Building an Interactive Online Fact Book with R Shiny

Mike Wallinga
Director of Institutional Research
Northwestern College, Orange City, IA
AIRUM Conference 2013

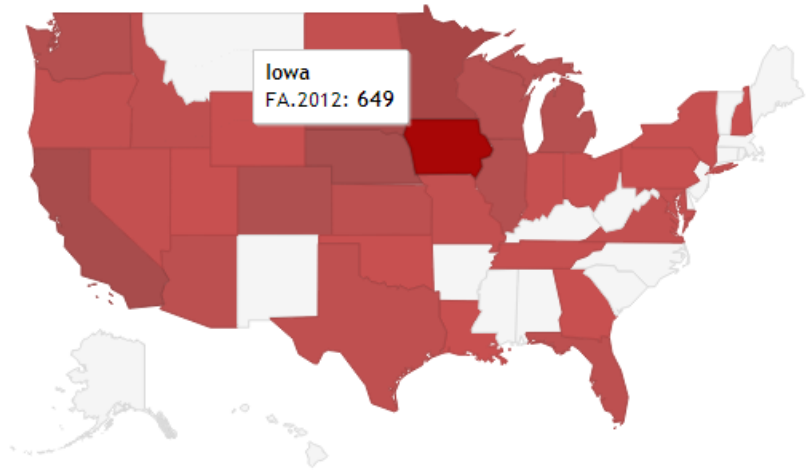
NORTHWESTERN
COLLEGE

My Annual Fact Book Struggle/Soapbox

- Creating pages was a highly manual process
  - Lots of editing in Excel, Word, Access, etc.
- Last year, I wrote some R scripts to extract the data and build static HTML pages using R Markdown

NORTHWESTERN COLLEGE

The Result



State	FA 2008	FA 2009	FA 2010	FA 2011	FA 2012
Alaska	1	1	1	1	0
Arizona	11	8	7	6	8
Arkansas	1	2	3	0	0
California	39	42	59	66	85
Colorado	30	25	29	24	32

NORTHWESTERN COLLEGE

Better, but not Ideal

- My new fact book was still too static – it didn't take advantage of its new interactive medium
- Even worse, the finished product still didn't serve everyone's needs:
  - Headcount vs. FTE vs. full-time vs. degree-seeking
  - Sorted alphabetically vs. numerically
  - Current year vs. historical trends

NORTHWESTERN COLLEGE

I Wanted Something Like This

NORTHWESTERN COLLEGE

(Reset Password) User Name: Password: Login

Home Academics Alumni Campus Life Career Development Help Desk Institutional Research

You are here: Institutional Research > Enrollment Summary

Institutional Research

- Home
- Enrollment Summary
- Graduation Rates

Quick Links

Northwestern Home Page

Red Raider Athletics

Enrollment Summary

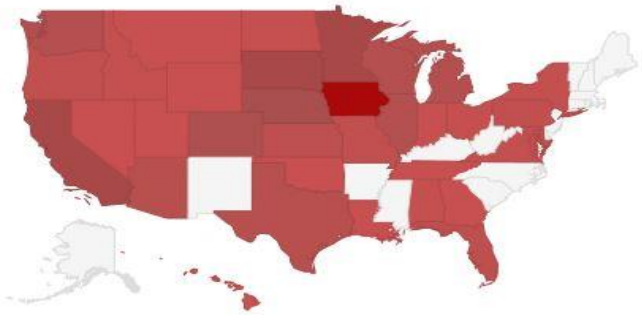
Choose Category: Enrollment

Choose Enrollment type: All Students

View enrollment by: Geography

Pick One: Home State

This Year | Historical



	Home State	Number of USA Students	Percentage
1	IA	627	52.3%
2	MN	114	9.5%
3	SD	106	8.8%
4	CA	105	8.8%
5	NE	59	4.9%
6	CO	38	3.2%
7	IL	22	1.8%
8	WI	19	1.6%
9	WA	18	1.5%
10	MI	15	1.3%

◄ ► 1 2 3 4

Ooooh… Shiny!!!

- **"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.
- 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."
  - http://www.rstudio.com/shiny/

NORTHWESTERN COLLEGE

That Sounds Good!

- Completely web-based (no reader software or web browser plug-ins needed)
- Interactive, with real-time updates of the page (real-time data updates optional)
- No HTML or JavaScript coding required
- Leverage my existing knowledge AND code
- Use existing, familiar tools (RStudio)

NORTHWESTERN COLLEGE

Getting Started

- Basic idea: two files
  - One defines the user interface elements
  - One contains the server-side logic for computing the output
- Let's start by making a basic interface

Basic UI Code

```
shinyUI(
  # This is the primary UI layout offered by Shiny, contains 3 areas:
  pageWithSidebar(

    # The header panelcontains the application title
    headerPanel("AIRUM 2013 Presentation"),

    # The sidebar panel typically holds most of your options/settings
    # (a single drop-down menu in this case)
    sidebarPanel(
      # The first argument is the ID,
      # the second is the label in the UI,
      #the third is the list of choices
      selectInput ("category",
                   "Choose Category:",
                   list("Major" = "major",
                                  "Home State" = "state",
                                  "Academic Classification" = "class")
    )),

    # The main panel typically displays your results (empty for now)
    mainPanel()
))
```

Let's Add Some Basic (Non-Interactive) Server Code

```
shinyServer(function(input, output) {

  # We get this output object for free
  # It contains all of the output values defined in the UI
  # Here, we declare a "caption" output value
  # ad set it to the text "Hello World"

  output$caption <- renderText("Hello World")

})
```

NORTHWESTERN COLLEGE

One Modification to the UI Code:

```
mainPanel(
   # Here's the only change to the UI -
   # add a text output space that will receive
   # output from the server logic
   # (in this case, the output is called "caption")
   textOutput("caption")
)
```

NORTHWESTERN COLLEGE

Plain Text is Boring…
Time for Something More Interesting

```
mainPanel(
  # Let's add a plot to our output values
  # and display it at the top of the main panel
  plotOutput("plot"),

  # And change this line from textOutput to htmlOutput
  # so that the text received is interpreted as HTML
  htmlOutput("caption")
)
```

NORTHWESTERN COLLEGE

Changes to the Corresponding Server Code

```
shinyServer(function(input, output) {

  # We still use renderText, but we can embed HTML tags,
  # and they will be interpreted correctly by the UI

  output$caption <- renderText("<h2>Hello World</h2>")

  # The server generates a histogram, stores in output$plot
  # The UI will receive it and display it on the screen

  output$plot <- renderPlot(hist(rnorm(100)))

})
```

NORTHWESTERN COLLEGE

So Far, So Good!

- We've built the desired basic UI structure
- Created information in the server code, displayed it in the UI code
- But, it's all made-up data, we want to use *real* data

My “Real” Data

	id	name	gender	major	state	class
1	1	Jim	M	Biology	IA	FR
2	2	Joe	M	Psychology	MN	FR
3	3	Sally	F	Chemistry	MN	SO
4	4	Sue	F	Psychology	IA	SO
5	5	Bob	M	Math	NE	SO
6	6	Mike	M	Math	SD	JR
7	7	Matt	M	Sociology	SD	JR
8	8	Jill	F	Math	IA	SR
9	9	Jane	F	Psychology	ND	SR
10	10	John	M	Biology	WI	FR

Read and Display from a CSV

```
shinyServer(function(input, output) {

  # Get some data
  data <- read.csv('data.csv')

  # Switch from renderText to renderTable
  # Note: This calls xtable behind the scenes,
  # which produces HTML output
  output$caption <- renderTable(table(data[,4]))

  # Switched this to a pie chart just for kicks
  output$plot <- renderPlot(pie(table(data[,4])))

})
```

Note: no changes to the UI code at this point

A Quick Refactoring Improvement

```
shinyServer(function(input, output) {

  data <- read.csv('data.csv')

  # Extract the column number and assign it to a variable
  major <- 4

  # Now we replace our hard-coded column 4 with the variable
  output$caption <- renderTable(table(data[,majors]))

  output$plot <- renderPlot(pie(table(data[,majors])))

})
```

NORTHWESTERN COLLEGE

Time to React!

- This is great, but we aren't reacting at all to the user input in the drop-down menu
- Now that we're using a variable to index into our data, it would be nice to change that variable when the user picks something new
- We can do this by creating a Shiny function called a "reactive"

Set our Column Variable Reactively

```
shinyServer(function(input, output) {
  data <- read.csv('data.csv')

  # Change our column number depending on
  # the value of the drop-down menu in the UI
  column <- reactive ({
    switch(input$category,
           "major" = 4,
           "state"= 5,
           "class"= 6
    ) # end the switch statement
  }) # end the reactive function

  # Everywhere we had "major" before, we want to use "column()"
  # In other words, we'll call our column() reactive,
  # and use the returned result as the column index
  output$caption <- renderTable(table(data[,column()]))

  output$plot <- renderPlot(pie(table(data[,column()])))
})
```

Note: still no changes to the UI code at this point

Adding a Second Set of Choices

- We'll add another drop-down list to the left-hand column in the UI to pick between all students, only males, or only females
- This new menu will combine with the existing menu's three choices and allow the user to display 3 * 3 = 9 different datasets
- We need a second reactive function in the server code to filter the data correctly

The New UI Code

```
shinyUI(
  pageWithSidebar(

    headerPanel("AIRUM 2013 Presentation"),

    # Add a new drop-down list to filter the data by gender
    sidebarPanel(
      selectInput ("gender", "Choose Gender:", list("Male and Female" = "both",
                                                    "Male only" = "male",
                                                    "Female only" = "female")

      ),
      selectInput ("category", "Choose Category:", list("Major" = "major",
                                                        "Home State" = "state",
                                                        "Academic Classification" = "class")

      )
    ),

    mainPanel(
      plotOutput("plot"),
      htmlOutput("caption")
    )
  )
)
```

NORTHWESTERN COLLEGE

The New Server Code

```
shinyServer(function(input, output) {
  data <- read.csv('data.csv')

  # Filter the data from the CSV
  # when the UI's Gender drop-down list changes
  filter <- reactive ({
    switch(input$gender,
           "both" = data,
           "male"= subset(data, gender == 'M'),
           "female"= subset(data, gender == 'F'))
  })
  column <- reactive ({
    switch(input$category,
           "major" = 4,
           "state"= 5,
           "class"= 6)
  })

  # Instead of using our "data" variable, we need to check
  # the filter() reactive to see which subset to use;
  # (Notice you can index a particular column of a reactive call!)
  output$caption <- renderTable(table(filter()[,column()]))
  output$plot <- renderPlot(pie(table(filter()[,column()])))
})
```

Adding Historical Data

- Our current example only deals with the current year's data
- What if we make our dataset more complex by adding a "year" column and including data for 2011, 2012, and 2013?
- We'll add a second tab to the UI, so a user can easily toggle between the single year and historical perspectives

NORTHWESTERN COLLEGE

	year	id	name	gender	major	state	class
1	2013	1	Jim	M	Biology	IA	SR
2	2013	2	Joe	M	Psychology	MN	SR
3	2013	3	Sally	F	Chemistry	MN	JR
4	2013	4	Sue	F	Psychology	IA	JR
5	2013	5	Bob	M	Math	NE	JR
6	2013	6	Mike	M	Math	SD	SO
7	2013	7	Matt	M	Sociology	SD	SO
8	2013	8	Jill	F	Math	IA	SO
9	2013	9	Jane	F	Psychology	ND	FR
10	2013	10	John	M	Biology	WI	FR
11	2012	1	Jim	M	Biology	IA	JR
12	2012	2	Joe	M	Psychology	MN	JR
13	2012	3	Sally	F	Chemistry	MN	SO
14	2012	4	Sue	F	Psychology	IA	SO
15	2012	5	Bob	M	Math	NE	SO
16	2012	6	Mike	M	Math	SD	FR
17	2012	7	Matt	M	Sociology	SD	FR
18	2012	8	Jill	F	Math	IA	FR
19	2012	9	Jerry	M	English	IA	FR
20	2012	10	Rachel	F	Theatre	SD	SO
21	2012	11	Tim	M	Chemistry	WI	SR
22	2012	12	Nancy	F	English	WI	SR
23	2012	13	Kim	F	Psychology	IA	SR
24	2012	14	Megan	F	Biology	NE	SR
25	2011	1	Jim	M	Biology	IA	SO
26	2011	2	Joe	M	Psychology	MN	SO
27	2011	3	Sally	F	Chemistry	MN	FR
28	2011	4	Sue	F	Psychology	IA	FR
29	2011	5	Bob	M	Math	NE	FR
30	2011	6	Rachel	F	Theatre	SD	FR
31	2011	7	Mark	M	Sociology	IA	JR
32	2011	8	Debbie	F	Math	MN	JR
33	2011	9	John	M	English	NE	JR
34	2011	10	Sarah	F	English	NE	SR

Adding a Second Tab to the UI

```
mainPanel(

  # We create a tabset inside of the main panel
  tabsetPanel(

    # One tab contains our existing panel contents
    tabPanel("This Year",
             plotOutput("plot"),
             htmlOutput("caption")),

    # This new tab contains the historical output
    tabPanel("Historical",
             plotOutput("hist_plot"),
             htmlOutput("hist_table"))
  )
)
```

Changes to the Server Code – Part 1

- Our existing filter() reactive will stay the same, but we'll need to subset the results for our existing "plot" and "caption" output values
- You could make an argument for subsetting for the current year in the filter() reactive instead, but I chose to keep the dataset as broad as possible, as long as possible

Changes to Existing Code

```
shinyServer(function(input, output) {
  data <- read.csv('data.csv')

  filter <- reactive ({
    switch(input$gender,
           "both" = data,
           "male" = subset(data, gender == 'M'),
           "female" = subset(data, gender == 'F'))
  })

  # Our data file has a new format, with a new column at the beginning,
  # so we need to increase our indices
  column <- reactive ({
    switch(input$category,
           "major" = 5,
           "state"= 6,
           "class"= 7)
  })

  # Since our data now contains multiple years,
  # we need to subset() for just the current year
  output$caption <- renderTable(table(subset(filter(), year==2013)[,column()]))
  output$plot <- renderPlot(pie(table(subset(filter(), year==2013)[,column()])))
```

Changes to the Server Code – Part 2

- Add a new reactive that will group and aggregate the data by year
  - Uses some ddply and dcast magic
- Add an output value that will display a line graph on the Historical tab
- Add an output that will display a table on the Historical tab

NORTHWESTERN COLLEGE

The New Reactive Code

```
# Add a new reactive that will aggregate the data by year (using ddply from plyr)
# and then reformat it to output it the way we'd like (using dcast from reshape2)
filter.hist <- reactive ({
  switch(input$gender,
    "both" = temp <- ddply(data,
                           c('year',as.character(names(data)[column()])) ,
                           nrow),
    "male" = temp <- ddply(subset(data, gender=='M'),
                           c('year',as.character(names(data)[column()])) ,
                           nrow),
    "female" = temp <- ddply(subset(data, gender=='F'),
                             c('year', as.character(names(data)[column()])) ,
                             nrow)
  )
  return(dcast(temp, year~temp[,2], value.var='V1'))
})
```

NORTHWESTERN COLLEGE

Output for the Historical Tab

```
# Display the data in a table on the Historical tab
output$hist_table <- renderTable(filter.hist())

# Create a line graph and add a line for each column in the set of results
# (Displayed on Historical tab)
output$hist_plot <- {
  renderPlot({
    colors <- rainbow(length(filter.hist()[1,]-1))

    plot(filter.hist()[,2] ~ filter.hist()[,1],
         type = "l", xaxt='n', ylim = c(0,5),
         xlab="Year", ylab="Students", col=colors[1], lwd=3)

    for (i in 3:ncol(filter.hist()))
      lines(x=filter.hist()[,1], y=filter.hist()[,i], type="l", col=colors[i-1], lwd=3)

    axis(1,at=2011:2013,labels=c('2011','2012','2013'))

    legend("topright",
           legend=names(filter.hist()[,2:length(filter.hist()[1,])]),
           fill=colors,
           ncol=2)
  })
}
```

Not Bad for 30 Minutes of Work! ☺

- This example doesn't solve everything I listed earlier, but it's a really good start
- On one hand, this example is rather bare-bones and not polished
- On the other hand, it's pretty complex!

NORTHWESTERN COLLEGE

Re-Using Existing Code

- One of the best parts of using Shiny is that you can use your favorite packages/existing code, and plug them into the output values in your server code
- Your existing code can take advantage of the reactive nature with minimal changes
- For my NWC fact book, I use the googleVis package
  - User-sortable tables!
  - I like their maps
  - Much easier to make line graphs ☺
  - Caveat: Requires live Internet connection

Moving a Shiny App into Production

- You can develop Shiny apps in RStudio using Windows, Linux, or Mac
- But currently, the Shiny web server only runs on Linux
  - The RStudio team provides very good documentation for installing under Ubuntu
  - But even so, it may be a barrier
  - It's also VM-friendly (that's the route I went)

NORTHWESTERN COLLEGE

End Result: Building a Foundation

- Do you want to change a graph type, or use a different visualization library? Add new pages to the fact book, or add new filters?
- I plan on developing new web apps beyond the fact book explorer – NSSE/CIRP survey data, admissions dashboards, etc.
- All of the above can use the foundation I've started with this code, and add new data sources, reactives, UI elements, etc.
  - But will create multiple web apps

NORTHWESTERN COLLEGE

Conclusions

- The code isn't necessarily easy, but once you get the hang of it, it is straightforward
  - There are syntax quirks… but it *is* R, so you knew that already ☺
- This solution isn't for everyone, but if you already use R, Shiny provides a free means of creating interactive data-driven web apps using a language you already know, allowing you to reuse code you've already written

NORTHWESTERN COLLEGE

Contact Info

- For a copy of the Shiny code demonstrated in this presentation, please contact me at wallinga@nwciowa.edu

NORTHWESTERN COLLEGE