# Intro to Shiny Workshop

*Ted Laderas*

*November 21, 2016*

## 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

An experimental weight loss drug was first tested at one site with volunteers (Site A). Given the small sample size, volunteers from an additional site were recruited (Site B).

1) Your goal is to conduct EDA on the two separate sites to assess whether there was an effect from the weight loss drug.
2) Are there site-specific differences in weight loss or demographics?

### Getting Started

First things first: make sure that you have the shiny Package installed:

```
install.packages(shiny)
```

Clone or download the github repo for today's assignment:

```
git clone http://github.com/laderast/CSE631Shiny
```

Unzip (if necessary) and open this folder. Open the `CSE631Shiny.Rproj` file in RStudio.

### Task 1 - Understanding the data and setting up the plot

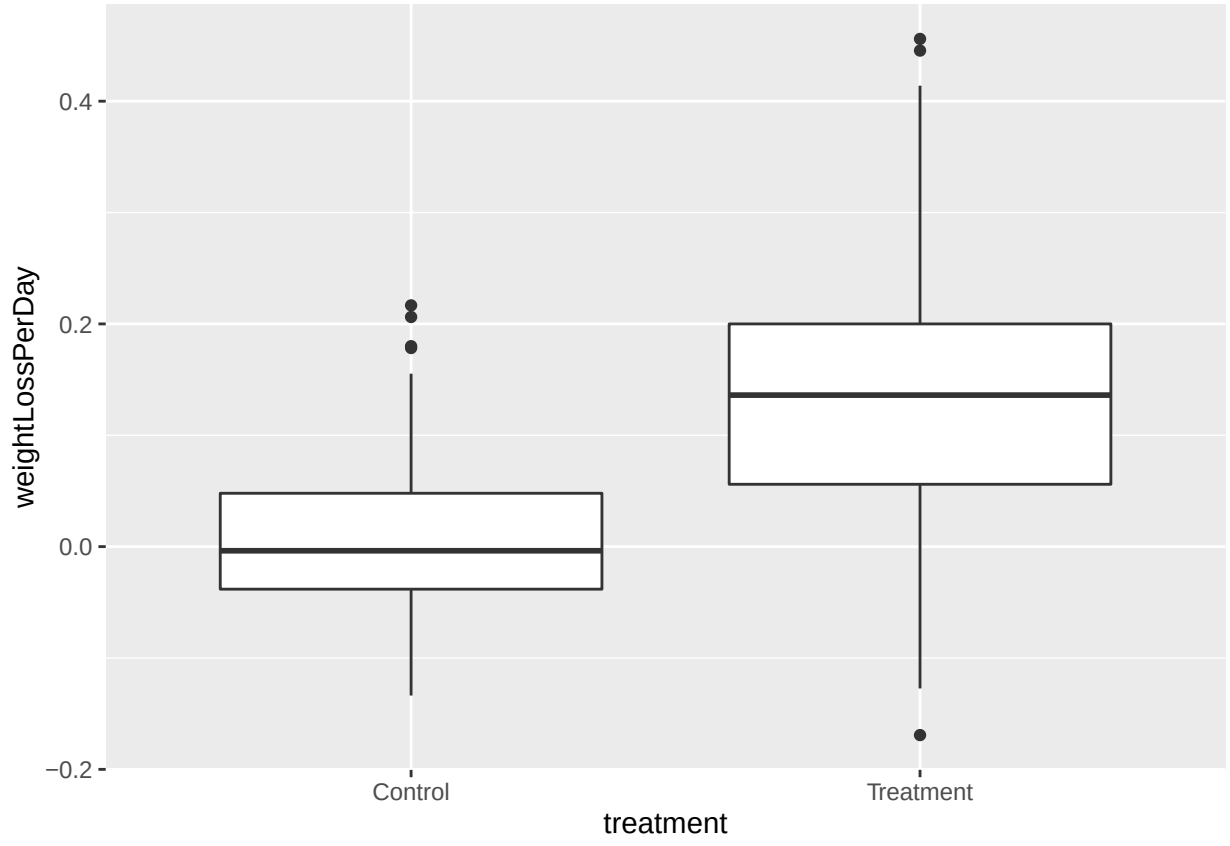
1. Once in the `CSE631Shiny` project, open the `ui.R`, `server.R`, and the `workshop.Rmd` files. For this tutorial, you will be copying code from the `workshop.Rmd` file into `ui.R` and `server.R`.
2. Load up the dataset and examine the covariates. What are you visualizing?

```
##Dataset loading code
weightLossData <- read.delim("weight-loss-study.txt")

summary(weightLossData)
```

```
##      treatment  weightLossPerDay   site      gender        age
##  Control  :61   Min.   :-0.16923   A:89   female:71   Min.   :20.00
##  Treatment:69   1st Qu.:-0.01081   B:41   male  :59   1st Qu.:24.00
##                 Median : 0.05657                      Median :27.00
##                 Mean   : 0.07953                      Mean   :40.85
##                 3rd Qu.: 0.15601                      3rd Qu.:67.75
##                 Max.   : 0.45594                      Max.   :88.00
```

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?

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

#run the plot (this is the code you want to paste)
 ggplot(weightLossData, aes_string(x="treatment", y="weightLossPerDay")) +
    geom_boxplot()
```




```
#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 load 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 for the X variable

So far, we are only visualizing two columns in the data set: `treatment`, and `weightLossPerDay`. Let's add some functionality so we can start to evaluate `weightLossPerDay` in the context of the other covariates.

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

```
selectInput("DataCol", "X covariate for Boxplot", choices =
                  list("treatment", "site", "gender"))
```

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? Well, you haven't yet connected it to anything in `server.R` yet, which does all of the statistical processing.

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 code in its place.

```
ggplot(weightLossData, aes_string(x=input$DataCol, y="weightLossPerDay")) +
      geom_boxplot()
```

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 `weightLossData` 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 select box. 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 to filter data

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 `weightLossData` data frame, we will call the `weightData()` reactive (note that the parentheses are mandatory to call the reactive version of the data). We'll use this reactive to filter our dataset by `age`.

1. Look at the reactive code for the `weightData()` 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 <- 45

#show number of rows in original dataset
nrow(weightLossData)

#filter out rows, returning only those those with a value of L which has filterValue or less
out <- weightLossData %>%
   filter(age <= 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(weightData(), aes_string(x="treatment", y="weightLossPerDay")) +
   geom_boxplot()
```

3. We want to filter out the data by `age`. Why would we want to do this? Add a slider after the selectInput element where it says . Note that we will have to add a comma after the `selectInput()` element, since we are passing a list of inputElements.

```
sliderInput("AgeFilter", "Filter by Age", min=20, max=88, value=45)
```

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-keen 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

Suggestions:

1. Change the dataset! (suggestions: `iris` or `ChickWeight`) Think about what you need to modify to accomodate the new dataset (i.e., should you transform the data into tidy data? How are UI elements going to change to accomodate these). Do you need to change code in `server.R` or `ui.R`?
2. Add some faceting code to your visualization using `facet_grid()`. Add another `selectInput` so you can select this facet.

```
#Hint: think about what inputs `facet_grid()` needs and how to put it together (you may have to use a
```

3. Add a `checkboxGroupInput` to filter out the data by site.
4. Play around with `global.R`. What are the advantages of putting data loading code in `global.R` versus `server.R` (hint: it has to do with `ui.R`)?

## Have Fun!

We've only scratched the surface. Here's a much more comprehensive tutorial: https://www.r-bloggers.com/building-shiny-apps-an-interactive-tutorial/

There are a ton more ways to modify these interactive visualizations. I usually take a look at the Shiny Gallery before I plan my apps. http://shiny.rstudio.com/gallery/

You can make your graphs way more interactive using the interactive tutorial (responding to brush, click, hover, and double-click events) here: http://shiny.rstudio.com/articles/plot-interaction.html

You can also make dynamic UI elements (that respond to data filtering and such) using `renderUI()`. Confusingly, `renderUI` is used in `server.R`, not `ui.R`.

If you want to make your graphs even more interactive (with tooltips), you can look into the `ggvis` package, which uses `Vega` underneath to make more interactive presentations: https://github.com/laderast/ggvis-intro

## Feedback

If you have any suggestions about how to improve this workshop, submit it as an issue to the GitHub: https://github.com/laderast/CSE631Shiny