Infant Mortality Rate Around the World - Now better and easier with advanced R technique

Jinsol Kim 12/3/2017

Introduction

This post intends to further learn about data analysis using advanced R techniques with add-ons like `Shiny App` and `ggvis`. As mentioned in my previous post, one of the best way that the data analytics could be utilized is solving social problems. With the data used in the post01–infant mortality rate from the United Nations Population Division's *World Population Prospects*–this post takes a totally different approach using advanced R techniques to perform data analysis process more easily, efficiently, and accurately.

Data

The raw data was imported and manipulated using the R script file.

File Structure

The file structure of this post folder looks like this:

```
post02/
  post02-jinsol-kim.Rmd
  post02-jinsol-kim.md
  post02-jinsol-kim.html
  data/
    rawdata/
      API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv
      API_SP.DYN.IMRT.IN_DS2_en_csv_v2-dictionary.md
      API_SP.DYN.IMRT.IN_DS2_en_csv_v2-dictionary.html
      Metadata_Country_API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv
      Metadata_Country_API_SP.DYN.IMRT.IN_DS2_en_csv_v2-dictionary.md
      Metadata_Country_API_SP.DYN.IMRT.IN_DS2_en_csv_v2-dictionary.html
    cleandata/
      clean_API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv
      clean_API_SP.DYN.IMRT.IN_DS2_en_csv_v2-dictionary.md
      clean_API_SP.DYN.IMRT.IN_DS2_en_csv_v2-dictionary.html
  code/
    csv-clean-script.R
  app/
    infmor_shiny.R
  images/
    tab1.png
    tab2.png
```

Preparation

First of all, the packages `readr` and `dplyr` was loaded in the script file using `library()`.

```
# install packages
library(readr)    # importing data
library(dplyr)    # data wrangling
```

Importing Data

Then, the data files were imported using `read_csv()` function.

```
# import the data API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv
rawindex <- read_csv('../data/rawdata/API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv', skip = 4)

# import the data Metadata_Country_API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv
metadata <- read_csv('../data/rawdata/Metadata_Country_API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv')
```

Note: since the first four rows of `API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv` was the title information, argument `skip` was used to exclude first four rows of the data file.

Merging Data Frames

After importing the data files, two data frames were merged using the function `inner_join()`. This enables the user to easily refer the metadata information without going back to the metadata file every time he or she needs certain information.

```
# merge two dataframes
index <- inner_join(rawindex, metadata, by = "CountryCode")
```

Manipulating Data

After merging two data frames, the functions `filter()` and `select()` were used to only select the data we need for analysis and reorder the column in a way that is easier for the user to visually examine the cleaned data frame.

```
# remove all rows before 1990
index <- filter(index, !Year < 1990)

# remove all rows with empty indicator
index <- filter(index, !Indicator %in% NA)

# reorder columns
index <- index %>% select(Region, CountryName, CountryCode, IncomeGroup, Year, Indicator)
```

Exporting Clean Data Frame

After the cleaning process is done, the function `write_csv()` was used to export the cleaned data frame to the `cleandata` folder.

```
# export the clean data frame of indices
write_csv(index, '../data/cleandata/clean_API_SP.DYN.IMRT.IN_DS2_en_csv_v2.csv')
```

Shiny App

To better carry out visual process, Shiny App was created using cleaned data.

Preparation

In preparation, required package was first loaded using the function `library()`

```
# required packages
library(shiny)
library(ggvis)
library(dplyr)
```

Then, the variables were converted as factors using the function `as.factor()` and given names

```
# Convert variables as factors
index$Region <- as.factor(index$Region)
index$IncomeGroup <- as.factor(index$IncomeGroup)

# Assign variable names
categorical_region <- c('North America', 'Latin America & Caribbean', 'Europe & Central Asia', 'East Asia & Pacifi
c', 'South Asia', 'Sub-Saharan Africa', 'Middle East & North Africa')
categorical_incomegroup <- c('Low income', 'Lower middle income', 'Upper middle income', 'High income')
```

UI

In the UI section of the Shiny App file, the title was given and the sidebar as well as main panel was defined. The user may embed list, slider bar, and radio button to reflect his or her choice to the output.

```
# Define UI for application that draws a histogram
ui <- fluidPage(

  # Application title
  titlePanel("Infant Mortality Rate"),

  # Sidebar with different widgets depending on the selected tab
  sidebarLayout(
    sidebarPanel(
      conditionalPanel(condition = "input.tabselected==1",
                       selectInput("var1", "Region", categorical_region,
                                   selected = "North America"),
                       sliderInput("opac", "Opacity",
                                   min = 0, max = 1, value = 0.5),
                       radioButtons("show1", "Show line",
                                    choices = list("none" = 1, "lm" = 2, "loess" = 3),
                                    selected = 1)),
      conditionalPanel(condition = "input.tabselected==2",
                       selectInput("var2", "Income Group", categorical_incomegroup,
                                   selected = "Low income"),
                       sliderInput("opac", "Opacity",
                                   min = 0, max = 1, value = 0.5),
                       radioButtons("show2", "Show line",
                                    choices = list("none" = 1, "lm" = 2, "loess" = 3),
                                    selected = 1))
    )
    ,
    mainPanel(
      tabsetPanel(type = "tabs",
                  tabPanel("Scatterplot", value = 1,
                           ggvisOutput("scatterplot1")),
                  tabPanel("Scatterplot", value = 2,
                           ggvisOutput("scatterplot2")),
                  id = "tabselected")
    ))
)
```

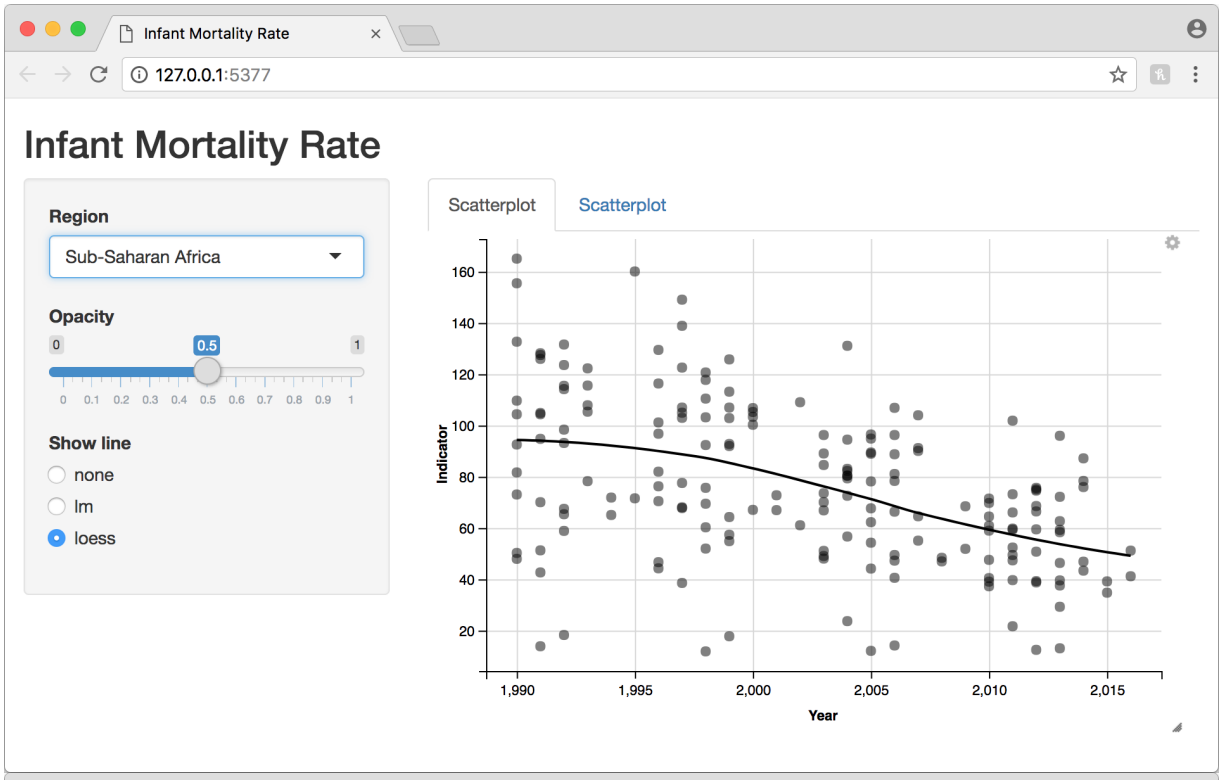
Server

In the server section of the Shiny App file, server logic was defined and the graph(scatterplot) was created. To take a look at the data in two different perspectives–region and income group–two conditional scatterplot was created. The user may create layers of condition to create the plot only with the data in certain conditions.

```
# Define server logic
server <- function(input, output) {

  # Scatterplot (for 1st tab)
  vis_scatterplot <- reactive({
    var1 <- prop("filter", input$var1)
    if (input$show1 == 1) {
      index %>%
      filter(index$Region == var1) %>%
      ggvis(x = ~Year, y = ~Indicator) %>%
      layer_points(fillOpacity := input$opac, fillOpacity.hover := 1)
      } else if (input$show1 == 2) {
        index %>%
        filter(index$Region == var1) %>%
        ggvis(x = ~Year, y = ~Indicator) %>%
        layer_points(fillOpacity := input$opac, fillOpacity.hover := 1) %>%
        layer_model_predictions(model = "lm")
        } else if (input$show1 == 3) {
          index %>%
          filter(index$Region == var1) %>%
          ggvis(x = ~Year, y = ~Indicator) %>%
          layer_points(fillOpacity := input$opac, fillOpacity.hover := 1) %>%
          layer_model_predictions(model = "loess")
      }
      })
  vis_scatterplot %>% bind_shiny("scatterplot1")

  # Scatterplot (for 2nd tab)
  vis_scatterplot <- reactive({
    var2 <- prop("filter", input$var2)
    if (input$show2 == 1) {
      index %>%
        filter(index$IncomeGroup == var2) %>%
        ggvis(x = ~Year, y = ~Indicator) %>%
        layer_points(fillOpacity := input$opac, fillOpacity.hover := 1)
    } else if (input$show2 == 2) {
      index %>%
        filter(index$IncomeGroup == var2) %>%
        ggvis(x = ~Year, y = ~Indicator) %>%
        layer_points(fillOpacity := input$opac, fillOpacity.hover := 1) %>%
        layer_model_predictions(model = "lm")
    } else if (input$show2 == 3) {
      index %>%
        filter(index$IncomeGroup == var2) %>%
        ggvis(x = ~Year, y = ~Indicator) %>%
        layer_points(fillOpacity := input$opac, fillOpacity.hover := 1) %>%
        layer_model_predictions(model = "loess")
    }
  })
  vis_scatterplot %>% bind_shiny("scatterplot2")

}
```

Output

The output of the Shiny App file with lines of code displayed above looks like this:



Note: the user may use the `lm` or `loess` button to look at the simple linear trend and curved trend to see how the trend is increasing or decreasing at certain point of time.

Take Home Message

As an intellectual living in a complicated world with immense amount of data, one must be able to efficiently and accurately process the data to point out important insights.

In 2017, social problems still exist worldwide, and infant mortality we looked at in this post is one of the most severe ones.

When given a data this large, advanced R techniques using add-ons like `dplyr` and `ggvis` can aid the user to effectively clean the data and extract insights that matters to tackle the problems based on data.

References

- https://data.worldbank.org/indicator/SP.DYN.IMRT.IN
- https://ggvis.rstudio.com/cookbook.html
- http://dplyr.tidyverse.org/reference/index.html
- https://blog.exploratory.io/filter-data-with-dplyr-76cf5f1a258e
- https://shiny.rstudio.com/reference/shiny/0.13.2/conditionalPanel.html
- https://shiny.rstudio.com/articles/dynamic-ui.html

- https://shiny.rstudio.com/articles/pool-dplyr.html
- https://shiny.rstudio.com/articles/plot-interaction-advanced.html