Stat 992 - Research Paper - Grammar of Graphics

Aaron Schram

Introduction

The Grammar of Graphics (GoG) can be considered an orthogonal system of seven classes or data flow from which one can form a mathematical foundation for creating statistical graphics (Wilkinson, 2010). The core purpose of this data flow or class system is to design a grammar that can define any plot of any form. For further defining the GoG as a base language for forming statistical graphics, Wilkinson (2010) further describes this decomposition of the grammar on statistical graphics into seven strict categories which are the Variables, Algebra, Scales, Statistics, Geometry, Coordinates, and Aesthetics.

Then, we define Variables as the input of data from a file to a transformation of the data using table joins, filters, selects, and any other data pre-processing tools, which is denoted as the Algebra. From the Algebra, we have our the true data that we are interested in deriving a statistical graphic from, so we move onto the Scale class. Wilkinson (2010) denotes this the Scale class as the least observable or the most ignored class in with regards to the most common of charts. The core idea is that we a preferred scale or measurement for the data. These scales could consist of just an identity transformation on the algebra all the way up to non-linear transformations, but furthermore this class is also associated with the structure of the data. Here, the measurement or data structure just has to do with the class of the data itself, such as numeric, binary, interval, and so on. This truly marks the end of the data pre-processing phase as the remaining classes have to do with how the data will be mapped and plotted.

As for the last 4 classes, Statistics is the last class after all the data has been processed. It has to do with how we want to perceive the data. For example, do we want view every point as in a scatter plot, or do we want to view some kind of summary statistic as in histogram or bar chart. We even define this as output from regression lines or interactions plots from statistical tests and procedures. From the Statistics, the next class is the Geometry which is closely tied to the Statistics, since Geometry is all about how the data is being represented on the plot itself. Geometry is the part where we specify if we are looking at lines, points, or bars, and how we perceive these on a graph. An easier way to examine this would be through

the use of geom functions used in ggplot2 package (v3. 3.3; Wickham, 2016). These represent the visualizations of each layer and the representations of the Statistics (Wickham, 2009). In other words, the Geometry is the plot being created.

Finally, our last two classes are straightforward. Second to last, we have Coordinates, and this is just the coordinate grid system that we are plotting in. Then, we have the Aesthetic class which produces are final visual given all other classes. Wilkinson (2010) breaks the Aesthetic class into seven sub-functions based off of Bertin's *Visual Variables* (1967), and calls them position, size, shape, orientation, brightness, color, and granularity. Therefore, the Aesthetics class can be perceived as everything to do with the visualization of the graphic from the size each point to the color scheme of the graph itself.

Minimizing the Grammar

The Grammar of Graphics introduces a total of seven different orthogonal classes that are fundamental building blocks for constructing statistical graphics, however not all classes are needed to construct graphics, and the construction is not necessarily a straight data flow. It will be argued that the Variable, Algebra, and Scales classes are not necessarily needed. Additionally, classes can be reordered and combined to form alternate classes or even layers. One of the biggest points of my argument comes from Wickham's *A Layered Grammar of Graphics* (2010), where ggplot2 and layering are introduced. Wickham (2010) argues that the DATA, TRANS, and Algebra steps of the original grammar can be fully ignored by leveraging tools within **R** to preform these data processing steps.

Using ggplot2 and **R** as stepping stone for the first argument, the Variable class can be fully ignored. We do not need to graphical software to implement pulling data into the program, since there are several pre-made programs that will pull and store the data. Such as, Standard Query Language functions will make the data usable for later, and they can even perform transformations and data pre-processing operations. This seemly eliminates the Algebra class as well or at least combines the Variable class and Algebra class into one single step. The bottom line or at least on par with Wickham's (2010) reasoning is that both of these steps can be handled and performed by functions or libraries outside the graphics tool. Then, the graphics tool can call upon the data set created.

Now, the Scales is an interesting case as we can perform simple functions to transform our data very easily outside a graphics, but we can also easily implement these transformation within said tool. For instance, in **R**, we could use base functions to scale and transform the data, such as log, exp, or scale (R Core Team, 2023). However, in a function like ggplot2, we use scale arguments to align layers of plots, but these are done after calling the Statistics and the Geometry classes, and these consist of log scales, continuous scales for x and y axes, and color scaling for heat maps (Wickham, 2010). These individual uses of scaling on layers allows us to realign our generated graphics before we overlay them. On the other hand, we can perform these basic transformations outside of the graphics tools, and the realigning of

our layers can be seen as part of the layering step within the Coordinates class in this case or even as part of the Aesthetics class. This is due to the fact that Wickham's (2010) scaling in ggplot2 is placed out of order from the original use of the grammar as specified above.

To compound further on my logic for why Scales should not be a class, consider two different lines of thought, pre-sent data of interest and Generalized Linear Models. We often receive data in a preset form on a preset scale of interest. This by nature invalidates the need for the Algebra and Scales class. For the second case, suppose we have data that will generate statistics on as part of the Statistics class and such data will belong to non-Gaussian right-skewed data. Naturally, we would fit Generalized Linear Model to the data, but this would require transforming the data using a link function to our linear predictor space. This transformations would generate our statistics on the wrong scale of interest, and we would use the inverse link to back transform our data to the correct scale of interest. Here, we performed scaling and transformations within the Statistics class. Therefore, the Scales class has been shown to be within two other classes of interest, and we have shown it to be easy to perform outside of the graphics tool as well.

The Essential Grammar

Previously, we described that the Variable, Algebra, and Scales classes were not entirely needed, but we did not mention the remaining four classes of grammar. This leaves us with Statistics, Geometry, Coordinates, and Aesthetics. Statistics and Geometry are the foundations for forming the statistical graphic as Statistics dictates how we want to display our data and Geometry physically displays the statistics on the graphic. We can think of these two classes as the most rudimentary form of data and the graph type. Then, we have aesthetics which allows us to customize and visualize our statistical graphic. Clearly, we want to be able take our data in a specified form or summary statistic, pick a plot type, and then display it. The classes Statistics, Algebra, and Aesthetics will do this at the bare minimum while retaining the ability to customize our data points positioning, size, and color.

Now, this leaves us with the Coordinate class. This particular class can go either one or two ways. We can view the Cartesian plane as a default choice as in the cases of several different graphical software implementations (Morel, 2018; Satyanarayan *et al*, 2017; Waksom, 2021; Wickham 2010). Therefore, from this standpoint, the Cartesian plane and by extension the Coordinate class is being implicitly called upon. If we are not truly specifying a grammar, then are not truly taking use of the class. Thus, the Coordinate class is inconsequential to the importance of the other classes if we only operate within the domain of Cartesian plane. Otherwise, if you user needs to operate in a different system, such polar coordinates or other dimensional plots, then the Coordinate class will have to be called specified from the default.

Additionally, on the topic of the Coordinate class, we will find we will find facets and layers that can and we require a degree of specification. Wickham (2010) asserts that facets are a form of an element that requires a specification of the Coordinate class at each instance,

while layering is also an element that makes use of the Statistics and Geom classes. However, Satyanarayan *et al* (2017) describe a unit as a whole as a specification for a single Cartesian plot, and then a layer is combination of units placed together as over layed Cartesian plots. In either definition, we see that facets and layers are heavily intertwined with the Coordinate class as each on can be seen using multiple instances of creating a new plot and recalling each of the grammar again, repetitively. The only difference being that facets specify different Coordinate class each iteration, but the point still stands that in the case of facets and layers that the Coordinate class is essential.

GGplot2 and the Grammar

To Further evaluate the Grammar of Graphics (GoG), we will examine to real life use cases by examining Wickham's ggplot2 from *a Layered Grammar of Graphics (2010)* and VegaLite from Satyanarayan *et al's Vega-Lite: a Grammar of Interactive of Graphics* (2017). Starting first with the ggplot2 package (v3. 3.3; Wickham, 2016) as specified from the reading and above, Wickham (2010) breaks use of the grammar in ggplot2 into five major groups: Defaults, Layer, Scale, Coord, and Facets. From the previous sections, Defaults corresponds to the Variable and Algebra which were considered as unneeded implementations, since **R** has several ways of importing and cleaning data.

Starting with Layers, Wickham (2010) subdivides this section into Data, Mappings, Geom, Stat, and Position, where a Layer is equivalent to an element within Wilkinson's *Grammar of Graphics* (2010). Intuitively, it is quite easy to see that a Layer is primarily composed the Statistics class and Geometry class as denoted by the inclusion of Geom and Stat, but also that stated in the essential grammar section of this paper that a layer forms an actual distinct plot. The minimum of which that is required to create is the Statistics and Geometry classes.

Next, we have Scale which is just the Scales found in the original grammar. The biggest difference is that the placement of the of scales in the original grammar is placed before the Statistics and Geometry classes with the grammar retaining an orthogonal data flow or forward structure. Wickham (2010) describes his version of Scale as mapping from the data to an aesthetic which differs greatly from the idea of a preferred measurement or scale, and He makes no mention of the inherent data types in this sections. Since there are differences, We should not consider this as the same Scales represented by the grammar. Instead, it is more of described as aesthetic choices for the Geometry, and therefore it seems to be more of a part of the Geometry class than anything else.

Finally, we have Coord and Facet classes. As previously explained, facets are equivalent to elements, and they require the combination of Statistics and Geometry or in this case Geom and Stats from the Layers section to create them as well as a specification of a coordinate grid system. Essentially, facets just create new plots entirely, while the Coord group is just the same as the Coordinate class.

Vega-Lite and the Grammar

Vega-Lite differs quite a bit from ggplot2 in that it is supposed to incorporate the Grammar of Graphics and the Grammar of Interaction using unit specifications (layer, concat, facet, and repeat operators) with mark types, while the actual Grammar of Graphics used is broken in Unit Specification, View Composition Algebra, and Nested Views (Satyanarayan *et al,* 2017). Here, a single Unit Specification refers to the data, mark, and visual channel, and then View Composition Algebra refers to multiple Unit Specifications or layers, which are additionally defined as Concatenations, Facets, and Repeats (Satyanarayan *et al,* 2017). Finally, Nested Views are just dashboards.

Starting with Unit Specification as reported by Satyanarayan *et al* (2017), a single unit creates in total a single Cartesian plot, so right-away the Coordinates class of the grammar is invoked. It would make sense, then that the rest of the inner category of Unit Specification should involve all previous classes if it were to strictly follow the Grammar of Graphics data flow, but we see that this level of grammar from Vega-Lite involves data, data transformations, mark type, and encoding at the Unit Specification. Now, data, data transformations, and mark type correspond to the Variable, Algebra, Geometry classes, respectively, while encoding incorporates channel, field, data-type, values, functions, scale and guide as additional sub-fields. Then, channel has to do with Aesthetics, field and data-type is the Statistics and Scale classes, and then we can view value as a sub-set of field and the others as a subset of Aesthetics. Therefore, a single Unit Specification is sufficient to invoke all of the Grammar of Graphic with all else being extra.

Implementations

Furthering the discussion of the Grammar of Graphics with ggplot2 and Vega-Lite, we will be examining the code for creating such graphics. Examining the code will allow us to determine the ease of the implementation of ggplot2 and Vega-Lite, but it will also allow us to further inspect the authors' incorporation of the grammar.

Starting with two examples from **R** documentation and from the Posit ggplot2 cheat sheet, we have: as written by R Core Team (2023),

"ggplot(data = sample_df, mapping = aes(x = group, y = value)) + geom_point() + geom_point(mapping = aes(y = group_mean), data = group_means_df, colour = 'red', size = 3)",

and then we have from Posit's website under resources and cheat sheets (https://posit.co/resources/cheatsheets/) for *Data Visualization with ggplot2 :: Cheat Sheet* (2024), a code template written as,

"ggplot(data = <Data>) + <Geom_Function>(mapping = aes(<Mappings>), stat = <Stat>, position = <Position>) + <Coordinate_Function> + <Facet_Function> + <Scale_Function> + <Theme_Function>".

From the first chunk of code, we the pre-processed data is already ran in using the data variable, and then mapping calls the x and y variables, and finally geom_point creates the Statistics class and the Geometry class being called. Meanwhile, arguments like colour and

size are purely part of the Aesthetics class. So, we see initially from default setting that ggplot mainly calls the Statistics, Geometry, and Aesthetics class without the need to specify any other arguments. In theory, we could reduce this further to the Statistics class and Geometry class if we refused to use or specify any of the choices allowed for the Aesthetics class.

From the second chunk of code show for ggplot2, we see the full bore scale of all potential arguments that ggplot2 is capable of taking. Specifically, we see the geom_function as the Geometry class, stat as the Statistics class, coordinate_function as Coordinate class, and the scale_function as the Scale class. The position variable is part of an element, and the theme_function is purely part of the Aesthetics class. Additionally, we do not see the extra aesthetic choices that could be made with arguments such as color, fill, or size options in this representation of the code, but otherwise ggplot 2 appears to make use of the Grammar of Graphics as specified in the earlier section.

Moving on to Vega-Lite, we have one example taken from the Vega-Lite website as written,

"{"data": { "values": [{"a": "C", "b": 2}, {"a": "C", "b": 7}, {"a": "C", "b": 4}, {"a": "D", "b": 1}, {"a": "D", "b": 2}, {"a": "D", "b": 6}, {"a": "E", "b": 8}, {"a": "E", "b": 4}, {"a": "E", "b": 7}] }, "mark": "bar", "encoding": { "y": {"field": "a", "type": "nominal"}, "x": { "aggregate": "average", "field": "b", "type": "quantitative", "title": "Mean of b" } } }" (Introduction to Vega-Lite, 2024).

Due to a general lack of experience with Vega-Lite, the interpretation of the code will be minimal at best and will rely mostly on previously stated arguments, but unlike ggplot2 we have actual Variable and Algebra classes with the data and values read in functions. Mark is the Geometry class as it has bar as in bar plot stated afterwards. Then, encoding as stated previously has field, which denotes the data type and scale, and we have an aggregate function to take the mean values of x as well as a title for the x-axis. Therefore, we see that this code is a single Unit Specification, and we have all seven of the classes represented as originally stated on this topic.

Overall, Vega-Lite incorporates more of the Grammar of Graphics than ggplot2 as well as the addition of the Interactive Grammar of Graphics, however the notation Vega-Lite is much more cumbersome to navigate. The reason being that Vega-Lite doesn't call specific variables at each step like ggplot2, so we are left we a lot of brackets, parentheses, and quotation marks. Additionally, Vega-Lite has re-categorized and renamed the classes of the grammar, and this makes unneeded complications, while ggplot2 still calls the class names at each step. Therefore, even though ggplot2 skips some grammar classes, we find that is more true to the original Grammar of Graphics.

Additions to the Minimal Grammar

Throughout this paper, we discussed and defined the seven orthogonal classes of the Grammar of Graphics, the minimal amount of language required to make a graphic, and two programs that incorporate the Grammar of Graphics. Recall, we defined the minimum cases to form a statistical graphic as the Statistic, Geometry, and Aesthetics classes, however this incredibly

limiting, so we should also include the Coordinate class to specify the plotting dimensions. With these four classes, we can now define this as our minimal grammar.

Building off of Wickham (2010), the first addition to the minimal grammar should be the inclusion of layers. Since layers are combination of Statistic and Geometry classes with the same specified Coordinate class, we should add a Layer class after Geometry that allows us to repeat through the Statistic and Geometry classes to continually create overlaying plots and then simultaneously picking a unified Coordinate class. This useful for showing additional information and trends, such as overlaying predictive models on raw data. It allows for comparisons of several like things on the same rendered graphed. Layers is an essential tool for combing and overlaying like graphs. Layer are seen incorporated by several different graphical software programs since ggplot2 and on (Morel, 2018; Satyanarayan *et al*, 2017; Waksom, 2021; Wickham 2010).

References

Data visualization with ggplot2 :: Cheat Sheet. (2024). Rstudio.github.io. https://rstudio.github.io/cheatsheets/visualization.html?_gl=1

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Hadley Wickham. (2009). *ggplot2*. New York, Ny Springer New York.

Introduction to Vega-Lite. (2024). Vega-Lite. https://vega.github.io/vega-lite/tutorials/getting_started.html

Morel, P. (2018). Gramm: grammar of graphics plotting in Matlab. *The Journal of Open Source Software*, *3*(23), 568. https://doi.org/10.21105/joss.00568

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., & Heer, J. (2017). Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 341–350. https://doi.org/10.1109/tvcg.2016.2599030

Uebe, G. (2007). Wilkinson, L.: The Grammar of Graphics. *AStA Advances in Statistical Analysis*, *91*(2), 221–222. https://doi.org/10.1007/s10182-007-0028-z

Waskom, M. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://joss.theoj.org/papers/10.21105/joss.03021

Wickham, H. (2010). A Layered Grammar of Graphics. *Journal of Computational and Graphical Statistics*, *19*(1), 3–28. https://doi.org/10.1198/jcgs.2009.07098