Shiny + Quandl + World Map Write Up

Today we are going to wrap our previous Quandl/world map Notebook into an interactive Shiny app that lets users choose both a country and a dataset for display. As usual, we did a lot of the heavy lifting in the Notebook to make our work more reproducible and our app more performant. The final app is available here.

Devotees of this Reproducible Finance blog series will note similarities to this Shiny app, but today's app will have a different and richer functionality set.

First, we are going to be pulling in data from Quandl, so we won't be using ticker symbols, but rather will be using country codes plus data set codes. That would also allow us to open this app up to the vast number of data sets available via Quandl.

Second, the user will be able to use the sidebar to select different datasets, rather than being restricted to a country ETF. For example, the user might want to select GDP per capita, or the exchange rate, or any data set that we wish to make available. There's a tradeoff here between focus and

Third, we will display a chart using `highcharter` instead of `dygraphs`. There's not a substantive reason to be doing this beyond that it's a good chance to start exploring highcharter, which is a popular tool in the financial world.

Lastly, we'll port raw data to a `data.table` and include a few buttons for easy download, in case our end users want to reproduce our charts or import data for their own work. We are going to use the `as_tibble` function from the fantastically useful new tidyquant package to facilitate our `data.table`, but consider that a teaser to a future post that explores the package in greater depth. In fact, we won't have space to cover the data download section in the post, but all the code is available in the live app.

Let's get to it.

The first line of substance in this app is to load in the `leaflet` map we constructed in our Notebook. We named the object `leaf_world` and saved it in a file called `wdiMapData.RDat`. If you want to refresh on how we did that, have a look back at the previous post. Apologies if that sounds tedious but hopefully it emphasizes the workflow of doing the heavy map building in the Notebook when possible.

To load that map, we run the following:

```
# Load the .RDat file where the leaflet map is saved.
load('wdiMapData.RDat')
```

Now we can access whatever R objects were saved in that file and, in this case, that means one R object called `leaf_world`. That might not be a great name, but it's the leaflet map of the world we built in the Notebook. When we are ready to build the map in our Shiny app, we'll simply render that object.

Before we get to the map, though, let's construct a sidebar where the user can choose which dataset to display. We are going to be working with a macroeconomic data source from the World Bank called World Development Indicators WDI. The Quandl code for WDI is `WWDI` and thus we'll append `WWDI/` to each data set call - but note that this will not appear in the sidebar because we won't actually pass any data to Quandl there. We have to wait for the user to click on a county and thus will handle the data passing in a future code chunk. This sidebar has one purpose, for the user to select the code for the economic indicator.

We want to give the user a drop down of choices, but we don't want the choices to be the Quandl codes. We want the choices to be the familiar name of the data set. In other words, the user will see as a choice 'GDP Per Capita', instead of the Quandl code `_NY_GDP_PCAP_KN`. For that reason, we'll first create an object called `dataChoices` that holds our name-value pairs. The name is the familiar title of the time series - for example, 'GDP Per Capita' - and the value is the Quandl code.

```
dataChoices <- c("GDP Per Capita" = "_NY_GDP_PCAP_KN",
                 "GDP Per Capita Growth" = "_NY_GDP_PCAP_KD_ZG",
```

```
                    "Real Interest Rate" = "_FR_INR_RINR",
                    "Exchange Rate" = "_PX_REX_REER",
                    "CPI" = "_FP_CPI_TOTL_ZG",
                    "Labor Force Part. Rate" = "_SL_TLF_ACTI_ZS")
```

Next, in our `selectInput` statement, we will set choices equal to `dataChoices` and this will allow the user to see intuitive names but choose the Quandl codes.

```
selectInput("indicatorSelect",
            "Choose an economic indicator",
            choices = dataChoices,
            selected = "GDP Per Capita")
```

Our simple sidebar is finished and the end result is that when a user makes a choice, we have an input reactive with the value of the data set code. Let's put that code aside for a minute and turn our attention to the map. That might be a bit of a confusing workflow but it's the order in which the user will experience our app: first choose an indicator, then click the map.

Remember, we did the map-building work on this in our Notebook, then loaded the object in the `setup` code chunk. That leaves us with a simple call to `renderLeaflet()` to pass it `leaf_world`.

```
leafletOutput("map1")

output$map1 <- renderLeaflet({
  leaf_world
  })
```

Alright, that wasn't too cumbersome and perhaps the simplicity of that step makes the hard work in the previous Notebook a bit more tolerable.

And now the fun part, wherein we build the machinery to let a user click the map and grab data from Quandl. We will proceed in four steps:

- capture the country code of the country that gets clicked
- call on the economic time series that the user selected in the sidebar
- paste the country code and the time series together to create one dataset code
- pass that one dataset code to Quandl and import the time series

On to step 1, capturing the clicked country.

```
clickedCountry <- eventReactive(input$map1_shape_click, {
  return(input$map1_shape_click$id)
})
```

It's same process as in this post, but there is one difference. Let's review what happened here.

Recall that we set `layerID = ~iso_a3` when we built the map in our Notebook. That's because the Quandl code appends the iso_a3 country codes to the data set code. In other words, if we want GDP-per-capita for Brazil, we need the Brazil country code 'BRA' (so we can later append it to the GDP-per-capita code '_NY_GDP_PCAP_KN').

In the code chunk above, we used an `observeEvent` function to capture the `layerID` of whatever shape a user clicks, and in the Notebook, we had set that ID to be the `~isa_a3` code. When a user clicks Brazil, our reactive captures the country code 'BRA'.

Next, we need to append that country code to the data set code and recall that the user chose this code in our side bar. We just need to grab his selection via another `reactive`.

```
indicatorValue <- reactive({input$indicatorSelect})
```

Let's pause and take inventory of what we have captured thus far: the country code from the click on the map, and the data set code from the sidebar. Now we want to paste those together and pass them to Quandl. We'll do that via another `reactive`.

In the code chunk below, notice that when we paste the codes together, we start with 'WWDI/'. That's because all of our data comes from the 'WWDI' data source on Quandl. We are not letting the user choose a different data source. (We could have done so in the sidebar, but consider how that would have complicated our data set code `reactive`).

```
countryData <- reactive({

  # WWDI is the World Bank data set.
  # We aren't giving the user a choice of data sources, but we could.
  # Think about how that would change the sidebar choices in our selectInput.

  dataSet <- paste("WWDI/",
                   # The country that was clicked.
                   as.character(clickedCountry()),
                   # The time series that was chosen in the sidebar.
                   as.character(indicatorValue()),
                   sep = "")
  # Now pass that pasted data set code object to Quandl.
  Quandl(dataSet, type = "xts")
})
```

The hard part is done now.

```
output$highchart <- renderHighchart({

  validate(need(input$map1_shape_click$id != "",
                "Please click on a country to display its ETF price history."))
  # Make a nice title for the chart
  indicatorName1 <- names(dataChoices[dataChoices == input$indicatorSelect])
  countryName1 <- countrycode(as.character(clickedCountry()), "iso3c", "country.name")
  title <- paste(countryName1, indicatorName1, sep = " ")
  highchart(type = "stock") %>%
    hc_title(text = title) %>%
    hc_add_series(countryData(), name = paste(countryName1, indicatorName1, sep = " ")) %>%
    # I don't like the look of the navigator/scrollbar, but you might.
    # Change these to enabled = TRUE and check out the results.
    hc_navigator(enabled = FALSE) %>%
    hc_scrollbar(enabled = FALSE)
})
```