Case Study 4: Shiny apps

AKSTA Statistical Computing

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

Additionally, if you upload your shiny app to the posit server shinyapps.io, all the team members will get an additional bonus point. In this case provide us also with the link to the application.

Groups with only one student can skip the `Multivariate analysis` *tab in the shiny app.*

Data

The data you will be using is `data_cia2.json` from TUWEL. This data set is familiar to you, as it is the one you used for Case Studies 2 and 3, but it contains some additional variables. As a reminder, the data contains 2020 information from the CIA factbook on

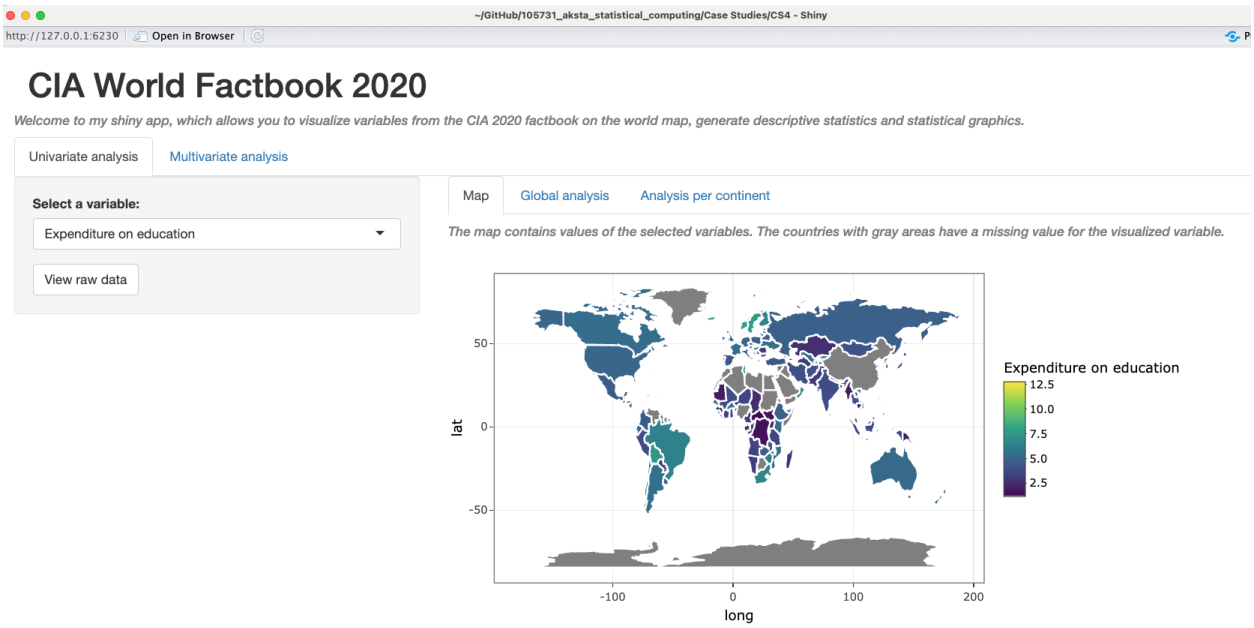
- (estimated) public expenditure on education as a percent of GDP;
- youth unemployment rate: percent of the total labor force ages 15-24 unemployed during a specified year;
- net migration rate: difference between the number of persons entering and leaving a country during the year per 1,000 persons (based on midyear population);
- electricity from fossil fuels: capacity of plants that generate electricity by burning fossil fuels (such as coal, petroleum products, and natural gas), expressed as a share of the country's total generating capacity;
- area: sum of all land and water areas delimited by international boundaries and/or coastlines in km^2.
- population growth rate: average annual percent change in the population, resulting from a surplus (or deficit) of births over deaths and the balance of migrants entering and leaving a country;
- life expectancy at birth: average number of years to be lived by a group of people born in the same year, if mortality at each age remains constant in the future.

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

Download the data set `data_cia2.json` from TUWEL, put it in the folder of your shiny app and load it in your `app.R`.

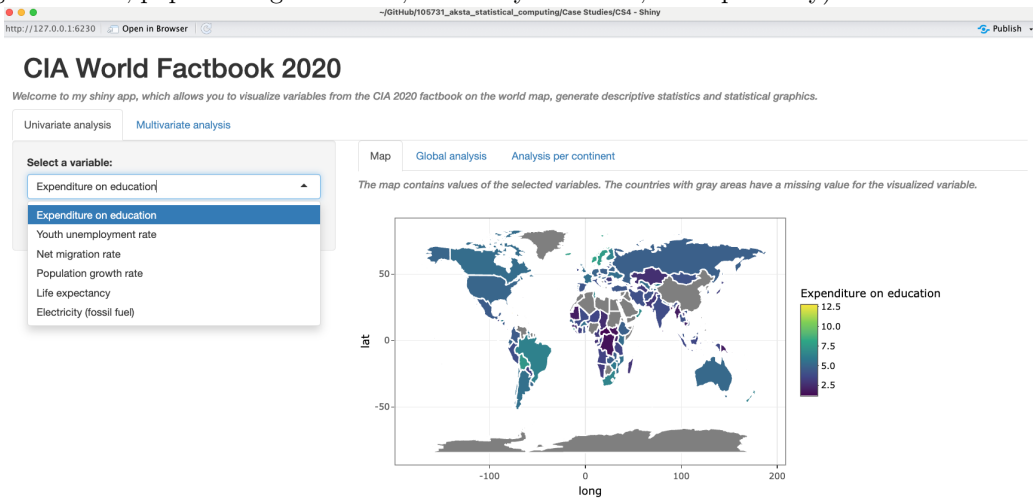
Shiny app description

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



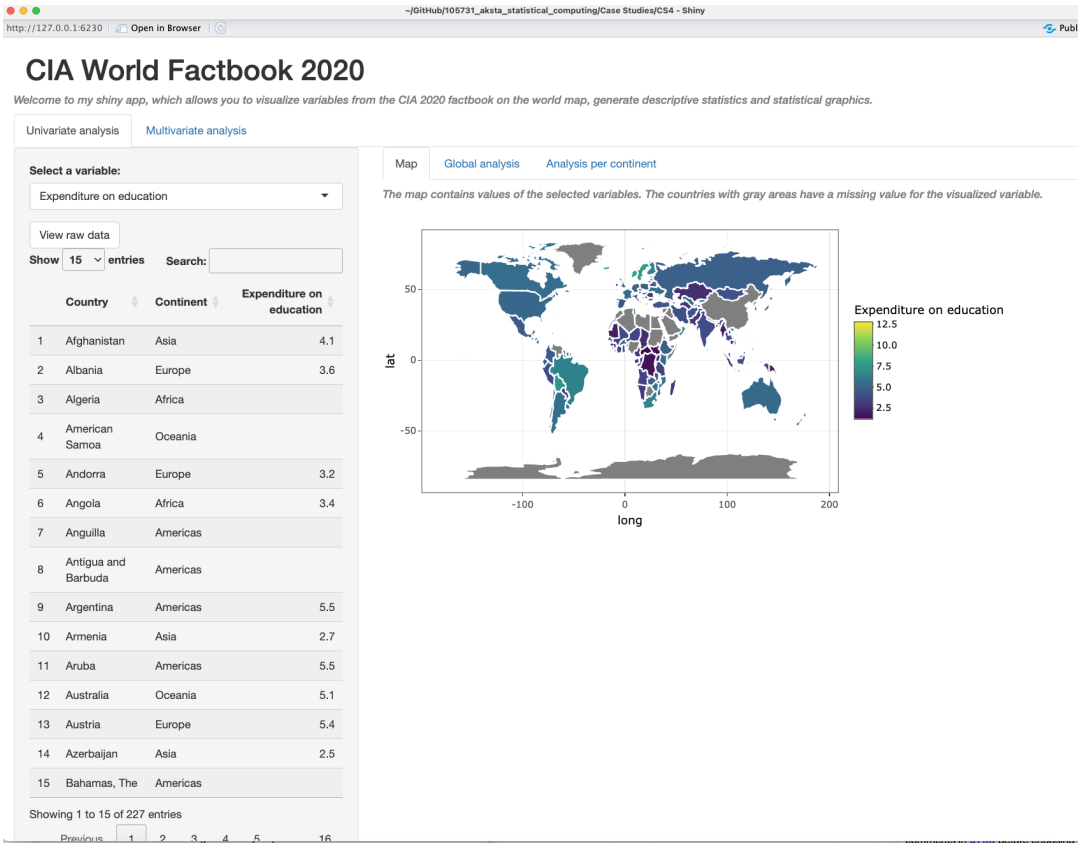
It consists of a

- Title,
- A "welcome" message briefly describing what the app does.
- two tabs: Univariate and multivariate analysis (see `tabsetPanel()` in **shiny**).
  - Univariate analysis contains a side bar and a main panel:
    - The side bar contains:
      - a selection input where a variable (among education expenditure, youth unemployment rate, net migration rate, population growth rate, electricity fossil fuel, life expectancy) can be selected.



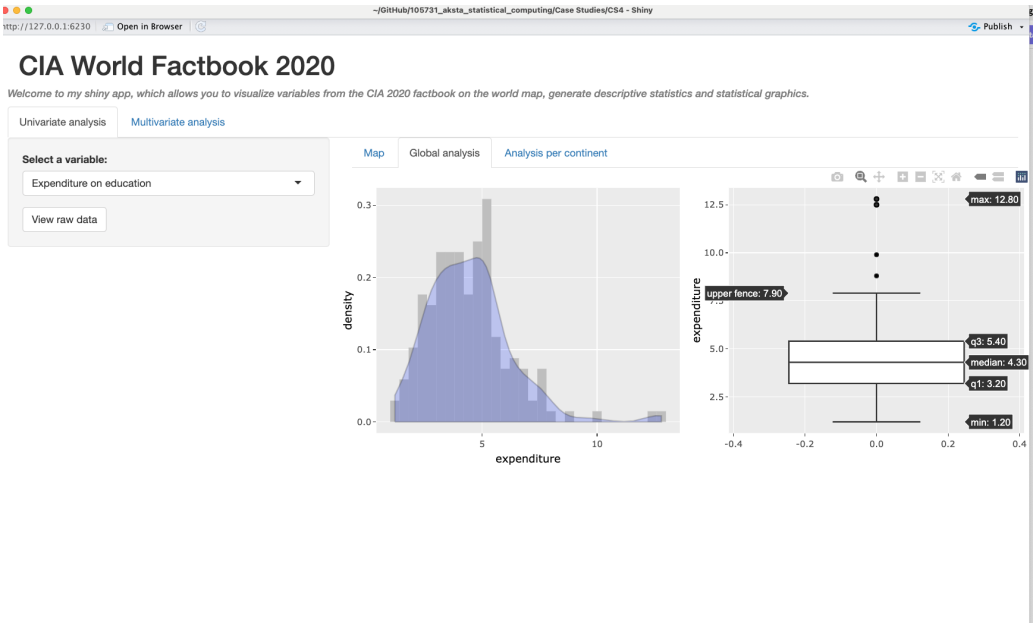
      - an action button `View raw data`. When pressed, a table appears in the side bar containing country, continent and the values of the selected variable. You can use `dataTableOutput()` and `renderDataTable()` in **shiny** or `DTOutput()` and `DT::renderDT()` (depending on

which version of shiny you are using). The maximum number of shown rows is 15.

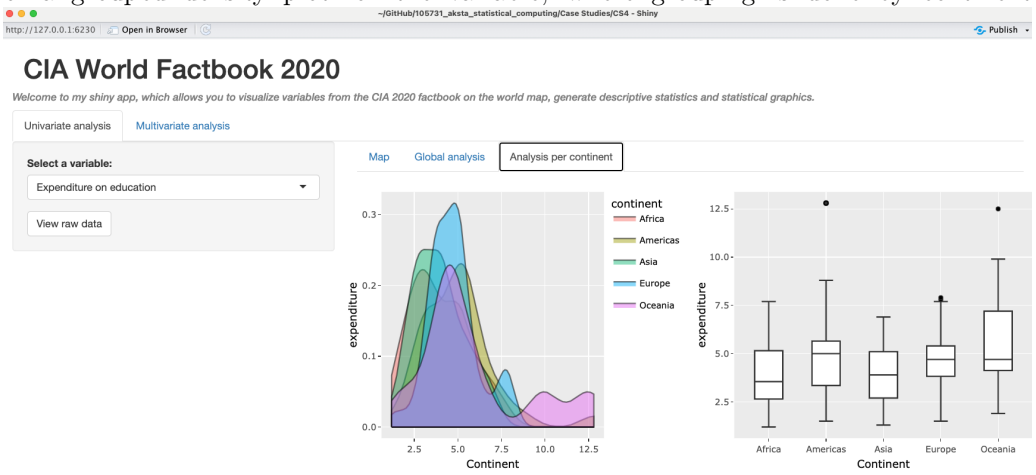


∗ The main panel contains 3 tabs:

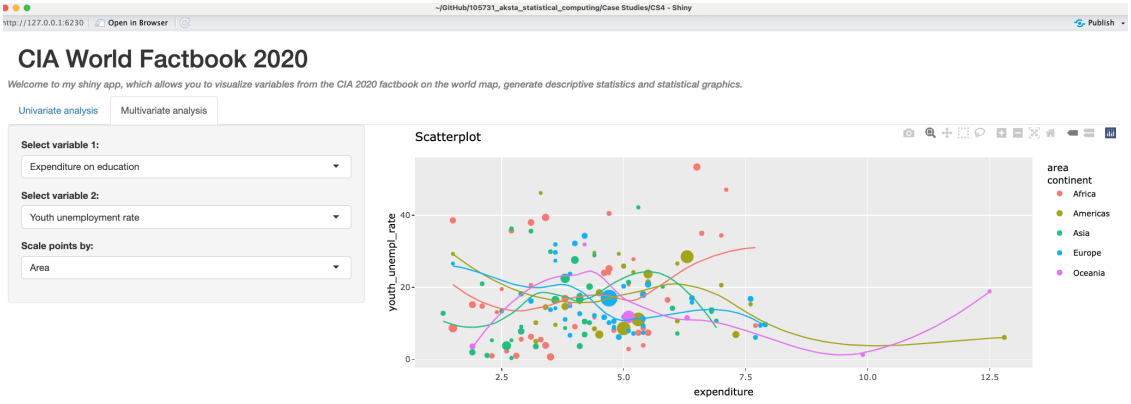
- Map: an interactive world map visualization using **ggplotly** is created where the color of the countries represents the value of the selected variable (to achieve this color scheme you can use `scale_fill_viridis_c()` in **ggplot2**). Note that when hovering the tooltip over the plot, we can see the name of the country and the value of the selected variable.
- Global analysis: a **ggplotly** version of a boxplot and **ggplotly** version of a histogram and density plot for the variable considering the whole data set and



· Analysis per continent: **ggplotly** version of a group boxplot and **ggplotly** version of a grouped density plot for the variable, where grouping is done by continent



– The multivariate analysis tab contains a side bar and a main panel:



∗ side bar with 3 selection inputs. The first two are like the one in the univariate analysis tab

and select variable. The third allows to select the variable by which points should be sized: population or area.

* main panel: an interactive scatterplot of the two selected variables, where the points are colored by continent and the size of the points is scaled according to either population or area, as specified in the side bar. Additionally, for each continent a smooth line (produced using the local regression method LOESS) is added to the plot by using `geom_smooth(method = "loess")`. The plot is created using **ggplotly**. **Hint: in order to not "size" the lines, you can use different aesthetics for `geom_point` and `geom_smooth`.**

Tasks

Create a file `app.R` which can be run to generate the shiny app above. Note that the shiny app in the screenshots is meets the minimal requirements but you can always improve it. Pay attention to the following:

- Make "nice" column names for `dataTableOutput`.
- Avoid using directly some codes or column names in R if they are not "user-friendly"
- For the variables selected from the selection tools use `reactive()`.

Points will not be subtracted if you choose e.g., different color schemes, map style etc. You are encouraged to use accessible color palettes (e.g., color blind friendly palettes). Points will be subtracted if the app is not user-friendly and not easy-to-use.

Hints

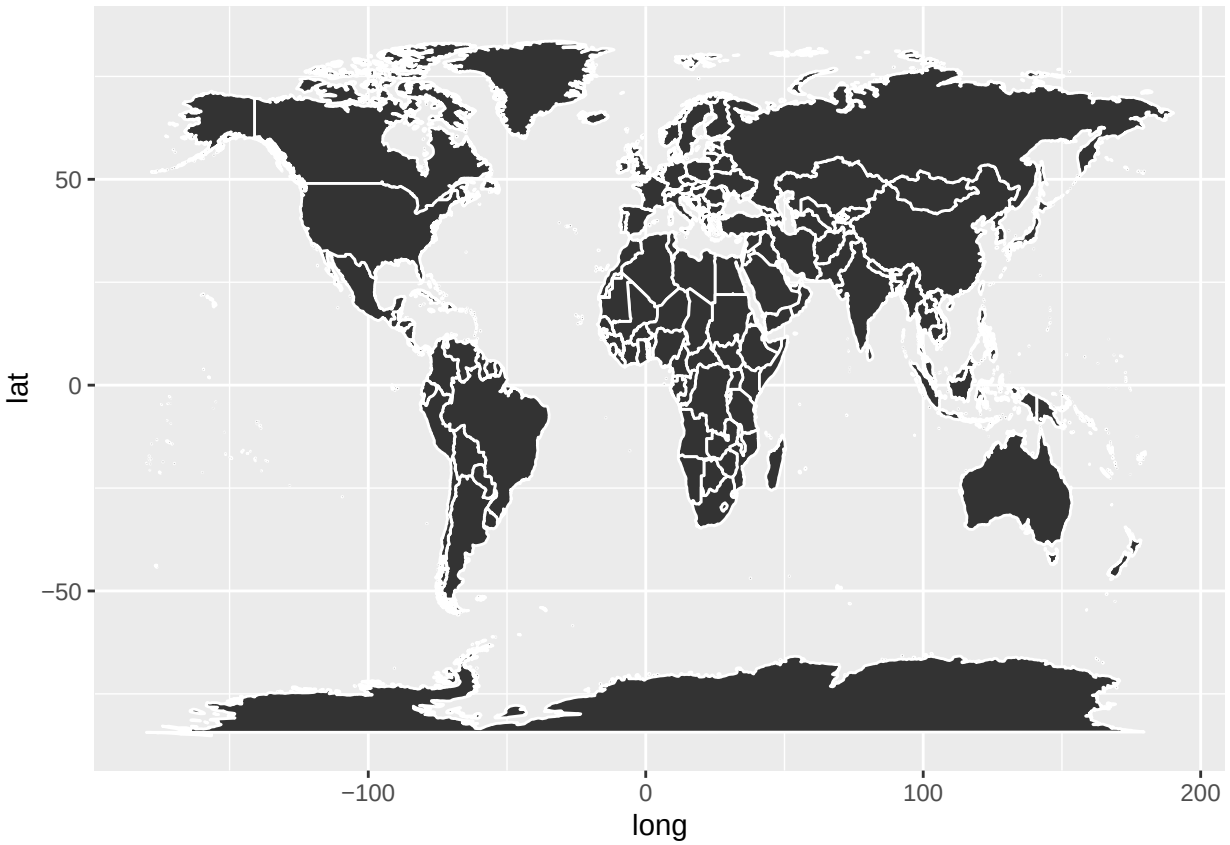
In order to prepare the data for creating the map, there are several options one can use. One option which you can use is the `map_data` function of package **ggplot2**.

```
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:

```
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:

```
# install.packages("countrycode")
library("countrycode")
world_map$ISO3 <- countrycode::countrycode(sourcevar = world_map$region,
                                           origin = "country.name",
                                           destination = "iso3c", nomatch = NA)
```

```
## Warning: Some values were not matched unambiguously: Ascension Island, Azores, Barbuda, Bonaire, Cana
```

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.