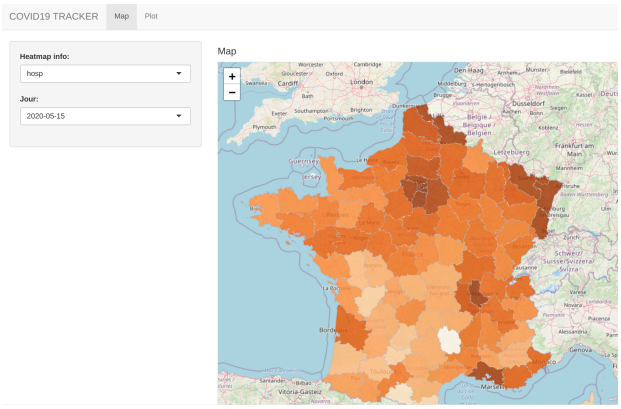
EpitechDocumentation

COVID19 Tracker

Data visualization of COVID19 in France with R



0.1

EPITECH.

COVID19 Tracker

language: R
build tool: no need here

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

R is a programming language and free software environment for statistical computing and graphics supported by the **R** Foundation for Statistical Computing. The **R language** is widely used among statisticians and data miners for developing statistical software and data analysis.

Shiny is a **R package**, which allows creation of web interactive pages on which it's possible to perform all analyzes / action available on **R**.

With the **COVID-19 pandemic**, **R** is a very effective tool for analyzing the various data linked to this situation.

For example, on the R-Shiny Gallery, you can find a web app called COVID-19 tracker made by Edward Parker to observe the epidemic on a global scale.

Today, we are going to created our own web application to visualize on a map and a graph the hospital data linked to COVID-19 in France.

Step 1 - Get datas

The very first step to start this application is to get datas.
We will need two things:

- a **csv file** with COVID-19 datas,
- a **geojson file** to put those datas on a map.

You are in luck, these two files are already given to you. The .csv file contains hospital data for each day since the beginning of March for each department. you can have more information on the format of each columns thanks to the **metadata file**. You can get a more up-to-date version of the dataset on this site if you wish.

The geojson file is made up of several elements:

- a list of **polygons** with GPS coordinates that represent each departments and their names.
- a **dataframe** with the list of department number and their names.

For this first step you will have to **open** these two files in **global.R**.

Look at *read.csv* and *geojsonio_read* function.

Before going further, you must **prepare the dataset** to be able to **merge** it with the geosjon file.
Indeed the dataset contains a **gender column** with values ranging from **0 to 2**. The values 0 are the sum of the values 1 and 2 so there is no need to keep these lines.

Filter the dataset to keep only the rows with **gender value at 0**.

Also, for now, we are going to display the values for a specific day. So filter the lines to keep **only one day** of datas. (e.q 2020-05-15)

Find out how to use *filter*.

If you want to execute one line of your script you can use CTRL+Enter on this line

Step 2 - Display the map

The dataset is ready, now you have to **merge** it with that of the geosjon. Do this in the *renderLeaflet* function of the ui.R file.

To merge two dataframes they must have a column sharing data in common.
sp::merge can perform this action, try to find out how merge functions work in R.

The geosjon now contains all the data but there is one thing to do before creating a map: making a **color palette**.

This palette will allow each department to be colored according to a value. This will allow you to better visualize datas.

Here is some way to create palettes

Now is the time to create a map and display our data there. Use *leaflet* to initialize the map and then use *addPolygons* to add a layer with our geosjon.

Department must be **filled** thanks to the previously created palette and the values to apply on the palette can be those of the **dc column**.

There is a very complete documentation about leaflet here

Feel free to use *addTiles* to display a map background if you want a better render

At this point, you can **launch** your application using the **Run App** button to see the result.

There should be a map with very contrasting colors. To find out what this is due to, add in the **popup** option of *addPolygons* a string containing the **department name** and its **value for each column** (hosp, rea, rad, and dc). This will display this string by clicking on a department on the map.

Try changing the **scale of values** used when creating the colors to decrease the variance. A **logarithmic scale** may give a better rendering.

Step 3 - Add buttons

The first page of our app now displays a map with information, but it has **no interactions** with it yet.

In the first step, you chose the data for a **specific day**, but why not leave it up to the user of the day to display on the map.

In ui.R, add a *selectInput* with **each day available of the dataset as choices** and **a default day already selected**.
Move the filtering of the day from global.R to server.R before merging the dataset with the geosjon and **filter with the selected day**.

Add another *selectInput* to leave **the choice** to the user **of the column used to color** the map by adding another *selectInput* (choices must be : hosp, read, rad or dc).

Take a look at the R-Shiny Gallery to learn how 'link' buttons of ui part with server part of a shiny app.

Step 4 - Draw a plot

From the beginning you used the dataset to make a map, but those datas can also be used to make plots.

In this part you will have to **create a plot** showing the **evolution** of the different data **over the time**.

We will still have to **prepare the dataset** to use it. You can keep the filtering of the gender column for this part. But we will have to merge the lines of the same departement **to get one line per day**.

group_by and *summarise* make a rather useful combo in this situation. Find out how to use them together.

The dataset therefore now contains one line per day. You must now create a plot with. Use *ggplot* function for this in server.R.

ggplot tutorial

To go further, you can also add a selectInput to this page. For example to be able to select the evolution of datas for a selected department.