Introduction to Data Products

Fabrice Niessen

2014-06-11

Contents

1. Week 1

Sorts of data products:

- Slidify lecture
- R Studio presenter
- Shiny App (little Web App using R)
- R package
- R charts
- interactive graphics

1.1 Shiny

Platform for creating **interactive R programs** (for example, a prediction algorithm; users can input the relevant predictors and obtain their prediction) embedded into a **Web page**.

```
library(shiny)
```

A basic Shiny project requires:

- a `server.R` file which controls what it does (render print value, plot output, etc.)

  Don't put system calls in your code: security concerns.
- a `ui.R` file which controls how it looks (or a `www/index.html` file with the specific HTML – in the same directory as `server.R`)

To run it, type `runApp()` in the directory that contains the code.

Things that Shiny can do:

- Have tabbed main panels
- Have editable data tables
- Have a dynamic UI
- User defined inputs and outputs
- Put a Submit button (non-reactive reactivity!) if the calculations take long.

It's generally good form to reshow the person the value they entered.

Distributing:

- Put the App on GitHub and the users can call runApp, or
- Create an R package, or

- Run a Shiny server (on Amazon AWS).

Details:

- Code put before `shinyServer` (in `server.R`) gets called once when you do `runApp()`.
- Code inside the unnamed function of `shinyServer` but not in a **reactive** statement will run once for every new user (or page refresh)
- Code in reactive functions (for example, `render*`) of `shinyServer` gets run repeatedly **as needed** when new values are entered.

  Shiny only executes the **component of the code** that's needed: the biggest **mistake** you can make in a Shiny app is to think of the Shiny function as something where the code just runs through it serially. These **reactive text statements** are **reacting to the widget input**.

Debugging techniques:

- **Important**: try `runApp(display.mode='showcase')` for seeing what Shiny is actually executing at the time it is.
- Use `cat` in your code to display output to `stdout` (so R console)

1.2 Manipulate

`manipulate` is an awesome function to create a **quick interactive graphic** (**if the intended users also use RStudio**): way, way better to do it this way than to create a Shiny app or something more complicated.

```
library(manipulate)
manipulate(plot(1:x), x = slider(1, 100)) ## x = placeholder
```

You can create:

- a slider,
- a checkbox,
- a drop-down,

and have more than one.

```
myPlot <- function(s) {
    plot(cars$dist - mean(cars$dist), cars$speed - mean(cars$speed))
    abline(0, s)
}
```

This function plots distance versus speed, each de-meaned and an associated line of slope s. The following code will make a manipulate plot that creates a slider for the slope:

```
manipulate(myPlot(s), s = slider(0, 2, step = 0.1))
```

1.3 rCharts

Way to create **interactive JavaScript visualizations** using R.

```
library(rCharts)
```

1.3.1 nPlot

The basic function for a specific type of `rCharts` is `nPlot` (which calls a specific JavaScript plotting library, `nvD3`).

```
n1 <- nPlot(...)                          ## n1 contains the plot
n1$save('fig/n1.html', cdn = TRUE)
cat('<iframe src="fig/n1.html" width=100%, height=600></iframe>') # embed it back into Slid
```

Interactivity:

- If you hover over a bar, it gives you the frequency
- You can even switch between grouped and stacked bar plots (nice animation when it does it)

1.3.2 rPlot

```
## Facetted barplot
r1 <- rPlot(...)
r1$print("chart1")                        ## print out the JS
r1$save('fig/r1.html', cdn = TRUE)        ## save as HTML file
cat('<iframe src="fig/r1.html" width=100%, height=600></iframe>') # bring it back into Slid
```

1.3.3 Publish an rChart

```
r1$publish('myPlot', host = 'gist')       ## save to Gist, rjson required
r1$publish('myPlot', host = 'rpubs')      ## save to rpubs
```

1.3.4 Morris (time series plot)

`mPlot`.

1.3.5 xCharts

1.3.6 Leaflet (mapping library)

Create a geographical map with pop-ups set at specific longitude and latitude locations.

1.3.7 Rickshaw

1.3.8 Highchart

(Quite confusing)

1.4 googleVis

Things you can do with googleVis:

- Interactive data visualizations made **available in websites** (with HTML code).
  - The R function creates an HTML page.
  - The HTML page calls Google Charts.
  - The result is an interactive (clickable) HTML graphic
- Dynamic visualizations can be built with Shiny, Rook, and R.rsp (and **shared with your friends**).
- **Embed them in interactive** R markdown based **documents**
  - Set `results="asis"` in the chunk options
  - Can be used with knitr and slidify.

Some types of charts:

- Motion charts: `gvisMotionChart`
- Interactive maps: `gvisGeoChart`
- Interactive tables: `gvisTable`
- Line charts: `gvisLineChart`
- Bar charts: `gvisColumnChart`
- Tree maps: `gvisTreeMap`

Combine multiple plots together (two at a time): `gvisMerge`.

```
demo(googleVis)                          ## for more info
```

2. Week 2

2.1 Presenting Data Analysis Writing a Data Report

Directory:

- `prompt.pdf`
- `data/` folder
- `code/` folder
  - `rawcode/` folder (for quick and dirty figures)
  - `finalcode/` folder (analysis that I will be sharing with other people)

    Structure of document:

    1. Preliminaries

2. Exploratory analysis
3. Modelling
4. Get the estimates and confidence intervals
5. Figure making

- `figures/` folder
- `writing/` folder

  Structure of the document:

  1. Introduction
  2. Methods
     - Data collection
     - Exploratory analysis
     - Statistical modelling
     - Reproducibility
  3. Results
  4. Conclusions
  5. References

2.2 Slidify

Presentation that you have to give once a month, and you want the presentation to be **updated** automatically with the new numbers. All you have to do is recompile the presentation (**rerun the code**).

Amalgamation of other technologies: knitr, Markdown, and several JavaScript libraries for **HTML5** (**slide decks**).

The JavaScript library **MathJax** correctly **renders** the **mathematical formulas**, and you type them out in LaTeX.

Slidify presentations are just HTML files, so you can view them with any Web browser and share them easily on GitHub, DropBox, or your own website.

2.2.1 Getting Slidify

```
install.packages("devtools")
library(devtools)
install_github("slidify", "ramnathv")
install_github("slidifyLibraries", "ramnathv")
library(slidify)
```

2.2.2 Getting started

```
setwd("~/sample/project")
author("first_deck")                    # create document (template with project name)
```

The `author` command does:

- Creating slide directory at first_deck...
- Copying files to first_deck...
- Finished creating slide directory...
- Switching to slide directory...
- Initializing Git Repo
- Initialized empty Git repository in *cygdrive/d/Users/fni/sample/project/first_deck*.git/
- [master (root-commit) 4c04da6] Initial Commit
- Checking out gh-pages branch...
- Switched to a new branch 'gh-pages'
- Adding .nojekyll to repo
- Opening slide deck...

2.2.3 Working it out

```
slidify("index.Rmd")                    # compile your Rmarkdown file
```

```
library(knitr)                          # for function browseURL
browseURL("index.html")                 # open the HTML file
```

2.2.4 Customizatiom

YAML-like code.

Other fields:

- `logo`
- `url` for `libs`, for `assets`
- (code) `highlighter`
- `hitheme`: `zenburn`

2.2.5 Publishing to GitHub

Create a new repository on GitHub, then:

```
publish_github(<user>, <repo>)
```

2.2.6 HTML5 Deck frameworks

- io2012

- html5slides
- deck.js
- dzslides
- landslide
- Slidy

Some of these HTML effects are actually pretty fancy. So, you should check 'em out.

2.2.7 HTML

You can include raw HTML it will get kept as HTML when it's slidified.

It is especially useful for more advanced stuff like images or tables where you need more control of the HTML options than Slidify actually gives you.

Similarly, you can incorporate JS or do anything else you can do in a Web page.

You can also **add interactive elements** to Slidify such as:

- quiz questions
- interactive R plots
- Shiny apps.

See examples: http://slidify.github.io/dcmeetup/demos/interactive/

2.3 RStudio Presenter

Authoring tool, a bit easier to learn than Slidify.

- Code is authored in a generalized markdown format
- Output is an HTML5 presentation
- Index file is .Rpres, which gets converted to an .md file and then to an HTML file if desired

Guide: http://www.rstudio.com/ide/docs/presentations/overview

- Single quote for inline code
- Triple quotes for block code

2.3.1 Getting started

New File > R presentation.

2.3.2 Customization

- Transitions.
- Hierarchical organization structure.

2.4 Slidify vs RStudio Presenter

- Slidify
  - Flexible control from the .Rmd file
  - Under rapid ongoing development
  - Larger user base
  - Lots and lots of styles and options
  - Steeper learning curve
  - More command-line oriented
- RStudio Presenter
  - Embedded in RStudio
  - More GUI oriented
  - **Very easy to get started**
  - Smaller set of easy styles and options
  - Default styles look very nice
  - Ultimately as flexible as Slidify with a little CSS and HTML knowledge