4 ANALYTICAL INTERACTION AND NAVIGATION

Although at times we sit silently in thought when analyzing data, most of the process requires dynamic interaction as we navigate from a state of unknowing to one of enlightenment.

Analytical Interaction

We can only learn so much when staring at a static visualization such as a printed graph. No matter how rich and elegant the display, if it's evocative it will invite questions that it wasn't designed to answer. At that juncture, if we can't interact with the data to pursue an answer, we hit the wall. The effectiveness of information visualization hinges on two things: its ability to clearly and accurately represent information and our ability to interact with it to figure out what the information means. Several ways of interacting with data are especially useful. In this chapter, we'll examine the following 13:

- Comparing
- Sorting
- Adding variables
- Filtering
- Highlighting
- Aggregating
- Re-expressing
- Re-visualizing
- *Zooming and panning*
- Re-scaling
- Accessing details on demand
- Annotating
- Bookmarking

Let's look at each type of interaction in detail.

Comparing

No interaction is more frequent, useful, and central to the analytical process than comparing values and patterns. An old joke goes something like this: A therapist asks a woman "How is your husband?" to which she replies "Compared to what?" Comparison is the beating heart of data analysis. In fact, what we do when we compare data really encompasses both comparing (looking for similarities) and contrasting (looking for differences). In this book I use the term comparing loosely to cover both of these actions.

Comparing magnitudes—for example, this is greater or less than that and by what amount—is a fundamental version of this activity. The following graph supports this activity, making it easy to compare the performance of salespeople to one another.

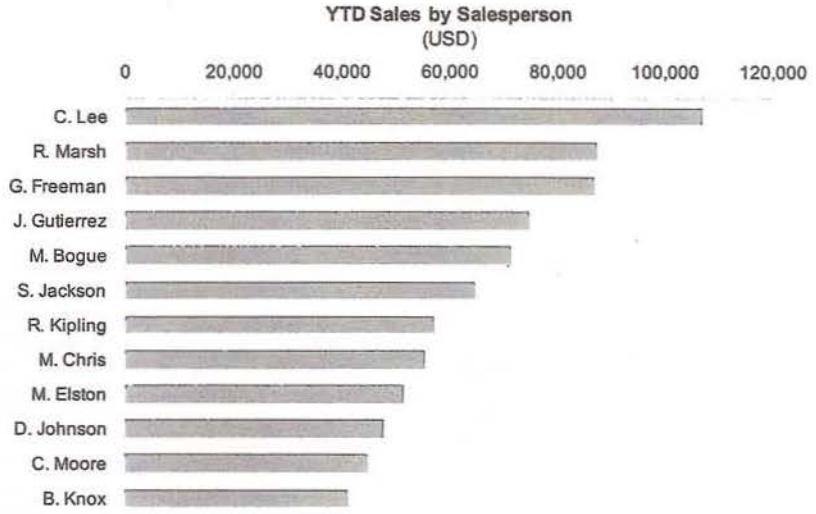
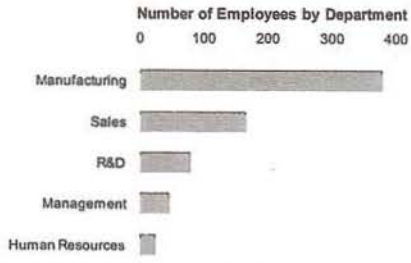
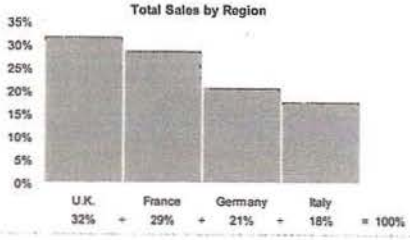
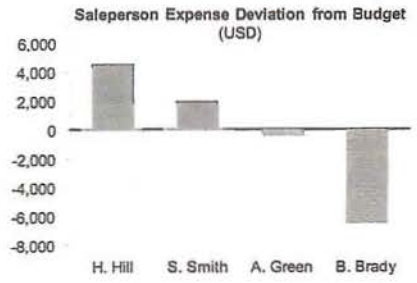
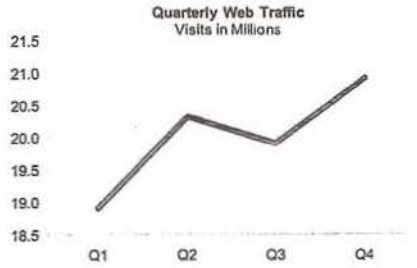


Figure 4.1

A few typical magnitude comparisons are:

Type	Description	
Nominal	Comparing values that have no particular order	
Ranking	Comparing values that are arranged by magnitude, from low to high or high to low	
Part-to-Whole	Comparing values that when combined make up parts of a whole	

Type	Description	
Deviation	Comparing the differences between two sets of values	
Time-series	Comparing measures that were recorded at different points in time to see how they changed	

In each of these cases, we are simply comparing the magnitudes of one value to another; in this sense, the cases are all the same. They differ, however, in the meanings that we can discover from the comparisons.

At the next level up in complexity, we compare patterns formed by entire series of values. The following graph makes it easy to compare domestic and international sales through time, exhibited in several patterns of change, including overall trends throughout the year and seasonal patterns.

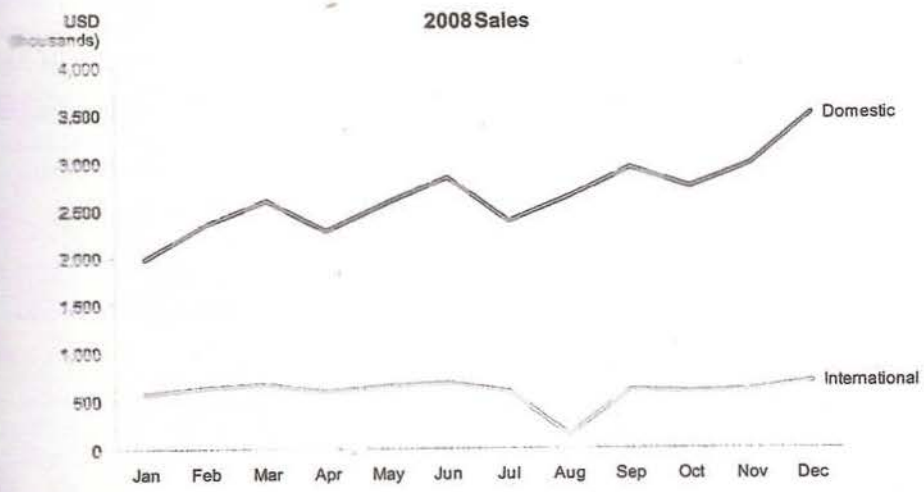


Figure 4.2

Different patterns are meaningful, depending on the nature of the data and what we're trying to understand. For instance, patterns that we might find in scatterplots while examining the correlation between two quantitative variables

are usually different from patterns that might surface as meaningful while we're examining how a set of values is distributed from lowest to highest, as illustrated below:

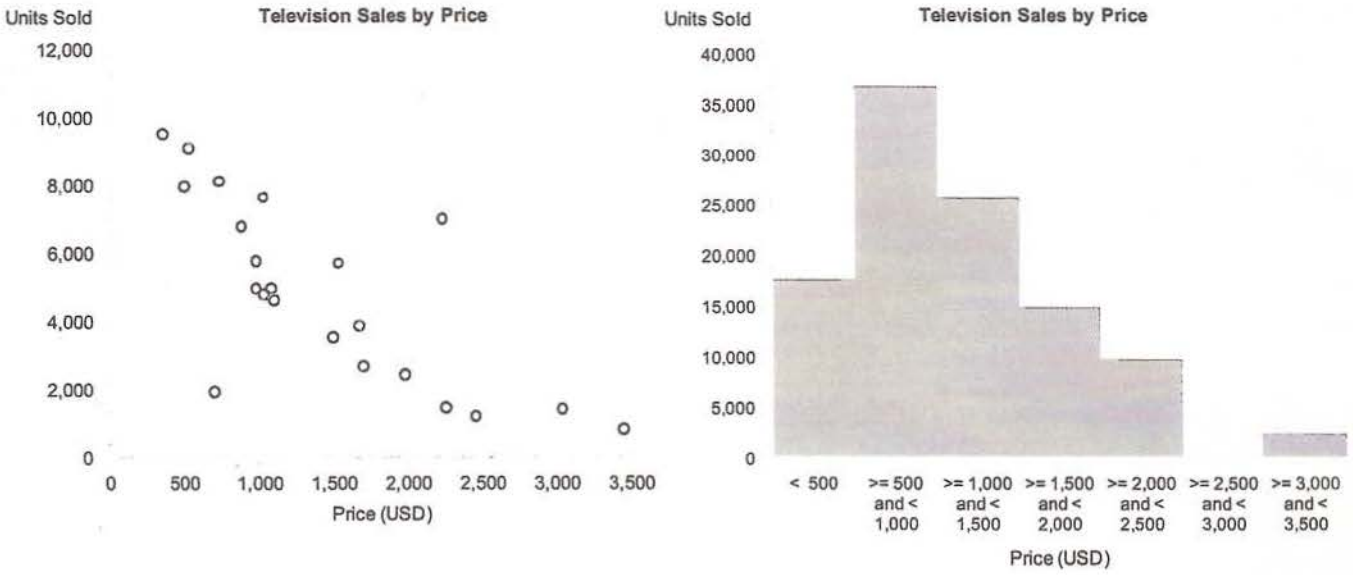


Figure 4.3

In later chapters, when we explore ways to perform particular types of analysis, such as time-series and distribution, we'll examine the particular types of comparisons and patterns that are meaningful in each.

Information visualization software should support comparisons in the following ways:

- Provide a selection of graphs that support the full spectrum of commonly needed comparisons
- Provide graphs that are designed for easy comparison of those values and relevant patterns without distraction
- Provide the means to place a great deal of information that we wish to compare on the screen at the same time, thereby avoiding the need to scroll or move from screen to screen to see the information

Many data analysis products fail to fully support comparisons of distributions because they don't provide box plots. Every general-purpose quantitative analysis product ought to include them. Another way that products often fail to support useful comparisons is by poorly rendering the objects that must be compared. Perhaps the most common example of this involves three-dimensional (3-D) graphs. Notice how difficult it is to compare the magnitudes of the values that are encoded as bars in the graph on the following page:

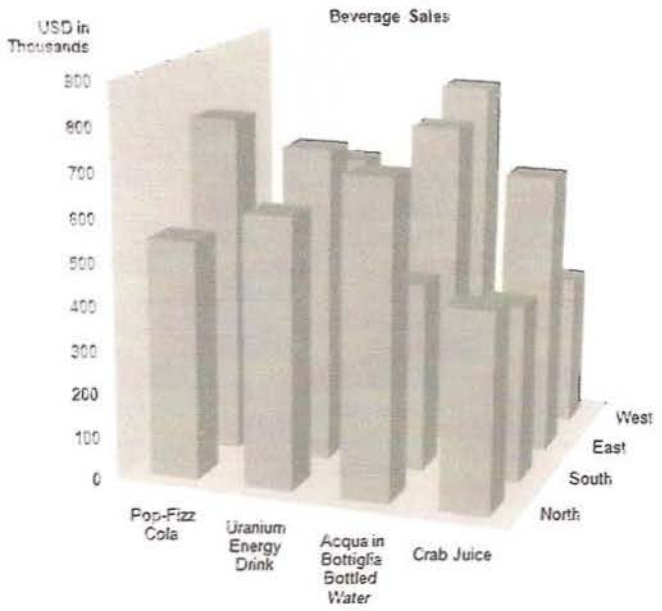


Figure 4.4

This graph suffers from occlusion: some bars are hidden behind others, which makes it impossible to compare them. When I complain about this to vendors, they often explain that this isn't a problem at all, because the graph can be rotated in a way that would allow the hidden bars to be seen. In addition to the fact that this is time-consuming and cumbersome, it undermines one of the fundamental strengths of graphs: the ability to see everything at once, which provides the big picture of relationships that we often need. Look at the 3-D graph below and try your best to interpret the values and compare the patterns of change through time.

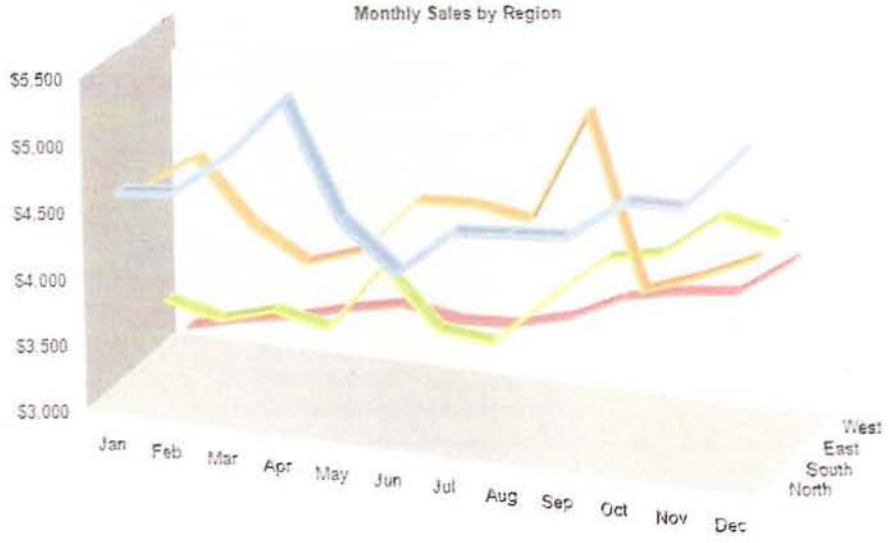


Figure 4.5. The east region is represented by the green line, but few people in my classes get this right when asked to guess.

Not only can you not interpret and compare what is going on in this graph, you probably can't tell which of the four lines represents the east region. If you don't even know which line represents which region, what good is the graph?

Even when a graph only has two axes, X and Y, if the objects that encode the data are rendered three dimensionally, the task of comparing the values is more difficult. Notice that it is easier to compare the bars in the left (2-D) graph below than those in the other two (3-D) graphs.

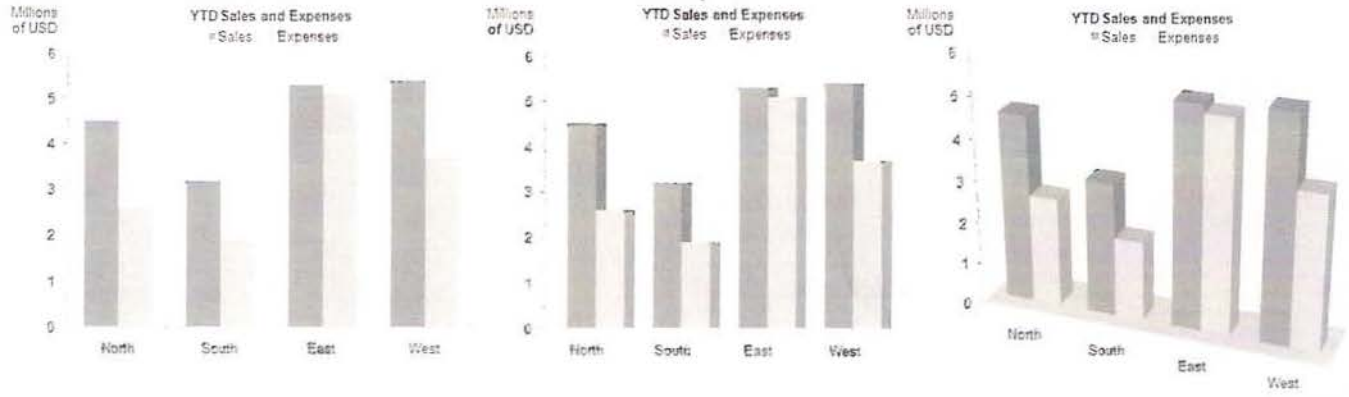


Figure 4.6

One other typical problem that undermines our ability to compare values accurately is illustrated in the next graph. How much greater is the number of "Yes" responses than "No" responses?

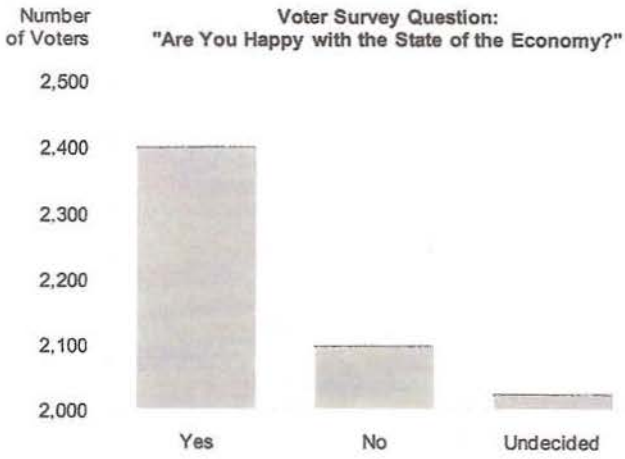


Figure 4.7

The relative heights of the bars suggests that "Yes" responses are four times greater than "No" responses, but this is not the case. When bars are used to encode values, their heights (vertical bars) or lengths (horizontal bars) will only represent the values accurately when the base of the bars begins at a value of zero. If we narrow the quantitative scale so that the bars begin at some value other than zero, their relative heights or lengths can no longer be accurately compared without first reading their values along the scale and doing math in our heads. In other words, a table of the same values could be used more efficiently to make these comparisons. There is no reason to use a graph unless its visual components can be used to make sense of the data. Software should not allow us to make the mistake illustrated above or at the very least should make it difficult to produce such a graph.

Another common comparison when analyzing data is making the distinction between values that appear normal—that is, within the quantitative range where most of the values are located—and those that appear abnormal. Values that fall outside the norm are called outliers or exceptions. When values are displayed in a well-designed visualization, it is usually easy to spot outliers and always worthwhile to examine them.

Sorting

Don't underestimate the power of a simple sort. It's amazing how much more meaning surfaces when values are sorted from low to high or high to low. Take a look at the following graph, which displays employee compensation per state:

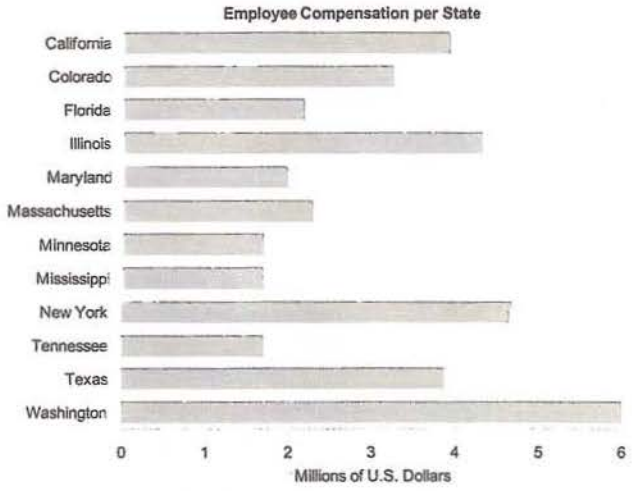


Figure 4.8

With the states in alphabetical order, the only thing we can do with ease is look up employee compensation for a particular state. It is difficult to see any meaningful relationships among the values. Now take a look at the same data, this time sorted from the state with the highest employee compensation to the one with the lowest.

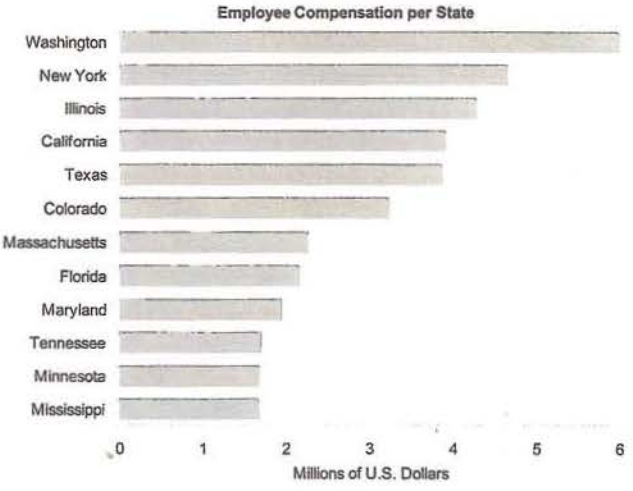


Figure 4.9

Not only is the story of how the states' compensation values relate to one another now clear, it is also easier to compare one value to another simply because values that are close to one another in magnitude are located near one another.

Let's go a step further and add another variable to the mix. In this next example, states are still sorted according to employee compensation, but a new column of bars has been added to display the number of employees for each state. Because the states are sorted by employee compensation to form a series of bars that decline in size from top to bottom, we can easily see that the related counts of employees per state do not perfectly correlate to compensation. This must be due to differences in how much employees are compensated, on average, in various states. For example, we can notice that although California and Texas pay roughly the same amount in total compensation, Texas has more employees (so presumably employees in Texas are paid less than those in California as roughly the same total amount of compensation stretches to cover more people in Texas). This graph is an illustration of how sorting can be used to examine multiple variables and analyze how the variables are correlated, if at all.

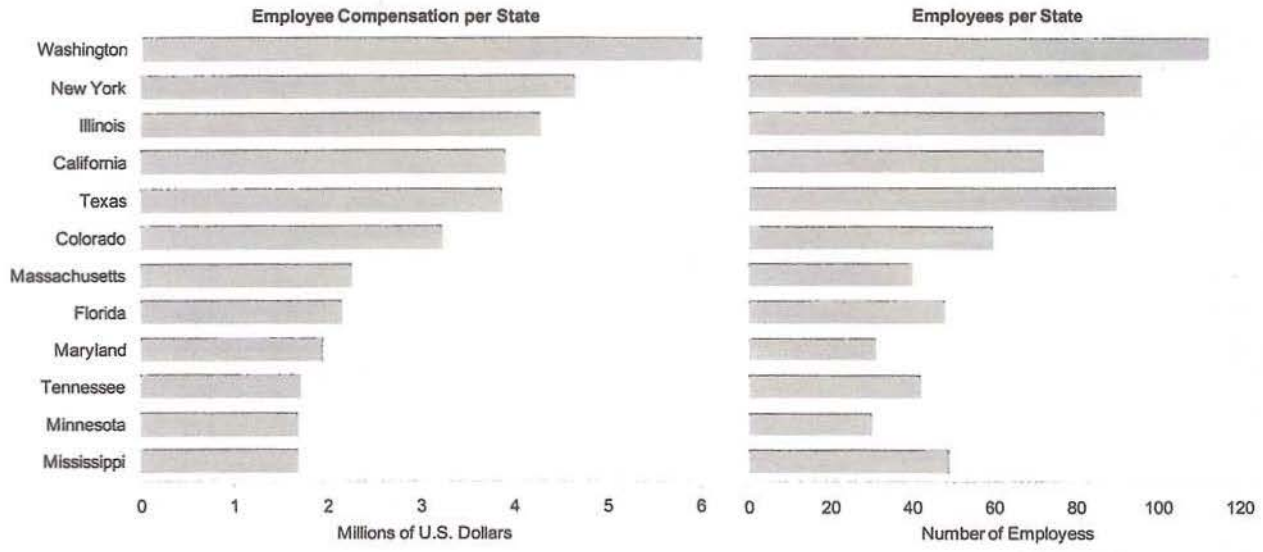


Figure 4.10

Information visualization software should support sorting in the following ways:

- Provide the means to sort items in a graph based on various values, especially the values that are featured in the graph
- Provide extremely quick and easy means to re-sort data in different ways, ideally with a single click of the mouse
- Provide the means to link multiple graphs and easily sort the data in each graph in the same way, assuming that the graphs share a common categorical variable (for example, state)

Adding Variables

We don't always know in advance every element of a data set that we'll need during the process of analyzing it. This is natural. Data analysis involves looking for interesting attributes and examining them in various ways, which always leads to questions that we didn't think to ask when we first began. This is how the process works because this is how thinking works. We might be examining sales revenues per product when we begin to wonder how profits relate to what we're seeing. We might at that point want to shift between a graph such as the one below on the left, to a richer graph such as the one on the right, which adds the profit variable.

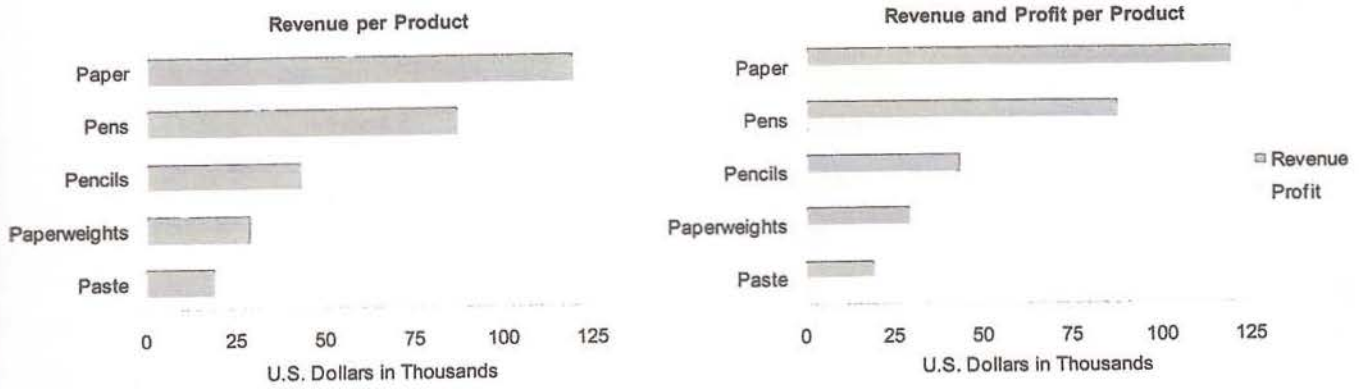


Figure 4.11

If we're examining revenue per product, sorted by product type from the best selling to worst as in the following example, we might decide to see how products are ranked by revenue without regard to product type.

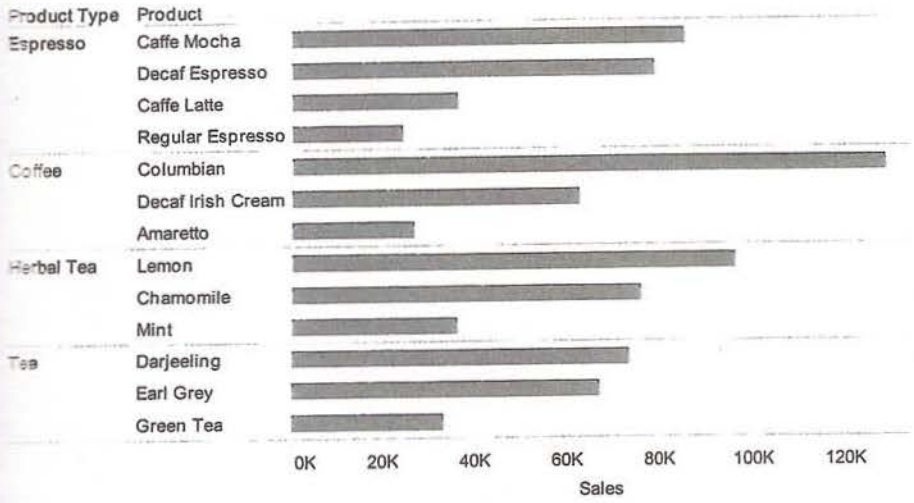


Figure 4.12. Created using Tableau Software

To do so, we would like a way to quickly remove product type from the display, switching from the view above to the one on the following page.

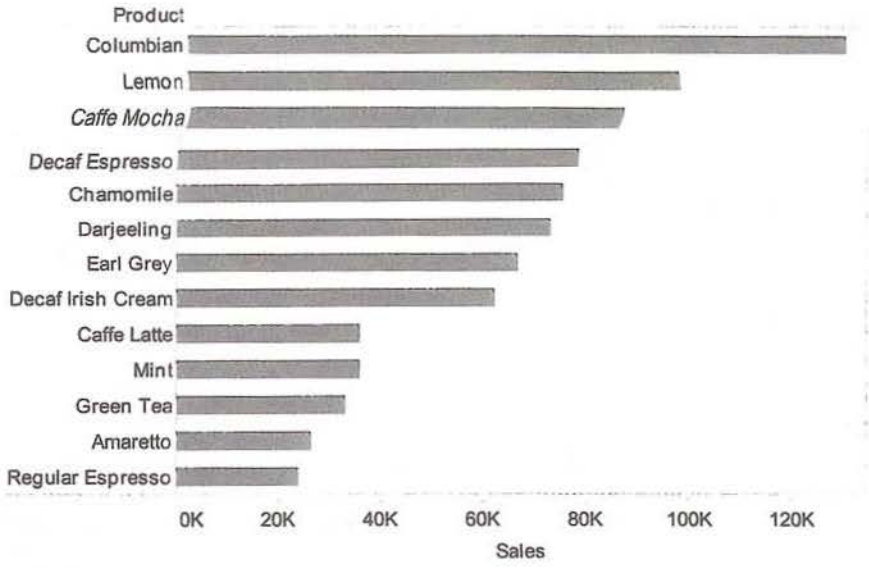


Figure 4.13. Created using Tableau Software

Information visualization software should support adding and removing variables in the following ways:

- Provide convenient access to every available variable that might be needed for analysis.
- Provide easy means to add a variable to or remove one from the display, such as by directly grabbing the variable and placing it or removing.

Filtering

Filtering is the act of reducing the data that we're viewing to a subset of what's currently there. From a database perspective, this involves removing particular data records from view. This is usually done by selecting particular items within a categorical variable (for example, particular products or regions) or a range of values in a quantitative variable (for example, sales orders below $20) and indicating that they (or everything but them) should be removed from view. Sometimes we do the opposite by restoring to view something that we previously removed. In both cases, we are working with filters, either by filtering (removing) or unfiltering (restoring) data.

The purpose of filtering is simple: to get any information we don't need at the moment out of the way because it is distracting us from the task at hand. On the next page, notice how much more easily we can examine and compare sales of shirts and pants in the right-hand graph when information regarding suits, coats, and shoes is no longer competing for attention as it is on the left.

Figure 4.14

Information visualization researchers refer to filters that operate in this manner as *dynamic queries*, based originally on work by Ben Shneiderman published as "Dynamic Queries for Visual Information Seeking," *IEEE Software*, 11(6), 70-77, 1994.

A great innovation of recent years is the development of filter controls that are so fast and easy to manipulate that we can apply a filter almost without taking our eyes off of the data. Below is a simple example of a filter control that uses radio buttons to filter regions.

Figure 4.15. Filter from Spotfire

The next example of a filter control is called a *slider*. This type of control of is especially useful for filtering ranges of quantitative values. The example below actually contains two sliders in a single control: one for the low end of the range ($1) and one for the high end ($447).

Figure 4.16. Filter from Spotfire

This allows us to select the precise range of sales orders that we wish to view by order amount, filtering out all other orders even if the range we want to view is in the middle (illustrated below) rather than at one of the ends of the scale.

Figure 4.17. Filter from Spotfire

Information visualization software should support filtering and unfiltering in the following ways:

- Allow easy filtering based on any information in the connected data sources, not just based on information that is currently being displayed. For example, even if the display only includes products, months, regions, and sales revenue, we might want to filter out all products belonging to a particular product type or all orders that were placed via the Web rather than by phone.
- Allow data to be filtered rapidly using simple controls, such as checkboxes or sliders. The lag time between issuing the filter command and seeing the results should be almost unnoticeable.
- Provide the means to directly select items in a graph (such as by using the mouse to click on them) and then remove them from display with a click or two.
- Once a filter has been applied, give some visible reminder that the filter is in effect along with an easy means to look up what has been filtered.
- Provide a means to define complex filter logic with multiple conditions (for example, remove all hardware products, except for those in the state of Colorado).
- Allow multiple graphs to be conveniently linked such that all can be filtered together in the same way in a single action.

Highlighting

Sometimes, rather than filtering out data we aren't interested in at the moment, we want to cause particular data to stand out without causing all other data to go away. Highlighting makes it possible to focus on a subset of data while still seeing it in context of the whole. At times, this involves data in a single graph only. In the following example, I have highlighted data points in red belonging to customers in their 20s who purchased products, without throwing out the other age groups. In this particular case, highlighting rather than the filtering allows us to see the relationship between the total number of purchases (along the X-axis) and the amount spent on groceries (along the Y-axis) by people in their 20s, in fairly good isolation from other age groups while still being able to see how their shopping habits compare to those of customers overall.

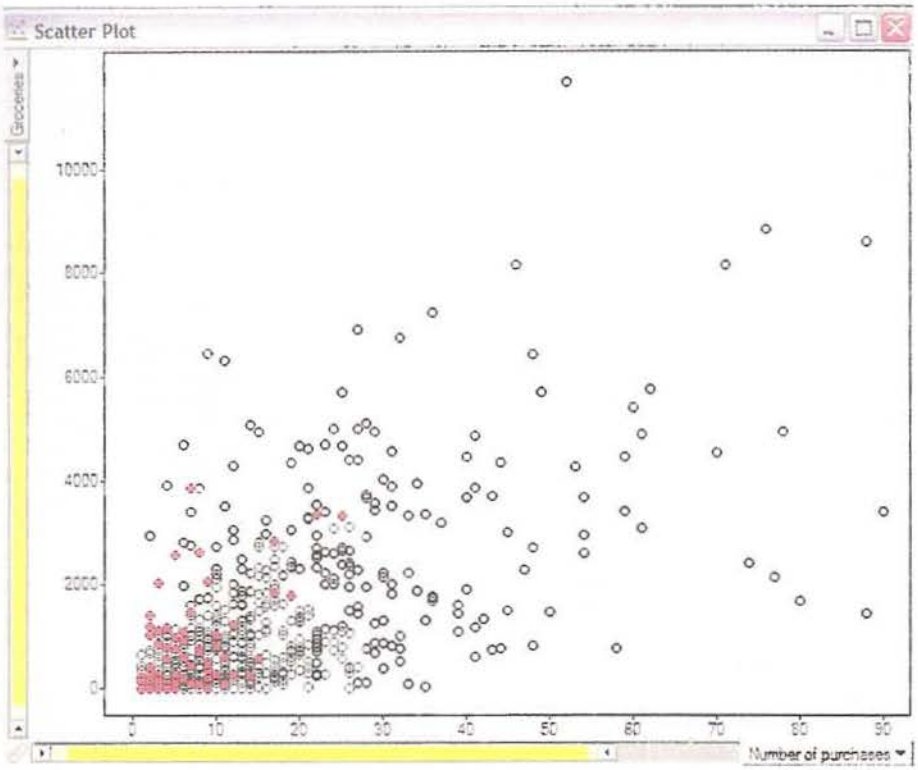


Figure 4.18. Created using Spotfire

Sometimes we need to highlight the same subset of data in multiple graphs at the same time. This is especially useful when we are simultaneously viewing several graphs, each of which displays the same set of data differently. For instance, consider the example on the following page in which we'll simultaneously view five graphs: the scatterplot from above, a bar graph with total purchases by age group, a bar graph with female vs. male purchases, a bar graph with purchases per store, and a bar graph with purchases based on the date of customers' initial purchases from the stores. We could, using this set of graphs, track the subset of customers who made their first purchases during the last three months (highlighted in red in the bottom bar graph) throughout the customer attributes featured in each of the other graphs. This technique of highlighting data in one graph, resulting in the same subset of data being highlighted in other associated graphs, is called *brushing* or *brushing and linking*. We'll look at brushing again more closely in *Chapter 5: Analytical Practices*.

Brushing was first introduced by W. S. Cleveland, R. A. Becker, and G. Weil in the paper "The Use of Brushing and Rotation for Data Analysis," originally published in *First IASC World Conference on Computational Statistics and Data Analysis*, International Statistical Institute, Voorburg, Netherlands,1988, pp. 114–147. It later appeared in the book *Dynamic Graphics for Statistics*, edited by W. S. Cleveland and M. E. McGill, published by Wadsworth and Brooks/Cole, Pacific Grove CA, 1988.

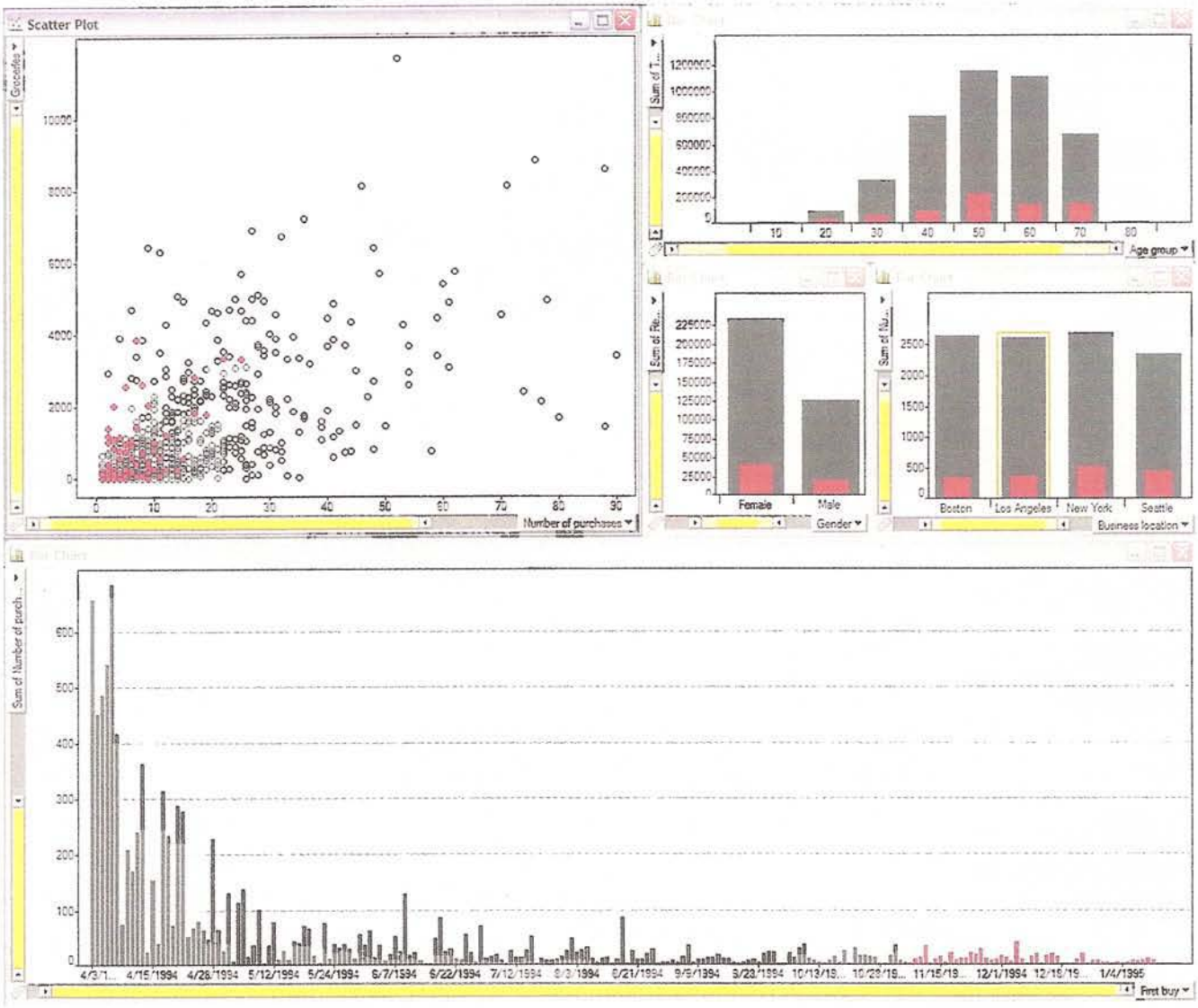


Figure 4.19. Created using Spotfire

Information visualization software should support highlighting in the following ways:

- Provide the means to highlight a subset of data by selecting from lists of categorical items (for example, departments from a list of those in the company) or from the ranges of values associated with quantitative variables (for example, expenses greater than $25,000).
- Provide the means to highlight a subset of data by directly selecting it in a graph, such as by using the mouse to draw a rectangle around the items or to click on each of the items.
- Highlight selected information so that it can be seen independently from the rest while still allowing viewers to see the entire set of data (both highlighted and not highlighted).
- Provide the means to highlight a set of items in one graph and have those same items automatically highlighted in other graphs that share the same data set.

Aggregating

When we aggregate or disaggregate information, we are not changing the amount of information but rather the level of detail at which we're viewing it. We aggregate data to view it at a higher level of summarization or generalization; we disaggregate to view it at a lower level of detail.

Consider the process of sales analysis. At its lowest level of detail, sales usually consist of line items on an order. A single order at a grocery store might consist of one wedge of pecorino cheese, three jars of the same pasta sauce, and two boxes of the same pasta. If we're analyzing sales that occurred during a particular month, most of our effort would not require knowing how many jars of the same pasta sauce were sold; we would look at the data at much higher levels than order line items. At times we might examine sales by region. At others, we might shift to sales by large groupings of products such as all pasta, grains, and rice products. Any time that a particular item looks interesting, however, we might dive down to a lower level of detail, perhaps sales per day, per individual product, or even per individual shopper. Moving up and down through various levels of generality and specificity is part and parcel of the analytical process.

Information visualization software should support aggregating in the following ways:

- Provide the means to easily aggregate quantitative data to the level of items in a categorical variable (for example, to the product type or regional level).
- Provide the means to easily aggregate data in a number of useful ways, especially summing (which should be the default), averaging (mean or median), and counting.
- Provide the means to easily aggregate data based on equal intervals of a quantitative variable. For example, if individual sales orders ranged in revenue roughly from $10 to $100, we might want to sum, average, or count the orders that fell within a series of $10 ranges ($10.00 through $19.99, $20.00 through $29.99, etc.).
- Process the transition from one level of aggregation to another without noticeable delay.
- Provide the means to create ad hoc groupings of items.

Although we usually aggregate data in conjunction with some categorical variable, resulting in a summarized value for each item that's associated with that variable (for example, for each individual product that a company sells), we sometimes want to combine particular items of a categorical variable into a single group and aggregate the data to that level. For instance, if we work for a company that groups products into firmly defined families, we might often want to aggregate values based on groups of products that belong to the same family. Let's say that our company sells hot beverages, both coffees and teas. Ordinarily, we group beverages into four families: espresso, coffee, tea, and herbal tea. However, today we want to analyze sales patterns related to caffeinated vs. decaf products. Because no such standard grouping exists, we might want to create an ad hoc grouping so we can aggregate sales based on that distinction.

A special type of aggregation and disaggregation that deserves individual attention is *drilling*. Drilling involves moving down levels of summarization (and also back up) along a defined hierarchical path. Some categorical variables relate to one another hierarchically in a multi-level arrangement of parent-to- child relationships. The hierarchical structure of any medium-to-large sales organization is a familiar example. The top of the hierarchy would be the total organization; the next level down might be several major geographical regions (such as the Americas, Europe, and Asia Pacific), followed by smaller regions (such as individual countries), followed by yet smaller regions (such as the individual states in the U.S.), followed by much smaller regions (such as counties), and ending with individual stores. Drilling during the process of analysis would involve moving from a higher to lower level, such as the country level to the state level of aggregation (drilling down) or the opposite, such as from the individual store level to the county level (drilling up).

Information visualization software should support drilling in the following ways:

- Provide the means to define hierarchical relationships among categorical variables.
- Provide the means to easily drill up or down through a hierarchy with no more than a click or two of the mouse.
- Provide the means to skip levels in a hierarchy when drilling.
- Automatically define the natural hierarchy associated with time, including years, quarters, months, days, and hours.
- Process the transition from one level of a hierarchy to another without noticeable delay.

Re-expressing

Sometimes quantitative values can be expressed in multiple ways, and each expression can lead to different insights. By the term re-expressing, I mean that we sometimes change the way we delineate quantitative values that we're examining. The most common example involves changing the unit of measure, from some natural unit, such as U.S. dollars for sales revenues, to another, such as percentages. Examining each product type's percentage of total sales might lead to insights that did not come to light when we were viewing the same values expressed as dollars.

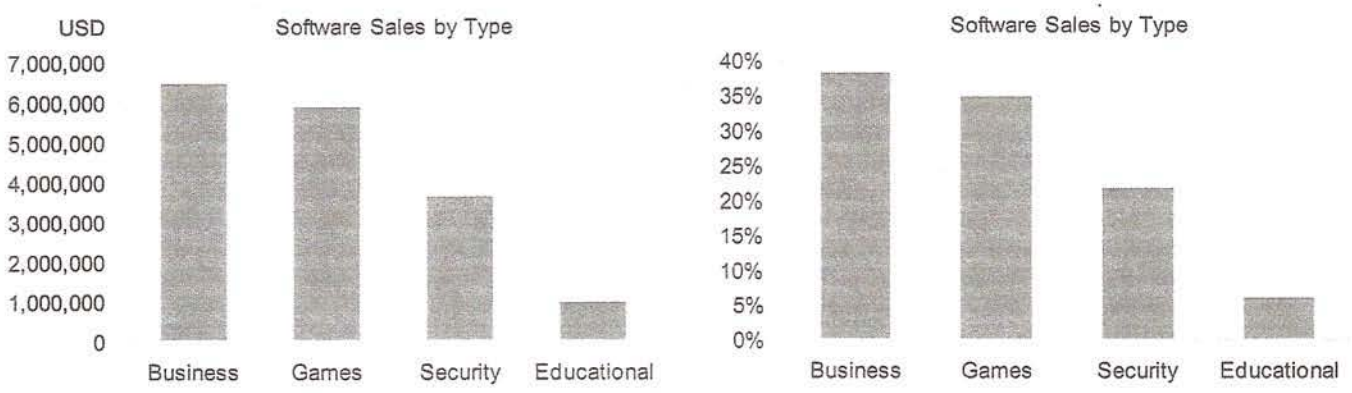


Figure 4.20

Re-expression can also take other forms. For instance, we might begin with the following graph, which paints a straightforward picture of change through time.

Figure 4.21

There might be an occasion when we would want to focus on the way that each month's sales compares to the sales in one particular month, such as January. We could use the graph above to do this, but it would take some work because this graph doesn't directly display this particular relationship between the monthly sales values. Look at the somewhat different perspective below where sales are re-expressed as the dollar amount of difference between each month and the month of January (January's sales are set at zero, and the line representing the rest of the months varies up or down in relation to January's value).

Figure 4.22

In this case, re-expression didn't involve changing the unit of measure (although it could have if we had chosen to express each month as its percentage difference from January's value). Rather, it involved a calculation that allowed us to focus on a different aspect of the year's sales data. Re-expression can take several forms, but only a few are common.

Sometimes it's useful to express time-series values as the difference between each interval's value and the value at some particular point in time, such as the month of January in the previous example. The reference value, however, could be the immediately prior period of time. The following graph displays the same data as the graph above, but this time each month is being compared to the immediately prior month and expressed as the percentage difference.

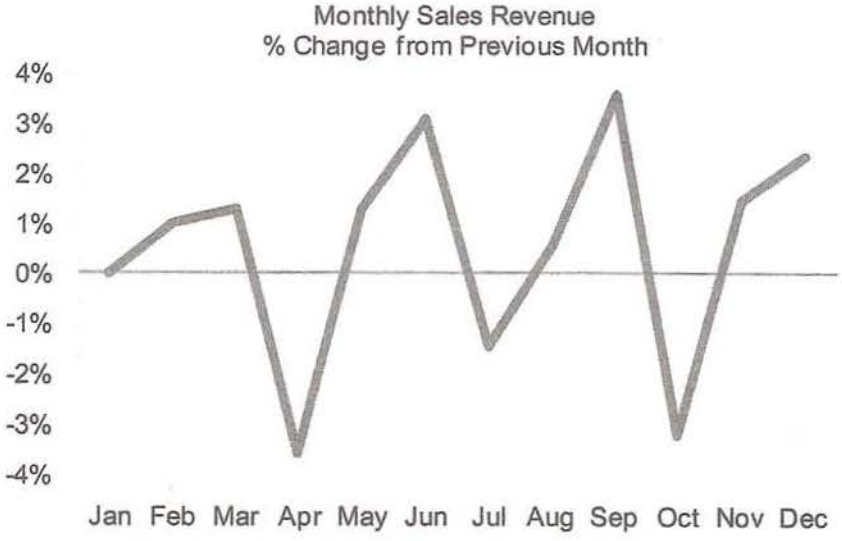


Figure 4.23

Information visualization software should support re-expression in the following ways:

- Provide an easy means to switch the current unit of measure in a graph to a percentage of the whole.
- Provide an easy means to re-express values in terms of how they compare to a reference value or as a rolling average.

At other times, especially when you wish to detect the general trend of what's happening through time, it's helpful to express values as a moving average, as in the following example that uses the same data as above but this time expresses each month's sales as the average of that month's sales and the previous two months' sales. This smoothes out some of the raggedness in the pattern, especially when values change radically from interval to interval, making it easier to see the overall trend.

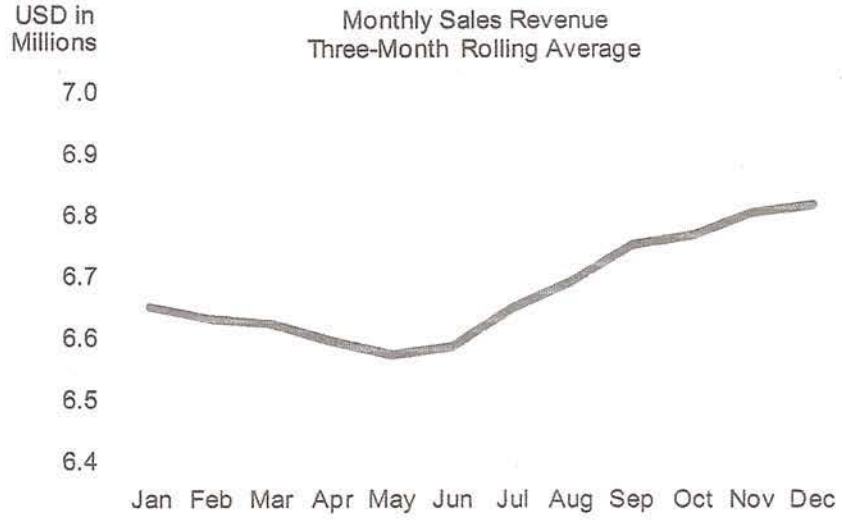


Figure 4.24

Re-visualizing

This activity pertains only to visual forms of analysis. It involves changing the visual representation in some fundamental way, such as switching from one type of graph to another. Being able to do this quickly and easily is essential. Bertin expressed the need well when he wrote: "A graphic is no longer 'drawn' once and for all: it is 'constructed' and reconstructed (manipulated) until all the relationships which lie within it have been perceived...A graphic is never an end in itself: it is a moment in the process of decision making."[1] No single way of visualizing data can serve every analytical need. Different types of visualization have different strengths. If we don't have the ability to switch from one to another as fast as we recognize the need, our data analysis will be fragmented and slow, and we will probably end the process prematurely in frustration, missing the full range of possible insights standing in the wings.

1. *Information Visualization*, Robert Spence, Addison-Wesley, Essex England, 2001, p. 15.

Imagine that we're comparing actual expenses to the expense budget for a year's worth of data using a bar graph. Bars nicely support magnitude comparisons of individual values, such as actual expenses to budgeted expenses.

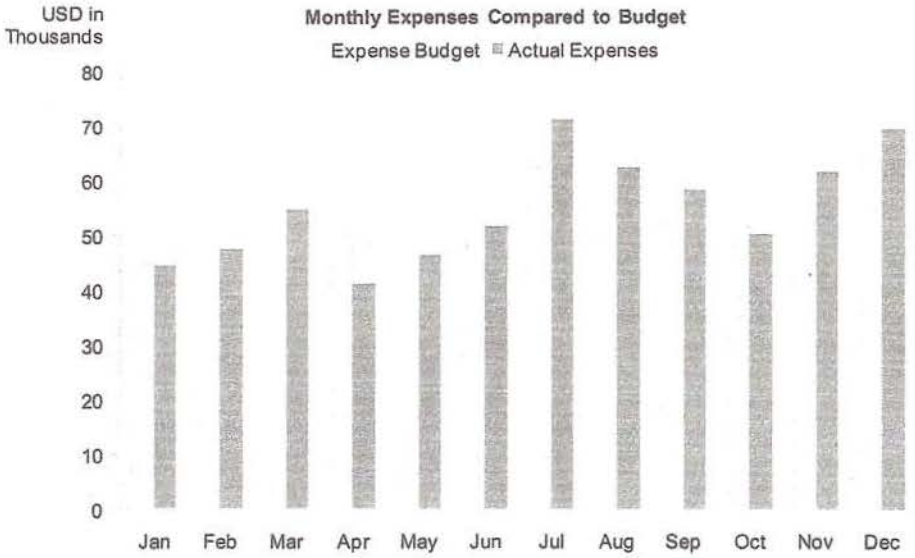


Figure 4.25

Before long, however, we want to see how the variation between actual and budgeted expenses changed through the year, which will be much easier if we switch from a bar to a line graph with a single line that expresses the difference between actual and budgeted expenses. We'll be grateful if we can switch the visualization from the one above to the one below with only a few keystrokes or movements of the mouse.

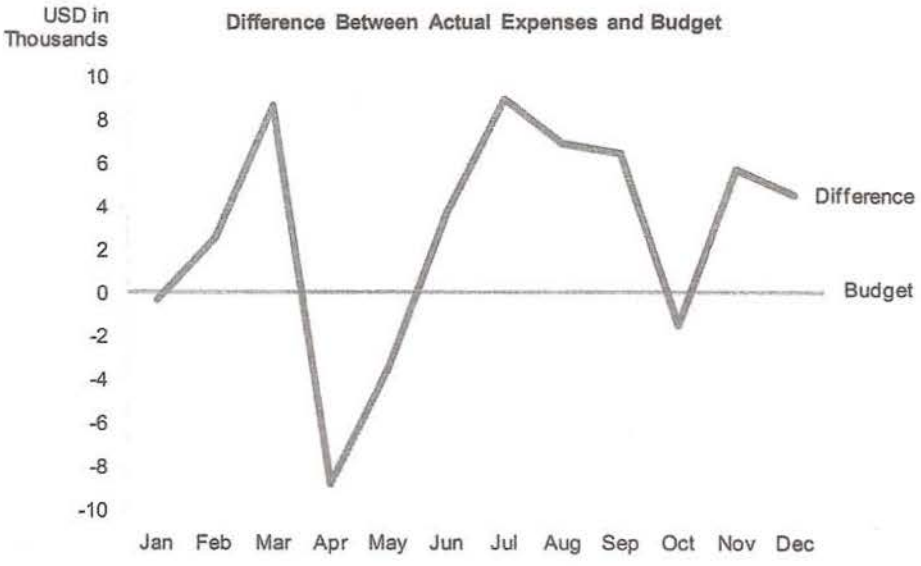


Figure 4.26

Information visualization software should support re-visualizing in the following ways:

- Provide a means to rapidly and easily switch from one type of graph to another.
- Provide a list of available graph types that is limited to only those that are appropriate for the data.
- Prevent or make more difficult the selection of a graph that would display the data inappropriately.

Zooming and Panning

When exploring and analyzing data visually, we sometimes want to take a closer look at a specific section of a graph. We can accomplish this by zooming in on the contents of a visualization, which enlarges the portion of the display that we wish to see more closely. The examples below illustrate this process. If we become particularly interested in what's happening during the specific period from February 14 through 20 while viewing the first graph below, we might want to zoom in on that portion, resulting in the bottom graph below.

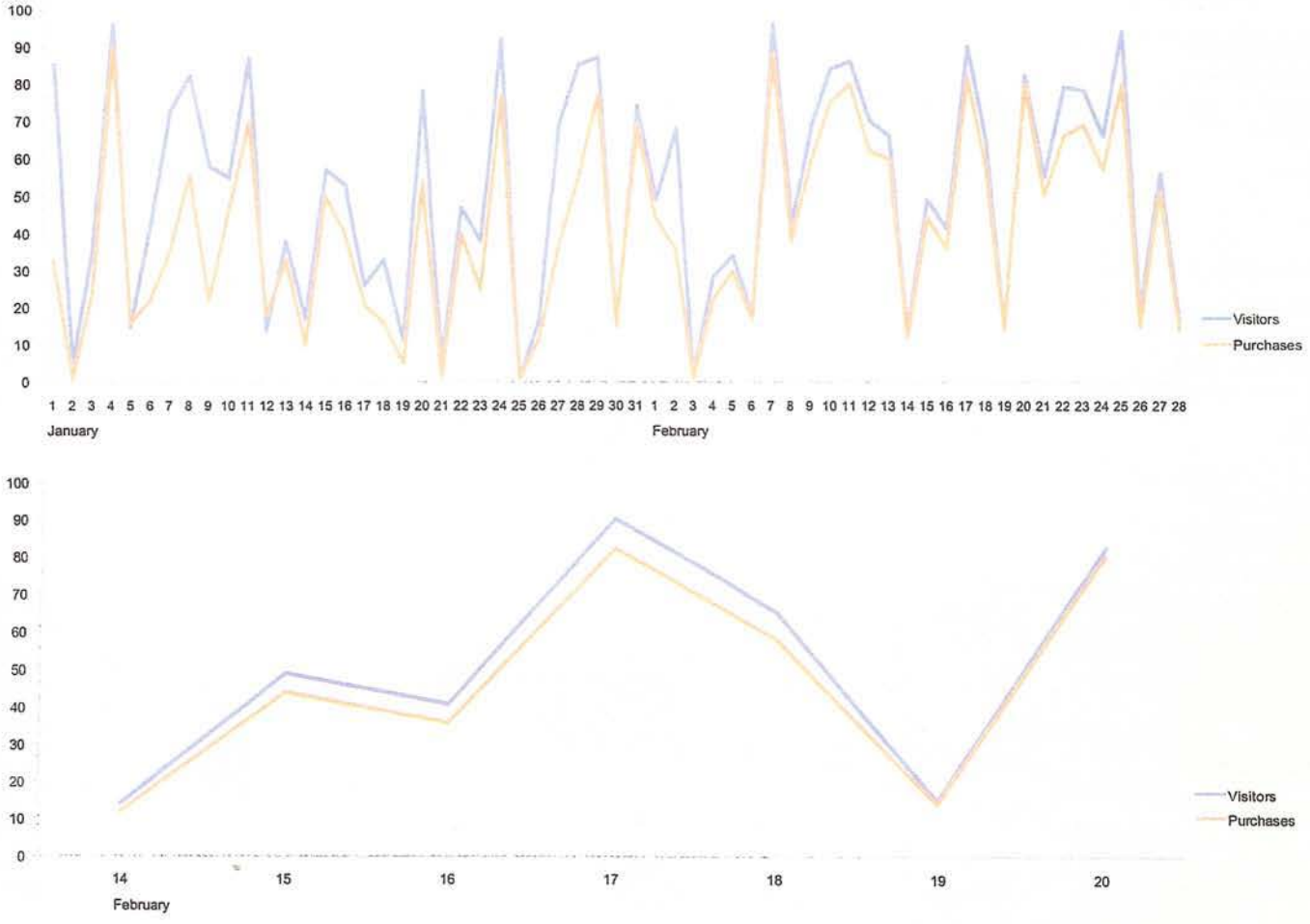


Figure 4.27

Zooming in on a graphical display can be accomplished in various ways: sometimes by using dedicated zoom controls and sometimes by simply narrowing a graph's quantitative scale (or both quantitative scales in a scatterplot) to include a subset of what's visible. Either way, the effect is much the same.

Usually, when we zoom in on a portion of the graph, it's as if we are using a magnifying glass to look at that specific section; that portion enlarges, and the other areas are no longer visible because they are outside the boundaries of the magnified view. We can always zoom back out to the graph's previous state to once again see the areas outside the magnified section, or we can change the view by panning, which involves moving up, down, right, or left in the larger display while maintaining the magnified (or "zoomed") scale. Panning brings into view areas of the graph that reside outside the boundaries of the current magnified portion. Panning is a familiar feature of many computer-based geographical displays, such as Google Maps. It is used in other graphical displays as well, such as panning from left to right in a line graph of time-series data to move the view to reveal different spans of time.

Information visualization software should support zooming and panning in the following ways:

- Provide the means to directly select an area of a graph and then zoom into it with a single click of the mouse.
- Provide the means to zoom back out just as easily.
- Provide the means, whenever a portion of what's in a graph is out of view, to pan in any direction directly with the mouse.

Re-scaling

This operation applies to quantitative graphs in particular. All graphs have at least one quantitative scale along an axis. Ordinarily, the quantitative scale places equal space between equal intervals of value. This common type of scale is sometimes called a *linear scale*. The following graph has a linear scale that ranges from $0 to $100,000 along equal intervals of $10,000 each. The distances between the tick marks are equal, reflecting that each jump in value is of equal size.

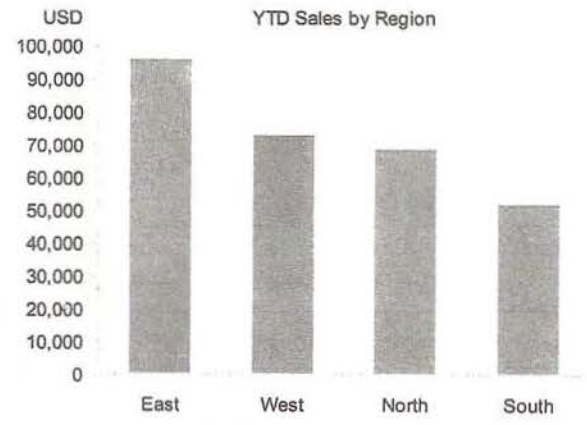


Figure 4.28

Another type of quantitative scale, which behaves differently, is a *logarithmic (log) scale*. Log scales occasionally come in handy. You might have avoided them until now because they were unfamiliar and perhaps a little intimidating, but they're really not that complicated. The little time it will take to get to know them will be worth the effort.

Along a log scale, each value is equal to the value of the previous interval multiplied by a base value. An example will clarify what I mean. Here's a log scale that has a base value of 10.

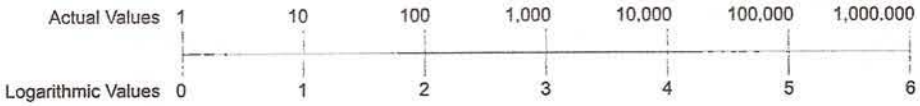


Figure 4.29

Notice that the second log value of 1 is equal to the actual value 10; that is, it is equal to the first actual value on the scale, 1, multiplied by the base value of 10 ($10 \times 1 = 10$). The third log value of 2 is computed by taking the second actual value, 10, and multiplying it by the base value, 10, which gives us 100. What's the fifth actual value along this scale? The answer is 10,000, which is the result of multiplying the actual value of 1 by the base value of 10 four times (that is, $1 \times 10 \times 10 \times 10 \times 10 = 10{,}000$).

So why bother with this seemingly unnatural scale? The primary reason, for our purposes at least, is because, by using a log scale to display time-series data, we can easily compare rates of change. We'll look at this use of log scales in *Chapter 7: Time-Series Analysis*, but here's a simple example to illustrate for now why they're handy. The graph below displays two sets of sales values: one for hardware sales and one for software sales. Which is increasing at a faster rate: hardware or software sales?

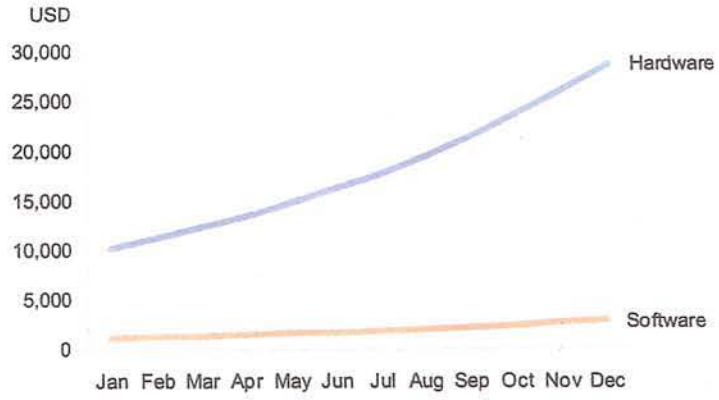


Figure 4.30

When I ask this question during classes, most people are quick to respond "hardware." The truth, however, is that both are increasing at the same 10% rate of change. The reason that hardware sales seem to increase at a faster rate is that a 10% increase in large dollar values produces a greater dollar increase than a 10% increase in low dollar values (software values are much smaller in this graph) even though the rate of change is the same. So the line for hardware has a steeper curve, but the rate of change, when normalized for the differences in

the magnitude of the sales prices, is the same. When we wish to compare rates of change and avoid this optical suggestion that rate of change for higher-priced items is greater than for lower-priced items, log scales are a convenient solution. Look at what happens when I do nothing to the graph above but change it from a linear to a log scale.

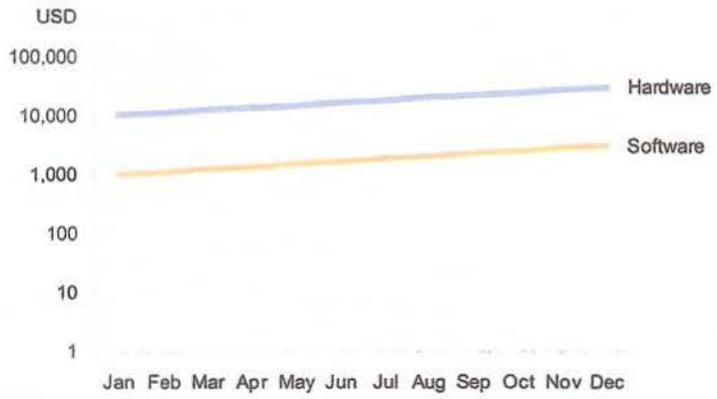


Figure 4.31

The slopes of the two lines are now identical. When using a log scale to display time-series values, identical slopes indicate identical rates of change. There are other scales besides the log scale that are sometimes useful for specific types of analytical problems, especially in science and engineering, but they are too specialized to include in this book.

Information visualization software should support re-scaling in the following ways:

- Provide a means to easily change the quantitative scale from linear to logarithmic and back.
- Provide the means to set a log scale's base. Although a base of 10 is typically used, others are sometimes useful. All bases display equal rates of change as equal slopes in lines, but the base value affects how many values appear on the scale. If, for example, a base 10 log scale results in too few values along the scale (for example, 1, 10, 100, and 1,000 only), switching to a lower base, such as base 2, will solve the problem.
- Provide a means to set the starting and ending values for the scale whether it's linear or logarithmic.
- Prevent or make inconvenient the use of a log scale with bar graphs and box plots. The height or length of a bar or a box encodes its value, but with a log scale we cannot rely on the bars or boxes as a means of comparing the actual values that they represent. Instead of regular values that are familiar, the heights or lengths of bars would encode log values, and most of us would find it difficult to meaningfully compare the magnitudes represented. For instance, in a log scale with a base of 10, a bar with a value of 100 would appear to have half the magnitude of a bar with a value of 10,000, even though it represented only 1/100th the value.

Accessing Details on Demand

Most of the time when we're exploring and examining data, we rely solely on visualizations, but from time to time we need details that either aren't included in a visualization or can't be discerned precisely enough. We want to call these details up instantly when we need them but keep them out of the way until then and put them out of the way once again after we've read them. A pop-up box (sometimes called a tool tip) containing details is the perfect solution. As shown in the example below, the box appears as a pop-up when we hover with the mouse over a particular item in a graph, such as a bar or point along a line. It disappears again when we move the mouse. In the field of information visualization, this is an example of what we call "details-on-demand."

Figure 4.32. Created using Tableau Software

Information visualization software should support accessing details on demand in the following ways:

- Provide an easy means to view details related to an item in a visualization when needed, in the form of text.
- Provide an easy means to make details disappear from view when they are no longer needed.

Annotating

When we think about things, it often helps to make notes. Notes help us clarify our thinking and build an external repository of our thoughts (both questions and insights), documenting them for ourselves and allowing us to pass them on to others. When our thinking is about visualizations that we are studying, it is most effective to annotate the visualizations themselves rather than keeping

notes in a separate, less accessible location. In the following example, I've annotated particular points in time to help me remember why traffic to my website exhibited particular behaviors.

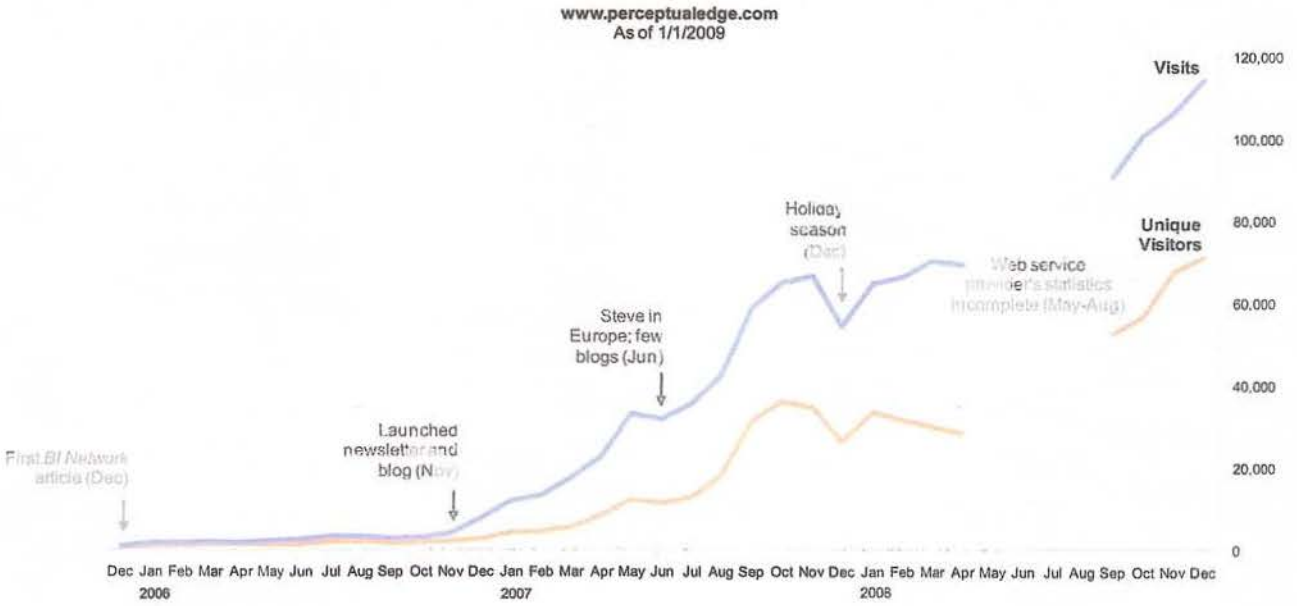


Figure 4.33

Unfortunately, because I can only annotate in Excel using text boxes that are independent of the graph, I'm forced to reposition the annotations every month when I update the graph with new statistics. Good visual analysis software supports richer annotation capabilities than this. In the two examples on the following page, I was able to attach the annotation to a specific data point in the scatterplot (left-hand example) so the annotation remained automatically tied to the point even when I filtered out some of the data, causing the annotated point to move (right-hand example). With annotation functionality such as this, we're encouraged to record our thoughts freely.

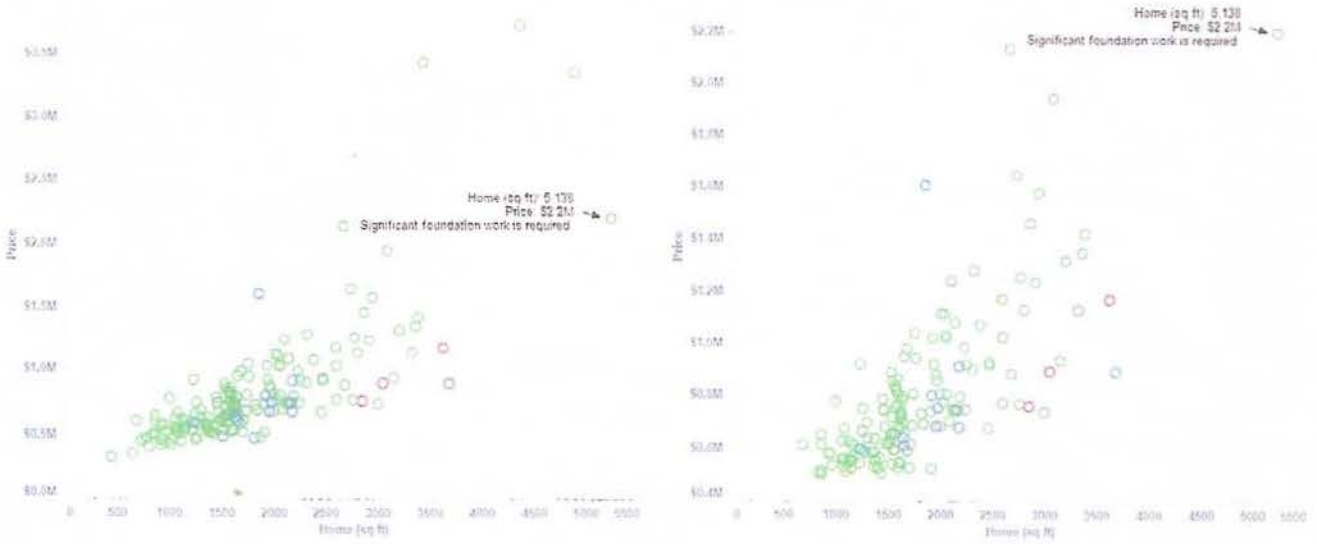


Figure 4.34. Created using Tableau Software

Information visualization software should support annotation in the following ways:

- Provide an easy means to add notes to a visualization so that they are associated with the visualization as a whole, a particular region, or one or more particular values.
- When notes are associated with particular items in a visualization and those items change position, the notes should automatically reposition to maintain the association.

Bookmarking

The analytical process does not follow a strict linear path; it meanders, bouncing back and forth. When we make an interesting discovery, it is often helpful to save that particular view, including its filters, sorts, and other features, so we can easily return to it later. This is similar to marking a page in a book or bookmarking a Web page for later access. Analytical software should allow us an easy, efficient way to save a particular view of the data so we can return to it whenever we wish. Sometimes this is accomplished by allowing us to save what we've created as a separate named worksheet or tabbed folder. How it's accomplished isn't important as long as it's easy to get back to it again with ease.

Sometimes we want to return to a previous view of the data that we didn't think to save at the time. This can only be accomplished if the software keeps a history of each view and allows us to navigate back through that history to get to particular prior states. It isn't as easy for software to maintain a history of the analytical process as it is to maintain a history of Web pages that we've visited during a single browsing session. What exactly constitutes a particular analytical state? We make so many changes during the analytical process, some big, such as changing from one type of graph to another, and some small such as turning on the grid lines or increasing font size. So it's hard to know where to draw the lines. Simply stepping backwards through a few individual changes, even small ones, is easy with a "back" or "undo" command, but what if the state that we wish to return to existed long ago? This requires some way to view the path we've followed in a manner that groups logically related changes together and perhaps branches off into separate paths when especially significant changes are made. This type of history tracking and navigation is more sophisticated than anything I've seen so far in a commercial visual analysis product, but some promising research has been done recently that is begging to be implemented.

The best work that I've seen so far for tracking and navigating analytical history was done by Jeffrey Heer, Jock D. Mackinlay, Chris Stolte, and Maneesh Agrawala and published in a paper titled "Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation," published in *IEEE Transactions on Visualization and Computer Graphics*, Volume 14, Number 6, November/December 2008.

Information visualization software should support bookmarking in the following ways:

- Provide an easy means to save the current state of an analysis (visualizations, filters, sorts, data, etc.) for later access without interrupting the flow of analysis.
- Maintain a history of the steps and states during the analytical process so that it's easy to return to a particular former step or state as needed.
- Provide a means to review the history of steps and views in the analytical process so that it's easy to find a particular previous state.

The analytical interactions that I've identified and described become fluid, integrated, and seamless: a natural extension of our thoughts. They cease to function effectively if too much work is required to move from one to another; this interrupts the free flow and evolution of thought that is required for effective analysis.

Analytical Navigation

The visual analysis process involves many steps and potential paths to get us from where we begin—in the dark—to where we wish to be—in the light (enlightened). Some methods of navigating through data are more effective than others. Tukey once wrote:

> *Data analysis, like experimentation, must be considered as an open-minded, highly interactive, iterative process, whose actual steps are selected segments of a stubbily branching, tree-like pattern of possible actions.*[2]

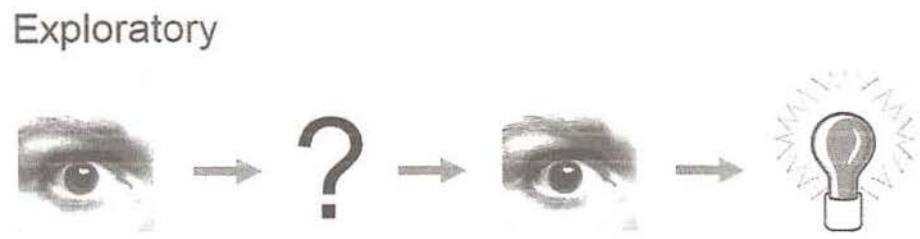
2. "Proceedings of the Symposium on Information Processing in Sight Sensory Systems," John W. Tukey and M. B. Wilk, California Institute of Technology, Pasadena CA, 1965, pp. 5 and 6.

There is no one correct way to navigate through information analytically, but some navigational strategies are helpful general guidelines within which we can learn to improvise as our expertise grows.

Directed vs. Exploratory Navigation

At a fundamental level, analytical navigation can be divided into two approaches: directed or exploratory. Directed analysis begins with a specific question that we hope to answer, searches for an answer to that question (perhaps a particular pattern), and then produces an answer. With exploratory analysis, however, we begin by simply looking at data without predetermining what we might find; then, when we notice something that seems interesting and ask a question about it, we proceed in a directed fashion to find an answer to that question.

Figure 4.35



Both approaches are vital. Data analysis sometimes requires us to begin with a blank slate and let the information itself direct us to features worth examining. I agree with Howard Wainer, who wrote, "A graphic display has many purposes, but it achieves its highest value when it forces us to see what we were not expecting."[3] William Cleveland expresses this opinion as well:

> *Contained within the data of any investigation is information that can yield conclusions to questions not even originally asked. That is, there can be surprises in the data...To regularly miss surprises by failing to probe thoroughly with visualization tools is terribly inefficient because the cost of intensive data analysis is typically very small compared with the cost of data collection.*[4]

Information visualization is ideal for exploratory data analysis. Our eyes are naturally drawn to trends, patterns, and exceptions that would be difficult or impossible to find using more traditional approaches, such as tables of text, including pivot tables. When exploring data, even the best statisticians often set their calculations aside for a while and let their eyes take the lead.

Shneiderman's Mantra

When new recruits are trained in spy craft by intelligence organizations such as the Central Intelligence Agency (CIA), they are taught a method of observation that begins by getting an overview of the scene around them while simultaneously using a well-honed awareness of things that appear abnormal or not quite right. When an abnormality is spotted, they rapidly shift from broad awareness to close observation and analysis. A similar approach is often the best approach for visual data analysis as well. This was simply and elegantly expressed by Ben

3. *Graphic Discovery: A Trout in the Milk and Other Visual Adventures*, Howard Wainer, Princeton University Press, Princeton NJ, 2005, p. 59.

4. *The Elements of Graphing Data*, William S. Cleveland, Hobart Press, Summit NJ, 1994, pp. 8 and 9.

Shneiderman of the University of Maryland, and is known by people in the information visualization research community as *Shneiderman's Mantra*:

> *Overview first, zoom and filter, then details-on-demand.*[5]

In the book *Readings in Information Visualization*, Shneiderman and his co-authors elaborated on this navigational approach:

> *Having an overview is very important. It reduces search, allows the detection of overall patterns, and aids the user in choosing the next move. A general heuristic of visualization design, therefore, is to start with an overview. But it is also necessary for the user to access details rapidly. One solution is overview + detail: to provide multiple views, an overview for orientation, and a detailed view for further work.*[6]

> *Users often try to make a "good" choice by deciding first what they do not want, i.e. they first try to reduce the data set to a smaller, more manageable size. After some iterations, it is easier to make the final selection(s) from the reduced data set. This iterative refinement or progressive querying of data sets is sometimes known as hierarchical decision-making.*[7]

5. *Readings in Information Visualization: Using Vision to Think*, Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, Academic Press, San Diego CA, 1999, p. 625.

6. *Ibid.*, p. 285.

7. *Ibid.*, p. 295.

Shneiderman's technique begins with an overview of the data, looking at the big picture. We let our eyes search for overall patterns and detectable points of interest. Let your eyes roam over the graph below, which displays daily unit sales of five clothing products during three months.

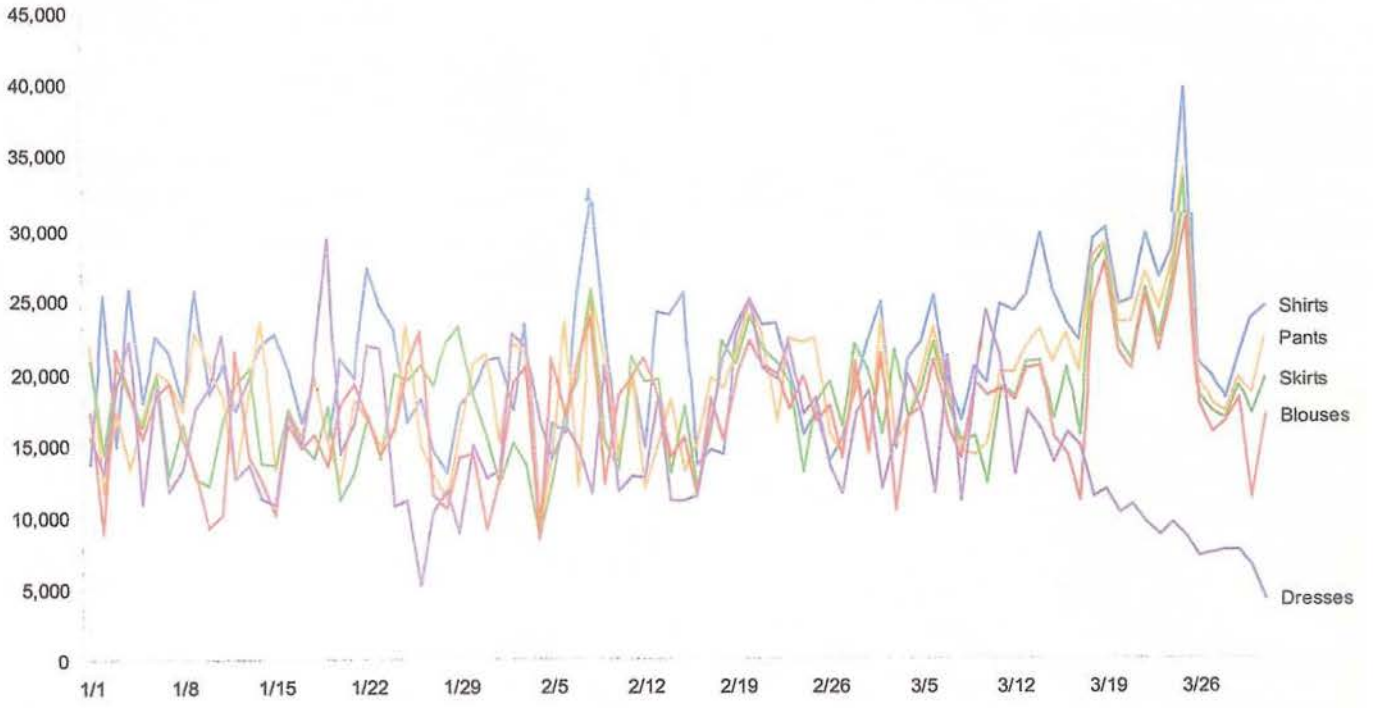


Figure 4.36

When we spot a particular point of interest, we can zoom in on it.

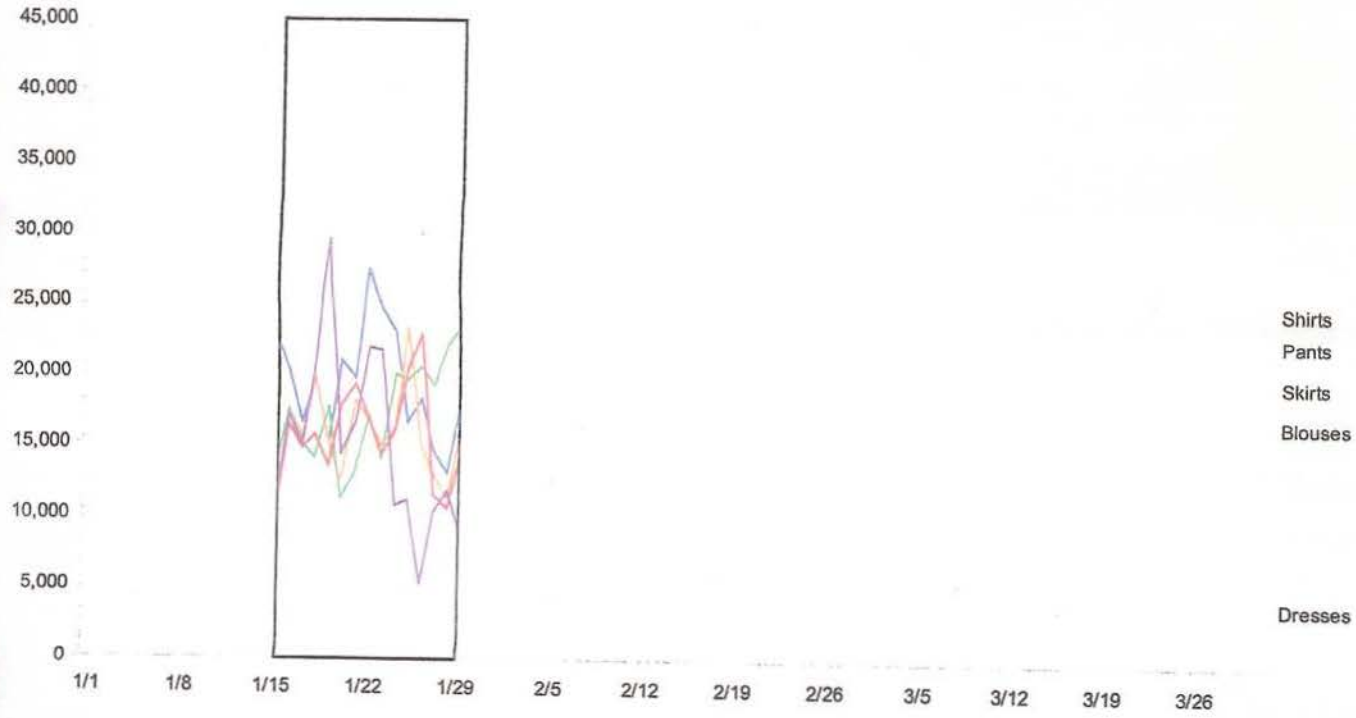


Figure 4.37

Once we've zoomed in on it, we're able to examine it more closely and in greater detail.

Figure 4.38

Often, to better focus on the relevant data, we must remove what's extraneous to our investigation.

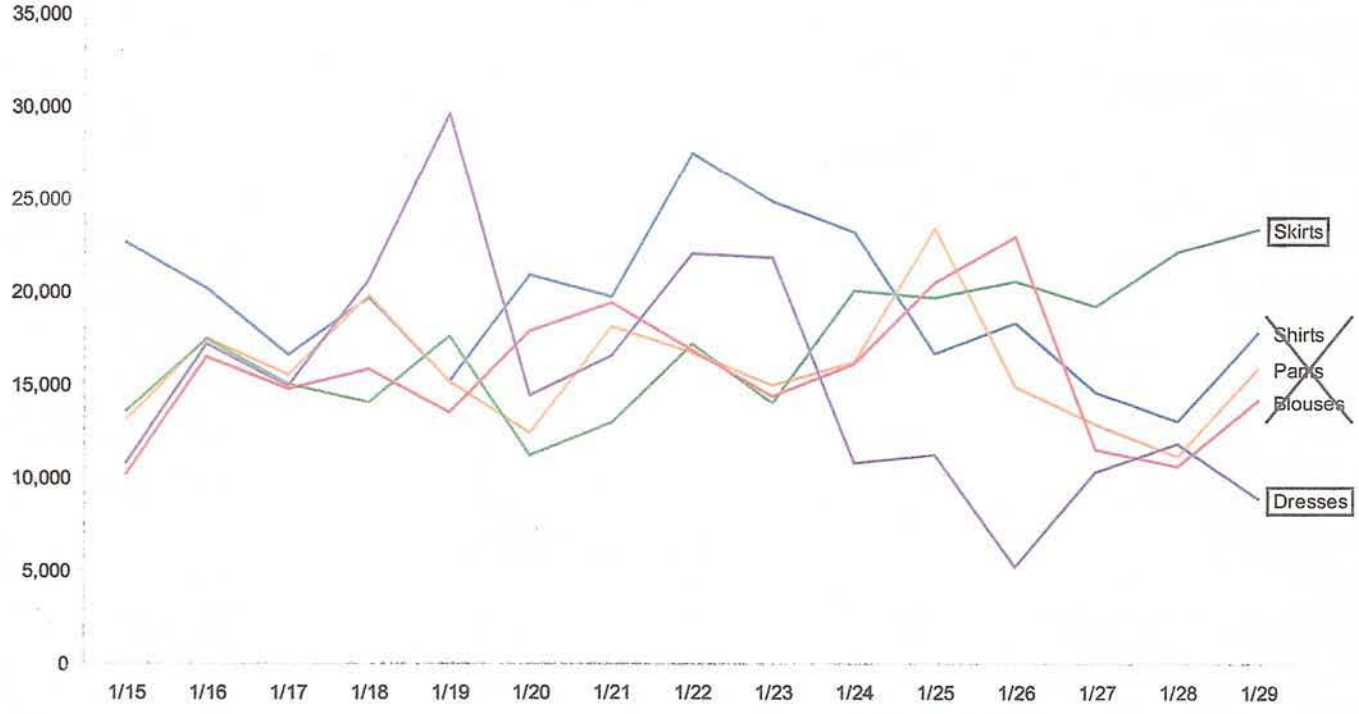


Figure 4.39

Filtering out extraneous data removes distractions from the data under investigation.

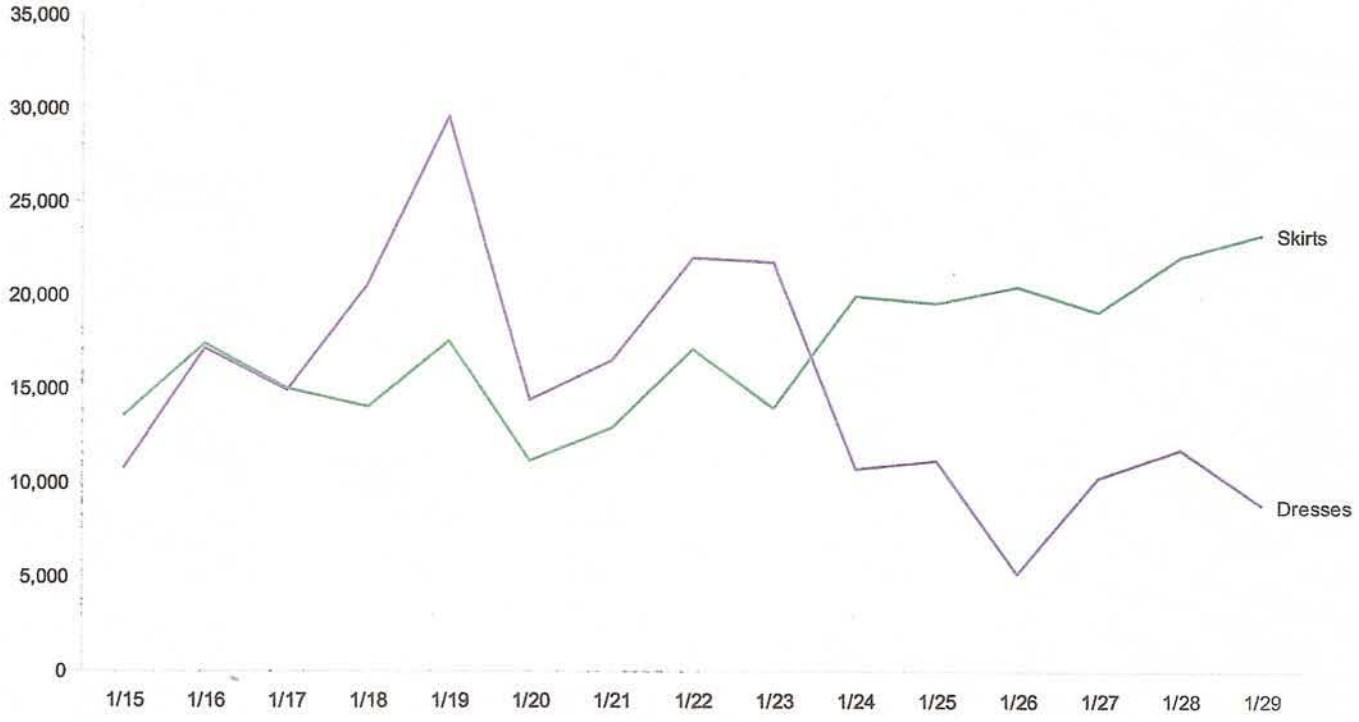


Figure 4.40

We rely mostly on the visual representations of information to reveal meanings that are embedded within data, but there are times when we need to see precise details that can't be discerned in the visualization. Here's another example of using a pop-up box in which information appears when we scroll over a specific area of an on-screen graph and disappears when we navigate away from the point.

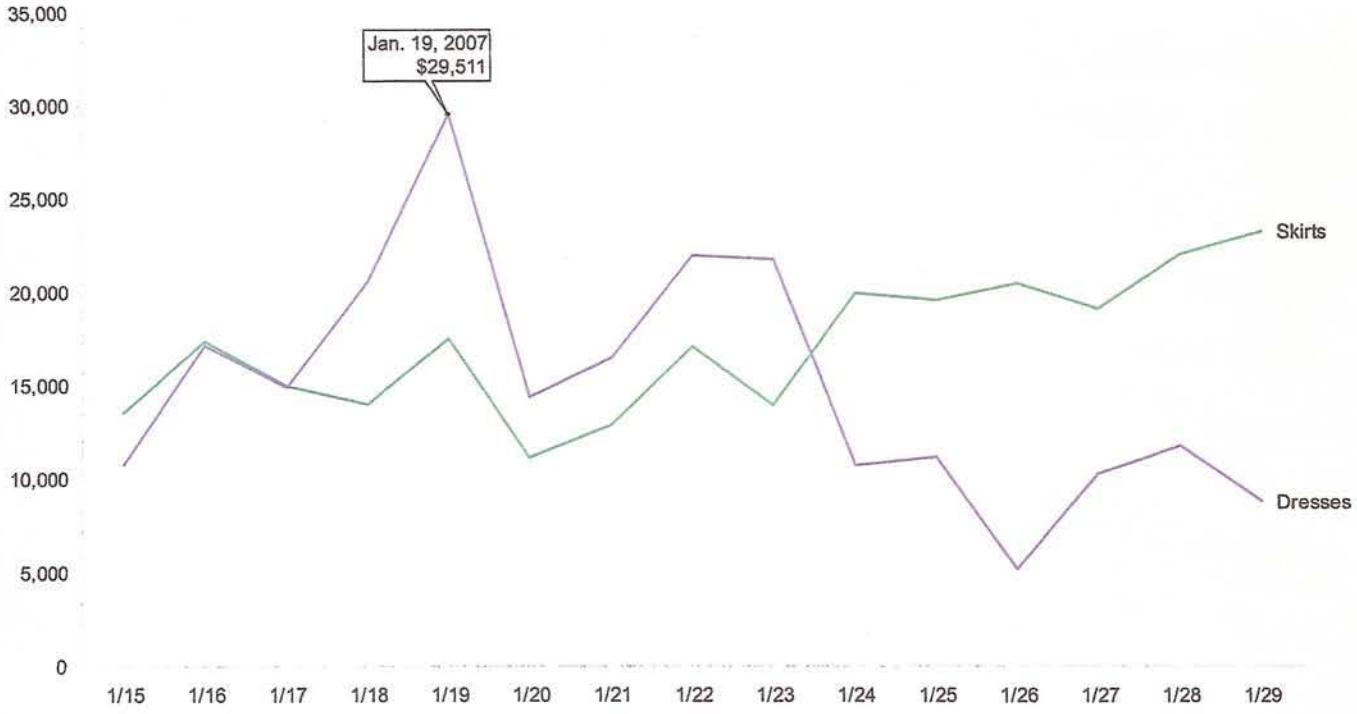


Figure 4.41

Although there is no one correct way to navigate analytically, Shneiderman's mantra describes the approach that often works best.

Hierarchical Navigation

It's frequently useful to navigate through information from a high-level view into progressively lower levels along a defined hierarchical structure and back up again. This is what I described earlier as drilling. A typical example involves sales analysis by region along a defined geographical hierarchy, such as continents at the highest level, then countries, followed by states or provinces, and perhaps down to cities at the lowest level. On the following page, the node-link diagram (also known as a tree diagram) illustrates a familiar way to display hierarchies.

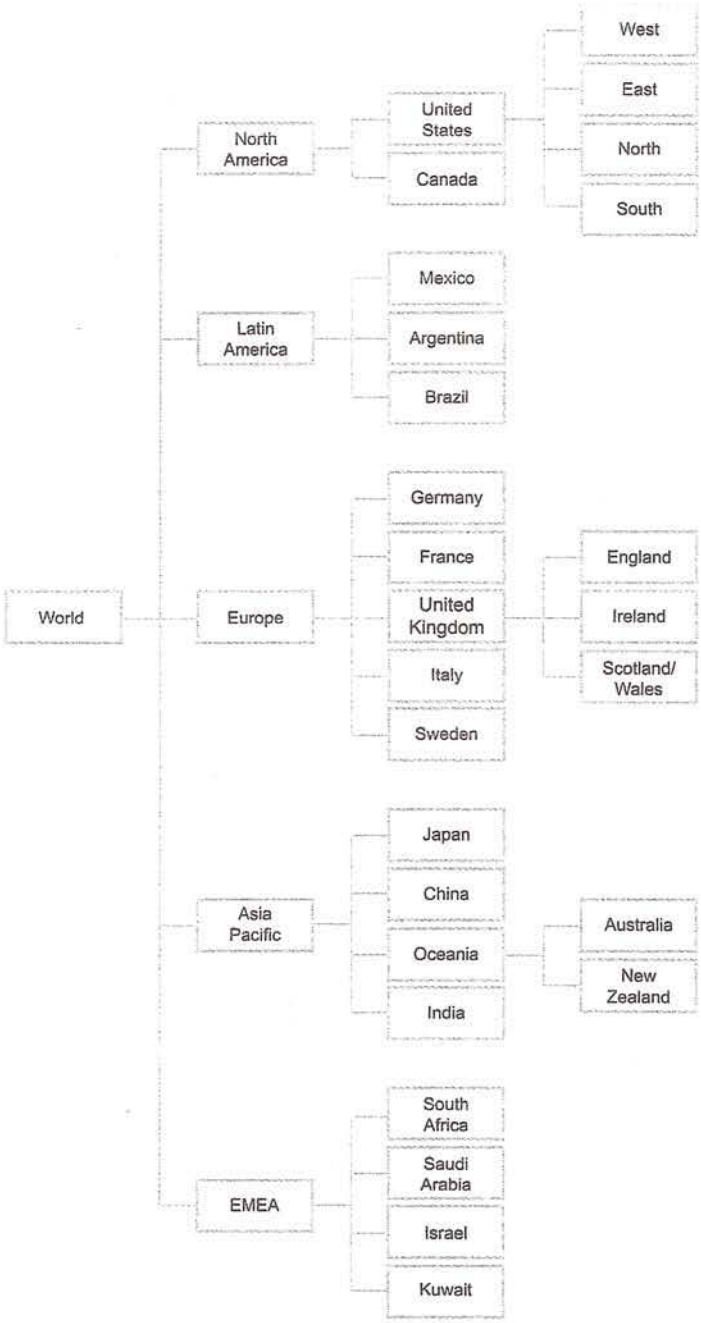


Figure 4.42

Node-link visualizations can be used to display quantitative values arranged hierarchically if variations in value are encoded using varying visual attributes such as the size or color intensity of each node. This approach quickly reaches the limits of what can be displayed on a single screen when we're dealing with large data sets. For this reason, Ben Shneiderman invented a visualization called a *treemap* (mentioned previously in *Chapter 3: Thinking with Our Eyes*), which he

designed to display quantitative data arranged hierarchically in a manner that takes full advantage of limited screen space to include as much information as possible. Rather than representing hierarchical relationships by connecting nodes with lines and arranging them from top to bottom or left to right, treemaps use containment—the placement of child objects inside of parent objects—to display these relationships. Treemaps arrange rectangles within larger rectangles, which fit neatly within one another, so many items can be displayed on a single screen, as shown below.

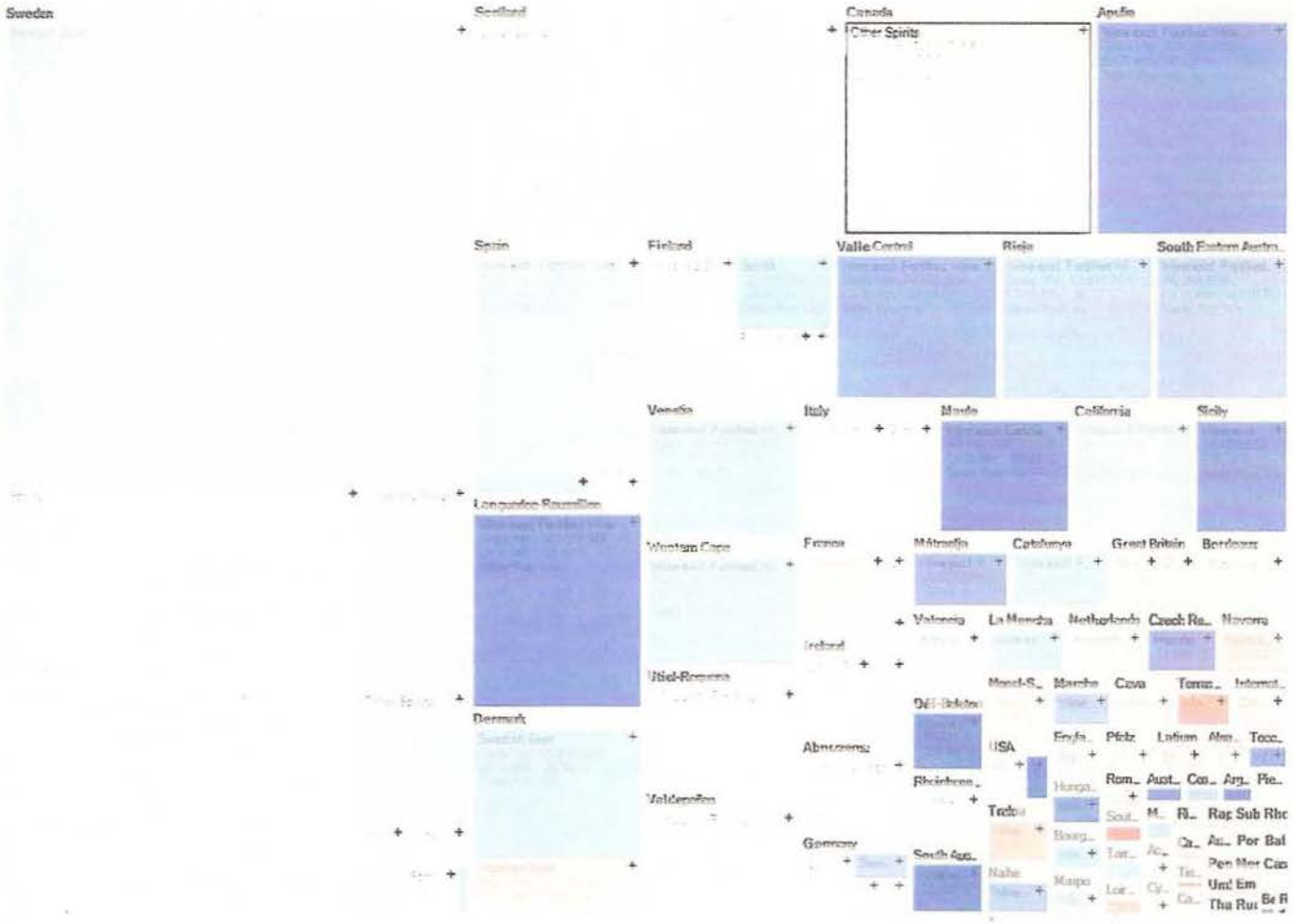


Figure 4.43. Created using Panopticon Explorer

This example displays sales of Swedish beverages (can you guess where Panopticon, the vendor whose software I used to create this example, is located?), per country or region divided into several categories (Swedish Beer, Spirits, and so on). A fourth level, individual producers, also exists in the hierarchy but isn't currently visible.

Treemaps can display two quantitative variables simultaneously: one represented by each rectangle's size and the other represented by its color. In this example, size represents year-to-date sales, and color represents the percentage change in sales from last month (blue for increases and red for decreases, with greater color intensities for greater degrees of change).

Treemaps can display a great deal of information quite powerfully but for a limited set of purposes. That is, treemaps were not designed to support precise quantitative comparisons, which we can't make based on relative sizes and colors. Instead, treemaps allow us to simultaneously view two quantitative variables for a large number of hierarchically arranged entities. I don't know any other visualization that can display as many hierarchically structured items at once, but I do know visualizations, such as bar graphs, that can display the same two variables more effectively for smaller sets of data.

Another reason to reserve treemaps for large data sets is that rectangles within rectangles don't display hierarchical relationships as clearly as other types of displays, such as node-link visualizations, do. When used for large data sets, however, treemaps allow us to readily spot extremes and predominant patterns. For instance, our eyes are drawn to the darkest red rectangle located near the bottom right corner, with a country name that begins "Sout..." When I hover over the rectangle with my mouse, details about this county immediately appear, as shown below.

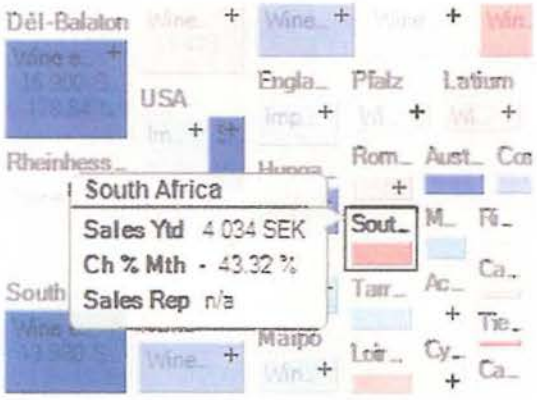


Figure 4.44. Created using Panopticon Explorer

We can also see a predominant pattern in this treemap: most of the countries with high sales (large rectangles) increased since the previous month (they're blue), with the notable exception of Spirits in Sweden, which decreased slightly.

Hierarchical navigation is easy with treemaps. If we want to take a closer look at sales of Spirits in Sweden to determine why they decreased, we can drill down into that category alone (by double-clicking it in this software program), causing it to fill the screen and automatically reveal the next level in the hierarchy (individual brands associated with each producer), shown on the following page.

Figure 4.45. Created using Panopticon Explorer

Now we can see that this decline cannot be attributed to any one product but appears to be fairly evenly distributed across all brands except for one: O. P. Anderson, the lone blue rectangle.

I've hardly done justice to treemaps in this short description. They merit much more exploration, but my current purpose is only to illustrate how they support hierarchical navigation especially when dealing with large quantities of data.

Now that we've examined ways to interact with information and navigate through the analytical process, it's time to move on to the techniques and best practices that will keep our work on track and lead us to rich insights.