Chapter 6: Communicating Visually

“The human visual system is a pattern seeker of enormous power and subtlety. The eye and the visual cortex of the brain form a massively parallel processor that provides the highest bandwidth channel into human cognitive centers.”

Colin Ware, “Information Visualization”

In Chapter 1, we briefly mentioned how data analysis is like how we imagine archeology to be: spending hour after hour with small tools in the hope of uncovering even the tiniest of insights in the earth. That analogy can be extended into the shared desire to create a narrative. Archeologists attempt to recreate the stories of history by digging up parts of a story and it’s the same with data analysts. There are stories buried in the data and it’s up to the data analyst to uncover that narrative, piece it back together and communicate that story to others. When it comes to data, with its unique blend of complexity and subtlety, nothing can tell a good story (a *data story*) like a well-crafted visualization.

A data story is built up from several attributes. The two most important of which are **truth** and **relevance**. While we can have a good story without truth, we cannot have a good *data story* without truth. We cannot affect meaningful and successful change if our stories are built on lies or half-truths. Therefore, we need all the skills to uncover the truth within the data, and then we need the visualization skills to be sure the story the consumer perceives matches the same story we uncovered within the data. The visual language should be a wrapper around the truth, thus it needs to be clear and unambiguous. Every point, line, color and shape we place into a visualization can (and should) carry some piece of information supporting the truth in the data and our data story.

A good story is only good if it is relevant and—hopefully—actionable to the consumer. We wouldn’t want to show a board-level executive the SIEM dashboard any more than we’d want to force market reports on the SIEM operator. Stories completely fail to communicate if the consumer doesn’t feel this applies to them. Therefore we have to know the audience for our visualizations. Are we trying to illicit and budget change or firewall change? A good question to ask yourself is “so what?” and if you struggle to answer that question for the consumer, rethink the approach. Another good mental exercise to run through a few other possible outcomes of the story, if the result of the visualization is the same (from the consumers perspective), then you should be rethinking the visualizations. For example, if we’re showing a line graph with an obvious upward slant, imagine if that line went down, would the consumer have a different reaction? If it went up much more than it does, *so what*?

We aren’t suggesting that all data should be visualized. If the story in the data is summarized with a sentence in an email, so be it. If the data can be expressed in a simple look-up table than so be it. The goal here is communicating the data. If we can communicate better, more succinctly or simpler in any other way, then we should go with that method. We also aren’t suggesting that visualizations be the center of the story. All data exists within a context and all our stories have a beginning, middle and end. Visualizations can play an important and supporting role in the entire communication process, but it should not be the communication by itself. Our focus is on the successful communication of the narrative and the method of communication is just a means to that end.

Why Visualize?

Our ability to visually process information is by far the most efficient path to human understanding. Like a good hacker, we want to learn about this system, understand how it functions (and how it doesn’t function) and then exploit this cognitive system to achieve our goal. In this case, our goal here is effectively and efficiently communicating the stories we find in our data. There are many advantages to using data visualization as a communication tool compared to other methods. To paraphrase Colin Ware (who we quoted to open this chapter), data visualization has the following advantages:

* **Data visualizations communicate complexity quickly.** Descriptive statistics (mean, median, variance, etc.) exist to describe and simplify data but tend to remove subtleties that may exist in the data. It’s possible to communicate millions of data points in seconds while minimizing the loss of detail and resolution through visualization.
* **Data visualizations enable recognition of dormant patterns.** Patterns that would never be apparent using statistical methods or scanning the data may be revealed through visualization. By visually representing the data, patterns in a single variable or relationships across many variables may leap off the screen at us.
* **Data visualizations enable quality control on our data.** Mistakes and errors in data collection or preparation can often be revealed through visualization. Data visualizations can serve as a good and quick sanity check on our work.
* **Data visualizations can serve as a muse**. It’s been said that most breakthroughs in science didn’t start with a “Eureka” but instead with a “Huh, that’s odd.” Laying out our data visually can give us new perspective and help facilitate our thinking and discovery process.

Unraveling Visual Perception

The system of how we process of visual information is incredibly complex and much of our knowledge around it is still evolving. There are a few key (and hopefully easy) concepts that we should understand since knowing how the brain visually processes information will help us create great visuals. Equally as important, it will also help us understand a few ways *not* to create visuals.

We begin this journey with visual stimulus in the form of light that our eyes convert into electrical signals for our brain. This information will pass through stages of our **visual memory,** each with a specific set of strengths, limitations and functions. Before we are consciously aware of it, our brains rapidly scan the visual field, which is called **preattentive processing**. Finally the brain will instruct the eyes to focus elsewhere, and through a series of **saccadic movements** our eyes will focus on various features to help build up the image in our mind. With these three concepts from our visual processing system, we should be able create a solid foundation for good visuals and dashboards.

Visual Thinking

We will step through the various stages of memory within our visual perception. **Iconic memory** is the first stop for the visual information. It is a very brief stop, lasting around half a second or until it’s replaced with new information. But what happens in this tiny window is critical to creating good visualizations and dashboards. With the information stored in iconic memory, the brain preprocesses the image prior to giving it any conscious attention. From an evolutionary perspective this is quite helpful, this preattentive processing can help us quickly identify possible threats in our environment. For example, anyone who has been driving when an animal dashes in front of the car has probably felt that urgent message from the brain when it recognizes a possible threat. We begin to react immediately even before we can process the full extent of the threat. While we hope our visualizations aren’t treated like a threat, it’s that visual searching and preattentive processing that we can leverage to draw attention and even communicate some basic attributes of our data to make processing much easier when we begin to consciously process it.

**Working memory** is the next stop and things get a little more complicated here. First the brain will gather up and group visual aspects into meaningful objects and hold these individually in working memory. There is a lot of flexibility within working memory as we can rapidly replace or drop these objects as we take in more information, but this flexibility comes at a cost in capacity. We can only hold three to five objects in working memory depending on the task and objects. This limit is important when designing visualizations and dashboards. If we create a visualization with a legend that has ten different attributes, the consumer will have to continually reference the legend in order to understand what they’re looking at. So as we communicate the stories in our data we want to limit each visual to no more than five objects (four to be safe).

**Long-term memory** is not directly important as we attempt to communicate our data. In order for something to move into long-term memory the consumer needs to visually “rehearse” the information to transition that visual chunk from working memory into long-term memory. Indirectly, we will leverage long-term memory to detect meaningful patterns and relationships within the data. This type of deeper understanding and processing is only available with long-term memory.

Tracking Eye Movements

When we focus on something like a dashboard or visual on a computer screen, we do not simply fix our gaze on it and take in the image as a whole. Our eyes actually dash around the screen, focusing on very small portions for very short periods of time in order to build up the image in our mind. One of these rapid eye movements is called a *saccade*, overall they are called saccadic movements and they are anything but random. The brain has a set of rules (guidelines really) for how the next fixation point is prioritized. As an example, when another person greets us, our eyes perform scanning saccades over their entire face, bouncing from the distinct features of the face (eyes, nose and mouth) and establishing the edges. The scanning saccades help us with recognition not only of the person, but also of their emotions. The same applies to our visualizations and dashboards. The eyes will fixate on an obvious feature and bounce around and between to the points it considers important. We will build up the entire picture over a series of these movements and over time. Understanding these movements can help a visualization flow and feel natural (or at least not strained) to the viewer

The saccadic motion itself is largely unconscious and is thought to be a ballistic movement. Meaning once the brain initiates a saccadic movement, the muscles take over and handle the rapid acceleration and deceleration from beginning to end. This is important for two reasons: once it is initiated it cannot be changed or stopped and during the motion we suppress much of the visual input. We will want to limit the distance of these motions by creating compact dashboards and visualizations.

We can pull together a few important learning points from saccadic eye movements. Knowing that the eyes will bounce around from feature to feature and the ballistic nature of the movement, we should keep several points in mind as we create our dashboards and graphics:

* **Don’t overload the dashboard with visual features**. Keep the number of attention-grabbing features under control because if everything is important visually, than nothing will be important visually and the analyst will have to put more effort in to understand the visual.
* **Make the important messages obvious visual features.** Just as we will scan the important parts of a human face, we will look for the similar attention-grabbing features on the screen. Make sure that those features are clear and are important to the viewer.
* **Limit time wasted on saccadic movements.**  Saccadic movements that jump longer distances take longer to execute. Do not push the visual features into the corners or towards the edges. Forcing the viewer to bounce across large distances will decrease the amount of time they are actually seeing the features (and increase the time spent in saccadic movements).

The role of saccadic movements influence dashboards much more than static data visualizations. A static visualization will typically have one, perhaps two, visual features we want draw attention to and the eye movements are contained in a relatively compact space. A dashboard may be designed to communicate several independent messages simultaneously with varying degrees of urgency. Good dashboard design, as we’ll cover in Chapter 10, will want to limit the time spent in a saccadic movement and exploit the eye movement for efficiency in our communications.

Preattentive Processing

The best way to describe preattentive processing is through pictures. Take a look at figure 6.1 and try to count how many capital X’s are in this completely random mix of letters and numbers.

Figure 6.1 Count the number of “X” characters [FILENAME 793725c06f001]

Because all of the letters are the same color and contained the same relative space, nothing about any of the characters really stands out. The brain simply sees a collection of shapes. In order to count the X’s we have to scan through each letter across the four rows. While we’re doing that we have to remember how many we’ve found as we scan so we don’t lose track. Now, let’s take a look at this completely random mix of letters and numbers with the X characters emphasized.

Figure 6.2 Count the number of “X” characters [FILENAME 793725c06f002]

Immediately we can see the X’s and count four of them. When we first look at this, the brain sees a background of gray symbols with four completely different objects that are similar to each other. Our preattentive processing will mentally create two groups: one of all the gray symbols and a second with the dark red X’s. A split-second later, we will consciously recognize the second group as what we’re interested in (the X’s), it becomes trivial to visually exclude the gray characters and now we can scan just through this group. Counting the X’s becomes a simple and quick task.

That mental grouping and ease of focus is what we are after. We want to enable our preattentive processing to effortlessly group similar objects and highlight where we want attention to be focused. But we have to keep in mind that the preattentive processing is not all that smart. There are only a handful of attributes that our preattentive processing will be able to pull out because the sole purpose of this processing is to recognize features in our visual environment. It will not be able to project meaning, interpret the objects or make meaningful associations (beyond simple visual grouping).

Through hundreds of studies, researchers have been able to differentiate visual attributes from what can be preattentively identified from those that can’t. Having looked through some of these studies they can get a little silly and abstract (how easy is *parallel* detected?), but looking at them as whole, we can create some high level categories of what can be preattentively processed. These categories are **form** (line, shape, size), **color** (hue and intensity), **spatial position** (two-dimensional, stereoscopic) and **motion** (blink, direction). The list of specifics within those categories can get quite long, but, thankfully, we can experiment here and iterate through various visual features in our graphics. If one version doesn’t highlight the data, try something different. Chances are good if it’s easy for you to pick out, it’ll be easy for others, and it’s a good idea to run things by others as a sanity check. Figure 6.3 gives a few visual examples of ways to differentiate based on preattentive attributes.

Figure 6.3 Examples of Preattentive Attributes [FILENAME 793725c06f003]

Not all preattentive attributes are created equal. Look at figure 6.3 again. While they all highlight the three data points, some make the three points slightly easier to see than others. For example in figure 6.3(e), if we would have chosen colors of pink and red, it would have been slightly more difficult to pick out the differences with the subtle difference in colors. The amount of “pop” for preattententive attributes depends on how different the attributes are. The shapes in example in 6.3(a) are more different from each other than the circles and squares in 6.3(b) and slightly easier to see. It’s still possible to distinguish the difference in 6.3(b), but it’s just not as quick to “pop”.

This concept of preattentive processing should be treated as just that—a concept. The line between our preattentive processing and conscious processing is gray and blurry. When looking at a visualization, we may slip between the two quickly and quietly. With repeated exposure too, we can actually train our preattentive processing. Meaning, over time, no matter how poorly designed a dashboard is, analysts will eventually pick up skills to quickly identify important features depending on environment and culture. But the point remains for our visualizations and dashboards. If we want to direct the consumer’s focus and attention we should leverage some basic elements like form and color to highlight the point we need to make in the data.

Finally, one last word of caution about preattentive processing: it’s possible to overload this process and negate any benefit. Take a look at Figure 6.4 below. In the first example 6.3(a), we separate three groups by color and it’s quite easy to pick them apart, not only are they spatially grouped, but the color highlights the difference. In 6.3(b), we attempt to communicate a difference with shapes and it’s a little harder to tell them apart, but we can still pick out the two groups. When we combine them in 6.3(c) things get a bit more complicated. Now to separate based on shape we have to actively inspect individual elements and separate them consciously. We have to be careful to keep the visuals as simple as possible to exploit the consumers preattentive processing for their benefit.

Figure 6.4 Too many attributes [FILENAME 793725c06f004]

Understanding The Components Of Visual Communications

We began by looking at how the brain visually processes information, how we can leverage our preattentive processing and saccadic movements to increase the consumer’s perception of a visual. Now we’re going to focus on the visual building blocks and material that we have to work with. We need to begin with our data and encode the values through various attributes like position, shape, length and size. Perhaps we want to encode changes over time with slopes or angles and separate categories by color hue, saturation or lightness. If we combine elements we can communicate relationships and groupings. Every choice we make in creating a visualization will affect how well others will decode the data.

Avoiding the third dimension

First and foremost, unless we are creating a physical data sculpture, we are dealing in two dimensions. The screens we look at, the reports we print out and slides we project on the wall are all limited to width and height dimensions. Of course we can simulate our perception of the third dimension of depth, but this brings a challenge. Simulating a third dimension will always be just that, a simulation. In order to simulate depth, we change the very attributes we are using to convey the meaning of our data. Elements that are closer in the simulation will need to be bigger and those further away will be smaller. The effect from the simulated perspective will modify consumer’s ability to compare and consume the data accurately. For this reason, we strongly recommend staying away from plotting in three dimensions. Plus, two dimensions offer a tremendous amount of flexibility. Even though readily available desktop tools like MS Excel makes 3-D charts incredibly easy, we should fight the urge if our goal is communicating our data to others.

We shouldn’t think of working with 2 dimensions as a limiting factor any more than just 12 notes in a chromatic scale is limiting to western music. Much research has been conducted into communicating in two dimensions and we will highlight two seminal papers published in the mid-1980’s by two statisticians William S. Cleveland and Robert McGill. They open the first paper, “Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods” with, “*The subject of graphical methods for data analysis and for data presentation needs a scientific foundation.*” And, they did just that. They conducted experiments where subjects were shown various graphics and measured how accurately they were able to visually decode the quantitative information in them. In their second paper, “Graphical Perception and Graphical Methods for Analyzing Scientific Data”, they updated their results and offered an ordered list of visual encodings and the relative accuracy in their decoding.

Figure 6.5 Accuracy of Decoding [FILENAME]

These are not mutually exclusive and the lines between these get a little blurry. For example to decode a simple bar chart, we may use position on a common scale to determine the quantity, but then use length to compare two bars within the same chart. In a pie chart, we may primarily use angles, but the area of the slice and arc length may also factor in to our perception. The findings from this research should serve as a guideline, if our goal is communicating quantitative data accurately; a bar chart is always better than a pie chart and a grouped bar chart is better than a stacked bar chart.

With all guidelines, we can deviate from this advice. Sometimes our goal is not to convey specific quantitative data, and the lack of accuracy in decoding is desired. As an example, let’s look at Figure 6.6. When looking at the pie chart on the left, it is relatively difficult to gauge the specific difference between the five slices. Looking at just the pie chart, we’d probably conclude that they are all about equal. However, if we look at the bar chart on the right, it’s relatively trivial to see the differences because we are using position on a common scale. Obviously, if we had confidence in the data its accuracy, the bar chart on the right is far easier to see the values and relationships. But, what if the data we have is from a small opinion survey? While we can calculate precise values, the differences in the values could easily be explained with sample error. In this case, we could justify using a less accurate method to communicate our data.

Figure 6.6 Comparing Pie and Bar Charts [FILENAME 793725c06f006]

type="note"

Save the Pies for Dessert

If you are new to data visualization, there are essentially two distinct (and sometimes very passionate) opinions when it comes to visualizations that use techniques lower on Cleveland’s accuracy list. Pie charts are often at the center of debate since they are used (and abused) more often than others. The core argument against pie charts is that the data can always be represented better and more accurately with other methods. As Stephen few said in his 2007 paper *Save the Pies for Dessert*, “*Of all the graphs that play major roles in the lexicon of quantitative communication, however, the pie chart is by far the least effective. Its colorful voice is often heard, but rarely understood. It mumbles when it talks*.” But on the other side is the point we made here, that the goal of communication may not be precision. There are other less convincing arguments in the defense of pie charts, but there is one piece of common ground: choose the visualization method deliberately and be sure it communicates the message you want to send.

Using Color

If you’ve never been tasked with selecting colors for a project this brief introduction may make color selection seem easy. There are a few guidelines on what types of color palettes go with what types of variables and a deep well of knowledge from color research has brought us a handful of easy rules for palette creation. However, it won’t be until you’re trying yet another set of colors in your visualization that you will truly appreciate the words of Edward Tufte. “Avoiding catastrophe becomes the first principle in bringing color to information: Above all, do no harm.”

There are many websites and tools that leverage color theory to make palette selection relatively painless (see the appendix for a complete list of resources, but Color Brewer (http://colorbrewer2.org/) and HCL Picker (http://tristen.ca/hcl-picker/) are our favorites). With some understanding of your data, picking colors that are good in [color] theory is the easy part. Colors also have to support and hopefully even highlight our message and be pleasing to the eye, which have a large element of subjectivity and are unique to each and every visual story. This creates the challenge with color: we have to balance function, aesthetics and theory across just a handful of colors.

Color is Relative

The first and perhaps most important aspect of color selection is that colors are always interpreted relative to the surrounding environment. For example, Figure 6.7 shows two rows of gray boxes on a gradient background. Even if we know each row has a consistent shade of gray, we will still see different shades on the same row as we scan from side to side. And to some, the upper left box looks the same color as the lower right. That’s because we see the shade in the boxes relative to the surrounding background. The boxes appear darker on a white background and lighter on a dark background. We can use this to our benefit as well. If we want to emphasize one variable above all else, we could choose a contrasting color from the rest. For example, red shapes will stand out among shades of light blue shapes, but will blend in with pink and orange shapes.

Figure 6.7 Visual signal and noise detection illusion [FILENAME 793725c06f007]

type="note"

We are the ~~99%~~ 10%

As we choose colors there is a

Palettes Depend on Data

We have not talked about it much before, but there are only a handful of high-level data types, and most of those fall into either categorical or quantitative values. **Categorical data** are represented as groups such as type of operating system or a programming language. **Quantitative data** are quantities (crazy, huh?) which are things we count or measure such as bytes, packets, sessions, number of servers and so on. Sometime categorical data may have a natural order to them. Rankings such as “first”, “second”, “third” or “high”, “medium”, “low” are treated like a categorical value but have an added sense of order to them. Sometimes the lines get blurry, TCP/UDP port numbers for example appear quantitative since they are sequential numbers going up to 65,535. But we have to treat them as categories: we would never add echo and two telnet ports to get DNS (yeah the math works out there). Another confusing data type is date/time. Most of the time we will treat this as an ordered categorical variable (such as the year, month, day of week, etc), but other times we’ll store it as a quantity (seconds since the epoch) to enable calculations on time, and *time series* data.

We have to be careful using colors to represent a quantity. We are relatively inaccurate when decoding quantity from color. But it may be used in circumstances where rough comparisons are enough. For example, back in Figure 5.7, it doesn’t matter if we can precisely see 1 in 724 people in Wyoming were infected with ZeroAccess. The color is simply communicating that Wyoming had more infections per person than any other state.

Figure 6.8 shows three types of color palettes, sequential, divergent and qualitative, from the Color Brewer website. We would select a palette of **sequential colors** to represent quantity or perhaps ordered categorical data. Sequential color palettes are built using a single hue (blue for example) and then adjust the lightness or saturation of that color to cover the range of the quantitative data. **Divergent colors** are also used on quantitative or ordered data, but help communicate above or below some middle value. Typically, the middle value is white and two divergent hues are used on either end. Divergent color scales may be used to convey two directions in the data such as above or below average (as it was used in figure 5.7). Finally we have **qualitative colors**, which are intended to not convey ordering and are used to represent categorical data.

Figure 6.8 Sample color palettes from Color Brewer [FILENAME 793725c06f008]

Putting it all together

We’ve laid some good ground work here, now let’s look at how these things come together to help communicate our data. We’ll spend less time talking about how to create these and more on why we create these as we do. But all of the source data and code to create these visualizations in this chapter are on the book website. Creating the basic types of plots are relatively easy within the R language and using ggplot2 and most of them are available as option in more familiar tools such as Excel.

Using Points

The easiest method to communicate and compare two quantitative variables is the basic scatter plot. Scatter plots position points along a common scale (both x and y scales) and allows the consumer to very accurately determine the quantities of variables and compare between various points. It is insanely simple to create scatter plots in R: (plot(x, y)) and, we’ll often do this just to “see” the data we are working with. For example, Figure 6.9 shows eight hours of firewall traffic. Each dot represents total number of packets (x-axis) and total number of bytes transferred (y-axis) processed by the firewall over 5 minutes.

Figure 6.9 Basic Scatter Plot [FILENAME 793725c06f009]

This is a good example of when a pattern quickly jumps out of a plot. We can see that the firewall traffic for the day ranges from around 7 gigabytes up to 19 gigabytes, and we range from 12 to 27 million packets. The linear relationship is very clear here: as we see more packets we see more bytes. Now this isn’t exactly a news flash or all that informative, but if we have data where we aren’t sure what’s in it, a simple scatter plot can do wonders. Figure 6.10 is an example of a scatter plot where we can quickly see something we didn’t know. This time we are putting the time of day along the x-axis against the number of sessions on the y-axis.

Figure 6.10 Dot Plot: Packets over Time [FILENAME 793725c06f010]

We did a couple of extra things here. We dropped some faint lines down from the points to give just a hint of a bar chart and visually tie the points (which are rather bunched up) back to the x-axis. We also wanted to highlight the repeating element of time so we darkened the line at the top of the hour and changed the points to be red every 30 minutes. It’s obvious to see why we did that. There is a noticeable dip at the top of the hour and not much significant change at the half-hour marks, but we wanted to emphasize those times for easier comparison (remember the preattiventive processing?). Perhaps it’s a meeting-heavy culture and people are walking to a new meeting around that time and not surfing. Who knows, but the pattern really jumps out with a simple scatter plot here.

Creating Directions with Lines

At some point, we heard someone say “lines are just points in motion” and that’s true, we see a lines having a sense of direction. Let’s take the same firewall traffic and separate out the types of devices on the network:

* Desktops
* Servers
* Printers
* Networking equipment.

Now let’s create two plots: first the same type of scatter plot of the time series and then do the same thing with a line plot.

Figure 6.11 Line Plot: Traffic by Device [FILENAME 793725c06f011]

It’s rather clear what’s going on with the line plot and it’s easy to follow the traffic over time for each of the four devices. The scatter plot on the left though is a little difficult to follow, though we can see trends and differences between the categories. Line plots are quite good at accurately communicating data since we are comparing points on the line along a common scale and we get the slope of the line as a sign of change; steep slopes, like on the printers. Line plots are most effective with at least one quantitative variable and some type of change within one or more categorical variables. In this case we are plotting number of packets (quantitative) on the y-axis against successive five-minute periods (ordered) with each line representing a category of device.

type="note"

Log Scales for Logs

Figure 6.11 has the y-axis plotted on a logarithmic scale. Notice how the values on the axis are increasing exponentially? If this plot was done on a normal linear scale, we’d see the workstation traffic at the top and then the other three lines would be reduced to visually zero. We changed to a linear scale because the tremendous skew in this data. By changing the scale to be logarithmic, we can see the patterns at the lower end of the spectrum. However, we have to be careful when we change to a log scale. We are used to seeing linear scales and we mentally will do the comparison like that. For example, we may think that the networking equipment here is about half the traffic of the workstations because it’s visually about half of the workstations. But in reality workstations are **generating about 10,000 times the traffic** network devices are and if the logarithmic scale isn’t clear to the consumer they could draw some incorrect conclusions.

Building Bar Charts

Bar charts are one of the most effective ways to communicate quantities of categories. There are a few variations on the basic bar chart. Figure 6.12 is showing three different ways of displaying vulnerability counts and severity classification per device. On the far left we have a typical bar chart with vertical bars. A simple modification of this is making the bars horizontal and the difference is largely for aesthetics and the context of where the chart will appear. The vertical bar chart is simple and shows the totals within each device type. This uses the common scale for comparison but we have the added feature of length. We can easily see that workstations have the most vulnerabilities and servers are close though with maybe 20% less or so. Then networking devices and printers are quite small in comparison.

Figure 6.12 Bar Charts: Vulnerability Counts [FILENAME 793725c06f012]

The other two bar charts have another categorical variable added to show severity of the vulnerability. The stacked and group bar chart applies a unique sequential color per severity and shows them slightly differently. With the stacked bar charts, we are still able to compare totals. It’s still clear that workstations have more vulnerabilities than all others. But comparing across severity is difficult as we lose the common scale. For example, attempt to visually compare the high vulnerabilities of workstations to servers. Since they are not aligned we purely judging by length on a non-aligned scale and we are less accurate. Now look at the grouped bar chart and it quickly becomes clear that servers have more high-severity vulnerabilities than workstations. The one draw back to the grouped bar chart is that we lose the overall count comparison. It is more difficult to tell that workstations have more vulnerabilities overall from grouped bar chart. The type of bar chart we choose is largely dependent on the message we are trying to send.

Leveraging Opacity

There is one more aspect we want to fit in here and that is leveraging the opacity or transparency of colors within graphs. If the data is overlapping or dense and we plot it with a solid opaque color we have no way of knowing just how many points are stacked up underneath that. Luckily, we can simply make the color we choose transparent. This will allow any points beneath to show through. Within R there are two methods of doing this. First within ggplot2 most of the geom’s allow for an alpha setting between 0 and 1. Or we can code the alpha right into the color with a 4th byte, meaning a red value of #FF0000 is the same as #FF0000FF (with the last FF setting opacity to maximum). If we want to set opacity to 50%, 255/2 = 128 = 0x80, so we can set the color to #FF000080 and now our red color is 50% opaque. This is far easier to see in Figure 6.13.

Figure 6.13 Bubble Chart: Opacity Shows stacking [FILENAME 793725c06f013]

Each graphic is showing the same 8 hours of firewall data for networking devices split into 5-minute totals. We are plotting the number of network sessions along the x-axis and the number of bytes on the y-axis. But, we have a lot of overlapping points. By setting the alpha value to 1/3 in the right picture we can see “through” the top level and get a glimpse of what’s underneath it. We’ve found it’s handy to set the alpha as a fraction (instead of .33 here) for our own benefit because an alpha of 1/3 means when stack up 3 values, it will appear as a solid color. This allows us to tweak the alpha for how many layers we have. If we think we have fifty layers deep (some of the maps in chapter 5 leverage small alpha values like this), we can set the alpha to 1/50 (as opposed to converting to 0.02 and typing that in).

Size encoding

We are encoding another quantitative variable in Figure 6.13 by mapping the size (*area*) of the circle to the number of packets in the same 5-minute period. Looking back at our accuracy chart we see that *area* is relatively low on the list and we’re compounding the problem here by not including a legend for the size. But all we want to communicate is the relative values here. In a real visualization we’d want to add a description to the title or use some other annotation to indicate the significance of the “bubble” size. For this purpose of this exercise, we are simply looking for any obvious patterns and this type of graphic shows relative sizes. Bubble charts like this serve a relatively crude purpose and are often downgraded to the level of pie charts for most people.

Since we’re hanging around at the bottom of Cleveland and McGill’s accuracy chart we might as well talk about another visualization that relies on area. Figure 6.14 is known as a treemap and it uses the size of rectangles to communicate a quantity and the color of the rectangle to communicate a different quantitative variable. Often times the rectangles are visually grouped to depict categorical relationships. Figure 6.14 attempts to group workstations, servers and networking devices and communicate the quantity of devices on the network with size, and the normalize amount of traffic they produce into color.

Figure 6.14: Treemap: Devices and Traffic on our Network [FILENAME 793725c06f014]

To clarify and simplify the definition: a treemap uses *area* and *color* to encode two *quantitative* values. In other words, a treemap combines two relatively inaccurate methods of encoding quantities. This makes treemaps difficult to do well and often confusing to consumers. The same rule applies to treemaps as for pie charts and bubble plots: there are usually better visualization methods to communicate the data.

Communicating Distributions

Histograms and Density Plots

Sometimes we just want to show the values within single variable and how they are distributed. Within classical statistics, we have descriptive statistics that attempt to reduce a distribution of numbers to single descriptive values. For example, if we go back to the 8-hours of firewall data, we could describe the distribution of total sessions within each 5-minute window like this:

|  |  |
| --- | --- |
| Description | Statistic |
| Min | 265,800 |
| Median | 356,500 |
| Mean | 350,500 |
| Standard Dev. | 32,093 |
| Max | 410,700 |
| Skew | -0.5 |
| Kurtosis | -0.457 |

Most people won’t be able to look at those numbers and understand what the data actually looks like. Nor will they be able to see any subtle patterns since descriptive statistics are about reducing a distribution of values to single numbers. This is where visualizations can help out considerably. Figure 6.15 shows a basic histogram on the left and a density plot on the right, both are showing the exact same distribution.

Figure Histogram and Density Plot: Firewall Sessions [FILENAME 793725c06f015]

A histogram uses a simple process called *binning.*  It works by creating equally spaced “bins” and then counting how many of our measurements are in each “bin”. In this example, we created bins that are 12,000 sessions wide. We can see at the peak, around 350,000 sessions, that we had about 18 sessions within that bin. Part of the criticisms of histograms is that we can affect how histograms appear by adjusting the size and position of the bins. But these plots are indispensable when we just want to get a feel for a distribution, as they are quite effective in communicating the basic shape.

The plot on the right in Figure 6.15 is a density plot. It uses the same approach as the histogram, but the bins are quite small and a smoothing process is applied over it. By projecting the original histogram behind it, we can see how it flattens the peaks and diminishes the valleys. There’s no right or wrong between the two. While we are exploring our data, it’s quite easy to pass in our data to hist() and get an immediate (though not all that pretty) histogram.

Boxing in Boxplots

Another method, which was developed by John Tukey (remember him from Chapter 1?), is the box plot. This is not something people will intuitively understand if they haven’t seen it before, so it may require a little more supporting material than other methods. In the fall of 2012, Jay had set up a very simple honeypot to just record the packets it saw on the Internet. How often is a host scanned when it’s on the Internet? We can get a feel for that as we show what a boxplot is able to communicate in Figure 6.16.

Figure Boxplot: Honeypot traffic [FILENAME 793725c06f016]

The boxplot begins with the median value of the distribution and it places the center bar there. Then it computes the 25th and 75th percentile. Meaning that 25% of the data is below the 25th percentile, 25% of the data is above the 75th percentile and 50% of the data is between the two. These two points form the length of the box and represent the *inter-quartile range*  or IQR*.*

There are a few different methods to represent the length of the lines, the most common is to place them one and half times the IQR away from the box. Other methods will place the end of the line at the minimum and maximum of the data. Figure 6.17 attempts to convey a large number of distributions within one graphic with boxplots.

Figure 6.17 Boxplots: Opportunistic Packets [FILENAME 793725c06f017]

What’s interesting about Figure 6.17 is that it was generated with over 100 million values. It not only conveys a large quantity of data, but it’s also able to represent confidence. In this case, just stating the mean or median would have been a disservice, since some of these have a very wide range of possible observations. How well could we have explained these values and the variations with anything other than a visualization of the distributions?

Visualizing Time Series

We have glossed over time series data in this chapter even though we’ve been working with it in most of our visualizations. Time series data are data collected over the same and repeated time intervals. For most of the firewall graphics in this chapter we parsed the log files and counted up the bytes, sessions and packets within each five-minute window of time. This allows aggregation of individual entries into more manageable data points. But depending on how we slice up time and aggregate the data, we can get and see different types of things.

Figure 6.18 Time Series: 21 days of traffic [FILENAME]

Figure 6.18 is looking at 21 days of firewall traffic sliced it into 5-minute chunks. This is quite a bit of data for a small line graph (over 6,000 data points in a few inches), and when we try to represent that data with a line plot, the lines are crisscrossing over one another so much that they look like one thick and jittery line. If we try to reduce the mess by simplifying the underlying data with an hourly average (middle plot in Figure 6.18), we lose the extremes and the details, which is not generally good in an industry where extremes matter. In the bottom plot, we replaced the lines in the first plot with points. This removes much of the mess and allows us to see both the general trends and the extreme points.

Time series data can get very dense to visualize when we are talking about log data. We even made it easier on ourselves by looking at five-minute slices instead of one-minute slices. How we prepare and visualize the data is dependent on what we are looking for in the data. If we are looking for specific spikes or gaps in traffic then a rolling average should be avoided, but if we want to understand general patterns, maybe averages are called for. We’ve covered quite a few techniques so far in this chapter. Feel free to get creative and try one or more techniques on your time series data. What we if tried showing each hour with a box plot? What if we used larger points and varied color based on size and turned down the alpha? Good visualizations are generally an iterative process, so take this as a license to experiment! Remember that we aren’t limited to static visualizations. We can create interactive visualizations (as we’ll see Chapter 11) or turn our time series into a video fit for YouTube, as well see in the next section.

Turning Your Data Into A Movie Star

We have focused primarily on foundational components of data visualizations. These will apply to static or interactive graphics, dashboards and as we’ll see, videos as well. One of the more fun “tricks” we’ve learned is how to turn our data into a video. In order to do this, we combine two techniques: automated sequential graphics and stop-motion software. If you aren’t familiar with stop-motion by name, you’re certainly familiar with it by sight. It’s the Claymation technology of setting up a scene, taking a picture and then changing it slightly, taking another picture and so on. When we string all of those pictures together we get the appearance of motion and we have a video. Same concept here, but instead of taking a picture, we want to generate a graphic and save that off as a picture. Then we use any number of stop-motion software packages (mencoder, ffmpeg, iMovie, etc.) to create a movie out of the pictures. If you’d like to get fancy with that most software packages allow including music or doing voice-overs so you can explain the data as it’s progressing.

For a sample of how this looks, try out this snippet of code in an open R session.

# random walk

**set.seed(1)**

# set up nine directions

**dirs <- matrix(c(rep(seq(-1, 1), 3),**

**rep(seq(-1, 1), each=3)), ncol=2, byrow=T)**

# start in the center

**cpos <- matrix(c(0, 0), ncol=2)**

# set full screen

**par(mar=c(0,0,0,0))**

**for(i in seq(200)) {**

**plot(cpos, type="p", col="gray80", xlim=c(-20, 20), ylim=c(-20,20),**

**yaxt="n", ann=FALSE, xaxt="n", bty="n")**

**cpos <- rbind(cpos, cpos[nrow(cpos), ] + dirs[sample(1:9, 1), ])**

**points(cpos[nrow(cpos), 1], cpos[nrow(cpos), 2],**

**type="p", pch=16, col="red")**

**Sys.sleep(0.1)**

**}**

# reset screen back to default

**par(mar=c(5.1,4.1,4.1,2.1))**

This code will setup a matrix of nine directions. Then loop 200 times, adjusting the point in some random direction, and drawing the new plot for it and sleeping for a tenth of a second so you can view the plot. On all but slow machines this looks like a random walking point on the screen. If you’d like a challenge, modify this script to write out each of the images (hint, take a look at help(png)) and then create a video of it. We’ve done this and it’s available on the book website if you’d like to see our wandering random walk in action!