post02-amy-zhu

Amy Zhu

December 3, 2017

Introduction

Today I will be writing a guide to create a shiny app using one of R's built in datasets (mtcars) so everyone has access. In class, we have not had the chance to explore the DataTables application in shiny app. So my post will be focused on building a shiny app with DataTables application. DataTables in shiny app are HTML tables that use the jQuery library DataTables. DataTables are important and useful if users want to post a large dataset without having to upload a csv or excel file. Being able to generate the HTML datatable will allow website visitors to easily get a glimpse at the data and make the experience more seamless.

Preparation

First, make sure to run the below so the data is loaded into the session.

```
data(mtcars)
```

And then you can check what type of dataset you are deadling with by calling the following.

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

```
head(mtcars, 6)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

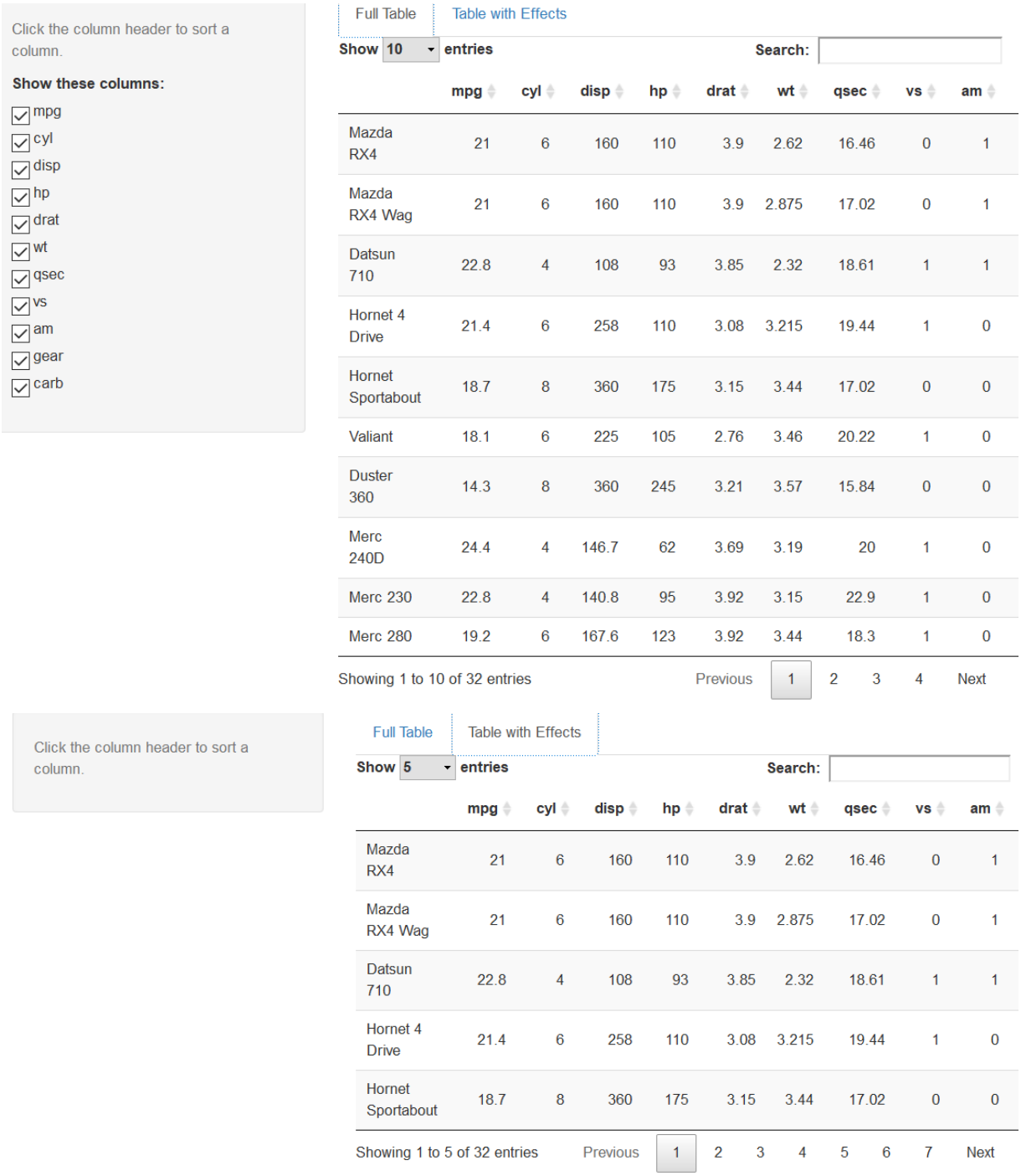
Now that we have loaded and peaked at the data, we have to load the libraries we will be using

```
library(shiny)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.2
```

Building UI

As with the other shiny apps, We first have to build a user interface portion for the application.



We will have 2 different tabs using the same dataset but exploring different options and features given in the DataTables application. The tabs ui can be achieved using conditionalPanel, which we've already used in homework 4. The text you see such as "Click the column header to sort a column" can be displayed with the following

```
helpText("Click the column header to sort a column")
```

Click the column header to sort a column

The checkboxes are widgets from shiny using

```
checkboxGroupInput("show_vars", "Show these columns:",
                   names(mtcars), selected = names(mtcars))
```

Show these columns:

- mpg
- cyl
- disp
- hp
- drat
- wt
- qsec
- vs
- am

- am
- gear
- carb

The rest of the UI are techniques we have learned or used before so I won't go into specifics. Below is the whole UI code

```
ui <- fluidPage(
  title = "DataTables Tutorial",
  sidebarLayout(
    sidebarPanel(
      conditionalPanel(
        'input.data === "Full Table"',
        helpText("Click the column header to sort a column."),
        checkboxGroupInput("show", label = "Show these columns:",
                           names(mtcars), selected = names(mtcars))),
      conditionalPanel(
        'input.data === "Table with Effects"',
        helpText("Click the column header to sort a column.")
      )
    ),
    mainPanel(
      tabsetPanel(
        id = 'data',
        tabPanel("Full Table", DT::dataTableOutput("table1")),
        tabPanel("Table with Effects", DT::dataTableOutput("table2"))
      )
    )
  )
)
```

Server

As suggested by the name of the tabs, the first tab will be simply the datatable of the whole dataset, including all the columns and all the rows along with a checkbox on the left that allows users to choose which columns they want to display. Everything in the table are the defaults, including the search bar and the length menu. These can be hidden in the options function we will talk about later. The rows displayed can be sorted by clicking on the column name by which you want to sort.

The following is the code that will render a table that takes in inputs from the checkbox column on the right and displays the data table accordingly.

```
 output$table1 <- DT::renderDataTable({
    DT::datatable(mtcars[, input$show_vars, drop = FALSE])
  })
```

Next is the second tab where we will specify features of the datatable. All the extra features are within the options function that would come after we indicate the dataset in datatabal constructor.

Table Elements

For example, if we only want to display the search bar or the length and only the table we would use the dom option where t indicates table only.

```
datatable(mtcars, options = list(dom = 't'))
```

Sorting

If we want the table to display a default order using more than one level of sort, we can also use the options function. With the following code, the initial table will first be sorted in ascending order based on column 1 and then in descending order based on column 2.

```
datatable(mtcars, options = list(order = list(list(1, 'asc'), list(2, 'desc'))))
```

We can also choose to color the column by which the table is sorted using the following

```
datatable(mtcars, options = list(orderClasses = TRUE))
```

Rows per Page

By default, datatables display 10 rows of data per page, but if you want to change that, you can use the pageLength option to display say only 5 rows of data per page.

```
datatable(mtcars, options = list(pageLength = 5))
```

If you want to display the whole table without separating the rows into multiple pages then you can use paging like below.

```
datatable(mtcars, options = list(paging = FALSE))
```

Length Menu

There is a length dropdown menu at the top left corner of the table that allows you to choose from a preset list of number of rows to show per

page. You can change that using the following, which indicates users can choose to display 5 rows per page, 20 rows per page, 50 rows per page, or 100 rows per page.

```
datatable(mtcars, options = list(lengthMenu = c(5, 20, 50, 100)))
```

In the end, you can combine whichever options you see fit for your datatable and render the table. Below is the whole app code for the images displayed earlier

```
library(shiny)
library(ggplot2)
library(DT)
```

```
## Warning: package 'DT' was built under R version 3.4.3
```

```
##
## Attaching package: 'DT'
```

```
## The following objects are masked from 'package:shiny':
##
##     dataTableOutput, renderDataTable
```

```
data("mtcars")
mtcars <- mtcars

ui <- fluidPage(
  title = "DataTables Tutorial",
  sidebarLayout(
    sidebarPanel(
      conditionalPanel(
        'input.data === "Full Table"',
        helpText("Click the column header to sort a column."),
        checkboxGroupInput("show", label = "Show these columns:",
                           names(mtcars), selected = names(mtcars))),
      conditionalPanel(
        'input.data === "Table with Effects"',
        helpText("Click the column header to sort a column.")
      )
    ),
    mainPanel(
      tabsetPanel(
        id = 'data',
        tabPanel("Full Table", DT::dataTableOutput("table1")),
        tabPanel("Table with Effects", DT::dataTableOutput("table2"))
      )
    )
  )
)

server <- function(input, output) {

  output$table1 <- DT::renderDataTable({
    DT::datatable(mtcars[, input$show, drop = FALSE])
  })

  output$table2 <- DT::renderDataTable({
    DT::datatable(mtcars, options = list(orderClasses = TRUE,
                                         lengthMenu = c(5, 20, 50, 100),
                                         pageLength = 5))
  })
}

shinyApp(ui, server)
```

Shiny applications not supported in static R Markdown documents

Conclusion

You can render DataTable Applications using the techniques gone over above and specify different features as needed using options.

References

https://rstudio.github.io/DT/ https://shiny.rstudio.com/gallery/widget-gallery.html https://shiny.rstudio.com/gallery/datatables-options.html
https://exploredata.wordpress.com/2014/10/05/embedding-images-in-r-markdown-documents/
https://shiny.rstudio.com/articles/datatables.html https://rstudio.github.io/DT/extensions.html
https://www.rdocumentation.org/packages/DT/versions/0.2/topics/dataTableOutput