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

- 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

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.'

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

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
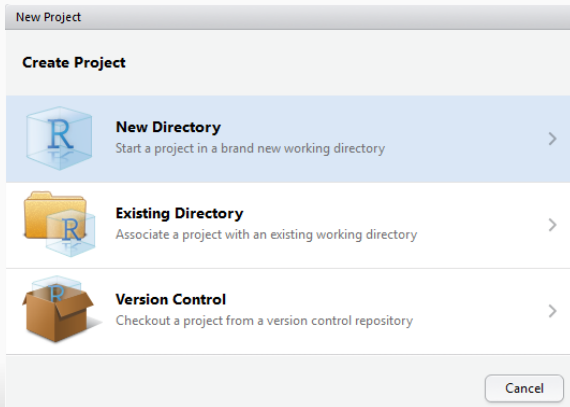- Integrated very well with R studio

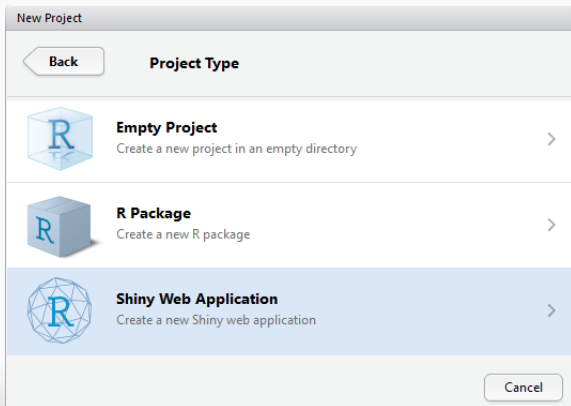Tools like Shiny improve ***accessibility*** and ***communication***

A minimal working example...

A minimal working example...

A minimal working example...

A minimal working example...

Introduction to Shiny

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

```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")
    )
  )
))
```

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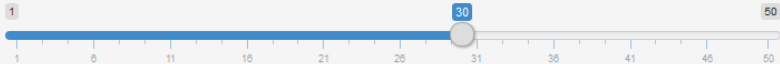
Step 1: User input to **ui.R**, 'bins'

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

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

```r
# generate bins based on input 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

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

Step 4: Rinse and repeat

✿EPA

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

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)

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

Apps are easily shared....

For the **single** user:

- Local RStudio 'project' as a standalone working directory

For **multiple** users:

- As a web application on your server, requires Shiny Server

- As a web application hosted on
  http://www.shinyapps.io/

Research application: Evaluating a statistical model to isolate and remove variance components from a time series

Scenario: Time series of dissolved oxygen provide information on ecosystem processes in aquatic systems

Problem: These time series are assumed to measure biological production/respiration, but noise from tidal cycles in coastal systems

A tidal and dissolved oxygen time series at Sapelo Island, Georgia
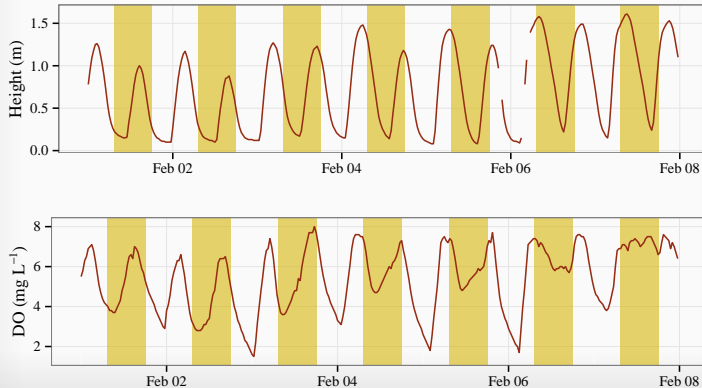
Grid-based evaluation of the statistical model using simulated
time series and varying model paremeters

- Time series varying by 4 characteristics, 3 levels per
  characteristic

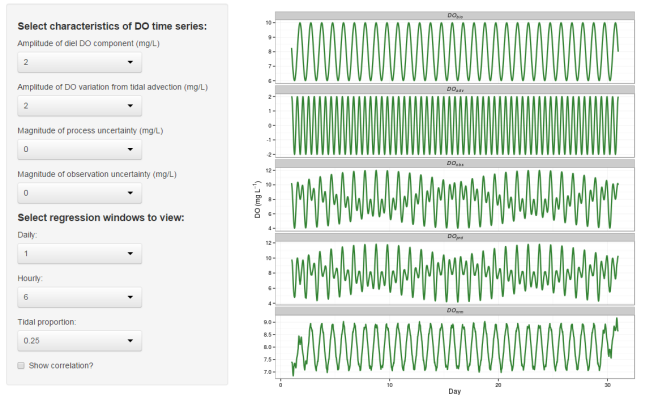- Model parameters varying by 3 characeristics, 3 levels per
  characterisic

2187 unique combinations: Very challenging to evaluate results,
use Shiny!

https://beckmw.shinyapps.io/detiding_sims/

Management application: Spatial and temporal assessment of water quality trends in NOAA estuary reserves

NERRS
National Estuarine Research Reserve System, established by Coastal Zone Management Act of 1972. Focus on ***long-term research***, ***monitoring***, ***education***, and ***stewardship*** for more effective coastal management.
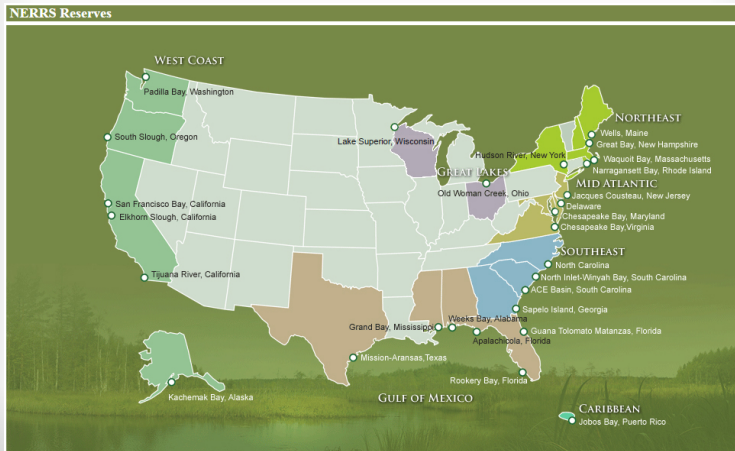
SWMP
System Wide Monitoring Program, initiated in 1995 to provide ***continuous monitoring*** data at over 140 stations in each of the 28 NERRS reserves

Shiny Examples

Location of NERRS estuary reserves with SWMP data

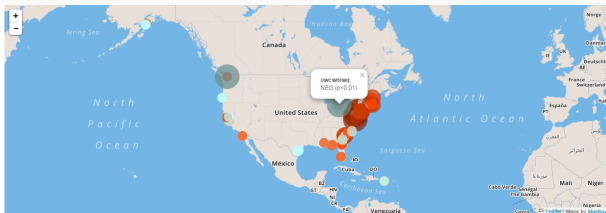Although SWMP data have been collected and processed using standardized methods...

- Long-term trends have not been evaluated between-reserves in several years

- Tools for simple trend analysis and visualization have been lacking

Solution: Use Shiny to bring results to users!

https://beckmw.shinyapps.io/swmp_comp/

Shiny has multiple benefits:

- Increased accessibility to information within and outside of the research community

- Available for use with minimal or no experience in web programming

- All open-source, no need for license and under active development

Opportunities for EPA Shiny Server?

Additional resources

✿EPA

RStudio Shiny tutorial:
`http://shiny.rstudio.com/tutorial/`

RStudio Shiny gallery:
`http://shiny.rstudio.com/gallery/`

Deploy Shiny apps:
`http://www.shinyapps.io/`

This presentation (`beck.marcus@epa.gov`):
`https://github.com/fawda123/shiny_pres/raw/master/`
`shiny_pres.pdf`