EDAV Fall 2019 PSet 4

Read *Graphical Data Analysis with R*, Ch. 9, 11

Read *Interactive Data Visualization for the Web*, Ch. 3, 5

The theme of this problem set is freedom. As you will see, you'll have more choices than usual in terms of data and packages.

Remember that you are expected to meet with your partner in person–even if you are working on different parts there is a benefit to having someone to ask questions to or spot the typo in your code. You are also expected to communicate frequently, pull your weight, and not make unilateral decisions. It's all about collaboration and partnership.

Grading is based both on your graphs and verbal explanations. Follow all best practices as discussed in class.

1. Missing values

You may choose your own data. The only requirement is that it has some missing values. If the dataset is large, you may choose to work with only some of the columns and a random selection of rows.

Use any R plotting package (`ggplot2`, `naniar`, `mi`, `extracat`, etc.) to answer the following questions. (Note that `extracat` is not currently on CRAN but you can install it from a CRAN read-only mirror on GitHub: https://github.com/cran/extracat)

(a) Visualize and describe missing column patterns. Which variable has the highest number of missing values?

(b) Visualize and describe missing row patterns. Which row has the highest number of missing values?

(c) Do correlations exist between the values of other variables and missing patterns? Investigate two missing patterns and draw conclusions such as:

"The number of missing values for the `last sale price` column seems to be correlated with `average income in zipcode`: there are more missing values in `last sale price` for homes in low-income neighborhoods."

"There does not seem to be any correlation between `apartment type` and missing values in `last sale price`"

Support your conclusions with graphs.

2. Time Series

(a) Use the **tidyquant** package to collect information on anything other than stock prices for which time series data is available. Create a multiple line chart to compare trends over time. Your chart should have at least 3 lines.

(b) Transform the data so each series begins at 100 and replot. (see Unwin, p. 229)

(c) Compare the graphs in (a) and (b).

3. Cause of Death

Data: https://wonder.cdc.gov/ucd-icd10.html

(a) Create a series of choropleth maps in which only one variable changes, such as level of a factor variable or time.

For inspiration, see these examples:

https://www.nytimes.com/interactive/2017/06/30/upshot/the-best-and-worst-new-york-neighborhoods.html

https://www.nytimes.com/interactive/2017/10/05/upshot/gun-ownership-partisan-divide.html

(b) Interpret the graphs you drew in (a).

4. Mosaic plot (SVG / D3)

(a) Manually create a 2 x 2 mosaic plot of party affiliation by gender for (House) representatives currently in the U.S. Congress using SVG. Data is available here in `.csv` form: https://github.com/unitedstates/congress-legislators

You may remove any Independents.

The SVG should be 500 x 400 pixels and included in your `.Rmd` file between svg tags (**not** in a code chunk):

```
<svg width="500" height="400">
```

Your svg code here.

```
</svg>
```

The axes and levels of each variable should be labeled.

(Don't despair, this is the only time you will have to create SVG by hand for this class!)

(b) Change your code from (a) so that with the exception of the blank SVG, the mosaic plot is completely created with D3, based on a dataset that you provide that contains the 4 values you calculated for part (a). Your code should work if the values change.

There are two options for including D3 directly in your `.Rmd` file:

i. You can include it as before in the "non code" section of the file using the template below. If you go this route, you will likely prefer to work in a text editor with a browser open on the other half of your screen, or in some other manner which will allow you to view your visualization as you go.

```
<svg id="partb" width="500" height="400">

<script src="https://d3js.org/d3.v5.min.js"></script>

<script>
```

Your JavaScript/D3 code goes here.

```
</script>
```

ii. Use the **r2d3** package. The setup here is different. You will create your visualization in a `.js` file and then call it from an R chunk. More information is available on the package site: https://rstudio.github.io/r2d3/ (Note that you no longer need the preview version of RStudio.)