Tips for Using Shiny

It took me a long time to understand inputs and outputs.

- Define things in server.R and use them in ui.R to show them in your app:
  - **output$*spam* = render*Bar*()** in server.R corresponds to ***bar*Output(*spam*)** in ui.R
- Use ui.R to let the app user define or select something, and use it in server.R:
  - ***thingy*Input(inputId = *"foo"*)** in ui.R corresponds to **input$*foo*** in server.R

The best way to start making shiny apps is to modify someone else's code. The following shiny related repositories can be found on github.com/debarros?tab=repositories

- WidgetList
  - extensive use of uiOutput and renderUI
- DirectCertLunches
  - input with whatever variable names you want
- ContinuedFractionConvergence
  - CSS in ui.R
  - plotOutput with a brush
  - observe statement that updates reactive values
- FlexAndScroll
  - custom CSS
  - customized widgets
  - shinyBS for collapsible panels
- BorderRadius
  - uiOutput with dynamically defined CSS style using a reactive value
- tabsetPanel2
  - uiOutput with static taglist defined in server.R
  - modified widget function stored in separate file
- CheckBoxColumns
  - uiOutput with dynamically defined CSS style using a reactive value
  - uiOutput used inside isolate statements in server.R
- Stack_DynamicUserInput
  - dynamically many inputs with last value defined by earlier values
- CSV.Checklist.Analysis
  - file inputs loaded in reactive objects
  - widgets defined in server.R and passed to ui.R using renderUI and uiOutput
  - tableOutput from renderTable defined using reactive objects and inputs

If you want to mess with how your app looks, you need to look into **Boostrap**. The front end generated by Shiny is based on the Boostrap framework (which has nothing to do with statistical bootstrapping)

- en.wikipedia.org/wiki/Bootstrap_(front-end_framework)
- www.w3schools.com/bootstrap/

Get to know **renderUI**. The ui.R code doesn't do a whole lot, and should mainly be used for laying out the interface. If any component of the interface needs to be calculated, that calculation needs to happen in server.R. renderUI allows you to dynamically define things in server.R that would have to be static if put in ui.R.