A *Shiny* application for visualising SARS-CoV-2 vaccination coverage in Portugal

Bruna Araújo	Matias Capitão	Rafael Antunes
A84408	A82726	A77457

Supervisor:
Cecília Castro

Abstract

For this project, was proposed to us the development of an application with the theme: "Development of a Shiny application to visualize data on the SARS-CoV-2 epidemic".

The main objective was to develop a Shiny application for visualization of data and analysis of various indicators that allow, in an interactive way, the monitoring of the SARS-CoV-2 epidemic in Portugal.

This project had two phases: the first one was a phase to study the data and workflow of an R Shiny application, and the second phase was the phase of implementation.

We end up to publish our application in *Shinyapps.io* that is an online service for hosting Shiny apps in the cloud.

Contents

List of Figures

Acronym List

IDE Integrated Development Environment

WHO World Healh Organization

EU European Union

ECDC European Centre for Disease Prevention and Control

Chapter 1

Introduction

1.1 The *R* Program and *Shiny* package

R is a program created in 1993 by Ross Ihaka and Robert Gentleman [1] defined as "a language and environment for statistical computing and graphics."

RStudio is an integrated development environment (IDE) for R. [8]

R has over 10,000 packages in the CRAN repository which are constantly growing. ***Shiny*** [2] is an *R* package that makes easy to build interactive web applications.

With this package we can combine the computational power of *R* with the versatility and interactivity of modern web. This feature it's relevant to develop plots and graphics that can be understood by everyone, and looking pretty while doing it.

Figure 1.1: Logos of R and RStudio

1.2 COVID-19 pandemic

As we can read in [9], "The COVID-19 pandemic, also known as the coronavirus pandemic, is an ongoing global pandemic of coronavirus disease 2019 (COVID-19), which is caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The virus was first identified in December 2019 in Wuhan, China. The World Health Organization declared a Public Health Emergency of International Concern on 30 January 2020, and later declared a pandemic on 11 March 2020. As of 27 June 2021, more than 180 million cases have been confirmed, with more than 3.91 million confirmed deaths attributed to COVID-19, making it one of the deadliest pandemics in history."

The main symptoms of COVID-19 are fever, cough, breathing difficulties, loss of smell and taste.

It's a disease that attacks people of all age and is transmitted mainly by air, via particles that we expel.

There are several reasons for the pandemic to spread globally, mainly the fact that a patient may be asymptomatic, which can cause him to transmit the virus without knowing it. For the same reason, the fact that the incubation period is about 14 days is also a factor to be taken into account.

1.2.1 Vaccination

Vaccines are substances made up of pathogens (viruses or bacteria), living or dead, or their derivatives. They stimulate the immune system to produce antibodies that act against pathogens that cause infections.

Vaccination is a way to protect people from harmful diseases. It uses the body's natural defenses to build resistance to specific infections and makes the stronger immune system. It reduces the possibility of contracting the disease and thus prevents it from spreading.

As soon as the COVID-19 pandemic took hold in the world, research for the

production of safe and effective vaccines began immediately. Vaccination is critical to ending the COVID-19 pandemic. WHO is working tirelessly with partners to develop, manufacture and distribute safe and effective vaccines. Until May 10, 13 vaccines have been authorized by at least one national regulatory authority for public use. However, in Europe, after consulting the list of authorized vaccines on the [4] official website of the European Union (EU), we have four vaccines available:

- BioNTech-Pfizer - (2020/12/21)
- Moderna - (2021/01/06)
- AstraZeneca - (2021/01/29)
- Johnson & Johnson - (2021/03/11)

But it's not the vaccines that will stop the pandemic, it's the vaccination.

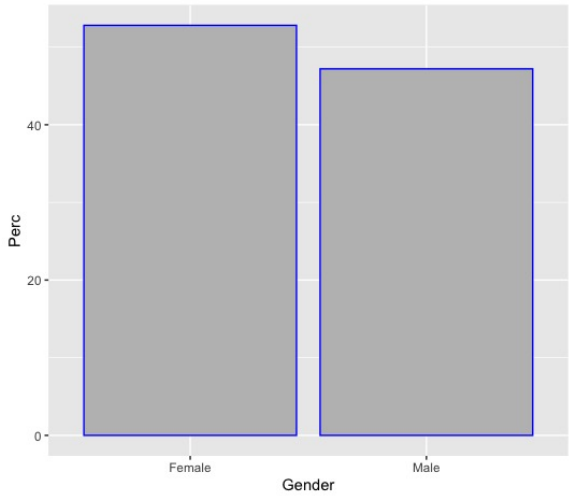
1.3 Portugal

1.3.1 Brief description

Portugal is a country in Europe with a resident population of around 10.2 million inhabitants.
According to the Nacional Statistics Institute, INE (Instituto Nacional de Estatística), [5] "Demographic Statistics - 2019":

- Concerning to **sex**,
  - **47,2%** of the population is **male**;
  - **52,8 %** of the population is **female**.
- In terms of **age**,
  - **13,6%** of **young people** (0-14 y/o);

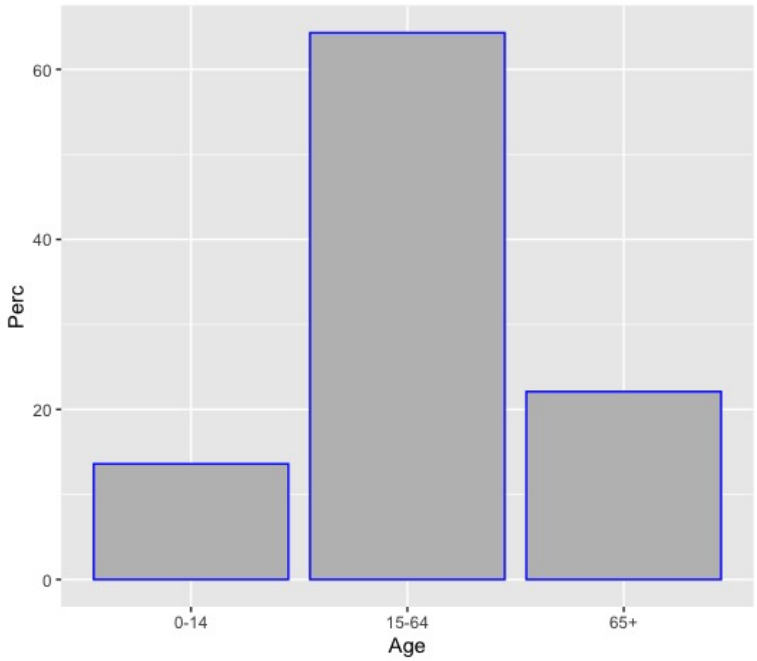
Figure 1.2: Portugal: Distribuição do Género



– **64.3%** of people of **active people** (15-64 y/o);

– **22.1%** of **elderly people** (65+ y/o).

Figure 1.3: Portugal: Distribuição da Idade



1.3.2 Regions

In terms of NUTS II regions, Portugal is divided in seven groups. The structuring of the Portuguese territory according to the New Territorial Units for Statistical Purposes, NUTS 2013, in application in the National Statistical System since 1 January 2015 is composed of seven NUTS II: North, Centre, Lisbon Metropolitan Area (AML), Alentejo and Algarve regions, on the mainland, and the two autonomous regions. [11]

Figure 1.4: Portugal's Regions

We'll describing them below, also mentioning some statistical data that may be interesting.

All information related to the labor market in the regions was obtained by consulting the [6] website of the European Commission. The demographic percentages presented are based on data from National Institute of Statistics of Portugal, INE.

North

The employment structure in the North region presents 3 sub-regions with specific characteristics: The Porto Metropolitan Area, with a strong incidence of services (mainly trade) with greater technological and knowledge intensity. A surrounding shore, where industrial employment is higher than the national average and rural areas, where nearly half of employment is concentrated in agriculture or non-commercial services.

In terms of **demographics**:

- Population density: 34,73% .
- Gender: 47,2% male and 52,8% female.
- Age:
  - 12,63 % aged 14 and younger;
  - 66,43% aged 15 to 64;
  - 20,94% aged 65 and older.

Center

In this region, the Services sector is the most relevant in terms of employment - with emphasis on Trade and Vehicle Repair, Health and Social Support Services and the Education.

In terms of **demographics**:

- Population density: 21,54% .
- Gender: 47,42% male and 52,58% female.
- Age:
  - 12,05 % aged 14 and younger;
  - 63,42% aged 15 to 64;
  - 24,53% aged 65 and older.

Metropolitan area of Lisbon

This is the region with the highest population density in the country. It's the region with the highest concentration of services, with emphasis on services provided mostly by the Public Sector; Education; Health and social support services.
In terms of **demographics**:

- Population density: 27,81% .
- Gender: 46,71% male and 53,29% female.
- Age:
  - 15,88% aged 14 and younger;
  - 62,04% aged 15 to 64;
  - 22,08% aged 65 and older.

Alentejo

This is the region with the lowest population density in the country. Most of the region's territory is dedicated to Agriculture, allied to cattle breeding and also forestry.
In terms of **demographics**:

- Population density: 6,84% .
- Gender: 47,97% male and 52,03% female.
- Age:
  - 12,4% aged 14 and younger;
  - 62,05% aged 15 to 64;
  - 25,55% aged 65 and older.

Algarve

The economic structure of this region is based on 5 strategic sectors associated with the region's natural resources: hospitality, catering and tourism, health, creative activities, agri-food and maritime activities. In terms of **demographics**:

- Population density: 34,73% .
- Gender: 47,2% male and 52,8% female.
- Age:
  - 12,63% aged 14 and younger;
  - 66,43% aged 15 to 64;
  - 20,94% aged 65 and older.

Azores autonomous region

The region's economy is fundamentally based on activities mainly from the Public Sector (Public Administration, Social Security, Education, Health and Social Support activities). The activities of Trade and Repair of Vehicles and Accommodation and Restoration are equally important in employment in the region.
In terms of **demographics**:

- Population density: 2,36% .
- Gender: 48,55% male and 51,45% female.
- Age:
  - 15,37% aged 14 and younger;
  - 69,69% aged 15 to 64;
  - 14,99% aged 65 and older.

Madeira autonomous region

In terms of more established work areas, this region is very similar to the Azores region. Tourism has an important role for both regions.
In terms of **demographics**:

- Population density: 2,47% .
- Gender: 46,67% male and 53,33% female.
- Age:
  - 13,11% aged 14 and younger;
  - 69,91% aged 15 to 64;
  - 16,98% aged 65 and older.

1.4 COVID-19 and Vaccination in Portugal

Portugal recorded the first confirmed case of COVID-19 on March 2 and the first death occurred on March 16. Since then we have added more than 800 thousand registered cases and more than 16 thousand deaths.

Portugal was severely affected by COVID-19, being considered, in the middle of January 2021, as the worst country in the world in terms of infection and mortality rates per million inhabitants and the worst country in Europe with the highest average of cases COVID-19 daily reports, with Portugal reaching a maximum of 16432 cases and 303 deaths on 28 January 2020.

1.4.1 Vaccination

The first vaccine administered in Portugal was on December 27, 2020. The first person to be inoculated with the first dose of a vaccine was António Sarmento, 65 years old, physician and director of the Infectious Diseases

Service, Hospital de São João, in Porto. Since then, more than 2 million doses of vaccine have been administered.

Although the entire Portuguese population has access to the vaccine, Portugal opted for a vaccination plan divided into 3 phases, depending on their vulnerability and clinical picture.

Priority groups were defined, as an example the health professionals, professionals and residents of residential structures for the elderly and similar institutions, were part of the first phase of vaccination.

Chapter 2

The Shiny App

2.1 Data :: Structure and Features

The data that we used to develop this application was downloaded from the European Centre for Disease Prevention and Control (ECDC) which is an EU agency aimed at strengthening Europe's defenses against infectious diseases.
As we can read in the ECDC website [7], they are providing an overview of the progress in the rollout of COVID-19 vaccines in adults across EU countries.

Figure 2.1: ECDC logo

The data is collected through The European Surveillance System (TESSy) and they publish the updated data, every week on Thursdays. In Portugal, the entity responsible to send the data is the Ministry of Health.

2.1.1 Structure of the data

The dataset downloaded in each interaction with the app in format `csv`, has the following columns:

- **YearWeekISO** - Date information in weeks: week number and year;
- **FirstDose** - Number of first dose vaccine administered to individuals during the reporting week;
- **FirstDoseRefused** - Number of individuals refusing the first vaccine dose;
- **SecondDose** - Number of second dose vaccine administered to individuals during the reporting week;
- **UnknownDose** - Number of doses administered during the reporting week where the type of dose was not specified;
- **NumberDosesReceived** - Number of vaccine doses distributed by the manufacturers to the country during the reporting week;
- **Region** - Region of the Reporting Country.

  In order to know the region to which these codes relate to, it was necessary another excel document. This document also brought another important component, namely, the population of each region. Having said this:

  - PTCSR01 - Alentejo; Population - 466 690
  - PTCSR02 - Algarve; Population - 438 406
  - PTCSR03 - Autonomous Region of Azores ; Population - 242 796
  - PTCSR04 - Centre; Population - 1 650 394
  - PTCSR05 - Metropolitan Area of Lisbon; Population - 3 674 534

  - PTCSR06 - Autonomous Region of Madeira; Population - 254 254
  - PTCSR07 - North; Population - 3 568 835

- **Population** - Age-specific population for the country (unfortunately, in Portugal, this hasn't been exactly right, it only has the total population of Portugal);
- **ReportingCountry** - The country that is providing the information;
- **TargetGroup** - Target group for vaccination;
- **Vaccine** - Name of vaccine;
- **Denominator** - Population denominators for target groups.

  We will show next, a little example of the dataset already filtered for Portugal.

	YearWeekISO	FirstDose	FirstDoseRefused	SecondDose	UnknownDose	NumberDosesReceived	Region	Population	ReportingCountry	TargetGroup	Vaccine	Denominator
1	2020-W52	8	*NA*	0	0	*NA*	PTCSR02	10295909	PT	ALL	COM	*NA*
2	2020-W52	4977	*NA*	0	0	9750	PT	10295909	PT	ALL	COM	8578859
3	2020-W52	1654	*NA*	0	0	*NA*	PTCSR05	10295909	PT	ALL	COM	*NA*
4	2020-W52	19	*NA*	0	0	*NA*	PTCSR01	10295909	PT	ALL	COM	*NA*
5	2020-W52	2671	*NA*	0	0	*NA*	PTCSR07	10295909	PT	ALL	COM	*NA*
6	2020-W52	614	*NA*	0	0	*NA*	PTCSR04	10295909	PT	ALL	COM	*NA*
7	2020-W52	169	*NA*	0	0	*NA*	PT	10295909	PT	Age18_24	COM	775701
8	2020-W52	3154	*NA*	0	0	*NA*	PT	10295909	PT	Age25_49	COM	3360653
9	2020-W52	1092	*NA*	0	0	*NA*	PT	10295909	PT	Age50_59	COM	1485319
10	2020-W52	561	*NA*	0	0	*NA*	PT	10295909	PT	Age60_69	COM	1299674
11	2020-W52	1	*NA*	0	0	*NA*	PT	10295909	PT	Age70_79	COM	981649

Figure 2.2: Excerpt from the dataset

2.2 Development

The Shiny R framework is an R / RStudio package that makes it incredibly easy to build interactive web applications using R *only* code.
Furthermore, it has the advantage of allowing us to create efficient and very attractive reports and data visualizations, on which the user can interact, exploring, for example, a dataset.

Basic Structure of the App

Shiny apps are divided into two main parts:

- `server.r`
- `ui.r (UI)`

UI specification

The `ui` specification defines:

- **Layout functions** to configure the visual structure of the html page that will be generated when the application is executed.
  The `fluidPage()` function embeds and configures all the necessary and sufficient HTML, CSS and JavaScript code for the application.
  To create more complex layouts, you need to call layout functions inside `fluidPage()`.

- **Input control functions** that will allow the user to interact with the application. Functions such as `sliderInput()`, `selectInput()`, `textInput()`, `numericInput()` can be used.
  All these functions have `inputID` as their first argument – a simple and unique string with the same restrictions as the object names of R. This identifier is used to bind the `ui` to the server. For example, if in `ui` the `inputID = \name"`, the server will access it by using `input$name`.
  The second argument, `label`, is also important as it contains the text that appears in the application's layout.

- **Output controls** that indicate where to place output with reactive behavior. Examples of output control functions are `textOutput()`, `tableOutput()`, among others.
  As in input control functions, the first argument must be the unique ID. If, for example, in `ui` an ID with the name "plot" is defined, on the server the access is `output$plot`.

Server specification

In the server specification, functions that allow calculations to be performed and updated (using reactivity) are implemented. The server controls what data will be through the UI. The server will be where you upload and collate the data and then set your options (ie graphics) using input from the UI.

For this purpose, specific rendering functions (render functions) are used. Each `render{Type}` function produces a specific type of output (for example, `renderTable()` produces tables, `renderText()` produces text). Usually these functions are paired with `{Type}Output` functions (for example `renderTable()` is paired with `tableOutput()`).

Even though we created two separate files for our application, namely `server.r` and `ui.r`, shiny supports single file applications.
A single file configuration puts both the server and user interface code in a single `app.R` file, whereas the multiple file configuration puts them in their own separate files.
Functionally, these configurations will produce the same app. The multiple file configuration is generally preferred, especially for larger applications, as it usually makes code easier to manage. For smaller apps, the single file configuration is likely a more efficient way to go.

2.2.1 Reactivity in shiny

Like in other web applications, in Shiny, you express server logic using reactive programming.
The main idea of reactive programming is to specify the dependency graph so that when an input is changed, all related outputs are also updated.
Reactivity is what makes applications responsive. Allows the application to update itself instantly whenever the user interacts with the application, requesting any visualization, with updated data.
In Shiny, reactivity creates the illusion that changes to input values auto-

matically flow to the outputs – graphics, text, and tables that use the input – and cause them to update. This flow behavior (such as current in a river or electricity) that pushes information from input to output is not real. In fact, in R an expression is only updated when it is executed (lazy evaluation).

So, the idea is: the server re-execute the instructions very frequently, so it knows the input change very quickly, and acts as if it were bound to the input or as if input pushed its new value to the output.

This is the approach that Shiny uses to create reactivity. This is why session R is busy when starting a Shiny application (for example, it is not possible to use the R console when the application is running). The server is using session R to monitor the application and rerun the expressions.

However, Shiny takes this approach a step further, creating an alert system that lets you know exactly what expressions need to be rerun.

Thus, although the server still checks the application at regular intervals (of microseconds), instead of rerunning each expression, it only executes the expressions that the alert system flagged as out of date. If alerts appear, the server executes all expressions that are out of date at the time - this event is called flush and is the key to reactivity as it allows the server to update the application as quickly as possible, appearing to flow instantly from inputs to outputs.

We can see several examples of the use of reactivity in the code of our app. For example, in `UI` (2.3) and in `server` (2.4)

```
output$myPlotPais <- renderPlot({

  res <- myReactiveDataPais()
```

Figure 2.3: reactivity:: UI

```
tags$div(style="col-12",checked=NA,plotOutput("myPlotPais")%>%
                                withSpinner(color="#cccccc"))),
```

Figure 2.4: reactivity:: server

2.2.2 Data visualization

Data visualization through maps, tables or graphs allows contextualizing and understanding information in a natural and efficient way.

R has libraries that allow us to get high quality graphics. In this report we mainly use `ggplot2`, based on its own graphics grammar, where the graphical representations are obtained by superimposing several stacking levels, allowing great flexibility in the construction of the graphics (Hadley Wickham, 2005). [10]
Layers consist on the data (data.frame format), aesthetic attributes (data mapping definition), geometric objects (visual elements), facets (`facets()`) functions that allow the execution of multiple graphics related to segments of data defined by factors), statistical transformations, coordinates (space where data is represented) and theme (`theme()`, functions allow changing individual components of a theme).

2.3 Implementation

2.3.1 User Interface

In terms of `User Interface`, it's pretty simple to implement an application with a good and intuitive layout. In our application, we have two main structures that were chosen, not only because, in our opinion, it's prettier, but also to show some different ways to see information. Our two structures are the followed:

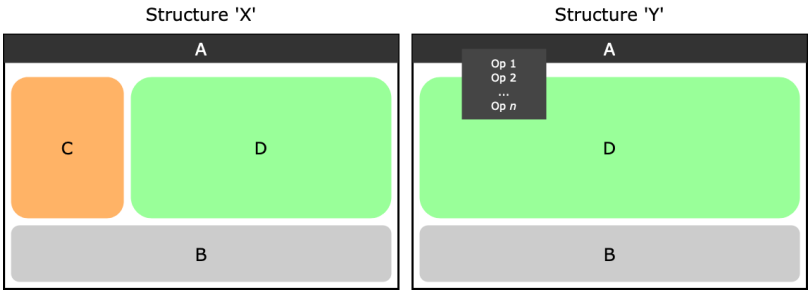


Figure 2.5: Structures overview

We named the structures as *'X'* and *'Y'*, merely to be easier to see and explain the differences between them.
As we can see, both layouts have something in common:

- **A** it's our navigation bar, commonly used in a lot of websites and web applications;
- **B** it's an area made by us with a few of `CSS` that will show some relevant information of the data;
- **D** it's the area where all the data and information will appear.

In the `Structure 'X'`, the area **C** represents an area where the user can select the plot/s that want to display. It's possible to display more than one plot at the same time.
In the `Structure 'Y'`, coming from **A**, there are plenty of options (**Op 1, Op 2, ..., Op *n***). After selecting one of these options, it will appear both areas **D** and **B**. It's relevant to point that, in this case, in **D** will appear several plots in a dashboard style.

We would like to make reference to the `thematic_shiny` function from **thematic** library. This function updates the graphics themes according to the theme chosen in the `shinyServer` function. It's really useful to prevent the

developer to change the theme of every plot, every single time that he want to change something.

2.3.2 Portugal

For the Portugal panel, we developed five graphs that we'll explain next:
First, after read the `csv` file from our source, using the `reavtiveVal()` function, we construct a reactive value with the information filtered to only have Portugal's data. After this, with a few skills in logic and programming, it's easier to work with the resultant dataset.

In short, at this moment we have a reactive value with information only from Portugal.
We will explain our developed plots, but only the first one will be fully described. The others are similar, so we will show only the way that we think and point some differences.

People totally vaccinated by age and laboratory

To make this plot, we started by examining the dataset relative to Portugal's data only, and realized that we needed to filter it by age group. At this stage, we create six new and filtered datasets, each one with a specific `TargetGroup`.
Then, we need to take each one of these newer datasets and filter them four times. This happens because, at this day, there are only four vaccines distributed in Portugal.
In this last step we also choose only two columns from the dataset: `YearWeekISO` and `FirstDose` or `SecondDose`, depending on the vaccine. This means that, at this moment, we have twenty four ($6 \times 4 = 24$) simple datasets with all the information simplified that we need.
At the end, we create one data frame with all this data, and also other in-

formation such as: age Group, laboratory of the Vaccine, to be possible to arrange the plot later.
The diagram (2.6) shows the complexity of the data processing.

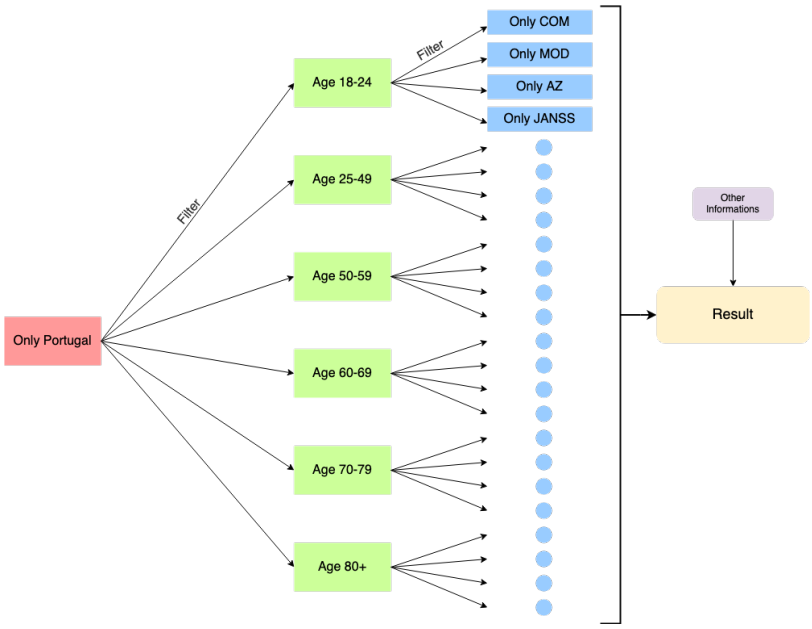


Figure 2.6: Data processing overview

The `Result` that we see in this figure, it's a reactive value with the data frame that will be used to render the plot.
This scheme, it's very similar in all data processing.

To render this plot, we use the `renderPlot()` function and save the result on a variable that will be the output displayed in the application. This behavior it's the same in all plots that we will show.
In this case, we call the data frame created at the end of the data processing, and we use the `ggplot()` function to create the plot. Then, as we want to display our plot in bars, with some visible information, we use the `geom_bar()` and `geom_text()` functions. We have some attributes in the `ggplot()` function that lets us handle with some theme and aesthetic de-

tails.

To be easier to understand, consider this:

- The 'Result' data frame will have the following structure:

Idade	**Vac**	**Doses**
Age group	Name of Laboratory/Vaccine	Number of people vaccinated
...	...	...

Next we will show the actual code to render the plot that we want:

```
  output$myPlotPais <- renderPlot({

res <- myReactiveDataPais() # Save the data set in the variable 'res'
ggplot(res, aes(x= Vac, y=Doses, fill = Idades)) +
  geom_bar(stat='identity', position = position_dodge())+
  geom_text(aes(label = Doses), vjust = -0.2,
            position = position_dodge(0.9), size = 3) +
  theme(
    axis.text.x = element_text(vjust = 1, size = 10,),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.title = element_text(face="bold", size=18),
    title = element_text(size = 20),
    panel.grid = element_blank(),
    panel.background = element_rect(fill ="#ffffff")
  ) +
  guides(fill= guide_legend(title = "Faixas etárias:")) +
  xlab('') +
  ylab('')
```

```
})
```

As you can see, this is a good example to show how we render plots, after the data processing. In the future, this example will not be displayed here again.

The result of this plot (2.7) is the following:

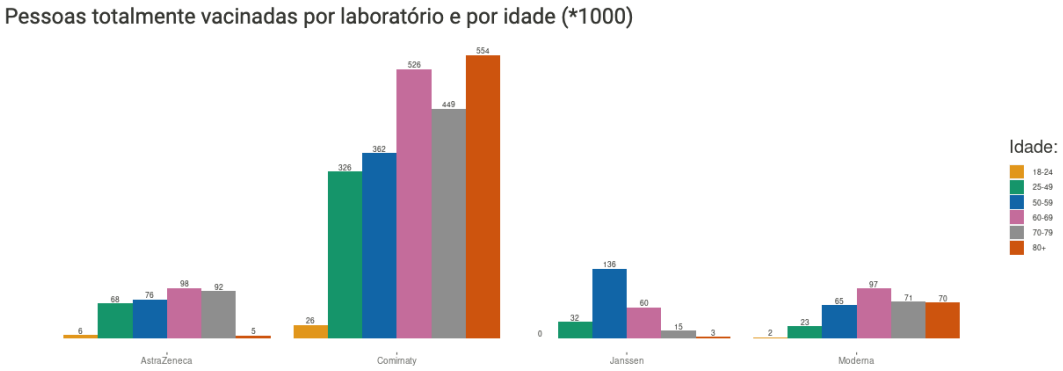


Figure 2.7: Plot of people totally vaccinated by age and laboratory

People totally vaccinated by age, by week

This is a time plot segmented by age, to visualise the behaviour of the number of vaccinated people, week by week, in each group age.

Since there are much information in this graphic, we choose to give the user options to see the data.

The user can choose if he wants to see, or not, the value in each age group by week.

It is possible to choose how many age groups you want to display simultaneously on the chart. This allows the user to compare those vaccinated according to age.

As mention earlier, the data processing it's pretty much similar, and this

is one of those cases. Such as the previous graphic, we will filter the data set that have information only from Portugal, creating new data sets with specific information that we need to move on. In each one of these data sets, we need to preserve the week of vaccination and, for each week, sum all the people vaccinated by age group. After the data processing, the result should be something with this structure:

Datas	***Vacinados18_24***	***Vacinados25_49***	***Vacinados_ageGroup***	...
Week_n	`int`	`int`	`int`	...
Week_{n+1}	`int`	`int`	`int`	...
...	...	...	...	...

To render this plot, the process is the same as the previous plot, except that, in this one, we'll save the `ggplot()` in a variable (this is a feature of the graphics produced with `ggplot2`). This makes possible that, in the future, we can add more information to this plot.
This information it's input passed by the user. When he chooses some option, our plot will update, adding or removing the information that he chose. Next we'll show different ways to observe the plot (2.8)

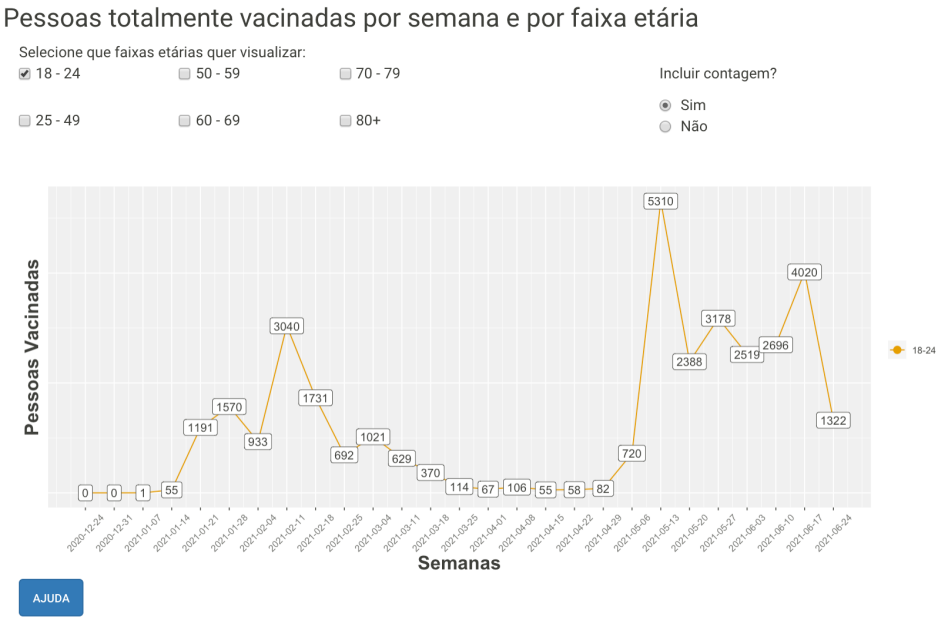


Figure 2.8: Plot with count, of the age group 18-24

As you can see, in the graphic (2.8), we include count and the age group selected is only 18-24.

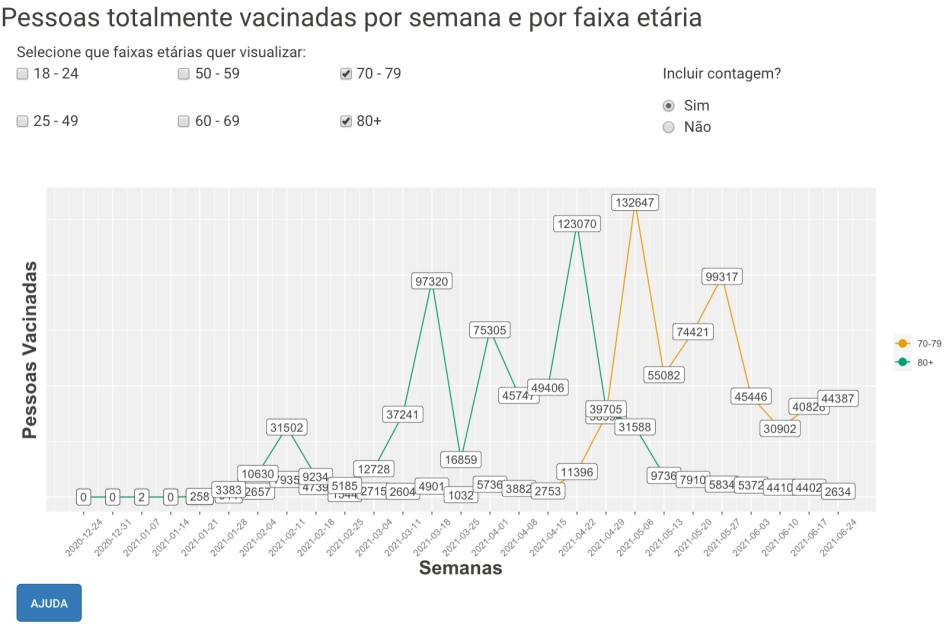


Figure 2.9: Plot with count, of the age groups 70-79 and 80+

In the graphic (2.9), we include count and the age groups selected are 70-79 and 80+. This makes possible for the user to compare information. As you can see, if you want to have an intuitive view of the evolution in every week, with the values displayed turns out to be hard.
That's the reason to give the option without the count. As shown next, Figure 2.10, it's possible even to see some information justified by the vaccination plane and more.

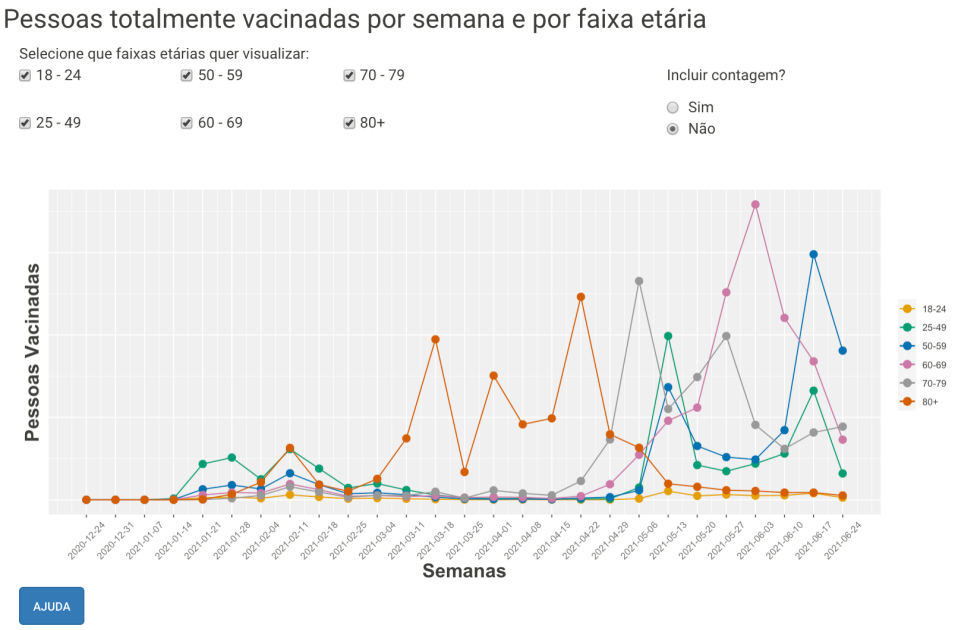


Figure 2.10: Plot without count, of all age groups

People totally vaccinated by region, by week

This one it's exactly the same as the one before, with the difference of filter the total of people vaccinated by region, instead of group age.

Vaccines administered by laboratory

In terms of data processing, there's not any new way to do things. The result will be the shown in (2.11).

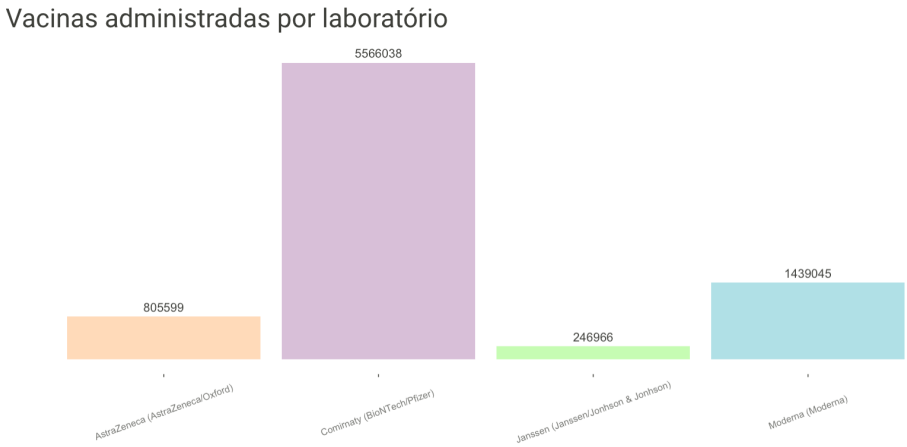


Figure 2.11: Plot with the number of administered vaccines by laboratory

In this plot, all the administered doses are referent of the first and second dose of each vaccine.

Percentages of vaccinated and unvaccinated people

In this graphic, the data process is similar to the previous graphics with the exception that now, we want to work with percentages. With this in mind, we know the population of Portugal and with simple math it's easy to create the reactive value to render the plot.

This graphic (2.12) uses a function from another package ('waffle'), that allow the function `renderPlot()` to display the information in a waffle plot.

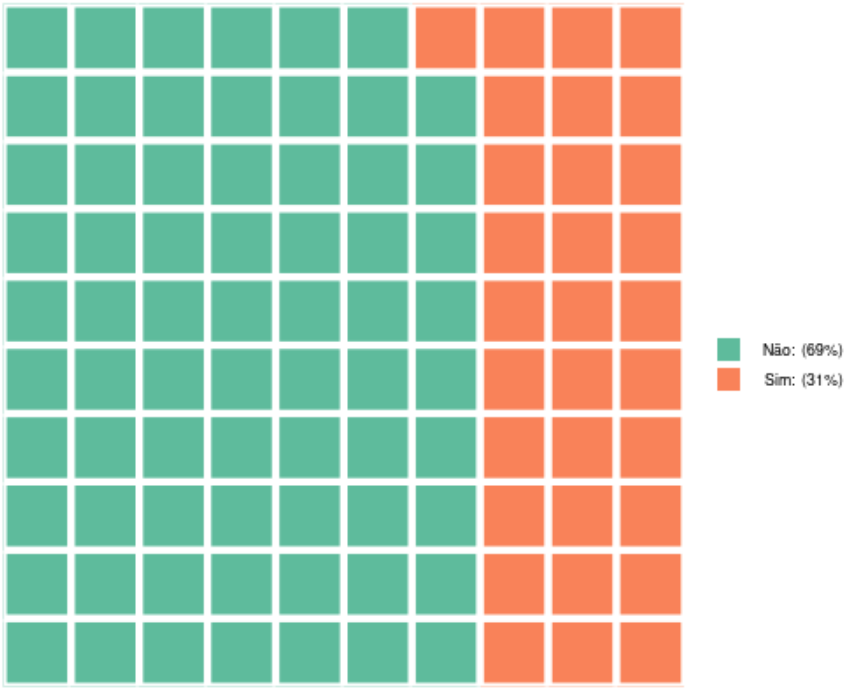


Figure 2.12: Plot with the percentages of vaccinated/unvaccinated people

We choose to develop this plot this way, not only because, in our opinion, it looks good, but also to show that **ggplot2** can be very versatile.

2.3.3 Regions

First of all we began by filtering our dataset as soon as the App begins to load so that further ahead when we need the data for the selected region it's already filtered. For this purpose we used the `reactiveVal()` function to create eight reactive value objects. One for each region and another one for the whole country. The package used for this endeavour was the `dplyr` package and was used as follows:

- We begin by creating the reactive object:

```
loaded_Alentejo <- reactiveVal()
```

- And end by loading it with the filtered data:

  ```
  Alentejo <- loadedData2() %>% filter(Region == "PTCSR01")
  loaded_Alentejo(Alentejo)
  ```

  Where `loadedData2()` is the data for the whole country.

Having done this and in order to know the selected region tab, we gave an id to the Regions tab upon drawing our User Interface so that at the point of which we started to manipulate our data we would know the region we were dealing with.

Administered vaccines by week

We begin by clearing the dataset of the unnecessary columns for the graphic. After this we are left with four columns: `YearWeekISO`, `FirstDose`, `SecondDose` and `Vaccine`.
Having done this we'll need to add the doses of every vaccine available in each and every week and place all those values in an array. What we mean by this is: If a week has 4 reports, each relative to a certain type of vaccine, we'll need to add every First and Second Dose of all those four types and associate that sum to the week. Finally we create a data-set with the Dates and with the Doses, and since we are taking care of this data with the reactive function, when we call such function in the renderPlot we'll receive the data-set.

In the renderPlot function all we'll have to do is give the ggplot function, from the ggpplot2 package, the data-frame, the aesthetic mappings of the contents of the data-frame, and add geom_bar() with stat='identity' so that ggplot2 knows that we'll provide the y values. Here's an example on how that would look like:

```
output$yourPlotID <- renderPlot({

  res <- myReactiveDat()      ## the data-frame

  ggplot(res, aes(x=Data, y= Doses,fill=Doses)) +
    geom_bar(stat='identity')+
    scale_fill_gradient(low="#63ACAA",high="#507D94")+          ##visual settings
    theme(axis.text.x = element_text(angle = 45, vjust = 0.5),
          axis.title = element_text(face="bold", size=18),
          title = element_text(size = 20)) +
    xlab('Week') +
    ylab('Doses')
})
```

Here's how this graphic would look like (2.13):

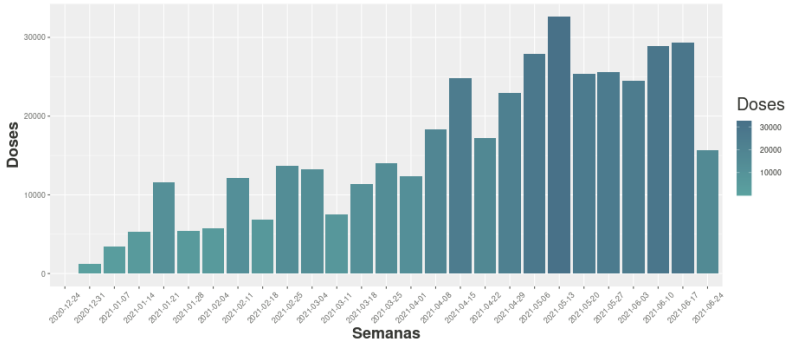


Figure 2.13: Administered vaccines by week

Cumulative value of vaccines by week

This graphic will follow the exact same principle of the previous one, however, this time we'll have to apply the "cumsum" function from RStudio to our Doses vector. This function returns a vector whose elements are the cumulative sums of the elements of the argument and was used as follows:

```
output$yourPlotID <- renderPlot({

  res <- myReactiveDat()

  (ggplot(res, aes(x=Data, y=cumsum(Doses),group=1))
    + geom_line(color="#63ACAA")
    + geom_point(color="#507D94")
    + theme(axis.text.x = element_text(angle = 45, vjust = 0.5),
            axis.title = element_text(face="bold", size=18),
            title = element_text(size = 20))
    + xlab('Week')
    + ylab('Doses')

  )
})
```

As you can see, the data used is the same as the one in the first graphic, except for the fact that we'll apply the function to the 'Doses' vector. And since we're trying to see the progress we use the geom_line function as well as the geom_point function for a better perception of the results.

Here's how this graphic would look like (2.14)

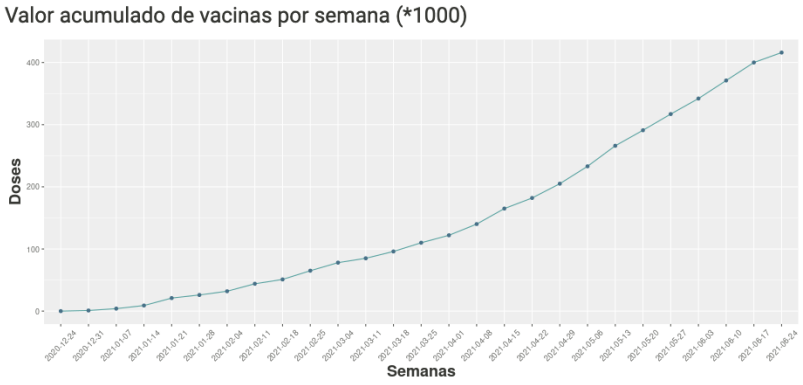


Figure 2.14: Cumulative value of vaccines by week

Percentage: Vaccinated/Incomplete Vaccination

For this chart we had to pay careful attention to the data for each region since there are certain Vaccine brands that are yet to be administered in some regions. Having this in consideration and the vaccination treatment of each vaccine, the procedure to find the intended values to draw the waffle chart will consist in getting the sum of all the Second Doses from Moderna, Pfizer and AstraZeneca vaccines and adding the sum of all the First Doses of the Jansen vaccine. This total will be the amount of people that are completely vaccinated and if we subtract it from the population of the region we'll get the amount of people that haven't completed the full vaccination treatment.

After getting both values, in order to get the percentages, all we had to do was dividing both numbers by the region population and multiplying the result by 100. Then we create a vector with both percentages and since we're working on this data with the reactive function, when we call it we'll receive said vector.

Finally we draw our Waffle chart (2.15)

```
output$yourPlotID <- renderPlot({
  resWaffle <- myReactiveWaffles()
```

```
  val_names <- sprintf("%s (%s%%)",
                       c("Vacinação incompleta: ",
                         "Completamente Vacinados: "),
                       resPie)
  names(resWaffle) <- val_names
  waffle::waffle(resPie)
})
```

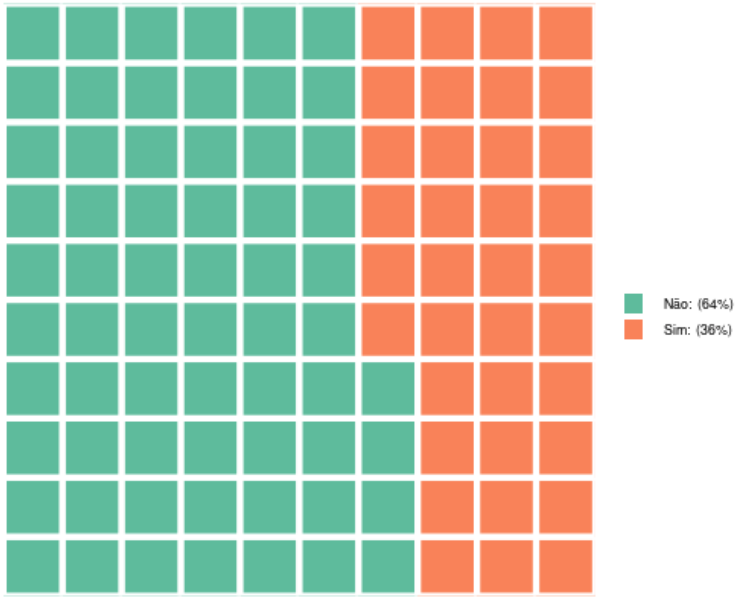


Figure 2.15: Percentage: Vaccinated/Incomplete Vaccination

Administered vaccines by laboratory

We start off by filtering the data for every vaccine. After this we'll need to verify whether the data frame of each vaccine is not empty just in case the region we're dealing with has not received any vaccines of that laboratory. If the data frames are not empty, we'll create a variable for each one with the sum of both the First Doses as well as the Second Doses. At last a vector

with these values is created so that, when we call the reactive function in the renderPlot, we can access this data.

Here's how the renderPlot function would look like:

```
output$yourPlotID <- renderPlot({


  res <- VacinasReg()
  ggplot(res, aes(x=Nomes, y=Totais, fill = c6)) +
    geom_bar(stat='identity')+
    scale_fill_manual(values = c("AZ"="#D8BFD8",
                                 "MOD"="#B0E0E6",
                                 "COM"="#FFDAB9",
                                 "JJ"="#c6ffb3")) +
    geom_text(aes(label=Totais), vjust = -0.6, size = 4)+
    theme(
      legend.position = "none",
      axis.text.x = element_text(angle = 20, vjust = 0.5),
      axis.title = element_text(face="bold", size=18),
      axis.text.y = element_blank(),
      axis.ticks.y = element_blank(),
      title = element_text(vjust = 1, size = 20),
      panel.grid = element_blank(),
      panel.background = element_rect(fill ="#ffffff")
    ) +
    xlab('') +
    ylab('')
})
```

As we can see, the res variable will consist of a data frame with both the names of the vaccines as well as the totals. Knowing this, the process

to draw the graphic is more or less the same as most of the others. We pass these values onto the ggplot function and after that we add the geom_bar in order to draw an bar plot (2.16).

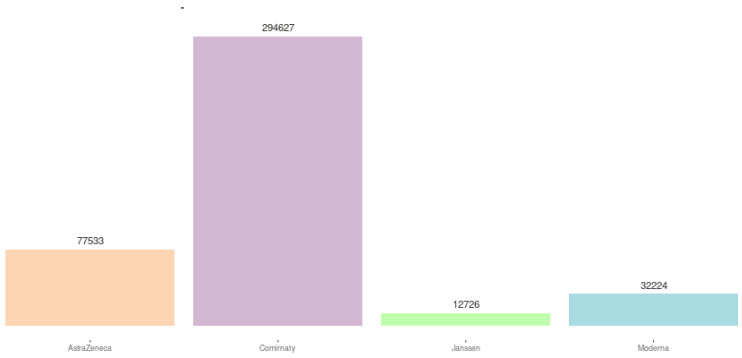


Figure 2.16: Administered vaccines by laboratory

2.4 Publication

It's possible to public the application for free, on the internet using `https://www.shinyapps.io`. [12] This is an online service for hosting Shiny apps in the cloud. RStudio takes care of all the details of hosting the app and maintaining the server. We found out pretty simple to publish the application. It's just creating an account and follow the well described steps. It tokes us a bit of time because we had to correct some errors that prevented publication correctly, but an experiment developer should do this steps quicker. Our application is located in the following link:

https://rafaelantunes.shinyapps.io/VacinacaoPortugal/

Chapter 3

Conclusion

This project made us realize the versatility of the Shiny package from RStudio. We were baffled by how easy it was to build a very useful app with such little tools. Even though we added a bit of HTML and CSS ourselves it was clear that someone with very little knowledge of front-end development and some R language knowledge could very well develop an interesting app.

Bibliography

[1] *https://www.r-project.org/about.html* , **About/R-Project website**

[2] *https://shiny.rstudio.com* , **Shiny website**

[3] *https://apps.who.int/.../origin-2020.1-eng.pdf* , **Origin of SARS-CoV-2 - World Healh Organization (WHO)**

[4] *https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/public-health/eu-vaccines-strategy$_p$t* , **EU vaccine strategy**

[5] *https://www.ine.pt/xurl/pub/71882686*, **Instituto Nacional de Estatística - Estatísticas Demográficas : 2019. Lisboa : INE, 2020.**

[6] *https://ec.europa.eu/eures/main.jsp?acro=lmilang=ptcountryId=PTcatId=57parentId=0*, **Labor Market Information - European Commission**

[7] *https://vaccinetracker.ecdc.europa.eu/public/extensions/COVID-19/vaccine-tracker.htmlnotes-tab* , **General Vaccine information - European Centre for Disease Prevention and Control**

[8] *https://www.rstudio.com/* , **RStudio website**

[9] *https://en.wikipedia.org/wiki/COVID-19$_p$andemic* , **Covid-19 information**

[10] ***https://en.wikipedia.org/wiki/Ggplot2*** **,**

Ggplot2 information

[11] ***https://ra09.ine.pt/xportal/xmain?xpid=INExpgid=ine***$_{faqs}FAQS faq_{b}oui = 242509855 FAQS modo = 1 xlang = pt$ **, Instituto Nacional de Estatística- Estruturação do território português**

[12] ***https://shiny.rstudio.com/articles/shinyapps.html***