**How To Customise ShinyApp With Bootstrap CSS, Javascript And Plotly**

What is the easiest way to make a data science product? My answer is to use [Shiny](https://shiny.rstudio.com/). You can code both front-end and server-side in R to create beautiful interactive web applications. You don’t even need to know any HTML. Shiny server is free, too. You [sign up](https://www.shinyapps.io/) and get it started right away! Check out the [shiny gallery](https://shiny.rstudio.com/gallery/) for inspiration. To learn Shiny, there are heaps of online tutorials like [this one](https://shiny.rstudio.com/tutorial/).

In my opinion, this is the area where the hurdle is higher for Python. Python doesn’t really have a specific framework to create data visualisation apps that run on a web server. Sure, you can use Django or Flask, but you really need to study web development before making anything. Most of us are interested in analysing data and delivering insights, not becoming a full-stack web developer. Shiny really enables us to focus on data analysis and insights creation rather than worrying about the web technologies.

Having said that, Shiny has its own limitations. It is not for creating a complex web application as you are coding the server-side with R. Certainly, this is not for an e-commerce platform. It is most suited for a one page web app with R logic behind it. If you are only coding the front-end with R, you will soon hit the wall in terms of how the page looks.

To customise Shiny App UI, the best approach is to use your own HTML, CSS and Javascripts. In this hack, I will show you how to customise ShinyApp UI with minimal effort. I will also show you how to make plotly work with custom HTML and import your own images, too.

[Plotly API](https://plot.ly/r/) enables you to make your R visualisation interactive. R graphs produced by plot() function, for example, is just an image without interactivity. Plotly uses Javascript to make it interactive in both web browsers and R Studio viewer.

To use plotly, you can simply pass the data and parameters to the ploty\_ly() function. See the difference it makes even with such a simple scatter graph.

[cc lang="matlab" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

library(plotly)

data(mtcars)

# (1) Basic scatter graph with R plot function

plot(mtcars$mpg, mtcars$hp, col='red', pch=16)

# (2) With plotly

plot\_ly(mtcars, x = ~mpg, y = ~hp)

[/cc]

What makes Plotly even more powerful is that you can use it on top of ggplot. Plotly has a function called ggplotly which takes a ggplot function as an argument. This means you can create a complex visualisation with ggplot first and then simply pass the function to ggplotly() for interactivity. It’s the ggplot on steroids. Check out how easily this can be done.

[cc lang="matlab" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

library(ggplot)

plot <- ggplot(data=mtcars, aes(x=mpg, y=hp)) + geom\_point(color='purple')

# (1) Scatter plot with ggplot

plot

# (2) Scatter plot with ggplot & plotly

ggplotly(plot)

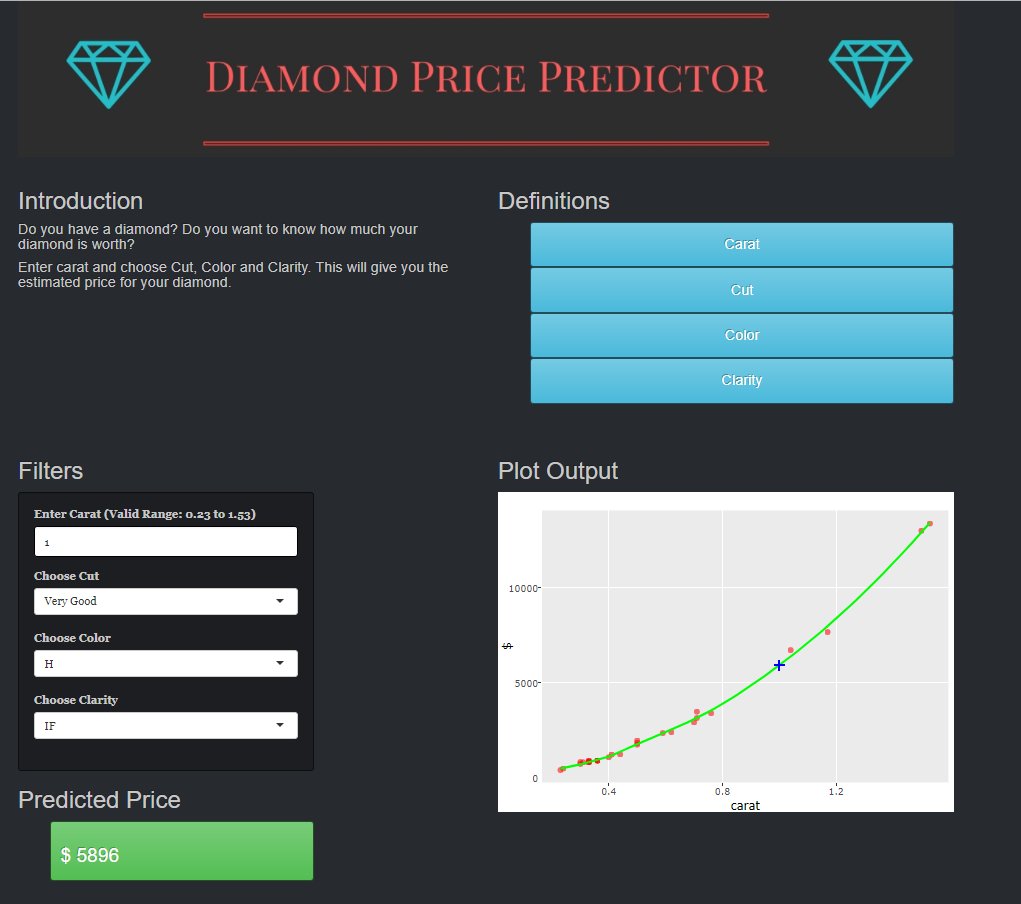
[/cc]

[Diamond Price Predictor](https://portf.shinyapps.io/diamondplotly/)

As an example, I created a [diamond price predictor](https://portf.shinyapps.io/diamondplotly/) that predicts diamond price based on carat, cut, color and clarity by using diamond dataset that comes with base R. You probably need to press ‘Reload’ as the free version of Shiny server goes to sleep if it’s not in use. It also uses plotly for plot interactivity. There are a few tricks to customise your UI with HTML, CSS and make Plotly work on the customised UI.

The summary of the steps are:

* Create the app with ui.R and server.R first.
* Run it in the browser (Run App and open in the browser) from R Studio and get HTML code by viewing the page source from the browser
* Create a new folder structure that allows you to import your own images, CSS and Javascript.
* Add images, CSS and Javascript files in the respective folder.
* Edit the html file and save it as index.html in the app folder.
* Deploy



**Steps**

1. Create the app with ui.R, server.R, and global.R. I recommend to use global.R for package import. It makes package management easy. The basic UI look like [this](https://portf.shinyapps.io/diamonds/).

Ui.R

[cc lang="matlab" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

[/cc]

Server.R

[cc lang="matlab" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

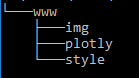
[/cc]

Global.R

[cc lang="matlab" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

[/cc]

1. Run the app and view the source code in the browser and get the source code and save it as index.html.
2. Create a new folder structure. The app folder name becomes your app name when you deploy it to the server. In this case, I have an app folder called diamondploty. Within the main folder, create the www folder. Inside it, create img, plotly and style folder.



1. I used [Canva](https://www.canva.com/) to create the header images. I fond Canva very easy to use for prototyping graphics for web development. I have two header images for dynamic content. When it is viewed on mobile screen, the header image changes (you can try to scale down the browser width to see if the header image changes).

For CSS, I grabed a cumtomised bootstrap theme from [Bootswatch](https://bootswatch.com/). You can choose whichever theme you want to use and download it. Save the file in the style folder. This CSS will replace the default CSS in Shiny.

To make Plotly work with custom HTML, you need to add 3 Javascript files and 1 css file in the plotly folder. If you are using ui.R, you don’t need to import them. But, once you start using custom HTML, these files are required.



You will find these files from the R library folder. Check your library path with .libPath(). You can either manually copy these files or use copy command from cmd or Unix equivalent. Make sure to copy them instead of moving them (mistake I have made so many times)!

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

copy <app folder>\www\plotly <R lib path>\htmlwidgets\www\htmlwidgets.js

copy <app folder>\www\plotly <R lib path>\plotly\htmlwidgets\plotly.js

copy <app folder>\www\plotly <R lib path>\plotly\htmlwidgets\lib\plotly-latest.min.js

copy <app folder>\www\plotly <R lib path>\plotly\htmlwidgets\lib\plotly-htmlwidgets.css

[/cc]

You can remove ui.R and keep both global.R and server.R. In the end, the app folder should look like this.



1. Move the html file that you made in Step 2 into www folder. First of all, you need to import files in the plotly folder in the head section as below. In the same way, you can import your own custom Javascript.

[cc lang="html5" tab\_size="4" lines="-1"]

<script src="plotly/htmlwidgets.js"></script>

<link href="plotly/plotly-htmlwidgets.css" rel="stylesheet" />

<script src="plotly/plotly-latest.min.js"></script>

<script src="plotly/plotly.js"></script>

[/cc]

To import your custom css file, replace the default css bootstrap link element with your own.

[cc lang="html5" tab\_size="4" lines="-1"]

<!-- <link href="shared/bootstrap/css/bootstrap.min.css" rel="stylesheet" /> -->

<link href="style/bootstrap.css" rel="stylesheet" />

[/cc]

In the header, you can also add your own style and java script.

Shiny UI uses bootsrap. The default class for the page div is set to “container-fluid”. This is the reason why you see the app wide. I changed this class to “container” so that I can have the app uses better looking width.

[cc lang="html5" tab\_size="4" lines="-1"]

<div class="container" style="height:100%;">

[/cc]

For header image, I added html to import from the img folder. With using the right CSS class, you can make this dynamic. In my case, switching two images depending on the screen size.

[cc lang="html5" tab\_size="4" lines="-1"]

<div class="container">

<img src="img/diamond.jpg" alt="Diamond Price Predictor Header Img" style="width: 100%;height:100%;" class="header-img-show">

</div>

<div class="container" class="header-txt-show">

<img src="img/diamond\_small.jpg" alt="Diamond Price Predictor Header Img small" style="width: 100%;height:100%;" class="header-txt-show">

</div>

[/cc]

For the rest, I updated HTML as I see fit. For example, I am making buttons more exciting by adding btn class to utilise bootstrap. I also added links for terms definition. I also have my own custom CSS and Javascript in the head section.

1. Once you are happy with the look, you can deploy it. Deployment is the easiest part of Shiny. The rsconnect package will take care of everything. You can get the authentication credentials from the website. You can either run code below or hit Publish on R Studio. You can check [deployment instruction](https://shiny.rstudio.com/articles/shinyapps.html) for further details.

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

install.packages('rsconnect')

library(rsconnect)

rsconnect::setAccountInfo(name=<>, token=<>, secret=<>)

deployApp()

[/cc]

Now you have your own shiny Shiny App.

Epic!