Getting to Know Shiny

Ted Laderas (laderast@ohsu.edu)

September 10, 2015

Welcome to the Wide World of Shiny Visualization!

Shiny is a framework for making interactive visualizations using R. Nearly any plot in R can be made into an interactive visualization by adding some simple interface elements and mapping these interface elements into the plot. It's an extremely powerful technique for visualization.

Motivation

The LACE score is a summary score used in clinical data that is derived from 4 key clinical data elements in order to predict the probability of 30-day readmissions. Here are the individual elements of the LACE score:

- L = Length of stay
- A = acuity of admission
- C = Comorbidities (scored by severity, and additive)
- E = Number of Emergency room admits 6 months prior to admission

In order to calculate the LACE score, the value of these data elements are simply added together (LACE = L + A + C + E). The goal of today will be to build an interactive data explorer for the LACE dataset in order to assess the contribution of each data element to the predicted probability.

The main dependent variable we will be visualizing is the predicted probability of readmission after 30 days, which is derived from a logistical regression of the LACE score versus the readmit30 variable (0 = no readmission, 1 = readmission within 30 days) in the data.

If You Get Stuck

You can always refer to the full implementation of the LACE Explorer code here: http://church.ohsu.edu:3838/laderast/laceExplorer/ - there is the full implementation of everything I've discussed here.

Also, a lot of your fellow students are pretty R-savvy and can help you out if you get stuck.

Task 0 - Setting Up on Church (*Skip this step if doing on your own computer*)

If you are unfamiliar with Rstudio, here is a simple overview of the interface here: http://dss.princeton.edu/training/RStudio101.pdf.

1. Sign into the Rstudio instance on church: church.ohsu.edu:8787 If you do not have an account on church, please let me know so I can add you.
2. Create a "ShinyApps" directory in your home folder using the file interface. Case is important.
3. Change your working directory to the ShinyApps directory using getwd().

```
#remember that "~/" is shorthand for your home directory
setwd("~/ShinyApps")
```

4. Download the tutorial zip file using the following commands:

```
git clone http://github.com/laderast/shinyTutorial
```

5. Ensure you did everything correctly by opening a browser window to http://church.ohsu.edu:3838/*yourlogin*/shinyTutorial/ where *yourlogin* is your login name. Ensure that you see a blank page with the "LACE Score Explorer" title in your browser screen.

Task 0 - Setting up on your own machine

1. Make sure that you have the following packages installed:

```
library(dplyr)
library(shiny)
library(ggplot2)
```

2. Clone the Git Repository:

```
git clone http://github.com/laderast/shinyTutorial
```

3. Open up the .Rproj file in the `shinyTutorial` folder.

4. To test, open up the `ui.R` folder. Click the `Run App` button to ensure everything is running. Or you can just use the `runApp()` command once you've loaded the shiny library. Ensure that you get a blank web page with the "LACE Score Explorer" title in your browser screen.

```
library(shiny)
runApp()
```

Task 1 - Understanding the data and setting up the plot

1. Set your working directory to `~/ShinyApps/shinyTutorial/` (if on your own machine, you don't need to do this). Open the **ui.R**, **server.R**, and the **shinyTutorial.Rmd** files. You will be copying code from the Rmd file into **ui.R** and **server.R**
2. Load the LACE scores dataset into your R workspace as laceScores in order to examine it.

```
#load Data into working directory
#note that this data is not visible to the app
laceScores <- read.delim("lace-scores.txt")
```

2. Look at the head of the laceScores data. What do you notice about each column? Try running summary(laceScores) to get a better idea of the content of each column.

```
#look at the first few rows of LACE Scores
head(laceScores)
```

```
##   patientid Admit_date Discharge_date readmit30 L A C E LACE        fit
## 1     11182 2014-01-01     2014-01-16         0 6 3 1 2   12 -0.5996031
## 2     13301 2014-01-01     2014-01-06         0 4 0 1 1    6 -2.0632486
## 3       316 2014-01-01     2014-01-02         0 1 3 1 0    5 -2.2330915
## 4       674 2014-01-01     2014-01-03         0 2 3 2 2    9 -1.7029891
## 5      2946 2014-01-01     2014-01-17         0 6 0 1 0    7 -1.6490326
## 6      9660 2014-01-01     2014-01-08         0 5 0 2 2    9 -1.6660236
##       se.fit        UL         LL PredictedProb   readmitLevels
## 1 0.06530688 0.3842372 0.32572061    0.35443450 Not Readmission
## 2 0.04033923 0.1208736 0.10505168    0.11272051 Not Readmission
## 3 0.06089239 0.1077683 0.08687198    0.09681797 Not Readmission
## 4 0.06551114 0.1715634 0.13807266    0.15407528 Not Readmission
## 5 0.05812194 0.1772467 0.14642106    0.16123974 Not Readmission
## 6 0.05062217 0.1726725 0.14613490    0.15895506 Not Readmission
```

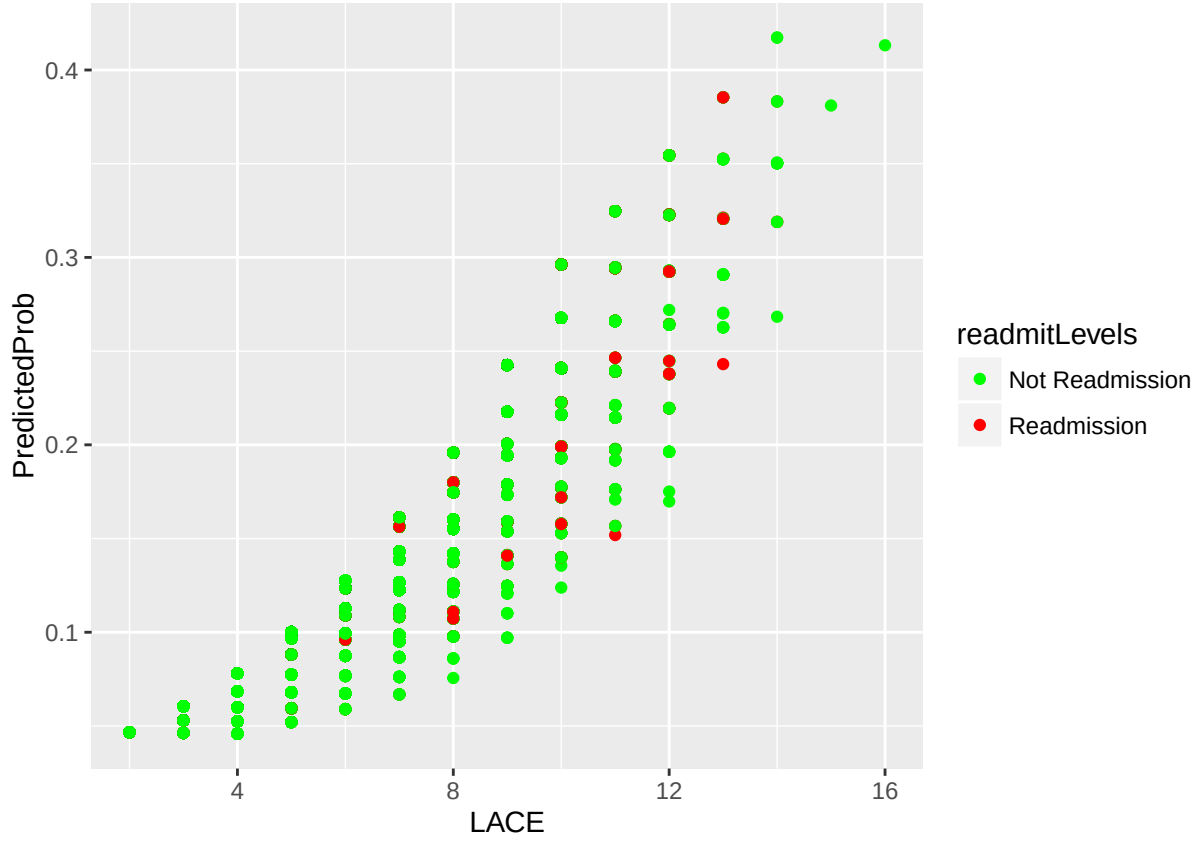
```
#look at the summary of the LACE Scores
summary(laceScores)
```

```
##    patientid          Admit_date       Discharge_date   readmit30
##  Min.   :    1   2014-01-01:14303   2014-01-04:2049   Min.   :0.0000
##  1st Qu.: 3576                      2014-01-02:1974   1st Qu.:0.0000
##  Median : 7152                      2014-01-03:1939   Median :0.0000
##  Mean   : 7152                      2014-01-05:1453   Mean   :0.1501
##  3rd Qu.:10728                      2014-01-06:1195   3rd Qu.:0.0000
##  Max.   :14303                      2014-01-07: 924   Max.   :1.0000
##                                     (Other)   :4769
##        L               A               C               E
##  Min.   :1.000   Min.   :0.000   Min.   :1.000   Min.   :0.0000
##  1st Qu.:2.000   1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000
##  Median :4.000   Median :0.000   Median :1.000   Median :0.0000
##  Mean   :3.627   Mean   :1.414   Mean   :1.447   Mean   :0.6853
##  3rd Qu.:5.000   3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :6.000   Max.   :3.000   Max.   :3.000   Max.   :7.0000
##
##       LACE             fit              se.fit              UL
##  Min.   : 2.000   Min.   :-3.0352   Min.   :0.03646   Min.   :0.05299
##  1st Qu.: 6.000   1st Qu.:-2.2145   1st Qu.:0.04205   1st Qu.:0.10868
##  Median : 7.000   Median :-1.8267   Median :0.05050   Median :0.15087
##  Mean   : 7.174   Mean   :-1.8417   Mean   :0.05338   Mean   :0.16299
##  3rd Qu.: 9.000   3rd Qu.:-1.4125   3rd Qu.:0.06207   3rd Qu.:0.20735
##  Max.   :16.000   Max.   :-0.3338   Max.   :0.16771   Max.   :0.48986
##
##        LL          PredictedProb            readmitLevels
##  Min.   :0.03878   Min.   :0.04586   Not Readmission:12156
##  1st Qu.:0.08687   1st Qu.:0.09845   Readmission    : 2147
##  Median :0.12566   Median :0.13864
##  Mean   :0.13811   Mean   :0.15011
##  3rd Qu.:0.18484   3rd Qu.:0.19585
##  Max.   :0.36592   Max.   :0.41730
##
```

3. Try running the plot code (make sure it looks like the plot below). How do we specify the x and y columns and the dataset in it? What do the colors specify?

```
#load in the ggplot2 plotting library
library(ggplot2)

#run the plot (this is the code you want to paste)
 ggplot(laceScores, aes_string(x="LACE", y="PredictedProb",
                               color="readmitLevels")) +
    geom_point() + scale_color_manual(values=c("Readmission"="red", "Not Readmission"="green"))
```




```
#end code to paste
```

4. Paste the plotting code into `server.R` where it says "#plotting code goes here". Note that you just need the ggplot statement, since we loaded the ggplot2 library at the beginning of the code.
5. Try loading your app in another browser window. Did you get an error? What is missing?
6. Paste the data loading code where "#data loading code goes here" in the server.R file. Why do we load the data at the beginning of the code?
7. Ensure that your basic plot works by opening up the app in another browser window. Please let me or fellow R experts know if you are having problems.

Task 2 - Add in a Select Box

So far, we are only visualizing two columns in the data set: *LACE score*, and the *predicted probability*. Let's add some functionality so we can start to evaluate the contribution of each data element.

1. The first thing we will be doing is adding the select box. Paste the following code into ui.R where it says "##selectInput code goes here".

```
selectInput("DataCol", "Data Covariate", choices =
                  list("Length of Stay"="L", "Acuity"="A", "Comorbidities"="C",
                       "ED Admits"="E", "LACE Score"="LACE"))
```

2. Ensure you put the interface element in correctly by loading your application again in another browser window. Try it out. Why isn't it working? You haven't yet connected it to anything in server.R yet.
3. Let's change the plotting code so we can change the x element in the plot. Remove the previous plotting code and paste the following in there:

```
ggplot(laceScores, aes_string(x=input$DataCol, y="PredictedProb", color="readmitLevels")) +
      geom_point()
```

4. Reload your app and try out the select box. Does it work now?
5. Before you proceed, understand what we just did. How did we map the different columns of laceScores into the x-element in the plot? How did we get the info from ui.R to server.R?
6. Let's set a default value for the slider. Read the documentation for `selectInput()` (use `?selectInput` to pull up the help page). Add a default value to your select box.
7. Ensure eveything is hunky-dory before you proceed any further.

Task 3 - Using Reactives and adding interactive filtering

We are now going to explore filtering using reactives and the dplyr package. The reactive is one way to add interactive filtering to the dataset. Instead of calling the `laceScores` data frame, we will call the `laceData()` reactive (note that the parentheses are mandatory to call the reactive version of the data).

1. Look at the reactive code for the laceData() reactive. Instead of using the actual `laceScores` data frame, we are going to use what's known as a *reactive* in shiny instead.

We use the dplyr package to provide filtering on different criteria. The "%>%" operator basically passes what is on the left side (in this case, our data frame, laceScores) to the next operation, which filters the data.

```
  #a quick dplyr demonstration.
  #test this out with different values of filterValue!
  library(dplyr)
  filterValue <- 5

  #filter out rows, returning only those those with a value of L which has filterValue or less
  out <- laceScores %>%
     filter(L <= filterValue)
  #return number of rows left
  nrow(out)
```

2. Let's change the plotting code to use the reactive. Remove the previous plotting code and paste the following into its place:

```
ggplot(laceData(), aes_string(x=input$DataCol, y="PredictedProb", color="readmitLevels")) +
    geom_point()
```

3. We want to filter out the data by length of stay. Add a slider after the `selectInput` element. *Note that we will have to add a comma after the `selectInput()` element, since we are passing a list of inputElements.*

```
sliderInput("LengthFilter", "Filter by Length of Stay", min=0, max=7, value=7)
```

4. Change the reactive code to use the input from the slider. Where do you add it into the reactive? Where is the sliderInput stored in input?
5. Reload your application and make sure that everything is peachy before moving on.

Task 4 (and beyond...)

This is where you start to be able to have fun and explore the data set and alternative visualizations.

Two resources that will be extremely helpful: Shiny Cheat Sheet and the ggplot2 Cheat Sheet

1. Add another plot into the `mainPanel`. For example, you can add a histogram that examines the distribution of LACE scores of the non-readmits versus the readmits. Where do you have to modify `ui.R` and where do you have to modify the `server.R` file?

```
#hint: use geom_hist(). You will only need to map the x aesthetic and the fill aesthetic.
```

2. Add text to the interactive plot using `annotate()` showing the number of readmits and non-readmits. This value should change when you filter the data.

```
#hint: use geom_text()
```

3. Add a `checkboxInput()` to filter out the non-readmits. (you will want to filter on the readmit30 column somehow in your reactive, since it is numeric.)

```
#hint: convert the true/false value the checkbox gives to a numeric value you can use for filtering.
```

4. Try changing the geom for the main plot to try another type of visualization, or add another geom, such as `geom_smooth()` (note that it will only work for the LACE score, since the other data elements have don't have that many possible values.)

```
#Hint: Look at ?geom_smooth() and ?geom_jitter() as possible geoms.
```