DATA VISUALIZATION (rCharts)

Leslie McIntosh and Connie Zabarovskaya

Today You Will Learn:

- Recap of ggvis and some updates
- How to create an interactive plot in rCharts
  - Explanation
  - Hands-On
  - More Tricks and Debugging
- How to publish an rCharts plot online
  - Explanation
  - Hands-On
- Where to get more resources, tools, etc.

Recap of ggvis and Updates

- summarize ggvis in two words/phrases
- from the RStudio Webinar
  - ggvis will eventually replace ggplot2
  - more development is underway (panel plots, etc.)
  - Linked Brushing

What is rCharts

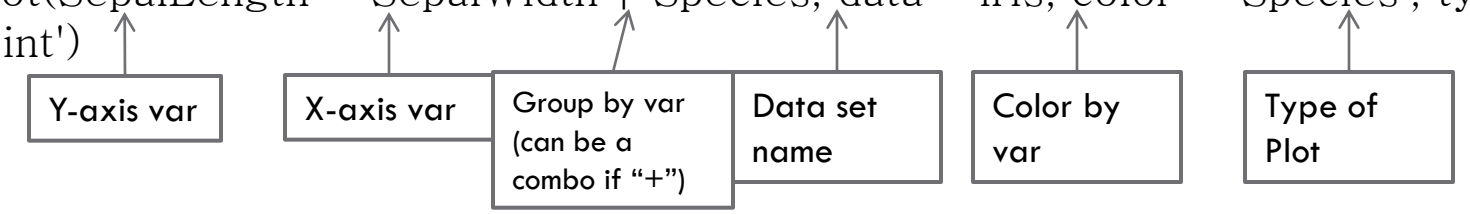
- rCharts is an R package to create, customize and publish interactive JavaScript visualizations from R
- Developed by Ramnath Vaidyanathan, creator of Slidify
- Uses a familiar lattice style plotting interface.
- R code is converted to JavaScript on the back end, see it when you type mychart$print() (can be embedded into website)
- rCharts by itself is interactive to a point, but within a Shiny app it allows input and modification of data in the chart
- Installation:
  - require(devtools)
  - install_github('rCharts', 'ramnathv')

Steps with rCharts

1. Intro to rCharts
2. Some examples with rCharts
3. Create an rCharts plot
4. How to publish it on Gist and RPubs
5. Integrate it with a Shiny app
6. Publish it on a Shiny server/shinyapps.io

rCharts

```
library(rCharts)
names(iris) = gsub("₩₩.", "", names(iris))
rPlot(SepalLength ~ SepalWidth | Species, data = iris, color = 'Species', type =
'point')
```



- Variables above are case sensitive, use the same as in data set.
- rCharts supports multiple javascript charting libraries, each with its own strengths.
- Sometimes you may have to to install new packages to take advantage of vast rCharts functionalities (potential error messages will identify that need)

More Examples with rCharts

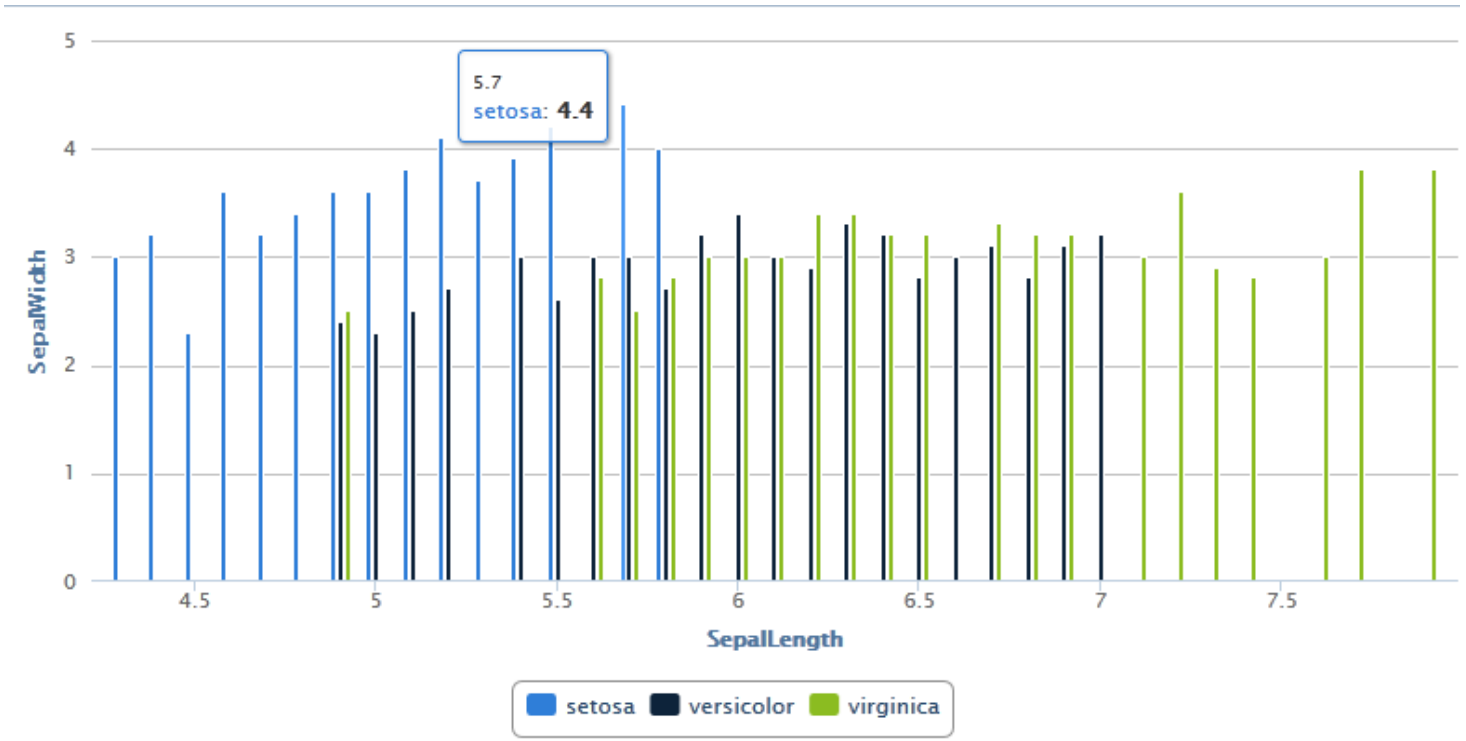
- Popular rCharts libraries:
  - Polychart (rPlot() function for basic, but powerful charts, inspired by ggplot2)
  - Morris (mPlot() function for pretty time-series line graphs)
  - NVD3 (nPlot() function based on d3js library for amazing interactive visualizations with little code and customization)
  - xCharts (xPlot() function for slick looking charts using d3js, made by TenXer)
  - HighCharts (hPlot() function interactive charts, time series graphs and map charts)
  - Leaflet (Leaflet$new() function for mobile-friendly interactive maps)
  - Rickshaw (Rickshaw$new() function for creating interactive time series graphs, developed at Shutterstock)

Example 1

Example below with Highcharts (courtesy of http://ramnathv.github.io/rCharts/)

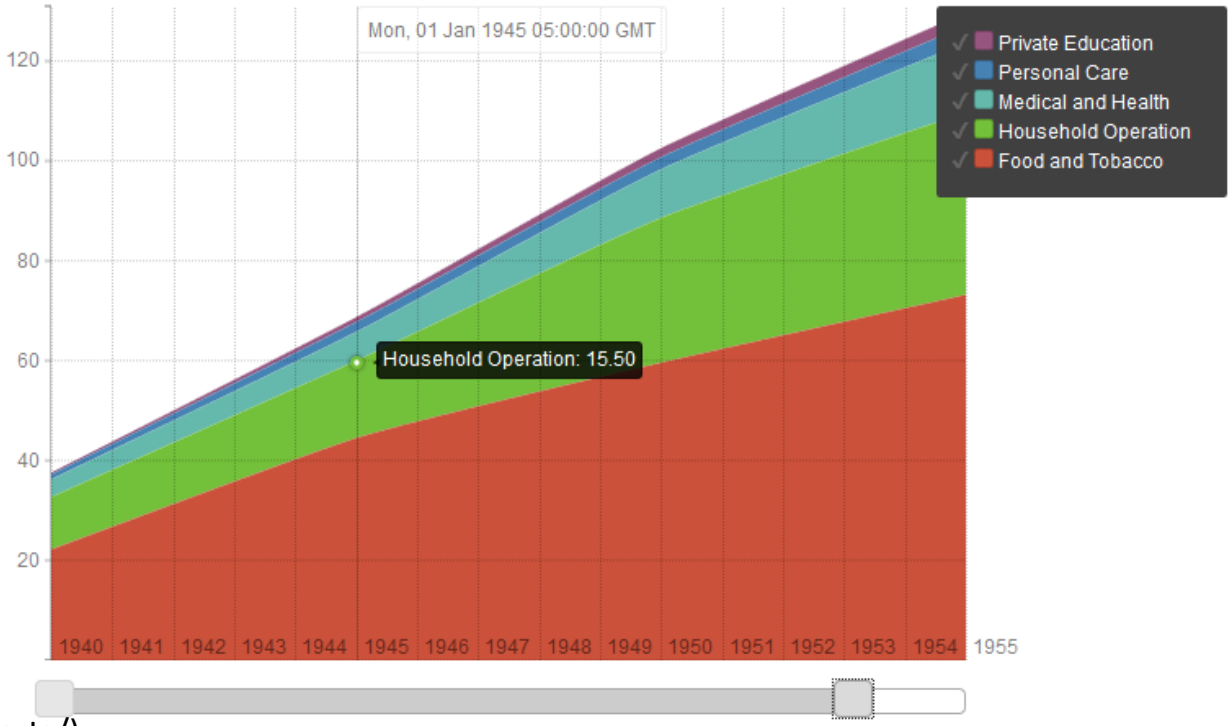
```
names(iris) = gsub("\\.", "", names(iris))
plot1 <- hPlot(x="SepalLength", y="SepalWidth", type="column", group
                ="Species", data=iris)
```

plot1



Example 2

```
usp = reshape2::melt(USPersonalExpenditure)
# get the decades into a date Rickshaw likes
usp$Var2 <- as.numeric(as.POSIXct(paste0(usp$Var2, "-01-01")))
p4 <- Rickshaw$new()
p4$layer(value ~ Var2, group = "Var1", data = usp, type = "area", width = 560)
# add a helpful slider this easily;
# other features TRUE as a default
p4$set(slider = TRUE)
p4
```



(courtesy of http://ramnathv.github.io/rCharts/)

Example 3

#uses the same dataset from ggvis presentation, see http://rpubs.com/conniez/datavis for details

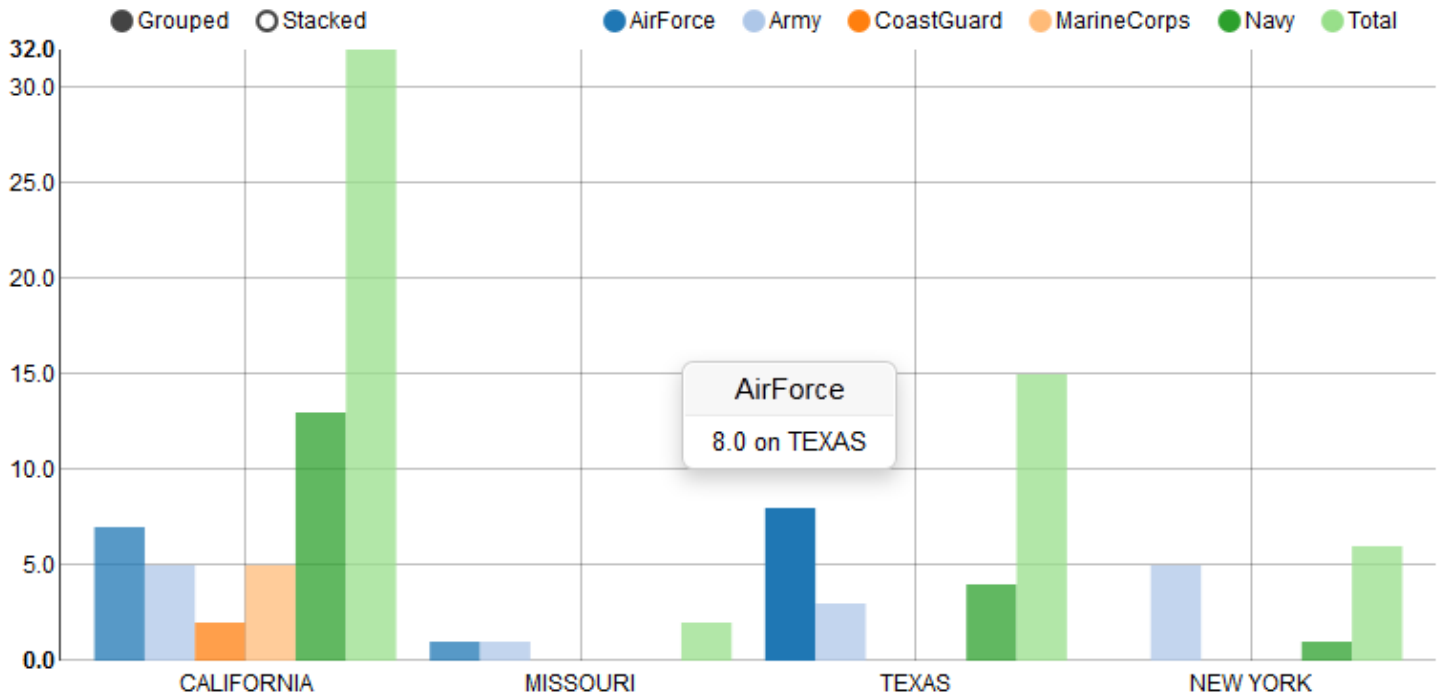
```
library(reshape2); library(rCharts)

ufo <- subset(mergeddata, State == "CALIFORNIA"| State == "TEXAS"| State == "NEW YORK"
                          | State == "MISSOURI")

ufo1 <- melt(ufo, id=c("State", "UFOReports"), na.rm = TRUE)

n1 <- nPlot(value ~ State, group = "variable", data = ufo1, type = "multiBarChart")

n1
```



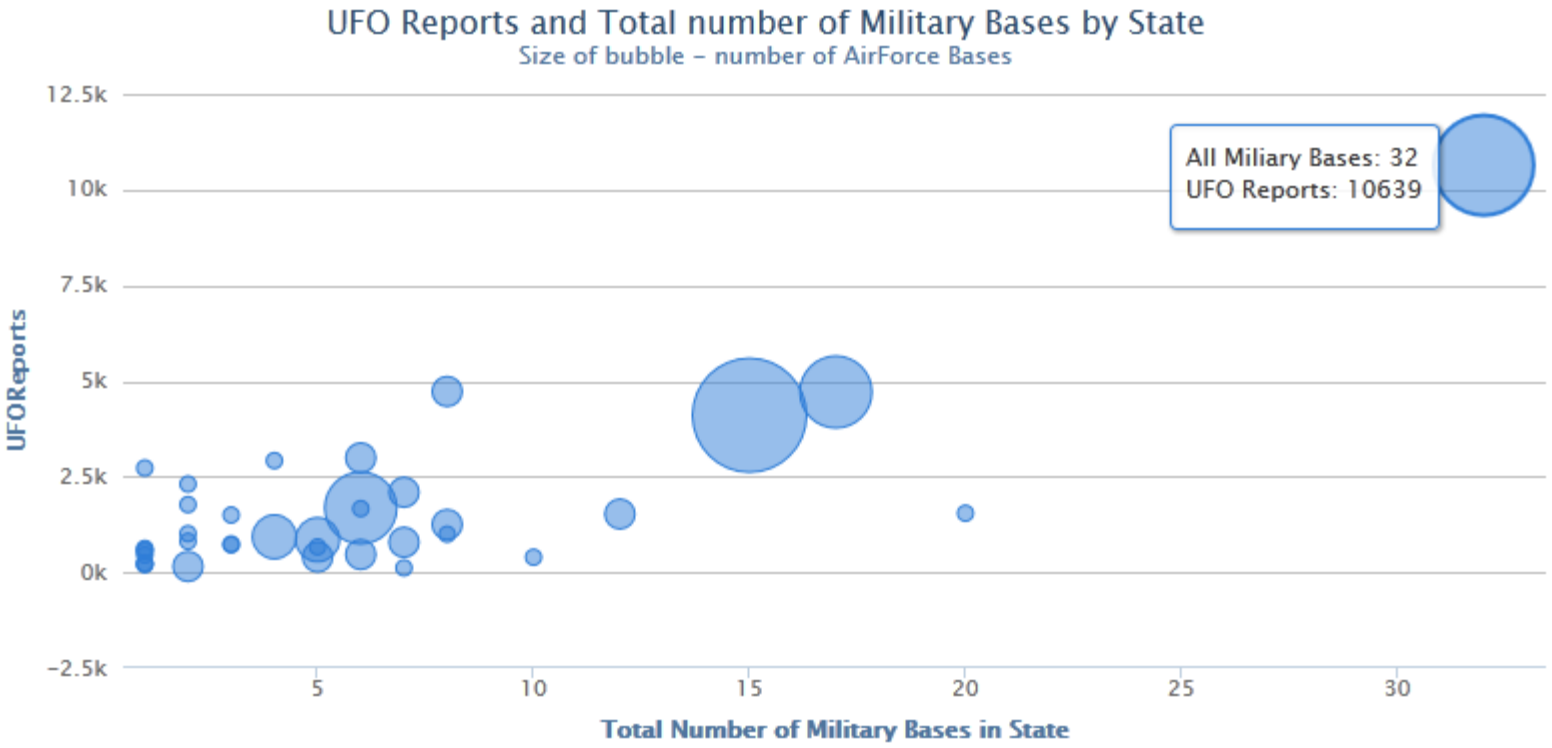
Example 4

- Use same data set as we used during ggvis presentation (see http://rpubs.com/conniez/datavis for details)
- Use hPlot() function to create a Highcharts plot, and then build on top of it

```
ufoplot <- hPlot(x="Total", y = "UFOReports", data = mergeddata, type = "bubble",
                 title = "UFO Reports and Total number of Military Bases by State",
                 subtitle = "Size of bubble - number of AirForce Bases",
                 size = "AirForce")
ufoplot$chart(zoomType = "xy") # will allow to draw a rectangle with mouse to zoom on
                 an area
ufoplot$tooltip( formatter = "#! function() { return 'All Miliary Bases: '     + this.point.x +
                 '<br />' + 'UFO Reports: '    + this.point.y  + '<br />'; } !#")
ufoplot$xAxis(title = list(text = "Total Number of Military Bases in State"))
ufoplot
```

The Result for Example 4

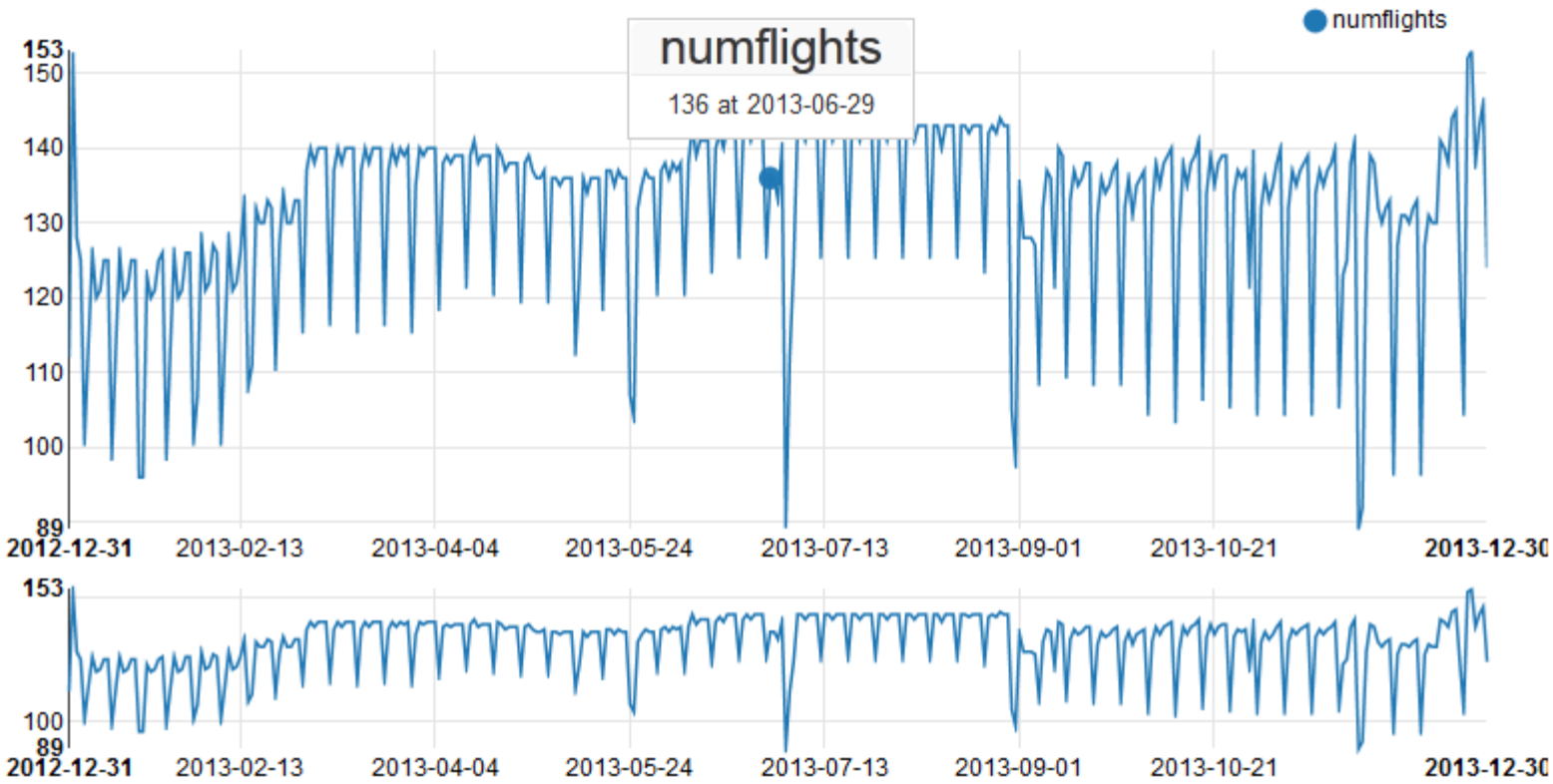
- (picture of the plot)



Example 5

```
library(nycflights13)
#transform the data to prepare for plotting
deltaflights <- subset(flights, carrier == "DL")
deltaflights$date <- as.Date(paste0(deltaflights$month, "/", deltaflights$day, "/",
                                    deltaflights$year),format = "%m/%d/%Y")
deltaflights <- as.data.frame(table(deltaflights$date))
names(deltaflights) <- c("date", "numflights")
#make sure the date column is in Date format
deltaflights$date <- as.Date(deltaflights$date,format = "%Y-%m-%d")
#plot using nvd3 library
deltaPlot <- nPlot(numflights ~ date, data = deltaflights, type = 'lineWithFocusChart')
deltaPlot$xAxis( tickFormat="#!function(d) {return d3.time.format('%Y-%m-%d')(new Date(d * 24
* 60 * 60 * 1000));}!#" )
deltaPlot
```

Result for Example 5



Hands- On Exercise 1 & 2

Exercise 1 (needs package installed)

1. Install package "nycflights13", which contains data about flights departing NYC in 2013
2. Load rCharts (first install it if you haven't yet), load the nycflights13 package and reshape2 package (for transforming data)
3. Transform the data for the plot, the purpose of which is to show average departure delay time (in min) for each airline (carrier), grouped by airport of origin. Feel free to use your preferred method of transformation. You can use aggregate() to summarize, use melt() and dcast() from reshape2 package to make sure there's a line for every combination of airline and airport. And you can use merge() to connect the flights to airlines data sets to get the names of airlines.
4. Using Highcharts library build a bar plot to show the mean departure delay by carrier, group it by origin, and use a relevant title, for example "Average Delay Time by Carrier"
5. Rename the y-axis and x-axis default values to be descriptive
6. Make x-axis type equal "category" (because we have categorical data mapped to x, even though it will be visually seem like y-axis)
7. Change the width of the chart to be 700px and height - 600px

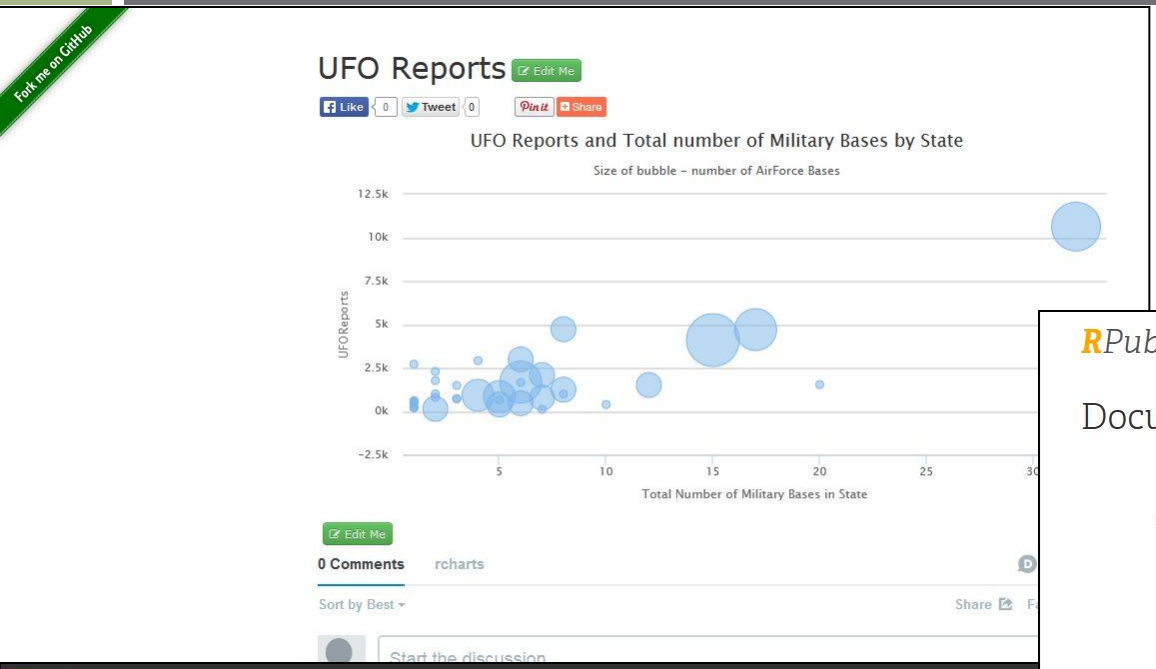
Exercise 2 (uses native dataset)

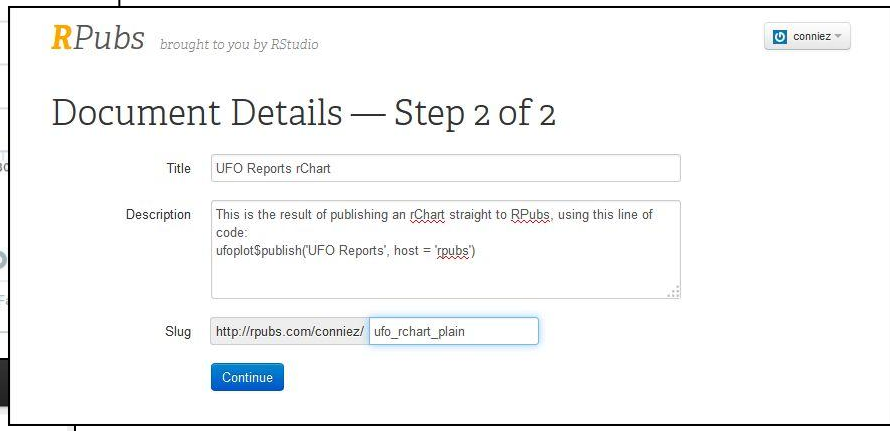
1. Load rCharts (first install it if you haven't yet)
2. Transform the "mtcars" data set for the plot, the purpose of which is to show average mpg grouped by number of cylinders and transmission type (automatic vs. manual).Feel free to use your preferred method of transformation. You can use aggregate() to summarize, use melt() and dcast() from reshape2 package to make sure there's a line for every combination of cylinders and transmission (this might not be necessary).
3. Using Highcharts library build a bar plot to show the mean mpg by number of cylinders, group it by transmission, and use a relevant title, for example "Average MPG by Number of Cylinders"
4. Rename the y-axis and x-axis default values to be descriptive
5. Make x-axis type equal "category" (because we have categorical data mapped to x, even though it will be visually seem like y-axis)
6. Change the width of the chart to be 700px and height - 600px

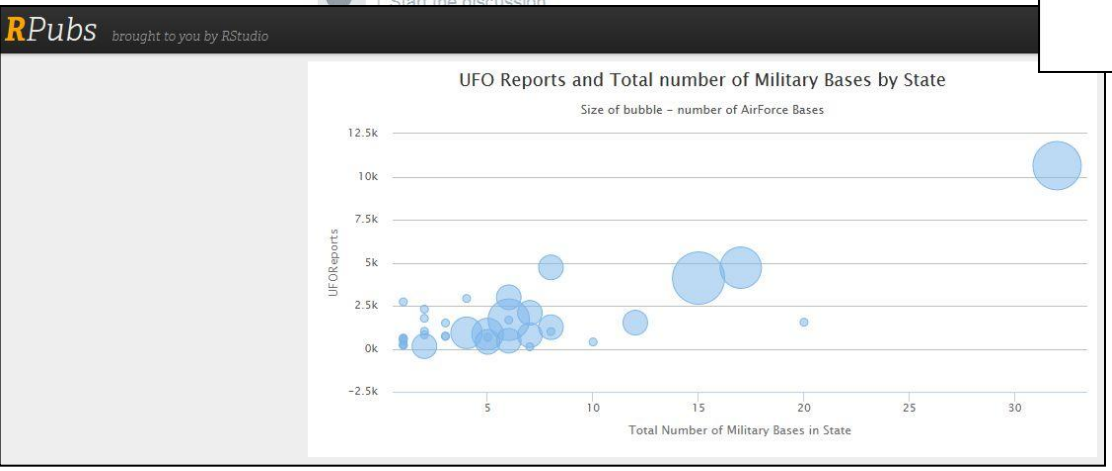
Let's Publish The Chart (Part 1)

- Ways to share:
  - Standalone html page (publish to gist.github.com or rpubs.com), the link is returned. Can be updated. Gist allows several files to be uploaded.
  - Within Shiny Application (functions renderChart & showOutput)
  - Embed into rmd doc, using knit2html, or into a blog post using slidify
- To publish on gist, use your GitHub acount username and password at the prompt after this command executes:
  - ufoplot$publish('UFO Reports', host = 'gist')
- To publish on RPubs, use your account info and tweak RProfile if necessary (see here: http://rpubs.com/conniez/ufo_rchart):
  - ufoplot$publish('UFO Reports', host = 'rpubs')

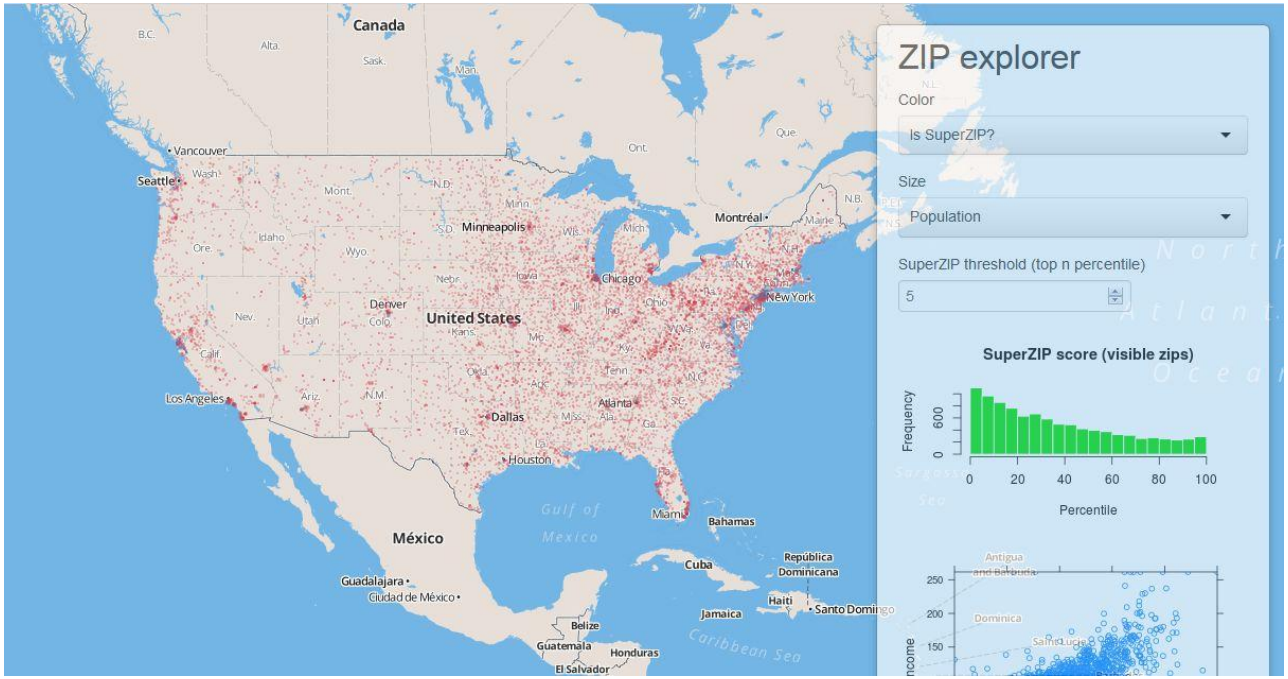
The Result


Fork me on GitHub
UFO Reports
Edit Me
Like 0 Tweet 0 Pin it Share
UFO Reports and Total number of Military Bases by State
Size of bubble – number of AirForce Bases
12.5k
10k
7.5k
5k
2.5k
0k
-2.5k
UFO Reports
5
10
15
20
25
Total Number of Military Bases in State
Edit Me
0 Comments
rcharts
Sort by Best
Share
Start the discussion


RPubs
brought to you by RStudio
conniez
Document Details — Step 2 of 2
Title
UFO Reports rChart
Description
This is the result of publishing an rChart straight to RPubs, using this line of code:
ufoplot$publish('UFO Reports', host = 'rpubs')
Slug
http://rpubs.com/conniez/
ufo_rchart_plain
Continue


RPubs
brought to you by RStudio
UFO Reports and Total number of Military Bases by State
Size of bubble – number of AirForce Bases
12.5k
10k
7.5k
5k
2.5k
0k
-2.5k
UFOReports
5
10
15
20
25
30
Total Number of Military Bases in State

Shiny Interactive Graphics



- Shiny is a web application framework for R, which allows you to create interactive web apps.
- Shiny can be used with any plotting tool in R

- Shiny Tutorial: http://shiny.rstudio.com/tutorial/lesson1/
- Shiny Cheatsheet: http://shiny.rstudio.com/articles/cheatsheet.html
- You can create Shiny apps by copying and modifying existing Shiny apps.

Shiny Online Publishing (web page)

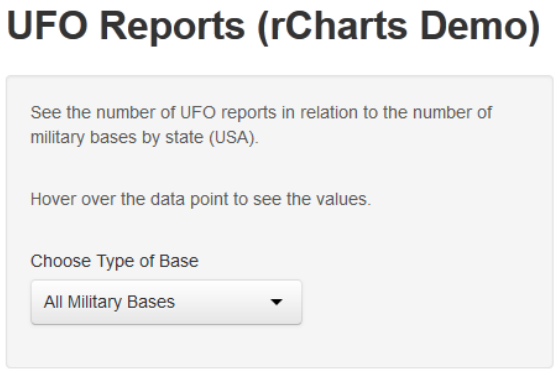
- Shiny Server and Shiny Server Pro are currently only supported on Linux. They provide 64-bit binary installers for Ubuntu/Debian and Red Hat/CentOS.
- You can host your apps in the cloud for free with ShinyApps.io. Just create an account and link to your computer. There are some performance and memory limitations.

Let's Publish The Chart (Part 2)

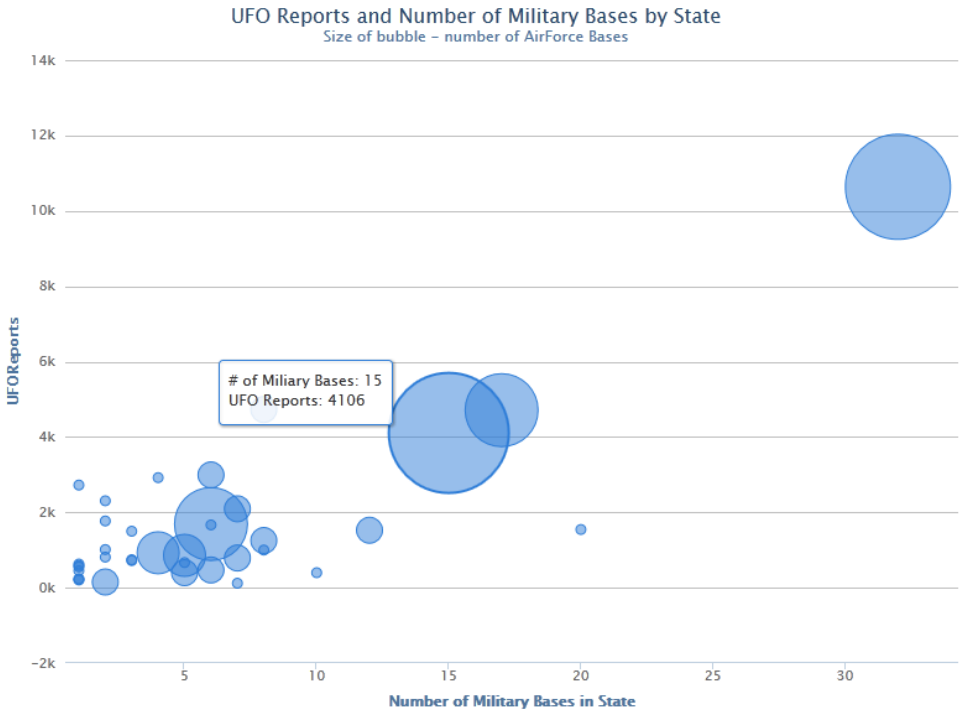
- To publish on Shiny, the steps are similar to steps with ggvis, but the functions for integration are different (see http://rpubs.com/conniez/datavis)
  - Create ui.R and server.R files for a Shiny app in a certain folder (folder name = short name of your app). Add data files in a folder inside of it (unless you plan to connect to them within the app)
  - From Rstudio test your application by clicking "Run App" button in your text editor window of Rstudio.
  - If you have set up your Rstudio with access to Shiny or Shinyapps.io, you will also see the "Deploy" button to publish it online.
  - Another option:
    - runApp("myapp")
    - library(shinyapps)
    - deployApp("myapp")

The Result

- Steps:
  - Create an app folder in wd and copy the data folder into the app folder
  - Reconstruct the code for server.R and ui.R and save them into the app folder
  - Publish the app on shinyapps.io or shiny server



For your practice: transform the code accordingly to work for the rChart you wrote in the exercise earlier



Summary of Tricks and Debugging

- How to connect to online data from a Shiny app (in global.R or server.R)
- Using global.R in a Shiny app will make data available to both ui.R and server.R. Demo: *rCharts-RV-empl* app
- Hands-On: finding bugs and why things are not working
  - missing or extra commas/braces/parens in Shiny app
  - Numerical vs. categorical data in charts
  - grouping data in bar/column charts for hPlot() where data is missing (don't leave NA's, a row must exist per each combination of variables)
  - omitting libraries in apps

More Resources on rCharts

- Getting Started: http://ramnathv.github.io/rCharts/
- What happens behind the scenes: http://rcharts.io/howitworks/
- Presentation on how to share: http://rcharts.io/NYC_May_2014/slides/02_share/#5

Some Differences

Features	ggvis	rCharts	Shiny
Interactive Input of Data	Allows for data input to modify the look	Limited capability for input of variables other than the ones included in the plot itself. Every library is different	Simple widgets for interactive data input
Complexity Level (Subjective)	Basic (one plot function, plus parameter “add-ons”)	Advanced(several JavaScript libraries – a plot function for each, plus add-ons, parameters may differ depending on the library)	Intermediate (simple guidelines for building the app, plus use any plotting tool)

Tips and Things to Remember

- Keep track of your commas, they are used to string things together in ui.R, but not in server.R file. If you get an error that looks similar to this: “Error in … argument is missing, with no default”, you probably have an extra comma in that place
- For Polychart library in rCharts use function showOutput("myChart", “polychart"). If you use highcharts or another library, insert their name into the showOutput() function. Both plot$addParams() and plot$set() work
- For rCharts, if you can’t see your chart in Viewer of Rstudio, try opening it in browser by clicking that button in Viewer
- Errors are inevitable, but the important part is to read the output, which is likely to lead you to a solution

More Resources

- Links to examples and tutorials on Slidify, rCharts, Shiny, etc.: http://www.r-bloggers.com/fantastic-presentations-from-r-using-slidify-and-rcharts/
- 100 interesting data sets: http://rs.io/2014/05/29/list-of-data-sets.html
- Google Dataset Engine: https://www.google.com/cse/publicurl?cx=002720237717066476899:v2wv26idk7m
- Missouri Data: https://data.mo.gov/

Resources Used

- Information about the nycflights2013 data: http://cran.r-project.org/web/packages/nycflights13/nycflights13.pdf
- Linked Brushing material: https://github.com/rstudio/webinars/blob/master/2014-01/4-linked-brush.Rmd
- Examples by creator: http://ramnathv.github.io/rChartsShiny/
- Example of shiny app with downloading data from internet: https://github.com/ramnathv/rChartsShiny/blob/gh-pages/rChartOECD/global.R
- Great examples of all types of charts in NVD3: http://ramnathv.github.io/posts/rcharts-nvd3/index.html
- Great examples with Highcharts: http://rpubs.com/kohske/12409 (and this one http://rstudio-pubs-static.s3.amazonaws.com/16699_4bc388ebe1454c84aaab3d22d17e3aaf.html)
- What chart to use when: http://timelyportfolio.github.io/rCharts_nvd3_systematic/cluster_weights.html
- Examples from Ramnath NVD3: https://github.com/ramnathv/rCharts/blob/master/inst/libraries/nvd3/examples.R
- How to embed into Rmarkdown: http://bl.ocks.org/ramnathv/raw/8084330/ (and this http://timelyportfolio.github.io/rCharts_share/showingoff.html)
- A very detailed explanation on how to use Highcharts API for rCharts: http://reinholdsson.github.io/rcharts-highcharts-api-docs/

rCharts and JavaScript

- All “domains” of the plot can be customized
- Example with http://api.highcharts.com/
- Converting from JavaScript to R for rCharts
  - casual “rules” of conversion
  - training example in sampleCode.R
  - print the js code to prove that everything converted correctly

JavaScript	R
$chart: {height : 800}	$chart(height = 800)

$plotOptionis: {
column: {
dataLabels: {
enabled: true,
align: "left“ } } }
	$plotOptions(column =
	list(dataLabels =
	list(enabled = TRUE,
	align = “left”)))

Thank you!

Questions?