Case Study 4: Shiny apps

AKSTA Statistical Computing

Submit a .zip in TUWEL containing the whole folder of your shiny app.

In this case study you will create a single page shiny app.

Download the data set `data_cia.rda` from TUWEL, put it in the folder of your shiny app and load it in your `app.R` using e.g.,

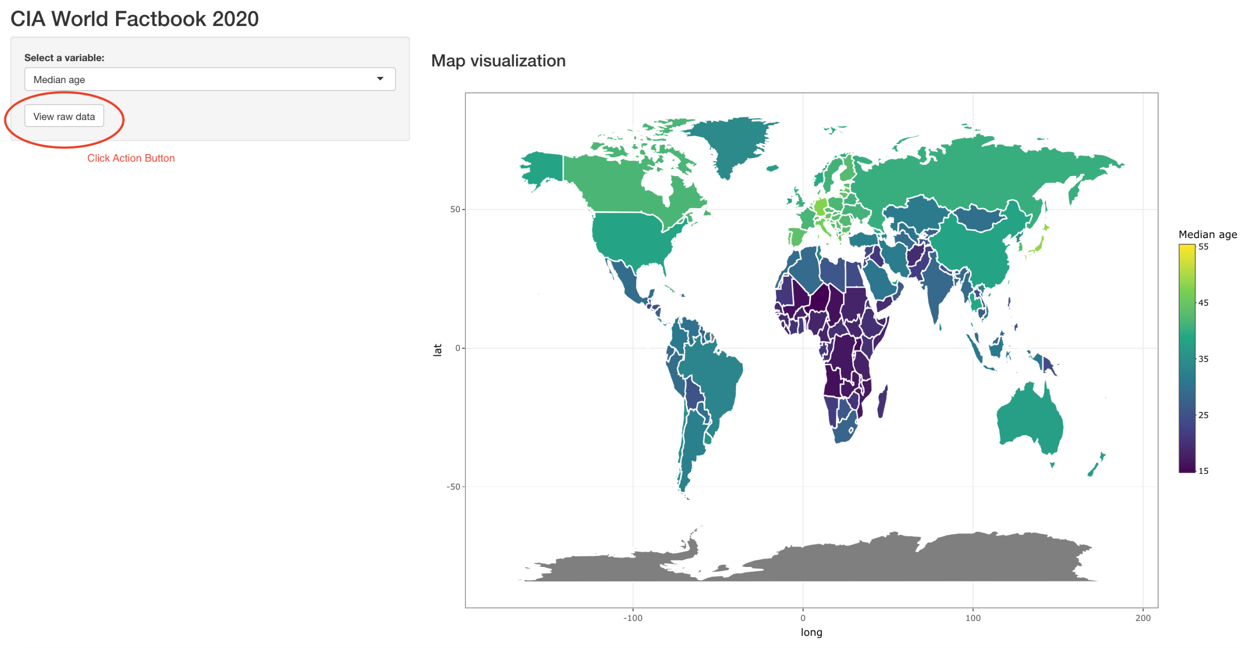
```r
load("data_cia.rda")
```

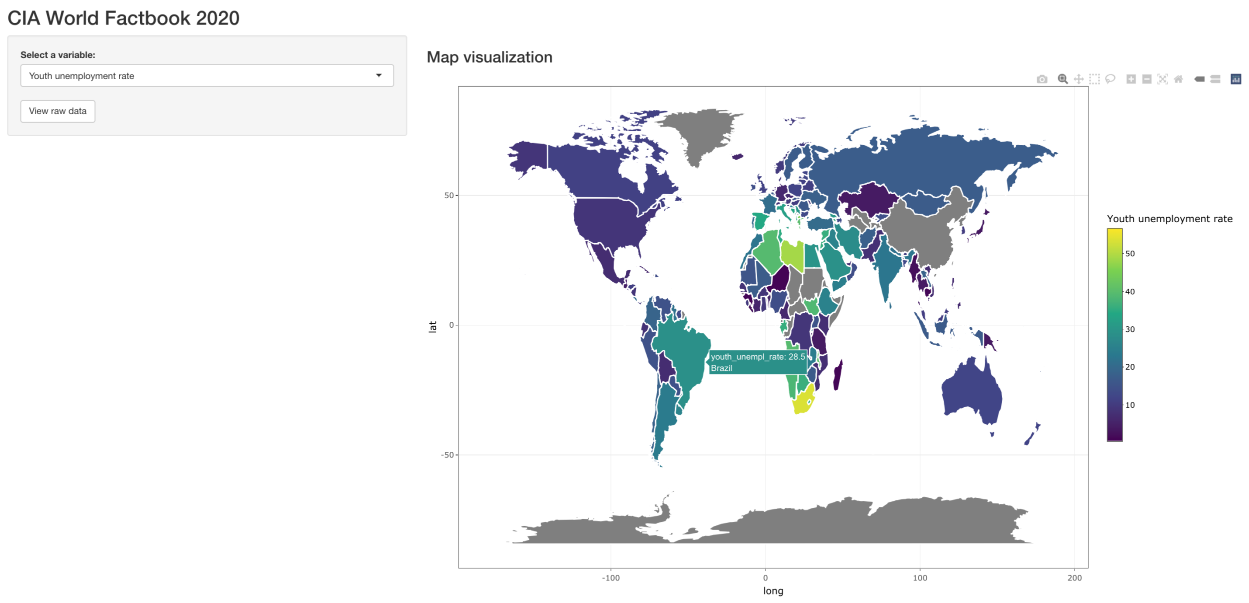
This is similar to the data you used in the previous case studies. As a reminder, the data contains 2020 information on

- median age
- youth unemployment rate

for most world entities. Additional information related to the region, sub-region and development status is provided.

The shiny app to be created should look in the following way:

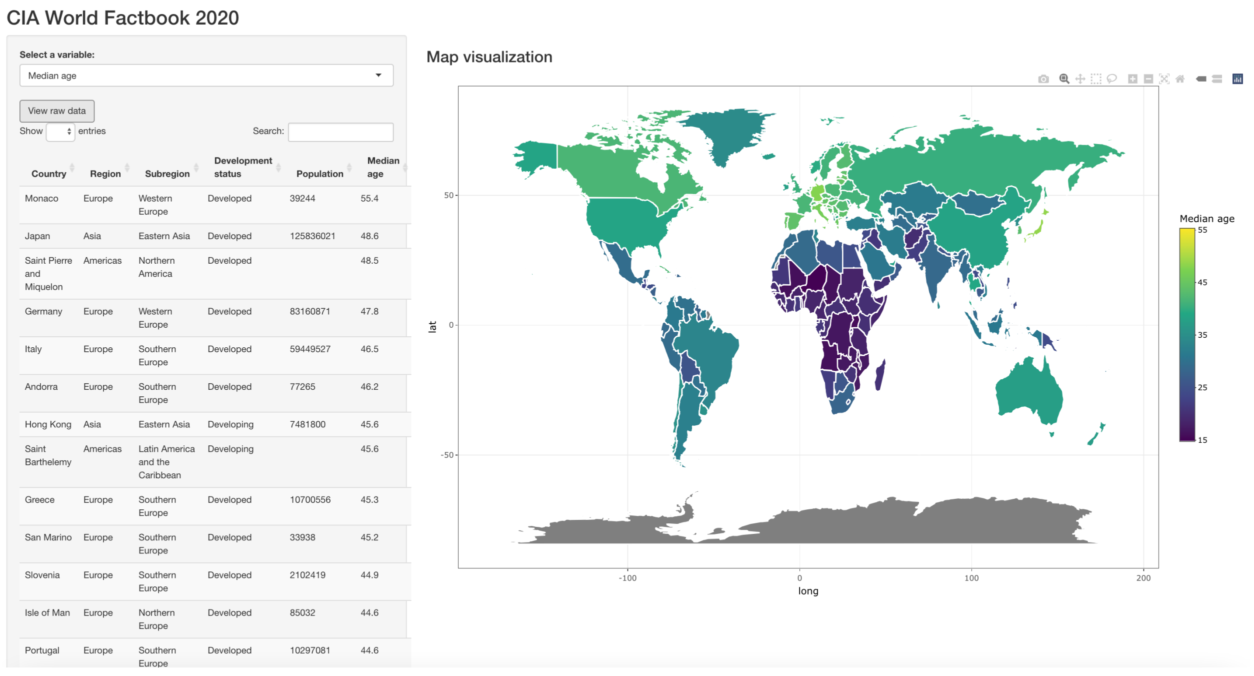




It consists of a

- title,
- side bar which contains a selection input where the variable (median age or youth unemployment rate) can be selected and an action button `View raw data`.
- in the main panel on the right hand side an interactive world map visualization using `ggplotly` is created where the color of the countries represents the value of the selected variable (to achieve this color scheme you can use `scale_fill_viridis_c()` in **ggplot2**). Note that when hovering the tooltip over the plot, we can see the name of the country and the value of the selected variable.

After pressing the action button, a data table (created by `dataTableOutput` in **shiny**) appears in the side bar. The maximum number of shown rows is 15 (you can set this in `renderDataTable`, see `?renderDataTable`).



CIA World Factbook 2020

Select a variable: Median age

View raw data

Show entries Search:

Country	Region	Subregion	Development status	Population	Median age
Monaco	Europe	Western Europe	Developed	39244	55.4
Japan	Asia	Eastern Asia	Developed	125836021	48.6
Saint Pierre and Miquelon	Americas	Northern America	Developed		48.5
Germany	Europe	Western Europe	Developed	83160871	47.8
Italy	Europe	Southern Europe	Developed	59449527	46.5
Andorra	Europe	Southern Europe	Developed	77265	46.2
Hong Kong	Asia	Eastern Asia	Developing	7481800	45.6
Saint Barthelemy	Americas	Latin America and the Caribbean	Developing		45.6
Greece	Europe	Southern Europe	Developed	10700556	45.3
San Marino	Europe	Southern Europe	Developed	33938	45.2
Slovenia	Europe	Southern Europe	Developed	2102419	44.9
Isle of Man	Europe	Northern Europe	Developed	85032	44.6
Portugal	Europe	Southern Europe	Developed	10297081	44.6

Tasks

Create a file `app.R` which can be run to generate the shiny app above. Pay attention to the details e.g.,:

- Make "nice" column names for `dataTableOutput`.
- Have a nice legend title.
- Have nice names for the variables in the selection tool (avoid using directly some codes or column names in R if they are not "user-friendly")
- ...

Points will not be subtracted if you choose e.g., different color schemes, map style etc. Points will be subtracted if the app is not user-friendly and not easy-to-use.

Hints

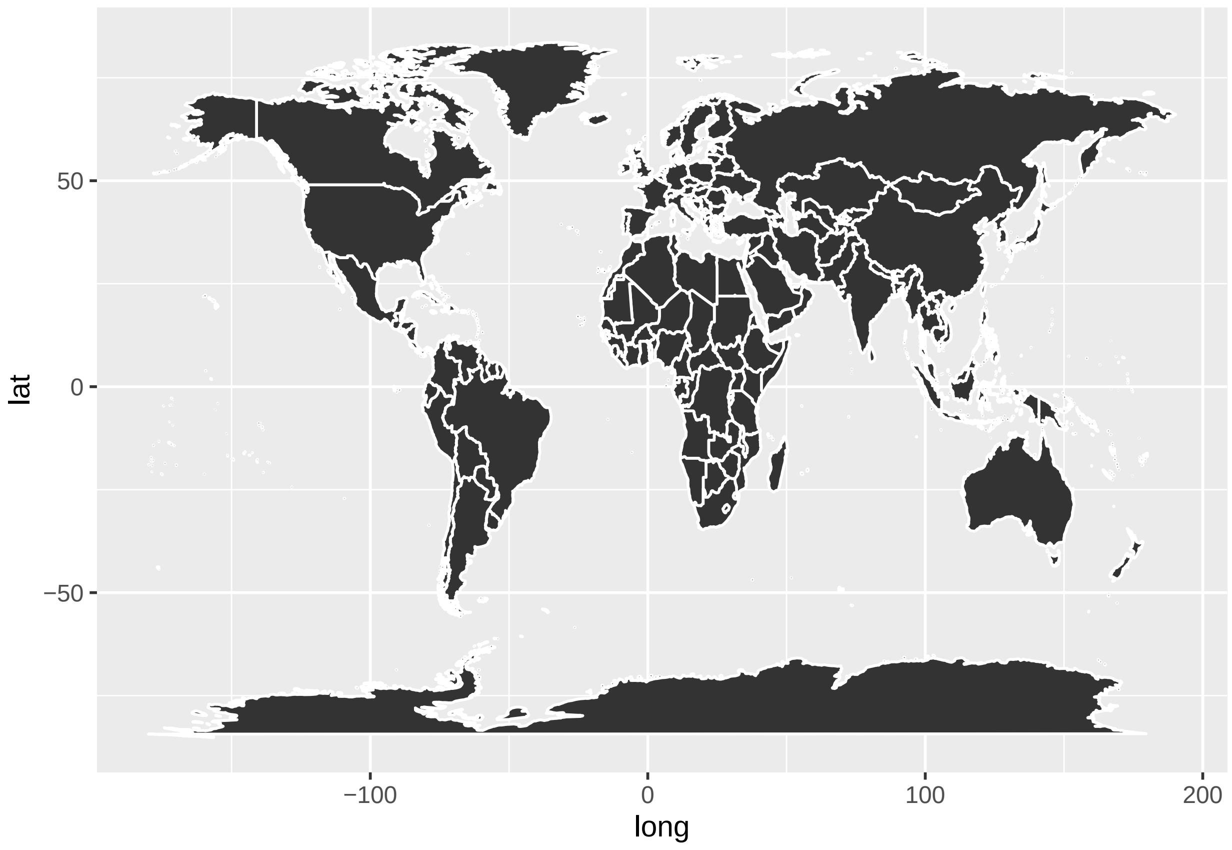
In order to prepare the data for creating the map, there are several options one can use. One option which you can use is the `map_data` function of package **ggplot2**.

```r
world_map <- map_data("world")
head(world_map)
```

```
##        long      lat group order region subregion
## 1 -69.89912 12.45200     1     1  Aruba      <NA>
## 2 -69.89571 12.42300     1     2  Aruba      <NA>
## 3 -69.94219 12.43853     1     3  Aruba      <NA>
## 4 -70.00415 12.50049     1     4  Aruba      <NA>
## 5 -70.06612 12.54697     1     5  Aruba      <NA>
## 6 -70.05088 12.59707     1     6  Aruba      <NA>
```

To plot the map you need:

```r
ggplot(world_map, aes(x = long, y = lat, group = group)) +
  geom_polygon(colour = "white")
```



Note that this built-in data set does not contain ISO codes so it will again be cumbersome to join it with `data_cia`. One useful tool in R is provided by the **countrycode** package, which offers some functionality to convert country names into one of many different coding schemes.

To get the ISO codes you can use the command:

```r
# install.packages("countrycode")
library("countrycode")
world_map$iso2c <- countrycode::countrycode(sourcevar = world_map$region,
                                            origin = "country.name",
                                            destination = "iso2c", nomatch = NA)
```

```
## Warning in countrycode_convert(sourcevar = sourcevar, origin = origin, destination = dest, : Some val
```

Matching is not perfect, but good enough.

Finally one should left join the world map data and `data_cia` based on ISO codes. Then you are good to go for the visualization.