Co-lab Shiny Workshop

Interactive Data Tables and Plots

October 22, 2020

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.

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.

1 Overview

- Preliminaries
  - What can Shiny, interactive data tables, and integrated plots do for you?
  - What are your expectations of this workshop?
- Examples
  - Example visualizations
  - Example Shiny apps
- Resources
- Anatomy of a Shiny App
- Example Table and Plot Shiny Apps
- Workshop Material
  - From RStudio Cloud
  - From github (execute locally)
- Review Apps From Previous Session

  - Graph delivery map
  - U.S. Office of Personnel Management (OPM) Central Personnel Data File
- HTML and Debugging
- OPM CPDF Analysis 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 features

2 Examples

2.1 Visualizations

- `ggplot` gallery: `https://www.r-graph-gallery.com/all-graphs.html`
- `ggplot` extensions: `https://mode.com/blog/r-ggplot-extension-packages`

2.2 Shiny apps

- Duke Data+ project, *Big Data for Reproductive Health*, `http://bd4rh.rc.duke.edu:3838`
- Duke Data+ project, *Water Quality Explorer*, `http://WaterQualityExplorer.rc.duke.edu:3838`
- Duke Med H2P2 Genome Wide Association Study, `http://h2p2.oit.duke.edu`
- Duke Nicholas School, *Health Effects of Airborne Pollutants*, `http://shindellgroup.rc.duke.edu`
- Duke Nursing School, *Urea Cycle Disorder SNOMED/RxNorm Graph Associations*
- Shiny gallery: `https://shiny.rstudio.com/gallery/`

2.3 Example Data Table apps

- Duke H2P2 GWAS phenotypic associations, `http://h2p2.oit.duke.edu/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 material: 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

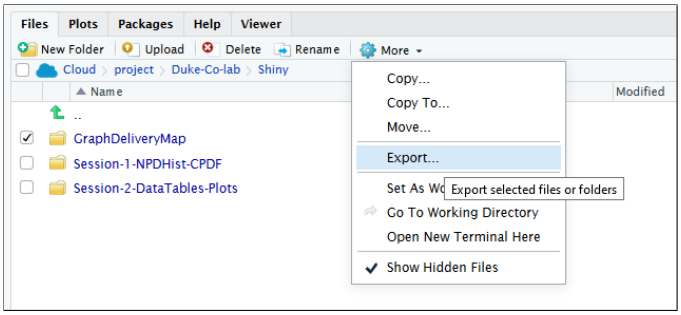
- `server()` function
  - The execution portion of the app
  - Contains a combination of standard R statements and function calls, such as to `apply()`, `lm()`, `ggplot()`, etc., along with calls to functions from the Shiny package that enable reading of on-screen values and rendering of results

- `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

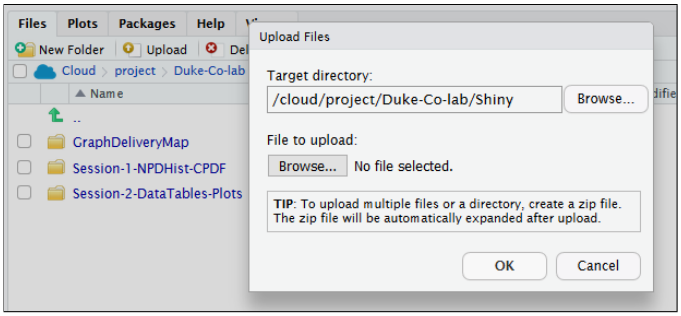
5 Workshop Material

5.1 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, example scripts, and data sets needed for the Shiny workshop are available in the Cloud environment
- Access workshop material
  - Create an account: `https://rstudio.cloud`
  - Create a convenient directory structure in your project
  - Copy workshop materials
    - Go to `https://rstudio.cloud/project/1768881` (Duke-Co-lab/Shiny directory)
    - Export a sub-directory



    - Save to your computer
    - Upload saved zip file to your RStudio Cloud project



  - Install packages:
    - Shiny: `install.packages("shiny")`
    - ggplot: `install.packages("ggplot2")`
    - Data Tables: `install.packages("DT")`
- All workshop scripts should function in RStudio Cloud, except OS shells that are used to automate execution

5.2 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 packages:
  - Shiny: `install.packages("shiny")`
  - ggplot: `install.packages("ggplot2")`
  - Data Tables: `install.packages("DT")`
- All workshop scripts should function locally

6 Review Apps from Previous Session

6.1 Graph delivery map

Material:

- https://rstudio.cloud/project/1768881 (Duke-Co-lab/Shiny/GraphDeliverMap subdirectory)
- https://github.com/tbalmat/Duke-Co-lab/tree/master/Examples/GraphDeliveryMap

This is a geographical location cluster modeling app that simulates a production facility, demand sites (customers), and carriers who make deliveries. Customer locations are clustered using the `kmeans()` function and carriers are assigned to clusters based on nearest proximity. Key features are:

- Use of `map_data()` for U.S. state boundaries
- `geom_polygon()` for display of state boundaries
- User selectable inputs for number of carriers and customers and service area latitude, longitude, and service area radius
- Input controls are reactive, so that changes on-screen cause immediate re-simulation of demand, clusters, and map
- Colors and shapes specified in `ggplot()` to distinguish clusters, facility, carrier, and regional centers

6.2 OPM Central Personnel Data File

- Material
  - RStudio Cloud
    - Outline: https://rstudio.cloud/project/1768881 (Duke-Co-lab/Shiny/Session-1-NPDHist-CPDF/CourseMaterial/Outline directory)
    - Scripts: https://rstudio.cloud/project/1768881 (Duke-Co-lab/Shiny/Session-1-NPDHist-CPDF/App/CPDF directory)
  - RStudio Cloud
    - Outline: https://github.com/tbalmat/Duke-Co-lab/blob/master/Session-1/CourseOutline
    - Scripts: https://github.com/tbalmat/Duke-Co-lab/blob/master/Session-1/App/CPDF
- Discussion: CourseOutline/Co-lab-Session-1-NPDHist-CPDF.pdf, 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 OPM CPDF Analysis Using Data Tables and Plots

The OPM Central Personnel Data File (CPDF) data set includes human capital characteristics on full-time U.S. federal employees in the GS pay plan. The data were sourced from Buzzfeed, who received it from OPM in response to a FOIA request.[1] Table 1 lists the variables included in the data.[2]

Restrictions on observations used 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.

To analyze 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 observation 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 (V1)

Script location (RStudio Cloud): https://rstudio.cloud/project/1768881
File: Duke-Co-lab/Shiny/Session-2-DataTables-Plots/App/V1/CPDF-Tables-1.r

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 proportion persons occupying clerical

[1] 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

[2] Additional information on CPDF data elements 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

positions (occupational category "C") and an increase in proportion 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 randomly selected, 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 175) One row is selected from `aggdat` and the observations corresponding to the specified independent variables are subset from the disaggregated data set (all observations)
- (line 178) An independent (x-axis) variable for the graph is specified (in `gindepVar`) that is different from any used in the aggregation step
- (lines 180-187) A data frame is prepared to be used as the data source of `ggplot()`
- (lines 184, 186) `gindepVar` is coerced to a factor to avoid the problem of `ggplot()` producing a single element x-axis when x (`gindepVar`) is continuous
- (line 186) Occupational category, if specified as an aggregation or graphical independent variable, is coerced to a factor with levels specified in the standard order P, A, T, C, O, for proper appearance
- (lines 189-196) 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 is closer to what is needed for the Shiny app. Features 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 non-empty. 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
- (line 280) Examining the structure of a composed `ggplot()` list reveals the inner structure of a plot and which list elements affect particular plot features, along with an idea of the behavior achieved by using certain parameter values (examination of `g` revealed the element `g[["layers"]][[length(g[["layers"]])]][["mapping"]][["colour"]]`, used on line 257 to apply the color aesthetic for `diffVar`)

Figure 1 is an example plot from this analysis. It reveals the expected general change (increase) in employee grade vs. education, within the Department of Health and Human Services, for occupation code 02, in fiscal years 1988-2011. The expected reduction in clerical positions (occupational category "C") throughout this period is also noticeable.

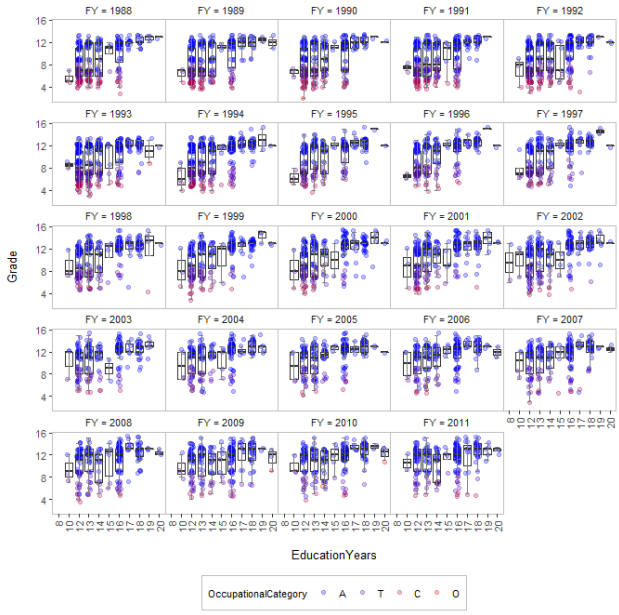


Figure 1: Change in employee grade, given education, along with change in distribution of occupational category, Health and Human Services, occupation code 02, FYs 1988-2011

8.2 Shiny app with basic table and plot controls (V2)

Script location (RStudio Cloud): https://rstudio.cloud/project/1768881
File: Duke-Co-lab/Shiny/Session-2-DataTables-Plots/App/V2/CPDF-Tables-2.r

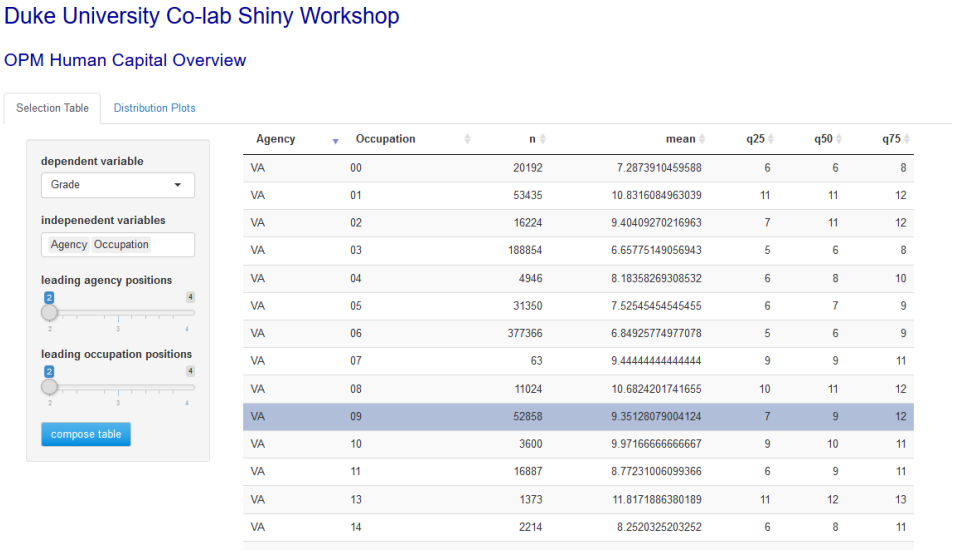
The analysis of section 8.1, implemented in R, reveals important features of federal employee human capital. To expedite and broaden the scope analysis of these data, we will develop a Shiny app that produces a summary table for specified dependent and independent variables. From the table, a row can be selected to subset observations for detailed plotting. Plot configuration includes controls to specify variables for the graph independent (x-axis) variable, paneling, and point category coloration. Additional controls are included for panel layout (rows/columns), point size, point jitter, and point transparency. Features and considerations include (line numbers are approximate):

- User interface (file `ui.r`)
  - (lines 57-60) A subset of observations is assembled corresponding to approximately one fourth of the employees represented in the data
  - (lines 62-66) Record cleaning is accomplished
  - (lines 70, 127) An agency selection list is constructed from agencies appearing in the data (disabled in this version)
  - (line 92) A `tabsetPanel` is configured for two tabs
  - (lines 94-116) The aggregation and selection tab (`t1`) is defined
  - (line 103) Multiple independent variables are selectable simultaneously (`t1IndepVar`). Note that the selection list is static. It can also be composed from the column names of the input data (as with `t2AgencyFilter` and `agencyList`).
  - (line 109) An action button (`t1ActionComposeTable`) is defined for initiating table construction
  - (line 114) The data table object (`t1Table`) is defined (selecting a row will activate the plot tab)
  - (lines 118-168) The plot configuration and review tab (`t2`) is defined
  - (line 147) An action button (`t2ActionPlot`) is defined for initiating plot construction
  - (line 156) The plot object (`t2Plot`) is defined (this is where plots will be displayed)
  - (line 164) A text output line (`t2Msg`) is defined for messaging
- Server (file `server.r`)
  - (lines 55-134) An observe event is defined for the action button `t1ActionComposeTable`. This event is executed whenever the state of the button changes (when it is clicked).
  - (line 62) It is confirmed that a dependent and at least one independent variable are selected. If violated, then (lines 128-129) the table is set to NULL (it disappears) and a message is displayed).
  - (lines 64-78) Observation subset indices are composed for each combination of levels of the independent variable(s) specified on the table tab (`t1IndepVar`). The result (`iagg`) is a list with one element per independent var level combination and is saved as a global variable to be made available outside of current function. Note the use of `lapply()` for the `by()` clause of `aggregate()`. A list is required here and it will contain one element per independent variable. This is responsible for observation subsetting.
  - (line 80) Subset index validation is accomplished
  - (lines 82-95) Observation count, mean, and quartiles are aggregated for each index subset.
  - `apply()` proceeds through each set of indices in `iagg` and produces a list of data frames that are then flattened into a single data frame (`aggdat`).
  - (line 97) `aggdat` validation is accomplished

- (lines 99-110) A data table is constructed and rendered (in `t1Table`), using `aggdat` results. Features include (this might be a good time to review the `dataTables` and `renderDataTable()` documentation listed in section 3):
  - Fixed page length (100) with no page length adjustment object (`bLengthChange=F`)
  - The global table search tool is disabled (bFilter=F)
  - Single row selection (multiple rows are possible, yielding a list that grows with selection, until it is reset)
  - Automatic column width assignment (based on max width of contained data)
  - The order of the second column is set to ascending (note that col IDs are 0-based)
- (lines 112-129) Messages are displayed to the user if no observations exist corresponding to the input subset parameter values
- (lines 137-234) A function is defined to generate a plot based on a selected `aggdat` table row and configuration parameter values taken from the plot configuration tab (`t2`). Features include:
  - (lines 144-148) Dependent, independent, and aggregation variables are validated (specification of a single variable as having multiple roles indicates an invalid plot structure). Invalidation causes a NULL plot to be returned, along with display of an error message (lines 220-232)
  - (line 151) The observation subset indices corresponding to the selected row are retrieved
  - (lines 153-176) A plot source data frame (`gdat`) is composed from the subset observations. Variables are converted to factors as needed and occupation category is order according to standard.
  - (lines 178-215) A composite box plot with overlayed points (for individual observation sin the data subset) is composed
  - (lines 183-184) Points are displayed with x-jitter to aid in distinguishing multiple points at shared coordinates. Note the use of `aes_string()`, since the x and y coordinates are contained in `gdat` indicated by `indepVar` and `depVar` (use of `aes()` would cause a search of `"indepVar"` and `"depVar"` in the column names of `gdat`).
  - (lines 185-194) The color differentiation aesthetic is assigned by direct editing of the `ggplot()` result list. This is an alternative to complex use of various `aes()` calls when aesthetics are conditional (two conditional aesthetics, one for color and one for size, would require four separate `aes()` calls embedded in if-then-else statements; three aesthetics would require eight statements, etc.).
  - (line 197) The box plot is generated with outliers suppressed (perhaps this should be done only when points are requested)
  - (line 198) error bars are included (although specified with fixed color and width, these can be made adjustable with additional on-screen controls)
  - (lines 200-210) Panels are produced, one for each level of a specified variable. A custom labeler function is specified.
  - (lines 212-215) Y-axis labels are formatted (thousands) and a previously prepared theme is applied
  - (line 218) The plot is rendered (displayed in `t2Plot`)
  - (lines 237-259) An action event is defined to `t1Table` row selection
  - (line 246) The selected row ID is saved in a global variable (`t1SelectedRow`) so that other functions may reference the currently selected row
  - (lines 248-253) A call to `t2RenderPlot` is made to compose and display a plot using the selected table row id and the current values of on screen objects. Note that the values of Shiny variables (preceded by `input$`) are passed as parameters instead of directly referencing Shiny variables within `t2RenderPlot` to avoid any possible attempt to create a reactive environment, where `t2RenderPlot` might be called when a Shiny variable is modified (this is not supposed to occur, but it does)
  - (line 256) Make the plot tab (`t2`) active, so that the user can view and configure the resulting plot

* (lines 262-279) An action event is defined for the `t2ActionPlot` button. This renders a plot using the previously specified table row (`t1SelectedRow`) and current on-screen parameter values

Figure 2 is an example screen shot of the aggregated summary table tab of the V2 app. Row highlighting indicates an observation subset (agency VA and occupation 09 here) selected for plotting. Figure 3 is an example screen shot of the plotting tab, with a plot composed from the observation subset corresponding to the row selected in figure 2.



Agency	Occupation	n	mean	q25	q50	q75
VA	00	20192	7.2873910459588	6	6	8
VA	01	53435	10.8316084963039	11	11	12
VA	02	16224	9.40409270216963	7	11	12
VA	03	188854	6.65775149056943	5	6	8
VA	04	4946	8.18358269308532	6	8	10
VA	05	31350	7.52545454545455	6	7	9
VA	06	377366	6.84925774977078	5	6	9
VA	07	63	9.44444444444444	9	9	11
VA	08	11024	10.6824201741655	10	11	12
VA	09	52858	9.35128079004124	7	9	12
VA	10	3600	9.97166666666667	9	10	11
VA	11	16887	8.77231006099366	6	9	11
VA	13	1373	11.8171886380189	11	12	13
VA	14	2214	8.2520325203252	6	8	11

Figure 2: Screen shot of V2 CPDF Shiny app, aggregated summary table tab

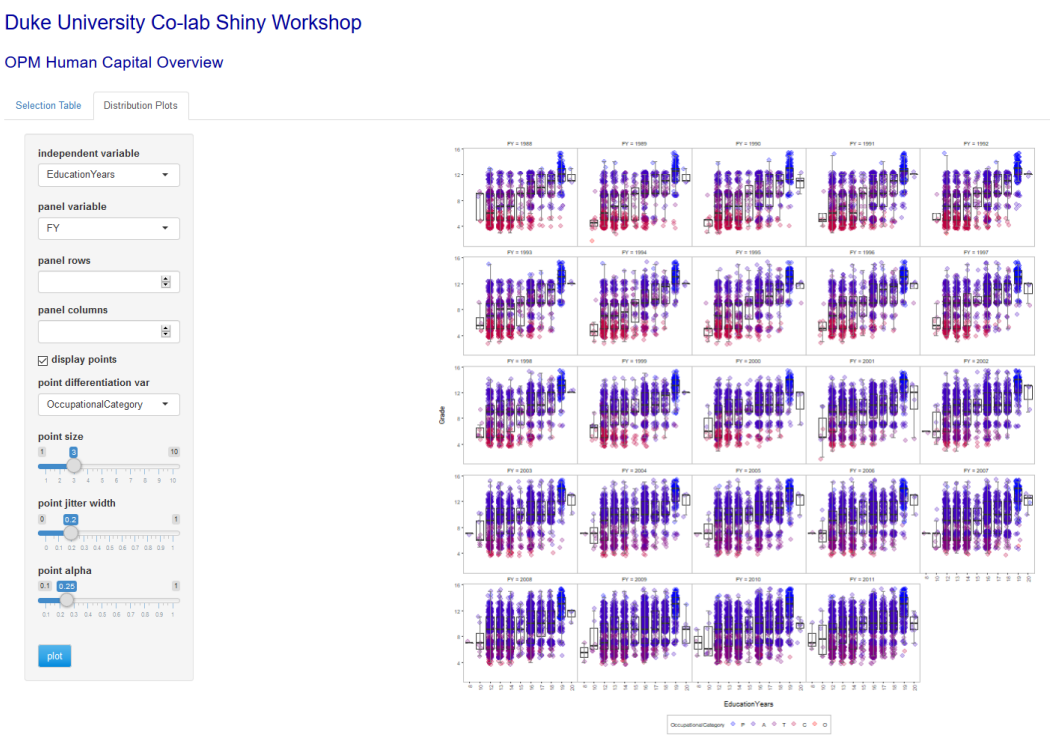


Figure 3: Screen shot of V2 CPDF Shiny app, plot configuration tab

Observations using the aggregation table:

- Aggregation of grade by fiscal year reveals a pattern of increased grade by, with median grade exhibiting greater increase than mean grade (does this indicate a widening of grade distribution with a tail forming for upper grades?)
- Aggregation of age by fiscal year reveals a pattern of increased age, indicating an older (more experienced?) workforce at the end of the study period
- Aggregation of education by fiscal indicates an increase in mean and median education over the study period
- Aggregation of grade by agency and occupational category indicates that VA employees tend to have lower grade than employees of similar category in other agencies

How can the plot tab be used to investigate the above observations to reveal differences between agencies, occupational categories, fiscal years, and so forth? Note that it would be interesting to explore a data table of grade aggregated by occupational category, then plot with fiscal year as the independent variable and agency as a panel variable. However, although our app allows truncation of agency on the aggregation tab, it does not on the plot tab. Therefore, panels are constructed of individual plots using four position agency codes, giving a somewhat detailed view. Yet, individual agencies with general increase in grade over the period are easily identified for each selected occupational category, which is informative.

8.3 Shiny app with additional table and plot features (V3)

Script location (RStudio Cloud): https://rstudio.cloud/project/1768881
File: Duke-Co-lab/Shiny/Session-2-DataTables-Plots/App/V3/CPDF-Tables-3.r

This version of the app implements the following features (affected script locations are indicated as u-nnn-ppp or s-nnn-ppp, where u indicates ui.r, s indicates server.r, nnn indicates the approximate beginning line number, and ppp indicates the approximate ending line number):

- Table features
  - (u-88) Browser tab title added
  - (u-90-95) Notification window formatted and moved to the center of the screen
  - (s-78-245) Table construction converted to a callable function
  - (s-234, 241) Query criteria errors reported using `showNotification()` with `type="error"`
  - (s-203) Default rows per page modified
  - (s-185-197) Rows per page adjustment control enabled
  - (s-185-197, 206) Global table search function enabled (with regex capability)
  - (s-178, 185-197, 225) csv download feature implemented
  - (s-122-134) HTML anchor tags implemented for agency and occupation links to reference sources
  - (s-156-159, 405-443) Row selection converted to cell selection (to prevent plot generation when clicking an anchor)
  - (s-217-222) Default order of rows limited to columns corresponding to independent variables
  - (s-141-145, 211-213) Numeric columns formatted
  - (s-161-166) Column headers customized with HTML
  - (s-169) Cell editing enabled
  - (s-175) Column search fields enabled
  - (s-147-150) HTML formatted caption displayed (composed from selected dependent and independent variables)

- Plot features
  - (s-364, 371) Errors reported using `showNotification()` with `type="error"`
  - (s-356-358) Progress indicator is displayed during rendering
  - (s-258-262, 376) Existing plot cleared prior to rendering new one (attempted)
  - (s-335-347) Observation subset count displayed in each facet panel
  - (u-148-149, s-304) Box intensity control added
  - (u-150-151, s-316-324) Point display (yes/no) control replaced with option for fixed or variable sized points (variable size are based on independent variable category frequency)