My Shiny App Documentation

Jin-Hoon An

Apr. 26, 2015

```
## Loading required package: knitr
```

Interactive Visualizations with rCharts and Shiny

In this project, I will take you through the details of creating an interactive visualization using `rCharts` and `Shiny` packages.

- First Steps

My first objective is to recreate the charts here without any menu controls. Once we are able to achieve that, we can add the menu controls easily using Shiny and convert it into an interactive app. This blog post will attemp to show you both how to use rCharts to create such visualizations, as well as how to convert them into an interactive app using Shiny.

— .class #id

Data

The data was collected by the International Labor Organization. I used a version of the dataset put together by the excellent data visualization blog: thewhyaxis.

```
         country countrycode year id gender value
1           OECD        OECD 1995  1    Men   9.4
3        Austria         AUT 1995  3    Men   9.5
4        Belgium         BEL 1995  4    Men  12.2
5         Canada         CAN 1995  5    Men  13.2
7 Czech Republic         CZE 1995  7    Men   8.0
8        Denmark         DNK 1995  8    Men   9.6
```

- Creating an interactive visualization

```
require(devtools)
install_github('ramnathv/rCharts')
```

— .class #id

Bar Plot Process

Now recreate the bar plot shown in the visualization.

```r
require(rCharts)
YEAR = 2011
# Step 1. create subsets of data by gender
men   <- subset(dat2m, gender == "Men" & year == YEAR)
women <- subset(dat2m, gender == "Women" & year == YEAR)

# Step 2. initialize bar plot for value by countrycode for women
p1 <- rPlot(x = list(var = "countrycode", sort = "value"), y = "value",
  color = 'gender', data = women, type = 'bar')

# Step 3. add a second layer for men, displayed as points
p1$layer(x = "countrycode", y = "value", color = 'gender',
  data = men, type = 'point', size = list(const = 3))

# Step 4. format the x and y axis labels
p1$guides(x = list(title = "", ticks = unique(men$countrycode)))
p1$guides(y = list(title = "", max = 18))

# Step 5. set the width and height of the plot and attach it to the dom
p1$addParams(width = 600, height = 300, dom = 'chart1',
  title = "Percentage of Employed who are Senior Managers")

# Step 6. print the chart (just type p1 if you are using it in your R console)
p1$print()
```

— .class #id

Bar Plot Description

The first step is to create subsets of the data by gender for a specific year. We then proceed to initialize a bar plot for `women`. The function `rPlot` uses an interface very similar to the `lattice` package. The first argument specifies that the x variable is going to be `countrycode` and we want it sorted by `value`. The remaining arguments specify different aesthetics of the bar plot.

The next step adds a second layer to this plot using the data for `men`, specifying that the values be displayed as `points`. The code in **Step 3** might seem a little strange to some of you, since there is no explicit assignment involved. The reason for this is that `rCharts` uses Reference Classes, which allow object oriented programming in R, leading to more concise code. In essence, `layer2` is a method of the object `p1` that adds a layer to the plot object.

The last few steps tweak the axis labels, width and height of the plot, before attaching it to a specific DOM element. At this point, you can run Steps 1 through 5 and type `p1` in your R console, and if everything goes right, you will be staring at the same plot created here.

— .class #id

Line Chart

We can now add a line chart for comparing the values for a specific country across years. We follow the same approach outlined above, except that we only need a single layer in this case.

```
COUNTRY = "Korea"
country = subset(dat2m, country == COUNTRY)
p2 <- rPlot(value ~ year, color = 'gender', type = 'line', data = country)
p2$addParams(width = 900, height = 800, dom = 'chart2')
p2$guides(y = list(min = 0, title = ""))
p2$print()
```

— .class #id

User Interface (ui.R)

Let us design the user interface first. Note that we need four components, two select boxes to allow the user to select the year and country, and two `div` tags to place the dynamically generated chart output.

```
require(rCharts)
options(RCHART_LIB = 'polycharts')
shinyUI(pageWithSidebar(
  headerPanel("% of Employed who are Senior Managers, by Sex"),

  sidebarPanel(
    selectInput(inputId = "year",
      label = "Select year to compare countries",
      choices = sort(unique(dat2m$year)),
      selected = 2011),
    selectInput(inputId = "country",
      label = "Select country to compare years",
      choices = sort(unique(as.character(dat2m$country))),
      selected = "Korea")
  ),

  mainPanel(
    showOutput("chart1", "polycharts"),
    showOutput("chart2", "polycharts")
  )
))
```

Although the above code is self explanatory, I want to draw your attention to a couple of things. First, we choose a layout where the controls are placed in a sidebar and the charts in the main panel. Second, we generate the choices for the two selection boxes using the unique values for `year` and `country` in the dataset. Third, `showOutput` is a function in `rCharts` that creates placeholders for the generated charts, with a given id.

— .class #id

Chart Output (server.R)

Now that we nailed the UI, we need to tweak the plotting code to interface with the UI. To achieve this, we need to do three things. First, we need to replace COUNTRY and YEAR by `input$country` and `input$year` so that their values are dynamically obtained from user input. Second, we need to wrap the plotting calls inside `renderChart`, which is a function in `rCharts` that adds allows the plots to react to user inputs. Third, we need to assign the output of these calls to their respective elements in the UI (chart1 and chart2).

```r
require(rCharts)
options(RCHART_WIDTH = 800)
shinyServer(function(input, output) {
  output$chart1 <- renderChart({
    YEAR = input$year
    men <- subset(dat2m, gender == "Men" & year == YEAR)
    women <- subset(dat2m, gender == "Women" & year == YEAR)
    p1 <- rPlot(x = list(var = "countrycode", sort = "value"), y = "value",
      color = 'gender', data = women, type = 'bar')
    p1$layer(x = "countrycode", y = "value", color = 'gender',
      data = men, type = 'point', size = list(const = 3))
    p1$addParams(height = 300, dom = 'chart1',
      title = "Percentage of Employed who are Senior Managers")
    p1$guides(x = list(title = "", ticks = unique(men$countrycode)))
    p1$guides(y = list(title = "", max = 18))
    return(p1)
  })
  output$chart2 <- renderChart({
    COUNTRY = input$country
    country = subset(dat2m, country == COUNTRY)
    p2 <- rPlot(value ~ year, color = 'gender', type = 'line', data = country)
    p2$guides(y = list(min = 0, title = ""))
    p2$guides(y = list(title = ""))
    p2$addParams(height = 300, dom = 'chart2')
    return(p2)
  })
})
```

Finally, we place the code required to read and process the data in `global.R` so that the data is accessible to both `ui.R` and `server.R`. You can see the resulting Shiny app here and the source code