Covid19 Analysis

Thanos Gentimis

2/8/2022

Contents

1 Description and preamble

This document will explain the inner workings of the code TG_TimeSeries_1_2.R which is used to analyze the covid19 number of cases and deaths in Louisiana. Make sure you have installed all the libraries bellow. You can do that from the tab packages in R-Studio.

1.1 Libraries and functions

The following libraries are quite useful when analyzing time series and creating graphs with them. fpp2, zoo, and urca are modeling and forecasting packages, whereas GGally, gridExtra and ggrepel are useful for creating better graphs with ggplot2. Tidyverse is the go to package for data analysis, lubridate handles date manupulations and foreign is used to read data into R from various formats. Finally, psych contains various statistical functions which are especially useful.

```r
rm(list = ls())
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
library(easypackages)
libraries("fpp2", "zoo", "tidyverse", "lubridate", "GGally",
    "gridExtra", "ggrepel", "foreign", "urca", "psych")
source("./Functions/TSConvert.R")  # adds the function
```

Note that the command at the end, utilizes the function TSConvert from the folder Functions. This is a custom function that I have created which quickly converts a variable to a timeseries from a dataframe that contains a Date column.

Also, for advanced versions of R studio the line setwd(. . .) automatically sets the working directory to the same location as the code. Still, if you are unsure that this worked you can always set the working directory to the source file location manually.

1.2 Read data, keep latest, format dates

```
df = read.csv("https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv")
df1 = df %>%
    filter(state == "Louisiana") %>%
    select(date, cases, deaths)
colnames(df1) = str_to_title(colnames(df1))
df1$Date = as.Date(df1$Date)
write.csv(df1, "../Data/Processed/Cleandata1.csv")
```

The data is read directly from a repository maintained by the New York times. So far this has been a reliable source with a stable flow of data. We then use tidyverse's commands to filter our data and keep only information about Louisiana and select only date, cases and deaths. The function str_to_title changes the names of the variables so that the first letter is capitalized. A good idea is to have the variable names with capital letter or at least the first letter capitalized, to distinguish them from the entries inside the dataframe.

We mentioned before that R most of the times understand Date data, but it is a good idea to make sure we set it like that with the command as.Date. This function allows us to change format if needed. Look it up!

Also, after you clean up the data, it is a good idea to save them as a csv file in the processed folder. Note that each time a new dataset comes in the cleandata1 will be overwritten!

```
dir.create(paste0("../Results/", df1$Date[nrow(df1)]), showWarnings = F)
```

Since we will need a place to save the results, and this process will be repeated, it is a good idea to create folders for the answers that are connected with the latest date of the data. The dir.create is a command that allows R to make folders. Mac and Linux users might need to give R special priviledges but it normally works just fine. Note also the use of the function paste0() which allows us to create path names with fixed parts and variables parts.

2 Data Exploration

It is always a good idea to get a feeling of your data before you even attempt to analyze it. Simple graphs can go a long way, as well as some summary statistics.

2.1 Graphs for Cumulative Cases and Deaths

```
p1 = ggplot(df1, aes(x = Date, y = Cases)) + geom_point(size = 0.5) +
    scale_x_date(breaks = "1 month", date_labels = "%b") + geom_line(color = "#00AFBB") +
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ xla
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ +
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ yla
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ Cas
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ +
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ #ge
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ =
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ 1,
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ nud
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ =
    xlab("") + ylab("Daily Cases") + #geom_text(aes(label=Cases),size=2,hjust = 1, nudge_x = 0.05)+ 0.0
ggtitle("Timeseries of Cases") + theme_classic()
#+theme(axis.text.x = element_text(angle = 90))
```

```
ggsave(paste0("../Results/", df1$Date[nrow(df1)], "/Cases.jpg"),
    width = 10, heigh = 10, p1)
```

```
p1 = ggplot(df1, aes(x = Date, y = Deaths)) + geom_point(size = 0.5) +
    scale_x_date(breaks = "1 month", date_labels = "%b") + geom_line(color = "#00AFBB") +
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ x
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ +
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ y
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ D
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ +
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ #
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ =
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ 1
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ n
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ =
    xlab("") + ylab("Daily Deaths") + #geom_text(aes(label=Deaths),size=2,hjust = 1, nudge_x = 0.05)+ 0
ggtitle("Timeseries of Deaths") + theme_classic()
#+theme(axis.text.x = element_text(angle = 90))
ggsave(paste0("../Results/", df1$Date[nrow(df1)], "/Deaths.jpg"),
    width = 10, heigh = 10, p1)
```

Note that the dataset does not contain the daily cases and deaths but the cumulative daily cases and deaths. Of course this is the same amount of information, but if we need to analyze the data we need to extract the daily cases and create some sort of moving average.

2.2 Creating moving averages and adding them to the dataset

Extracting the first day of the dataset and use that to define our time series. Note the use of strsplit so we can obtain the year-month-day breakdown

```
StartDay = df1$Date[1]
yr_mt_da = as.numeric(strsplit(as.character(StartDay), "-")[[1]])
yr_mt_da
```

```
## [1] 2020    3    9
```

2.2.1 Creating the Daily Cases and their moving Average

```
Casests <- ts(df1$Cases, start = yr_mt_da, frequency = 365.25)
Casets1 = c(Casests[1], diff(Casests, 1))
Casests_ma = ma(Casets1, 7)
df1$DailyCases = as.vector(Casets1)
df1$Cases_ma = as.vector(Casests_ma)
```

2.2.2 Creating the daily deaths and their moving average

```
Deathsts <- ts(df1$Deaths, start = yr_mt_da, frequency = 365.25)
Deathsts1 = c(Deathsts[1], diff(Deathsts, 1))
Deathsts_ma = ma(Deathsts1, 7)
df1$DailyDeaths = as.vector(Deathsts1)
df1$Deaths_ma = as.vector(Deathsts_ma)
```