Co-lab Shiny Workshop

Interactive Data Tables and Plots

March 31, April 7, 2021

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

R provides a wide range of features and functions for transforming data, fitting models to data, 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 and initiates an action (a button or immediate computation) then 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.

In this session, we will develop a Shiny app to explore covariate relationships in a human capital data set by selecting subsets of observations indicated by rows of a summary aggregation table. Aggregation variables are specified by the user and corresponding data subsets are used to construct various plots based on further selection of dependent and independent variables. Values of on-screen Shiny controls are used in an R script to control table and graph construction. Design of the interaction between on-screen Shiny controls and internal R script variables using knowledge of the domain and data being studied is the key to producing an effective data exploration app. Development will begin with an R script (using no Shiny features) to demonstrate the types of analyses that will be converted into dynamic Shiny apps. A basic Shiny app will be developed from the analysis script and it will be extended in a series of apps that include increasingly useful Shiny and R features. The final version is representative of a professional, visually appealing app acceptable for published research. Important features include:

- Data table row selection
- Interactive data table formatting and search features (`DT` package)
- Interactive plot construction from data table subset (row) selection
- Control of `ggplot` parameters using on-screen Shiny controls
- Draggable panels for user control of screen layout
- Inspection and modification of themes for attractive app appearance
- User file search and upload capability
- Download link for review of sample input file format
- Aggregation table download capability
- Progress indicators
- Dynamic insertion of HTML windows containing supplemental project info and links

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
  - Example table and plot Shiny apps
- Resources
- Shiny Use Cases
- Anatomy of a Shiny App
- Reactivity
- Download Workshop Material and Configure R
- Example Data Set - U.S. Office of Personnel Management (OPM) Central Personnel Data File
  - Explore Increase in Mean Age with Fiscal Year
  - Explore Increase in Mean Age with Fiscal Year, Paneled by Grade
  - Explore Increase in Mean Age with Fiscal Year, Paneled by Grade, Differentiated by Occupational Category
  - Explore Increase in Proportion Employees by Grade and Fiscal Year
- Development of a Shiny App With Data Tables and Plots
  - Development of Analysis in R
  - Shiny App with Basic Table and Plot Features
  - Shiny App with Updated Appearance Using Navigation Bars and Themes
- Plot Animation - Fiscal Year Slider Bar
- Debugging

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 H2P2 project, *Host Pathogen Genome Wide Association Study*, `http://h2p2.oit.duke.edu`
- Duke iCPAGdb project, *Cross-Phenotype Analysis of GWAS*, `http://cpag.oit.duke.edu`
- Duke Nicholas School, *Health Effects of Airborne Pollutants*, `http://shindellgroup.rc.duke.edu`
- 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.
- Duke iCPAGdb review tab, `http://cpag.oit.duke.edu`. This app presents a table with each row describing a set of precomputed cross-phenotype analysis results. Computation time required to produce a selected analysis prohibits effective browsing of results. Further, presenting descriptions in a convenient, selectable format draws attention to important findings. Complete data used in a selected analysis are displayed in tables and heatmap plots.
- 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.

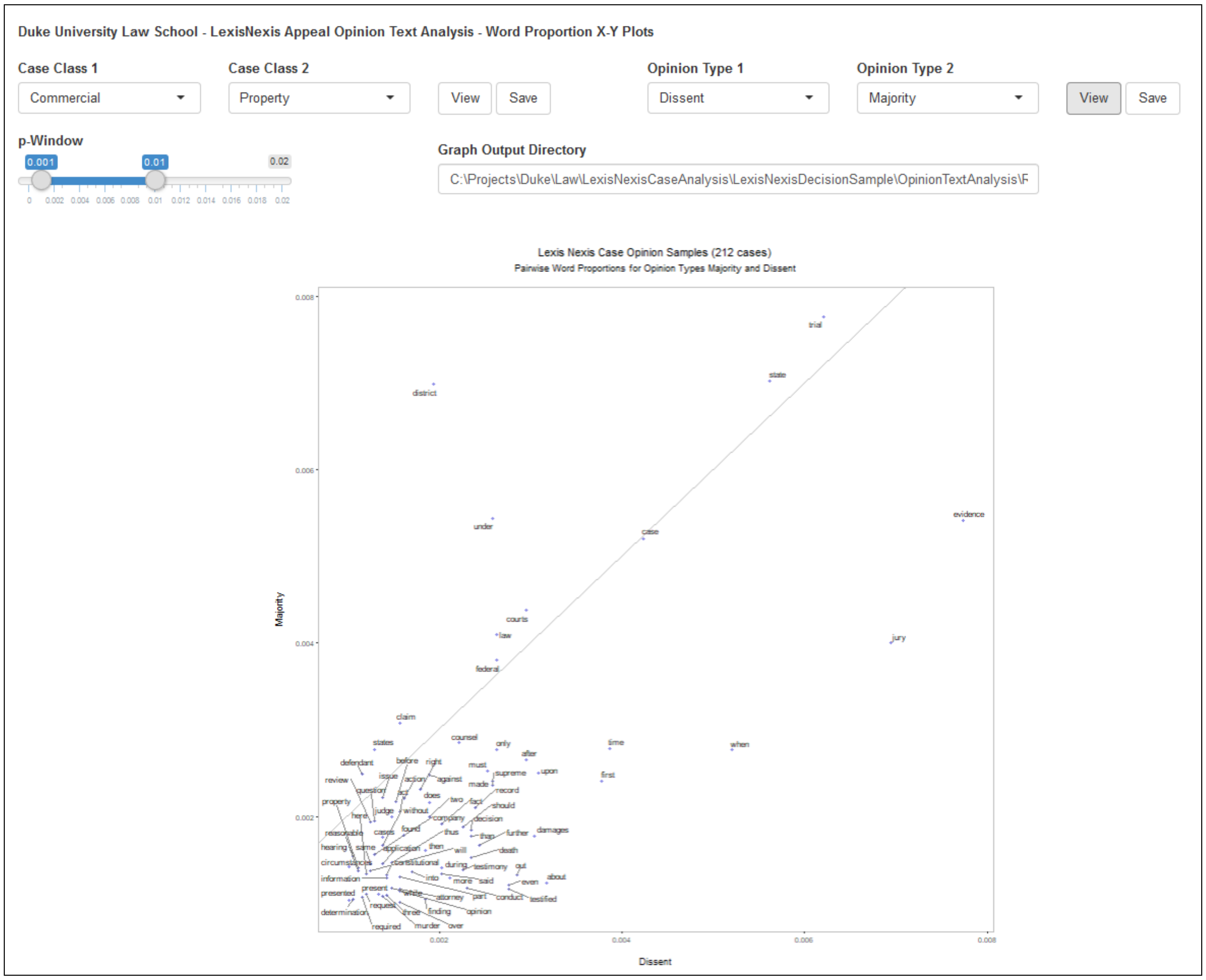
3 Resources

- 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`

4 Shiny use cases

4.1 Personal Use

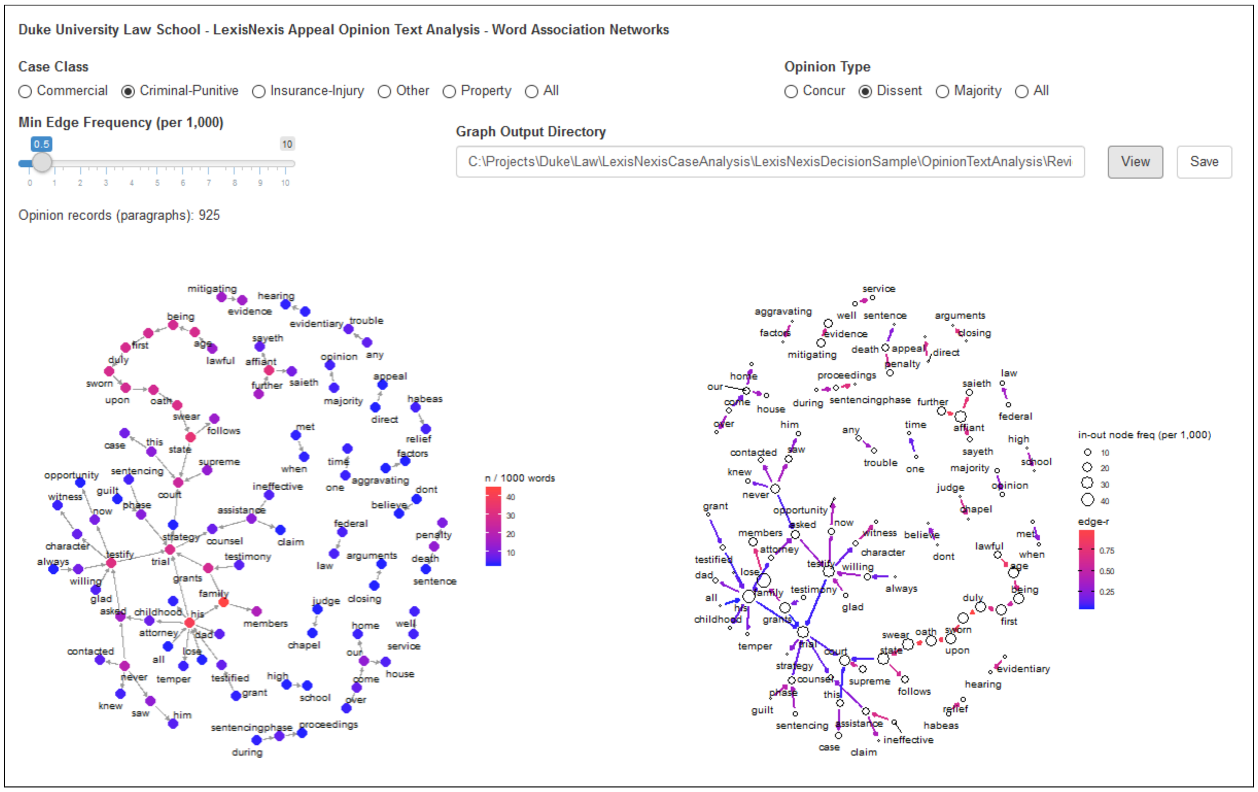
Shiny can expedite identification of model and visualization parameter values that illustrate important properties of systems that you want to document or publish. Instead of repetitious modification of "hard coded" values in scripts, followed by execution, Shiny can re-execute a script using parameter values taken from on-screen prompts (text inputs, radio buttons, slider bars, etc.). Figure 1 is a screen-shot of a Shiny app that was developed to compare incidence proportion of words appearing in case opinions of two categories. The adjustable p-window parameter aids in identifying filtering bounds on proportion that reveal significant associations (high p in both dimensions), while eliminating spurious ones (low p in either dimension). Slider bar adjustment of p-value replaces iterative R script editing and manual plotting. Shiny scripts are available at https://github.com/tbalmat/Duke-Co-lab/tree/master/Examples/Law/OpinionWordProportionXY


Duke University Law School - LexisNexis Appeal Opinion Text Analysis - Word Proportion X-Y Plots

Case Class 1: Commercial
Case Class 2: Property
Opinion Type 1: Dissent
Opinion Type 2: Majority
p-Window: 0.001 – 0.01
Graph Output Directory: C:\Projects\Duke\Law\LexisNexisCaseAnalysis\LexisNexisDecisionSample\OpinionTextAnalysis\F

Figure 1: Shiny app developed to explore two-way incidence proportions of words in legal text.

Figure 2 is a screen-shot from another legal text analysis app that identifies, for various case and opinion types, edge frequencies that reveal word pair proportions and correlation. Choice of small edge frequency values fails to reveal all important associations, while large values cause a cluttered graph that hides primary associations. The edge frequency slider input replaces iterative R script parameter value specification and manual plotting. Shiny scripts are available at https://github.com/tbalmat/Duke-Co-lab/tree/master/Examples/Law/OpinionTextCorrelationGraph


Duke University Law School - LexisNexis Appeal Opinion Text Analysis - Word Association Networks

Case Class

Commercial [✓ Criminal-Punitive] Insurance-Injury Other Property All

Opinion Type

Concur [✓ Dissent] Majority All

Min Edge Frequency (per 1,000): 0.5

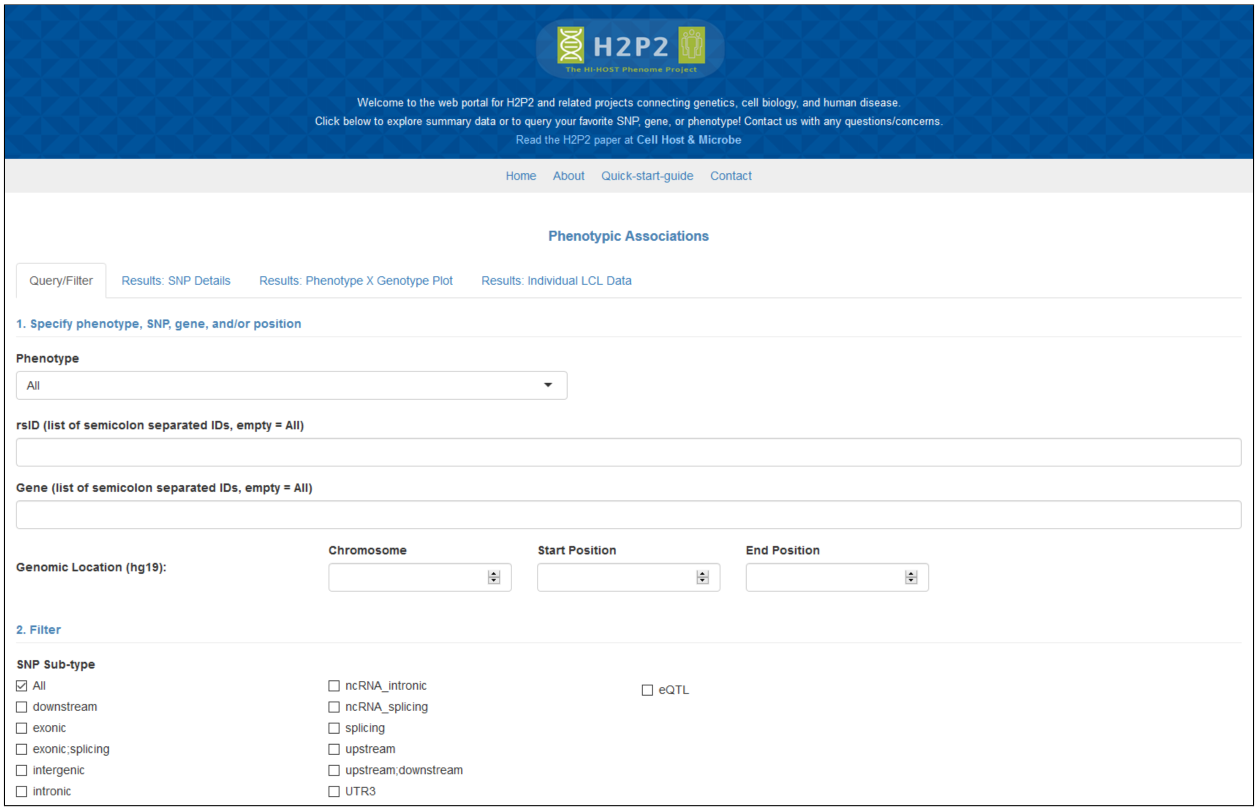
Graph Output Directory: C:\Projects\Duke\Law\LexisNexisCaseAnalysis\LexisNexisDecisionSample\OpinionTextAnalysis\Revi

Opinion records (paragraphs): 925

Figure 2: Shiny app developed to explore leading and trailing word pair correlation in case opinion text, using a graph with filtered edge density.

4.2 Public App

Alternatively, apps can be developed for public use, so that others can investigate your data and models and become better informed from your research. Figure 3 is a screen-shot from an app developed by the H2P2 (Hi-Host Phenome) Project at Duke. Researchers from around the world use it to explore associations between 150 cell pathogens and 15,000,000 chromosomal positions. The site url is http://h2p2.oit.duke.edu. Example Shiny scripts are available at https://github.com/tbalmat/Duke-Co-lab/tree/master/Examples/H2P2



H2P2
The HI-HOST Phenome Project

Welcome to the web portal for H2P2 and related projects connecting genetics, cell biology, and human disease.
Click below to explore summary data or to query your favorite SNP, gene, or phenotype! Contact us with any questions/concerns.
Read the H2P2 paper at **Cell Host & Microbe**

Home About Quick-start-guide Contact

Phenotypic Associations

Query/Filter | Results: SNP Details | Results: Phenotype X Genotype Plot | Results: Individual LCL Data

1. Specify phenotype, SNP, gene, and/or position

Phenotype
All

rsID (list of semicolon separated IDs, empty = All)

Gene (list of semicolon separated IDs, empty = All)

Genomic Location (hg19): **Chromosome** **Start Position** **End Position**

2. Filter

SNP Sub-type

☑ All	☐ ncRNA_intronic	☐ eQTL
☐ downstream	☐ ncRNA_splicing	
☐ exonic	☐ splicing	
☐ exonic;splicing	☐ upstream	
☐ intergenic	☐ upstream;downstream	
☐ intronic	☐ UTR3	

Figure 3: Shiny app developed to query and plot results from a Duke hosted genome wide association study.

5 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

6 Reactivity

Reactivity is the single most important feature that Shiny offers. Variables are defined in your `ui()` function with an `input$` prefix and when these variables appear in `observe()` functions within in your `server()` function, execution events are triggered by on-screen changes to the corresponding `ui()` variables. In addition to referencing `input$` variables `observe()` functions include standard R commands, including those supported by any valid R package, so that the reactive variables become parameters to your R functions, enabling dynamic analysis of data. Output is rendered in the app by targeting `ui()` variables defined with an `output$` prefix. A simple example follows. It has a single, numeric input (`x`) and one plot output (`plot`). Changes in `x` cause the `observeEvent()` to be executed. The `observeEvent()` generates a histogram of `x` random, normal values. The histogram is a suitable input value to `renderPlot()`. Assignment of the `renderPlot()` result to `output$plot` causes the histogram to be displayed as defined in `ui()`. Notice how a Shiny input variable (`input$x`) is used as a parameter to an R function (`rnorm()`) and the result of an R function (`plot()`) is used as a parameter to a Shiny function (`renderPlot()`). Note that modifying `x` to its current value does not cause execution of the `observeEvent()` (try it).

```
library(shiny)

# Define uI
ui <- function(req) {
  numericInput("x", "x"),
  plotOutput("plot")
}

# Define server function
server <- function function(input, output, session) {
  observeEvent(input$x, {
    output$plot <- renderPlot(hist(rnorm(input$x)))
  })
}

# Execute
runApp(list("ui"=ui, "server"=server), launch.browser=T)
```

7 Download Workshop Material and Configure R

- Copy course outline, scripts, and data from https://github.com/tbalmat/Duke-Co-lab/tree/master/Spring-2021/Session-1-2-DataTables-Plots
  - App.zip
  - Co-lab-Session-1-2-DataTables-Plots.pdf
  - Data.zip
  - SupplementalMaterial.zip
- Expand zip files (one subdirectory per file)
- Launch RStudio
- Install packages:
  - Shiny: `install.packages("shiny")`
  - Shiny: `install.packages("shinythemes")`
  - Shiny: `install.packages("shinyjs")`
  - ggplot: `install.packages("ggplot2")`
  - Data Tables: `install.packages("DT")`

8 Example Data Set - OPM CPDF

The U.S. Office of Personnel Management (OPM) maintains records on the careers of millions of current and past federal employees in what is called the Central Personnel Data File (CPDF). The Human Capital and Synthetic Data projects at Duke have conducted various research using these data. Although the complete data set contains data elements that are private and not released to the public, OPM has released data sets with private elements omitted and with certain variables (age, for instance) induced with statistical noise. We will review a subset of results made available by Buzzfeed. The accuracy of the publicly available elements has been confirmed by comparison of data procured by Duke through FOIA requests. The data used here are highly aggregated, so that analysis is limited to broad patterns, typically involving means of pay, age, education, etc. for large groups of employees. Additional information on the OPM data set, research at Duke, and Buzzfeed is available at

- The Office of Personnel Management: https://www.opm.gov/
- 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
  - https://github.com/tbalmat/Duke-Co-lab/blob/master/Docs/US-OPM-Guide-To-Data-Standards.pdf
- Duke Synthetic Data Project, Annals of Applied Statistics paper:
  - https://projecteuclid.org/euclid.aoas/1532743488
  - https://github.com/tbalmat/Duke-Co-lab/blob/master/Docs/AOAS1710-027R2A0.pdf
- U.S. Federal Grade Inflation supplement to Synthetic Data paper (section 9): https://github.com/tbalmat/Duke-Co-lab/blob/master/Docs/SynthDataValidationSupplement.pdf
- Buzzfeed OPM data: https://www.buzzfeednews.com/article/jsvine/sharing-hundreds-of-millions-of-federal-payro

Notes on the data set:

- Observations are limited to:
  - Fiscal years 1988 through 2011
  - General schedule (GS) grades 01 through 15
  - Full-time employees
  - Grade between 01 and 15
  - Occupational category in P, A, T, C, O
  - Education level between 01 and 22
  - Adjusted basic pay > 10 (thousands per year)
  - Top five agencies (left two positions) by observation frequency
- Columns include:
  - fy - U.S. federal government fiscal year
  - agency - federal agency employed (synthetically generated for workshop)
  - age - employee age (five year increments, noised induced by OPM)
  - grade - general schedule (GS) grade
  - occCat - occupational category
  - yearsEd - years of education
  - n - number of observations (employees) in fy, agency, age, grade, occCat, yearsEd combination
  - sumPay - sum of basic pay in fy, agency, age, grade, occCat, yearsEd combination (in 2011 $U.S.)
- There is one record for each unique combination of fy, agency, age, grade, occCat, yearsEd combination

- n and sumPay are aggregated within fy, agency, age, grade, occCat, yearsEd combinations

For motivation, we will use a Shiny app to expose a few trends in the data. 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 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 the output produced by our app.

Executing the app:

- From the `V9-CPDF-FYSliderBar` directory (downloaded in section 7)
  - Edit `ui.r` and modify the statement

    `setwd("C:/Projects/Duke/Co-lab/Shiny-Spring-2021/Session-1-2-DataTables-Plots/App/V9-CPDF-FYSliderBar")`

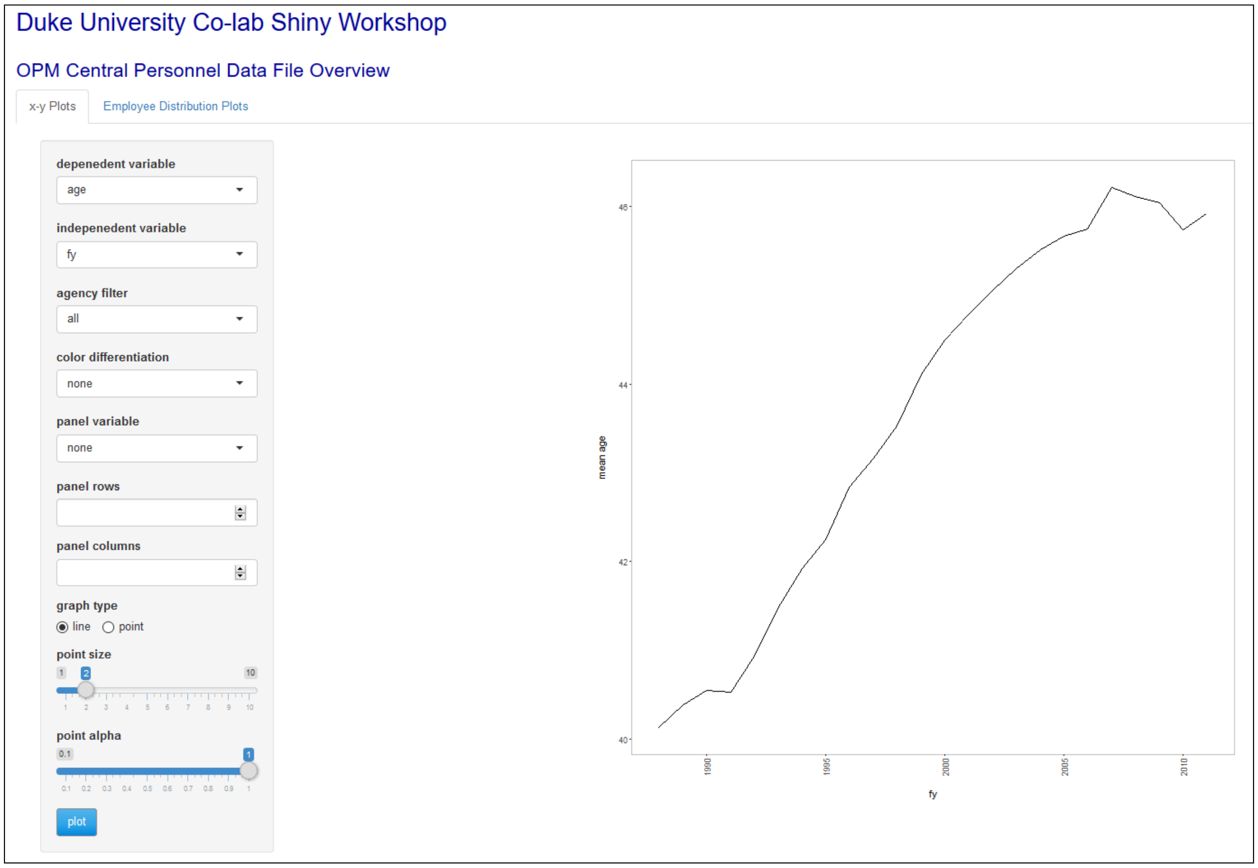
    to reference the `V9-CPDF-FYSliderBar` directory on your computer
  - Edit `CPDF-FYSliderBar.r` and modify the statement

    `ad <- "C:/Projects/Duke/Co-lab/Shiny-Spring-2021/Session-1-2-DataTables-Plots/App/V9-CPDF-FYSliderBar"`

    to reference the `V9-CPDF-FYSliderBar` directory on your computer
- Load `CPDF-FYSliderBar.r` into RStudio
- Execute the loaded script

8.1 Motivation Graph 1, Increase in Mean Age with Fiscal Year

On the x-y Plots tab of the app, select `age` for dependent variable and `fy` for independent variable then click the `plot` button. As seen in figure 4, mean aggregate age does increase with fiscal year throughout the study period.



Duke University Co-lab Shiny Workshop

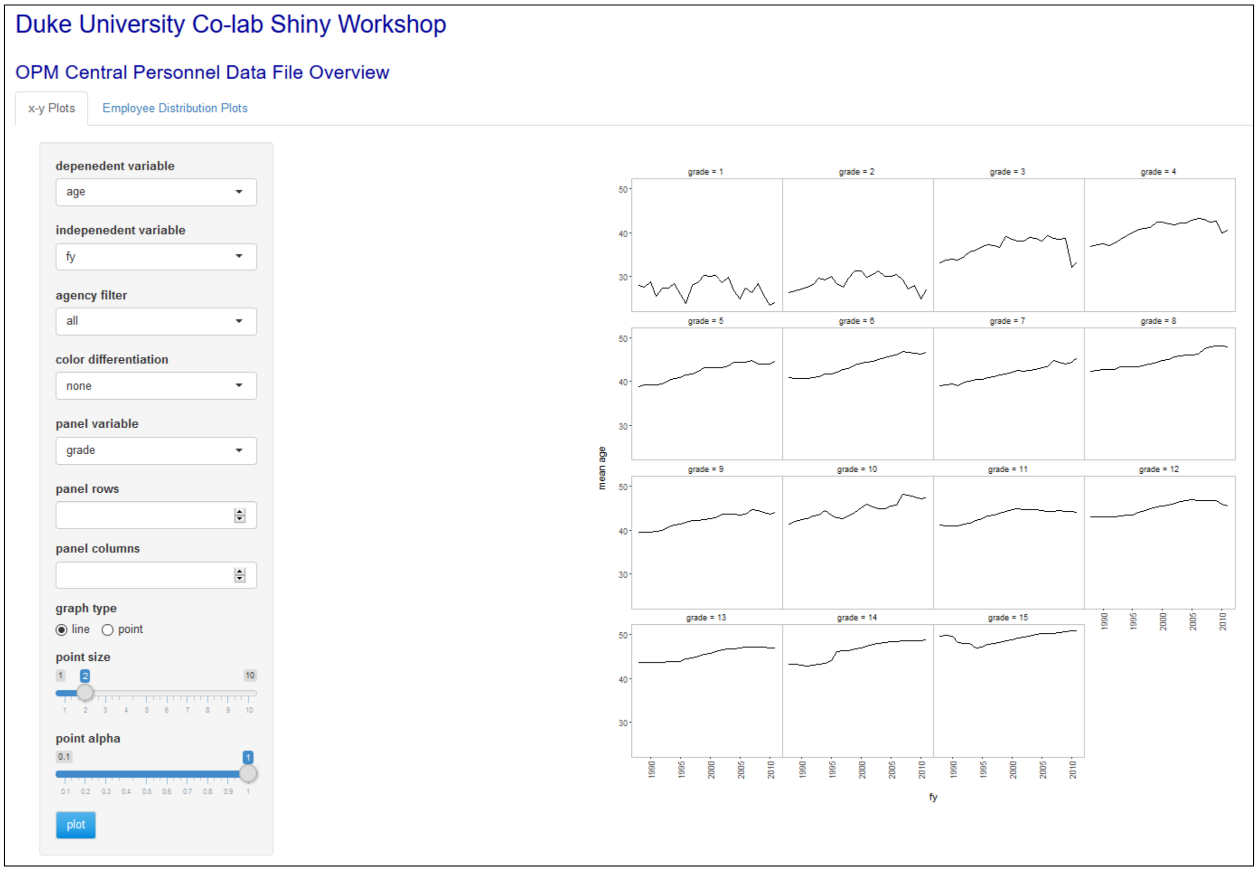
OPM Central Personnel Data File Overview

dependent variable: age
indepenedent variable: fy
agency filter: all
color differentiation: none
panel variable: none
panel rows:
panel columns:
graph type: [✓ line] point
point size: 2
point alpha: 1

Figure 4: OPM CPDF graph of mean employee age vs. fiscal year

8.2 Motivation Graph 2, Increase in Mean Age with Fiscal Year, Paneled by Grade

Selecting `age` for dependent variable, `fy` for independent variable, and `grade` for panel variable produces figure 5. This panel reveals differences in mean age by grade (generally, greater age with grade) and slight variation in the increase of mean age by fiscal year for various grades.


Duke University Co-lab Shiny Workshop

OPM Central Personnel Data File Overview

x-y Plots | Employee Distribution Plots

depenedent variable: age

indepenedent variable: fy

agency filter: all

color differentiation: none

panel variable: grade

panel rows:

panel columns:

graph type: [✓ line] point

point size: 2 (1–10)

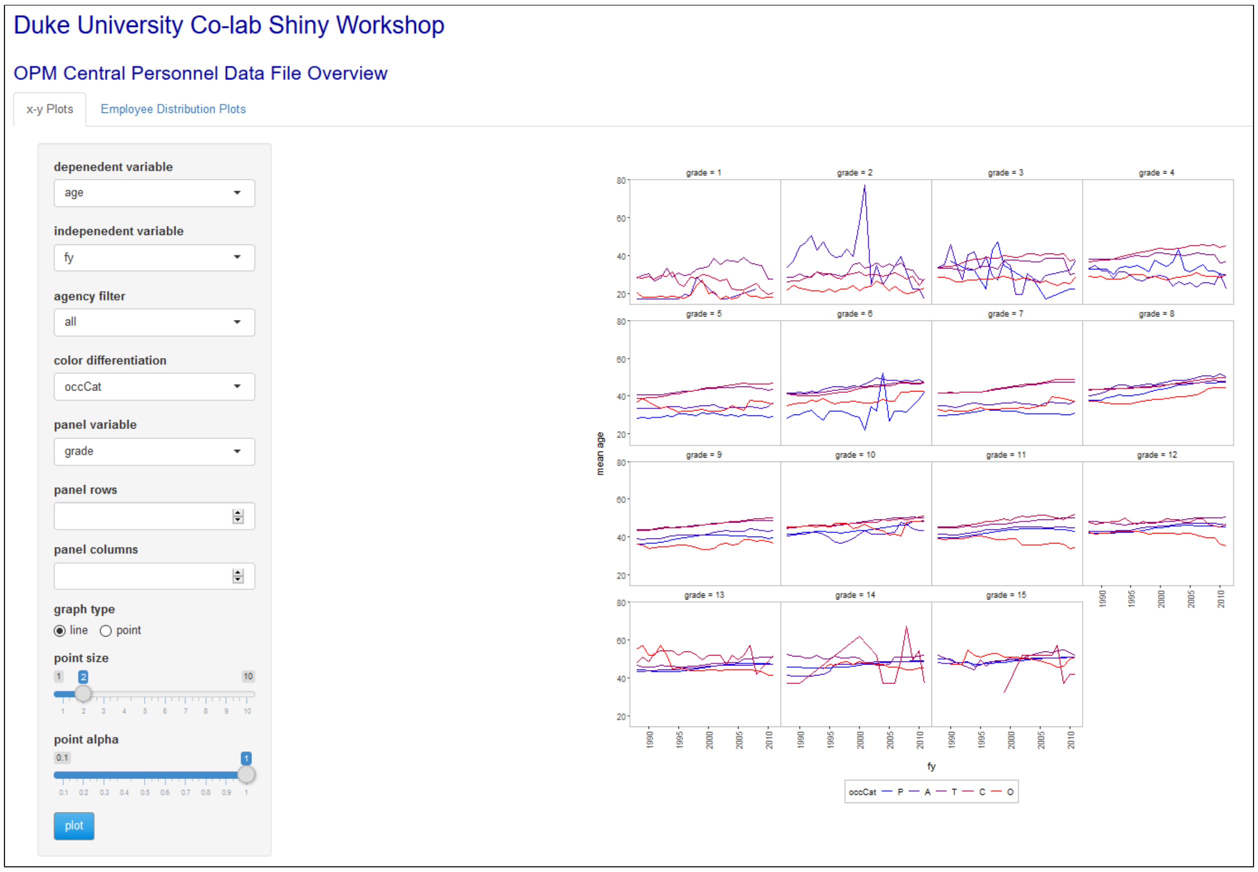
point alpha: 1 (0.1–1)

plot

Figure 5: OPM CPDF graph of mean employee age vs. fiscal year, paneled by grade

8.3 Motivation Graph 3, Increase in Mean Age with Fiscal Year, Paneled by Grade, Differentiated by Occupational Category

Selecting `age` for dependent variable, `fy` for independent variable, `grade` for panel variable, and `occCat` for color differentiation produces figure 6. This panel reveals further differences in the rate of change in mean age by fiscal year for various grades and occupational categories.


Duke University Co-lab Shiny Workshop

OPM Central Personnel Data File Overview

x-y Plots | Employee Distribution Plots

depenedent variable: age

indepenedent variable: fy

agency filter: all

color differentiation: occCat

panel variable: grade

panel rows

panel columns

graph type: [✓ line] point

point size: 1 2 10

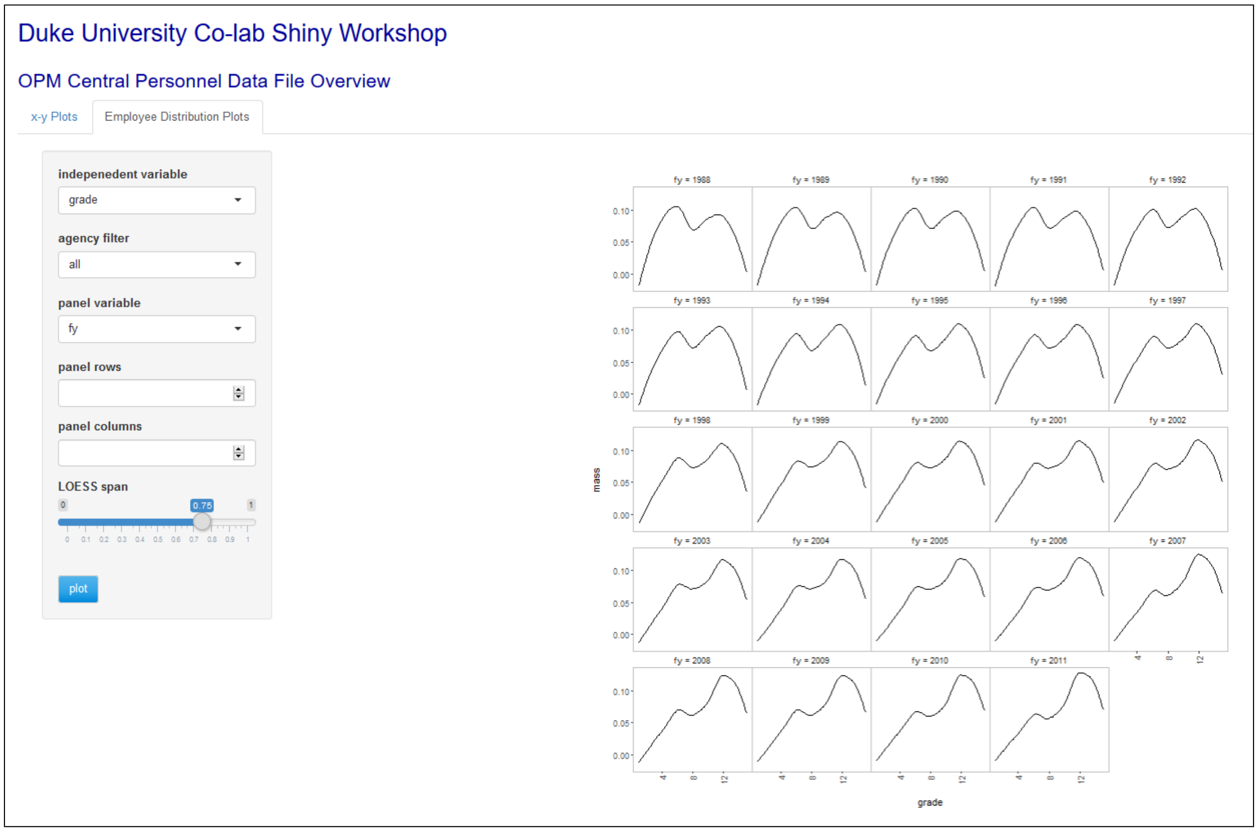
point alpha: 0.1 1

plot

Figure 6: OPM CPDF graph of mean employee age vs. fiscal year, paneled by grade, differentiated by occupational category

8.4 Motivation Graph 4, Increase in Proportion Employees by Grade and Fiscal Year

On the Employee Distribution Plots tab of the app, select `grade` for independent variable and `fy` for panel variable then click the `plot` button. As seen in figure 7, the proportion of employees assigned to higher grades increases with fiscal year.



Duke University Co-lab Shiny Workshop

OPM Central Personnel Data File Overview

independent variable: grade

agency filter: all

panel variable: fy

panel rows:

panel columns:

LOESS span: 0.75

Figure 7: OPM CPDF graph of mean employee age vs. fiscal year

Figures 4, 5, 6, and 7 demonstrate simple, yet effective, exploration of the CPDF data. With little effort, known historical human capital patterns were verified. Experimentation with the on-screen Shiny controls may give the analyst new insight into unknown trends or covariate relationships within an area being studied. The opportunities presented through dynamic data exploration using Shiny and `ggplot` should be apparent from these examples.

9 Development of a Shiny App With Data Tables and Plots

To analyze CPDF observations, we might aggregate a dependent variable by joint levels of a combination of independent variables and produce a summary table with one row per joint level of the independent variables. If done within a Shiny app, using the `DT` (data tables) package, clicking a table row can signal an event such that the app, with identifying row information, can conduct additional, detailed analysis on the subset of observations associated with the row. For example, a table of median pay by agency and occupation can be presented 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.

9.1 Analysis Script in R (V1)

Before developing a Shiny app for this analysis, we will examine a basic R script that aggregates observations for fixed independent and dependent variables and plots covariate relationships for a fixed set of additional variables. Once an R script is designed, implemented, and tested, converting it a Shiny app requires little modification to the R instructions. The challenge in developing an effective Shiny app is in superimposing the Shiny structures to transform static R variables into dynamic, reactive variables that reflect a variety of user inputs and analysis requirements. For this script, and subsequent Shiny apps, we will use a random sample of observations from the Buzzfeed data, as opposed to aggregated values.[1] Table 1 lists the variables included in the data.[2]

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.

Restrictions on observations used in our analysis 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)
- Limited to top five agencies (left two positions) by observation frequency

[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

Script location: App/V1/CPDF-Tables-1.r (App was downloaded in section 7)

Features and considerations include (line numbers are approximate):

- (lines 43-60) One of two sample CPDF files is read, either a random sample of a quarter of the entire data set or a random sample of 100,000 records (the smaller 100,000 record sample is useful for script development)
- (lines 64-89) A common `ggplot` theme is defined once and specified on all `ggplot` calls (this avoids redundant specification of themes)
- (lines 95-99) Dependent and independent aggregation variables are specified
- (lines 101-121) A table (data frame labeled `aggdat`) of aggregated mean and quartiles is constructed using specified dependent and independent variables
- (lines 124-163) An alternative aggregation method truncates `agency` and `occupation` to two positions (to achieve a high level of aggregation)
- (line 171) One row is selected from `aggdat`
- (line 174) An index vector is constructed that subsets the disaggregated data (all observations) corresponding to the selected `aggdat` row
- (line 177) An independent (x-axis) variable for the graph is specified (in `gindepVar`) that is different from any used in the aggregation step
- (lines 182-185) A data frame (`gdat`) is prepared to be used as the data source of `ggplot()`
- (lines 183, 185) `gindepVar` is coerced to a factor to avoid the problem of `ggplot()` producing a single element x-axis when x (`gindepVar`) is continuous
- (line 185) Occupational category, if specified as a graphical independent variable, is coerced to a factor with levels specified in the standard order P, A, T, C, O, for proper alphabetic appearance
- (lines 188-195) The plot (`g`) is constructed in a step-wise manner, as a template for applying further conditional geoms and appearance features
- (line 197) The plot is rendered
- (lines 199-276) 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
  - (line 250) Jitter (`geom_jitter()`) is added in the x-dimension to diminish the effect of point overlay
  - (line 256) 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 257) Actual point category colors are generated using a `colorRampPalette` from blue to red
  - (line 262) Default display of outlier points is suppressed in `geom_boxplot()`
  - (line 263) Error bars (`stat_boxplot()`) are included to indicate the inter-quartile range of each independent variable level

    – (lines 266-268) A custom function is defined to label facet panels
    – (line 272) The previously prepared theme (`ggTheme`) is used to control overall plot appearance

- (line 279) 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 256 to apply the color aesthetic for `diffVar`)

Figure 8 is an example plot from this analysis, showing the distribution of grade by joint combinations of education and fiscal year, for employees in agency Health and Human Services with two-position occupation code 02 (agency and occupation were selected from the initial aggregation data frame - there is no data table yet from which to review and select rows - Shiny is needed for that). It reveals an expected general increase in employee grade vs. education and a general increase in grade throughout the study period, regardless of education (although this may confounded with the visible reduction in employees in clerical and other occupational categories).

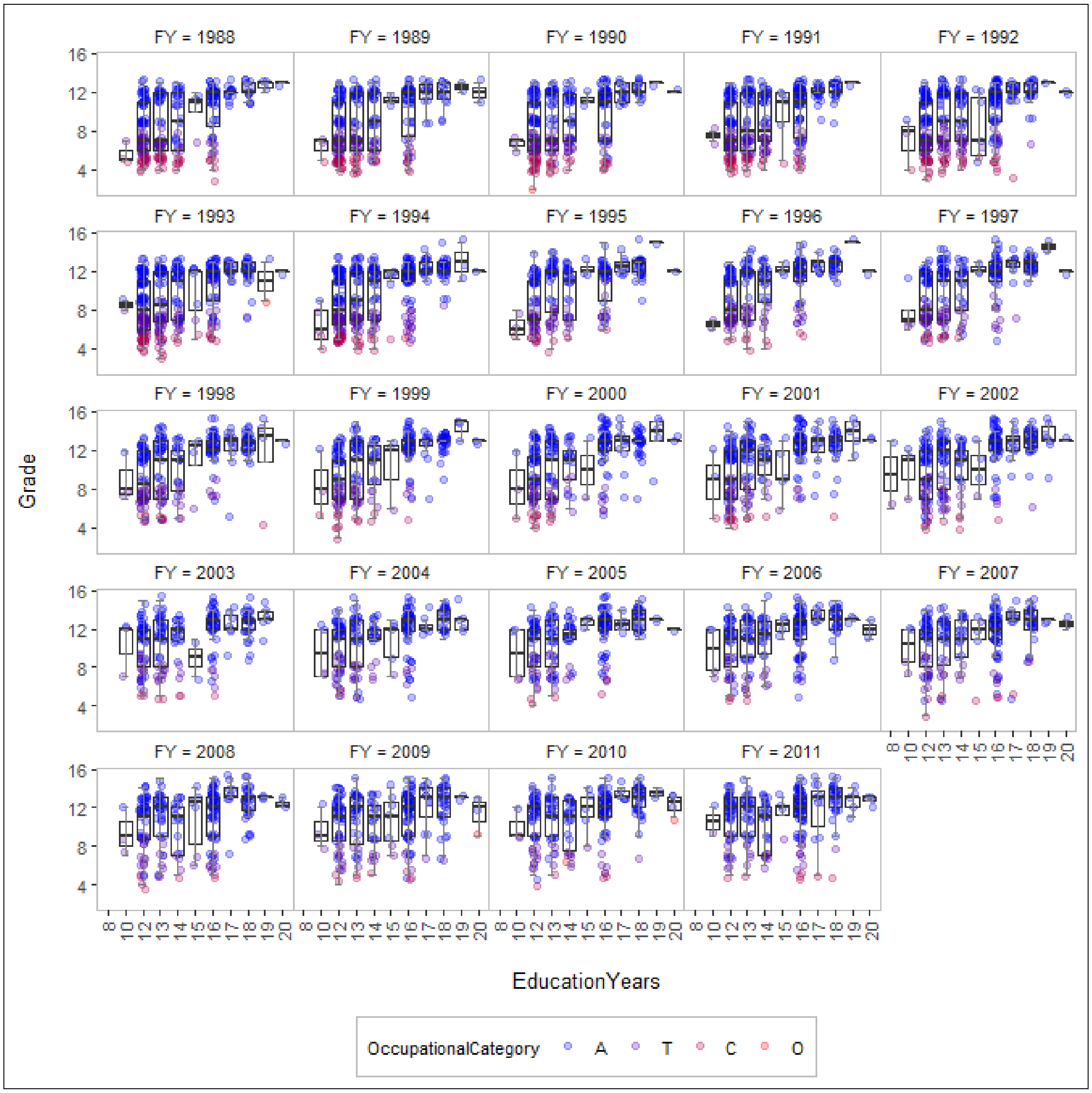


Figure 8: Change in employee grade, given education, along with change in distribution of occupational category, Health and Human Services, occupation code 02, FYs 1988-2011

9.2 Shiny App with Basic Table and Plot Features (V2)

The analysis of section 9.1, 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 user selected 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 of this app include:

- Use of a tab panel with individual tabs for aggregation and plot controls
- Use of cascading style sheet for HTML formatting

- Immediate reactive controls for preparing the aggregation table (which is efficient)
- An action button for plot rendering (as opposed to immediate reactivity, since plotting time may be "long")
- Multiple selection of aggregation variables
- Functions for aggregating data and preparing plots
- Table row selection for subsetting source data to be plotted

Script location: App/V2/CPDF-Tables-2.r (App was downloaded in section 7)

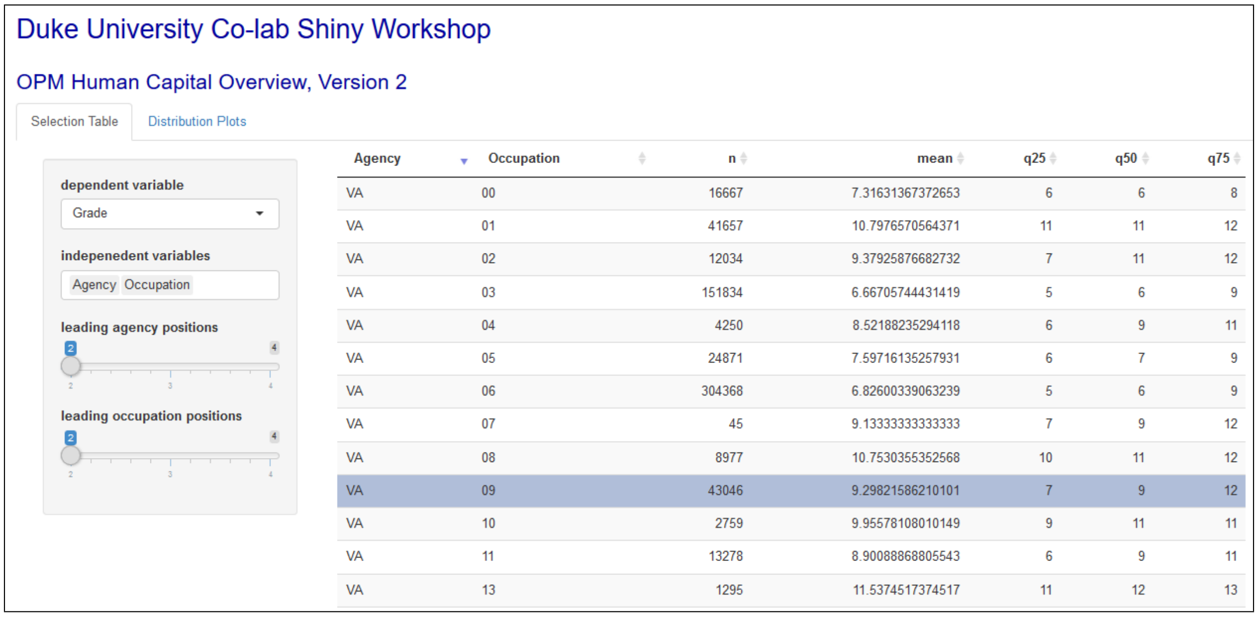
Features and considerations include (line numbers are approximate):

- User interface (file `ui.r`)
  - (lines 59-72) A subset of observations is assembled corresponding to approximately one fourth of the employees represented in the data
  - (lines 74-78) Record cleaning is accomplished
  - (lines 82, 151) An agency selection list is constructed to be used in an agency selection input field (agency selection is disabled in this version)
  - (line 116) A `tabsetPanel` is configured for two tabs: one for the aggregation table, one for plotting
  - (lines 118-140) The aggregation and selection tab (`t1`) is defined
  - (lines 121, 137) Two columns are defined: one of width 2 and one of width 10. There are twelve columns available in the UI.
  - (line 123) A `sidebarPanel` is defined to contain the user input controls. It is defined within a column of width 12, the entire width of the column in which it is defined.
  - (line 127) Multiple independent variables are simultaneously selectable (`t1IndepVar`, `multiple=T`). 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 138) The data table object (`t1Table`) is defined (selecting a row will activate the plot tab)
  - (lines 42-196 The plot configuration and review tab (`t2`) is defined
  - (line 171) An action button (`t2ActionPlot`) is defined for initiating plot construction
  - (line 182) The plot object (`t2Plot`) is defined (this is where plots will be displayed)
  - (line 190) A text output line (`t2Msg`) is defined for messaging
- Server (file `server.r`)
  - (lines 28-52) A `ggplot` theme is defined to control plot appearance
  - (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).
  - (lines 54-133) A function is defined to aggregate observations
  - (line 60) 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 65-77) 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 79) Subset index validation is accomplished
  - (lines 83-94) 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 96) `aggdat` validation is accomplished
- (lines 100-110) 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 135-232) 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 142-146) 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 149) The observation subset indices corresponding to the selected row are retrieved
  - (lines 151-174) 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 177-213) A composite box plot with overlayed points (for individual observation sin the data subset) is composed
  - (lines 181-182) 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 183-192) 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 195) The box plot is generated with outliers suppressed (perhaps this should be done only when points are requested)
  - (line 196) error bars are included (although specified with fixed color and width, these can be made adjustable with additional on-screen controls)
  - (lines 198-208) Panels are produced, one for each level of a specified variable. A custom labeler function is specified.
  - (lines 210-213) Y-axis labels are formatted (thousands) and a previously prepared theme is applied
  - (line 216) The plot is rendered (displayed in `t2Plot`)
- (lines 234-246) An action event is defined to respond to changes in any of the tab 1 input variables. This renders an aggregation table based on current tab 1 input values
- (lines 248-270) An action event is defined for `t1Table` row selection
- (line 257) The selected row ID is saved in a global variable (`t1SelectedRow`) so that other functions may reference the currently selected row

- (line 261) 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 267) The plot tab (`t2`) is made active, so that the user can view and configure the resulting plot
- (lines 272-289) An action event is defined for the `t2ActionPlot` button. This renders a plot using the previously specified table row (`t1SelectedRow`) and current tab 2 input values
- (lines 245, 269, 288) `ignoreInit=T` instructs that an observe is not to be triggered during script startup, when reactive values change from NULL to their default values

Figure 9 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 10 is an example screen shot of the plotting tab, with a plot composed from the observation subset corresponding to the row selected in figure 9, using education as independent variable, fiscal year as paneling variable, and display of points with color differentiating occupational category.


Duke University Co-lab Shiny Workshop

OPM Human Capital Overview, Version 2

dependent variable: Grade

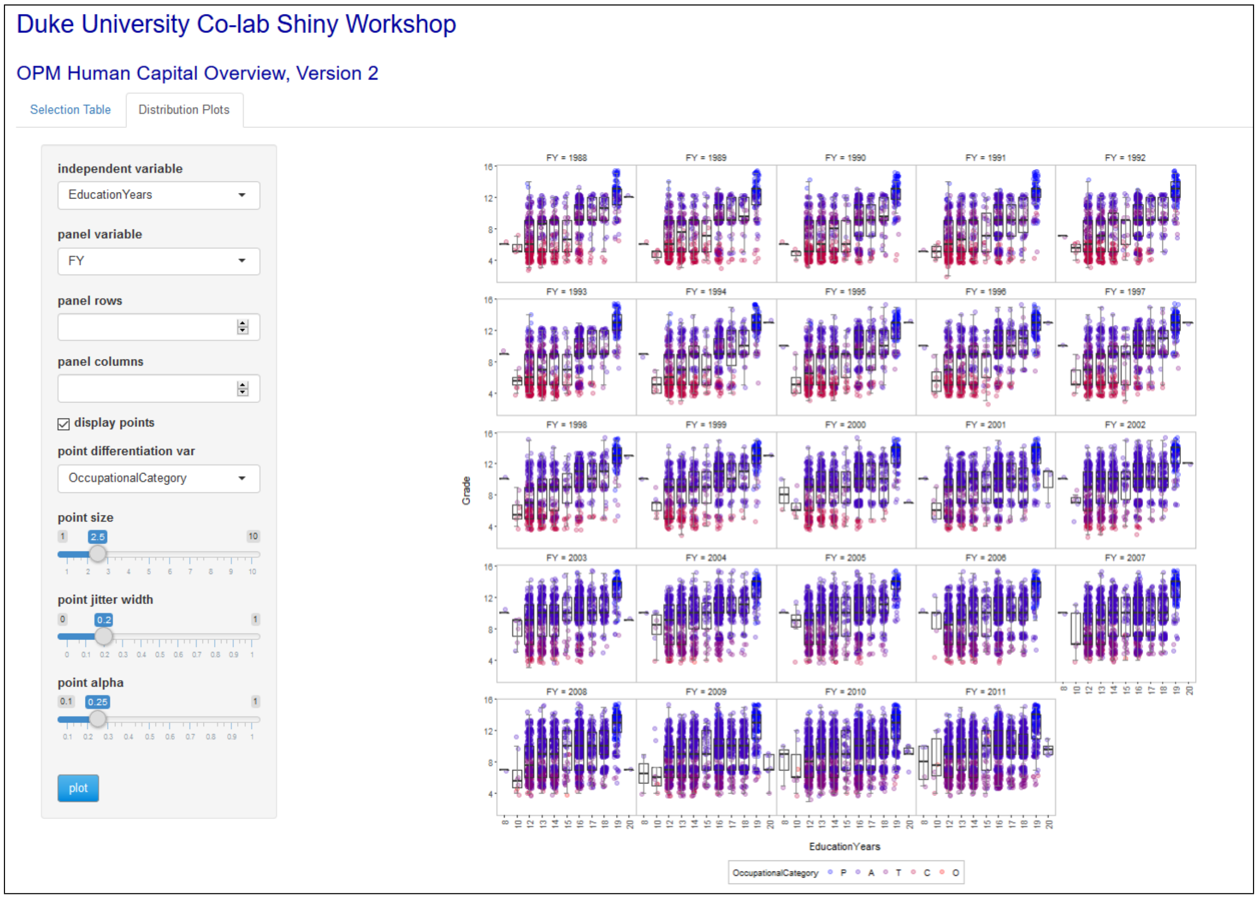
indepenedent variables: Agency Occupation

leading agency positions: 2

leading occupation positions: 2

Agency	Occupation	n	mean	q25	q50	q75
VA	00	16667	7.31631367372653	6	6	8
VA	01	41657	10.7976570564371	11	11	12
VA	02	12034	9.37925876682732	7	11	12
VA	03	151834	6.66705744431419	5	6	9
VA	04	4250	8.52188235294118	6	9	11
VA	05	24871	7.59716135257931	6	7	9
VA	06	304368	6.82600339063239	5	6	9
VA	07	45	9.13333333333333	7	9	12
VA	08	8977	10.7530355352568	10	11	12
VA	09	43046	9.29821586210101	7	9	12
VA	10	2759	9.95578108010149	9	11	11
VA	11	13278	8.90088868805543	6	9	11
VA	13	1295	11.5374517374517	11	12	13

Figure 9: Screen shot of V2 CPDF Shiny app, aggregated summary table tab


Duke University Co-lab Shiny Workshop

OPM Human Capital Overview, Version 2

independent variable: EducationYears

panel variable: FY

panel rows:

panel columns:

☑ display points

point differentiation var: OccupationalCategory

point size: 2.5

point jitter width: 0.2

point alpha: 0.25

Figure 10: Screen shot of V2 CPDF Shiny app, plot configuration tab

Observations from review of the aggregation table:

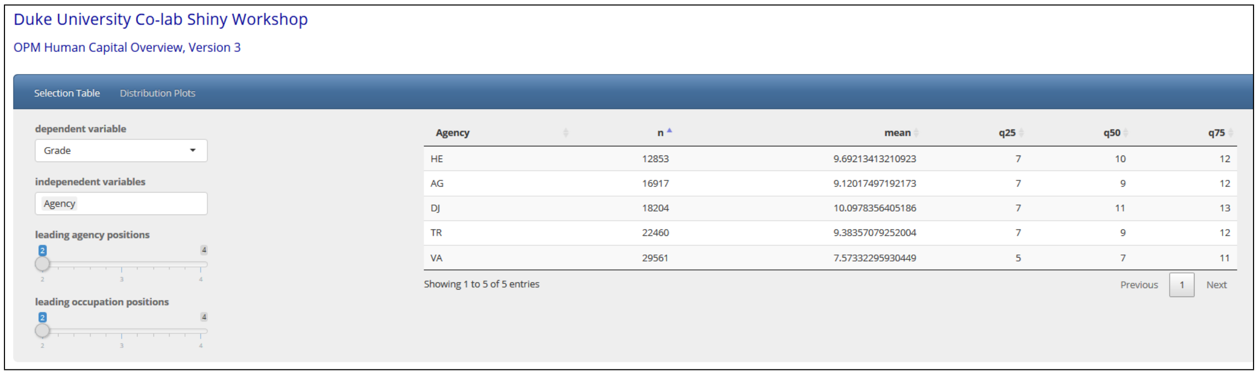
- Aggregation of grade by fiscal year reveals a pattern of increased grade by year, with median grade exhibiting greater increase than mean grade (does this indicate a widening of grade distribution with a tail forming for upper grades?)
- Aggregation of age by fiscal year reveals a pattern of increased age by year, indicating an older (more experienced?) workforce at the end of the study period
- Aggregation of education by fiscal year 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.

9.3 Shiny App with Updated Appearance Using Navigation Bars and Themes (V3)

Script location: App/V3/CPDF-Tables-3.r (App was downloaded in section 7)

This version of the app extends the V2 app by replacing the basic `tabsetPanel` with a `navbarPage` and use of themes and modification of cascading style sheet (CSS) elements. V2 and V3 app functions are very similar. The primary difference is in their appearance. Figures 11 and 12 are screen-shots of this app. Discussion will follow.


Duke University Co-lab Shiny Workshop
OPM Human Capital Overview, Version 3
Selection Table
Distribution Plots

dependent variable: Grade

indepenedent variables: Agency

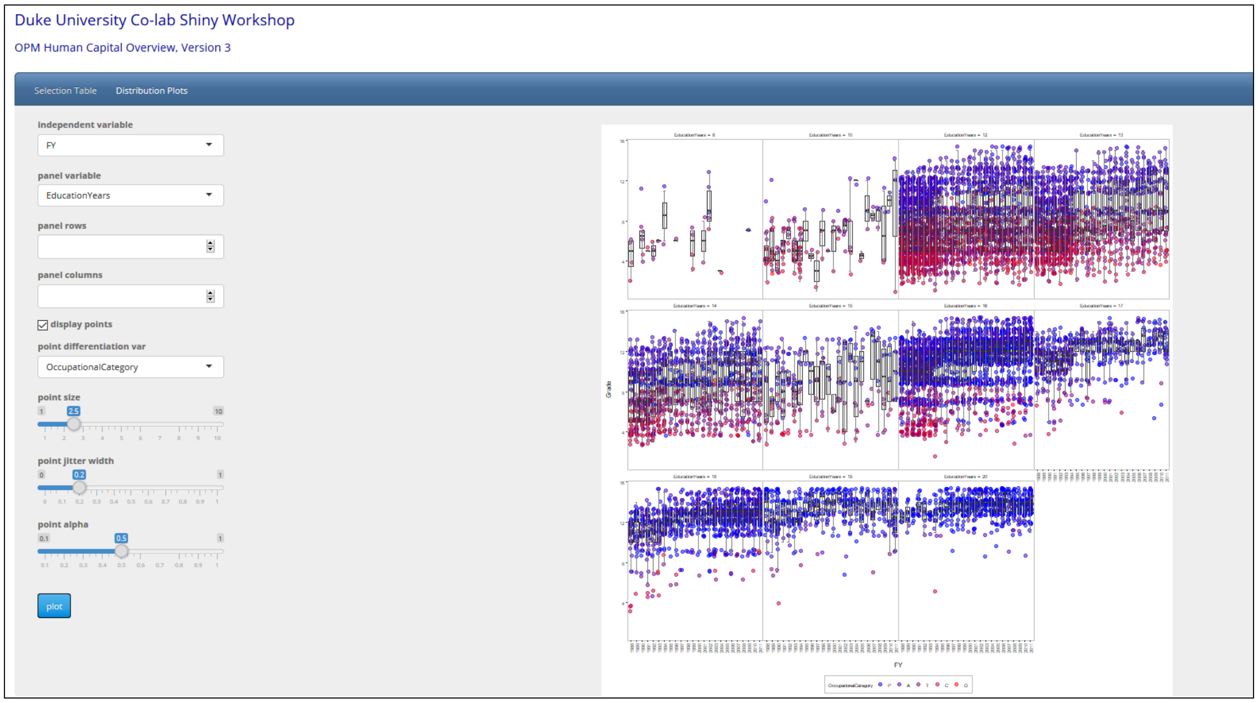
leading agency positions: 2

leading occupation positions: 2

Agency	n	mean	q25	q50	q75
HE	12853	9.69213413210923	7	10	12
AG	16917	9.12017497192173	7	9	12
DJ	18204	10.0978356405186	7	11	13
TR	22460	9.38357079252004	7	9	12
VA	29561	7.57332295930449	5	7	11

Showing 1 to 5 of 5 entries

Figure 11: Screen shot of V2 CPDF Shiny app, aggregated summary table tab


Duke University Co-lab Shiny Workshop
OPM Human Capital Overview, Version 3
Selection Table
Distribution Plots
plot
Grade
FY
OccupationalCategory

independent variable: FY

panel variable: EducationYears

panel rows:

panel columns:

☑ display points

point differentiation var: OccupationalCategory

point size: 2.5

point jitter width: 0.2

point alpha: 0.5

Figure 12: Screen shot of V2 CPDF Shiny app, plot configuration tab