Shiny

Data Products

Brian Caffo, Jeff Leek, Roger Peng

What is Shiny?

- Shiny is a platform for creating interactive R programs embedded into a web page.
- Suppose that you create a prediction algorithm, with shiny you can *very easily* create web input form that calls R and thus your prediction algorithm and displays the results.
- Using Shiny, the time to create simple, yet powerful, web-based interactive data products in R is minimized.
- However, it lacks the flexibility of full featured (and more complex) solutions.
- Shiny is made by the fine folks at R Studio.

What else is out there?

- Creating any solution requiring fairly deep knowledge of web client/server programming
- OpenCPU by Jerome Ooms, is a really neat project providing an API for calling R from web documents
- And he even hosts an OpenCPU server, but you can create your own

Getting started

- Make sure you have the latest release of R installed
- If on windows, make sure that you have Rtools installed
- `install.packages("shiny")`
- `library(shiny)`
- Great tutorial at http://shiny.rstudio.com/tutorial/
- Basically, this lecture is walking through that tutorial offering some of our insights
- Note, some of the proposed interactive plotting uses of Shiny could be handled by the very simple `manipulate` function rstudio manipulate
- Also, `rCharts` is will be covered in a different lecture.

ui.R

```
library(shiny)
shinyUI(pageWithSidebar(
  headerPanel("Data science FTW!"),
  sidebarPanel(
    h3('Sidebar text')
  ),
  mainPanel(
      h3('Main Panel text')
  )
))
```

To run it

- In R, change to the directories with these files and type `runApp()`
- or put the path to the directory as an argument
- It should open an browser window with the app running

R functions for HTML markup

`ui.R`

```
shinyUI(pageWithSidebar(
  headerPanel("Illustrating markup"),
  sidebarPanel(
      h1('Sidebar panel'),
      h1('H1 text'),
      h2('H2 Text'),
      h3('H3 Text'),
      h4('H4 Text')

  ),
  mainPanel(
      h3('Main Panel text'),
      code('some code'),
      p('some ordinary text')
  )
))
```

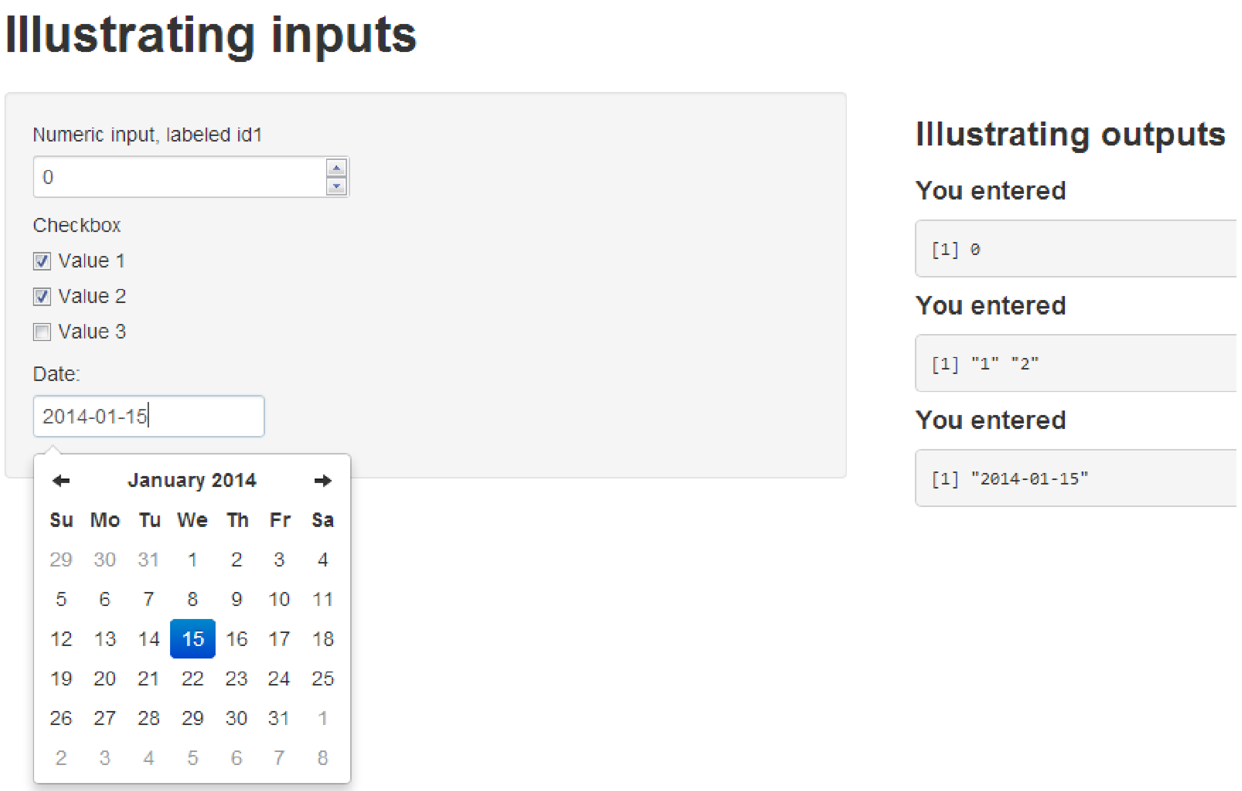
Illustrating inputs ui.R

```
shinyUI(pageWithSidebar(
  headerPanel("Illustrating inputs"),
  sidebarPanel(
    numericInput('id1', 'Numeric input, labeled id1', 0, min = 0, max = 10, step = 1),
    checkboxGroupInput("id2", "Checkbox",
                       c("Value 1" = "1",
                         "Value 2" = "2",
                         "Value 3" = "3")),
    dateInput("date", "Date:")
  ),
  mainPanel(

  )
))
```

Part of ui.R

```
  mainPanel(
        h3('Illustrating outputs'),
        h4('You entered'),
        verbatimTextOutput("oid1"),
        h4('You entered'),
        verbatimTextOutput("oid2"),
        h4('You entered'),
        verbatimTextOutput("odate")
  )
```



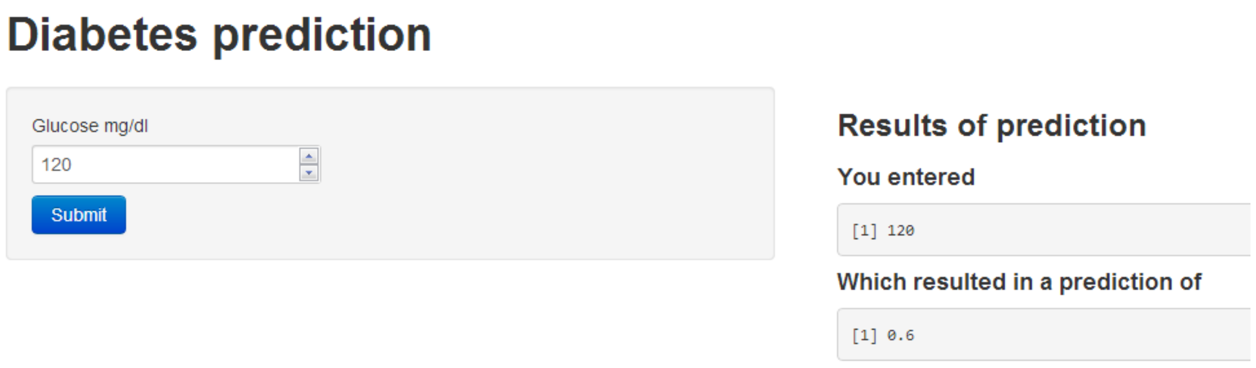
```
[1] 0
[1] "1" "2"
[1] "2014-01-15"
```

Figure 1: outputs

```
shinyUI(
  pageWithSidebar(
    # Application title
    headerPanel("Diabetes prediction"),

    sidebarPanel(
      numericInput('glucose', 'Glucose mg/dl', 90, min = 50, max = 200, step = 5),
      submitButton('Submit')
    ),
    mainPanel(
        h3('Results of prediction'),
        h4('You entered'),
        verbatimTextOutput("inputValue"),
        h4('Which resulted in a prediction of '),
        verbatimTextOutput("prediction")
    )
  )
)
```

The result



Results of prediction

You entered

```
[1] 120
```

Which resulted in a prediction of

```
[1] 0.6
```

Figure 2: prediction model

ui.R

```
shinyUI(pageWithSidebar(
  headerPanel("Example plot"),
  sidebarPanel(
    sliderInput('mu', 'Guess at the mean',value = 70, min = 62, max = 74, step = 0.05,)
  ),
  mainPanel(
    plotOutput('newHist')
  )
))
```

The output

Other things Shiny can do

- Allow users to upload or download files
- Have tabbed main panels
- Have editable data tables
- Have a dynamic UI
- User defined inputs and outputs
- Put a submit button so that Shiny only executes complex code after user hits submit

---

Distributing a Shiny app

- The quickest way is to send (or put on github or gist or dropbox or whatever) someone the app directory and they can then call `runApp`
- You could create an R package and create a wrapper that calls `runApp`

- Of course, these solutions only work if the user knows R
- Another option is to run a shiny server
- Requires setting up a (Shiny server)[http://www.rstudio.com/shiny/server/]

  - Probably easiest if you use one of the virtual machines where they already have Shiny servers running well (for example, on AWS)
- Setting up a Shiny server is beyond the scope of this class as it involves some amount of linux server administration
- Groups are creating a Shiny hosting services that will presumably eventually be a fee for service or freemium service
- BTW, don't put system calls in your code (this is one of the first things many of us do for fun, but it introduces security concerns)