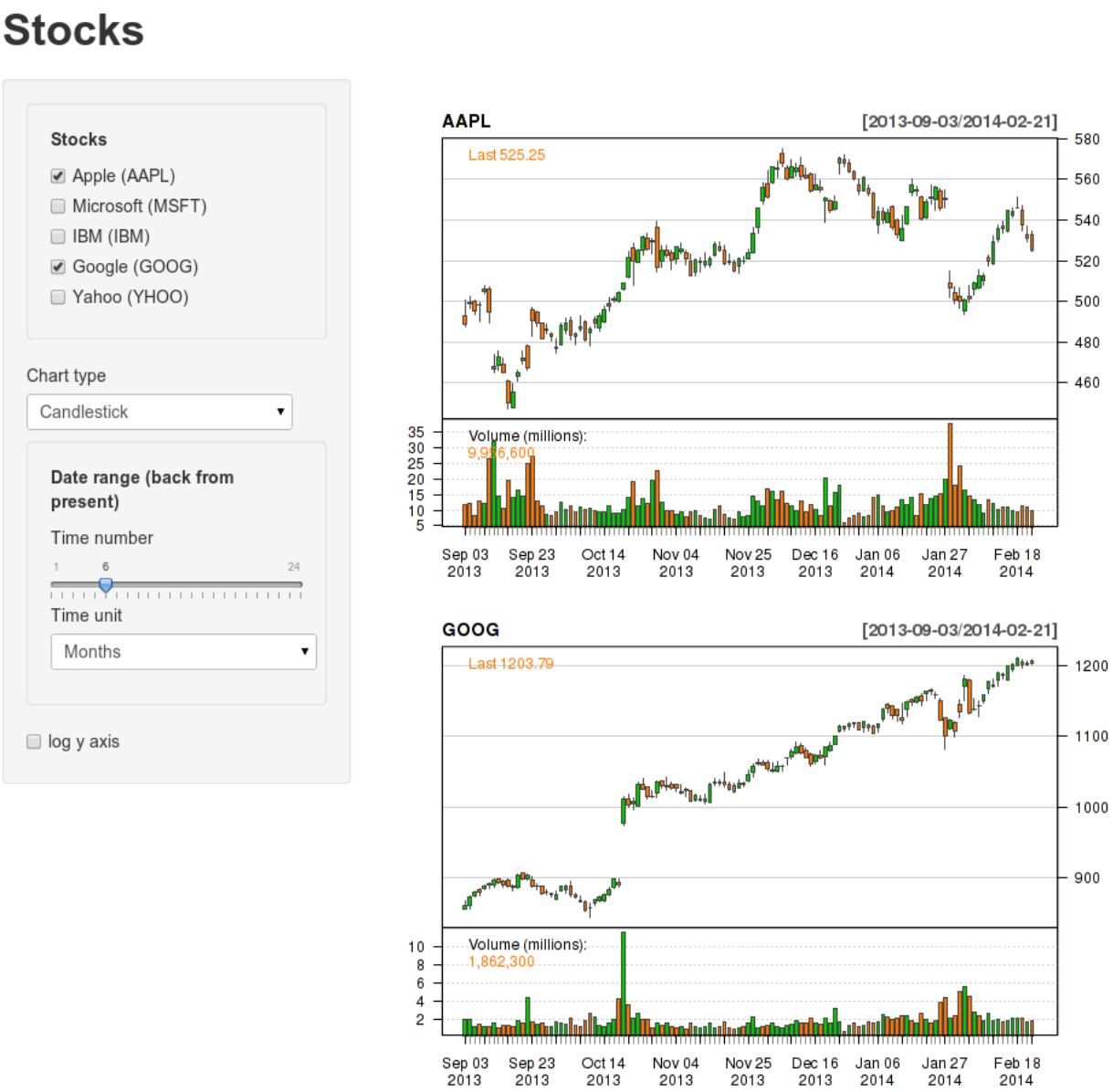
Shiny

Di Cook, Eric Hare

May 14, 2015

California Dreaming - ASA Travelling Workshop

It's So Shiny!



* Shiny** is an R package that allows you to easily create interactive web applets using R.

- Produced by RStudio

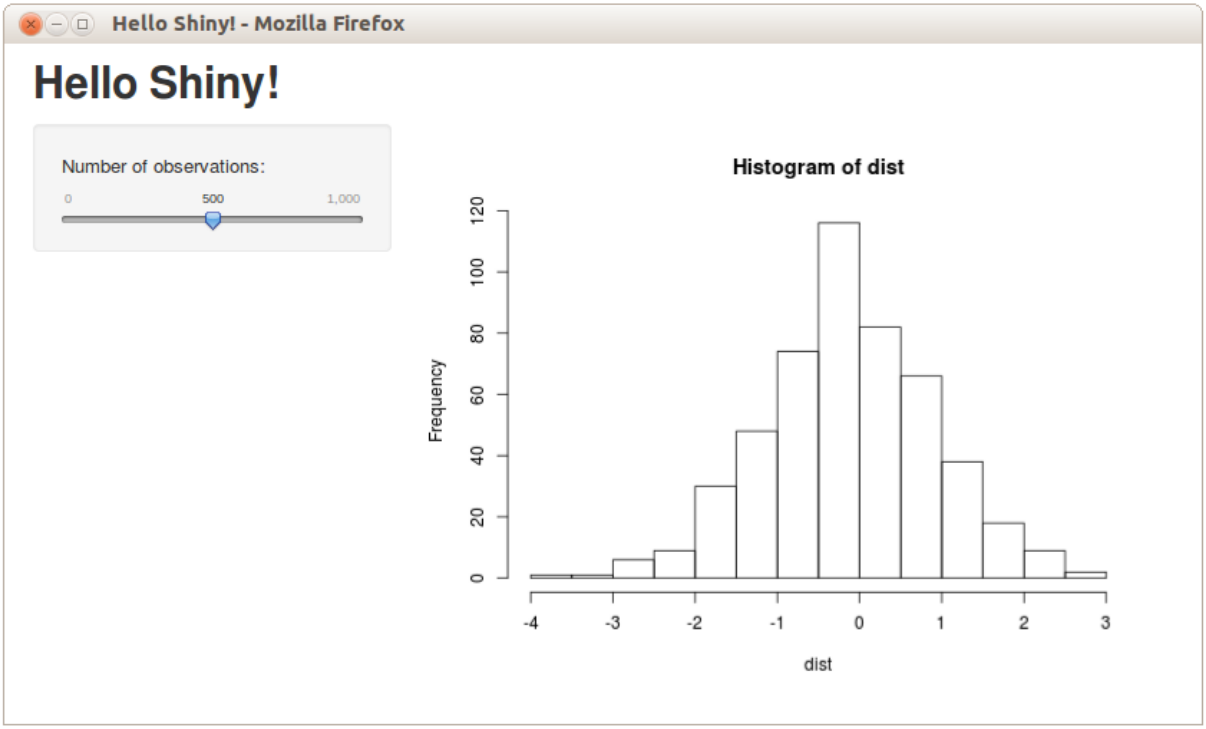
- Code can be entirely within R (or customize with HTML/JavaScript)
- Take a look at RStudio's showcase of Shiny applets

Shiny Documentation and Help

- RStudio Tutorial
  Much of tonight's material is motivated by this tutorial
- Shiny Setup, Showcase, and Server setup
  Useful if you want to use shiny on your own webserver
- Shiny Github Page
  See the latest updates
- Shiny Mailing List
  Check the tutorial first, then search the mailing list.

A Simple Example

```
library(shiny)
runApp("shinyApps/01_Hello")
```



A Tale of Two Files

The quickest/easiest way to create a shiny app is to define the user interface in a file named **ui.R** and the server side functionality in **server.R**.

- **ui.R** defines the page layout and user interface
- **server.R** contains the R code to create any output

ui.R

```
library(shiny)

# A simple/common user interface template
shinyUI(fluidPage(

  # Application title
  titlePanel("Title"),

  sidebarPanel(
    # Define some inputs here
  ),

  mainPanel(
    # output (from the server) go here
  )

))
```

server.R

```
library(shiny)

# Server side logic
shinyServer(function(input, output) {
  # do something
})
```

Reactivity

shiny is built on the idea of **reactive programming**. The idea that outputs should be automatically updated whenever an input value changes.

```
input values => R code => output values
```

Reactive expressions keep track of what values they read and what values they change. If those values become "out of date", they know their return value is out of date and will automatically recalculate.

An example

```
shiny::runApp("shinyApps/02_Reactivity", display.mode = "showcase")
```

Your Turn

- Consider extending the hello world example:

```
shiny::runApp("shinyApps/01_Hello", display.mode = "showcase")
```

- **Challenge 1:** add an input to change the mean and standard deviation (Hint: see `?numericInput`).
- **Challenge 2:** add an input to simulate from a gamma as well as a normal (Hint: you can simulate from gamma distribution with `rgamma`).
- **Challenge 3:** Extend **Challenge 2** so that there are dynamic inputs according to the desired distribution. That is, display mean and std dev inputs for normal distribution and a shape input for gamma (Hint: see `conditionalPanel`).

Shiny Inputs

Shiny has many different input options: - `actionButton()` - creates a clickable button - `checkboxInput()` and `checkboxGroupInput()` - `dateInput()` - calendar to select a date - `dateRangeInput()` - select a range of dates - `fileInput()` - upload a file - `numericInput()` - input a numeric value - `radioButtons()` - select one or more items - `sliderInput()` - slide along a range of values - `textInput()` - input a string

Shiny Outputs

Shiny also has many output options: - `renderDataTable()` - outputs an interactive, sortable data table - `htmlOutput()` - output html elements - `renderPlot()` - output an R plot - `renderPrint()` - output text from print() in R - `renderTable()` - output an HTML table - `renderText()` - output text from R - `renderUI()` - output a custom part of the user interface - `renderImage()` - print an image to the page

Other User Interface Options

- `tabsetPanel()` - make multiple different output views (i.e. a plot in one tab, a data table in another)
- `helpText()` - create additional text to help users navigate your applet
- `submitButton()` - only update outputs when this button is clicked
- `conditionalPanel()` - only show certain UI options when conditions are met (i.e. if a certain tab is open, or a certain input is selected)

Your Turn

Using your own data or the OkCupid dataset provided, create a simple Shiny app. Use the 03_OKCupid app as a starting point.

```
shiny::runApp("shinyApps/03_OKCupid")
```

- Ideas:
- Plot some aspect of the data with color based on another aspect of the data
- Use `subset()` and `checkboxInput()` to plot user-selected subsets
- Challenges:
- Use `tabsetPanel()` to display different summaries or plots
- Allow multiple input variables
- Use if statements to output different plot types and let the user choose which plot type to output