Week 3 Journal

Aaron Schram

Grammar Implementations

From last week's journal, the Grammar of Graphics was mostly defined as the orthogonal classes which consisted of variables, algebra, scales, statistics, geometry, coordinates, and aesthetics with the idea that knowing this specific grammar allows the construction of almost any graph without having to specifically call the name of graph, such as a scatter plot (Wilkinson, 2010). With that in mind, I started with reading *A Layered Grammar of Graphics*, Wickham (2010) to directly compare to the grammar as graphics as I thought this would have been the closest in actual syntax. Additionally, I know a lot more about R and base R plot than the others. As for the other articles of comparison, I chose *Gramm: Grammar of Graphics Plotting in Matlab*, Morel (2018) and *seaborn: statistical data visualization*, Waskom (2021) with the idea of comparing them both to Matlab plot and plt.plot() from matplotlib pyplot, respectively.

Starting with GGplot2 in R, the two immediate deviations from the Grammar of Graphics are the concepts of layers and a somewhat disregard for the first three defined classes of the Grammar stated above. Wickham defines a layer as a combination of data, the context for data, and a statistical summary of the data (Wickham, 2010). In a less technical sense, this allow us to define those three things as elements from the Grammar of Graphics, but it also allows this new mapping of those definitions to be over-layered on top of each other, so we have a definition or class that allows the combining of graphs types, such as scatter plots and regression lines. For the first three classes of the grammar, we see in GGplot2 that these are not actually performed. R has lots of ways to perform these operations, and GGplot2 will do transformations by the coordinate grid used.

As for GGplot2 versus base R function plot, plot's syntax mainly allows the specification of x, y, line type, and axis titles. The plot function doesn't adhere to the grammar in any sense that GGplot2 does, where we have a step-by-step building of the grammar from picking the mark, the mark shape, the graph type, aesthetics, coordinate grid, facets, and layers. For the base plot function to do any of these, you would need to use functions outside of itself. Therefore, the plot is limited to basic graphs on the Cartesian plane. I would it say it primarily just uses geometry and aesthetics in its design.

Moving on to Matlab and Gramm, Gramm is somewhat similar in construction to GGplot2 or that it tries to be. It is limited by Matlab's low graphing ability. Unlike GGplot2, Gramm appears to allow data manipulation and transformations within its structure. With the idea of using grouped variables as the backbone for the plotting reasons, the Gramm function has built it grouping with color, columns, and subset. The subset function is the biggest difference between GGplot2 and Gramm with another difference being that the geometry and layer is created outside of the Gramm function. It is worth noting that Gramm does allow using an update function outside of it as well.

Comparing Gramm to Matlab plot, the plot function appears to be used specifically for line plots and plotting tables. There is little going on here in terms of grammar. We specify the x and the y coordinates, we can layer them, denote color, and add line specifications. We cannot manipulate the data within the function, create layers or facets. It feels like a watered-down version of R's plot function. There is not a lot of grammar if at all being used here. We have line plots and not much else.

Finally, Seaborn is the most recent interpretation of the Grammar of Graphics and GGplot2 in Python, ignoring plotnine. Seaborn seems to be almost exactly like GGplot2 in that we start with calling a dataset, specifying x and y, and then we recursively build off of that initial using so.add, instead of + in GGplot2. The biggest difference between GGplot2 and Gramm is that seaborn allows data transformations within the function, such as aggregation. It even allows use of a scaling function for the data, like a log transform. So, GGplot2 drops the first three grammars, Gramm tries to make use of variable sub setting using the algebra class, and Seaborn uses the scales class to manipulate the data. Those are the biggest differences between the three. Otherwise, they all allow faceting, layering, and a plethora of aesthetics.

For Python's pyplot function, there is no one plot function that you can use to create a series of plots or use grammar to create plot. You would have to specify what plot function specifically. Even then, you still need to provide other functions to apply labels and aesthetics to the graph created. The base plotting function is most comparable to Matlab's base plot function in simplicity, but it has added functions to add on to the created graph, while R's base plot function has several of these within the plot function. Although the pyplot function is not wrapped all in one command, it still allows the user to adjust the scale of the data, annotate graphs, create facets, adjust sizes of scatter plot points, but does not look like it allows the use of layers. I believe the R base plot can be made to layer plots with use of other functions or specifying "add = TRUE". Pyplot at least adds far more flexibility than Matlab's base plot function.