Interactive R: Shiny

Workshop April 19, 10:30am

Chardon 115

Dr. Wolfgang Rolke

Log-on and setup

- Username: .\esma
- Password: Mate1234
- Open browser, go to http://academic.uprm.edu/wrolke/shiny/workshop/workshop.htm
- Download workshop.zip, copy-paste it to folder My Documents, extract it
- Double-click on RShinyWorkshop.Rdata

What is Shiny?

- Shiny allows us to run R inside any standard browser (IE, Firefox, Chrome etc.)
- Shiny is interactive, so the user can make changes to what the R routine does on the fly.
- It is useful for teaching, illustrating concepts, publishing research results…
- What can it do? Really, your imagination is the limit..

Table of Content

1) Prerequisites to writing shiny apps
2) the basic parts of an app
3) How to run and debug an app
4) Some useful elements of shiny
5) How to deploy an app
6) Some examples
7) Homework / Competition

1) Prerequisites to writing shiny apps

“underneath” an app runs R, so some knowledge of R is needed. How much depends on what the app is supposed to do. (I often start a new app by writing the necessary R code inside of R)

The app is run in a browser, so it is a webpage, so some knowledge on how to make webpages is needed. Basic HTML coding.

2) the basic parts of an app

Each app has to have two basic files, written in ASCII with any standard program editor:

1) ui.R creates the basic layout of the app
2) server.R generates the content to be displayed

To begin writing a new app, create a new folder with the name of the app, and within the folder create the ui.R and server.R files

I have a folder called template, which contains a ui.R and a server.R with some elements that I generally want. So I simply make a copy of this folder and rename it.

Let's say we want to write an app that generates data from a standard normal distribution and draws the histogram. We want to let the user choose how many observations are generated and how many bins are used.

ui.R

```
shinyUI(fluidPage(

  titlePanel("Workshop - Example 1 -
          Basic Histogram"),
  sidebarLayout(
      sidebarPanel(
           numericInput(inputId="n“,
                label="Number of
            observations",value=1000),

numericInput(inputId="bins",
              label="Number of
                          bins",value=50)
      ),
      mainPanel(
           plotOutput("plot")
      )
  )
))
```

server.R

```
shinyServer(function(input, output) {

   data <- reactive({
       x <- rnorm(input$n)
       x
   })
   output$plot <- renderPlot({
             hist(data(),input$bins,
             main="",xlab="x")
   })
})
```

Running and Debugging

Now we have to open R, and start with

> library(shiny)

(I usually do this in a .First file, if you don't know what that is read my SIDIM 2015 talk **Some Features of R You Might Not Yet Know**)

Running and Debugging

Next type

> runApp(“c:/ (folderpath..)/app1”)

and the our first app is up and running!

Here (folderpath..) is the folderpath were the app is located.
If your app folder is located in the same directory as the .Rdata file you used to start R you can now use

> runApp(paste(getdir().“/app1”,sep=“”))

Running and Debugging

There are three basic types of coding errors:

1) Error in ui.R

Let's eliminate one of the parentheses and see what happens

2) Error in server.R

a) Let's take out the last parentheses in hist()

Look at the R Console to get some info on the problem.

Also, we can add print statements to server.R to track down where the error occurs.

b) Some times errors also appear in the browser

Write hist(data,..)

Running and Debugging

To see the details of shiny commands use the usual ?

> ?numericInput

Or google “shiny numericInput”

A great place for help on thorny issues is http://stackoverflow.com but you should make a serious attempt at finding the answer yourself before posting questions there.

How to add text

```
shinyUI(fluidPage(

  titlePanel("Workshop - Example 1 – Basic
Histogram"),

  sidebarLayout(
      sidebarPanel(
            numericInput(inputId="n",
                  label="Number of observations",
                  value=1000),
            numericInput(inputId="bins",
                 label="Number of bins",value=50)
      ),
      mainPanel(
         uiOutput(“text”),
        plotOutput("plot")
      )
  )
))
```

```
shinyServer(function(input, output) {

   data <- reactive({
       x <- rnorm(input$n)
       x
   })

   output$text <- renderText({
          "<h2>My first Shiny app!</h2>"
  })
   output$plot <- renderPlot({
              hist(data(),input$bins,main="",
                           xlab="x")
   })
})
```

Input widgets

- textInput(“nam”,”Name”)
- selectInput(“gen”,”Gender”,choices=c(“Male”,”Female”),selected=“Female”)
- sliderInput(“age”,”Ages”, min=0, max=100, value=20, step = 1)
- radioButtons(“grad", “Grade”,choices =c(“A”,”B”,”C”,”D”,”F”,”W”), selected = “A”)

More examples can be found at http://shiny.rstudio.com/gallery/widget-gallery.html

Practice Exercise

Change app1 so that

The user can enter different values for the mean and standard deviation

The mean and standard deviation are shown in the mainPanel

The user can enter a title for the histogram

Workshop - Example 1 - Basic Histogram

Number of observations

1000

Number of bins

50

Mean

10

Standard Deviation

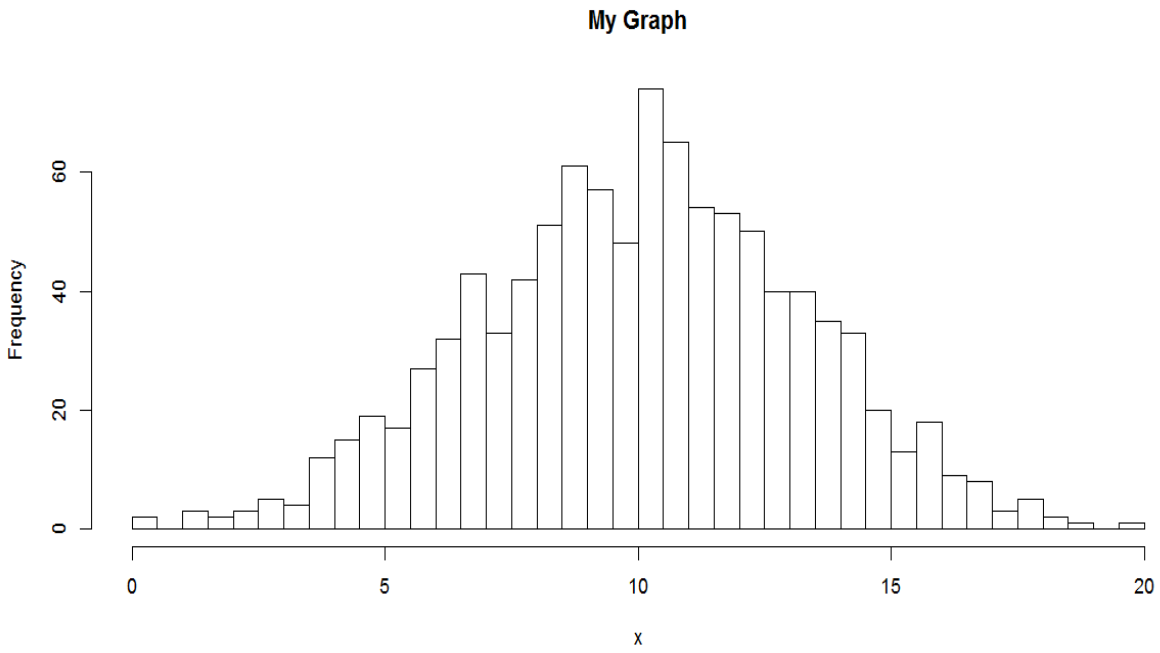
3

Graph Title

My Graph

Mean: 10

Std: 3



```
shinyUI(fluidPage(

  titlePanel("Workshop - Example 1 - Basic Histogram"),
  sidebarLayout(
    sidebarPanel(
        numericInput(inputId="n", label="Number of
                    observations",value=1000),
        numericInput(inputId="bins", label="Number of
                     bins",value=50),
        numericInput(inputId="mu",
                     label="Mean",value=0),
        numericInput(inputId="sig", label="Standard
                     Deviation",value=1),
        textInput("ttl","Graph Title",value="")
    ),
    mainPanel(
        uiOutput("text"),
        plotOutput("plot")
    )
  )
))
```

```
shinyServer(function(input, output) {

  data <- reactive({
      x <- rnorm(input$n,input$mu,input$sig)
      x
  })

  output$text <- renderText({
        line <- paste("<h4>Mean:",input$mu,"</h4>")
        line[2] <- paste("<h4>Std:",input$sig,"</h4>")
        line
  })

  output$plot <- renderPlot({
       hist(data(),input$bins,main=input$ttl,xlab="x")
  })

})
```

Practice Exercise

The user can choose to do a histogram or a boxplot

Workshop - Example 1 - Basic Graphs

Number of observations

1000

Number of bins

50

Mean

0

Standard Deviation

1

Graph Title

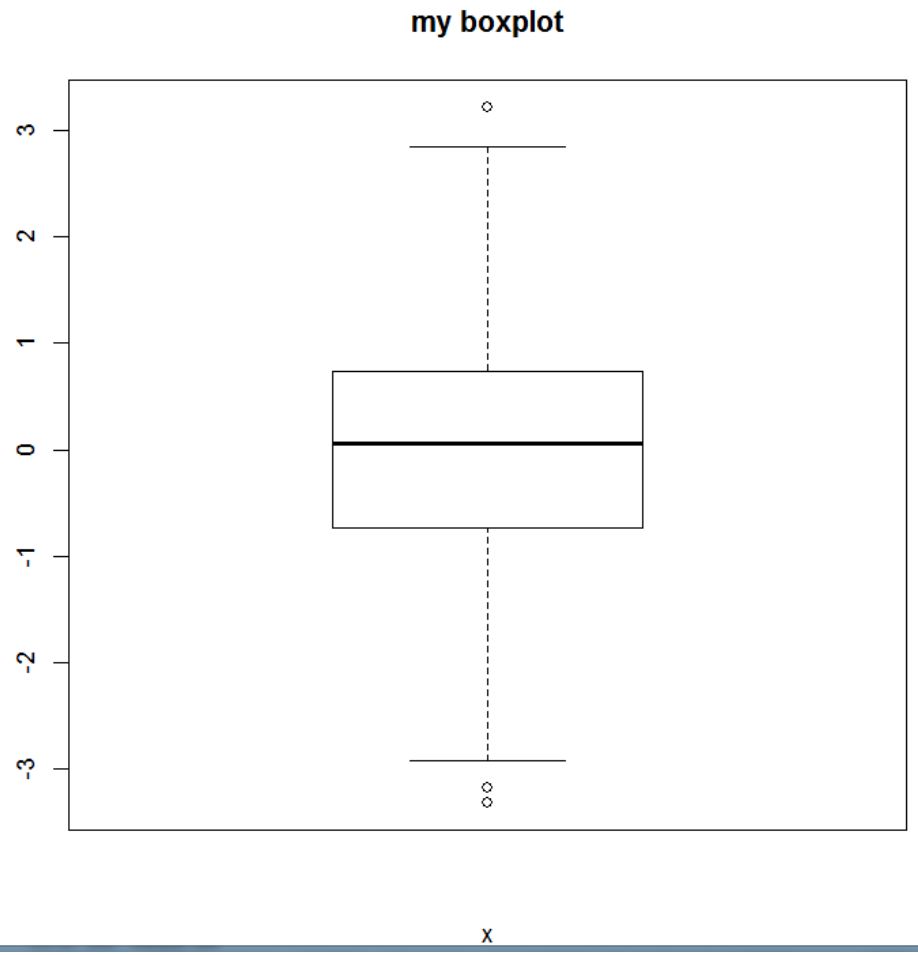
my boxplot

Which Graph?

- Histogram
- Boxplot

Mean: 0

Std: 1



```
shinyUI(fluidPage(
  titlePanel("Workshop - Example 1 - Basic Graphs"),
  sidebarLayout(
    sidebarPanel(
        numericInput(inputId="n", label="Number of
observations",value=1000),
        numericInput(inputId="bins",
            label="Number of bins",value=50),
        numericInput(inputId="mu",
            label="Mean",value=0),
        numericInput(inputId="sig",
            label="Standard Deviation",value=1),
        textInput("ttl","Graph Title",value=""),
        radioButtons(inputId="whichgraph",
            label="Which Graph?",
            choices=c("Histogram","Boxplot")
      ),width=3
    ),
    mainPanel(
        uiOutput("text"),
        plotOutput("plot", width = "500px", height =
"500px")
    )
  )
))
```

```
shinyServer(function(input, output) {
  data <- reactive({
     x <- rnorm(input$n,input$mu,input$sig)
     x
  })
  output$text <- renderText({
        line <- paste("<h4>Mean:",input$mu,"</h4>")
        line[2] <- paste("<h4>Std:",input$sig,"</h4>")
        line
  })
  output$plot <- renderPlot({
    if(input$whichgraph=="Histogram")
      hist(data(),input$bins,main=input$ttl,xlab="x")
    else
      boxplot(data(),main=input$ttl,xlab="x")
  })
})
```

Do it the way you want it

Just about anything can be made to look the way you want it. Two examples already in the last app:

1) change the size of the sidebar (,width=3)
2) Change the aspect ratio of the graph plotOutput("plot", width = "500px", height = "500px")

conditionalPanel

An ugly feature of our app: the input field Number of bins only makes sense for the histogram, not for the boxplot, so it should not appear when we do a boxplot.

Solution: conditionalPanel

```
shinyUI(fluidPage(
  titlePanel("Workshop - Example 1 - Basic Graphs"),
  sidebarLayout(
     sidebarPanel(
          numericInput(inputId="n", label="Number of observations",value=1000),
          conditionalPanel(condition = "input.whichgraph=='Histogram'",
               numericInput(inputId="bins", label="Number of bins",value=50)
          ),
          numericInput(inputId="mu", label="Mean",value=0),
          numericInput(inputId="sig", label="Standard Deviation",value=1),
          textInput("ttl","Graph Title",value=""),
          radioButtons(inputId="whichgraph",label="Which Graph?",
               choices=c("Histogram","Boxplot")),width=3
     ),
     mainPanel(
          uiOutput("text"),
          plotOutput("plot", width = "500px", height = "500px")
     )
  )
))
```

File Input

We want to do the graph for some predefined data sets

First we need to save the data sets in the same folder as ui.R and server.R, say with dump

Then we can read the data in with

```
 if(input$dataset=="Newcomb's Speed of Light") {
        source("newcomb.R")
        return(newcomb)
}
```

You can download the data sets from here:

http://academic.uprm.edu/wrolke/shiny/workshop/workshop.htm

```
shinyUI(fluidPage(
 titlePanel("Workshop - Example 1 - Basic Graphs"),
 sidebarLayout(
    sidebarPanel(
       selectInput("dataset", HTML("<h5>Choose a dataset:</h5>"),
            choices = c("Newcomb's Speed of Light",
                 "Weight of Euro Coins","Forbes
                 500","Random"),selected="Random"),
       conditionalPanel(condition = "input.dataset=='Random'",
          numericInput(inputId="n", label="Number of
                observations",value=1000),
          conditionalPanel(condition =
                   input.whichgraph=='Histogram'",
             numericInput(inputId="bins", label="Number of
                                bins",value=50)
          ),
          numericInput(inputId="mu", label="Mean",value=0),
          numericInput(inputId="sig", label="Standard
                                Deviation",value=1),
          textInput("ttl","Graph Title",value="")
       ),
       radioButtons(inputId="whichgraph",label="Which Graph?",
            choices=c("Histogram","Boxplot")),width=3
    ),
    mainPanel(
       uiOutput("text"),
       plotOutput("plot", width = "500px", height = "500px")
    )
 )
))
```

```
shinyServer(function(input, output) {
  data <- reactive({
     if(input$dataset=="Random")
        return(rnorm(input$n,input$mu,input$sig))
     if(input$dataset=="Newcomb's Speed of Light") {
       source("newcomb.R")
       return(newcomb)
     }
     if(input$dataset=="Weight of Euro Coins") {
       source("euros.R")
       return(euros)
     }
     if(input$dataset=="Forbes 500") {
       source("forbes.R")
       return(forbes$Assets)
     }
  })
  output$text <- renderText({
     if(input$dataset!="Random") return("")
     line <- paste("<h4>Mean:",input$mu,"</h4>")
     line[2] <- paste("<h4>Std:",input$sig,"</h4>")
     line
  })
  output$plot <- renderPlot({
    if(input$dataset=="Random") ttl<-input$ttl
    else ttl <- input$dataset
    if(input$whichgraph=="Histogram")
      hist(data(),input$bins,main=ttl,xlab="x")
    else
      boxplot(data(),main=ttl,xlab="x")
  })
})
```

Text Output

In the text area we want a table of summary statistics.

The idea here is to use R syntax to create a character vector which has the lines of the HTML code.

```
output$text <- renderText({
    x <- data()
    line <- "<table border=1>"
    line[2] <- "<tr><th>Sample Size</th>
                <th>Mean</th><th>Standard Deviation</th></tr>"
    line[3] <- paste("<tr><td>",length(x),
                "</td><td>",round(mean(x),2),
                "</td><td>",round(sd(x),3),"</td></tr>")
    line[4] <- "</table>"
    line
})
```

Tables

These tables rarely look very good. To change their appearance we need to use cascading style files. The easiest way is to include that in the ui.R. For example I almost always use

```
shinyUI(fluidPage(
  tags$head(
     tags$style(HTML("
        table, th, td {
           text-align:right;
        }
        th, td {
           padding: 10px;
        }
      "))
  ),
  titlePanel("Workshop - Example 1 - Basic Graphs"),
```

Panels

Often it is a good idea to have several panels to show different things. Say we want to separate the text from the graph.

```
mainPanel(
        tabsetPanel(
            tabPanel("Statistics",uiOutput("text")),
            tabPanel("Graphs",plotOutput("plot", width =
                                        "500px", height = "500px")),
            id="Tabs"
        )
    )
```

conditionalPanel

Again there are items on the left that only make sense for the graphs, so they should only appear when the Graph panel is selected. Again conditionalPanel to the rescue!

```
conditionalPanel( condition = "input.Tabs == 'Graphs'",
                  radioButtons(inputId="whichgraph",
                               label="Which Graph?",
                               choices=c("Histogram","Boxplot")),
                  conditionalPanel(condition =
                                     "input.dataset=='Random'",
                                   textInput("ttl","Graph Title",value=""))
                )
```

Animation

When generating random data we might want to do this a number of times. Slowly, so one can watch the changes. Here is how:

In ui.R:

```
sliderInput("k","Repeat!",min=1, max=10, value=0,step=1,
            animate=animationOptions(interval = 500,playButton="Go!")
          )
```

In server.R:

```
if(input$dataset=="Random") {
        for(i in 1:input$k) mu<-input$mu
        return(rnorm(input$n,input$mu,input$sig))
      }
```

Using libraries

Say we want to do the graphs with ggplot2

In server.R:

```
require(ggplot2)
shinyServer(function(input, output) {

…
 output$plot <- renderPlot({
      if(input$dataset=="Random") ttl<-input$ttl
      else ttl <- input$dataset
      dta<-data.frame(x=data())
      if(input$whichgraph=="Histogram") {
              bw <- diff(range(data()))/input$bins
              plt <- ggplot(data=dta,aes(x))+
              geom_histogram(aes(y = ..density..),color="black",fill="white", binwidth = bw)
      }
      else
        plt <- ggplot(data=dta,aes(factor(1,length(x)),x))+
              geom_boxplot()
      plt <- plt + xlab(ttl)+ylab("")
      print(plt)
  })
```

How to deploy an app

There are a number of ways to make shiny apps available to the public:

1) Email them the folder with the ui.R and server.R (and any other parts needed such as data sets)

2) Make a compressed zip file of the folder and put it on the internet. They can then be run with runUrl.

>runUrl("http://academic.uprm.edu/wrolke/shiny/workshop/app10.zip")

How to deploy an app

3) Put code on github, run with runGitHub

runGitHub('wolfgangrolke/app10')

4) Put app on shinyapps.io

https://drrolke.shinyapps.io/app10/

5) Get shinyapps pro, set up your own server

6) Some examples

1) Sampling – use animation

2) Confidence Intervals – use of simulation in teaching

3) Problem Generator

4) Taylor Series

7) Homework / Competition

Write an app that illustrates the concept of a probability distribution.

Email me your folder with the ui.R and server.R and anything else needed.

I will put all solutions online and ask for a vote

Winner gets a bottle of champagne (if under 21 a box of cookies) Minimum: 5 submissions

Deadline: May 1.