AHRQ
Agency for Healthcare
Research and Quality

Shiny applications without Shiny

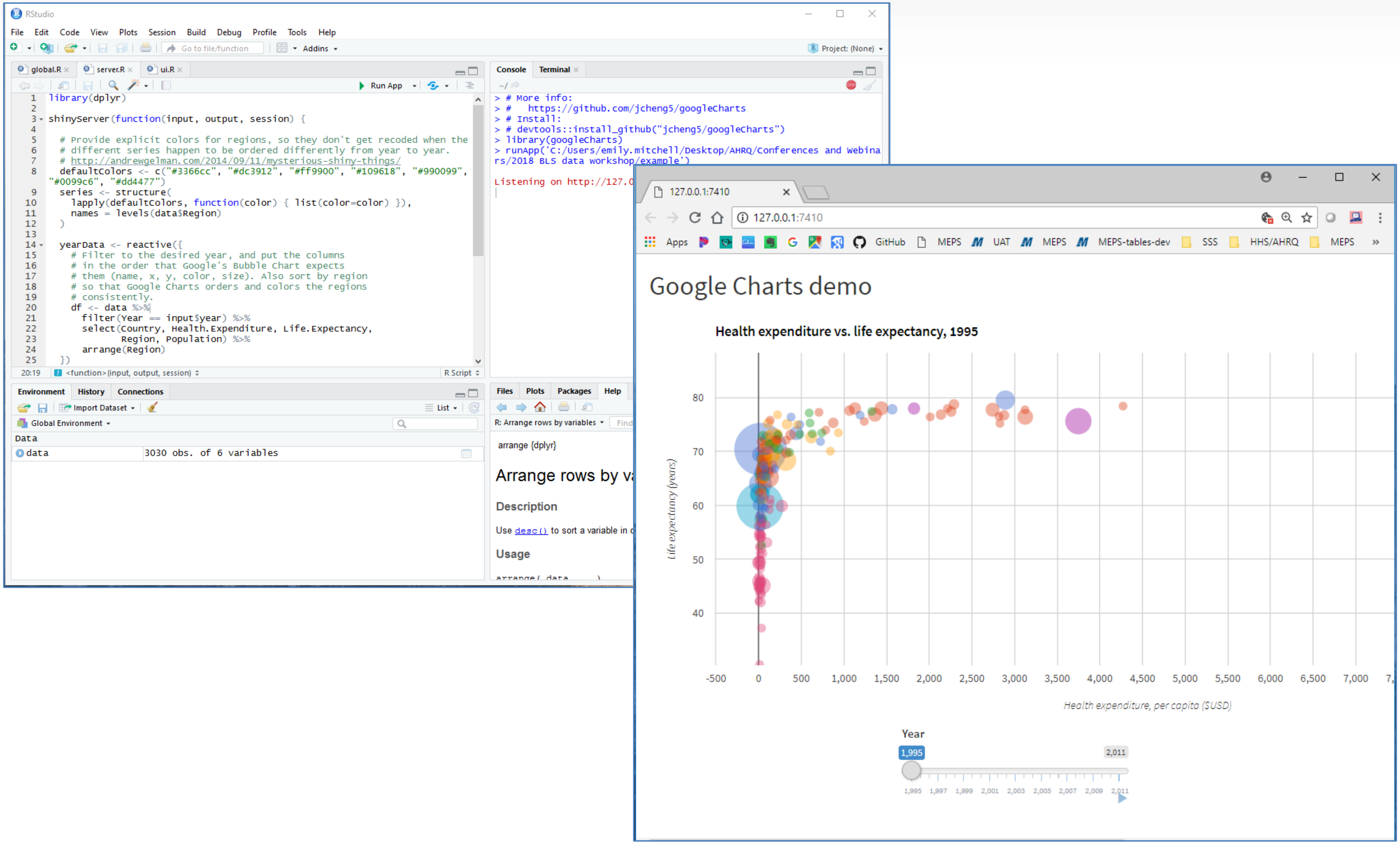
Emily Mitchell

October 25, 2018

Disclaimer

The views expressed in this presentation are those of the author and no official endorsement by the Department of Health and Human Services, the Agency for Healthcare Research and Quality is intended or should be inferred.

What is Shiny?



```r
library(dplyr)

shinyServer(function(input, output, session) {

  # Provide explicit colors for regions, so they don't get recoded when the
  # different series happen to be ordered differently from year to year.
  # http://andrewgelman.com/2014/09/11/mysterious-shiny-things/
  defaultColors <- c("#3366cc", "#dc3912", "#ff9900", "#109618", "#990099",
"#0099c6", "#dd4477")
  series <- structure(
    lapply(defaultColors, function(color) { list(color=color) }),
    names = levels(data$Region)
  )

  yearData <- reactive({
    # Filter to the desired year, and put the columns
    # in the order that Google's Bubble Chart expects
    # them (name, x, y, color, size). Also sort by region
    # so that Google Charts orders and colors the regions
    # consistently.
    df <- data %>%
      filter(Year == input$year) %>%
      select(Country, Health.Expenditure, Life.Expectancy,
             Region, Population) %>%
      arrange(Region)
  })
```

```
> # More info:
> #   https://github.com/jcheng5/googleCharts
> # Install:
> # devtools::install_github("jcheng5/googleCharts")
> library(googleCharts)
> runApp('C:/Users/emily.mitchell/Desktop/AHRQ/Conferences and Webina
rs/2018 BLS data workshop/example')

Listening on http://127.0
```

Data

data 3030 obs. of 6 variables

arrange {dplyr}

Arrange rows by va

Description

Use desc() to sort a variable in d

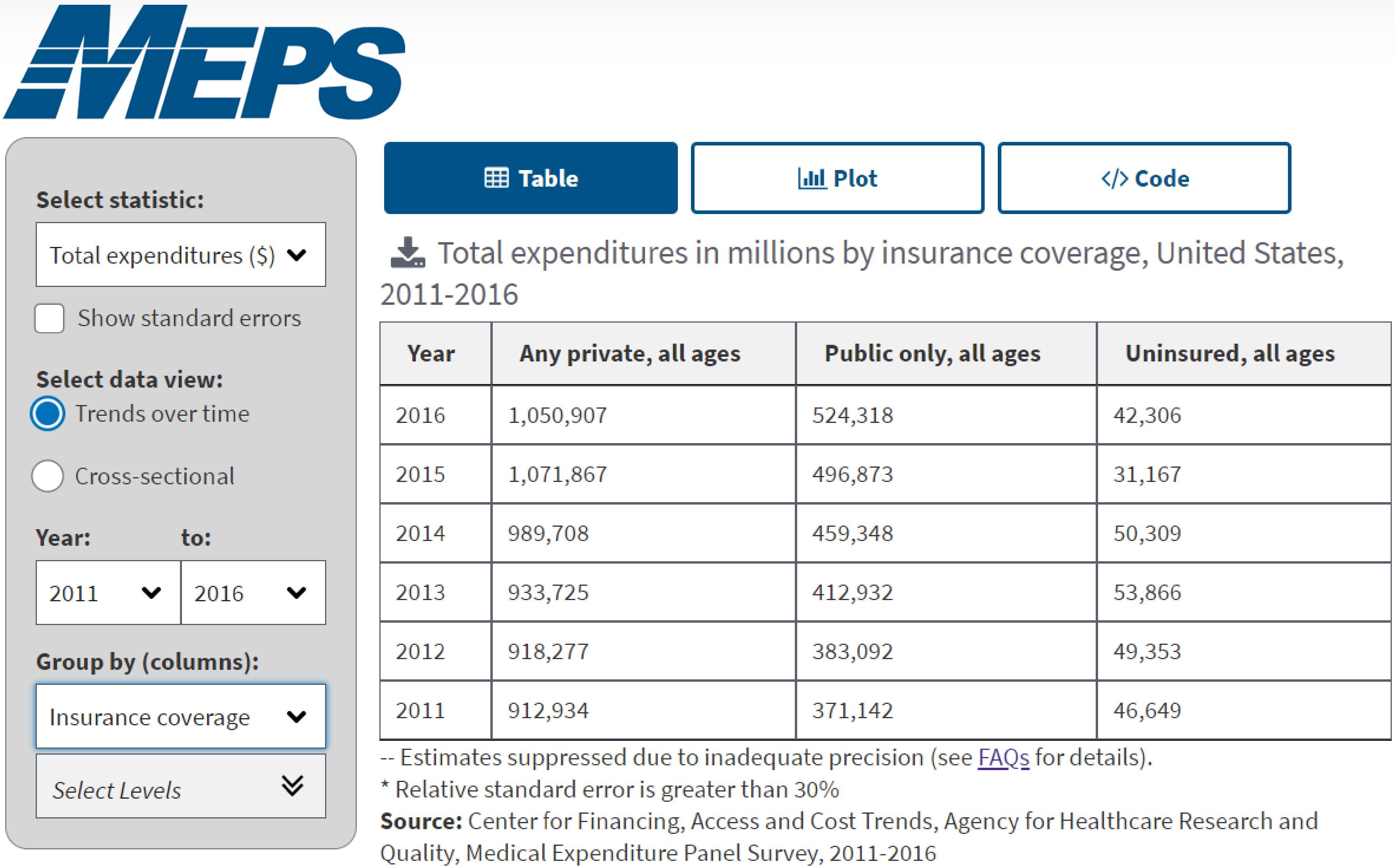
Usage

arrange(data, ...)

Google Charts demo

Health expenditure vs. life expectancy, 1995

Motivation



Select statistic: Total expenditures ($)

□ Show standard errors

Select data view:

● Trends over time

○ Cross-sectional

Year: 2011 **to:** 2016

Group by (columns): Insurance coverage

Select Levels

Total expenditures in millions by insurance coverage, United States, 2011-2016

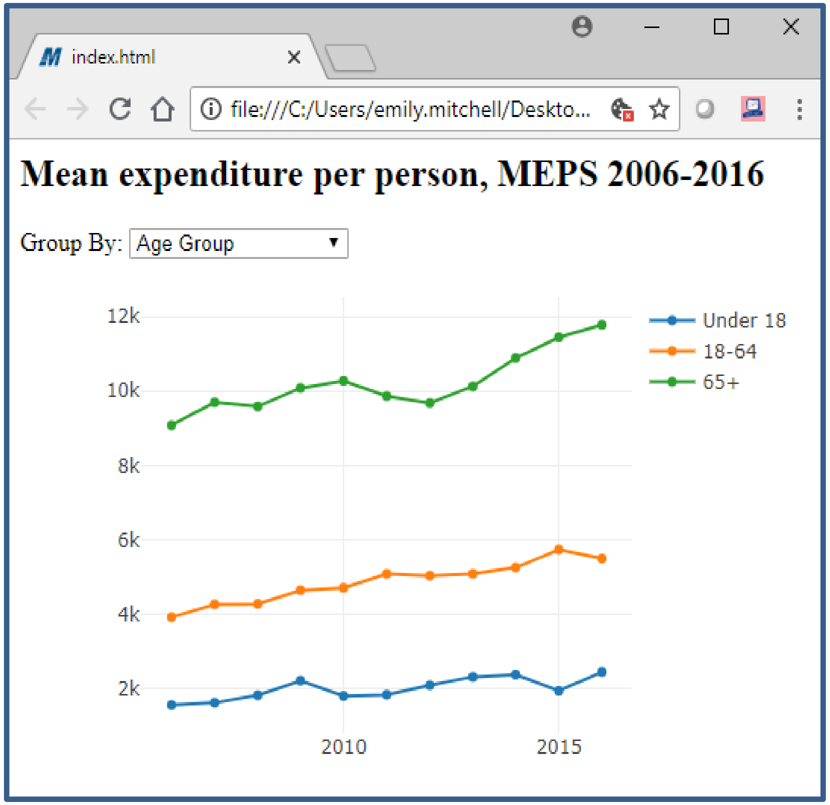
Year	Any private, all ages	Public only, all ages	Uninsured, all ages
2016	1,050,907	524,318	42,306
2015	1,071,867	496,873	31,167
2014	989,708	459,348	50,309
2013	933,725	412,932	53,866
2012	918,277	383,092	49,353
2011	912,934	371,142	46,649

-- Estimates suppressed due to inadequate precision (see FAQs for details).

* Relative standard error is greater than 30%

Source: Center for Financing, Access and Cost Trends, Agency for Healthcare Research and Quality, Medical Expenditure Panel Survey, 2011-2016

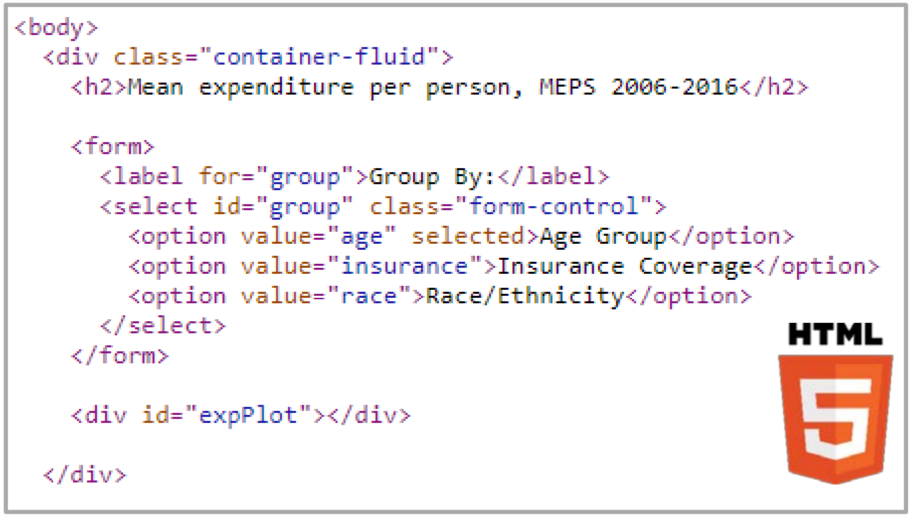
Anatomy of a web page



Mean expenditure per person, MEPS 2006-2016

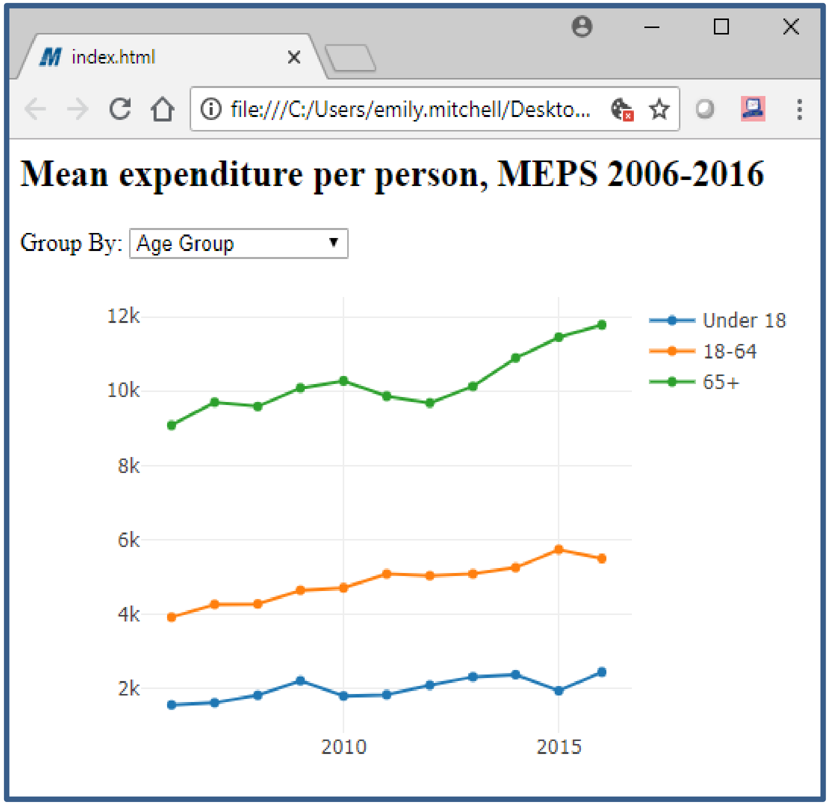
Group By: Age Group ▾

HTML



```
<body>
  <div class="container-fluid">
    <h2>Mean expenditure per person, MEPS 2006-2016</h2>

    <form>
      <label for="group">Group By:</label>
      <select id="group" class="form-control">
        <option value="age" selected>Age Group</option>
        <option value="insurance">Insurance Coverage</option>
        <option value="race">Race/Ethnicity</option>
      </select>
    </form>

    <div id="expPlot"></div>

  </div>
```

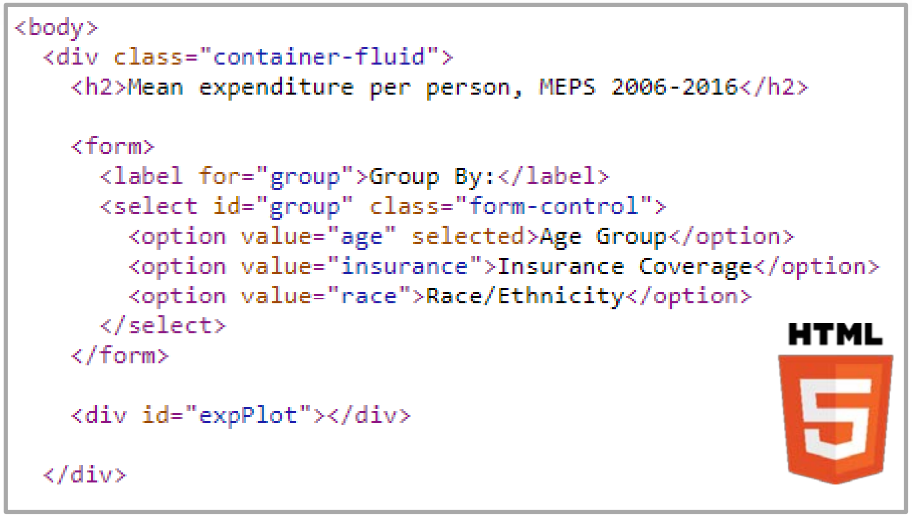
Anatomy of a web page



Mean expenditure per person, MEPS 2006-2016

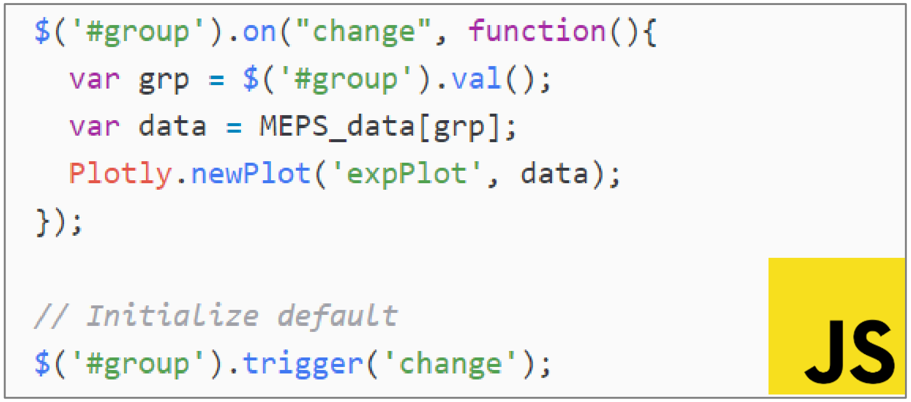
Group By: Age Group ▾

HTML



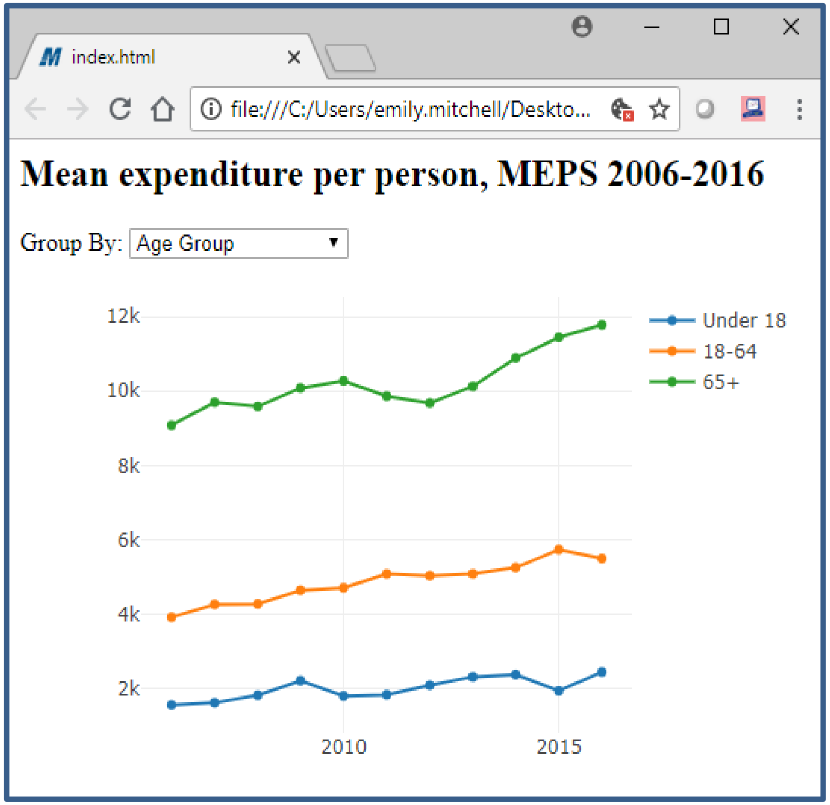
```html
<body>
  <div class="container-fluid">
    <h2>Mean expenditure per person, MEPS 2006-2016</h2>

    <form>
      <label for="group">Group By:</label>
      <select id="group" class="form-control">
        <option value="age" selected>Age Group</option>
        <option value="insurance">Insurance Coverage</option>
        <option value="race">Race/Ethnicity</option>
      </select>
    </form>

    <div id="expPlot"></div>

  </div>
```

JavaScript (JS)



```javascript
$('#group').on("change", function(){
  var grp = $('#group').val();
  var data = MEPS_data[grp];
  Plotly.newPlot('expPlot', data);
});

// Initialize default
$('#group').trigger('change');
```

Anatomy of a web page



Mean expenditure per person, MEPS 2006-2016

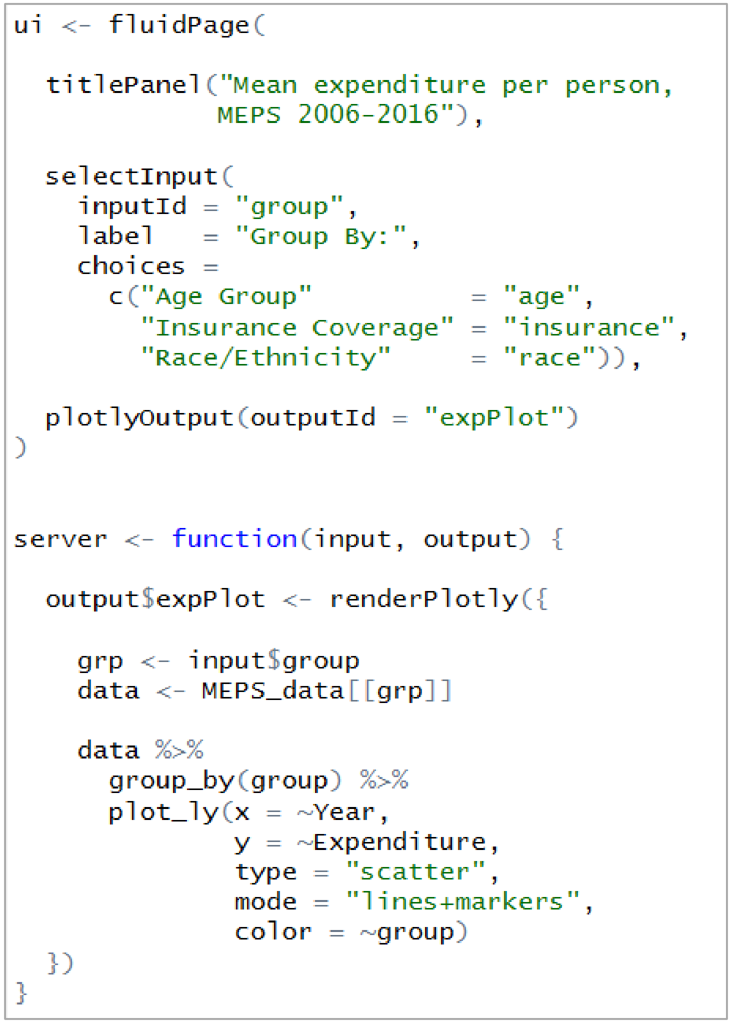
Group By: Age Group ▾

Cascading Style Sheets (CSS)

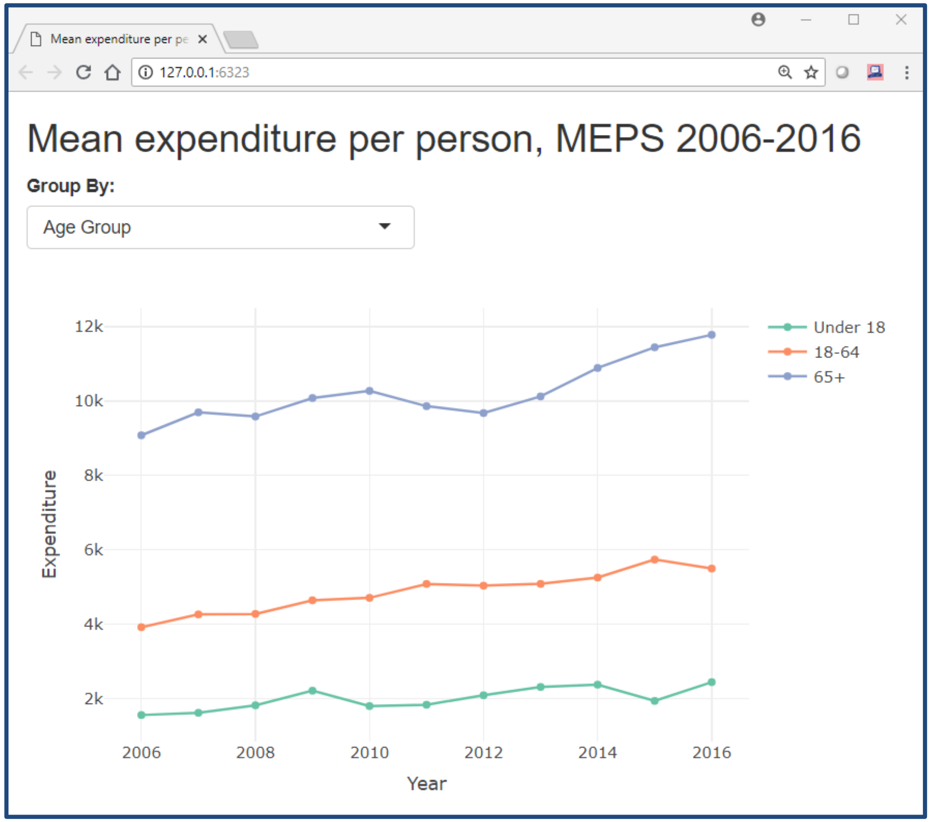
```
table {table-layout: auto;}
td, th {white-space: nowrap;}

footer a {text-decoration: none;}
header a {text-decoration: none;}

.tooltip {
  bottom: auto;
  background: none;
  width: auto;
  font-size: 14px;
}
```

Anatomy of a Shiny app

R – Shiny app



```
ui <- fluidPage(

  titlePanel("Mean expenditure per person,
             MEPS 2006-2016"),

  selectInput(
    inputId = "group",
    label   = "Group By:",
    choices =
      c("Age Group"              = "age",
        "Insurance Coverage" = "insurance",
        "Race/Ethnicity"     = "race")),

  plotlyOutput(outputId = "expPlot")
)


server <- function(input, output) {

  output$expPlot <- renderPlotly({

    grp <- input$group
    data <- MEPS_data[[grp]]

    data %>%
      group_by(group) %>%
      plot_ly(x = ~Year,
              y = ~Expenditure,
              type = "scatter",
              mode = "lines+markers",
              color = ~group)
  })
}
```



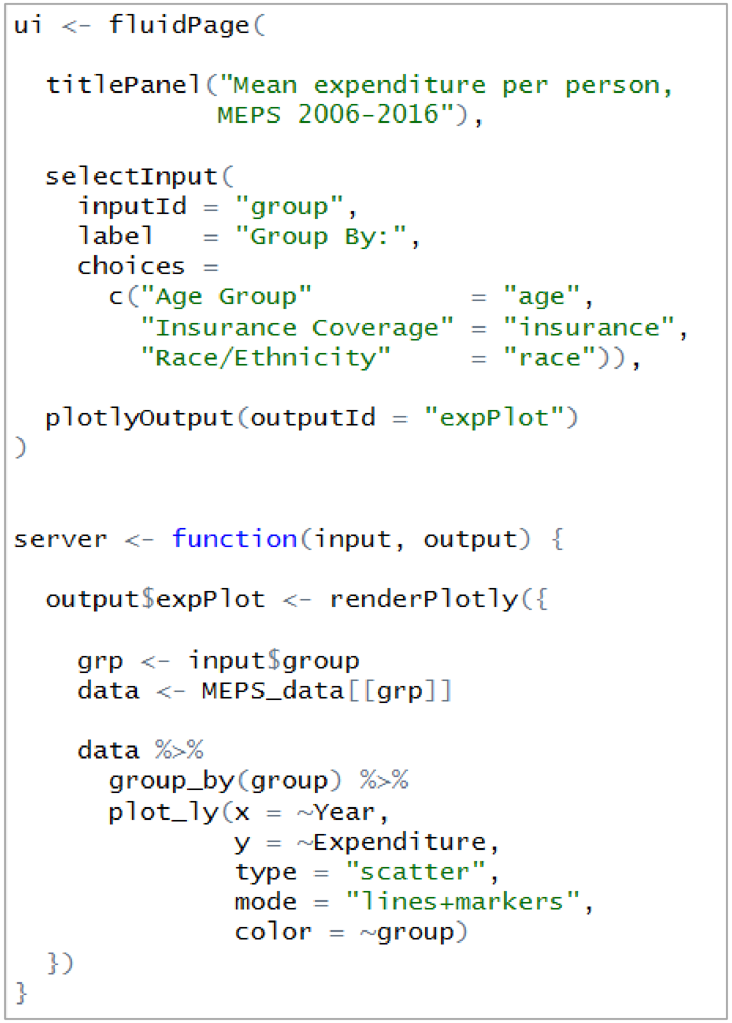
Mean expenditure per person, MEPS 2006-2016

Group By:

Age Group

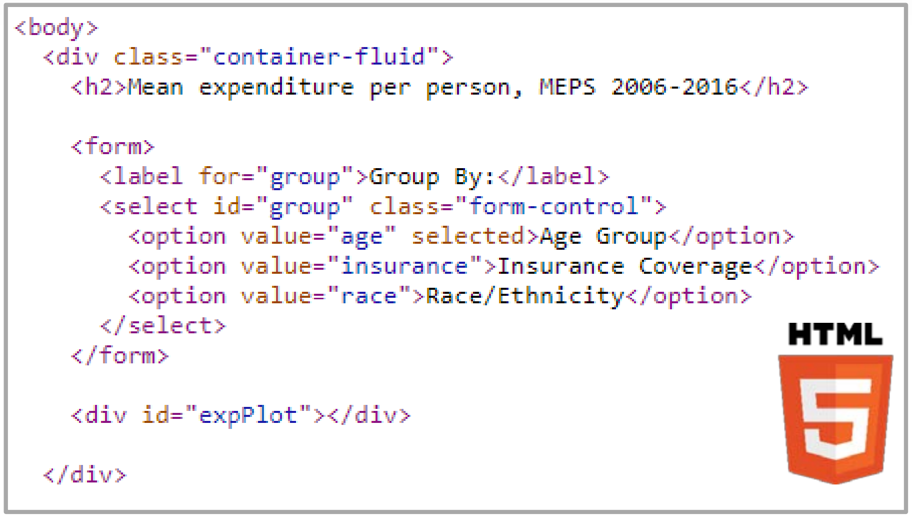
Anatomy of a Shiny app

R – Shiny app



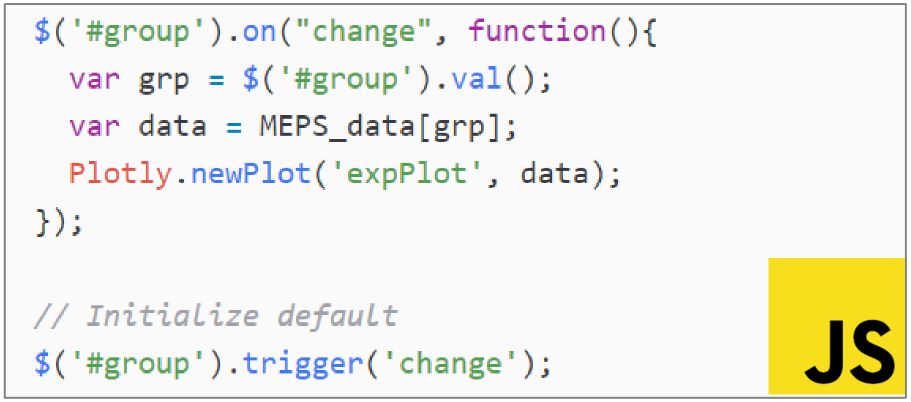
```r
ui <- fluidPage(

  titlePanel("Mean expenditure per person,
             MEPS 2006-2016"),

  selectInput(
    inputId = "group",
    label   = "Group By:",
    choices =
      c("Age Group"            = "age",
        "Insurance Coverage" = "insurance",
        "Race/Ethnicity"      = "race")),

  plotlyOutput(outputId = "expPlot")
)


server <- function(input, output) {

  output$expPlot <- renderPlotly({

    grp <- input$group
    data <- MEPS_data[[grp]]

    data %>%
      group_by(group) %>%
      plot_ly(x = ~Year,
              y = ~Expenditure,
              type = "scatter",
              mode = "lines+markers",
              color = ~group)
  })
}
```

HTML



```html
<body>
  <div class="container-fluid">
    <h2>Mean expenditure per person, MEPS 2006-2016</h2>

    <form>
      <label for="group">Group By:</label>
      <select id="group" class="form-control">
        <option value="age" selected>Age Group</option>
        <option value="insurance">Insurance Coverage</option>
        <option value="race">Race/Ethnicity</option>
      </select>
    </form>

    <div id="expPlot"></div>

  </div>
```

JavaScript (JS)



```javascript
$('#group').on("change", function(){
  var grp = $('#group').val();
  var data = MEPS_data[grp];
  Plotly.newPlot('expPlot', data);
});

// Initialize default
$('#group').trigger('change');
```

Tips for transitioning

1. Start with UI => HTML

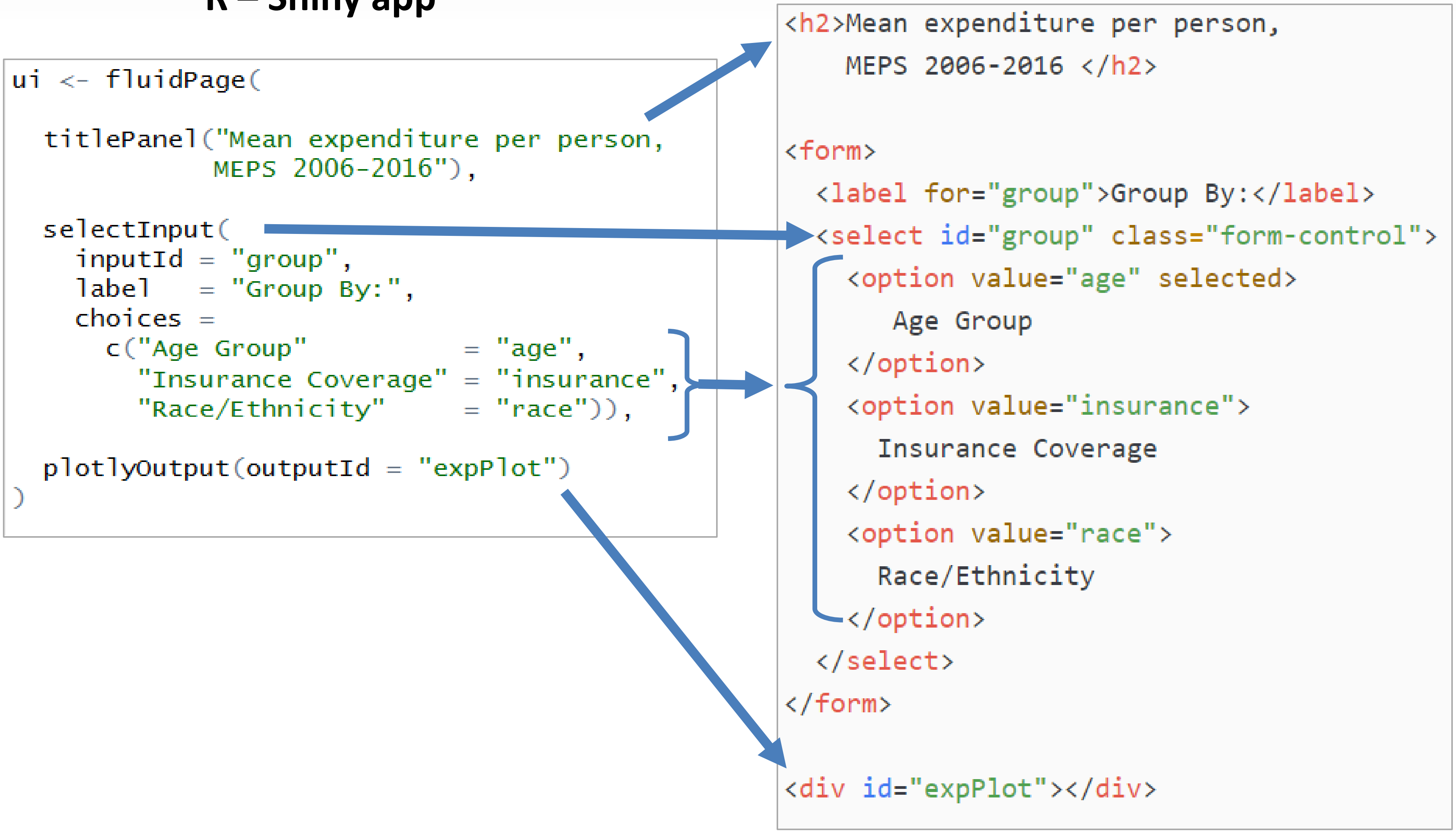
2. Add JavaScript for reactivity

3. Add CSS (optional)

1. Start with UI => HTML

R – Shiny app

HTML

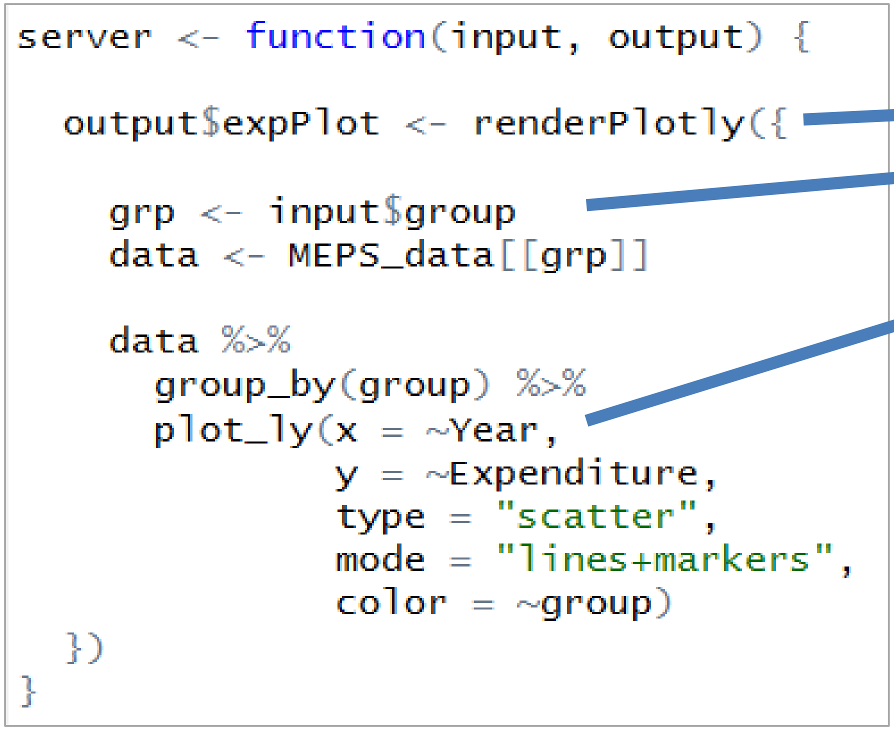


```r
ui <- fluidPage(

  titlePanel("Mean expenditure per person,
             MEPS 2006-2016"),

  selectInput(
    inputId = "group",
    label   = "Group By:",
    choices =
      c("Age Group"          = "age",
        "Insurance Coverage" = "insurance",
        "Race/Ethnicity"     = "race")),

  plotlyOutput(outputId = "expPlot")
)
```

```html
<h2>Mean expenditure per person,
    MEPS 2006-2016 </h2>


<form>
  <label for="group">Group By:</label>
  <select id="group" class="form-control">
    <option value="age" selected>
       Age Group
    </option>
    <option value="insurance">
      Insurance Coverage
    </option>
    <option value="race">
      Race/Ethnicity
    </option>
  </select>
</form>


<div id="expPlot"></div>
```

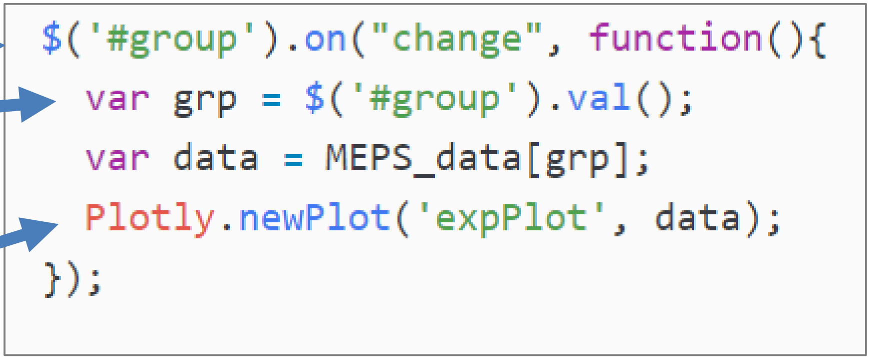
2. Add JavaScript for reactivity

R – Shiny app



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

  output$expPlot <- renderPlotly({

    grp <- input$group
    data <- MEPS_data[[grp]]

    data %>%
      group_by(group) %>%
      plot_ly(x = ~Year,
              y = ~Expenditure,
              type = "scatter",
              mode = "lines+markers",
              color = ~group)
  })
}
```

JavaScript (JQuery)

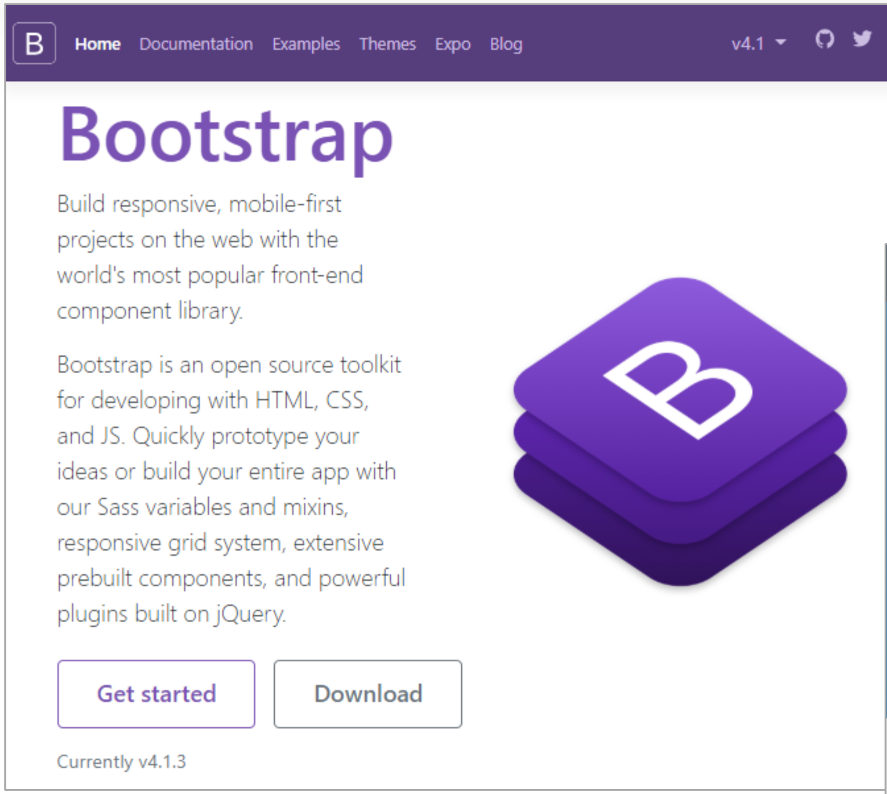


```
$('#group').on("change", function(){
  var grp = $('#group').val();
  var data = MEPS_data[grp];
  Plotly.newPlot('expPlot', data);
});
```

jQuery

write less, do more.

3. Add CSS (optional)

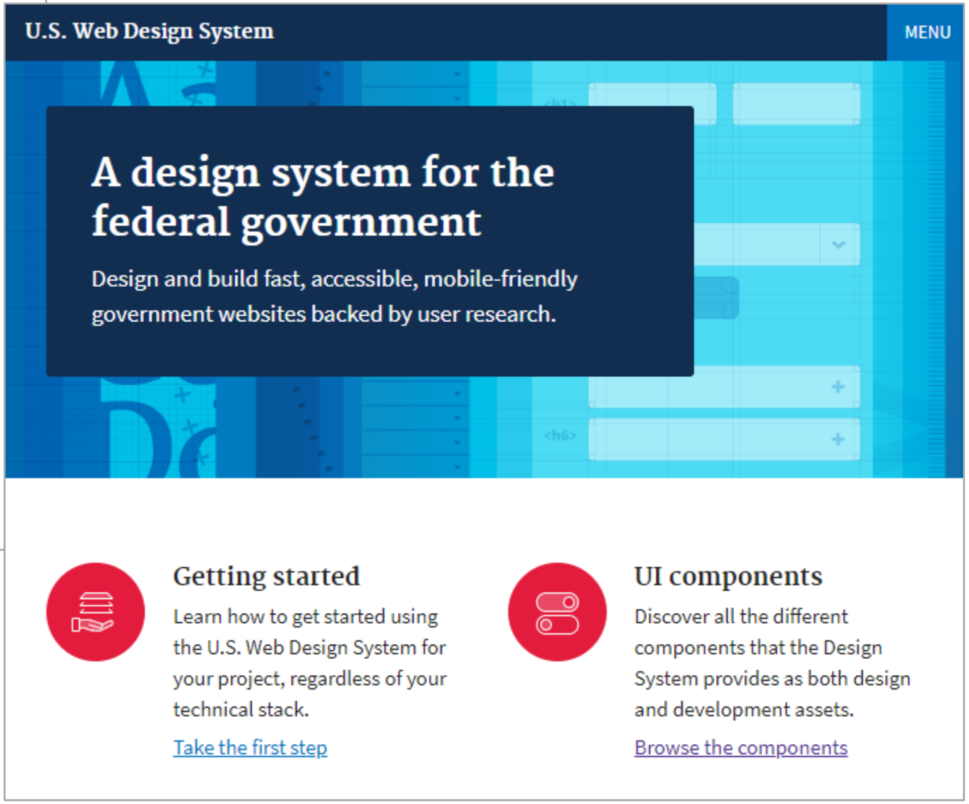


Bootstrap

Build responsive, mobile-first projects on the web with the world's most popular front-end component library.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

Currently v4.1.3



A design system for the federal government

Design and build fast, accessible, mobile-friendly government websites backed by user research.

Getting started

Learn how to get started using the U.S. Web Design System for your project, regardless of your technical stack.

Take the first step

UI components

Discover all the different components that the Design System provides as both design and development assets.

Browse the components

Shiny vs. JavaScript

	Shiny	JavaScript
Statistical programming	🙂	☹️
Data manipulation	🙂	☹️
Learning curve	🙂	😐
Trouble-shooting	😐	😐
Customization	☹️	🙂
Speed		

You may want to transition if:

❑ **You don’t really need R** ★

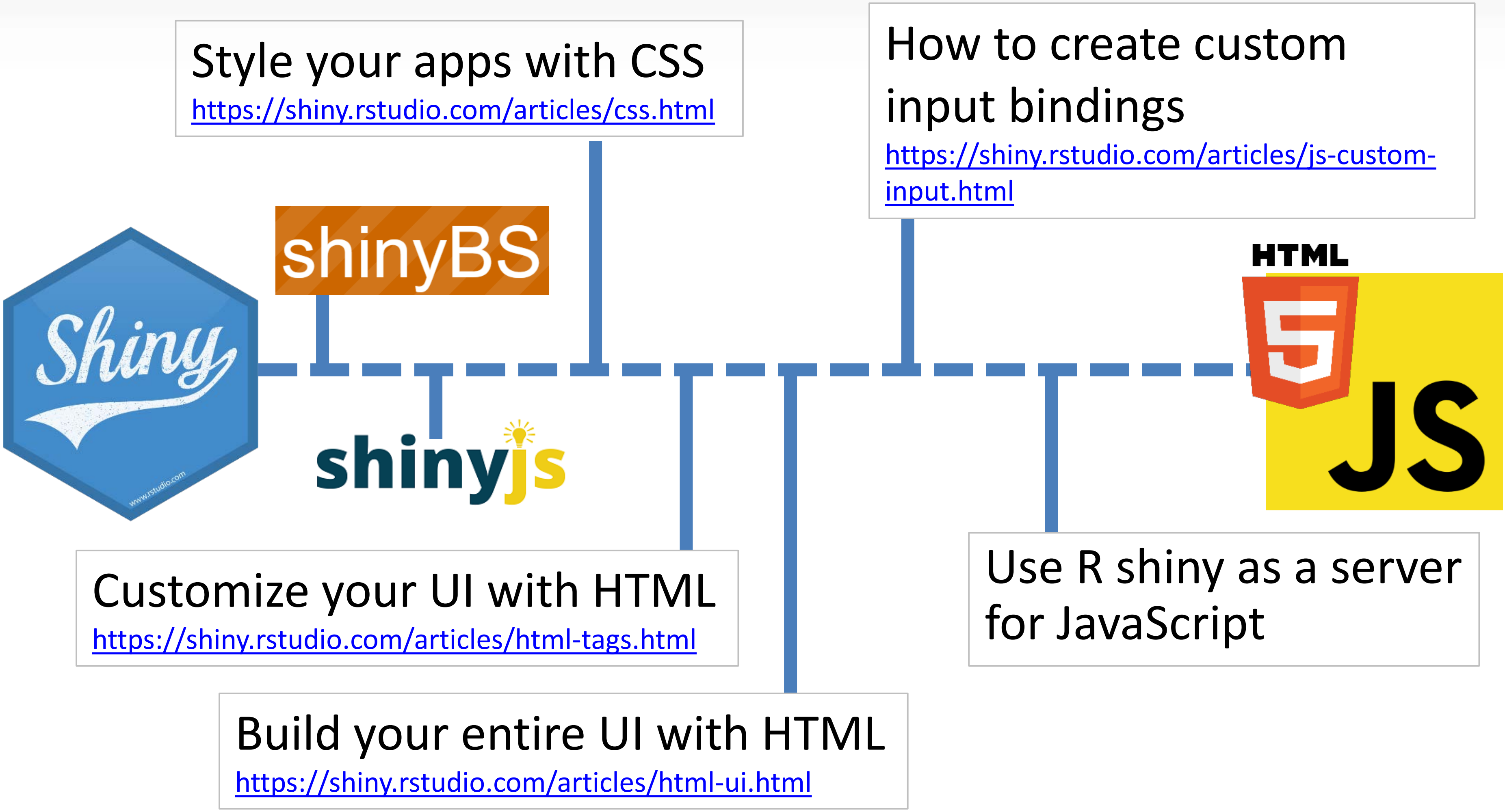
❑ You use Shiny for display, not computation

❑ You write your own R functions to create UI objects

❑ You frequently use *shinyBS* and *shinyJS* packages, and find they don’t have all the features you want

❑ You are writing and embedding custom HTML/JS/CSS into your Shiny app

A happy medium?



Style your apps with CSS
https://shiny.rstudio.com/articles/css.html

Customize your UI with HTML
https://shiny.rstudio.com/articles/html-tags.html

Build your entire UI with HTML
https://shiny.rstudio.com/articles/html-ui.html

How to create custom input bindings
https://shiny.rstudio.com/articles/js-custom-input.html

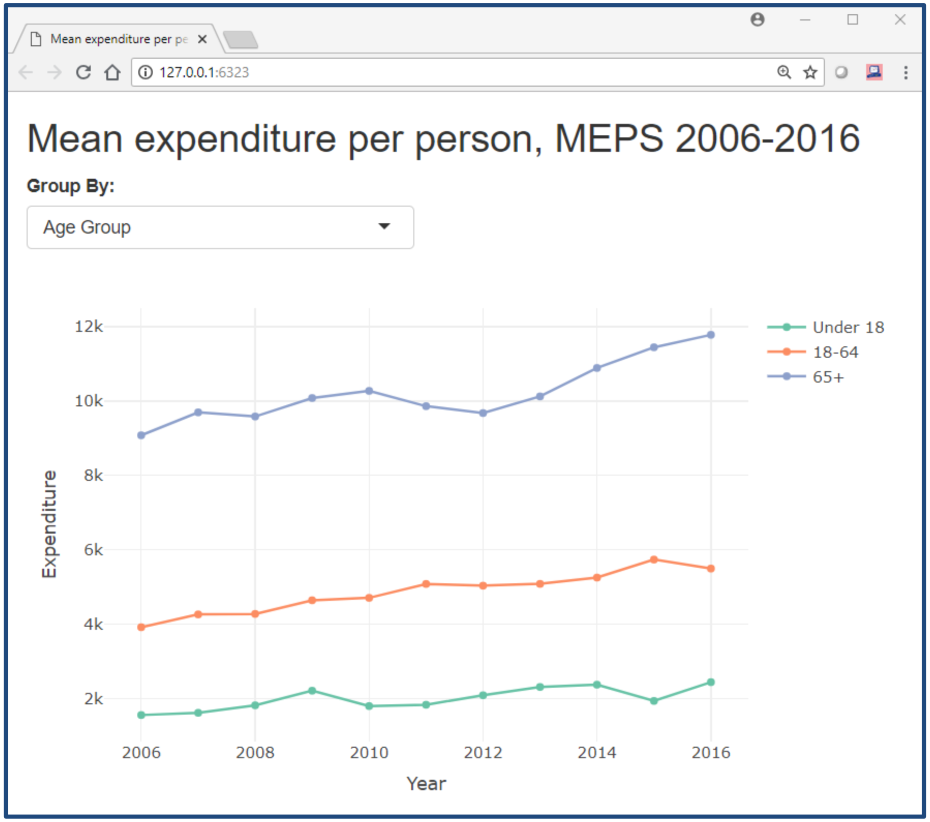
Use R shiny as a server for JavaScript

Image credits: rstudio.com, deanattali.com/shinyjs/, ebailey78.github.io/shinyBS/

Transition example

https://github.com/e-mitchell/Shiny_JS_transition

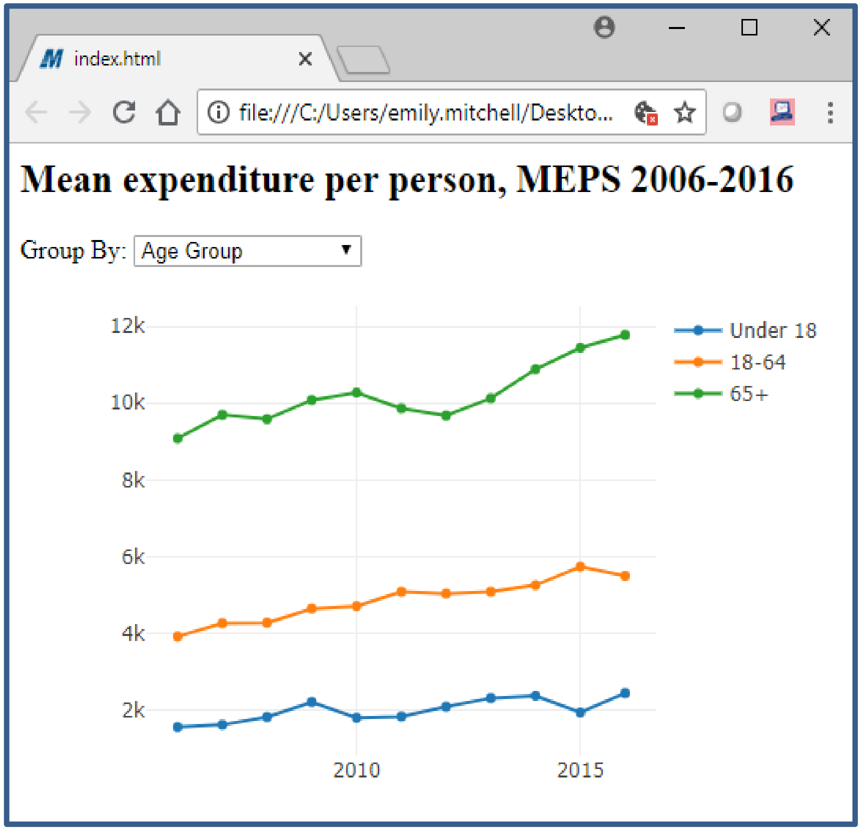
R – Shiny app



Mean expenditure per person, MEPS 2006-2016

Group By: Age Group

HTML / JS

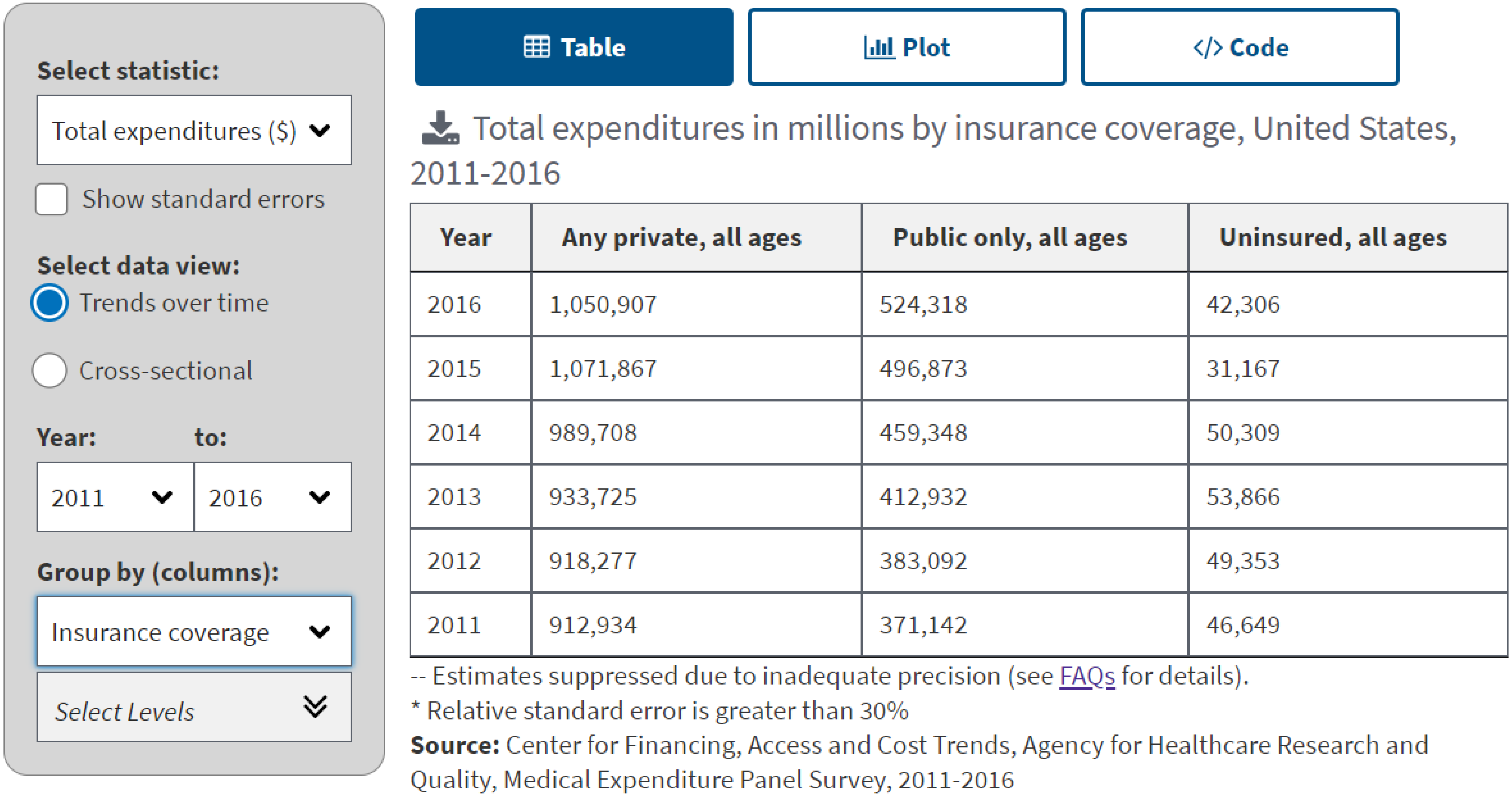


Mean expenditure per person, MEPS 2006-2016

Group By: Age Group

MEPS summary tables

https://github.com/HHS-AHRQ/MEPS-summary-tables



Select statistic: Total expenditures ($)

□ Show standard errors

Select data view:

● Trends over time

○ Cross-sectional

Year: 2011 to: 2016

Group by (columns): Insurance coverage

Select Levels

Total expenditures in millions by insurance coverage, United States, 2011-2016

Year	Any private, all ages	Public only, all ages	Uninsured, all ages
2016	1,050,907	524,318	42,306
2015	1,071,867	496,873	31,167
2014	989,708	459,348	50,309
2013	933,725	412,932	53,866
2012	918,277	383,092	49,353
2011	912,934	371,142	46,649

-- Estimates suppressed due to inadequate precision (see FAQs for details).

* Relative standard error is greater than 30%

Source: Center for Financing, Access and Cost Trends, Agency for Healthcare Research and Quality, Medical Expenditure Panel Survey, 2011-2016

AHRQ Agency for Healthcare Research and Quality

Resources

Shiny tutorials	https://shiny.rstudio.com/articles
Learning HTML / CSS / JS	coursera.org/specializations/web-design w3schools.com
Bootstrap	getbootstrap.com
US Web Design Standards (USWDS)	standards.usa.gov
Web Accessibility (508 compliance)	webaim.org

emily.mitchell@ahrq.hhs.gov