> Dublin R

West of Ireland Data Science - Shiny Demonstrat

rstudio

RStudio

- 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.

Shiny - Slides Set 1

- Overview of Demonstration
- Resources (i.e. Shiny Tutorial Page)
- Minimal Examples
- Widgets
- A bit about JavaScript
- 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, youre 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!)*

Shiny Resources

Resources

- Shiny Tutorial (`http://rstudio.github.io/shiny/tutorial`)
- Chris Beeley's New Book (Sample Chapter Available)
- Stack-Overflow and GitHub

SNAP Scenarios Network for Alaska & Arctic Planning

About Tools and Data Methods Projects Resources

Exploring our future in a changing Arctic

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
Managing for the Future in a Rapidly Changing Arctic
A REPORT TO THE PRESIDENT

Matthew Leonawicz (SNAP - Uni. Alaska Fairbanks)

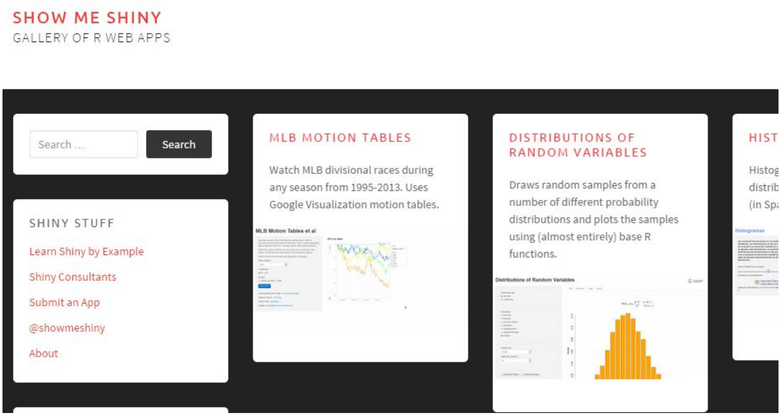
github.com/ua-snap/shiny-apps

twitter.com/leonawicz

github
SOCIAL CODING

github - code sharing

Showmeshiny.com


SHOW ME SHINY

GALLERY OF R WEB APPS

Search ... Search

SHINY STUFF

Learn Shiny by Example

Shiny Consultants

Submit an App

@showmeshiny

About

MLB MOTION TABLES

Watch MLB divisional races during any season from 1995-2013. Uses Google Visualization motion tables.

DISTRIBUTIONS OF RANDOM VARIABLES

Draws random samples from a number of different probability distributions and plots the samples using (almost entirely) base R functions.

HIST

Histog

distrib

(in Spa

Shiny - Part 2

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**

Shiny Basics

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.

ui.R

- 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.

server.R

- `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.

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