

# Post2

## Simple&Fast Web Data Tool: Shiny

### Intro

Shiny App is a powerful tool to create web app for data engineers & scientists. It makes web visualization and interaction become very easy for data engineers. In the past, I really cannot tell the difference between R and python in their data processing ability. But after experiecing with shiny, I have to say R has a better visualization ability for its customers. I have been a web programmers for quite a long time, and I know how time-consuming it is to make website. If a data scienists has to spend a lot of time on web creation, it's going to lower their data analysis quality. I am a computer science major student and I have been through a lot of difficulties in creating websites before. It's a pain to spend a lot of time to build something that's only served to display some other contents. Therefore, in this Post, I'll make a comparsion between normal web development process and R web process and demonstrate some of Shiny's advantages.

First, let's understand why web visualization is important first. In fact, one of the advantage we have in Shiny App is that it gives us a better visualization than just a R plotted histogram. Consider the ugly histogram below:

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.3.2
```

```
# vector of data types (for each column)
col_types <- c(
  'character', 'character', 'factor', 'character', 'double', 'integer', 'integer', 'integer', 'integer', 'integer',
  'integer', 'integer', 'integer', 'integer', 'integer', 'integer', 'integer', 'integer', 'integer', 'integer', 'integer',
  'integer', 'integer', 'integer'
)

df1 <- read.csv(
  'nba2017-player-statistics.csv',
  header = TRUE,
  colClasses = col_types,
  sep = ",",
)

df2 <- read_csv(
  'nba2017-player-statistics.csv',
  col_types = list(col_character(), col_character(), col_factor(c("C", "PF", "PG", "SF", "SG")), col_character(),
    col_double(), col_integer(), col_integer(), col_integer(), col_integer(), col_integer(), col_integer(), col_integer(),
    col_integer(), col_integer(), col_integer(), col_integer(), col_integer(), col_integer(), col_integer(), col_integer(),
    col_integer(), col_integer(), col_integer(), col_integer())
)

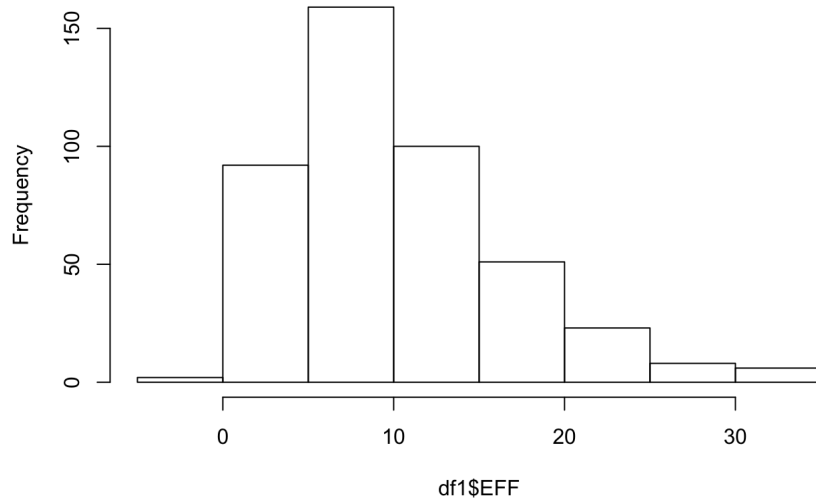
df1 <- data.frame(df1)
df2 <- data.frame(df2)
df1$Experience <- chartr("R", "0", df1$Experience)
df1<-transform(df1, Experience = as.integer(Experience))
df2$Experience <- chartr("R", "0", df2$Experience)
df2<-transform(df1, Experience = as.integer(Experience))

df1$Missed_FG <- df1$FGA - df1$FGM
df1$Missed_FT <- df1$FTA - df1$FTM
df1$PTS <-df1$Points3*3 + df1$Points2*2 + df1$FTM
df1$REB <-df1$OREB + df1$DREB
df1$MPG <- df1$MIN / df1$GP
df1$EFF <- (df1$PTS + df1$REB + df1$AST + df1$STL + df1$BLK - df1$Missed_FG - df1$Missed_FT - df1$TO) / df1$GP
summary(df1$EFF)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.600   5.452   9.090  10.140  13.250  33.840
```

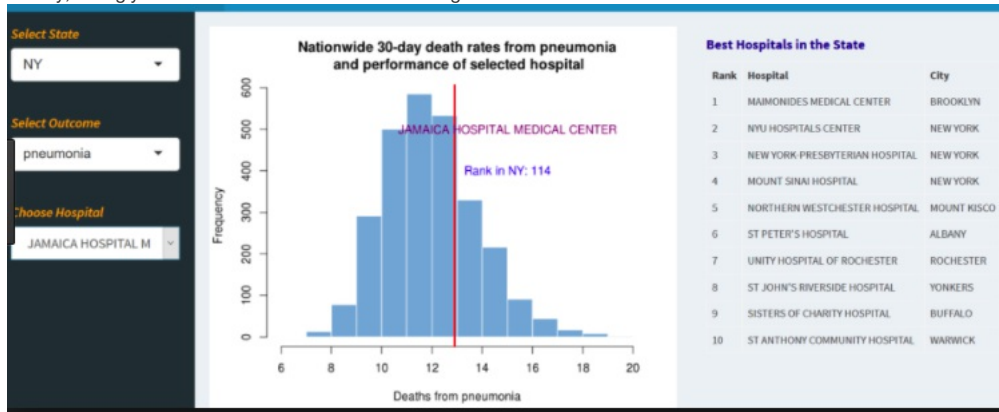
```
print (hist(df1$EFF), main="Histogram of EFF", xlab="EFF")
```

Histogram of df1\$EFF



```
## $breaks
## [1] -5  0  5 10 15 20 25 30 35
##
## $counts
## [1]  2  92 159 100  51  23  8  6
##
## $density
## [1] 0.0009070295 0.0417233560 0.0721088435 0.0453514739 0.0231292517
## [6] 0.0104308390 0.0036281179 0.0027210884
##
## $mids
## [1] -2.5  2.5  7.5 12.5 17.5 22.5 27.5 32.5
##
## $xname
## [1] "df1$EFF"
##
## $sequidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

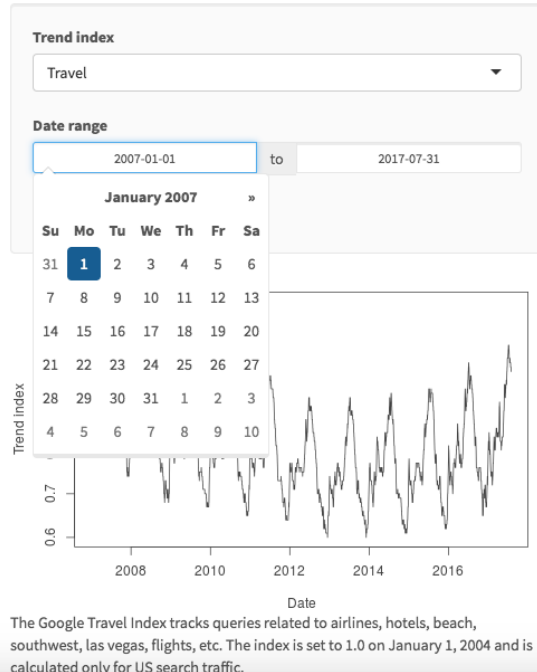
Clearly, it's ugly. What about an interactive web histogram?



Much better right?!

This is how a data analytics would normally create an interactive website using shiny package in R. This demo is from <https://shiny.rstudio.com/>. It has a nice option control as well as date range input fields. Also, the chart is very clear on data's relationship.

## Google Trend Index



Description

app.R

```
# Load packages
library(shiny)
library(shinythemes)
library(dplyr)
library(readr)

# Load data
trend_data <- read_csv("data/trend_data.csv")
trend_description <- read_csv("data/trend_description.csv")

# Define UI
ui <- fluidPage(theme = shinytheme("lumen"),
  titlePanel("Google Trend Index"),
  sidebarLayout(
    sidebarPanel(
      # Select type of trend to plot
      selectInput(inputId = "type", label = strong("Trend index"),
        choices = unique(trend_data$type),
        selected = "Travel"),

      # Select date range to be plotted
      dateRangeInput("date", strong("Date range"), start = "2007-01-01", end = "2017-07-31",
        min = "2007-01-01", max = "2017-07-31"),

      # Select whether to overlay smooth trend line
      checkboxInput(inputId = "smoother", label = strong("Overlay smooth trend")),

      # Display only if the smoother is checked
      conditionalPanel(condition = "input.smoother == true",
        sliderInput(inputId = "F", label = "Smoother span:",
          min = 0.01, max = 1, value = 0.67, step = 0.01,
          animate = animationOptions(interval = 100)),
        HTML("Higher values give more smoothness."))
    ),
    # Output: Description, lineplot, and reference
    mainPanel(
      plotOutput(outputId = "lineplot", height = "300px"),
      textOutput(outputId = "desc"),
      tags$a(href = "https://www.google.com/finance/domestic_trends", "Source")
    )
  )
)
```

On the other hand, If a person tries to create a normal interactive website, he first needs to learn how to write Javascript code. As shown below, writing website codes could be very time consuming. Below is a snapshot of javascript code that merely creates a dropdown list, let alone import data. You can find js tutorial on <https://www.w3schools.com/js/default.asp>.

Run »
Result Size: 579 x 59

```
<h2>Make a table based on the value of a drop down menu.</h2>

<select id="myselect" onchange="change_myselect(this.value)">
<option value="">Choose an option:</option>
<option value="customers">Customers</option>
<option value="products">Products</option>
<option value="suppliers">Suppliers</option>
</select>

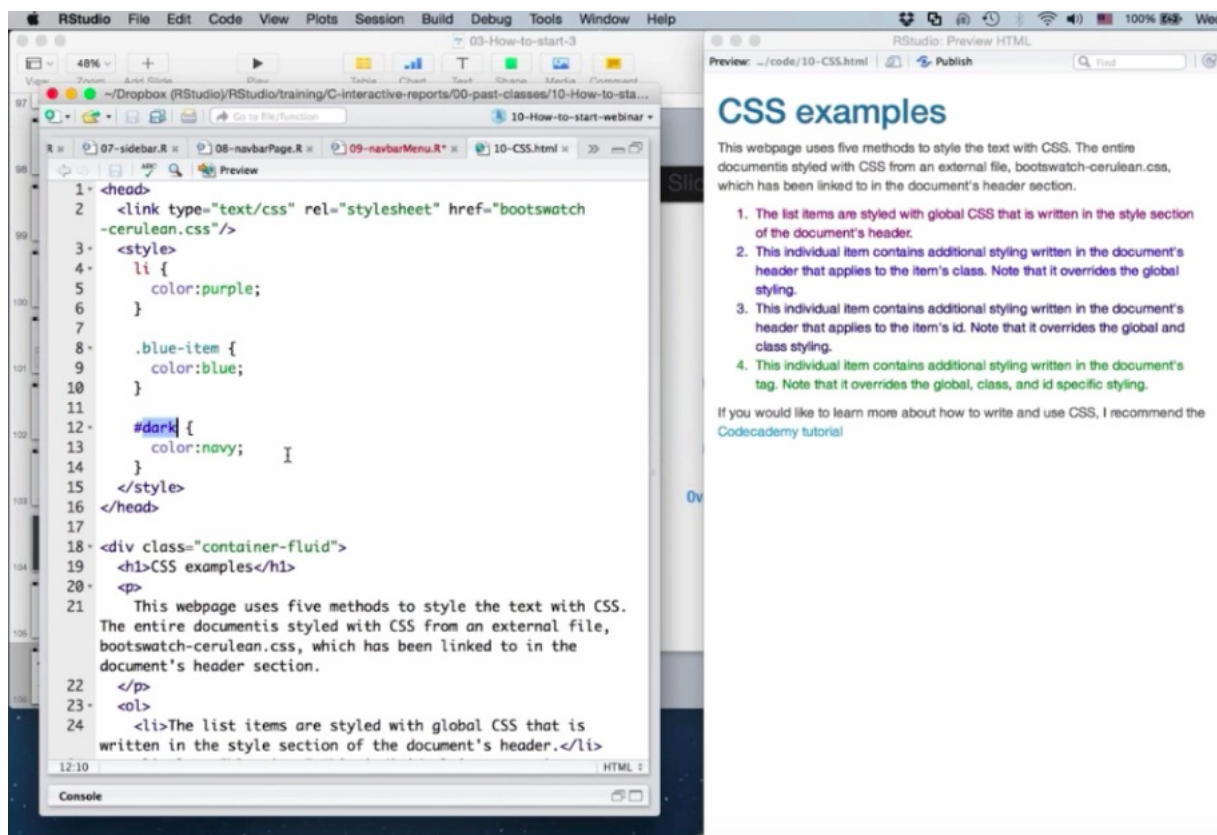
<p id="demo"></p>

<script>
function change_myselect(sel) {
var obj, dbParam, xmlhttp, myObj, x, txt = "";
obj = { "table":sel, "limit":20 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
myObj = JSON.parse(this.responseText);
txt += "<table border='1'>"
for (x in myObj) {
txt += "<tr><td>" + myObj[x].name + "</td></tr>";
}
txt += "</table>"
document.getElementById("demo").innerHTML = txt;
}
};
xmlhttp.open("POST", "json_demo_db_post.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
}
</script>
```

### Make a table based on the value of a drop down menu.

Choose an option: ▼

One may raise a question that if shiny app specifies everything, isn't the format very similar between each other? Well, the main concern for data analytics is to showcase their data analysis not the beauty layout. In addition, it's actually possible for data engineers to modify the css property, according to the video tutorial at 2:15:55 in <https://vimeo.com/rstudioinc/review/131218530/212d8a5a7a/#t=1h11m36s>.



what's more, Shiny app allows you to easily share your data visualization as well as your data analysis work to your colleagues. Because they're all integrated in the same R framework, all the files are R files. For example, this pic is from my github repository, and people who want to have a demo on their local computer can easily do it by clone the repo.

app	first commit
code	first commit
data	first commit
output	first commit
.DS_Store	first commit
README.md	Update README.md

title	author	date	output
README.md	Lai Wei	11/26/2017	github_document

Above is a basic overview of the architecture of shiny app and its advantages. Here I will give a more specific layout of its design. A shiny app is typically consist of 3 parts.

```

{r}
library(shiny)

# Define UI ----
ui <- fluidPage(

)

# Define server logic ----
server <- function(input, output) {

}

# Run the app ----
shinyApp(ui = ui, server = server)

```

And it's very user-friendly. In fact, you can find many templates and inspirations from shiny gallery. By filling in some data path, you can easily demonstrate your work using the pre-built templates.

## Dynamic user interface

These examples show how to create a user interface that changes dynamically.



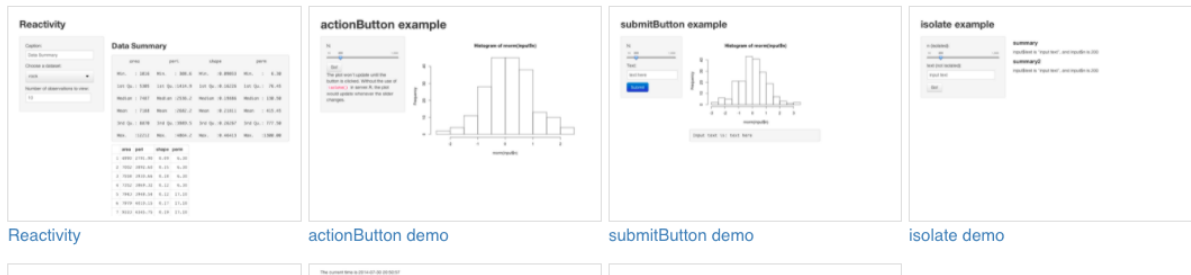
conditionalPanel demo

Dynamic UI

Update input demo

## Reactive programming

These examples illustrate some useful features and idioms of Shiny's reactive programming framework.



Reactivity

actionButton demo

submitButton demo

isolate demo

However, Shiny App is not 100% all good. For example, it's actually a non-free service. It charges certain amount of fees if you want to actually deploy it on websites and have some meaningful functionalities.

FREE	STARTER	BASIC	STANDARD	PROFESSIONAL
<b>\$0</b> /month	<b>\$9</b> /month ( or \$100/year )	<b>\$39</b> /month ( or \$440/year )	<b>\$99</b> /month ( or \$1,100/year )	<b>\$299</b> /month ( or \$3,300/year )
New to Shiny? Deploy your applications for FREE.	More applications. More active hours!	Take your users to the next level!	Password protection? Authenticate your users!	Professional has it all! Personalize your domains.
5 Applications	25 Applications	Unlimited Applications	Unlimited Applications	Unlimited Applications
25 Active Hours	100 Active Hours	500 Active Hours	2,000 Active Hours	10,000 Active Hours
Community Support	Premium Support	Performance Boost	Authentication	Authentication
RStudio Branding		Premium Support	Performance Boost	Account Sharing
			Premium Support	Performance Boost
				Custom Domains
				Premium Support

So can we run a shiny app on our Apache server? The answer seems to be no according to the answer found on stackoverflow.

[Questions](#)
[Developer Jobs](#)
[Tags](#)
[Users](#)

[active](#)
[oldest](#)
[votes](#)

▲

3

▼

✓

So you don't run a Shiny app on Apache, Shiny Apps must be run on a Shiny Server. You have several options for deploying a Shiny app to your website. The simplest approach is to publish your app to shinyapps.io then add an IFrame to your website that will display your deployed Shiny app.

Since you have asked about how to run a Shiny App on Apache, I assume you want to deploy your own Shiny Server, which can be deployed on the same machine as your Apache server or on a different machine. Instructions for doing this may be found here:  
<https://www.rstudio.com/products/shiny/download-server/>

Once you shiny server is installed you will need to publish your app there. This is accomplished by moving your r files to the appropriate location on your Shiny Server. For details see:  
<http://docs.rstudio.com/shiny-server/#host-a-directory-of-applications-1>

Once you have done this and have your app running on Shiny Server, you need to set up Apache(Nginx or another webserver) to act as a proxy to Shiny Server. Here is a link that walks you through the steps: <https://support.rstudio.com/hc/en-us/articles/213733868-Running-Shiny-Server-with-a-Proxy>

If you have not done this before and do not have experience with Linux or servers, I would expect it to take at least 4 hours.

However, it's not acceptable, here is the price for heroku, which is used to deploy a lot of websites for many developers.

PROFESSIONAL	
<p><b>Free</b></p> <p>Ideal for experimenting with cloud applications in a limited sandbox.</p> <p>CORE PLATFORM FEATURES</p> <p>SLEEPS AFTER 30 MINS OF INACTIVITY</p> <p>USES AN ACCOUNT-BASED POOL OF FREE DYNOS HOURS</p> <p>CUSTOM DOMAINS</p> <p>512 MB RAM   1 web/1 worker</p> <p><b>Free</b></p>	<p><b>Hobby</b></p> <p>Perfect for small scale personal projects and hobby apps.</p> <p>CORE PLATFORM FEATURES</p> <p>NEVER SLEEPS</p> <p>FREE SSL &amp; AUTOMATED CERTIFICATE MANAGEMENT FOR CUSTOM DOMAINS</p> <p>APPLICATION METRICS</p> <p>MULTIPLE WORKERS FOR MORE POWERFUL APPS</p> <p>512 MB RAM   10 Process Types</p> <p><b>\$7</b> per dyno/month <small>prorated to the second</small></p>
<p><b>Standard</b> 1X 2X</p> <p>Enhanced visibility, performance, and availability for powering your professional applications.</p> <p>ALL HOBBY FEATURES +</p> <p>SIMPLE HORIZONTAL SCALABILITY</p> <p>THRESHOLD ALERTS</p> <p>PREBOOT</p> <p>LANGUAGE RUNTIME METRICS</p> <p>512MB OR 1GB RAM</p> <p>∞ Process Types</p> <p><b>\$25 - \$500</b> per dyno/month <small>prorated to the second</small></p>	<p><b>Performance</b> M L</p> <p>Superior performance when it's most critical for your super scale, high traffic apps.</p> <p>ALL STANDARD FEATURES +</p> <p>MIX WITH STANDARD 1X, 2X DYNOS</p> <p>DEDICATED</p> <p>AUTOSCALING</p> <p>2.5GB OR 14GB RAM</p> <p>∞ Process Types</p> <p><b>\$25 - \$500</b> per dyno/month <small>prorated to the second</small></p>

So it seems like shiny's pricing isn't that bad.

While shiny has less control for programmers compared to normal javascript website, it is still a very convenient tool for data engineers or data researchers to show their work.

Reference: <https://shiny.rstudio.com/>

<https://www.shinyapps.io/>

<https://vimeo.com/rstudioinc/review/131218530/212d8a5a7a/#t=1h11m36s>

[https://www.heroku.com/pricing?c=70130000001xDpdAAE&gclid=EAlalQobChMIwOvSmprt1wIVklp-Ch09tQ14EAAYASABegJOyPD\\_BwE](https://www.heroku.com/pricing?c=70130000001xDpdAAE&gclid=EAlalQobChMIwOvSmprt1wIVklp-Ch09tQ14EAAYASABegJOyPD_BwE)

<https://www.w3schools.com/js/default.asp>

<https://shiny.rstudio.com/gallery/>

<https://stackoverflow.com/questions/43527041/run-r-shiny-app-on-apache-server>