An Introduction to Shiny Apps in R with Applications to Statistics Education

Daniel W. Adrian

Grand Valley State University
Department of Statistics

Department Seminar
March 24, 2016

Outline

Song: "Shiny (H)appy People", by R.E.M.

Link to YouTube Video

My website

`http://facweb.gvsu.edu/adriand1/index.html`

- PDF of these slides
- Link to ePoster for Teaching and Learning with Technology Symposium
- Math in Action
  - slides
  - sample apps and their R code

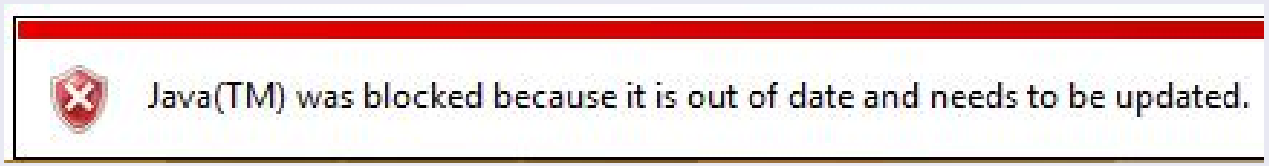
Note: All web addresses in this presentation are hyperlinks on the PDF.

Motivation

- Shiny apps are similar to Java Applets but without the following problems:

Problems with Java Applets

- Require updates or downloads



- Blocked due to security concerns
- Browsers no longer supporting Java Applets
  - Google Chrome (after version 45)
  - Microsoft Edge
- Not customizable unless you know Java.

Shiny Apps are written in R

- Free!
- Three Downloads:
  1. The Base System: `https://cran.rstudio.com/` (may need to update if your version is a few years old)
  2. User Interface: `https://www.rstudio.com/products/rstudio/#Desktop`
  3. Shiny Package (In R, Tools $\rightarrow$ Install Packages...)
- Students do not need to know R to use Shiny Apps.
- Note: Version of R on GVSU computers does not have Shiny.

Outline

Demonstrations

ePoster

- ePoster is a Shiny App
- Contains 12 tabs organized in menus
- Each tab is a learning activity

Math in Action

- "Single tab" apps at
  `http://facweb.gvsu.edu/adriand1/mia.html`
- Click on "shinyapps.io" or "GVSU server" next to the following apps:
  - Slope-intercept Form of a Line
  - The Unit Circle: sine and cosine functions (try the animation)
  - Equation of a circle
  - Equation of a parabola (vertex form)
  - Normal distribution
- R code for each app given.

Discussion

- Shiny Apps provide for student learning that is
  - dynamic
  - interactive
  - exploration-based
  - more fun
- Illustrate dynamic visual concepts very well

Outline

Excellent Online Tutorial about Shiny Apps

- Website: `http://shiny.rstudio.com/tutorial/`
- I borrow from it in this presentation.

Two Parts of a Shiny App

User Interface (UI)

What is shown on the webpage

A computer (server) running R

Performs calculations to update the webpage

Outline

The Template

```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

- Every time you make a new app, you should start with this template.
- Available on my Math in Action webpage

The template

```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

- `library(shiny)`: Loads the package "shiny" into the current R session.
- `ui <- fluidPage()`: What goes between `()` determines what is shown on the user interface (UI), i.e. webpage.
- `server <- function(input, output)`: Tells the server what to do to update the webpage.
  - Takes `input` from the webpage (like slope and intercept)
  - Produces `output` to the webpage (like updated graph)
- `shinyApp(ui = ui, server = server)`: R does its "magic" to create the app from your code.

Outline

`*Input()` and `*Output()` functions

In general

- Go in the `ui <- fluidPage()` part.
- `*Input()`: input from user of webpage $\rightarrow$ server
- `*Output()`: output from server $\rightarrow$ user of webpage

My apps use lots of...

- `sliderInput()`
- `plotOutput()`

Family of `*Input()` functions

Buttons

Action

Submit

`actionButton()`
`submitButton()`

Single checkbox

Choice A

`checkboxInput()`

Checkbox group

Choice 1
Choice 2
Choice 3

`checkboxGroupInput()`

Date input

2014-01-01

`dateInput()`

Date range

2014-01-24 to 2014-01-24

`dateRangeInput()`

File input

Choose File No file chosen

`fileInput()`

Numeric input

1

`numericInput()`

Password Input

`passwordInput()`

Radio buttons

Choice 1
Choice 2
Choice 3

`radioButtons()`

Select box

Choice 1

`selectInput()`

Sliders

`sliderInput()`

Text input

Enter text...

`textInput()`

Link to Shiny "widget" gallery

Family of *Output() functions

Function	Inserts
`dataTableOutput()`	an interactive table
`htmlOutput()`	raw HTML
`imageOutput()`	image
`plotOutput()`	plot
`tableOutput()`	table
`textOutput()`	text
`uiOutput()`	a Shiny UI element
`verbatimTextOutput()`	text

For documentation (help): `?functionname`

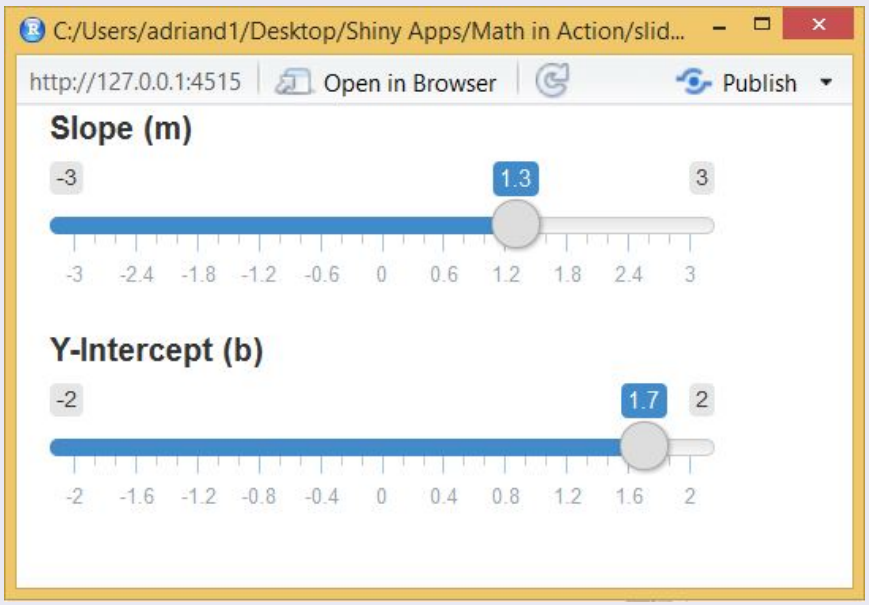
Outline

Example: Slope-intercept app

Code

```
ui <- fluidPage(
  sliderInput(inputId="m", label='Slope (m)',
              value=1.3, min=-3, max=3, step=.1),
  sliderInput(inputId="b", label='Y-Intercept (b)',
              value=1.7, min=-2, max=2, step=.1),
  plotOutput('myplot')
)
```

Webpage



Building the output (plot) in the `server` function

3 steps:

1. Save to `output$`
2. Use `render*()` functions to produce output – in this case, `renderPlot()`.
3. Incorporate inputs with `input$`

1. Save to output$

Code

```
ui <- fluidPage(
  sliderInput(inputId="m", label='Slope (m)',
              value=1.3, min=-3, max=3, step=.1),
  sliderInput(inputId="b", label='Y-Intercept (b)',
              value=1.7, min=-2, max=2, step=.1),
  plotOutput('myplot')
)

server <- function(input, output) {
  output$myplot <- #code
}
```

- Save to `output$`
- The name following `output$` needs to match the name in `plotOutput`.
- Here: `output$myplot` matches `plotOutput('myplot')`

2. Use render*()

Code

```
ui <- fluidPage(
  sliderInput(inputId="m", label='Slope (m)',
              value=1.3, min=-3, max=3, step=.1),
  sliderInput(inputId="b", label='Y-Intercept (b)',
              value=1.7, min=-2, max=2, step=.1),
  plotOutput('myplot')
)

server <- function(input, output) {
  output$myplot <- renderPlot({
    #code
  })
}
```

- `render*()` functions make the app "reactive".
- Other `render*()` functions:
  - `renderTable()`
  - `renderText()`

3. Use input$ to incorporate inputs

Code

```
ui <- fluidPage(
  sliderInput(inputId="m", label='Slope (m)',
              value=1.3, min=-3, max=3, step=.1),
  sliderInput(inputId="b", label='Y-Intercept (b)',
              value=1.7, min=-2, max=2, step=.1),
  plotOutput('myplot')
)

server <- function(input, output) {
  output$myplot <- renderPlot({
    x <- seq(from=-3, to=3, by=1)
    y <- input$m * x + input$b
    plot(x, y, type='l')
  })
}
```

Note:

- `input$m` matches `sliderInput(inputId='m', ...)`
- `input$b` matches `sliderInput(inputId='b', ...)`

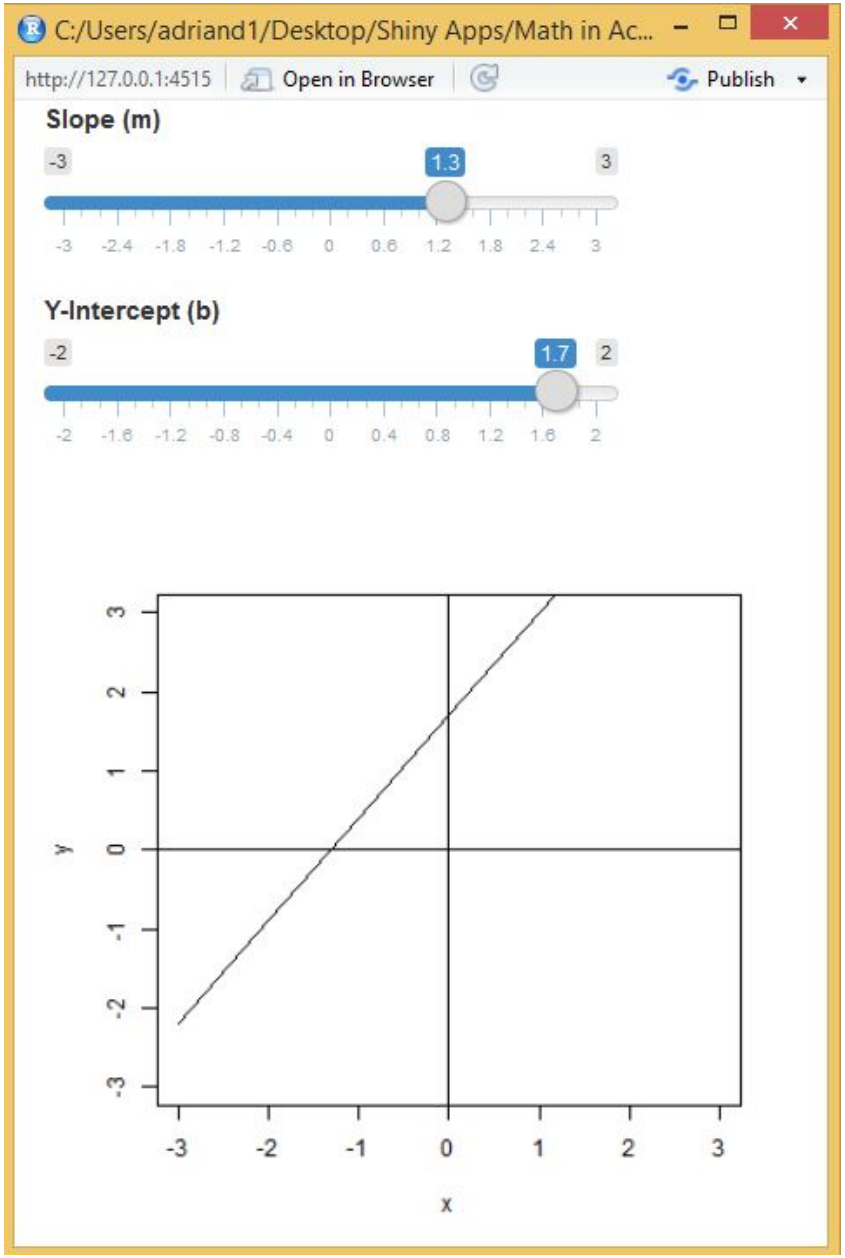
Add to the plot

Code

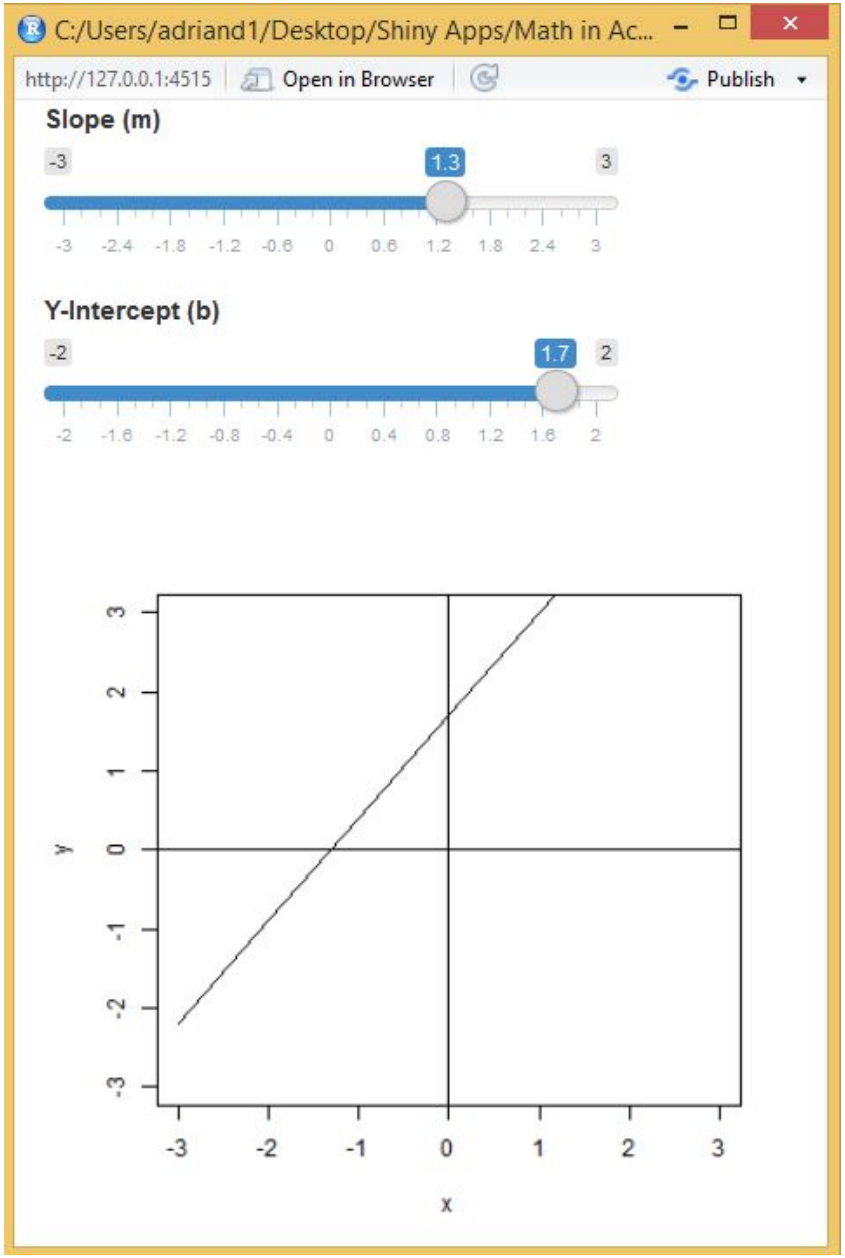
```
server <- function(input, output) {
  output$myplot <- renderPlot({
    x <- seq(from=-3, to=3, by=1)
    y <- input$m * x + input$b
    plot(x, y, type='l', ylim=c(-3,3))
    abline(h=0)
    abline(v=0)
  })
}
```

- `ylim=c(-3,3)` fixes the y-axis limits (so we can see the effect of the slope)
- `abline(h=0)` and `abline(v=0)` add the x- and y-axes.

Resulting app (code on MIA website – `example.R`)



Resulting app (code on MIA website – `example.R`)



Outline

Layout: fluidRow() and column()

Code

```
ui <- fluidPage(
  fluidRow('Slope-intercept app'),
  fluidRow(column(4,
            sliderInput(inputId="m", label='Slope (m)',
                        value=1.3, min=-3, max=3, step=.1),
            sliderInput(inputId="b", label='Y-Intercept (b)',
                        value=1.7, min=-2, max=2, step=.1)
            ),
            column(8, plotOutput('myplot'))
            )
)
```

- column() goes inside fluidRow()
- Arguments of column():
  1. width (must be integer, whole window: 12)
  2. objects in that column

wellPanel()

Code

```
ui <- fluidPage(
  fluidRow(wellPanel('Slope-intercept app')),
  fluidRow(column(4, wellPanel(
            sliderInput(inputId="m", label='Slope (m)',
                        value=1.3, min=-3, max=3, step=.1),
            sliderInput(inputId="b", label='Y-Intercept (b)',
                        value=1.7, min=-2, max=2, step=.1)
            )),
            column(8, wellPanel(
              plotOutput('myplot')
              ))
            )
)
```

Separates and outlines rows and columns.

Adding HTML code in Shiny

- HTML code consists of tags.
- For example, the HTML tag to make the text the largest headline is "h1".

HTML code

```
<h1> Slope intercept form </h1>
```

- The corresponding R code is:

R code

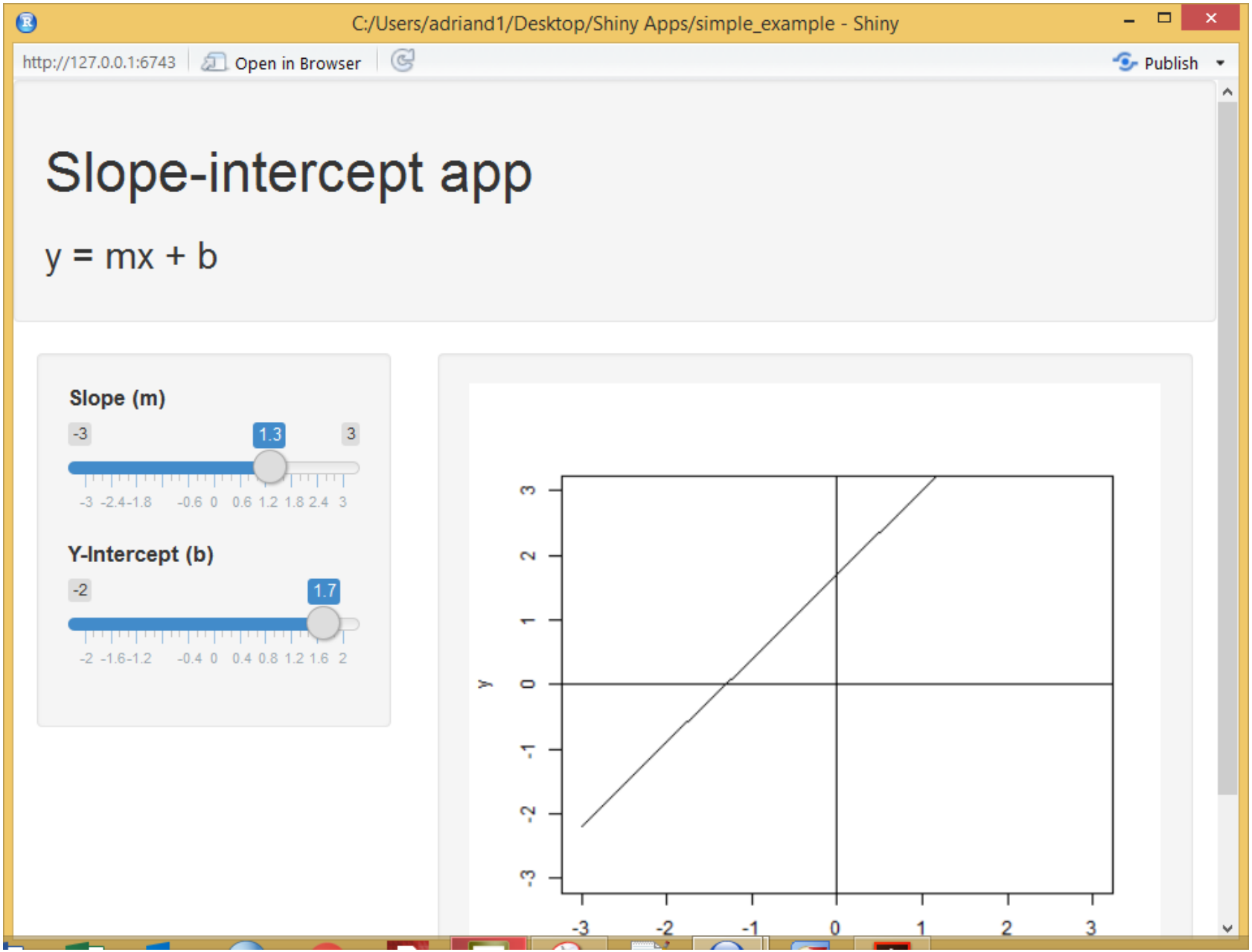
```
tags$h1('Slope intercept form')
```

Completed example

Code

```
ui <- fluidPage(
  fluidRow(wellPanel(
    tags$h1('Slope-intercept app'),
    tags$h3('y = mx + b')
  )),
  fluidRow(column(4, wellPanel(
            sliderInput(inputId="m", label='Slope (m)',
                        value=1.3, min=-3, max=3, step=.1),
            sliderInput(inputId="b", label='Y-Intercept (b)',
                        value=1.7, min=-2, max=2, step=.1)
            )),
            column(8, wellPanel(
              plotOutput('myplot')
              ))
            )
)
```

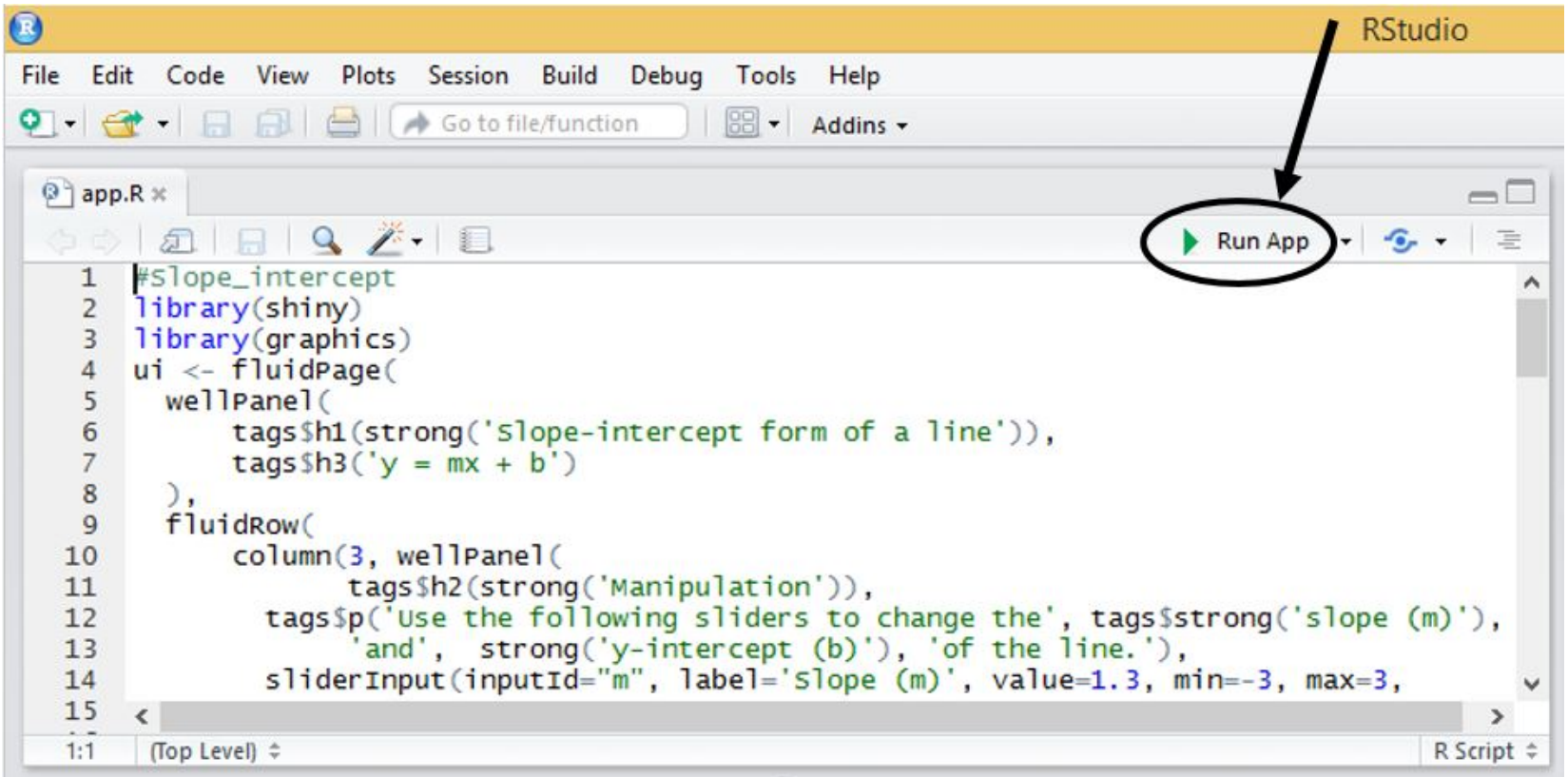
Completed app



Outline

Use R as a local server

- Save code as the file `app.R`.
- Click on "Run App"



- Great for developing apps.
- Disadvantage: For other users to run apps, need R (with package Shiny) installed.

shinyapps.io: from Rstudio

- Allows user without R to use apps through a web browser.
- Free Plan - Limitations:
  - Only 5 Apps
  - Only 25 active hours per month
- Other plans available with more apps and hours but for $$$.
- Info:
  `http://shiny.rstudio.com/articles/shinyapps.html`

Shiny Server: Linux

- Use Linux to create your own server.
- Advantage: no limits on number of apps or hours
- Info: `www.rstudio.com/products/shiny/shiny-server/`
- Not recommended unless you are an advanced Linux user.

- **Thanks Dave!** – Dave has set up a Shiny Server for our department.

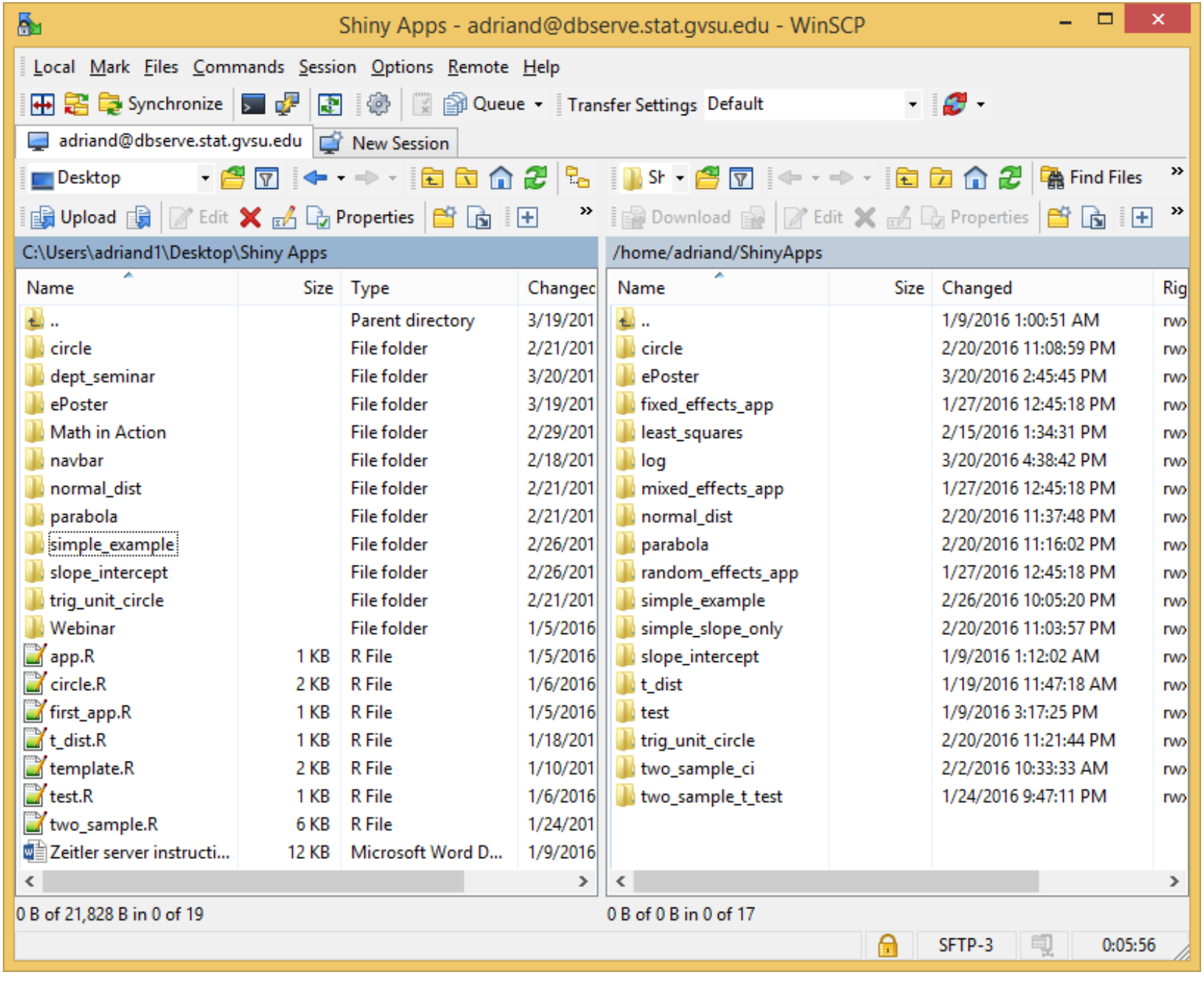
Using Dave's Shiny Server

1. Ask Dave for an account.
2. Download WinSCP or another file transfer program.
3. Make "ShinyApps" directory. For example, for me: `/home/adriand/ShinyApps`

4. Put a folder containing app.R (and other files) in the ShinyApps folder.
5. Name of this folder: name of app.

6. Example: `/home/adriand/ShinyApps/ePoster/app.R`
7. Website for app: `http://dbserve.stat.gvsu.edu:3838/adriand/ePoster/`

WinSCP Screenshot



Using Dave's Shiny Server (cont.)

- Let's "publish" the app we made and call it "seminar_app".
- Caution: To use apps on the server, users either need to be on the GVSU network or logged in through VPN.

TISE paper on Shiny Apps

- California Polytechnic State University
- Reference: Doi, J., Potter, G., Wong, J., Alcaraz, I., & Chi, P. (2016). Web Application Teaching Tools for Statistics Using R and Shiny. Technology Innovations in Statistics Education, 9(1).
- Website: `http://www.statistics.calpoly.edu/shiny`

Thank you.