

A Critical Examination of the Grammar of Graphics

Tyler Wiederich

September 30, 2024

This paper explores the philosophy and applications of the Grammar of Graphics (Wilkinson 2010). The Grammar of Graphics uses seven orthogonal classes to deconstruct the process of creating statistical graphics. Through evidence-based reasoning, the classes can be reduced to a minimally sufficient set of features. I will then propose new additions to the minimum set of features for a more robust grammar for creating statistical graphics.

Outline

Introduction

- Definition of GoG
- Philosophy
- Why GoG should be considered

Implementation

- Minimum required specs for plots
- Paragraph of each step and whether or not it is essential

VegaLite

-

ggplot2

-

New Grammar

-

- Proofreading (check that graph is good)

Introduction

Data visualizations are an essential method in conveying information (Tukey 1965). When presented with data, these graphics provide faster comprehension, albeit slightly less accurate, than data presented in a table or text format (Meyer, Shinar, and Leiser 1997; Prasad and Ojha 2012; Brewer et al. 2012). The computer boom in the later half of the twentieth century allowed for new innovations in graphics, along with the ability to create graphics faster and more easily (Tukey 1965; Beniger and Robyn 1978). The importance of data visualization is evident from its widespread use in science and media, which means that creating these graphics requires careful considerations to display meaningful information.

However, the mapping of data to the visual space inherently has creative freedom which results in nearly infinite representations. An example of this is produced in Figure 1, where the same bivariate data is represented in three different forms. Good practices have developed over the years for producing readable and effective graphics (Tuft 1983; Kosslyn 1989; Cleveland and McGill 1984; Heer and Bostock 2010). A byproduct of graphics research is the recommendations for the forms for which graphics can take. Most modern graphical software packages, such as `ggplot2` (Wickham 2016) from R (R Core Team 2024), are able to create many types of graphics with minimal user effort. These packages inherently need to follow a well-designed process to produce statistical graphics.

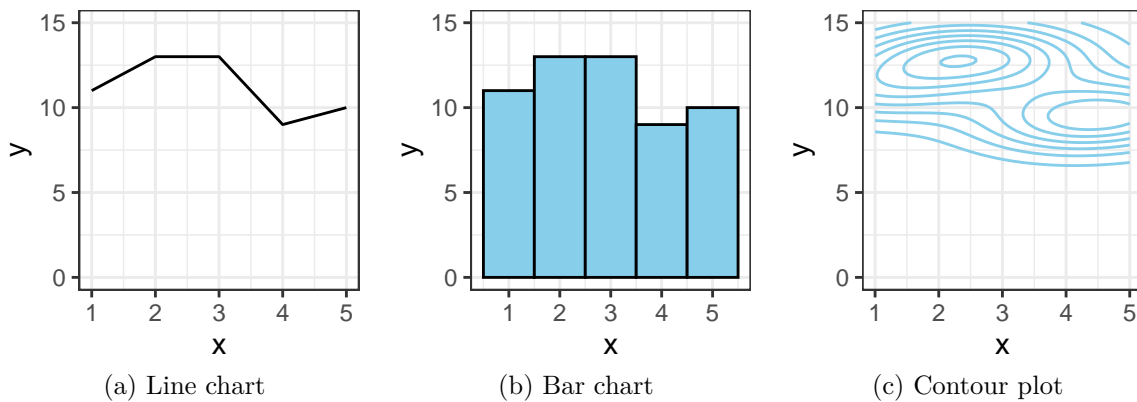


Figure 1: Three graphical representations of identical data. The line chart in (a) is useful for showing a trend in y across x . When x is categorical, as in (b), y can be compared across the categories of x . Lastly, the contour plot in (c) shows the bivariate density of x and y . Each graphic conveys information from different contexts of the data.

Since the graphical representation of data can take many forms, it is important to establish a set of rules, or grammar, to define the process of creating data visualizations. One such proposal was the Grammar of Graphics by Wilkinson (2010). The Grammar of Graphics defines the process as a sequential set of steps, where each step equally contributes to the rendered graphic. This process includes the following steps: variables, algebra, scales, statistics, geometries,

coordinates, and aesthetics. Taken together, these steps provide a system of creating graphics by reducing each step into its most basic form.

In this paper, I will discuss the Grammar of Graphics by Wilkinson (2010) and its applications in creating graphics. The grammar, while created with good intentions, has some philosophical flaws that I will address. These shortcomings are both in theory and application of the grammar, from which easy adaptations will allow it to functionally behave with existing graphics tools.

Grammar of Graphics

Wilkinson (2010) describes the Grammar of Graphics as a system of seven orthogonal classes. These classes produce an abstract sequential formulation for creating graphics. Wilkinson states nearly all statistical graphics can be created using the classes, but deviation from the classes or sequence can result in meaningless graphics. The Grammar of Graphics consist of the following seven classes: variables, algebra, scales, statistics, geometries, coordinates, and aesthetics.

Variables

Wilkinson (2010) states an important distinction between the definition of a *variable* and a *varset*. A *variable* is a mapping of objects to values, whereas a *varset* is a mapping of values to all possible ordered lists of objects. This distinction is claimed to simplify the process of algebra, although many graphics packages do not acknowledge the difference. The Graphics Production Language (GPL) is one such example of where the distinction is directly implemented (Wilkinson and Wills 2005), as well as at `graphic` package for Flutter (“Graphic | Flutter Package,” n.d.). In the case of GPL, the distinction allows for data sources to take abstract forms that do not limit graph types to specific data structures (Wilkinson et al. 2000).

Algebra

The algebra class is a set of operations for producing combinations of variables given varsets. This step consists of three operations: cross (*), nest (/), and blend (+). Cross is the horizontal, or column, concatenation. Nest, while similar to cross, is essentially a left join that creates a nested structure. Blend is the vertical concatenation, like append, that should be used only when the varsets have matching or compatible units. These operations allow for a restructuring of the data, but the ending result resembles tidy data (Wickham 2014).

Scales

Scales are transformations on the units of the varsets. These are done before statistical calculations to avoid issues with meaningless results. This is easy to see in the case of averaging logarithmic-transformed data, where given a set of observations x_i for $i = 1, \dots, n$

$$\log(\bar{X}) = \log\left(\frac{1}{n} \sum_{i=1}^n x_i\right) \neq \frac{1}{n} \sum_{i=1}^n \log(x_i) \quad x_i > 0$$

For example, the field of food science tends to use the logarithmic colony-forming units (log CFU) when measuring bacteria counts due to extremely large responses (Johnston et al. 2005; Abadias et al. 2008). The interest is in the log CFU units, so averages and other statistical calculations are performed on the log scale. Adjustments to the scales before calculating statistics helps to avoid misrepresented values.

Statistics

The statistics class takes a varset and outputs a summary varset. Some cases simply use the identity for its statistic, where each observation gets mapped. In other cases, summaries of the varsets can be calculated, such as counts, averages, predicted values from models (such as regression or clustering).

Geometries

The abstract mapping of data into the visual space is performed with the geometry class. This takes the output of the previous classes and arranges the form that their visual representation will take. This includes the geometric shapes of points, lines/paths, areas, bars/intervals, schemas (boxplots), polygons, contours, and edges. The geometry class produces geometric sets, but still requires further specification before it enters the visual space.

Coordinates

The coordinate class is how geometries are mapped into the visual space. The most common system is the Cartesian coordinate system, where up to three dimensions can be shown at a time. In most cases, only two dimensions are used since three dimensional representations provide less accurate comparisons (Croxtton and Stein 1932; Cleveland and McGill 1984; Barfield and Robless 1989; Fischer 2000; Hughes, n.d.). Polar coordinates are another common coordinate system when circular geometries are desired, such as pie charts.

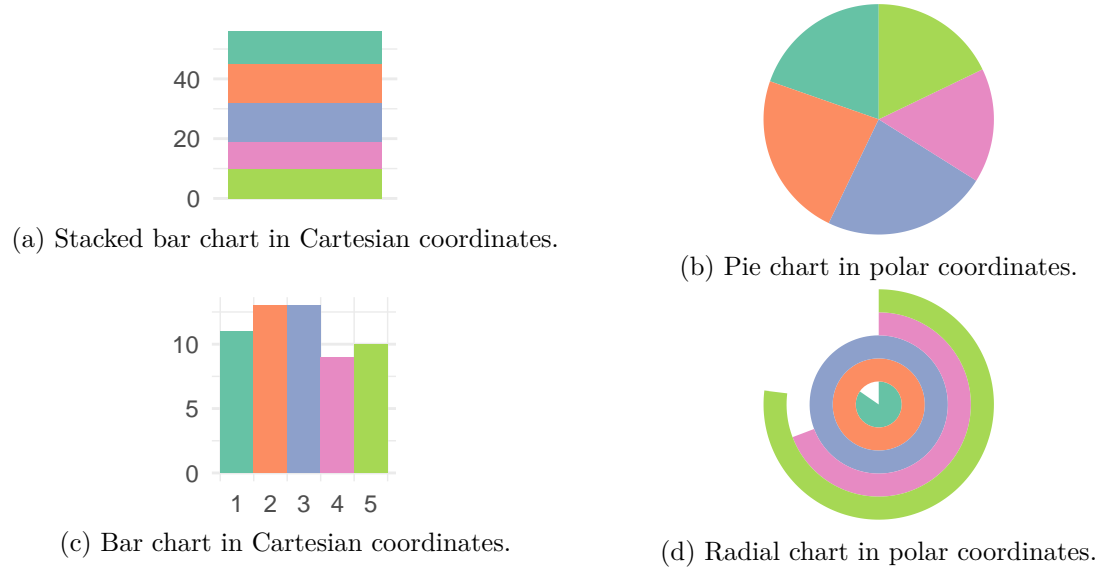


Figure 2: Two examples of bar charts in both Cartesian and polar coordinates. Figures (a) and (b) are identical with the exception of the coordinate system. Similarly for figures (c) and (d).

Aesthetics

The final class, aesthetics, is the actual mapping of geometries onto the coordinate space. This includes mapping varsets to the coordinates of the geometries and other components, such as color, shape, and size. Aesthetics can also include sound and motion, extending the Grammar of Graphics beyond a static visual space.

Philosophy

The purpose of the Grammar of Graphics (Wilkinson 2010) was to establish a mathematics-based foundation by using a series of mapping functions. Data is initially mapped to varsets before using set operations to join varsets together. These varsets then have direct numeric operations with the scales and statistics. Finally, the varsets are mapped into the visual space with geometries, coordinate systems, and aesthetics. The steps should compliment each other to produce other meaningful statistical graphics.

The ability to create statistical graphics using the Grammar of Graphics does not guarantee that the all permutations of classes produce meaningful graphics. For example, consider Figure 3 with the penguins (Horst, Hill, and Gorman 2020) dataset. Here, the Cartesian coordinate display of a contour plot is much easier to understand than the same data represented

in polar coordinates. While the Grammar of Graphics is capable of producing many types of charts, it does not provide guidance on what charts are useful for users.

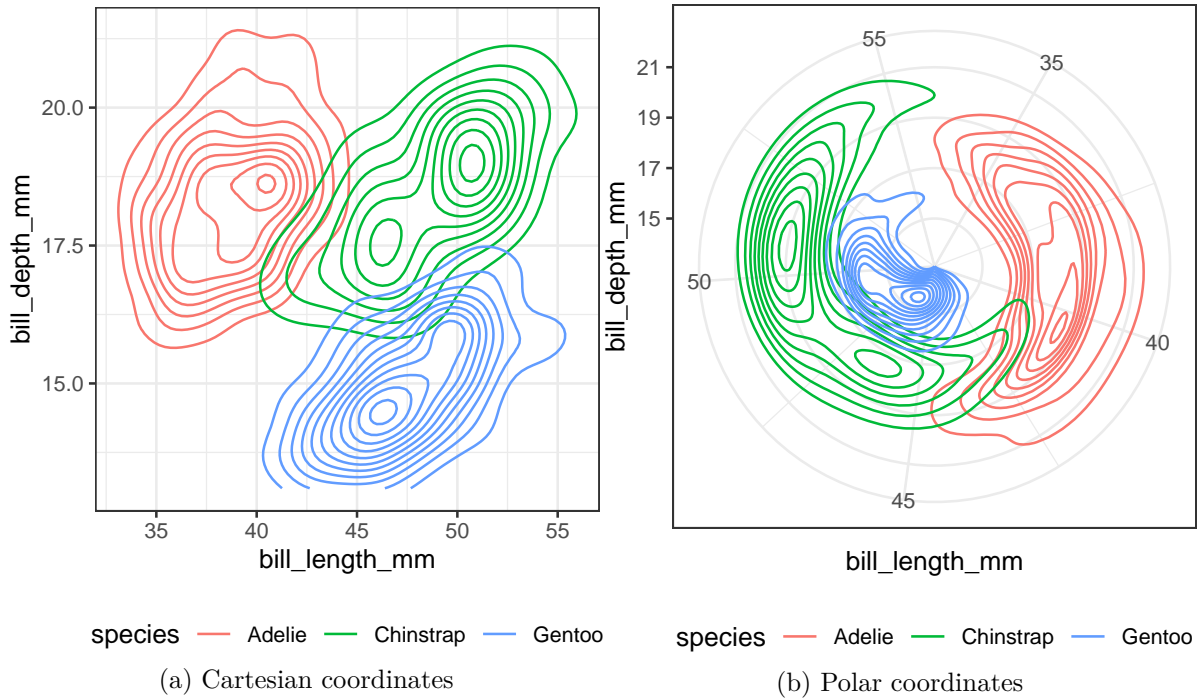


Figure 3: Contour plots of `penguins` dataset from the `palmerpenguins` package in R. The left panel is in Cartesian coordinates and the right panel is in polar coordinates. The polar coordinates are more difficult to understand than the Cartesian coordinates when extracting information about the different species.

Modern Visualization Tools

The classes contained within the Grammar of Graphics (Wilkinson 2010) are used in many modern graphics packages (Wickham 2016; Waskom 2021; Satyanarayan et al. 2017). These packages typically use similar terms found in the Grammar of Graphics, such as data, scales, geometry, coordinates, and aesthetics. Code examples are provided with Listing 1 and Listing 2. Using `ggplot2` with R, data is provided in the `ggplot()` function, and aesthetics are mapped using a dedicated `aes()` function. Here, the coordinate positions are included within the aesthetics along with color. Python’s `seaborn` module initially calls the geometry with `os.scatterplot()`, using function arguments to identify data, coordinate positions, and color. The plot from Listing 1 is produced in Figure 4.

The Grammar of Graphics was built around GPL, and thus needs adaptations for general use. Consider Listing 2. To create a graphic, all that was minimally required was data and

Listing 1 Code to produce scatterplot of penguins data using `ggplot2` in R.

```
library(palmerpenguins)
data("penguins")
ggplot(data = penguins, mapping = aes(x = bill_length_mm,
                                      y = bill_depth_mm,
                                      color = species)) +
  geom_point()
```

Listing 2 Code to produce scatterplot of penguins data using `seaborn` in Python.

```
import seaborn as os
os.scatterplot(data=r.penguins, x="bill_length_mm",
              y="bill_depth_mm",
              hue="species")
```

a geometry mapping to coordinates x and y (hue was not necessary to create the plot). The same can be said for Listing 1, although the geometry step is more explicit as its own step. A commonality is that both figures were produced with tidy data (Wickham 2014). However, most classes from the Grammar of Graphics are either used inherently or by default settings. Therefore, an adaptation of the Grammar of Graphics can be considered by redefining classes and their practicality in modern data visualizations.

Evaluation of the Seven Classes

Establishing variables from data is an essential step in the Grammar of Graphics. This is obvious from the fact that statistical graphics are the visual mapping of data. The only requirement from the variable class is to create varsets, the mapping of values to all possible ordered lists of objects. From a practical point of view, a variable is simply a $n \times 1$ vector of values that share the same data type.

The algebra step is the first class from the Grammar of Graphics that is implicitly performed by the data structure. For a single data set that is already in a tidy format (Wickham 2014), the algebra step can be ignored since no crossing, nesting, or blending is needed. However, data often is stored across multiple sources, such as in a relational database (Harrington 2016). The joining of data results in another dataset, which can then become the new data source. Consider Listing 3 using the `band_members` and `band_instruments` from the `dplyr` package (Wickham et al. 2023). Here, data is stored separately and then joined to create a new data object. Now considering the joined data as the new data source, the algebra step is no longer required. This shows that algebra is only an intermittent step in producing data.

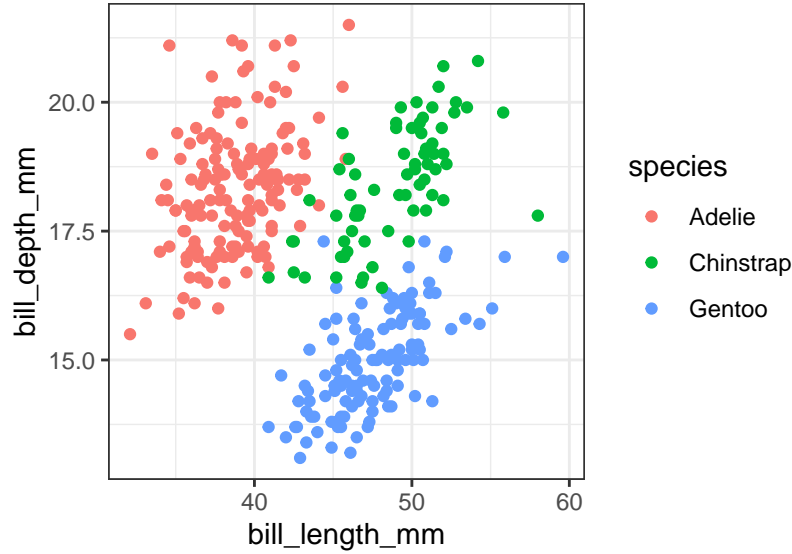


Figure 4: Scatterplot of `penguins` from Listing 1 using `ggplot2` in R.

Listing 3 Full join of `band_members` and `band_instruments` from the `dplyr` package in R. The joined data is stored into `band_info` object.

```
band_info <- dplyr::band_members %>%
  full_join(dplyr::band_instruments, by = 'name')
```

The next two classes, scales and statistics, follow a similar logic to that of the algebra class. For scales, a transformation onto a variable can be used to create a new dataset with the newly defined variable. Statistics follow a similar process, where summaries and models can be calculated and arranged into a tidy format. Listing 4 shows an example of a scale transformation and averages, where the resulting data is saved into a new data object.

Thus far, the classes of algebra, scales, and statistics are argued to be non-essential in the Grammar of Graphics framework. The reasoning is that these classes are able to result in a new dataset that foregoes the necessity to reevaluate the data aggregation steps. In reality, the application of these classes is more fluid than having a strict ordering. For example, the `magrittr` package (Bache and Wickham 2022) in R allows for piping, a “this-then-that” approach, where data does not need to be saved into a new object. Additionally, many graphical packages allow for transformations withing the geometry class. Consider Figure 5 with the `penguins` dataset (Horst, Hill, and Gorman 2020) where the binning statistic is used within the `geom_bar()` function. The binning could have taken place before introducing the data into the `ggplot()` function, but including the transformation with the geometry increases readability and makes it easier to create the graphic. It is not always necessary to calculate relevant transformations and/or statistics as their own steps, so they can be considered non-essential

Listing 4 Data aggregation using scales and statistics on the `penguins` dataset from the `palmerpenguins` package in R. Data is saved into a new data set called `penguins_summary`.

```
penguins_summary <- penguins %>%  
  mutate(bill_length_cm = bill_length_mm/10,  
         bill_depth_cm = bill_depth_mm/10) %>%  
  group_by(species, island, sex, year) %>%  
  summarise(avg_bill_length_cm = mean(bill_length_cm, na.rm = T),  
            avg_bill_depth_cm = mean(bill_depth_cm, na.rm = T))
```

classes of the Grammar of Graphics methodology.

```
ggplot(penguins, mapping = aes(x = bill_length_mm)) +  
  geom_bar(stat = 'bin', fill = 'skyblue', color = 'black') +  
  theme(aspect.ratio = 1/2)
```

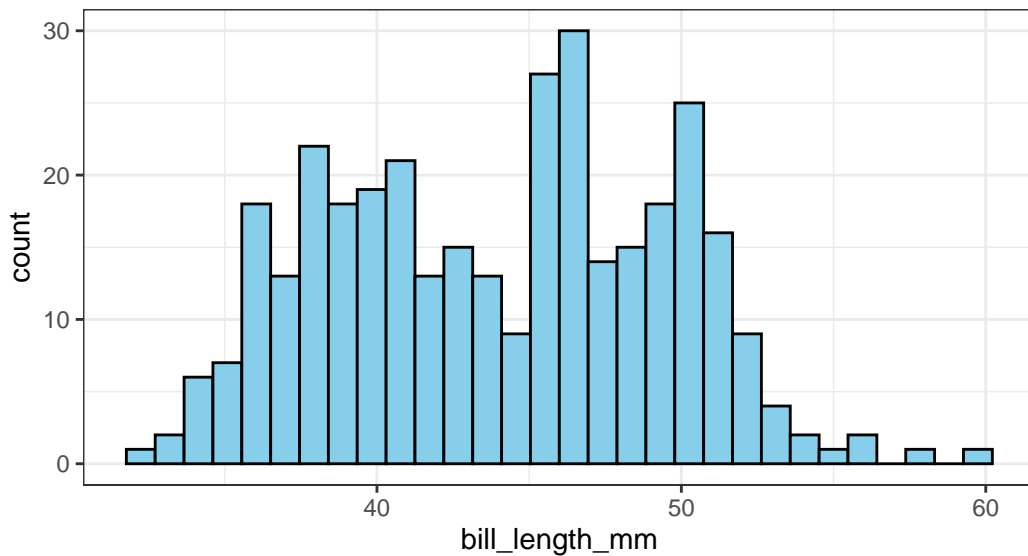


Figure 5: Histogram of bill length (mm) from the `penguin` dataset. Binning is performed inside the geometry of the bar chart.

The mapping of data into an abstract geometric space includes the classes of geometry and coordinates. The geometry class is essential in defining the visual form of the data, which is the key point of making graphics. However, the placement of geometry requires a coordinate system, which makes the coordinate class inherently essential. Given the limitations of the physical world and computers, displays of dimensional data are restricted to a maximum of

three dimensions. The standard Cartesian coordinates have been around since at least the 17th century, making them an ideal default system (Vince 2010, chap. 5). The geometry and coordinate classes work in tandem to create the form that graphic will use in its final product.

The final class is aesthetics. By technical definitions, mapping data to aesthetics is not necessary in producing charts, with the exception of the coordinates of the geometries. Figure 5 is one such example where only geometries are mapped to locations. However, it is often useful to encode additional information through one or more aesthetics (Wilkinson and Wills 2005). Figure 6 shows separate tasks from aesthetics, where only geometry mapping is essential in creating a graphic. Aesthetics are

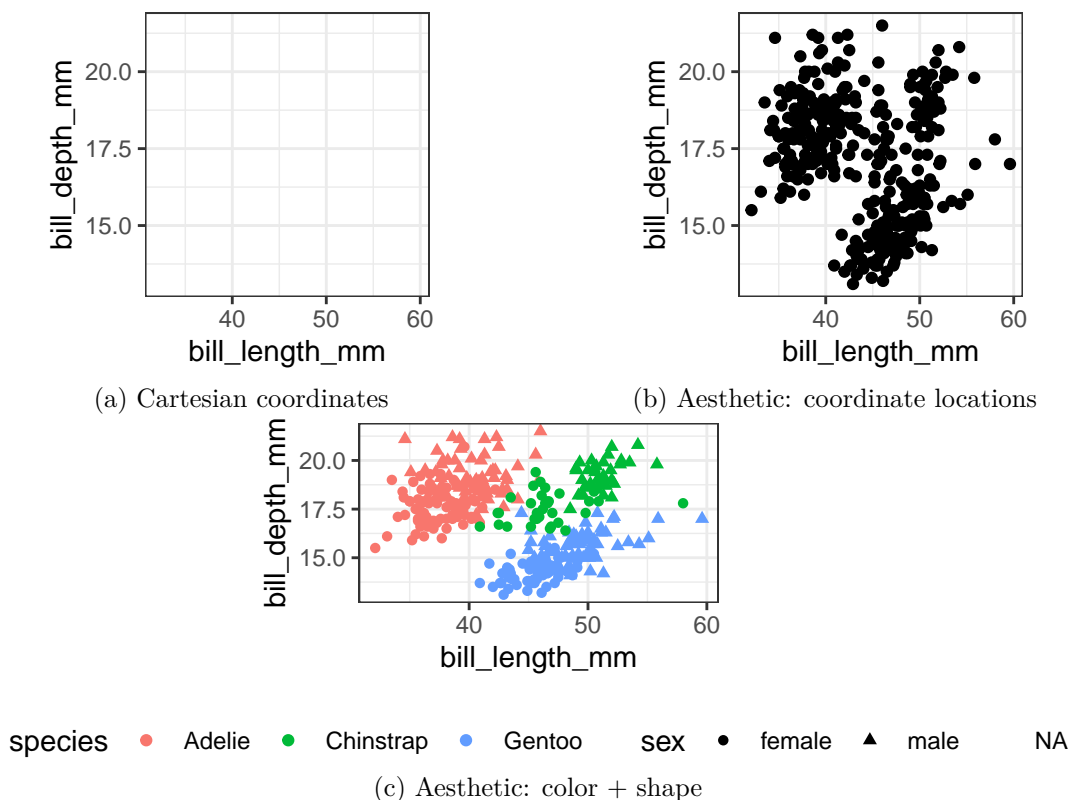


Figure 6: Demonstration of each visual class from the Grammar of Graphics. In (a), a coordinate grid is created based on the identify statistic of bill length (mm) and bill depth (mm). Panel (b) uses the point geometry to map the variable pairs onto the coordinate space. Panel (d) map species to color and sex to shape. Only (b) was required to create a graphic from data, but (c) adds additional context.

Two Applications of Essential Classes

In this section, I will evaluate the graphics software of ggplot2 in R (Wickham 2010) and VegaLite in JavaScript (Satyanarayan et al. 2017). The ggplot2 package was created from the Grammar of Graphics philosophy, using a layering approach navigate through the classes. VegaLite was similarly created using the Grammar of Graphics, but opted to use differing terminology and incorporate interactive elements. Each graphics package has its own implementation of the Grammar of Graphics and the minimally required classes.

ggplot2

- Example with minimal classes
-

VegaLite

VegaLite uses *units* to define graphics, which consists of three required classes: data, mark-types, and encodings. Data follows the same form as tidy data (Wickham 2014) and is clearly an essential component required from the Grammar of Graphics. Mark-types define the geometry of the data before using the visual space. Lastly, encodings are the visual space of geometry location and other features, such as color and shape.

Comparing ggplot2 and VegaLite

- Main separating factor is layers vs. JSON
- Terminology
- Extent (static vs. interactive)

A New Grammar

- Algebra, scales, and statistics all can form new data
- Example with exp(normal) data, use model to get data from model before plotting
- Geometry, coordinates, and aesthetics all used together
- Need to have readable graph, so use

Conclusion

References

- Abadias, M., J. Usall, M. Anguera, C. Solsona, and I. Viñas. 2008. “Microbiological Quality of Fresh, Minimally-Processed Fruit and Vegetables, and Sprouts from Retail Establishments.” *International Journal of Food Microbiology* 123 (1-2): 121–29. <https://doi.org/10.1016/j.ijfoodmicro.2007.12.013>.
- Bache, Stefan Milton, and Hadley Wickham. 2022. *Magrittr: A Forward-Pipe Operator for r*. <https://CRAN.R-project.org/package=magrittr>.
- Barfield, Woodrow, and Robert Robless. 1989. “The Effects of Two- or Three-Dimensional Graphics on the Problem-Solving Performance of Experienced and Novice Decision Makers.” *Behaviour & Information Technology* 8 (5): 369–85. <https://doi.org/10.1080/01449298908914567>.
- Beniger, James R., and Dorothy L. Robyn. 1978. “Quantitative Graphics in Statistics: A Brief History.” *The American Statistician* 32 (1): 1–11. <https://doi.org/10.2307/2683467>.
- Brewer, Noel T., Melissa B. Gilkey, Sarah E. Lillie, Bradford W. Hesse, and Stacey L. Sheridan. 2012. “Tables or Bar Graphs? Presenting Test Results in Electronic Medical Records.” *Medical Decision Making* 32 (4): 545–53. <https://doi.org/10.1177/0272989X12441395>.
- Cleveland, William S., and Robert McGill. 1984. “Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods.” *Journal of the American Statistical Association* 79 (387): 531–54. <https://doi.org/10.1080/01621459.1984.10478080>.
- Croxtan, Frederick E., and Harold Stein. 1932. “Graphic Comparisons by Bars, Squares, Circles, and Cubes.” *Journal of the American Statistical Association* 27 (177): 54–60. <https://doi.org/10.1080/01621459.1932.10503227>.
- Fischer, Martin H. 2000. “Do Irrelevant Depth Cues Affect the Comprehension of Bar Graphs?” *Applied Cognitive Psychology* 14 (2): 151–62. [https://doi.org/10.1002/\(SICI\)1099-0720\(200003/04\)14:2%3C151::AID-ACP629%3E3.0.CO;2-Z](https://doi.org/10.1002/(SICI)1099-0720(200003/04)14:2%3C151::AID-ACP629%3E3.0.CO;2-Z).
- “Graphic | Flutter Package.” n.d. <https://pub.dev/packages/graphic>.
- Harrington, Jan L. 2016. *Relational Database Design and Implementation*. 4th ed. Amsterdam, [Netherlands: Morgan Kaufmann.
- Heer, Jeffrey, and Michael Bostock. 2010. “CHI ’10: CHI Conference on Human Factors in Computing Systems.” In, 203–12. Atlanta Georgia USA: ACM. <https://doi.org/10.1145/1753326.1753357>.
- Horst, Allison Marie, Alison Presmanes Hill, and Kristen B Gorman. 2020. *Palmerpenquins: Palmer Archipelago (Antarctica) Penguin Data*. <https://doi.org/10.5281/zenodo.3960218>.
- Hughes, Brian M. n.d. “Just Noticeable Differences in 2D and 3D Bar Charts: A Psychophysical Analysis of Chart Readability.”
- Johnston, Lynette M., Lee-Ann Jaykus, Deborah Moll, Martha C. Martinez, Juan Anciso, Brenda Mora, and Christine L. Moe. 2005. “A Field Study of the Microbiological Quality of Fresh Produce.” *Journal of Food Protection* 68 (9): 1840–47. <https://doi.org/10.4315/0362-028X-68.9.1840>.
- Kosslyn, Stephen M. 1989. “Understanding Charts and Graphs.” *Applied Cognitive Psychology*

- 3 (3): 185225. <https://doi.org/10.1002/acp.2350030302>.
- Meyer, Joachim, David Shinar, and David Leiser. 1997. “Multiple Factors That Determine Performance with Tables and Graphs.” *Human Factors: The Journal of the Human Factors and Ergonomics Society* 39 (2): 268–86. <https://doi.org/10.1518/001872097778543921>.
- Prasad, Gollapudi V. R. J. Sai, and Amitash Ojha. 2012. “2012 IEEE Fourth International Conference on Technology for Education (T4E).” In, 126–31. Hyderabad, India: IEEE. <https://doi.org/10.1109/T4E.2012.18>.
- R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.
- Satyanarayan, Arvind, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. “Vega-Lite: A Grammar of Interactive Graphics.” *IEEE Transactions on Visualization and Computer Graphics* 23 (1): 341–50. <https://doi.org/10.1109/TVCG.2016.2599030>.
- Tufte, Edward R. 1983. *The Visual Display of Quantitative Information*. Cheshire, Conn. (Box 430, Cheshire 06410): Graphics Press.
- Tukey, John W. 1965. “The Technical Tools of Statistics.” *The American Statistician* 19 (2): 23–28. <https://doi.org/10.2307/2682374>.
- Vince, John. 2010. *Mathematics for Computer Graphics*. Undergraduate Topics in Computer Science. London: Springer London. <https://doi.org/10.1007/978-1-84996-023-6>.
- Waskom, Michael L. 2021. “Seaborn: Statistical Data Visualization.” *Journal of Open Source Software* 6 (60): 3021. <https://doi.org/10.21105/joss.03021>.
- Wickham, Hadley. 2010. “A Layered Grammar of Graphics.” *Journal of Computational and Graphical Statistics* 19 (1): 3–28. <https://www.jstor.org/stable/25651297>.
- . 2014. “Tidy Data.” *Journal of Statistical Software* 59 (10): 123. <https://doi.org/10.18637/jss.v059.i10>.
- . 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.
- Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. <https://CRAN.R-project.org/package=dplyr>.
- Wilkinson, Leland. 2010. “The Grammar of Graphics.” *WIREs Computational Statistics* 2 (6): 673–77. <https://doi.org/10.1002/wics.118>.
- Wilkinson, Leland, Daniel J. Rope, Daniel B. Carr, and Matthew A. Rubin. 2000. “The Language of Graphics.” *Journal of Computational and Graphical Statistics* 9 (3): 530–43. <https://doi.org/10.1080/10618600.2000.10474897>.
- Wilkinson, Leland, and Graham Wills. 2005. *The Grammar of Graphics*. 2nd ed. Statistics and Computing. New York: Springer.