

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



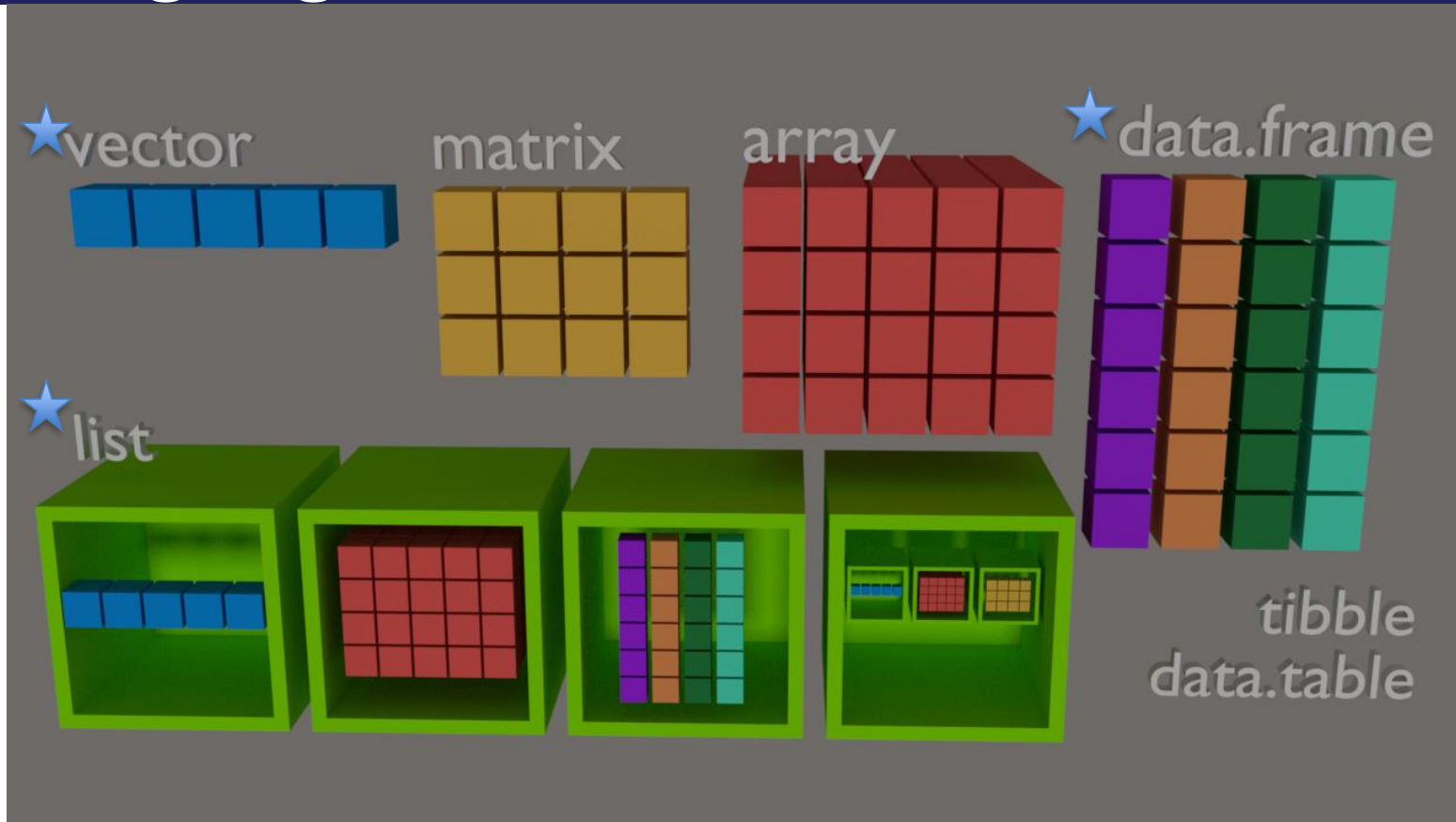
- Allowing a two-way flow of information between a computer and a computer-user; responding to a user's input
- Responses should be quick, without too much hassle or limitations

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

The screenshot shows the RGui (64-bit) interface. The R Editor window displays the following code:

```
mtcars
plot(mtcars$mpg, mtcars$hp)
```

The R Console window shows the output of the commands:

```
> mtcars
      mpg  cyl  disp  hp  drat   wt  qsec vs  am  gear  carb
Mazda RX4      21.0   6 160.0 110  3.90 2.620 16.46  0  1   4   4
Mazda RX4 Wag  21.0   6 160.0 110  3.90 2.875 17.02  0  1   4   4
Datsun 710      22.8   4 108.0  93  3.85 2.320 18.61  1  1   4   1
Hornet 4 Drive  21.4   6 258.0 110  3.08 3.215 19.44  1  0   3   1
Hornet Sportabout 18.7   8 360.0 175  3.15 3.440 17.02  0  0   3   2
Valiant         18.1   8 225.0 105  2.76 3.460 20.22  1  0   3   1
Duster 360     14.3   8 360.0 245  3.21 3.570 15.84  0  0   3   4
Merc 240D      24.4   4 146.7  62  3.69 3.190 20.00  1  0   4   2
Merc 230       22.8   4 140.8  95  3.92 3.150 22.90  1  0   4   2
Merc 280       19.2   6 167.6 123  3.92 3.440 18.30  1  0   4   4
Merc 280C      17.8   6 167.6 123  3.92 3.440 18.90  1  0   4   4
Merc 450SE     16.4   8 275.8 180  3.07 4.070 17.40  0  0   3   3
Merc 450SL     17.3   8 275.8 180  3.07 3.730 17.60  0  0   3   3
Merc 450SLC    15.2   8 275.8 180  3.07 3.780 18.00  0  0   3   3
Cadillac Fleetwood 10.4   8 472.0 205  2.93 5.250 17.98  0  0   3   4
Lincoln Continental 10.4   8 460.0 215  3.00 5.424 17.82  0  0   3   4
Chrysler Imperial 14.7   8 440.0 230  3.23 5.345 17.42  0  0   3   4
Fiat 128       32.4   4  78.7  66  4.08 2.200 19.47  1  1   4   1
Honda Civic    30.4   4  75.7  52  4.93 1.615 18.52  1  1   4   2
Toyota Corolla 33.9   4  71.1  65  4.22 1.835 19.90  1  1   4   1
Toyota Corona  21.5   4 120.1  97  3.70 2.465 20.01  1  0   3   1
Dodge Challenger 15.5   8 318.0 150  2.76 3.520 16.87  0  0   3   2
AMC Javelin    15.2   8 304.0 150  3.15 3.435 17.30  0  0   3   2
Camaro 228     13.3   8 350.0 245  3.73 3.840 15.41  0  0   3   4
Pontiac Firebird 19.2  4 400.0 175  3.08 3.845 17.05  0  0   3   2
Fiat X1-9      27.3   4  79.0  66  4.08 1.935 18.90  1  1   4   1
Porsche 914-2  26.0  4 120.3  91  4.43 2.140 16.70  1  1   5   2
Lotus Europa   30.4   4  95.1 113  3.77 1.513 16.90  1  1   5   2
Ford Pantera L 15.8   8 351.0 264  4.22 3.170 14.50  0  1   5   4
Ferrari Dino   19.7  6 145.0 175  3.62 2.770 15.50  1  1   5   6
Maserati Bora  15.0   8 301.0 335  3.54 3.570 14.60  1  1   5   8
Volvo 142E     21.4  4 121.0 109  4.11 2.780 18.60  1  1   4   2
```

The R Graphics Device 2 (ACTIVE) window shows a scatter plot of mtcars\$hp (Y-axis, 0 to 300) versus mtcars\$mpg (X-axis, 10 to 30). The plot displays a negative correlation between miles per gallon and horsepower.

The screenshot shows the Rterm (64-bit) window. The command prompt displays the following text:

```
C:\Program Files\R\R-3.5.1\bin>r
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

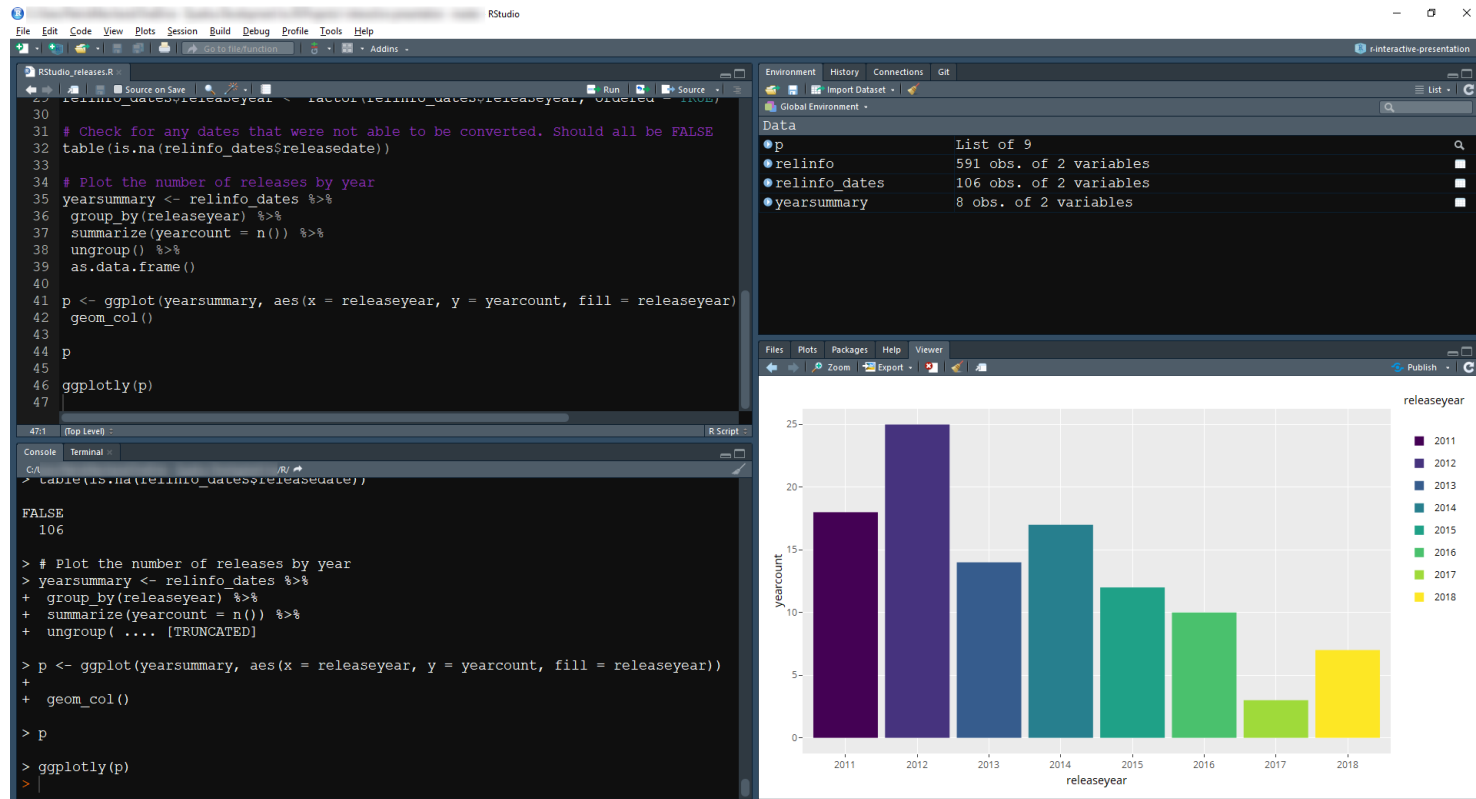
  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

RStudio



Package: plotly

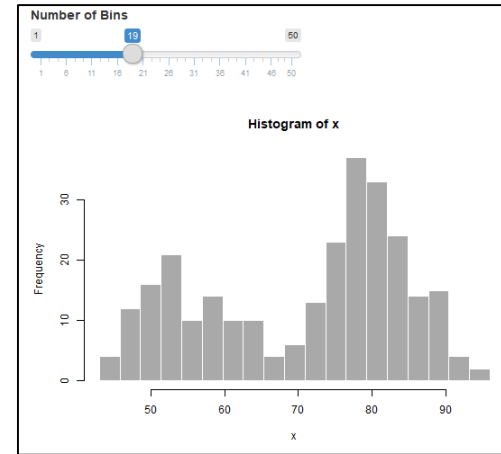
- Open source graphing library
- Produces interactive graphs/charts with built-in controls
- Has its own syntax for generating plots
- Can also make ggplot output interactive*
- Available for other languages as well

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]
```

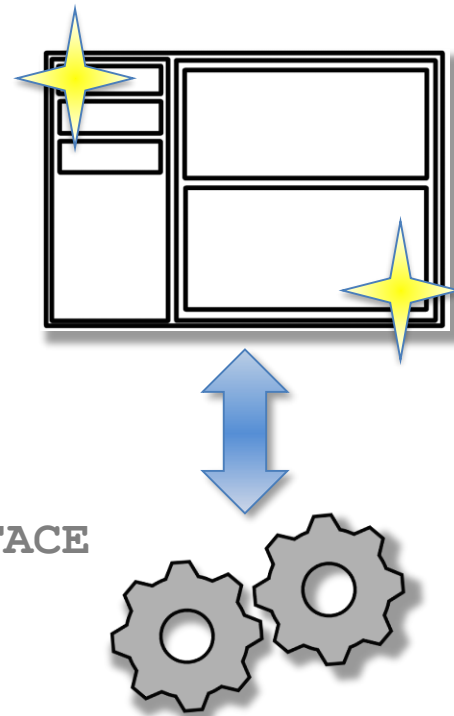
```
)
```

```
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>

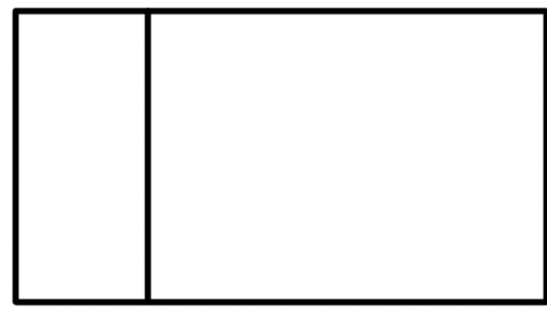
Package: shiny (server code)

- 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

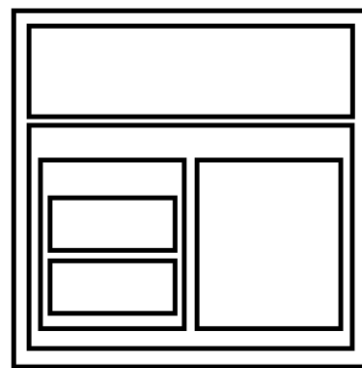
- Pre-defined layout:

- `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.

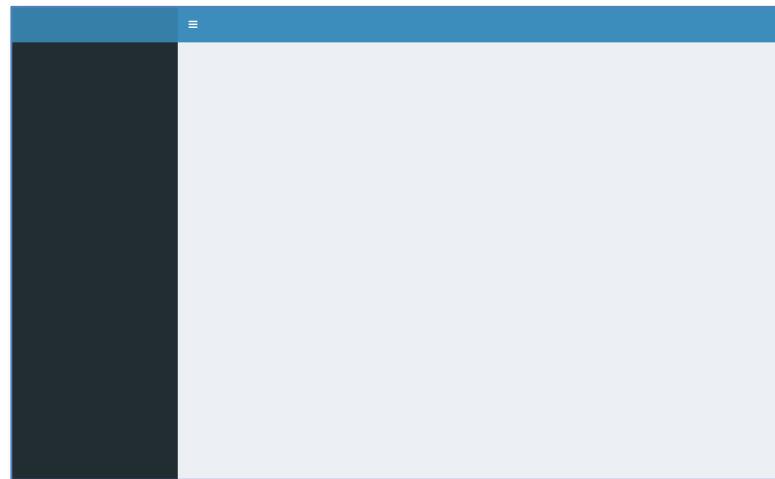
Package: shinydashboard – Template

```
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}

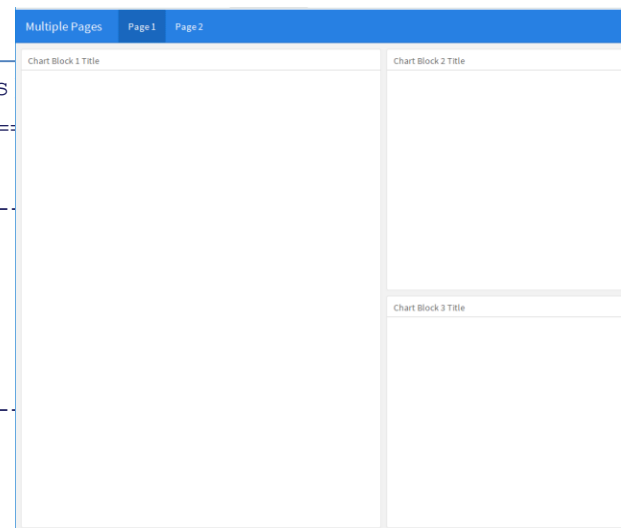
```
### Chart Block 1 Title  
```{r}  
```
```

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