Shiny and Flexdashboards

Creating dashboards and interactive websites in R

Joel Herndon, Center for Data and Visualization Sciences

Prerequisites

- Rstudio and R
- The following packages installed
  - tidyverse
  - flexdashboard
  - shiny

Goals for Today's Workshop

1. Building flexdashboards
2. Creating shiny apps
3. Understanding reactivity

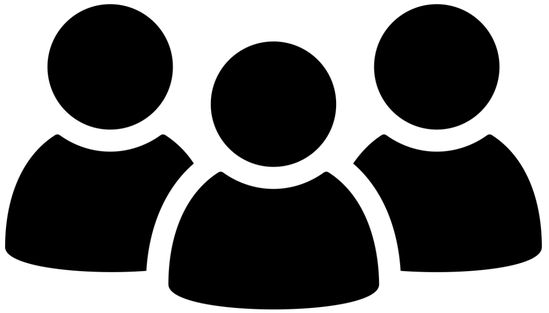
Plus:

1. Workflow tips/considerations
2. Resources for getting help
3. Hands-on exercises

Working assumptions

1. You have R experience, but are new to Shiny and flexdashboards
2. We're going to focus on a bit more on concepts than code
3. It's hard to see the screen in the back of this room! I'll try to make sure you can follow slides on your computer.

Let me know if you have questions or email askdata@duke.edu later.



2 minute meet and share

Dashboard Benefits

- Visually compelling
- Expands the audience for your data/argument
- Empowers you, your team, and your client/audience
- Can save time for both the dashboard creator and audience

Dashboard Types

Static Dashboards

- Easier to design
- Easy to share by email, web, slides
- Can be refreshed to reflect changes

Dynamic Dashboards

- Allow user interaction
- Require specialized code
- (Shiny) Requires a server to host

Data used in today's workshop

National Electronic Injury Surveillance System (NEISS)

- Focus: consumer product related injuries
- Sample: nationally representative sample of hospitals
- Coverage: United States, 2017
- Source: Hadley Wickham (https://github.com/hadley/neiss)

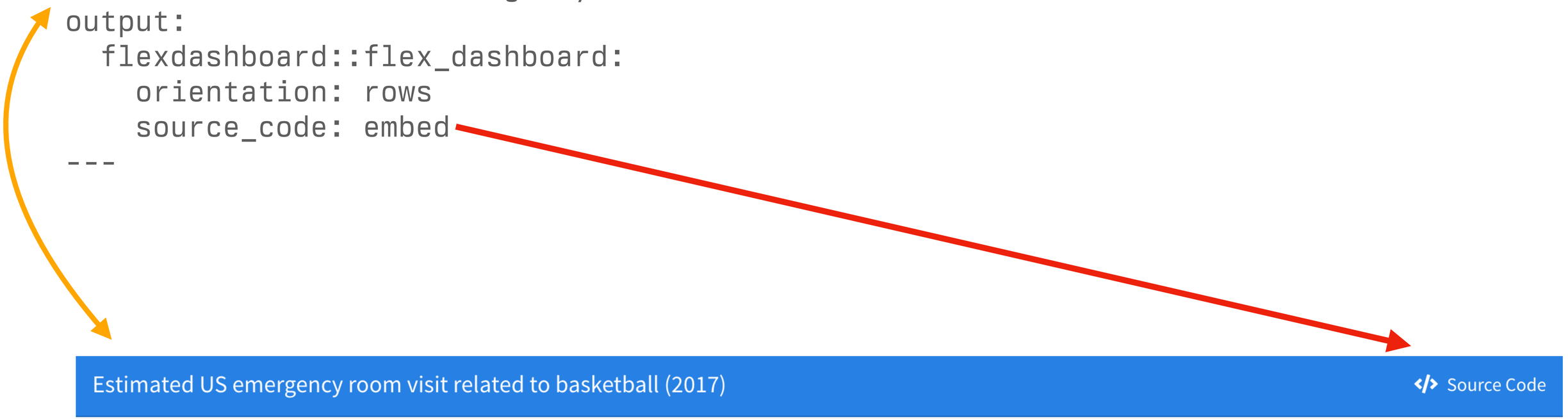
Building flexdashboards

Static Flexdashboards

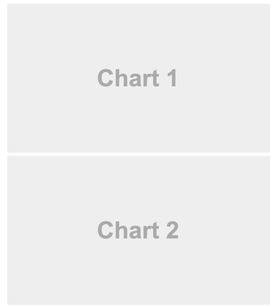
“Sometimes simpler is easier!”

1. YAML Header

```
---
title: "Estimated US emergency room visit related to basketball (2017)"
output:
  flexdashboard::flex_dashboard:
    orientation: rows
    source_code: embed
---
```



2. Pick a layout




```
---
title: "Focal Chart (Left)"
output: flexdashboard::flex_dashboard
---

Column {data-width=600}
-------------------------------------

### Chart 1

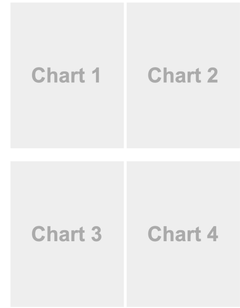
```{r}
```

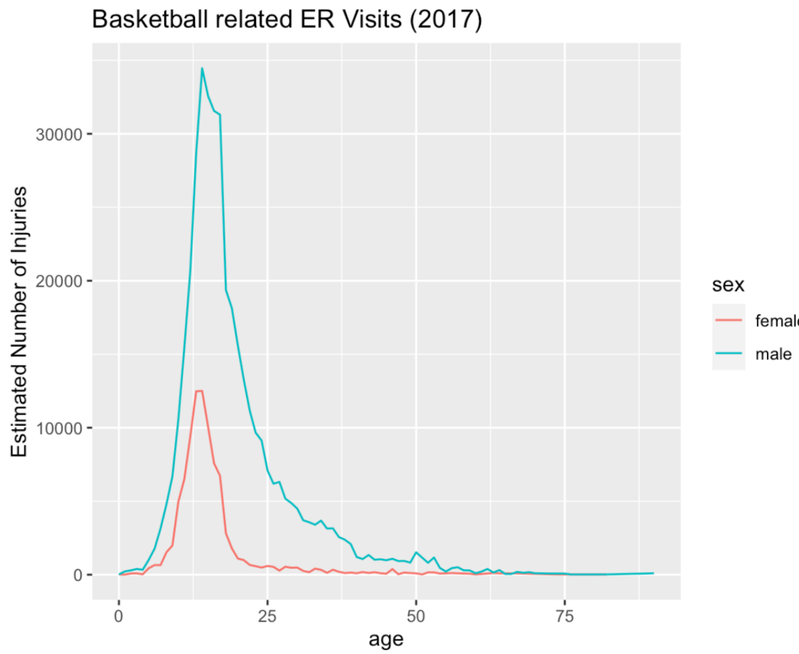
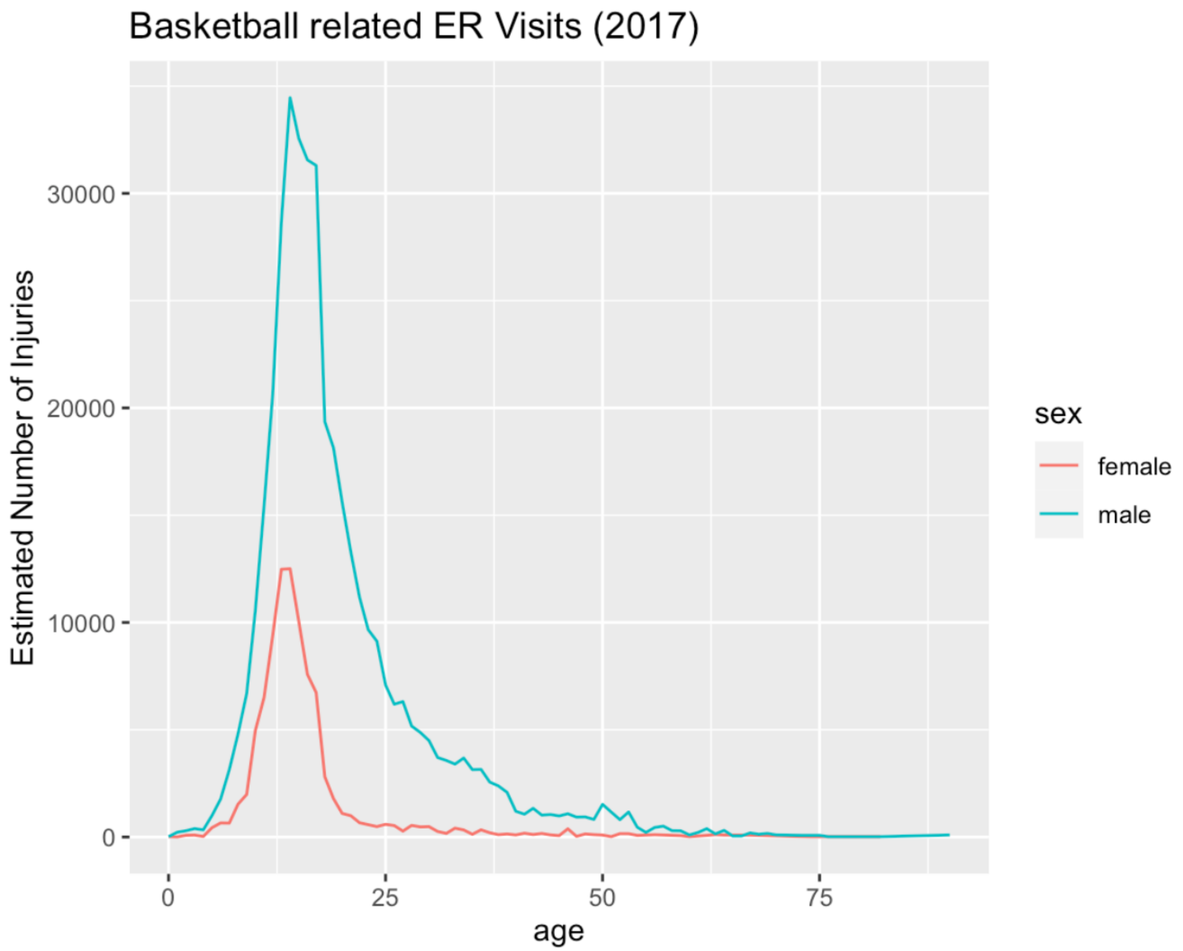
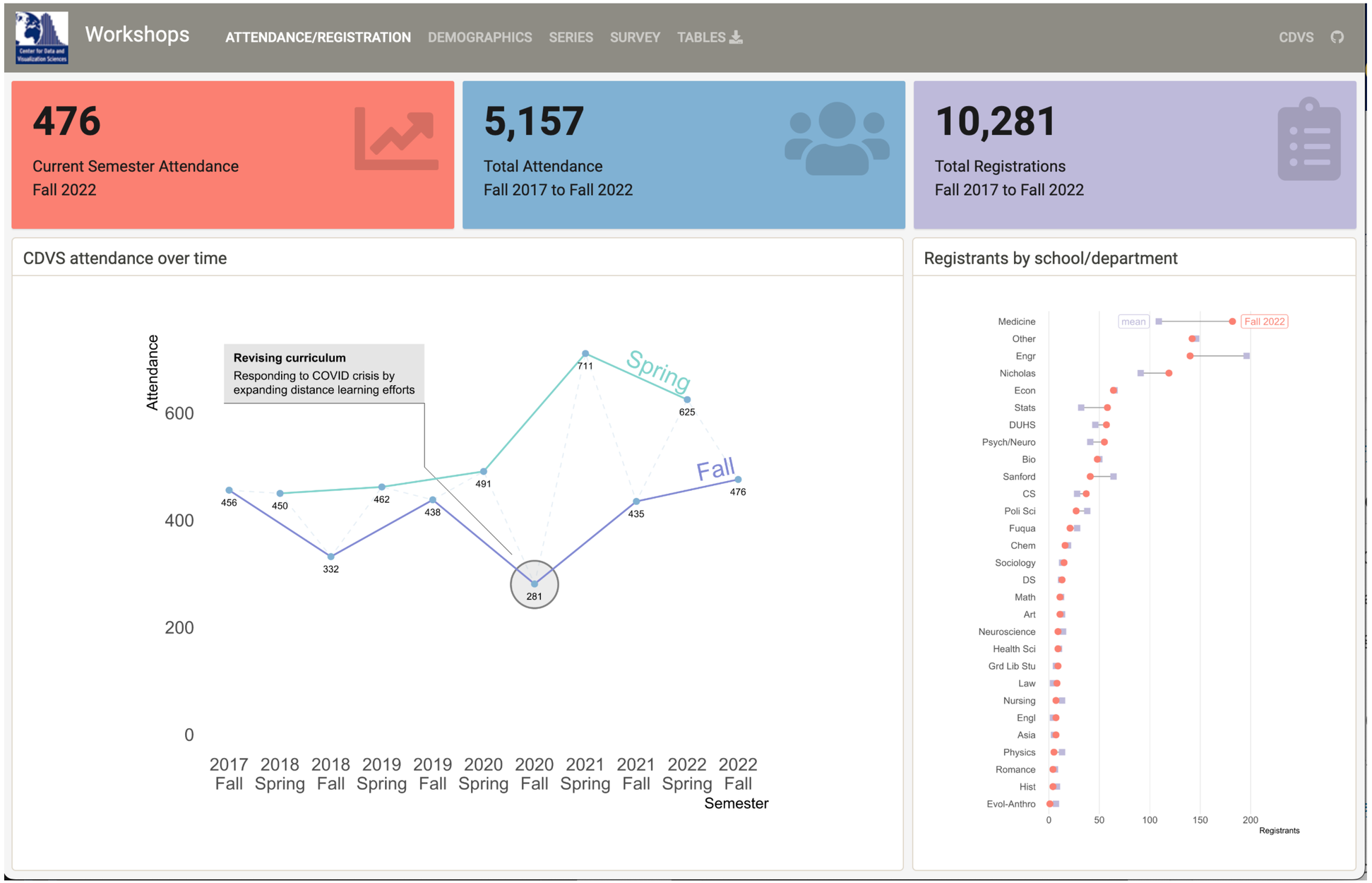
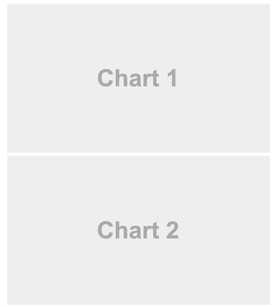
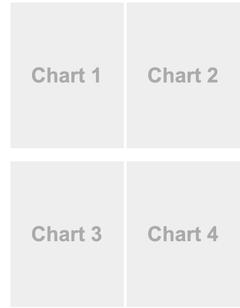
Column {data-width=400}

Chart 2

```{r}
```

Chart 3

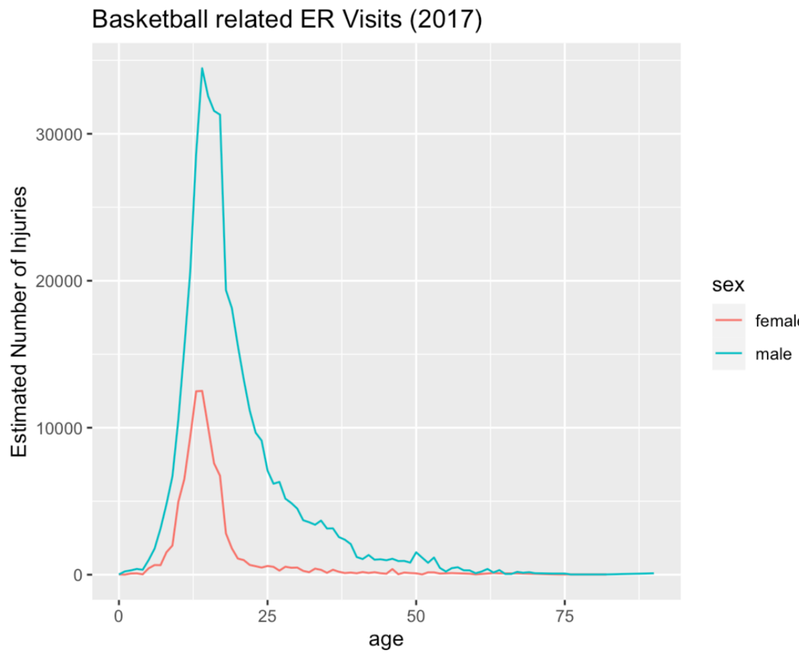
```{r}
```
```



For more layouts and the associated code: https://pkgs.rstudio.com/flexdashboard/articles/layouts.html

3. Add R (or shiny) code

````
### Basketball related ER visits {data-width=650}

```{r}
basketball_injuries_demographics <- injuries |>  filter(prod_code == 1205) |>
  count(age, sex, wt = weight, sort = TRUE)
basketball_injuries_demographics |>  ggplot(aes(x = age, y = n, color = sex)) +
  geom_line() +
  labs(title = "Basketball related ER Visits (2017)",
       y = "Estimated Number of Injuries")
```
````

Estimated US emergency room visit related to basketball (2017)

Basketball related ER visits



flexdashboards/01_static_flexdashboard.html

Estimated US emergency room visit related to basketball (2017)

Source Code

Basketball related ER visits



Top 20 demographic groups for basketball related ER visits (2017)

age	sex	n
14	male	34456.708
15	male	32565.140
16	male	31558.694
17	male	31300.324
13	male	28695.674
12	male	20628.453
18	male	19360.890
19	male	18137.962
20	male	15600.353
11	male	15425.811
21	male	13281.578
14	female	12503.775
13	female	12478.700
22	male	11176.448
10	male	10569.878
15	female	10059.873
23	male	9664.258
12	female	9389.971



Dashboard: John Little, CDVS

Dynamic Flexdashboards

1. YAML Header

```
---
title: "Estimated US emergency room visit related to basketball (2017)"
output:
  flexdashboard::flex_dashboard:
    orientation: rows
    source_code: embed
runtime: shiny
```

Enables interactive shiny code in the dashboard

2. Pick a layout




```
---
title: "Focal Chart (Left)"
output: flexdashboard::flex_dashboard
---

Column {data-width=600}
-------------------------------------

### Chart 1

```{r}
```

Column {data-width=400}

Chart 2

```{r}
```

Chart 3

```{r}
```
```



For more layouts and the associated code: https://pkgs.rstudio.com/flexdashboard/articles/layouts.html

3. Add R (or shiny) code

````
### Basketball related ER visits {data-width=650}

```{r}
basketball_injuries_demographics <- injuries |>  filter(prod_code == 1205) |>
  count(age, sex, wt = weight, sort = TRUE)
basketball_injuries_demographics |>  ggplot(aes(x = age, y = n, color = sex)) +
  geom_line() +
  labs(title = "Basketball related ER Visits (2017)",
       y = "Estimated Number of Injuries")
```
````

Estimated US emergency room visit related to basketball (2017)

Basketball related ER visits



flexdashboards/02_interactive_flexdashboard.Rmd

Estimated US emergency room visits (2017) | Dashboard | Data | Source Code

NEISS 2017 Explorer

The National Electronic Injury Surveillance System provides a representative sample of US emergency room visits associated with a consumer product. Data in this dashboard cover emergency room visits in 2017.

Product

basketball (activity, apparel or equipment)

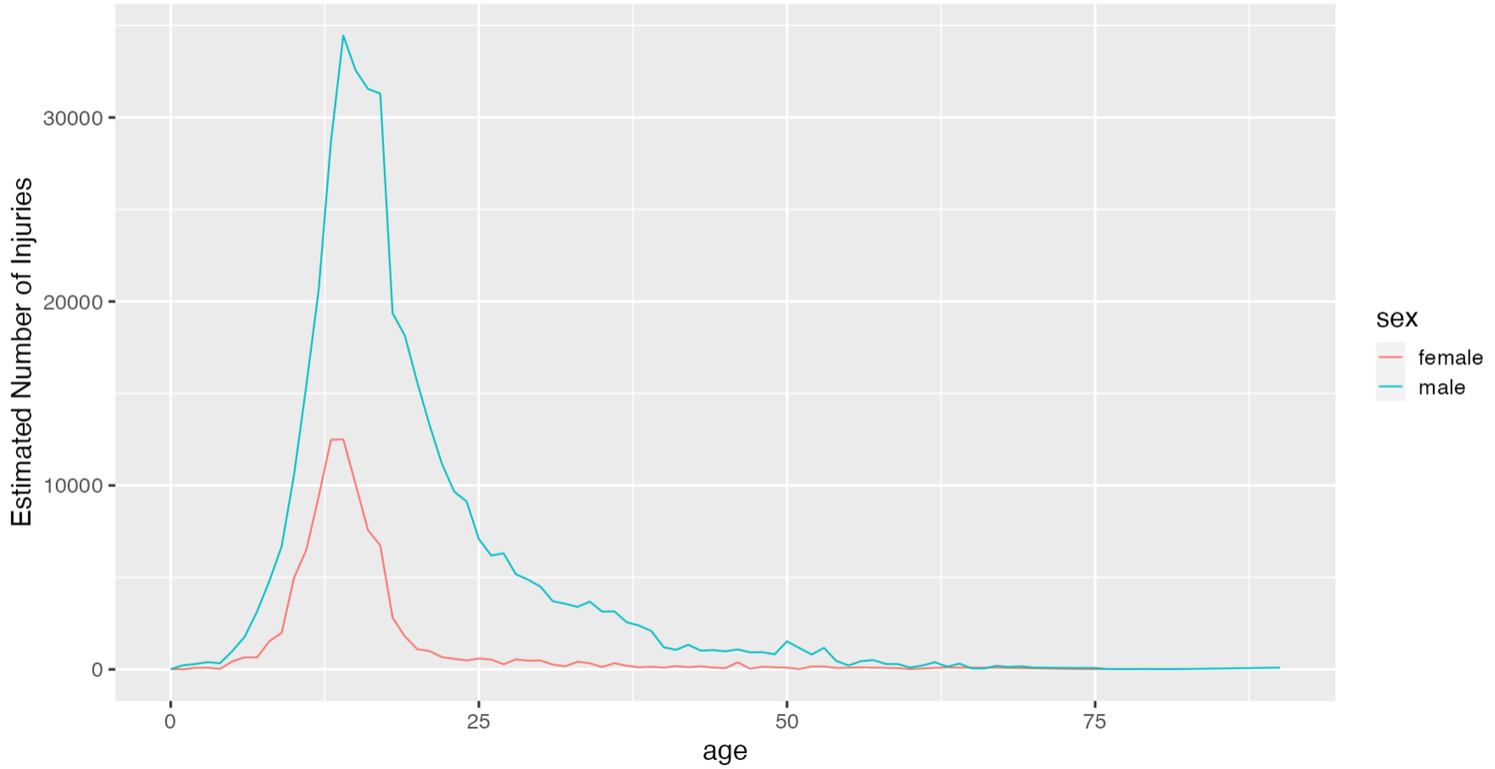
493,039

Number of injuries in the US (2017)

5.13%

Percentage of injuries with this product

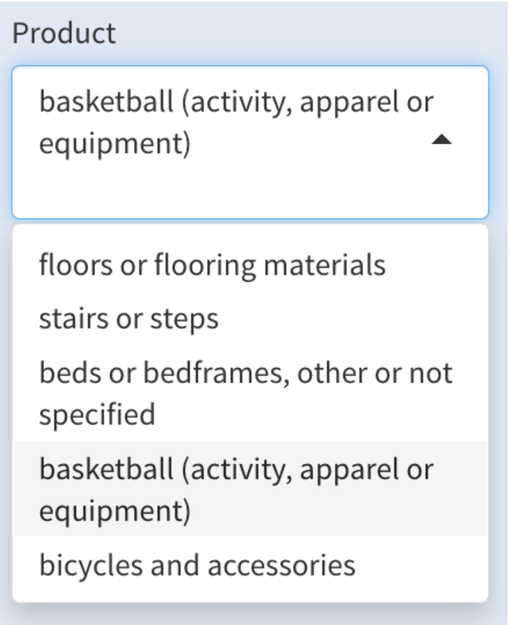
Emergency room visits related to the product



Let's try it!

flexdashboards/02_interactive_flexdashboard.Rmd

```
selectInput("code", label = "Product",
            choices = c( "floors or flooring materials" = 1807,
                         "stairs or steps" = 1842,
                         "beds or bedframes, other or not specified" = 4076,
                         "basketball (activity, apparel or equipment)" = 1205,
                         "bicycles and accessories" = 5040),
                          selected = 1205)
```



- Open "02_interactive_flexdashboard.Rmd"
- Change the default injury to "bicycles and accessories" (Hint: code = 5040)
- Click "Run Document"
- Change the default view option (next to "Run Document") to "Preview Window" or "Preview Pane"
- Click "Run Document" again

03:00

Building apps in

Building apps in

“But why not just use flexdashboards?”

Google trend index

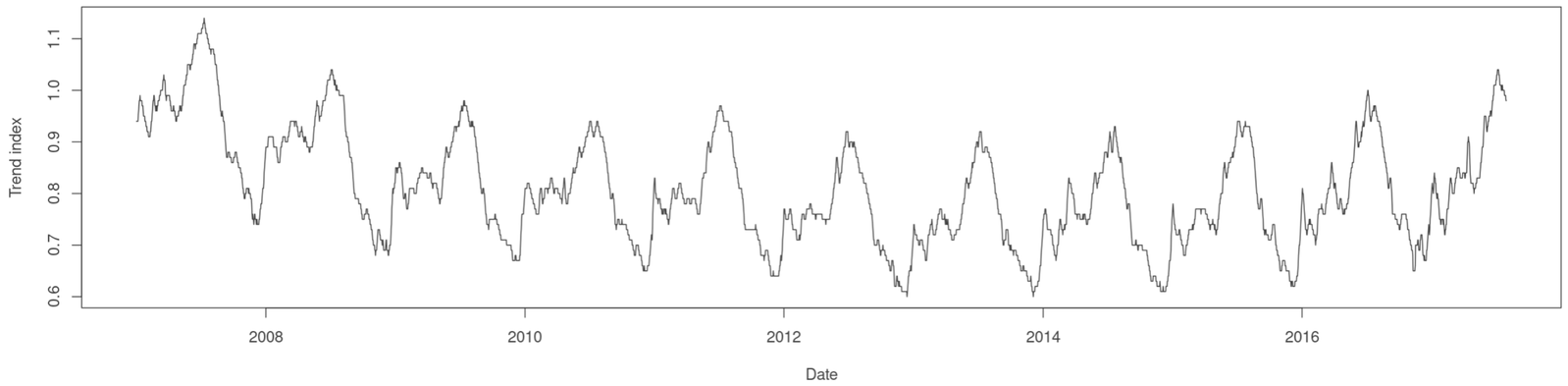
Trend index

Travel

Date range

2007-01-01 to 2017-07-31

Overlay smooth trend line



The Google Travel Index tracks queries related to airlines, hotels, beach, southwest, las vegas, flights, etc. The index is set to 1.0 on January 1, 2004 and is calculated only for US search traffic.
Source: Google Domestic Trends

bit.ly/g-trend

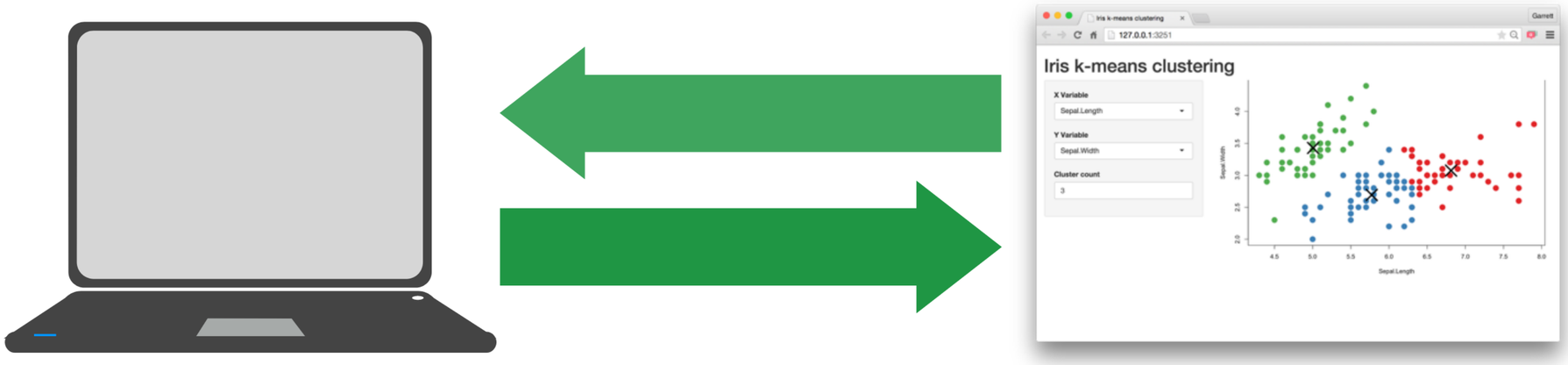
Let’s try it!

- Open a new file, and choose “Shiny Web App”
- Open (it should be open) the app.R file, and click “Run App”
- Select another output option by using the dropdown menu on “Run app”
- Click the stop sign icon in the Console to stop the app
- Change the default view option (next to “Run Document”) to “Preview Window” or “Preview Pane”

03:00

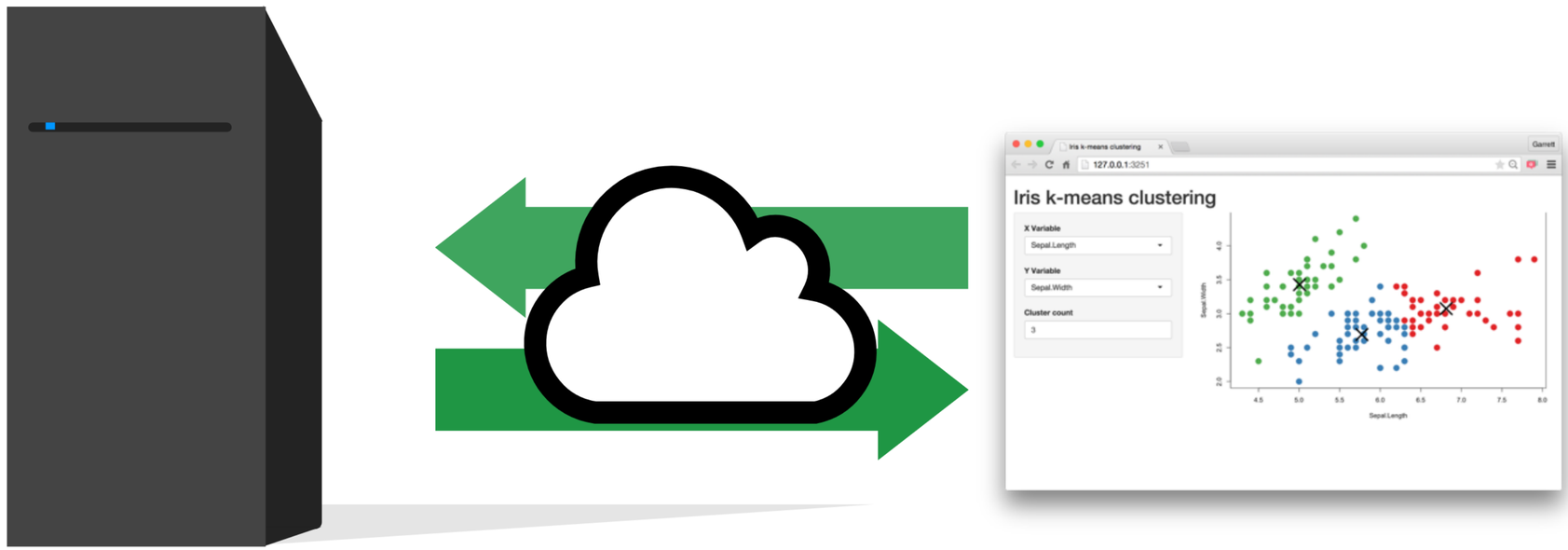
So how does shiny work?

Every Shiny app is maintained by a computer running R

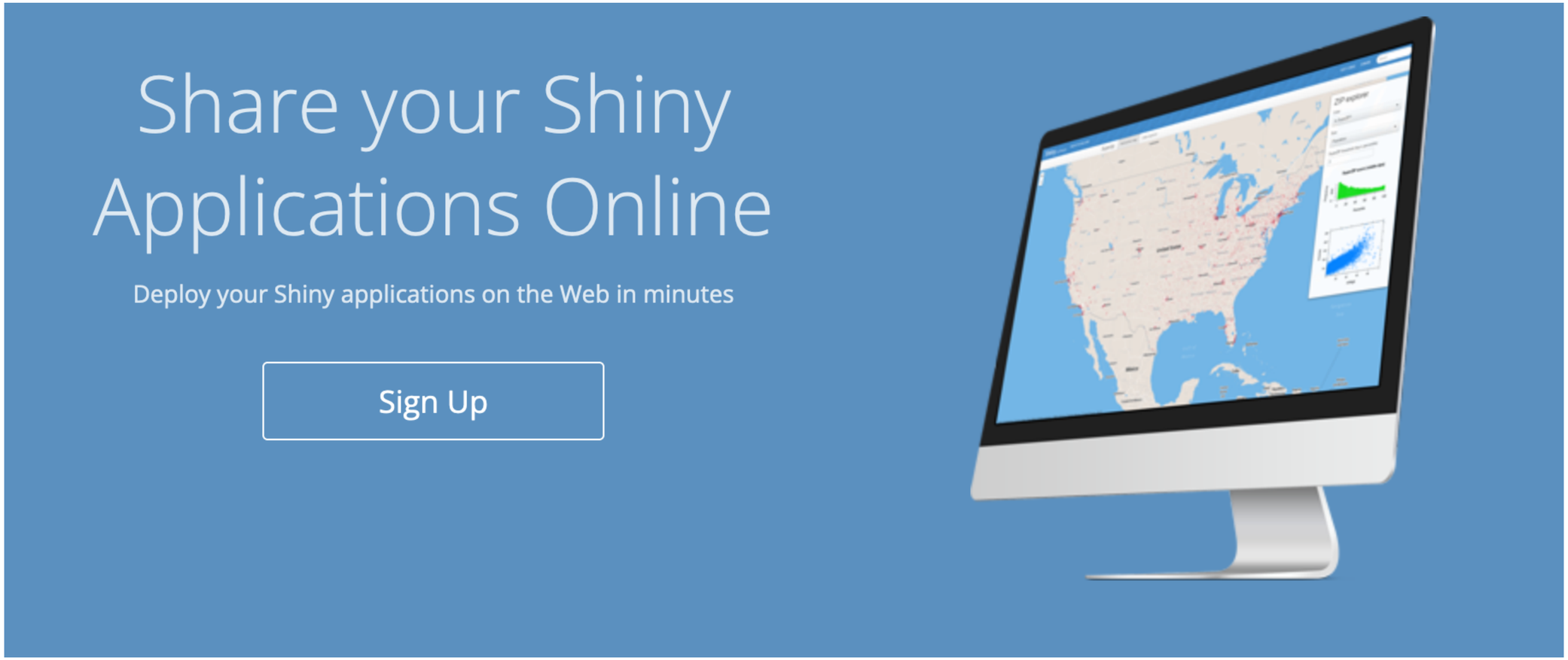


Source: Rstudio - https://bit.ly/408L3iB

Every Shiny app is maintained by a computer running R



Source: Rstudio - https://bit.ly/408L3iB



https://shinyapps.io

Server Instructions

User Interface (UI)

Source: Rstudio - https://bit.ly/408L3iB

Shiny apps have three elements

1. A **user interface** or “**ui**”

2. A **server** component

3. A s**hinyApp** function

```
library(shiny)

ui <- fluidPage(

)

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

}

shinyApp(ui, server)
```

Building a NEISS app using Shiny

National Electronic Injury Surveillance System (NEISS)

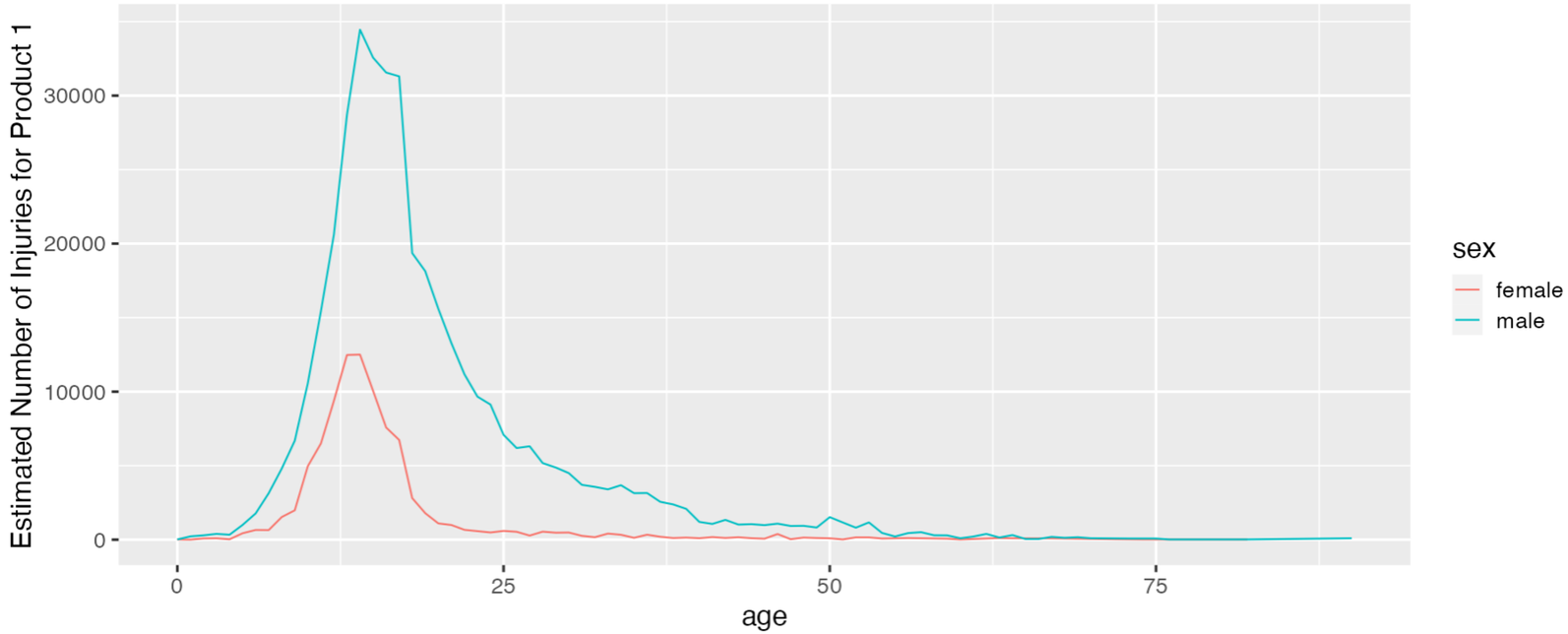
- Focus: consumer product related injuries
- Sample: nationally representative sample of hospitals
- Coverage: United States, 2017
- Source: Hadley Wickham (https://github.com/hadley/neiss)

NEISS Demographic Comparisons by Product

Product 1

basketball (activity, apparel or equipment)

- floors or flooring materials
- stairs or steps
- beds or bedframes, other or not specified
- basketball (activity, apparel or equipment)
- bicycles and accessories



shiny/neiss_shiny/app.r

App outline

```
library(shiny)
library(tidyverse)

injuries <- read_tsv("../../data/NEISS/injuries.tsv")
```

Header

```
ui <- fluidPage(

)
```

User Interface (ui)

```
server <- function(input, output, session) {

}
```

Server function

```
shinyApp(ui = ui, server = server)
```

shinyApp

App outline

```
library(shiny)
library(tidyverse)

injuries <- read_tsv("../../data/NEISS/injuries.tsv")
```

Header

```
ui <- fluidPage(

)
```

User Interface (ui)

```
server <- function(input, output, session) {

}
```

Server function

```
shinyApp(ui = ui, server = server)
```

shinyApp

Run once when app launches

App outline

```
library(shiny)
library(tidyverse)

injuries <- read_tsv("../../data/NEISS/injuries.tsv")
```

Header

```
ui <- fluidPage(

)
```

User Interface (ui)

```
server <- function(input, output, session) {

}
```

Server function

Runs once for each user

```
shinyApp(ui = ui, server = server)
```

shinyApp

user interface

```
ui <- fluidPage(

  # Application title
  titlePanel("NEISS Demographic Comparisons by Product"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      selectInput(inputID = “code", label = "Product 1",
                  choices = c( "floors or flooring materials" = 1807,
                               "stairs or steps" = 1842,
                               "beds or bedframes, other or not specified" = 4076,
                               "basketball (activity, apparel or equipment)" = 1205,
                               "bicycles and accessories" = 5040),
                  selected = 1205),
    ),

  # Show a plot of the generated distribution
    mainPanel(
    plotOutput("injurePlot1"),
    )
  )
)
```

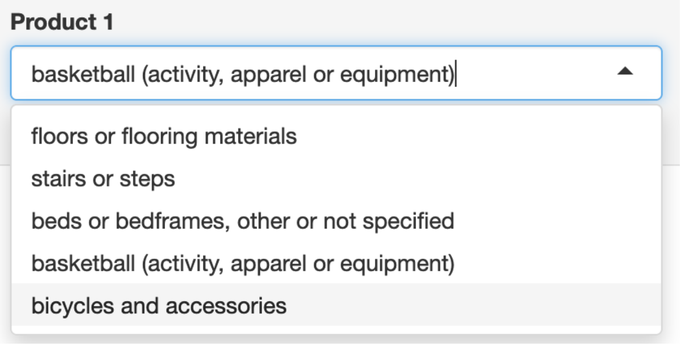
Create a responsive page

Add a title

We want a layout with a sidebar

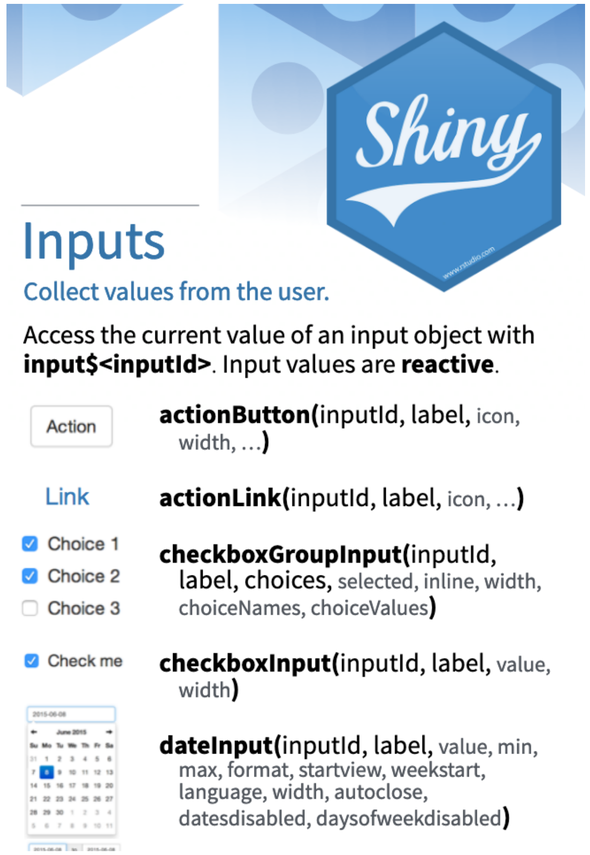
input$code

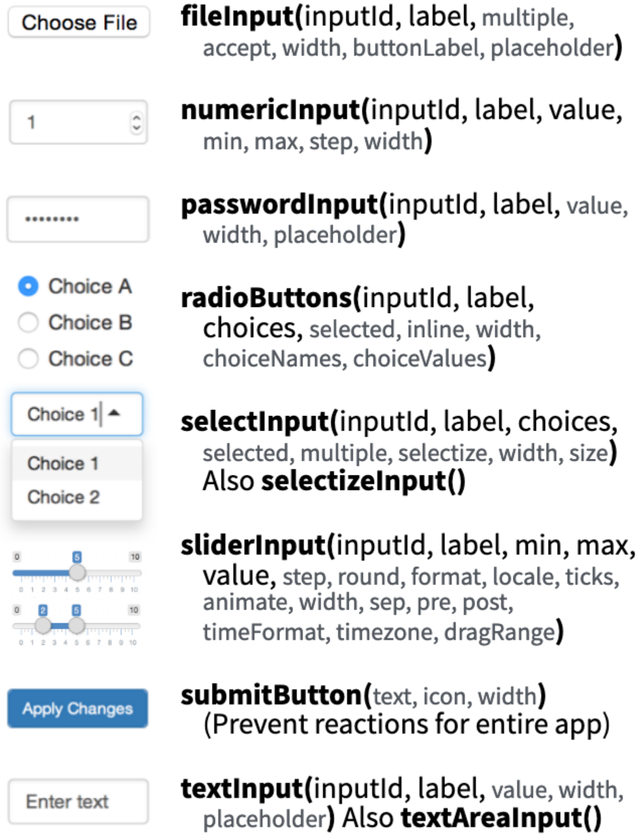
We create an output for our plot.



Let’s add an input!

Shiny offers a range of inputs!





https://shiny.rstudio.com/articles/cheatsheet.html

Exercise: Let’s add a slider

Instructions

- Open “shiny/neiss_shiny/app.r”
- Add the code for a **sliderInput (see pink box)**
- Enter the following properties:
  - inputID = “text_size”
  - label = “Magnify Labels”
  - min = 1
  - max = 25
  - value = 16
  - step =1

shiny/neiss_shiny/app.r

```
# Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            selectInput(inputId = “code",
                label = "Product 1",
                choices = c( "floors " = 1807,
                selected = 1205),
            hr(),

            sliderInput(inputId = "text_size",
                label = “”,
                min = ,
                max = ,
                value = ,
                step =
             ),
        ),
```

03:00

NEISS Demographic Comparisons by Product

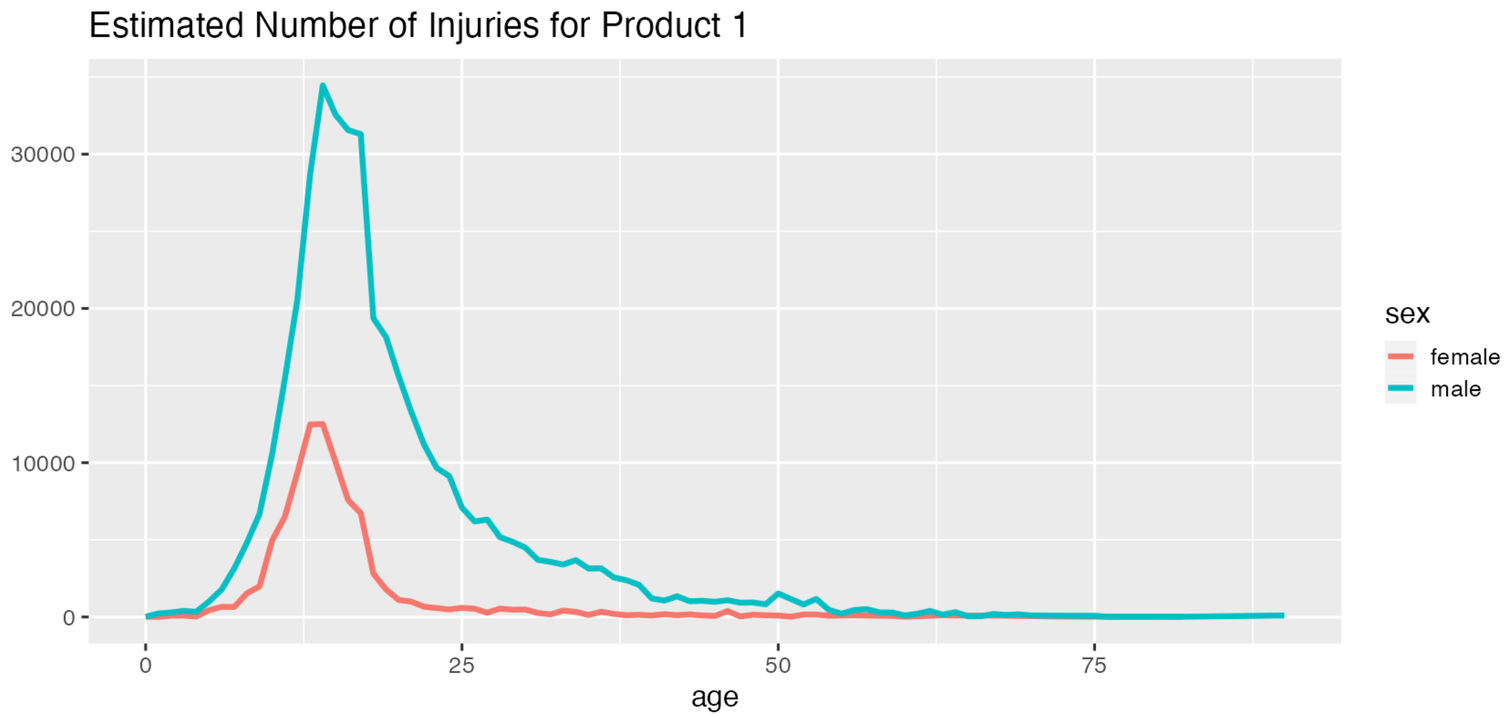
Product 1

basketball (activity, apparel or equipment)

Magnify labels

1 16 25

1 4 7 10 13 16 19 22 25



shiny/neiss_shiny_ui_change_themes/app.r

Server

```
server <- function(input, output) {
    injuries_demographics <- reactive ({
      injuries |>  filter(prod_code == input$code) |>
        count(age, sex, wt = weight, sort = TRUE)
    })

    output$injurePlot1 <-renderPlot({injuries_demographics() |>
        ggplot(aes(x = age, y = n, color = sex)) +
        geom_line() +
        labs(title = "", y = "Estimated Number of Injuries for Product 1") +
        theme_grey(base_size = 16)
    })
}
```

Subset the data

Server

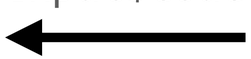
```
server <- function(input, output) {

    injuries_demographics <- reactive ({
      injuries |>  filter(prod_code == input$code) |>
        count(age, sex, wt = weight, sort = TRUE)

    })

    output$injurePlot1 <-renderPlot({injuries_demographics() |>
        ggplot(aes(x = age, y = n, color = sex)) +
        geom_line() +
        labs(title = "", y = "Estimated Number of Injuries for Product 1") +
        theme_grey(base_size = 16)
    })
}
```

input$code



Server

```
server <- function(input, output) {
    injuries_demographics <- reactive ({
      injuries |>  filter(prod_code == input$code) |>
        count(age, sex, wt = weight, sort = TRUE)
    })

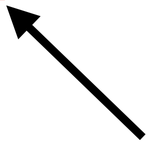
    output$injurePlot1 <-renderPlot({injuries_demographics() |>
        ggplot(aes(x = age, y = n, color = sex)) +
        geom_line() +
        labs(title = "", y = "Estimated Number of Injuries for Product 1") +
        theme_grey(base_size = 16)
    })
}
```

Create a graph

Server

```
server <- function(input, output) {
    injuries_demographics <- reactive ({
      injuries |>  filter(prod_code == input$code) |>
        count(age, sex, wt = weight, sort = TRUE)
    })

    output$injurePlot1 <-renderPlot({injuries_demographics() |>
        ggplot(aes(x = age, y = n, color = sex)) +
        geom_line() +
        labs(title = "", y = "Estimated Number of Injuries for Product 1") +
        theme_grey(base_size = 16)
    })
}
```

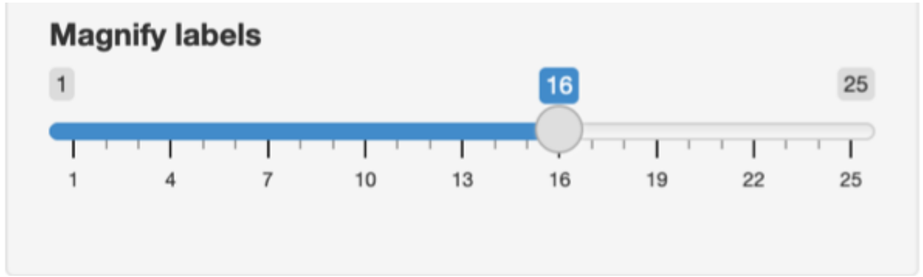


We are setting the size of the axis labels and titles here. We want the slider to control this.

Server

```
server <- function(input, output) {
    injuries_demographics <- reactive ({
      injuries |>  filter(prod_code == input$code) |>
        count(age, sex, wt = weight, sort = TRUE)
    })

    output$injurePlot1 <-renderPlot({injuries_demographics() |>
        ggplot(aes(x = age, y = n, color = sex)) +
        geom_line() +
        labs(title = "", y = "Estimated Number of Injuries for Product 1") +
        theme_grey(base_size = input$text_size)
    })
}
```



NEISS Demographic Comparisons by Product

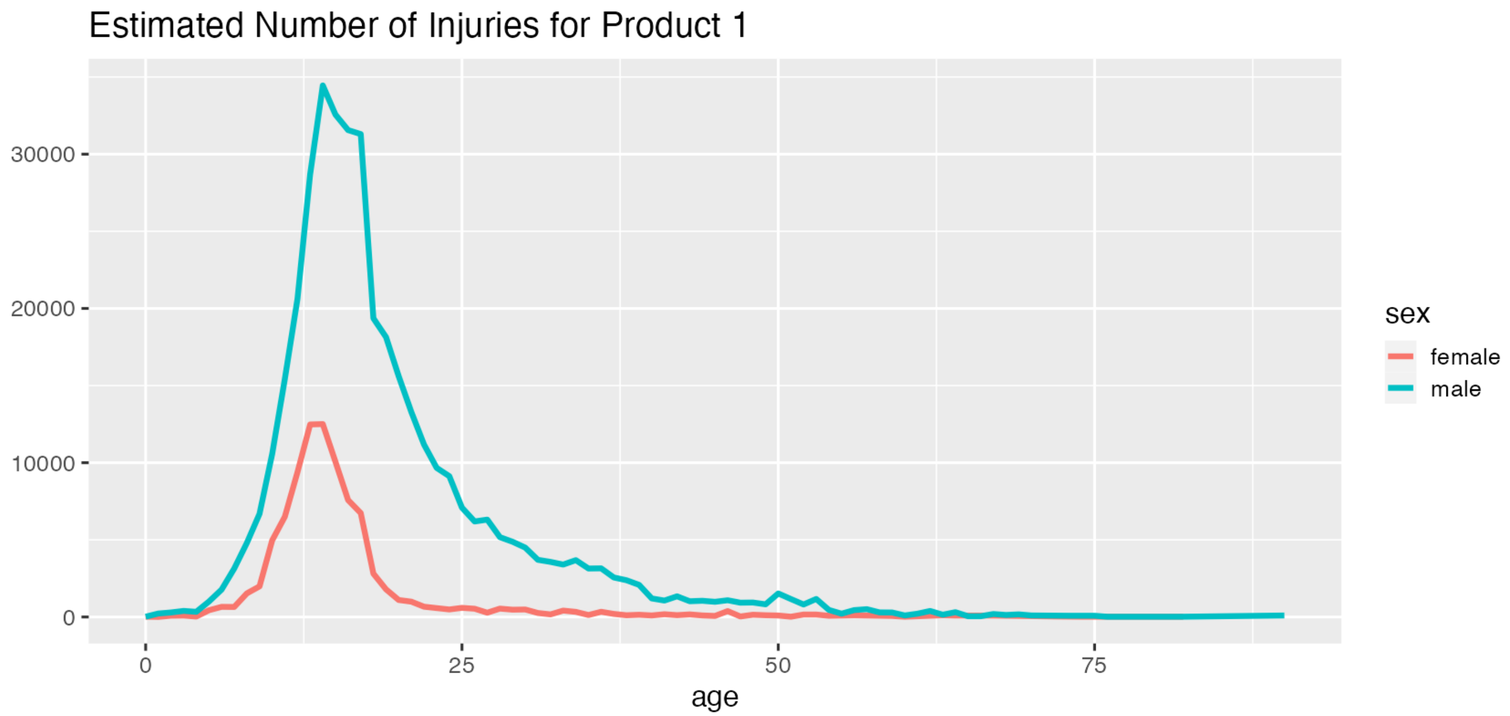
Product 1

basketball (activity, apparel or equipment)

Magnify labels

1 16 25

1 4 7 10 13 16 19 22 25



shiny/neiss_shiny_ui_change_themes/app.R

shinyApp

```
# Run the application
shinyApp(ui = ui, server = server)
```

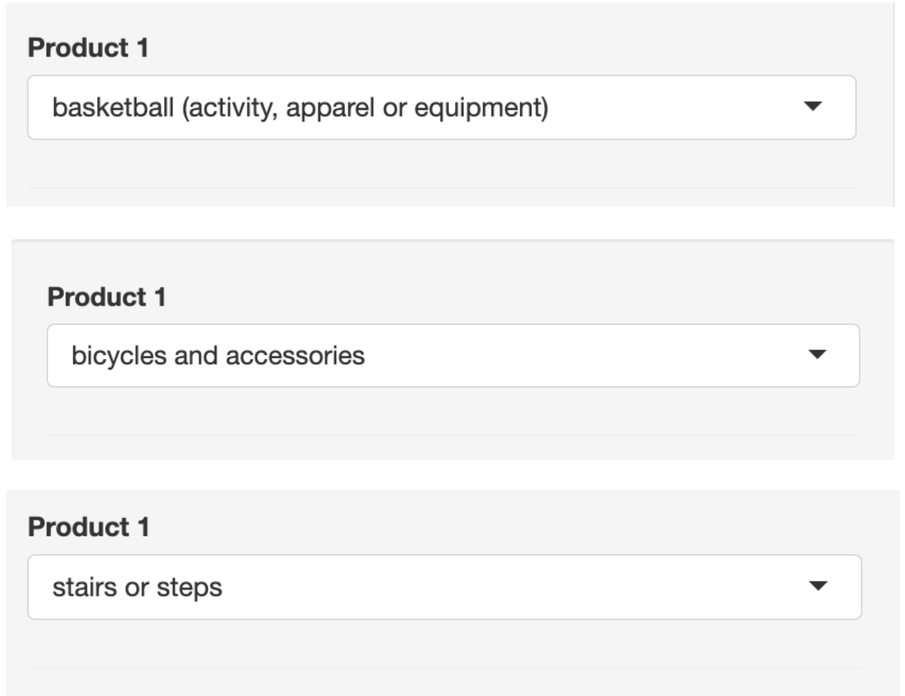
Questions?

Reactivity

(Shiny behind the interface)

Inputs

```
selectInput(inputID =“code",
            label = "Product 1",
            choices = c(“floors…” = 1807,
                        “Stairs…" = 1842,
                        “Beds…”   = 4076,
                        "basketball = 1205,
                        "bicycles" = 5040),
            selected = 1205),
```



input$code=1205

input$code=5040

input$code=1842

input$code

We are building a list of inputs as we select items in the user interface.

Outputs

Reactivity occurs when we use one of our inputs to create (render) an output.

```
server <- function(input, output) {
output$injurePlot1 <-renderPlot({injuries_demographics() |>
        ggplot(aes(x = age, y = n, color = sex)) +
        geom_line() +
        labs(title = "", y = "Estimated Number of Injuries for Product 1") +
        theme_grey(base_size = input$text_size)
    })
```

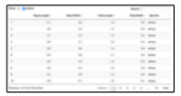


```
output$injurePlot1
```

Shiny offers a range of outputs!

Outputs

render*() and *Output() functions work together to add R output to the UI.

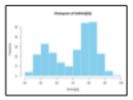


DT::**renderDataTable(**expr, options, searchDelay, callback, escape, env, quoted, outputArgs**)**

dataTableOutput(outputId**)**

renderImage(expr, env, quoted, deleteFile, outputArgs**)**

imageOutput(outputId, width, height, click, dblclick, hover, brush, inline**)**

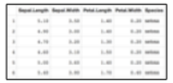


renderPlot(expr, width, height, res, …, alt, env, quoted, execOnResize, outputArgs**)**

plotOutput(outputId, width, height, click, dblclick, hover, brush, inline**)**

renderPrint(expr, env, quoted, width, outputArgs**)**

verbatimTextOutput(outputId, placeholder**)**



renderTable(expr, striped, hover, bordered, spacing, width, align, rownames, colnames, digits, na, …, env, quoted, outputArgs**)**

tableOutput(outputId**)**

foo

renderText(expr, env, quoted, outputArgs, sep**)**

textOutput(outputId, container, inline**)**

renderUI(expr, env, quoted, outputArgs**)**

uiOutput(outputId, inline, container, …**)**
htmlOutput(outputId, inline, container, …**)**

These are the core output types. See **htmlwidgets.org** for many more options.

https://shiny.rstudio.com/articles/cheatsheet.html

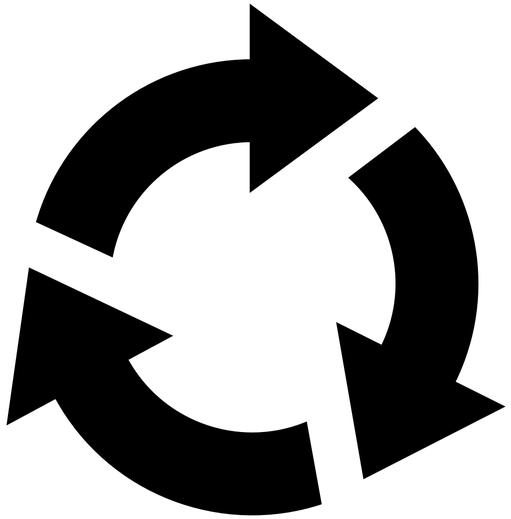
Reactive Outputs

Reactive expressions that return a value based on a reactive value or expression are also common.

```
server <- function(input, output) {
    injuries_demographics <- reactive ({
      injuries |>  filter(prod_code == input$code) |>
        count(age, sex, wt = weight, sort = TRUE)
    })
```


```
 output$injurePlot1 <-renderPlot({injuries_demographics() |>
         ggplot(aes(x = age, y = n, color = sex)) +
         geom_line() +
         labs(title = "", y = "Estimated Number of Injuries for Product 1") +
         theme_grey(base_size = input$text_size)
     })
```

Reactivity Considerations



Questions and next steps…

Working with Dashboards in R

Closing thoughts on dashboards in R

1. Develop your content for your dashboard independently of the design
2. Embrace iterative design (start simple and add features)
3. Consider your end user (interface design, hosting of dashboard, concepts shared)

A special thanks to:

- Mine Çetinkaya Rundell for pedagogy and structure suggestions for this workshop
- Garrett Grolemund and RStudio for inspiration on the slides in the “shiny basics” section
- Hadley Wickham for his subset on the NEISS data and Mastering shiny

Online resources for learning more…

Online Reference and Tutorials

- **Flexdashboard:** pkgs.rstudio.com/flexdashboard
- **Shiny**: shiny.rstudio.com
  mastering-shiny.org

Training videos

- **Shiny Workshop:** rfun.library.duke.edu/portfolio/shiny_workshop

Reactivity Exercise

1. Add a second select menu to create a second injury plot

- `inputId = “code2”`
- `label = “Product 2”`
- `choices = (you can use the same choices as the first select)`
- `selected = 5040`

2. Create a second plot by

- Copy the code inside the server function and paste a copy below the existing code
- In the copied code:
  - Change `injuries_demographics` to `injuries_demographics2`
  - Change `input$code` to `input$code2`
  - Change `output$injurePlot1` to `output$injurePlot2`
  - Change `injuries_demographics()` to `injuries_demographics2()`

3. Run the new app

05:00

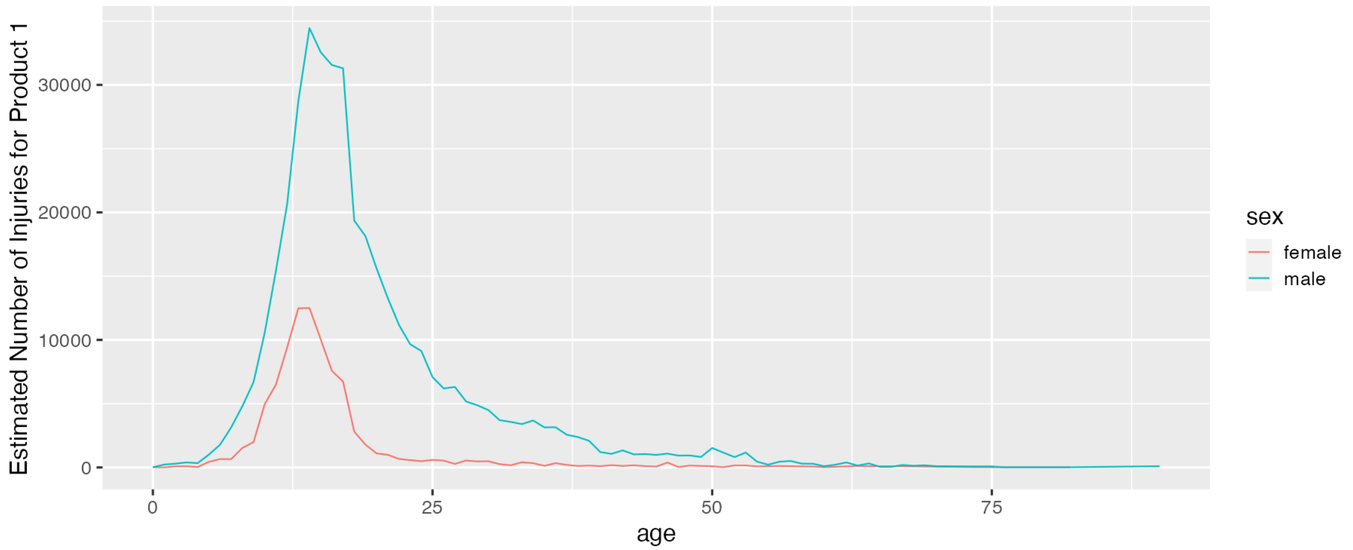
NEISS Demographic Comparisons by Product

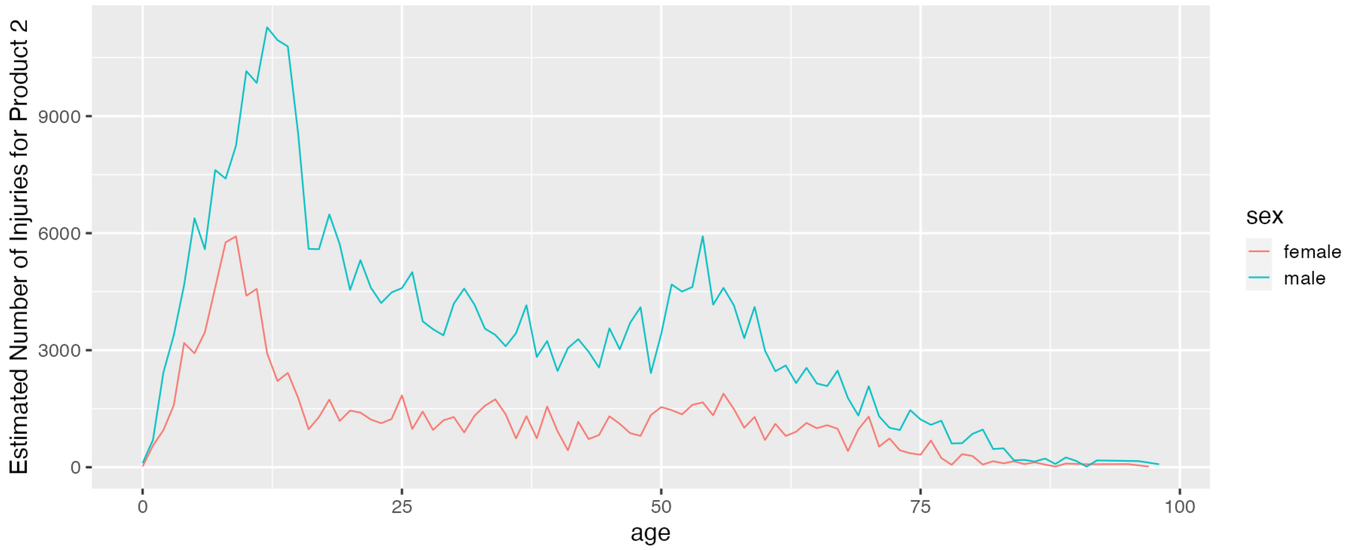
Product 1

basketball (activity, apparel or equipment)

Product 2

bicycles and accessories





shiny/neiss_shiny_product_compare/app.R