actuartech

Introduction to Dashboarding in R with the use of COVID-19 Data

Actuartech Academy

info@actuartech.com

Introduction

With the rise of the COVID-19 pandemic, there presents an on going need to monitor data regularly in order for businesses to interpret and respond to the impact of the pandemic.

This presentation is concerned with actuarial teams; and how they may use lightweight dashboarding techniques as part of an internal reporting strategy; allowing them to understand or forecast trends; understand the impact on their business and share information required to these impacts.

Objectives

Our aim is for users to have a set of relevant pages displaying key COVID-19 statistics interactively, like the ability to change the country, date, and statistic they wish to view.

To make it more readable, we want to display these in separate tabs. One page displaying COVID-19 statistics at a glance, labeled as "Overview", another displaying an interactive table the user can search through ("Table"), and finally an interactive plotting environment where the user can compare countries across various features (e.g. daily deaths, cumulative tests conducted). We also want to give users the ability to add moving averages. This smooths out the visualised data while preserving trends.

R Shiny Dashboarding

Overview

R Shiny is a dashboarding technique found in the R programming language. RStudio contains templates and helpful guides to get started. Shiny dashboards are essentially web-based applications that can also run locally. Even though it is web-based, simple examples do not require any Javascript, HTML, or CSS knowledge.

Dashboarding Purposes

Implemented correctly, a dashboard should be intuitive to the user to operate and produce clear visuals and insight from data. The user typically will not have to add code nor require knowledge on how to code, as the code operates in the background. This offers users more flexibility than static visualisations but without requiring the users to know how to produce visualisations from scratch.

Reactive Programming

To implement a Shiny dashboard in R requires a shift from traditional programming structures, known as *reactive programming*. This essentially means code can be run and objects may change in real-time while the program is running. To contrast this to usual R programming, when we run the code to train a model and produce a summary of the output, we cannot alter that model while it is running and see results in real time. It requires that we run the piece of code again before changes can be seen. With Shiny's reactive programming, we only need to run the R script once to load the dashboard, then by interacting with the dashboard, we "re-run" part of the code again with new changes made. As you will see, we are able to switch from country to country and alter the dates and the output (for example: case numbers, deaths, and vaccines) update automatically.

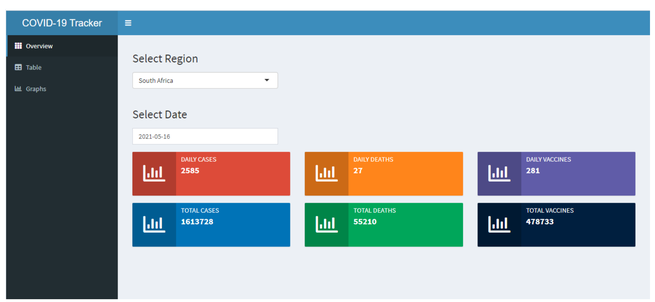

Figure 1:Structure of the Dashboard

COVID-19 Dataset Used

We utilise the most recent version of *Our World in Data*'s COVID-19 dataset, found on their GitHub repository `https://github.com/owid/covid-19-data/tree/master/public/data/`.

The dataset contains the following information relevant to our dashboard:

1. **Reported Cases and Deaths** Daily and cumulative values for each country from the first reported case worldwide to present day. This is provided from the COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University.
2. **Reported Tests:** Daily and cumulative amounts compiled by *Our World in Data*.
3. **Vaccinations:** Daily and cumulative amounts compiled by *Our World in Data* using published reports.

Before being used in the dashboard, the dataset is checked for quality and structure by checking for missing values, obvious outliers, and visualisation spot-checks.

Packages Used

In addition to the R programming language, we also require external packages that allow us to build the dashboard, namely `shiny`, `shinydashboard`, `DT`, `lubridate`, `tidyverse`, and `zoo`. Note that this is performed using R 4.0.3 for Windows 10 x64.

Package	Version	Description
DT	0.18	Display tables on HTML pages. Data can be easily filtered, sorted, and more on your dashboard
lubridates	1.7.9.2	Makes it easier to work with dates and times. Also found in `tidyverse`.
shiny	1.6.0	Allows us to build interactive dashboards in R
shinydashboard	0.7.1	Improves the user interface beyond `shiny`
tidyverse	1.3.0	A collection of packages designed to make data management and visualisation easier
zoo	1.8.8	This package will only be used for its `rollmean()` function to create moving-averages.

Table 1:Packages Used

Methodology

Loading and Pre-Processing the Data

The first step is to load the required packages and data onto the local system. Any missing values are set to zero and we ensure the dates present are `Date` objects. The data is then stored as a `reactive` object. We then extract the following objects in order to create drop-down lists:

- A list of unique countries.
- Unique, relevant column names (features)
- Set of all unique dates

The output from user selections will be captured in the `server` function and updated in the `ui` object.

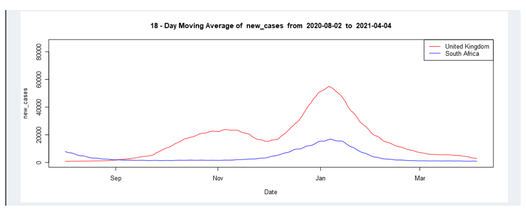

Figure 2:Customisable Moving Average Plot

Methodology Cont.

Overview Tab

We make use of info boxes to display information on daily & cumulative cases, deaths, and vaccinations. This requires us to perform a subset command in the server side based on the user input and update the `ui` with the output.

The aim is to provide key statistics colour-coded at a glance. Developers can add functionality such as choosing which statistics to display.

Data Table Tab

We then want to provide users with an interactive space to query the data for themselves, by country. The `DT` package makes this very straightforward and quick to implement. The result is an interactive table that can be queried and sorted.

This interactive element allows for complete datasets to be loaded succinctly. Further features can be added such as user-inputted SQL commands through the `sqldf` package.

Graphs Tab

This requires generalised plots that update based on user input. The user can set two countries they wish to view, their date range, the feature they want to plot, and the option to include a moving average plot.

Special care must be taken to update the headings and legends to match the user inputs. These can be saved and used when compiling internal reports. Users are able to generate multiple plots without having to produce any new code.

The moving average functionality allows users to stipulate the number of days they want to average over, and a plot is produced below the regular plot. The `rollmean()` function found in the `zoo` package is used.

Deployment

When dashboards are created, they are deployed locally and accessed through localhost. While this is useful for testing features but it is inflexible if the aim is to distribute it with others, since it would require them to install R and all dependencies on their local machine. Some solutions are:

Shinyapps.io

Shinyapps.io allows developers to quickly deploy their dashboards on pre-existing hardware that is secured by RStudio themselves. Free-tier gives developers a limited number of active hours but up to 5 dashboards on a preset domain, which scales up through various paid tiers allowing for multiple dashboards and personalised domains.

Shiny Server

This is an open-source solution through Amazon Web Servers or a similar cloud provider. Developers have to build up the environment themselves through Linux-supported binaries which may be advanced for some but offers flexibility and personalised control over their deployment and management of their server.

RStudio Connect

This is a collaborative environment for all R hosting from dashboards to R markdown reports through RStudio. It is a commercial licence and may be suitable for a team working collaboratively in R using RStudio's suite that wish to deploy their software and receive the necessary support from RStudio.

References and Further Reading

[1] Hosting and deployment.
https://shiny.rstudio.com/deploy/.

[2] Learn shiny.
https://shiny.rstudio.com/tutorial/.

[3] Open source & professional software for data science teams, May 2021.
https://www.rstudio.com/.

[4] Diana Beltekian Hannah Ritchie, Esteban Ortiz-Ospina.
Coronavirus pandemic (covid-19).
Our World in Data, 2020.
https://ourworldindata.org/coronavirus.