An overview of Shiny applications using R and RStudio

Marcus W. Beck[1]

[1]USEPA National Health and Environmental Effects Research Laboratory, Gulf Ecology Division, beck.marcus@epa.gov

Dec. 10, 2015

Who am I?

- ORISE post-doc for 2.5 years, fed postdoc since last week
- NHEERL Gulf Ecology Division
- Research focus on water quality assessment and indicator development
- Specific interests in statistical modelling, data assimilation, graphics

Who am I?

- R user since 2007
- Maintainer of two packages on CRAN:

SWMPr

Tools for retrieving, organizing, and analyzing data from the System Wide Monitoring Program of the National Estuarine Research Reserve System.

NeuralNetTools

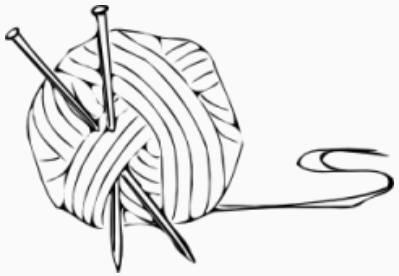
Visualization and analysis tools to aid in the interpretation of neural network models

Reproducible research workflow

General workflow for ***reproducible research*** - reproduce results from an experiment or analysis conducted by another.

From Wikipedia... 'The ultimate product is the ***paper along with the full computational environment*** used to produce the results in the paper such as the code, data, etc. that can be ***used to reproduce the results and create new work*** based on the research.'

Reproducible research workflow



The use of these tools increases transparency and transfer of information = ***better science***

Data prep, analysis, report, and sharing can all be done in RStudio IDE

Introduction to Shiny

Where does Shiny fit with reproducible research?

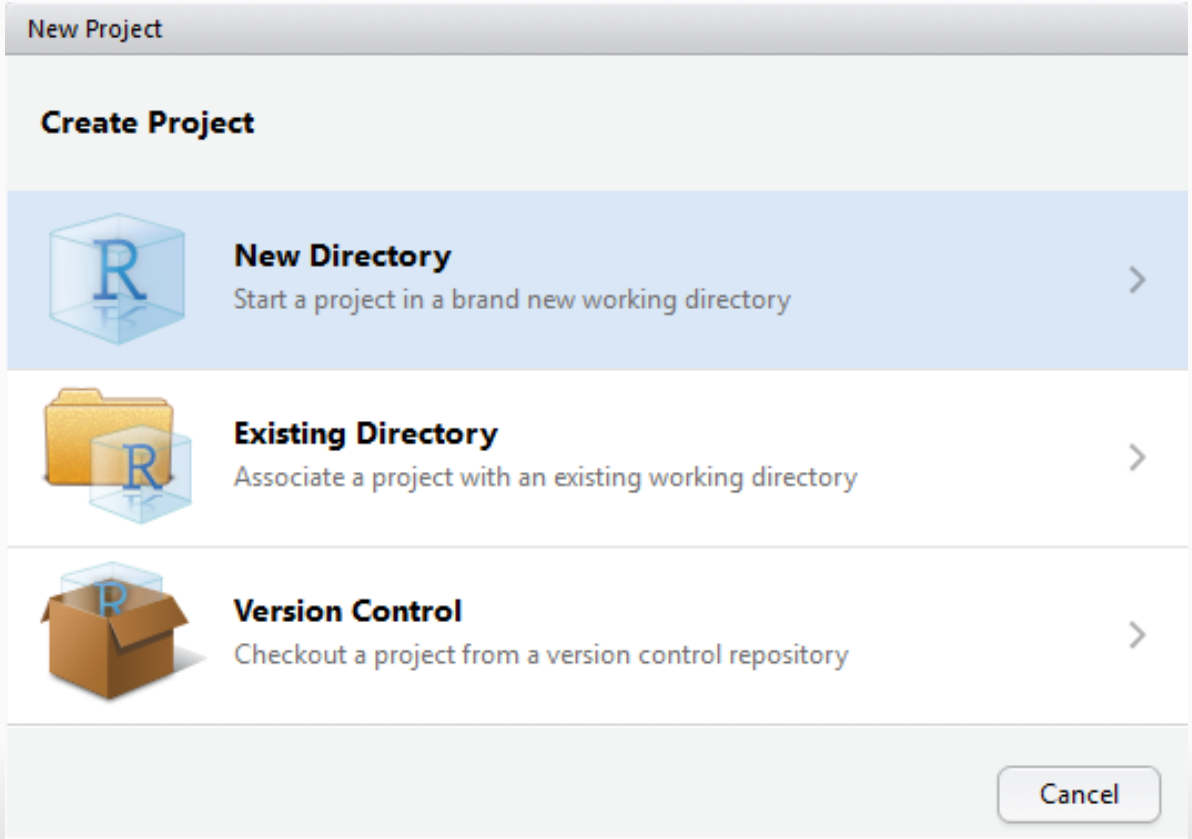
Shiny is a web application framework for R

- From the command line to a graphical user interface
- Make your code interactive
- Do not need to know anything about web programming
- Integrated very well with R studio

Tools like Shiny improve ***accessibility*** and ***communication***

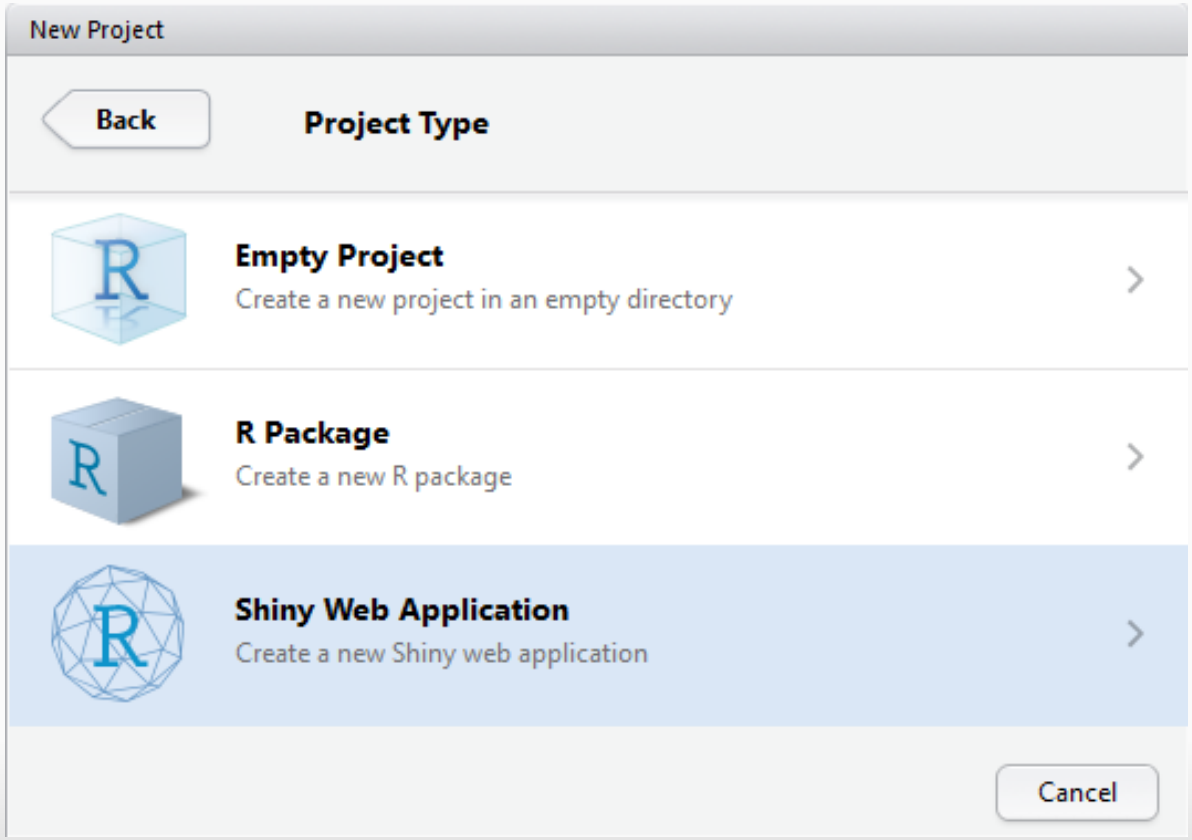
Introduction to Shiny

A minimal working example...



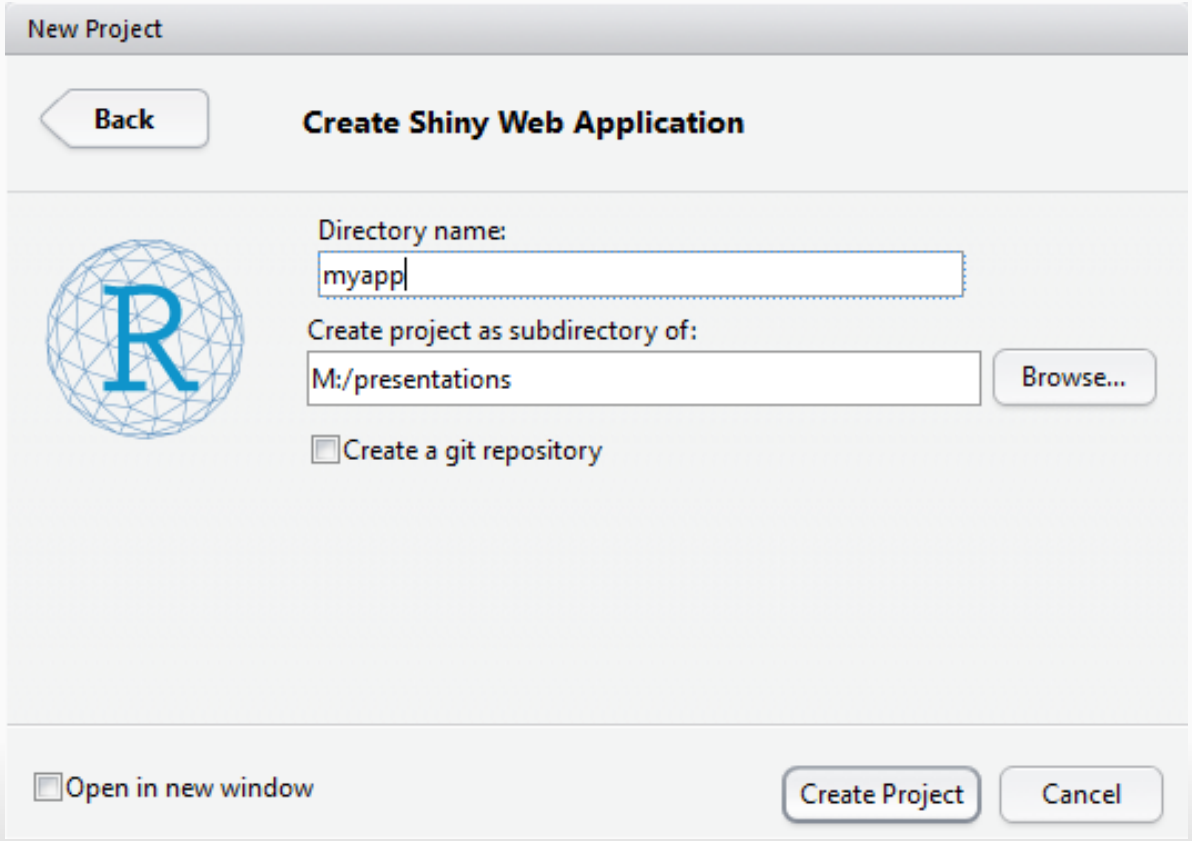
Introduction to Shiny

A minimal working example...



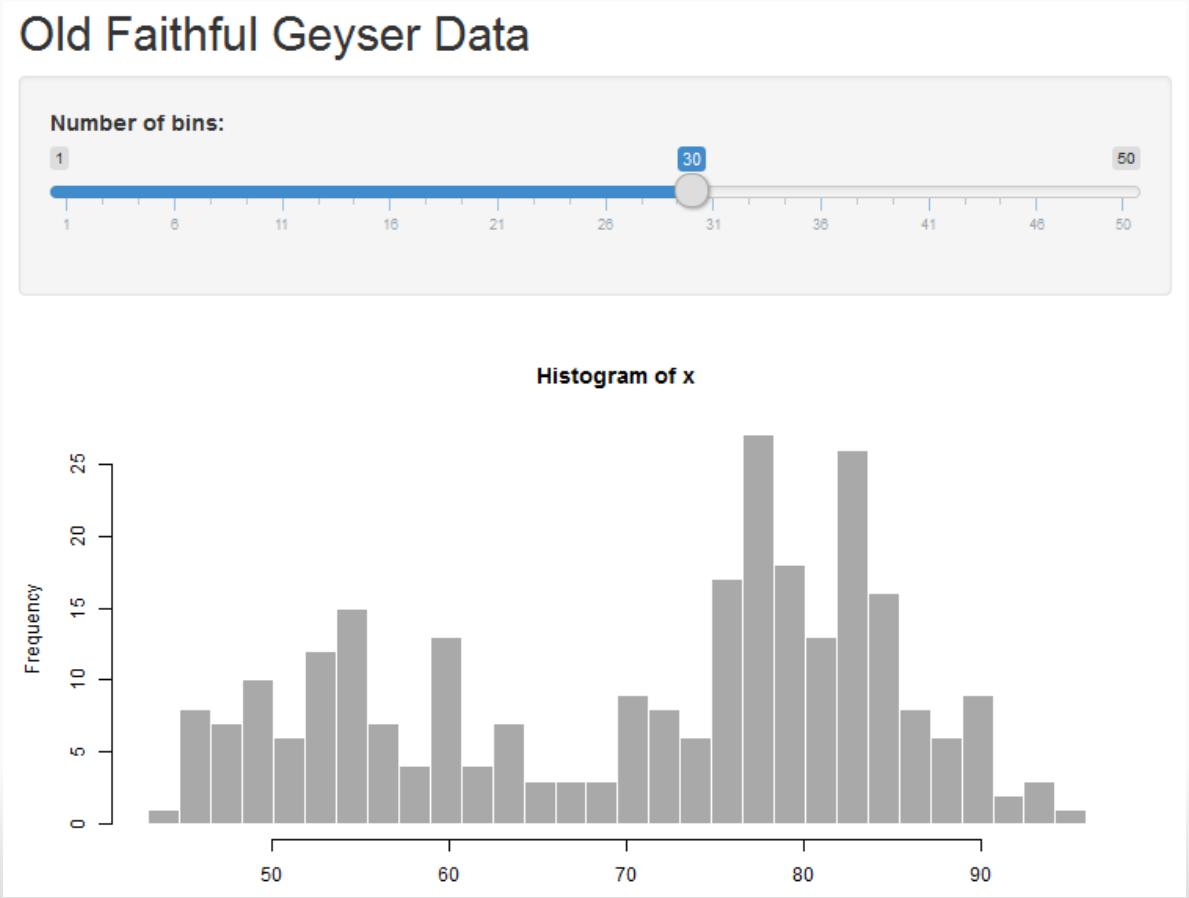
Introduction to Shiny

A minimal working example...



Introduction to Shiny

A minimal working example...



Introduction to Shiny

What's under the hood? Two files... ***server.R***

```
# This is the server logic for a Shiny web application.
# You can find out more about building applications with Shiny here:
#
# http://shiny.rstudio.com
#

library(shiny)

shinyServer(function(input, output) {

  output$distPlot <- renderPlot({

    # generate bins based on input$bins from ui.R
    x    <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')

  })

})
```

Introduction to Shiny

What's under the hood? Two files... ***ui.R***

```
# This is the user-interface definition of a Shiny web application.
# You can find out more about building applications with Shiny here:
#
# http://shiny.rstudio.com
#

library(shiny)

shinyUI(fluidPage(

  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

Introduction to Shiny

The files contain only R code!

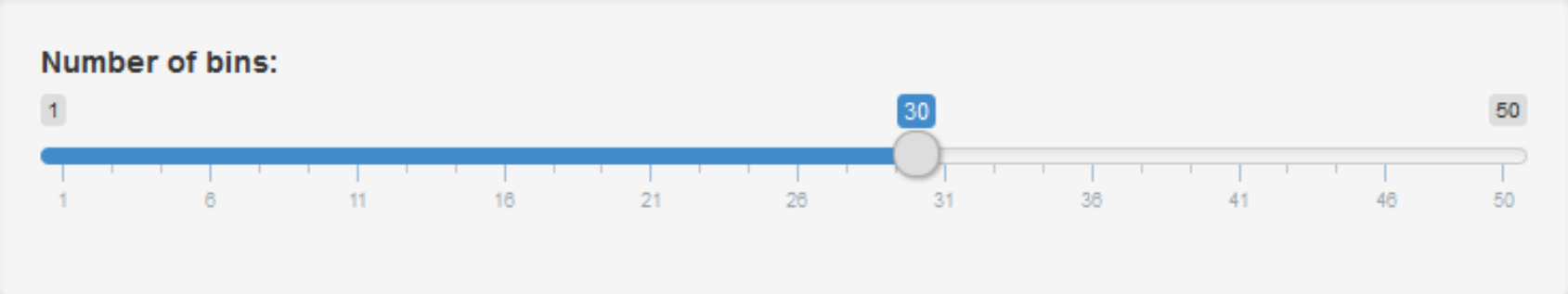
- ***server.R***: Contains instructions to build the app, e.g., plots, functions, etc.
- ***ui.R***: Controls the layout and appearance of the app, i.e., panel types, widgets, etc.

Executing a Shiny app will run both scripts, user input to ***ui.R*** sent to ***server.R***, output from ***server.R*** sent to ***ui.R*** for display

Introduction to Shiny

Step 1: User input to ***ui.R***, 'bins'

```
sliderInput("bins",
            "Number of bins:",
            min = 1,
            max = 50,
            value = 30)
```



Introduction to Shiny

Step 2: Input from ***ui.R*** sent to ***server.R***, executed

```
# generate bins based on input£bins from ui.R
x <- faithful[, 2]
bins <- seq(min(x), max(x), length.out = input$bins + 1)

# draw the histogram with the specified number of bins
hist(x, breaks = bins, col = 'darkgray', border = 'white')
```

Step 3: Output from ***server.R*** sent to ***ui.R***, plotted on app

```
plotOutput("distPlot")
```

Step 4: Rinse and repeat

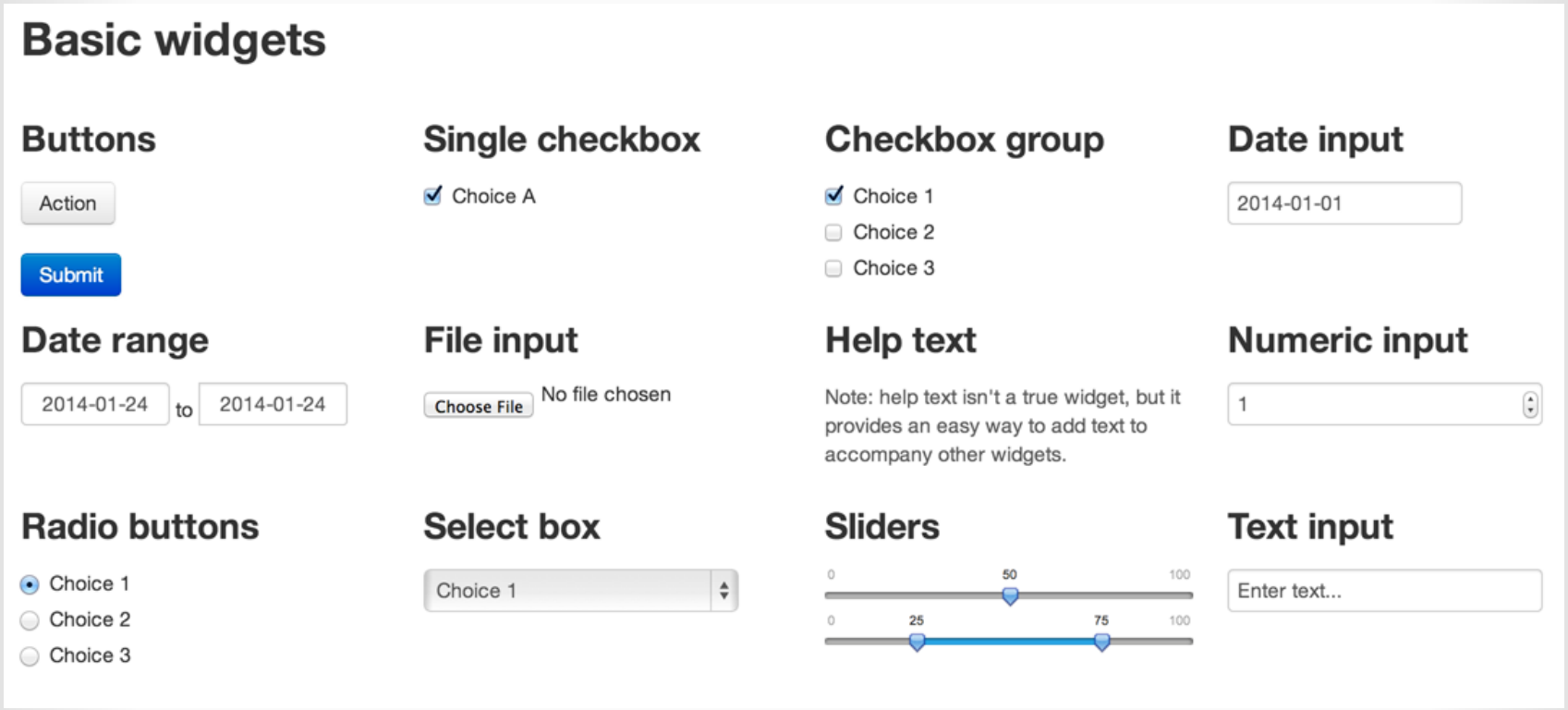
Introduction to Shiny

This style of programming and execution is ***reactive*** - re-executes automaticallly when inputs change

This has tremendous value:

- Quick code execution after initial setup
- Ease of use for others given application infrastructure
- Ease of use for the developer - no knowledge of web programming needed

Shiny applications are very flexible: widgets



Introduction to Shiny

Shiny applications are very flexible: outputs

function	expects	creates
`renderDataTable`	any table-like object	DataTables.js table
`renderImage`	list of image attributes	HTML image
`renderPlot`	plot	plot
`renderPrint`	any printed output	text
`renderTable`	any table-like object	plain table
`renderText`	character string	text
`renderUI`	Shiny tag object or	UI element (HTML)

Introduction to Shiny

Shiny applications are very flexible: use of HTML or Javascipt libraries

- Refined layouts: shinydashboard, htmlwidgets, shinyBS
- Interacive graphics: dygraphs, metricsgraphics, plotly
- Mapping: leaflet

Use of these libraries can create applications comparable to any other web application for data viz

Introduction to Shiny

Apps are easily shared: http://www.shinyapps.io/

Sign up for free account, push local shiny applications to a server, share URL

```
library(shinyapps)
deployApp()
```