



Shiny Demonstration

West of Ireland Data Science

26th February 2015



shiny.rstudio.com

- ▶ Makers of Shiny : RStudio
(JJ Allaire, Hadley Wickham etc etc)
- ▶ RStudio? IDE for R. See
`www.rstudio.org` for more.
- ▶ Shiny's Lead Developers : Winston Chang
and Joe Cheng.

Overview of Demonstration

Overview of Demonstration

- ▶ Resources (i.e. Shiny Tutorial Page)
- ▶ Minimal Examples
- ▶ Widgets
- ▶ A bit about JavaScript
- ▶ Special Design Considerations
- ▶ Deploying Shiny

What is Shiny?

Easy web applications in R

(Source: *Shiny's Website*)

- ▶ **Shiny** makes it super simple for R users like you to turn analyses into interactive web applications that anyone can use.
- ▶ Let your users choose input parameters using friendly controls like sliders, drop-downs, and text fields.
- ▶ Easily incorporate any number of outputs like plots, tables, and summaries.

What is Shiny?

Easy web applications in R (contd.)

(Source: Shiny's Website)

- ▶ No **HTML** or **JavaScript** knowledge is necessary. If you have some experience with R, you're just minutes away from combining the statistical power of R with the simplicity of a web page.
- ▶ *(Remark: They do appear to be really handy - based on several examples available on the internet!)*

Resources

- ▶ Shiny Tutorial -
(shiny.rstudio.com/tutorial/)
- ▶ Chris Beeley's Book
(Sample Chapter Available)
- ▶ Stack-Overflow and GitHub



Scenarios Network for Alaska & Arctic Planning

[About](#) [Tools and Data](#) [Methods](#) [Projects](#) [Resources](#)

Exploring our future in a changing Arctic

[share](#)

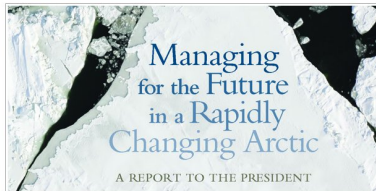
[contact](#) | [blog](#)

Managing for the Future in a Rapidly Changing Arctic

SNAP and ACCAP contributed to a report to the President of the U.S. The report highlights the need to coordinate and use the best available science to integrate cultural, environmental and economic factors in decision-making about development and conservation in the Arctic.

[read more >>](#)

1 2 3 4 5 6



Matthew Leonawicz (SNAP - Uni. Alaska Fairbanks)

github.com/ua-snap/shiny-apps

twitter.com/leonawicz

github

SOCIAL CODING



github - code sharing

Main Components of a Shiny Web App

- ▶ The shiny app is structurally a folder. The name of the app is the name of the folder.
- ▶ Shiny programs are the easiest to build and understand using two scripts, which are kept within this folder. They must be named `server.R` and `ui.R`.
- ▶ The input elements are defined in `ui.R` and processed by `server.R`, which then sends them back to `ui.R`
- ▶ Consideration: **Reactive Programming**

Reactive Programming

Simple R example re: reactivity

```
> A <- 5
> B <- A + 3
> A <-6           #Update A
>
> c(A,B,A+3)
[1] 6 8 9
>
```

Compare this with Microsoft Excel Spreadsheets

Basic structure of a Shiny program

- ▶ Selection of simple input widgets (checkboxes and radio buttons)
- ▶ Selection of simple output types (rendering plots and returning text)
- ▶ Selection of simple layout types (page with sidebar and tabbed output panel)
- ▶ Handling reactivity in Shiny

Running a Shiny App

To run a Shiny program on your local machine you just need to do the following:

1. Make sure that `server.R` and `ui.R` are in the application subfolder (`appName`).
2. Make the main folder R's working directory (using the `setwd()` command, for example `setwd(" /shinyFiles")`).

```
>...\shinyFiles\appName
```

3. Load the Shiny package (`library(shiny)`).
You should always do that in both `server.R` and `ui.R` files.

runApp

- ▶ Type `runApp("appName")` at the console.
- ▶ If you are in the application folder, just type `runApp()`
- ▶ **Important** - Just remember that it is a directory and not a file that you need to point to.

User Interface

- ▶ The `ui.R` file is a description of the UI and is often the shortest and simplest part of a Shiny application.
- ▶ All of the UI elements are defined within this instruction.
- ▶ The standard shiny layout is a three panel layout, with a header panel, a sidepanel controls on the left, and the main panel on the right - with the output.
- ▶ This layout is called **pageWithSidebar**. There are other layouts too - such as **basicPage** and **threePage**.

Inputs

The arguments are pretty typical among most of the widgets and are as follows:

inputId : This argument names the variable so it can be referred to in the server.R file

label : This argument gives a label to attach to the input so users know what it does

value : This argument gives the initial value to the widget when it is set up.

All the widgets should have sensible defaults for this argument.

Main Panel

- ▶ The final function is `mainPanel()`, which sets up the output window.
- ▶ HTML helper functions - make a little title `h3("...")`. Knowledge of HTML is very useful!
- ▶ There are several of these functions designed to generate HTML to go straight on the page; e.g. type `?p` at the console for the complete list.

Main Panel

- ▶ The other element that goes in `mainPanel()` is an area for handling reactive text or plots generated within the `server.R` file
- ▶ For example - a call to `textOutput()` with the name of the output as defined in `server.R`, in the upcoming "minimal case" examples.

Layout of a Shiny Web Application

`fluidPage` - **updated**

- ▶ To create a display with a fluid, unbroken layout, Shiny `ui.R` scripts need the function `fluidPage`. Shiny knows where to put your app's elements when it reads them in the `fluidPage` function.
- ▶ The following `ui.R` script creates a user-interface that has a title panel, a sidebar panel, and a main panel.
- ▶ Note that these elements are placed within the `fluidPage()` function.

Layout of a Shiny Web Application

```
# ui.R

shinyUI(fluidPage(
  titlePanel("title panel"),

  sidebarLayout(
    sidebarPanel("sidebar panel"),
    mainPanel("main panel")
  )
))
```

Layout of a Shiny Web Application

- ▶ `titlePanel` and `sidebarLayout` are the two most popular elements to add to `fluidPage`. They create a basic Shiny app with a sidebar.
 - ▶ `sidebarLayout` always takes two arguments:
 - ▶ `sidebarPanel` function output
 - ▶ `mainPanel` function output
- ▶ These functions place content in either the sidebar or the main panels.
- ▶ By default the sidebar appears on the left side of your apps display. To move the sidebar to the right, in `sidebarLayout` set position to right.

Layout of a Shiny Web Application

```
# ui.R

shinyUI(fluidPage(
  titlePanel("title panel"),

  sidebarLayout(position = "right",    #<- HERE
    sidebarPanel( "sidebar panel"),
    mainPanel("main panel")
  )
))
```

- ▶ `shinyServer(.....)` defines the bit of Shiny that's going to handle all the data.
- ▶ On the whole, two types of things go in here.
- ▶ **Reactive objects** (for example, data) are defined, which are then passed around as needed (for example, to different output instructions),
- ▶ Outputs are defined, such as graphs.

Shiny - Special Topics

- ▶ Conditional Panels - Outside Document
- ▶ Formatted Text - Outside Document
- ▶ HTML - Outside Document
- ▶ MathsJax

Sliders Widgets and Tabs

Customizing Sliders

- ▶ Shiny slider controls are extremely capable and customizable.
- ▶ Features supported include:
 - ▶ The ability to input both single values and ranges
 - ▶ Custom formats for value display (e.g for currency)
 - ▶ The ability to animate the slider across a range of values
- ▶ Slider controls are created by calling the `sliderInput` function.

Widgets

Worked Example - ex10

```
sidebarPanel(  
  selectInput("dataset", "Choose a dataset:",  
    choices = c("rock", "pressure", "cars")),  
  
  numericInput("obs", "Number of observations to view:", 10),  
  
  helpText("Note: while the data view will show only the specified  
    "number of observations, the summary will still be based",  
    "on the full dataset."),  
  
  submitButton("Update View")  
)
```

Tab Panels

- ▶ Tabsets are created by calling the `tabsetPanel` function with a list of tabs created by the `tabPanel` function.
- ▶ Each tab panel is provided a list of output elements which are rendered vertically within the tab.
- ▶ In this example we updated our Hello Shiny application to add a summary and table view of the data, each rendered on their own tab.

Widgets

Worked Example - ex11

```
mainPanel(  
  tabsetPanel(  
    tabPanel("Plot", plotOutput("plot")),  
    tabPanel("Summary", verbatimTextOutput("summary")),  
    tabPanel("Table", tableOutput("table"))  
  )  
)
```

Deploying Shiny apps

- ▶ The Shiny package itself is designed to run Shiny applications locally.
- ▶ To share Shiny applications with other R users, you can send them your application source as a GitHub gist, R package, or zip file.

Sharing Apps to Run Locally

- ▶ Once you've written your Shiny app, you can distribute it for others to run on their own computers; they can download and run Shiny apps with a single R command. All that this requires is that they have R and Shiny installed on their computers.
- ▶ If you want your Shiny app to be accessible over the web, so that users only need a web browser, see [Deploying Shiny Apps over the Web](#).

Gist

- ▶ One easy way is to put your code on `gist.github.com`, a code pasteboard service from **GitHub**.
- ▶ Both `server.R` and `ui.R` must be included in the same gist, and you must use their proper filenames.
- ▶ See *<http://gist.github.com/3239667>* for an example.

- ▶ Your recipient must have R and the Shiny package installed, and then running the app is as easy as entering the following command:

```
shiny::runGist('3239667')
```

- ▶ In place of '**3239667**' you will use your gists ID; or, you can use the entire URL of the gist (e.g. '*https://gist.github.com/3239667*').

Deploying Shiny

Advantages of using Gist

- ▶ Source code is easily visible by recipient (if desired)
- ▶ Easy to run (for R users)
- ▶ Easy to post and update

Cons

- ▶ Code is published to a third-party server

GitHub repository

- ▶ If your project is stored in a git repository on GitHub, then others can download and run your app directly. An example repository is at **`http://github.com/rstudio`**
- ▶ The following command will download and run the application:

```
shiny::runGitHub('shiny_example', 'rstudio')
```

In this example, the GitHub account is 'rstudio' and the repository is 'shiny example'; you will need to replace them with your account and repository name.

Github: Advantages

- ▶ Source code is easily visible by recipient (if desired)
- ▶ Easy to run (for R users)
- ▶ Very easy to update if you already use GitHub for your project
- ▶ Git-savvy users can clone and fork your repository

Disadvantages

- ▶ Developer must know how to use git and GitHub.
- ▶ Code is hosted by a third-party server.

Deploying Shiny

Making it into a Package

- ▶ If your Shiny app is useful to a broader audience, it might be worth the effort to turn it into an R package. Put your Shiny application directory under the `packages` inst directory, then create and export a function that contains something like this:

```
shiny::runApp(system.file('appdir',  
package='packagename'))
```

where `appdir` is the name of your apps subdirectory in `inst`, and **packagename** is the name of your package.

Deploying Shiny

Making it into a Package:

Advantages

- ▶ Publishable on CRAN
- ▶ Easy to run (for R users)

Disadvantages

- ▶ More work to set up
- ▶ Source code is visible by recipient (if not desired)

Deployment over the Web

- ▶ You can also deploy Shiny applications over the web, so that users need only a web browser and your applications URL.
- ▶ For this, you'll need a Linux server and our Shiny Server software.
- ▶ Shiny Server is free and open source, though in the future RStudio will offer a commercially licensed edition with additional features for larger organizations.
- ▶ RStudio also working on a subscription-based hosting service for Shiny.

Deploying Shiny apps : Shiny Server

Shiny Server

- ▶ Shiny Server is if you want to use your own server instead of hosting it on Rstudio's server (i.e. **glimmer**).
- ▶ This is really important for those who can't let their code or data out of their organization, or want more computational/storage resources than glimmer can offer, or need their apps to access their internal network.