Beginning Data Visualization with R

by Matthew Renze

Course Overview

Course Overview

Hi, I'm Matthew Renze with Pluralsight, and welcome to Beginning Data Visualization with R. R is a very popular programming language for data analysis and data visualization. Its interactive programming capabilities, powerful data analysis tools, and data visualization capabilities make it many expert's first choice for creating professional data visualizations. In this course we'll learn how to create and interpret data visualizations using the three main plotting systems in R, that is the base graphic system, lattice, and ggplot2. In addition, we'll learn about best practices for creating professional data visualizations for a wide audience. As an overview of this course, first we'll learn the basics of creating data visualizations with the three a main plotting systems in R. Next we'll learn how to create and interpret data visualizations that involve either one or two categorical or numeric variables. Finally, we'll move down the basics and learn about more advanced data visualizations, best practices for creating professional data visualizations, and we'll look at a few alternatives to using R to create our data visualizations. By the end of this course you have the skills necessary to create and interpret a variety of data visualizations using all three of the main plotting systems in R. Before beginning the course you should be familiar with the basic syntax of R and the RStudio integrated development environment. If you do not meet these prerequisites please feel free to watch module one of my Exploratory Data Analysis with R course

first. This module will cover all of the basics. So please join us today at Pluralsight and learn how to transform your data into actual insight with Beginning Data Visualization with R.

Introduction

Introduction

Hi, my name is Mathew Renze with Pluralsight and welcome to Data Visualization with R. In this course you'll learn how to create and interpret data visualizations using the programming language R. When you're finished with this course you'll be able to answer day to day questions about your data by creating data visualizations in R and interpret the results of these data visualizations, so let's get started. Imagine for me if you will that you're in a foreign country and you don't know how to speak, read, or write the native language there. This world would be full of strange sounds and symbols that would carry no meaning for you. All of the information you acquire about your environment would be through your senses and based on what others have translated for you. Imagine how difficult it would be to operate in this world without the ability to turn those strange sounds and symbols into knowledge that you can act upon. If you've ever been in this situation before you know how disorienting the experience can be. Unfortunately, many of us in the IT world are essentially stuck in the same situation, we live and breathe in a world filled with data, but we've never learned how to communicate in the language of data. So, we're unable to make sense of what these data are telling us, and we must rely on others to interpret these data for us. We're essentially living in a place that speaks a foreign language that we've never bothered to learn to speak, read, or write. Luckily for you, however, you've made the first step in learning one of these languages of data, a visual language for understanding data. In this course we'll essentially learn how to speak, read, and write in the language of data visualization. In more precise terms, we'll learn how to communicate, interpret, and create data visualizations. We'll do this with R, the most popular, free, open source language for data visualization. Learning to create and interpret data visualizations is extremely important as an IT professional because we live in a world filled with data. In addition, the amount of data is growing exponentially every year, the value of actionable data is increasing as well, and those with the knowledge and skills to work with data are rapidly becoming some of the most valuable resources in our information economy. So, if you want to continue to stay relevant and to be a highly valued asset in the new data-driven economy, it's critical that you learn how to work with and derive value from data. By the end of

this course you'll have the skills necessary to transform data into actionable insight using a variety of types of data visualizations.

Overview

Before we begin though, I think it's important that I provide you with a brief overview of what we'll be covering, who the intended audience is, and what the prerequisites for the course are. As an overview, first, we'll start with an introduction to data visualization, introduce the basics of creating data visualizations in R, and discus the various types of data visualizations that we'll be creating during this course. Next, we'll learn how to create and interpret data visualizations for qualitative, univariate analysis, that is the analysis of a single categorical variable. Then, we'll learn how to create and interpret data visualizations for quantitative univariate analysis, that is the analysis of a single numeric variable. Next, we'll learn how to create and interpret data visualizations for qualitative bivariate analysis, that is the analysis of their relationship between two categorical variables. Then, we'll learn how to create and interpret data visualizations for quantitative bivariate analysis, that is the analysis of the relationship between two numeric variables. Next, we'll learn how to create and interpret data visualizations for bivariate analysis for both a qualitative and a quantitative variable, that is the analysis a numeric variable grouped by a categorical variable. Finally, we'll move beyond the basics and learn about more advanced types of data analysis, more advanced types of data visualizations, best practices for creating professional data visualizations, and we'll learn about a few alternatives to using R to create data visualizations. The intended audience for this course are developers who want to learn how to visualize the data that they work with every day, data analysts who want to learn how to use R to create their data visualizations, and IT professionals with a bit of programming experience who want to learn how to make better decisions using data. Essentially, this course if for anyone in information technology with a desire to transform data into actual insights by creating and interpreting data visualizations using R. There are two prerequisites for this course. First, it is recommended that you have at least a basic understanding of the R programming language. As long as you know the basic language syntax you should do just fine. In addition, if you're a developer that has worked with at least one C-like programming language, for example, C++, C#, Java, or JavaScript, you'll most likely be able to follow along without too much difficulty. Second, it is recommended that you have basic familiarity with RStudio. We will be using RStudio for the duration of this course instead of the out of the box R IDE, that is integrated development environment. However, if you're a developer that has spent a lot of time working in IDEs you will likely be able to follow along without too much difficulty. If you don't meet the two prerequisite

conditions for this course I highly recommend that you watch the first module of my previous course, Exploratory Data Analysis with R. Module one of that course will provide you with all of the prerequisite information necessary for this course. It's only 25 minutes long and you can always skip the parts that aren't directly applicable to you. However, in my opinion, even though I might be a bit biased about this, I think it's a great introduction to both R and RStudio.

Data Visualization

Data visualization is that representation of data by a visual means. We do this because the human brain is exceptionally good at visual pattern recognition. So, we're able to quickly learn about a set of data by transforming it from its raw, textural, numeric form into a form that can be visually inspected. The essential idea is to map the dimensions of data, that is our variables, to visual characteristics. That is, we can map the quantities of our data to locations, sizes, shapes, and colors in our visualizations. For example, if we take a look at this table of data it would take us a bit of time to determine whether we sold more pizzas or more salads during the dates listed in the table. However, if we represent these data as a bar chart we can quickly see that we sold quite a bit more pizzas than salads. Just by representing our data visually we can often recognize patterns in our data more quickly than if we were to perform a numerical analysis. Our data visualizations should always start with a question. For example, did we sell more pizzas or more salads yesterday? Starting with a question helps us focus on a specific goal. It also helps us to determine the best way to answer the question using the data visualization that is most appropriate given the question and the data required to answer that question. R is currently the most popular open source tool for data analysis. Its interactive programming capabilities and powerful data visualizations tools make R the first choice for many developers, data analysts, and data scientists for creating data visualizations. We're going to be creating all of our data visualizations using three different plotting systems in R. First, we'll learn about the base graphic system in R. Next, we'll learn about lattice. And finally, we'll learn about ggplot2. Each of these plotting systems has various pros and cons, which we'll learn more about throughout this course. First, we'll create data visualizations using the base graphic system built into R. This is an extremely flexible system for plotting data visualizations that allows you to build your plots through a series of high-level plotting commands mixed with low-level graphics commands. For example, we can create a chart using the high-level plot command and make modifications to defaults by specifying additional parameters or calling low-level graphics functions. Second, we'll create data visualizations using the lattice extension package. Lattice is based on Trellis graphics, which was originally developed for the statistical programming languages S and S+, two

predecessors of R, is designed to make it easy to create multi-variate data visualizations, that is, data visualizations involving many variables. It works very well for basic graphing, but it's also flexible enough for most non-standard needs. Third, we'll create data visualizations using the ggplot2 extension package. Ggplot2 is based on The Grammar of Graphics, a book on statistical graphics written by Leland Wilkinson in 1999. It attempts to simply the creation data visualizations by using a high-level language to represent the various aspects of data visualizations. With ggplot2 we work with higher-level abstractions, like layers, geoms, aesthetics, scales, and more. In addition, we can build up our data visualizations by chaining various commands together. Now let's walk through a quick set of demos to become more familiar with these three plotting systems.

Demo Setup

For our first set of demos we're going to walk through the process of creating basic data visualizations using all three of the plotting systems that we'll be using for this course. First, we'll start with the base graphic system. Next, we'll create the same data visualization with lattice. Finally, we'll create this data visualization with ggplot2. We're going to keep things very simple to start. Based on the prerequisites of this course I'm assuming that you should have no problem understanding the code in R that I'm about to present or the layout and behavior of RStudio. If, however, by the end of this first demo you've had difficulties following along I highly suggest that you watch module one of my Exploratory Data Analysis with R course to familiarize yourself with the basic syntax of R in the use of RStudio. Our demos in this course will work like this, I'm going to be writing scripts in the Scripts pane in the upper left-hand corner of the screen. The code will be executed and the output will be displayed in the Console pane in the lower left-hand corner. The variables will be displayed in the Environment pane in the upper right-hand corner. And the plots will be displayed in the Plots pane in the lower right-hand corner. Simple enough? Excellent, let's visualize some data using the base graphic system in R.

Demo (Base)

First, we're going to create a very simple chart in the base graphic system of R. To do this, we'll need a table of data to plot. We'll begin by creating a simple data frame to store these data. The data frame will contain two columns, that is, a column called Name and column called Value. The data frame will also contain three rows with the values a, b, c, and 1, 2, 3, populated in columnar fashion. Next, we'll display this data frame so we can inspect its contents. As you can see, we have

the two columns titled Name and Value, with three rows containing a, b, c, and 1, 2, 3 populated in columnar fashion. Now, we'll plot the data frame in the simplest way possible, we'll just call the plot command and pass in the data frame as an argument to the plot command. This will plot the data frame using all of the default parameters. As we can see, this creates a very minimal bar chart. On the X-axis we have the name of the row of data from our data frame, that is a, b, and c. On the Y-axis we have the value of the row of data from our data frame, that is 1, 2, and 3. The height of the horizontal line corresponds to the value of each name row of data in our data frame, that is the higher the horizontal line the higher the value. For example, item a has a value of 1, item b has a value of 2, and item c has a value of 3. In addition to plotting the data frame and having R decide how best to plot the data contained within the data frame we can also be more specific about what we want to plot. We can do this by specifying the X and Y components of our plot. For example, we can specify that we want the X-axis to map to the data in the Name column in our data frame, and the Y-axis to map to the Value column in our data frame. This works because the first argument in the plot command is X, and the second argument in the plot command is Y. If the argument passed into the X parameter is a data frame, like in our previous example, the command figures out what columns to use for both X and Y on its own. However, if we specify the column to use for X and the column to use for Y, the plot command will use those columns for the X and Y axis respectively. Please note that the axis label on the X-axis, that is Name, and the axis label for the Y-axis, that is Value, have disappeared. When we're passing in the entire data frame the plot command is able to infer the titles of the axis labels from the names of the columns in the data frame. However, when we pass the data contained in the column to the X and Y-axis arguments explicitly the plot command doesn't have the column name property available in order to provide a label. Instead, we'll have to explicitly provide axis labels, which we'll do shortly. I should also note that we're not actually passing in a column into the X and Y arguments, rather, we're passing in a vector of data, that is a one-dimensional array of values all containing the same data type. A data frame can be looked at as a set of vectors of the same length representing each column of data. In our data frame we have two vectors, that is Name and Value containing three elements each, that is the vectors both have a length of three. The data types of the elements in the Name vector are all the same, and the data type of the elements in the Value vector are also all the same. In addition to passing in arguments by the default order of the parameters we can also explicitly use Name parameters instead. For example, we'll create the exact same plot as the one above, but instead we'll be specific about which parameter is X and which parameter is Y. Now, the order that we pass in arguments no longer matters. We could just as easily reverse the order of these name parameters and the code would execute exactly the same. For all the demos in this course if there's only a single parameter I'll typically just pass it in

as an argument without naming it. However, for two or more arguments I'll generally pass them all in by name. This should make it much easier for you to comprehend the code in this course. Finally, now that we've seen the basic syntax of plotting let's create an actual bar chart of our data frame. We'll use the bar plot command instead of the more generic plot command, set that names argument to our Name column, set the height argument to our Value column, set the color to skyblue, set the main chart title to Hello World, set the X-axis label to Name, and set the Y-axis label to Value. This creates a basic bar plot of the data contained in our data frame. On the X-axis we have the name of the row of data from our data frame, that is, a, b, and c. On the Y-axis we have the value of the row of data from our data frame, that is, 1, 2, and 3. The height of each bar corresponds to the value of each named row of data in our data frame. That is, the taller the bar, the larger the value. In addition, we have our main chart title at the top, our X-axis label on the bottom, and our Y-axis label in the left-hand side. If you need any help learning how to use the base graphic system in R there are plenty of ways to get assistance. The first step is to view the Help file for the plotting command that you're using. We can view the Help file for the generic plot command by typing? plot and pressing Enter. This will display the Help file in the lower right-hand pane of RStudio. We can also view Help files for the specialized plot commands like barplot in the same way. Or, if you're interested in learning how to modify graphical parameters for plot, you can view the Help file for parameters for using? par, for parameters. Now, let's see how to create the same data visualizations in lattice.

Demo (Lattice)

Now we're going to learn how to create basic data visualizations using lattice. To begin, we'll need to download and install the lattice package. We do this by using the install. packages command and pass in the name lattice as our argument. This'll download and install the lattice package on our machine. Next, once we have lattice installed we need to load the package into memory so we can use it. We do this using the library command passing in lattice as an argument. Let's create another data frame that is identical to the data frame we created in our previous demo. We'll do this using the exact same steps. Next, let's create a plot with lattice using the default parameters. To do this we'll use the dotplot command, set the x parameter to the formula Value, tilde, Name, which we read as Value as a function of Name, and we'll set the data parameter to our data frame. As we can see this creates a basic dotplot of the data in our data frame. On the X-axis we have the name of the row of data in our data frame. On the Y-axis we have the respective values for each row in our data frame. Each point represents a row of data in our data frame. The location of the point of the X-axis corresponds to the category a, b, or c, respectively.

And the location of each point on the Y-axis corresponds to the values 1, 2, or 3, respectively. Notice how rather than specifying the X and Y values separately we passed a formula in as an argument into the X parameter instead. Then we pass a data frame in as an argument into the data parameter. The variable names in our formula must match the column names in our data frame. Lattice uses formulas to specify what should be plotted. This is the most obvious initial difference between lattice and the base plotting system. This might seem more complicated now, however, doing so will make creating more complex data visualizations much easier later in the course. In addition, please note that the syntax of the formula is Y, tilde, X, which we read as Y as a function of X. This might seem backwards to start with the Y variable and then specify the X variable, but once again, this will make more sense as we get into more complex formulas. Now let's create a plot by specifying additional parameters. We'll create the same dot plot using the dotplot command, but we'll also set the main title parameter to Hello World, set the X-axis label parameter to Name, and set the Y-axis label parameter to Value. As we can see this added the main chart title and the axis labels to our plot. Finally, let's create a bar chart using lattice, we'll do this using the barchart command and we'll set the x parameter to the formula, Value as a function of Name, set the data parameter to our data frame, set the color to skyblue, set the main chart title to Hello World, set the X-axis label to Name, and set the Y-axis label to Value. As we can see this creates a bar chart of our data frame. This bar chart is essentially the same as our previous bar chart, however it has slightly different aesthetics. For example, the bars are a bit thinner, the fonts are a bit smaller, and there's a border around the chart. But, in essence, they both communicate the same information. If you're having trouble using any of the lattice plotting commands you can view the Help file for a specific command by typing?, and the name of the command, for example,? barchart, and pressing Enter. This will display the Help file in the lower right-hand pane in RStudio. In addition, if you need help constructing a formula for your data visualization,? formula is a good place to start. Now, let's learn how to create the same data visualizations in ggplot2.

Demo (ggplot2)

We're now going to learn how to create basic data visualizations using ggplot2. To begin, we need to download and install ggplot2. We do this using install. packages and pass ggplot2 in as an argument. Next, we need to load the ggplot2 package into memory. We do this with the library command passing ggplot2 in as an argument. Now, let's create our data frame using the same syntax we saw in the two previous demos. Next, let's create a basic plot with ggplot2 using the default parameters. We use the ggplot command and set the data parameter to our data frame,

set the aesthetics with x set to our Name column and y set to our Value column, and we'll set the geom to use a point geom. As we can see this create a basic dot plot like we saw with lattice. On the X-axis we have our names, that is a, b, and c. On the Y-axis we have our values, that is, 1, 2, and 3. Each point represents a row in our data frame. The location of each point on the X-axis corresponds to the name of the data element. The location of each point on the Y-axis corresponds to the value of the data element. Notice we needed to specify data, aesthetics, and geoms for ggplot, which is quite different from the previous two plotting systems we've seen. Aesthetics describe how the variables in the data are mapped to the visual properties. For example, Name is mapped to the X-axis and Value is mapped to the Y-axis. Geoms, which is short for geometrics objects, describe the type of plot we will create, for example, points, lines, bars, etc. Also, notice how the plus sign operator is being used to chain the geom point command to the base ggplot command. With ggplot2 we can chain together multiple visual elements to compose our data visualizations by individual layers. Aesthetics, geoms, and layers are all concepts from The Grammar of Graphics, the underlying visual language that ggplot2 is based on. Now this might seem more complex for creating simple data visualizations, however, creating data visualizations in this manner makes it very easy to create complex data visualizations with a clear and consistent language. Now let's take things a step further and explicitly set the parameters of our plot. With ggplot2 we have the both the concepts of parameters and layers. Some properties are specified using parameters while other are specified by adding another layer to the plot. We'll recreate our previous chart, but we'll add a main chart title and explicitly set our X and Y-axis labels. Instead of setting parameters like we did with the base plotting system and lattice, instead we'll add a chart title layer and set the title to Hello World, add an X-axis label layer and set the label to Name, and add a Y-axis label layer and set the label to Value. We can see that this now added a title to our chart and explicitly set the X and Y-axis labels. Conceptually, a plot in ggplot2 is composed of multiple layers that are all rendered one on top of the other, so by adding a new layer we're essentially drawing a new layer on top of our previous layers. When the chart is rendered we see the composite of all of these layers rendered one on top of the other. Finally, let's create a bar chart in ggplot2. To do this we'll start with the ggplot command. We set the data for the plot to our data frame, set the aesthetics with the X-axis set to the Name column, and the Y-axis set to the Value column, add a bar geom to the chart indicating that we want to create a bar chart, set the statistical transformation to identity, which means that we should set the height of the bar to the value of the data itself, set the fill color to skyblue, add a title layer with the title set to Hello World, add an X-axis label layer with the X-axis label set to Name, and add a Y-axis label layer with the Y-axis label set to Value. As we can see, this creates a bar chart of our data frame. This bar chart is essentially the same as the two previous bar charts we created, but once

again, it has slightly different aesthetics. For example, the bars a bit wider, the fonts are different, and there's a light gray background with thin white gridlines. But, in essence, they all communicate the same information. If you need assistance with any of the plotting commands in ggplot2 simply type? ggplot and press Enter. This will display the Help file for creating a new ggplot in the lower right pane of RStudio. In addition, to learn about the various parameters you can use for aesthetics, type? aes, for aesthetics, and press Enter. If you're interested in learning more about a specific geom, like the bar geom, simply type? and the name of the geom, for example,? geom_bar. You can also learn about any ggplot layer, like the main title layer, by typing?, and the name of the layer, for example,? ggtitle.

Demo Conclusion

This concludes our first set of demos. All of our demos will happen roughly the same way. First, we'll learn how to create plots using the base plotting system. Next, we'll learn how to create the same plots in lattice. Finally, we'll learn how to create them in ggplot2. This might seem a bit redundant, but there are some very good reasons as to why I've chosen to approach the course this way, in fact there are three main reasons for doing this. First, each of these three plotting systems has various pros and cons, it's easy to create some types of data visualizations in one plotting system, but either difficult or impossible to create the same data visualization in the others. So, it's likely that you'll use all three of these plotting systems at some point and time in your career to create plots. Knowing which plotting system works best to create each plot will make your job much easier. Second, repetition will help make learning how to create and interpret these data visualizations much easier. If you see how to do the same thing three different times in three different ways you'll be much more likely to retain this information. Third, you'll be seeing data visualizations from three different perspectives. Each of these three plotting systems has a unique way of looking at data visualization. The base plotting system is largely command oriented, that is, you describe your data visualization using high-level commands, then you add additional visual elements with low-level commands as needed. Lattice is formula-oriented, that is, you describe your data visualization using a formula for the data to be visualized. And ggplot2 is based on the Grammar of Graphics, that is, you describe your data visualization in terms of aesthetics, geoms, layers, etc. Having three different and unique perspectives on data visualization will help us to understand these data visualizations at a much deeper level. Now let's take a look at the various types of data visualizations that we can create.

Types of Data Visualizations

We don't want to go too deep into statistics for this course, but in order to data visualization we need to understand a few basic terms from statistics. First, we have observations, which are essentially rows in the table. They are referred to as observations because in statistics we're typically concerned with observations of some kind of physical phenomenon. For example, if we have a temperature sensor each recorded temperature over time would correspond to an observation of the temperature. Equivalently, we can also have transactions, like pizza sales transactions, or entities, like feature-length films, as phenomenon we are observing. However, for our purposes we'll refer to them all generically as observations. Next, we have variables, which are the columns in the table. They're called variables because their values vary across each observation. For example, in the table on the right, Date, Customer, Product, and Quantity are all variables that can change value across each row in the table. There're two types of variables, first we have qualitative variables. Qualitative variables contain categorical values, for examples, customers and products. In addition, they have no natural sense of order. We can, however, impose an arbitrary order upon them, like using the alphabetical sort order of their names. However, this is still just an artificial, not a natural means of sorting them. Qualitative variables are often referred to as nominal variables because they are named values. Finally, we have quantitative variables. Quantitative variables contain numeric values, for example, the quantity of products sold. In addition, they do possess a natural sense of order, for example, two pizzas sold is more than one pizza sold. Quantitative variables can also be subdivided into either discreet values, that is whole numbers represented as integers, or continuous values, that is all possible points on the real number line, typically represented as decimal precision numeric values. In addition, quantitative variables can also be subdivided into ordinal, interval, and ratio subtypes. However, these subdivision, their differences, and statistical limitations are outside of the scope of this course. When we're performing visual data analysis the types of data visualizations that we can create are determined by the type of variables and the number of variables. First, we have univariate analysis, which is just a fancy term for the analysis of a single variable. If we have just a single qualitative, that is categorical, variable we can perform qualitative univariate analysis, which means the analysis of a single categorical variable. If we have just a single quantitative, that is numeric, variables we can perform quantitative univariate analysis, which is the analysis of a single numeric variable. Next, we have bivariate data analysis, which is just a fancy term for the analysis of the relationship between two variables. If we have two qualitative, that is categorical, variables we can perform qualitative bivariate analysis, which is the analysis of the relationship between two categorical variables. If we have two quantitative, that is numeric, variables we can perform quantitative bivariate analysis, which is the analysis of the relationship between two numeric variables. If we have both a qualitative and a quantitative variable we can perform bivariate

analysis for both a qualitative and a quantitative variable, that is the analysis of a numeric variable grouped by a categorical variable. There are more types of data analysis beyond these five types, however, to keep things simple for our course we're going to focus on just these first five types of data analysis. We will, however, discuss the other types of data analysis beyond these five types in our last module. Now I realize that this information might be a bit difficult to digest all at once, especially given the statistical terminology that you might not be familiar with. However, this chart is very important as it will act as a roadmap for the next modules in this course. The next five modules in this course correspond to each of the five types of data analysis. In addition, this chart is how you will mentally organize all of the data visualizations that we're about to learn throughout the course. Essentially, each of these types of data analysis have corresponding data visualizations, which are appropriate for that specific type of analysis. The best way that I've found to mentally organize all of these various types of data visualizations is by having a thorough understanding of each of these types of data analysis, and then learning which types of data visualizations belong with each type of data analysis. Don't worry if you don't have this chart memorized however, we'll be coming back to it at the start of each of our remaining modules. Finally, before we wrap things up for this first module I'd like to quickly show you all of the different types of data visualizations that we're going to be learning in this course. Partially, this is to provide you with a glimpse of the types of data visualizations that we'll learning throughout this course, and partially to get you excited about how much we're going to be learning over the next few modules. First, in module two, that is Visualizing One Categorical Variable, we're going to learn how to create and interpret frequency tables, frequency bar charts, horizontal bar charts, cleveland dot plots, and pie charts. In module three, that is Visualizing One Numeric Variable, we're going to learn how to create and interpret dot plots, jitter plots, box plots, histograms, and density plots. In module four, that is Visualizing Two Categorical Variables, we're going to learn how to create and interpret contingency tables, grouped frequency bar charts, stacked frequency bar charts, 100% stacked frequency bar charts, spine plots, and mosaic plots. In module five, that is Visualizing Two Numeric Variables, we learn how to create and interpret scatter plots, linear regression plots, Bin Frequency Heat Maps, hexagonal bin frequency heat maps, contour plots, level plots, mesh plots, surface plots, step charts, line charts, and area charts. Finally, in Module six, that is Visualizing Both a Categorical and a Numeric Variable, we'll learn how to create and interpret bivariate bar charts, bivariate box plots, notched box plots, and bivariate violin plots. As you can see, we have quite a bit ahead of us to learn, and I hope you're as excited as I am learn about all of these various types of data visualizations.

Summary

In this module, first, we learned about the learning objectives for the course, what topics we will cover, and why we want to learn how to create and interpret data visualizations with R. Next, we learned about data visualization and how it can be used to transform data to actionable insight. Then, we learned how to create basic data visualizations using the base R graphic system, lattice, and ggplot2. Finally, we learned about the various types of data visualizations that we can create based upon the type and number of variables involved. In the next module we'll learn how to create an interpret data visualizations for a single categorical variable.

Visualizing One Categorical Variable

Introduction

Welcome back to data visualizations with R. I'm Matthew Renze with Pluralsight, and in this module we'll learn how to create and interpret data visualizations for a single categorical variable using R. As an overview of this module, first we'll learn about data visualizations for qualitative univariate analysis, that is the analysis of a single categorical variable. Next, we'll learn how to create these data visualizations using the base graphic system. Then we'll learn how to create the same data visualizations using lattice. Finally, we'll create these data visualizations using ggplot2. To help orient ourselves in relation to the rest of the course, this module will cover qualitative univariate analysis, that is the analysis of a single categorical variable. In addition, we'll be building the foundation for the remaining modules in this course. For the duration of this course we're going to be working with the data set from the open movies database containing a selection of over 3, 000 movies released in the United States between the year 2000 and the summer of 2015. Our data set only includes movies that have a Motion Picture Association of America rating of G, PG, PG-13, or R that also have a reported box office revenue. The data set contains various data about the movies, such as the title of the movie, the year the movie was released, the MPAA rating of the movie, that is G, PG, PG-13, and R, the movie's runtime in minutes, the genre of the movie, for example, action, comedy, drama, etc., the critic score, that is, the percentage of critics who gave the movie a favorable review, the box office revenue in millions of US dollars, a flag indicating whether the movie has won one or more awards, and a flag indicating whether the movie was distributed internationally, or if it was only released in the United States. We'll be using

this data set in one form or another for all the slides and demos in this course. In order to keep things interesting for the slides and demos in this course we're going to create a fictional narrative to help motivate the remainder of our course. Imagine for me if you will that we work as a software developer for an online store that specializes in selling digital movies on demand, we'll call this company NetFilms. Our day to day job at NetFilms is to create software that streams digital movies to our end users. However, over the years we've also developed a reputation for being the go-to person to answer business questions using data. So, our co-workers in sales and marketing often come to us for help when they're unable to get answers to their sales and marketing questions. In fact, we're actually getting pretty good at being able to answer their questions using data.

Qualitative Univariate Analysis

The first type of analysis that we can perform is univariate analysis of a qualitative variable. This is just a fancy term for the analysis of a single categorical variable. When we're performing qualitative univariate analysis we're typically interested in answering questions about the frequency of occurrences of observations that occurred in a specific category. For example, if we have a data set containing a collection of movies and are interested in the number of movies that are comedies we can find this value by scanning down the table to the comedy row and inspecting the frequency. In addition to frequency, we might also want to answer questions about how the proportion of observations in a specific category relates to all observations, that is the part of the whole. This value it typically expressed as a percentage. There are other measures that we might be interested in while performing qualitative univariate analysis, but these should cover the basics for the purposes of our data visualizations. The first type of data visualization that we can create for qualitative univariate analysis is a frequency bar chart. On the X-axis we have the categories of our variable, in this case, we have movie rating categories, like G, PG, PG-13, and R, and on the Y-axis we have the frequency of observations, for example, the number of movies. The height of each bar corresponds to the frequency of observations in each category, for example, the number of movies in each rating category. This type of data visualization can be used to answer questions about the frequency of observations contained within a specific category of a data set. For example, how many PG movies are contained in the data set? As we can see from our bar graph there are approximately 500 PG movies in our data set. In addition, this data visualization works very well for answering questions that involve a comparison between the number of observations in two or more categories. For example, are there more PG-13 movies, or are there more R-rated movies in our data set? As we can see from our data visualization there

are more R-rated movies than PG-13 movies. Frequency bar charts also work well for answering questions about the minimum or maximum values in the data set. For example, which category contains the least number of movies and which category contains the most movies? As we can quickly see from our bar chart the category containing the least number of movies in the G-rated category. And, the category containing the most number of movies in the R-rated category. In addition to displaying bar charts vertically, we can also display bar charts horizontally. On the X-axis we have the frequency of observations, for example, the number of movies in this case. On the Y-axis we have the categories, for example, the movie genres. The length of each bar now corresponds to the number of movies contained in each category. This data visualization is useful when you have a large number of categories, for example, we have more than 10 categories, or when you have long category names. This data visualization works better in these cases because the horizontal orientation provides more room for the category labels, which makes them easier to read. The alternative would be to rotate the labels on the X-axis of the vertical bar graph, which makes the labels more difficult to read. We can answer similar questions involving frequency, comparison, and minimum or maximum values using horizontal bar charts. For example, if we want to know whether there are more action movies or more adventure movies we can simply compare the length of the corresponding bars and determine that there are more action movies. An alternative to horizontal bar charts is type of a data visualization called a Cleveland dot plot, named after its creator, William S. Cleveland, an expert in data visualization. On the X-axis we have the frequency of observations, for example, the number of movies. On the Y-axis we have the categories, in this case, the movies genres. The location of the point along the X-axis corresponds to the number of movies in a specific category. The location of the point on the Y-axis corresponds to the respective category. Cleveland dot plots answer the same types of questions that both vertical and horizontal bar charts answer. However, they answer these questions in a way that maximizes the data-to-ink ratio for the data visualization. Data-to-ink ratio is the amount of information that is conveyed by our data visualization relative to the amount of ink needed to create the visualization. Essentially, you want to communicate as much information as possible with the minimum amount of ink, or digital ink in our case. The primary reason for this is that clean data visualizations are much easier to read than cluttered, or noisy data visualizations. Thus, as a general principle, we want to always strive to maximize the data-to-ink ratio in our data visualizations and create clean data visualizations where possible. In addition to comparing the frequency of observations across categories we might also want to answer questions about how the number of observations in a category relates to all observations, that is, the proportion, or part of the whole. We can answer questions about proportion using a pie chart. In this pie chart we have the proportion of movies released in each rating category. The interior

angle of each wedge represents the proportion of movies released in a specific category. With pie charts we can answer similar questions involving comparisons of the number of observations in categories. However, it is now within the context of one or more categories in relation to all observations. I should note, however, that pie charts are generally discouraged by data visualization experts because the human brain isn't as good at comparing values using the angles of a pie chart relative to using the length of bars in a bar chart. This is especially the case when there's a large number of categories in a pie chart. However, if you keep the number of categories limited to a very small number, for example, two categories, and your visualization isn't intended for precise comparison, there're places where pie charts may be acceptable. For example, in this pie chart, we have the number of movies by award status, that is, whether the movie has won one or more awards, or not. Since there are only two categories we are never comparing more than two angles at a time. In general, using a two category pie chart is preferred to using a pie chart with more than two categories. Ultimately, please use your best judgment, and if all else fails, just use a bar chart. There are other data visualizations we can create for qualitative univariate analysis, but these should cover the basics. Now, let's see how to create these types of data visualizations using the base graphic system in R, lattice, and ggplot2.

Demo (Base)

Our co-worker Natalia works in the marketing department at NetFilms. One day she comes to us with a series of questions she has about the movies contained in the NetFilm streaming video library. First, she wants to know how many movies are contained within each movie rating category. She wants to compare them to make sure that we have the right balance of movies for each age group. Second, she wants to know what proportion of movies have won at least one or more awards. This will help her to determine whether we have enough award-winning films, which are more expensive, versus the less expensive movies that have not won any awards. We offer to help her answer these questions using the data visualizations techniques that we just learned. So, let's get started with our data analysis by using the base graphic system in R. First, we need to set our working directory to the folder containing the data for our analysis. We do this with a setwd command, which stands for set working directory. We pass the UNIX-style directory path of the folder containing our data file into the command as an argument. Next, we need to load the CSV, that is, comma separated values data, from our CSV file. We use the read. csv command and we pass the file name in as an argument into the command. Then, we assign the data return from the command into a variable that we'll call movies. In addition, just so that we can see the structure of the data that we just loaded, let's take a quick peek at the data using the head

command. The head command displays just the first six rows of data from our data frame. Now let's begin creating univariate data visualizations for a qualitative variable, that is, a single categorical variable. Natalia's first question involves wanting to know how the number of movies in each rating category compares to one another. This means that a frequency bar chart is likely a good choice of data visualization to answer this question. So, let's create a frequency bar chart for these data. To do so, we use the plot command and we set x to the vector of values contained in the Ratings column of our movies data frame, add a main chart title, add an X-axis label, and add a Y-axis label. The plot command is smart enough to figure out that we want to visualize these data as a bar chart. Now Natalia can compare the number of observations in each rating category. If we were dealing with a large number of categories, or if our categories had long names, we might want to display our results as a horizontal bar chart. To do so, we simply execute the plot command with the same data as before. However, this time, we'll set the horizontal parameter to true indicating that we want to display the data horizontally. We also need to swap our X and our Y-axis labels since we've rotated the chart. Now, we have a horizontal bar chart representing the same data as our previous bar chart. In addition, if we want to create the same data visualization, but with a higher data-to-ink ratio we can create a Cleveland dot plot of these data. To do so, we use the dot chart command and we pass in a frequency table of ratings using the table command to create the frequency table. We can also set the pch parameter to 16 indicating that we want the plot character to be a solid black point rather than the default empty small circle to make the plot a bit easier to see. Each of the plot characters are represented as an integer and 16 is the index of the solid circle plot character. In addition, we'll set the main chart title, X-axis label, and Y-axis label. This produces a more minimalistic data visualization than the two previous bar charts. However, we'll leave it up to Natalia to decide which data visualization she prefers. If we were interested in comparing these categories in the context of their proportion to all movies contained in the data set we could use a pie chart. To create a pie chart we use the pie command and we set the x parameter to the same frequency table that we created for the previous chart and we'll set the main chart title. However, as we discussed earlier, pie charts are probably not the best way to present these data for comparison purposes. Natalia is, however, interested in seeing a pie chart of the proportion of movies that have won awards to all movies in the data set. This would be a better case for using a pie chart since there are only two categories involved in this data visualization. To do so we'll use the pie command, we'll pass in a frequency table of movie awards, and this time, we'll set the clockwise parameter to TRUE so that the pie chart populates in a clockwise fashion. Finally, we'll set the main chart title. As we can see the clockwise parameter made the value of FALSE, that is no awards, first, and the value of TRUE, that is movies that have won awards, second in a clockwise

manner. This should allow Natalia to quickly see what proportion of movies in our streaming video library have won awards versus those that have not. Now, let's see how to create the same data visualizations in lattice.

Demo (Lattice)

We'll now visualize the same qualitative data using lattice. We already have the lattice package downloaded and installed from our first set of demos in module one. However, we still need to load the lattice package using the library command, like we saw previously. In addition, we already have our working directory set from the previous demo, and we've already loaded the CSV, that is comma separated values data, from our CSV file into a variable called movies. So, we don't need to do either of those two steps again for the remaining demos in this module. Let's create a frequency bar chart using lattice. Before we can create a bar chart though we need to convert the data into a frequency table containing the number of observations for each rating category. First, we create the frequency table using the table command and we'll assign it a variable called table. Next, to make things easier, we're going to convert that table into a data frame and assign it to a variable called ratings. Then, we'll set the name of the first column to rating and set the name of the second column to count. Just so you can see what we've created we'll display the frequency table using the print command. As you can see we have a data frame containing two columns, that is, rating and count, with the rows populated for each rating and its respective number of observations. Now that we have our data in the correct format for visualization, let's create our bar chart. To do this we'll use the barchart command, set the formula to Count as a function of Rating, since we want Count on the Y-axis and Rating on the X-axis, set the data for the chart for the data frame that contains the frequency table we just created, set the main chart title, and set the X-axis label. Note that the Y-axis is automatically set by lattice for bar charts so we don't need to manually specify it. Now we can compare the number of observations in each rating category. We can create a horizontal bar chart simply by changing the order of the variables in our formula. Once again, we'll use the barchart command, however, this time we'll set the formula to Rating as a function of Count since we want the Rating variable on the Y-axis and the Count variable on the X-axis. Then, we'll set the data to the rating frequency table, just like we did before, set the main chart title, and this time we'll need to specify the Y-axis label since lattice doesn't render it by default for horizontal bar charts. Note that the change in formula changes the orientation of our bar chart to horizontal. To create a Cleveland dot plot, we use the same formula as we did for the horizontal bar chart. However, this time we use the dot plot command instead. All of the other parameters will be the same as the horizontal bar chart. By changing the

command form bar chart to dot plot lattice creates a Cleveland dot plot instead. Unfortunately, pie charts do not exist in lattice, however, as an alternative, lattice provides a bar chart that displays the percent of total on the Y-axis. To create this data visualization we use the histogram command, set the formula to tilde Rating indicating that we want to display a single variable, that is Rating, set the data to the movies data frame, set the main chart title, and this time we don't need to specify either the X or Y-axis labels because lattice adds them both for us with the histogram command. Notice that lattice uses a command called histogram to produce this part-of-whole bar chart. However, the data visualization that is being produced is not technically a histogram since it involves displaying the frequency of observations of a categorical variable. The term histogram, which we'll learn more about later in this module is typically reserved for observing the frequency of observations of a continuous numeric variable grouped into bins. In addition, please note that the Y-axis displays the percent of movies in each rating category instead of the count of movies, like we had in our previous frequency bar charts. This difference mirrors the idea that we're interested in the part-of-whole relationship among each of the rating categories. We also changed the main chart title to reflect this change as well. Now, let's see how to create the same data visualizations in ggplot2.

Demo (ggplot2)

We'll now visualize the same qualitative data using ggplot2. We've already downloaded and installed the ggplot2 package in our first set of demos. However, we need to load the ggplot2 package using the library command, like we saw in the ggplot2 demo in module one. In addition, we already have our working directory set and our movies loaded from the first demo in this module, so, we're ready to start visualizing our data. Now, let's create a frequency bar chart of our movie rating category in ggplot2. We'll use the ggplot command, set the data to our movies data frame, set the aesthetics with our X-axis set to the Rating variable, we'll add a bar geom to specify that we want to create bars as our geometry object, and we'll add a main chart title. Note that the plus sign in used to indicate that we're adding a layer to our ggplot command. As we can see, this produces a basic frequency bar chart for our movie rating categories. To create a horizontal bar chart we use the exact same command as we did before, however, we'll add a Cartesian coordinate flip operation to our existing command. This tells ggplot 2 to flip the X and Y-axis orientation so that we'll end up with a horizontal bar chart. To create a Cleveland dot plot we use the same command as we did for the horizontal bar chart, however, instead of using the bar geom we'll use the point geom instead. In addition, to specify the location of the points along the X-axis we'll set the statistical transform to count. This tells ggplot2 that we want to use the

frequency of the observations as the value to determine the location of the points on the X-axis. To create a pie chart in ggplot2 we have to use a bit of a trick. We'll actually be creating a stacked bar chart, but re-projecting it into a polar coordinate system rather than projecting it as a straight line. We use the ggplot2 command, set the data to movies, set the aesthetics with X set to an empty string indicating that there'll be no variable on the X-axis, and the fill color set to the Rating category indicating that we want each rating to be represented by a different color. Add the bar geom indicating that we want to display a single-stacked bar chart, then re-project our data into a polar coordinate system, which maps the data to a circle, rather than a straight line, and set the theta value to Y indicating that we want the angle of our pie chart wedges to map to the number of observations in each rating category. Finally, we'll add the chart title. Now, we can visualize the proportion of movies in Rating category as a part of the whole. Finally, we'll answer Natalia's second question about the proportion of movies that have won awards versus movies that have not won any awards. We'll create this pie chart the same way we created our previous pie chart, however, we'll use the Award variable instead of our movie Rating variable. As you can see, this produces a better usage of a pie chart than our previous pie chart. However, it still has some limitations. We present our data visualizations to Natalia and she's very impressed with our analysis. We've answered both of her questions in way that can be easily communicated to her, and with a bit of extra work could be potentially be communicated to a wider audience.

Summary

In this module, first we learned about data visualizations for qualitative univariate analysis, that is, the analysis of a single categorical variable. Next, we learned how to create these data visualizations using the base graphic system. Then, we learned how to create these same data visualizations using lattice. Finally, we created these data visualizations using ggplot2. In the next module, we'll learn how to create and interpret data visualizations for a single numeric variable.

Visualizing One Numeric Variable

Introduction

Hello again, and welcome to our next module on Data Visualization with R. I'm Matthew Renze with Pluralsight, and in this module we'll learn how to create and interpret data visualizations for a single numeric variable using R. As an overview of this module, first, we'll learn about data visualizations for quantitative univariate analysis, that is the analysis of a single numeric variable. Next, we'll learn how to create these data visualizations using the base graphic system. Then, we'll learn how to create these same data visualizations using lattice. Finally, we'll create these data visualizations using ggplot2. To help orient ourselves in relation to the rest of the course this module will cover quantitative univariate analysis, that is, the analysis of a single numeric variable. In addition, we'll be building upon what we learned in qualitative univariate analysis, and preparing ourselves for bivariate data analysis as well.

Quantitative Univariate Analysis

The second type of analysis we can perform is quantitative univariate analysis, which is just a fancy term for the analysis of a single numeric variable. Quantitative univariate analysis allows us to answer questions about data concerned with the location of the data, that is measures of central tendency, the spread of the data, that is measures of dispersion, and the shape of the data for a single numeric variable. We'll take a look at four types of data visualizations that can help us answer these types of questions next. The first type if data visualization for quantitative univariate analysis is a dot plot. With a dot plot we can quickly see how the data are distributed along the variable. On the X-axis we have the Movie Runtime in minutes. There's no dimension on the Y-axis since the plots are all on a single one-dimensional line. Each point on the dot plot represents an occurrence of an observation with the corresponding value on the X-axis, so we are essentially seeing a visual representation of the observations along the dimension of a single quantitative variable. Dot plots can be used to inspect the location of individual observations and how they are distributed along a numeric variable. For example, if we're interested in knowing the minimum, and maximum Movie Runtimes, we can see that the minimum is approximately 40 minutes, and the maximum is approximately 220 minutes. In addition, we can see that there are generally more observations in the middle of the dot plot than on the ends of the dot plot. However, we cannot determine much more from this data visualization. Dot plots are the simplest of the quantitative univariate data visualizations as a result of this simplicity, however, they're of very limited use. In addition, they only work well for visualizing quantitative variables with a small number of observations. Otherwise, overlapping points hide the true number of observations at any given point along the dot plot. However, there are visual techniques, like using alpha transparency, like is shown in this plot, that can help improve the usefulness of dot plots. In addition to using alpha

transparency, we can also use other techniques, like jitter, that can improve the usefulness of dot plots. This type of dot plot, often referred to as a jitter plot, contains the same data as our dot plot, but uses both empty circles and jitter to improve the usefulness of the dot plot. Jitter, in the context of a dot plot, is a technique of adding random noise along the Y-axis to each point so that they do not visually overlap with one another. So, we still do not have a dimension of the data mapped to the Y-axis, but we're now randomly placing the points along the Y-axis. This makes it easier to see how the points are actually located at a specific point along the X-axis. Jitter plots are quite useful when you have a larger sample size that you're trying to visualize, and they're also especially useful if you have a discreet numeric variable, that is a variable consisting of only whole number values rather than a continuous numeric variable, that is, a variable where the numeric values can fall at any position along the real number line. This is because discreet numeric values, like Movie Runtime in minutes, always fall on whole number, that is a whole number of minutes. So dot plot visualizations tend to stack the points one on top of another. Jitter plots visually unstack these values along the Y-axis so that it's easier to see the true number of observations at each whole number interval of the variable. However, there are other data visualizations that are often better at providing us with the same information. The next type of data visualizations for quantitative numeric variables is a box plot, also known as a box and whiskers plot. In this plot we're looking at the same data that we saw in the two previous plots. On the X-axis we have our Movie Runtime in minutes, we have no dimension again on our Y-axis. A box plot provides us a visual representation of the five-number summary statistics for a single quantitative variable. A five-number summary consists of the minimum value observed for the variable, excluding any outliers, the lower, or first quartile, the median, the upper, or third quartile, the maximum value observed for the variable, excluding any outliers, and any outliers in the data set represented as small circles, which are values greater than 1. 5 times the interquartile range added to the upper quartile, or less than 1. 5 times the interquartile range subtracted from the lower quartile. Outliers are values that fall outside of the statistically likely minimum or maximum values entailed in a distribution of values. Outliers may be naturally occurring, or the result of measurement error, and thus may be of interest, or may need to be excluded from our analysis. A box plot allows us to quickly answer questions about the location of the data. We can quickly determine where the minimum, maximum, and middle values of our variable are located. In addition, a box plot allows us to answer questions about the spread of the data. For example, are the values widely distributed or narrowly distributed? With a box plot we can quickly determine the range of the data, that is, the difference between the minimum and maximum values, and the interquartile range, that is, the difference between the upper and lower quartiles. Finally, a box plot allow us to answer questions about outliers for a single numeric variable. For example, does the variable

contain any outliers, what are the values of the outliers, and how many outliers does our variable contain? These key features of a box plot make them an extremely useful data visualization for deriving information about the distribution about values for a single numeric variable. Next, we can represent the same data as a histogram. A histogram shows us an approximation of the shape of a distribution of values by grouping them into equal with partitions called bins. On the X-axis we have our Movie Runtime in minutes. On the Y-axis, we have our number of observations contained within each bin. The height of the bar represents the number of observations contained within a specific bin. The width of the bins are all equal. Bins are essentially equal with buckets containing the number of observations that occur within the minimum and maximum values of the bucket. In other words, each bar shows us the number of values that occur between the minimum and maximum value of each bin. Histograms also allow us to answer questions about the location and spread of data in addition to answering questions about the shape of the distribution. For example, are the data positive skewed, that is, is the mass of the distribution on the left side, and thus the right tail of the distribution longer than the left, or are the data negatives skewed? Is the distribution skewed to the left? Are the data normally distributed, that is, does the distribution of values follow a bell-shape curve? And is the peak of the distribution more sharply peaked than the normal distribution, or is it flatter than a normal distribution? We can also change the size of the bins to create either, a more course grain representation of the distribution, or a more fine grain representation of the distribution. Histograms are often confused with bar charts. However, while a bar chart is showing a measure across multiple categories, a histogram is showing the frequency of observations of a numeric variable grouped into bins. Finally, we can represent these same data as a density plot. Density plots, more precisely called kernel density plots, show us the shape of the distribution of values similar to a histogram. On the X-axis we have our Movie Runtime in minutes. On the Y-axis, we have the probability, which ranges from 0 to 1, of a movie occurring at a specific location on the X-axis, that is, the likelihood that a movie will be of a specific length. Density plots allow us to answer similar questions about the location, spread, and shape of data as histograms do. However, where a histogram showed us the step-wise approximation of the distribution using bins, a density plot shows us a smooth representation of the distribution as a function. In addition, while the histogram showed us the number of observations in each bin, the density plot shows us the probability density function. A probability density function tells us the likelihood expressed as a number from 0 to 1 that an observation will be found at a specific point on the line. We can use the area under the curve to compute the likelihood that an observation will occur within a range of values. Thus, the entire area under the curve integrates to a value of 1. To explain how all this works requires a bit of calculus, but we won't need to know any of that for the demos in our course, we're just going to

be looking at the shape of our data, not calculating probabilities using a probability density function.

Demo (Base)

Natalia was so impressed by how quickly we were able to answer her previous questions that she comes to us with a more difficult set of questions. She's interested in learning about movie runtimes, that is, the length in minutes of each movie in our streaming video library. She has several questions she wants to answer, for example, what is the average movie runtime? Are there any outliers, that is, movies that are either significantly shorter or longer than the rest? How spread out are the values, that is are they tightly clustered around a single movie runtime, or are they spread out over a wide range of values. And, what is the shape of the distribution of movie runtimes? For example, is a bell-shape distribution, a bi-modal distribution, or some other shape? Given these questions we decide to create some quantitative univariate data visualizations to help answer her questions. Let's begin our quantitative univariate analysis by creating a dot plot of movie runtimes. We already have our working directory set and our movie data loaded, so let's get started. To create a dot plot of movie runtimes we use the plot command, set the X-axis to our Movie Runtime variable, and set the Y-axis to 0 for each movie. We do this using the replicate command, which creates a vector of the same length as our Movie Runtime variable, but with each value in the vector set to 0. Doing so will draw each plot on a single, straight line parallel to the X-axis. In addition, we'll set the main chart title and the X-axis label. We'll set the Y-axis label to an empty string to prevent the plot from rendering an irrelevant axis label, and we'll set the Y-axis text to n to suppress rendering of the Y-axis text. These last two steps are necessary because we don't have a dimension of data assigned to our Y-axis so we don't want R to render any labels or text for that axis. As we can see, this creates a dot plot of our movie runtimes. To make is easier to identify overlapping points we'll add alpha transparency to our previous dot plot. To do this we'll use the same command as our previous dot plot, however, we'll set the pch parameter to 16, which tells R to set the plot character to a solid circle. In addition, we'll set the color of the plot character to black with an alpha transparency of 0. 1. We do this with the rgb command setting the red, green, and blue parameters to 0, which creates the color black, and we set the alpha transparency parameter, that is the opacity to 0. 1, indicating that we want our points to be 10% opaque, in other words, we want them to be 90% transparent. This means that as points overlap their opacity will be added together so multiple overlapping points will become darker. However, since we're only using 10% opacity we'll only be able to see up to 10 overlapping movies where the opacity will become 100%. Any more than 10 movies with the same runtime will just

look like a solid black circle. A better solution to the problem of overlapping points is to add jitter to the points. We'll just be applying jitter along the Y-axis so that the jitter won't affect where our observations are located on the X-axis. To do this we'll use the same command as our first dot plot. However, we'll use the jitter command on the Y-axis values. As we can see this creates a dot plot with correct values in the X-axis, but randomly assigns locations to the Y-axis. This makes it much easier to see where observations are concentrated along the Movie Runtime dimension. As we can see, the average movie runtime appears to be about 100 minutes long. This information will help Natalia answer her first question about the average movie runtime. An even better visualization for our distribution of movie runtimes is a box plot. We can create a box plot using the boxplot command. We set the x parameter to our Movie Runtime variable, set the orientation to horizontal, set the main chart title, and set the X-axis label. As we can see, we now have a visual representation of our five-figure summary statistics. In addition, we have a visual representation of the outliers contained in the Movie Runtime variable. So now, we can show Natalia that there are in fact many outliers contained within our streaming movies library. This visualization also show us that the median movie runtime is approximately 100 minutes, the data appeared to be relatively concentrated around the center of the distribution, and the data may be positively skewed. That is, the tail on the right appears to be longer than the tail on the left. However, we'll need to look at other data visualizations to know for sure. To create a histogram of movie runtimes we used the histogram command, set the X-axis to movie Runtime, set the main chart title, and set the X-axis label. This creates a histogram with the default number of bins, that is, 20 equal with bins. To create a more core-screen histogram, that is a histogram with fewer, yet wider bins, we use the same command as our previous histogram. However, we change the number of breaks from the default of 20 to 10 breaks instead. This produces a histogram with 10 wider bins. To create a more finely-grained histogram, that is, a histogram with more, yet narrower bins, we use the same command as our first histogram, however, we change the number of breaks to 30 breaks instead. This produces a histogram with 30 narrower bins. To create a density plot, that is kernel density estimation plot, we use the plot command, set the x parameter to the density of the Movie Runtime variable, using the density command, set the main chart title, and set the X-axis label. As we can see, this produces a density plot of our Movie Runtime distribution. With this visualization we can now show Natalia that the distribution of movie runtimes is tightly concentrated around the center of the distribution, that is around 100 minutes, is positive skewed, that is, the tail on the right is longer than the tail on the left, and the distribution is unimodal, that is it has one peak like the normal distribution does. Finally, to demonstrate how all of these quantitative univariate data visualizations are just different ways to visualize the same underlying data we'll plot them all one on top of the other. To do so, we'll tell

our R plot our next four visualizations on a visual grid containing four rows and one column. We do this using the par command, which stands for graphical parameters, and set the mfrow parameter, which is the multi-figure row-wise parameter to a vector containing the number of rows, which we'll set to four, and the number of columns, which we'll set to one. Now, we'll create our dot plot, box plot, histogram, and density plot. Finally, we'll set our graphical parameters back to a single row and single column layout. As we can see, this produces all four of our data visualizations stacked top to bottom sharing the same scale and X-axis. It's very easy to see now that these are four visual representations of the same underlying data.

Demo (Lattice)

Now let's see how create the same quantitative univariate data visualizations with lattice. First, we'll start with a dot plot. To create a dot plot we use the stripplot command. Set the x parameter to the formula tilde Runtime, which tells lattice that we want to display the runtime variable by itself, set the data parameter to our movies data frame, set the main chart title, and set the X-axis label. As we can see, this produces a basic dot plot of the Movie Runtime variable. If we want to add jitter to our dot plot in lattice we use the same command as before. However, we also set the jitter parameter to TRUE, and specify the amount the jitter we want added, which in this case we'll set to 0. 5. This will randomly position our points along the Y-axis to make it easier to see our Movie Runtime distribution. To create a box plot we use the bwplot command. The b and w stand for box and whiskers. Set our x parameter to tilde Runtime, set our data to movies, set our main chart title, and set our X-axis label. This produces a box plot of our Movie Runtime. To create a histogram we use the histogram command, set our x parameter to tilde Runtime, set our data to movies, set our main chart title, and set our X-axis label. This creates a histogram of our Movie Runtime variable. To create a density plot we use the densityplot command, set the x parameter to tilde Runtime, set the data parameter to movies, set the main chart title, and set the X-axis label. As we can see this produces a density plot of our Movie Runtime variable. Finally, we'll create another small multiples representation of all four of our quantitative univariate data visualizations, like we did at the end of the last demo. To do so with lattice requires a slightly different approach than with the base graphic system. Instead, we'll assign each of our plots to a variable, and then we'll print each variable by specifying the position using a vector containing four numbers, that is xmin, ymin, xmax, and ymax indicating the lower left and upper right corners of a rectangle in which the specified plot will be printed. We'll do this for our dot plot, box plot, histogram, and density plot. As we can see, these four data visualizations are just different visual representations of the same underlying data. In addition, I should note that these data

visualizations don't perfectly line up with one another just due to the way lattice renders in the grid format.

Demo (ggplot2)

Finally, we'll see how to create the same quantitative univariate data visualizations using ggplot2. First, we'll create a dot plot using the ggplot command, set the data to our movies data frame, set the aesthetics with x set to Runtime, and the statistical transform set to count indicating that we want to display counts of our movie runtimes, set our geom to the dot plot geom, with the binwidth set to 1 indicating that we want to use a single point to represent each movie runtime, set our main chart title, and set our X-axis label. As we can see, this creates a dot plot of our movie runtime distribution, however, each dot is stacked one on top of the other on the Y-axis. This makes it very easy to see the shape of the distribution in the same way as a histogram. In addition to stacking the dots upward along the Y-axis we can also stack them uniformly around the Y-axis line instead. We refer to this a violin-style dot plot in reference to a violin plot, which gets its name because its shape often resembles that of a violin. To create a violin-style dot plot we use the same command as our previous dot plot, but instead we set the stack direction parameter of our dot plot geom to center. As we can see, this renders the dot plot with the dots stacked around the Y-axis line. To create a box plot of our Movie Runtime variable we use the ggplot command, set the data to movies, set the aesthetics with x set to Movie Runtime, and Y also set to movie Runtime, set our geom to the boxplot geom, flip the coordinates so that our box plot is displayed horizontally along the X-axis, set our main chart title, set our X-axis label to an empty string, which will clear our Y-axis label since we rotated the chart, set our Y-axis label to Runtime in minutes, which will set our X-axis label, since we rotated the chart as well. And finally, we need to clear our Y-axis text and tick marks using a theme, and set our Y-axis text to a blank element, and set our Y-axis tick marks to a blank element as well. As we can see, this creates a box plot of our Movie Runtime variable. Please note that it's a bit confusing setting the X and Y-axis labels once we've flipped our coordinate system, however, this is just something you have to learn to get used to with ggplot2. To create a histogram we use the ggplot command, set the data to movies, set the aesthetics with x set to Runtime, set our geom to the histogram geom with a binwidth of 10, set our main chart title, and set our X-axis label. As we can see, this creates a histogram of our Movie Runtime variable. Note that with ggplot2 we set the width of the bin rather than setting the number of bins, like we did with the base plotting system. To create a density plot we use the ggplot command, set our data to movies, set the aesthetics with x set to Runtime, set our geom to the density geom, set our main chart title, and set our X-axis label. This

creates a density plot of our Movie Runtime variable. Finally, we'll create a small multiples representation of our four quantitative univariate data visualizations. Doing this in ggplot2 is different from both the way we did it in the base plotting system, and in lattice. First, we'll clear the existing plot window using the dev. off command. This command shuts off the current graphics device, which is our plot window. Next, we'll load the grid package, which allows us to plot data visualizations in a grid in our graphics device. Then we'll create a viewport with a grid layout containing four rows and one column. Next, we'll push that viewport under the root of our viewport tree, which in this case is our graphics device. Then we'll create our dot plot. We'll print our dot plot to the first row and first column of our grid. We'll create our box plot and print our box plot to the second row and first column of our grid. We'll create our histogram, and print our histogram to the third tow. We'll create our density plot, and print our density plot to the fourth row. As we can see from our small multiples representation these four data visualizations are just different ways of presenting the same underlying univariate quantitative data. Using the univariate quantitative data visualizations that we created Natalia was able to answer all of her questions regarding the location, spread, and shape of the distribution of movie runtimes in our streaming video library. She thanks us for our help and tells all of her co-workers how impressed she was with our data visualization skills.

Summary

In this module, first, we learned about data visualizations for quantitative univariate analysis, that is, the analysis of a single numeric variable. Next, we learned how to create these data visualizations using the base graphic system. Then, we learned how to create these same data visualizations using lattice. Finally, we created these data visualizations using ggplot2. In the next module we'll learn how to create and interpret data visualizations for the relationship between two categorical variables.

Visualizing Two Categorical Variables

Introduction

Welcome back to Data Visualization with R. I'm Matthew Renze with Pluralsight, and in this module we'll learn how to create and interpret data visualizations for the relationship between two categorical variables using R. As an overview of this module, first, we'll learn about data visualizations for qualitative bivariate analysis, that is, the analysis of the relationship between two categorical variables. Next, we'll learn how to create these data visualizations using the base graphic system. Then, we'll learn how to create these same data visualizations using lattice. Finally, we'll create these data visualizations using ggplot2. To help orient ourselves in relation to the rest of the course this module will cover qualitative bivariate analysis, that is the analysis of two categorical variables. In addition we'll be expanding upon what we learn from univariate analysis and preparing ourselves for the remainder of bivariate analysis.

Qualitative Bivariate Analysis

The first type of bivariate analysis that we can perform is bivariate analysis of a qualitative variable. This means the analysis of the relationship between two categorical variables. When we're performing qualitative bivariate analysis we're typically interested in the joint frequency of observations, that is, the frequency of occurrences of observations at the intersection of two categories. For example, movie genres and movie ratings. We can display bivariate frequency distributions using a contingency table, which is a table in matrix format containing the frequency of observations at the intersection of the rows and columns. For example, in this table, if we're interested to see how many PG-rated comedies there are in our data set we would find the value contained at the intersection of the comedy row and the PG column. In addition, our contingency table can contain row totals on the right-hand margin of each row showing the sum of all observations for the rows, and column totals on the bottom margin showing the sum of all observations for each column. These are referred to as marginal frequencies. If we're interested in proportions or parts of the whole our contingency table can also contain the relative frequency of observations expressed as either a percentage or value between 0 and 1. The percentages and all cells of the table would sum to 100%, or if you were using a value between 0 and 1 the value in all cells of the tables would sum to 1. A contingency table is often referred to as a two-dimensional frequency table, a two-way table, or a cross-tabulation matrix. The first type of qualitative bivariate data visualization that we can create is a grouped frequency bar chart, also known as either a clustered or dodged frequency bar chart. On the X-axis we have the category of our first variable. For example, movie rating categories. on the Y-axis we have the frequency of observations, for example, the number of movies. For each category along the X-axis we have one bar for each category contained in the second variable, for example, movie award status. We use

color to help us easily identify which bar belongs to which category of the second variable. The height of each bar corresponds to the joint frequency of observations at the intersection of the first and second category. For example, if we were interested in seeing if there were more G-rated movies that didn't win awards versus R-rated that won awards we could find the G-rating category on the X-axis, and then the bar representing the movies that did not win any awards within the G-rating category, and compare it to the bar for the R-rated categories that won awards. As we can see, there were significantly more R-rated movies versus G-rated movies that did not win awards. This type of data visualization works well when we're interested in answering questions that involve comparing the joint frequency of observations across two categorical variables. It makes it very easy to compare the joint frequencies within each category of the first variable across categories of the second variable, for example, R-rated movies that won awards versus R-rated movies that didn't win any awards. It makes it relatively easy to compare the joint frequencies within a single category of the second variable and across multiple categories of the first variable, for example, G-rated movies that won awards versus R-rated movies that won awards. And, it makes it relatively easy to compare the joint frequencies across both categories, for example, G-rated movies that didn't win any awards versus R-rated movies that won awards. The second type of data visualization that we can create for qualitative bivariate analysis is a stacked frequency bar chart. On the X-axis we have the categories of our first variable, for example, movie rating categories. On the Y-axis we have the frequency of observations, for example, the number of movies. We use color to subdivide each bar by the second categorical variable, for example, movie award status. The height of each segment of each bar corresponds to the number of observations at the intersection of both categorical variables. In addition, the height of the bar, as a whole, represents the marginal frequency of each category of the first variable, in this case, each movie rating category. For example, we can again compare the number of G-rated movies that did not win awards versus the number of R-rated movies that won awards. This type of data visualization works very well to answer questions that involve a comparison of the marginal frequency of one of the two categorical variables that also require knowledge of the joint frequency of observations across both categories. In addition, this data visualization captures the essence of a part of the whole for the second categorical variable grouped by the first categorical variable. This knowledge of the marginal frequency of one of the two categorical variables comes at the expense of making it more difficult to compare the joint frequencies across categories because we've stacked our bar segments. In addition, we cannot easily derive the marginal frequency of the second category, in this case, the award status, using this data visualization. Also, we are only able to see the part of the whole relationship for the second categorical variable, that is award status, grouped by the first categorical variable, that is ratings.

Because of this, it's important that we choose this data visualization when comparing the marginal frequency of one category is more important that comparing the joint frequency across categories. In addition, we need to carefully choose which variable we want to map to our X-axis since this is the variable whose marginal frequencies we will be able to easily compare across it's categories, and which variable we want to map to our bar segments since this is the variable that will convey a part of the whole relationship. The third type of data visualization that we can create for qualitative bivariate analysis is a 100% stacked frequency bar chart. This chart is identical to the previous stacked bar chart, however, we are now stretching the bars along the Y-axis to fill 100% of the chart area. So now, the height of each bar segment represents the proportion of observations of each of the categories of the second categorical variable grouped by the first categorical variable. In this example, the bar segments represent the proportion of movies that have won awards versus the proportion of movies that have not won any awards grouped by the movie rating categories. This type of data visualization works well when we want to answer questions that involve a comparison of the relative frequency of observations of one category grouped by another category. For example, if we're interested in knowing if there're proportionately more G-rated movies that won awards versus R-rated movies that won awards we can compare these two bar segments. As we can see, there're proportionately more G-rated movies that won awards than R-rated movies that won awards. In this data visualization we give up our knowledge of the marginal frequency of each category of the X-axis variable in exchange for knowledge of the relative frequency of observations of one of our two variables, that is, the second variable, which is grouped by the X-axis variable. So, we should choose this data visualization when we're interested in knowing the relative frequency of one variable grouped by another variable. In addition, we must choose our X-axis variable, and our bar segment variable with care, so that we communicate the correct relative frequency in our chart. Note that we can flip each of these three previous types of bar charts on their side to make them horizontal bar charts. However, since we already learned how to do this we'll skip it in this module and the remaining modules that contain other types of bar charts. The fourth type of data visualization that we can create for qualitative univariate analysis is a spine plot. On the X-axis we have our first categorical variable, for example, movie rating. On the Y-axis we have our second categorical variable, for example, movie award status. The width of each bar represents a proportion of observations in each of the categories on the X-axis. The height of each bar segment on each stacked bar represents the proportion of observations in each category on the Y-axis. So the area of each rectangular bar segment is proportional to the relative frequency of observations at the intersection of the two categories. The color corresponds to the category on the Y-axis. Color is used to help visually distinguish categories since they may vertically overlap from one bar

segment to the next. For example, in this spine plot, if we're interested in comparing the number of G-rated movies that have won awards relative to the number of R-rated movies that have not won any awards, we could find the bar segment at the intersection of the G-rated movie column and the row containing movies that have won awards using color as our guide. Then we would compare it to the area of the bar segment at the intersection of the R-rated movie column and the row containing movies that have not won any awards, again using color as our guide. Spine plots are a useful data visualization when we're attempting to answer questions that involve the comparison of the relative frequency of observation in two categorical variables as a part of the whole relationship. This chart is different from the stacked bar chart in the sense that we can see the relative frequency of observations across both of the categorical variable, rather than just one variable. Spine plots do, however, have some limitations. For example, categories with no observations are represented by the absence of a specific colored bar segment in a specific column. There's a variation on spine plot called a mosaic plot where the spacing between the bars and visual segments produces a cleaner data visualization. It works the same as a spine plot in that the width of the tiles represent the proportion of observations on the X-axis, and the height of the tiles represent the proportion of observations on the Y-axis. Thus we can use the area of the tiles to determine the number of observations at the intersection of each of the two variables. Color is once again used to help distinguish categories on the Y-axis since they may not line up vertically. Black dashes indicate that there are no observations at the intersection of the two categories. In this mosaic plot, however, we have data in all combinations of all categories, so we don't see any of the black dashes. Once again, if we're interested to see how many G-rated movies there are in our data set that won awards we would find the tile at the intersection of the G-rated column and the row containing movies that won awards. A mosaic plot can be used to answer the same questions as a spine plot, however, a mosaic plot has cleaner aesthetics than a spine plot, and thus is generally preferred over a spine plot in most cases.

Demo (Base)

Our co-worker Natalia showed our data visualizations to her boss Tony, who is the director of sales in marketing at NetFilms. Afterwards, Tony approached us to ask if we could help him, answer a question that he has been pondering as well. Tony wants to know how many award-winning movies there are in each of our rating categories so he can compare them to see if we have an equal balance in each category. As we dig a bit deeper, we discover that what he actually wants to know is what proportion of movies in each rating category are award winning films versus the proportion of movies that have not won any awards. So let's help Tony answer his

questions by creating the qualitative bivariate data visualizations that we just learned about. To create our qualitative bivariate data visualizations we first want to create a contingency table for our rating and award categories. Our first three data visualizations are going to want the data to be structured with ratings on the columns of the contingency table and awards on the rows of the table. However, the remaining data visualizations are going to prefer the data in the opposite order, that is, ratings on the rows and awards on the columns, but we'll get to that shortly. To create a contingency table we use the table command, set the rows of the contingency table to the award column in our movies data frame, set the columns of our contingency table to our rating column, and we'll assign this table to a variable called awards. Now, let's display the contingency table so we can see our results. As we can see, we have our movie ratings on the columns and our award status on the rows of the contingency table. Each cell in the contingency table represents the frequency of observations at the intersection of the award category on the rows, and the rating category on the columns. For example, we have 849 R-rated movies that have won one or more awards. First, let's create a grouped frequency bar chart using the base plotting system. To do so we'll use the barplot command, set the height parameter to our contingency table, set the beside parameter to TRUE indicating that we want our bars to be grouped beside, that is alongside of one another, set the main chart title, set the X-axis label, set the Y-axis label, and we'll create a legend containing the labels No and Yes for each of our award categories. We'll set the position of the legend to the top left-hand corner of the plot area, and set the title of our legend to Awards. As we can see, this creates a grouped frequency bar chart. We can use this chart to show Tony the number of movies that have won awards versus the movies that have not won awards in each rating category. Next, let's create a stacked frequency bar chart. To do so we'll use the same command and parameters as before, however, this time, we'll set the beside parameter to false, or we can just exclude it, since the default is false. As we can see, this creates a stacked frequency bar chart of the same data. This chart would be useful if Tony thinks that comparing the rating categories is more important than comparing the award categories within each rating category. However, this is isn't the case based on the qualifying questions that we asked Tony before we began creating our data visualizations, so we'll move on to a more appropriate data visualization to answer his questions. Third, we can create a 100% stacked frequency bar chart. To do so we'll have to take our contingency table containing the counts of the movies and convert those values into relative frequencies, that is, a number between 0 and 1 that represents the proportion of movies in each combination of rating and award categories. We'll do this using the apply command to apply a function over the columns of the awards table. We specified the table that we want to apply the function to, that is the awards table, the number 2, which indicates that we want to apply the function over the columns of the

table, note that we would use a 1 if we wanted to apply the function over the rows, and we specify the function that we want applied, which in this case, is the function x divided by the sum of x, which will compute the proportion of movies in each award category relative to all movies contained within each rating category. To clarify what we've done we'll first display the original awards contingency table, and next we'll display the new proportional contingency table. As you can see, what we've done in convert these joint frequencies into relative frequencies within each rating category. And please note that there relative frequencies are just the relative frequencies within each rating category, not eh frequencies relative to all observations, that is, these values sum to 1 within each rating category rather than summing to 1 for the entire table. We'll be working with the second type of relative frequency table with our spine plots and mosaic plots coming up soon. Now, we can use this relative frequency table to create our 100% stacked frequency bar chart. To do so we'll use the barplot command using the same parameters as the previous stacked frequency bar chart. However, we'll change the height parameter to our new relative frequency table called Proportions. In addition, we'll update our labels to reflect that we're dealing with proportions rather than counts. As we can see, this produces a 100% stacked bar chart of our data. This should make it much easier for Tony to see that we have similar balance of award-winning movies across each rating category, but slightly more in the G-rated movie category than the other three categories. Unfortunately, due to the way that the plots in R prefer to have their data structured to be rendered correctly we're going to have to replace our existing contingency table with a new contingency table with the rows and columns flipped so that now Ratings will be on the rows and Awards will be on the columns. We refer to this process of swapping rows and columns as transposing, or pivoting rows and columns within the table. We'll also rename the award status categories from TRUE and FALSE to Yes and No respectively to make labeling our remaining data visualizations easier. Just to clarify what we've done, let's display the new transposed contingency table, as we can see, we have Ratings on our rows and Award status on our columns. Each of the cells, however, still contains the frequency of observations at the intersection of each rating and award category. If we'd like to take our relative frequency data visualizations a step further we can create a spine plot to show the relative frequency of observations in each combination of categories relative to all observations. To do so we'll use the spineplot command, set the x parameter to our new awards table, set the main chart title, set the X-axis label, and set the Y-axis label. As we can see, this creates a spine plot showing the relative frequency of observations for each combination of categories relative to all observations. Tony could use this data visualization is he's interested in seeing how the observations compare as parts of the whole. Finally, if we'd like derive the same information as a spine plot, with a cleaner set of aesthetics, we can create a mosaic plot instead. To do so, we'll use

the mosaicplot command and set all of the parameters to the same values as their spine plot. As we can see, this produces a similar plot as our spine plot. However, there are now gaps between the Y-axis breaks of each of the tiles. In addition, the order of the tiles on the Y-axis is reversed by default. And the X-axis labels are placed at the top of the chart rather than the bottom of the chart. There's no color coding of the categories along the Y-axis by default, however, we can easily add color for clarity if we choose. Now, let's learn how to create the same data visualizations in lattice.

Demo (Lattice)

We'll now learn how to create our qualitative bivariate data visualizations in lattice. First, we'll create a grouped frequency bar chart in lattice, to do so, we'll use the barchart command, set the x parameter to our awards contingency table from the previous demo, set the stack parameter to false indicating that we want our bars to placed side by side for each group, set the horizontal parameter to FALSE indicating that we want our bars to be oriented vertically, set our main chart title, set our X-axis label, set our Y-axis label, and we'll add a legend to the upper left-hand side of our plot area by setting the x location of the legend to 0. 05, that is 5% of the distance along the X-axis, setting our Y location of the legend to 0. 95, that is 95% of the way up the plot area, set our legend title to awards, and set the legend labels to Yes and No for true and false respectively. As we can see, this creates a grouped bar chart of ratings and awards. Next, we'll create a stacked frequency bar chart. We'll use the same command and parameters s our previous bar chart, however, this time we'll set the stack parameter to TRUE instead. As we can see, this produces a stacked frequency bar chart of our data. To create our 100% stacked frequency bar chart we'll need to create a relative frequency contingency table from our contingency table containing counts. We'll do this using the same apply function as we used in the last demo. However, this time we'll set the margin parameter to 1 instead of 2, indicating that we want apply this function over the rows rather than the columns like we did in the last demo. In addition, because this apply command puts our ratings on the columns and awards in the rows in the resulting matrix, we need to transpose this matrix so that the ratings are back on the rows and the awards are on the columns. Just to clarify what we've done here, we'll display the original awards contingency table containing counts and the new contingency table containing relative frequencies within each rating category. Now let's create a 100% stacked frequency bar chart. To do so, we'll use the bar chart command with all of the same parameters as our previous stacked bar chart, however, instead, we'll set the x parameter to our relative frequency contingency table and we'll change the labels to mirror the fact that we're now dealing with proportions rather than

counts of movies. We'll also move the legend outside of the plot area by placing it in the upper right-hand corner of the chart area. As we can see, this creates a 100% stacked bar chart of the same data we saw before. Unfortunately, neither spine plots nor mosaic plots currently exist in lattice, so to create these two plots in R we would need to create them using the base plotting system, like we did in the previous demo. Finally, let's see how to perform these same data visualizations in ggplot2.

Demo (ggplot2)

Now we'll learn how to create our qualitative bivariate data visualizations in ggplot2. First, we'll create a grouped frequency bar chart in ggplot2. To do this we'll use the ggplot command, set the data to our movies data frame, set our aesthetics with x set to our movie rating variable, our fill color set to our award variable, we'll add a bar geom with the position set to dodge indicating that we want our bars to be placed side by side within each group, add a main chart title, and to set the labels of our legend without having to rename the underlying data in our data frame, we'll set a discreet fill color scale, and relabel the first item to No and the second item to Yes. This last step is just a simple way of overriding the labels used in the legend. As we can see, this creates a grouped frequency bar chart for both our rating and award variables. Next, we'll create a stacked frequency bar chart. To do so we'll use the same command in parameters as our previous bar chart, however, this time, we can either set our position parameter to stack, or just omit the parameters since stack is a default. As we can see, this produces a stacked frequency chart of our data. Finally, we'll create a 100% stacked frequency bar chart. To do so we'll use the same command as our previous bar chart, however, this time, we'll set the position parameter of our bar geom to fill to indicate that we want the bars to fill the Y-axis, that is stretch to cover the entire Y-axis. We'll also update our chart and axis titles to reflect the fact that we're now dealing with proportions instead of counts. As we can see, this creates a 100% stacked bar chart of the same data. notice how for all three of these data visualizations using ggplot2 we didn't have to create a contingency table like we did for the base plotting system in lattice. In addition, we only need to change a single parameter in each of these three data visualizations to produce the correct chart. This is where the additional, initial complexity of ggplot2 is starting to pay for itself by making it much easier to create more advanced data visualizations than using the base graphic system. Please also note that both spine plots and mosaic plots do not currently exist in ggplot2, so unfortunately, we can't use ggplot2 to create these two data visualizations. We present our data visualizations to Tony and help him answer his questions about the proportion of movies that have won awards to the movies that have not won awards in each rating category. In

fact, after we showed him the mosaic plot of our movies data he told us that he has all sorts of ideas for other questions he wants to try answering using additional data visualizations. However, we're going to have to learn some more data visualizations to help him out with these other questions.

Summary

In this module, first, we learned about data visualizations for qualitative bivariate analysis, that is the analysis of the relationship between two categorical variables. Next, we learned how to create these data visualizations using the base graphic system. Then, we learned how to create these same data visualizations using lattice. Finally, we created these data visualizations using ggplot2. In the next module we'll learn how to create and interpret data visualizations for the relationship between two numeric variables.

Visualizing Two Numeric Variables

Introduction

Hello, and welcome back to Data Visualization with R. I'm Matthew Renze with Pluralsight, and in this module we'll learn how to create and interpret data visualization for the relationship between two numeric variables using R. As an overview of this module, first, we'll learn about data visualizations for quantitative bivariate analysis, that is the analysis of the relationship between two numeric variables. Next, we'll learn how to create these data visualizations using the base graphic system. Then, we'll learn how to create these same data visualizations using lattice. Finally, we'll create these data visualizations using ggplot2. To help orient yourselves in relation to the rest of the course this module will cover quantitative bivariate analysis, that is the analysis of the relationship between two numeric variables. In addition, we'll be expanding upon what we've learned so far and preparing ourselves for the last type of bivariate analysis.

Quantitative Bivariate Analysis

The second type of bivariate analysis that we can perform is quantitative bivariate analysis, which is the analysis of two numeric variables. Rather than using our movies data set for the first part of our quantitative bivariate analysis topic, we're going to use a data set that's a bit more visually interesting than our quantitative variables in our movies data set. In the chart on the left we have two variables pertaining to Old Faithful, one of the most predictable geysers on Earth, located in Yellowstone National Park. On the X-axis we have the duration of eruption in minutes, this is the independent, or predictor variable. On the Y-axis we have the duration of time before the next eruption in minutes, this is the dependent, or outcome variable. There's a relationship of interdependence between the eruption duration and the waiting duration that we would describe as a strong positive correlation between these two variables. We can see this relationship of interdependence by the linear regression line that's drawn on this plot. In quantitative bivariate analysis we're typically trying to answer questions that involve the relationship between two numeric variables, for example, how does one variable influence, predict, or correlate with a second variable? In addition, we often want to answer questions about the location, spread, and shape of a distribution of two variables at the same time. We can quickly assess these aspects by inspecting the two-dimensional kernel density estimation of the two variables distributions. In cases where one of our two quantitative variables is the dimension of time we typically want to answer questions changes in one variable over time. We refer to this type of analysis as time series analysis. There are also other types of questions that we might want to answer with quantitative bivariate analysis, but once again, these three should cover the basics. The first type of quantitative bivariate data visualization that we can create is a scatter plot. On the X-axis we have our first quantitative variable, which is typically our independent, or predictor variable, for example, the duration of the eruption in minutes. On the Y-axis we have our second quantitative variable, which is typically our dependent, or outcome variable, for example, the duration of time before the next eruption in minutes. Each point of the scatter plot represents an observation of a geyser eruption. The position on the X-axis corresponds to the duration of the eruption for that observation. And the position on the Y-axis corresponds to the wait time before the next eruption for that observation. Scatter plots are very useful when answering questions that involve the correlation between two variables. For example, do these points visually indicate a strong correlation, a weak correlation, or no correlation at all? In addition, is the correlation positive, or is it negative? In the case of our Old Faithful data visualization, visually we can see a relatively strong positive correlation exists between these two variables because we could imagine drawing a straight line through the center of these clusters of points that would fit these points very well. In addition, visually we can see that this is positive correlation because the line points to the upper right-hand corner of the plot area. This line that we've drawn through the points is referred

to as a linear regression line. Unfortunately, a detailed explanation of regression is outside of the scope of this course. However, we'll see how to draw a simple linear regression line on our scatter plot during our demos. Scatter plots are also useful for answering questions that involve the general location, spread, and shape of data based on the frequency of observations or density of the distribution of the two variables. For example, we can visually see that there are two clusters of data, one located in the lower left-hand of the plot area, and the other located in the upper right-hand of the plot area. In addition, we can see that these data are bimodal, that is, there are two separate clusters with separate centers to each cluster. However, scatter plots aren't the best data visualization for answering questions about density. The second type of data visualization we can create for quantitative bivariate analysis is a two-dimensional binned frequency heat map, sometimes referred to as a binned scatter plot or 2D histogram. On the X-axis we have our first quantitative variable, for example, the duration of the eruption in minutes. On the Y-axis we have our second quantitative variable, for example, the duration of time before the next eruption in minutes. The color of each square bin corresponds to the frequency of observations contained within the extent of the square. The darker the color the more observations contained within the bin. Frequency heat maps are useful for answering questions that involve the joint frequency of observations for two quantitative variables. For example, we can see that there are two clusters, their location, spread, and general shape within the visualization. In addition, frequency heat maps are very useful when you have a large number of data points, or observations that overlap visually, like discreet values that fall in whole numbers. Metaphorically, you can think of a frequency heat map as a two-dimensional histogram where we use color rather than heights of bars to represent the number of observations contained within each bin. Next, we have hexagonal binned frequency heat maps. These plots are used to answer the same questions as rectangular binned frequency heat maps, however, they use hexagonal-shaped bins rather than square or rectangle-shaped bins. Hexagonal bins are generally preferred to square or rectangular bins because hexagons produce a more accurate representation of the frequency of data in a two-dimensional plane. The explanation for why this is the case is rather complicated, but if you're interested in knowing more I highly recommend researching hexagonal binning further. In general though, you should prefer to use hexagon bins over square or rectangular bins. In addition to visualizing the frequency of observations as colored bins on a square, rectangular, or hexagonal grid, we can also visualize the two-dimensional kernel density estimation using one of four different types of data visualizations that involve representing density as elevation along the Z-axis. The Z-axis is the axis perpendicular to both the X and the Y-axis, which in this case, would be pointing through the screen towards you. The first of these four types of data visualizations that represent density as elevation along the Z-axis is a contour plot. On the X-axis we have our first

quantitative variable, which is typically our independent of predictor variable, for example, the duration of eruption in minutes. On the Y-axis we have our second quantitative variable, which is typically our dependent, or outcome variable, for example, the duration of time before the next eruption, The three-dimensional shape of the distribution is represented as a two-dimensional projection of elevation using contour lines. Contour lines work the same way in this data visualization as they do in topographical maps. For example, we can see that there are two peaks in the lower-left and upper right-hand corners of the distribution that taper off in all directions until we hit flat land in the upper-left and lower right-hand corners. Contour maps are useful for answering questions about the location, spread, and shape using the density of a joint probability distribution for two quantitative variables. However, unlike binned data visualizations, we can much more easily see the location, spread, and shape of the distribution. Where we can metaphorically think of our previous binned frequency plots as a two-dimensional extension of the histogram, the contour plot in the next three visualizations we'll see can be equivalently thought of as two-dimensional extensions of the density plot. Level plots are similar to contour plots, however, to increase the clarity of the data visualization, color is added to each level of the contour plot, that is each area with equal elevation. Level plots can used to answer the same questions as contour plots, but may, or may not, provide a visualization that's easier to read than contour plots. Some people intuitively understand level plots better than they do contour plots, however, please use your best judgement and choose the data visualization most appropriate for your audience. Also note that level plots have a lower data-to-ink ratio than contour plots because they require significantly more digital ink to communicate the same information. The next type of visualization for joint density distributions is a mesh plot. A mesh plot is three-dimensional rendering of the joint density distribution of two quantitative variables. On the X-axis we have our independent, or predictor variable, for example, the duration of eruption in minutes. On the Y-axis we have our dependent, or outcome variable, for example, the duration of time before the next eruption. On the Z-axis we have the density of our distribution at the corresponding x and y location. The three-dimensional shape of the distribution is represented as a three-dimensional projection of the surface of the distribution draped with a warped, rectangular grid, called a mesh. There are several pros and cons to using mesh plots. For example, the warped mesh helps us to more easily see the elevation of the distribution at any given point along the X and Y-axis. however, this ease of seeing the elevation in a more natural way comes at the expense of a more complex visualization, hidden surfaces, and sometimes a less than ideal visual perspective. Once again, it's important to keep in mind your audience and choose a visualization that is most appropriate for their needs. As a side note, I've seen variations on mesh plots where the contour lines were mapped to the surface of the distribution instead of a warped

mesh. However, I decided not to include this our course since it's essentially just a mesh plot with contour lines instead of a grid. The fourth type if visualization for joint density distributions is a surface plot. Surface plots are like mesh plots except that they do not have a warped mesh draped on the surface on the plot, rather, they drape color on the surface of the plot to indicate the density or elevation at any particular x or y coordinate. Surface plots can be used to answer similar questions as contour, level, and mesh plots. However, much like these three visualizations they have some pros and cons as well. For example, by giving up the warped mesh we can no longer easily trace any x or y gridline along the surface of the plot to see what elevation is at that point. However, the surface plot does make it easier to convey the relative elevation of multiple peaks despite the information distortions caused by the perspective. For example, it might be difficult to determine how much taller or shorted the first peak was relative to the second peak, especially if the perspective made it appear larger because it was closer to the virtual camera. However, with color, we can much more easily see that the first peak is much shorter than the second peak because the color of the first peak is much lighter than the color of the second peak. Once again, please keep your audience and these pros and cons in mind when choosing the best data visualization to answer your audiences question. Now let's switch gears to our three-time series-based data visualizations. In addition, we'll switch back to our movies data set to discuss these data visualizations. First, we have a step chart. On the X-axis we have our variable that represents time, for example, the year a movie was released. On the Y-axis we have our second quantitative variable, for example, total box-office revenue in millions of dollars. The height of each bar represents the value of the second variable for a given time period, in this case, the box office revenue for each year. This chart is referred to as a step chart because the values instantaneously increase or decrease between each time period. Step charts are useful for answering questions involving change in value over time where the value remains constant throughout each time period. For example, if our online streaming movie company changes the price of its membership once per year on the first of the year, but the price stayed the same throughout the rest of the year, this would produce an accurate visual representation of these data. We can also create step charts without the vertical segments called risers to leave the horizontal line segments for an even higher data-to-ink ratio. Our next time-series based data visualization is the line chart. Line charts are similar to step charts, except that rather than displaying values as constant throughout each time period they show the rate of change from one time period to the next. The steepness of the angle of the line provides a great visual representation of the rate of change from one time period to the next. Line charts are useful for answering questions that involve change in value over time for values that are continuously changing throughout time. In addition, line charts work better than step charts for inspecting the

rate of change over time using the angle of the line segments. In general, line charts are used more often than step charts, so, if in doubt, just use a line chart. Finally, our last time series based data visualization is the area chart. Area charts are similar to line charts except that they display the area underneath the line as a solid fill color. Area charts are also useful for answering questions that involve change of value over time for values that are continuously changing throughout time. However, area charts also convey a sense of the value or summation of the underlying data. For example, it would make sense to use an area chart for a total box office revenue by year because we're summing all of the dollars of revenue for each year. However, it might not make sense to use an area chart to display increases or decreases in the rate of change of wait times between geyser eruptions since these values represents the difference from one year to the next, so there's nothing actually being summed. In general, line charts are used more often than area charts, so if in doubt, just use a line chart. Now let's see how to create these quantitative bivariate data visualizations using the three plotting systems in R.

Demo (Base)

Now that Tony has seen how effective we are at answering questions using our data visualization skills, he comes to with a more difficult set of questions. He's now interested in knowing about how movie runtime and box office revenue are related. He's wondering if longer movies make more money on average and is curios if we should be adding more lengthy, yet costly, three hour cinematic epics to our streaming movies library. In addition, he is wondering how the average box office revenue in theaters has changed over time. He is curious to know if the theatrical release of movies is generating more or less revenue as more and more people stream their movies online. So, let's help Tony answer his questions using the new quantitative bivariate data visualizations that we just learned about. For our first quantitative bivariate data visualization, let's create a scatter plot in the base graphic system. To do so we use the plot command, set the X-axis to our movie runtime variable, set the Y-axis to our box office revenue variable, set a main chart title, set our X-axis label, and set our Y-axis label. As we can see, this creates a scatter plot with Movie Runtime on the X-axis, and Box Office Revenues on the Y-axis. To add a linear regression line to this scatter plot to show the correlation between the two variables, we're going to have to create a linear model. Statistical modeling is outside of the scope of this course so I'm just going to skip over how this is all working and just explain what we need to do in order to add the linear regression line to our plot. First, we need to create linear model using the lm command, which stands for linear model. We use a formula to describe the relationship between the variables in our linear model. In this case we use the formula box office revenue as a linear function of movie

runtime. Now we need to draw this linear model as a linear regression line on top of our scatter plot. To do so we use the lines command, set the x coordinates of our line to our Movie Runtime variable, which we have previously mapped to the X-axis in our scatter plot, set the y coordinates of our line to the fitted property of our linear model, which is the y coordinates for the line of best fit using the linear regression algorithm. We'll set the color of the line to red, and increase the line with the three to make is more visible. As you can see, this draws a linear regression line on top of our scatter plot. We can now show Tony that these two variables are positively correlated with one another. However, we would need to look at the numeric values in the linear model to determine how strongly these variables are correlated and to know precisely how much a movie's box office revenue increases for each additional minute of movie runtime. As a side note, I've done a more rigorous statistical analysis of these data in the past and the correlation is relatively weak. This is why it's important that we understand how to perform both statistical analysis and visual analysis of our data. So Tony shouldn't reasonably expect to make millions of dollars just by adding a bunch of extra hours of cutting room floor to each of our films. Next, we'll create a hexagonal binned heat map for our previous scatter plot using the hexbin package. To do so, first we need to download, install, and load the hexbin package. Now we'll create our hexbin plot using the hexbin command and we'll assign it to a variable called hexbin. Set our X-axis to our Movie Runtime variable, set our Y-axis to our Box Office Revenue variable, set the number of bins along the X-axis to 30, which will create a hexagonal grid with 30 cells along the X-axis, set our X-axis label, and set our Y-axis label. Now, we'll plot the hexbin object using the plot command, setting our x parameter to the hexbin object, and we'll set the main chart title. As we can see, this creates a hexagonally binned representation of our previous scatter plot. Please note that the X and Y-axis labels in the values labels appear to overlap. Unfortunately this appears to be a glitch in the current version of the hexbin package. We could fix it with some extra work, however, to keep things moving along we're just going to skip those steps for now and we'll see how to create a better hexbin plot in both lattice and ggplot2 later. To create our contour plot, level plot, mesh plot, and surface plot in the base graphic system, we need to create a two-dimensional kernel density estimation of our two variables using a package called mass. This package contains functions and data sets contained in the book titled Modern Applied Statistics with S. First, we need to download, install, and load the mass package. Next, we need to create our two-dimensional kernel density estimation using the kde2d command, which stands for kernel density estimation in two dimensions. Set our x coordinate to the data of our Movie Runtime variable, set our y coordinate to our Box Office Revenue variable, we'll set the number of grid points in both the x and y direction to 50, and we'll set the estimation function to a variable called density2d. Now to create a contour plot of our kernel density estimation we use the contour command, set

the X-axis to the x coordinates of our density function, set the Y-axis to the y coordinates of our density function, set the Z-axis to the z coordinates, that is, the density of our density function. We'll add a main chart title, an X-axis label, and a Y-axis label. As we can see, this creates a contour plot of our two-dimensional density function for our two variables. Unfortunately most of the density of this distribution is located at the center of this distribution so it doesn't make for all that interesting of a data visualization. You can probably now see why I chose to use geysers eruptions at Old Faithful for our plots during our discussion about the types of quantitative bivariate data visualizations. Next, we'll create a level plot of density. However, this level plot isn't exactly a level plot like we described during our discussion of level plots, that is a contour plot with each of the contour lines color coded by density. Instead, this level plot is essentially a rectangular binned frequency heat map except that we're using density instead of frequency to represent elevation. Using either frequency will produce an identical heat map though, so really, we should be referring to this as a density heat map. However, it's the closest thing I've found to a level plot in the base graphics system. To create this data visualization we'll use the image command, which is actually a low-level graphics command for programmatically creating images pixel by pixel, set the X-axis of our image to the x coordinate of our density function, set the y-axis to the y coordinates, set the Z-axis, which is the color component, to our z component, which is the density, set the color palette to the topographical color palette to 100 color levels, set our main chart title, set our X-axis label, and set our Y-axis label. As we can see this produces a level plot of density, or more specifically a density heat map for our two variables using the topographical color palette. We could have used a different color palette, like the heat map color palette, however, this palette seemed to work well for these data. Now we'll create a mesh plot of density. To do so we'll use the perspective command, set the X-axis to the x coordinates of our density function, set the Y-axis to the y coordinates, set the Z-axis to the z coordinates, that is the density, add a main chart title, an X-axis label, a Y-axis label, and we'll add a Z-axis label as well. As we can see, this creates a three-dimensional rendering of our density function with Runtime on the X-axis, Box Office Revenue on the Y-axis, and density of the Z-axis. To drape color on top of our mesh plot we'll use the same command as our mesh plot. However, first, we'll set the color palette to the topographical color palette with 100 color levels. Next, we'll execute the same command as we did for our mesh plot, however, we'll set the color parameter to the Z-axis component of our density. We use the cut command to divide the density values into 100 equal value intervals so that we can easily map them to our 100 colors in our color palette, then we'll reset our palette back to the default color palette. As we can see, this produces a surface plot where we've draped color representing the density on top of the surface of the plot. Using these four data visualizations we can show Tony how highly concentrated the movie runtimes are

around the 100 minute movie runtime. And that both runtime and box office revenue drop off sharply once we go too far above or below the 100 minute runtime. This might help convince Tony that trying to increase the average movie runtime in our streaming movies library might not have a positive impact on our revenue. To create our time series data visualizations, first, we need to load a time series data set containing the average box office revenue for each year. We'll do this by reading the Timeseries. csv file using the read. csv command and assign the resulting data frame to a variable called time series. Next, we'll take a peek at the time series data using the head command. As you can see, we have a column called Year containing the year a movie was released, and a column called Bow Office containing the average box office revenue for each corresponding year. For our first time series data visualization we'll create a step chart. To do so, we'll use the plot command, set the x parameter to our timeSeries data frame, set the type of our plot to s for step, set the Y-axis limit with the minimum Y-axis value equal to 0 and the maximum Y-axis value to the maximum box office revenue in our data frame. We do this so that our Y-axis starts at 0 in order to help us being misleading with our data visualization. We'll discuss ways to avoid information distortions and misleading data visualizations more in the last module or our course. We'll set our main chart title, set our X-axis label, and set our Y-axis label. As we can see, this creates a step chart of the average box office revenue by year. We can see that the average box office revenue has gone up and down over the years, but has stayed around 40 million dollars over this 15 year time period. Also, notice how the box office revenue shoots up to over 56 million dollars in 2015. Interestingly, this is simply because all of the movies haven't yet been added to our database for 2015 yet, so we have more of the years blockbuster movies and less of the box office failures for that year, which is dramatically throwing off our average annual revenue figures. This is why it's important to understand surrounding the data you're working with, which we'll discuss more in the last module, otherwise, you and Tony might have prematurely concluded that 2015 was going to be an off the charts year for movie revenue. The next time series data visualization we'll create is a line chart. To do so, we'll use the same command as we did for the step chart. However, we'll set the type parameter to l, for line. As we can see, this produces a line chart of average box office revenue by year. This data visualization is probably more appropriate for Tony than a step chart since it allows him to see rates of change over time. Regarding our final time series data visualization, unfortunately area charts are not easy to create in the base graphic system. However, they are easy to create in both lattice and ggplot2, so now let's switch over to create these same data visualizations in lattice, followed by ggplot2.

Demo (Lattice)

Now we'll learn how to create the same quantitative bivariate data visualizations in lattice. First, we'll create a scatter plot in lattice. To do so, we'll use the xyplot command, set our formula to box office revenue as function in movie runtime, set our data to our movies data frame, set our main chart title, our X-axis label, and our Y-axis label. This creates a scatter plot with Movie Runtime on the X-axis, and Box Office Revenue on the Y-axis. If we want to add a linear regression line to our scatter plot we can do so very easily with lattice. We'll use the same xyplot command as our scatter plot with all the same parameters, however, we'll set the type parameter to a vector containing the character p and r. The p stands for points indicating that we want to draw points for out plot character, and note that this the default for the xyplot, and the r stands for regression indicating that we want to add a linear regression line to plot as well. As we can see this creates the same scatter plot as before, however, we've now added a linear regression line to the plot. In addition, the plot has shifted the Y-axis upward to draw more of the linear regression line. Lattice takes care of constructing the underlying linear regression model for us so we don't need to worry about creating one when we're using lattice. To create a hexagonal binned frequency heat map in lattice we first need to load the hexbin package. We already did this once during our previous demo, however, just as a reminder that we need to load the hexbin package we'll do it once more. Lattice uses the hexbin package under the hood to create its hexbin data visualization. To do so, we use the hexbin plot command, set our formula to Box Office as a function of Runtime, set our data to the movies data frame, set the number of bins along the X-axis to 30, set tour main chart title, set our X-axis label, and set our Y-axis label. As we can see, this creates a hexagonal binned frequency heat map, like we saw in the previous demo, however, it has slightly aesthetics and the X and Y-axis labels no longer overlap with the axis value labels like we saw in the base graphic system demo. To create our contour plot, level plot, mesh plot, and surface plot in lattice we need to convert our two-dimensional kernel density estimation into a data frame. To do this we take the 2D kernel density estimation that we created using the kde2d command in the mass package and convert it into a data frame using the grid. expand command. We'll assign this data frame to a variable called grid, and we'll create a column called x containing the x coordinate of our 2D kernel density estimation function, create a column called y that contains the y coordinates, and we'll add a column called z, which contains the z coordinates, that is the density for each x and y coordinate. Just to clarify what we've done let's take a peek at the grid data frame. As we can see, we have one row for each combination of the x and y coordinate pairs and its corresponding z value. These rows will be ordered starting will all x coordinates for y coordinate until we've enumerated all combinations of x and y coordinate pairs. For example, since we had 50 rows and 50 columns in this two-dimensional grid of our density function, our new data frame will have 2500 rows since we had 50 x coordinates multiplied by 50

y coordinates for a total of 2500 x and y coordinate pairs. First, we'll create a contour plot using our new kernel density estimate data frame. To do so, we'll use the contour plot command, set our formula to z as a function of x and y. Note that we're using as asterisk to represent the and component of x and y in our formula. This formula tells lattice that we want to display our density that is z as a function of every combination of x and y. We'll set our data to our kernel density estimation data frame, set our main chart title, set our X-axis label, and set our Y-axis label. As we can see, this creates a contour plot of the density of our two variables, with our Movie Runtime variable on the X-axis, and Box Office Revenue on the Y-axis. Unfortunately, this data visualization isn't all that interesting in lattice using this combination of variables from our movies database. Next, we'll create a level plot. To do so, we'll use the same parameters as we did for our contour plot. However, we'll use the level plot command instead. As we can see, this creates a level plot of the density of our two variables. Now we'll create a mesh plot. To do so, we'll use the same parameters as our last two plots, however, we'll use the wireframe command we'll set the Z-axis label as well. As we can see, this creates a mesh plot of our data. Notice that this mesh plot is much easier to interpret than the mesh plot created in the base graphic system. Finally, we'll create a surface plot of density. To do so we'll use the same command and parameters as our mesh plot. However, we'll set the drape parameter to TRUE telling lattice to drape color on the surface of our mesh plot. As we can see, this creates a surface plot of our data using color to enhance the readability of the density on each location of the plot. Now we'll create our time series based data visualizations in lattice. First, we'll start by creating a step chart. To do so, we'll use the xyplot command, set our formula to Box Office as a function of year, set our data to our time series data frame that we loaded in our previous demo, set our plot type to s for step plot, set our Y-axis minimum limit to 0, and our Y-axis maximum limit to the maximum average box office revenue, set our main chart title, set our X-axis label, and set our Y-axis label. As we can see, this creates a step chart of average box office revenue by year. Next, we'll create a line chart. To do so, we'll use the same command and parameters as we did for our step plot. However, we'll set the plot type to l for line plot. As we can see, this creates a line chart of our average box office revenue by year. Finally, we'll create an area chart. In order to do so we'll first need to download, install, and load a package called lattice extra. It contains additional features that don't exist in the core lattice package, like area charts. To create an area chart we use the same command parameters as we did for the line chart. However, this time instead of setting the plot type to l, we'll set the panel parameter to panel. xyarea, which tells lattice to render as an area chart instead of a line chart. As we can see, this creates an area chart of average box office revenue by year. Now let's see how to create the same data visualizations in ggplot2.

Demo (ggplot2)

Finally, let's learn how to create the same quantitative bivariate data visualizations with ggplot2. We'll start by creating a scatter plot. To do so, we'll use the ggplot command, set the data to our movies data frame, set the aesthetics with x set equal to Runtime and Y set equal to Box Office, add a point geom, add a main chart title, add an X-axis label, and add a Y-axis label. As we can see, this creates a scatter plot with our Movie Runtime variable on the X-axis and our Box Office Revenue variable on the Y-axis. To add a linear regression line to our scatter plot we use the same command, parameters, and layers as our previous scatter plot. However, this time we'll add a smooth conditional mean geom, and we'll set the method to lm, which stands for linear model. as we can we this adds a linear regression line to our scatter plot. In addition, by default, ggplot2 adds shaded confidence intervals around the linear regression line. Unfortunately, confidence intervals are a statistical concept that's outside of the scope of this course. Next, we'll create a two-dimensional binned frequency heat map. To do so, we'll use the same command as our scatter plot, however, this time instead of adding a point geom we'll add a stat_bin2d statistic instead. As we can see, this creates a heat map of the two-dimensional frequency of observations using a rectangular grid of bins. If we prefer to create a hexagonal bin frequency heat map we use the same command as our rectangular bin frequency heat map, however, this time we add a stat_binhex statistic instead of a stat_bin2d statistic. As we can see, this create our heat map with hexagonal bins instead of rectangular bins. To create a contour plot of density we use the same command as before, however, this time we use a density2d geom instead. As we can see, this creates a contour plot of the density of our two variables with our Movie Runtime variable on the X-axis and our Box Office Revenue variable on the Y-axis. To create a level plot of density we use the same command as our previous data visualizations, however, this time, we use the stat_density2d statistic instead, setting the aesthetics with the fill equal to the computed level of density, that is the incremental elevation of the density and the geom set to polygon. Note that we use the two dots before and after the word level when setting our fill parameter. The two dots before and after is a special notation in ggplot2 that the variable is not present in the original data set, but has been computed by the statistic, which in this case is our density2d statistic. As we can see, this creates a level plot of the density of our two variables. I should note that this is just one of many ways to create contour and level plots in ggplot2. There are other combinations of geoms and statistics that could be used to create the same result. This is actually the case with many of the data visualizations in ggplot2. It's so flexible that there's often multiple ways to create the same data visualizations. Unfortunately, we can't create either our mesh plot or our surface plot in ggplot2 because 3d data visualizations do not currently exist in ggplot2. We would

need to create these data visualizations using either the base plotting system or lattice, like we saw in the previous two demos. Now let's switch gears and learn how to create time series data visualizations in ggplot2. First, we'll create a step plot. To do so, we'll use the ggplot command, set our data to our time series data frame, set our aesthetics with x set equal to Year, and y set equal to Box Office, add a step geom, expand our Y-axis limit so that our Y-axis begins at 0, which we're doing again to avoid being misleading, add a main chart title, add an X-axis label, and add a Y-axis label. As we can see, this creates a step chart of the average box office revenue by year. To create a line chart we use the same command, parameters, and layers as our previous command. however, this time we'll use a line geom instead. As we can see, this creates a line chart of the average box office revenue by year. Finally, to create an area chart we use the same command, parameters, and layers as our two previous commands, however, we use an area geom instead. As we can see, this creates an area chart of the average box office revenue by year. Notice how easy it is to create these more advanced data visualizations in ggplot2 and lattice. We're generally using a relatively consistent set of commands, parameters, and layers. Typically, we're only changing one or two things across each data visualization. This flexibility, consistency, and simplicity make both lattice and ggplot2 excellent tools for creating more advanced data visualizations. We show our data visualizations to Tony and help him answer his questions about the relationship between movie runtime and box office revenue, and the change the in the average box office revenue from year to year. He was quite impressed, especially with the 3D data visualizations that were able to create. This gives him some ideas for additional questions he'd like to answer with our data. However, we're going to have to learn some data visualizations before we can help him answer his remaining questions.

Summary

In this module, first, we learned about data visualizations for quantitative bivariate analysis, that is the analysis of the relationship between two numeric variables. Next, we learned how to create these data visualizations using the base graphic system. Then, we learned how to create these data visualizations using lattice. Finally, we created these data visualizations using ggplot2. In the next module we'll learn how to create and interpret data visualizations for a numeric variable grouped by a categorical variable.

Visualizing Both a Categorical and a Numeric Variable

Introduction

Welcome back to Data Visualizations with R. I'm Matthew Renze with Pluralsight, and in this module we'll learn how create and interpret data visualizations for both a categorical variable and a numeric variable using R. As an overview of this module, first, we'll learn about data visualizations for bivariate analysis of both a qualitative and quantitative variable, that is the analysis of a numeric variable grouped by a categorical variable. Next, we'll learn how to create these data visualizations using the base graphic system. Then, we'll learn how to create these same data visualizations using lattice. Finally, we'll create these data visualizations using ggplot2. To help orient yourselves in relation to the rest of the course, this module will cover bivariate analysis of both a qualitative and a quantitative variable, that is the analysis of a numeric variable grouped by a categorical variable. In addition, we'll be wrapping up our study of the various types of univariate and bivariate analysis, and preparing to learn about more advanced types of data analysis that go beyond these five basic types.

Qualitative and Quantitative Bivariate Analysis

The third type of bivariate analysis that we can perform is bivariate analysis for both a qualitative and a quantitative variable. That is, the analysis of a numeric variable grouped by a categorical variable. When we're performing this type of bivariate analysis we're comparing aggregate values of the numeric variables for our observations grouped by categories. Let's break this down into simpler terms. First, we're typically looking at an aggregate value of a numeric variable, this could be the sum, mean, median, quartiles, standard deviation, or some other aggregate measures. Second, these values are grouped by categories. This means that when we're aggregating the numeric values of our observation we're doing so within each category as a group. Finally, we're typically interested in answering questions that involve a comparison of these aggregate values across each category. For example, in this bar chart we're interested in comparing the total box office revenue by rating category. In this example, our aggregate measure is the sum, that is, the total box office revenue. Our groups are our rating categories, and we're comparing the total box

office revenue from one rating category to the next. Now, let's learn about a few data visualizations that we can use for answering questions involving this type data analysis. First, as we saw in previous slide, we can use a bar chart to perform this type of analysis. On the X-axis we have our qualitative, that is categorical variable, for example, movie rating categories. On the Y-axis we have our quantitative, that is our numeric variable, for example, box office revenue in millions of US dollars. The height of each bar corresponds to the value of the aggregate measure of our numeric variable. In this case, it's the average movie runtime. This type of data visualization works well for answering questions that involve the comparison of a single aggregate measure. For example, is the average box office revenue for PG-rated movies larger than the average box office revenue for R-rated movies? As we can see from this bar chart, the average box office revenue for PG movies is significantly larger than it is for R-rated movies. We can essentially choose any aggregate measure, whether is be some, average, standard deviation, etc., for our bar chart that can be compared in an apples to apples fashion. However, with this data visualization we can only focus on a single aggregate measure at once. To focus on multiple aggregate measures at the same time we have to look at additional data visualizations. The second type of data visualization is a bivariate box plot. On the X-axis we have our qualitative, that is, our categorical variable, for example, movie rating categories. On the Y-axis we have our quantitative, that is our numeric variable, for example, box office revenue in millions of US dollars. In the plot area we have the box and whiskers for each of our categories on the X-axis. As you recall from our previous module, the box and whiskers provide us with a visual representation of the five-figure summary statistics for our numeric variable. The five figures are the minimum value, excluding outliers, the first, or lower quartile, the median, the third, or upper quartile, the maximum value excluding any outliers, and any outliers, that is values greater than 1. 5 times the interquartile range added to the upper quartile, or less than 1. 5 times the interquartile range subtracted from the lower quartile. We can use bivariate box plots to answer questions involving a comparison of the location of our data across each category, a comparison of the spread of the data across each category, and a comparison of the outliers across each category. There's a modified version of a box plot that can be used to perform an additional type of visual comparison, called a notched box plot. The difference is that this box plot has notches around the median value of each box plot. The notches represent the confidence interval around the median. In this case, the notches are representing a 95% confidence interval around the median. While I can't go too deep into confidence intervals or hypothesis testing in this course, in a nutshell, what the notches tell us is that if the notches do not visually overlap then there's strong evidence that the medians of the two categories differ significantly, in the sense of statistical significance. If the notches do however, overlap, then the difference in the median if likely do to random chance at a

95% degree of confidence. Please note, however, that this is not a formal statistical test, so we should only use these notches for visual inspection, we would want to run a more rigorous statistical test and compare the numeric results to know if the medians actually differ significantly, or not. The final data visualization for this section of bivariate data analysis is a bivariate violin plot. On the X-axis we have our qualitative, that is categorical variable, for example movie rating categories. on the Y-axis we have our quantitative, that is our numeric variable, for example, box office revenue in millions of US dollars. In the plot area, we have a visual representation of the shape of the density of the distribution of numeric values drawn uniformly around the center of each category on the X-axis. This is done using a kernel density estimation like we saw in our density plot. However, this time, we're creating a density function for each category and the density function is drawn uniformly around the center of each categories access line. A bivariate violin plot allows us to answer questions that involve a comparison of the shape of the distribution of a numeric variable across each category. For example, we can quickly see that the peak of the distribution of R-rated movies is significantly sharper than that of G-rated movies. In addition, we can see that all of our distributions are positive skewed, that is, the tail on the right, or in this case, the top, is much longer than the tail on the left, or bottom in this case. There are other types of data visualizations that we could create to analyze both the qualitative and quantitative variable, however, these four should cover the basics. Now let's see how to create these four types of data visualizations using the three plotting systems in R.

Demo (Base)

Tony says that he has one more set of questions that he needs help with for his annual sales and marketing plan. First, he's interested in knowing how the average box office revenues compare across each movie rating category. He can use this information to help him determine the rating category to focus on to maximize revenue. Second, he's also interested in knowing how similar are the spreads of the box office revenues across each movie rating category. He wants to know if one or more categories has higher revenue simply because it has a few blockbuster outliers. In addition, he'd like see if the median values differ in a statistically significant way, or just to random chance. Finally, he wants to compare the shapes of the distributions across rating categories to see if there's anything that looks odd or interesting about them that might warrant further analysis. So, let's help Tony answer his final set of questions using the data visualizations we just learned about. For our first bivariate data visualization for both a qualitative and quantitative variable, let's begin by creating a bivariate bar chart. To do so, first we'll create an array containing the average box office revenue by each rating category. We'll do this using the tapply

function, which allows us to apply a function to a numeric variable grouped by a categorical variable. For example, in our case, we want to apply the mean function to box office revenue grouped by each movie rating category. Just to clarify the two-dimensional array of values that we've just created we'll print out this array. As we can see, we have a two-dimensional array containing movie rating categories in the first row, and the average box office revenue for each rating category in the second row. Now, use the barplot command, set the height parameter of our array to the average box office revenue group by rating category, set a main chart title, set an X-axis label, and set a Y-axis label. As we can see, this creates a bivariate bar chart of the average box office revenue in millions of US dollars for each movie rating category. Tony can use this chart to answer his questions about how the average box office revenue compares across categories. Next, we'll create a bivariate box plot. To do so, we'll use the plot command, set the X-axis to our Rating variable, set the Y-axis to our Box Office Revenue variable, set a main chart title, set an X-axis label, and set a Y-axis label. As we can see, this creates a bivariate box plot of our two variables. Tony can use this chart to answer his questions about how the location and spread of the data compare across each category. He can also see that there are significantly more outliners in the PG-13 category compared to the other categories, especially the G-rated movie category. This might be useful information for him to know as well. Now let's create a notched box plot. To do so, we use the same command and parameters that we used in our bivariate box plot. However, this time, we'll set the notch parameter to TRUE. As we can see, this creates a notched box plot of the same data. If we look at the overlap in the notches, we can see that the median box office revenues for G, PG, and PG-13-rated movies all overlap. This means that the difference in the median box office revenue of the first three categories is likely due to random chance. However, the notch in the R-rated category does not overlap with any of the first three categories. This tells us that the difference in the median box office revenue is likely not due to random chance, but rather some other factor. This might be valuable information for Tony's analysis as well. There's no violin plot available in the base graphic system of R, however, there are some extension packages, like vioplot, that can produce these visualizations. Instead of using one of these extension packages we'll just move on to creating these same data visualizations with lattice and ggplot2 to learn how to create violin plots using those graphics packages.

Demo (Lattice)

Now let's learn how to create the same data visualizations for both a qualitative and quantitative variable in lattice. We'll start again with the bivariate bar chart. To create a bivariate bar chart we'll need a table containing our average box office revenue by rating category. However, instead of

using the tapply function this time, we're going to take a different approach to creating this table. We're going to use a package called dplyr, which provides several features for slicing, dicing, and aggregating data. It's one of the most popular packages for manipulating data, so I thought it'd be valuable to see it in action. We won't go too deep into how it works, I just want to give you a bit of exposure to it so that you can see how powerful it is. To create our table, first, we need to download, install, and load the dplyr package. Now, we'll take our movies data frame, select two columns from the data frame, that is Rating, and Box Office, group the values by rating, summarize the values using the arithmetic mean function applied to the box office revenue, convert the resulting data into a data frame, and we'll assign this data frame to a variable called average, which will override our previous variable called average. We'll print this data frame so we can see what we've created. As you can see, this created essentially the same result as when we use the tapply command, however, the result is a data frame, instead of an array. And ratings and average box office revenue is populated in columnar fashion. While the apply functions like tapply work well for applying functions across data in a relatively narrow set of ways, dplyr allows significantly more power and flexibility in manipulating data. So, at some point in time I highly recommend that you learn how to use dplyr, it will make many of your data manipulation tasks with R much easier. Now we can create our bivariate bar chart using the data frame we just created. To do so, we'll use the bar chart command, set our formula to Box Office as a function of Rating, set our data to the data frame we just created, set our main chart title, set our X-axis label, and set our Y-axis label. As we can see, this creates a bivariate bar chart of average box office revenue grouped by movie rating category. To create a bivariate box plot we'll use the same parameters of our bar chart, however, we'll use the bwplot command, that is, the box and whiskers plot command, instead. As we can see, this creates a box plot of our two variables. To create a notched box plot we'll use the same command and parameters as our bivariate box plot. However, this time we'll set the notch parameter to TRUE instead. As we can see, this adds notches to our bivariate box plot. Finally, we'll create a violin plot using lattice. To do so, we'll use the same command parameters as our bivariate box plot, however, this time we'll set the panel parameter to panel. violin instead. As we can see, this produces a violin plot of the same data. Notice that lattice renders these distributions below the 0 line on the Y-axis. However, we won't see the same thing with violin plots in ggplot2, so let's wrap things up by creating the same data visualizations with ggplot2.

Demo (ggplot2)

Finally, let's learn how to create the same qualitative and quantitative bivariate data visualizations in ggplot2. First, we'll create a bivariate bar chart. To do so, we'll use the ggplot command, set our data to our data frame containing average box office revenue grouped by rating category from our previous demo, set our aesthetics with x set to rating and y set to box office, add a bar geom with the statistical transformation set to identity, which tells ggplot2 to use the value from our Y-axis variable, as is without statistical transformation, add our main chart title, add our X-axis label, and add our Y-axis label. As we can see, this creates a bivariate bar chart of the average box office revenue grouped by rating category. Next, we'll create a bivariate box plot. To do so, we'll use the same command, parameters, and layers as our bar plot command, however, this time we'll add a box plot geom instead of the bar geom. As we can see, this create a bivariate box plot of our two variables. To create a notched box plot, we use the same command, parameters, and layers as our bivariate box plot. However, this time, we'll set the notch parameter equal to TRUE. As we can see, this creates a notched box plot of our data. Finally, we'll create a violin plot. To do so, we use the same command, parameters, and layers as our previous data visualizations. However, this time, we'll use the violin geom instead. As we can see, this creates a violin plot of the density of our distribution of data grouped by movie rating categories. Notice how unlike lattice, ggplot2's violin plot abruptly stops drawing the distribution at the 0 line on the Y-axis. So, you need to make a decision as to the whether you want your data visualization to render the distribution smoothly down to a density of 0 below the Y-axis line, like lattice does, or to cut off abruptly at 0 Y-axis line, like ggplot2 does. I personally prefer the way ggplot2 renders violin plots as it's less likely to misrepresent our data. But, we can leave it up to you and Tony to decide which version you both prefer. So now, Tony can use this plot to compare the shape of the density of the distribution across each rating category. For example, we can easily see that the R-rated movie distribution in much more sharply peaked than the rest of the distributions. This could provide useful insight for Tony during his analysis. Tony uses our data visualizations to create his annual sales and marketing plan. He thanks us for our help, and promises to put in a good word for us when he presents these data visualizations during his annual sales and marketing presentation to the board of directors.

Summary

In this module, first, we learned about data visualizations for bivariate analysis of both a qualitative and a quantitative variable, that is the analysis a numeric variable grouped by a categorical variable. Next, we learned how to create these data visualizations using the base graphic system in R. Then, we learned how to create these same data visualizations using lattice.

Finally, we created these data visualizations using ggplot2. In the next module we'll move beyond the basics and learn about more advanced types of data visualization, best practices for creating professional data visualizations, and we'll also look at a few alternatives to using R to create data visualizations.

Moving Beyond the Basics

Introduction

Hello again, and welcome to our final module on Data Visualization with R. I'm Matthew Renze with Pluralsight, and in this module we'll move beyond the basic concepts of data visualization and take a look at a few advanced topics. As an overview of this module, first, we'll learn about a few more types of advanced data analysis in data visualizations. Next, we'll learn about best practices for creating professional data visualizations, then we'll see a demo of how to produce clean, professional data visualizations, next we'll learn about a few alternatives to using R to create our data visualizations, finally, we'll wrap up this module and the course as a whole.

Advanced Data Visualization

I really want to emphasize that what I've shown you so far in this course is just the tip of the iceberg. I can stress how much more R provides beyond what we've seen in terms of data visualizations capabilities. There's significantly more types of data analysis that we can perform and data visualizations that can be created as well. I would, however, like to give you a bit of exposure to these advanced types of data analysis and their corresponding data visualizations for a few reasons. First, I think it's important to see the bigger picture so you'll know what other types of data analysis and data visualizations exist beyond what we've already seen. Second, I want to help motivate you to continue learning about more advanced types of data visualizations beyond what we've already learned in this course. And finally, I have two additional course at Pluralsight on intermediate and advanced data visualizations with R. So this next section will act as a preview for what you'll learn if you decide to take either of those two courses as well. Please note that at the time this course was released, the intermediate and advanced data visualizations courses were still in development so they may not be part of the Pluralsight catalog just yet. However, if they aren't currently available, they will be in the near future. So now, let's quickly run through some additional types of more advanced data analysis and their corresponding data

visualizations. In this course we learn about the five types of data analysis contained within univariate analysis and bivariate analysis for categorical and numeric variables. However, beyond this, we have several other types of data analysis. Beyond univariate and bivariate analysis we have trivariate analysis, which as you've probably already guessed, is the analysis of the three variables. If we have three qualitative, that is categorical variables, we can perform qualitative trivariate analysis. If we have two qualitative and one quantitative variable, we have a type of trivariate analysis for that combination as well. If we have one qualitative and two quantitative variables, we also have a type of trivariate analysis for that combination. And if we have three quantitative, that is numeric variables, we can perform quantitative trivariate analysis. Beyond three variables we have the remainder of multivariate analysis, which just means the analysis of many variables. We have special types of data visualization and analysis that work well for dealing with an arbitrary number of variables. Finally, we have a few special types of data analysis that don't really fit anywhere else. If we have a set of variables that include a position on the map, typically expresses a longitude and latitude, we can perform spatial analysis. If we have a set of variables that contain a parent/child relationship we can perform hierarchical analysis. If we have a set of variables that contain relationships between the entities we can perform graph or network analysis. And, if we have a body of text as our data source we can perform textual analysis. In addition, we can also create both animated and interactive versions of all these types of data visualizations as well. In this course, that is Beginning Data Visualization with R, we covered creating and interpreting data visualizations for univariate and bivariate analysis. The intermediate course will cover trivariate and multivariate analysis, and the advanced course will cover all the special types of data analysis, including animated and interactive data visualizations. We'll begin the intermediate course with trivariate data analysis by learning how to visualize three categorical variables using facetted clustered frequency bar charts, faceted stacked frequency bar charts, and facetted frequency bar charts. Next, we'll learn how to visualize two categorical and one numeric variable using clustered bar charts, stacked bar charts, facetted bar charts, and heat maps. Then, we'll learn how to visualize one categorical and two numeric variables using discreet color scatter plots, shape coded scatter plots, facetted scatter plots, multi-series line charts, stacked area charts, and facetted line charts. Next, we'll wrap up trivariate data analysis by learning how to visualize three numeric variables using gradient color scatter plots, divergent color scatter plots, and bubble charts. Finally, we'll end the second course with multivariate data analysis by learning how visualize an arbitrary number of variables with tools like a scatter plot matrix. We'll begin the advanced data visualization course with spatial data analysis by learning how to visualize spatial data using dot maps, density maps, bubble maps, and _____ plus. Next, we'll learn about hierarchical analysis by visualizing hierarchical data using qualitative tree maps

and quantitative tree maps. Then, we'll learn about graph and network analysis by visualizing graph and network data for small graphs, medium-size graphs, and large graphs. Next, we'll learn about textual data analysis by visualizing textual data using frequency word clouds, word clouds sized by numeric variables, and word clouds using two variables. Then, we'll learn how to create animated data visualizations to help us tell the story of our data over time. Finally, we'll wrap up the advanced course and the three-part series on data visualizations with R by building interactive data visualizations so that our users can answer their own questions with our data visualizations. If you found this course valuable and you want to continue learning how to create an interpret more advanced types of data visualizations with R, please be sure to check out these other two courses in the series.

Best Practices

Now let's spend a few minutes discussing best practices for creating professional data visualizations. In addition, we want to learn how to avoid making errors and invalid claims with our data visualizations. This is a very important topic because it's very easy to unintentionally mislead or misrepresent data when we create our data visualizations. These best practices are general tips that are advisable in most contexts. However, context is very important so there may be some specific contexts where this advice is not applicable. So, it's important that you learn the general rules and learn when it's appropriate to follow them and when it's advisable to do something else. First, we should start each data visualization with a question that we're attempting to answer. For example, our question might be, how many movies were released in each rating category from the year 2000 to 2015? Once we have our question in mind we can attempt to answer that question using a data visualization. In this case, our question involves a comparison of the frequency of observations for a qualitative variable, that is movie ratings. So, we can create a bar graph of the number of movies by rating category to visualize the answer to our question. Starting with a question in mind helps us to focus on a specific goal, and helps us to choose a data visualization that's appropriate to answer the question. Second, it's important that we use the right tool for the job. This starts with understanding the question that we're trying to answer for the audience. Ask yourself, what information is the audience trying to learn from our visualization? Are they trying to compare two values to see which is larger? Are they trying to see rates of change over time? Or are they trying to see how values are distributed across a variable? For example, if our audience is interested in comparing movie rating categories a pie chart may not be the best way to present this information. However, a bar chart is an excellent tool to communicate this information for comparison purposes. Once we know the question our

audience is asking we then need to know which type of data visualization communicates the answer in the most effective way for our audience. This requires learning which types of data visualizations are most appropriate to answer types of questions in specific contexts. Third, it's important that we know our audience. We should always consider who will be viewing our data visualizations, and then create our data visualizations accordingly. We could be producing our data visualization for a small audience who is very familiar with the data and context, or we could be producing our data visualization for a large audience who is unfamiliar with the data or context. We should invest time and effort in proportion to the size of the audience, their familiarity with the data and domain, and based on the potential impact of decisions that will be made using this data visualization. Data visualizations created for small audiences for exploratory purposes are typically referred to as exploratory data visualizations, whereas data visualizations created for explanatory purposes for a wide audience are typically referred to expository, or explanatory data visualizations. If we're performing exploratory data analysis we're typically creating our initial data visualizations for an audience of one, that is, we're creating our data visualizations for ourself during the data exploration process. We typically do this in a very quick and dirty way because we're only concerned about learning from our data exploration, not communicating information to a wider audience. In addition, we typically have a deep understanding of the data and the context, so we don't need to clean up the data visualizations and create human-readable titles, labels, etc. The out of the box defaults are typically good enough four our data exploration process. However, if we're planning to create data visualizations to be shown to a wide audience we would need to spend extra time and effort cleaning up the data visualizations, adding human-readable titles, labels, legends, etc. We'll cover some of the basics of creating high-quality expository data visualizations during our upcoming demo. Fourth, we want our data visualizations to be learn and simple. Our goal is to maximize the data-to-ink ratio of our data visualizations, that is, we want to communicate as much information as possible using the minimum amount of digital ink. This means that we should strive to eliminate all chart junk as data visualization expert Edward Tufty likes to call it, that is, anything our chart that doesn't directly help communicate the core information of our data visualization. For example, here's a data visualizations that contains a tremendous amount of chart junk. It is a 3D graph, but the third dimension communicates no relevant information. It has each bar color-coded, and a legend, but this is redundant information since each bar has an X-axis label. Each bar also has unnecessary special effects, like rounded corners, surface reflectivity, and drop shadows. It has a base texture on the main canvas, and the back wall of the bar graph has a granite texture. It uses multiple fonts that are difficult to read, are heavily bolded, and uses 100% black text. In addition, we have thick, solid black Y-axis grid lines that add even more clutter to this already heavily-

cluttered graph. Instead, we could communicate the exact same information with a much simpler graph, like this. We've eliminated the unnecessary 3D bars, special effects, and backgrounds. We've eliminated the redundant color-coding of the categories and the legend. We've cleaned up the fonts, removed the bolding, and lightened up the text. And we've eliminated the Y-axis gridlines. However, both of these charts communicate the exact same information, the second chart is just doing so in a way that is much more efficient and effective. When we're creating data visualizations we want to keep things clean and simple, oftentimes, less is more. Finally, we want to avoid information distortions in our data visualizations. It's very easy to lie with data, whether these lies are intention or unintentional. In fact, there's so many ways to visually misrepresent data using data visualizations that entire books have been written on the subject. For example, let's say we're interested in comparing the total number of PG-13 movies to R-rated movies. We could create a graph where the Y-axis begins at an arbitrary location, for example, 1, 200 movies. This makes it appear as if there's a significantly larger number of R-rated movies than PG-13 movies. However, if we create the same graph with the Y-axis starting at 0, these two values don't look all different after all. Simply by changing the start and end of our Y-axis we can unintentionally distort the reality of our data. Other examples of information distortion involve using exaggerated 3D perspectives, using the size of shapes rather than their area for comparison, and breaking established data visualization conventions. It's important that we learn how someone can lie with data visualizations in order to avoid accidentally lying with our data visualizations. In addition, it's a very important skill for recognizing when we're being lied to by someone else's data visualizations. Now, let's see a quick demo of how to apply these best practices in R.

Demo

Now let's learn how to incorporate a few of these best practices with our data visualizations in R. Tony stops by our office an tells us that his presentation with the board of directors went extremely well, so well, in fact, that they've asked him if they can include of few of our data visualizations in the presentation for the annual shareholders meeting. He asks us if we could prepare a few of these data visualizations to be presented to a wider audience. So, we agree to help Tony, and the board of directors, create these more polished data visualizations for their annual shareholders meeting. We're just going to focus on a single data visualization for the last demo in our course. We began the data visualizations with the question, how many movies were released in each rating category from 2000 to 2015. Starting with the question is the first step in the process of creating professional data visualizations. Once we understand the question we can

then choose the right tool for the job, which is our second step. Given the question, we decide that frequency bar chart is the best data visualization to answer this question. So, we'll create a frequency bar chart of our movie rating categories using the default parameters. We'll do this with a plot command, passing in movie ratings as only parameter. As we can see, this creates a frequency bar chart with movie rating categories in the X-axis, and the frequency of observations on the Y-axis. The audience for this data visualization are the shareholders of NetFilms. Given that they are a wide, general audience, we can't assume that they are familiar with the internal business terminology of NetFilms. In addition, since they're not intimately familiar with these data, we need to provide sufficient context for them to understand this data visualization without investing much time or effort. We can do this by providing very clear context around the data. Information is data with context. So, by providing context we turn these raw data into information for the shareholders. To do so, we'll add a main chart title, something very clear, like Count of Movies by MPAA Rating Category. The acronym MPAA, which stands for Motion Picture of America Association, might be well-known to the shareholders, so we might need to clarify that in the body of text surrounding the data visualization in a footnote, or in a caption for the data visualization. In addition, we might want to be more specific about the context of these data by clarifying that the movies contained in this data visualization are movies released in the US from the year 2000 to summer of 2015 with an MPAA rating of G, PG, PG-13, or R, and with a reported box office revenue. This is obviously too much information for the title itself, so this might be more appropriate in the body of the document in the caption, or in a footnote. Let's also create a very clear X-axis label like MPAA rating category, and a clear Y-axis label, like count of movies. As we can see, this adds labels to our data visualization which provide the audience with the context that they need to clearly understand what these data represent. Also, notice how I didn't say anything about frequency or observations in any of the chart labels. These terms are statistical jargon and not appropriate for the audience in question. However, if we were presenting these data visualizations for statisticians or data scientists, this terminology may be more appropriate. The next step is to clean up our data visualization to prepare it for publication. Luckily for us, our mix is quite easy as the defaults generally produce relatively clean data visualizations with reasonable data-to-ink ratios. However, we're going to add a bit color to our data visualization at the request of the board of directors, and remove some unnecessary ink from our data visualization. First, let's create a color palette using the Color Brewer extension package. To do so, we'll need to download, install, and load the Color Brewer extension package. Color Brewer was created to make it easy to create color palettes that are appropriate for a wide variety of data visualizations and maps. These color palettes are extremely well designed. They're safe for viewing on a wide range of print and digital mediums, and they solve several problems that are

found with arbitrarily chosen color palettes, like being accessible for individuals with certain types of color blindness. Essentially, they take the guesswork out of creating professional color palettes for data visualization. In fact, my recommendation is to use Color Brewer anytime you need to use color in your professional data visualizations. I highly recommend you invest time learning how to use the various color palettes, and the theory behind the design of the palettes in Color Brewer. Once we have the Color Brewer extension package loaded, we can easily create one of many types of color palettes. We're just going to create the default pastel color palette with nine colors, even though we're only going to using one of these colors for our data visualization. Now, we'll recreate the same bar chart, however, this time we'll add the second color from our pastel color palette, which is a light blue. Next, we'll increase the data-to-ink ratio by removing the borders around each of the bars. Since we filled our bars with color, the borders are now unnecessary, and thus their presence decreases our data-to-ink ratio. As we can see, this added a light-blue fill color to our bar chart and removed the border around each of the bars. We could also spend more time adjusting the fonts, numeric formats, grid lines, access labels, etc., however, I think this gives us a basic idea for how to clean up our data visualizations. The rest we can learn from the Help files as needed. Next, we want to check our data visualizations for any information distortions. We're using a flat 2D bar chart, so we don't have any information distortion from improperly using a 3D perspective. In addition, our bars are equal with rectangles rather than scaled, real-world shapes, or geometric objects, which might lead to other types of information distortion. Also, by default, the bar chart in the basic graphic system started our Y-axis origin at 0 so we don't have any potential information distortion due to an offset Y-axis origin. Essentially, it looks like data visualization is properly representing our data, so, we can proceed to the next step. Finally, we want to export our data visualization to a file so that it can be embedded in the document for our shareholders. To do this, there's several commands in R that allow us to export data visualizations to various file formats. For example, we can export our data visualizations into various image and document formats, such as Bitmap, PNG, JPEG, TIFF, SVG, or PDF. It's important to learn about the pros and cons of each of these types of document and image formats so that you know in what context they work better or worse. However, in general, if we want to use our data visualizations as an image on the web we should save it as a PNG, that is portable graphics file. To save it as a printable document we should generally export it as a PDF, that is Adobes portable document format. Or, to export it as a scalable vector graphic we should export it as an SVG, that is scalable vector graphics file. In our case, we're going to export our data visualization as a PNG file. To do so, first we need to create a PNG graphics device using the png command. Set the file name to Count of Movies by Rating. png, set the width of the image to 640 pixels, and set the height of the image to 480 pixels. Now, we render the plot into the PGN

graphics device we created by executing our plot command with the same parameters. Finally, we use the dev. off command to close the graphics device. Now, we can navigate to our working directory in our file, which we originally set to our Pluralsight directory on the C drive in our first demo. And now, we can open and view our newly created PNG file. In addition, RStudio has a built-in data visualization export feature as well. To use it, we simply click the Export button at the top of the Plot window, select Save as Image, specify the image format, which in our case we'll choose PNG, set the directory, which we'll set to our Pluralsight directory, specify a file name, which in our case we'll use Count of Movies by Rating 2. png, specify our width, which we'll set to 640 pixels, specify our height, which we'll set to 480 pixels, we'll check the View plot after saving checkbox so that we can see the data visualization that we've exported, and we'll click the Save button. As we can see, this exported our data visualization to a PNG file with the specified parameters. There's also a command built into ggplot2 to export data visualizations. However, rather than going into the details of how to use it, I'm just going to point you to the Help file instead. To view the Help file, just type? ggsave and press Enter. This will load the Help file for the ggsave command. Finally, if you have any questions about any of the plotting commands, and their parameters, I'd like to remind you again to look at the Help file for these commands. You can view the Help files for a specific plot command by typing? plus the name of the plot command and pressing Enter. It will explain the command and list all of the applicable parameters. For more generic plotting parameters, just type? par for graphical parameters and press Enter. This Help file will provide you with detailed information on all of the generic plotting parameters. Getting to know these plotting commands and parameters very well will help you tremendously when you're trying to fine-tune your data visualizations for publication. We finished creating our production-quality data visualizations and they are presented at our annual shareholders meeting. There are even a few comments made about how much easier it was to understand the information that was presented this year because the data visualizations were so clean and easy to understand. In fact, one of the presenting board members approaches us after the meeting to thank us for all or our help. She also tells that there's going to be great opportunities at NetFilms for individuals with the ability to work with and communicate actual insight using data because the direction they're taking the company. However, we're going to have to learn about more advanced types of data visualizations if a future course to find out what happens next with our career at NetFilms.

Alternatives to R

Now that we've seen how to use R to created professional data visualizations, I think it's important that we take a few minutes to look at a few alternative tools for creating data

visualizations. We'll take a look at these alternatives in order from least difficult to learn to most difficult to learn, based on my own personal experience with them. First, spreadsheets like Microsoft Excel are still the most popular choice in the business world for creating quick data visualizations. Creating data visualizations in spreadsheet software is quick and easy, and they're relatively customizable as well. If you'd like to give Microsoft Excel a try, you can download a trial copy at www. office. com. Next, we have interactive data visualizations tools like my open source project called Data Explorer. It's highly visual, highly interactive, and it's very easy to use, even for users with little to no experience with data visualization. Best of all, it's free, open source software, so you can use it without cost. If this looks interesting to you, you can download a free copy of Data Explorer at www. data-explorer. com. Another step up in complexity we have more advanced interactive data visualization tools, like Tableau. This is like my open source Data Explorer, but with much more advanced data visualizations and analysis features. However, Tableau software is commercial software, that is, it's not free, open source software, and it takes a bit of time to get up and running. You can give Tableau a try by visiting www. tableau. com. Moving further up in complexity we have statistical analysis software with rich user interfaces like SPSS. These provide many of the same data visualization features as R, but in a much more user-friendly interface. Choosing a program like SPSS over R has a lot to do with personal preference and whether you enjoy working directly with programming languages, or prefer to use a graphical user interface instead. You can download a trial copy of SPSS at www. ibm. com/software/analytics/spss. In addition to R, we also have other statistical programming languages like SAS. They are similar in power and complexity to R, but unlike R they are not as popular as R, and most are not free like R. Languages like SAS, however, are still heavily used in the business world for data analysis. If you'd like to learn more about SAS, please visit www. sas. com. And finally, you can use other general purpose programming languages like C#, Python, and JavaScript, along with graphics in charting libraries to create all of the same data visualizations as R. R, however, was designed for data analysis, so while you may gain flexibility with general purpose programming languages, it will often take you considerably more time and effort to create the equivalent data visualizations. I typically use R for day to day data visualizations. However, as a software developer I also frequently use general purpose programming languages with charting or graphics libraries to embed data visualization capabilities into the applications that I create for my clients. For example, I do quite a bit of consulting work creating interactive data visualizations for the web using JavaScript with a d3 data visualization library. In addition, as we've seen with my open source Data Explorer project, I've also created data visualization software without third party charting libraries, that is, by drawing the plots directly on the screen using low-level graphics routines. Once again, it's important to choose the right tool for the job,

and the first step is becoming familiar with all of the tools so that you know all of their strengths and weaknesses. Now, let's wrap things up for this module and for the course as a whole.

Course Summary

Now let's wrap things up for this module and for the course as a whole. If you'd like to learn more about the topics we discussed in this course I have links to a few resources you might be interested in. First, I recommend that you head to the R website and download a free copy of R. This way, you'll have it installed and ready to go when you're ready to start creating data visualizations. Next, head to the RStudio website and download a free copy of our RStudio. With RStudio you'll get up and running much quicker than by using R out of the box. If you're interested in additional online training, Pluralsight has other courses on both R, and data analysis available. So be sure to check our my other courses on data analysis with R, as well as the other R courses at Pluralsight as well. Coursera has a data science specialization track taught by professors at Johns Hopkins University that provides in depth coverage of using R for many different types of data analysis and data visualization. Nathan Yau's Flowing Data website is an excellent source for learning about data visualization with R. Finally, my website contains quite a bit of useful information on R, data visualization, and data science topics as well. Finally, one last thing before we wrap things up for this course, feedback is very important to me. I use your feedback to improve each and every one of my courses. So, please be sure to take a moment and rate this course, ask questions in the discussion board if there's something about this course that you didn't understand or would like me to clarify, leave comments to let me know what you liked about this course, or if you think there's something I could do to improve upon a course in the future. And, feel free to send me a Tweet on Twitter if you liked this course and would like to provide me with feedback in public as well. My Twitter handle is @matthewrenze. In this course, first, we started with an introduction to data visualization, introduced the basics of creating data visualizations in R, and discussed the various types of data visualizations that we created during this course. Next, we learned how to create and interpret data visualizations for qualitative univariate analysis, that the analysis of a single categorical variable. Then, we learned how to create and interpret data visualizations for quantitative univariate analysis, that is the analysis of single numeric variable. Next, we learned how to create and interpret data visualizations for qualitative bivariate analysis, that is the analysis of the relationship between two categorical variables. Then, we learned how to create and interpret data visualizations for quantitative bivariate analysis, that is the analysis of the relationship between two numeric variables. Next, we learned how to create and interpret data visualizations for bivariate analysis for both a qualitative

and a quantitative variable, that is the analysis of a numeric variable grouped by a categorical variable. Finally, we moved beyond the basics and learned how to create more advanced types of data visualizations, best practices for creating professional data visualizations, and we learned about a few alternatives to using R for creating data visualizations. This course would not be possible without the help of many people that were involved in the creation of this course. However, I would like to give special thanks to Basia Fusinska, Anne Herlache, and Sarah Weno for providing me with technical feedback on all of my course materials. And I would especially like to thank you for joining us for this course and for sticking with it until the end. I hope you've learned some valuable new skills that you'll be able to put to great use, and I hope to see you again in another Pluralsight course in the future.

Course author

Matthew Renze

Matthew is a data science consultant, author, and international public speaker. He has over 17 years of professional experience working with tech startups to Fortune 500 companies. He is a...

Course info

Level	Beginner
Rating	★★★★½ (83)
My rating	★★★★★
Duration	3h 1m
Released	11 Mar 2016

Share course