Co-lab Shiny Workshop

Interactive Data Tables and Plots

November 5, 2019

thomas.balmat@duke.edu
rescomputing@duke.edu

R provides a wide range of features and functions for transforming data, analyzing models, and presenting results. However, to evaluate models and results under a set of possible scenarios, the analyst must iteratively adjust parameter values and manually execute R scripts. Shiny offers a means of presenting controls (text prompts, selection lists, radio button groups, etc.) on a web page for prompting user input and dynamically integrating the user's web environment with the analytical portion of an R script. Instead of altering the values of R variables, the user adjusts on-screen graphical controls, R scripts are executed, and results presented, according to the particular Shiny features implemented in your app.

1 Overview

- Preliminaries
  - What can Shiny, interactive data tables, and integrated plots do for you?
  - What are your expectations of this workshop?
  - Interest in R Day?
- Links to material from prior sessions
  - Session 1 - Hello Shiny! `https://github.com/tbalmat/Duke-Co-lab/tree/master/Session-1`
- Example table/plot Shiny apps
- Resources
- Anatomy of a Shiny app
- Workshop material
  - From github (execute locally)
  - From RStudio Cloud
- Review previous app, OPM Central Personnel Data File
- HTML and debugging
- OPM CPDF overview using data tables and plots
  - Development of analysis in R
  - Shiny app with basic table and plot controls
  - Shiny app with additional table and plot controls

2 Example Data Table Apps

- Duke H2P2 GWAS phenotypic associations, http://h2p2.oit.duke.edu/H2P2/PhenotypicAssociations. This app queries a large database of phenotypic and genotypic associations, produces a summary table of results in order of significance, and renders a boxplot of individual phenotypic response values corresponding to a single SNP (row) selected from the table. The plot and data subset can be further examined using on screen controls.
- Shiny gallery, basic data table, https://shiny.rstudio.com/gallery/basic-datatable.html
- Review your data set on-line, https://shiny.rstudio.com/gallery/file-upload.html
- Shinyapps.io example, https://yihui.shinyapps.io/DT-rows/. This app highlights the plotted point corresponding to a selected data table row. Note the distinction between client and server side tables. The documentation for `renderDataTable()` (https://shiny.rstudio.com/reference/shiny/0.14/renderDataTable.html) states that only server side tables are implemented.

3 Resources

- R
  - Books
    - Norm Matloff, *The Art of R Programming*, No Starch Press
    - Wickham and Grolemund, *R for Data Science*, O'Reilly
    - Andrews and Wainer, *The Great Migration: A Graphics Novel*, https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1740-9713.2017.01070.x
    - Friendly, *A Brief History of Data Visualization*, http://datavis.ca/papers/hbook.pdf
  - Reference cards
    - R reference card: https://cran.r-project.org/doc/contrib/Short-refcard.pdf
    - Base R: https://rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf
    - Shiny, `ggplot`, `markdown`, `dplyr`, `tidy`: https://rstudio.com/resources/cheatsheets/
- Shiny
  - `?shiny` from the R command line
  - Click `shiny` in the `Packages` tab of RStudio
  - https://cran.r-project.org/web/packages/shiny/shiny.pdf
- `dataTables`
  - `?DT` from the R command line
  - Click `DT` in the `Packages` tab of RStudio
  - https://cran.r-project.org/web/packages/DT/DT.pdf
  - https://rstudio.github.io/DT/
  - java-centric: https://datatables.net/reference/option/
- `ggplot`
  - `?ggplot2` from the R command line
  - Click `ggplot2` in the `Packages` tab of RStudio
  - https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf
- Workshop materials: https://github.com/tbalmat/Duke-Co-lab/tree/master/Session-2

4 Anatomy of a Shiny App

A Shiny app is an R script executing in an active R environment that uses functions available in the Shiny package to interact with a web browser. The basic components of a Shiny script are

- `ui()` function
  - Contains your web page layout and screen objects for inputs (prompt fields) and outputs (graphs, tables, etc.)
  - Is specified in a combination of Shiny function calls and raw HTML
  - Defines variables that bind web objects to the execution portion of the app
- `server()` function
  - The execution portion of the app
  - Contains a combination of standard R statements and function calls, such as to `apply()`, `lm()`, `ggplot()`, etc., along with calls to functions from the Shiny package that enable reading of on-screen values and rendering of results
- `runApp()` function
  - Creates a process listening on a tcp port, launches a browser (optional), renders a screen by calling the specified `ui()` function, then executes the R commands in the specified server() function

5 Access Workshop Material

5.1 Execute Locally (copy workshop material from github repo)

- Copy scripts and data from https://github.com/tbalmat/Duke-Co-lab/tree/master/Session-1
- In a separate directory, copy scripts and data from https://github.com/tbalmat/Duke-Co-lab/tree/master/Session-2
- Install Shiny package: R command `install.packages("shiny")`
- Install plotting package: R command `install.packages("ggplot2")`
- Install data tables package: R command `install.packages("DT")`
- All workshop scripts should function locally

5.2 RStudio Cloud

- What is RStudio Cloud?
  - *We* [RStudio] *created RStudio Cloud to make it easy for professionals, hobbyists, trainers, teachers and students to do, share, teach and learn data science using R.*
- With RStudio Cloud
  - You do not need RStudio installed locally
  - Packages and data are available without installation and transfer
- Access workshop material
  - Create an Account: https://rstudio.cloud
  - Workshop project link: https://rstudio.cloud/project/580472
- All workshop scripts should function on RStudio Cloud, except OS shells that are used to automate execution

6 Review OPM CPDF Overview App from Session 1

- Material: https://github.com/tbalmat/Duke-Co-lab/blob/master/Session-1/CourseOutline/Co-lab-Session-pdf
- Discussion: section 11
- Development of analysis in R: section 11.1
  - Required libraries
  - Use of `lapply()` as the `by` parameter of `aggregate()`
  - `ggplot()` features
    - Conditional, stepwise construction of plot (`g <- g + ...`)
    - `aes_string()` is used for aesthetic declaration, since supplied variables contain text referencing actual variable names
    - `size` and `color` are specified as aesthetics
    - `facet_wrap` is specified with `panelVar`, as opposed to `panelVar`, since
    - `panelVar` contains text referencing the actual variable to panel by
    - Panel labels are customized using `as_labeller()`
    - Axis labels are formatted with a function
    - A prepared theme (`ggTheme`, a list) is used to control general appearance
- Shiny app, version 1: section 11.2
  - Considerations
    - Required libraries
    - Maintaining `ui()` and `server()` functions in separate files
    - `runApp()` features
      - `launch.browser=T`
      - `appDir`
      - `host`
      - `port`
    - Terminating the app (esc key, stop sign)
    - Residual browser effects
    - `ui()` features
      - `fluidPage()`
      - `tabsetPanel`
      - `sidebarPanel`
      - `fluidRow()`
      - `column()`
      - Use of `HTML()`
      - Common error (`"Warning: Error in tag: argument is missing, with no default"`) when delimiting comma missing between parameters of `fluidPage()`, `fluidRow`, `column()`, etc.
      - The `Agency` selection list is constructed from values observed in the data
    - `server()` features
      - Since referencing `input$variableName` within `renderDataTable()` or `renderPlot()` establishes a *reactive* environment, where the associated table or plot would be instantaneously updated whenever `input$variableName` is modified, we instead use `observeEvent()` functions bound to action buttons and pass `input$variableName` to separate functions for data aggregation and plot generation.

    - Communicating with your app: `print()` and `cat()`

- Shiny app, version 2: section 11.3
  - `ui()` features
    - Conditional panel to hide slider bar (`t1PlotSlider`) when graph independent variable is FY
    - `t1PlotSlider` is configured with animation options with a timer interval of 500 ms
  - `server()` features
    - An observe event for `t1PlotSlider` generates a plot for each year in the slider range, in 500 ms intervals (it would be interesting to update this interval dynamically)
    - `ignoreInit=T` is specified for the `t1PlotSlider` observe event, which prevents rendering of a plot as the slider transitions from NULL to 1988 during initialization

7 HTML and Debugging

Sections 9 and 10 of *Co-lab-Session-1-NPDHist-CPDF.pdf*

8 CPDF Overview Using Data Tables and Plots

Shiny's input objects (`textInput`, `selectInput`, `sliderInput`, etc.) provide a means of presenting, to the user, placeholders for values to be used in an R script for querying data sets, subsetting observations, specifying model parameter values, configuring plot variables, and whatever else may be needed to conduct supported analyses. Additionally, using an appropriate update function (`updateSelectInput()`, `updateRadioButtons()`, etc.) the list of possible values presented to the user can be modified based on ranges of values observed in the data. Along with a fixed set of controls, we can provide a summarized view of the data, in tabular form, and use each row as a virtual menu entry that, when selected, will advance the user to another level of analysis, limited to data corresponding to that row. For instance, with the CPDF observations, we might summarize median pay by agency and occupation in a table with one row per agency, occupation combination. When a row is selected, we can advance the user to a plot configuration screen where various relationships between pay, fiscal year, grade, education, and age can be viewed for the selected agency, occupation subset. This approach separates the analysis into two phases: a summary phase to present the structure of the data and an investigation phase to explore relationships within key data subsets identified in the summary phase.

8.1 Development of Analysis in R

The OPM CPDF data set used here includes human capital characteristics on full-time U.S. federal employees in the GS pay plan for fiscal years 1988 through 2011. The data were sourced from Buzzfeed, who received the data from OPM in response to a FOIA request. For information on Buzzfeed and their hosting of these data, see https://www.buzzfeednews.com/article/jsvine/sharing-hundreds-of-millions-of-federal-payroll-records. Table 1 lists the variables included in the data. Additional information on data elements and their meaning is available in the OPM Guide to Data Standards, https://www.opm.gov/policy-data-oversight/data-analysis-documentation/data-policy-guidance/reporting-guidance/part-a-human-resources.pdf. Restrictions on observations used here are:

- FY between 1988 and 2011
- WorkSchedule=F
- PayPlan=GS
- Grade between 01 and 15
- OccupationCategory in P, A, T, C, O
- EducationLevel between 01 and 22
- AdjustedBasicPay > 10 (thousands per year)
- Top five agencies (left two positions) by observation frequency

Table 1: Buzzfeed OPM data set

Column	Description
PseudoID	unique (OPM randomly assigned) employee ID
FY	U.S. federal government fiscal year
Agency	federal agency employed (synthetically generated for workshop)
Grade	general schedule (GS) grade
OccupationalCategory	occupational category
Occupation	occupation
Age	employee age (five year increments, noised induced by OPM)
EducationYears	years of education
BasicPay	adjusted basic pay, in 2011 $U.S.

Important features of federal employee human capital include a general increase in age, education, pay grade, and pay throughout the study period along with a decrease in persons occupying clerical positions (occupational category "C") and an increase in persons occupying professional and administrative positions (occupational categories "P" and "A"). These trends should be clearly revealed in our app. The R script for development and analysis is in workshop file /App/V1/CPDF-Tables-1.r. Features and considerations include (line numbers are approximate):

- (lines 53-55) For demonstration and to limit data load time, one of four data sets is used, each containing observations for a randomly selected set of approximately one fourth of the total number of employees represented in the data
- (lines 92-122) A table (data frame labeled `aggdat`) of aggregated mean and quartiles is constructed using specified dependent and independent variables
- (lines 125-164) An alternative aggregation method truncates `agency` and `occupation` to two positions to achieve a high level of aggregation)
- (lines 166-198) One row is selected from `aggdat` and the observations corresponding to the specified independent variables are subset from the disaggregated data set (all observations read)
- The selected observation subset are used as the source to `ggplot()` to prepare a box plot using the dependent variable specified during aggregation and an independent variable (R script variable `gindepVar`) different from any used in the aggregation step
- (lines 180-187) `gindepVar` is coerced to a factor to avoid the problem of `ggplot` producing a single element x-axis when the `gindepVar` x is continuous
- Occupational category, if specified as an aggregation or graphical independent variable, is coerced to a factor with levels specified in the OPM standard order P, A, T, C, O
- The plot (`g`) is constructed in a step-wise manner, as a template for applying further conditional geoms and appearance features
- (lines 200-280) A more developed plot is prepared that represents what we need for the Shiny app. Feature include:
  - Faceting based on a specified panel variable (`panelVar`)
  - Specification of the number of panel rows or columns to arrange (`panelRows`, `panelCols`)
  - Ability to display points for individual observations (`pointDisplay`)
  - Ability to specify point transparency (`pointAlpha`) to diminish the effect of point overlay
  - Ability to specify a point coloration variable (`diffVar`) to aid in distinguishing categories of observations

- (lines 250-260) Jitter (`geom_jitter()`) is added in the x-dimension to diminish the effect of point overlay
- (line 257) The point color aesthetic is implemented by direct editing of the list produced by `ggplot`, but only if `diffVar` is true. This is an example of conditional geom modification.
- (line 258) Actual point category colors are generated using a `colorRampPalette` from blue to red
- (line 263) Normal display of outlier points is suppressed in `geom_boxplot`
- (line 264) Error bars (`stat_boxplot()`) are included to indicate the inter-quartile range of each independent variable level
- (line 269) A custom function is defined to label facet panels
- (line 273) A previously prepared theme (`ggTheme`) is used to control overall plot appearance

8.2 Shiny App With Basic Table and Plot Controls

Table, grade by FY, reveals increase in mean grade. increase in median more pronounced.

8.3 Shiny App With Additional Table and Plot Controls