Learning to use package googleVis

Yoon Sung Hong

11/18/2017

1. Introduction

It's extremely important that you download the entire folder instead of just the html. This is so that the interactive charts and the images can be displayed as they should be for the post. Thank you!!

Thank you for taking the time to read this. I know you're really busy, but I hope this post gives you knowledge that will come in handy for you later on.

In class, we learned to use in-game data from NBA and make the analysis as well as visual representation of some of the processed and unprocessed statistics. Similarly, this post aims to elaborate on, in particular, the visual representation part. For example, we want to be able to see how the data extracted applies in a more relatable form for everyone; if you're someone who doesn't understand R, but wants to know about the proportion of rookies drafted into the NBA from Cal, it would be easier for you to learn about this through a more diagrammatic report. By the end of the post, we want to be able to present advanced data in a more tangible and complex ways and analyze the visualized data accordingly, in particular using the package `googleVis`. Further, we should be able to utilize some aspects of packages such as `rjson` and `RCurl`.

You may be asking, *why is this important*? Think from a strategic standpoint. First and foremost, it is easier to **visualize and strategize data in a relatable manner**; often, we want to get people to understand the trend with data, and this requires a language that is more relatable to them, which in this case is visual representation. Second, we will learn how to extract data from online sources when neat csv files are not made available. `rjson` package will allow us to retrieve such data, in particular those in JSON format.

2. Background knowledge

First you may ask, what are `googleVis` and `rjson` packages? `googleVis` is a package that brings in Google Charts API, allowing one to make interactive charts similar to `ggVis`. `rjson` is a package that allows one to bring in JSON (JavaScript Object Notation) objects into R objects. This will be the package we will be utilizing to *import data from NBA's API*, as they do not *directly offer any prepared data files*.

For this post, I will be using mostly data related to NBA (basketball). For those who may not watch basketball every day, some of these concepts we discuss in this post may come across as strange to you. Thus, before we go further with the post, I will explain some of the concepts that may seem confusing later in the post.

We will be considering the 2017 NBA drafts dataset when learning to use `rjson` package and `googleVis` package. The NBA drafts happen annually, to bring new faces into the league with hopes of developing them into the next Michael Jordan's and Lebron James's. There are in total 30 teams in the NBA, and fundamentally, **each of the teams walks out with 2 players each year at the draft**; 1 in the first round and 1 in the second round. However, teams can include these draft picks (for future seasons) for trade with other teams so some teams may end up with other teams' draft picks, ending up drafting more than 2 or less than 2 rookies. These players are drafted from high school, college, and European leagues.

3. Preparation

3.1. Loading packages

We will first load `googleVis`, `rjson`, `sqldf`, and `dplyr` as packages to help us.

```
library(googleVis)
library(rjson)
library(dplyr)
library(sqldf)
```

3.2. Fetching the data

We will only be considering the 2017 NBA draft, where nationally known players such as *Lonzo Ball* were drafted.

The picture above was the picture taken after the 2017 NBA draft, with the chief NBA commissioner Adam Silver and the rookies drafted. NBA.com provides the statistics for NBA rookie draft every year, but it is visible through their API. This requires us to first find the **json** format of the statistics, then use `rjson` package to import the data into R. I will explain this through screenshots and instructions. For your information, my browser is Google Chrome.

First, click on the view tab, then Developer, and Developer Tools on your browser

---

Second, look at the right half of the screen and click the Network tab, then XHR tab right underneath it.

If you already had your target page open, it may not show anything under the tabs after you click them. **Don't worry**, you just need to hit **refresh page**.

When you hit the refresh page, the page now should look something like this.

In our case, you'll see that there are four different "objects" listed. One of those is likely to be the object with all the data stored. If you click on these objects, you'll learn that it opens new tabs, like ones in the diagram below.

Use the preview tab to see what is inside the object. The data you'll be looking for will have a lot of information stored similar to what you see in the API (i.e. objects like **Year**, **Player Names** such as **Lonzo Ball**). Therefore, it is important that you make the comparison between preview tab and original page's data.

Through trial & error, you can find that the 2nd object labeled "drafthistory" is the one storing all the data. You may be asking: how do I know this?

You can see that with the *drafthistory* object, a lot of the data in the **rowSet** part resembles what we see in the website's API. Therefore, it is highly likely that *drafthistory* object is the one we should be using to import the data into our post.

Now, we need to get the URL for this object that we can then use with `rjson` feature to import into R. We can find this in the tab next to *Preview*, at *Headers*.

---

4. Data processing and presenting

4.1. Importing and cleaning the data

Now, using the url, we can import the data with `fromJSON` function in `rjson` package. Let's first create an object for the URL, then import the JSON data. Note that the imported data will be in a list format. The imported data will then be in a form of a list, with list inside a list. In the list of the list, 2nd component is the data collected and 3rd component is the names of the columns. This will make more sense in the code chunk presented.

```
#2017 Drafting data
URL <- paste("https://stats.nba.com/stats/drafthistory",
             "?College=&LeagueID=00&Overall",
"Pick=&RoundNum=&RoundPick=&Season=2017&TeamID=0&TopX=",
sep = "")
dat <- fromJSON(file = URL, method = "C")
rook <- data.frame(matrix(unlist(dat$resultSets[[1]][[3]]),
                          ncol = 12,
                          byrow = TRUE))
colnames(rook) <- dat$resultSets[[1]][[2]]
```

Let's use `dplyr` to arrange the data so that it can be oragnized to be used for the Sankey diagram. To do this, **data must be grouped and aggregated in terms of the number of people that each university has sent to different teams**, since some teams sent more than one player to a team (for example, Los Angeles Lakers drafted 2 rookies from UCLA). You can see the code chunk below to see what I mean.

```
rook$FULLTEAM <- rep(0, nrow(rook))
for(i in 1:nrow(rook)) {
  rook$FULLTEAM[i] <- paste(rook$TEAM_CITY[i], rook$TEAM_NAME[i], sep = " ")
}
rook$COUNT <- rep(1, nrow(rook))
data <- rook %>%
  select(FULLTEAM, ORGANIZATION_TYPE, ORGANIZATION, COUNT) %>%
  group_by(ORGANIZATION,FULLTEAM) %>%
  summarise(count = sum(COUNT))
```

4.2. Sankey Diagram

Now that the data is ready, let's try using the `googleVis` and its Sankey diagram feature through `gvisSankey` function. **from** part of the function should be the university/institute rookie is from, and **to** part of the function should be the team that the rookie was drafted into. It will be weighed by the aggregate numbers we got from above to show the difference in numbers when necessary. I also added some color, opacity, and resizing formats in the function. **Note that I am using print function to make sure that the gvisSankey is displayed in the same html document, not as a separate pop up.**

```
# Sankey diagram using googleVis
print(gvisSankey(data, from="ORGANIZATION", to="FULLTEAM", weight="count",
    options=list(height=800, width=850,
    sankey="{
     link:{color:{fill: 'lightgray', fillOpacity: 0.7}},
     node:{nodePadding: 5, label:{fontSize: 12}, interactivity: true, width: 20},
   }")
  ), 'chart')
```

You can specifically click on the categories, to see the distribution of the players to the selected teams (for the case of this html file you're reading, you cannot. But if you make the googleVis sankey yourself, you will be able to). For example:

In this case, we selected Cal as the organization, and we see that two players from Cal were drafted into the NBA in this year's draft, with one going to Orlando Magic and another going to Boston Celtics. Now, let's see how many players the Philadelphia 76ers have drafted this season.

We can see that 5 players were drafted into the 76ers, which is an unusually high number. However, this makes sense as 76ers are known to be rebuilding their team around young players in the recent years!

4.3. Visually customized scatterplots

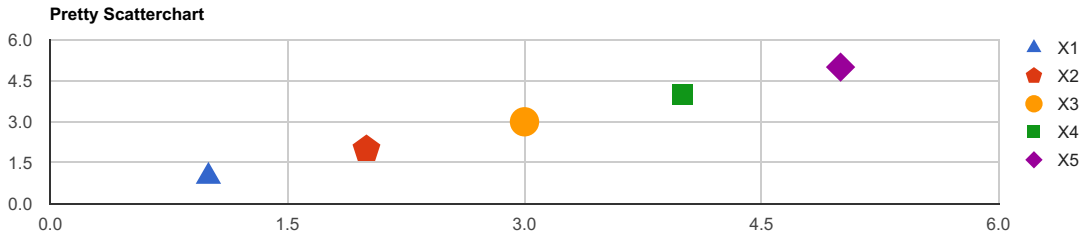
One of the things `googlevis` does extremely well is to make the visualization pleasant to look at. They are catchy, colorful, and neat looking. With scatterplots, you can change the shapes of the plots through customizing the function.

Before we do that, let's first create an arbitrary data we can use to make a scatterplot.

```
#making a dataframe
a <- matrix(nrow = 5, ncol = 5)
a[col(a) == row(a)] <- 1:5
dat <- data.frame(x = 1:5, a)
```

Now, let's try making the scatterplot, but using `googleVis` and its customization feature inside the `gvisScatterChart` function.

```
#First, make an object
scatchart <- gvisScatterChart(dat,
                              options = list(
                                title = "Pretty Scatterchart",
                                legend = "right",
                                pointSize = 20,
                                series="{
                                0: { pointShape: 'triangle' },
                                1: { pointShape: 'polygon' },
                                2: { pointShape: 'circle' } ,
                                3: { pointShape: 'square' } ,
                                4: { pointShape: 'diamond' },
                                5: { pointShape: 'star' }
                              }"))
print(scatchart, 'chart')
```

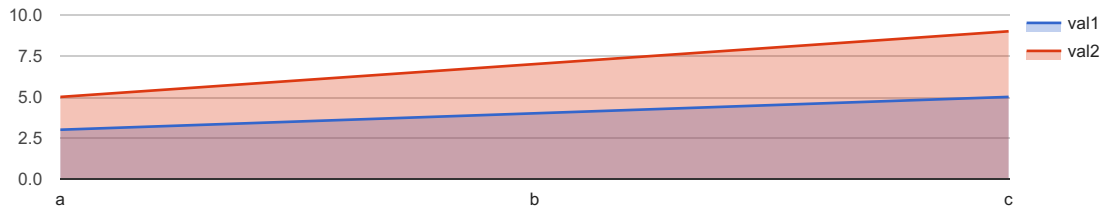


So pretty!

4.4. Area chart

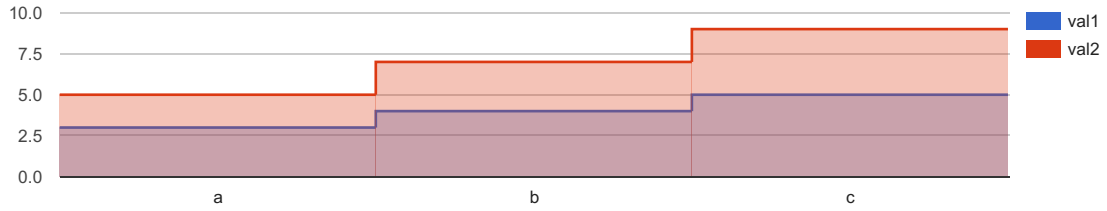
`googleVis` also has a nice, neat version of area chart. Again, let's try making an Area Chart with another arbitrarily created object.

```
#Creating an object
data <- data.frame(var = c("a", "b", "c"),
                   val1 = c(3,4,5),
                   val2 = c(5,7,9))
#Now plotting
areachart <- gvisAreaChart(data)
print(areachart, 'chart')
```



There is also another form of Area Chart - Stepped Area Chart. For this, it is important that we define the x variable and y variable to get the full effect.

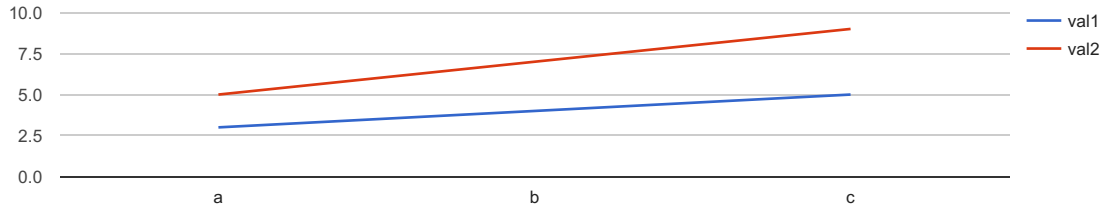
```
areastepchart <- gvisSteppedAreaChart(data,
                                      xvar = "var",
                                      yvar = c("val1", "val2"))
print(areastepchart, 'chart')
```



4.5. Line chart

Let's just use the same defined data frame for the line chart.

```
linechart <- gvisLineChart(data)
print(linechart, 'chart')
```

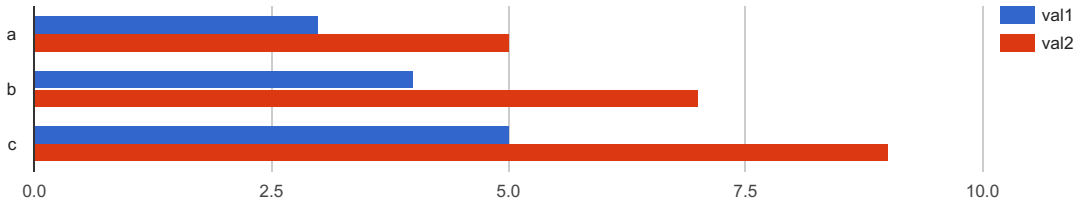


Personally, I think these line charts are sleeker than the r-base line charts.

4.6. Bar charts

Bar charts are generated sideways, again with the typical red and blue colors **google style**. If you want the ones that are generated vertically, try the `gvisColumnChart`, which is the column chart for `googleVis`.

```
barchart <- gvisBarChart(data)
print(barchart, 'chart')
```

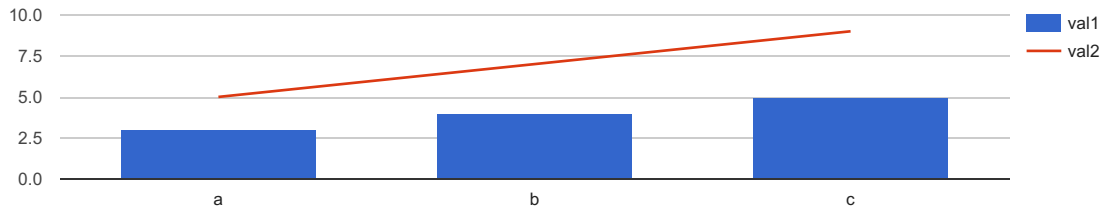


4.7. Combo chart

You can also combine multiple chart types into one, using `gvisComboChart`. You'll have to specify these combinations in the `options` part of the function.

Let's try making a Combo Chart with barchart and line chart.

```
combochart <- gvisComboChart(data, xvar = "var",
                             yvar = c("val1", "val2"),
                             options = list(seriesType = "bars",
                                            series = '{1: {type:"line"}}'))
print(combochart, 'chart')
```



It is clear that there is no right or wrong way to present the data visually. googleVis and its diverse features offer different ways to quantify and visually present the data. I think through this post, it was realizable that it is important to try to present different forms of graphs and compare them to see which ones apply best to the given data set. I am certain that different cases with different data frames offer alternate graphical solutions that work best for the analysis; if the purpose of the analysis was to see the different colleges sending their rookies to NBA every season, a Sankey diagram would be more ideal than a scatterplot. Therefore, a case-by-case representation of diagrams would be necessary.

In terms of rjson package, it shows and proves that there are many sources where data can be scraped, particularly using the API's set up on the websites.

In terms of the analysis itself relating to NBA rookie draft, it seems to be that same/similar list of colleges send their students to the NBA as rookies every year. In other words, it seems to be that basketball-prestigious colleges always seem to send some players to the NBA every year. This is likely because talented high school basketball players tend to choose their colleges based on their reputations in relation to basketball, creating a cycle between incoming athletes and sustained reputation of these colleges.

5.1. What is the take-home message?

Graphically presenting things often provide better conceptual/general ideas. However, it is important to make a clear decision about the type of analysis you want to do and the research question you want to focus on so that you can have the most ideal visual representation to look and analyze for the given research question.

6. References

1. https://cran.r-project.org/web/packages/googleVis/vignettes/googleVis_examples.html
2. http://www.gregreda.com/2015/02/15/web-scraping-finding-the-api/
3. https://stats.nba.com/draft/history/
4. https://thedatagame.com.au/2015/12/14/visualising-the-2015-nba-draft-in-r/
5. https://cran.r-project.org/web/packages/dplyr/dplyr.pdf
6. https://cran.r-project.org/web/packages/rjson/rjson.pdf
7. https://cran.r-project.org/web/packages/googleVis/googleVis.pdf
8. https://stackoverflow.com/questions/2617600/importing-data-from-a-json-file-into-r