Web Application Tools for Statistics Using R and Shiny

Rikkyo University

June 16, 2016

Jimmy Doi jdoi@calpoly.edu

California Polytechnic State University
San Luis Obispo
Department of Statistics

Introduction

Introduction

Based on the following work:

Web Application Teaching Tools for Statistics Using R and Shiny*

Jimmy Doi[1], Gail Potter[1,2], Jimmy Wong[1,3], Irvin Alcaraz[1,4], and Peter Chi[1,5]

[1]Department of Statistics, California Polytechnic State University, San Luis Obispo
[2]The EMMES Corporation (Rockville, MD)
[3]Food and Drug Administration, Center for Drug Evaluation and Research
[4]OpenX
[5]Department of Mathematics and Computer Science, Ursinus College

*(March 2016) *Technology Innovations in Statistics Education*, 9(1)

Introduction

- There is a large collection of applet tools on the web
  - Rossman/Chance Applet Collection
  - Statistics Online Computational Resource
- Eventually an instructor/researcher can come across a problem in finding an existing applet to perfectly suit his/her needs
- One can try to customize an existing applet ...
  - this requires access to original source code (not always available)
  - even if available, customization requires fluency in source code language
- If the desired functionality of application is novel (e.g., based on newly proposed research), existing applets are most likely unsuitable

Introduction

- In these situations the instructor/researcher is left to consider building his/her own web-based tool applet
- Not a trivial task – can require knowledge in
  - Java, Javascript
  - HTML
  - CSS
  - PHP
  - Server-side management
- This burden can be a sufficient obstacle, keeping an instructor/researcher from creating applets

Introduction

- An alternative method to create web-based tool applications is provided by `Shiny` – a web application framework for `R`
- It is not uncommon for instructors/researchers to build their own tools via algorithms written in `R`
- It is not difficult to convert existing `R` algorithms into `Shiny` web applications – known simply as '`Shiny` apps'
- With `Shiny`, one can build applications that are interactive, dynamic, user-friendly, visually appealing, and, with similar functionality to Java/Javascript applets; the only requirement is some familiarity in `R`

Motivating example

Motivating example

- Introductory Statistics Course
- Topic: Behavior of a simple stochastic process
  - Binary outcomes (H or T) where $P(\text{H}) = P(\text{T}) = 0.5$
- Popular class activity – coin flipping (find the fake sequence)

Sequence A

```
T T H H T H T T H H H T T H H T T H T H H H T H T
H H T T T H T T H T H H T H T T H T H H T H T H H
```

Sequence B

```
H H H H H H T T H H H H H T H T T T T T H H T H T
H H H T T T T H H H H T T H H T H H T H H T T H T
```

Motivating example

- Introductory Statistics Course
- Topic: Behavior of a simple stochastic process
  - Binary outcomes (H or T) where $P(\text{H}) = P(\text{T}) = 0.5$
- Popular class activity – coin flipping (find the fake sequence)

Sequence A

T T H H T H T T H H H T T H H T T H T H H H T H T
H H T T T H T T H T H H T H T T H T H H T H T H H

Sequence B

H H H H H H T T H H H H H T H T T T T T H H T H T
H H H T T T T H H H H T T H H T H H T H H T T H T

Motivating example

- Tried to find an applet that simulated coin flips and allowed user to identify head/tail runs of a particular length
- No such applet could be found
- I made two R functions to accomplish this task
  - `flip.gen()` – simulates the outcomes of a fair coin flipped a given number of times and determines run lengths
  - `plot.flips()` – displays the simulation outcomes
- In-class presentation using the R console and these functions
  - Set $n = 100$, show randomizations, highlight various run lengths
  - Set $n = 200$, show randomizations, highlight various run lengths

Motivating example

‘Traditional Method’ (working from the console) – disadvantages:

- Must work from the console
- Awkward pauses and choppy (delays can detract from the presentation content)
- Difficulties arise when having others compile your **R** code

Alternative Presentation Method using **Shiny**

- Longest Run of Heads or Tails App

Motivating example

Presentation with `Shiny` – advantages:

- A more fluid presentation
  - Eliminates the awkward pauses – no need to show console
  - All adjustments done within the app itself by moving sliders/clicking buttons
  - Updates are virtually instantaneous
- Improved accessibility
  - Users can access the app outside of the brief exposure during the presentation
  - No need for users to work with R directly (avoiding many potential problems) – just launch a web browser
- A means to convert existing R algorithms into web-applications
  - `Shiny` can facilitate development of new web-based tools or applications in a very feasible manner
  - Helpful when introducing concepts not in the standard curriculum or are based on recent research

Shiny App Teaching Tools Collection

Shiny App Teaching Tools Collection

- Currently our *Shiny App Teaching Tools Collection* has 18 apps
  - Wide range of topics – coin flipping, random variable generator, hierarchical models, ...
  - Just about every type of `Shiny` layout and widget can be found in our apps

- Cal Poly `Shiny` Site: www.statistics.calpoly.edu/shiny

- All `Shiny` source code available at: gist.github.com/calpolystat

Shiny App Teaching Tools Collection

App #	Author	App Name
1	Alcaraz	Correlation and Regression Game
2		Multiple Regression Visualization
3		Probability Distribution Viewer
4	Chi	Gambler's Ruin
5		Random Variable Generation
6	Doi	Length/Coverage Optimal Confidence Intervals
7		Benford's Law: Sequences
8		Benford's Law: Data Examples
9		Chaos Game: Two Dimensions
10		Chaos Game: Three Dimensions
11		Longest Run of Heads or Tails
12	Potter	Testing Violation of the Constant Variance Condition for ANOVA
13		Maximum Likelihood Estimation for the Binomial Distribution
14		Sampling Distributions of Various Statistics
15	Wong	t-test with diagnostics
16		Performance of the Wilcoxon-Mann-Whitney Test vs. t-test
17		Heaped Distribution Estimation
18		Hierarchical Models

`Shiny` app examples

- Length/Coverage Optimal (LCO) Confidence Intervals
- Benford's Law: Data Examples
- Multiple Regression Visualization

Shiny app examples

Length/Coverage Optimal (LCO) Confidence Intervals

Schilling, M., and Doi, J. (2014) "A Coverage Probability Approach to Finding an Optimal Binomial Confidence Procedure". The American Statistician, 68, 133–145

- Statistical research projects often involve the creation of novel computational algorithms (e.g., written in R)
- Journal publications based on such research often include references to these algorithms
  - Code attached in Appendix section
  - Code location specified by URL citation
  - Include in publication: "Please contact author for code."
- Problem: General user will not go to the trouble to access the computational algorithms from your research.
- If the code was converted into a **Shiny** app, and app's URL is cited in the paper/website, you will reach a greater audience.
  - LCO **Shiny** app completed shortly after paper publication, its URL was mentioned in our response to "Letter to the Editor"

Shiny app examples

- Length/Coverage Optimal (LCO) Confidence Intervals
- Benford's Law: Data Examples [uses webscraping]
- Multiple Regression Visualization

Shiny basics

Shiny basics

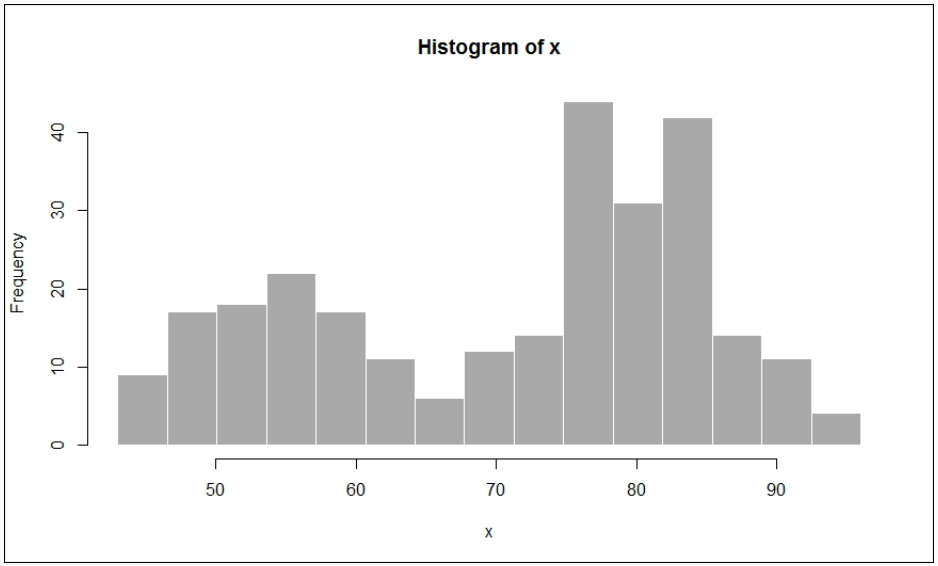
- A Shiny app is comprised of two files: `UI.R` and `SERVER.R` *
  - `UI.R` – instructions for the layout and appearance of the app
  - `SERVER.R` – (usually) the app's computational components
- The dynamic and interactive nature of a Shiny app is made possible through the ongoing interplay that occurs between `UI.R` and `SERVER.R`.

*As of version 0.10.2, Shiny allows for single-file applications where the components of `UI.R` and `SERVER.R` can be stored in one file called `APP.R`.

Shiny **basics:** HISTOGRAM.R

HISTOGRAM.R

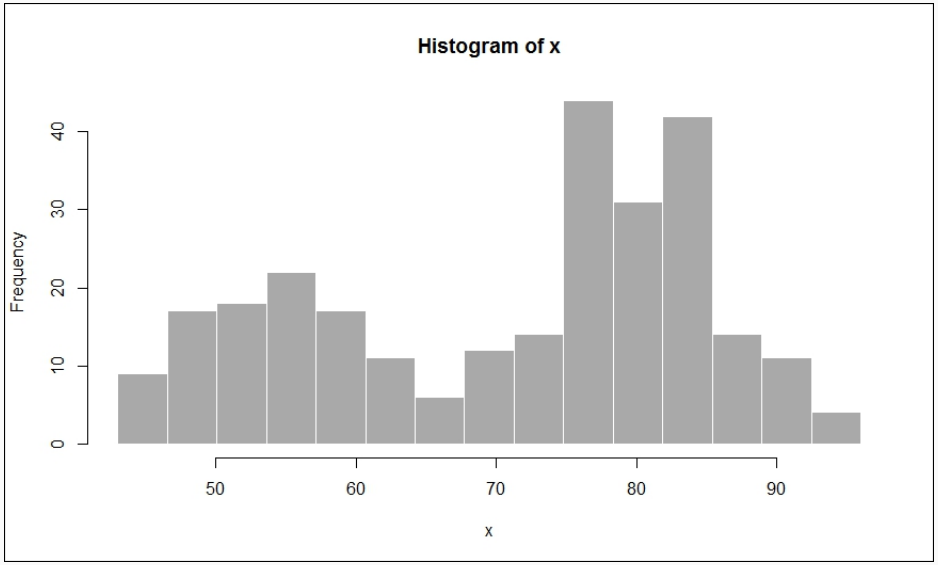
```
bin.num <- 15
x <- faithful[, 2]
bins <- seq(min(x),max(x),
   length.out = bin.num+1)
hist(x, breaks = bins,
     col = "darkgray",
     border = "white")
```

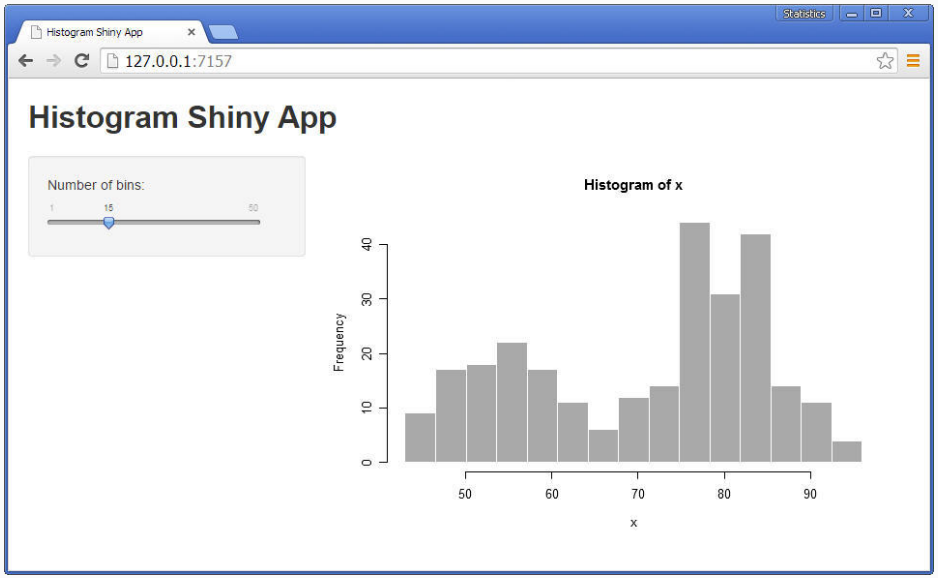


Shiny **basics:** HISTOGRAM.R

HISTOGRAM.R

```
bin.num <- 15
x <- faithful[, 2]
bins <- seq(min(x),max(x),
   length.out = bin.num+1)
hist(x, breaks = bins,
     col = "darkgray",
     border = "white")
```





Shiny basics: SERVER.R and UI.R

HISTOGRAM.R

```
bin.num <- 15
x <- faithful[, 2]
bins <- seq(min(x),max(x),
   length.out = bin.num+1)
hist(x, breaks = bins,
     col = "darkgray",
     border = "white")
```

SERVER.R

```
shinyServer(function(input,output){
 output$distPlot <- renderPlot({
   x <- faithful[, 2]
   bins <- seq(min(x), max(x),
      length.out = input$numBin+1)
   hist(x, breaks = bins,
        col = "darkgray",
        border = "white")
 })
})
```

Shiny basics: SERVER.R and UI.R

HISTOGRAM.R

```
bin.num <- 15
x <- faithful[, 2]
bins <- seq(min(x),max(x),
   length.out = bin.num+1)
hist(x, breaks = bins,
     col = "darkgray",
     border = "white")
```

SERVER.R

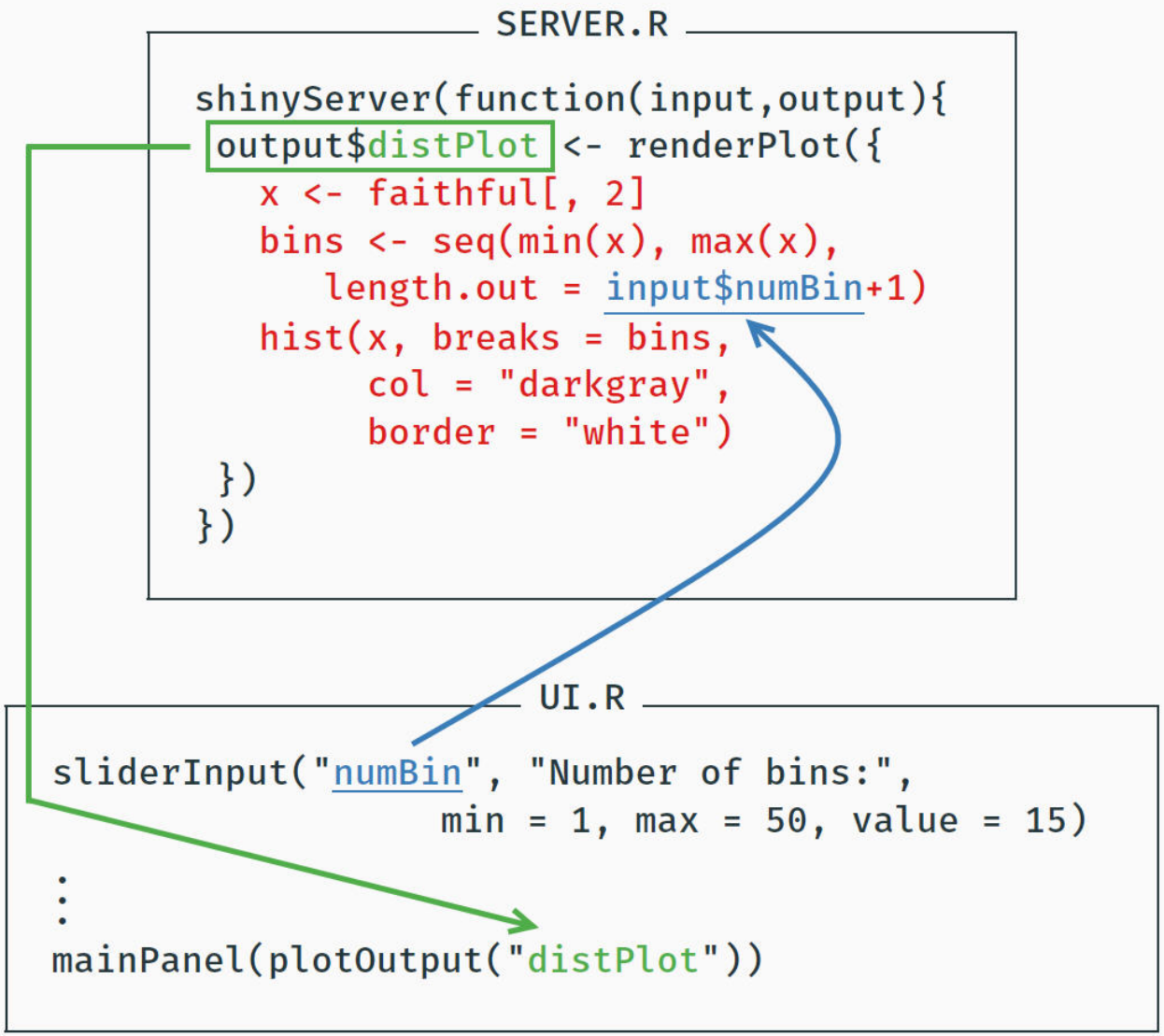
```
shinyServer(function(input,output){
 output$distPlot <- renderPlot({
   x <- faithful[, 2]
   bins <- seq(min(x), max(x),
      length.out = input$numBin+1)
   hist(x, breaks = bins,
        col = "darkgray",
        border = "white")
 })
})
```

UI.R

```
sliderInput("numBin", "Number of bins:",
                  min = 1, max = 50, value = 15)

:
:
mainPanel(plotOutput("distPlot"))
```

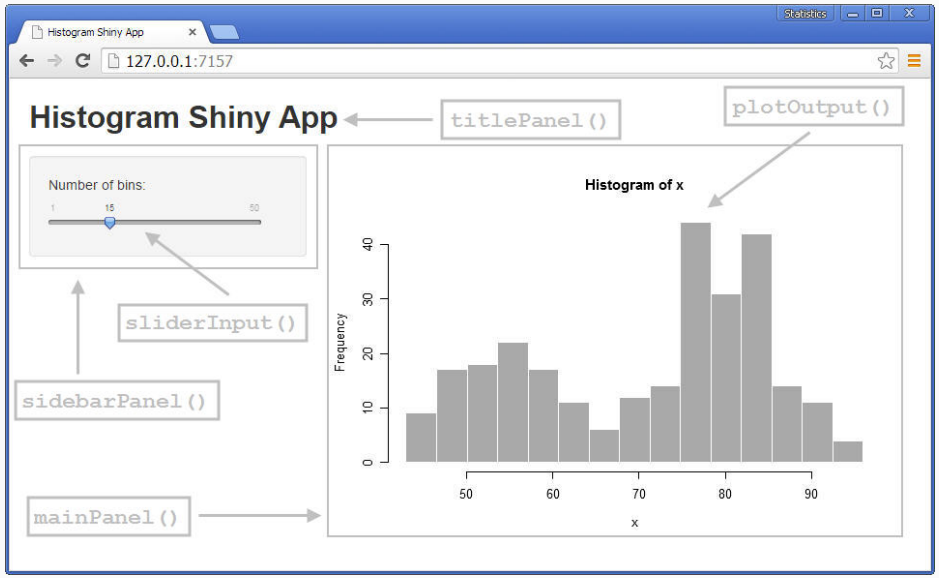
Shiny basics: SERVER.R and UI.R



Shiny basics: UI.R

UI.R

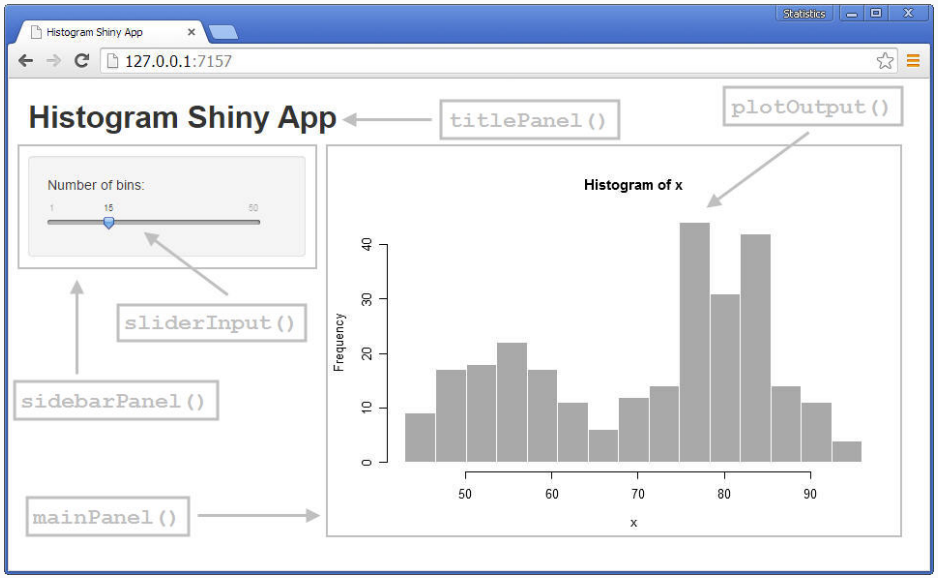
```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – shinyUI(fluidPage())

UI.R

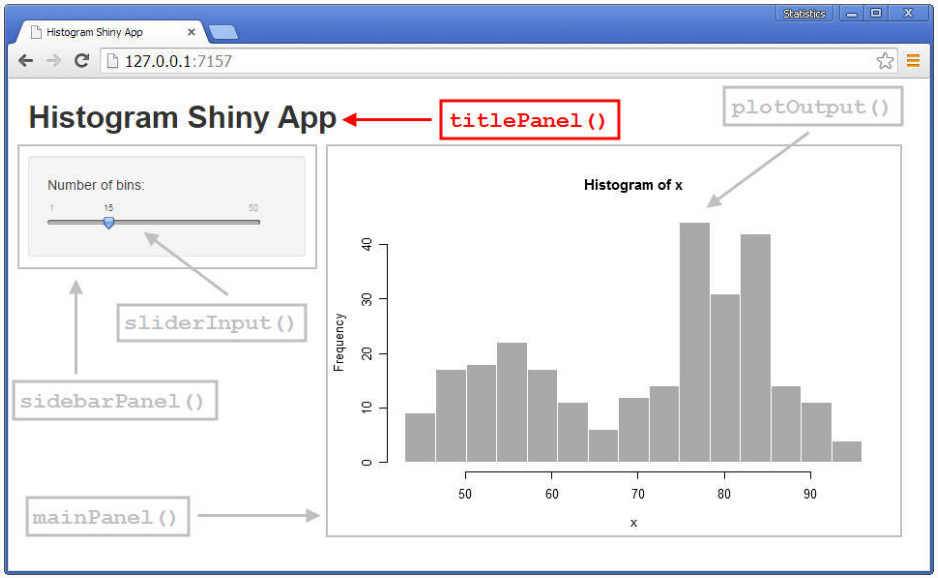
```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – titlePanel()

UI.R

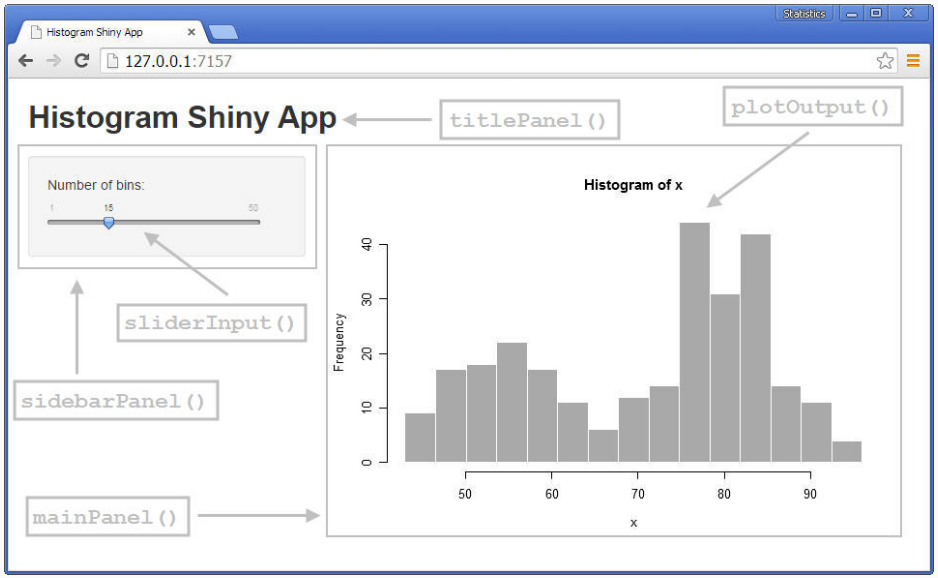
```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – sidebarLayout()

UI.R

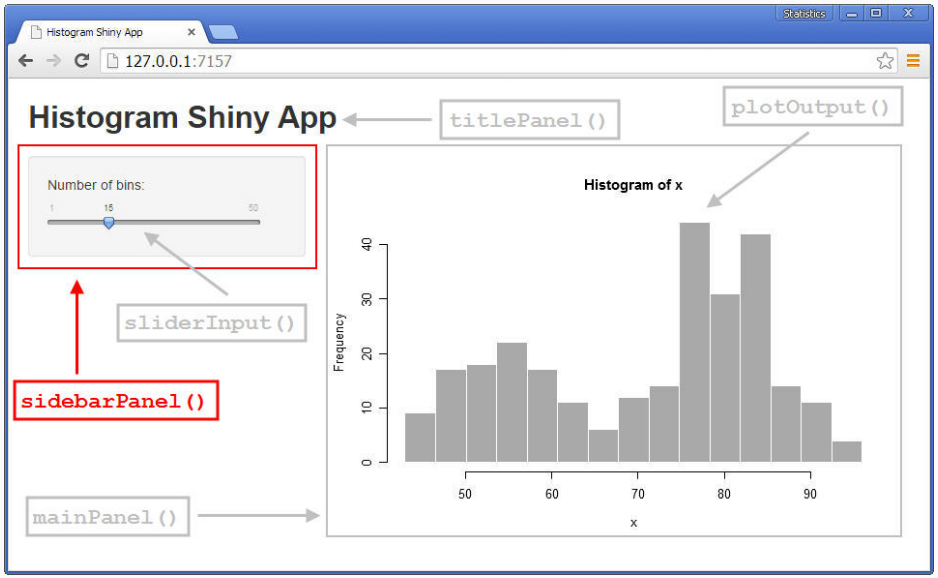
```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – sidebarPanel()

UI.R

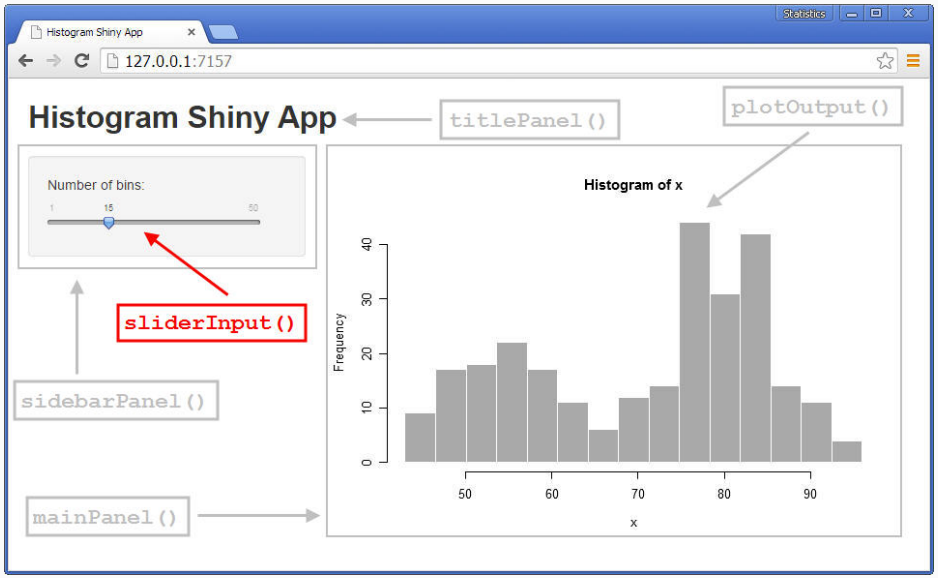
```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – sliderInput()

UI.R

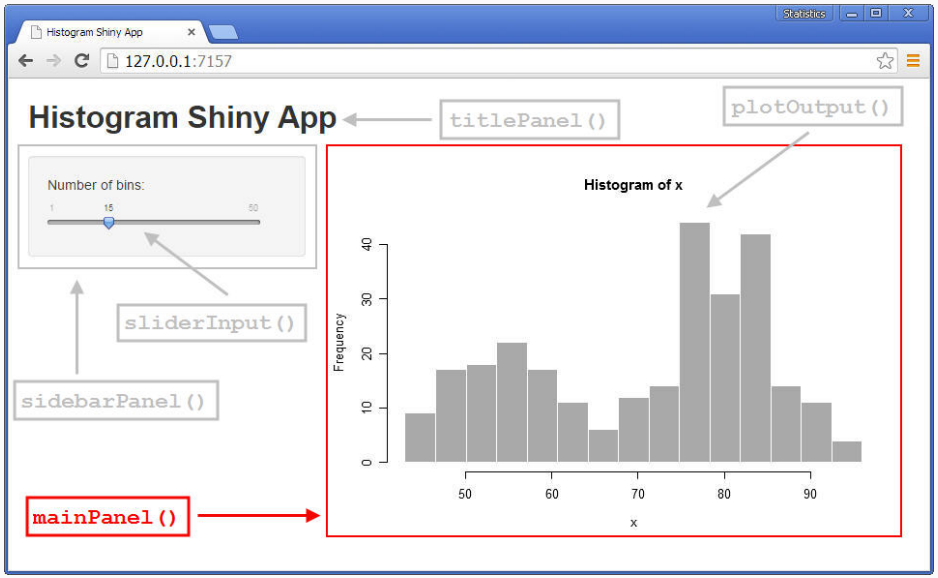
```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – mainPanel()

UI.R

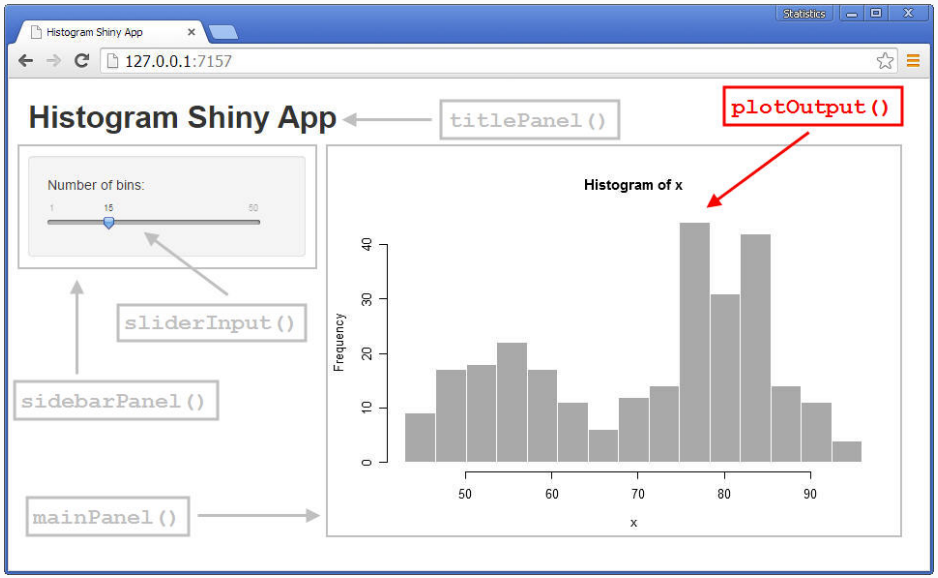
```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – plotOutput()

UI.R

```
shinyUI(fluidPage(
  titlePanel("Histogram Shiny App"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("numBin",
                  "Number of bins:",
                  min = 1, max = 50,
                  value = 15)),
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```



Shiny basics: UI.R – Widgets

- A widget is a web element that users can interact with
- Each widget is an input device that sends feedback to the Shiny app
- shiny.rstudio.com/gallery/widget-gallery.html

Launching and deploying `Shiny` apps

Launching and deploying `Shiny` apps

Launching apps (locally accessible on a PC)

- Use the RStudio software (works very well with `Shiny`)
  - Edit `UI.R` and `SERVER.R`
  - Use CONTROL-SHIFT-ENTER to launch app
  - No need to close app window to re-edit/re-launch

- Store your files at a cloud-based service like Dropbox
  - allows user to revert back to a code version between backups

Launching and deploying `Shiny` apps

Deploying apps (publicly accessible via a server) – e.g., Benford App

- RStudio server at shinyapps.io
  - Choose from several accounts based on tiered pricing system
  - Free account available but with restrictions

- Open Shiny Server and Shiny Server Pro* (both for Linux servers)

  *(annual fee starts at $10,000 USD → 約107万円)

- Cal Poly **Shiny** Server Pro (100 concurrent users)*

  *(Our equivalent annual fee: $25,000 USD → 約268万円)

- RStudio Academic Pricing Policy
  (Research → 50% discount, Teaching → 100% discount)

- I am happy to work with your IT staff to discuss install process

RStudio Academic Pricing Policy

Stat 418, Analysis of Cross-Classified Data
Winter 2016, Cal Poly State University San Luis Obispo

Class Info:	Section 01, TR 2:10 – 4:00pm (Bldg 14, Room 247)
Instructor:	Dr. Jimmy A. Doi, Department of Statistics Email: `jdoi@calpoly.edu` ← *best way to reach me*
Office Hours:	MTWR 12:10 – 1:00pm (and by appointment)
Office Info:	Faculty Office Building East 25 - 108
Text:	*An Introduction to Categorical Data Analysis* (SECOND edition) by Alan Agresti, 2007
Computer Access:	**We will use the following software –** **SAS, R, RStudio, Shiny Apps** • SAS is available on many on-campus computers • Download R at `cran.r-project.org` • Download RStudio at `www.rstudio.com/products/rstudio/download` • URL for all Shiny Apps will be announced in class

Shiny at Academic Institutions

Shiny Server usage worldwide at academic institutions:

- Hundreds of universities worldwide are using Shiny Server
- Currently several universities in Japan are using Shiny Server

Example: Scenarios Network for Alaska and Arctic Planning (SNAP)

SNAP is part of the International Arctic Research Center at the University of Alaska Fairbanks.

- Daily Precipitation for Alaska

Utility of `Shiny` for Rikkyo University

`Shiny` can be useful for any department/group at Rikkyo.
For example:

- Center for Statistics and Information
- College of Economics
- College of Business

shiny.rikkyo.ac.jp/CSI
shiny.rikkyo.ac.jp/Econ
shiny.rikkyo.ac.jp/Bus

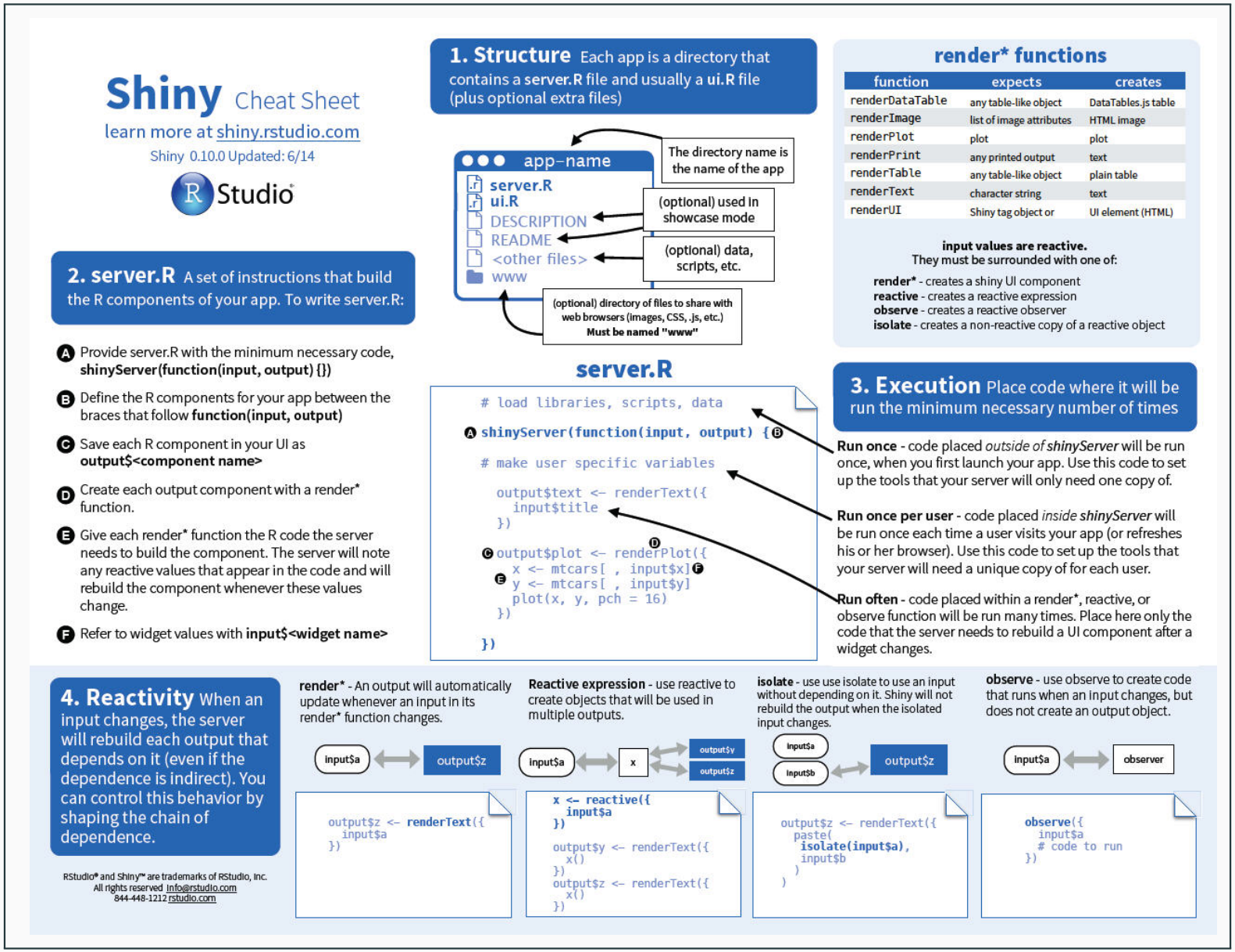
shiny.rikkyo.ac.jp/yamaguchi
shiny.rikkyo.ac.jp/ohashi
shiny.rikkyo.ac.jp/tanno

`Shiny` resources

Resources

- A great starting point: The **Shiny** Tutorial at RStudio:
  shiny.rstudio.com/tutorial
- A large gallery (over 150) of various **Shiny** apps:
  www.showmeshiny.com
- Using **Shiny** in **R Markdown**:
  rmarkdown.rstudio.com/authoring_shiny.html
- Reactive Programming in **Shiny**:
  shiny.rstudio.com/articles/reactivity-overview.html
- Adding Google Analytics to **Shiny**:
  shiny.rstudio.com/articles/google-analytics.html
- **Shiny** Cheat Sheet: www.rstudio.com/resources/cheatsheets

Resources: Shiny Cheat Sheet



Shiny Cheat Sheet

learn more at shiny.rstudio.com

Shiny 0.10.0 Updated: 6/14

RStudio

1. Structure Each app is a directory that contains a server.R file and usually a ui.R file (plus optional extra files)

app-name
- server.R
- ui.R
- DESCRIPTION — (optional) used in showcase mode
- README
- <other files> — (optional) data, scripts, etc.
- www — (optional) directory of files to share with web browsers (images, CSS, .js, etc.) Must be named "www"

The directory name is the name of the app

2. server.R A set of instructions that build the R components of your app. To write server.R:

A. Provide server.R with the minimum necessary code, **shinyServer(function(input, output) {})**

B. Define the R components for your app between the braces that follow **function(input, output)**

C. Save each R component in your UI as **output$<component name>**

D. Create each output component with a render* function.

E. Give each render* function the R code the server needs to build the component. The server will note any reactive values that appear in the code and will rebuild the component whenever these values change.

F. Refer to widget values with **input$<widget name>**

server.R

```
# load libraries, scripts, data

shinyServer(function(input, output) {

  # make user specific variables

  output$text <- renderText({
    input$title
  })

  output$plot <- renderPlot({
    x <- mtcars[ , input$x]
    y <- mtcars[ , input$y]
    plot(x, y, pch = 16)
  })

})
```

render* functions

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)

input values are reactive.
They must be surrounded with one of:

render* - creates a shiny UI component
reactive - creates a reactive expression
observe - creates a reactive observer
isolate - creates a non-reactive copy of a reactive object

3. Execution Place code where it will be run the minimum necessary number of times

Run once - code placed *outside of* ***shinyServer*** will be run once, when you first launch your app. Use this code to set up the tools that your server will only need one copy of.

Run once per user - code placed *inside* ***shinyServer*** will be run once each time a user visits your app (or refreshes his or her browser). Use this code to set up the tools that your server will need a unique copy of for each user.

Run often - code placed within a render*, reactive, or observe function will be run many times. Place here only the code that the server needs to rebuild a UI component after a widget changes.

4. Reactivity When an input changes, the server will rebuild each output that depends on it (even if the dependence is indirect). You can control this behavior by shaping the chain of dependence.

render* - An output will automatically update whenever an input in its render* function changes.

input$a → output$z

```
output$z <- renderText({
  input$a
})
```

Reactive expression - use reactive to create objects that will be used in multiple outputs.

input$a → x → output$y, output$z

```
x <- reactive({
  input$a
})

output$y <- renderText({
  x()
})
output$z <- renderText({
  x()
})
```

isolate - use use isolate to use an input without depending on it. Shiny will not rebuild the output when the isolated input changes.

input$a, input$b → output$z

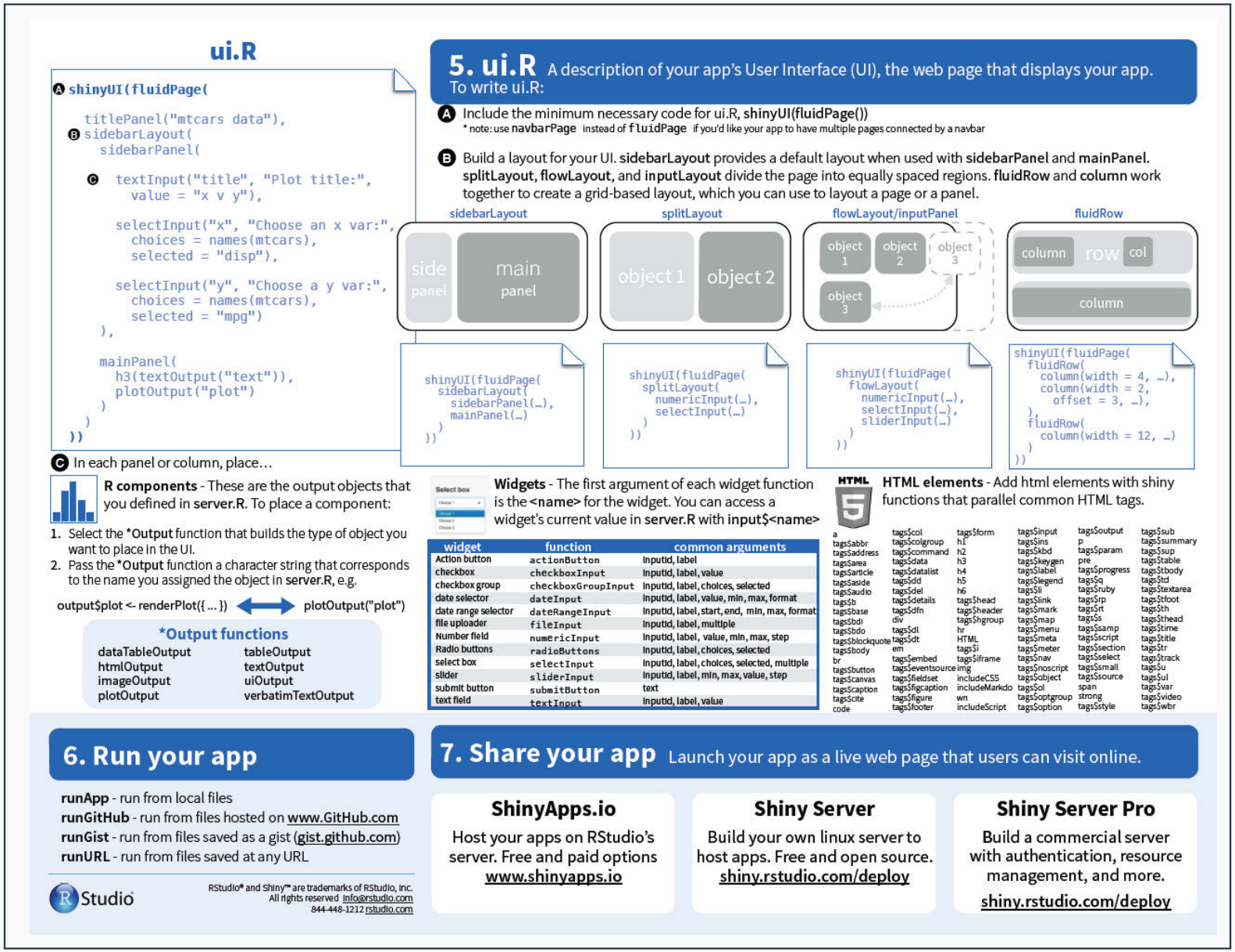
```
output$z <- renderText({
  paste(
    isolate(input$a),
    input$b
  )
)
```

observe - use observe to create code that runs when an input changes, but does not create an output object.

input$a → observer

```
observe({
  input$a
  # code to run
})
```

Resources: Shiny Cheat Sheet



ui.R

```
A shinyUI(fluidPage(

    titlePanel("mtcars data"),
  B sidebarLayout(
      sidebarPanel(

    C   textInput("title", "Plot title:",
          value = "x v y"),

        selectInput("x", "Choose an x var:",
          choices = names(mtcars),
          selected = "disp"),

        selectInput("y", "Choose a y var:",
          choices = names(mtcars),
          selected = "mpg")
      ),

      mainPanel(
        h3(textOutput("text")),
        plotOutput("plot")
      )
    )
))
```

C In each panel or column, place…

R components - These are the output objects that you defined in **server.R**. To place a component:

1. Select the ***Output** function that builds the type of object you want to place in the UI.
2. Pass the ***Output** function a character string that corresponds to the name you assigned the object in **server.R**, e.g.

output$plot <- renderPlot({ … }) ⟷ plotOutput("plot")

***Output functions**

dataTableOutput	tableOutput
htmlOutput	textOutput
imageOutput	uiOutput
plotOutput	verbatimTextOutput

5. ui.R A description of your app's User Interface (UI), the web page that displays your app.

To write ui.R:

A Include the minimum necessary code for ui.R, **shinyUI(fluidPage())**
* note: use **navbarPage** instead of **fluidPage** if you'd like your app to have multiple pages connected by a navbar

B Build a layout for your UI. **sidebarLayout** provides a default layout when used with **sidebarPanel** and **mainPanel**. **splitLayout**, **flowLayout**, and **inputLayout** divide the page into equally spaced regions. **fluidRow** and **column** work together to create a grid-based layout, which you can use to layout a page or a panel.

sidebarLayout


```
shinyUI(fluidPage(
  sidebarLayout(
    sidebarPanel(…),
    mainPanel(…)
  )
))
```

splitLayout


```
shinyUI(fluidPage(
  splitLayout(
    numericInput(…),
    selectInput(…)
  )
))
```

flowLayout/inputPanel


```
shinyUI(fluidPage(
  flowLayout(
    numericInput(…),
    selectInput(…),
    sliderInput(…)
  )
))
```

fluidRow


```
shinyUI(fluidPage(
  fluidRow(
    column(width = 4, …),
    column(width = 2,
      offset = 3, …),
  ),
  fluidRow(
    column(width = 12, …)
  )
))
```


Widgets - The first argument of each widget function is the **<name>** for the widget. You can access a widget's current value in **server.R** with **input$<name>**

widget	function	common arguments
Action button	`actionButton`	inputId, label
checkbox	`checkboxInput`	inputId, label, value
checkbox group	`checkboxGroupInput`	inputId, label, choices, selected
date selector	`dateInput`	inputId, label, value, min, max, format
date range selector	`dateRangeInput`	inputId, label, start, end, min, max, format
file uploader	`fileInput`	inputId, label, multiple
Number field	`numericInput`	inputId, label, value, min, max, step
Radio buttons	`radioButtons`	inputId, label, choices, selected
select box	`selectInput`	inputId, label, choices, selected, multiple
slider	`sliderInput`	inputId, label, min, max, value, step
submit button	`submitButton`	text
text field	`textInput`	inputId, label, value

HTML elements - Add html elements with shiny functions that parallel common HTML tags.

a	tags$col	tags$form	tags$input	tags$output	tags$sub
tags$abbr	tags$colgroup	h1	tags$ins	p	tags$summary
tags$address	tags$command	h2	tags$kbd	tags$param	tags$sup
tags$area	tags$data	h3	tags$keygen	pre	tags$table
tags$article	tags$datalist	h4	tags$label	tags$progress	tags$tbody
tags$aside	tags$dd	h5	tags$legend	tags$q	tags$td
tags$audio	tags$del	h6	tags$li	tags$ruby	tags$textarea
tags$b	tags$details	tags$head	tags$link	tags$rp	tags$tfoot
tags$base	tags$dfn	tags$header	tags$mark	tags$rt	tags$th
tags$bdi	div	tags$hgroup	tags$map	tags$s	tags$thead
tags$bdo	tags$dl	hr	tags$menu	tags$samp	tags$time
tags$blockquote	tags$dt	HTML	tags$meta	tags$script	tags$title
tags$body	em	tags$i	tags$meter	tags$section	tags$tr
br	tags$embed	tags$iframe	tags$nav	tags$select	tags$track
tags$button	tags$eventsource	img	tags$noscript	tags$small	tags$u
tags$canvas	tags$fieldset	includeCSS	tags$object	tags$source	tags$ul
tags$caption	tags$figcaption	includeMarkdo	tags$ol	span	tags$var
tags$cite	tags$figure	wn	tags$optgroup	strong	tags$video
code	tags$footer	includeScript	tags$option	tags$style	tags$wbr

6. Run your app

runApp - run from local files
runGitHub - run from files hosted on www.GitHub.com
runGist - run from files saved as a gist (gist.github.com)
runURL - run from files saved as any URL

7. Share your app Launch your app as a live web page that users can visit online.

ShinyApps.io
Host your apps on RStudio's server. Free and paid options
www.shinyapps.io

Shiny Server
Build your own linux server to host apps. Free and open source.
shiny.rstudio.com/deploy

Shiny Server Pro
Build a commercial server with authentication, resource management, and more.
shiny.rstudio.com/deploy

Resources: dplyr and tidyr Cheat Sheet (日本語)

dplyrとtidyrを使ったデータラングリングチートシート

RStudio

文法 - ラングリングに役立つおまじない

dplyr::tbl_df(iris)
データフレームからテーブルへ変換。テーブルの情報は画面に収まるよう表示され、データフレームよりも扱いやすい。

```
Source: local data frame [150 x 5]

  Sepal.Length Sepal.Width Petal.Length
1          5.1         3.5          1.4
2          4.9         3.0          1.4
3          4.7         3.2          1.3
4          4.6         3.1          1.5
5          5.0         3.6          1.4
..         ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::glimpse(iris)
テーブルの要約を確認

utils::View(iris)
スプレッドシート形式でデータを確認（Vは大文字）

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::%>%
左辺のオブジェクトを、右辺関数の第一変数（または .で指定した引数）として渡す。

```
x %>% f(y)     は  f(x, y)     と同じ
y %>% f(x, ., z)  は  f(x, y, z)  と同じ
```

%>% を使うとコードが読みやすくなる。例えば:

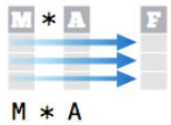
```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - Rでのデータ整形初歩

tidyのデータセットでは:

各変数は各列に保存される
各オブザベーションは各行に保存される

Tidy dataは、Rのベクトル操作を補完する。Rは変数(列)を操作しても、自動的にオブザベーション(行)を保つ。



データを変形する - データセットのレイアウト変更

tidyr::gather(cases, "year", "n", 2:4)
各列を行に展開
（ワイド型からロング型へ変換）

tidyr::spread(pollution, size, amount)
各行を列に展開
（ロング型からワイド型へ変換）

tidyr::separate(storms, date, c("y", "m", "d"))
一列を、複数列に分割

tidyr::unite(data, col, ..., sep)
複数列を、一列に集約・結合

dplyr::data_frame(a = 1:3, b = 4:6)
複数のベクトルを結合し、データフレーム化(最適化)
dplyr::arrange(mtcars, mpg)
列の値で、行を並び替え(昇順)
dplyr::arrange(mtcars, desc(mpg))
列の値で、行を並び替え(降順)
dplyr::rename(tb, y = year)
データフレームの列名を変更

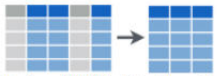
行の一部を取り出す

dplyr::filter(iris, Sepal.Length > 7)
条件式の評価に合う行を抽出
dplyr::distinct(iris)
重複行を削除
dplyr::sample_frac(iris, 0.5, replace = TRUE)
無作為に指定割合のサンプルを抽出
dplyr::sample_n(iris, 10, replace = TRUE)
無作為にn行のサンプルを抽出
dplyr::slice(iris, 10:15)
Select rows by position. 行を位置で指定し、選択する
dplyr::top_n(storms, 2, date)
上位のnエントリーを並替え抽出(グループ化済ならグループ毎に)

Rの演算子 - ?Comparison, ?base::Logic

<	より小さい	!=	等しくない
>	より大きい	%in%	グループメンバシップ
==	等しい	is.na	NA（欠損値）である
<=	等しいか、より小さい	!is.na	NA（欠損値）でない
>=	等しいか、より大きい	&,|,!,xor,any,all	ブール型演算子

列の一部を取り出す



dplyr::select(iris, Sepal.Width, Petal.Length, Species)
列を、名前またはヘルパー関数を使って抽出

Selectのヘルパー関数 - ?select

select(iris, contains("."))
列名にある文字列を含む列を選択
select(iris, ends_with("Length"))
列名がある文字列で終わる列を選択
select(iris, everything())
Select every column. 全ての列を選択
select(iris, matches(".t."))
列名がある正規表現に一致する列を選択
select(iris, num_range("x", 1:5))
列名がx1, x2, x3, x4, x5である列を選択
select(iris, one_of(c("Species", "Genus")))
列名が列挙した名前に一致する列を選択
select(iris, starts_with("Sepal"))
列名がある文字列で始まる列を選択
select(iris, Sepal.Length:Petal.Width)
Sepal.LengthとPetal.Widthの間にある全列を選択（二列を含む）
select(iris, -Species)
Species以外の全列を選択

devtools::install_github("rstudio/EDAWR") for data sets
Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15
Translated by Tomo Masuda // @tomo_makes

Resources: dplyr and tidyr Cheat Sheet (日本語)

データを要約する

dplyr::summarise(iris, avg = mean(Sepal.Length))
複数行のデータを一行に要約
dplyr::summarise_each(iris, funs(mean))
summary関数を各列に適用
dplyr::count(iris, Species, wt = Sepal.Length)
変数の個数を重複を除き数え上げ(重み付け可)

summarise()は、summary関数を使う。summary関数は、複数値のベクトル入力に対し、ひとつの値を返す
例:

dplyr::first
ベクトル1つ目の値
dplyr::last
ベクトル最後の値
dplyr::nth
ベクトルn番目の値
dplyr::n
ベクトルの要素数
dplyr::n_distinct
ベクトルの重複を省いた要素数
IQR
ベクトルのIQR
min
ベクトルの最小値
max
ベクトルの最大値
mean
ベクトルの平均値
median
ベクトルの中間値
var
ベクトルの不偏分散
sd
ベクトルの不偏標準偏差

データをグルーピングする

dplyr::group_by(iris, Species)
Speciesが同値のものは1行にグループ化
dplyr::ungroup(iris)
データフレームからグループ化情報を削除

iris %>% group_by(Species) %>% summarise(…)
各グループ毎に要約を計算

新しい列を追加する

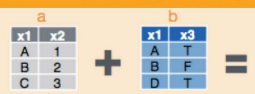
dplyr::mutate(iris, sepal = Sepal.Length + Sepal. Width)
1つ以上の列を計算・生成し追加
dplyr::mutate_each(iris, funs(min_rank))
window関数を各列に適用
dplyr::transmute(iris, sepal = Sepal.Length + Sepal. Width)
1つ以上の新しい列を計算・生成し元列を削除

mutate()は、window関数を使う。window関数は、複数値のベクトル入力に対し、別のベクトルを返す
例:

dplyr::lead
前にずらす(先頭消し末尾にNA)
dplyr::lag
後ろにずらす(末尾消し先頭にNA)
dplyr::dense_rank
順位(同着は順位飛ばさず)
dplyr::min_rank
順位(同着は最小順位へ合せ次の順位飛ばす)
dplyr::percent_rank
0〜1に標準化した順位(順位%)
dplyr::row_number
順位(同着なし、上から順。行番号)
dplyr::ntile
n個の群に分割
dplyr::between
値がaとbの間か否か
dplyr::cume_dist
順位の%の積み上げ
dplyr::cumall
部分条件判定(all)
dplyr::cumany
部分条件判定(any)
dplyr::cummean
部分平均
cumsum
部分和
cummax
部分最大
cummin
部分最小
cumprod
部分積
pmax
要素間の最大
pmin
要素間の最小

iris %>% group_by(Species) %>% mutate(…)
グループ毎に新しい変数を計算

データセットを結合する



Mutating Joins

dplyr::left_join(a, b, by = "x1")
bをaに対応付け、結合する(aが全て残る)

dplyr::right_join(a, b, by = "x1")
aをbに対応付け、結合する(bが全て残る)

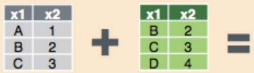
dplyr::inner_join(a, b, by = "x1")
内部結合。a, b両方にある行のみ残す。

dplyr::full_join(a, b, by = "x1")
外部結合。全ての値と行を残す。

Filtering Joins

dplyr::semi_join(a, b, by = "x1")
bに対応する行をもつaの行を抽出

dplyr::anti_join(a, b, by = "x1")
bに対応する行を持たないaの行を抽出



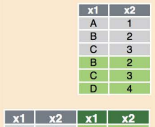
Set Operations

dplyr::intersect(y, z)
yとz両方に現れる行を抽出

dplyr::union(y, z)
yとz両方または片方に現れる行を抽出

dplyr::setdiff(y, z)
yに現れるが、zに現れない行を抽出

Binding



dplyr::bind_rows(y, z)
yに、zを新しい行として追加

dplyr::bind_cols(y, z)
yに、zを新しい列として追加
注意: 行は位置ベースで対応づけ

devtools::install_github("rstudio/EDAWR") for data sets
Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15
Translated by Tomo Masuda // @tomo_makes

Resources

- Our paper can be downloaded from TISE or from my website
  statweb.calpoly.edu/jdoi/web/research/index.htm
- Appendix A from the paper:
  Brief tutorial on how to install and get started in `Shiny`

Appendix

A. Shiny Basics

A.1. Getting Started

The version of Shiny that we used is 0.11.1 which requires R version 3.0.0 or higher. R version updates are available at cran.r-project.org.

At the R console submit the following commands to install Shiny:

```
install.packages("shiny")
library(shiny)
```

To confirm successful installation, submit the following command to launch one of the built-in Shiny example apps:

```
runExample("01_hello")
```

Resources

- Our paper can be downloaded from TISE or from my website
  statweb.calpoly.edu/jdoi/web/research/index.htm
- Appendix A from the paper:
  Brief tutorial on how to install and get started in **Shiny**
- Appendix B from the paper:
  Some teaching materials based on our **Shiny** apps
- Additional resources can be found in the paper

Summary

Summary

- Developing `Shiny` apps – very feasible
  - Required **R** background $\neq$ Master **R** Developer Level
- Start with The `Shiny` Tutorial at RStudio then **SLOWLY** convert a working **R** algorithm into an app
- Open a free account at shinyapps.io and deploy your first app
- Open a free account at gist.github.com or at github.com to store your library of source code
- Visit our Cal Poly `Shiny` Site, access our source code, and experiment on your own
- Please **do not hesitate** to send me questions as you build `Shiny` apps – I'm very happy to help

Acknowledgments

Special thanks to ...

- Dr. Kazunori Yamaguchi for inviting me to Rikkyo University

I hope to see you at ICOTS-10, July 2018 in Kyoto.

Email: jdoi@calpoly.edu
Web: www.calpoly.edu/~jdoi