LSC 563 Lecture 6: Introduction to graphs and graphing relationships

Doug Joubert

2021-10-22

Table of Contents

[Learning Objectives 2](#_Toc84072679)

[Semiotics of Graphics 2](#_Toc84072680)

[Coordinate systems and axes 3](#_Toc84072681)

[Cartesian Coordinates 3](#_Toc84072682)

[Nonlinear Axes and Log Transformations 5](#_Toc84072683)

[Coordinate Systems with Curved Axes 7](#_Toc84072684)

[Introduction to Graphs 7](#_Toc84072685)

[Elements of a Graph 8](#_Toc84072686)

[A Brief History of Graphs 8](#_Toc84072687)

[When to Use Graphs 9](#_Toc84072688)

[Visual coding of graphs 9](#_Toc84072689)

[Marks 9](#_Toc84072690)

[Points 10](#_Toc84072691)

[Lines 11](#_Toc84072692)

[Area 12](#_Toc84072693)

[Channels 16](#_Toc84072694)

[Position 17](#_Toc84072695)

[Shape 18](#_Toc84072696)

[Size 19](#_Toc84072697)

[Angle 19](#_Toc84072698)

[Color 21](#_Toc84072699)

[Making a Plot 21](#_Toc84072700)

[How ggplot Works 21](#_Toc84072701)

[Mappings Link Data to Things You See 24](#_Toc84072702)

[Build Your Plots Layer by Layer 28](#_Toc84072703)

[Mapping Aesthetics vs Setting Them 30](#_Toc84072704)

[Aesthetics Can Be Mapped per Geom 33](#_Toc84072705)

[Saving Your Work 38](#_Toc84072706)

[Quick Overview of File Formats 38](#_Toc84072707)

[Saving plots using ggplot 38](#_Toc84072708)

[Commit and push the changes to GitHub 39](#_Toc84072709)

# Learning Objectives

By the end of this section, students should be able to:

1. Discuss the connection between data, aesthetics, & the grammar of graphics
2. Outline the grammatical layers in ggplot
3. Label the basic elements of a graph
4. List the three primary marks used in data visualization
5. List the 5 channels discussed during lesson 4
6. Distinguish between “viewing” and graph in R and saving a graph in R.

# Semiotics of Graphics

Semiotics is the study of symbols and how they convey meaning. In his great masterwork, Semiology of Graphics, Bertin (1983) attempted to classify all graphic marks in terms of how they could express data.

For the most part, this work is based on his own judgment, although it is a highly trained and sensitive judgment. There are few references to theories of perception or scientific studies.

Figure 6.1 shows examples of languages that have some claim to being visual.

The first example of visual language is based on a cave painting. We can readily interpret that these are animal.

The second example is a schematic diagram showing the interaction between lab equipment and some sort of medium.

The third example is the expression of a mathematical equation that is utterly obscure to all but the initiated.

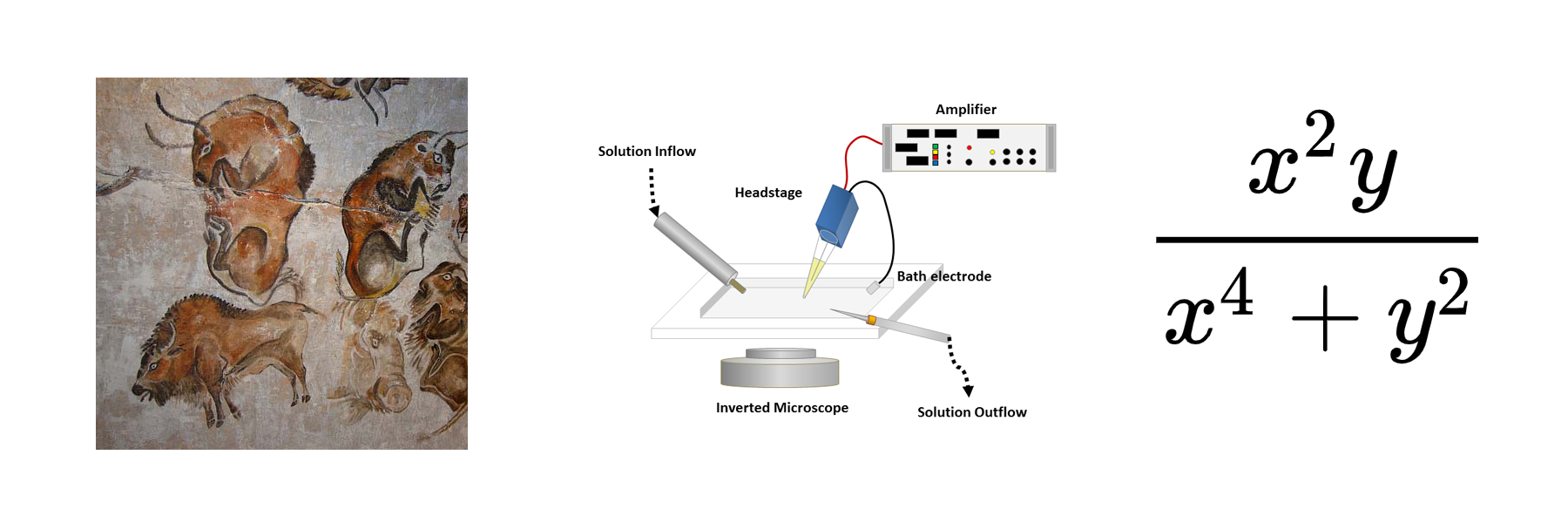


Figure 6.1: Examples of different “languages” (Ware, 2019a).

These examples clearly show that some visual languages are easier to “read” than others. But why?:

* Perhaps it is simply that we have more experience with the kind of pictorial image shown in the cave painting and less with the mathematical notation.
* Perhaps the concepts expressed in the cave painting are more familiar than those in the equation.

# Coordinate systems and axes

To make any sort of data visualization, we need to define position scales, which determine where in a graphic different data values are located (Wilke, 2019). We cannot visualize data without placing different data points at different locations, even if we just arrange them next to each other along a line.

In 2D visualizations, two numbers are required to uniquely specify a point, and therefore we need two position scales (Wilke, 2019). These two scales are usually the x and y axes of the plot (Wilke, 2019). We also have to specify the relative geometric arrangement of these scales (Wilke, 2019).

## Cartesian Coordinates

The most widely used coordinate system for data visualization is the 2D Cartesian coordinate system, where each location is uniquely specified by an x and a y value (Wilke, 2019). As illustrated in Figure 6.2, the data values are placed in an even spacing along both axes, the two axes are continuous position scales, and they can represent both positive and negative real numbers (Wilke, 2019).

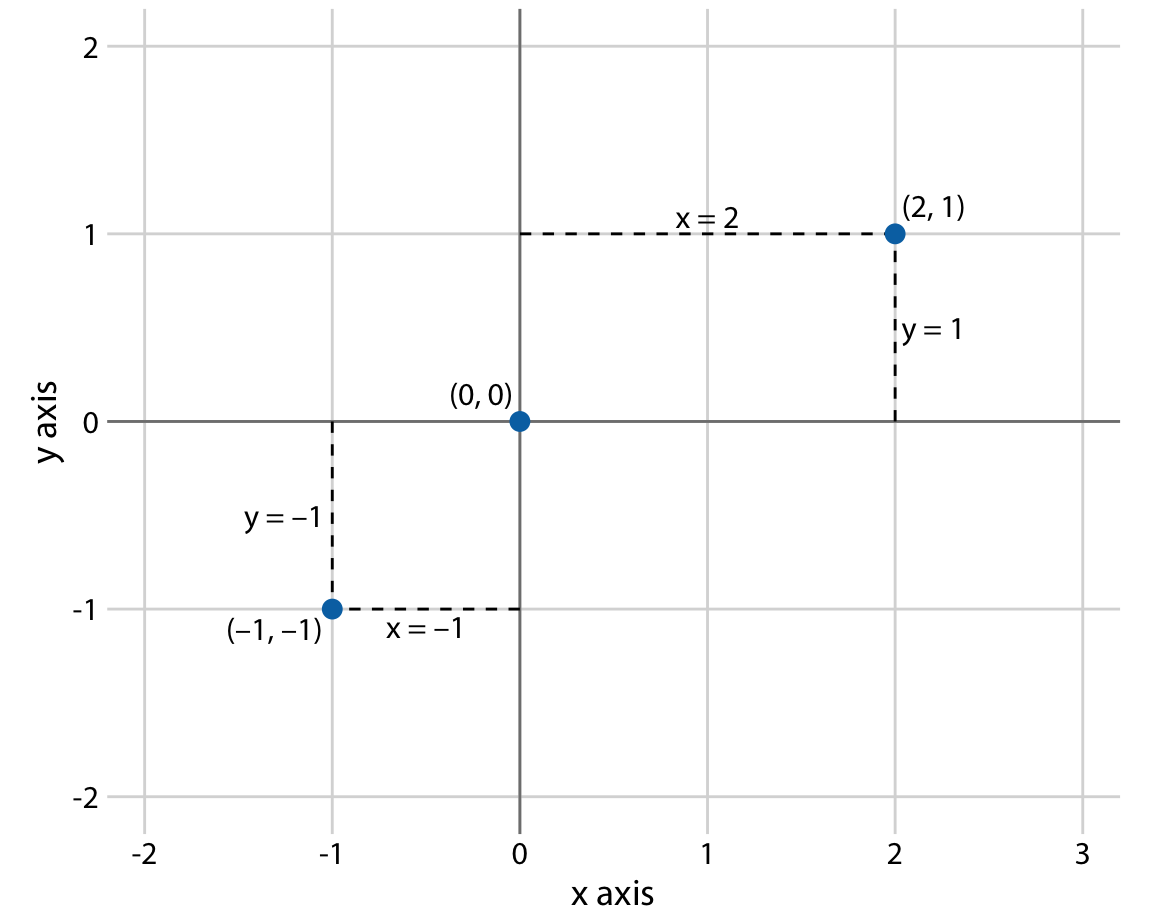


Figure 6.2: Standard Cartesian coordinate system [Wilke (2019).

In Figure 6.2, both the x and the y grid lines are separated by units of one, and the x axis runs from -2.2 to 3.2 and the y axis runs from -2.2 to 2.2 (Wilke 2019). Any data values between these axis limits are placed at the respective location in the plot. Any data values outside the axis limits are discarded (Wilke, 2019).

It is important to remember that data values usually aren’t just numbers (Wilke 2019). They come with units (Wilke, 2019). For example, if we’re measuring temperature, the values may be measured in degrees Celsius or Fahrenheit (Wilke 2019). Similarly, if we’re measuring distance, the values may be measured in kilometers or miles (Wilke 2019).

You may wonder what happens if you change the units of your data. A change in units is a linear transformation, where we add or subtract a number to or from all data values and/or multiply all data values with another number.

Therefore, you can change the units of your data and the resulting figure will not change if you change the axes accordingly. As an example, compare how the data is displayed in Figures L4.3 (a) and L4.3 (b).

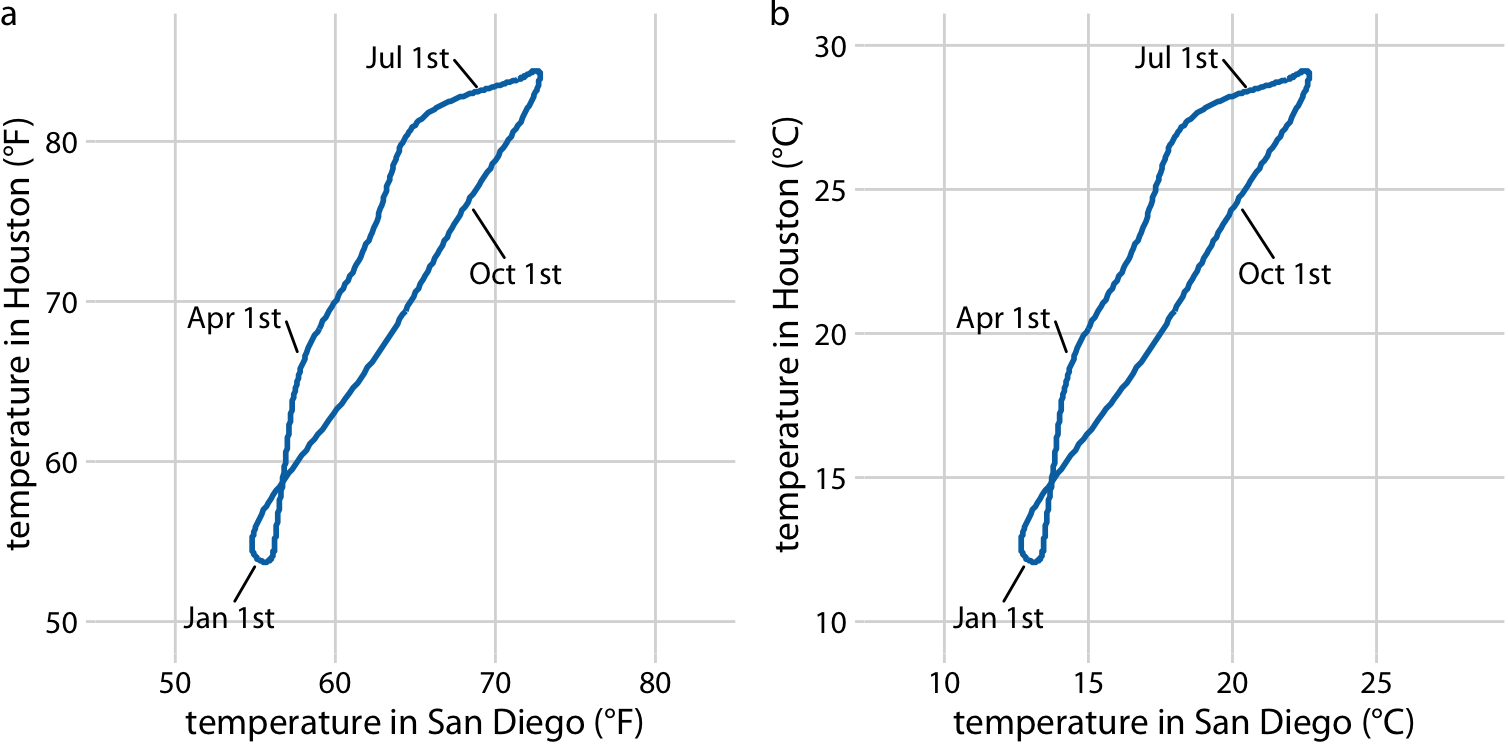


Figure 6.3: Daily (normal) temperature for Houston, TX, plotted versus the (normal) temperature of San Diego, CA (Wilke, 2019). Data source: NOAA.

## Nonlinear Axes and Log Transformations

In a Cartesian coordinate system, the grid lines along an axis are spaced evenly both in data units and in the resulting visualization. We refer to the position scales in these coordinate systems as linear (Wilke, 2019).

There are scenarios where nonlinear scales are preferred. The most used nonlinear scale is the logarithmic scale, or log scale for short (Wilke, 2019). Log scales are linear in multiplication, such that a unit step on the scale corresponds to multiplication with a fixed value (Wilke, 2019).

To create a log scale, we need to log-transform the data values while exponentiating the numbers that are shown along the axis grid lines (Wilke, 2019). Importantly, the correct axis title for a logarithmic scale is the name of the variable shown, not the logarithm of that variable.

In most cases, the labeling for a logarithmic scale is preferable, because it places less mental burden on the reader to interpret the numbers shown as the axis tick labels (Wilke, 2019).

It is always recommended that you verify the base when working with log-transformed data (Wilke, 2019). When plotting log-transformed data, always specify the base in the labeling of the axis, to avoid confusion.

Log scales are the natural choice for any data that has been obtained by multiplication or division. For example, ratios should generally be shown on a log scale (Wilke, 2019). As an example, I have taken the number of inhabitants in each county in Texas and divided it by the median number of inhabitants across all Texas counties (Figure L5.14).

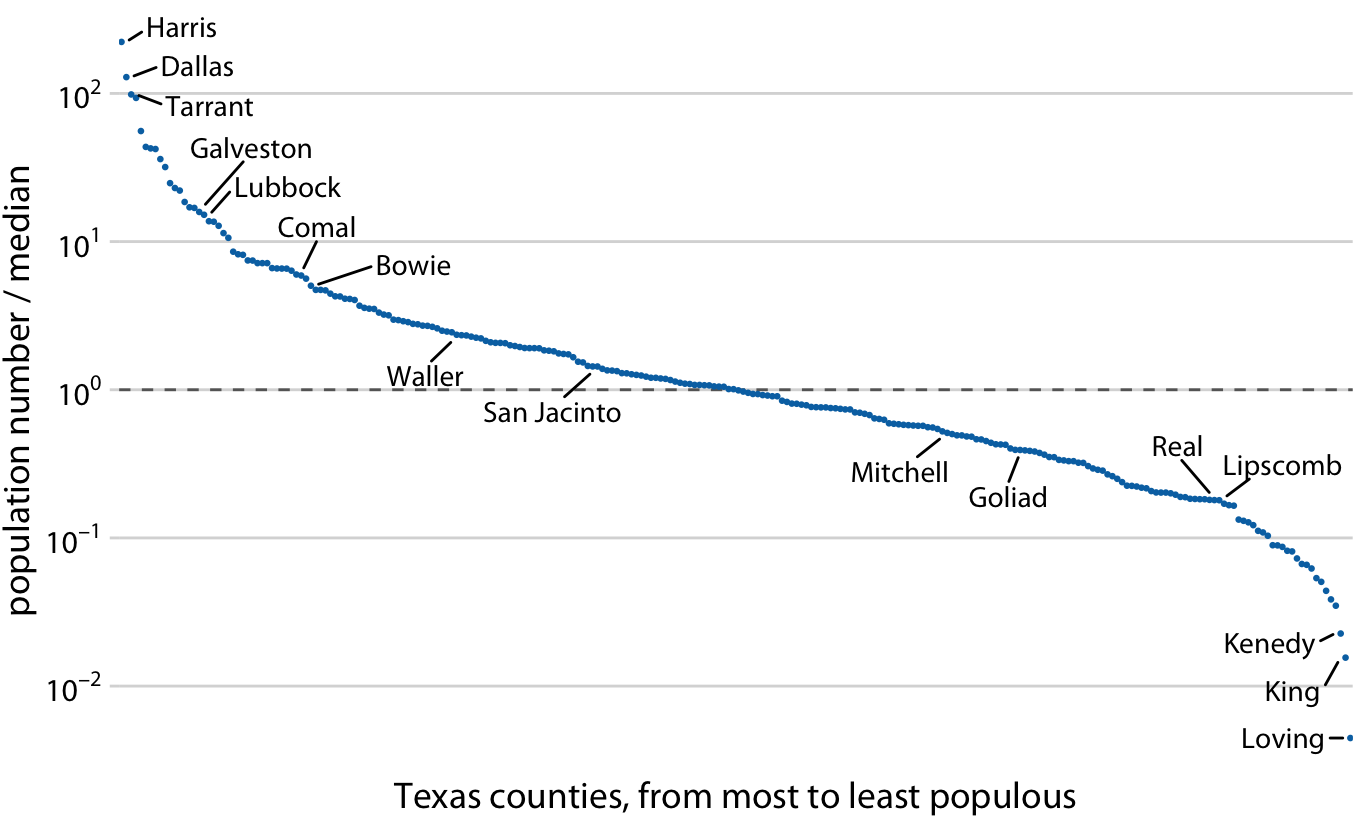


Figure Figure 6.4: Texas counties, from most to least populous using a log scale 10 transformation (Wilke, 2019).

Look how different the same data appears when graph using a linear scale. Using a linear scale obscures the differences between a county with median population number and a county with a much smaller population number than median (Figure 6.5) (Wilke, 2019).

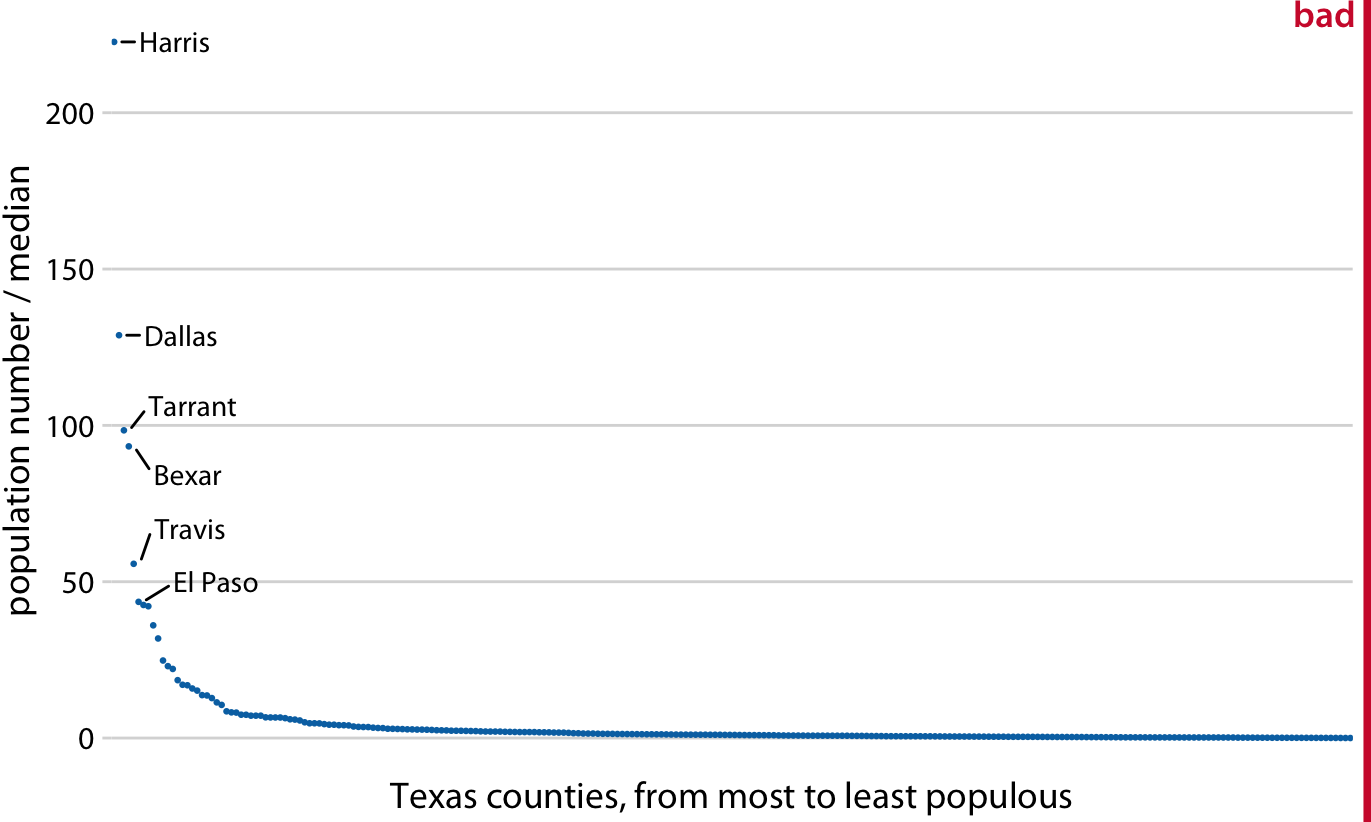


Figure 6.5: Texas counties, from most to least populous, using a linear transformation (Wilke, 2019).

Log scales are frequently used when the dataset contains numbers of very different magnitudes (Wilke, 2019). For the Texas counties shown in Figures L4.4 and L4.5, the most populous one (Harris) had 4,092,459 inhabitants in the 2010 US Census while the least populous one (Loving) had 82 (Wilke, 2019). So, a log scale would be appropriate even if we hadn’t divided the population numbers by their median to turn them into ratios (Wilke, 2019).

## Coordinate Systems with Curved Axes

All the coordinate systems we have encountered so far have used two straight axes positioned at a right angle to each other (Wilke, 2019). There are other coordinate systems, where the axes themselves are curved (Wilke, 2019). In particular, in the polar coordinate system, we specify positions via an angle and a radial distance from the origin (Figure 6.6) (Wilke, 2019).

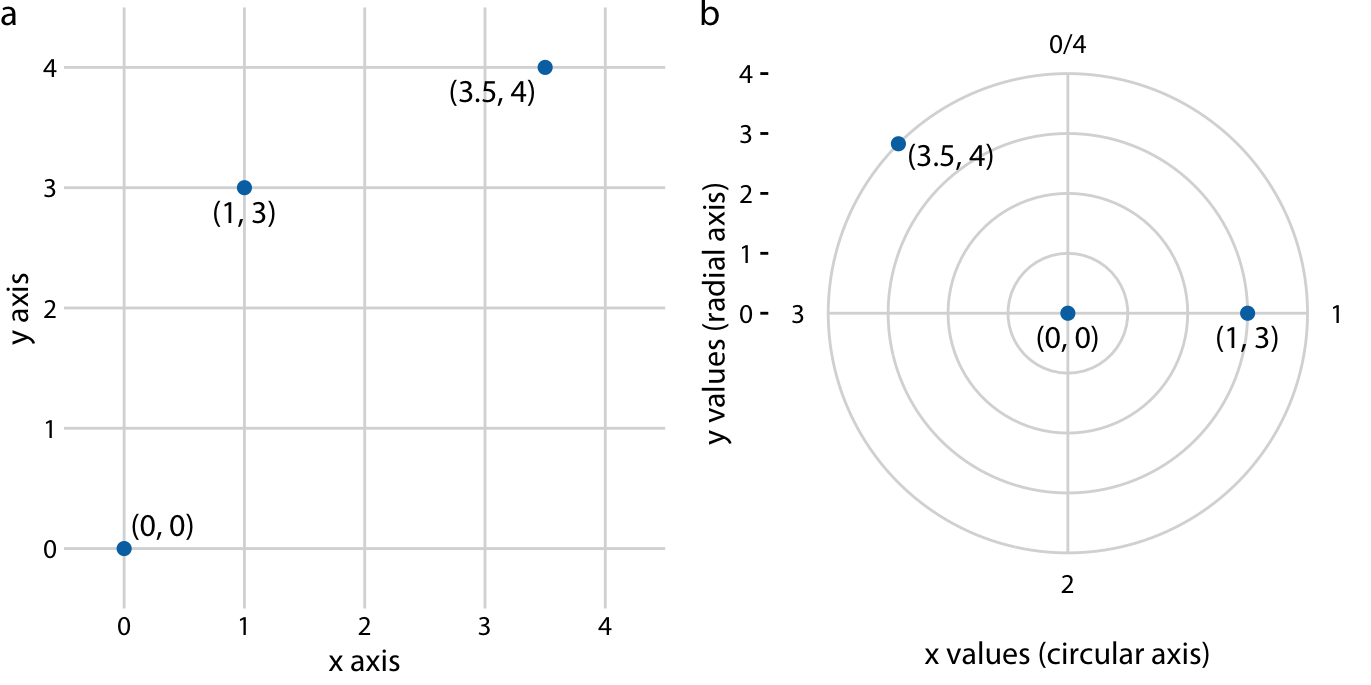


Figure 6.6: Relationship between Cartesian and polar coordinates (Wilke, 2019).

In Figure 6.6, the three data points shown in Cartesian coordinate system (a) are compared to the same three data points shown in a polar coordinate system (b) (Wilke, 2019). A radial layout is used to display the data. A common example is pie charts. Radial layouts will be covered the parts-to-wholes lecture.

A second setting in which we encounter curved axes is in the context of geospatial data (Wilke, 2019). We will be talking about geospatial data in a later lecture.

# Introduction to Graphs

A graph is a visual display of quantitative information. Graphs exhibit the following characteristics (Few, 2012a):

* Values are displayed within an area delineated by one or more axes
* Values are encoded as visual objects positioned in relation to the axes
* Axes provide scales (quantitative and categorical) that are used to label and assign values to the visual objects. For clarification, axes delineate the space that is used to display data in a graph.

## Elements of a Graph

Figure 6.7 is an example of a simple graph, which displays the sum of sales and shipping costs.

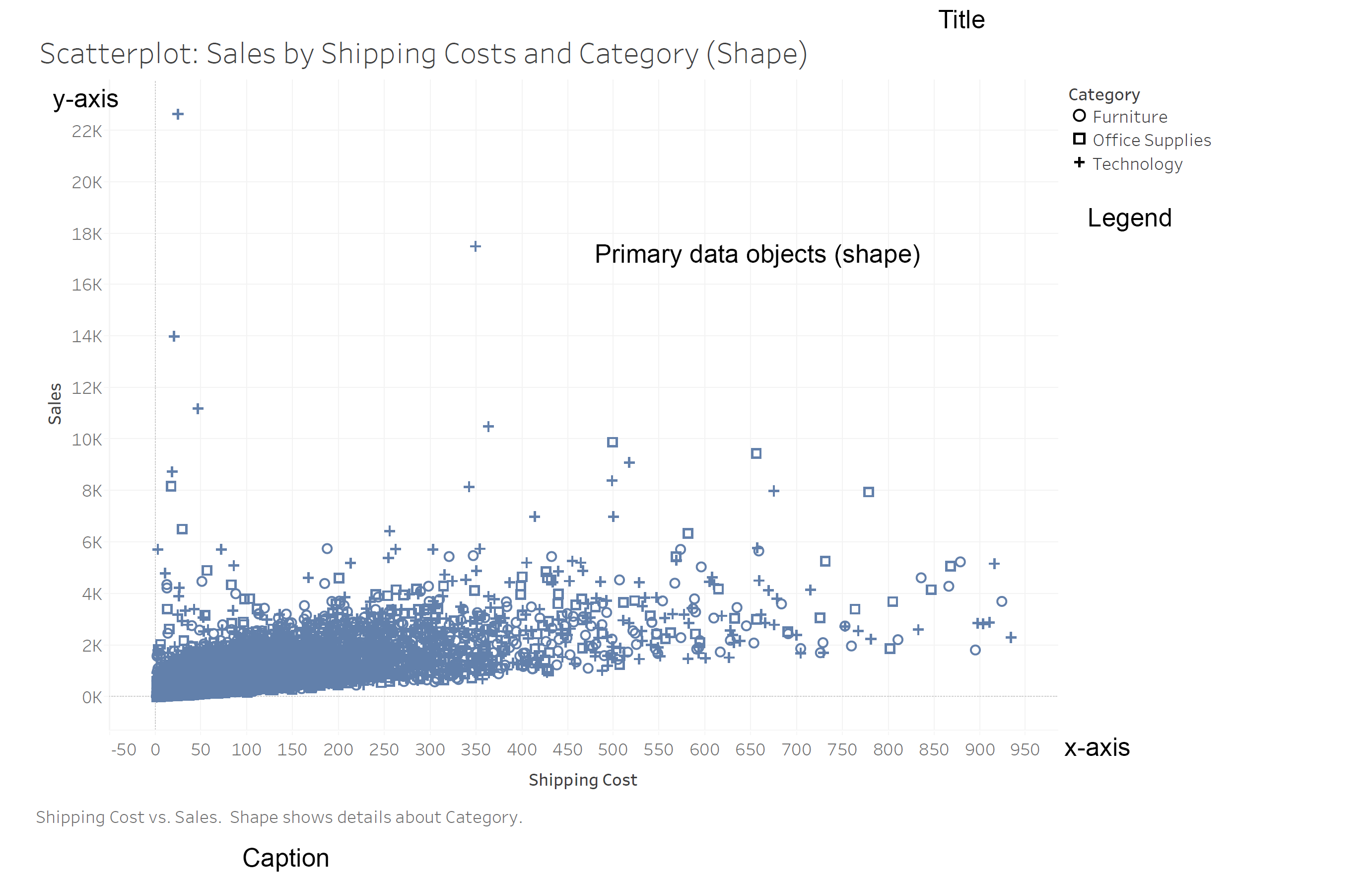


Figure 6.7: Sum of Sales and Shipping Costs by Category.

This graph in Figure 6.7 has two axes: one that runs horizontally, called the X axis, and one that runs vertically, called the Y axis. In this graph, the quantitative scale, which labels the costs associated with shipping a product, resides along the X axis, and the quantitative scale, which represents the sum of sales resides along the Y axis. Shape is used to divide these values by the category of the product. Shape in this instance is a channel, one of the two major visual components of graphs. More about that below.

## Contextual Graph Components

There are other visual components that are important to consider. However, we will be discussing them later in the semester. These are:

* Annotations
* Axis
* Grids
* Labels
* Legends
* Reference Lines

## A Brief History of Graphs

Please see the following journal article for a wonderful overview of the history of visualizations (Friendly, 2008).

## When to Use Graphs

Graphs allow us to visualize the individual values in the dataset and also highlight the overall shape of the data. Let us see this in action by loading some data and then graphing that data.

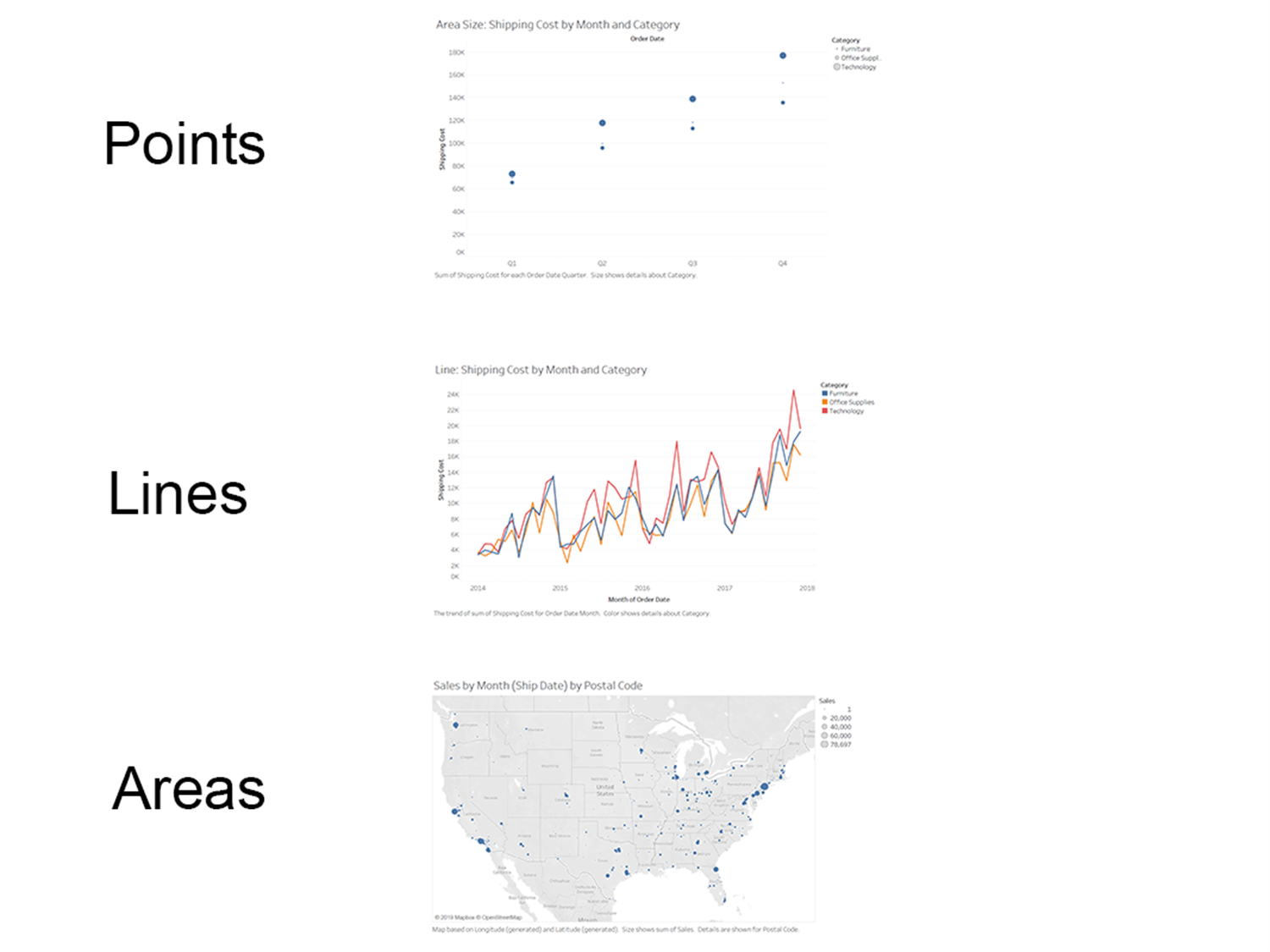
# Visual coding of graphs

There are several ways to think about coding data in graphs, so more formal than others. For example, Stephen Few calls these elements “components,” and Healy and Munzner call them “marks and channels” [Few (2012b)}; [Healy (2018)}; [Healy (2018)}; [Bertini (2016)}. Despite the differences in terms, they are all talking about the same thing: (1) the individual elements of a graph, and (2) how we control the appearance of those elements to tell a meaningful story about our data.

## Marks

A mark is a basic graphical element in a graph (Munzner, 2014). Marks are geometric objects classified according to the number of spatial dimensions they require Figure 6.8 (Munzner, 2014):

1. Point: A zero-dimensional (OD) mark
2. Line: a one-dimensional (1D) mark
3. Area: A two-dimensional (2D) mark. Areas can also be a three-dimensional (3D) volume mark, but they are not frequently used in data visualization.

 Figure 6.8: Three primary types of marks.

### Points

By a point, I mean any simple and small geometric shape that is used to mark a specific location on a graph. A point often consists of a dot. In Figure 6.9 a scatterplot is used to encodes each quantitative value. In a scatterplot every mark represents a data item, and typically its position represents two values (Few, 2012b).

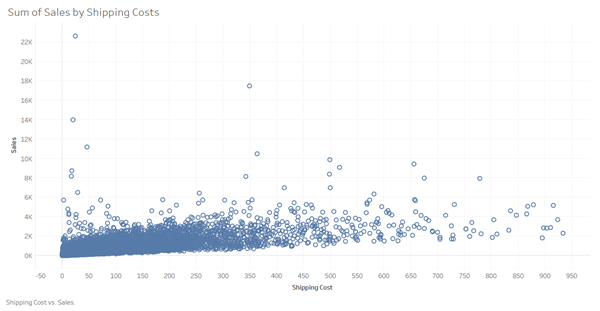


Figure 6.9: Sum of sales by shipping costs.

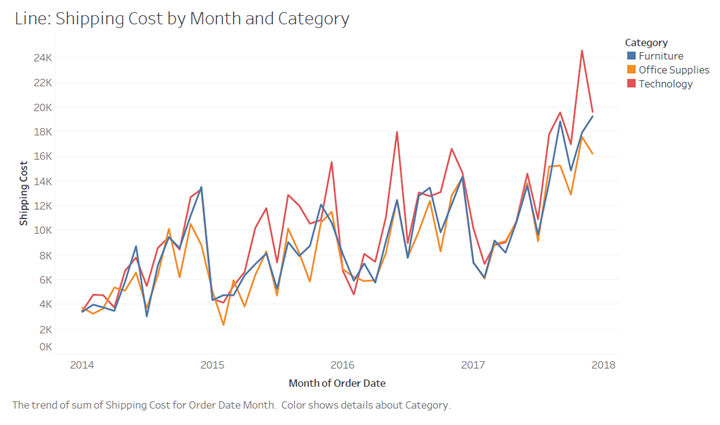
The point closest to the top left represents the value where Sales is equal to 22, 638 on the Y axis and Shipping Cost is equal to 24.3 on the X axis. In addition to a dot, any simple symbol (e.g., a small square or triangle) may be used to mark values on a graph.

Points are often used in maps, as a proportional symbol map. Just like in scatterplots, if you see the points clustering on a symbol map, it might mean something is going on in that region. We will be looking at examples of these types of maps when we get to the mapping portion of this class.

### Lines

The most common example of using this type of mark is the line chart, Figure 6.10. Lines usually connect a series of values in a graph (Few, 2012b).

As you can see in Figure 6.101, line charts typically show how values change over time.

 Figure 6.10 Shipping Costs by Month and by Category.

Although this is not the best view of the data, Figure 6.10 allows us to view the entire series of values as a single pattern from left to right. This is especially useful for displaying how values change through time because it is quite easy to think of the up and down patterns as degrees of change (Few, 2012b).

Lines are routinely used in graphs to represent values in two ways (Few, 2012b):

1. To connect individual data values (as in Figure 6.10)
2. To display the overall trend of a series of values (such as in the form of a trend line in a scatter plot)

Lines can also be use in network diagrams that are made up of nodes, which are connected using lines. In network graphs, the role of the lines is to show the relationship between different nodes.

### Area

There are several different ways that area marks can be used, the simplest is the pie chart. A pie chart is part of a larger family of area graphs, which encode quantitative values as the sizes of 2-D areas (Few, 2012b).

Pie charts use segments of a circle (i.e., slices of a pie). The size of each piece of the pie is equal to its value compared to the total value of all the slices [Figures L4.11].

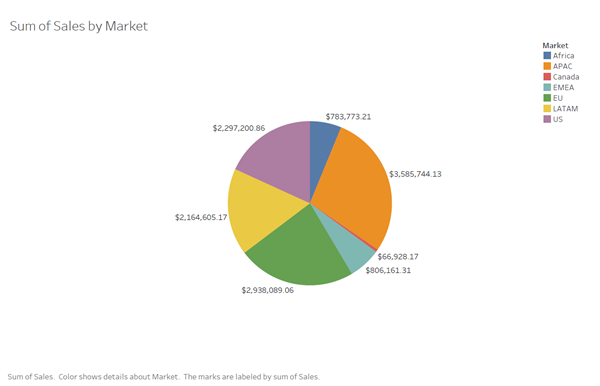


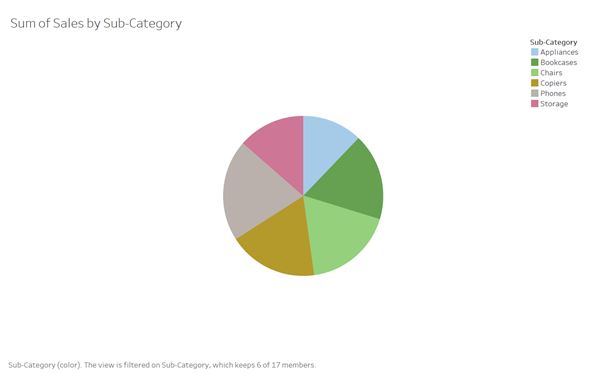
Figure 6.11: Sum of sales by market.

One of the defining characteristics of a graph is that it has at least one axis. So, you might be asking yourself, “where is the axis on this graph?” Though it isn’t apparent, a pie chart does have an axis. However, unlike most graphs, this axis isn’t a straight line. In pie charts, the perimeter of the circle serves as a circular axis (Few, 2012b).

There is another issue with pie charts. A fundamental problem with all types of area graphs but especially with pie charts, is that they are not great at communication information (Few, 2012b).

Our visual perception is not designed to accurately assign quantitative values to 2-D areas, and we have an even harder time when a third dimension of depth is added (Few, 2012b).

If the slices of a pie chart are close in size, it’s difficult to tell which slice is bigger. Take a minute to examine Figure 6.12 and rank each slice from largest to smallest.

 Figure 6.12: Pie chart with similar sized sections.

You might think that these values are extraordinarily close in size, but as you can see when the same values are displayed in a bar graph, that’s not the case [Figure 6.13].

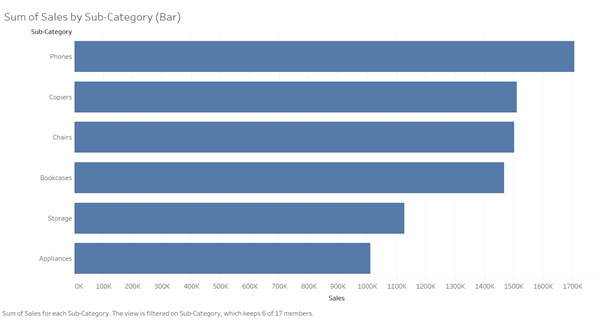


Figure 6.13: Market data displayed as a line chart.

Also, when they’re not close in size, the best you can do is determine that one is bigger than the other, but you can’t judge by how much.

The reason we can see the differences easily in the bar graph but with difficulty in the pie chart is that visual perception is highly tuned for seeing differences among the lengths of objects that share a common baseline (Few, 2012b),(Ware, 2019b)

Another means of encoding values using 2-D areas involves varying the sizes of points, such as in a scatter plot or on a map. Points that vary in size are usually called bubbles. A bubble plot, such as the example below, is simply a scatter plot that uses the vertical and horizontal positions of objects to encode two variables (sales and shipping costs), and the size to encode a third variable (region) [Figure 6.14].

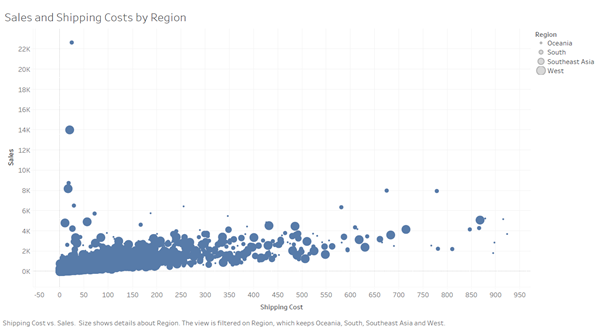


Figure 6.14: Bubble chart of sales and shipping cost by region.

Figure 6.10 displayed the data for shipping costs by month and by category using a line chart. Figure 6.15 is displaying that same data using area marks.

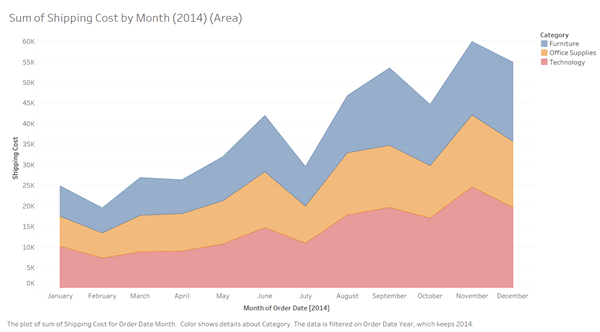


Figure 6.15: Shipping Costs by Month and by Category.

This area chart is showing the same data that we saw in the line charts. However, this chart is showing the percentage that each category is contributing to the shipping costs for each month. For example, the total shipping costs for October 2014 were 44,622. The biggest “slice” is Technology which contributed 16,989 to the total shipping costs.

## Channels

A visual channel is a way to control the appearance of marks, which is independent of the mark’s dimension (Munzner, 2014). Munzner classifies channels as an identity channel or a magnitude channel.

An identity channel, tells us information about what something is or where it is (Munzner, 2014)

* What shape do I see?
* What color (hue) do I see?

Magnitude channels tell us how much of something there is (Munzner, 2014)

* How much longer is line segment A versus line segment B?
* How much area is contained in this square?
* What is the volume of this cube?

Figure 6.16 shows a few of the many visual channels that can encode information as properties of a mark. The basic categories are:

1. Shape
2. Position
3. Angle/tilt
4. Color
5. Size

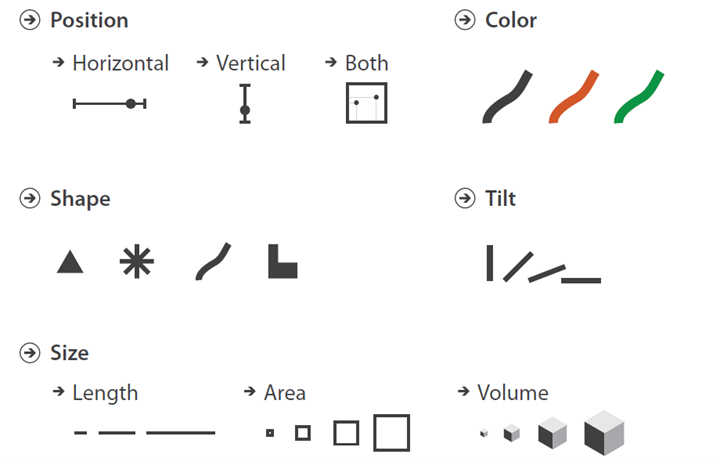


Figure 6.16: Visual channels control the appearance of marks.

### Position

Almost all visualization that you encounter will use position. A scatterplot is an example of graph that uses both horizontal and vertical position. As illustrated in Figure 6.17, this is the positional intersection between the x-axis (349.1) and the y-axis (17,500).

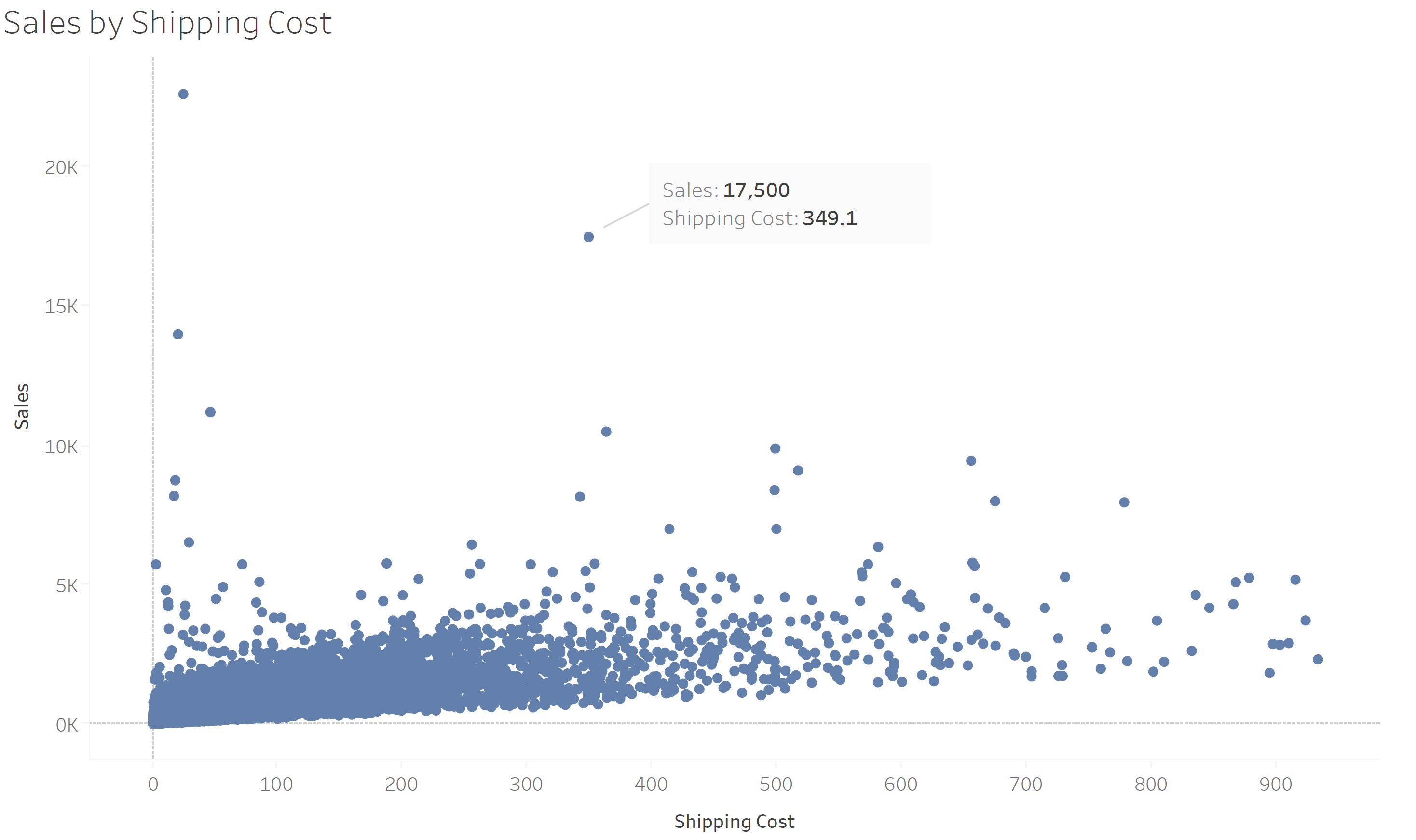


Figure 6.17: Sales by Shipping Costs and Order Priority.

### Shape

For the purposes of analyzing visual encoding with marks and channels, Munzner considers shape as an identity channel that can be used with point and line marks (Munzner, 2014).

In Figure 6.18 we are looking sales and shipping costs by category. However, this time we are using the shape channel to distinguish between order priority. Specifically, circles, squares, plus signs, and x-marks.

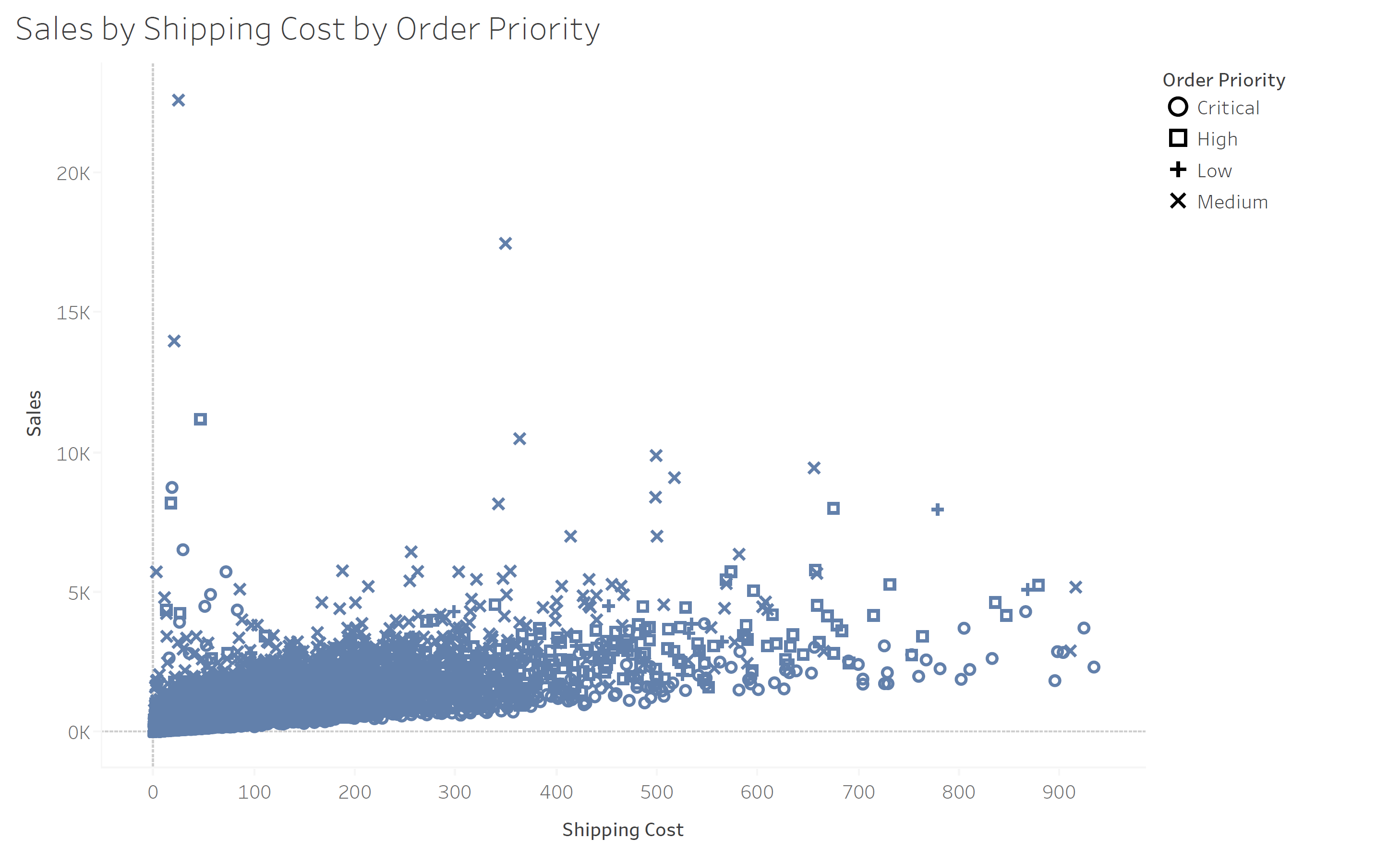


Figure 6.18: Sales by shipping costs and order priority.

If the point size is sufficiently large, the number of discriminable bins for the shape channel is dozens or even hundreds (Munzner, 2014). However, there is a strong interaction between shape and size (Munzner, 2014). Thus, when the region in which the shape must be drawn is small, then far fewer discriminable bins exist. Shape can also interfere with other channels. Please consult (Munzner, 2014) for examples.

### Size

Size is also a very common channel in data visualization, that is appropriate for ordered data (Munzner, 2014). Size interacts with most other channels: when marks are too small, encodings in another channel such as shape or orientation simply cannot be seen (Munzner, 2014). Size interacts particularly strongly with color hue and color saturation.

Length is one-dimensional (1D) size. Height is vertical size and width is horizontal size (Munzner, 2014). Area is two-dimensional (2D) size, and volume is three-dimensional (3D) size (Munzner, 2014).

### Angle

The terms angle, tilt, and orientation are often used as synonyms (Munzner, 2014). The angle channel encodes magnitude information based on the orientation of a mark: the direction that it points (Munzner, 2014). There are two slightly different ways to consider orientation that are essentially the same channel (Munzner, 2014):

* Angle, the orientation of one line is judged with respect to another line
* Tilt, an orientation is judged against the global frame of the display

A line chart is a perfect example of using the tilt channel. In Figure 6.19, the slope of each line is meaningful, in this case the rate of change (because we are using time on the x-axis).

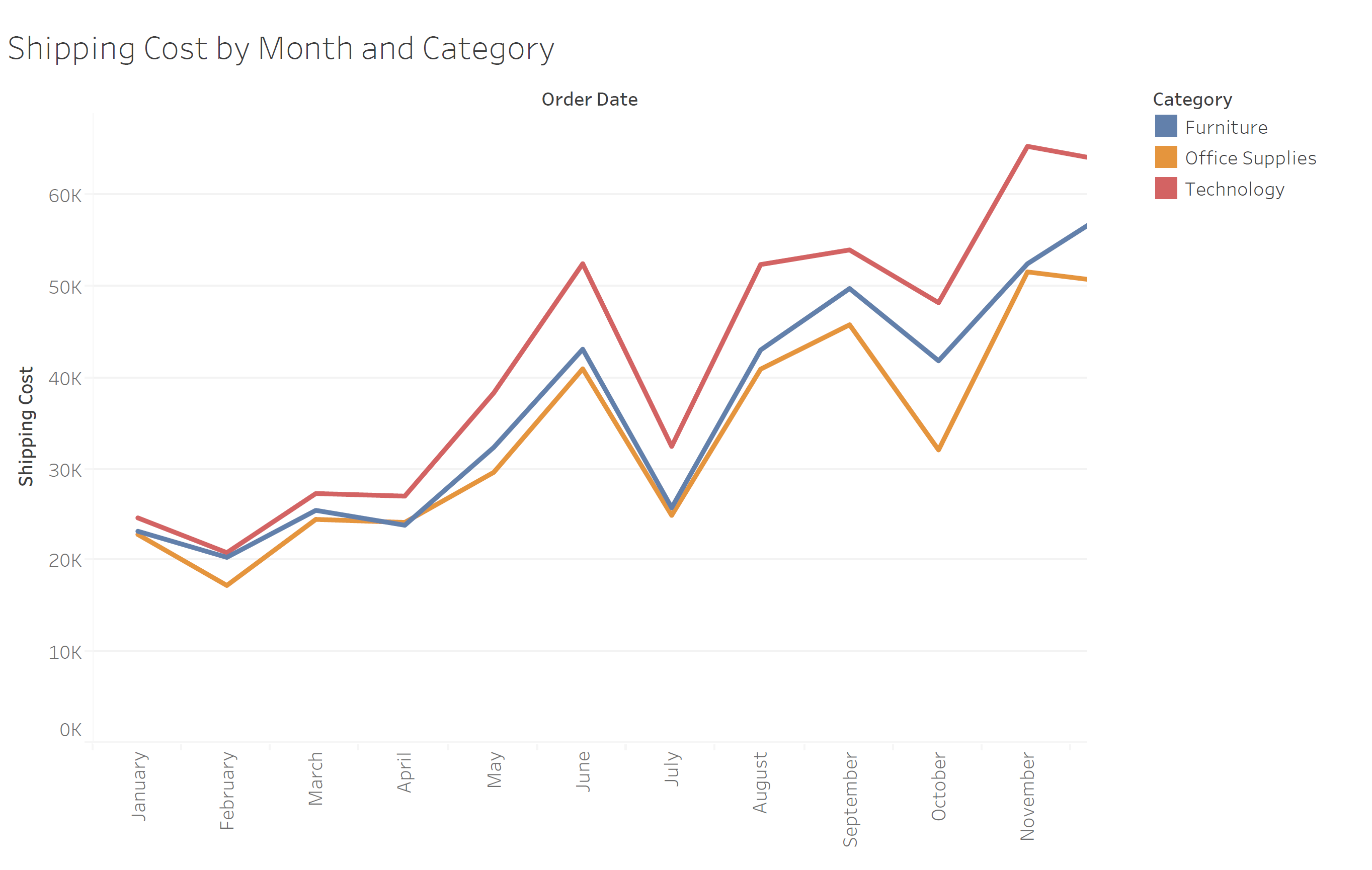


Figure 6.19: Shipping cost by month and category.

A pie chart is another example of a graph that uses tilt. In Figure 6.20 each segment represents the proportion that the sales in each country. The angle is typically mapped to the value represented, in this case the percentage that each market contributes to the total sales.

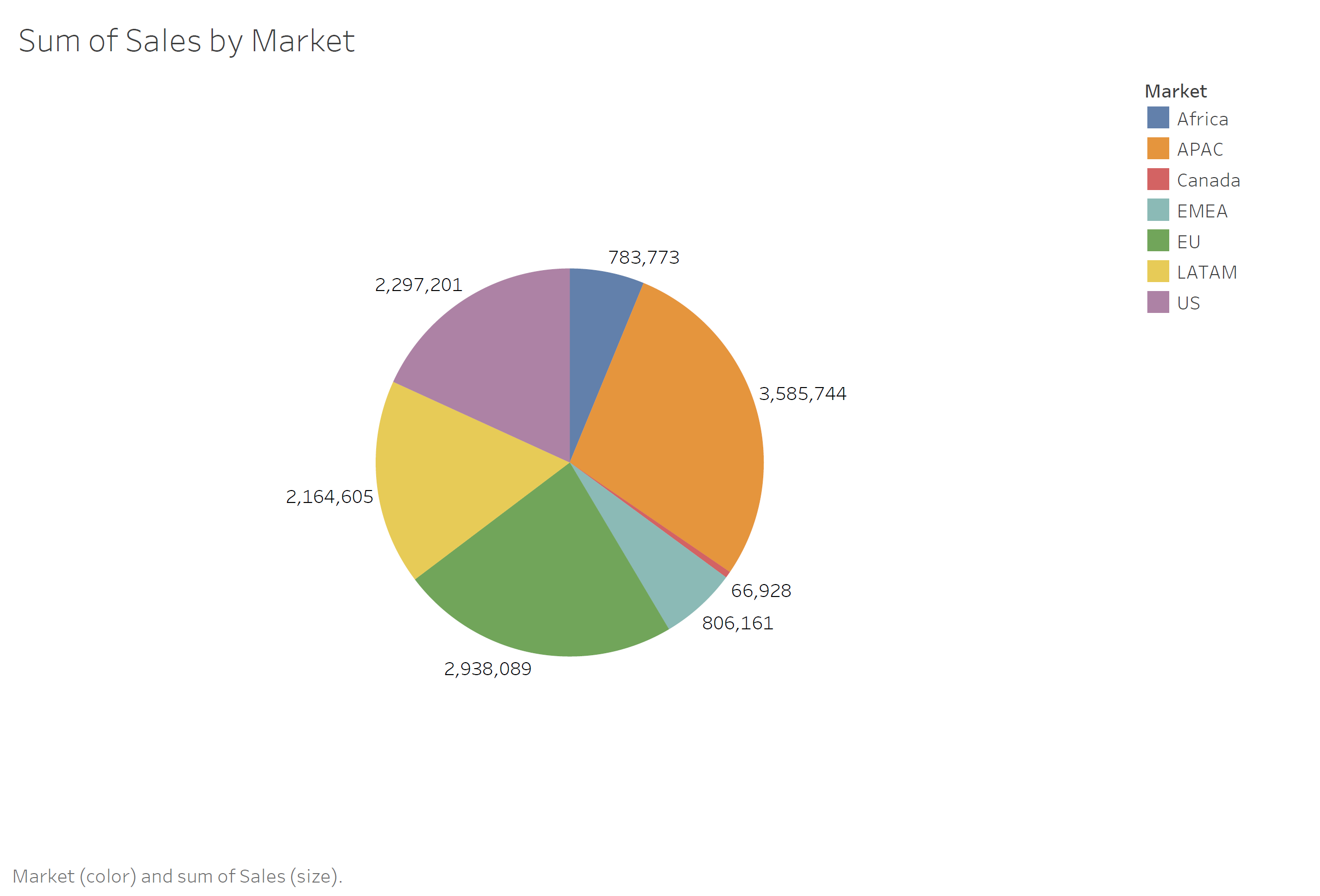


Figure 6.20: Pie chart of sum of sales by country.

### Color

When we think about how to encode attributes using color, we must remember that it has multiple channels: (1) hue [the name of the color], and (2) lightness. Color can also be confusing as it sometimes used as a magnitude channel and sometimes as an identity channel (Munzner, 2014:)

* Hue is very good at encoding categorical information (identity channel)
* Lightness is good for encoding quantitative information (magnitude channels)

We will be exploring the use of color for encoding, next week.

# Channel Rankings

The topic of channel effectiveness is really a very complex topic. So, it is useful summarize/provide examples of the effectiveness of certain channels.

The ranking of channels in this section is based on empirical evidence, meaning experiments were conducted to rank the channels. We will be reading one of the classic empirical studies this week (Cleveland and McGill).

Figure 6.24 presents effectiveness rankings for the visual channels broken down according to the two expressiveness types of **ordered** and **categorical data**. The rankings range from the most effective channels at the top to the least effective at the bottom.

Chart

Description automatically generated

Figure 6.24: Channel rankings from most effective to least effective.

## Identity Channels (Categorical Attributes) Ranking

Categorical attributes should be shown with the identity channels.

1. The most effective channel for categorical data is spatial region.
2. This is followed by color hue. Hue is covered in Chapter 10.
3. The motion channel is also effective, particularly for a single set of moving items against a sea of static ones.
4. The final identity channel appropriate for categorical attributes is shape.

While it is possible in theory to use a magnitude channel for categorical data or an identity channel for ordered data, that choice would be a poor one because the expressiveness principle would be violated.

It is important to note that the highest ranked channel in both the Magnitude and the Identify Channels are **spatial**. Specifically, aligned and unaligned spatial position are at the top of the list for ordered data, and spatial region is at the top of the list for categorical data.

Moreover, the spatial channels are the only ones that appear on both lists; none of the others are effective for both data types.

The choice of which attributes to encode with position is the most central choice in visual encoding. The attributes encoded with position will dominate the user's mental model-their internal mental representation used for thinking and reasoning-compared with those encoded with any other visual channel.

## Magnitude Channels (Ordered Attributes) Ranking

Ordered attributes should be shown with the magnitude channels.

1. The most effective channel is aligned spatial position.
2. This is followed by unaligned spatial position.
3. The third most effective channel is length, which is one-dimensional size.
4. This is followed by tilt or angle.
5. Next is area, which is two-dimensional size.
6. The next two channels are roughly equally effective: luminance and saturation. Luminance and saturation are aspects of color discussed in Chapter 10.
7. The final two channels, curvature and volume (3D size), are also roughly equivalent in terms of accuracy.

## Channel Ranking Examples

### Position on a Common Scale

The most common example is a bar graph. In a bar graph we are using the position channel, on a common scale. Length is also important. However, it is not the primary channel being used.

A screenshot of a cell phone

Description automatically generated

Figure 6.32: Position on a common scale example, bar graph.

The same thing happens with a scatterplot. In a scatterplot, there are two positions: the x-axis (horizontal) and the y-axis (vertical). Every dot corresponds to the values ( x and y) represented by the data.

A close up of a map

Description automatically generated

Figure 6.33: Position on a common scale example, scatterplot.

### Position on an Unaligned Scale

Figure 6.34 is looking at sales by state and sub-category, on an unaligned scale. The y-axis is the same for each portion of the graph. However, each state is on a separate portion of the graph, making this unaligned. Thus, it would be harder to compare sales for chairs in California with sales of chairs in New York.

A screenshot of a cell phone

Description automatically generated

Figure 6.34: Position on an unaligned scale.

### Length

In a stacked bar chart, like Figure 6.35, lets you compare the sales (by state) using the individual segments that make up a stacked bar chart. You are no longer using position because you are comparing the length of each segment of the bar.

A screenshot of a cell phone

Description automatically generated

Figure 6.36: Graph that uses length channel.

### Tilt or Angle

In line charts you are actually using the slope of the line to visually evaluate the change from quarter to quarter. So, if I wanted to compare the sales from 2017 Q2 to that of 2018 Q2 I would compare the slope for each of the two quarters.

A close up of a map

Description automatically generated

Figure 6.37: Line chart of sales by segment illustrating the tilt or angle channel.

We will explore most of the other channels in later lectures.

# Relative versus Absolute Judgements

In a very general definition, Weber’s Law is that the human perceptual system is fundamentally based on relative judgements, not absolute ones. For instance, the amount of length difference we can detect is a percentage of the object's length.

Weber’s Law is fundamental to data visualization. The fact that our senses work through relative rather than absolute judgements has far-ranging implications. When considering questions such as the accuracy and discriminability of our perceptions, we must distinguish between relative and absolute judgements. For example, when two objects are directly next to each other and aligned, we can make much more precise judgements than when they are not aligned. Another example is when two objects are separated with many other objects between them.

Figures L4.24 - L4.26, illustrate this principle. Figure 6.24, illuminates why position along a scale can be more accurately perceived than a pure length judgement of position without a scale.

The length judgement in Figure 6.24 is difficult to make with unaligned and unframed bars.

Chart

Description automatically generated with medium confidence

Figure 6.24: Unframed and unaligned position.

The length judgement is easier with framing, as in Figure 6.25. When making a judgement without a common scale, the only information is the length of the bars themselves.

Placing a common frame around the bars provides another way to estimate magnitude. This allows us to check the length of the unfilled bar.

Chart, bar chart

Description automatically generated

Figure 6.25: Framed and unaligned position.

The length judgement is easiest with alignment, as in Figure 6.26 where the bars can be judged against a common scale.

Chart, bar chart

Description automatically generated

Figure 6.26: Unframed and aligned position.

# Making a Plot

## How ggplot Works

Figure 6.21 displays the main elements of ggplot’s grammar of graphics. We will go through these steps in detail.

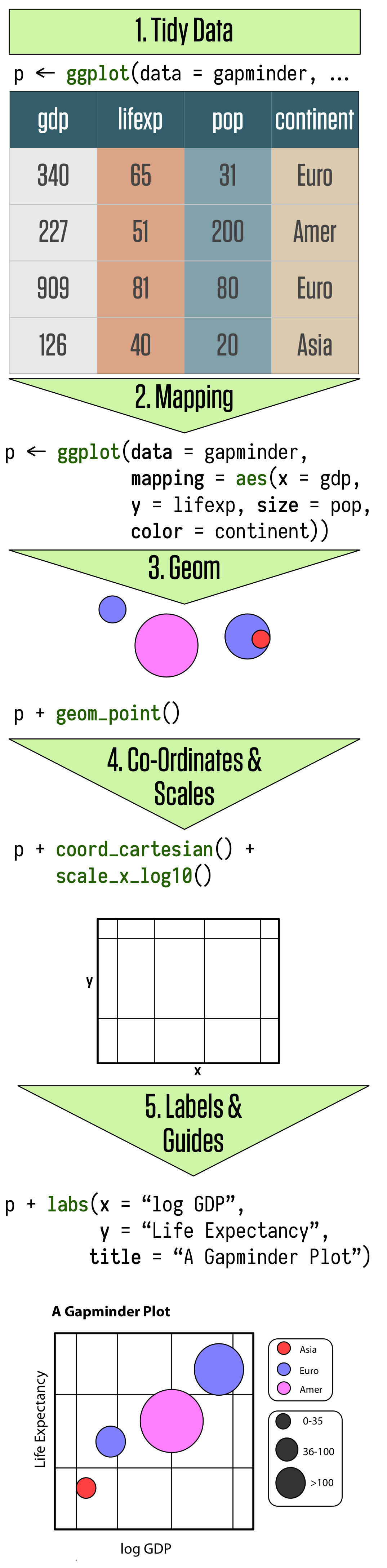


Figure 6.21: Schematic of how ggplot works.

The most important thing to get used is mapping out the logical structure of your plot (Healy, 2018). The code you write specifies the connections between the variables in your data, and the colors, points, and shapes you see on the screen (Healy, 2018).

In ggplot, these logical connections between your data and the plot elements are called aesthetic mappings or just aesthetics (Healy, 2018). You begin every plot by telling the ggplot() function what your data is and how the variables in this data logically map onto the plot’s aesthetics (Healy, 2018).

To build a ggplot, we will use the following basic template that can be used for different types of plots:

ggplot(data = , mapping = aes()) + geom\_xx()

In this class, we will save the basic plot parameters to an object called p, which will contain the core information for our plot. (The name p is just a convenience.) Then we choose a plot type, or geom, and add it to p. From there we add more features to the plot as needed, such as additional elements, adjusted scales, a title, or other labels.

For this lesson we will use the gapminder data to make our first plots. Make sure the library containing the data is loaded, along with the tidyverse.

#Load the tidyverse and the gapminder data  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.2 v dplyr 1.0.6  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(gapminder)

Let us type in the name of the object into R:

gapminder

## # A tibble: 1,704 x 6  
## country continent year lifeExp pop gdpPercap  
## <fct> <fct> <int> <dbl> <int> <dbl>  
## 1 Afghanistan Asia 1952 28.8 8425333 779.  
## 2 Afghanistan Asia 1957 30.3 9240934 821.  
## 3 Afghanistan Asia 1962 32.0 10267083 853.  
## 4 Afghanistan Asia 1967 34.0 11537966 836.  
## 5 Afghanistan Asia 1972 36.1 13079460 740.  
## 6 Afghanistan Asia 1977 38.4 14880372 786.  
## 7 Afghanistan Asia 1982 39.9 12881816 978.  
## 8 Afghanistan Asia 1987 40.8 13867957 852.  
## 9 Afghanistan Asia 1992 41.7 16317921 649.  
## 10 Afghanistan Asia 1997 41.8 22227415 635.  
## # ... with 1,694 more rows

At this point ggplot knows our data but not the mapping. That is, we need to tell it which variables in the data should be represented by which visual elements in the plot. It also doesn’t know what sort of plot we want.

## Mappings Link Data to Things You See

Many of these concepts are from a book by Leland Wilkinson, which have been summarized in a book by Hadley Wickham Figure 6.22. The [book](https://ebookcentral.proquest.com/lib/cua/detail.action?docID=4546676) is available from the CUA library, and it is also available [online](https://ggplot2-book.org/).

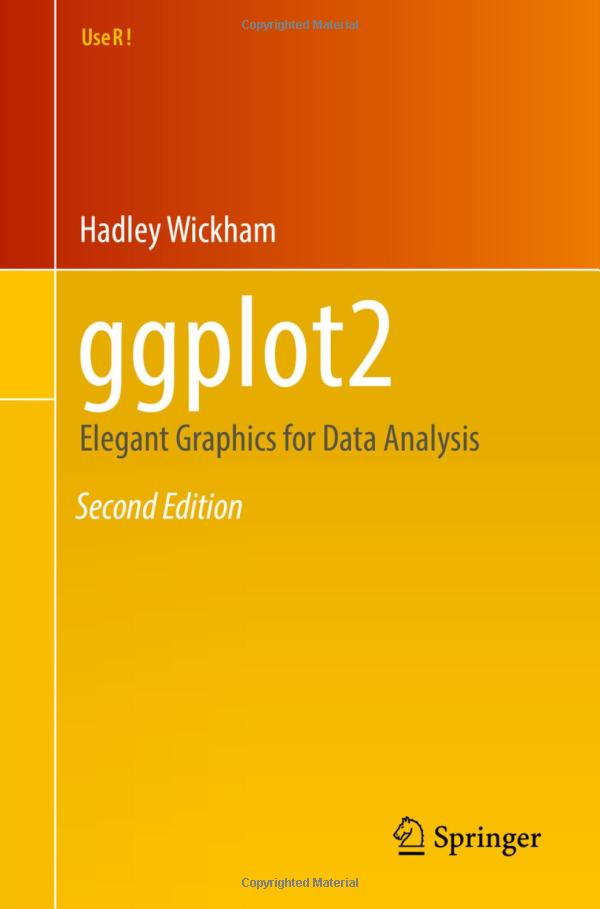


Figure 6.22: Wickham ggplot2 book.

You can think of this mapping process as:

1. Taking data, which is simply a column in you dataset
2. Using the aesthetic attributes of ggplot to display the visual property of your graph. For example the position, shape, or color elements on the graph.

Each geom function in ggplot2 takes a mapping argument, mapping = .

This defines how variables in your dataset are mapped to visual properties. The mapping argument is always paired with aes():

**mapping = aes()**

aes() allows us to select the variables to be plotted and specify how to present them in the graph, e.g. as x/y positions or characteristics such as size, shape, or color.

What does this look like in ggplot? These are some possible options:

|  |  |  |
| --- | --- | --- |
| Data | Aesthetic | Graphic/Geometry |
| Longitude | Position(x-axis) | Point |
| Latitude | Position(y-axis) | Point |
| Army size | Size | Path |
| Army direction | Color | Path |
| Date | Position(x-axis) | Line + text |
| Temperature | Position(y-axis) | Line + text |

Table 01: Mapping data to aesthetics in ggplot, adapted from: [Heiss](https://datavizs21.classes.andrewheiss.com/slides/03-slides.html#14)

When you extend this to particular geoms in ggplot, in might look like this:

|  |  |  |
| --- | --- | --- |
| Data | aes | Graphic/Geometry |
| Longitude | x | geom\_point() |
| Latitude | y | geom\_point() |
| Army size | Size | geom\_path() |
| Army direction | Color | geom\_path() |
| Date | x | geom\_line() + geom\_text() |
| Temperature | y | geom\_line() + geom\_text() |

Table 02: Mapping data to aesthetics in ggplot, adapted from: [Heiss](https://datavizs21.classes.andrewheiss.com/slides/03-slides.html#15)

The code that we would give R to make this happen, might look something like this:

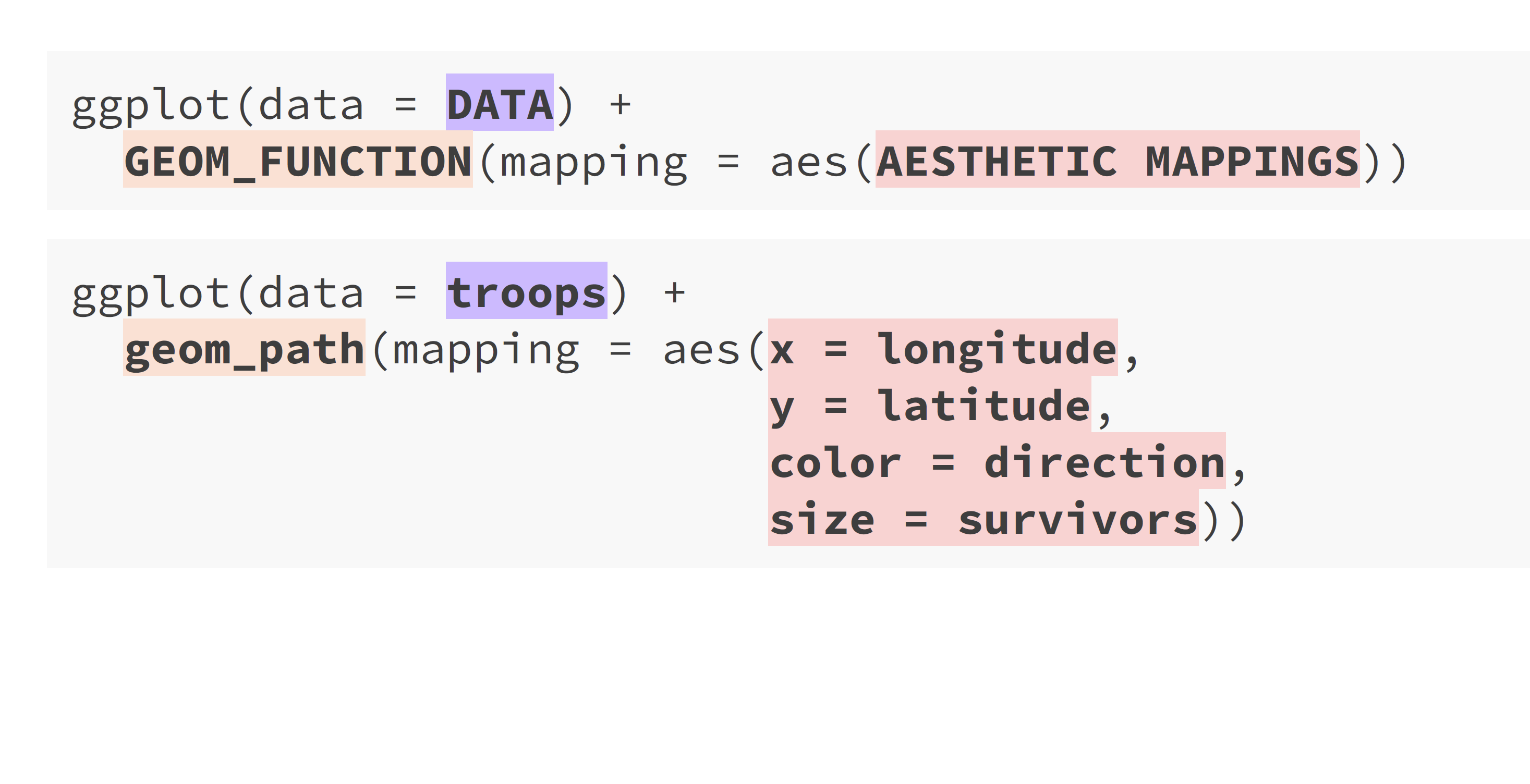


Figure 6.23: ggplot template for mapping troop data to ggplot aesthetics. Image [source](https://datavizs21.classes.andrewheiss.com/slides/03-slides.html#16).

Using the Gap Minder data, let us plot life expectancy against per capita GDP for all country-years in the data.

Let us add aes() to our plot. We are going to plot the following variables, which we will place inside the (x = gdpPercap, y = lifeExp), and save this to the p object.

p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))

So what have we done so far? We have provided the ggplot() function two arguments: dataand mapping. The dataargument tells ggplotwhere to find the variables it is about to use. This saves us from having to tediously dig out the name of each variable in full.

Next, the mapping. The mapping argument is a function (Healy, 2018). The arguments we give to the aesfunction are a sequence of definitions that ggplot will use later [Healy (2018).

For the code that we wrote above this portion mapping = aes(x = gdpPercap, y = lifeExp) says that the variable on the x-axis is going to be gdpPercap, and the variable on the y-axis is going to be lifeExp. The aes() function does not say where variables with those names are to be found. That’s because ggplot() assumes that this comes from object given to the data argument (Healy, 2018).

The mapping=aes() argument links variables to things you will see on the plot (Healy, 2018). The x and y values are the most obvious ones. However, other aesthetic mappings can include, color, shape, size, and line type ( whether a line is solid, dashed) (Healy, 2018).

A mapping does not say what particular colors or shapes will be on the plot. Rather it says which variables in the data will be represented by visual elements like a color, a shape, or a point on the plot area (Healy, 2018).

What happens if we just type p at the console at this point and hit return? The result is shown in Figure 6.24:.

p

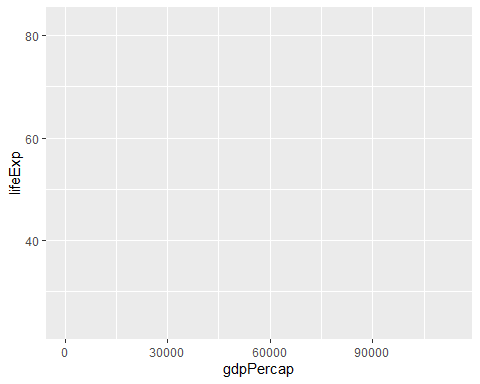


Figure 6.24: Loading ggplot and adding data and mapping the aesthetics.

As we can see from Figure 6.24, the p object has been created by the ggplot() function and already has information in it about the mappings we want, together with a lot of other information added by default (Healy, 2018).

If you want to see just how much information is in the p object already, str(p).

Next, we need to add a layer to the plot. This means picking a geom\_ function. We will use geom\_point().

p +  
 geom\_point()

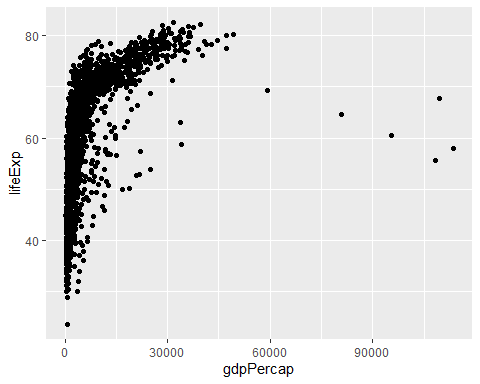


Figure 6.25: Plot with the geom added.

## Build Your Plots Layer by Layer

We just went through the basics of building up plots, layer by layer. We began with creating a mapping between a variable and an aesthetic element. The good news is that, from now on, not much will change conceptually about what we are doing (Healy, 2018).

We will start with a table of data that has been tidied, and then we will do the following:

1. Using data = (), tell the ggplot() function define our dataset.
2. Using mapping = aes(), tell ggplot() *what* relationships we want to see on the plot. For convenience we will put the results of the first two steps in an object called p.
3. Tell ggplot *how* we want to see the relationships in our data by defining the xand yaxes.
4. Using + layer on geomsas needed, by adding them to the pobject one at a time.
5. Use some additional functions to adjust the scales, labels, tick marks, and titles.

It is important to note that you usually cannot add functions to objects, in R (Healy, 2018). Outside of ggplot, functions take objects as inputs and produce objects as outputs (Healy, 2018). This functionality of ggplot() makes it easier to assemble plots one piece at a time, and to inspect how they look at every step (Healy, 2018). For example, let’s try a different geom\_ function with our plot [Figure 6.26].

p +  
 geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

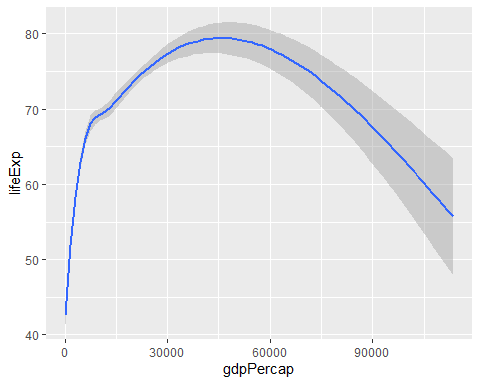


Figure 6.26: Life expectancy vs GDP, using a smoother.

As we can see, some geoms do a lot more than simply put points on a grid. Here the geom\_smooth() has calculated a smoothed line for us and shaded in a ribbon showing the standard error for the line (Healy, 2018). If we want to see the data points and the line together, we simply add the geom\_point() back in Figure 6.27:

p +  
 geom\_point() +  
 geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

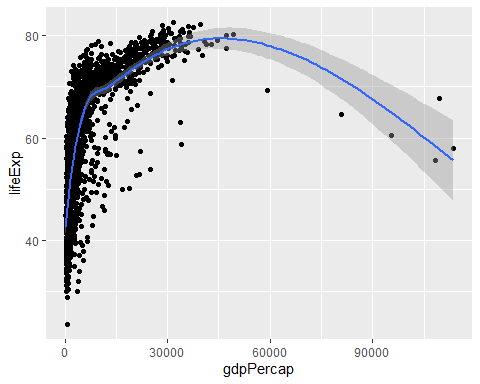


Figure Figure 6.28: Life expectancy vs GDP, showing both points and a GAM smoother.

For figures L4.27 and L4.28 we did not have to tell geom\_point() or geom\_smooth()where their data was coming from, or what mappings they should use. They inherit this information from the original p object (Healy, 2018). As we’ll see later, it’s possible to give geoms separate instructions that they will follow. However, in the absence of any other information, the geoms will look for the instructions they need in the ggplot() function, or the object created by it (Healy, 2018).

## Mapping Aesthetics vs Setting Them

An *aesthetic mapping* specifies that a variable will be expressed by one of the available visual elements, such as size, or color, or shape. For example:

p <- ggplot(data = gapminder , mapping = aes(x = gdpPercap, y = lifeExp, color = continent))

What happens when we graph this?

p +  
 geom\_point()

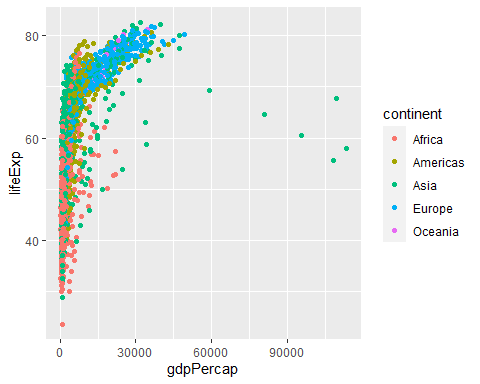


Figure 6.29: Mapping the continent variable to the color aesthetic.

In Figure 6.29, the individual data points have been colored by continent, and a legend with a key to the colors has automatically been added to the plot. There is one for each unique value of the continent variable.

How do you think this code will be interpreted, when graphed?

p <- ggplot(data = gapminder , mapping = aes(x = gdpPercap, y = lifeExp, color = "purple"))

p +  
 geom\_point()

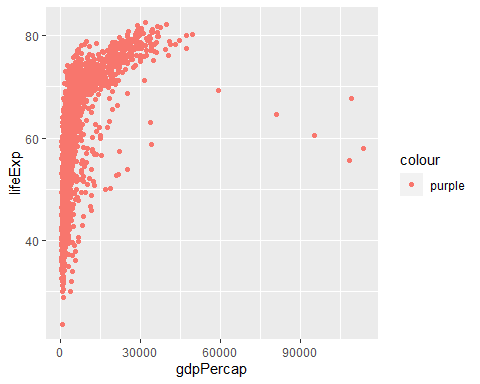


Figure 6.30: What has gone wrong here?

Why is there a legend saying “purple,” and why have the points all turned pinkish-red instead of purple (Healy, 2018)? The aes () function is where that mapping is specified, and the function is trying to do its job (Healy, 2018). It wants to map a variable to the color aesthetic, so it assumes you are giving it a variable (Healy, 2018). aes() will do its best to treat “purple” as though it were a variable (Healy, 2018).

If we want to set a property, like color, we do it in the geom\_ we are using, and outside the mapping =aes (...) step. For example:

p <- ggplot(data = gapminder , mapping = aes(x = gdpPercap, y = lifeExp))  
  
p +  
 geom\_point(color = "purple")

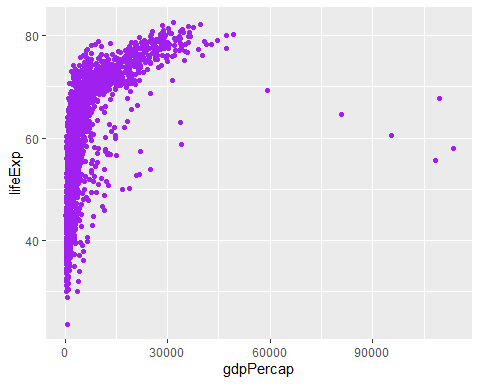


Figure 6.31: Setting the color attribute of the points directly.

## Aesthetics Can Be Mapped per Geom

By default, geoms inherit their mappings from the ggplot() function. We can change this by specifying different aesthetics for each geom. To do this, we use the same mapping = aes(...) expression. However, this time we use it in the geom\_ functions.

p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))

Mappings specified in the initial ggplot( ) function (x and y) will carry through to all subsequent geoms.

p +  
 geom\_point(mapping = aes(color = continent)) +  
 geom\_smooth(method = "loess") +  
 scale\_x\_log10()

## `geom\_smooth()` using formula 'y ~ x'

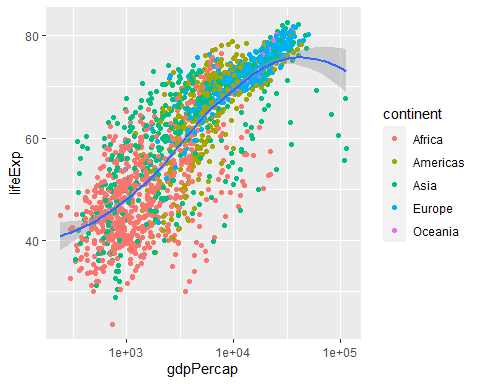


Figure 6.32: Mapping aesthetics on a per geom basis.

As you can see in Figure 6.32, color is mapped to the continent variable for the points but not the smoother.

Finally, it is worth paying a little more attention to the way that ggplot draws its scales (Healy, 2018). Because every mapped variable has a scale, we can learn a lot about how a plot has been constructed, by seeing what the legends look like (Healy, 2018). For example:

p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp, color = continent, fill = continent))  
  
p +  
 geom\_point() +  
 geom\_smooth(method = "loess") + scale\_x\_log10()

## `geom\_smooth()` using formula 'y ~ x'

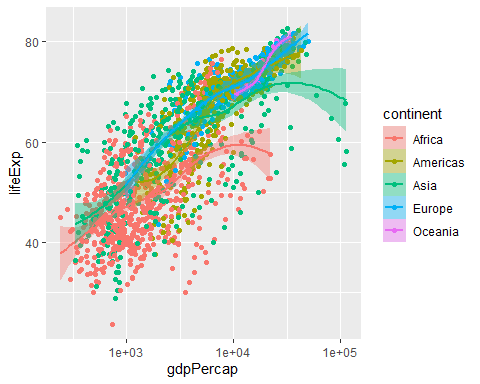


Figure 6.33: Mapping the continent variable to the color aesthetic, and correcting the error bars using the fill aesthetic.

In Figure 6.33, we mapped the continent variable to both color and fill. We then drew the figure with geom\_point()and fitted a line for each continent with geom\_smooth (). Points have color but the smoother understands both color (for the line itself) and fill (for the shaded standard error ribbon) (Healy, 2018). Each of these elements is representedi n the legend: the point color, the line color, and the ribbon fill [Figure 6.34].

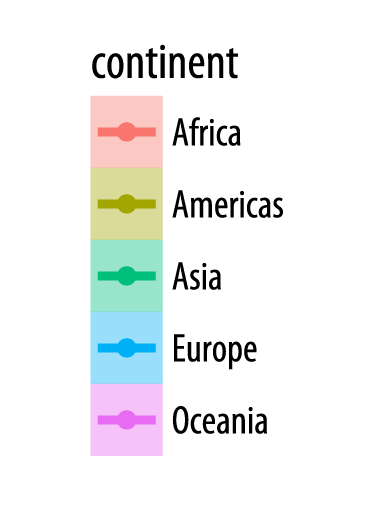


Figure 6.34: Legend that faithfully reflects the mappings for the chart: color, dots, lines, and fill.

In the legend for Figure 6.34, we see several visual elements. The key for each continent shows a dot, a line, and a filled background for the eror bars.

Let us try this with another graph.

p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))  
  
p +  
 geom\_point(mapping = aes(color = continent)) +  
 geom\_smooth(method = "loess") +  
 scale\_x\_log10()

## `geom\_smooth()` using formula 'y ~ x'

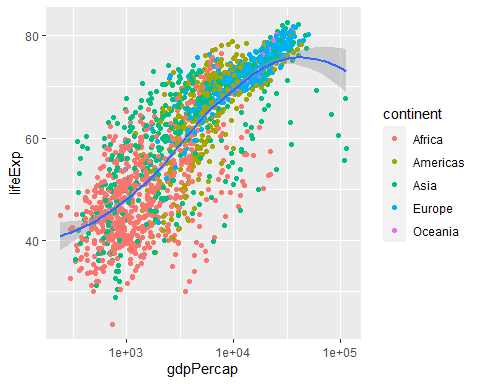


Figure 6.35: Mapping aesthetics on a per-geom basis. Here color is mapped to continent for the points but not the smoother

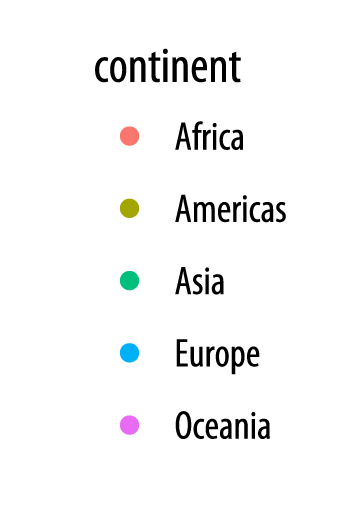


Figure 6.36: Legend that faithfully reflects the mappings for the chart: dots, lines, and color.

The key for Figure 6.36, has only a dot for each continent, with no shaded background or line.

If you look again at the code for Figures L4.35 and L4.36, you will see that in the L4.35 we mapped the continent variable to both color and fill. We then drew the graph with geom\_point() and fitted a line for each continent with geom\_smooth(). In L4.35 points have color but the smoother understands both color (for the line itself) and fill (for the shaded standard error ribbon).

In contrast, in L4.35 we decided to simplify things by having only the points be colored by continent. Then we drew just a single smoother for the whole graph. Thus, in the legend for that figure, the colored line and the shaded box are both absent. We only see a legend for the mapping of color to continent in geom\_point().

# Saving Your Work

## Quick Overview of File Formats

You can save your figure in a variety of different formats, depending on your needs. The most important distinction to bear in mind is between vector formats and raster formats. A file with a *vector* format, like *PDF* or *SVG*, is stored as a set of instructions about lines, shapes, colors, and their relationships (Healy, 2018). Representing the figure this way allows it to be easily resized without becoming distorted (Healy, 2018). This makes a vector-based format like PDF the best choice for submission to journals (Healy, 2018).

In contrast, a raster based format stores images essentially as a grid of pixels of a pre-defined size with information about the location, color, brightness, and so on of each pixel in the grid (Healy, 2018). This makes for more efficient storage, especially when used in conjunction with compression methods that take advantage of redundancy in images in order to save space (Healy, 2018). Formats like JPG are compressed raster formats (Healy, 2018). A PNG file is a raster image format that supports lossless compression (Healy, 2018). For graphs containing an awful lot of data, PNG files will tend to be much smaller than the corresponding PDF (Healy, 2018). However, raster formats cannot be easily resized without becoming pixelated or grainy (Healy, 2018). Formats like JPG and PNG are the standard way that images are displayed on the web. The more recent SVG format is vector-based format but also nevertheless supported by many web browsers.

## Saving plots using ggplot

You will often need to save your figures individually, as they will end up being dropped into slides or published in papers that are not produced using RMarkdown. Saving a figure to a file can be done in several different ways. When working with ggplot, the easiest way is to use the ggsave() function. To save the most recently displayed figure, we provide the name we want to save it under.

There are several file formats are available. See the function’s help page for details. For our first saved graph, we will save the figure as a PNG file, a format suitable for displaying on web pages.

ggsave(filename = "../figures/my\_first\_figure\_png.png")

## Saving 5 x 4 in image

## `geom\_smooth()` using formula 'y ~ x'

If you want a PDF instead, change the extension of the file:

ggsave(filename = "../figures/my\_first\_figure\_pdf.pdf")

## Saving 5 x 4 in image

## `geom\_smooth()` using formula 'y ~ x'

Remember that, for convenience, you do not need to write filename = as long as the name of the file is the first argument you give ggsave(). You can also pass plot objects to ggsave(). For example, we can put our recent plot into an object called p\_out and then tell ggave() that we want to save that object.

p\_out <- p +  
 geom\_smooth(method = "loess") +  
 scale\_x\_log10()

Now, we can save our file

ggsave(filename = "../figures/my\_second\_figure\_pdf.pdf", plot = p\_out)

## Saving 5 x 4 in image

## `geom\_smooth()` using formula 'y ~ x'

When saving your work, it is sensible to have a subfolder (or more than one, depending on the project) where you save only figures. You should name your saved figures in a sensible way. fig\_1.pdf or my\_figure.pdf are not good names. Figure names should be compact but descriptive, and consistent between figures within a project.

In addition, it is also wise to play it safe and avoid file names containing characters likely to make your code choke in future. These include apostrophes, backticks, spaces, forward and back slashes, and quotes.

## Commit and push the changes to GitHub

After you have created the R Markdown document and finished making your changes, it is time to commit them to GitHub. Here is a [resource](https://cfss.uchicago.edu/setup/git-with-rstudio/) and [another](https://rfortherestofus.com/2021/02/how-to-use-git-github-with-r/), that goes into more detail.

1. In RStudio click the Git tab in the upper right pane.
2. Click Commit.
3. In the Review changes view, check the staged box for all files.
4. Add a commit message, for example Add initial speed and distance report.
5. Click Commit.
6. Click the Pull button to fetch any remote changes.
7. Click the Push button to push your changes to the remote repository.
8. On GitHub, navigate to the Code tab of the repository to see the changes.

# Bibliography

Bertini, E. (2016). *Marks and channels* [Unpublished Work]. New York University.

Few, S. (2012a). Differing roles of tables and graphs [Book Section]. In *Show me the numbers : Designing tables and graphs to enlighten* (2nd ed., pp. 39–52). Analytics Press.

Few, S. (2012b). Fundamental variations in graphs [Book Section]. In *Show me the numbers : Designing tables and graphs to enlighten* (2nd ed., pp. 87–136). Analytics Press.

Friendly, M. (2008). A brief history of data visualization [Book Section]. In C. Chen, W. Härdle, & A. Unwin (Eds.), *Handbook of data visualization* (pp. 15–56). https://doi.org/<https://doi.org/10.1007/978-3-540-33037-0_2>

Healy, K. (2018). Make a plot [Book Section]. In *Data visualization: A practical introduction* (pp. 54–72). Princeton University Press.

Munzner, T. (2014). Marks and channels [Book Section]. In *Visualization analysis and design* (pp. 95–115). CRC Press, Taylor & Francis Group.

Ware, C. (2019a). Foundations for an applied science of data visualization [Book Section]. In *Information visualization: Perception for design* (4th ed., pp. 1–30). Morgan Kaufmann (Elsevier). <https://doi.org/10.1016/B978-0-12-812875-6.00001-3>

Ware, C. (2019b). Visual salience: Finding and reading data glyphs [Book Section]. In *Information visualization: Perception for design* (pp. 143–182). Morgan Kaufmann (Elsevier). https://doi.org/<https://doi.org/10.1016/B978-0-12-812875-6.00005-0>

Wilke, C. (2019). Coordinate systems and axes [Book Section]. In *Fundamentals of data visualization: A primer on making informative and compelling figures* (First edition., pp. 13–26). O’Reilly Media.