Graph Augmentation: Increase the Number of Dimensions and Show More Variables

Jack Ji

November 18, 2017

Introduction:

When we are at the stage of data visualization, we sometimes need to use more than two variables to answer a research question. For example, when we examine the reasons why a country might have so high a life expectancy, we might want to look at many factors like total population, GDP per capita, etc. simultaneously. However, since most types of graphs that we employ are only 2-D, we are restricted to using 2 or 3 variables per graph. Motivated by the need to unshackle the dimensional constraint and reflect information of more than 2 variables in a graph, in this post we will explore ways to increase the number of dimensions of a graph, including using packages `ggiraph` and `plotly` to augment bivariate graphs and create trivariate graphs so that we are ultimately able to achieve simultaneous comparison of multiple variables.

Body:

Both `ggiraph` and `plotly` extend the package `ggplot2`, so we need to import `ggplot2` as well as `dplyr` for data manipulation. `gapminder` is a package that contains the data of life expectancy, GDP per capita and population from 142 countries from 1952 to 2007. Importing `gapminder` is similar to reading in a CSV file and creating a data frame. (You might need to update your R to version 3.4.1 in order to install `gapminder` package.)

```
library(gapminder)
library(dplyr)
library(ggplot2)

# Set the theme of ggplot2 to a white background as opposed to the default grey background
theme_set(theme_bw())
```

Special bivariate graphs:

Since we can only render graphs on a 2-D computer screen, most statistical graphs are bounded by 2 dimensions. This type of graphs are intended to illustrate the relationship between two variables. Nonetheless, it is possible for some augmented 2-D graphs to include information of more than 2 variables when we want to compare many variables simultaneously. We will see a few ways to do so below.

Labeled points

On a scatterplot, we may include information of the third variable directly in the points. The `ggiraph` package would enable us to do that. For the example below, we want to include the information about our third variable `country` in a scatterplot whose x-values are GDP per capita and y-values are life expectancy. The graph below is hence a simultaneous comparison of three variables–GDP per capita, life expectancy and country. While we are able to observe that Asian countries have lower GDP per capita and life expectancy than European countries, we may also compare the GDP per capita and life expectancy between two specific countries (for instance, China has higher life expectancy but lower GDP per capita than Thailand).

Usage: hover the cursor above a point to see a label indicating the country of this data point

```
library(ggiraph)
# Select the data only in the year of 2007 and from countries in Europe and Asia (to limit the number of points)
gapminder_2007 <- filter(gapminder,
                         year == 2007 & (continent == "Europe" | continent == "Asia"))

# Create a graph with GDP per capita as x-values and life expectancy as y-values
g_ggiraph <- ggplot(gapminder_2007,
                    aes(x = gdpPercap, y = lifeExp, size = pop, color = continent)) +
             geom_point_interactive(aes(tooltip = country))

# Display the graph
ggiraph(code = print(g_ggiraph))
```

Other features of package `ggiraph`:

Since the points are relatively close to each other, we could make the black box of the point label transparent so as to minimize its visual interference.

Usage: hover the cursor above a point to see how the label has changed

```
# Display the graph with altered setting for point label
ggiraph(code = print(g_ggiraph), tooltip_extra_css = "background-color:transparent;")
```

Since the points are rather clustered, we could enable the magnifier so that readers would be able to zoom in on the graph.

Usage: double-click or scroll up to zoom in; click the leftmost of the three buttons at the top-right corner or scroll down to zoom out

```
# Display the graph with magnifiying capability
ggiraph(code = print(g_ggiraph), zoom_max = 5)
```

We could increase the number of dimensions even further. For the example below, a webpage is attached to each point in the graph. Since each webpage may include much information about many other variables of the country that each point represents, the number of dimensions of this graph is also increased. From the webpages, we may find out about other variables such as the area, HDI, Gini, currency, language and even something that a variable cannot capture like history.

Usage: open this html file in a browser to see it work–click on a point which would direct the reader to the Wikipedia page of the country

```
# Join the main web address of Wikipedia with the name of each country
gapminder_2007$ext_links <- sprintf("window.open(\"%s%s\")",
  "http://en.wikipedia.org/wiki/", as.character(gapminder_2007$country))

# Create the graph with GDP per capita as x-values and life expectancy as y-values  with each point linked to its
country's Wikipedia page
g_ggiraph <- ggplot(gapminder_2007,
                    aes(x = gdpPercap, y = lifeExp, size = pop, color = continent)) +
             geom_point_interactive(aes(tooltip = country, onclick = ext_links))

# Display the graph
ggiraph(code = print(g_ggiraph))
```

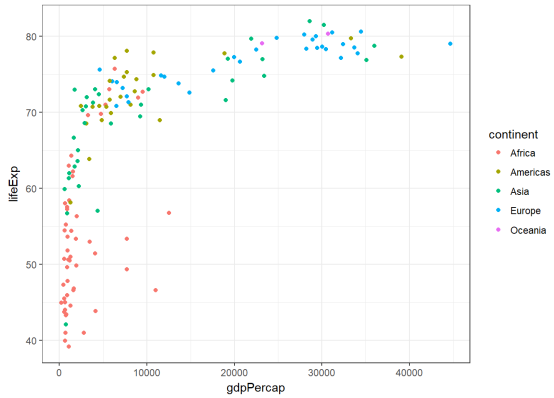
Mapping a variable to an attribute of the graph

We have used this technique to include a third variable in our graph before in this class, although it was never formally introduced to us. We may map the values that a variable can take to a variance of an attribute of the graph. More concretely, we can let properties like the size, the shape, or the color of a point to indicate the value of a variable. This notion is in line with the notion in linear algebra–we are simply finding a bijection that maps a value that a variable can take to a particular instance of a graph attribute. One advantage that this technique has over the one above (labeling points) is that the third variable is reflected visually on the graph, which obviates the need for some operation (clicking on the point) to access the variable. (Read more about graph attributes of `ggplot2`)

This example maps the continent of each country or point to a specific color that a point may have. For visual ease, color is good for categorical variables that only has a few possible values. By eyeballing the graph, we could conclude that most Asian and African countries have low GDP per capita and that most European countries tend to have higher GDP per capita and life expectancy.

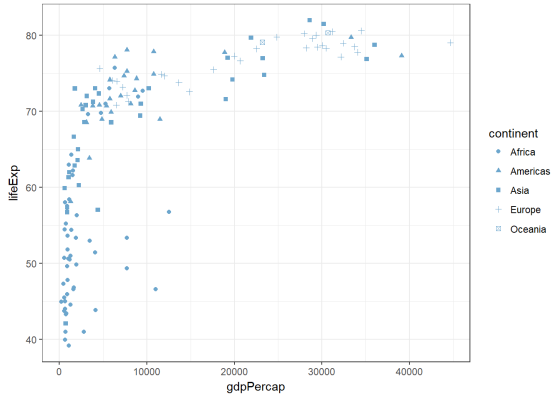
```
# Select the data only of the year 2002
gapminder_2002 <- filter(gapminder, year == 2002)

# Create a graph with GDP per capita as x-values and life expectancy as y-values (point color reflects continent)
ggplot(gapminder_2002, aes(x = gdpPercap, y = lifeExp, color = continent)) +
             geom_point()
```



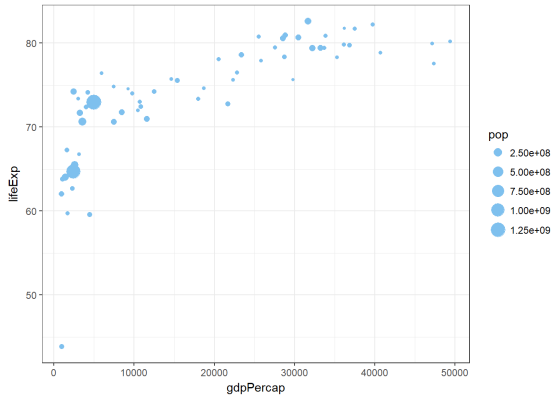
This example maps the continent of each country or point to a specific shape that a point may have. Like the technique of color mapping, shape is only good for categorical variables that only has a few possible values.

```
# Create a graph with GDP per capita as x-values and life expectancy as y-values (point shape reflects continent)
ggplot(gapminder_2002, aes(x = gdpPercap, y = lifeExp, shape = continent)) +
               geom_point(col = "skyblue3")
```



This example maps the population of each country or point to a specific size that a point may have. Unlike the previous techniques of color mapping and shape mapping, size is good for continuous variables because size itself is a continuous variable. We may conclude from this graph that countries with very large population tend to have low GDP per capita and life expectancy.

```
# Create a graph with GDP per capita as x-values and life expectancy as y-values (point size reflects population)
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp, size = pop)) +
               geom_point(col = "skyblue2")
```



These three techniques above could be combined and together bring the number of variables reflected in a graph to 5. Nevertheless, there is a problem–it might be slightly difficult for human eyes to see, for example, the relationship between the size or color of the points and the x-values.

Animation

Generally, there is a way to eschew the restriction that 2-D graphs have placed on the number of variables that our graph can have–to render a 2-D graph animated. By doing so, we can map another variable from our data to the implicit time attribute of our graph–the data from each year is mapped to a particular frame of the animation of the graph.

Starting from this point, we will begin to use the package `plotly`. We first create the same scatterplot that we have been using above, whose x-values is GDP per capita and y-values are life expectancy. Then, we add the variable year to the graph via animation. (Each point may also be programmed to show information about many other variables.) Therefore, each frame may be viewed as a "derivative" from "differentiating" of the data with respect to the third variable (in this case, year). In this graph, we observe that both GDP per capita and life expectancy increase with year, but GDP per capita tend to float about for certain countries and even slightly decreases in 1992 and 2007.

Usage: click the "Play" button to see how the data evolve over the span of 55 years; also use the slider to see the data at a particular year (only multiples of 5); hover the cursor over a point to see more information

```
library(plotly)

# Create a graph with GDP per capita as x-values, life expectancy as y-values and year as frames
g_plotly_animated <- ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point(aes(size = pop, frame = year, ids = country)) +
  scale_x_log10()
ggplotly(g_plotly_animated)
```

Multivariate graphs:

Since the maximal number of dimensions that we can perceive is 3, the phrase "multivariate graphs" is generally synonymous with 3-D graphs. In this type of graphs, we can include 3 variables with ease. We are allowed many things–simultaneous comparison of 3 variables, or even create a topographic map as below. A 3-D topographic map is a model of an geographic object in real life (in a 3-D Euclidean space) and hence has the variables x, y and z. Although our eyes can only perceive 2-D objects, a 3-D graph may be freely oriented so that readers can observe the data points/lines/surfaces at different angles and imagine the 3-D object in their head, rendering a 3-D graph as effective and pellucid as its 2-D counterpart.

You need not understand this paragraph–just some background information of the data. `volcano` is a matrix that contains the topographic data of a volcano in the Auckland volcanic field. The row number and column number of this matrix form a 2-D coordinate and each entry in the matrix is the third variable–height. This matrix has sufficient data for us to construct a 3-D map of this volcano and see the shape of the volcanic vent.

Usage: drag the graph to view it at different angles; scroll up and down to zoom in and out; hover the cursor above the figure to see information of that particular point. (If the space below is blank, move the cursor above the blank space to "wake up" the graph.)

```
# Create a sequence from 100 to 100 + the number of rows of volcano
x_plotly <- seq_len(nrow(volcano)) + 100

# Create a sequence from 500 to 500 + the number of columns of volcano
y_plotly <- seq_len(ncol(volcano)) + 500

# Create the 3-D topographic map
plot_ly() %>%
  add_surface(x = ~x_plotly, y = ~y_plotly, z = ~volcano)
```

However, below is a more common usage of a 3-D graph. For the example below, we augment the 2-D scatterplot above (with x-values replaced by year) by adding a third variable population as height in the graph and create a 3-D scatterplot. This graph allows a direct comparison between a variable of interest and the other two related variables, divulging a mutual relationship of the 3 variables, whereas drawing two 2-D graphs can only reveal pairwise relationship between 2 variables. The layers at the bottom of the graph that are horizontal to the year axis indicate that life expectancy increases (shifts upwards) with year. Asian and African countries generally have low life expectancy while American and European countries have relatively high life expectancy. Most countries have rather constant population over years except for few Asian countries. Also, population tend to increase along with life expectancy.

Usage: same as the one above

```
# Create a graph with year as x-values, life expectancy as y-values and population as z-values
plot_ly(gapminder, x = ~year, y = ~lifeExp, z = ~pop) %>%
  add_markers(color = ~continent)
```

Take home message

To summarize, we examined some ways to increase the number of dimensions of a graph. When we need to conduct simultaneous comparison of more than two variables, we may augment a 2-D graph with package `ggiraph` by including information of other variables in the data points–the value of another variable, a short blurb, or even a link to a few paragraphs–and animating the graph to show the progression of data with respect to the change of a third variable, which is a great aid to time series analysis. Or we could create a 3-D graph with package `plotly` and effectively make use of the last dimension, z, of the three that we are able to perceive to reflect another variable within our data in order to maximize the number of dimensions of our graph.

References:

https://cran.r-project.org/web/packages/gapminder/README.html

https://davidgohel.github.io/ggiraph/articles/an_introduction.html

https://plot.ly/r/

https://cran.r-project.org/web/packages/ggplot2/vignettes/ggplot2-specs.html

https://en.wikipedia.org/wiki/Bijection

https://cran.r-project.org/web/packages/ggplot2/vignettes/ggplot2-specs.html

https://www.rdocumentation.org/packages/datasets/versions/3.4.1/topics/volcano

http://pages.csam.montclair.edu/~mcdougal/SCP/statistical_graphs1.htm