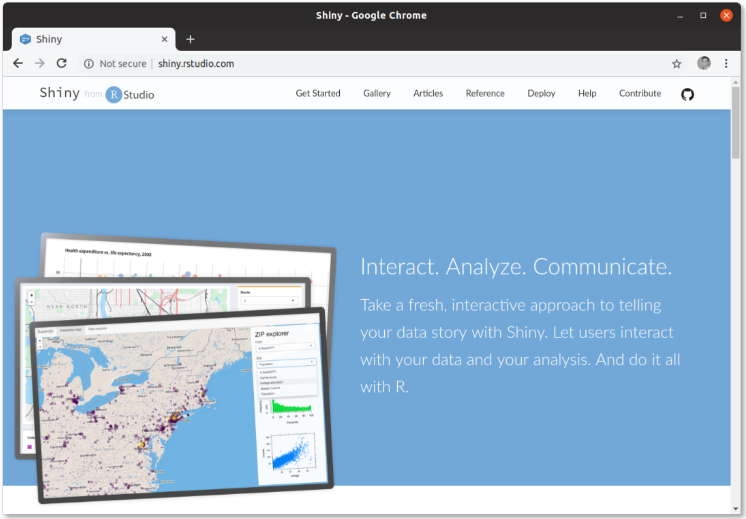
Dashboards with Shiny

Lecture 19

Dirk Eddelbuettel

STAT 430: Data Science Programming Methods (Fall 2019)
Department of Statistics, University of Illinois

Shiny



Interact. Analyze. Communicate.

Take a fresh, interactive approach to telling your data story with Shiny. Let users interact with your data and your analysis. And do it all with R.

Many resources at shiny.rstudio.com (and other) sites

ILLINOIS

Overview

- A tremendously *useful and popular* framework
- *Simple* enough to get started quickly
- *Powerful* enough to create interesting applications
- *Extensible* enough to cover many usage pattern
- Often run from a server but local development possible
- Very active development both from RStudio and community extensions

First Example

```
# install.packages("shiny")  # as needed
library(shiny)
shinyAppDir(system.file("examples/01_hello",
                        package="shiny"))
```

This run the example `01_hello/` from the Shiny package.

A new window should pop up. Alternatively,

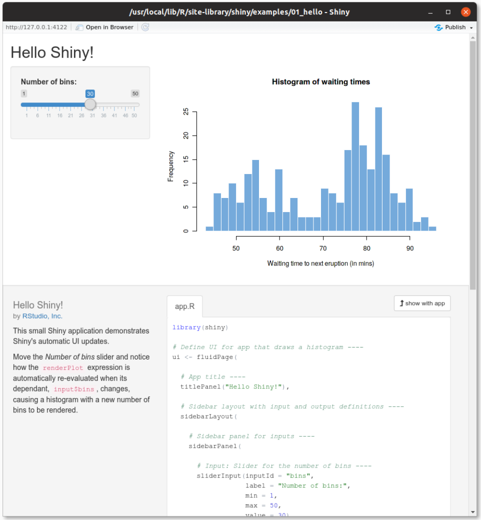
```
runExample("01_hello")
```

runs the example *and* shows the code.

Shiny

`runExample("01_hello")`

launches window with application *and* code.



Hello Shiny!
by RStudio, Inc.

This small Shiny application demonstrates Shiny's automatic UI updates.

Move the *Number of bins* slider and notice how the `renderPlot` expression is automatically re-evaluated when its dependant, `input$bins`, changes, causing a histogram with a new number of bins to be rendered.

```r
library(shiny)

# Define UI for app that draws a histogram ----
ui <- fluidPage(

  # App title ----
  titlePanel("Hello Shiny!"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(

    # Sidebar panel for inputs ----
    sidebarPanel(

      # Input: Slider for the number of bins ----
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
```

First Example code: 1 of 2

60 lines of code (with lots of whitespace) in `app.R`

```r
# Define UI for app that draws a histogram ----
ui <- fluidPage(
  # App title ----
  titlePanel(”Hello Shiny!”),
  # Sidebar layout with input and output definitions ----
  sidebarLayout(
    # Sidebar panel for inputs ----
    sidebarPanel(
      # Input: Slider for the number of bins ----
      sliderInput(inputId = ”bins”,
                  label = ”Number of bins:”,
                  min = 1, max = 50, value = 30)
    ),
    # Main panel for displaying outputs ----
    mainPanel(
      # Output: Histogram ----
      plotOutput(outputId = ”distPlot”)
    )
  )
)
```

First Example code 2 of 2

60 lines of code (with lots of whitespace) in `app.R`

```
# Define server logic required to draw a histogram ----
server <- function(input, output) {
  # Histogram of the Old Faithful Geyser Data ----
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call to renderPlot to indicate that:
  #
  # 1. It is ”reactive” and therefore should be automatically re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({
    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    hist(x, breaks = bins, col = ”#75AADB”, border = ”white”,
         xlab = ”Waiting time to next eruption (in mins)”,
         main = ”Histogram of waiting times”)
    })
}

# Create Shiny app ----
shinyApp(ui = ui, server = server)
```

First Example Discussion

- A Shiny app conists of
    - a UI component definining the User Interface
    - a Server component defining the computation
    - each is implemented in one function
- The `ui()` function defines
    - a title in a panel
    - a layout in a sidebar containing a slide
    - the slider has a label, mix/max and default
    - the slide provides its value under id `bins`
    - a main panel with a plot `dispPlot`

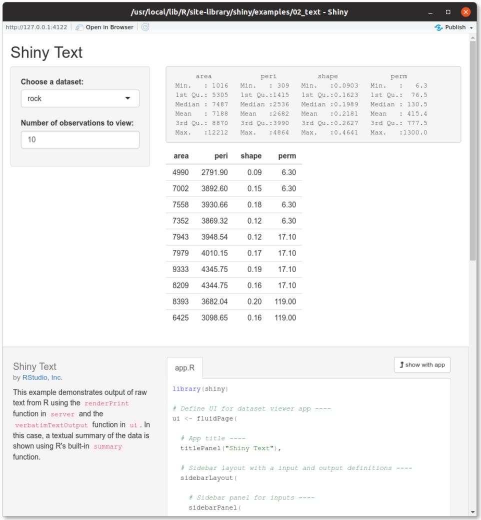
First Example Discussion

- The `server()` function has
  - arguments `input` and `output`
  - `input` contains `input$bins` – our slider
  - `output` is assigned `distPlot` – what is plotted
- The `renderPlot()` function is pretty standard:
  - access the waiting times of Old Faithful as `x`
  - define a sequence of `bins+1` values from min to max
  - use it in `hist()` along with some formating

Shiny

```
runExample("02_text")
```

launches window with application *and* code.



Shiny Text

Choose a dataset: rock

Number of observations to view: 10

```
      area            peri           shape              perm       
 Min.   : 1016   Min.   : 309   Min.   :0.0903   Min.   :   6.3  
 1st Qu.: 5305   1st Qu.:1415   1st Qu.:0.1623   1st Qu.:  76.5  
 Median : 7487   Median :2536   Median :0.1989   Median : 130.5  
 Mean   : 7188   Mean   :2682   Mean   :0.2181   Mean   : 415.4  
 3rd Qu.: 8870   3rd Qu.:3990   3rd Qu.:0.2627   3rd Qu.: 777.5  
 Max.   :12212   Max.   :4864   Max.   :0.4641   Max.   :1300.0  
```

area	peri	shape	perm
4990	2791.90	0.09	6.30
7002	3892.60	0.15	6.30
7558	3930.66	0.18	6.30
7352	3869.32	0.12	6.30
7943	3948.54	0.12	17.10
7979	4010.15	0.17	17.10
9333	4345.75	0.19	17.10
8209	4344.75	0.16	17.10
8393	3682.04	0.20	119.00
6425	3098.65	0.16	119.00

Shiny Text
by RStudio, Inc.

This example demonstrates output of raw text from R using the `renderPrint` function in `server` and the `verbatimTextOutput` function in `ui`. In this case, a textual summary of the data is shown using R's built-in `summary` function.

```
library(shiny)

# Define UI for dataset viewer app ----
ui <- fluidPage(

  # App title ----
  titlePanel("Shiny Text"),

  # Sidebar layout with a input and output definitions ----
  sidebarLayout(

    # Sidebar panel for inputs ----
    sidebarPanel(
```

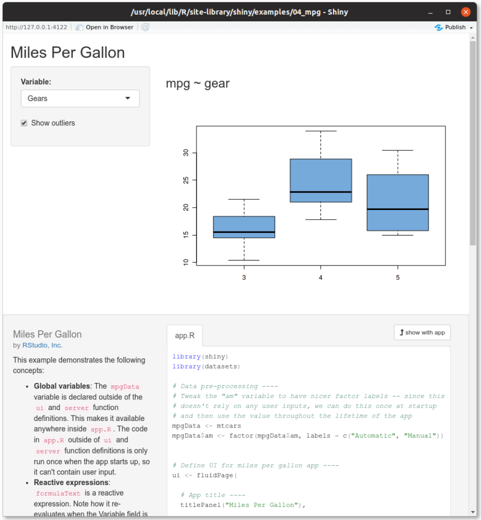
Second and Third Example

```
shinyAppDir(system.file(”examples/02_text”, package=”shiny”))
shinyAppDir(system.file(”examples/03_reactivity”, package=”shiny”))
```

- A simple text reactivity example
- The dropdown selects a dataset
- Changing the selection immediately changes the display
- Changing the number of obs to show changes the length
- Code run automatically (!!) after value changes
    - that is *the* key feature of Shiny: reactivity

```
runExample("04_mpg")
```

launches window with application *and* code.



Miles Per Gallon
by RStudio, Inc.

This example demonstrates the following concepts:

- **Global variables**: The `mpgData` variable is declared outside of the `ui` and `server` function definitions. This makes it available anywhere inside `app.R`. The code in `app.R` outside of `ui` and `server` function definitions is only run once when the app starts up, so it can't contain user input.
- **Reactive expressions**: `formulaText` is a reactive expression. Note how it re-evaluates when the Variable field is

```r
library(shiny)
library(datasets)

# Data pre-processing ----
# Tweak the "am" variable to have nicer factor labels -- since this
# doesn't rely on any user inputs, we can do this once at startup
# and then use the value throughout the lifetime of the app
mpgData <- mtcars
mpgData$am <- factor(mpgData$am, labels = c("Automatic", "Manual"))


# Define UI for miles per gallon app ----
ui <- fluidPage(

  # App title ----
  titlePanel("Miles Per Gallon"),
```

Fourth Example

```
shinyAppDir(system.file(”examples/04_mpg”, package=”shiny”))
```

- Select different variables
- Shown in boxplot to study relationship to mpg (miles per gallon)
- Display of outliers can be turned on/off

More simple examples in the package

- `05_sliders` different slides with one or more values
- `06_tabsets` ‘tabs’ for the output allowing multiple views
- `07_widgets` another display illustration
- `08_html` shows results directly in a (barebones) webpage
- `09_upload` lets the user upload a csv (or alike) file
- `10_download` inverts this and offers downloads of datasets
- `11_timer` displays the current time, constantly updated

Use helper `runExample()` as *e.g.* in `runExample(”05_sliders”)` to run one of these examples and see the code behind it.

Running Shiny Apps

- We saw the first example with a single file containing
  - functions `ui()` and `server()`
  - called by one function `shinyApp()`
- The `shinyAppDir()` function can run a Shiny application
  - typically organized as one per directory
  - see the eleven examples
- Shiny applications can also be split into files
  - `ui.R` definining the *user-interface* and
  - `server.R` defining the *backend computation*.

RMarkdown document

- Shiny can be used well along with RMarkdown:
    - Select ‘File -> New File -> R Markdown -> Shiny’
    - Last choice between (html) document and presentation
- With ‘document’ mode, `Shiny` becomes runtime for RMarkdown
- This creates a dynamic documents …
- … which retain full markdown formatting options.

Beyond Basic Shiny

RMarkdown document example

```r
rmarkdown::run("example.Rmd")
```

Beyond Basic Shiny

ILLINOIS

Dynamic Graphs

- Many add-on packages for R combine Shiny with Javascript-based display widgets
- This offers interactivity in the browser (where Javascript runs)
- Example: `dygraphs` for interactive (zoomable) time series:
  - `https://rstudio.github.io/dygraphs/`
- *Many* other choices, and *e.g.* `leaflet` for maps very popular
  - `https://rstudio.github.io/leaflet/`
- See the Shiny Gallery for *much* more

Beyond Basic Shiny

ILLINOIS

Styling

- As Shiny is delivered via the web browser, many web frameworks and styling options available for Shiny too
- “Bootstrap” (the CSS/JS framework) one of many options
- “Material Design” (Google’s style) another option
- See CRAN and various Github repos

Responsive Styling

- Initial Shiny apps were often set up for a fixed (pixel) resolution
- But responsiveness is desirable:
    - allow phone/tablet/desktop use
    - allow resizing and different size browser windows
- One solution: `flexdashboard`
    - See https://rmarkdown.rstudio.com/flexdashboard/
- Can be used with and without Shiny
- My default layout choice

Beyond Basic Shiny

Hosting

- One possibility: run the (open source) Shiny server
- Another possibility: run the (commercial) Shiny Pro server
- Have it hosted: `http://www.shinyapps.io/`
    - Basic (free) tier open to all
    - Limited number of service hours

Final Example

Flexdashboard with ggplot2

- One of the examples at the `flexdashboard` site
- Combines the `mtcars` data with `ggplot2`
- Additional feature: “brushing”
    - an interactive selection method
- Implemented by `flexdashboard`:
    - we get the selected data back
    - selection affects other display

Summary

Shiny

- An excellent framework to quickly construct dashboard
- Simple yet flexible and extensible enough for sophisticated use
- Wide variety of built-in components for input and styling
- Full ecosystem of community-added extensions & applications
- Documentation starting point at shiny.rstudio.com