",
                                         metric=metric, trControl=control)
    pre_svm <- predict(fit.svm, test1)
    cf_svm <- caret::confusionMatrix(pre_svm, test1$Risk_Rank)
    cf_svm
```

Figure 153: Programming of SVM RBF

Figure 154 presents the programming of random forest and confusion matrix.

```
# 5.8 Random Forest
    library(randomForest)
    set.seed(seed_model)
    fit.rf <- train(Risk_Rank~., data=train1, method="rf",
                                         metric=metric, trControl=control)
    pre_rf <- predict(fit.rf, test1)
    cf_rf <- caret::confusionMatrix(pre_rf, test1$Risk_Rank)
    cf_rf
```

Figure 154: Programming of Random Forest

Figure 155 presents the programming of Neural Network and confusion matrix.

```
# 5.9 Neural Network
    set.seed(seed_model)
    fit.nnet <- train(Risk_Rank~., data=train1, method="nnet",
                                         metric=metric, trControl=control)
    pre_nnet <- predict(fit.nnet, test1)
    cf_nnet <- caret::confusionMatrix(pre_nnet, test1$Risk_Rank)
    cf_nnet
```

Figure 155: Programming of Neural Network

Figure 156 presents the programming of Boosting and confusion matrix.

```
# 5.10  Gradient Boosting with Component-wise Linear Models
    library(mboost)
    set.seed(seed_model)
    fit.boost <- train(Risk_Rank~., data=train1, method="glmboost",
                                         metric=metric, trControl=control)
    pre_boost <- predict(fit.boost, test1)
    cf_boost <- caret::confusionMatrix(pre_boost, test1$Risk_Rank)
    cf_boost
```

Figure 156: Programming of Boosting

Figure 157 presents the programming of Bagging and confusion matrix.

```
# 5.11  Bagging (Bootstrap aggregation) CART
    set.seed(seed_model)
    fit.bagging <- train(Risk_Rank~., data=train1, method="treebag",
                                         metric=metric, trControl=control)
    pre_bagging <- predict(fit.bagging, test1)
    cf_bagging <- caret::confusionMatrix(pre_bagging, test1$Risk_Rank)
    cf_bagging
```

Figure 157: Programming of Bagging

Figure 158 presents the programming of Naïve Bayes and confusion matrix.

```
# 5.12  Naive Bayes
    library(klaR)
    set.seed(seed_model)
    fit.nb <- train(Risk_Rank~., data=train1, method="nb",
                                         metric=metric, trControl=control)
    pre_nb <- predict(fit.nb, test1)
    cf_nb <- caret::confusionMatrix(pre_nb, test1$Risk_Rank)
    cf_nb
```

Figure 158: Programming of Naïve Bayes

Figure 159 presents the programming of ctree and confusion matrix.

```
# 5.13  Conditional Inference Tree
    set.seed(seed_model)
    library(party)
    fit.ctree <- train(Risk_Rank~., data=train1, method="ctree",
                                        metric=metric, trControl=control)
    pre_ctree <- predict(fit.ctree, test1)
    cf_ctree <- caret::confusionMatrix(pre_ctree, test1$Risk_Rank)
    cf_ctree
```

Figure 159: Programming of ctree

5.4 Programming of Result Summary

Figure 160 to 165 present collecting data from each model to create comparison bar chart and line chart of accuracy, AUC, sensitivity, specificity, and precision.

```
# SET DATA
overall_glm <- data.frame(cf_glm$overall)
Accuracy_glm <- overall_glm[1,]
class_glm <- data.frame(cf_glm$byClass)
AUC_glm <- class_glm[11,]
TP_glm <- class_glm[1,]
TN_glm <- class_glm[2,]
Prec_glm <- class_glm[5,]

overall_lda <- data.frame(cf_lda$overall)
Accuracy_lda <- overall_lda[1,]
class_lda <- data.frame(cf_lda$byClass)
AUC_lda <- class_lda[11,]
TP_lda <- class_lda[1,]
TN_lda <- class_lda[2,]
Prec_lda <- class_lda[5,]
```

Figure 160: Collection of accuracy, AUC, sensitivity, specificity, and precision – part 1

```
overall_cart <- data.frame(cf_cart$overall)
Accuracy_cart <- overall_cart[1,]
class_cart <- data.frame(cf_cart$byClass)
AUC_cart <- class_cart[11,]
TP_cart <- class_cart[1,]
TN_cart <- class_cart[2,]
Prec_cart <- class_cart[5,]

overall_knn <- data.frame(cf_knn$overall)
Accuracy_knn <- overall_knn[1,]
class_knn <- data.frame(cf_knn$byClass)
AUC_knn <- class_knn[11,]
TP_knn <- class_knn[1,]
TN_knn <- class_knn[2,]
Prec_knn <- class_knn[5,]
```

Figure 161: Collection of accuracy, AUC, sensitivity, specificity, and precision – part 2

```
overall_svml <- data.frame(cf_svml$overall)
Accuracy_svml <- overall_svml[1,]
class_svml <- data.frame(cf_svml$byClass)
AUC_svml <- class_svml[11,]
TP_svml <- class_svml[1,]
TN_svml <- class_svml[2,]
Prec_svml <- class_svml[5,]

overall_svm <- data.frame(cf_svm$overall)
Accuracy_svm <- overall_svm[1,]
class_svm <- data.frame(cf_svm$byClass)
AUC_svm <- class_svm[11,]
TP_svm <- class_svm[1,]
TN_svm <- class_svm[2,]
Prec_svm <- class_svm[5,]
```

Figure 162: Collection of accuracy, AUC, sensitivity, specificity, and precision – part 3

```
overall_rf <- data.frame(cf_rf$overall)
Accuracy_rf <- overall_rf[1,]
class_rf <- data.frame(cf_rf$byClass)
AUC_rf <- class_rf[11,]
TP_rf <- class_rf[1,]
TN_rf <- class_rf[2,]
Prec_rf <- class_rf[5,]

overall_nnet <- data.frame(cf_nnet$overall)
Accuracy_nnet <- overall_nnet[1,]
class_nnet <- data.frame(cf_nnet$byClass)
AUC_nnet <- class_nnet[11,]
TP_nnet <- class_nnet[1,]
TN_nnet <- class_nnet[2,]
Prec_nnet <- class_nnet[5,]
```

Figure 163: Collection of accuracy, AUC, sensitivity, specificity, and precision – part 4

```
overall_boost <- data.frame(cf_boost$overall)
Accuracy_boost <- overall_boost[1,]
class_boost <- data.frame(cf_boost$byClass)
AUC_boost <- class_boost[11,]
TP_boost <- class_boost[1,]
TN_boost <- class_boost[2,]
Prec_boost <- class_boost[5,]

overall_bagging <- data.frame(cf_bagging$overall)
Accuracy_bagging <- overall_bagging[1,]
class_bagging <- data.frame(cf_bagging$byClass)
AUC_bagging <- class_bagging[11,]
TP_bagging <- class_bagging[1,]
TN_bagging <- class_bagging[2,]
Prec_bagging <- class_bagging[5,]
```

Figure 164: Collection of accuracy, AUC, sensitivity, specificity, and precision – part 5

```
overall_nb <- data.frame(cf_nb$overall)
Accuracy_nb <- overall_nb[1,]
class_nb <- data.frame(cf_nb$byClass)
AUC_nb <- class_nb[11,]
TP_nb <- class_nb[1,]
TN_nb <- class_nb[2,]
Prec_nb <- class_nb[5,]

overall_ctree <- data.frame(cf_ctree$overall)
Accuracy_ctree <- overall_ctree[1,]
class_ctree <- data.frame(cf_ctree$byClass)
AUC_ctree <- class_ctree[11,]
TP_ctree <- class_ctree[1,]
TN_ctree <- class_ctree[2,]
Prec_ctree <- class_ctree[5,]
```

Figure 165: Collection of accuracy, AUC, sensitivity, specificity, and precision – part 6

Figure 166 presents creating comparison bar chart of accuracy, AUC, sensitivity, specificity, and precision.

```
# prep for chart
      model <- c(1:12)
      model_name       <- c("glm", "lda", "cart", "knn", "svml", "svm", "rf",
                            "nnet", "boost", "bagging", "nb", "ctree")

      Accuracy_results <- c(Accuracy_glm, Accuracy_lda, Accuracy_cart, Accuracy_knn,
                            Accuracy_svml, Accuracy_svm, Accuracy_rf, Accuracy_nnet,
                            Accuracy_boost, Accuracy_bagging, Accuracy_nb, Accuracy_ctree)

      AUC_results      <- c(AUC_glm, AUC_lda, AUC_cart, AUC_knn, AUC_svml, AUC_svm,
                            AUC_rf, AUC_nnet, AUC_boost, AUC_bagging, AUC_nb, AUC_ctree)

      TN_results       <- c(TN_glm, TN_lda, TN_cart, TN_knn, TN_svml, TN_svm,
                            TN_rf, TN_nnet, TN_boost, TN_bagging, TN_nb, TN_ctree)

      TP_results       <- c(TP_glm, TP_lda, TP_cart, TP_knn, TP_svml, TP_svm,
                            TP_rf, TP_nnet, TP_boost, TP_bagging, TP_nb, TP_ctree)

      Prec_results     <- c(Prec_glm, Prec_lda, Prec_cart, Prec_knn, Prec_svml, Prec_svm,
                            Prec_rf, Prec_nnet, Prec_boost, Prec_bagging, Prec_nb, Prec_ctree)

      par(mfrow=c(1,5))
      barplot(Accuracy_results,names.arg=model, main = "Accuray", col=rainbow(12, alpha=0.5))
      barplot(AUC_results,names.arg=model, main          = "AUC", col=rainbow(12, alpha=0.5))
      barplot(TN_results,names.arg=model, main    = "Sensitivity", col=rainbow(12, alpha=0.5))
      barplot(TP_results,names.arg=model, main    = "Specificity", col=rainbow(12, alpha=0.5))
      barplot(Prec_results,names.arg=model, main    = "Precision", col=rainbow(12, alpha=0.5))
```

Figure 166: Creating comparison bar chart

Figure 167 presents creating comparison line chart by ggplot.

```
# for line chart by ggplot -----------------------------------------------------------------
      library(ggplot2)
      df1 <- data.frame(cbind(model, model_name, Accuracy_results))
      df1$type <- "Accuracy"
      names(df1)[names(df1)=="Accuracy_results"] <- "Results"

      df2 <- data.frame(cbind(model, model_name, AUC_results))
      df2$type <- "AUC"
      names(df2)[names(df2)=="AUC_results"] <- "Results"

      df3 <- data.frame(cbind(model, model_name, TP_results))
      df3$type <- "Sensitivity"
      names(df3)[names(df3)=="TP_results"] <- "Results"

      df4 <- data.frame(cbind(model, model_name, TN_results))
      df4$type <- "Specificity"
      names(df4)[names(df4)=="TN_results"] <- "Results"

      df5 <- data.frame(cbind(model, model_name, Prec_results))
      df5$type <- "Precision"
      names(df5)[names(df5)=="Prec_results"] <- "Results"

      df <- rbind(df1, df2, df3, df4, df5)
      str(df) # check data type
      df$model    <- as.numeric(as.character(df$model))
      df$type     <- as.factor(df$type)
      df$Results <- as.numeric(as.character.numeric_version(df$Results))
      str(df) # check data type

      ggplot(df, aes(x = model, y = Results, colour = type, group = type)) +
        geom_line( size = 1.5 ) +
        ggtitle("Comparison of Machine Learning Results in line chart") +
        scale_y_continuous(breaks=seq(0,1.2,0.2)) +
        scale_x_continuous(breaks=seq(0,12,1)) +
        geom_point(aes(shape=type), size = 3)
```

Figure 167: Creating comparison line chart

Figure 168 presents the programming of result collection to CSV format

```
# Result export ------------------------------------------------------
      Result_table <- cbind(df1[,c(1:3)], df2[,3], df3[,3],
                            df4[,3], df5[,3])
      # set header
      seed_no <- paste("Seed", seed_split, sep="")
      header <- c(seed_no, "model_name", "Accuracy", "AUC",
                  "Sensitivity", "Specificity", "Precision")
      colnames(Result_table) = header
      Result_table

      Result_export <- paste("Resuslt","_",seed_no,".","csv", sep="")
      Result_export

      # file export
      setwd("C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/SS for report/Result_table")
      write.csv(Result_table, Result_export)

      setwd("C:/Wakako/NCI/2020_Data_Analytics_MSc/Program/")
# ---------------------------------------------------------------------
```

Figure 168: Result export to csv file

Figure 169 to 174 presents programming of AUC-ROC Curve confidence interval zone (original source: ROC Curve - Joseph Rickert[10])

```
# AUC-ROC Curve -----------------------------------------------------------------
      library(pROC)
      # 5.2 Generalized Linear Model (GLM)
      par(mfrow=c(2,3))
      pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_glm),
                      smoothed = TRUE,
                      ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                      plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                      print.auc=TRUE, show.thres=TRUE,
                      main = "Generalized Linear Model", col = 'red')
      sens.ci <- ci.se(pROC_obj)
      plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
      plot(sens.ci, type="bars")

      # 5.3 Linear Discriminant Analysis (LDA)
      pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_lda),
                      smoothed = TRUE,
                      ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                      plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                      print.auc=TRUE, show.thres=TRUE,
                      main = "Linear Discriminant Analysis", col = 'red')
      sens.ci <- ci.se(pROC_obj)
      plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
      plot(sens.ci, type="bars")
```

Figure 169: Programming of AUC-ROC Curve Part1

```
      # 5.4 Classification And Regression Tree (CART)
      pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_cart),
                      smoothed = TRUE,
                      ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                      plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                      print.auc=TRUE, show.thres=TRUE,
                      main = "CART", col = 'red')
      sens.ci <- ci.se(pROC_obj)
      plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
      plot(sens.ci, type="bars")

      # 5.5 k-Nearest Neighbors (KNN)
      pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_knn),
                      smoothed = TRUE,
                      ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                      plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                      print.auc=TRUE, show.thres=TRUE,
                      main = "kNN", col = 'red')
      sens.ci <- ci.se(pROC_obj)
      plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
      plot(sens.ci, type="bars")
```

Figure 170: Programming of AUC-ROC Curve Part2

[10] https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/

```r
# 5.6 Support Vector Machines (SVM) with Linear Kernel
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_svml),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "SVM Liner", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")

# 5.7 Support Vector Machines (SVM) with Radial Basis Function (RBF) Kernel
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_svm),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "SVM Radial", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")
mtext("Comparison of AUC-ROC curve Part1", outer=TRUE,  cex=1.2, line=0)
```

Figure 171: Programming of AUC-ROC Curve Part3

```r
# 5.8 Random Forest
par(mfrow=c(2,3))
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_rf),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "Random Forests", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")

# 5.9 Neural Network
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_nnet),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "Neural Network", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")
```

Figure 172: Programming of AUC-ROC Curve Part4

```r
# 5.10  Gradient Boosting with Component-wise Linear Models
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_boost),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "Boosting", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")

# 5.11  Bagging (Bootstrap aggregation) CART
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_bagging),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "Bagging", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")
```

Figure 173: Programming of AUC-ROC Curve Part5

```
# 5.12  Naive Bayes
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_nb),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "Naive Bayes", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")

# 5.13  Conditional Inference Tree
pROC_obj <- roc(as.numeric(test1$Risk_Rank), as.numeric(pre_ctree),
                smoothed = TRUE,
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE,
                main = "Ctree", col = 'red')
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col=rgb(0,1,1,0.1))
plot(sens.ci, type="bars")
mtext("Comparison of AUC-ROC curve Part2", outer=TRUE,  cex=1.2, line=0)
```

Figure 174: Programming of AUC-ROC Curve Part6

5.4.1 Comparison by AUC-ROC Curve graph

Figure 175 and 176 present AUC-ROC Curve with confident interval zone of experiment 1 to 12 (programming code are presented in Fig 169 to 174). Each graph presents specificity as x-axis and sensitivity as y-axis, grey zone as AUC (area under curve), blue areas as 95% confidence interval zone and the value of AUC and confident interval are written in red. As GLM, LDA, CART, SVML, SVMR, boosting, Naïve Bayes and Ctree are over 70% accuracy it shows the shapes close to a square. It can be seen kNN and NNet are very unsatisfactory shapes.

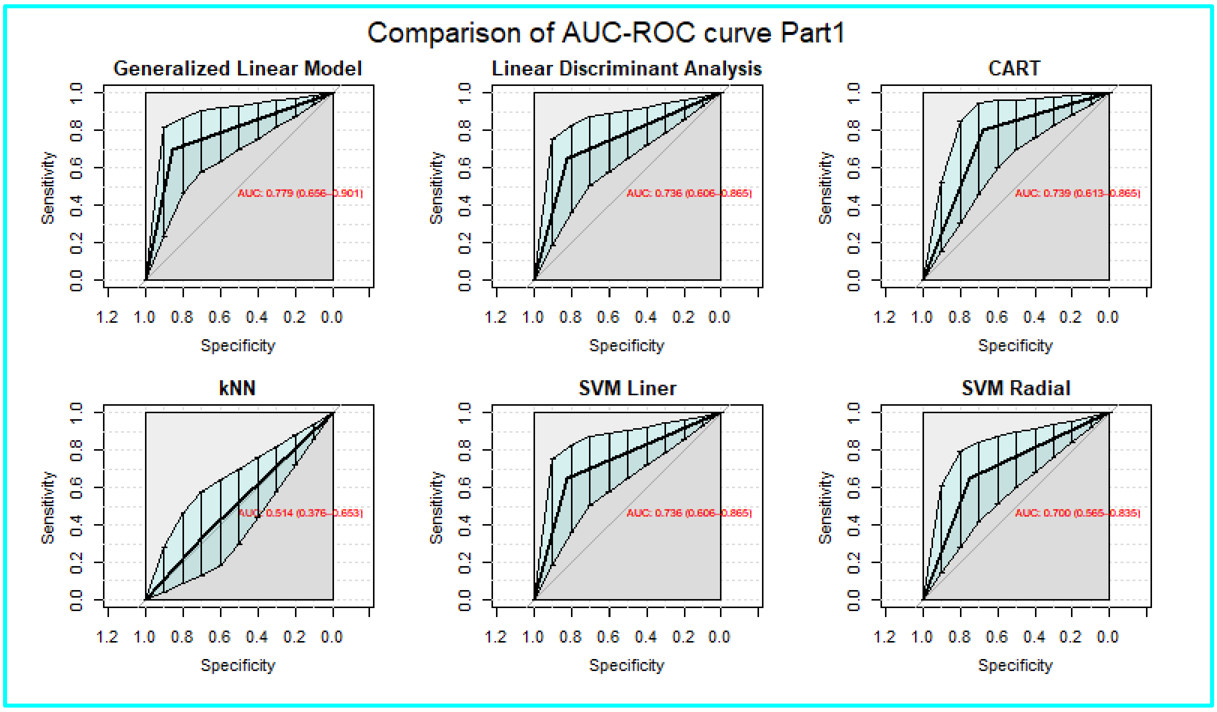


Figure 175: AUC-ROC Curve Part1

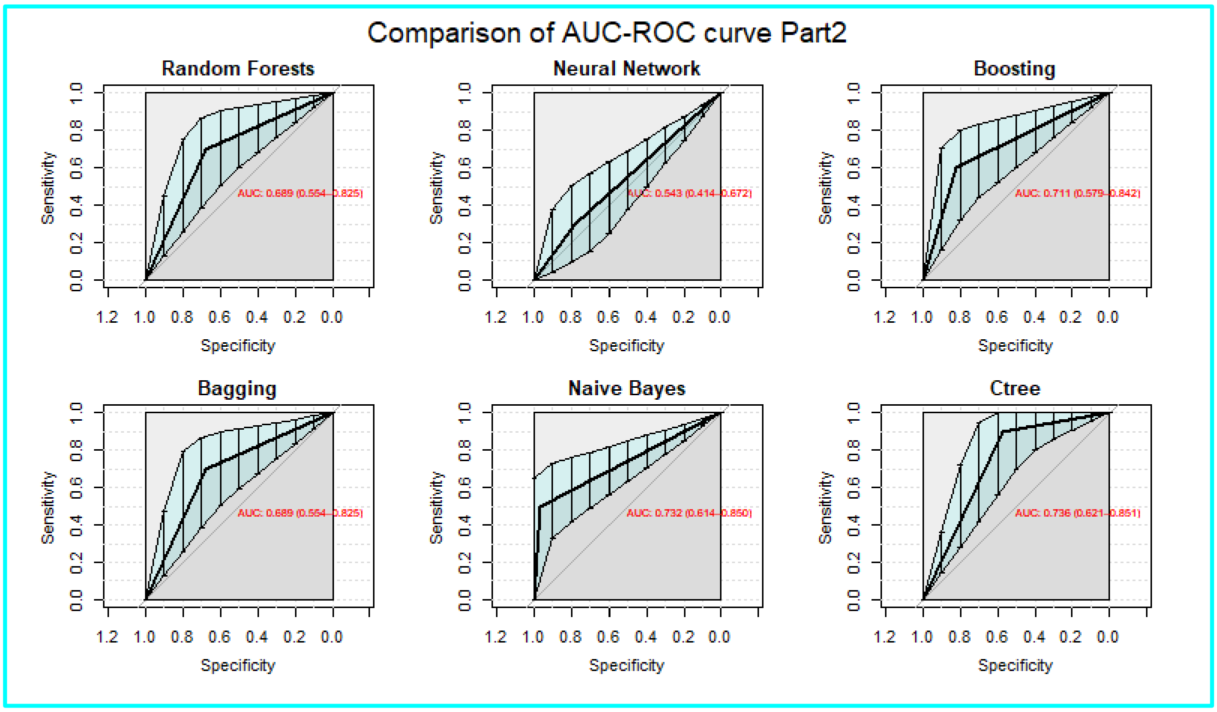


Figure 176: AUC-ROC Curve Part2

6 Tableau Presentation

This section presents the creation of Tableau dashboard and story board. Two csv files were provided for creating three bar charts and Sankey chart.

6.1 Bar charts

The detail of data collection is presented in the technical report 5.14.5. Those data were imported and created following three bar charts in three different work sheets in Tableau. Figure 177 presents proportion of high/low/misclassification in each machine learning model.

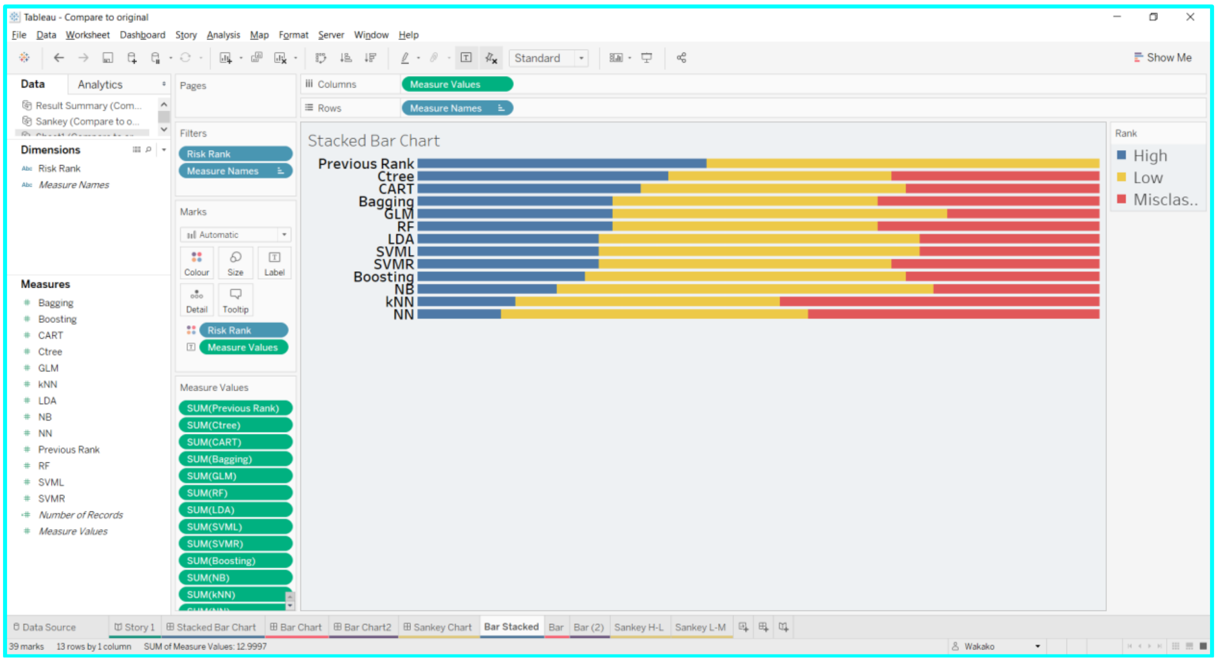


Figure 177: Stacked bar chart in Tableau

Figure 178 presents the majority of high/low/misclassification by machine learning model.

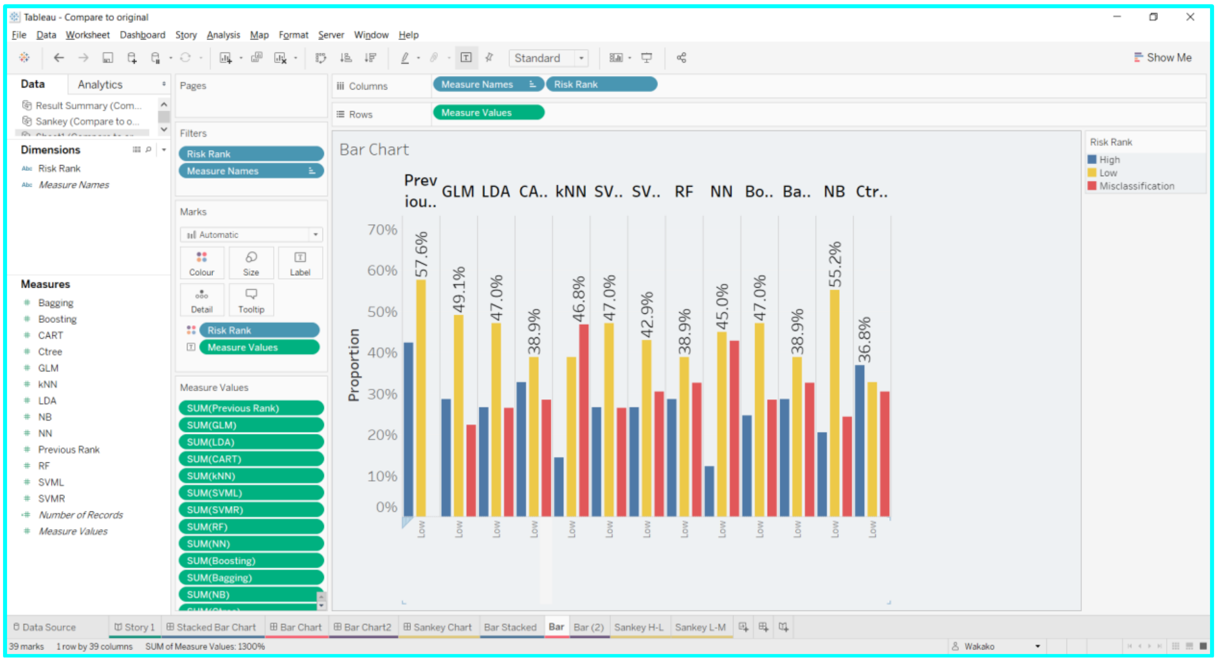


Figure 178: Bar chart in Tableau

Figure 179 presents comparison in each rank (high/low/misclassification) by 12 machine learning models.

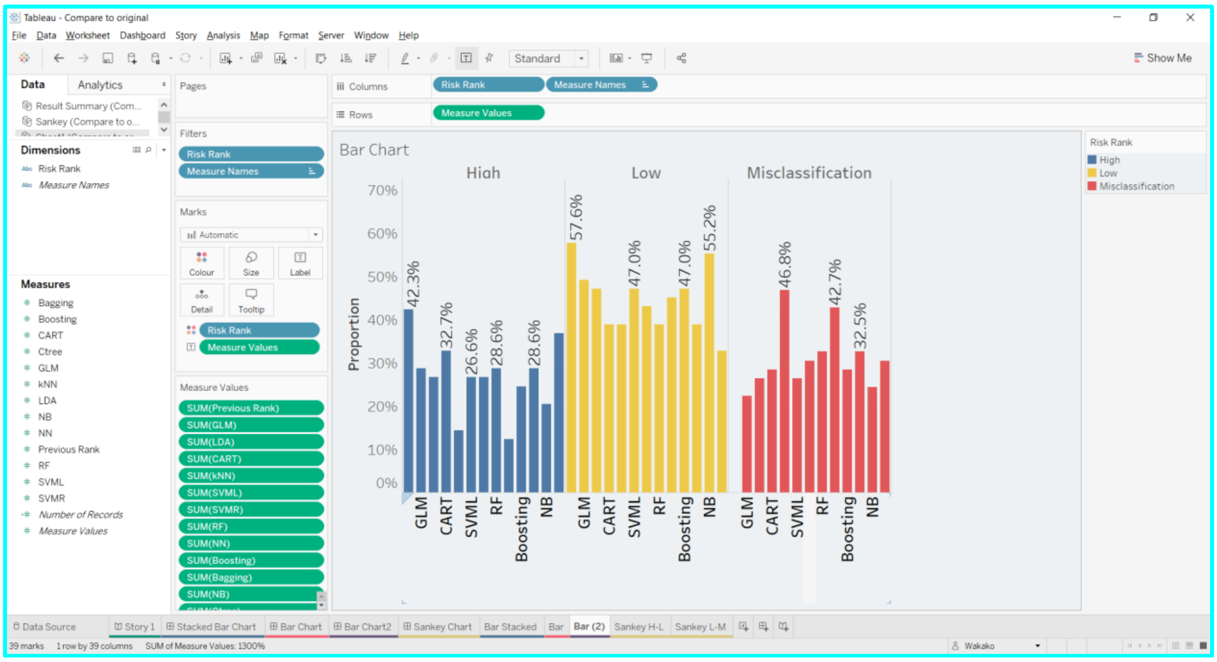


Figure 179: Bar chart in Tableau

Figure 177 to 179 were set to individual dashboard.

6.2 Sankey Chart

From the data sheet created from 6.1, ranking of 1 to 13 are applied to high, low and misclassification. Table 3 presents the details of the rankings.

Table 3: Ranking of high/low/misclassification

Risk Rank	High	Low	Misclassification
Previous Rank	1	1	13
Ctree	2	13	6
CART	3	9	7
GLM	4	3	12
RF	5	11	3
Bagging	6	12	4
LDA	7	4	9
SVML	8	5	10
SVMR	9	8	5
Boosting	10	6	8
NB	11	2	11
kNN	12	10	1
NN	13	7	2

Because Sankey chart uses sigmoid function, 49 points (-6 to 6 in 0.25 intervals) were provided for each observation on the data of Table3 to show the smooth curve in the chart. Figure 180 presents the part of modified data for Sankey chart. This data has 637[11] rows. The details of Sankey chart creation were referred the website guidance[12].

Risk Rank	High	Low	MisClass	T
Previous Rank	1	1	13	-6
Previous Rank	1	1	13	-5.75
Previous Rank	1	1	13	-5.5
Previous Rank	1	1	13	-5.25
Previous Rank	1	1	13	-5
Previous Rank	1	1	13	-4.75
Previous Rank	1	1	13	-4.5
Previous Rank	1	1	13	-4.25
Previous Rank	1	1	13	-4
Previous Rank	1	1	13	-3.75
Previous Rank	1	1	13	-3.5
Previous Rank	1	1	13	-3.25
Previous Rank	1	1	13	-3
Previous Rank	1	1	13	-2.75
Previous Rank	1	1	13	-2.5
Previous Rank	1	1	13	-2.25
Previous Rank	1	1	13	-2
Previous Rank	1	1	13	-1.75
Previous Rank	1	1	13	-1.5
Previous Rank	1	1	13	-1.25
Previous Rank	1	1	13	-1
Previous Rank	1	1	13	-0.75
Previous Rank	1	1	13	-0.5
Previous Rank	1	1	13	-0.25
Previous Rank	1	1	13	0
Previous Rank	1	1	13	0.25
Previous Rank	1	1	13	0.5
Previous Rank	1	1	13	0.75

Figure 180: Part of data for Sankey chart

Figure 181 presents the ranking relationship between high and low, and Figure 182 presents the ranking relationship between low and misclassification. Figure 183 presents a dashboard which merged Figure 181 and 182.

[11] 637 = 13 * 49
[12] https://www.dataplusscience.com/RecreationinTableau2.html

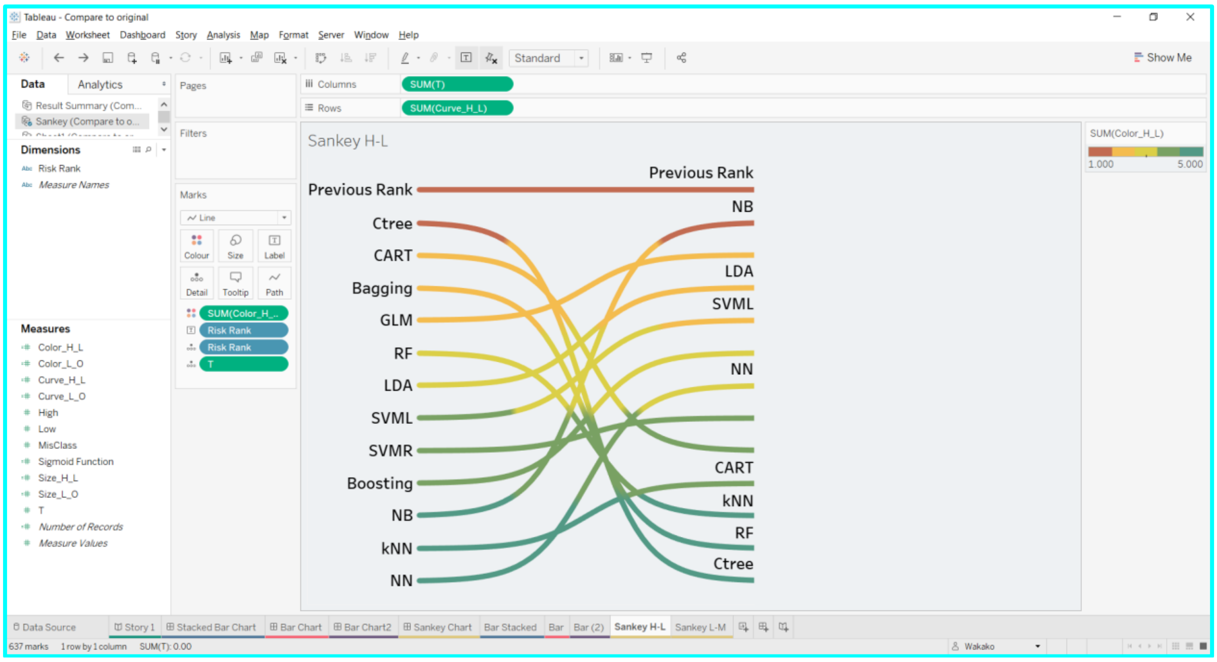


Figure 181: Sankey chart part 1

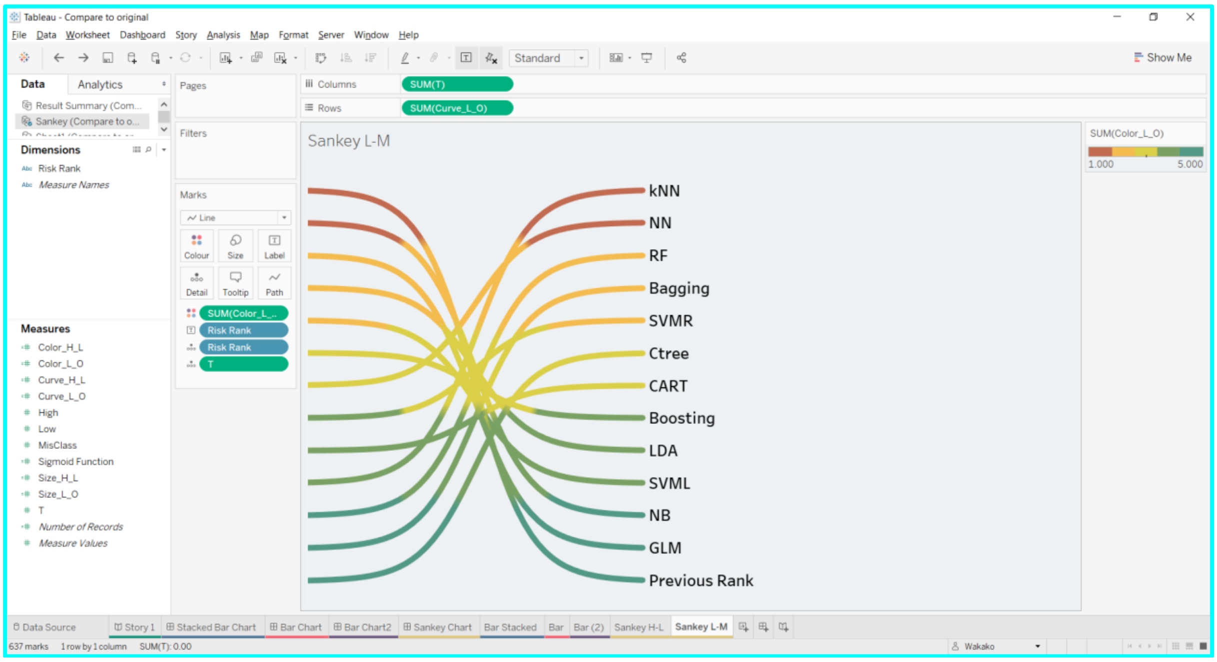


Figure 182: Sankey chart part 2

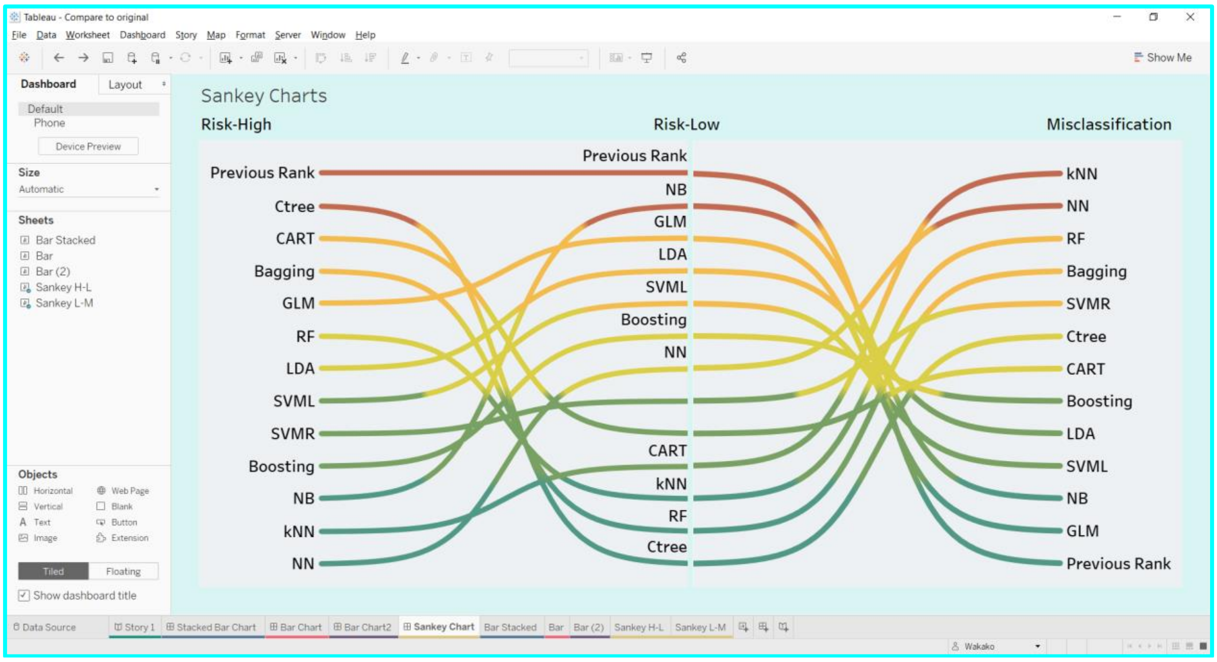


Figure 183: Sankey chart on the dashboard

All four dashboard which are three bar charts and a Sankey chart, are set a story board in Tableau and uploaded to Tableau public sever[13], so that all presentation can been seen in one place. Figure 184 presents the image from the Tableau public sever.

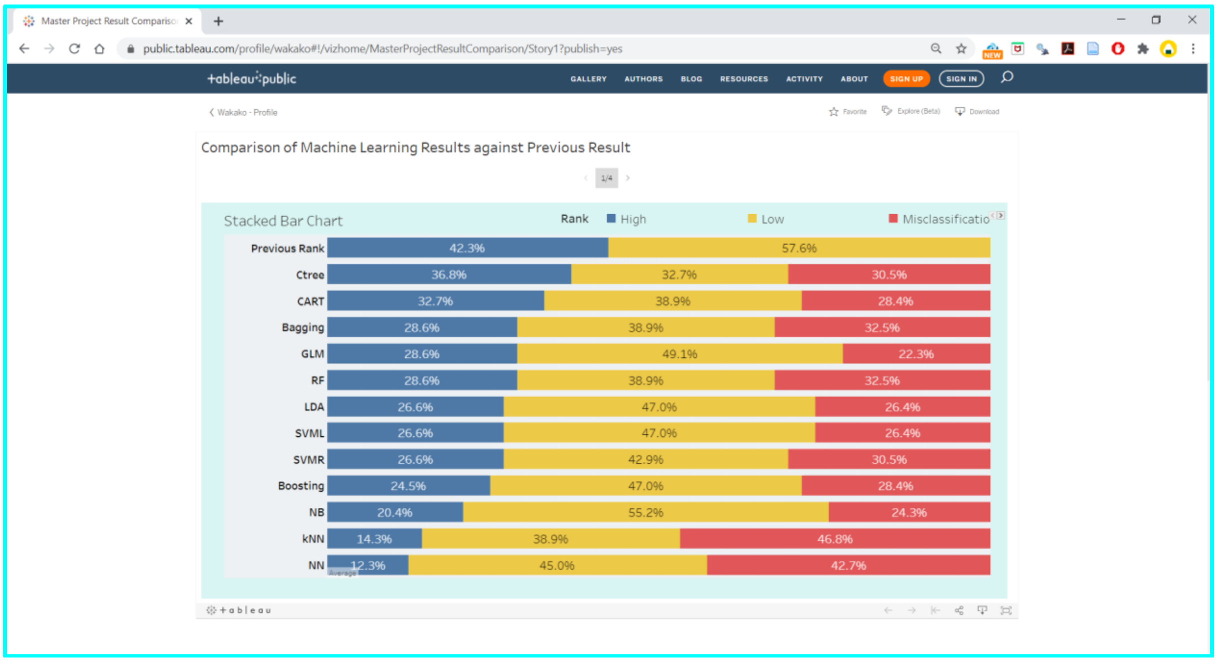


Figure 184: Image from Tableau public server

[13] https://public.tableau.com/profile/wakako#!/vizhome/MasterProjectResultComparison/Story1?publish=yes

7 References

ALEX, 2019. *The Scale Location Plot: Interpretation in R.* [Online]
Available at: https://boostedml.com/2019/03/linear-regression-plots-scale-location-plot.html
[Accessed 11 July 2020].

Chen, M., Cao, J., Bai, Y. & Cai, i., 2019. Development and Validation of a Nomogram for Early Detection of Malignant Gallbladder Lesions. *Clinical and Translational Gastroenterology,* 10(10), p. e00098.

Deo, S., 2016. *R Tutorial : How to use Diagnostic Plots for Regression Models.* [Online]
Available at: http://analyticspro.org/2016/03/07/r-tutorial-how-to-use-diagnostic-plots-for-regression-models/#:~:text=Scale%20%E2%80%93%20Location%20Plot%20Scale%20location%20plot%20indicates,that%20residuals%20have%20uniform%20variance%20across%20the%20range.
[Accessed 11 July 2020].

Hagerman, I., 2017. *Residual Plots Part 1 — Residuals vs. Fitted Plot.* [Online]
Available at: https://medium.com/data-distilled/residual-plots-part-1-residuals-vs-fitted-plot-f069849616b1
[Accessed 11 July 2020].

Hagerman, I., 2017. *Residual Plots Part 2 — Normal QQ Plots.* [Online]
Available at: https://medium.com/data-distilled/residual-plots-part-2-normal-qq-plots-c220ee9ed9fc
[Accessed 11 July 2020].

Kim, B., 2015. *Understanding Diagnostic Plots for Linear Regression Analysis.* [Online]
Available at: https://data.library.virginia.edu/diagnostic-plots/
[Accessed 11 July 2020].

Medicine, T. A. f. C. B. &. L., 2018. *Accuracy, precision, specificity & sensitivity.* [Online]
Available at: https://labtestsonline.org.uk/articles/accuracy-precision-specificity-sensitivity
[Accessed 18 July 2020].

Narkhede, S., 2018. *Understanding AUC - ROC Curve, towards data science.* [Online]
Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
[Accessed 14 July 2020].

SIRSAT, M., 2019. *Data Science and Machine Learning.* [Online]
Available at: https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html
[Accessed 14 July 2020].