Web Application Tools for Statistics Using R and Shiny

A Presentation for Faculty and Students

Jimmy Doi  jdoi@calpoly.edu
March 10, 2016

Cal Poly Statistics Department
# Table of contents

1. Introduction and Motivating example

2. *Shiny App Teaching Tools Collection*

3. *Shiny basics*

4. Launching and deploying *Shiny* apps

5. How students can use *Shiny*

6. *Shiny resources*

7. Summary
Introduction and Motivating example
Introduction

Based on the following work:

Web Application Teaching Tools for Statistics Using R and Shiny*

Jimmy Doi¹, Gail Potter¹,², Jimmy Wong¹,³, Irvin Alcaraz¹,⁴, and Peter Chi¹,⁵

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

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

• There is a large collection of applet teaching tools on the web
  • Rossman/Chance Applet Collection
  • Statistics Online Computational Resource

• Eventually an instructor can come across a problem in finding an existing applet to perfectly suit his/her needs
• There is a large collection of applet teaching tools on the web
  • Rossman/Chance Applet Collection
  • Statistics Online Computational Resource

• Eventually an instructor 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
Introduction

• There is a large collection of applet teaching tools on the web
  • Rossman/Chance Applet Collection
  • Statistics Online Computational Resource

• Eventually an instructor 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 is left to consider building his/her own teaching 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 from creating applets
• An alternative method to create web-based teaching tool applications is provided by Shiny – a web application framework for R
• An alternative method to create web-based teaching tool applications is provided by Shiny – a web application framework for R
• It is not uncommon for instructors to build their own teaching tools via scripts written in R
• It is not difficult to convert existing R scripts into Shiny web applications – known simply as ‘Shiny apps’
Introduction

• An alternative method to create web-based teaching tool applications is provided by Shiny – a web application framework for R
• It is not uncommon for instructors to build their own teaching tools via scripts written in R
• It is not difficult to convert existing R scripts 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

• STAT 130 – Intro to Statistical Reasoning
• Topic: Myths about Probability and Chance
• Popular class activity – coin flipping (find the fake sequence)

**Sequence A**


**Sequence B**

Motivating example

• STAT 130 – Intro to Statistical Reasoning
• Topic: Myths about Probability and Chance
• Popular class activity – coin flipping (find the fake sequence)

<table>
<thead>
<tr>
<th>Sequence A</th>
</tr>
</thead>
<tbody>
<tr>
<td>T T H H T H T T H H H T T H T H H H T H T H T H H T H T H T H H T T</td>
</tr>
<tr>
<td>T H T T T H T T H T H H T H T T H T H H T H T H H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence B</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
</tr>
<tr>
<td>H H H T T T T H H H H T T H H T H H T H T H T T H T</td>
</tr>
</tbody>
</table>
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
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
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 $n = 50$, show randomizations, highlight various run lengths
  - 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 students compile your R code
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 students 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
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
  • Students can access the app outside of the brief exposure in class (use as lab activity, HW assignment, or project)
  • No need for students to work with R directly (many difficulties there) – just launch a web browser
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
  • Students can access the app outside of the brief exposure in class (use as lab activity, HW assignment, or project)
  • No need for students to work with R directly (many difficulties there) – just launch a web browser
• A means to convert existing R scripts into web-applications
  • Shiny can facilitate development of new teaching 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
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](http://www.statistics.calpoly.edu/shiny)

- All *Shiny* source code available at: [gist.github.com/calpolystat](https://gist.github.com/calpolystat)
<table>
<thead>
<tr>
<th>App #</th>
<th>Author</th>
<th>App Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alcaraz</td>
<td>Correlation and Regression Game</td>
</tr>
<tr>
<td>2</td>
<td>Multiple Regression Visualization</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Probability Distribution Viewer</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Chi</td>
<td>Gambler’s Ruin</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Random Variable Generation</td>
</tr>
<tr>
<td>6</td>
<td>Doi</td>
<td>Length/Coverage Optimal Confidence Intervals</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Benford’s Law: Sequences</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Benford’s Law: Data Examples</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Chaos Game: Two Dimensions</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Chaos Game: Three Dimensions</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Longest Run of Heads or Tails</td>
</tr>
<tr>
<td>12</td>
<td>Potter</td>
<td>Testing Violation of the Constant Variance Condition for ANOVA</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Maximum Likelihood Estimation for the Binomial Distribution</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Sampling Distributions of Various Statistics</td>
</tr>
<tr>
<td>15</td>
<td>Wong</td>
<td>t-test with diagnostics</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Performance of the Wilcoxon-Mann-Whitney Test vs. t-test</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Heaped Distribution Estimation</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Hierarchical Models</td>
</tr>
</tbody>
</table>
Shiny app examples

- Benford’s Law: Data Examples
  - Dr. Ted Hill, Research Scholar in Residence at Cal Poly

- 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 interplay that occurs between `UI.R` and `SERVER.R`.

---

1As 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`. 
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**

```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**

```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")
  })
})
```
### HISTOGRAM.R

```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

```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"
  })
})
```
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")

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")
  })
})

sliderInput("numBin", "Number of bins:",
               min = 1, max = 50, value = 15)
::
mainPanel(plotOutput("distPlot"))
Shiny basics: SERVER.R and UI.R

**HISTOGRAM.R**

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

**SERVER.R**

```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**

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

mainPanel(plotOutput("distPlot"))
```
Shiny basics: SERVER.R and UI.R

SERVER.R

```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

```r
sliderInput("numBin", "Number of bins:",
           min = 1, max = 50, value = 15)
::
mainPanel(plotOutput("distPlot"))
```
Shiny basics: SERVER.R and UI.R

SERVER.R

```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

SERVER.R

```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")
  })
})
```
```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())`

```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()`

```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()`

```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()`

```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()`

```r
library(shiny)

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()`

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

- RStudio server at shinyapps.io
  - Choose one of several accounts based on a tiered pricing structure
  - Free account available but with restrictions
Deploying apps (publicly accessible via a server)

- RStudio server at shinyapps.io
  - Choose one of several accounts based on a tiered pricing structure
  - Free account available but with restrictions

- Open Shiny Server and Shiny Server Pro (costs up to $14,995)
  - For Linux based servers
Launching and deploying Shiny apps

Deploying apps (publicly accessible via a server)

- RStudio server at shinyapps.io
  - Choose one of several accounts based on a tiered pricing structure
  - Free account available but with restrictions

- Open Shiny Server and Shiny Server Pro (costs up to $14,995)
  - For Linux based servers

- RStudio Academic Pricing Policy
  (Research → 50% discount, Teaching → 100% discount)
Launching and deploying Shiny apps

Deploying apps (publicly accessible via a server)

• RStudio server at shinyapps.io
  • Choose one of several accounts based on a tiered pricing structure
  • Free account available but with restrictions

• Open Shiny Server and Shiny Server Pro (costs up to $14,995)
  • For Linux based servers

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

• Cal Poly Shiny Server Pro (100 concurrent users)
  • All made possible thanks to Dr. Phil Bailey (Dean of COSAM)
Limitations with Shiny apps

• Java/Javascript applets can perform more efficiently and faster than corresponding Shiny apps

• Shiny does not currently offer the same dynamic animation capabilities that applets can provide.
  • For example, Least Squares Regression applet at www.rossmanchance.com/applets/RegShuffle.htm

• However, Shiny is still at Version 0.13.1 ...
How students can use Shiny
How students can use Shiny

• Recent conversation with tenure track candidate on data science jobs

• Employers are putting more emphasis on seeking candidates with strong programming skills
How students can use Shiny

• Recent conversation with tenure track candidate on data science jobs

• Employers are putting more emphasis on seeking candidates with strong programming skills

• You can build Shiny apps and show future employers your programming skills
  • J. Wong and I. Alcaraz did just that on their interviews
How students can use Shiny

• Recent conversation with tenure track candidate on data science jobs

• Employers are putting more emphasis on seeking candidates with strong programming skills

• You can build Shiny apps and show future employers your programming skills
  • J. Wong and I. Alcaraz did just that on their interviews

• Not all companies know about Shiny – you may be the one to introduce them to it
How students can use Shiny

- Convert R scripts created for homework assignments/class projects into Shiny apps

- If appropriate, integrate Shiny into your senior project (subject to approval from your advisor). Some students are already doing this now.

- Create a free account at shinyapps.io for starters and host your apps there

- Make your own GitHub account (great way to showcase your library of code)
Shiny 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

• Shiny Cheat Sheet: www.rstudio.com/resources/cheatsheets

• Shiny app for calculus: www.r-bloggers.com/integration-take-two-shiny-application
Shiny Cheat Sheet

Resources: Shiny Cheat Sheet

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

- Provide server.R with the minimum necessary code, `shinyServer(function(input, output) { })`
- Define the R components for your app between the braces that follow `function(input, output)`
- Save each R component in your UI as `output$<component name>`
- Create each output component with a `render*` function
- Give each `render*` function the R code the server needs to build the component. The server will note any reactive values that appear in the code and will rebuild the component whenever those values change.
- Refer to widget values with `input$<widget name>`

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.
- Reactive expression - use reactive to create objects that will be used in multiple outputs.
- `isolate` - use `isolate` to use an input without depending on it. Shiny will not rebuild the output when the isolated input changes.
- `observe` - use `observe` to create code that runs when an input changes, but does not create an output object.
Resources: Shiny Cheat Sheet

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

- Include the minimum necessary code for ui.R, shinyUI(fluidPage())
  Use a new fluidPage() instead of fluidPage() if you’d like your app to have multiple pages connected by a navtab

- 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.

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 at 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: Shiny app for calculus

Area from $-1$ to $1 = 2.6667$

Integration from $-1$ to $1 = 2.5922$
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
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:

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

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

```r
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
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

• STAT 331 Shiny assignment (G. Potter) available at my website
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

- STAT 331 Shiny assignment (G. Potter) available at my website

- Additional resources can be found in the paper
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

• STAT 331 Shiny assignment (G. Potter) available at my website

• Additional resources can be found in the paper

• These slides can be downloaded from my website
Summary
Issues to Consider when Selecting Technology in a Statistics Class

We believe that no one tool can do it all and that there are many good tools available to use, many of which are free. Therefore, rather than thinking about one technological tool for students to use, we encourage teachers to think about what sets of tools will help students best learn statistics in each unit of the course.²

Issues to Consider when Selecting Technology in a Statistics Class

The GAISE College Report lists some issues to consider when selecting technological tools to use in helping students learn statistics:

- Ease of data entry, ability to import data in multiple formats
- Interactive capabilities
- Dynamic linking between data, graphical, and numerical analyses
- Ease of use for particular audiences
- Availability to students, portability

Shiny is a tool, not THE tool
Shiny is a tool, not THE tool.

Issues to Consider when Selecting Technology in a Statistics Class

The GAISE College Report lists some issues to consider when selecting technological tools to use in helping students learn statistics:

- Ease of data entry, ability to import data in multiple formats
- Interactive capabilities
- Dynamic linking between data, graphical, and numerical analyses
- Ease of use for particular audiences
- Availability to students, portability

We believe Shiny can do all of these things.
Summary

- Developing Shiny apps – very feasible
  - Required R background ≠ Master R Developer Level
• Developing Shiny apps – very feasible
  • Required R background ≠ Master R Developer Level

• Start with The Shiny Tutorial at RStudio then SLOWLY convert a working R script 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 showcase your library of source code

• Visit our Cal Poly Shiny Site, access our source code, and experiment on your own

• Invitation to contribute to the Cal Poly Shiny App Teaching Tools Collection

• Help spread the word: Shiny talk to be given at Joint Statistical Meetings 2016 Chicago by Dr. Gail Potter.
Summary

- Developing Shiny apps – very feasible
  - Required R background ≠ Master R Developer Level
- Start with The Shiny Tutorial at RStudio then SLOWLY convert a working R script 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 showcase your library of source code
- Visit our Cal Poly Shiny Site, access our source code, and experiment on your own

• Invitation to contribute to the Cal Poly Shiny App Teaching Tools Collection
• Help spread the word: Shiny talk to be given at Joint Statistical Meetings 2016 Chicago by Dr. Gail Potter.
Summary

- Developing **Shiny** apps – very feasible
  - Required R background ≠ Master R Developer Level
- Start with **The Shiny Tutorial at RStudio** then **SLOWLY** convert a working R script 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 showcase your library of source code
- Visit our **Cal Poly Shiny Site**, access our source code, and experiment on your own
- Invitation to contribute to the Cal Poly **Shiny App Teaching Tools Collection**
Summary

- Developing Shiny apps – very feasible
  - Required R background ≠ Master R Developer Level
- Start with The Shiny Tutorial at RStudio then SLOWLY convert a working R script 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 showcase your library of source code
- Visit our Cal Poly Shiny Site, access our source code, and experiment on your own
- Invitation to contribute to the Cal Poly Shiny App Teaching Tools Collection
- Help spread the word: Shiny talk to be given at Joint Statistical Meetings 2016 Chicago by Dr. Gail Potter.
Acknowledgments

Special thanks to ...

- Co-authors: Peter Chi, Gail Potter, Jimmy Wong, Irvin Alcaraz
- Allan Rossman and Beth Chance for initial consultation and suggesting we submit our paper to TISE.
- Mark Schilling and Ann Watkins for their feedback on our apps and on a draft of our paper.
- Dean Phil Bailey and Roger Oberg of RStudio for their support in helping to establish the Shiny Server at Cal Poly.
- COSAM Computing Support, especially to Brian Zelenke and Chip Dupre, for setting up and maintaining our Shiny Server.
Extra embedded commas in R functions often do not lead to errors

The following pairs of commands yield the same results:

\[
\begin{align*}
\text{summary}(x) & \iff \text{summary}(x, ) \\
\text{plot}(x, y) & \iff \text{plot}(x, y, ) \\
\text{lm}(y \sim x) & \iff \text{lm}(y \sim x, )
\end{align*}
\]

An extra comma at the final position of a function usually doesn’t lead to problems.
Shiny basics: Extra embedded comma error

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

```r
shinyUI(fluidPage(
  f1(A),
  f2(
    f3(
      f4(B, C, D, E, F)),
    f5(
      f6(G))
  )
))
```
Shiny basics: Extra embedded comma error

```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: Extra embedded comma error

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

```ui.R
shinyUI(fluidPage(
  f1(A),
  f2(
    f3(
      f4(B, C, D, E, F)),
    f5(
      f6(G)))
)
))
```

`↑` `↓`

```ui.R
shinyUI(fluidPage(f1(A), f2(f3(f4(B, C, D, E, F)), f5(f6(G)))))
```

`^` = the final position of a function
An extra embedded comma in `UI.R` in the final position of a function can lead to an error such as this:

```
Error Message

Warning: Error in shinyUI: unused argument ()
Stack trace (innermost first):
  40: shinyUI
   1: shiny::runApp
Error in shinyUI(fluidPage(titlePanel("Histogram Shiny App"),
               sidebarLayout(sidebarPanel(sliderInput("numBin", :
               unused argument ()
```

This was the most common error I encountered. Traditional block commenting doesn’t help here. Shiny debugging functions may offer more help in these situations.
An extra embedded comma in `UI.R` in the final position of a function can lead to an error such as this:

```
Warning: Error in shinyUI: unused argument ()
Stack trace (innermost first):
  40: shinyUI
  49: shinyUI
  1: shiny::runApp
Error in shinyUI(fluidPage(titlePanel("Histogram Shiny App"),
  sidebarLayout(sidebarPanel(sliderInput("numBin", :
  unused argument ()
```

This was the most common error I encountered.

Traditional block commenting doesn’t help here.

**Shiny** Debugging functions may offer more help in these situations.
```{r, echo=FALSE}
inputPanel(
  selectInput("n_breaks", label = "Number of bins:",
    choices = c(10, 20, 35, 50), selected = 20),

  sliderInput("bw_adjust", label = "Bandwidth adjustment:",
    min = 0.2, max = 2, value = 1, step = 0.2)
)

renderPlot({
  hist(faithful$eruptions, probability = TRUE,
    breaks = as.numeric(input$n_breaks),
    xlab = "Duration (minutes)",
    main = "Geyser eruption duration")

  dens <- density(faithful$eruptions, adjust = input$bw_adjust)
  lines(dens, col = "blue")
})
```