Calling R functions from JS charts

The OpenCPU libraries allow a user to turn their R instance into a server that is accessible from Javascript. Yellowfin data can then be passed to an R function as a JSON object, and returned as either a JSON object or as an R plot into an SVG container.

Getting this working requires a bit of setup, but once its going, adding or editing functions can be done very quickly, and the same JS template can be used repeatedly with little adjustment.

1. Install and start the OpenCPU server

These instructions are for a single-user local server. In an ideal setup, this would be hosted so that the functions could be accessed remotely. One important note, is that the single-user server does not handle concurrency (so only one chart can load at a time), while the server version does. These instructions are for R v3.3.3.

Within your R Console:

```
install.packages("devtools")
install.packages("opencpu")
library(opencpu)     #Will automatically start the server on a random port, this is easily
customized in the server edition, but in single user, we have to stop it then start again on a
specified port
opencpu$stop()
opencpu$start(8585)
```

Complete guide on the various installations: https://www.opencpu.org/download.html

2. Create a package for your R function

There are multiple ways to do this, but this is the simplest. First:

```
Install RTools: https://cran.r-project.org/bin/windows/Rtools/
```

Then, within your R Console:

```
devtools::install_github("klutometis/roxygen") #Install needed packages
install.packages("assertthat")
install.packages("crayon")
install.packages("rprojroot")
library(roxygen2)
create("yfR")  #Create a package. A new folder appear under this name in your working directory
```

3. Create your script and save to project folder

In this example I simply accept 4 data arrays, as well as an array of column names. I then create a pairs plot colored by the results of a simple kmeans model. (For advanced visualizations use a library such as ggplots: https:/www.r-bloggers.com/multiple-regression-lines-in-ggpairs/)

Once you have created a function, save it as an R script (*myscript.R*), and place it into the "R" directory of the package folder created in the previous step. (*yourdirectory/yfR/R/*)

Below is code for the function *"kmeanspairs"*.

```
kmeanspairs <- function(x1,x2,x3,x4,colz){  #define function

    a1=(x1-min(x1))/(max(x1)-min(x1))  #normalize the data
    a2=(x2-min(x2))/(max(x2)-min(x2))
    a3=(x3-min(x3))/(max(x3)-min(x3))
    a4=(x4-min(x4))/(max(x4)-min(x4))
    df=data.frame(a1,a2,a3,a4) #throw into a dataframe
    colnames(df)<-colz  #add colnames to dataframe
    model=kmeans(df,3)  #run k=3 kmeans model
    cols <- character(nrow(df)) #create an index list of colors
    cols[] <- "black"
    cols[model$cluster == 1] <- "blue"  #color by cluster assignment
    cols[model$cluster == 2] <- "red"
    cols[model$cluster == 3] <- "green"
    pairs(df, col=cols)   #call pairs plot

}
```

4. Install and load the package

Within your R Console:

```
setwd("./yfR")
document()  #for these purposes we will just call this without providing documentation
setwd("..")
install("yfR")
library("yfR")
```

The tedious part is over! From this point on, you will only need to use the last two lines (re-install and load) after making any additions or changes to your R package. You can add as many functions as you like to the package that can all be called by name from JS.

5. Create a JS chart in Yellowfin

```
generateChart = function(options) {

    require(['http://cdn.opencpu.org/opencpu-0.4.js'], function(aarr) {  //As JQuery is already
included, we only need the opencpu library
    ocpu.seturl("http://localhost:8585/ocpu/library/yfR/R");  //point opencpu to your package's R
directory
    debugger;
    height=options.dataset.chart_information.height;
    width=options.dataset.chart_information.width;
    datax1=[];
    datax2=[];
    datax3=[];
    datax4=[];
    for (i=0;i<options.dataset.data.invoiced_amount.length;i++){
    datax1.push(Number(options.dataset.data.invoiced_amount[i].raw_data));
    datax2.push(Number(options.dataset.data.athlete_latitude[i].raw_data));
    datax3.push(Number(options.dataset.data.athlete_longitude[i].raw_data));
    datax4.push(Number(options.dataset.data.age_at_camp[i].raw_data));
    }
    cols=["Inv_Amt","Lat","Long","Age"]; //colnames
    var chartDrawDiv =
$(options.divSelector).css({'height':height,'width':width}).rplot("kmeanspairs",{x1:datax1,x2:dat
ax2,x3:datax3,x4:datax4, colz:cols,});   // call the kmeanspairs function and return it as a plot
into the div
   });
};
```