Quadrus
Improving Enterprises

INTERACTIVE

A guide to interactivity in R

Patrick Marchand
Senior Technical Consultant

Agenda

- Interactivity
- R fundamentals
  - Particularly those that will relate to code shown later on
- Interactivity in R
  - The R Language
  - Development Environment
- Packages
  - plotly, shiny, shinydashboard, and others
  - Walk through demonstrations

Definition

in·ter·ac·tive

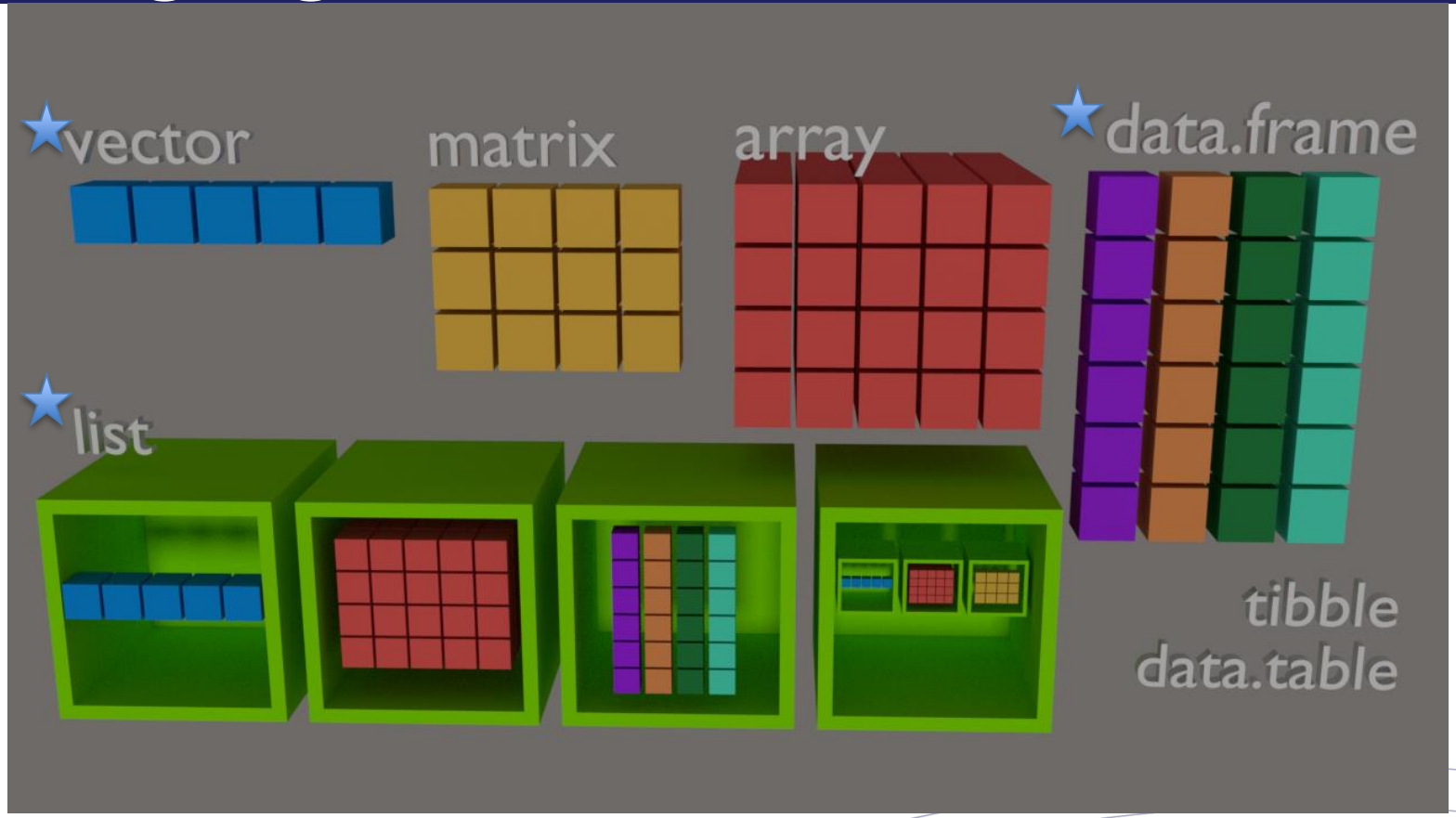
- Allowing a two-way flow of information between a computer and a computer-user; responding to a user's input
- Responses should be quick, without too much hassle or limitations

Interactive Content

- *“By its very nature, interactive content engages participants in an activity: answering questions, making choices, exploring scenarios. It’s a great way to capture attention right from the start. Individuals have to think and respond; they can’t just snooze through it.”* - Scott Brinker

https://www.copyblogger.com/interactive-content/

R Language - Data Structures



R Language - Syntax

- R is a functional language
- R is a Case-Sensitive language
  - `selectInput()` not `selectinput()`
- Assignment can be done using “<-” or “=” operators
  - E.g. `var <- rnorm(100)` or `var = rnorm(100)`
- But recommend using “=” for passing parameter values, “<-” to assign results to a variable
  - E.g. `var <- rnorm(100, mean = 5, sd = 2)`

R Language - Syntax

- Can spread long lines of code over many lines, but you should include a “continuation” operator at the *end* of a line:

```
mtcars %>% filter(mpg > 20)
```

as:
```
mtcars %>%
    filter(mpg > 20)
```

not:
```
mtcars
    %>% filter(mpg > 20)
```

- Parentheses () are normally for indicating a function call
- Curly braces {} are for multi-statement blocks of code
- Get ready for ({…})

R Language - Syntax

- User-defined functions:
  `myfn <- function(<parameters>) {<function body>}`
- Need to differentiate the result of a functions being passed a parameters and a block of code with functions passed as a parameter:

  A function that is called with parameter values from 3 functions:

```
callme(
    x(), # commas to separate parameter values
    y(),
    z()
)
```

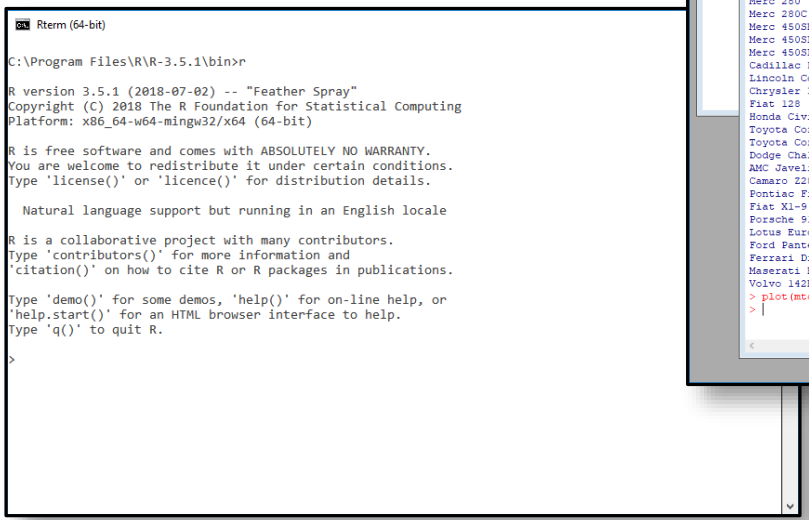
  vs a function that accepts the result of 3 statements:

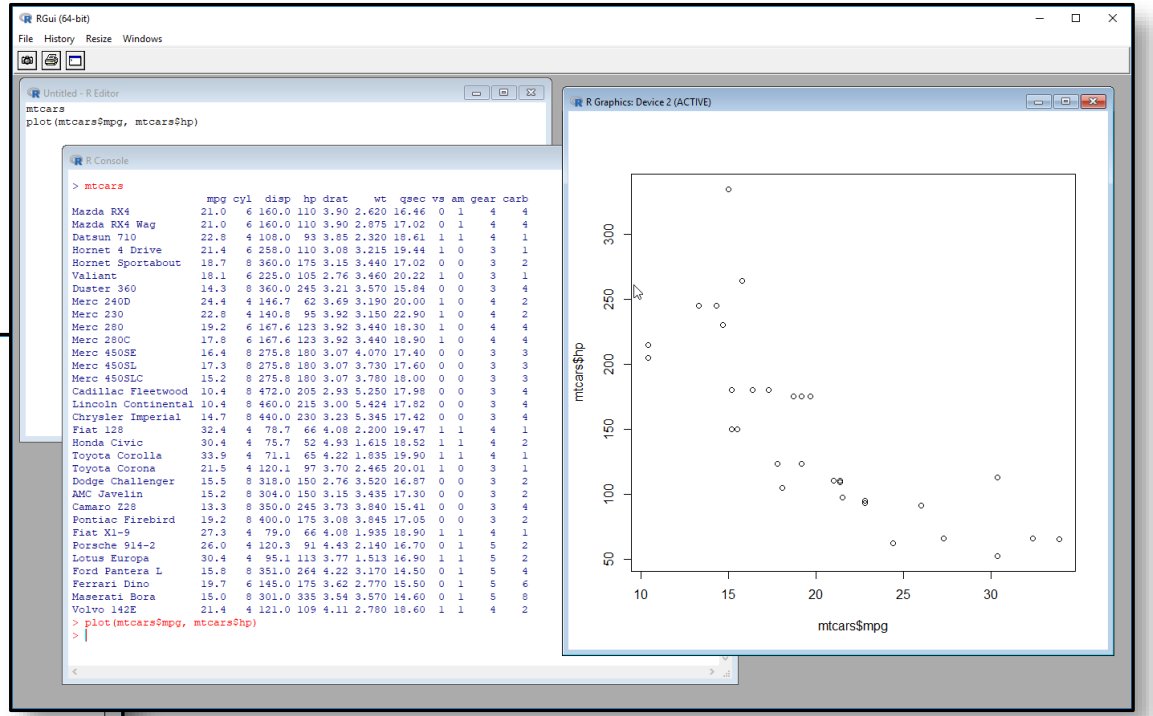
```
funkyfn({  # the parameter is an expression
    dothis() # no trailing commas after each statement!
    dothat()
    dosomethingelse()
})
```

Interactivity in R

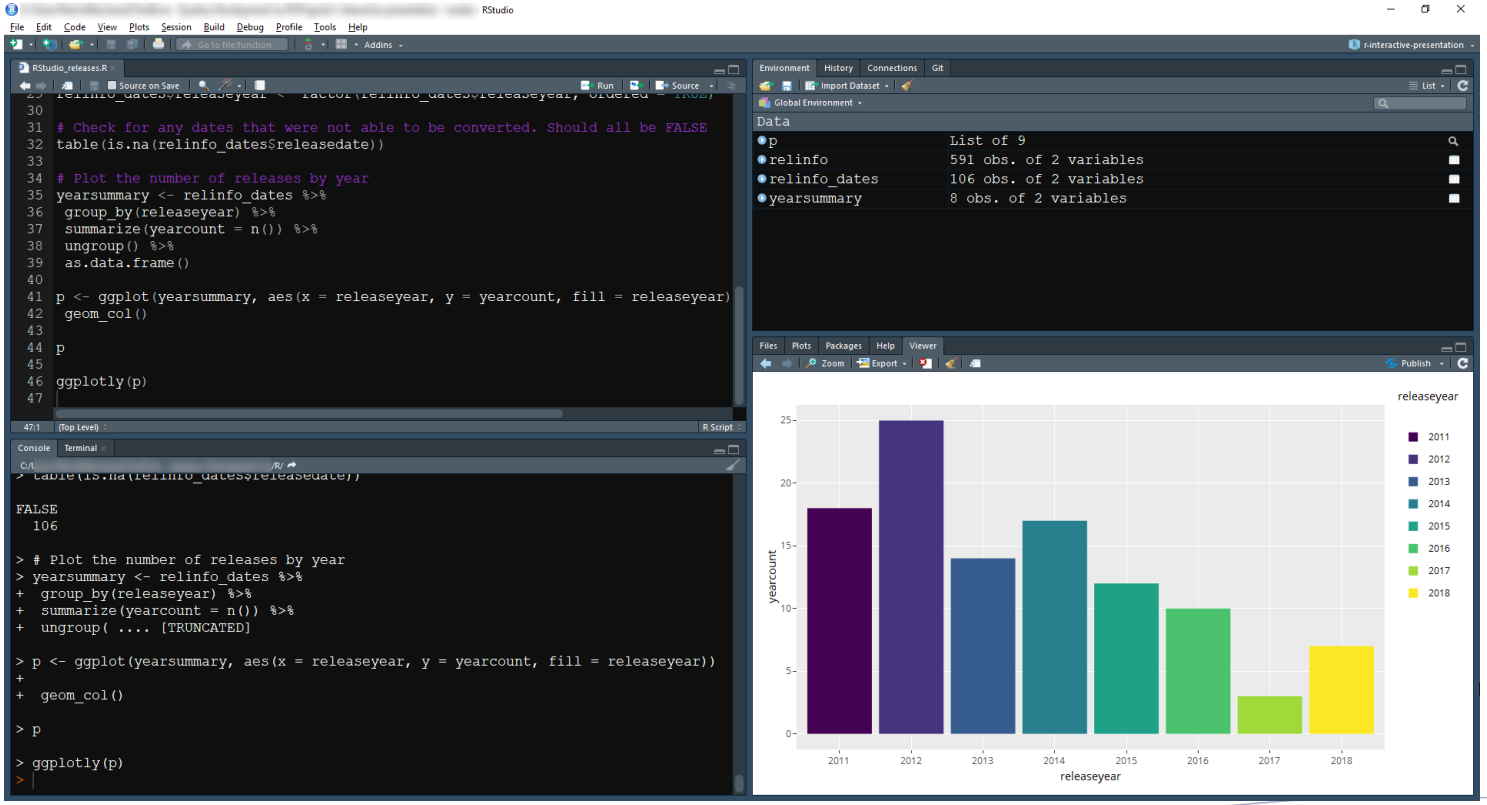
- R is an interactive language in the sense that:
  - No compilation is required
  - You can try something, get a result, try something, and adapt code as you go along
- Incremental feedback is very useful when in the exploratory stage of data analysis

In the Beginning… Rterm / RGui





RStudio



Package: plotly

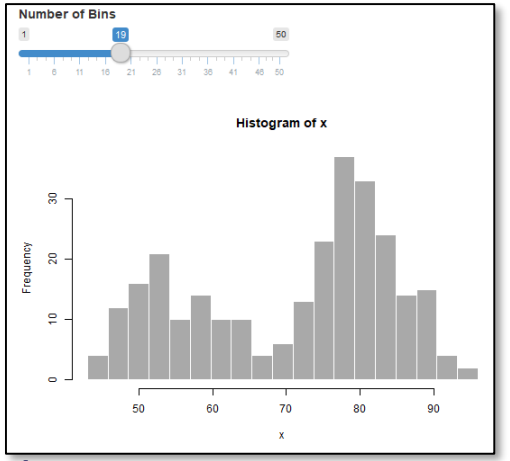
- Open source graphing library
- Produces interactive graphs/charts with built-in controls
- Has its own syntax for generating plots
- Can also make ggplot output interactive*
- Available for other languages as well

Package: plotly

PLOTLY DEMO

Package: shiny

- Open source package for R
- Framework for building interactive web applications using R code
- Reactive programming framework
  - https://www.rstudio.com/resources/videos/effective-reactive-programming/
- Brought to you by the folks at RStudio (https://shiny.rstudio.com)



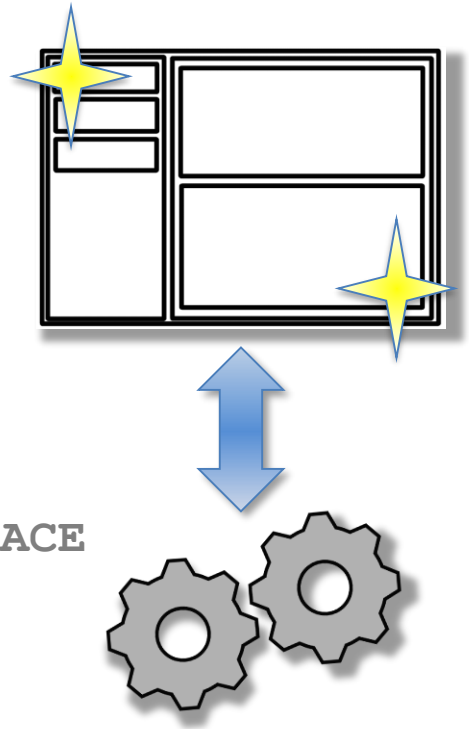
Package: shiny – Template

```
library(shiny)

[SETUP R CODE]

ui <- fluidPage(
   [DEFINE YOUR USER INTERFACE HERE]
)
server <- function(input, output, session) {
   [CODE TO REACT TO CHANGES IN THE USER INTERFACE
    AND GENERATE OUTPUT]
}

shinyApp(ui, server)
```



Package: shiny

SHINY DEMO

Package: shiny (UI code)

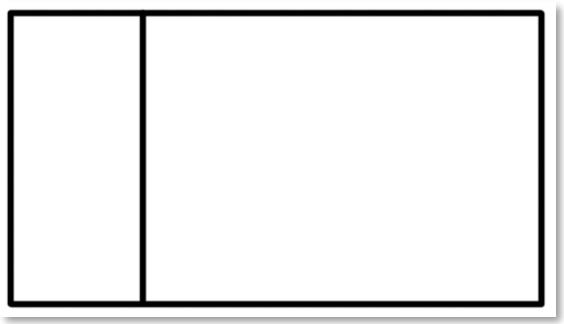
- Functions for defining inputs and outputs:
  - To define input controls:
    ```
    <type>Input(inputId = "<inputname>", ___)
    E.g. sliderInput(inputId = "slider", ___)
    ```
  - To define output areas:
    ```
    <type>Output(outputId = "<outputname>", ___)
    E.g. plotOutput(outputId = "mpgplot")
    ```
- In the server code:
  - Inputs will be referenced as **`input$`**`<inputname>`
  - Outputs will be referenced as **`output$`**`<outputname>`

Package: shiny (server code)

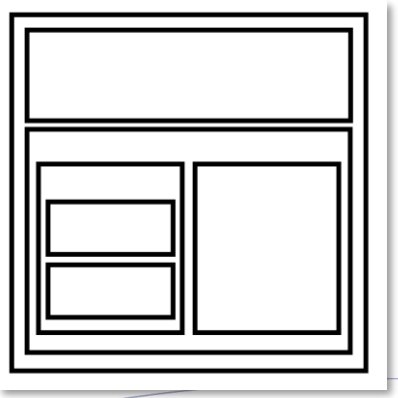
- Functions used in the server code:
  - Generate output for the appropriate type: `render<type>()`
    `E.g. renderPlot()`
  - To update input controls: `update<type>Input()`
    `E.g. updateSelectInput()`
  - To manipulate data based on a change in the UI and have the result stored: `reactive()/EventReactive()`
  - Create a side-effect when something changes: `observe()/observeEvent()`

Package: shiny

- Pre-defined layout:
  - `sideBarLayout(`
    `sideBarPanel(),`
    `mainPanel()`
    `)`



- Defining rows and columns:
  - `fluidRow()`
  - `column()`



shiny apps - Deployment

- For the most up-to-date information for On Premise and Cloud deployments please visit:
  https://www.shinyapps.io/#pricing
  https://www.rstudio.com/pricing/

Package: shinydashboard

- A package to make it easy to develop dashboards using shiny code
- shiny code is used for interactive controls and logic; shinydashboard code is for a dashboard layout
- Facilitates the creation of the standard sidebar layout with menus, tab groups, notifications, etc.

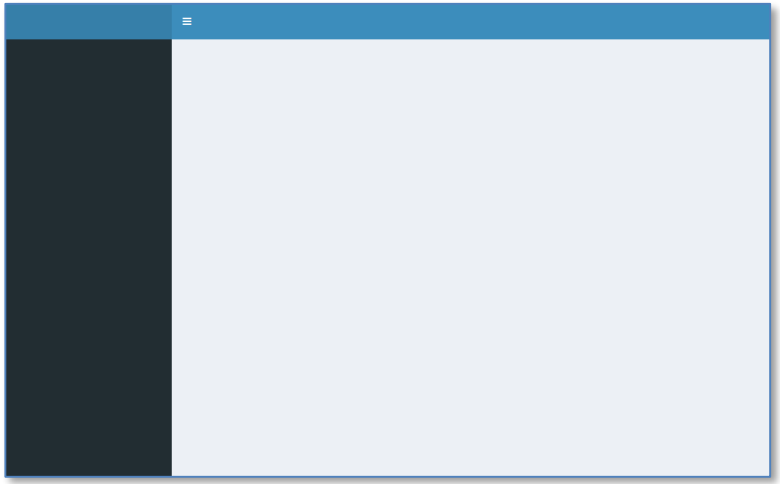
Package: shinydashboard – Template

```
library(shiny)
library(shinydashboard)

ui <- dashboardPage(
          dashboardHeader(),
          dashboardSidebar(),
          dashboardBody()
      )

server <- function(input, output, session)
           {…}

shinyApp(ui, server)
```



Package: shiny

SHINYDASHBOARD DEMO

Challenges

- Code formatting becomes extremely important
- Common gotcha's: missing/unmatched ()'s and {}'s, missing commas to separate functions in the UI code
- Debugging using `print()` or `cat(file=stderr, …)`
  - https://shiny.rstudio.com/articles/debugging.html
- Variables defined as "reactive expressions" are referenced with ()'s
  - e.g. `myvar <- reactive({…})` and elsewhere `myvar()`

Challenges

- Easy to make a basic app but may need JavaScript knowledge for advanced functionality
- Reactivity in shiny can be a little odd at first e.g. when to use `observe()` vs `reactive()`
- Need to save file as “app.R” for some things to work as expected

Other shiny packages

- There are dozens of other “shiny” related packages (not always with the same prefix)
  - shinydashboardPlus – more dashboard features
  - shinyDND – Drag aNd Drop functionality
  - shinyjs – more control of JavaScript
  - shinythemes – change the look of a shiny app
  - dashboardthemes (BETA) – change the look of a dashboard
  - shinyWidgets – more widgets!

Package: flexdashboard

- Coding a dashboard-type layout using R Markdown
- May be a little easier to get started than shinydashboard
- Can make use of plotly and shiny code
- For static or dynamic dashboards
- Results can also be deployed to a Shiny server
- Several examples can be found at: https://rmarkdown.rstudio.com/flexdashboard/examples.html

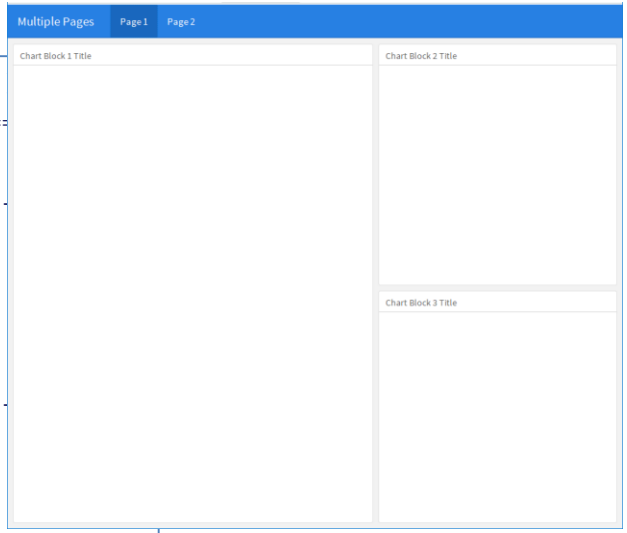
Package: flexdashboard (RMarkdown)

```
---
title: "Multiple Pages"
output: flexdashboard::flex_dashboard
---
Page 1
====================================
Column {data-width=600}
------------------------------------
### Chart Block 1 Title
```{r}
```
Column {data-width=400}

Chart Block 2 Title
```{r}
```
Chart Block 3 Title
```{r}
```
```

```
Page 2 {data-orientation=rows
==============================
Row {data-height=600}
------------------------------
### Chart Block 1 Title
```{r}
```
Row {data-height=400}

Chart Block 2 Title
```{r}
```
Chart Block 3 Title
```{r}
```
```



https://rmarkdown.rstudio.com/flexdashboard/examples.html

Outside the Box

- leaflet package – access to the popular open-source JavaScript libraries for interactive maps
- NodeJS + R + Plotly = Power BI Custom Visual
- Displayr.com – data science platform, drag-and-drop functionality, based on R

Get Interactive

- Shiny Webinars:
  - https://www.rstudio.com/resources/webinars/shiny-developer-conference/
- Dashboard Tutorials:
  - https://www.rstudio.com/resources/videos/building-dashboards-with-shiny-tutorial/
- Awesome R Packages:
  - https://awesome-r.com/#awesome-r-html-widgets
- Online courses (Data Camp, Udemy, Coursera, etc.)

Questions?

patrickm@quadrus.com