REPORT OF GGVIS PACKAGE

THI TRANG, LE (Student ID 616074)

16 August, 2021

Contents

```
##                          file words chars nonws words_chunk chars_chunk
## 1 SPL21_GGVIS_Le_Thi_Trang.Rmd  1702 12010  9766        1361        9158
##   nonws_chunk path
## 1        7426    .
```

1. What is GGVIS?

GGVIS is "an implementation of an interactive grammar of graphics, taking the best parts of 'ggplot2', combining them with the reactive framework of 'shiny' and drawing web graphics using 'vega' ".[1]

2. Components of a plot:

There are three main components of a plot:[2]

- Data
- Geometry: Geometry represents what you see on the plot: points, lines, histograms, bars… In another word, geometry answers the question of what type of plot can be drawn to understand the data.
- Aesthetic mapping: Aesthetic mapping is associated with a set of properties, such as size, color, shape, opacity… that control how a plot looks like. A property can be a constant value, a range of values, or mapped to other variables in the data.

Here is also three key components of GGVIS plots that have affected GGVIS syntax too.

3. Syntax:

The general syntax formula is:

- DATA %>% **GGVIS**() %>% LAYER 1() %>% LAYER 2() %>% … %>% LAYER N()
- Global and local aesthetic mapping: Global properties can be set by defining the blank of GGVIS function and local properties by defining the blank of each layer.

[1]https://cran.r-project.org/web/packages/ggvis/ggvis.pdf
[2]https://rafalab.github.io/dsbook/ggplot2.html, Chapter 7, part 7.1

To illustrate mentioned theories, the next part of this report will show some examples, using GGVIS package to visualize the data called "World Happiness Score in 2019".

4. Examples:

First of all, we need to install GGVIS package, load the package into the current session, and import the data by the following syntax:

```
install.packages("ggvis")
```

```
library(ggvis)
library(knitr)
data2 <- read.csv("data2.csv")
```

And then, `str(data2)` is used to understand the structure of the data

```
str(data2)
```

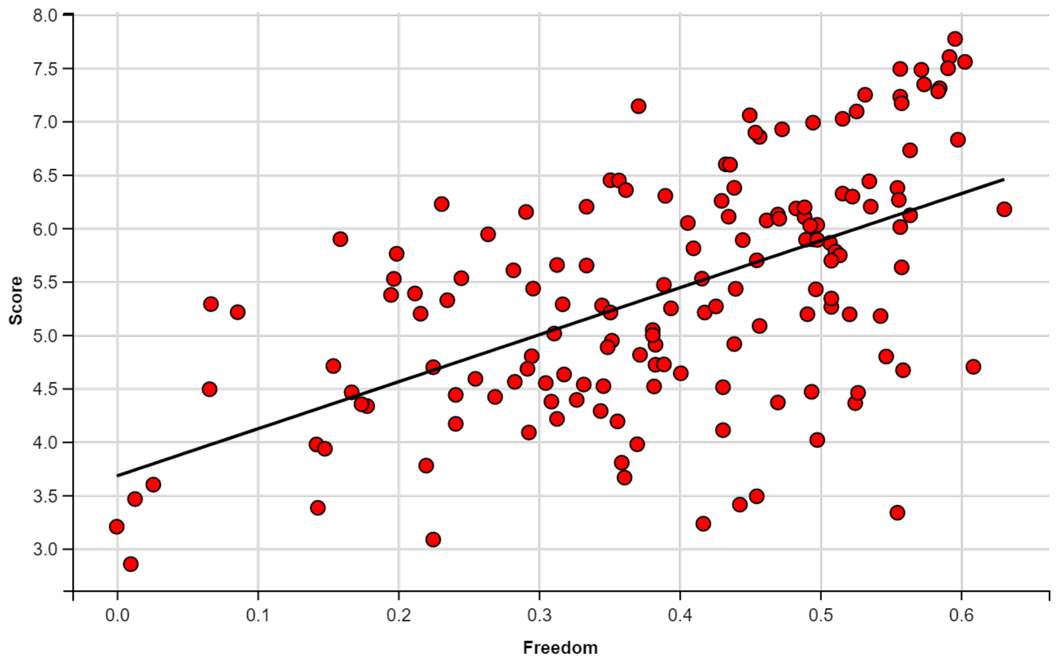
```
## 'data.frame':    156 obs. of  11 variables:
##  $ Overall_rank       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Country            : Factor w/ 156 levels "Afghanistan",..: 44 37 106 58 99 134 133 100 24 7 ...
##  $ Region             : Factor w/ 12 levels "","Australia and New Zealand",..: 12 12 12 12 12 12 12
##  $ Continent          : Factor w/ 6 levels "","Africa","America",..: 6 6 6 6 6 6 6 5 3 6 ...
##  $ Score              : num  7.77 7.6 7.55 7.49 7.49 ...
##  $ average_GDP        : num  1.34 1.38 1.49 1.38 1.4 ...
##  $ Social_support     : num  1.59 1.57 1.58 1.62 1.52 ...
##  $ Longevity          : Factor w/ 120 levels "0","0.105","0.168",..: 120 103 109 108 104 115 105 108
##  $ Freedom            : num  0.596 0.592 0.603 0.591 0.557 0.572 0.574 0.585 0.584 0.532 ...
##  $ Generosity         : num  0.153 0.252 0.271 0.354 0.322 0.263 0.267 0.33 0.285 0.244 ...
##  $ Trust_in_Government: num  0.393 0.41 0.341 0.118 0.298 0.343 0.373 0.38 0.308 0.226 ...
```

This data includes 156 observations and 11 variables. The main purpose of this data is to score the level of happiness of different countries in different regions and continents in the world in 2019. To score the level of happiness, this data considers 6 factors, they are: average_GDP, Social_support (the level of support from family), Longevity, Freedom (the level of freedom to make life choices), Generosity, Trust_in_Government.

Example 1:

GGVIS allows adding more layers on the same plot by using the symbol %>%

```
data2 %>%
    ggvis(~Freedom, ~Score) %>%
    layer_points(`:=`(fill, "red"), `:=`(stroke, "black"), `:=`(size, 60)) %>%
    layer_model_predictions(model = "lm")
```



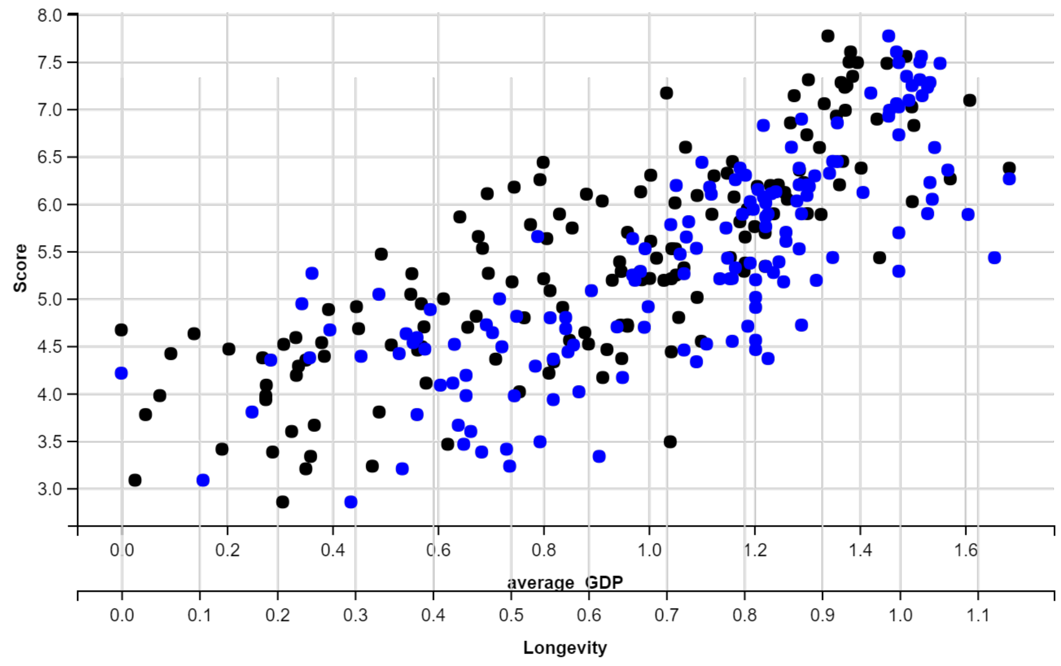
- As explained earlier, a plot contains three components: data, geometry, and aesthetic mapping. The data in this plot is the World Happiness Score in the year 2019; the geometries are points and regression line. For aesthetic mapping, the x-axis represents the variable "Freedom" and the y-axis represents the variable "Score".
- About syntax: To draw scatter plot and line regression, `layer_points` and `layer_model_predictions` are used. The arguments `"~Freedom, ~Score"` are put in GGVIS function, then they would apply to both layers. Meanwhile, the arguments that are put in each layer only control locally the properties of that layer. Specifically, `fill := "red", stroke := "black"` are to set the color and the color border of the points respectively; `model = "lm"` is to set linear model to the regression line. In this example, each property of each layer receives a constant value.

Example 2:

GGVIS can create two identical geometries in the same plot.

Here are 2 scatter plots in one plot, one demonstrates average_GDP and Score, other Freedom and Score. As the measures for average_GDP and Freedom are different, a dual-axis chart need to be built by overriding the default name of a scale to have more scales than the default

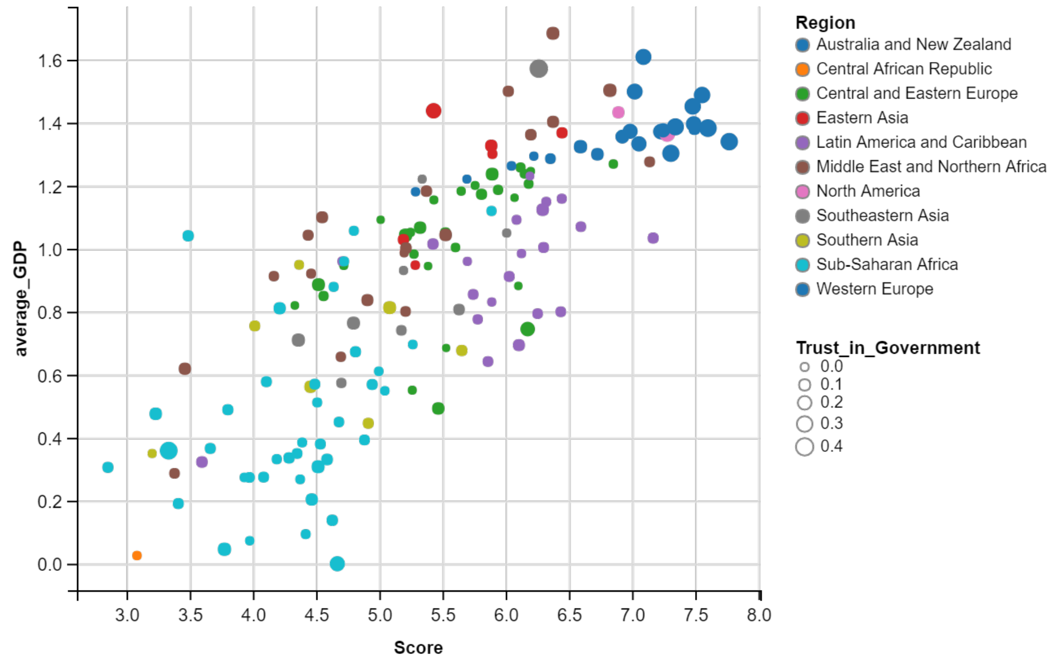
```
data2 %>%
    ggvis(y = ~Score) %>%
    layer_points(prop("x", ~average_GDP, scale = "xdisp")) %>%
    layer_points(prop("x", ~Longevity, scale = "xwt"), `:=`(fill, "blue")) %>%
    add_axis("x", "xdisp", orient = "bottom") %>%
    add_axis("x", "xwt", orient = "bottom", offset = 40)
```



Example 3:

GGVIS can create a scatter plot

```
data2 %>%
    ggvis(~Score, ~average_GDP) %>%
    layer_points(fill = ~Region, size = ~Trust_in_Government) %>%
    add_legend("fill") %>%
    add_legend("size", properties = legend_props(legend = list(y = 200)))
```



- Here, these properties are mapped to other variables in the data. Specifically, the color of the points is mapped to Region and the size of the points is mapped to Trust_in_Government.

In the above examples, GGVIS can draw static graphs as GGPLOT2 does. This package is also used to create interactive graphs.

5. Interactive graphs:

Why do we need interactive plots?

- We can add more options for property inputs and observe how the plot interacts with the change of property input.
- Using interactive graphs to visualize data is also a colorblind-friendly method.

As shown in the definition, GGVIS consists of GGPLOT2, Shiny, and Vega.

What is Shiny?

Shiny is an R package that makes it easy to build interactive web apps straight from R. Because of this component, GGVIS can create interactive graphs.

What is Vega?

- Vega is a language to describe the visual appearance and interactive behavior of visualization and then generate web-based views.[3]
- Because of the combination between Vega and Shiny, the interactive graphs created by GGVIS package work smoothly on web browse, as being shown in this link: link to Shinyapp.io

Notes:

- To escape with interactive plots that are currently run in R console, press the stop button in Rstudio, or close the browser window and then press Escape or Ctrl + C in R.
- Every interactive GGVIS plot must be connected to a running R session. As a result, it is impossible to publish a PDF report for interactive plots. As you still can run the code in R session to see how interactive plots work, I will present the result in a PDF file by inserting the pictures of the interactive plots. These pictures were taken by screenshotting instead of downloading the PNG format of interactive graphs. I did that because PNG-format pictures fail to show the interactive parts, for example, sliders, drop-down lists.
- The report containing interactive graphics produced by GGVIS can be created in web browsers. This task can be done with R markdown. As Shiny was also integrated into R markdown, it can create an HTML document with interactive components. Here is the interactive report was done by that: link to Shinyapp.io

How to make an interactive graph?

As well as mapping visual properties to variables or setting them to specific values, properties can be also connected to interactive controls to create interactive graphs. In this report, five common ways to create interactive graphs will be introduced.

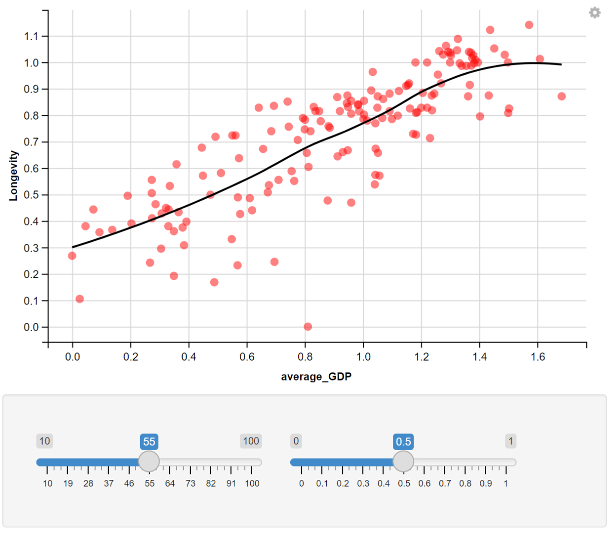
1. **Create a slider:**

- A slider contains a range value for the property, depending on the position of the cursor on the slider.
- In this example, to create a slider for the size of points, the syntax `"size = 60"` (as in the code in Example 1) was replaced by the new syntax `"size := input_slider(10, 100, label = "Size")"`.
- By doing the same with other properties, many sliders are created for one graph.

[3](https://vega.github.io/vega/)

```
data2 %>%
    ggvis(~average_GDP, ~Longevity) %>%
    layer_points(`:=`(fill, "red"), `:=`(size, input_slider(10, 100, label = "Size")),
        `:=`(opacity, input_slider(0, 1, label = "Opacity"))) %>%
    layer_smooths()
```

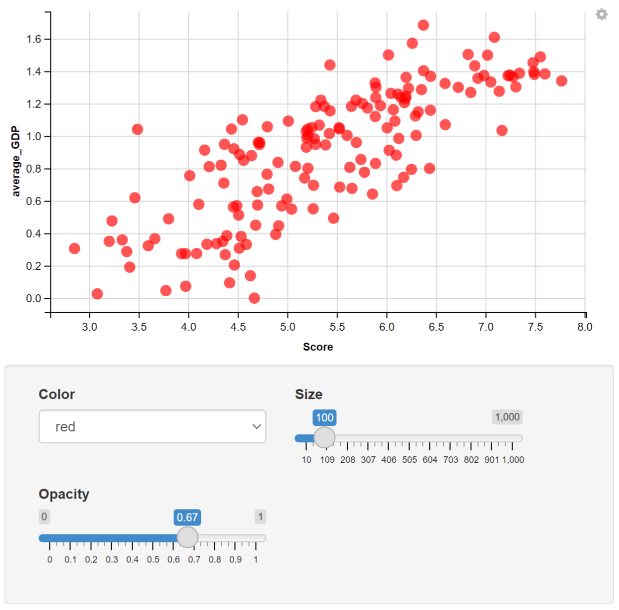
```
## Error in storage.mode(y) <- "double": invalid to change the storage mode of a factor
```



2. **Create a drop-down list:**

   In this example, a drop-down list was created to choose the color of the point. To do that, in layer_points, instead of specifying the color of the point by the syntax `"fill = "red""` as previous example, I coded `"input_select(c("red", "blue", "purple"), label = "Color")"`.

```
data2 %>%
    ggvis(~Score, ~average_GDP, `:=`(fill, input_select(c("red", "blue", "purple"),
        label = "Color")), `:=`(size, input_slider(10, 1000, 100, label = "Size"))) %>%
    layer_points()
```



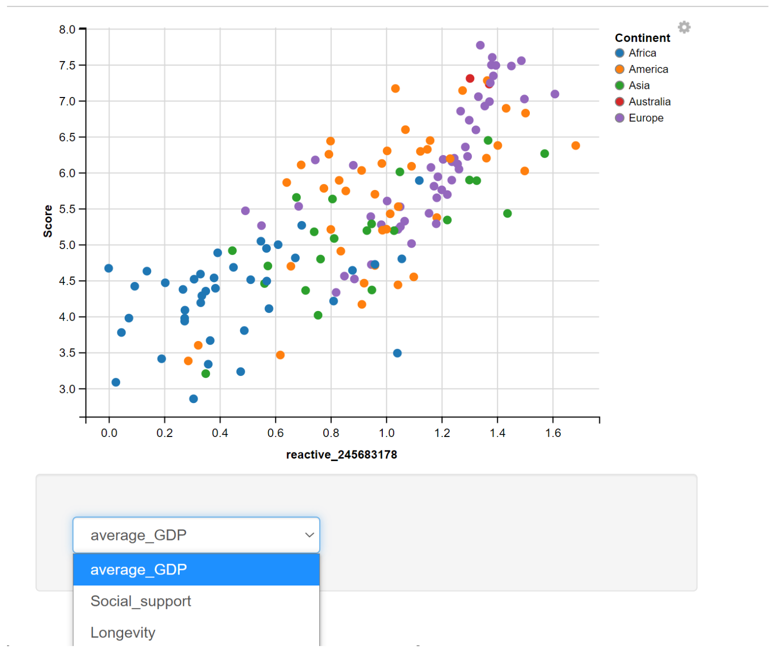
Color
red

Size
100

Opacity
0.67

A drop-down list is also applied to change the variable in the x-axis as following:

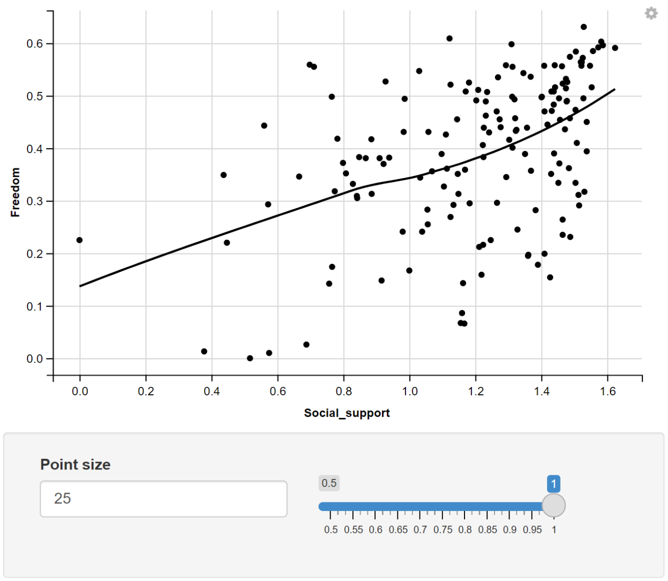
```
data2 %>%
    ggvis(x = input_select(c("average_GDP", "Social_support", "Longevity", "Freedom",
        "Generosity"), map = as.name)) %>%
    layer_points(y = ~Score, fill = ~Continent)
```



3. **Input numeric:**

   Here we want to input numeric value for the size of the points. The syntax is "`size := input_numeric(value = 25, label = "Point size"))`", `value = 25` means that the default size is 25

```r
data2 %>%
    ggvis(x = ~Social_support, y = ~Freedom) %>%
    layer_points(`:=`(size, input_numeric(value = 25, label = "Point size"))) %>%
    layer_smooths(span = input_slider(0.5, 1, value = 1))
```



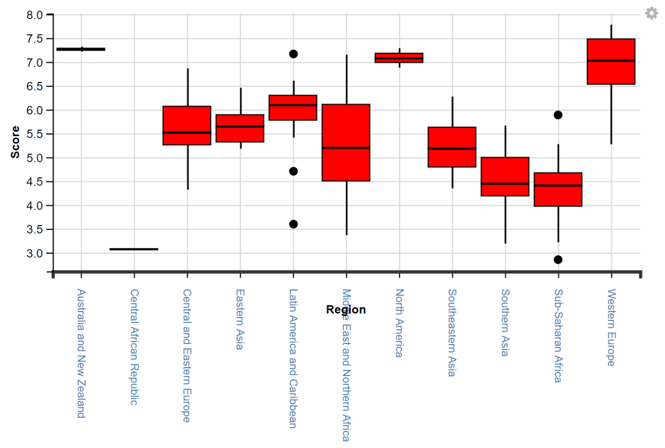
Point size

25

4. Input text:

The function to input text is ‘`input_text, value = "red"` means that red is the default color.

```r
fill_text <- input_text(label = "Point color", value = "red")
data2 %>%
    ggvis(~Region, ~Score, `:=`(fill, fill_text)) %>%
    layer_boxplots() %>%
    add_axis("x", properties = axis_props(labels = list(fill = "steelblue", angle = 90,
        fontSize = 10, align = "left", baseline = "middle", dx = 7), axis = list(stroke = "#333",
        strokeWidth = 3)))
```



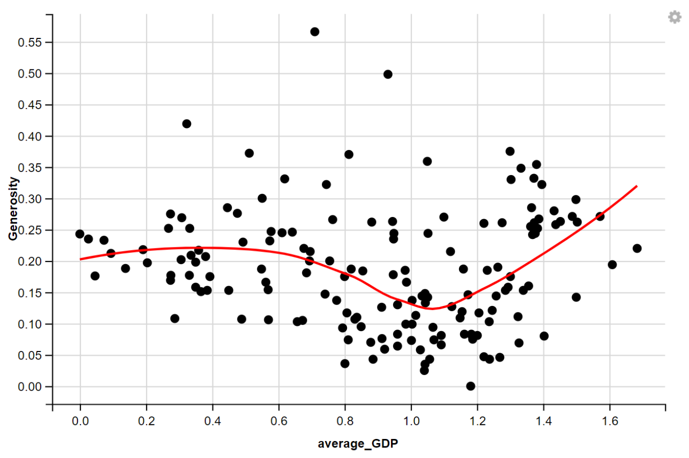
Point color

red

5. **Choose one button:**

The function to create buttons is `input_radiobuttons`

```
data2 %>%
    ggvis(x = ~average_GDP, y = ~Generosity) %>%
    layer_points() %>%
    layer_model_predictions(model = input_radiobuttons(c(LOESS = "loess", Linear = "lm"),
        label = "Model type"), `:=`(stroke, input_radiobuttons(c(Red = "red", Green = "Green"),
        label = "Line color")))
```



Model type

(● LOESS) ○ Linear

Line color

(● Red) ○ Green

In general, to create interactive graphs, we can use these functions: `input_numeric`, `input_text`, `input_select`, `input_radiobuttons`. The argument for them follows the same format: `the (default) value` and its `label`.

6. Conclusion:

GGVIS is a great package that enables interactive graphics and publishes on a web browser. However, comparing to ggplot2, many of the features in GGVIS is still unavailable (for example: facetting). But in the future, GGVIS has been continuously developed and would be a powerful package for data visualization.

7. References

- Winston Chang, Hadley Wickham, Interactive Grammar of Graphics, December 4, 2020, at https://cran.r-project.org/web/packages/ggvis/ggvis.pdf
- Rafael A. Irizarry, Introduction to Data Science - Data Analysis and Prediction Algorithms with R, July 03, 2021, at https://rafalab.github.io/dsbook/
- Data source: https://www.kaggle.com/unsdsn/world-happiness