The Cosumnes River College HawkTrack:
Creating a student success dashboard with R and R Shiny

Paul Meinz, Ph.D.
Cosumnes River College
Outline

• Describe the purpose of the HawkTrack – a dashboard programmed in R - and provide a brief tour of its functionality

• Describe some of the key steps I took when making the HawkTrack:
  – Give a brief overview of R Shiny
  – Programming a dashboard (basic logic, useful packages, etc.)
  – Improving your dashboard (with other packages in R)
  – Deploying dashboards on a local server
  – Workflow and iterative development

• Future dashboard directions
The HawkTrack: Purpose

- Cosumnes River College began implementing guided pathways in Fall 2017.

- We needed a tool that could track:
  - Enrollment milestones (e.g., full-time, math/English, etc.)
  - Achievement milestones (e.g., math/English completion)
  - Completion
  - Disaggregate outcomes

- The data needed to be live and up-to-date
The HawkTrack: Tour

• The HawkTrack follows cohorts of students at CRC and tracks demographics, enrollment, and achievements for three years.

• https://research.crc.losrios.edu/HawkTrack/
R Shiny: Overview

- R is a free and open source programming language primarily designed for statistical programming (https://www.r-project.org/)

- Shiny is a library for R that provides a web application development framework.
  - Lets you leverage html, CSS, javascript all in R code.

- If you intend on working with R and Shiny, you should also download Rstudio (https://www.rstudio.com/)
R Shiny: Overview

• Downsides of Shiny
  – Steep learning curve
  – R is slower than other languages when dealing with large amounts of data (Python and Javascript are faster).

• Benefits of Shiny
  – Powerful dashboard interface and a powerful statistical programming language
  – R and Shiny are both free and open source
  – Can be deployed using local resources (e.g., no additional cost)
Programming an R Shiny Application
Application Development: Overview

• All Shiny applications are designed around the concept of *reactivity*.

• *Reactivity* can be observed when the *outputs* of an application change based on user *inputs*:
  – When the HawkTrack filters change depending on the selected cohort
  – When a plot changes depending on the outcome/disaggregation.

• Inputs and outputs for a Shiny app can be specified in two files named: ui.R and server.R
  – There is also a one file solution
Application Development: Inputs

- Inputs are defined in ui.R file. The ui file also:
  - Specifies the format and styling of your dashboard
  - Specifies where outputs are displayed

Radio buttons
- Choice 1
- Choice 2
- Choice 3

Checkbox group
- Choice 1
- Choice 2
- Choice 3

Action button
- Action

Slider

Text input
- Enter text...
Application Development: Inputs

ui.R

```r
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(
  # Header
titlePanel("RP Group Demonstration"),
  # Sidebar with a slider input for number of bins
  sliderInput("bins",
    "Number of bins: ",
    min = 1,
    max = 50,
    value = 30),
  # Show a plot of the generated distribution
  plotOutput("distPlot")
))
```
Application Development: Outputs

• Outputs are built from inputs in the server.R file

• Output types:
  – Text
  – Charts
  – Images
  – HTML
  – Tables
  – Other input elements
Application Development: Outputs

server.R

```
library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {
  # Render a chart
  output$distPlot <- renderPlot(
    # Do some calculations
    x <- faithful[, 2]
    bins <- seq(min(x), max(x),
                length.out = input$bins + 1)
    # Make a chart
    hist(x, breaks = bins,
         col = 'darkgray', border = 'white')
  )
})
```
Application Development: Reactivity

**ui.R**

```r
shinyUI(
  header(),
  slider input(),
  output here()
)
```

**server.R**

```r
shinyServer(
  render my output(
    do some calculations with the input data
    create a histogram with the calculations
  )
)
```
Application Development: Reactivity

Slider Input

Hey I changed!

Retrieve the new value

Recalculate + Re-Render

Render my Output()
Application Development: Reactivity

- A full tutorial for the Shiny package can be found at: https://shiny.rstudio.com/tutorial/
Improving your application
Application Development: Styling

• There are many libraries and standard formats that can be leveraged with functions in the Shiny package:
  – Bootstrap html pages
  – Tab pages
  – Control and main panel pages

• You can also incorporate CSS elements for styling.
Application Dev.: Useful Packages

- dplyr
  - Vastly speeds up and simplifies data manipulation and data disaggregation.

```r
server.R

shinyServer(
  render my output(
    do some calculations with the dplyr data manipulation input data
    create a histogram with the calculations
  )
)
```

Application Dev.: Useful Packages

- **dplyr**
  - Vastly speeds up and simplifies data manipulation and data disaggregation.

- **rCharts**
  - Allows you to code interactive plots from the Javascript library d3.js
  - [https://d3js.org/](https://d3js.org/)

---

**server.R**

```r
shinyServer(
  render my output(
    dplyr data manipulation
    create a histogram with the calculations
  )
)
```

---

Application Dev.: Useful Packages

- Additional useful packages
  - shinyjs
    - [https://CRAN.R-project.org/package=shinyjs](https://CRAN.R-project.org/package=shinyjs)
  - shinyBS
      [https://CRAN.R-project.org/package=shinyBS](https://CRAN.R-project.org/package=shinyBS)
Deploying Your Application to the Internet
Deploying your application: Basics

• Shiny applications can be deployed on a local server with a Linux operating system.

• Requires the installation of Shiny Server on the Linux operating system.
  – Step by step instructions for installation can be found at https://www.rstudio.com/products/shiny/download-server/
Deploying your application: Basics

• Once the server is installed a new directory will be created on the Linux server:
  – /srv/shiny-server/

• Create a folder with your application name:
  – /srv/shiny-server/HawkTrack/

• Drop your ui.R/server.R files into the folder with any data/files they reference.
  – Ask IT to open port 3838 on your server
Deploying your application: Basics

http://research.crc.losrios.edu:3838/HawkTrack/

- server URL
- port
- folder name

Requests → Shiny Server
Deploying your application: Additional Security

Denial of Service Attack (DOS)

Man in the Middle Attack (MITM)
Deploying your application: Additional Security

- Solution a reverse proxy through Apache Web Server
  - Apache Web Server can add SSL certification to protect from man in the middle attacks.
  - Apache Web Server can identify and block DOS attacks.
Deploying your application: Basics

• Modify/paste this code into Apache configuration file to create a reverse proxy:

• Work with your local IT to:
  – Get an SSL certificate and set-up Apache for SSL
  – Add directives to the Apache configuration to prevent DOS

• https://research.crc.losrios.edu/HawkTrack/
Work Flow

• Development time is substantially reduced once your server is set-up.
  – First dashboard in R Shiny (the HawkDash) took 4 weeks.
  – The HawkTrack took about 1-2 weeks.

• When developing large applications it is helpful to use version control software
  – https://github.com
  – https://github.com/paulmeinz/hawkTrack

• Workflow using github:
  – Develop the application on your local computer.
  – Push changes up to your github account.
  – Pull final versions down onto the server for deployment.
Work Flow: Interacting with your Linux Server

• PuTTY to interface with the Linux server
  – https://www.putty.org/

• WinSCP to transfer data files or other files (e.g. images)
  – https://winscp.net/eng/download.php
Future Dashboard Directions

• Automate all the data required for our local dashboards

• Add statistical projections to HawkTrack

• Create an SLO evaluation dashboard using Canvas data
Thank you!

Questions?