Exploratory Data Analysis

Lesson 1: What Is EDA? - Notes

Quick links

Data Is Ubiquitous

Introducing your instructors

Chris: Hi, and welcome to Exploratory Data Analysis, or EDA. In the last few months, I learned about R, a programming language, and EDA with the help of my friends from Facebook. And in this course, I'll teach you how to use R to conduct Exploratory Data Analysis.

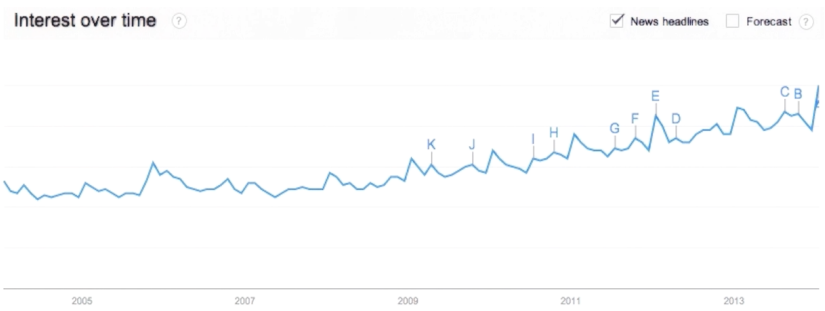
Dean: EDA is one part of the larger process of collecting, learning from and acting on data. In this course, I'll share my advice about working with data and visualizations.

Solomon: By the end of this course, you should feel confident when exploring new data sets to uncover meaningful patterns. In the last lesson, I'll walk you through an analysis of the diamond market with an eye toward building predictive models.

Moira: We're excited to teach this course, and to show you how EDA can be used to answer questions with data. But before we dig in to EDA, let's talk about data. Data is ubiquitous. You can find information about hurricanes, forest fires, and state finances on websites like data.gov. Social networking sites like Facebook, where we work, collect petabytes of data everyday. (One of Facebook's tools, Presto, which is mainly used for adhoc analysis, processes over 1 petabyte of data per day.)And some people have started tracking their own personal data using calendars, mobile apps, and physical activity trackers.

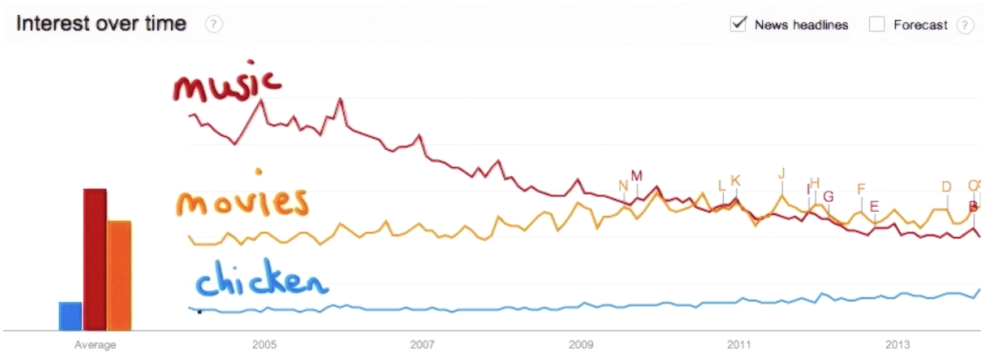
Exploring Google trends

Chris: There is so much data out there in the world, and so little of it has been explored. For example, I was poking around on Google trends and here's what I found. I searched for the word chicken, because I had read a news article that morning about chicken food poisoning and salmonella.



And when I search for the word, Google Trends gave me this graph. This graph shows the interest in searching for the word chicken over time. In reality, it's counting how many times we see chicken in any newspaper headlines in a given month. So from 2005 to 2013, we can see that occurrences of chicken has been increasing. Now, I didn't want to stop my search there, so I decided to enter two more generic word to see what I could find. So I added the word music and I added the word movies. And then I got the graph below.

Quiz: What do you notice in the time series graph below?



Now when you look at this graph, tell me some things that you notice. You can talk about anything interesting that you find in this graph, or you might compare the first graph to this one and write some things that you find. Now, keep in mind there is no right or wrong answer, I just want to get you thinking.

Answer:

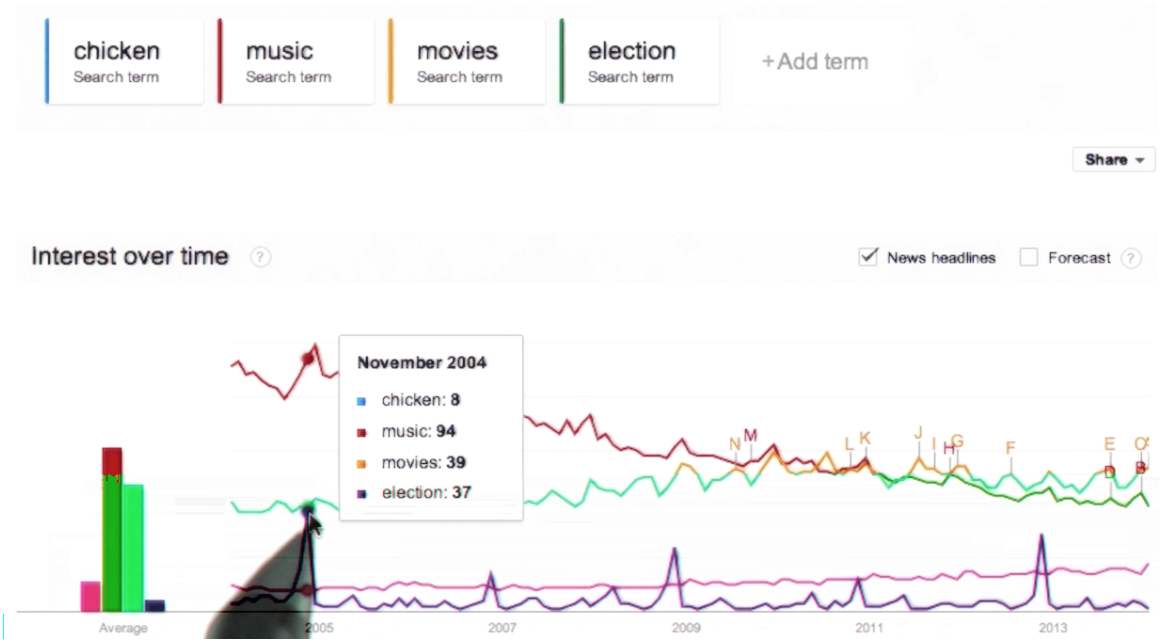
There are a ton of observations to make here, and you might have said something about a particular spike in the graph, or spoken more generally about the trends. Here's what I noticed, I noticed that the vertical scale had changed, since the blue line goes all the way down to the bottom. This means that the number of times that

we see music or movies in headlines is much greater than the number of times we see the word chicken. I also noticed that music used to be more prevalent in headlines, but it's declined in recent years. In fact, the word movies overtook the word music for headline presence around June 2010. That might not have been obvious on the static image. But if you played with the interactive visualization, you could find dates and other interesting things too. If you notice something else that isn't covered in the solution video, share it on the forum.

Go Exploring

Continuing the investigation

I wanted to continue my investigation, so I decided to add another term. This time I used the word election. And here's what was interesting, I noticed these peaks spaced evenly out over time. And then I realized that these peaks made a lot of sense



because the US Presidential election occurs every four years in November. If I hover right over the graph, I can see these peaks in November.

Finally, I wanted to see if there was a word that appeared more than any of these other words in the last few years. So, I typed in the word Facebook. This created a drastic change in the vertical scale in the graph. And I noticed Facebook didn't even come out on the scene until around 2007. I think it's pretty cool that a word such as Facebook dwarfs words like chicken, music and movies. Facebook is certainly the giant when it comes to this graph. This is one small example of how data is

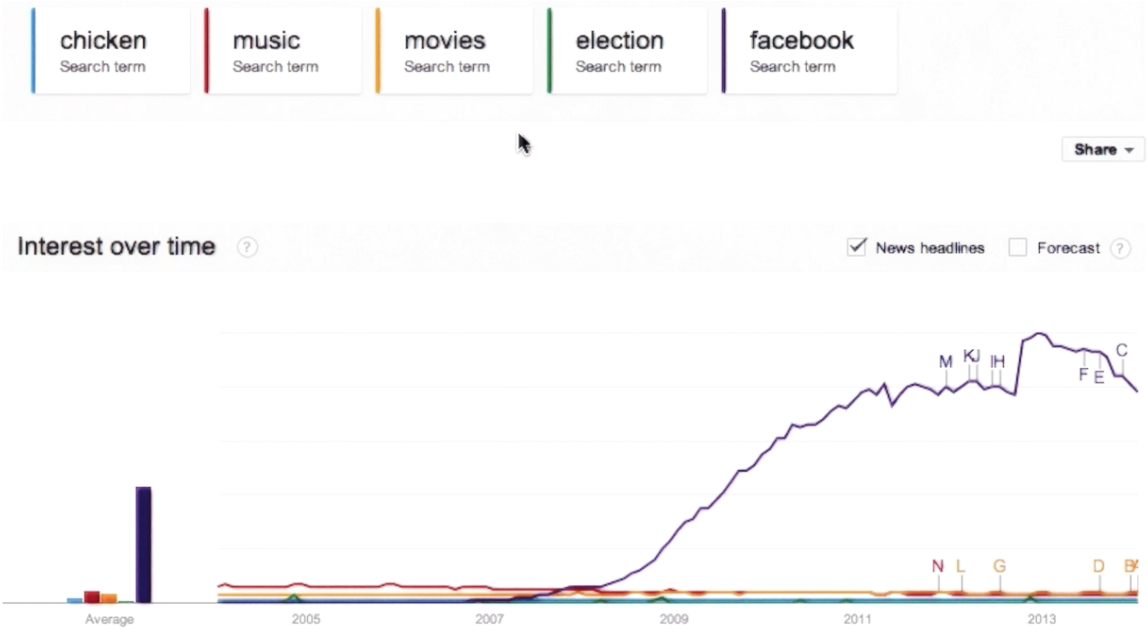
everywhere around us and it's waiting to be explored.

Quiz: Do your own exploration.

Let's have you get your hands dirty, by performing your own trend analysis. Search for celebrities, music, movies, or anything else that you find interesting. You can use the time series graph like I used, or something else on the site. Once you've got your exploration, share it on the forums, and comment on any posts that you find interesting.

So what's getting ubiquitous and cheap?
Data.
And what is complementary to data?
Analysis.

-Hal Varian



Why learn EDA?

Dean: You might be wondering why you should be learning about exploratory data analysis in the first place. Hal Varian, the chief economist at Google, said, “What's

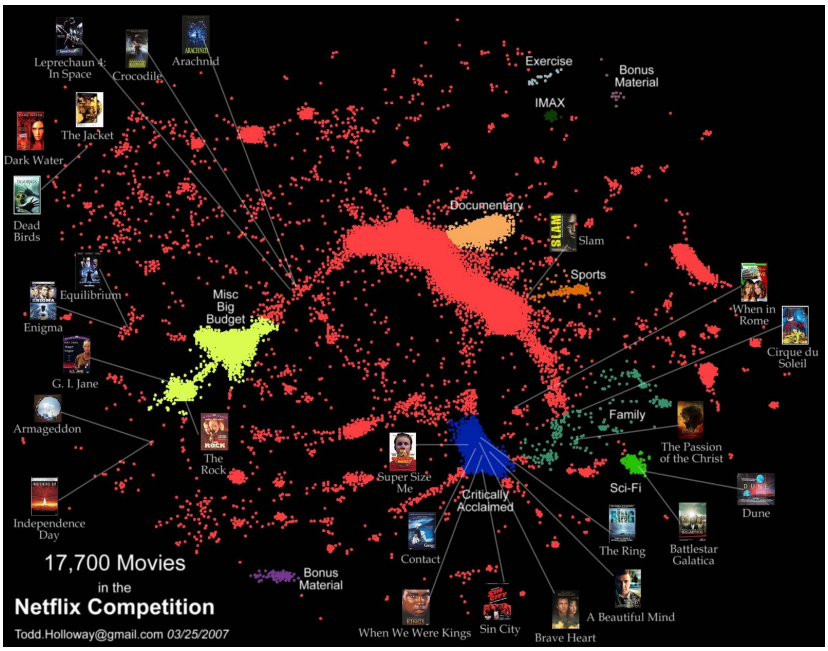
becoming ubiquitous and cheap? Data. And what is complementary to data? Analysis." In general terms, exploratory data analysis or EDA. Is the examination of data and relationships among variables, through both numerical, and graphical methods. It often takes place before more formal, more rigorous statistical analyses.

EDA is often the first part of a larger process. It can lead to insights, or new questions, or even feed into the process of building predictive models. It's also an important line of defense against bad data. It's an opportunity to check some of your assumptions and intuitions about a data set. Many times, business decisions are made using unpolished visualizations that come out of exploratory data analysis. Other times, you might polish some of those visualizations or summaries before presenting them to a larger audience. Let's hear from Chris about an example of how EDA can fit into the larger process of building predictive models.

Netflix Prize Example

Chris: A great example of EDA comes from the Netflix prize competition. Teams competed for a million dollar prize to improve the Netflix movie recommendation system by at least 10%. You can read more about the contest and results by clicking on the following links (Netflix prize dataset visualization, Interactive visualization).

The visualization to the bottom right was created by Todd Holloway to show clusters of movies based on movie ratings. Movies that are closer together, receive similar movie ratings, whether those ratings are high or low. And it's exactly this type of visualization that allows us to get a feel for the data and determine what could be a part of it and what could be explored in more depth.



So, even before performing complex data analyses on the data, we can learn a lot about visualizing our data. Nathan Yau, a noted statistician and visualization guru provides his reflections about this plot. You can read his thoughts and check out the interactive visualization from the instructor notes as well. So, whether you want to learn an in-demand skill or compete in $1,000,000 competitions. EDA has something to offer to you. At the least, learning about EDA will

improve your ability to reason through data, sharpen your communication skills, and expand your career opportunities.

Quiz: Why do YOU want to learn EDA?

Now I want you to take a few minutes to think about your previous work and your educational experiences. Write a few sentences about why you would like to learn about EDA and once you've got your ideas down, share your thoughts and your experiences in the forum. And remember, there's no right or wrong answer here. This question is just for you.

Aude's Interest in Data

Chris: While creating this course, I visited Facebook with Dean, Moira, and Solomon and spoke with many data scientists. I learned a lot about EDA, and I'm excited to share those stories with you throughout the course. First, you'll hear from Aude who will talk about what she finds most interesting about being a data scientist.

Aude: So what I really like is, being able to go from like really raw data - we have like billions and trillions of rows of data. You don't know what's interesting. It's like this gigantic thing and you need to investigate and have some ideas on what might be interesting, and then try investigating it, like, in different ways. Sometimes you use, charts, sometimes you use maps and sometimes you come across something that's interesting and that you want to dig a little bit more into. And then eventually, once we have figured out something interesting that we've developed the corresponding algorithms, then the goal is to have an impact on, either understanding the world or changing the Facebook product. And being able to go from…we have all this data, we don't know what to do with it. So like, we're going to make a significant impact on like, understanding mobility or improving Facebook. That's like a, a real awesome flow of work.

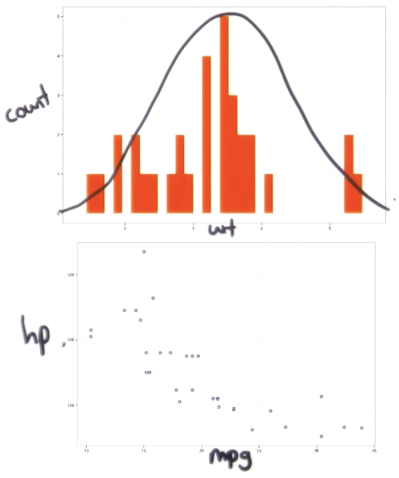
Goals of EDA

So what exactly is exploratory data analysis? You can check out the Wikipedia definition in the instructor notes but simply put it's an approach to understanding data using visualization and statistical tools. Think of EDA as your initial interaction with data.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4

For example, let's say we wanted to look at a set of cars. This data comes from the MT cars data set, which comes with R Studio. Now we couldn't make any sort of conclusions just by reading through the table. It'd be pretty difficult. So instead what we want to do is we want to understand the distribution of the variables for things like miles per gallon, cylinders and perhaps horsepower. We might create histograms, which we'll see later in Lesson three.

The histogram to the right shows the distribution of weights for all the cars. Or we might examine correlations between variables like miles per gallon and horsepower. In that case we'd want to create a scatter plot. We'll examine these in Lesson four.



Our second goal is to assess and validate assumptions. In which future inferences will be based. So we might ask questions like which variables are normally distributed. Or we might be wondering if a variable is biased toward a particular value. Now this might not be the case for our cars data set but certainly in other cases like social behavior or users interacting on the web you might find that.

Third you might want to understand the data before performing and intelligent hypothesis. Well EDA can be the source of an idea for an experiment. It's not a formal

process of hypothesis testing and predictive modeling. Ultimately, we're developing intuition of our data set and how it came into existence. By examining our data we can generate better hypothesis, determine which variables have the most predictive power. And select appropriate Statistical tools, to build our predictive models.

The Growth of Televisions

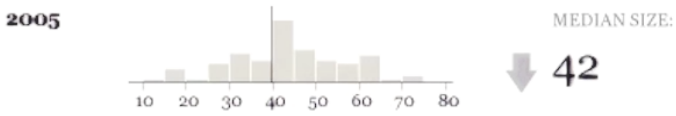
Let's look at another example of EDA. We've linked you to a blog posting on Nathan Yau's blog about Flowing Data. In the post Nathan analyses the size of televisions over the past couple of years. I want you to read through the post and think about how Nathan conducted his EDA. And pay careful attention to the date of visualization and the source of his data. I want you to gather some thoughts about some specific things that Nathan did in his work. And how they relate to the definition and the goals of EDA.

Quiz: Some questions about Nathan's post

According to Nathan's post, what was the median television size in 2006? The largest increase in television size occurred between which two years? What else stands out to you about Nathan's work? What connections are there between his work and the definition and goals of EDA?

Answer:

This video doesn't contain an exhaustive list of everything you might have related to exploratory data analysis. But here are some of the things that we thought about. If you have other ideas, please share your thoughts on the forum. The first thing I notice is that Nathan used histograms to show the distribution of TV sizes in any given year. You can also see that for each year, Nathan included a dark line in the middle. Which represents the median size television for that year. So in 2005, it looks like the median TV size was about 40 inches. So half the TVs were larger and half of them were smaller. Second, Nathan describes where the data originated.

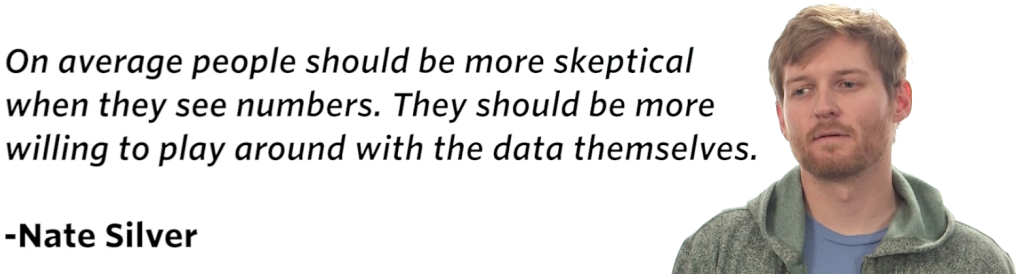


In this case he got his data from CNET Reviews. Nathan describes a caveat in the source of his data. He mentioned that the screen sizes are based on the TVs that were on the market at the time and not on how many people actually bought them. The average television size that people bought could be quite different from the average television size on the market. I think the second part is key and is something we should remember when approaching any data set. We should always be skeptical of what story the data may tell. We should ask questions about our data, inspect it and

consider its context.

Finally, Nathan conducted a time series analysis. So each year after 2002, Nathan indicates whether the median size of televisions increased or whether or not it decreased. In the case of the circle, we know that the television size remained the same. This leaves me wondering what television sizes are doing now. Maybe you can find some data out there and let us know by the end of the course.

Our Approach for This Course



Dean: As you go through this course, and when you're analyzing data on your own, we want you to keep the following in mind. Always be curious and skeptical when you're taking a look at data. Nate Silver a statistician noted for his work in baseball and in politics, captured the sentiment as well. He said, “People should be more skeptical when they see numbers. They should be willing to play around with the data themselves.” Exploratory data analysis is one of the best tools that we have for playing with the data. For letting the data speak really directly. When conducting an exploratory data analysis. We want to test our intuitions about the data set and develop new intuitions.

When you're going through this course and analyzing the data sets that we're going to take a look at. We want you to be excited to play with the data. But we also want you to be open to detecting oddities in the data. Especially through the visualizations and summaries that you produce.

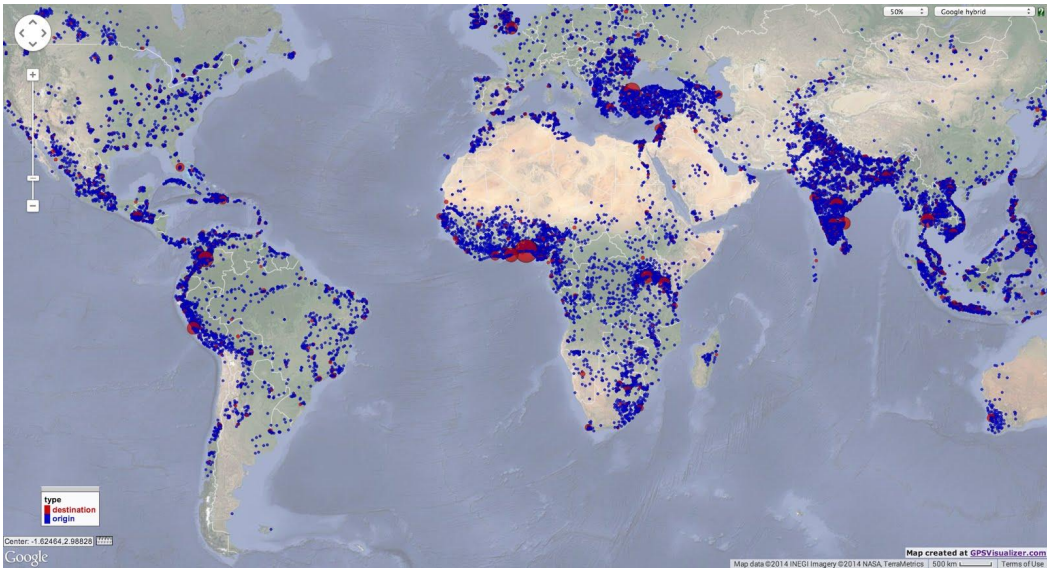
Aude Explores Coordinated Migration

Chris: We want you to develop a mindset of being both curious and skeptical, when you work with data. To help you get into this mindset, I want to share another conversation that I had with Aude. In this next video, I want you to listen to Aude's work and look out for how she demonstrated this

exact mindset.

Aude: So we gathered all the hometowns and current cities from the users and I was looking at conditional probabilities given a hometown. What is the probability that you currently live in each different cities? Like, for example given that your hometown is New York, what is the probability that you live in Chicago or that you still live in New York or that you live in San Francisco or Paris and so on. And what I was expecting is that, at least, the most likely city, where you would live right now would be your hometown.

If you grew up in Chicago, the most likely place that you're going to be now is still Chicago. You could be moving as well but the most likely place would remain your hometown. But I saw a fair number of cases where the most likely current city was different from the hometown and that was, was a fairly high probability. I was really

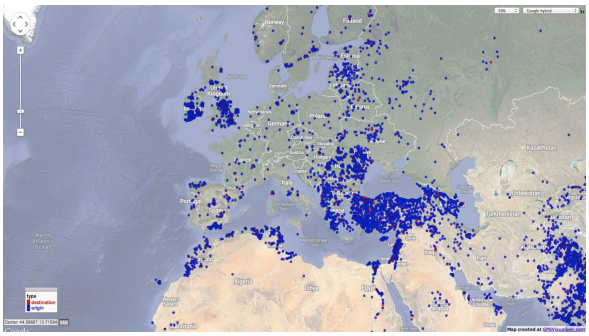


surprised. I was wondering if I had a problem in my computations, if there was some issues upstream of what I was doing.

So I decided to put all the cities on a map. All the pairs of hometowns and current cities for which the most likely current city was different from the hometown. And what we saw on this map was really fascinating because it was really not what we we're expecting. It was not a bug in the code. We were really seeing patterns arise.

Here we only plotted pairs of hometown and current city, so there's no movement between the pairs but what we see is that a lot of these cities for which the most likely current city is different from the hometown arise in western Africa or in India or in like

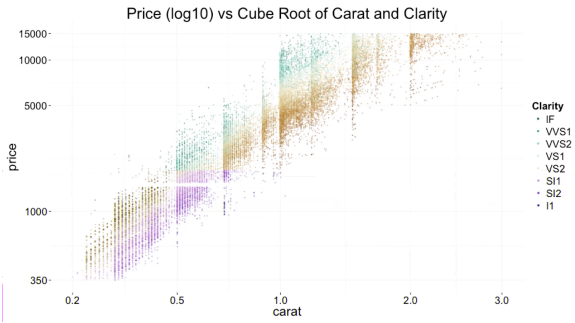
Turkey, which we were not necessarily expecting at the beginning. And there were a lot of small cities all moving to the same current city and so we decided to dig a bit more into it.



One thing that happens is that sometimes the distribution of the current city is very flat. Given that you grew up in, let's say Paris, maybe you're still living in Paris but maybe you live in like one of the thousand cities in the suburbs and so the distribution is really flat and so we have to decide what was considered as a coordinates demarcation. We decided yeah, the probability to move to that city is high enough that we're considering that.

And the other thing we have to think about is that if you look at the map at the world scale or if you zoom to a very specific area, you don't want to see the same things. So,we also want it to have interactivity in the visualization. And so we decided to use D3, which is a Javascript-based visualization framework, which enables you to have a lot of interactivity with with your data and enabled us to do a lot of that exploration and so on. (You learn more about Aude's work here or about global urbanization trends here).

Course Overview



Chris: Visualizations like this one are a core component of exploratory data analysis. We're going to be using the R programming language to create these types of visualizations and to explore data throughout the course. You can read more about the R programming language and how it was developed by clicking on the link. We'll also learn about RStudio. It's a graphical user interface for programming in R. You'll learn all about RStudio in the next lesson, and future lessons will cover visualization and statistical techniques for data exploration. We'll start by examining one variable at a time in a dataset. This will give us an idea of its underlying distribution and what type of values the variable takes on. Eventually our visualizations will become more complex as we look at pairs of variables and multiple variables all at once. At the end of the course you'll have the chance to

conduct your own exploratory data analysis. I hope you're getting excited to explore data. See you in the next lesson.

Exploratory Data Analysis

Lesson 2: R Basics - Notes

Quick links

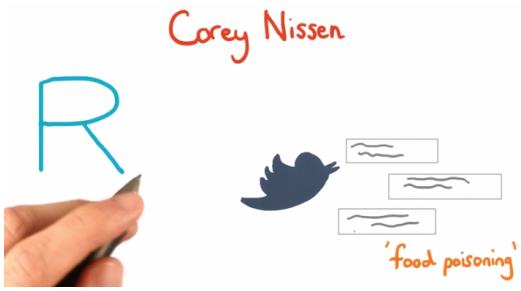
The Power of R

Intro to Lesson 2

Welcome to lesson two. In this lesson, we'll learn about the advantages of using R to perform data analyses. We'll install R Studio. And we'll learn the basic data types and commands of R.

Analyzing Tweets in Chicago

Before we get started, I want to share with you how one data scientist used R to improve the health outcomes for the citizens of Chicago, Illinois. Data scientist Corey Nissen used the R programming language to analyze tweets from people in the city of Chicago. Corey analyzed tweets in real time by searching for the words food poisoning in each of them. I want you to read more about his analysis and his work by clicking on the link in the instructor notes. Try to find the answers to these next three questions as you read through the article.



Quiz

What was the name of the R package used by Corey? How many Tweets per day did the system flag? (lower and upper limits) What did you find most interesting about the article?

Answers

Textcat was the name of the R package that Corey used. We'll discuss R packages in the next video and what they allow us to do. For the second question, the automated system captures between ten tweets to 20 tweets a day. For each of the tweets, the system will recommend that whoever sent the tweet file a report. Now, for this last question, we accepted any text answer here.

I thought that the most interesting part was that Cory showed the open source R code classifier on GitHub. GitHub is a repository for sharing code. You can learn more about GitHub by following the links in the instructor notes. It's the most popular repository for open source projects and it's really easy to get started.

Why R?

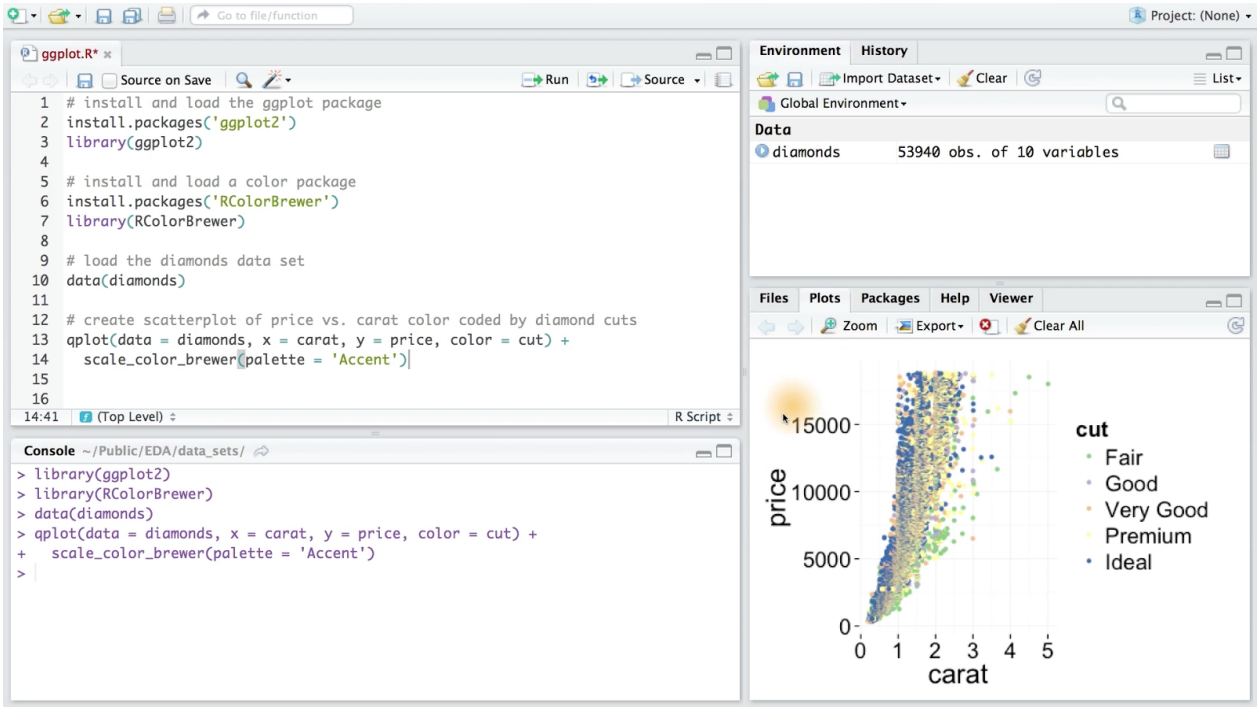
About R

R is the leading programming language for statistics and data analysis. One of the main advantages of using R is that we can build up an analysis line by line in code. We can save all of our work in a file and go back to see what we investigated at a later date. Having R scripts allows you to easily share your work with others. And you can see what others are doing with data. R also has over 2,000 user contributed packages that increase its functionality. One example of this is the text analysis package, TextCat, that you just saw.

ggplot2

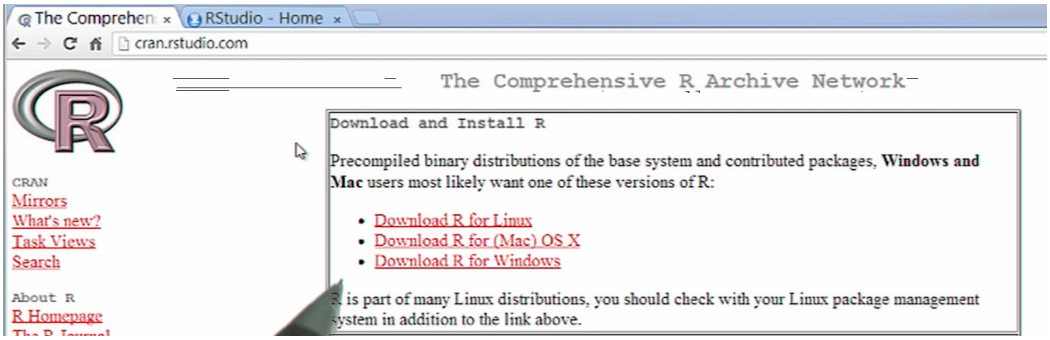
Now, one of the main packages we'll use throughout this course is ggplot2. ggplot2 is a graphics package that lets us create plots and graphs with just a few commands. We'll learn more about ggplot in the next lesson but just to give you a taste of it, here's an example of a plot you can create. Don't worry about memorizing or understanding all of this code. You'll have plenty of practice with this later in the course. I just want to show you how a few lines of code, can create amazing graphics.

I'm going to load up the ggplot library and a color library. Then I'm going to load the diamonds data set, and with this function I'll create a scatter plot. Let's check out



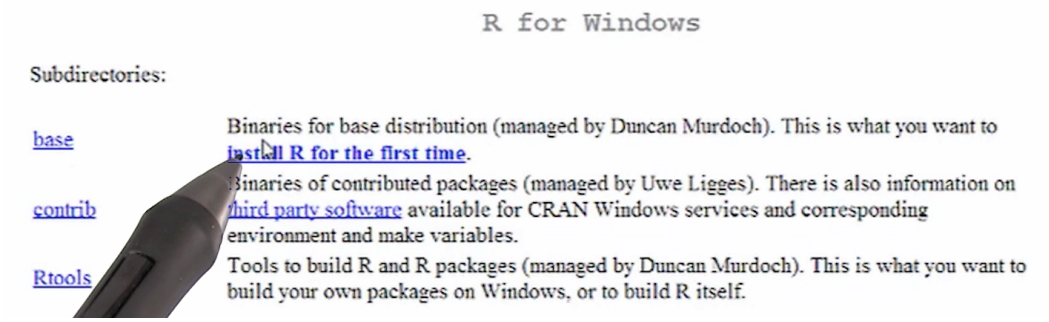
this plot in detail. This part shows the relationship between price and carat of almost 54,000 round cut diamonds. I'd say R is doing very well for such few lines of code. The last thing I want to mention is that you can use R anywhere. It's free open source software that works on any operating system. And as a result of this, R has a large active and growing community of users.

Install RStudio on Windows

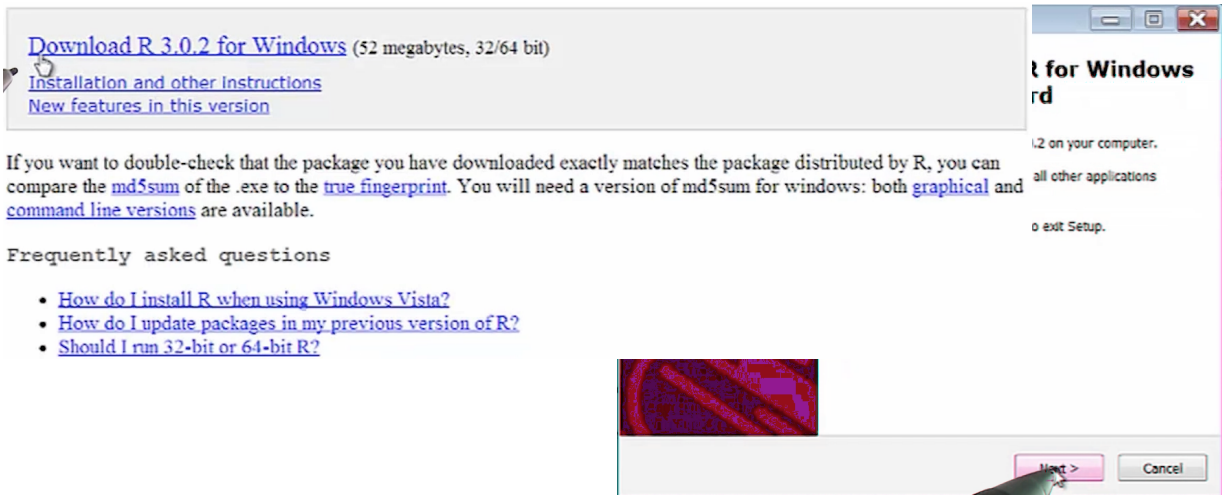


R Programming Language Installation

If you're using windows, I'll show you how to install R and R Studio. To get started you want to go to cran.rstudio.com. And then you'll click on the link to download R for Windows. If you never installed the R programming language before, then click on this first link. Since it's your first time downloading it. Now depending on your version of Windows, you might need to see the frequently asked questions. I'm set for my computer, so I'm going to go ahead and download R for Windows. I get a warning message and I want to keep the file. Now this might take some time to download, so just be patient. Then you want to open the file, and then run the program. A set up wizard should appear and now you just want to walk through all the steps. I'll set my language, and then I'll just continue installing by hitting next. I'm just going to accept the defaults here and then install it to R. I'll leave these as is, so that way a desktop

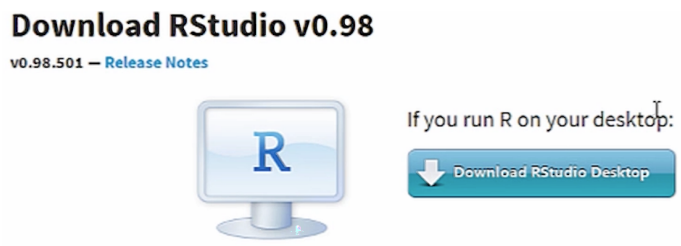


icon will be created. And it's going along. Great. Now I'm finished with my setup.

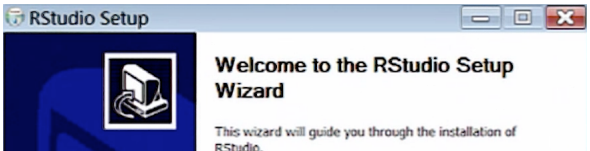


RStudio Installation

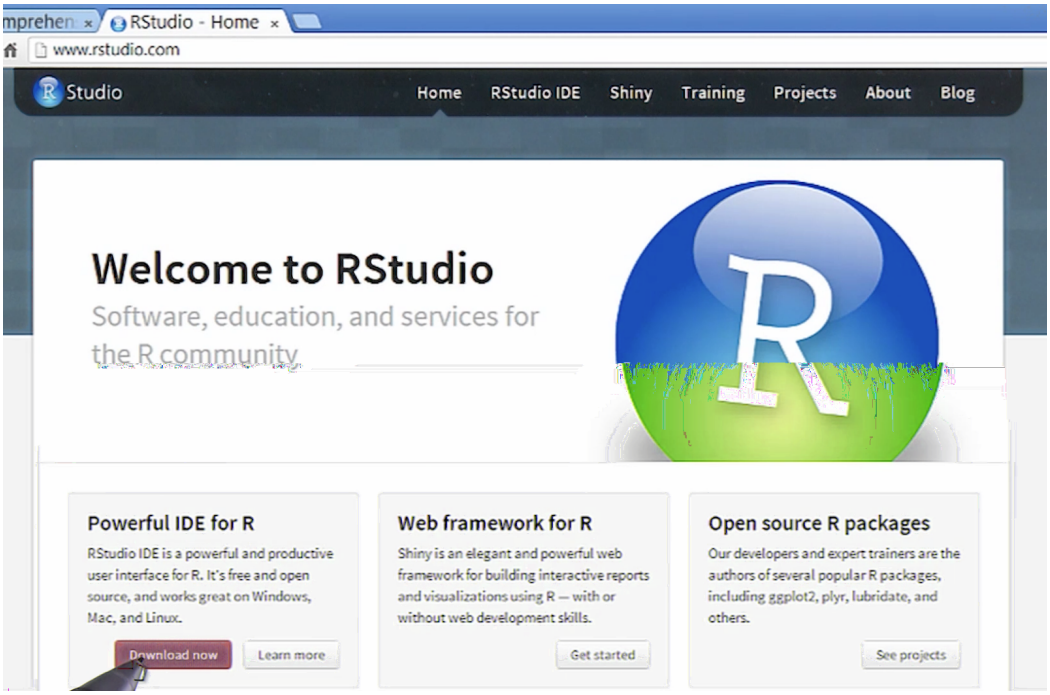
So that was just part one of the installation process. That just gave us the



programming language R. Now we need to go and install RStudio. RStudio is a graphical user interface for programming in R. It's basically a point and click system that lets us work with data really easily. So first let's just go to download now to get RStudio. We're going to get RStudio for our desktop. So just click this first button. Now there's going to be a lot of links here, and you really just want this first link. It's going to be the version of RStudio that's recommended for your computer, and for

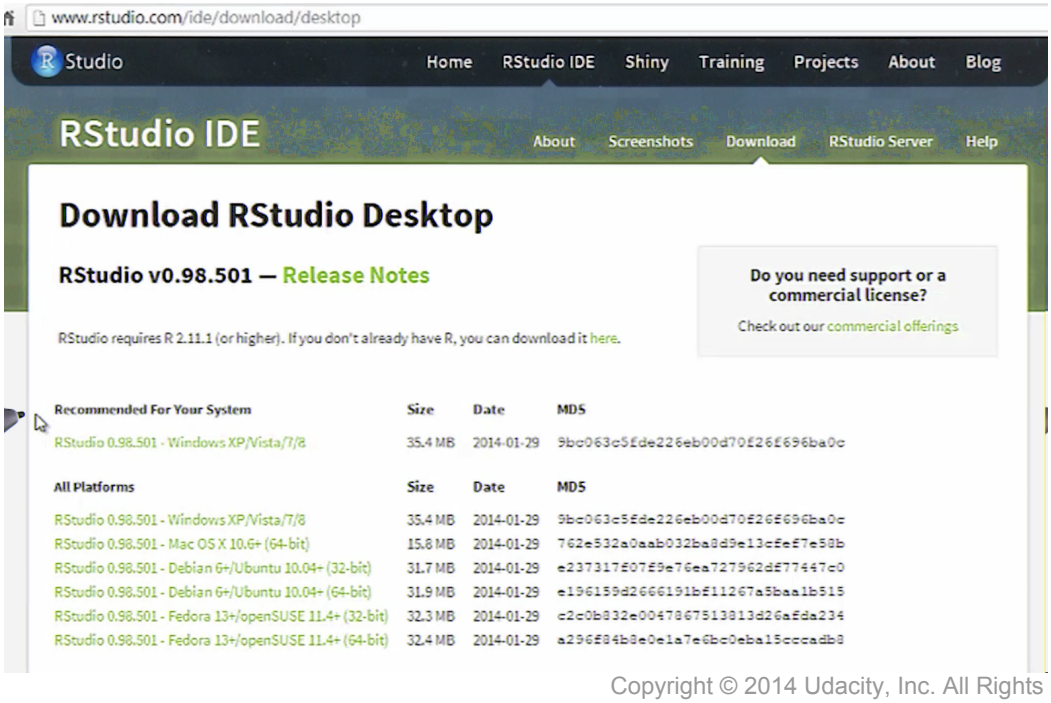


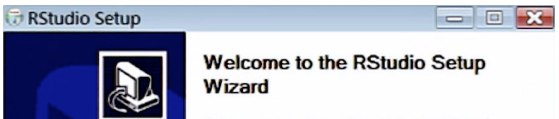
your operating system. Notice how this one's for Windows XP, Vista, Windows 7, or Windows. Now that it's ready, so I'm going to open this up, and I get the wizard you can see on the right. I'll follow the instructions, then install RStudio into this path. And create a Start Menu folder as well. Great. I'm done. And this is what RStudio looks like. Now yours might look slightly different than mine and that might be because the file's open. So if I go to File > New File > RScript this opens a basic RScript



which allows me to start programming in R.

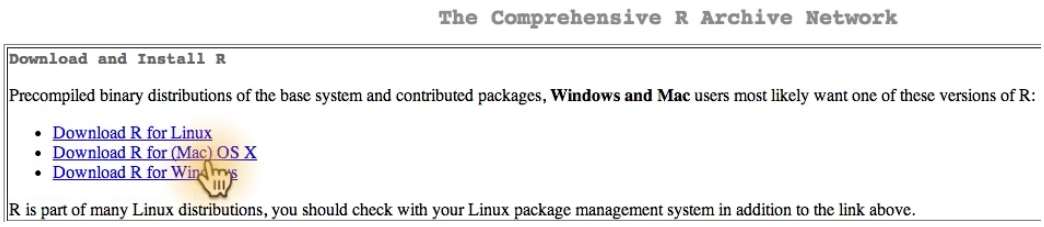
Install RStudio on a Mac



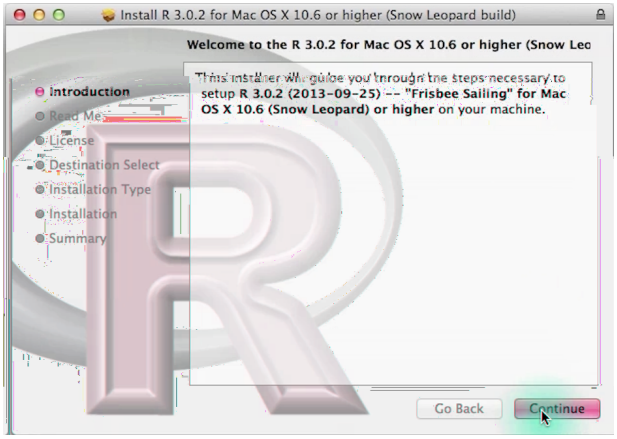


In this video, I'll show you how to install R Studio on a Mac, which is a graphical interface for programming with the R language. Simply open up any web browser and go to RStudio.com then you want to go to download now. You are going to run R Studio from your desktop so click on this link. Now here's where I want you to be very careful. R studio requires R first, so if you don't already have the programming language R installed, you need to download it here. Now this is a really easy link to miss so make sure you click here first to install R, before installing R studio. Once you're at cran.rstudio.com, you want to select the download for the Mac OSX version. This will bring up another page and then you can download the latest version of R. It may take some time to download so you might want to take a break and then come back to the computer.

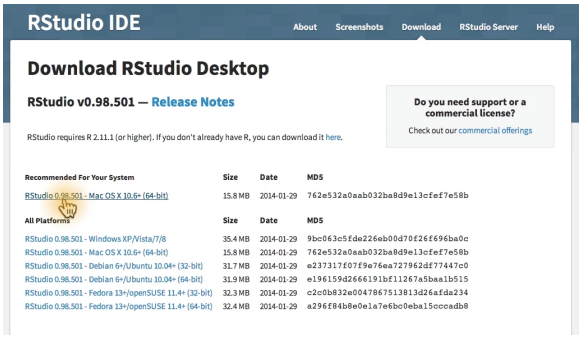
Once you've opened up the package, follow the instructions for installing R. You should end up getting a successful message for the installation. Now you're ready to go and install R studio. This website will recommend the best version of RStudio for you so just click on whatever one comes up for you first. Once you've downloaded this, you want to open that package. Now, just drag RStudio into your applications

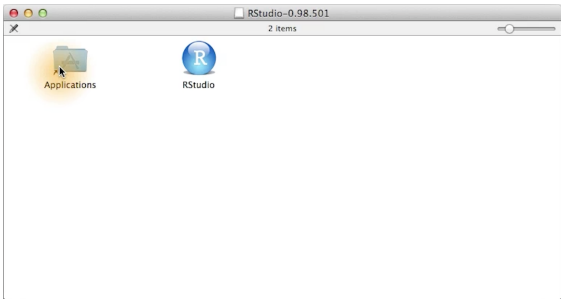


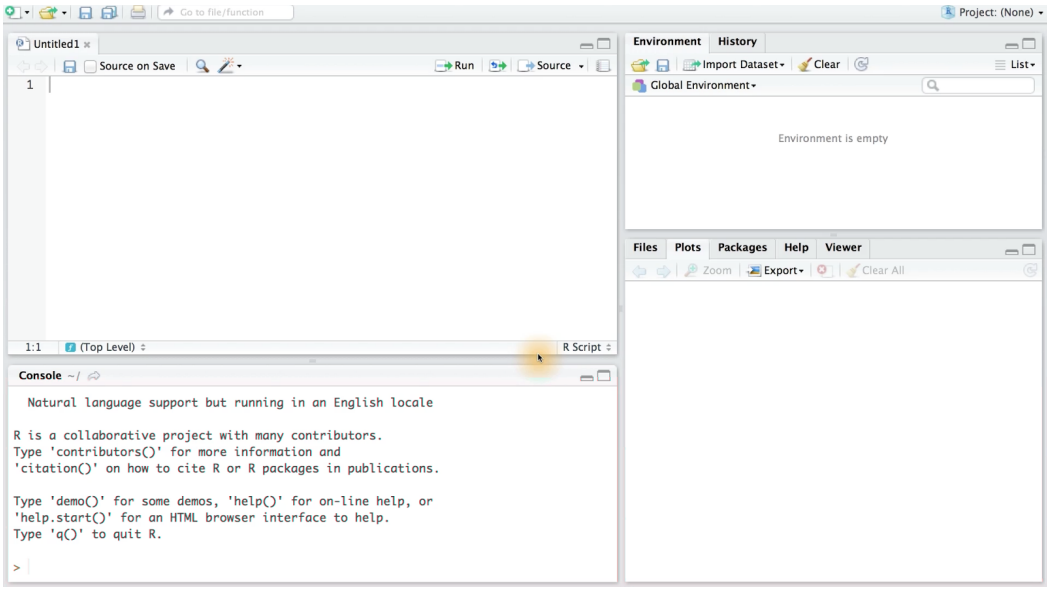
and you'll be good to go.



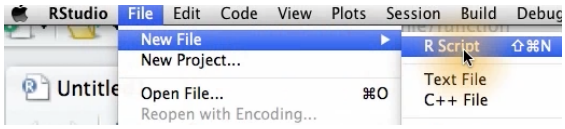
RStudio Layout

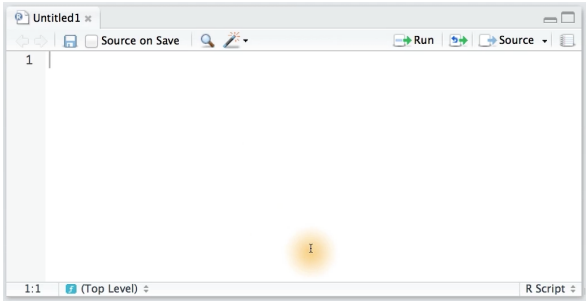






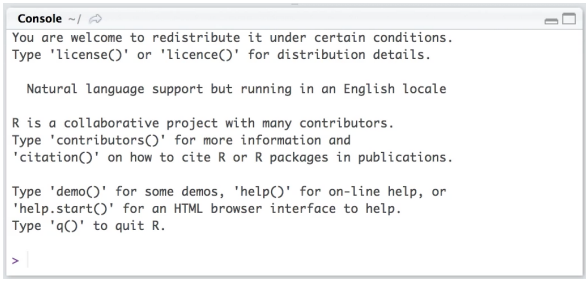
This is R studio, it's an IDE or an Integrated Development Environment and it's what we'll be using to create R scripts and to create plots. It's a



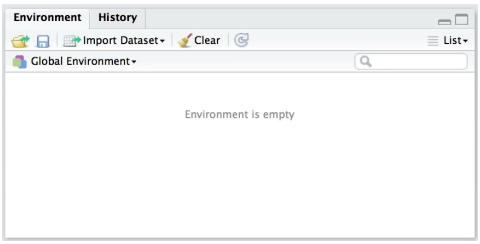


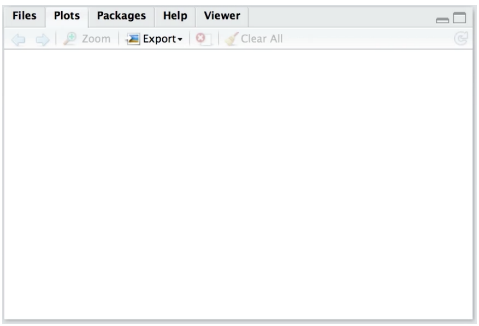
point and click system that organizes our workspace and it has some nice features for creating and sharing work. Here's a basic tour of it. The main interface contains four panels that can be resized as needed. Now, you may only see three panels when you open R studio. If so, just open a new R script by clicking on the file, and then New File > Rscript.

So, in the top left here, we have Rscripts. These are the files in which we will type our R commands. We'll also save these files, so we can alter them or share them at a later time. In the bottom left, we have the R Console. You can type in R commands here. However, these R commands will only be saved in a temporary history. The commands won't be saved to a file like up in here that can be reviewed at a later date.



In the top right of R studio, we have two tabs. One called Environment and one called History. The Environment will contain all the objects, functions and values that are in the current working memory for R. History on the other hand will keep a running log of any of the R commands that we run. Now we can run R scripts directly from our files but we can also type them into the console. No matter which way we type them in the History will capture them.

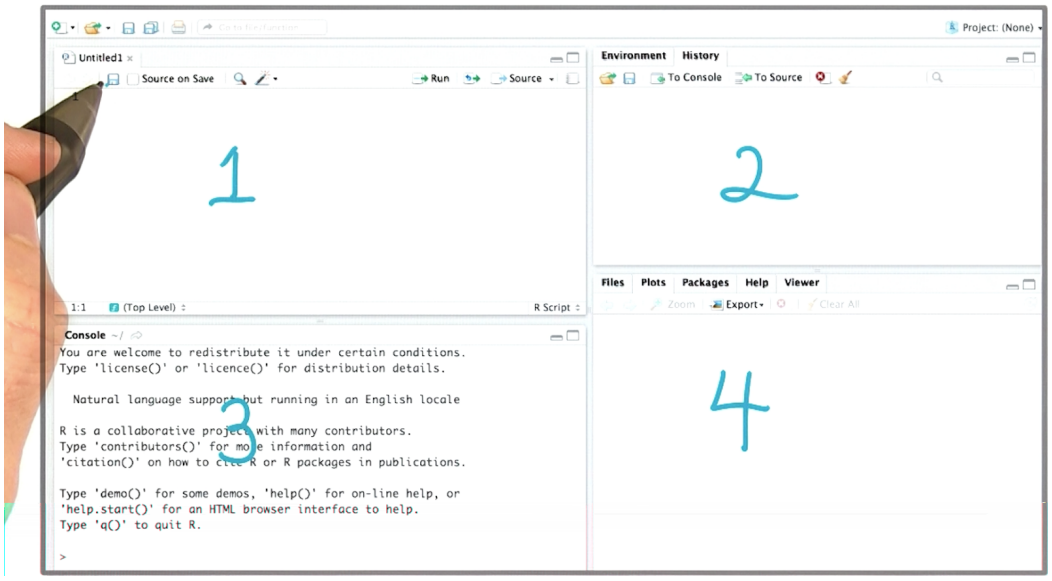




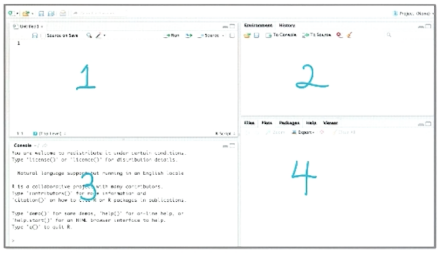
And finally, in the lower right, we have the Files, Plots, Packages, Help and Viewer section. You'll learn more about these later but this is where our plots or graphs will first appear when we run an R command that creates a visualization.

Quiz

Now that we've gone over the layout of R Studio, I want you to match some actions that you might do in each panel of R. Remember this first panel is for R files or Rscripts. This second area is for the workspace or the history. This third space is



the Console where we get our output. And this fourth panel is for Files, Packages, Plots, and Help. So for example, if I asked you to match the activity to one of these panels, like save an R script that would correspond to box number one. You can save them by clicking on this button. Now we could of course use the main menu, but I'm actually include as any one of your options here. The main purpose of this quiz is just to get you familiar with R studio. So here are your list of actions. And I want you to pair it with the panel where you would do it in R studio. Put the number of the corresponding panel in each of the boxes.



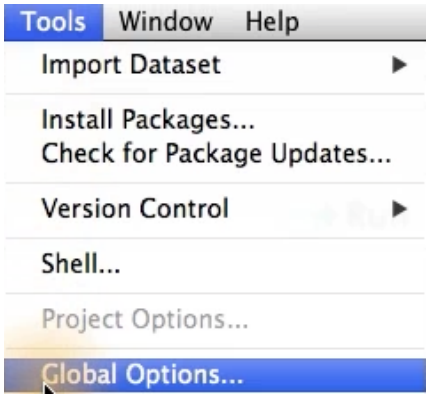
1 R scripts / files
2 Environment / History
3 Console
4 Files / Plots / Packages / Help

- [x] 1 Save an R script
- [ ] Review the log of commands entered
- [ ] Read help documentation
- [ ] Clear the workspace
- [ ] Run a bunch of commands from a file
- [ ] Look at a plot
- [ ] See a list of objects in memory
- [ ] Read the results from functions or calculations

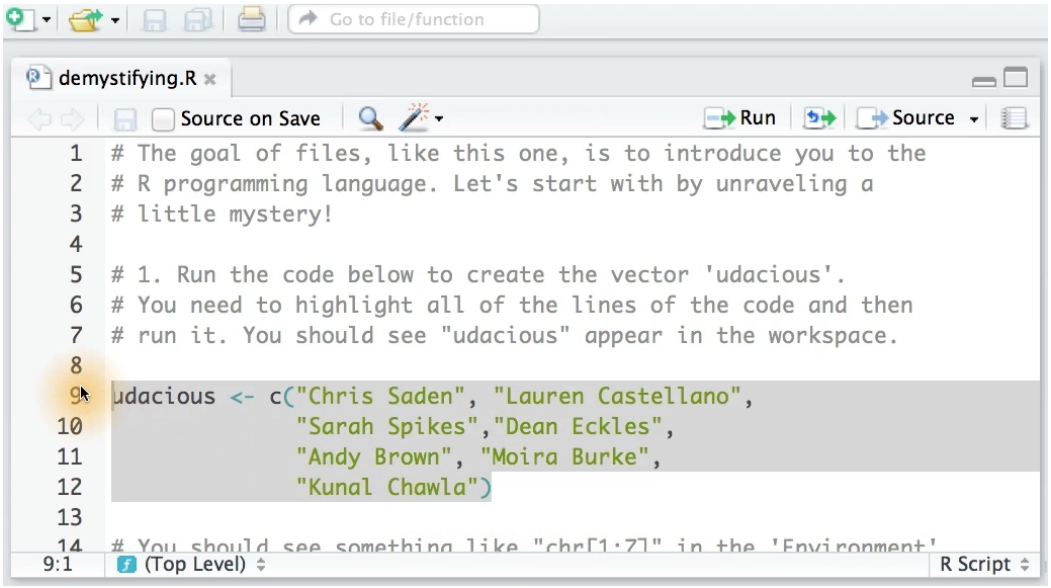
Answer

You can review the log of command entered by going to the history so that would be panel two. Help documentation appears in the lower right hand side which would be panel four. To clear the workspace you can either use panel two or panel three. You can run a command in here to clear the workspace or you can use the icon located in panel two. It should look like a sweeping broom. We can run a bunch of commands from a file from panel one. Plots appear down here in the lower right, so that would be panel four, and all of our objects and memory appear up here, in our environment or our work space. So that's panel two. And finally to read the results from functions or from calculations, we simply look at the console, where our output comes, panel three.

Demystifying R



Now that you know the basic layout of our studio, you're welcome to customize it, by going to Tools, and then Options. From here you can choose the default directory if you'd like. You can change formatting options for editing code, or you can even change the font and the appearance. Here, I'm using the tomorrow theme, but there are of course some others. You can also make adjustments to the panels so, feel free to play around with this to get a configuration that you like. Now, you might not have any idea what you like first so just wait until you play with r a bit, and then you'll find a configuration that suits you. I'm going to leave this as is.



Alright, we've got R set up, so let's start using the programming language. To get started, I want you to download the R script (demystifying.R) and open it here. When you open the file, it should look like this. This R script contains both comments as plain text and R code. You can run R commands by highlighting a chunk of the code and then hitting Cmd+Enter on a Mac, Or Ctrl+Enter on a PC. The lines of text that start with a hash or a pound symbol are comments. There lines are plain text that we don't want to run as R code. It's the standalone text that you really want for the R code.

Now, your task is to read through this file and run and write code when you're prompted to do so. When you get to the end of the file, you'll get to a question, and you'll need to answer this question to move on to the next video in this lesson. Now, if you're already familiar with R, I suggest trying to answer the questions, and then you might want to go back to the files to see if you missed anything.

We know you learn best by doing, so it's important that you execute and run this code as you go, and that you make sense of the output. Any output will appear down here in the console. I also hope that you have moments of inspiration. So if something does come to mind I want you to code it up and just run it in R. You won't break anything and the worse thing that would happen would be an error message or a warning message in this box. If you do happen to get stuck at any point post in the discussions so that way your peers or one of the instructors can help.

Quiz

Download the demystifying.R file and open it in R Studio. As you read through it, read and write code when prompted. When you get to the end of the file answer this question: what is the average mpg (miles per gallon) for all of the cars in the mtcars dataset?

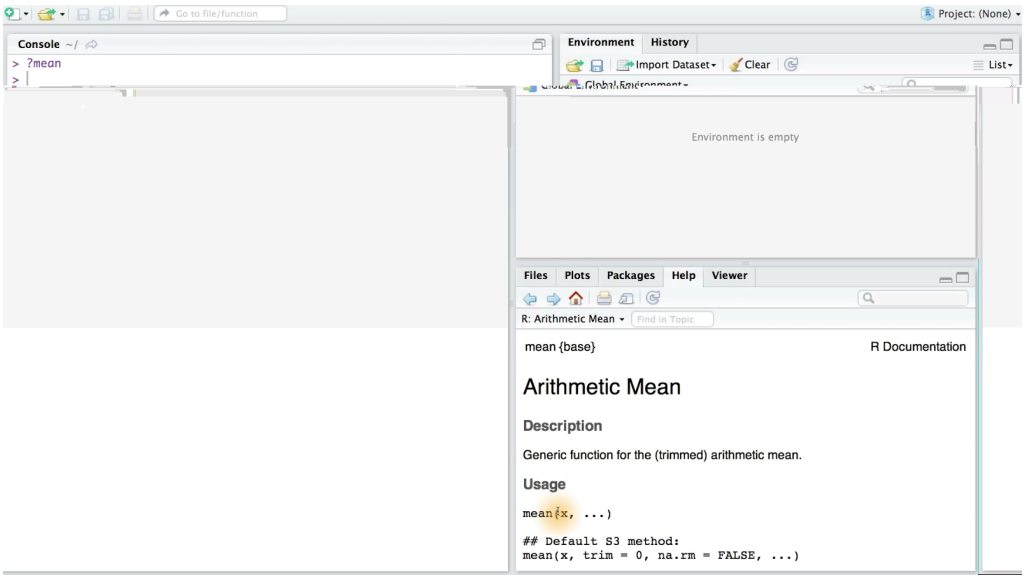
Answer

This question should have been pretty simple. You just needed to run this bit of code, "**mean(mtcars$mpg)**", and that would have given you the value to enter.

Getting Help

As you go through this course, you might encounter commands that you've never seen before. And we don't expect you to know everything, but we do want you to keep the following in mind when you encounter any sort of challenges or difficulties. First, take an active role in problem solving and be aware of the resources at your disposal. You can type a question mark and the name of any function to bring up help documentation in R.

So for example I can look up the documentation for the function mean. Here it will tell me any parameters that mean takes. And it will give me examples usually at the bottom of the documentation. I can copy and paste these into the console. And press enter to see what the function does. Here, I created something called **x**, and then if I hit **x** into the console, I can see what this is. This is just the vector of numbers zero to ten, and then the number 50. I can take the mean of **x**, and print it out. And finally, I have some sort of trend



parameter. Maybe you can figure out what this does.

```
Console ~/
> ?mean
> x <- c(0:10, 50)
> x
 [1]  0  1  2  3  4  5  6  7  8  9 10 50
> xm <- mean(x)
> xm
[1] 8.75
> c(xm, mean(x, trim = 0.10))
[1] 8.75 5.50
```

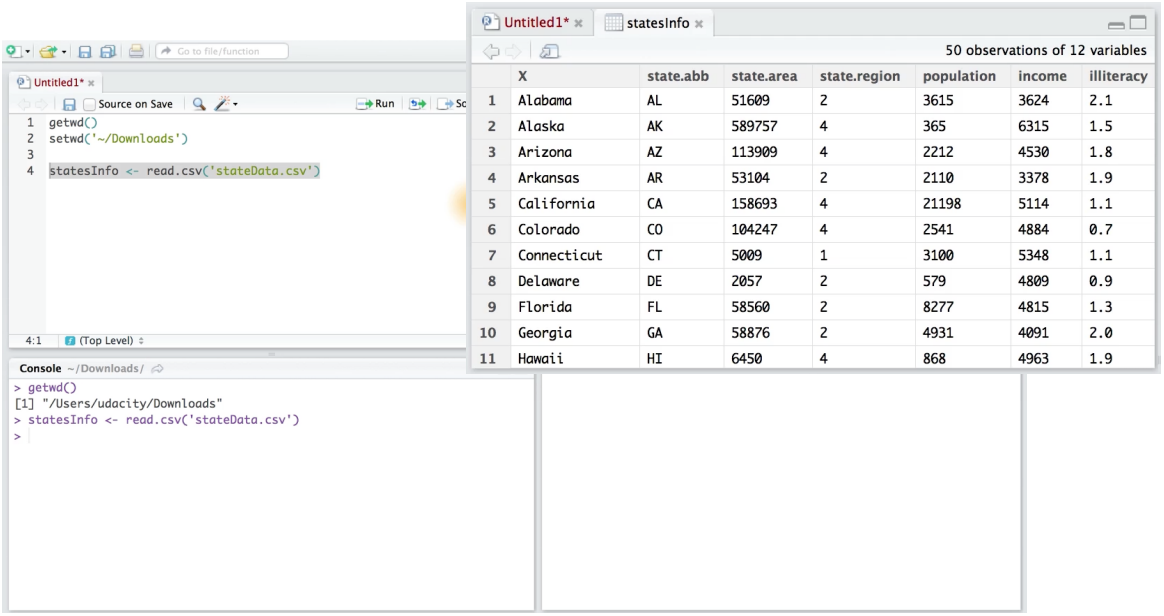
Now, if you can't solve your problem using the documentation I would suggest Googling it. StackOverflow is also another great site to browser for help and for an FAQ. One of my personal favorites when I learned R was Quick-R, or statmethods.net. This site has excellent examples and short pieces of code that can refresh your memory and get you going on any data analysis. I've linked to a couple other websites like the R Cookbook and R-bloggers, and you can also find these in the course materials under R Resources on the course wiki.

Read and Subset Data

In the next three lessons, we will explore a data set that resembles user activity on Facebook but before we get to that data set I need to show you how to read in data that doesn't come with R and how to subset that data. It turns out that not all data sets can be easily loaded like the mtcars data set. You can find a link to a new data set here. This data set is going to contain information on the fifty states in the US. Most of the data comes from about 1970.

Before we can read in the data, we need to set our current working directory. So, to figure out what directory you're in now, you can type in **getwd()**. We can run

this command and see the output to the console. It looks like I'm already in the downloads file, and that also appears here in the top of my console. So I don't need to change my directory, but maybe you do. To change your directory, you can type **setwd('***directory***')**. This will take a string which will be the file path to whatever directory you want to go to. My guess is that your data set is in your downloads file. So I would probably run this command (**setwd('~/Downloads')**). Now it's important to note that whether or not you are on a Mac or Windows machine you still need to separate your paths or your folders with a forward slash. Also be sure that you use quotes around your path.



	X	state.abb	state.area	state.region	population	income	illiteracy
1	Alabama	AL	51609	2	3615	3624	2.1
2	Alaska	AK	589757	4	365	6315	1.5
3	Arizona	AZ	113909	4	2212	4530	1.8
4	Arkansas	AR	53104	2	2110	3378	1.9
5	California	CA	158693	4	21198	5114	1.1
6	Colorado	CO	104247	4	2541	4884	0.7
7	Connecticut	CT	5009	1	3100	5348	1.1
8	Delaware	DE	2057	2	579	4809	0.9
9	Florida	FL	58560	2	8277	4815	1.3
10	Georgia	GA	58876	2	4931	4091	2.0
11	Hawaii	HI	6450	4	868	4963	1.9

Now in order to load up the data, we can use the **read.csv('***filename***')** command. This command takes a string, which is the name of the file. And here we're going to pass it to a variable called **statesInfo**. **statesInfo** is going to save all of our data into a data frame. When I run this code, I can see that states info appears in my environment. I can double click on the data frame in the workspace, and this will let me see the table of values in R Studio.

Now, let's say I wanted to get information on states located in only the Northeast. Those states would be states like Connecticut and they have a state region of one. I'm going to go back to my R-script and write a command that pulls in this data. This subset command would look like this. Here I'm passing the data frame states info to subset and I'm asking for it to retrieve any states that have a **state.region** equal to **1**.

```
6 subset(statesInfo, state.region == 1)
```

When I run this code I can see the output down below. We have Connecticut, Maine, and many others. Now, there is another way to subset this data frame. I didn't want to show you it, just so that way, you don't get confused later if you see it. It uses bracket notation, where we have the name of our *dataSet*, followed by two brackets. And we'll have a comma in between. This first spot is for the *rows* of our data set that we want to keep. And the second spot is for the *columns* that we want to keep.

```
> subset(statesInfo, state.region == 1)
                X state.abb state.area state.region population income
7     Connecticut        CT       5009            1       3100   5348
19          Maine        ME      33215            1       1058   3694
21 Massachusetts         MA       8257            1       5814   4755
29 New Hampshire         NH       9304            1        812   4281
30    New Jersey         NJ       7836            1       7333   5237
32      New York         NY      49576            1      18076   4903
38  Pennsylvania         PA      45333            1      11860   4449
39  Rhode Island         RI       1214            1        931   4558
45       Vermont         VT       9609            1        472   3907
```

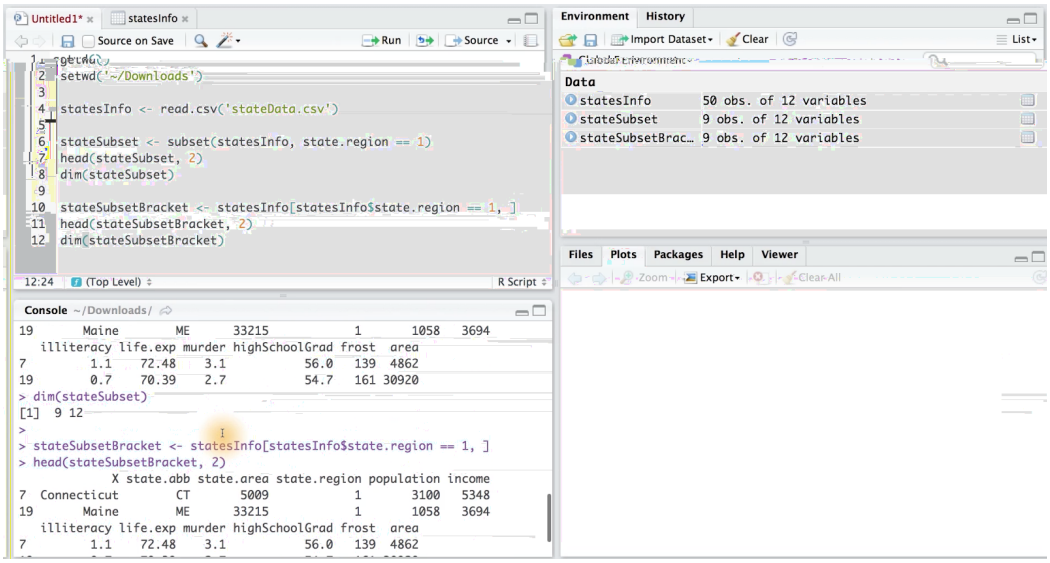
```
dataSet[ROWS,COLUMNS]
```

So if I want only the states in the Northeast, I would write this code. The name of the data set is states info and then I want the rows that have a state region equal to one. Now I can't just use state region here, I need to access the actual variable, so I have to put states info and the dollar sign. This gives me the actual variable value and I can see if it's equal to one. If it is equal to one, I want to return every single column in the

```
8  statesInfo[statesInfo$state.region == 1, ]
```

data frame. So for example, with Connecticut if it's state region is equal to one. I want to return every single column in this row.

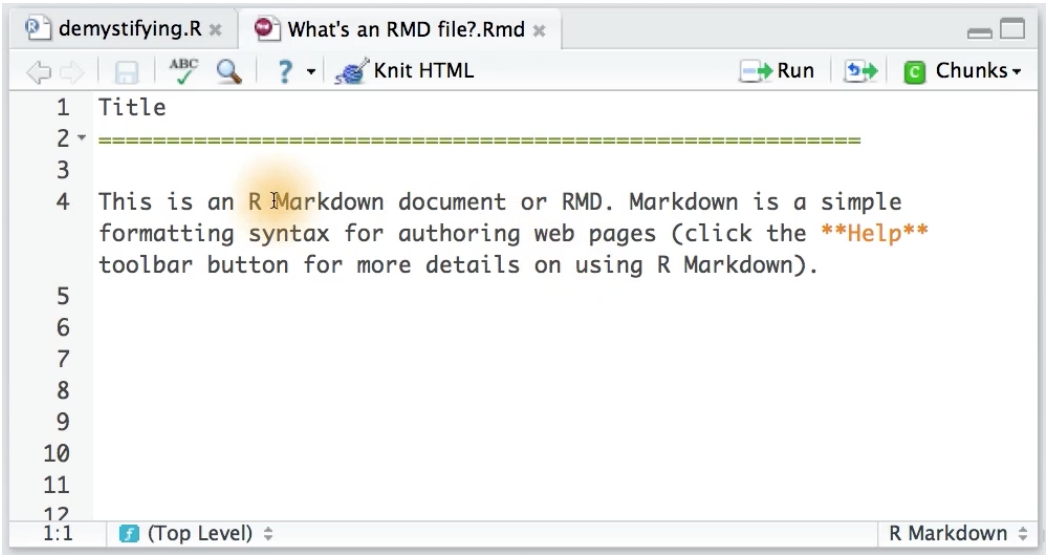
To return all of the columns, I'll just leave this blank. So this code searches for rows that have a state region equal to one. And then it takes all the columns in that row. And all of this will be sent to the console as a new data frame. Now it might not be so helpful to have this output just in the console. So we can save these subsets into new variables. I'm also going to include some functions to print out the first two rows of each data frame, and also their dimensions. Hopefully, I've convinced you that they're the same data set.



Now, I really want you to pay careful attention to the syntax in both of these examples. Throughout this course, we tend to make use of the subset command, but there might be instances where we use the other method. Just know that both methods produce the same result. Now, I recommend that you try subsetting this data frame for other regions of the country on your own. You could also try finding out which states have an illiteracy rate of 0.5%, or which states have high school graduation rates above 50%. Feel free to play around.

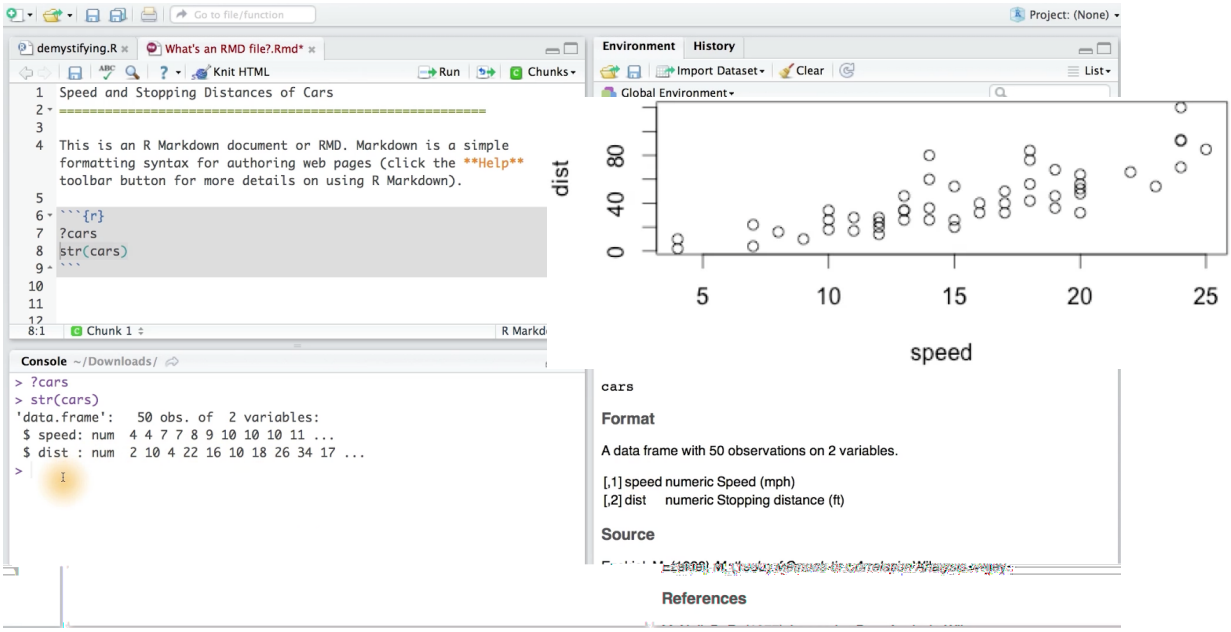
R Markdown Documents

Let's get some more practice working with data frames. You downloaded an R script earlier and saw how we can save our work and run an R code from it. For your next task, you're going to download an RMD file and run code in it. The file will look something like this. Notice how this file is slightly different from the file that you saw before. An R script can only have R code and comments. This file, however, the RMD file, allows us to do so much more. It's an R Markdown document, or RMD.



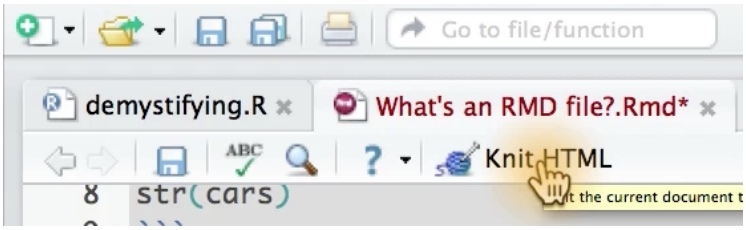
Now, you might be wondering what Markdown is. And basically, it's a simple formatting language that lets us author web pages. If you want to know all you can do with Markdown then go to Help and then go to Markdown Quick Reference. This would bring up the documentation for Markdown, and you can scroll through some of the examples in the formatting. We've also included a video that will go through Markdown.

So far this file only contains text that will be formatted using Markdown. Let's add some R code to this. We can do this by clicking on Chunks and then going to Insert Chunk. Now if you're friendly with the keyboard, you can use the shortcut Cmd+Option+I. There's many other shortcuts in here and you can see them here in this menu. Here I've added some code to see what this data set is. This is the cars data set that also comes with R, and it contains 50 observations of speeding and



stopping distances for cars in the 1920s. I'm going to add this text as the title to this document, so that way it's more descriptive of my file. I'll also run the **str** command on this data frame, so that way I can see what's inside of it. It looks like I have two variables, speed and distance, which are measured in miles per hour and feet. Not only can we run R code and get the output from these files, we can also embed images. This **plot** function comes with a base graphics package in R. But don't worry learning more about it. We'll be using the **ggplot2** package which is a powerful graphics language in the coming lessons.

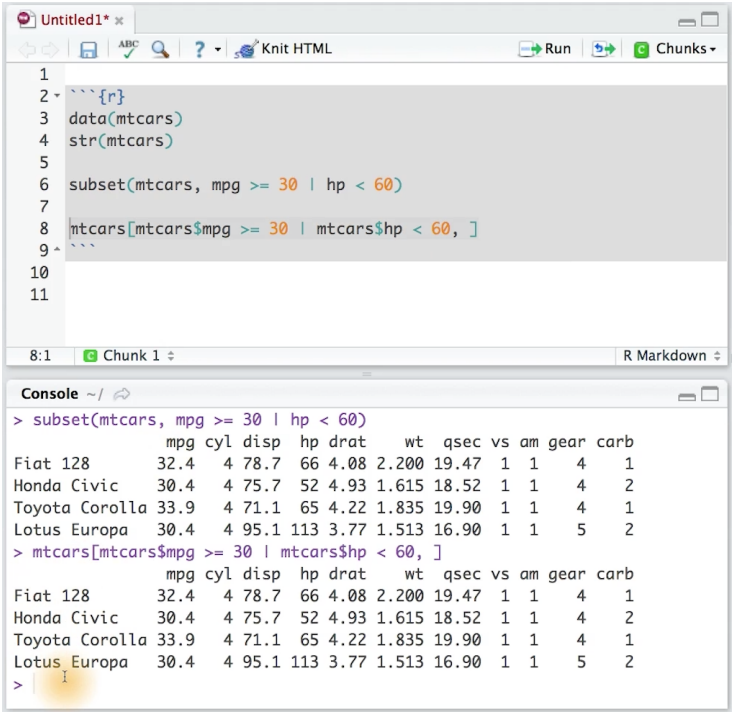
When I run this code - **plot(cars)** - I get a simple plot of speed versus distance. Now, what's really amazing about this document is that we can click the Knit HTML button. This will generate a web page that includes both content, as well as the output of any embedded R code. First, you'll need to install and load the package **knitr** in order to use the Knit HTML button. Run the commands to the right in RStudio console install and load **knitr**. This simple button allows us to easily share and publish our work. That's enough about RMD files. Let's have you work in one now. Download [the second RMD file] if you haven't already, and run and write code when you're prompted to do so. At the end of the file, you'll come across a question. I want you to answer that question for this next quiz. And then you can continue on with the rest of the lesson. Good luck and have fun.




```
install.packages('knitr', dependencies = T)
library(knitr)
```

Answer

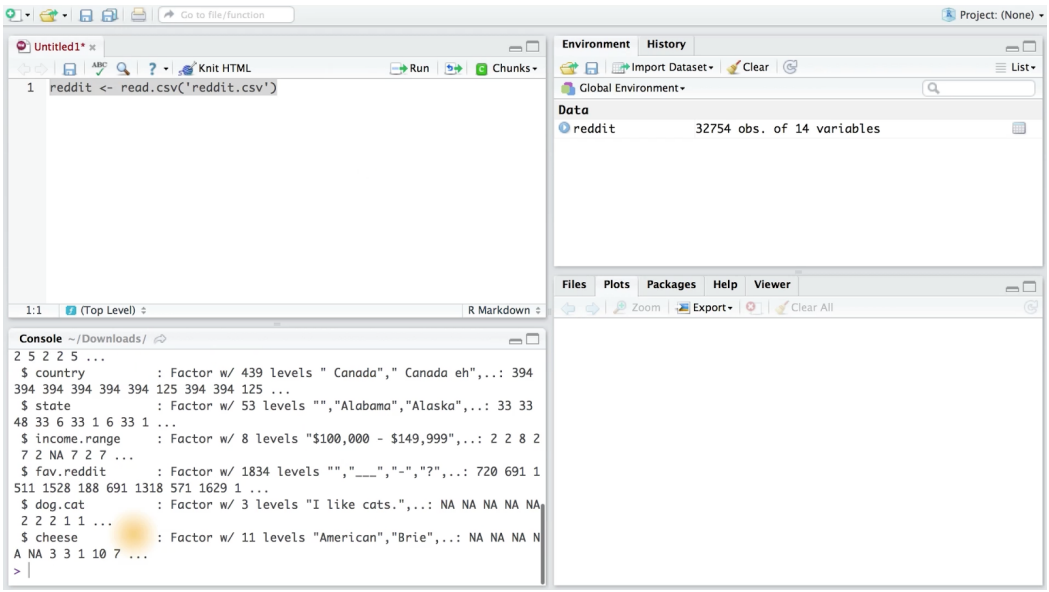
The cars that satisfy either this condition or this condition were the Fiat 128, the Honda Civic, the Lotus Europa, and the Toyota Corolla. Let's see how we can figure this out using "R". Your task was to determine which cars in the **mtcars** data set have MPG greater than or equal to 30 and an HP less than 60. To answer this question you needed to subset the data frame. I'll show you two ways of doing this. In the first method, I'll use the **subset** command on empty cars, and I want to get the cars which have an mpg greater than or equal to 30 or whose horse power is less than 60. That would be the Fiat, the Honda Civic, the Toyota Corolla, and the Lotus Europa. Using the bracket notation, the coats and tacks would look like this, and there's the same output.



Factor Variables

Now that you're familiar with the basic R commands, let's look at some more data. This time we'll be looking at data collected from a survey of Reddit users. Reddit's a social and entertainment website where users can post links and comments about trending news. This survey asks users about demographic information such as gender, age, nationality and employment status. It even asks users what type of cheese they would be. And whether they prefer dogs, cats, or turtles. Download the data set from the instructor notes and load it into R. Once you've done that, take a look at the data by using the **str(***data***)** function.

Now when I try to read in the file, sometimes I might get an error, and this is pretty common, so I would suggest looking at your current working directory to figure out the problem. Often times your directory isn't where your file is stored. Alright, so I've set my directory, and now I'm going to try this code up here again. And there we go,



there's our data. Running the **str(reddit)** command, we can see that there's lots of data here. Most of these variables have a type of factor.

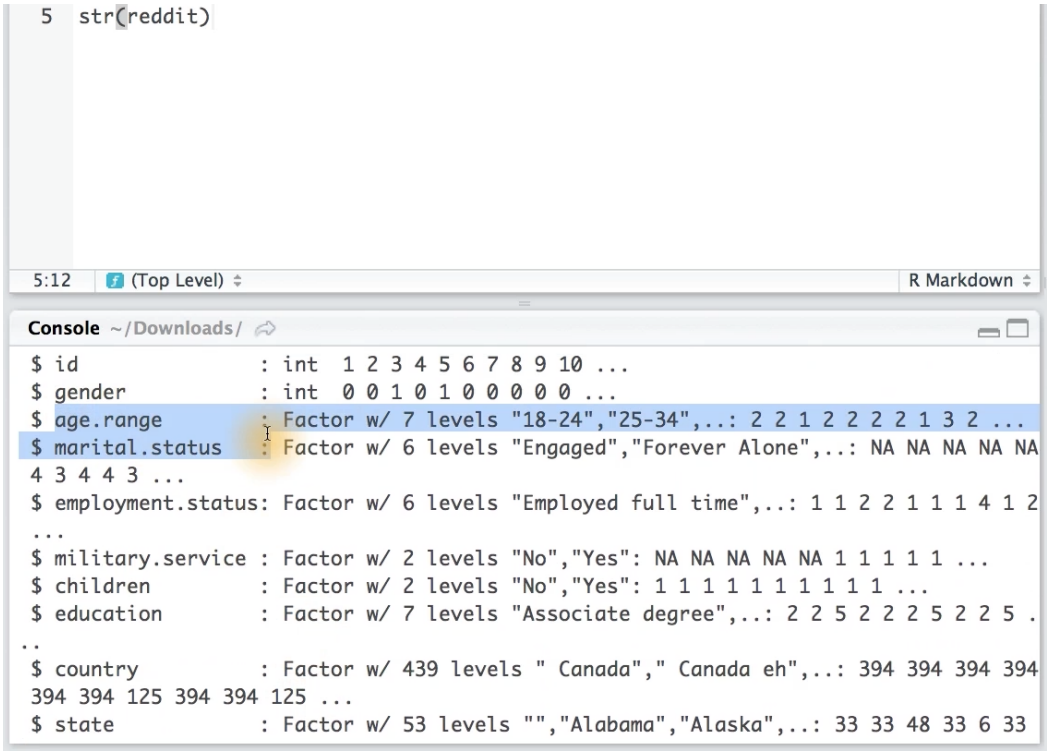
Now, a factor is a categorical variable that has different flavors or levels to it. An example of this would be employment status. This variable has many different levels such as employed full time or employed part time or not working. One thing we might be interested in is how many people are in each group of employment status. We can table that variable to see the number in each of these groups. Running this code - **table(reddit$employment.status)** - I can see the table.

```
> table(reddit$employment.status)

                   Employed full time                               Freelance
                                14814                                    1948
Not employed and not looking for work    Not employed, but looking for work
                                  682                                    2087
                              Retired                                 Student
                                   85                                   12987
```

We can also get these counts and other data points by running the **summary(reddit)** function on our data frame. I encourage you to check out the output of this for yourself. In addition to factor variables like employment status, the R programming language also has other data types like list and matrices, but we really won't be working with them in this course. We've included a reference for data types, so if you would like to learn more about those, check it out.

Ordered Factors

Let's look more closely to these factor variables. For now I want to draw your attention to the **age.range** variable right here. Notice that it says that we have a factor variable with seven different levels. We can examine the levels of a variable, by

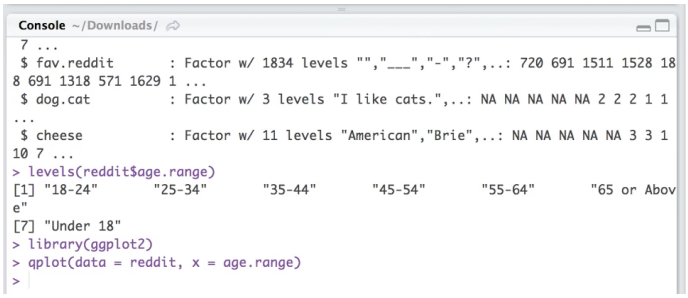


typing in the command **levels(***factor variable***)** and then putting in the variable to the argument. In the console we can see the seven levels of the **age.range** variable. Now,

```
> levels(reddit$age.range)
[1] "18-24"       "25-34"       "35-44"       "45-54"       "55-64"       "65 or Abov
e"
[7] "Under 18"
```

instead of creating a table of the **age.range** variable, let's create a plot that shows how many users are in each bin. That is, we want to figure out how many surveyed respondents are between the ages of 18 and 24, 25 and 34, and so on. I'm gonna create this plot using the **ggplot2** package, and the **qplot** function that comes with it.

Again, don't worry about understanding this code too much, we'll have practice with

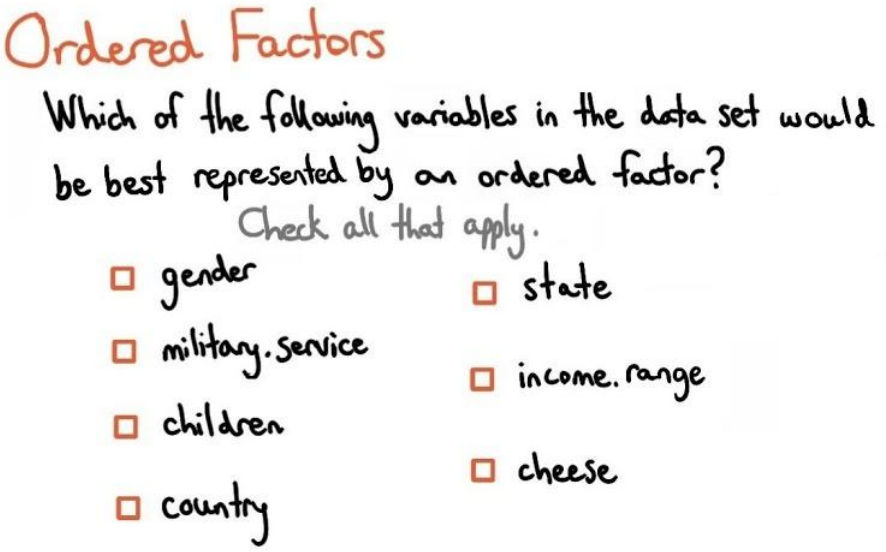


this in the next lesson. When I run this code, I get my plot over here. I want you to notice that the age groups appear to be in order. This is true for everyone except the survey takers who are under the age of 18. Now, it would be really helpful if the under 18 bar was really on the left of the 18-24 bar. That way we could make comparisons across the groups more easily. Now this is why we would want to have ordered factors.

The variable **age.range** just contains factors with seven levels, but these levels aren't arranged in any particular order. Sometimes you want to introduce order into our data set. So that way we can make more readable plots. So, knowing a little bit about ordered factors, let's see if you can answer this next question.

Quiz

If you haven't already done so, download the Reddit survey data and look at its structure. After you looked at the structure of the variables, try and answer this question. Which of these variables in the data set could also be converted to an ordered factor? Just like **age.range**. Check any of the variables that apply.

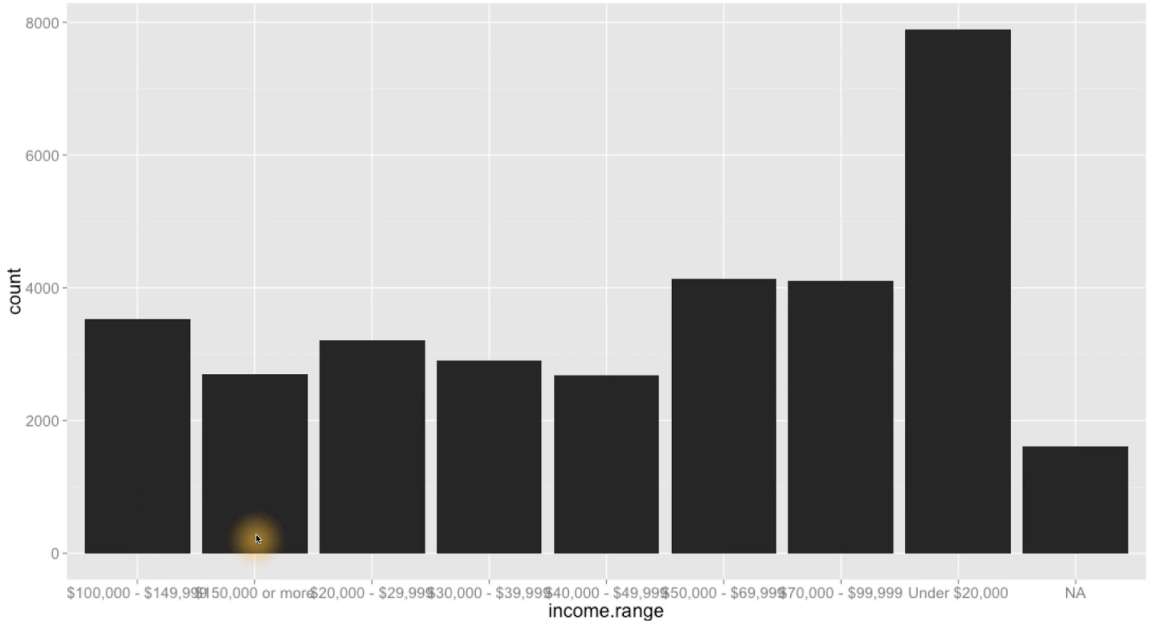


Answer

The answer here is just income level. Gender would not be considered an ordered factor, since it doesn't make sense to rank male and female groups. The military service and children variables are binary variables with yes or no values, so these really wouldn't make sense to order either. Using the same thinking, it doesn't really make sense to order the levels of country and state variables, since we wouldn't rank those either. We wouldn't really say one location is better than another. And finally, cheese falls into the same bucket as the country and state variables since they wouldn't necessarily be ranked either. The income levels actually have a natural order to them where we can group the buckets from lowest to highest, or from highest to lowest. That's what make it an ordered factor.

Setting Levels of Ordered Factors

So here's our problem. If we try to see the amount of users in each age group, it's not in order and it's hard to compare immediately. This problem is even more noticeable on a different plot. If we try to create the same plot for users in each income bracket, we see a very similar problem. And here it's much worse. These first two bins are for a 100,000 and over 150,000. And this last bin is for people who make under 20,000. Our eyes tend to read pages from left to right , so this graph is pretty hard to interpret. Or even better said, it'd be very hard to make comparisons naturally with our eye. We have to always scan down to the bottom to figure out what group we're in. So let's order the factors in our **age.range** variable.



Quiz

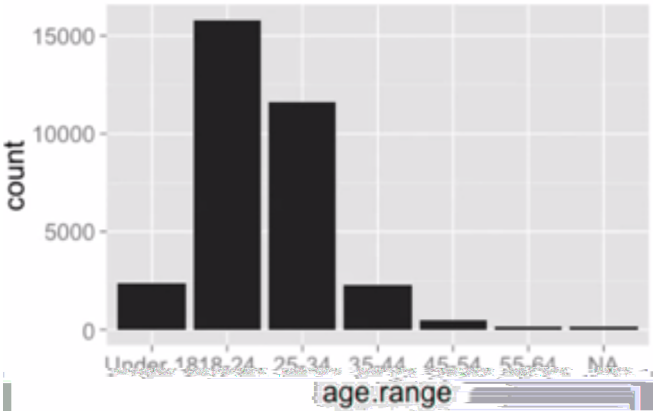
What I want you to do is to look up the documentation for the **factor** function. Or you can read through the example here. Once you're ready, try to write the code in order

to order the levels of the **age.range** variable. And just as a reminder, the level should take on the following values, 18-24, 25-34, 35-44, 45-54, 55-64, 65 and over, and under 18.

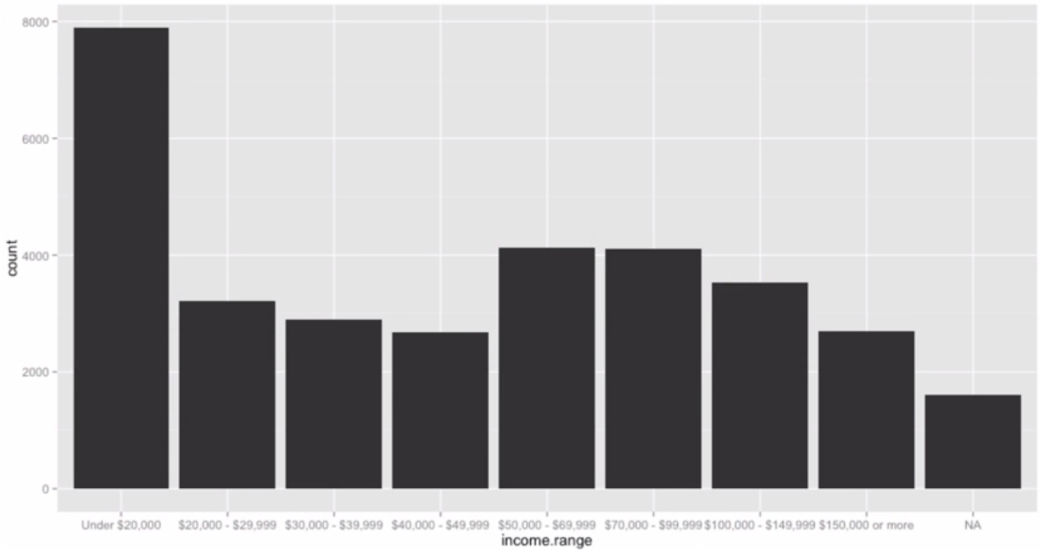
```
# Setting Levels of Ordered Factors Solution
reddit$age.range <- ordered(reddit$age.range, levels = c('Under 18', '18-24',
'25-34', '35-44', '45-54', '55-64', '65 of Above'))

# Alternate Solution
reddit$age.range <- factor(reddit$age.range, levels = c('Under 18', '18-24',
'25-34', '35-44', '45-54', '55-64', '65 of Above'), ordered = T)
```

Answer

For this programming assignment, you needed to have rearranged the levels of the **age.range** variable, so that way, under 18 appeared first. One way to do this is to use the **ordered** function on the variable. We'll use the ordered function on **age.range,** and then we'll set the levels. I can run all this code to set the levels, and then, I can make my plot again. Pretty nice, huh? Another way to achieve the same result would be to use this code. Here, we're using the factor function. We're taking our age.range variable, setting the levels, and then making the order be true since we want these ordered. I'll run this code. And then, I'll make sure that the graph is the same output. And yes, it is. Now, if you want another challenge, try reordering the levels for the income range variable, and see if you can produce this same plot.



Data Munging



It's important to note that many of the data sets that we've used so far in this course are what I would call tidy data sets. What I mean is that these data sets were

manipulated into a specified format of rows and columns. You can learn even more about tidy data here. This is what allowed us to so easily import these data sets into R. You should know that not all data sets are going to be so nice. Sometimes, you may be pulling data from different sources, like webpages, audio files, or even PDFs. Other times, you may need to reshape or rearrange your data into different formats. We're going to cover this later in lesson 5.

mtcars | reddit | statesInfo

32 observations of 11 variables

	row.names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
6	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4

The important thing to know is that this is a necessary step prior to conducting EDA and it's called data munging. There are plenty of tools for doing this kind of work, and if you're interested in learning more about how to wrangle and adjust data, check out our data munging course. Now, this course EDA won't cover data munching, but techniques for doing so are vital for any data scientist.

Advice for Data Scientists

Before we wrap up this lesson, let's hear from Eytan Bakshy and Sean Taylor, who are data scientists at Facebook. They'll offer their advice for how to work with data.

Eytan: My advice to data scientists, or future data scientists is find data sets that you're interested in, and, and work with them, and play with the data. And just developed experience with playing with data.

Sean: Good data science comes from good questions, not from fancy techniques, or from, you know, having the right data. It comes from motivating your research with an idea that you care about, and that you think other people will care about. It's almost, you almost have to think about it like a journalist would. You have this audience and they're going to consume the work that you produce, and you want them to care about it, and be interested. And so, starting with the right questions, in particular, something that motivates you as a person is really the right starting point, not, like, you know, some fancy technique that you want to learn.

Congratulations

Congratulations on finishing lesson two. In this lesson you learned how to use R Studio and you learned about the basic commands in R. If you found something particularly helpful or if you have ways that we can improve the course, let us know by posting in the forum. In the next lesson, you'll learn how to visualize and summarize single variables within a data set. We hope to see you there.

Exploratory Data Analysis

Lesson 3 Notes

Explore One Variable

Welcome!

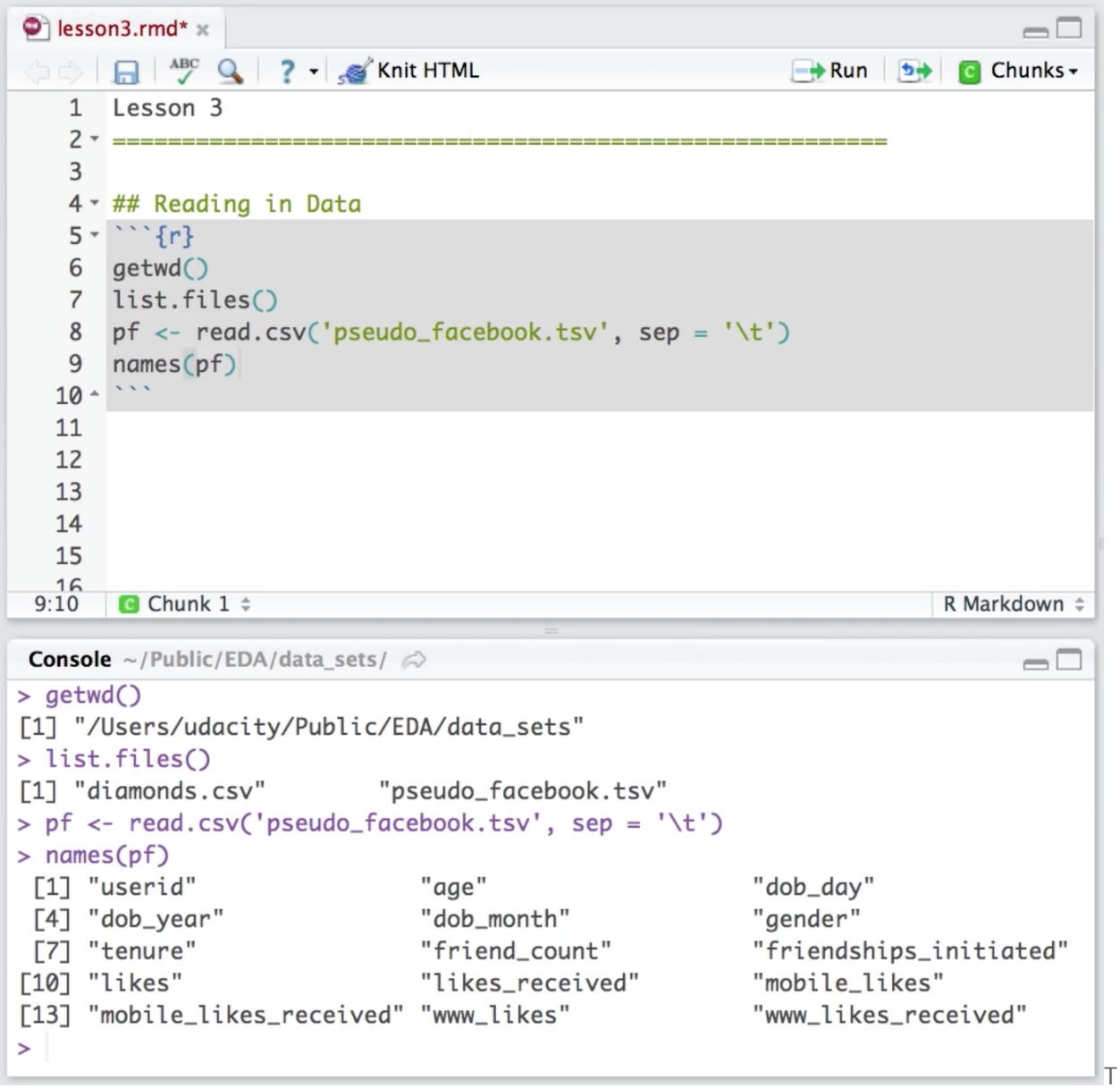
Welcome to lesson three. In this lesson, we'll learn how to investigate a single variable within a data set. You'll learn about visualizations like the box plot and the histogram and you'll learn the R syntax to create them. But, before we do that, let's hear from Dean about how he approaches looking at data.

What to Do First

Exploratory data analysis is an opportunity to let the data surprise you. Think back to the goals of your investigation. What question are you trying to answer? That question might be relatively simple or one dimensional like the comparison of two groups on a single outcome variable that you care about. Even in such a case, exploratory data analysis is an opportunity to learn about surprises in the data. Features of the data that might lead to unexpected results. It can also be an opportunity to learn about other interesting things that are going on in your data. What should you do first? Well certainly you want to understand the variables that are most central to your analysis, often, this is going to take the form of producing summaries and visualizations of those individual variables.

Pseudo-Facebook User Data

Thanks Dean. Throughout this lesson and others, Dean will give us more advice and discuss the big picture when doing EDA. For now, let's start by loading up some data. You can find a data file in the instructor notes. This data file contains all the Facebook data that we're going to be looking at. To load up the data, let's first make sure that we're in the right directory. Here is my current directory and I can see that I'm in a folder that has the data sets. I can use the list.files command to figure out what that files are within this directory and here is the data set that I want.

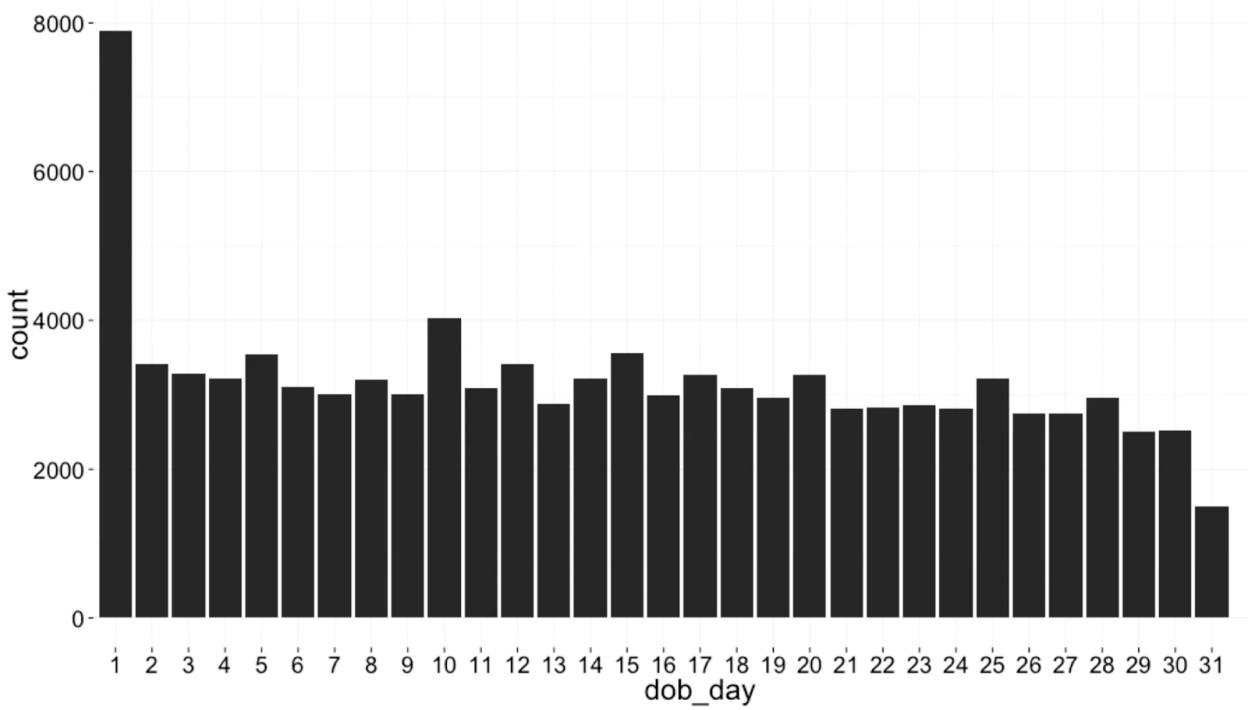


The pseudo_facebook data set. To read in the data set I'm going to use the read.csv command. But I need to be a little bit careful. I need to indicate that the separator is really a tab. And that's because we're working with a tab separated values file. Running the command, I can see that I get my data set up in R. Now this data set is similar to the kind that data scientists see from time to time at Facebook. And it's especially the type of data that they'll use when doing EDA. This file has approximately 99,000 rows or observations in it with 15 variables. Each observation represents a user and that user will have different information such as their age, their name or their date of birth. We can see all these variables by running the names command. Now in this entire process our goal is to understand our user's behavior and their demographics. We're going to want to understand what they're doing on Facebook and what they use. This is why you see things like friend_count, www_likes and mobile_likes. We created this data set specifically for this course to have many features common to such user level

behavioral and demographic data. But I need to say that this data isn't actual Facebook data. So not all statistics in this data set are going to be accurate representations of how people use Facebook on a day to day basis. So our lawyers want me to make the disclaimer that this isn't actual data from Facebook users. But we did use a complex model to generate it, and I think we can learn some very interesting things about it.

Programming Quiz: Histogram of Users Birthdays

Let's start by looking at birthdays using ggplot. Ggplot is a graphics package that allows us to make visualizations. We're using ggplot since it's easier than the base graphics that comes with R. It has some nice features, such as the format of plots and legends that are automatically generated. The first thing we need to do is to download and install the ggplot library. Now that we have got the ggplot loaded, let's use the qplot function to plot a histogram showing the number of users whose birthdays fall on any given day. I can run their name pf command to get an output of all my variables. Now, I'm trying to create the date of birth by day histogram for all my users. So, I'm going to use the date of birth day variable, here. For the qplot function, I'll pass it two parameters, x and data. X is going to take the variable of data birthday, and data's going to take the variable pf, which is where all my data comes from, from the pseudo Facebook dataset. Running this command, I see my histogram come out in a plot window below.



Now, you might notice that this plot is slightly different in color. If you're curious about setting color and themes in R, see the instructor notes. Getting back our code in console, you can see that we get this warning message, when we ran the code. This is fine for now but, you should think about what this warning message means. So, here's our histogram so far but, I want to add one more layer to this

plot. Let's fix the label on the x axis, so we can see every day of the month here. So to adjust the x axis, I'm going to add a + sign right after my code and then hit Return. This immediately takes me down the next line with a little bit of indentation right here. I'm going to add the layer scale x discreet and then give it the parameter breaks from 1 to 31. This is because I want all my days to be 1 to 31, the days of the month. And running this code, we can see we've done a little bit better. What are some things that you notice about this histogram?

Answer:

Here're some things that I noticed. On the first day of the month I see this huge bin of almost 8,000 people. This seems really unusual since I would expect most people to have the same number of birthday's across every day of the month. Now this would be true of course except for the 31st. Not all months have 31 days so the lower number of people in this bin makes a lot of sense.

Moiras Investigation

As you continue learning about EDA in the next lessons, you'll get to hear from Moira. Moira's done some exciting work at Facebook and she'll be sharing one of her projects with you. Let's hear from her.

So one of the things that I've been looking at lately is whether people's perception of their audience on Facebook matches up to the reality. Who's actually seeing the content that they're sharing. Because who you think is in your audience really affects how you present yourself. So to investigate this problem, I worked with Michael Bernstein, Brian Carr, and Eytan Bakshy. And together we ran a survey where we pointed people to a post that they'd made in the last couple of weeks, and we said, how many people do you think saw this post? And then we took their responses to that question and we actually counted how many people had viewed that post for about a second at least and we found that there was a pretty big mismatch between people's perceived audience size and their actual audience size.

Programming Quiz: Estimating Your Audience Size

Moira just described the investigation where she estimated people's audience size. I want you to think about a time when you posted a specific message or shared a photo on Facebook. What was that message, and how many of your friends do you think actually saw that exact post? And finally, what percent of your friends do you think saw the post? The results of Moria's investigation will be revealed a little bit later in the course. But just so you know, there's no right or wrong answers for these. Keep your guesses in mind, and then you can compare them to the results that Moira found in her study. Write about your message or your post here and then enter two numbers here and here. And there's

no need to include a percent sign.

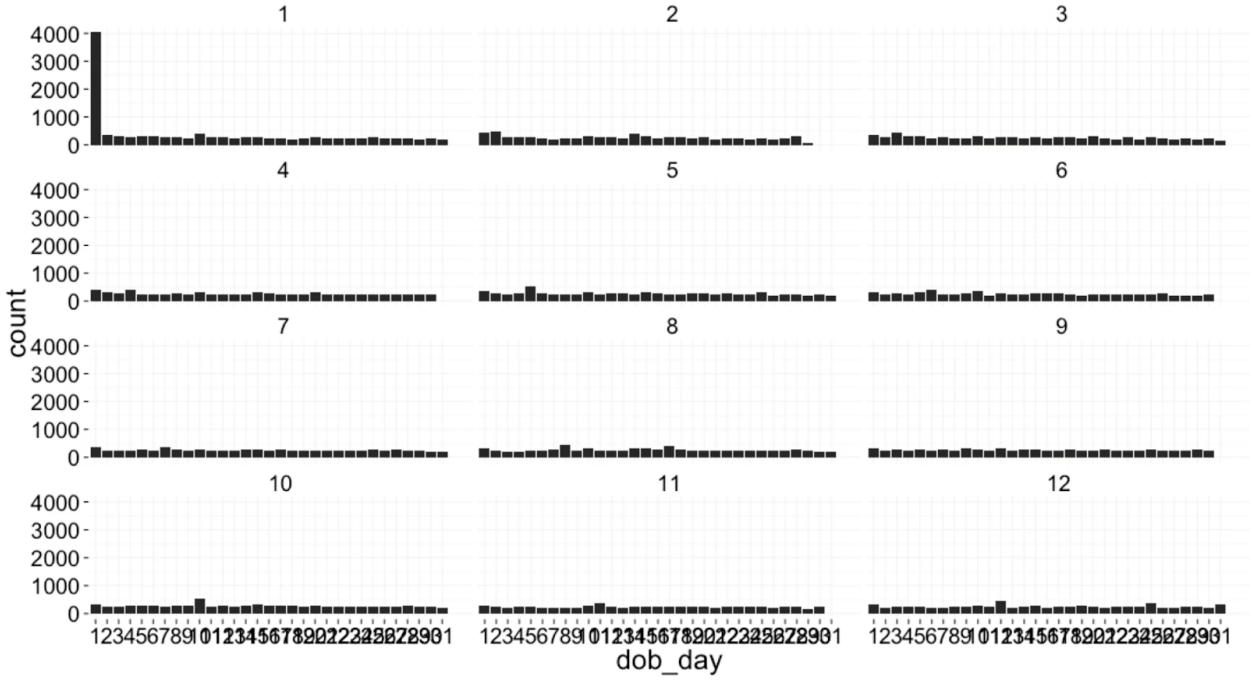
Perceived Audience Size

What we found, was that people dramatically underestimated the size of their audience. Typically, they guessed that their audience was about a quarter of the size that it actually was. When I first approached this problem, I, plotted a bunch of histograms. I wanted to know how many people do people think are in their audience. So, the first thing I did was, you know, I opened up R, and in ggplot, I plotted a histogram of people's guesses.

Programming Quiz: Faceting

Cool. It looks like Moira got started the same way as us when we investigated birthdays. Lets make more sense of our data, especially for this bin of people born on the first. We can break this histogram into twelve histograms, one for each month of the year. The code would look like this. First, I'm going to add a layer and that layer is going to be called facet wrap. Facet wrap takes a formula where we have a tilde, and then we're going to use
the variable that we're going to split our data over. In this case, DOB month. We have the number of columns set equal to three. And then we can run this code and see the output. So notice, we took our one histogram and split it into 12 histograms. Since we set ncol as the number of columns equals to three, we can see the three columns in our one plot. Now, this one stands for January. This two would be for February, and so on. And if I wanted to, I could have used ncol equals four. It would have just given me a slightly different plot with four columns. Before we keep coding anymore, I want to focus on this facet wrap. And specifically, this formula right here. in general, facet wrap takes in a formula inside of its parentheses. The formula contains a tilde sign followed by the name of the variable that you want to facet over. This allows you to create the same type of plot for each level of your categorical variable. In our case, we wanted to make histograms of dob day, one for each month of the year. A similar layer to facet wrap is facet grid. Facet grid also takes a formula, but it's in a little bit

d ifferent form.



It's formula contains variables that we want to split over in the vertical direction, followed by a tilde sign, and then the name of the variables we want to split in the horizontal direction. In general, where you place the variables can change how the graph is laid out, and the orientation of them. Now, facet grid is generally more useful to use when you're passing over two or more variables. If it's just one I would use facet wrap. You can learn more about facet wrap, facet grid, and how to format plots by following the link in the instructor notes.

facet_wrap(formula)

facet_wrap (~ variable)

ved.

Answer:

Now, you may have noticed some peaks in May or perhaps in October, but I think what's really interesting is this huge spike on January first. There's almost 4,000 users in this bin. Now, this could be because of the default settings that Facebook uses or perhaps users are choosing the first choice in the drop down menus. Another idea is that some users may want to protect their privacy and so they just go with January first by default. Whatever the case may be, I think it's important that we make our considerations in the context of our data. We want to look out for these types of anomalies.

Be Skeptical Outliers and Anomalies

People often talk about the importance of detecting and dealing with outliers in your data. But there are many types of outliers and anomalies and how your analysis proceeds should depend on what type you're dealing with. Outliers can have many causes. For example, an outlier might be accurate data about an extreme case. For example, someone represented in your data set might really be tweeting 1,000 times a day. On the other hand, sometimes outliers, or anomalies represent bad data, or the limitations of your data. For example, what otherwise would be a normal value of a variable might be replaced with an extreme value. Or, in other cases, extreme values might be replaced with a more normal value. For example, in a lot of census data or surveys, income information is top coded. So individuals with very large incomes have their incomes replaced with some other value. Let's get back to Moira's story about perceived audience size, and hear about a particular anomaly that popped up there.

Quiz: Moira's Outlier

And the first thing you get is this really terrible plot with one big, tall bar. And that's because most people guessed a kind of small number, but at least one person guessed 10 million people saw my post which was not true. The first thing I needed to do was adjust the axes so that I could actually see the bulk of the data. So, I cut this one outlier out. And then, I looked basically at people who guessed a few thousand or, or below. Then, then you get a slightly better histogram by adjusting the axes. And, you can still see that the bulk of the data was in the smaller ranges, but it still was kind of hard to see how large people thought their audiences were.

Moira just explained how she encountered an outlier in her data. And I want you to think about what type of outlier do you think it was. So, try answering this question. Do you think Moira's outlier was an example of bad data about a non-extreme case, bad data about an extreme case, or was it good data about an extreme case? Choose the best answer here.

[ ] Bad data about a non-extreme case
[x] Bad data about an extreme case
[ ] good data about an extreme case

Answer:

In this case, the outlier was an example of bad data about an extreme case. The respondent in Moira's study said that they had over 1 million friends. Moira's outlier had a value of 10 million, which isn't possible using Facebook. So that would be an example of bad data in an extreme case since we're at the higher end of people who saw the post. In general, I think it's really important that we think about What type of case or categories that our outliers fall into. This may help us think about how we can adjust for the outliers, or replace them or exclude them for our data, if we want to make those decisions.

Programming Quiz: Friend Count

As Moira mentioned, we may need to adjust our axes to get a better look at the data. Let's see an example of this using our Facebook data set. I want you to create a histogram of friend counts. Try this in "R" and then paste your code in the box. Once you've got your plot, I want you to think about how this plot is similar to Moira's first plot.

Answer:

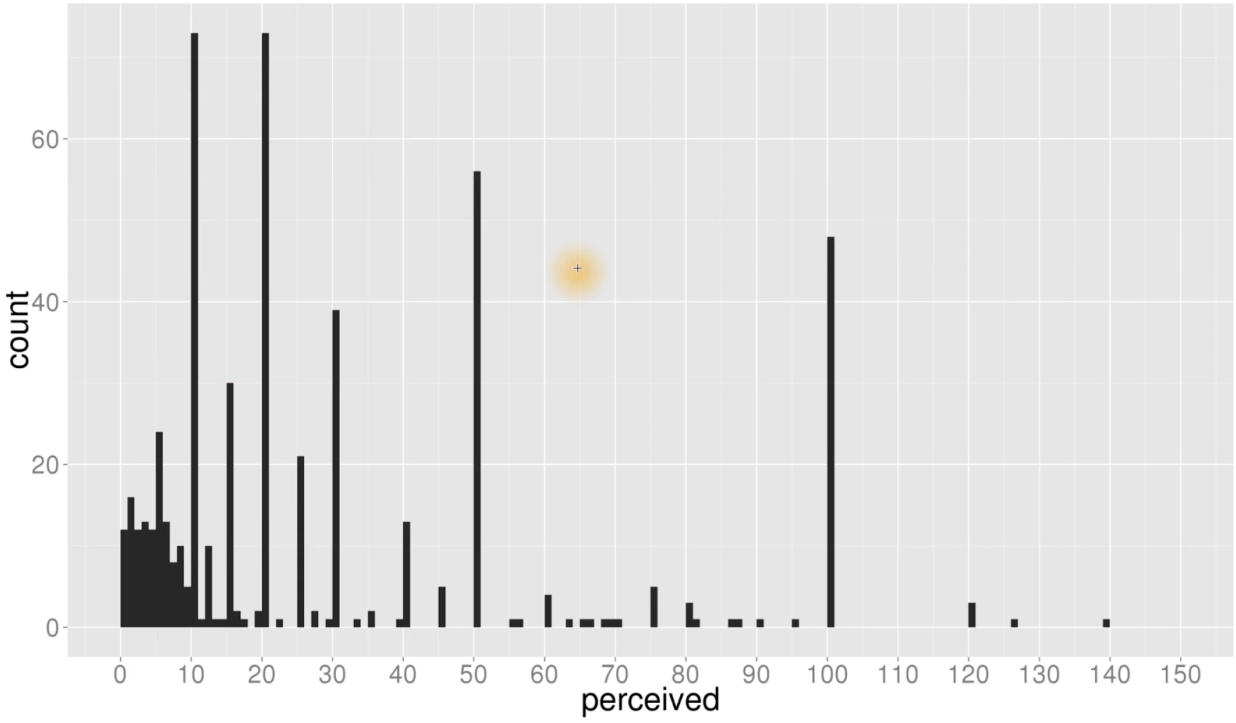
To create the histogram, we'll pass friend count to x inside of qplot, and we'll pass pf into data. When I run the code, I can get my histogram. Looking at the plot, we can see that the data is squished on the left side of the graph, just like Moira's plot, and this graph extends all the way to the 5000 mark. This is what we call long tail data. And this data can be common for some user-level data. Most users have friend counts under 500, so we get really tall bins on the left. But there are a few users in our dataset with really high values. The higher values are closer to 5,000, which is the maximum number of friends a user can have. Moira was interested in examining the bulk of her data. And, it's the same for us. While, we may want to investigate some observations in this tail, we really want to examine our users with friend counts well below 1,000. All of these people here. To do that, we'll need to learn something else, in order to adjust our code, and our plot.

Limiting the Axes

To avoid looking at all this long tail data, we can use the xlim parameter inside of qplot. This parameter takes a vector where we have the start position and the ending position of our axes. In our case, we want to examine friend counts less than 1000. Running the code, I can see that I have my plot with friend counts less than 1000. Now there is another way to create the same plot. Here I won't use the xlim parameter. Instead I'm going to add a layer. I use limits instead of xlim as the parameter inside of here since I know this adjustment is already for the x axis. That's why its called scale x continuous. There's also a counterpart for the y axis called scale y continuous. One of the neat concepts of ggplot is that you can build up your plot in layers. We're going to discuss layers later in this lesson, but for now I'm going to keep using the qplot syntax. So far we've learned how to create histograms, how to facet them, and how to adjust our axes. But up until now all of our histograms have had a default setting. We've been getting this error message for a while about the bin width defaulting to a range over 30. It turns out we can set the bin width ourselves to reveal interesting trends in the data. Let's hear what happened during Moira's investigation when she altered the bin width of her histogram.

Exploring with Bin Width

When you adjust the bin width, and here the bin width is set to one, you really see these dramatic patterns, these tall vertical lines. This is because when you say, how many people do you think saw this post, people typically say, oh, 10, 20, 50, 100.

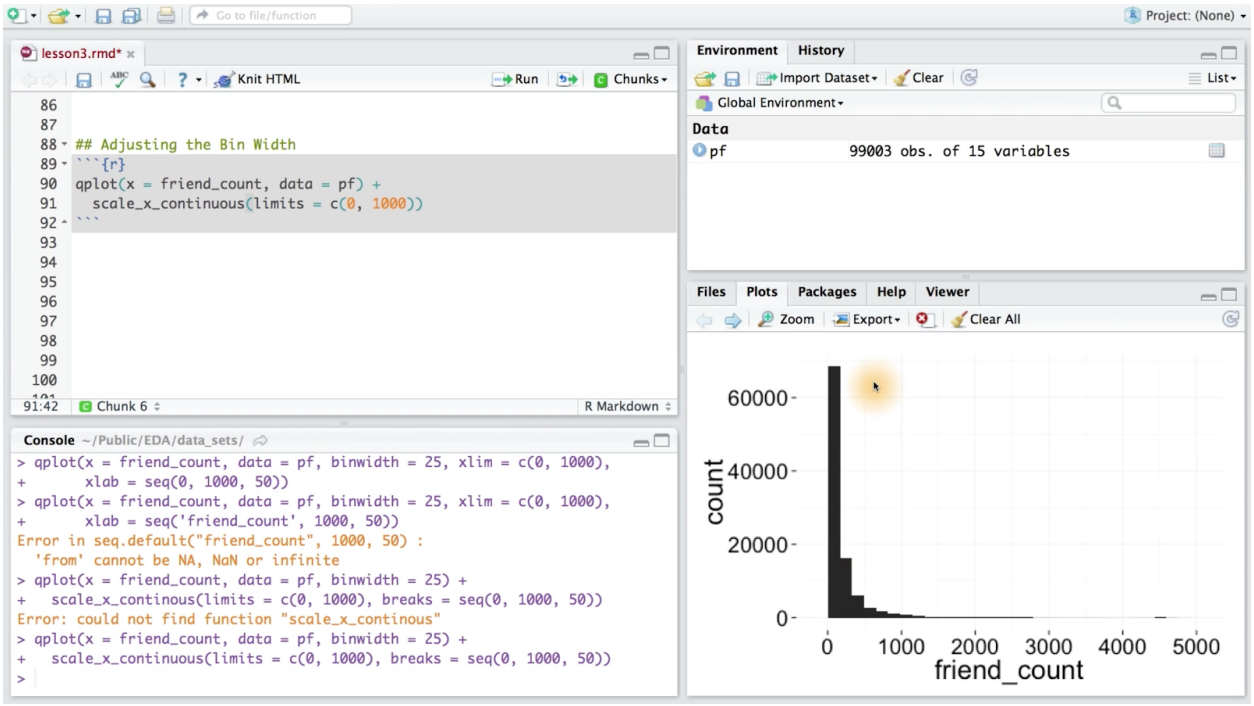


People guess these regular numbers, and they don't tend to guess numbers in between. So this really helped me get a feel for how big people thought their audience size was. The most common guess

was I think 20 with 50 and 100 close behind. In most cases though, this was only about a quarter the size of the actual audience.

Adjusting the Bin Width

Now that we've heard from Moira about the benefits of altering the bin width, let's try it for ourselves. Looking at the histogram we created earlier, we can see that it's skewed. We've just zoomed in on the left side of the graph from the histogram from before. Working from his new histogram, let's add some better labels and some binning. To adjust the bin width, I can pass the bin width parameter to qplot. And for this data, I'm going to choose a value of 25. That seems reasonable since we are going from 0 to 1000. The other thing that I want to do is break up this x axis every say 50 units. To do that, I'm going to pass the breaks parameter to our scale to x scale continuous layer. Breaks is going to take a sequence that has a starting point, an ending point, and a step or the interval space we want on our axis. So in this case, I'm going from zero to a 1000, and I want to step every 50 units. You can learn more about how to manipulate the scale and the breaks on the x or y axis by reading the link in the instructor notes.



When I run the code I can see that I get a completely different histogram. Taking a closer look, it's easy to see that many users have less than 25 friends. These users are probably new. Now let's ask a more interesting question. Which gender group on average has more friends: men or women? To answer this question we need to split this histogram into two parts for males and females. Let's see if you can do that.

Faceting Friend Count

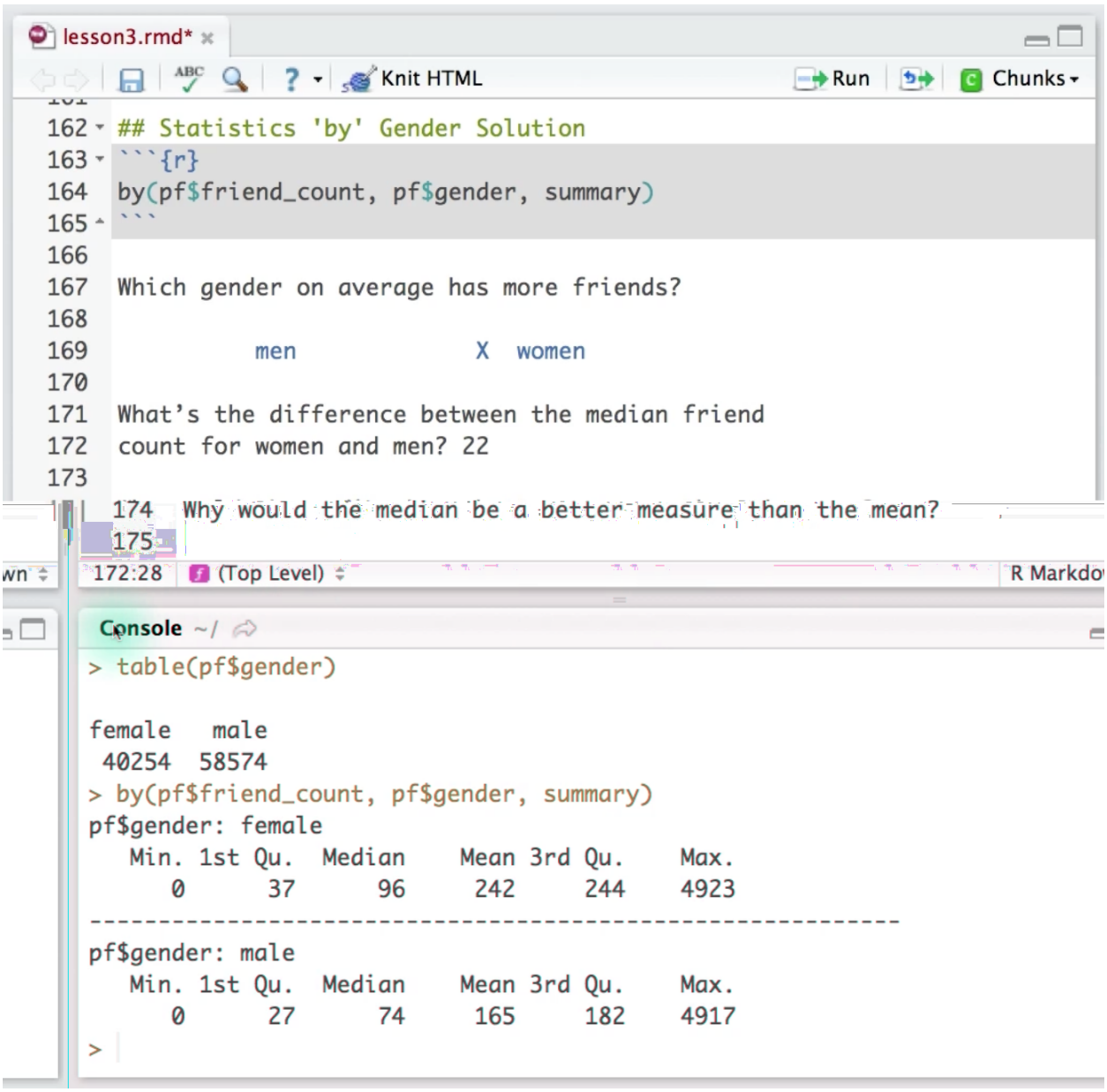
So, to start answering our question of who has more friends, men or women, on average, let's take our histogram and split it up into two histograms for males and females. We do this by adding a facet wrap as another layer. This time, our facet wrap is going to take a formula where we have the tilde sign and then gender as our splitting variable, and when we run this code as you might find out, we get something quite unexpected. We don't just have two histograms, we have three. One for females, one for males, and one for NA?

Omitting NA Observations

So, we made these three histograms, and let's take a closer look at them. We have one for females, one for males, and one for NA. NA stands for Not Applicable. And the third histogram we just created contains users who are not labeled as male or female. This is how R handles missing values. Missing values take on a value of NA. Let's try creating our histogram for friend counts split by gender one more time. But this time we'll take a subset of our data and ignore the observations where the gender is NA. To do that, I'm going to subset our data. I can use the subset command and use a condition for the second parameter. The first parameter is just the data set. Here, I'm using the is.NA function with a bang (!). This removes any rows that have NA for the gender variable. So if I run this code this time I shouldn't see the NA and fixing a small typo I can see I have my two histograms for females and males. Now I do want to point out there's another way that we could have subset the data. If, instead, I use the na.omit function, I would remove any observations that have NA in them, not necessarily just for gender. You want to be careful when using the na.omit wrapper since we might end up ignoring some users that are labeled as male or female...but, they might have different variables that do have NA values. Say for like friend_count or something else.

Programming Quiz: Statistics by Gender

Let's take a closer look at these two plots. Now, it might be hard to determine which gender has more friends on average. Remember, that's the question we're investigating. Who has more friends, males or females? So, instead of looking at these histograms, we can use the table command to see if there are more men versus women. When I run that command, I get this output. So, it looks like there's slightly more males than females in our sample. To look at the average friend count by gender, we'll need another new command. This command is the 'by' command. The 'by' command takes three inputs: a variable, a categorical variable, or a list of indices to subset over, and a function. In our case, we want to use the Summary as the function to get basic statistics on our friend count. So, friend count's the variable, gender is our categorical variable, or the variable that contains our segments of users. And, we want a summary of the friend count by gender. Now, I'm not going to run this code, but I want you to run it in R Studio and then answer these next questions.



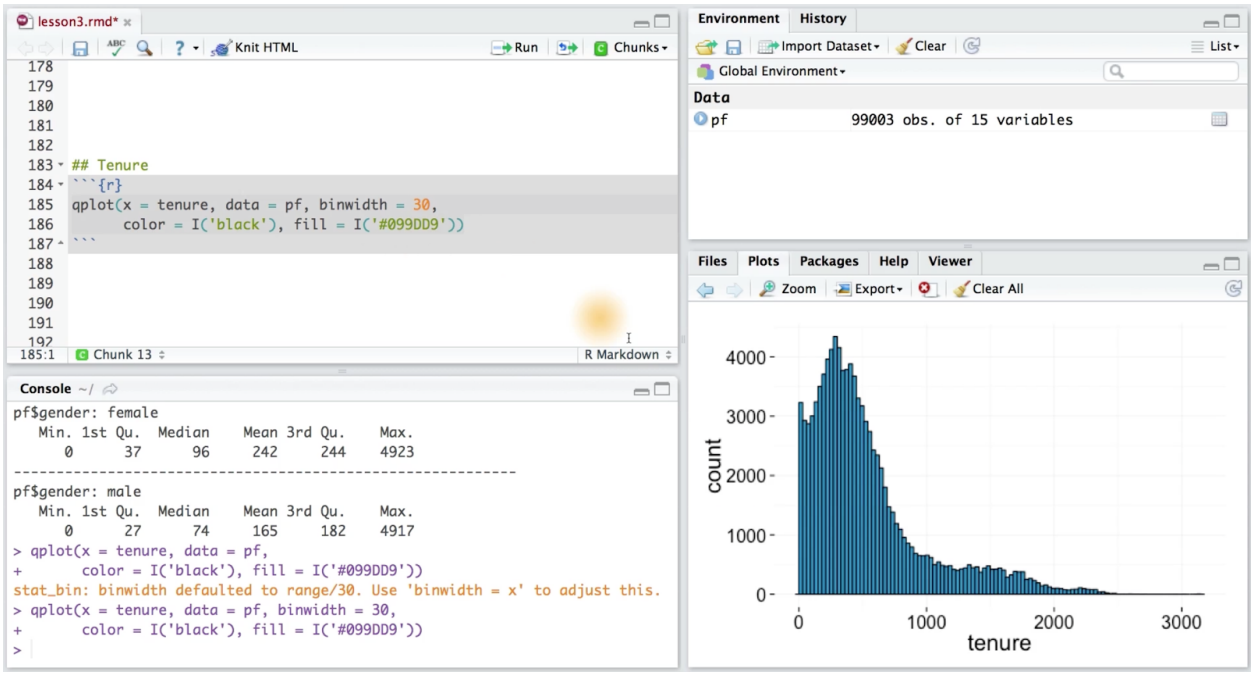
Answer:

When we run the code, we get this output. And notice how we have friend count split by females, and then by males. And it looks like the median and the mean friend count is greater for females. So here the answer was women. Now the difference between the median friend count for men and women is 22. That's just subtracting these two numbers. Now, something I want to draw your attention to is the mean Notice how the mean for friend count is higher than the median for both females and for males. This should make sense since our data is long tailed. So these higher values out and the higher friend counts our kind of pull our mean to the right. For this last question, the median is a better measure

than mean because it's a more robust statistic. A few people with huge friend counts drag the mean upwards which isn't necessarily representative of most users. What's nice is that the median's resistant to change, since it marks the halfway point for all data points. So as long as we trust half of our values, we can report a reliable location of the center of our distribution.

Programming Quiz: Tenure

Let's continue looking at some other variables so we can better understand our users in our sample data set. This time I'm going to examine the distribution of tenure, or how many days someone has been using Facebook. I'm also going to start introducing color to jazz up our plots a little bit. To do that I'm going to use the color and the fill parameter. Now, I'm not going to explain this code in detail. Check out the instructor notes for an explanation of this code. There's also a link to ggplot's themes to learn about all the different adjustments besides color that you can make. Now, if this code has thrown you for a loop, I don't want you to focus on it too much, you can really just ignore it. I'll typically include it as the second line of parameters inside of qplot. Running this code, you can see we get a nice histogram. And again, here the histogram has defaulted to the automatic bin width. I'm going to make one last adjustment and set it for ourselves. By setting the bin width equal to 30, we can see we get a finer view of the distribution. Let's see if you have a handle on histograms and bin width. We're going to see if you can change the distribution in this next programming exercise. In it, you're going to be creating a histogram of tenured that's measured in years, rather than in days. You'll also want to think about an appropriate bin width for the data. 30 here makes sense, since I'm measuring days, and 30 days is about a month.

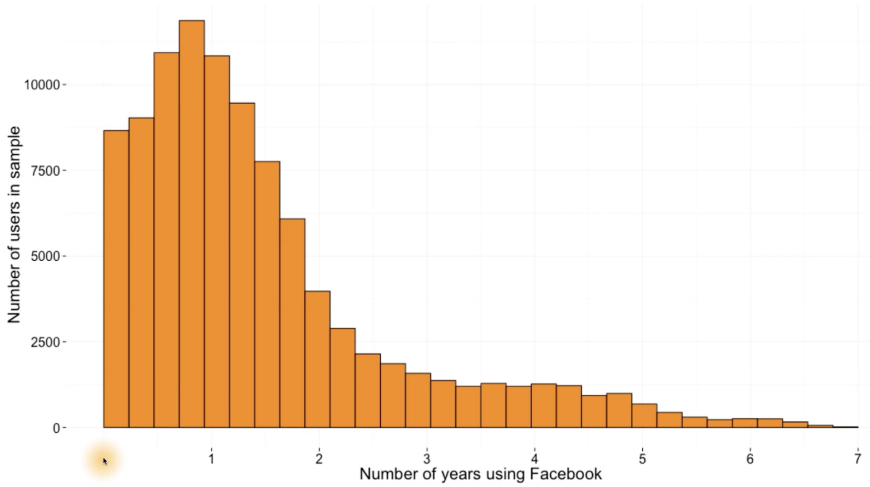


Answer:

For this programming exercise, you need to alter the variable tenure, so that way, it was measured in years instead of days. And this should make a lot of sense, since 365 days would be one year, and 730 days would be two years. Now for the bin width, let's set it equal to one so we get a count for yearly users. Running this code, we can see our histogram. And notice here that we have a different color. And I did that by changing the hex code. To get a finer view of our data, we could set the bandwidth equal to .25. It looks like the bulk of our users had had less than two and a half years on Facebook. To improve this plot in one more way, I can change the x axis, so that way it increments by one year. To do that, I can add the layer of scale x continuous, and set the breaks from one to seven. I'm also going to limit our data so that we only see users from zero to seven years. Writing this code, we can see we have a very nice histogram.

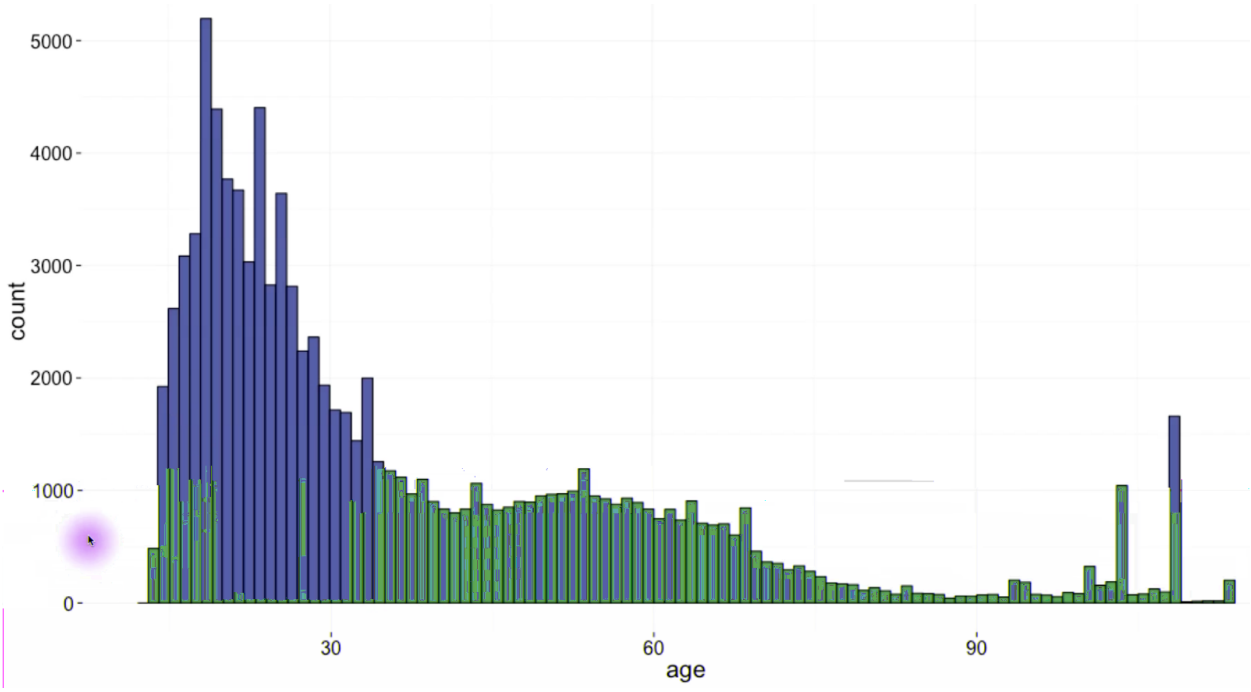
Labeling Plots

We haven't talked about this yet, but you do want plots to speak for themselves. If you look back at some of the plots we've created, you'll notice that the labels on the x axis and the y axis are automatically generated. By default, ggplot2 uses the variable names for the labels. That's why we can see that tenure divided by 365 appears right here. We pass that to x, so that's going to appear in our graph. Now, this might not be the best label for the x axis, so let's change this. To do so, we can use the xlab and ylab parameters. Now, making sure I have a comma after each of these new parameters, I can run this code and see the labels on their plot. Yeah, that's a lot better. Now, your x-axis might look slightly different from mine, and that's because I added scale-x continuous as a layer, and I made the breaks go from zero to seven. That's what this bit of code does here. Now, when you conduct EDA, your plots don't need to be perfect. I don't want you to stress about formatting, but do focus on making sensible choices for scales and limits on each axis. When you return to your plots and to your code later, labels can function as comments about what the code was intended to do. Likewise, if you were to share a plot with a colleague, labels can make your plots more understandable.



Programming Quiz: User Ages

Lets try another plotting exercise. This time, I want you to create a histogram of Facebook users based on their age. You should play around with the bin width the breaks and the labels. The grader will accept any answer here, but I want you to compare your code and your observations of your histogram to those in the solution video. After you create your histogram, think about what you notice and remember that you can create a couple plots so don't just stick to one. Figure out which pot reveals the most information to you.



Answer:

To create our histogram of user ages, let's start by using the Q plot function. We'll pass it the variable age. And our data set is pseudo Facebook so pf. And I'm going to add some color using the color and fill parameters. Running this code we see we get our histogram. Now you might have tried a variety of bin widths. And in this case I think a bin width of one is the best. So let's add that in here. I chose a bin width of one since we're going by years or ages. This gives me a histogram of counts of users by individual years, so I can get a finer view of the data for say, like, 25 years old. By setting the bin width equal to one, it allows us to more easily spot unusual spikes in our data. Points like here, here, and definitely out here. Let's zoom in on this plot and take a closer look. I see the largest spikes in the mid to late 20's. And we don't have any users who put in an age lower than 13 at the left of the histogram. This makes sense because users must be at least some unusual spikes above 100, but my guess is that some users may be exaggerating their age. Do you have any friends who that are 100

years or older on Facebook? I know I certainly don't. And one final comment that I want to make is that your x-axis might look slightly different than mine. Remember that we can use scale x discrete or scale x continuous to change this appearance. So I adding scale x discrete here, I can set the breaks so they run from 0 to 113, and then increment every five units, or every five years. Now you might be wondering why I chose 113 here, and I have a simple answer for you. I just used the maximum age in our user data set. Remember I can find that out by running a summary on that variable. So I can see the minimum age is 13, and the maximum age is 113.

The Spread of Memes

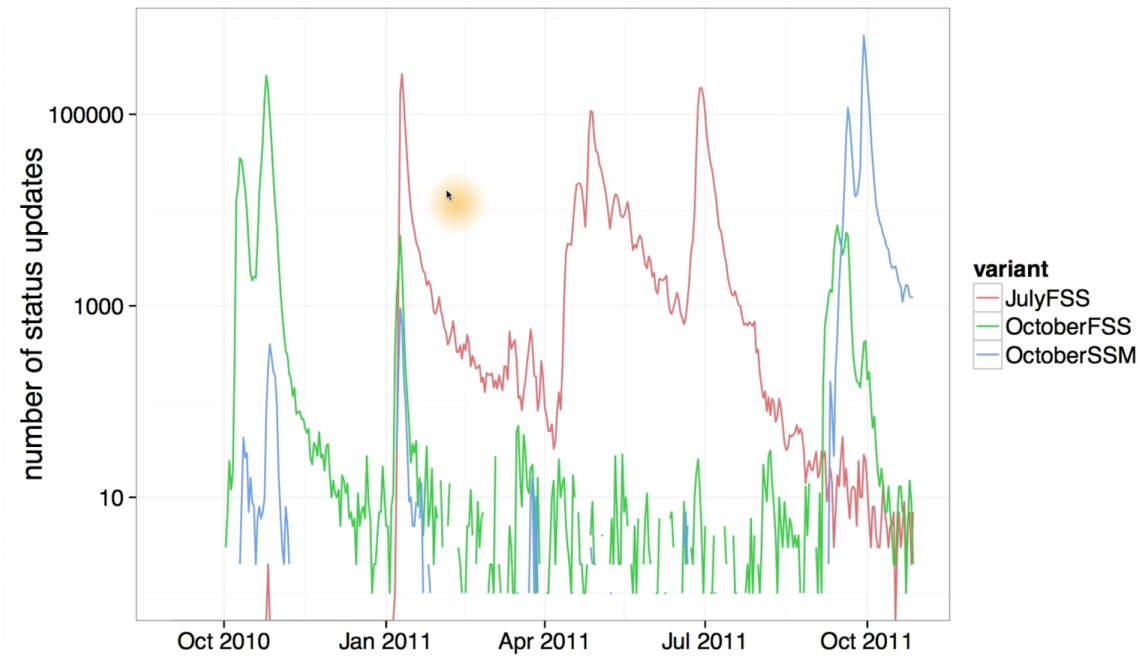
Now there are other changes that we can make besides bin width. I spoke to Lada who's a Facebook data scientist and she's been studying memes. Let's hear from her about how she made some adjustments to plots during her exploration.

Ladas Money Bag Meme

In terms of your work at Facebook, what are some recent projects that you've been working on?

So, one thing I'm really fascinated by is how information flows through networks. So, in my early work, this was just how web pages linked together. But, with online social networks being such a prevalent means of communicating these days there's a lot of transmission from individual to individual or from individual to their social network. And, there are these things called memes which are ideas that seemed to almost take advantage of this to replicate themselves. So, some Facebook status updates, for example, will have a little piece of text that says copy and paste or repost or post this so that all your friends will see if you agree with this statement. One interesting thing about memes is that they tend to recur over time. So, sometimes you see a meme and it's like oh, yeah, I saw that one a year ago and now it's back and how does it come back? So, we looked at one particular meme. This is the Moneybags meme, which says that you know, this year, October has five Fridays, Saturdays, Sundays. This happens only once every 823 years and this is a lucky occurrence, blah, blah, blah and anyone who actually looks it up, sees that this kind of

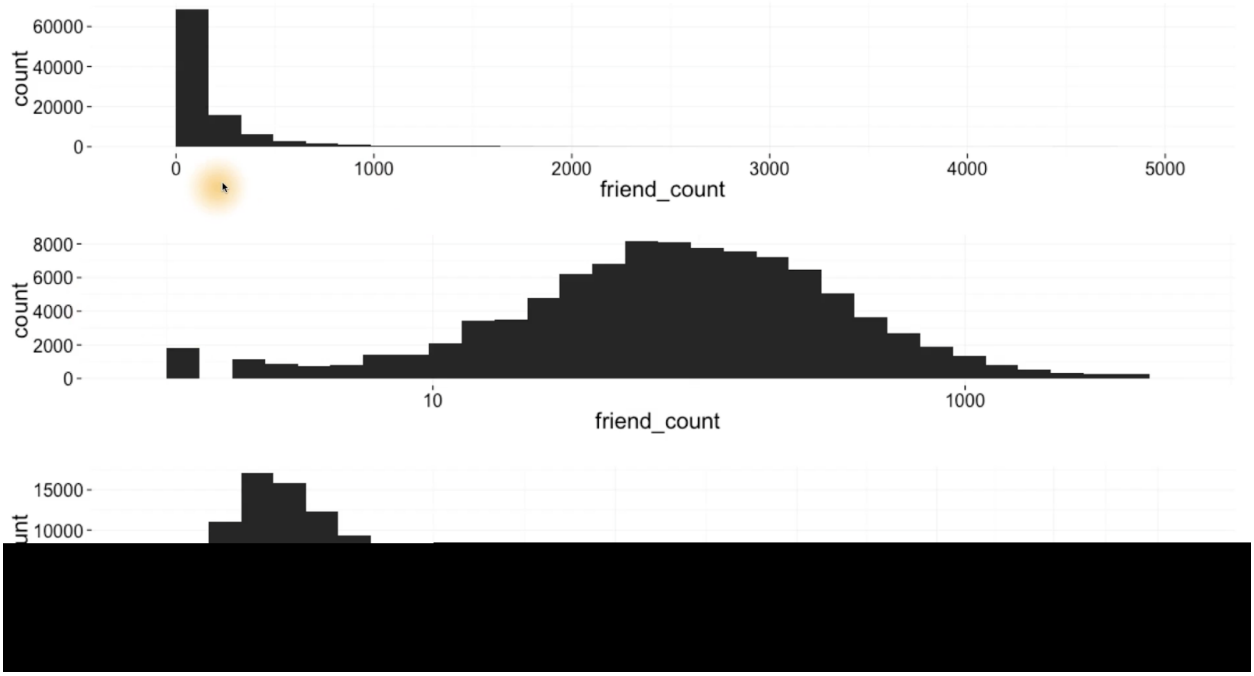
occurrence is much more frequent and in fact, you'll have different variants of the meme. For example, here in red, you have the one that says, July has five Fridays, Saturdays, Sundays. And this is true in 2011 and if you just look at this plot, which will, on a linear scale, show you the counts for this meme, it may appear as though it just disappears completely in between. So, you might think, well, maybe it's being introduced from some other source and not Facebook. But the weird thing is that the meme is adapted to Facebook. It says, copy and paste, which doesn't make so much sense for other media. So, is it possible that the meme is actually staying within Facebook but in very low numbers, and then flares up? And so if you, instead of using a linear scale, use a log scale here at the same time, you can see counts that are 100,000 and counts that are 10. And so here when you look, you can see that even though there's a rapid decay in interest, it actually looks like it's maybe parallel and you can see the meme persisting in low numbers. And then different variants, perhaps being prompted by a resurgence of this meme, then subsequently occurring. These were done in ggplot, just with a simple line geome and grouping by the particular meme variant and then rescaling the ax, the y-axis to log in one of the versions.



Programming Quiz: Transforming Data

Let's use Lada's approach of transforming variables on our Facebook data set. Most of our variables, such as friend count, likes, comments, wall posts and others are variables that I would call engagement variables, and they all have very long tails. Some users have 10 times, or even 100 ,the

median value. Another way to say this is that some people have an order of magnitudes, more likes, clicks, or comments, than any other users. In statistics, we say that the data is over dispersed. Often it helps to transform these values so we can see standard deviations, or orders of magnitudes, so we are in effect, shortening the tail. Here was our histogram of friend count from before, and notice, we still have that long tail. We can transform this variable by taking the log, either using the natural log, log base 2, or log base 10. We could use other functions, such as the square root, and doing so helps us to see patterns more clearly, without being distracted by the tails. A lot of common statistical techniques, like linear regression, are based on the assumption that variables have normal distributions. So by taking the log of this variable, we can transform our data to turn it into maybe a normal distribution or something that more closely resembles a normal distribution, if we'd be using linear regression or some other modelling technique. Now, I know we're not doing modelling here, but let's just see what it looks like to transform the variable. First, I'm going to just do this in the summary command. So, here's our regular summary of friend count. Looks like the median friend count is 82, and the mean is 196. I can take the log base 10 of this friend count and get a different table. Now this seems a little bit unusual since I have negative infinity for the minimum and negative infinity for the mean. Hm.



So what must be going on? Well, some of our users have a friend count of zero. So, when we take the log of base 10 of zero, that would be undefined. For those familiar with Calculus, the limit would be negative infinity, which is why that appears here. To avoid this, we're going to add one to friend count, so that way we don't get an undefined answer, or negative infinity. There, that looks much better. And, just to show you another function, let's also use the square root on friend count. This would be another type of transformation. For me, log base 10 is an easier transformation to wrap my head around, since I'm just comparing friend counts on orders of magnitude of 10. Basically a tenfold scale like the pH scale. Now that you've seen transformations within summaries, let's see if you can

apply a similar transformation to the histogram. Check out the instructor notes to learn how to use scales and how to create multiple graphs on one page. Once you've read through those links, and you think you're ready, try this next programming exercise. In it, you're going to create three different histograms. The first one will be our original friend count histogram, and then the second one will have the friend count transformed using log 10. And then the last histogram will have the friend count transformed using square root. Now, I haven't given you all the information to do this exercise, but I want you to be resourceful. Part of being a good programmer is being able to read documentation and know how to find your answers to the questions you may have. Now, we've already done the legwork for you by providing the resource. So, let's see if you can figure it out.

Answer:

Your task was to create three histograms on one output plot. And to do that you needed to install the grid extra package and load it into our studio. Once you've got that, now it's just a matter of creating each of the three histograms. And saving them to three variables. I'll use p1, p2 and p3 for each of the plots that I'm going to create. Now the first histogram is the same as before. We just have friend_count, pass to x, and pf, pass to data. For the second histogram, we'll take the log10 of friend_count. And remember to add 1 to it first, so that we don't get any undefined results. And finally for our third histogram, we'll simply take the square root of friend count.

Now, if I run all these lines of code, I can see that I create three new variables that saves each of my plots. And now I just need to pass each of these plots to grid dot arrange. And set end col equal to 1, since I just want one column with all my histograms. And running my code, I can see that I get all three histograms. And here's a closer view of the three. Notice that our second plot is much better. Since we have a normalish kind of distribution. The square root transform is also better than no transformation at all, since we don't have as much of a long tail. We do still have a long tail, it's just that much lower friend count since we transformed the variable. The tail's just not as bad as before. Now there was another way to create these same three histograms using a different type of syntax. Now the syntax is a little more complicated at first. But I want to give you a preview right now since you'll see it in upcoming lessons.

This time, we're going to use the ggplot syntax to create our histogram. Ggplot is very similar to the q plot function, since it still takes the same parameters, x and data. The key difference is that we need to include any x or y variables inside this aesthetic wrapper or AES. The other thing we need to do is we need to tell ggplot what type of plot we want to create. Or what kind of geom we want. The geom that we want is geom histogram. Running this bit of code I can see that I get my original histogram. Which is the first one we created on friend count. Running the entire line I can save this histogram into p1. Now I've got a plot that I can just alter using scales. So for the second plot, I'm going to take my first plot and just add a scale_x_log10 to it. This is going to transform the x-axis, or the x variable, using log base 10. Similarly, for the third graph, I'm going to add scale x square root. Now, I don't want

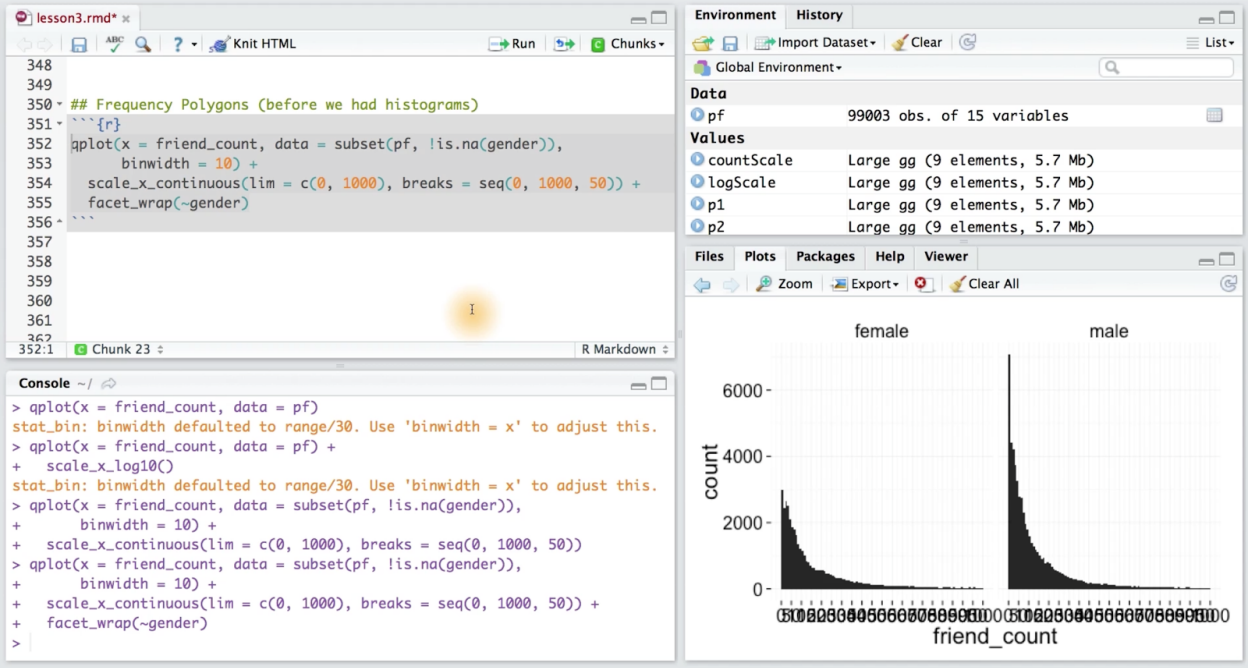
to use p2 here, since I've already got a scale log10 on it. I want to use my original graph, p1. And add scale x square root. Now saving all these plots and writing our grid dot arrange function, we get the same output as we had before. Now there is a slight difference here based on the x-axis labeling. And I want you to see if you can figure that out. If you're not sure of the differences, don't worry. I'll explain it in the next video.

Add a Scaling Layer

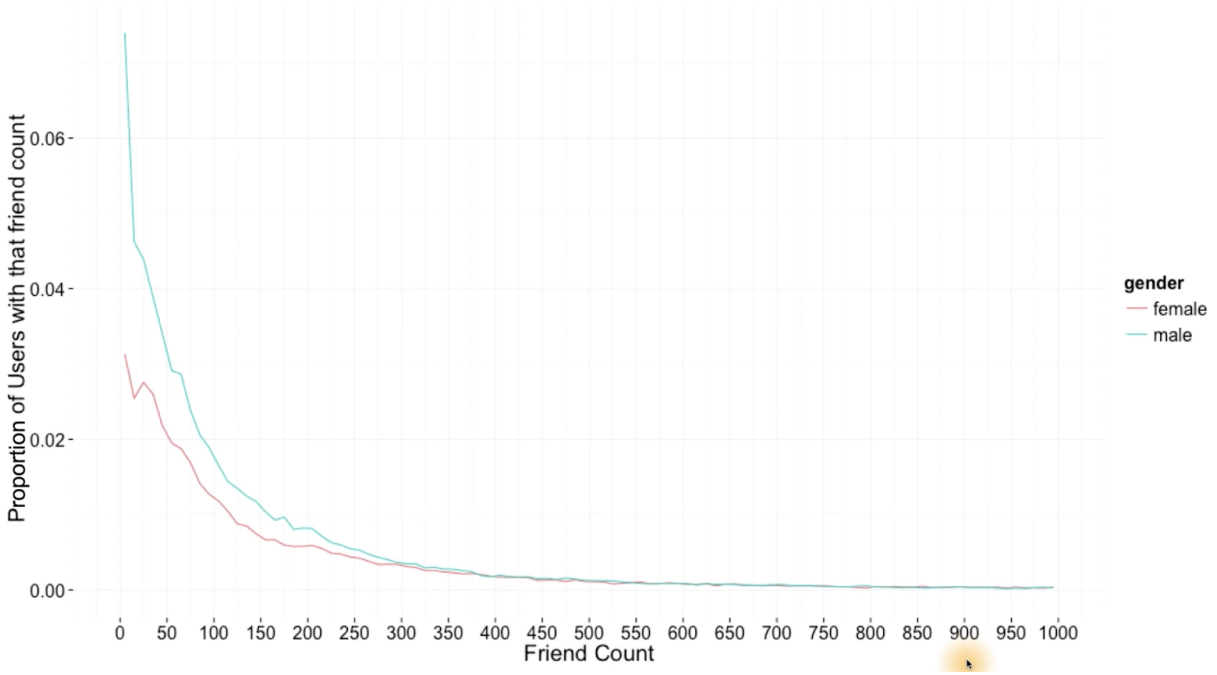
If you watched the solution video, you would have noticed that I used two different methods for transforming our variable. The first method used a wrapper right around the variable, and then the second used a scaling layer. Let's look at the differences between these two plots and see what the two adjustments really do. I'm going to save this first plot into log scale and I'm going to save the second plot into count scale. I'm going to use each of these variables and pass it to grid.arrange, so that way I can plot them side by side, and that's why the ncal is equal to two. Running this code, we can see that we get our two histograms. When looking at the two plots, we can see that the difference is really in the x axis labeling. Using scale_x_log10 will label the axis in actual friend_counts. Whereas using the log10 wrapper will label the x axis in log units. This is just something to keep in mind as you make more plots. In general, I think it's easier to think about actual counts, so that's why I prefer using the scale_x_log10 as a layer. Now, this does mean that you'll need to learn the ggplot syntax, but don't worry, you'll have plenty of practice in lesson four, and if you're feeling really adventurous, you can add a layer to any of the plots that you create. So, take our original histogram for friend count. Let's just add scale_x_log10 to it, and there you go. You have it, the new histogram transformed using log base 10.

Programming Quiz: Frequency Polygons

So far we've seen how to examine a variables distribution using histograms, and how to check our hunches with visualizations and numerical summaries. But there's another type of plot that lets us compare distributions, and it's called the frequency polygon. Frequency polygons are similar to histograms, but they draw a curve connecting the counts in a histogram. So this allows us to see the shape and the peaks of our distribution in more detail. You might remember from before that we were looking at friend count using this code. This code gives a histogram of our user's friend counts, and then we added a facet wrap and broke it out by gender.



Remember we were trying to answer the question, who on average has more friends, men or women. We said we couldn't tell based on this histogram, so we ran some numerical summaries instead. And instead of having these polygon and overlay these histograms together. Here's how we can create that frequency polygon. I'll copy and paste the same code, except I need to make an addition. By default qplot is going to create a histogram when I pass it just one single variable. So I need to tell it to create a frequency polygon instead. I can use the geom parameter to do that. Here, I'll pass it frequency polygon, so that way we create a different type of plot, and I won't need this facet wrap by gender anymore. To get the distributions of each gender on the plot, I'm going to pass the parameter color to qplot and set it equal to gender. So here's my color parameter, and now I'm going to pass it gender. When I run this code, I get one plot with notice how gender has been assigned a color, so color is indicating which frequency polygon I'm on. And this is what the frequency polygon is really good for. We can compare doesn't really answer our question who has more friends on average men or women. Let's change the y-axis to show proportions instead of raw counts. This is going to involve some funky syntax, so I want to explain it. To change this count variable, we're going to pass in y to our qplot function. I'm going to assign the parameter y this expression. This allows me to get proportions instead of the actual raw counts on the y-axis. And I'm just move this around so that way my code looks a little bit cleaner.



And finally, let me change the labels to more accurately explain the plot. Alright, it looks like I have everything. Let's run this code and see the differences. Zooming in on the plot, we can see that we've changed the y-axis scale, and we have our labels appearing here and here. And while it may appear that males have higher friend counts on average than women, we can see that many males or a high percentage of them have low friend counts. It's probably in this tail region of the graph where females overtake males. I encourage you to play around with this yourself in our studio to see where women overtake men in this side of the x-axis. Try using limits or try using breaks to explore more. We've learned some new syntax here, so let's see if you can put it into practice. Your task is to determine which gender creates more likes on the world wide web. To answer that question, I want you to create a frequency polygon and explore it in many different ways. Remember that the first plot that you make doesn't need to be final. I want you to create a couple of plots and make adjustments to the limits and the breaks on the x axis until you're satisfied.

Answer:

Remember, our goal in analyzing this Facebook user data is to understand our users and their behavior. This question is just another dimension where we're trying to understand which segment of our population uses a certain feature on the website. In this case, we're wondering whether or not males or females end up using likes on the World Wide Web more often. So, first I'm just going to remind myself of what variables are in my data set. I'm wanting to compare likes between the genders, so I'm going to use www_likes. This code would give me a histogram which isn't what I want. And remember I also need to subset the data to remove any values for gender that are NA. Now this bit of code is starting to look more like what I want. But, I've only got one frequency polygon on

here, and I need two. So I'm going to use the color parameter, and pass a gender. That seems a little bit better. I'm also going to add scale x continuous, since I know that World Wide Web likes is on a continuous scale. Alright, now we're looking at a reasonable plot. Zooming in, it looks like males typically have more likes on the web. But I can't really make sense of the tail end of this graph. This is long tail data, so let's use a log transformation to see if we can get a better look at what's happening down here. I'm going to go back to my code and add a scale x log 10. Running this code we get a different plot with much more information. It looks like males have more likes on the web at first, but we can see that females overtake males at this point in the graph. So this still leaves me wondering who really has more likes, men or women?

Programming Quiz: Likes on the Web

Let's answer our question using a numerical summary instead. I want you to run some code in R to answer the following questions. First, what's the www_like count for males, and second, which gender has more www_likes? Is it the males or the females? These two questions will automatically be graded so you don't really need to go to the solution video unless you get stuck.

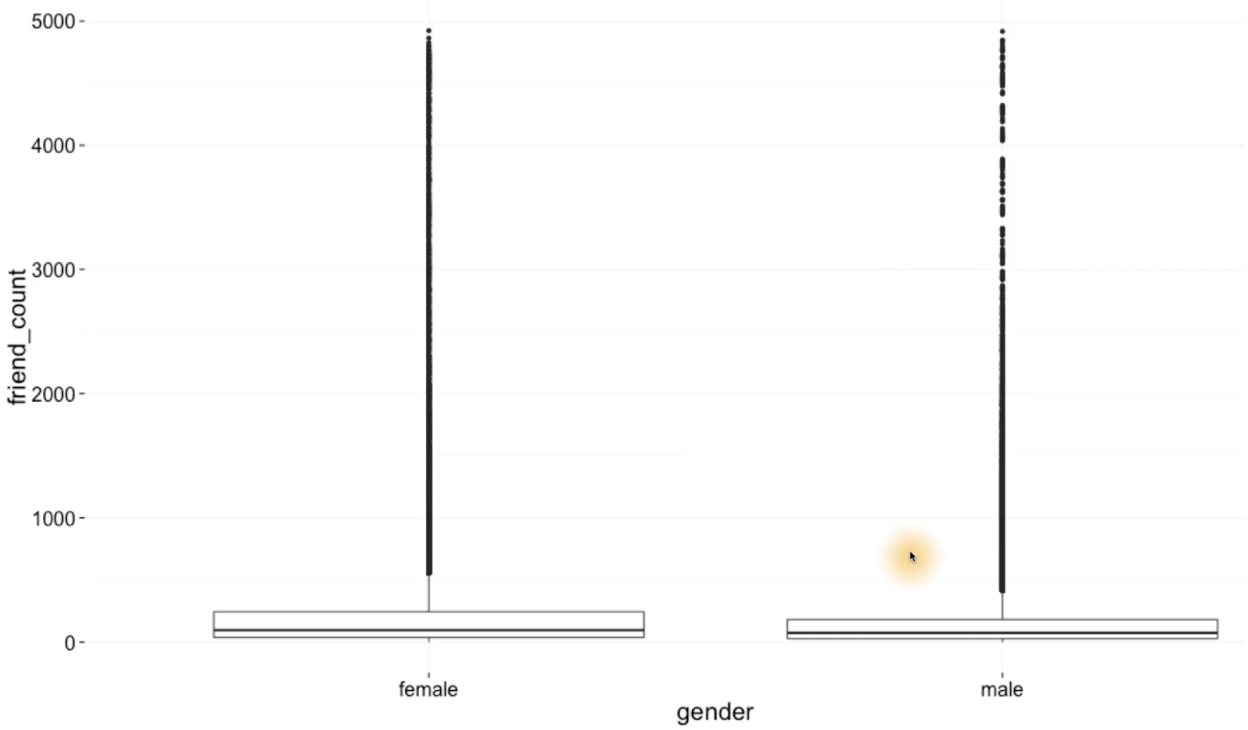
Answer:

The by command is going to allow us to answer this question. So first I'm going to create an r block of code and I'm going to use the by command. I'll pass a www_likes as the first parameter and then I'll pass gender as the second, since that's the variable I would have split our www_likes over. I want to total of likes for each gender, so I'm going to use the sum function here. Writing this code we can see our output. The likes for males is at 1430175. But it's the females that have way more likes. There are over three and a half million. It looks like females have more than two times the number of likes as men. And while this might seem trivial, information like this can help websites or other businesses decide which features are used most often by different subgroups. This might help a business or a website decide which feature they should continue to invest in, or which ones they should just leave behind.

Programming Quiz: Box Plots

Let's use another type of visualization that's helpful for seeing the distribution of a variable called a box plot. Now if you're unfamiliar with a box plot, you can find resources in the instructor notes, and there's also a link to Udacity's statistic class so you can test your own knowledge. You may recall earlier that we split friend count by gender in a pair of histograms using facet wrap. The code looked like this. Instead of using these histograms we're going to generate box plots of friend count by gender, so we can quickly see the differences between the distributions. And in particular we're going to see the difference between the median of the two groups. And remember again the the q plot

function automatically generates histograms when we pass it a single variable. So we need to add a parameter to tell q plot that we want a different type of plot. To do that, we're going to use the gym called box plot.



Now, I'm going to use the same data set as before. So I'm going to keep this and q plot. Now, what's different about box plots is that the y axis is going to be our friend count. The x axis, on the other hand, is going to be our categorical variables for male and female, or gender. Notice that we use the continuous variables. Friend count as y. And the grouping, or the categorical variable as x. This will always be true for your box plots. I forgot a parenthesis here and then let me just reformat my code so it looks a little bit cleaner. There we go. Running this code, we can see that we get our two box plots. Let's zoom in to get a closer look. The boxes here and here cover the middle 50% of values, or what's called the inner quartile range. And I know these boxes are hard to see, since we have so many outliers on this plot. Each of these tiny little dots is an outlier in our data. We can also see that the y axis is capturing all the friend counts from zero all the way up to 5,000. So we're not omitting any user data in this plot. And finally, this horizontal line, which you may have noticed at first, is the median for the two box plots. And you might be wondering what makes an outlier an actual outlier. And well, we usually consider outliers to be just outside of, one and a half times the IQR from the media. Since there's so many outliers in these plots, let's adjust our code to focus on just these two boxes. We'll have you do this in the next programming exercise. See, if you can alter our code to make that adjustment.

Answer:

To change what our box plots look like, we just need to adjust the Y axis. We can use the ylim parameter inside of Q plot to do so. Here, I can set the upper and lower limits. The lower limit will be 0 and the upper limit will be 1000. Now, you may or may not have gotten a warning message here about removing data points. When we use the ylim parameter, we're actually removing data points from our calculations. So the original box plots that we have might look slightly different from this. Another way to create the same plot is to use the scale y continuous layer. So we can use this same bit of code here and then just close off our parenthesis. Adding the scale wide continuous layer, I can set the limits to be between 0 and 1000. Running this code, we can see that we get the same exact box plots. What I want to draw your attention to, is the female box plot. Notice how the top of the box is just below 250. So this value might be around 200 or 230. But this value might not be accurate for all of our data. When we use the ylim parameter or the scale_y_continuous layer, we actually remove data points from calculations. So a better way to do this is to use the cord Cartesian layer to set the y limits instead. So, we'll use the same bit of code and then instead of adding scale wide continuous layer, we'll add the cored Cartesian layer. And here I'll set the y limits from 0 to a 1000. Notice how the top of the box has moved slightly closer to next video. We'll also discuss what the top of the box means. What the bottom of the box means. And what this black line means.

Programming Quiz: Box Plots, Quartiles, and Friendships

It looks like females on average have slightly more friends than men. Since I can see that this median line is slightly higher. That's what this black line is. It represents the median or the middle 50% of friend counts for females and for males. Now this difference isn't very large. So let's zoom in to take a closer look. This box for females and this box for males Represents the middle sense that we zoom in even more to take a closer look. We should consider any values less than 250. Now, there's no exact choice here, I'm just choosing something that seems reasonable, since the bulk of my data is down here. After running this code, we can now see that the bulk of user friend count is similar for the middle 50% of men as it is for the middle 50% of women. Its just our females are slightly higher for friend count. Lets look at actual values though and compare the values to what we see in our box plot. We can look at those values by using the by command and running a summary of our friend count split by gender. So first, I want to include my friend count which is the variable I want a summary of. I want to split it over gender and I want a summary. Running this code, I get an output of my table, which shows me the minimum maximum values for both genders, as well as the core tiles. The first core tile for women is 37 and that looks about right in our graph. The third quartile or the 75% mark is at 244 and that's all the way up here. This means that 75% of female users have friend counts below 244. Or another way to say this is that 25% of female users have more than 244 friends. Similarly for the men, we can see how the first quartiles and the third quartiles match up to the box plot. Now, you might have remembered that we used coord_cartesian in the solution video from before. We did this so that way, the table output would match our box plots. If we would have just used the ylim parameter inside of qplot, we would have gotten different quantiles that wouldn't match our picture. This is just a subtle difference that you should be aware of when working in R. Now, it's your turn to

answer a different question. On average, who initiated more friendships in our sample? Was it men or was it women? Used some of the techniques that we just covered and then write a few sentences explaining how you came up with your answer. This second question won't be automatically graded, but it's important that you know how to communicate your analysis to other people.

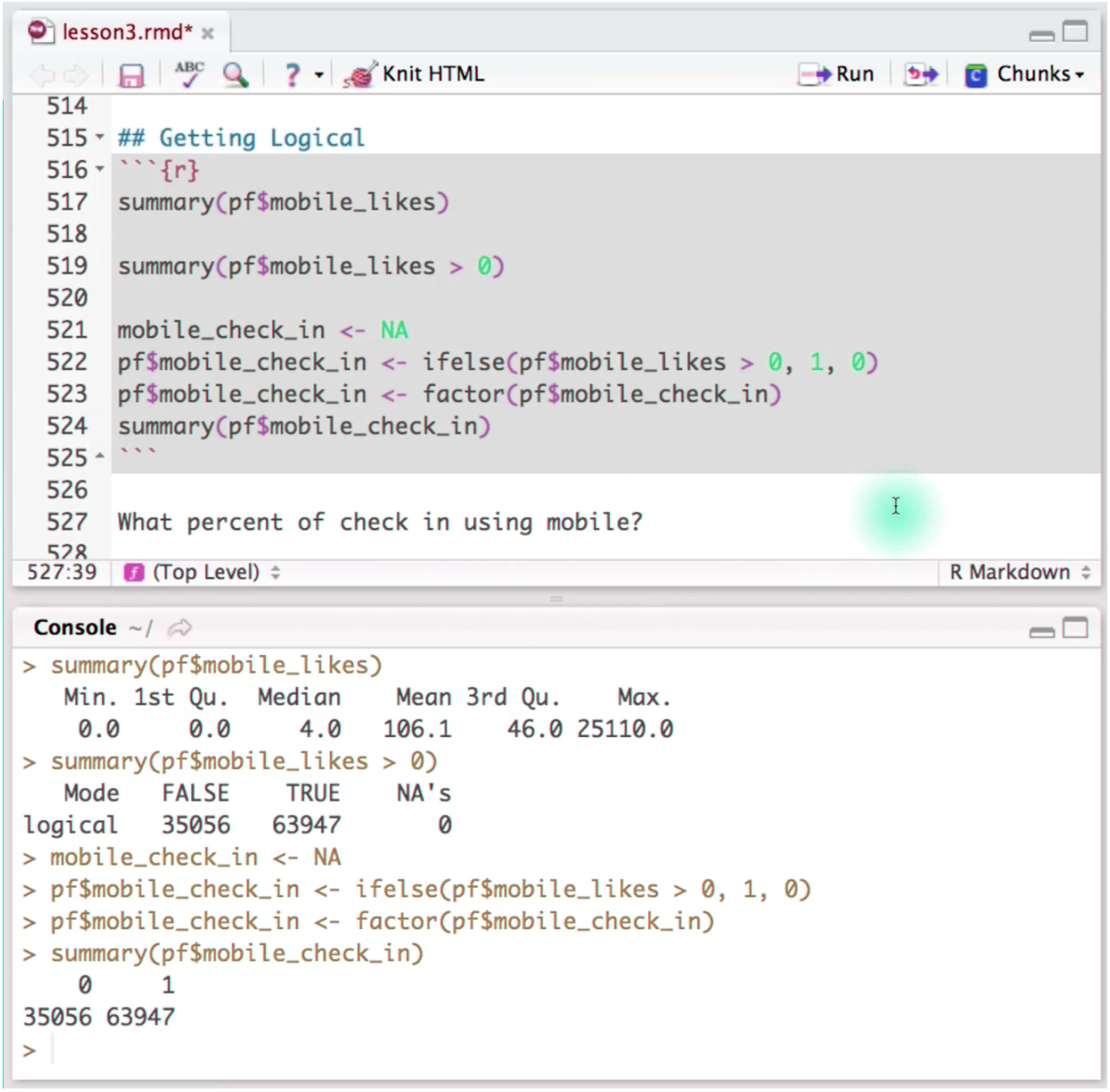
Answer:

To answer this question we want to write code that looks very similar to the code from before. We want to create a box plot to compare the distribution of the number of friendships initiated for our male and female users. So lets just remind ourselves what variables we have in our data frame. So for friend requests, I think I'll use friendships_initiated that sounds right. To make my box plot I'll pass x the variable gender, since that's my categorical variable. And then I'll pass y, friendships_initiated. Now I need to subset my data frame just like from before. And then I'll need to tell cube plot that I want to box plot. So I'll set the geom. Looking at this plot, we see that most users make less than 500 friend requests. So let's set our y limb setting zero as our minimum and 500 as our max. So I'll add the cored_cartesian layer and set my limits. It's really close, but it looks like females have slightly more friend requests. Since the median black line is slightly higher for females than this median black line for males. Now, this might be too close to call. So, I say we should zoom in again. This time, we'll change the upper limit to 150. Yeah, it looks like females initiate slightly more friendships on average. Let's also check this with a numerical summary. I can use the by command and split friendships_initiated by gender and then, find their summary. So, I'll take my friendships_initiated, split it by gender and then run a summary. And here's our output. And we can see that the median for females is 49 whereas the median for males is 44. Now, you might be wondering, why should we even create this box spot to begin with if we can answer our question with a numerical summary. Well, it's helpful to understand the distribution of the data. So in the case of the box plot, we can see the middle 50% of values for each segment of our categorical variable. Our box plots also let us get a sense of alliers. So in one way, they're much more rich with information than just this table.

Programming Quiz: Getting Logical

So far, we've looked at a number of visualizations to examine the distribution of a single variable. And, we also looked at transforming a variable to get a better look at the data. Now, there are other ways that we can transform a variable, beside using something like the square root or a log. You often want to convert variables that have a lot of zero values to a new binary variable that has only true or false. This is helpful because we may want to know whether a user has used a certain feature at all, instead of the number of times that the user has actually used that feature. For example, it may not matter how many times a person checked in using a mobile device. But, whether the person has ever used mobile check-in. Here's what I mean. Here's a summary of the mobile likes in our dataset. The median is four, which means we have a lot of zeroes in our dataset. If I run this summary, I'm going to get a

different type of table.



Notice that in the table I get logical counts, because I use this comparison operator. I wanted to see whether or not someone had actually checked in. So, instead of tracking the count of mobile likes, let's create a new variable that tracks mobile check-ins. We'll call this variable, mobile_check_in. The first thing I'll do is assign it a bunch of NA values. Next, we can use the if/else function to assign a value of one if the user has checked in using mobile and a value of zero if the user has not checked in. So in this if/else statement, if this condition is true, we'll assign our user a value of 1. Otherwise, we'll give them a value of 0. And the last thing I'll do is convert it to a factor variable. Running this code, I get my new variable saved. And then taking a summary of the results, I can see that about 64,000 just shy of that, have checked in using mobile while 35,000 have not. So, using this data can you tell me what percent of users check-in using mobile. Try to use actual code here instead of using these wrong

numbers. Round your answer to the nearest percent and don't include the percent sign when you enter it here.

Answer:

To answer this question we need to know the number of users who did check in using mobile and the total number of users in our sample. Now I could have just used these two numbers and divided them, but I want to do this programmatically. To do that I can take the sum of my mobile check in variables when it's equal to one. And then divide that by the length of that vector. This is how many users are in our sample. Running that code I can see that I get about set, so it wouldn't make a whole lot of sense to continue the development of the mobile experience. At least based on this sample data set. So when you're answering questions, it's important not to just think about what kind of data you're looking at but maybe what types of transformations you can make to the variables themselves. Sometimes you want raw counts, and other times you just want a binary yes or no, did a user use this. It's important to think about what kind of data you need to answer a specific question. So not every sort of transformation you do has to be a functional transformation like square root or log.

Analyzing One Variable

Before we finish this lesson, I want you to take some time to reflect and think about the key ideas in this lesson. What are some strategies that you could use with other data sets? Once you've submitted your answer, you can see what we thought was important in the solution video.

Analyzing One Variable

Congratulations on finishing lesson three. Here we learned that it's important to take a close look at the individual variables in your data set. To understand the types of values they take on, what their distributions look like, and whether there are missing values or outliers. We did all this in part by using histograms, box plots, and frequency polygons, which are some of the most basic and important tools for visualizing and understanding individual variables. We also made adjustments to these plots. For example, we changed the limits on some of our axes. We adjusted the bin width on our histograms. And we transformed variables either logarithmically or turned them into binaries to uncover hidden patterns in the data. These are some of the main conclusions that we had, and, maybe you have more. We encourage you to share your ideas in the discussion forum, and we'll see you again in lesson four, where we explore the relationship between two variables.

Exploratory Data Analysis

Lesson 4 Notes

Explore Two Variables

Welcome!

Welcome to lesson four. In this lesson you'll learn how to investigate two variables, and you'll learn how to visualize and quantify that relationship. You'll learn about data aggregation, conditional means, and scatter plots. You'll also get to hear from Moira about how she used some EDA techniques and her work in perceived audience size. Let's get this lesson started by hearing from Moira.

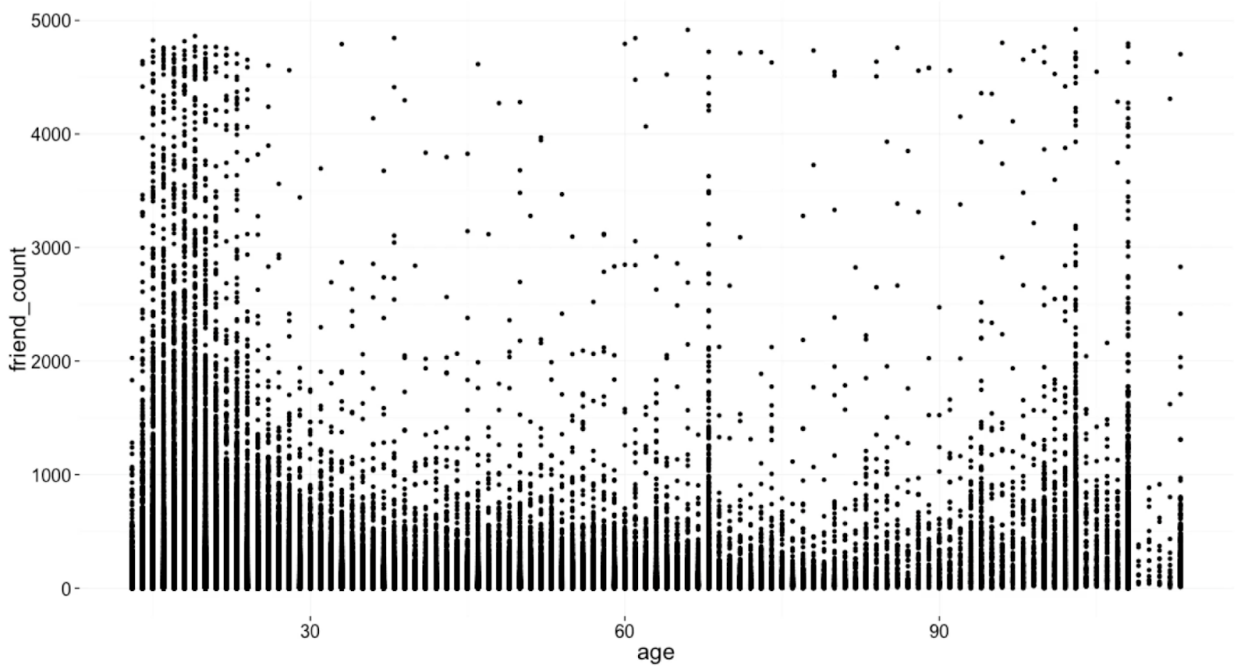
Scatterplots and Perceived Audience Size

So after plotting histograms and looking at peoples guesses about their audience size, I wanted to see how that matched up with their actual audience size. So then at that point I turned to scatter plots. So here's an example of the first scatter plots that we did looking at this. So on the x axis is how many people actually saw that post. And on the Y axis is how many people our survey respondents guessed saw the post. If they guessed perfectly accurately, their guess would fall along this dotted diagonal line. That's the perfect accuracy line. But one of the things that stands out with, in the scatter plot, is that you see these horizontal stripes. That's again because people are guessing these regular numbers. You see a really clear stripe at about 50 and another really clear stripe at 100, and a little bit of a stripe at 200. And you really see that these all of this, the points on the scatter plot are just this big cluster at the bottom. So people are guessing very small numbers. 20, 50, something like that, when in reality their audience size is 100 or 200.

Programming Quiz: Scatterplots

Lets take Moira's advice and move away from using histograms. Up to this point we've looked at distributions of single variables. For example, friend count, and at most we looked at different subsets of friend count by splitting it, using a factor, or in our case it was gender. Now we're going to look at two continuous variables at the same time. So, to get started, make sure you're in the right working

directory, and then go ahead and load your data set and load the ggplot library. Usually it's best to use a scatter plot to examine the relationship between two continuous variables. Q plot chooses the scatter plot automatically when we pass two continuous variables to the x and y parameters, so let's go ahead and do that. I'll pass h to the x parameter, and I'll pass friend count to the y parameter, and finally I'll indicate that my data set is pf, my pseudo Facebook users. Now, there's over 99,000 observations in our data, so when we create this plot, it might take a few moments to render. And there it is. We could also write this code which will produce the same exact plot. This time, I'm not using the x and y parameters explicitly. And that's okay, because qplot knows which variables to use on which axis. X will come first, and y will come second. I'll run this code, just so that way you can see the same plot being produced. And there we go, we have a pretty ugly scatter plot, but what are some things that you noticed, right away?

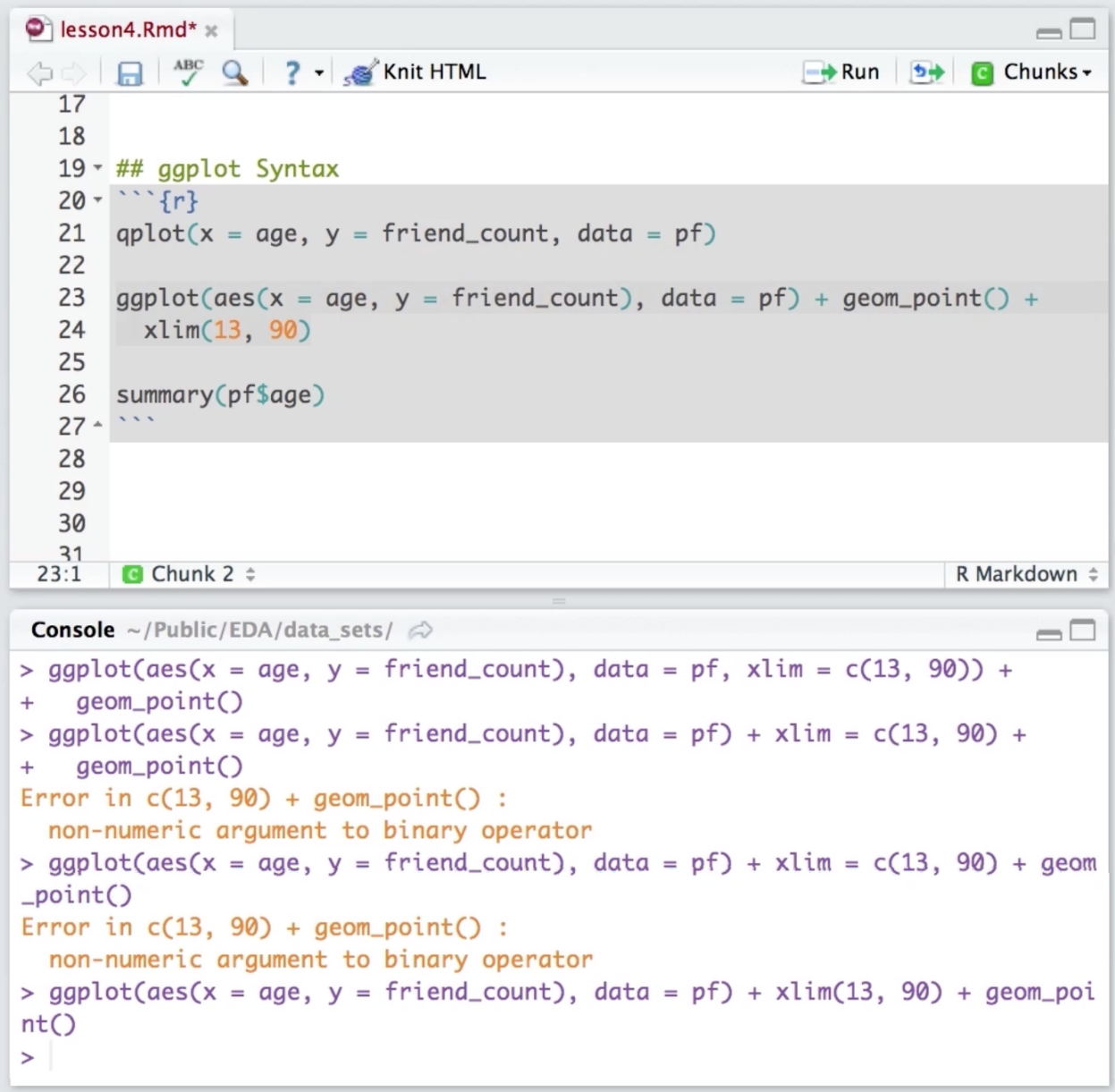


Answer:

Here are some of the things that I noticed. It looks like younger users have a lot of friends. These young users seem to have thousands of more friends than most users over the age of 30. You also might have noticed that there are some vertical bars where people have lied about their age, like 69 and also about 100. Those users are also likely to be teenagers, or perhaps fake accounts, given these really high friend counts. If there's something else that you noticed that we didn't cover, share it in the discussions.

ggplot Syntax

Let's make some improvements to our scatter plot. This time we're going to start by switching from the qplot syntax to the more formal and verbose ggplot syntax. I mention this in Lesson 3. The ggplot syntax will let us specify more complicated plots. So just as a reminder, this is the qplot syntax in order to create the scatter plot. Ggplot is another plotting function similar to qplot, and it has slightly different syntax. Here's the equivalent code to produce this scatter plot. The main difference between qplot and ggplot is that we have to say what type of geom or chart type that we want. In this case, we want a scatter plot.

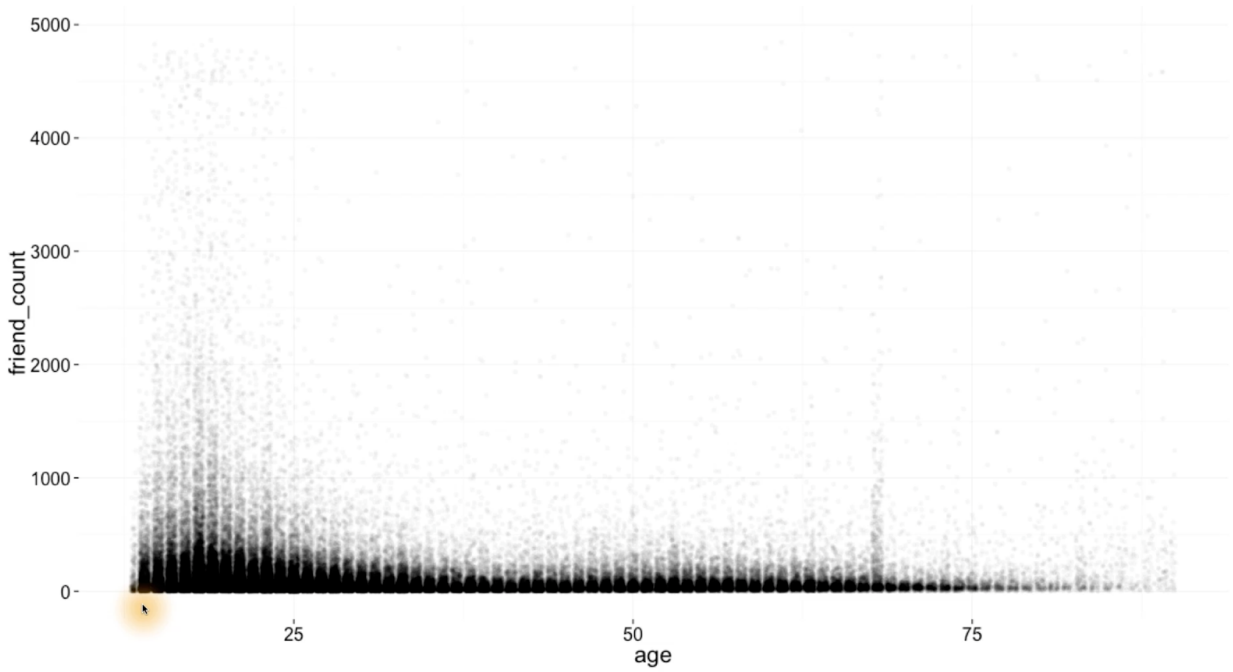


So we're going to use the geom_point. You can see the full list of chart types in the ggplot reference, which is linked in the instructor notes. The second big difference between the two plotting functions is that ggplot uses this aes wrapper. We have to wrap our x and y variables inside this aesthetic wrapper. Don't worry about this too much. We'll cover this in more depth later. Now, I want to get some ranges

on my age, so I'm going to run the summary command on age to figure out the lower and upper limits. The minimum age of a user is 13, and the maximum is 113. Now, let's click the x-axis at age 90 and at age 13. This seems reasonable since users who are younger than 13 are not permitted to use Facebook. And we're really not that confident whether people who report being older than age 90 are telling the truth. To do this, we're going to use the xlim layer. Now I'm not going to pass xlim into ggplot. Instead, I'm going to use it as an additional layer outside of it. Notice that we use the plus operator to add a new layer to our figure. And this is going to change the appearance of our x-axis. I really recommend that you add one layer at a time when building up plots. This allows you to debug and find any broken code. And there we go. There's a nicer plot with our users ranging from 13 to 90 years old.

Programming Quiz: Overplotting

Now you may notice that some of these points are spread out from one another, while others are stacked right on top of each other. This area of the graph is considered to be over plotted. Over plotting makes it difficult to tell how many points are in each region. So we can set the transparency of the points using the alpha parameter and geom point. I'm going to add this on a new line so that we can see our layers more clearly. Next I set the alpha parameter equal to 1 over 20. So this means it's going to take 20 points to be the equivalent of one of these black dots. Writing this code, we can see that the bulk of our data lies below the 1000 threshold for friend count. Now let's also add a little jitter here too. We can swap out geom_point with the geom_jitter. Age is a continuous variable but you really only have integer values, so we are seeing these perfectly lined up columns. Which isn't a true reflection of age. These columns should feel intuitively wrong and we want to make sure that we can see more of the points. So using jitter we can add some noise to each age so we get a clearer picture of the relationship between age and friend count. Writing our code with geom_jitter we can see that we get a more disperse distribution. What stands out to you in this new plot? Keep in mind that the alpha is set to.

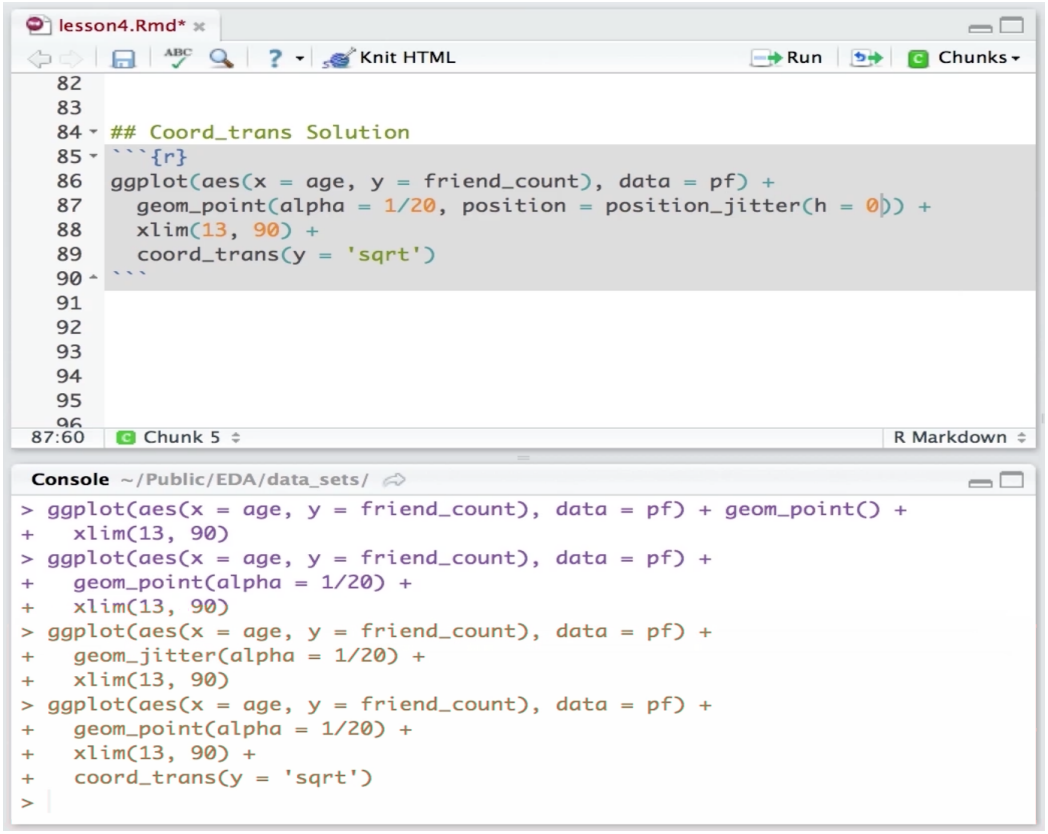


Answer:

With this new plot, we can see that the friend counts for young users aren't nearly as high as they looked before. The bulk of young users really have friend counts below 1000. Remember, alpha is set to 0.05, so it takes 20 points for a circle to appear completely dark. You also still might have seen this peak around 69, which is what we originally saw when we looked at friend count in our data. It's faint because we have the alpha parameter set to 0.05, but I would say that this is comparable to users in say, the 25 or 26 age group.

Programming Quiz: coord trans()

Now, let's make some more adjustments to our plot. This time we're going to use a transformation on the y-axis, so we change the friend count. We're going to do this so that we can get a better visualization of the data. Let's see if you can figure this out on your own. I want you to look at the documentation for coord trans and add it as a layer to this plot. The function you're going to use to transform friend count will be the square root function. See if you can make the plot and think about any observations you have, once you've made it. Now here again is an instance where we haven't given you all the knowledge upfront, but I think you can figure this one out.

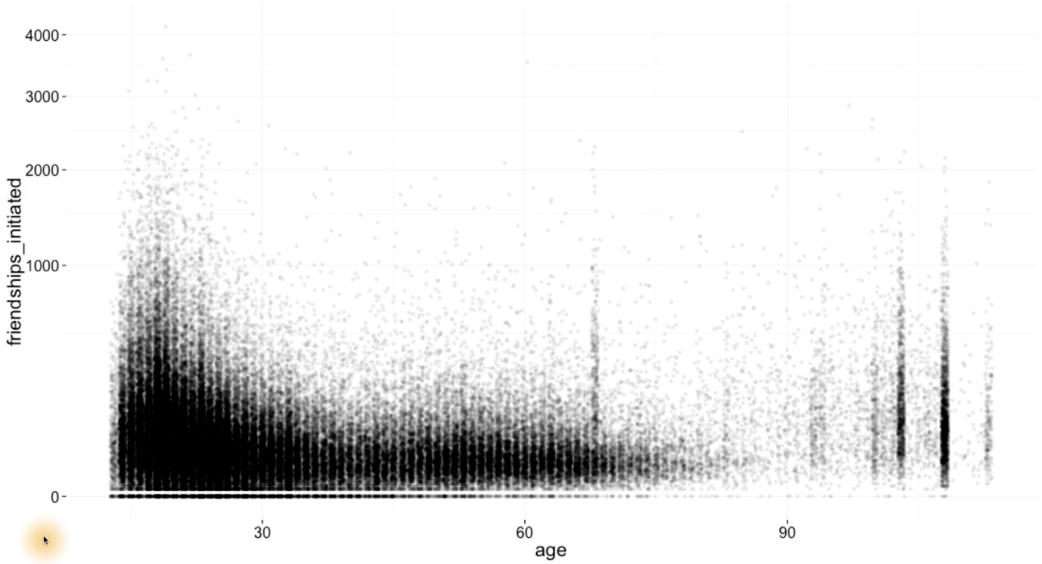


Answer:

For this programming exercise, you just needed to add a new layer onto our code, and that was the coord trans layer. I'll pass at the y variable, and I'll set that y equal to square root, which is the function we'll use to transform the y axis. With this plot, it's much easier to see the distribution of friend count, conditional, and age. For example, we can see thresholds of friend count above which there are very few users. Now, you might have noticed I went back from geom_jitter to geom_point. If we wanted to also add jitter to the points, we'd have to use a bit more elaborate syntax to specify that we only want to jitter the ages. We also need to be careful since some people have a friend count of 0. If we add noise to 0 friend counts, we might end up with negative numbers for some of our friend counts and those square roots would be imaginary. Okay so to make this adjustment I'm going to set the position parameter equal to position jitter and then I'll pass it a minimum height of 0. This is a bit more advanced in terms of syntax but it prevents us from having that warning message and getting negative friend counts over here. Remember that jitter can add positive or negative noise to each of our points.

Programming Quiz: Alpha and Jitter

Now that you've seen alpha and jitter in action I'd like you to use your new knowledge of them to explore the relationship between friends initiated and age. Build your plot up in layers and be sure to use the ggplot syntax. I really recommend playing around with this in RStudio, and then once you've got your finalized plot, copy and paste your code into the browser and submit it. There's no right or wrong answer here and there's a couple of different ways to make the plot. So just be sure that once you have your final plot, you copy and paste all of your code into the browser and submit. After you've created your plot, share any of your findings in the discussions.



Answer:

Your task was to explore the relationship between friends initiated and age. For the ggplot syntax, we're just going to pass it, x, which will be age, and y, which will be friends initiated. We'll put those in the aesthetic wrapper and then pass pf, as our data frame. I also need to remember to specify the geom, so here I'll use geom point for a scatter plot. This should get me going. It looks like that I used the wrong variable here, so looking at my data I can see that friendships initiated should be my variable, rather than friends initiated. Running this code, we can see our scatter plot. Now, I still get this discreteness with age, so I'm going to jitter our points. Here, I'll use an alpha set to one tenth, so I'll need ten points to make just one of these. That's looking a little bit better. Another way to jitter the points is to still use geom_point. But then to set the position equal to jitter. This code will produce the same plot that we have here. Let's run it to be sure. Sure enough, we have the same plot. And since I still have really high values for some of my friendships initiated, I'm going to use a coord transformation and take the square root of the y axis. When I add the coord trans layer and run this code I see that I get an error message. This should make sense since some of my friendships initiated have a count of zero. So, when I take the square root of that number with some noise, it might be negative which would be an imaginary result. So, I'm going to fix this code like we had before, so that way we have the positions set equal to h equals 0. So, we'll set position equal to position jitter

and we'll pass it h equals 0. This is the plot that I came up with. Now, you might have had something slightly different, or you might have even used a different transformation. Whatever you found, and whatever your observations may be, share them in the discussions. We'd love to hear from you.

Overplotting and Domain Knowledge

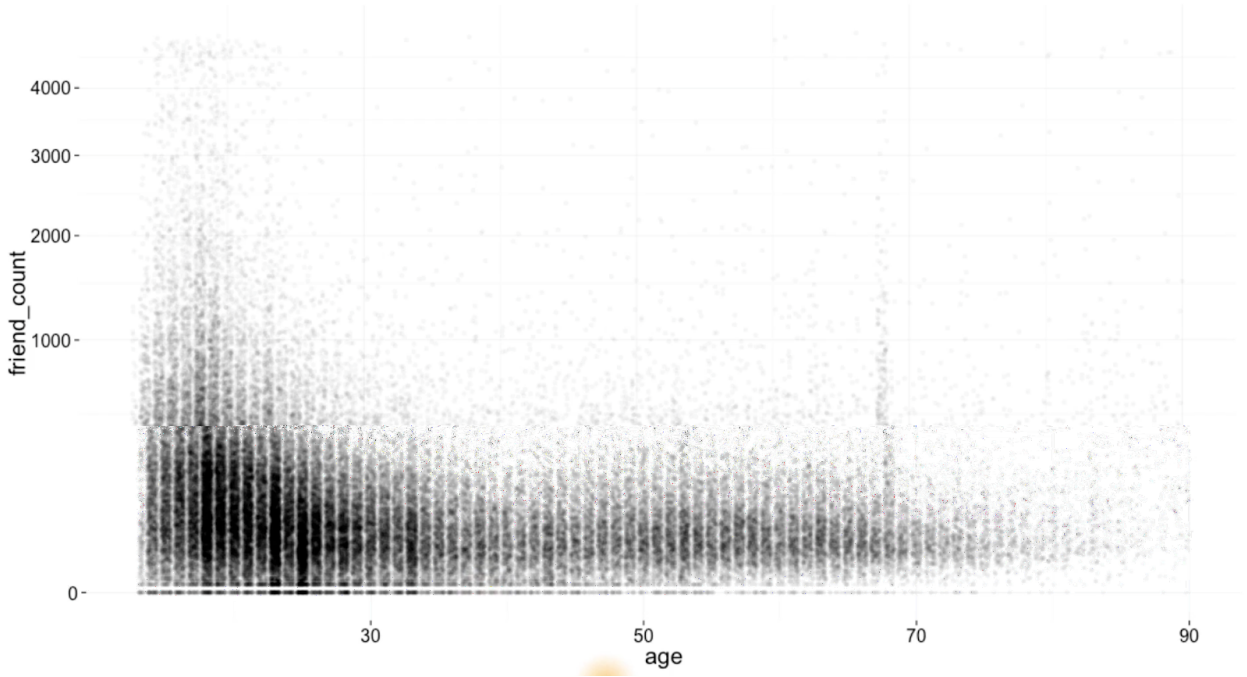
In the last exercise, we used alpha and jitter to reduce over plotting, but it turns out that there's more that we can do. Let's hear from Moira about how she used her domain knowledge and a transformation to make an adjustment to her scatter plot.

The next thing that I did, was to take again, the perceived audience size, and their actual audience size, but this time I transformed the axes. So this time, it's as a percentage of their friend count. Some people in this study had it actually makes more sense to think about your audience size as a percentage of the possible audience. All of the people in the study had shared their post with friends only privacy, so you'd expect that it would be bounded by their friend count. So, what we found when we plotted it this way was that all of the points are below this line of perfect accuracy, this diagonal line, really well below. And one other thing I should note about this plot, we actually ran two different surveys. We ran one survey where we asked people in a single post, how many people do you think saw your post? But we also asked a different set of people, in general, how many people do you think see the content that you share on Facebook? So that's what this plot is showing. This is the in general question. And their guesses are a little bit higher. But still, people typically think that maybe 10% of their friends see their posts when in reality it's more like 40% or 50%, even 60% of their friends will see their content in a given month. So that's what this plot is showing, is the percentage of friends who actually saw their content in the last month, again, they're underestimating.

Programming Quiz: Conditional Means

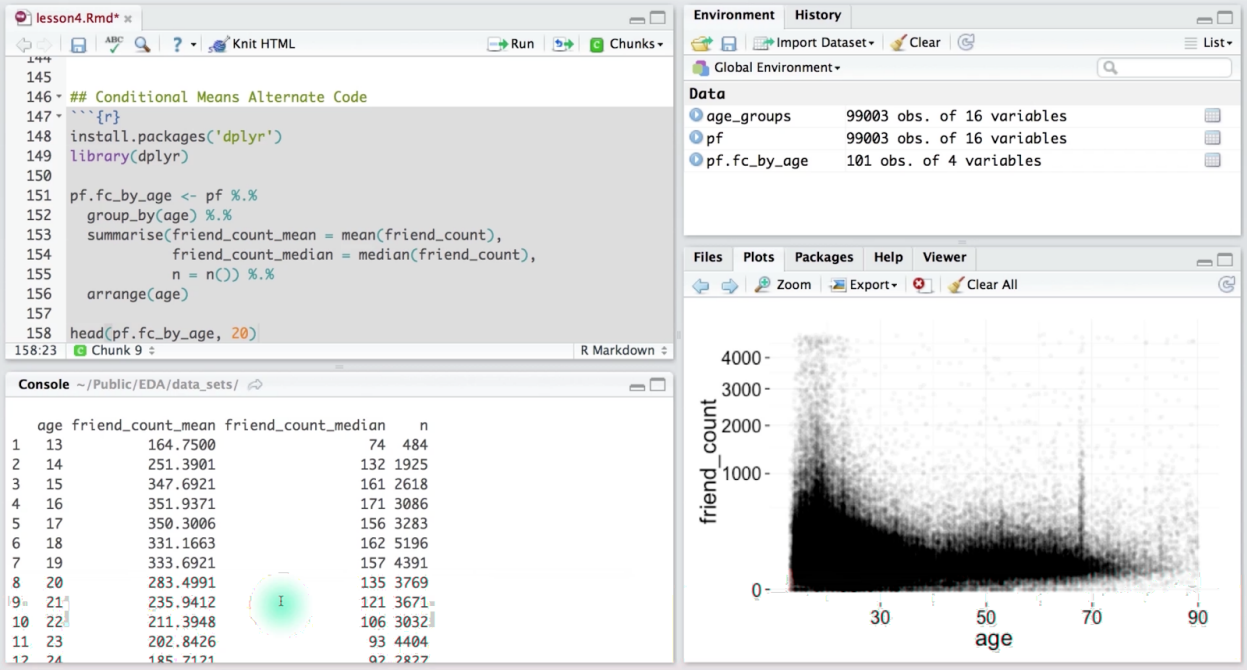
Let's take a step back and look at our scatter plot between friend count and age. This scatter plot keeps us really close to the data because we are visualizing every point in the data set. In general, it's not possible to judge important quantities from such a display. Sometimes you want to understand how the mean or median of a variable varies with another variable. That is it can be helpful to summarize the relationship between two variables in other ways rather than just always plotting every single point. For example we can ask how does the average friend count vary over age. To do this, we could start by creating a table that for each age it gives us the mean and median for income. To do this we're going to need to learn some new code. To create the table, we're going to use the R package called dplyr. I'm going to install and load that package now. The dplyr package lets us split up a data frame and apply a function to some parts of the data. Some of the common functions you might use are filter, group by, mutate and arrange. You can learn more about the dplyr package and browse there some examples from the links and the instructor notes. For now we'll work through an

example together.



So the first thing that I want to do is that I want to group my data frame by age. I'm going to save this grouping in a new variable called age groups. Next I want to summarize this new group in enough data And create new variables of mean thread count, median friend count and the number people in each group. So we're going to summarize our variable that we just created, age groups. Now right after I enter the data frame that I want to work on. I'm going to enter the variables that I want to create. So I want the friend count mean, and I get that by just taking the mean of the variable friend count. And I want the friend count median. And finally I want the number of users in each group. This in function can only be used for summarise, and it reports how many people are really in each group. The last thing I want to do is save this result into a new variable. I'll use the same data frame abbreviation, and then add fcbyage, since we have friend count by age. Running this code, I can see that I get a new dataframe with 101 observations, or groups, and four different variables. Using the head command, I can print out the first couple rows to examine the data frame. So notice, I have age, friend count mean, friend count median, and n, the number of users in each group. Now the state of frame isn't in any order. So I'm going to rearrange my data frame so that way the ages go from lowest to high. I'll just use the arrange function on the current data frame and arrange it by age. I'll save the result over the variable I just had and now heading out the data frame I can see that I have everything in order. Now this may seem like a lot of code, so I really encourage you to take your time and to review the code and the example. Make sure that you know what each piece is doing so that way you can write this code on your own. The two things that I really want to point out is that we need to pass in a data frame, or a grouping, at the beginning of each function. We also need to save the result into a new variable, and we pass that into the next function. This is what makes it difficult to understand this code at first, so I'm going to show you one other way to get this same table. To start, we're just going to take our data set and apply some function to it. To do that, I'm going to use the percent period

percent (%.%) symbol. This allows me to chain functions onto our data set. So I'm going to perform one function at a time. One after another on pf. The first thing I'll do is group my data set by age.



Now I'm going to chain on one more function. I'm going to summarize the result using friend count mean. Friend count median, and n. And finally, I'll add one more function, using this chain command, the percent period percent (%.%), and this time I'll arrange my data frame, by age. All of this code will produce, this exact data frame, so I want to make sure I save it to a variable. I'll save it to pf.fc_by_age, and I'll head the data frame just like before so that way we can check our result. Running this code, we see that we get the same exact result, we have age, friend count mean, friend count median, and, and the number of users in each age group. Printing out more rows, we can carefully scrutinize the table to learn about the relationship between age and friend count. We can already notice that on average. Users who are age 13 have slightly lower friend counts than those who are 14. It also looks like the mean friend count peaks at age 16 and age 17. Now, of course, we don't want to be summarizing and digging through tables like this. This is very tedious. We could show these observations more effectively with a visualization. So let's plot this table of averages. This is going to be your next programming assignment. I want you to plot the average friend count versus age. Make sure that you use the the appropriate data set. We're using this new data set here that we just created. And be sure that you have the appropriate variable names. They're different from before. Create your plot in RStudio and then check your work with ours.

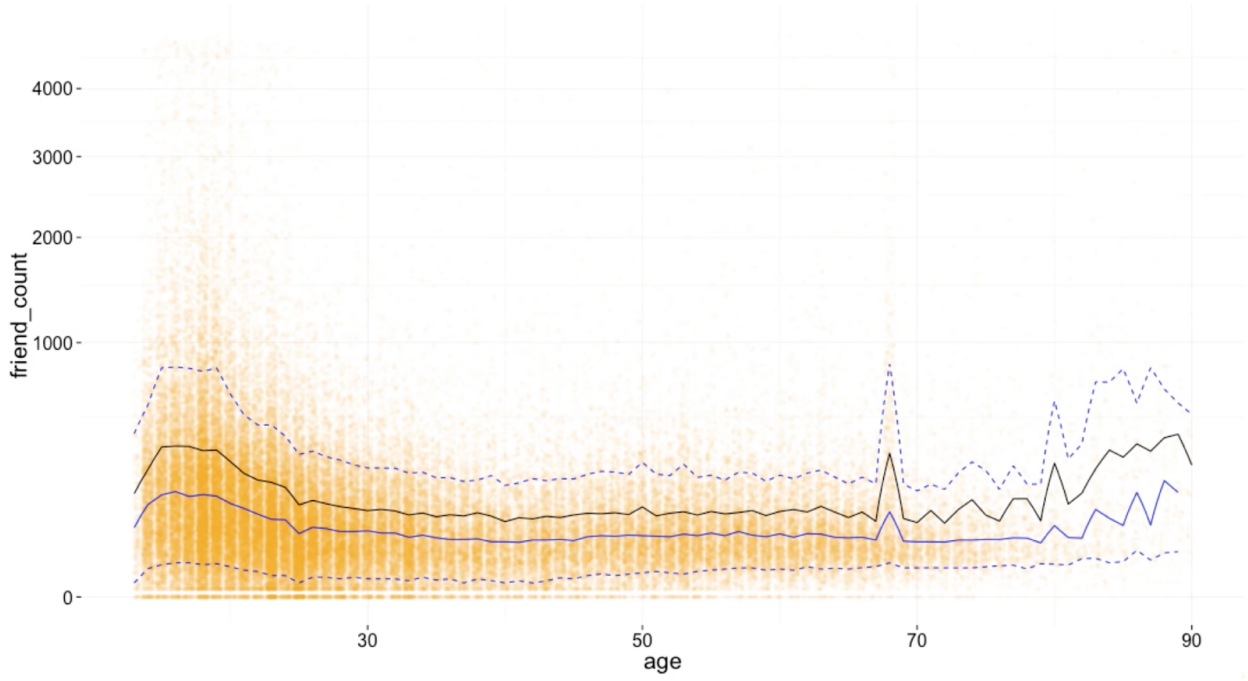
Answer:

This programming assignment was a way for you to practice creating scatter plots, and to make sure that you knew how to work with our new data frame. We'll use our ggplot function to get our histogram, and we'll pass it the two variables that we had in our data frame, age and friend count

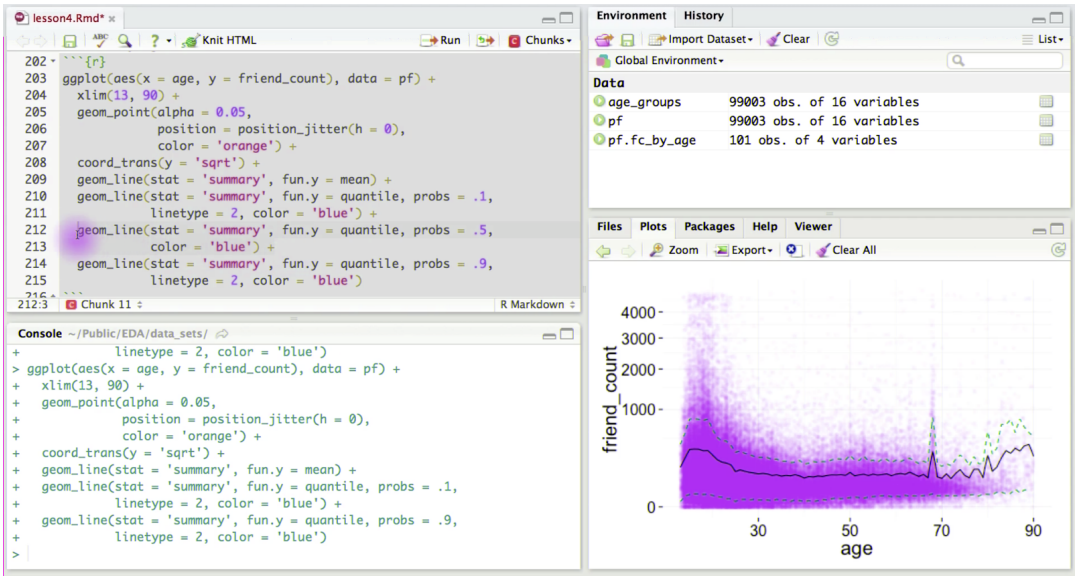
mean. And for the data variable, I want to pass it pf.fc_by_age, instead of pf, since this was the new data frame we're working with. Writing this code I see that I get an error, and that's because I forgot my geom. I needed to add geom point. So adding that as another layer, I can run my code and get the result. Now we can do slightly better than this plot. Let's connect our dots in order of age by using geom line instead of geom point. This plot immediately makes clear the patterns we mentioned before, as well as the oddness at older ages which are highly variable for friend count mean. They're jumping up and down, sort of all over the place. And for our young users, they still have high friend counts, and for the ages between 30 and 60, the mean count is hovering just over 100.

Programming Quiz: Overlaying Summaries with Raw Data

Ggplot allows us to easily create various summaries of our data and plot them. This could be especially useful for quick exploration and for combining plots of raw data, like our original scatter plot with displaying summaries. This plot is one of those displaying summaries and I want to be able to display it over the original plot we had for friend_count versus age. Let's see that first original scatter plot again. Now since all these points are black, I'm going to change the color of these. So that way when I overlay the summary, it's easier to see. I'm going to make the color here orange. So now, I've got my scatter plot and I want to overlay the summary that we have from before. I want to put this on top of this. I can add a geom_line to our plot to do so. Here I'm going to pass the parameter stat and set it equal to summary, and I'm going to give it a function for y. The fun.y parameter takes any type of function, so that way we can apply it to the y values. In this case, I want to take the mean. And there it is, this is my summary line for the mean friend count by age, over my raw data or my scatter plot.



This plot immediately reveals the increase in friend_count for very young users and the subsequent decrease right after that. We can add even more detail to this plot by displaying multiple summaries at the same time. Despite having this conditional mean plotted, we can't immediately see how dispersed around the mean. For example, are the median friend_counts for age 30 in this region or did they span all the way up to 2,000? Certainly we can see that most users, even young ones, don't have more than 2,000 friends. We can help ourselves understand this conditional distribution of friend_counts by also plotting quantiles of the data. So let's use the 10%, 50% or median and to this plot. So I'll add another geom_line, I'll pass it a stat of summary and then for the function, I'm going to pass it quantile instead of mean. I need to set the probability equal to one tenth or 0.1. This code gives me different from the mean.



So, I'm going to add some details to color it and to make it dash. I'll set the line type equal to two to make it dashed, and I'll set the color equal to blue. There that's much better. Now to add the 90% quantile, I would just need to change the probability here to 0.9 instead of 0.1. So I'm going to add another line. I'll use the same parameters as before and then just change the probability. So here we can see that 90% of users have friend counts below this line. The last thing I'll do is add in the 50% mark, which is the median. Now I won't make this geom_line dash, but I will make the color blue. Now this is quite a plot. I want you to try creating this same plot in R and try adding a coord_cartesian layer to zoom in on different parts of this graph. Look at the documentation for coord_cartesian if you need help. And definitely be sure that you understand all of this code. I know that there is a lot going on here. Once you've got a strong idea of this plot and you've explored it a bit, I want you to make some observations about the plot. What do you notice?

Answer:

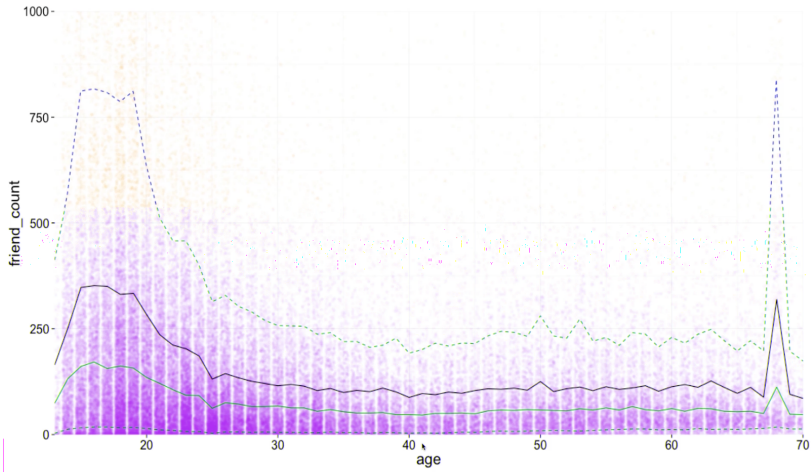
Without making any adjustments to this plot, I notice that having more than 1,000 friends is quite rare. Even for very young users we can see that this is where it peaks, with the exception of this

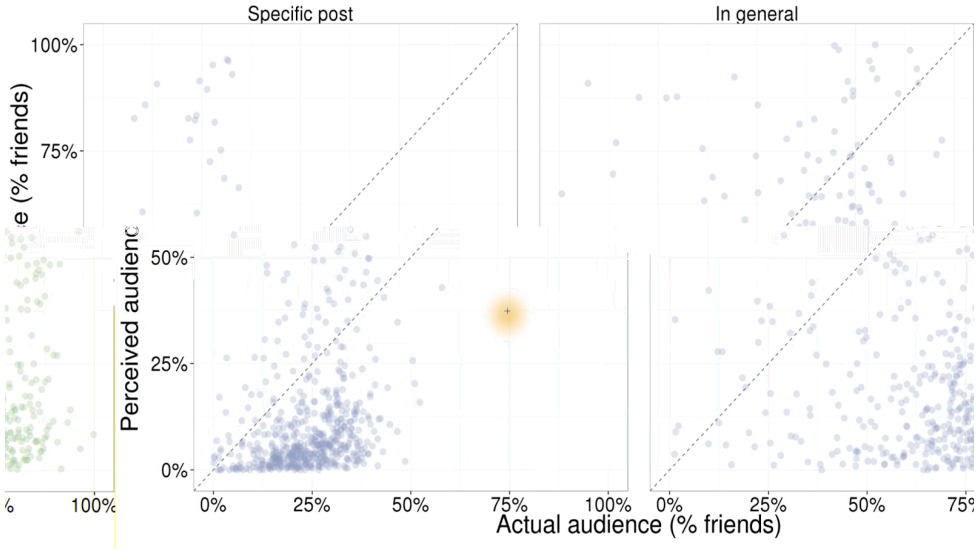
older crowd. And that the zoomed in on this plot using the coord_cartesian layer. To use it, we need to get rid of the coord_trans and the xlim layers first. So I'll delete those lines of code. Adding in the coord_cartesian layer, we can just set the x and y limits to whatever we think is appropriate. For our users, I want to explore users between the age of 13 and 70. And for friend counts, the 90th percentile is a 1,000, so that seems like a good upper limit here. Now it looks like I got an error and that's because I forgot to add a plus sign after this layer. I want to add all these layers on top of each other. Now with this plot, we can see that we get a little bit more detail. For example, we can see that for 35 year olds to 60 years olds 90% of those users have less than 250 friends, at least according to Facebook.

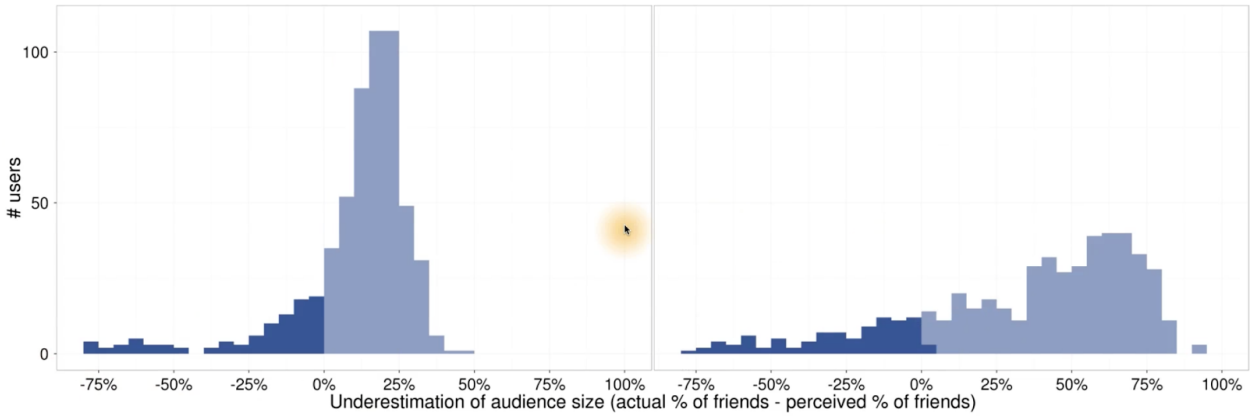
Moira Histogram Summary & Scatterplots

So, we've learned a little bit more about the relationship between age and friend count. And we did that using conditional means. We'll return to these two variables later and we'll see the relationship to a third variable in lesson five. But for now, lets see how more used a slightly different approach to pair a histogram to summarize raw data. So here's an example of how we paired raw data with summary data. So, here we have these scatter plots, where we're showing people's raw guesses of their perceived audience size against their actual audience size. And you can see they all fall below the perfect accuracy line. So people underestimate. But it's still kind of hard to get the big picture from these scatter plots and so, we have some histograms to capture more summary data. In this case, these two histograms show how much people over or

underestimated their audience size. So here on the left, at the 0 mark on the x axis, that's how many people guessed their audience size perfectly. If they had a lighter color blue, those are people that underestimated their audience. So you can kind of see how many people underestimated their audience by about 25%, versus 50%, versus 75% and so on. The darker blue are people who overestimated their audience size. And you can just see from the chart far fewer people overestimated their audience size. If you look on the right side of these summary plots, this is for people's over underestimations when you ask them about the whole month. How many people in general do you think see your post? And you can see that people don't underestimate quite as badly.







Programming Quiz: Correlation

Thanks Moira. Let's get back to our scatter plot of friend count versus age. It might be appealing to further summarize the relationship of age and friend count. Rather than having the four extra layers of geom line, we could try to summarize the strength of this relationship in a single number. Often, analyst will use a correlation coefficient to summarize this, now, if you're not familiar with correlation, you should review the links in the instructor notes. We've even included some practice from Stat 95, that's our statistics course. We're going to use the Pearson product moment correlation, noted with a lower case r, to measure the linear relationship between age and friend count. What I want you to do is to look up the documentation for the cor.test function. Figure out how to find Pearson's r using any two variables. What's the correlation between age and friend count? Round your answer to three decimal places.

Answer:

To look at the documentation, you'll simply type in ?cor.test. That will bring up this page. It looks like cor.test takes two vectors x and y. And then it will compute the correlation coefficient. It looks like we have a couple methods for determining that coefficient and we could either use pearson, kendall or spearman. For our purposes, we'll be using the pearson method. So your code might have looked like this. Writing this code, we get a correlation coefficient of 0.0274. This indicates that there's no meaningful relationship between the two variables. A good rule of thumb is that a correlation greater than 0.3 or less than minus 0.3, is meaningful, but small. Around pretty large. Another way to compute the same coefficient is to use this code. Here, I'm using the with function around the data frame. The with function let's us evaluate an R expression in an environment constructed from the data. Now, I know I haven't shown you this function yet but I wanted to introduce it to you. Running this bit of code we see that we get the same result.

Programming Quiz: Correlation on Subsets

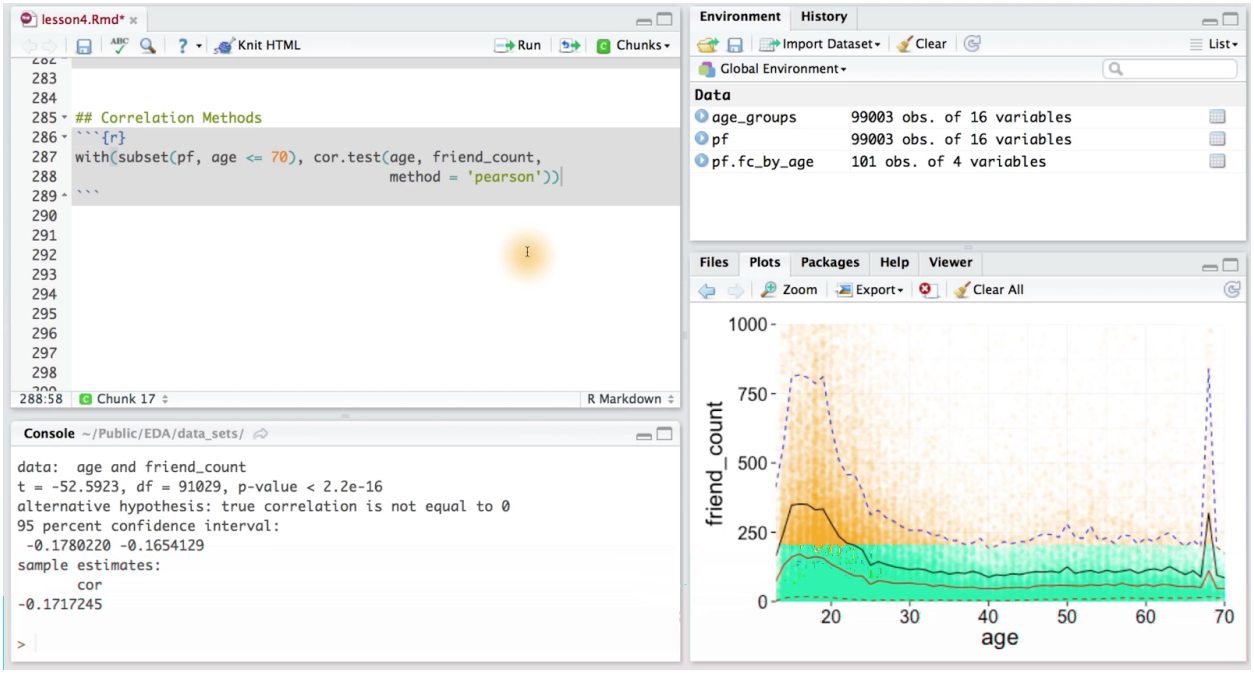
Based on the correlation coefficient in this plot, we just observed that the relationship between age and friend count is not linear. It isn't monotonic, either increasing or decreasing. Furthermore, based on the plot, we know that we maybe don't want to include the older ages in our correlation number. Since older ages are likely to be incorrect. Let's re-calculate the same correlation coefficient that we had earlier with users who are ostensibly age 70 or less. What command would you use to subset our data frame so that way we can get a correlation coefficient for just this data?

Answer:

We need an expression here in order to subset our data frame. So really I'll just run the subset command on our pf data frame or pseudo-Facebook users, and it will take ages that are less than 70. Now the question said 70 or less so I should really use less than or equals here. Running this code we get a very different summary statistic. In fact, this tells a different story about a negative relationship between age and friend count. As age increases, we can see that friend count decreases. It's important to note that one variable doesn't cause the other. For example, it'd be unwise to say that growing old means that you have fewer internet friends. We'd really need to have data from experimental research and make use of inferential statistics. Rather than descriptive statistics to address causality. You may have noticed that I left off the method parameter from cor.test. And that's because cor.test defaults to using the Pearson Product-Moment Correlation. No matter what we do. Adding this in as a parameter, we should get the same result. And running this code, sure enough, I do.

Correlation Methods

The Pearson product-moment correlation measures the strength of relationship between any two variables, but there can be lots of other types of relationships. Even other ones that are monotonic, either increasing or decreasing. So we also have measures of monotonic relationships, such as a rank correlation measures like Spearman. We can assign Spearman to the method parameter and calculate the correlation that way.



Here we have a different test statistic called row, and notice how our value is slightly different as well. You can learn more about the assumptions and uses of the different methods in the instructor notes. From these last couple correlation videos, the main point I want to make here is that single number coefficients like this are useful, but they are not a good substitute for looking at a scatter plot and computing conditional summaries like we did earlier. We have a richer understanding by looking at such plots like this one for friend_count and age.

Programming Quiz: Create Scatterplots

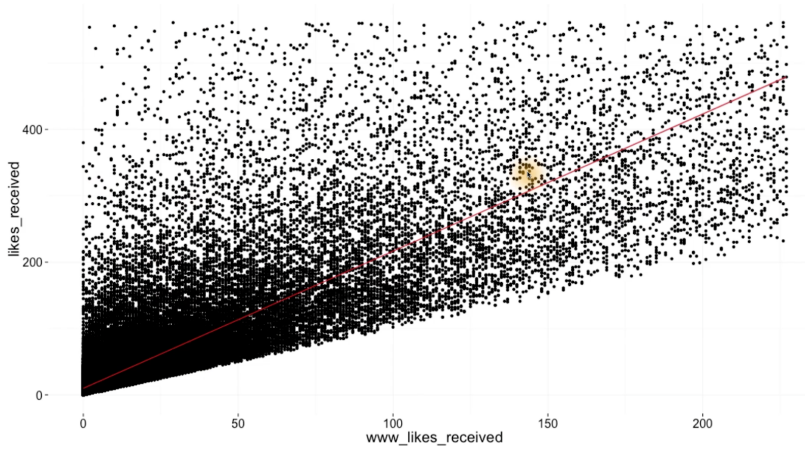
Let's continue our investigation by looking at some variables that are highly correlated. This time we'll look at the number of likes users received from friends on the desktop version on the site. This is the www likes received. We'll compare this variable to the total number of likes users received which is the likes received variable. Now you could of course get likes received through the mobile version but we're not going to look at that here. So now I'm going to hand it off to you. I want you to create a scatter plot of this variable versus this variable and it should make sense that likes received should be higher than www likes received since some of the likes may be coming from other sources, either mobile or unknown.

Answer:

To create this plot, we're going to use our ggplot function.We'll pass it www likes received to the x variable, and we'll pass likes received to the y variable. Then we'll set our data frame equal to our pseudo Facebook users and then we'll tell ggplot that we want a geom_point or scatterplot. Now I actually have forgotten something to do here, I need to wrap my x and y variables inside the aesthetic wrapper. So I'll add that in. Now you might not have used the ggplot syntax, but I highly encourage you to continue to use it and to get practice with it throughout this lesson. We'll be making more use of the ggplot syntax in the next lesson, so it's important that you really know it backwards and forwards. Try making some other scatter plots with different variables to get more familiar with the syntax.

Programming Quiz: Strong Correlations

Looking at this plot, we can see that we have some funky outliers in here. And down here is the bulk of our data. To determine good x and y limits for our axis, we can look at 95th percentile, using the quantile command. This will let us see the 95th percentile of www_likes_received. And the 95th percentile of likes received.



And hopefully, we should get rid of some of these points. To do that, I'll use the x lim layer and the y lim layer. We'll pass zero as the lower bound for x lim, and for the upper limit, we'll use the ninety-fifth percent quantile for the www likes received. Similarly, for likes received, we'll use the same sort of syntax and just replace the variable. Zero will be our lower bound, and the ninety fifth percentile for likes received will be our upper bound. When I run this code, we're in effect zooming in on that lower portion of the graph that we had over here. The slope of the line of best fit through these points is the correlation. And, we can even add to the plot by using some code. We do that by adding a smoother, and setting the method to a linear model or lm. Notice too that I also colored it red so that we could see it through the black points. Let's quantify this relationship with a number. So what's the correlation between our two variables? I don't want you to have to subset the data, so just include all the data points and then round your answer to three decimal places.

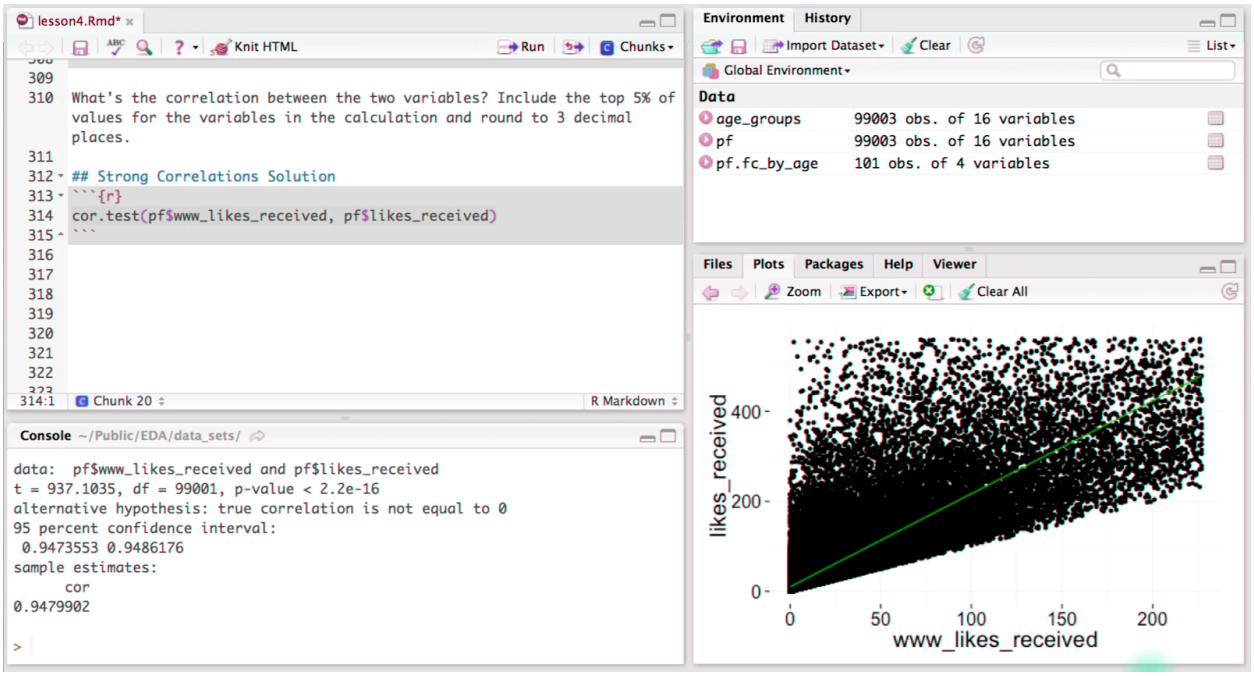
Answer:

To determine the correlation coefficient, we're just going to run our cor.test function. We'll pass it our two variables, www likes received, and likes received. This gives us a correlation of .948. This is a strong positive correlation, and in reality most variables are not correlated that closely. The correlation that we just found was an artifact of the nature of the variables. One of them was really a superset of the other.

Moira on Correlation

Strong correlations might not always be a good thing. Let's hear from Moira about why we look at correlation in the first place, and what it can tell us about two variables.

So in addition to doing scatter plots where you can visually see how related two variables are typically, I will actually measure their correlation coefficient to really quantify how correlated they are. This is really important, because a lot of the data that I work with is correlated with each other. So, for example, looking at Facebook data, how many status updates someone posts in a month is really highly correlated usually with how many days in the last month they logged in, or how many friends they have, or how many photos they uploaded in the last Month. All of these variables are typically very highly related, and it's usually because they all kind of measure the same thing. It's how engaged someone is. And so typically, when I'm working on a problem and I'm going to be doing some kind of regression where I'm modeling something, I'm going to be throwing some of these variables into the regression. And one of the assumptions of regression is these variables are independent of each other. And so if any two are too highly correlated with each other, it will be really difficult to tell which ones are actually driving the phenomenon. And so it's important to measure the correlation between your variables first, often because it'll help you determine which ones you don't actually want to throw in together, and it might help you decide which ones you actually want to keep.



Programming Quiz: More Caution with Correlation

As Moire put it out, correlation can help us decide which variables are related. But even correlation coefficients can be deceptive if you're not careful. Plotting your data is the best way to help you understand it and it can lead you to key insights. Let's consider another data set that comes with the alr3 package. You'll need to install this package first and then make sure you load it. The data set that we're going to load is called the Mitchell Data Set. The Mitchell Data Set contains soil temperatures from Mitchell, Nebraska. And they were collected by Kenneth G Hubbard from 1976 to 1992. By working with this data set, we'll see how correlation can be somewhat deceptive. So, for your first task, I want you to create a scatter plot of temperature versus months.

Answer:

To create this scatter plot, we're going to use our GG plot function. And we're going to pass it the Mitchell data frame. We'll pass our month variable to x and our temperature variable to y and we'll wrap that in the AES wrapper. And the last thing I'll do is add GM point so that way we create a scatter plot. And there it is. If you use the qplot syntax, it would have been a little bit shorter. And we still get the same plot. Again, q plot is pretty smart here since it automatically knows that x is the month variable and y is the temperature variable. These variables are passed in alphabetically to the parameters x, and then y. Because both x and y are used, qplot is to make a scatter plot.

Programming Quiz: Noisy Scatterplots

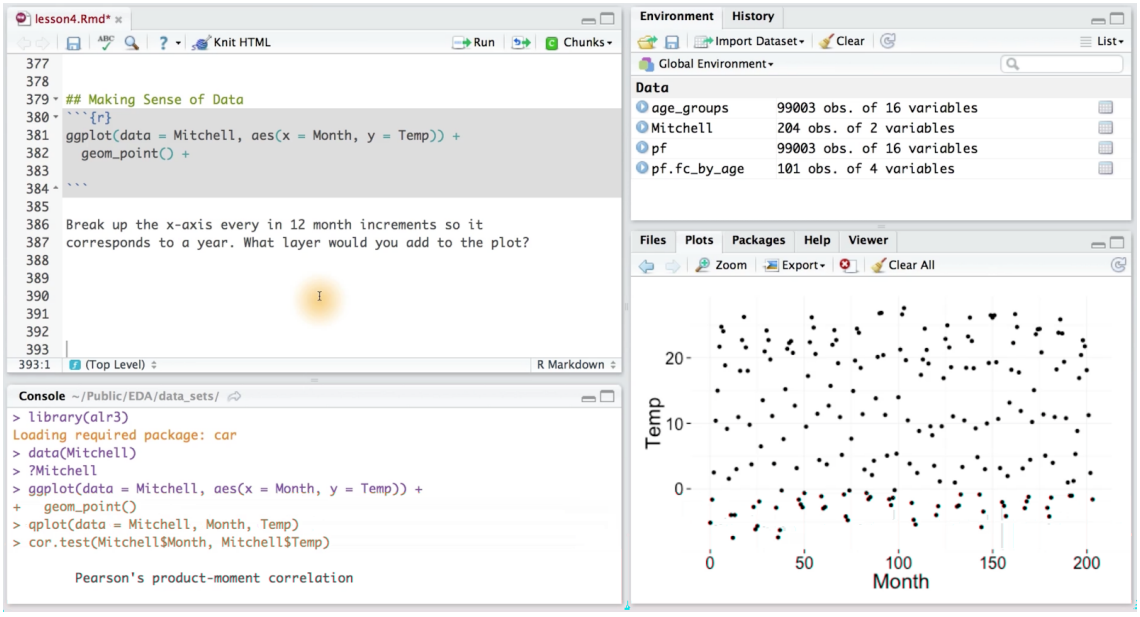
Now that we've got a scatter plot. I want you to take a guess as to what you think the correlation coefficient is, between these two variables. Write that answer in this box. After you've made your guess, I want you to calculate the actual correlation between the two variables and put that number here. Be sure you round your answer to the thousandths place or to three decimal places.

Answer:

Now, if I was looking at the scatter plot, I would say that the correlation between these two variables is about zero. Now, the autograder would have accepted anything between negative 2/10 and positive 2/10. We just wanted something reasonable. Now, to actually calculate the correlation, we'd run cor.test function on Mitchell Month and Mitchell Temp. It turns out that the actual value is .57000, seems like a pretty weak correlation to me.

Programming Quiz: Making Sense of Data

While there might not appear to be a correlation between the two variables, let's think about this a little bit more. What's on the x-axis? Months. And what's on the y-axis? Temperature. So, we know that this month variable is very discreet. We'll have the months from January to December, and they'll repeat over and over again. Let's be sure we add that detail to our plot. Here's the code that we used to create the plot using the ggplot syntax. What layer and syntax would you add to this plot in order to make that change? Type in your code here. You'll want to break open the x-axis every 12 months, so that way it corresponds to a year. I'll let you figure out what the lower and upper limits of the x axis should be. So, what do you think? What layer and syntax should we add?



Answer:

For this question, you need to add a layer to this plot to make the months be discrete. To do this, we can add our scale_x_discrete layer. We're going to pass it the parameter breaks, and set the breaks from zero to 203. Now I'm going to step every 12 months. Since 12 months is a year. And you might be wondering how I decided upon 203. If I type in range mitchell month I can see the range of the month variable, and it runs from zero to 203. So this is my lower limit. And my upper limit. We can run this code, and we'd get a little bit of a better plot with our months in discrete units.

Programming Quiz: A New Perspective

Now we've got a plot here, and I want you to take a new perspective on it. Take your plot and stretch it out horizontally as much as you can in r. Make the horizontal axis be longer than the vertical axis. What do you notice?

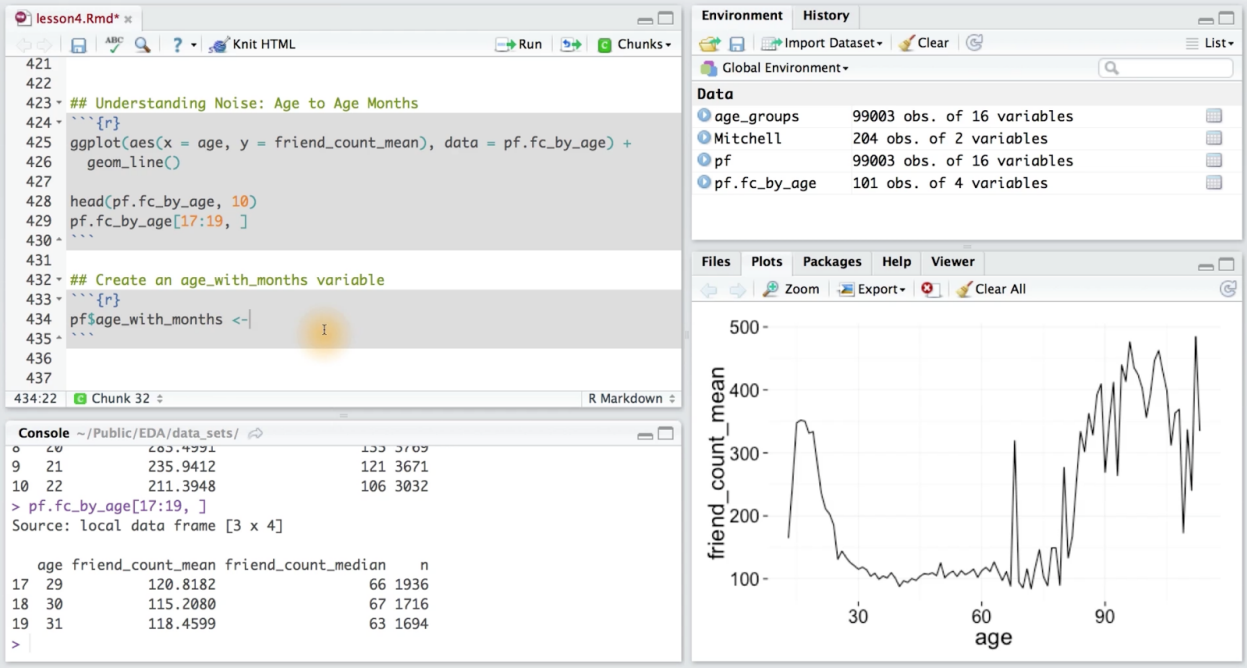
Answer:

When I stretch out of the graph, I notice that I get more of a cyclical pattern. It's almost like a sine or a cosine graph. And this makes sense with what the story the data's telling. I mean, there are seasons in Nebraska, so we should see fluctuation in the temperature every 12 months. This is one example of how it's so important to get perspective on your data. You want to make sure you put your data in context. Another important point to make here is that the proportion and scale of your graphics do matter. Pioneers in the field of data visualization such as Playfair and Tukey studied this extensively. They determined that the nature of the data should suggest the shape of the graphic. Otherwise, you should tend to have a graphic that's about 50% wider than it is tall.

Programming Quiz: Understanding Noise Age to Age Months

Let's return to our scatter plot that summarized the relationship between age and mean friend count. Recall that we ended up creating this plot from the new data frame that we created using the dplyr package. The plot looked like this. As you can see, the black line has a lot of random noise to it. That is, the mean friend count rises and falls over each age. Let's print out some of our data frame to have a closer look. As we can see, the mean friend count increases, then decreases later. In one particular case, we can see that for 30 year olds, the mean friend count is actually lower compared to the 29 year olds and the sense, such as the spike at age 69. But others are likely just to be noise around the true smoother relationship between age and friend count. That is, they reflect that we just have a sample from the data generating process. And so the estimated mean friend count for each age is the true

mean plus some noise. We can imagine that the noise for this plot would be worse if we chose finer bins for age. For example, we could estimate conditional means for each age, measured in months instead of years. Over the next few programming exercises, you're going to do just that. You're going to create a plot just like this one with a new variable that measures ages in months instead of years.



Then you'll plot the conditional mean for ages in months, and we'll compare this graph to the one that you create. To start, you're going to create the age with months variable, and save it into the data frame. This variable will have each user's age measured in months rather than in years. So, if a user is 36 years old and was born in March, the user's age would be 36.75. Try coding this up in R for yourself. And then once you have the code, copy and paste it into the browser and submit. Now, this is one of the exercises where the grader will automatically check your output. Don't worry if you don't get this one right on your first try. It's pretty tough. I really recommend thinking about ages and people being born in different months. How would that affect the variable age with months? Working with actual values might help you here.

Answer:

To convert age to age with months, we know we're going to need to add some fraction to the age variable. Since there are 12 months in a year, we know 12 will be the denominator. Now, let's think carefully about age. For a given year, someone who was born in March would be older than someone born in September. So we need to subtract the birth month from 12 to reflect this. This should make sense, since someone born in March would be born on the third month of the year. So our numerator here would be seven. So for the 36 year old born in March, their age would be 36.75. If the user was born in September and was also 36, then the user's age would be 36.25. So let's run this bit of code and convert our age variable, measured in years, to age with months. And it looks like I made a

mistake. I forgot to include my data frame for DOB month. Now we're ready to go.

Programming Quiz: Age with Months Means

We're on our way to plotting the conditional means for a to months. Remember we're trying to generate this plot again but only for smaller bin widths. We'll have more data points since age will be measured in months rather than in years. Now that we've got our age with months variable from before we can go ahead and use the d apply r functions. To get a new data frame with an average friend count. And the median friend count for each age with months. So here comes your second programming task. Create a new data frame called pf.fc_by_age_months that contains the mean friend count, the median friend count, and the number of users in each group of age with months. There will be detailed instructions on this on the next page and an example of what your output should look like. Now this is one of the more challenging programming exercises. So if you get stuck, check out the instructor notes for two hints. This exercise will also be automatically graded and will check the output of your data frame.

Answer:

Your goal is to create a similar data frame to the fc_by_age one. This time though, we wanted it to have friend counts by age months instead of age years. Here's one solution that you might have done, by chaining the operations of dplyr together. So first, I want to make sure that the library is loaded and then I'm going to create my data frame, pf.fc_by_age_months. So, I'm going to take my original data frame and apply a bunch of functions to it, from the dplyr package. First, I'll group by age with months, then I'll chain on a new command called summarize, and here I want to summarize by friend count mean, and friend count median, and the number of users in my group. And finally, I'll chain on one more command that will arrange my data frame by age with months. Running this command, I can see that I have my new data frame. Notice too, that I have a lot more observations. And that should make sense, because I went from age years to age months. And just to be sure, I'll print out a couple rows on my data frame to examine it. There's my age measured in months, my friend count mean, my friend count median, and n, the number of users in each group. Now, there was another way to get the same data frame. Let's see how we can do that. Instead of chaining the commands together, I can use the data frame and then apply commands to it. So first, I'll create age_with_months.groups. I'll use that using the groupby command. I'll pass it my data frame, and then I want to group, by age_with_months. That's the variable. Now that I have my groups, I want to summarize them using mean friend count, median friend count, and in, which is the number of users in each group. So here I'll summarize age-with-months, and I'll save it into this new variable. Now that I have my groups, I want to summarize them using the Summarize command. I'll create a new data frame called, pf.fc_by_age_months2. I want to summarize this data frame, since it's already in groups. So I'll pass it here. Age_with_month groups. Now I just need to add the variables that I want

to summarize. I want the mean of friend count, so I'll save that to a variable, I want the median friend count so I'll also save that to a variable. And finally I want the number of users in each age group. Now I just want to take this data frame and arrange it by age_with_months. So I'll pass this data frame into the arrange function, and then I'll tell it to arrange by age_with_months. Notice, too, that I'm saving this new data frame into our old data frame, so I'm just writing it over. Running all this code, we can see that we get our new data frame, and we can also check it just by using our head function. Sure enough, the same exact data frame.

Programming Quiz: Noise in Conditional Means

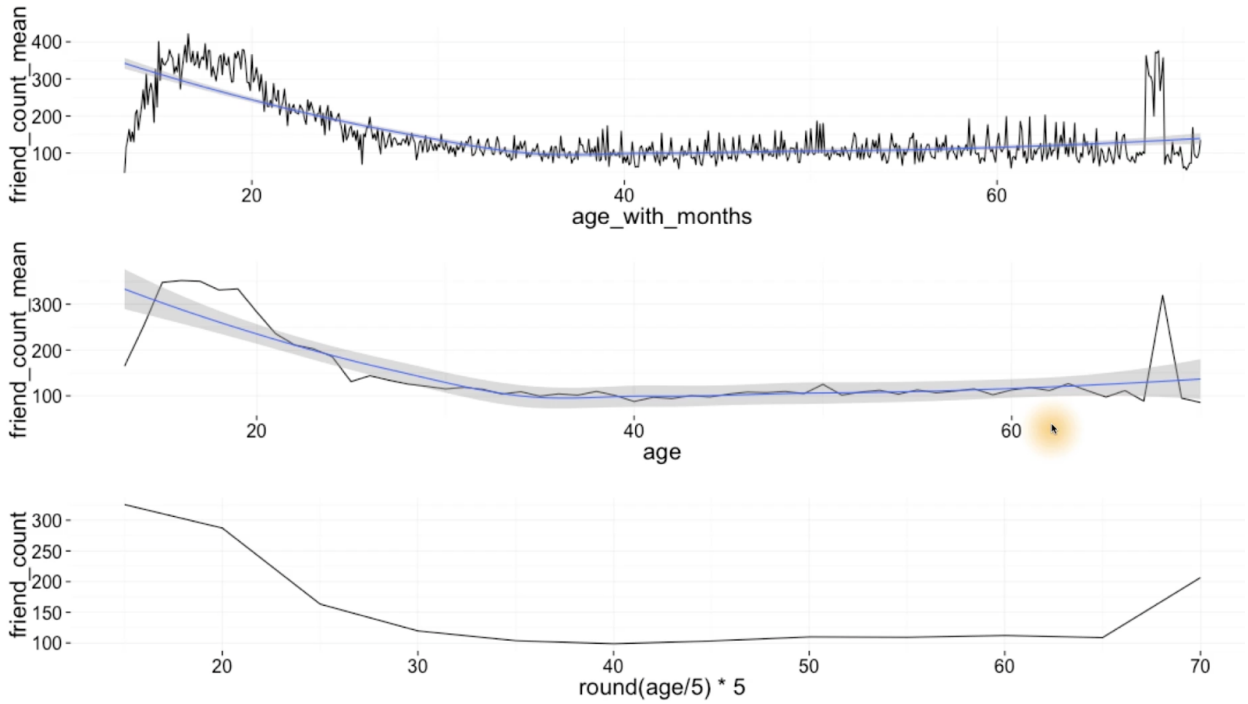
This is great. We've got our data frame, with our conditional means measured in months. Now it's just time to plot them. Your final programming task here is to make the plot of mean friend count versus age measured in months. Be sure that you use a line to connect all the points just like we did in this plot. Now I'm going to make this a little bit harder. I want you to subset the data frame as well. I want you to only investigate users with ages less than 71. The plot that you create won't be automatically graded so I encourage you to check out what we did in the solution video.

Answer:

For this exercise, you need to create a similar scatter plot that had friend count mean against age with months. We'll use ggplot to create our figure. I'll pass age with months to x, and I'll pass friend count mean to y. And then I just need to remember to wrap this in aes. Now comes the data frame. I need to be careful that I don't use pseudo Facebook users, since I really want this data frame that we just created. Now that I've got my data frame, I need to subset it. I'll only take the users whose age with months is less than 71. Then, I'll tell ggplot what type of geom I want, in this case geom live. So here's our much noisier plot, a friend count mean versus age with months.

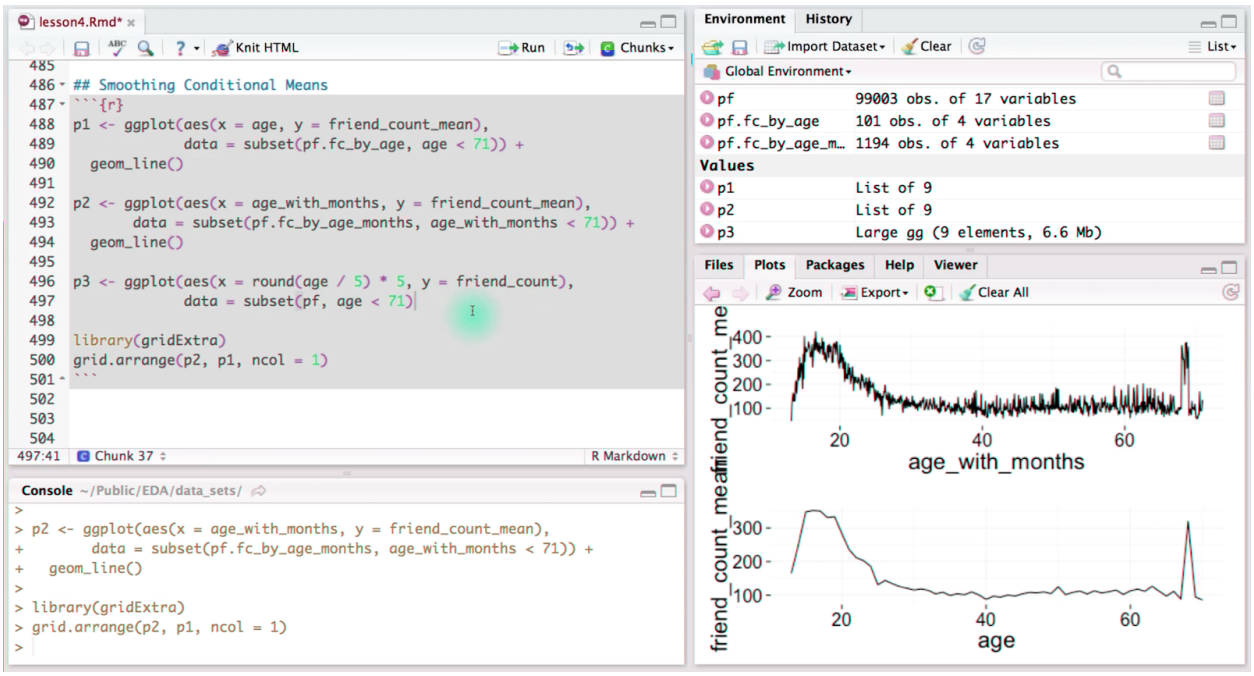
Smoothing Conditional Means

In this lesson, you created two plots for conditional means. Let's take a closer look at both of the plots and see how they're different. This second block of code gave us this plot. And this first block of code gave us this plot. Now, you subset this data frame to only consider uses who are age 71 or less. So, let's do the same up here. Running the code, we can see that we're limiting our x axis. Now, what I want to do is put these two plots side by side so we can look at them together. Now, you know this before, we basically just say, each plot into a variable, and then we plot those variables in one column. So, here's the difference between age and age with months.



By decreasing the size of our bins and increasing the number of bins, we have less data to estimate each conditional mean. We can see that the noise is a lot worse on this graph since we have finer bin choices. On the other hand, we could go the other direction and increase the size of the bins. Say, we could lump everyone together whose age falls under a multiple of five. Essentially what we'll do is, we'll cut our graph in pieces and average these mean friend counts together. So, users who are within two and a half years of 40 will get lumped into one point. The same will be true for users who are within two and a half years of 50 and for users who are in two and a half years of 60. I'll show you what I mean in code. Here, I'm creating a plot with age that's been divided by five, rounded and then multiplied by five. I've also subsetted our data frame, just like the other plots. The last thing I'll do is I'll add a geom line with a stat summary. I don't really want to plot the friend count, I want to plot the mean friend count.

So I'll pass summary to stat, and I'll pass mean to fun.y. I'll save this plot, and add it in with the others. So, see how we have less data points here? And wider bin widths. By doing this, we would estimate the mean more precisely, but potentially miss important features of the age and friend count relationship. These three plots are an example of the bias variance tradeoff, and it's similar to the tradeoff we make when choosing the bin width in histograms. One way that analysts can better make this trade off is by using a flexible statistical model to smooth our estimates of conditional means. ggplot makes it easier fit such models using geom smooth. So, instead of seeing all this noise, we'll have a smooth modular function that will fit along the data. We will do the same for this plot as well. Here, I've added the geom smooth layer to both our first plot and our second plot. I'm just using ggplot's defaults so all the decisions about what model we'll be using will be made for us. If you're interested in exploring the models and the parameters, take a look at the geom smooth documentation.



So, I'll save these two plots and then I'll run the code again. So, here's our smoother for age_with_months, and here's our smoother for age. While the smoother captures some of the features of this relationship, it doesn't draw attention to the non-monotonic relationship in the low ages well. Not only that, but it really misses the discontinuity at age 69. This highlights that using models like lowess or smoothing splines can be useful. But, like nearly any model, it can be subject to systematic errors, when the true process generating our data isn't so consistent with the model itself. Here the models are based on the idea that true function is smooth. But, we really know that there's some discontinuity in the relationship.

Which Plot to Choose

So, we've been looking at lots of different plots of the same data, and talking about some of the trade-offs that are involved in data visualization. So, which plot should you choose? One important answer is that you don't have to choose. In exploratory data analysis, we'll often create multiple visualizations and summaries of the same data, gleaning different incites from each. So, throughout the course, as we iteratively refine a particular plot of the same data, it's not that the later versions are always better than the previous versions. Sometimes they are. But, sometimes they're just revealing different things about the same data. Now, when it comes time to share your work with a larger audience, you may need to choose one or two visualizations that best communicate the main findings of your work.

Programming Quiz: Analyzing Two Variables

Wow, that was a lot. We covered scatter plots, conditional means, and correlation coefficients. Now I want you to take some time to think about everything that you learned in this lesson. Capture some of your ideas and then write them down in the text box that will appear right after this video. Once you've submitted your thoughts, you can compare your thoughts to our instructors and the solution.

Answer:

Congratulations on finishing lesson four. In this lesson, we learned how to explore the relationship between two variables. Our main visualization tool, was the scatter plot. But we also augmented the scatter plot, with conditional summaries, like means.

We also learned about the benefits and the limitations of using correlation. To understand the relationship between two variables and how correlation may affect your decisions over which variables to include in your final models.

As usual, another important part of this lesson was learning how to make sense of data through adjusting our visualizations. We learned not to necessarily trust our interpretation of initial scatter plots like with the seasonal temperature data. And we learned how to use jitter and transparency to reduce over plotting.

We'd love to hear your thoughts in the discussion forum and we hope you learned a lot. Next, we'll move onto lesson five, where we talk about multivariate analysis.

Exploratory Data Analysis

Lesson 5 Notes

Explore Many Variables

Multivariate Data

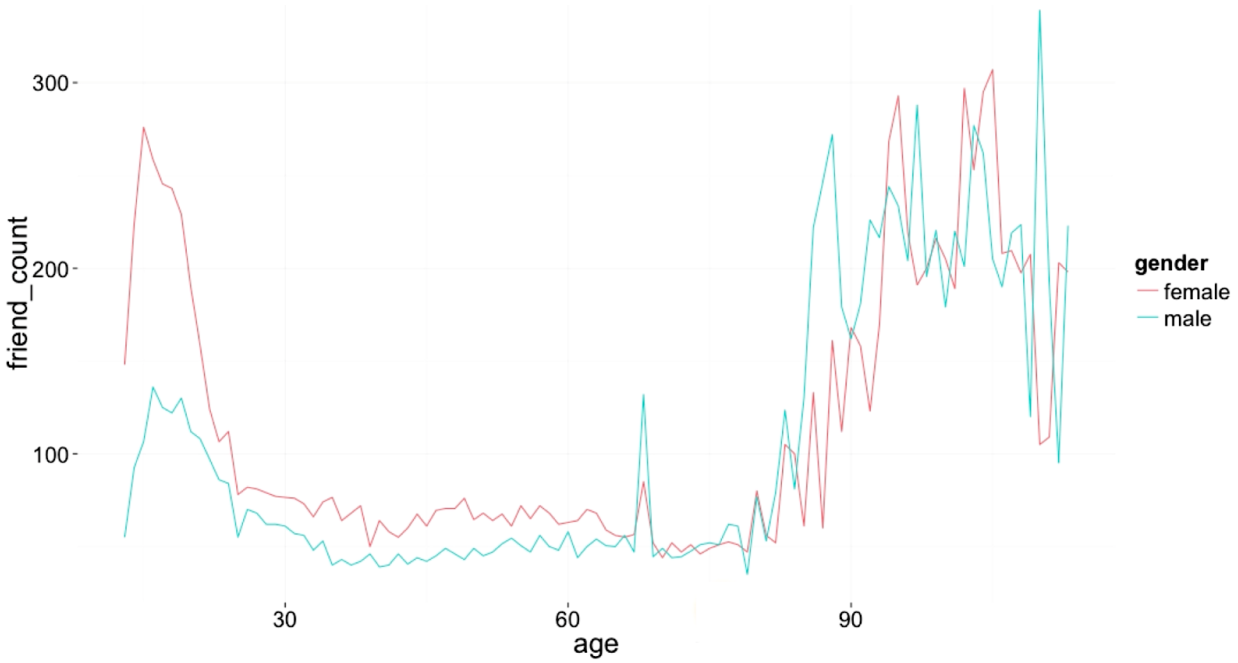
In the last lesson, you learned how to examine the relationship between two quantitative variables. In this lesson, you'll learn how to examine three or more variables at a time. Let's get started by hearing from Moira about how she added one more variable to her scatter plot of perceived audience size.

Perceived Audience Size by Age

My next question was, well, okay, so people aren't very good at guessing their audience sizes. But maybe, maybe people who are older, maybe they have a better sense, than teenagers, for example. So the next plot that we did was another scatter plot, but this time we added a third level, where we added color to represent the age of the survey respondent. And you can see again we have this horizontal stripes for people who are guessing that there are 50 or 100 people in their audience. But I don't see any pattern in the color. I think this is actually kind of a dead end. This is one example of kind of a failure. I can tell if younger people were more accurate than older people, there is too much over plotting. In this plot there are too many dots on top of each other and color really doesn't add much.

Programming Quiz: Third Qualitative Variable

It looks like Moira didn't really get anywhere by adding age as color to her plot. But I want you to know that's okay. Sometimes when we conduct exploratory data analysis we do reach dead ends. Let's see if we can get any further in examining our relationship between friends count and age by adding a third variable. Previously we noted that female users have more friends on average than male users. And, we might wonder, is this just because female users have a different age distribution? Or, maybe conditional on age, the differences are actually larger.



Here's a box plot of ages by gender. Now, I'm going to add the mean for each gender to the box plots, using stat summary. Here we can see the averages marked by an x since I used shape=4. Since male users are a bit younger, we might actually think a simple male to female comparison doesn't capture their substantial differences in friend count. Let's look at median friend count by age and gender instead. A lot of this code should look familiar, and if it doesn't, I want you to post any questions that you have about it in a discussion. When I run this code, we can see that nearly everywhere the median friend count is larger for women than it is for men. Now there are some exceptions, and this includes these noisy estimates for our very old users. Now, I'm using old with quotation marks here, since we're not really confident about these reported ages. And notice that users reporting to be of reported gender. You're going to reproduce the same plot, but first let's see if you can create the summary data to create it. Recall that we can produce the same summary data underlying this plot by using the dplyr package. We can divide the data by age and gender and then compute the median and mean friend count for each sub-group. In this next program assignment you're going to do just that by using the group buy, summarize, and arrange functions from the dplyr package.

Answer:

For this question, you need to create the data frame that would give us the data to construct this plot. Here's how I would go about constructing the code. First, I'll load the dplyr package and then I'm just going to leave myself a comment that I'm going to chain these functions together. So I'm using this symbol. I'm going to save my data frame as pf.fc by age gender, and I'm going to work from my existing data frame and group it. So, here's my first chain command, and I'm going to group by age and gender. Now, I'm going to summarize getting the mean friend count, the median friend count, and n, the number of people in each group. Now, summarize will remove one layer of grouping when

it runs, so we'll remove the gender layer. So, we need to run ungroup one more time to remove the age layer and finally I'll arrange my data frame by age. Now, it looks like that I have everything, but I actually forgot to filter or subset the data. I could use the subset command but I'm actually going to show you the filter command. I'll filter the data so that I remove any people that have a gender marked as NA and then, I just need to remember to chain that together with the rest of the functions. Alright, let's run this code and see if our data frame looks good. It looks like I actually forgot to run my library first, so let me do that. We have our new data frame up here, so let's print out some of the rows to make sure we're right. I'll print out a couple of the first rows to the console, and I can see that I have my groups split by age and gender, the mean friend count, the median friend count and n, the number of groups.

Programming Quiz: Plotting Conditional Summaries

Now that you've got this data, I want you to construct this plot that shows the median friend count for each gender as age increases. Now, the plot should be identical to this one that we created before. You code just will look slightly different, and you want to make use of this data frame.

Answer:

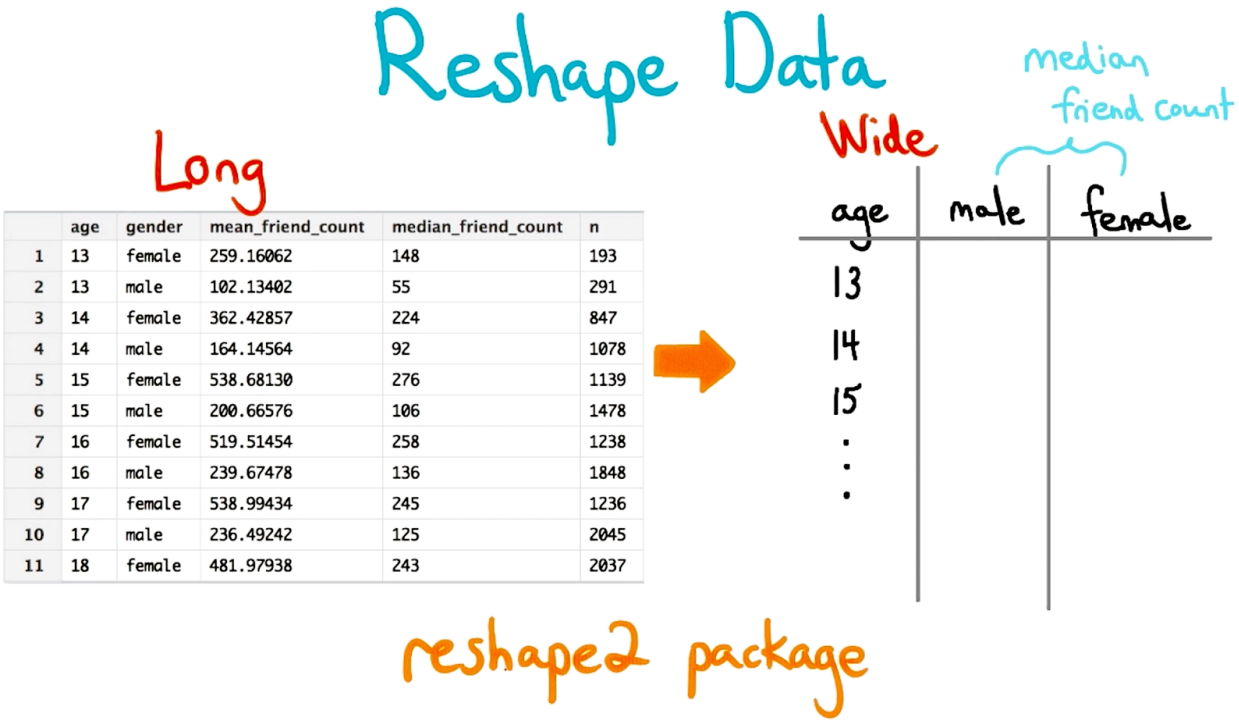
To create this plot, we're going to pass in our variables to ggplot as usual. We'll set x equal to age, and y equal to median friend count. And we'll set data equal to the data frame that we just created. So pf.fc by age, gender. Now I just need to add gm line and pass up the color parameter inside the aesthetic wrapper. This time the color will take on the value of gender. This will give me one line for males and one line for females. Running this code, we get the same exact plot. Hopefully you're feeling like a data guru at this point.

Thinking in Ratios

This can be useful if we want to inspect these values, or carry out further operations to help us understand how the difference between male and female users varies with age. For example, looking at this plot, it seems like the gender difference is largest for our young users. It would be to put this in relative terms though. So, let's answer a different question. Let's answer the question, how many times more friends does the average female user have than the male user? Maybe, females have twice as many friends as male users, or maybe it's ten times as many friends.

Wide and Long Format

To answer that question, we need to rearrange our data a little bit. Right now, our data is in long format. We have many rows. And, notice how that the variables that we grouped over, male and female, have been repeated. They're repeated for each year. So let's do something else besides this long data format. What we're going to do is convert it to a wide format. This new data frame will have one row for each age, and then we'll put the median friend count inside of males and females. Many times when computing with and exploring data, it's helpful to move back and forth between these different arrangements. To carry this out in R, we're going to be using the reshape2 package. Now, if the difference between wide and long data is still a little bit fuzzy to you, I recommend pausing the video right now. You can read through another example in the instructor notes before moving on.



	age	gender	mean_friend_count	median_friend_count	n
1	13	female	259.16062	148	193
2	13	male	102.13402	55	291
3	14	female	362.42857	224	847
4	14	male	164.14564	92	1078
5	15	female	538.68130	276	1139
6	15	male	200.66576	106	1478
7	16	female	519.51454	258	1238
8	16	male	239.67478	136	1848
9	17	female	538.99434	245	1236
10	17	male	236.49242	125	2045
11	18	female	481.97938	243	2037

Reshaping Data

If you're ready, let's go through the code to reshape our data frame into a new one. First, let's install and load the R reshape two package. Now, let's create a variable for a new data frame that will be in wide format. I'll use the same variable name for the data frame and just add wide to the end. Now, we're going to make use of the dcast function, which comes with the R shape two package. The letter d is used since we want the result to be a data frame. If we wanted an array or a matrix we could use a cast. We specify the data set we are going to change and then we put in a formula. So, here's the data frame I want to modify and then this is where I'll enter my formula. Now, the first part of the formula, or the part to the left of the tilde sign, will list the variables I want to keep with an addition sign in between them. Here I just want to keep h. On the right side of the tilde, we use the gender variable since we want male and female users to have their own columns for median friend count in the data frame. And finally, we set value dot var equal to median friend count because value dot var holds the

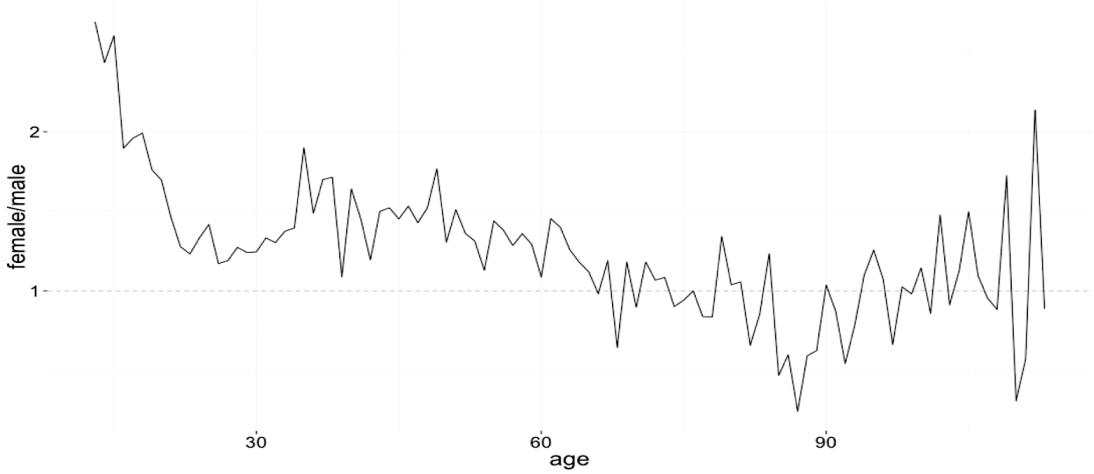
key measurements or their values in our new data frame. And it looks like that I forgot quotes around this variable.

So let me add that. Now, we should get a new data frame. And let's look at some of these data frames, so that way, we can make sure we understand this dcast function. Here are some of the rows printed out and notice I have my age column, my female median friend count and my male median friend count. See if you can recreate these steps on your own and try playing around with the dcast and melt functions, and the reshape to package. The melt function will allow you to convert the wide data back to the original long format.

Programming Quiz: Ratio Plot

We've got our new data frame so let's make use of it. I want you to plot the ratio of females to males to determine how many times more friends the average female user has, compared to the number of friends the average male user has. You'll also add some cool features to the plot, but I'll describe that in the programming exercise.

Answer:



Now, this plot might have been a little bit tricky, so let's walk through it together. I'll assign age to the X parameter, and then I'll assign females divided by males, to the Y parameter. This will give me my ratio. And then I just need to make sure that I pass my newest data frame to data. We'll add a geome line to connect the points, and then we'll also add a horizontal line. This geom_hline will take a couple parameters. We'll set the y-intercept to one, the alpha equal to 0.3, and the line type equal to two. Running this code, we get out ratio plot. We can easily see that for very young users, the median female user has over two and a half times as many friends as the median male user. Clearly, it was helpful to condition on age in understanding the relationship of gender with friend count. This helped assure us this pattern is robust for users of many different ages. And it also highlighted where this difference is most striking. Now, there are many processes that can produce this difference, including the biased distribution from which this pseudo Facebook data was generated. One idea which shows the complexity of interpretation here, is that people from particular countries who more recently joined Facebook are more likely to be male with lower friend counts.

Programming Quiz: Third Quantitative Variable

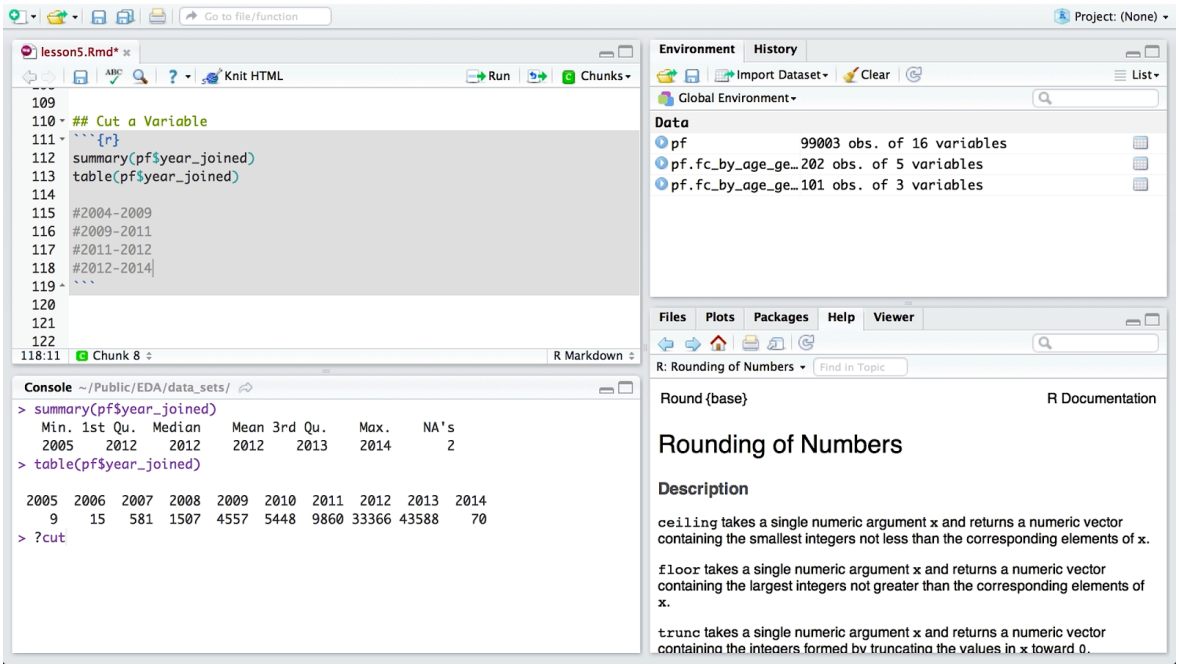
In the previous examples, we were looking at our data Age and Friend Count across the categorical variable Gender. Usually, color or shape tend to be aesthetics for representing such changes over a categorical variable. But what if we looked at Age and Friend Count over, say, another numerical variable? For example we might notice that since users are likely to accumulate friends over time using Facebook that Facebook tenure is important for predicting friend count. Tenure or how many days since registering with Facebook is associated with age. The first people to start using Facebook were college students as of 2004 and 2005. I was lucky enough to be part of that group and joined the site on February and have over 1,200 of them. On the other hand, 14 year-old users, have had less time to accumulate the same number of friends. One way to explore all four variables friend count, age, gender and tenure is using a two-dimensional display like a scatter plot. And we can bend one of the quantitative variables and compare those bends. In this case, we can group users by the year that they joined. So let's create a new variable called year_joined in our data frame. This variable is going to hold the year that our users first joined Facebook. In our next program and exercise, you're going to create this variable, year_joined, and put it inside the data frame. You need to make use of the variable tenure and use 2014 as the reference year.

Answer:

For this programming assignment, you need to create the variable year_joined, and assign it to the data frame pf. To do that, we just need to take our reference year 2014, and subtract off our tenure, or the number of days someone's been active on Facebook. Now, tenure is measured in days rather than in years, so we need to divide this by 365. I also need to make sure that I actually access the variable so I need to include pf right here with a dollar sign. Now, this number will give me a year with a little bit of extra if there's a decimal. Now, I don't really care about the decimal, since that doesn't make a full year, so I just want to floor this number. The function floor will return the greatest integer that's not larger than this number. When I run the code, it doesn't look like much happened. But if I check my data frame, I can see that I have another variable. There it is, year_joined.

Programming Quiz: Cut a Variable

We've got our new variable your joined so let's look at its summary it looks like most of our users joined in 2012 or 2013 and since the values for this variable are discreet and the range is pretty narrow I'm going to go ahead and table this variable as well. Here we can see the distributions of users and each year joined. Notice that there isn't much data here about early joiners. To increase the data we have in each tenure category, we can group some of these years together. The cut function is often quite useful for making discrete variables from continuous or numerical ones, sometimes in combination with the function quantile. Now what I want you to do is to look at the documentation for cut, and refer to the link in the instructor notes to complete this next programming exercise. Your task is to cut the variable year joined to create four bins, or buckets of users. The bins will be from 2004 to 2009, 2009 to 2011, 2011 to 2012, and then 2012 to 2014.



Answer:

For this programming exercise, you need to create a variable called, year joined bucket. That bin together users depending on which year they joined Facebook. So, for example, this would be one bucket, this would be one bucket, this would be a bucket, and then these users would be another bucket. To do this, we just need to use the cut function on the variable year joined. I just need to tell cut on what years I should split my data. So I'm going to split at 2004, 2009, 2011, 2012, and 2014. Running this code, I can see that I got a new variable inside of my data frame.

Programming Quiz: Plotting It All Together

We've done two things up to this point. We created a variable called year_joined, based on the tenure variable, and we converted year_joined, to the variable year_joined_bucket. A categorical variable. That bin their users into different groups. Let's table this new variable to see the distribution in each group. Here we can see that we have our four bins of users, depending on when they joined Facebook, and it looks like two people have a value of NA. Let's use this new year joined bucket variable to create a line graph Like we did for gender at the start of the lesson. As a reminder, here's the code that generated this plot earlier. Also, notice how we compute the median friend count for each age using the fun.wide parameter, and the summary for the stat parameter. Using a similar code structure that we see here. I want you to create a plot for friend count versus age, so that each year join bucket has its own line on the graph. In other words, each bucket will be a color line that tracks the median friend count across the age of users, just as it did in this plot for genders.

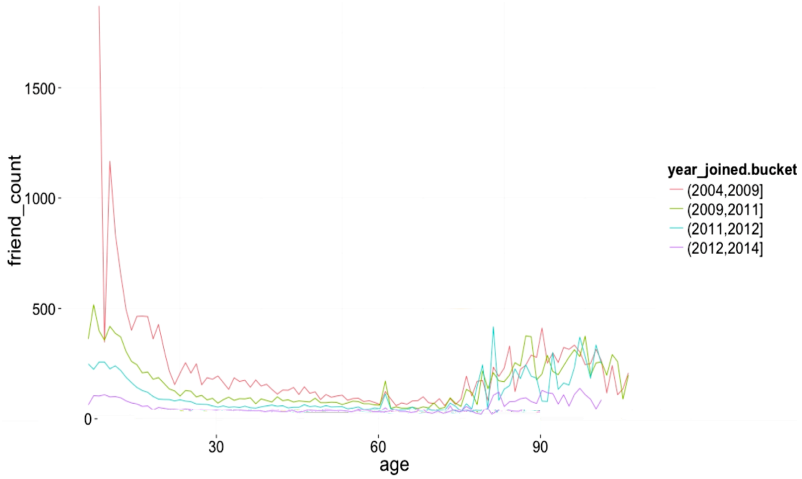
Answer:

To create this new plot, we really just need to switch out gender with our year_joined.bucket variable. This is the new categorical variable that will take the place of color. Now, I'm also using year_joined.bucket here so that way, I can exclude the two people who have values of NA. I'll clean up my code just a little bit, and then I'll run it to create my plot. And now, here's the plot that examines the relationship between friend_count and age, split up by year_joined.bucket variable.

Programming Quiz: Plot the Grand Mean

Looking at this plot, we can see that our suspicion is confirmed. Users with a longer tenure tend to have higher friend counts, with
the exception of our older
users, say, about 80 and up. To
put these cohort specific
medians in perspective, we can

change them to cohort specific means. And then plot the grand mean down here as well. The grand mean is the line we saw in lesson four when we plotted average friend count across the ages. Let's see if you can change the code for this plot to plot the means instead of the medians for each cohort. You'll also want to plot out the grand mean as well, that we talked about in lesson four.



Answer:

To plot the means and study the medians for each of the cohorts, we just need to change our fun.y parameter. This needs to be mean instead. Now, to get the grand mean, we just add a geom line and then set the parameters. We'll have a stat of summary. We'll set fun.y to equal the mean, and then we'll set the line type equal to two. Here, I'm using two so that way, I get a dash line so it stands out in comparison to these others. Plotting the grand mean is a good reminder that much of the data in the sample is about members of recent cohorts. This is the type of more high level observation that you want to make as you explore data.

Programming Quiz: Friending Rate

Since the general pattern continues to hold after conditioning on each of the buckets of year joined, we might increase our confidence that this observation isn't just an artifact of the time users have had to accumulate friends. Let's look at this relationship in another way. We could also look at tenure and friend count as a rate instead. For example, we could see how many friends does a user have for each day since they've started using the service. Let's see if you can create a summary of this rate, that shows how many friends a user has for each day since the user started using Facebook. Subset the data so you only consider users with at least one day of tenure. Once you have that summary answer these two questions. What's the median rate? And, what's the maximum rate?

Answer:

Here we want a summary of the friend rate. So, we can use the with command and subset the data so we only consider users with tenure of at least one day. Now we just want a summary of friend count divided by tenure, which gives us the friends per day since the user's been active. When we run the code, we see that the median rate is about 0.22, and the maximum rate is 417. Now, this is definitely

an outlier, considering our data, since the third quartile is only about 0.5.

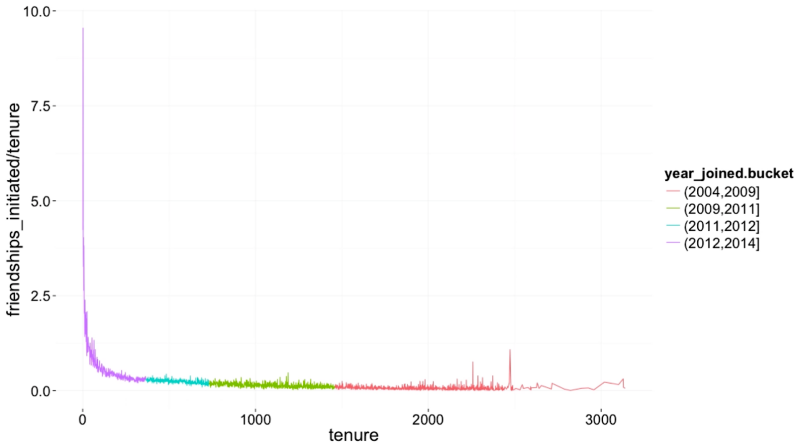
Programming Quiz: Friendships Initiated

We've learned even more about the relationship between age and friend count by making use of the variables' gender and tenure. We've seen that users who have been on the site longer, typically have higher friend counts across ages. Now, this leaves me wondering if friend requests are the same or different across groups. Do new users go on friending sprees? Or do users with more tenure initiate more friendships? Let's explore this with a plot. I want you to create a line graph of friendships initiated per day, versus tenure. You need to make use of the variables age, tenure, friendships initiated, and year_joined.bucket. The color of each part of your one line should correspond to each bucket of year_joined. You'll also need to subset the data to only consider users with at least one day of tenure.

Answer:

To create this plot, we're going to have much of the same code as before. We'll pass tenure to our x variable, and then we'll pass friendships initiated divided by tenure to our y variable, and we'll subset our data frame so that we only consider users who have a tenure of at least one day. We'll color our line by year_joined.bucket and then we'll plot the mean of the y variable across tenure. Taking a closer look, it appears that users with more tenure typically initiate less friendships.

Programming Quiz: Bias Variance Trade off Revisited

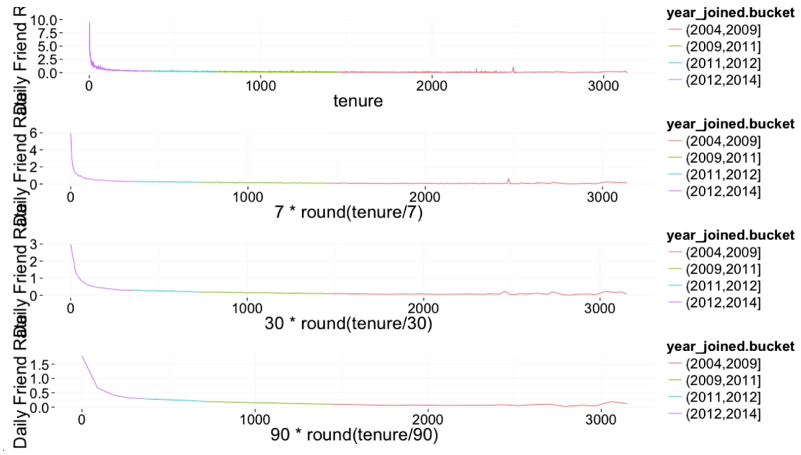


There's a lot of noise in our graph since we are plotting the mean of y for every possible tenure x value. Recall from lesson four that we can adjust this noise by bending our x-axis differently. Let me show you one of those changes in the bend width. Here's our code from before, and instead of using tenure here, I'm going to replace this with a different version or formula so that way I can bend some of the tenures together. Now let's see the difference when I plot this graph. Notice how I have slightly less noise in my plot. We still see some of the same peaks from before, especially here, but it's much

smoother in general. Here's another one using the number 30 instead of the number seven and here's a graph with very high bias, but much less variance using the number 90. Here I plotted all the graphs so we could compare it to the original one. Notice that as the bin size increases we see less noise on the plot. Our estimates are adjusted since we have more data points for our new values of tenure. In lesson four we introduce smoothers as one tool for an analyst to use in these types of situations. So instead of using gm line here, I'd like you to use gm smooth, to add a smoother, to this plot. Try doing this, in this next programming assignment.

Answer:

To add a smoother to this plot, we need to change geom line to geom smooth. We also need to get rid of this fun.y parameter and the stat parameter. We'll still keep year_joined.bucket assigned a color, so that way we see this segment in our graph. And here I'm using the defaults for geom smooth. So R will automatically choose the appropriate statistical methods based on our data. All of that additional information can be found in the output down here. So here again, in this smooth version of the graph, we still see that the friendships initiated declines as tenure increases.

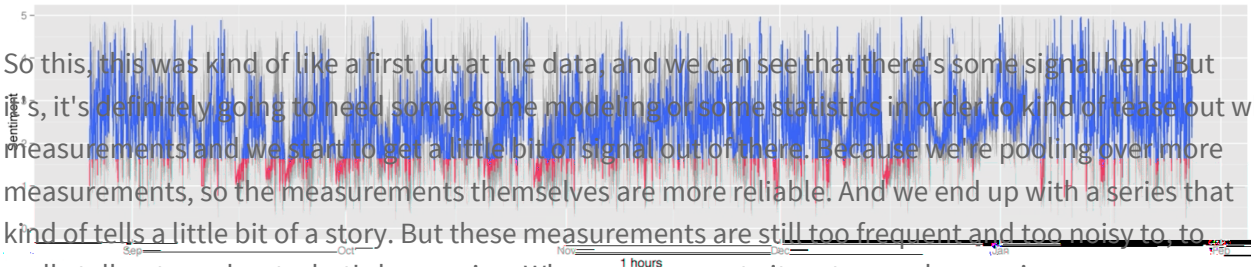


Seans NFL Fan Sentiment Study

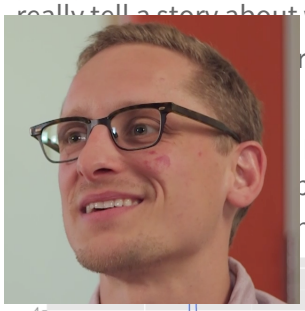
You learned about the bias- ‘‘variance tradeoff in this lesson and the last lesson. Now I want you to hear from Sean. Hear about his work at Facebook and pay careful attention to his models and the trade offs in the visualizations he made.

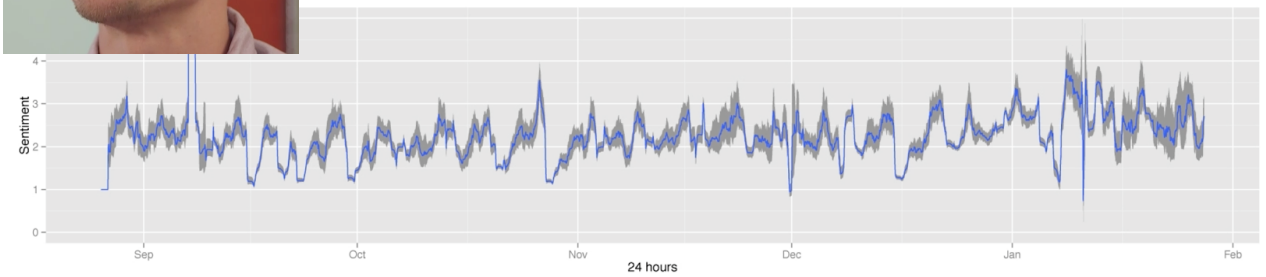
Some recent work I've been doing on measuring fan sentiment for NFL teams over the course of the season that I'm really excited about. it's, it's been a really fun project be I myself am, am an NF, NFL fan. I'm a huge Eagles fan and I go through all of the, emotions tha goes through over the course of the season. You know, the highs of when your team wins and t the lows after a couple losses in a row of feeling kind of hopeless. And so I got the idea of mayb could measure this and kind of tell a story. Not just for my team but for all the other teams in t and come up with some idea, some way of visualizing this, this experience of being a fan. And s counted ratios of positive to negative words at five-minute increments over the course of the w months of the NFL season. And because we're taking a ratio, we end up with some measurements that are extremely

high. Like, you know, positive to negative word ratios, oh, over a hundred even though the average in somewhere in the two to three range.

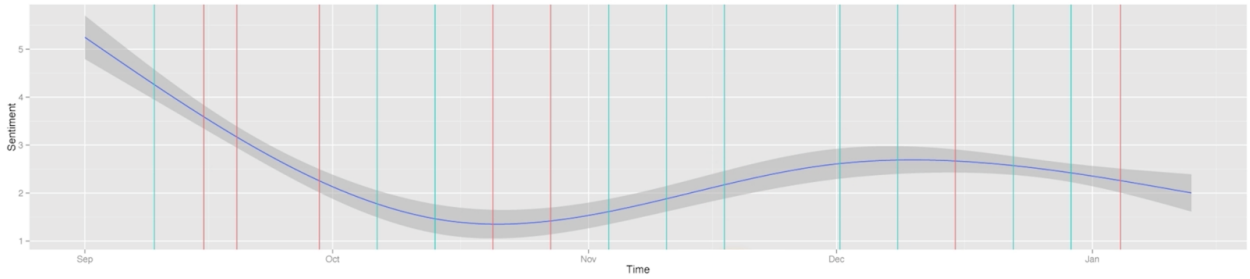


So this, this was kind of like a first cut at the data, and we can see that there's some signal here. But it's, it's definitely going to need some, some modeling or some statistics in order to kind of tease out what's actually ha measurements and we start to get a little bit of signal out of there. Because we're pooling over more measurements, so the measurements themselves are more reliable. And we end up with a series that kind of tells a little bit of a story. But these measurements are still too frequent and too noisy to, to really tell a story about what's happening. When we aggregate it up to one day moving averages, we rns emerge. I guess one of the key features of this dataset was that I knew what of time. Because I, I am an Eagles fan, I experienced the highs and lows of the plot like this immediately and tell you this is not telling the story that I want. pply some modeling to that. We want a model that predicts sentiment as a ne of the things that comes to mind right away is a natural spline.

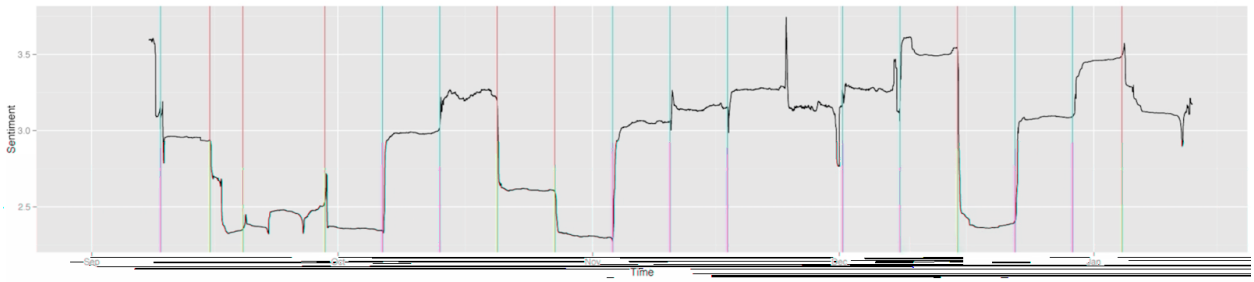




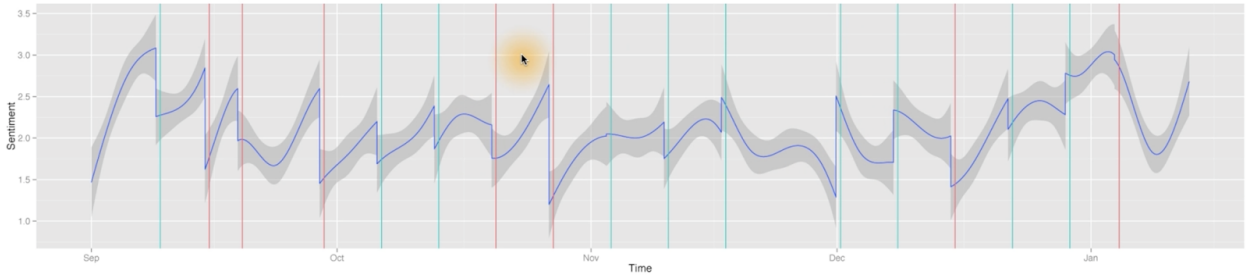
You can see that this actually tells a nice story. These color lines are the dates of wins and loses. It kind of gives you an idea of why the lines are upward sloping or downward sloping. So you can see here, kind of like the exuberance at the beginning of the season as people are really optimistic. And then, you know, three losses in a row and see how the sentiment dips. This tells a nice story, but it doesn't have the feature of, that we'd expect, which is that on game day, things change really abruptly. At the very end of the game, when you know if you've won or you've lost, you're much happier or much sadder than you were at the beginning of the game. So we expect to see really discrete jumps in sentiment that we don't see from a model like this. And this is because this is a bad model of the underlying data generating process.



So we need something more flexible. One way to do that is just to use a seven day moving average which is going to allow us to include only kind of the last game sentiment in the moving average. So we're going to pick a moving average, like I showed you before. Let's smooth it out into over a whole seven day period. And when we do that, we get something that actually tells a really nice story about the season. And has all the kind of characteristics that I would expect as a fan in having gone through this. Which is, the kind of, the big bumps up on game days where you win, the big bumps down on game days where you lose. And then, kind of, these plateaus in between, which are these periods of stability when you don't have any information about how your team is doing. We see that a week off around Thanksgiving but then there's a big spike in happiness because people are just happy around Thanksgiving. This big low point right after a loss that could have knocked them out of the playoffs. And then a big kind of ascension to their playoff game which they ended up losing and the subsequent dip. This I think is a really nice depiction of what happened and it took a little bit of averaging to get the story to come out.



When you're looking at all this data, what sort of things come up for you in terms of bias and variance tradeoffs. When you're computing just a simple moving average like this, you're dealing with one of the most, it's just a really flexible statistic. And so you're, not imposing any structure on the data. You're letting the data kind of speak. When I use a moving average here and I plotted standard errors that were kind of rolling along with the data. They were gigantic. The mean sentiment for the season is somewhere in the three range. And the standard errors were over two or three. We can say very precisely what's happening at any given point. But our variance on that estimate is huge. So as we started to add more lags, higher number of lags to the, moving average. We end up with kind of smoother looking plots, that have lower variance but but then are getting progressively more bias.

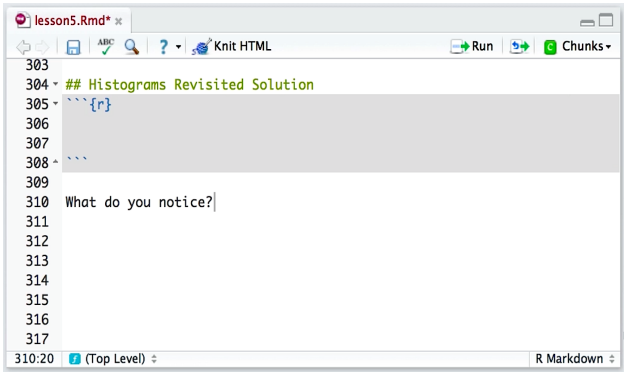


So as we go back further we are including more data and were getting more bias. Because were including data from parts that actually aren't applicable to that exact point. But in exchange for that we get a lower variance plot, one that doesn't move as wildly. When we combine that with splines, we can end up fitting a model, that has kind of the best of both worlds. Which has the smoothing, aspects of the splines, with the discrete jumps, of what happens on game day. And so this is a spline where we add it dummy variables for post game periods. In this model, we end up with kind of all the same thing. We get the big jumps that we expect, so it jumps down on loses, jumps up on game days where they win. And then also kind of the smooth transitions in between. So it's kind of a nice story of taking one style of model, which is a spine, which is just too specific for the data generating process, and maybe not a good fit. And in doing some exploratory data analysis where we see that averaging over seven days tells a really nice story and gives us the discreteness that we want. And then combining those two together into kind of an aggregate of the two types of models. Where We're able to better account for the fact that game days, are, are an important thing.

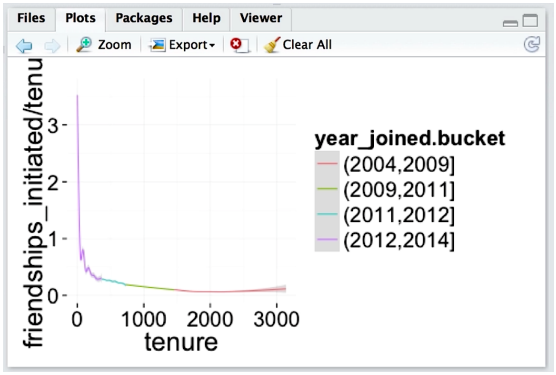
Introducing the Yogurt Dataset

Throughout all of our analyses of the pseudo-Facebook user data set, we've come to learn a lot about our users. From their birthdays, to their friend counts, to their friendships initiated, we've really come to understand their behaviors and how they use the Facebook platform. But, now I think it's time for something completely different. In the next couple of segments, we'll look at another data set, and then we'll return to this Facebook data set to draw some comparisons Here's Dean to introduce this new data set, and why we might look at it in the first place. >> Because of online purchases, credit cards, and loyalty cards, lots of retail purchase data is associated with individuals or households, such that there is a history of purchase data over time. Analysts in industry often mine this panel scanner data, and economists and other behavioral scientists use it to test and develop theories about consumer behavior. We are going to work with a data set describing household purchases, of five flavors of Dannon yogurt in the eight-ounce size. Their price is recorded with each purchase occasion. This yogurt data set has a quite different structure than our pseudo-Facebook data set. The synthetic Facebook data has one row per individual with that row giving their characteristics and counts of behaviors over a single period of time. On the other hand, the yogurt data has many rows per household, one for each purchase occasion. This kind of microdata is often useful for answering different types of questions than we've looked at so far.

Programming Quiz: Histograms Revisited



I'd like you to get started by looking at the structure and the summary of the Yerger data set on your own computer. You may want to pause the video now to give yourself some time to load in the data set and to download it first from the instructor notes. This Yerger data set contains over variables. The observations are for households that buy Dannon yogurt over time. Now one thing that you might notice is that most of the variables in here are integers. But I want to convert one of the variables to a factor, and that's the ID variable. We'll see how this comes in handy later in the course. But just make sure it says factor right here on your data set as well. If ID doesn't say factor for the type of variable, make sure you run this command in order to change ID to a factor variable. Once you've done all this, I want you to create a histogram of the Yogurt prices. Copy and paste your R code in to the R chunk on the quiz screen that will appear here. And then write a few sentences about what you notice. So that way we can be consistent as we work through the data set together, I want you to be sure to read in the data set into the variable called yo. You can, of course, use other variable names. But our auto grader will only accept answers for programming assignments that have this data frame labeled as yo.



Answer:

This code should be really familiar to you because it's what we saw in lesson three. One of the first plots you created was a histogram, so let's see what this one looks like. We can immediately notice some important discreetness to this distribution. There appear to be prices at which there are many observations, but then no observations in adjacent prices. This makes sense if prices are set in a way that applies to many of the consumers. There are some purchases that involve much lower prices, and if we are interested in price sensitivity, we definitely want to consider what sort of variations is in these prices. Now, I also want you to note that if we chose a different bin width we might obscure this discreetness. Say if I chose a bandwidth equal to ten. In this histogram, we would miss the observation for some of the empty spaces for the adjacent prices. So, it's no surprise that for this very discreet data this histogram is a very biased model.

Programming Quiz: Number of Purchases

There are other ways we could have noticed this important feature of the data set. But, there are also other ways, that by jumping into a different analysis, we might have missed this. For example, if we just look at a five number summary of the data, we might not notice this so easily. One clue to the discreteness is that the 75th percentile is the same as the maximum. We could also see this discreteness by looking at how many distinct prices there are in the data set. So here it looks like there's about 20 different prices. Tabling the variable we get an idea of the distribution like we saw in the histogram. So now that we know something about the price, let's figure out on a given purchase occasion how many eight ounce yogurts does a household purchase. To answer this we need to combine accounts of the different yogurt flavors into one variable. For example, for this particular household, on one purchase occasion, they bought three different types of yogurt. To figure this out for all the households, we need to make use of a new function. The function is called the transform function. Now I haven't introduced this function yet, but I want you to look up the function and run the examples of the code at the bottom of the documentation to figure out what it does. See if you can create a new variable called all.purchases, which gives the total counts of yogurt for each observation or household purchase. Keep in mind that you need to save this variable to the data frame, so this may or may not be in your syntax.

Answer:

For this programming assignment you needed to create the variable all purchases and add it to our Yo data frame. The transform function takes in a data frame and allows us to add different variables to it by recombining variables that are already within the data frame. So here I'll create the variable All Purchases and then I'll set it equal to the sum of all the yogurt flavors strawberry, blueberry, pina colada , plain, and mixed berry. When I run the code, I can see that my number of variables changed. And, just to be sure, I can print a summary of this variable as well. You might have also used this code. This code is fine, it's just a bit more verbose than we need.

Programming Quiz: Prices Over Time

Now that we have an extra column, our variable in our data frame, we can create a histogram of it. This histogram reveals that most households buy one or two yogurts at a time. To dive deeper into

our yogurt prices and household behavior. Let's investigate the price over time in more detail. This is where our Facebook data was deficient. In this data set, we can examine changes in prices because we have data on the same households over time. So, now I want you to use your knowledge of scatter plots and over-plotting to create an appropriate visualization. Your visualization should be a scatter plot of price versus time. Now, there's no right answer here, so just create a plot that's sensible and that allows you to see the data. You can assess your code and your plot with what we did in the solution video.

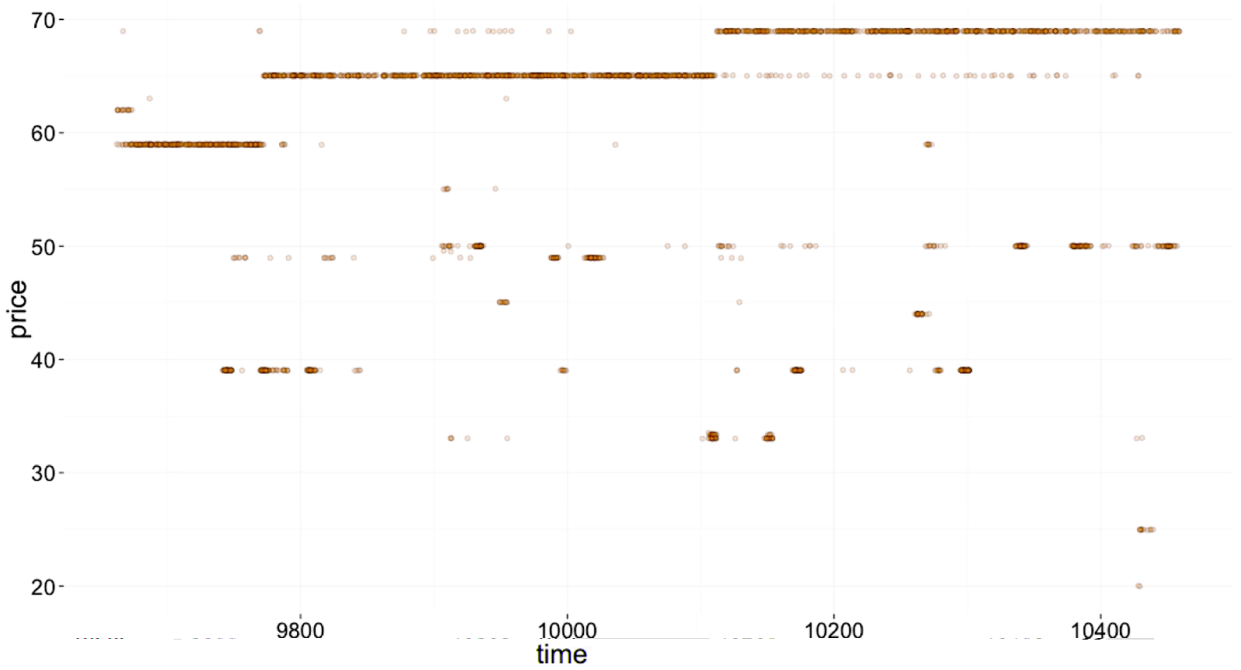
Answer:

For this solution, I'm showing you the ggplot code to create the scatter plot of price versus time. Now, I've added geom jitter here. So, that way, my points will show up a little bit transparent. And I'll color them with an orange hue. Looking at the plot, we can that the mode or the most common prices, seem to be increasing over time. We also see some lower price points scattered about the graph. These may be due to sales or, perhaps, buyers using coupons that bring down the price of yogurt.

Sampling Observations

Now I know our work up to this point with this yogurt data set has been review and it should be, but lets hear from Dean about how we might proceed differently with this type of a data set.

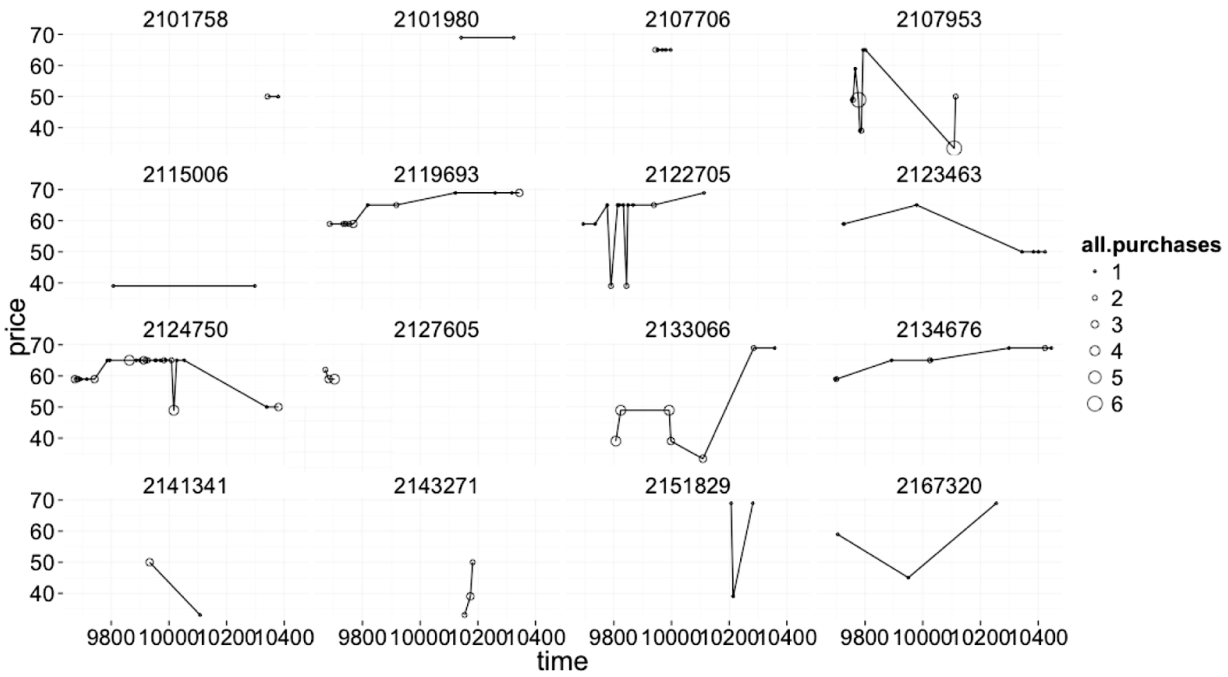
When familiarizing yourself with a new data set that contains multiple observations of the same units, it's often useful to work with a sample of those units so that it's easy to display the raw data for that sample. In the case of the yogurt data set, we might want to look at a small sample of households in more detail so that we know what kind of within and between household variation we are working with.



This analysis of a sub-sample might come before trying to use within household variation as part of a model. For example, this data set was originally used to model consumer preferences for variety. But, before doing that, we'd want to look at how often we observe households buying yogurt, how often they buy multiple items, and what prices they're buying yogurt at. One way to do this is to look at some sub-sample in more detail. Let's pick 16 households at random and take a closer look.

Programming Quiz: Looking at Samples of Households

Thanks again, Dean. Let's cut up the exploration Dean just described. First, we should use the set seed function to make this reproducible. Running this one line of code will also allow you to see the same households in my explorations, if you use the same seed number. Now, let's sample 16 of the households from our yogurt data set, from yogurt ID. Notice that I'm sampling from the levels because those are all of the different households that I have. I'll run the code for the sample ID's. And then, I'll print out the sample ID's. There's the 16 of them. Now we can plot each purchase occasion for each of the households that we sampled. We have the time of the purchase, the price per item of the yogurt, and the number of items. Here, I'm using the size parameter to add more detail to my plot. I'm passing at the all purchases variable, so that way I can consider the number of items in terms of size of the point on the plot. So, before I run this code, I want you to take a minute to think about what type of plot this code will produce. You maybe want to pause for a moment and give yourself some time to think. Now you might also notice this new syntax here. This percent in percent sign. This just tells us to loop over the IDs, so that when we go through and create a panel plot, for each ID of our households. And here's the plot. Notice how we get panels for each of the households that we had in our sample.



From these plots, we can see the variation and how often each household buys yogurt. Here, a lot and here, not to much. And it seems that some household purchases more quantities than others with these larger circles indicating not here. For most of the households, the price of yogurt holds steady, or tends to increase over time. Now, there are, of course, some exceptions, like in this household and in this household, and even here, we might think that the household is using coupons to drive the price down. Now, we don't have the coupon data to associate with this buying data, but you could see how that information could be paired to this data to better understand the consumer behavior. So now, let's have you do something similar. Your task is to create a similar plot and provide your own insights for another set of your seed number and the title and in the post, you can include an image of your plot and any observations that you make. You may want to try this for a few households to get a better feel for the data and the plot. There won't be any solution video here, so we'd love to hear from you all and see you all discuss what's going on in the discussions.

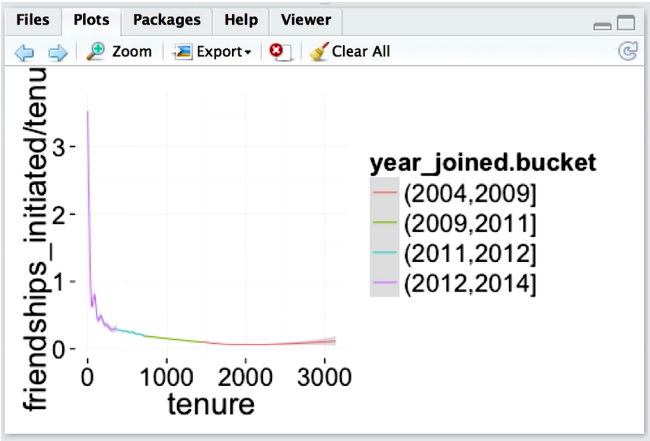
Answer:

The general idea is that if we have observations over time, we can facet by the primary unit, case, or individual in the data set. For our yogurt data it was the households we were faceting over. This faceted time series plot is something we can't generate with our pseudo Facebook data set. Since we don't have data on our sample of users over time. Let's get back to that plot to see that limitation. The Facebook data isn't great for examining the process of friending over time. The data set is just a cross section, it's just one snapshot at a fixed point that tells us the characteristics of individuals. Not the individuals over, say, a year. But if we had a dataset like the yogurt one, we would be able to track friendships initiated over time and compare that with tenure. This would give us better evidence to explain the difference or the drop in friendships initiated over time as tenure increases.

Many Variables

Before we wrap up our work with the pseudo-Facebook data, let's hear from Dean again about another technique that you can use in your work as a data analyst.

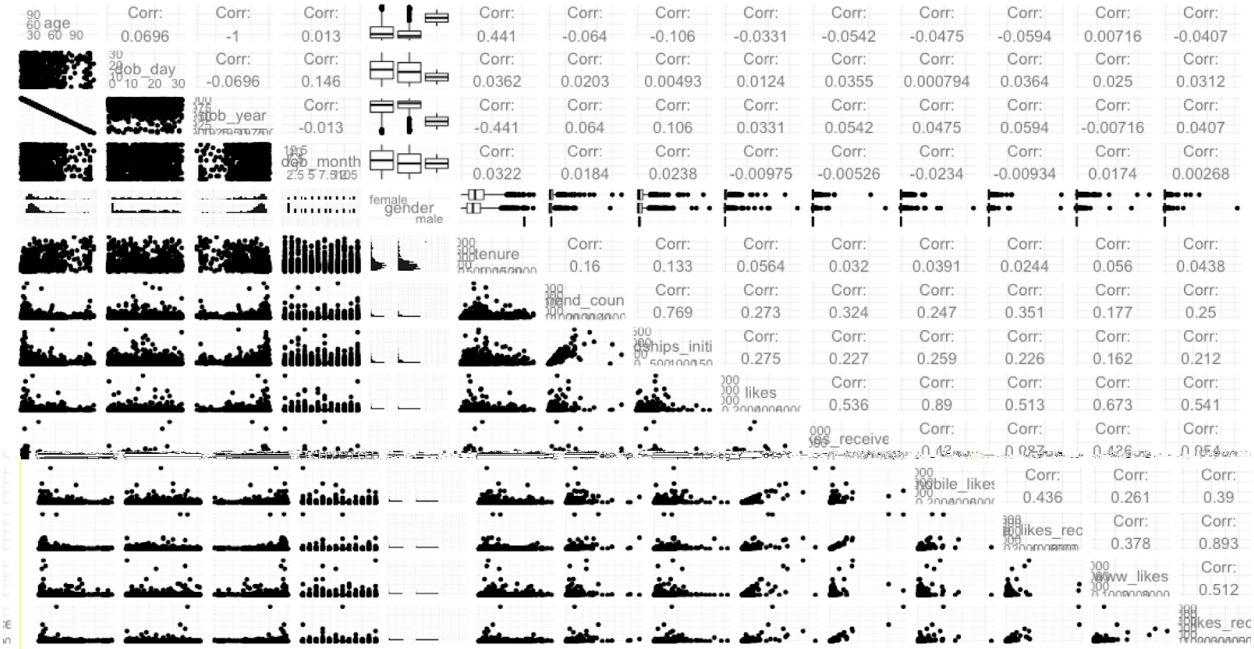
Much of the analysis we've done so far focused on some pre-chosen variable, relationship or question of interest. We then used EDA to let those chosen variables speak and surprise us. Most recently, when analyzing the relationship between two variables we look to incorporate more variables in the analysis to improve it. For example, by seeing whether a particular relationship is consistent across values of those other variables. In choosing a third or fourth variable to plot we relied on our domain knowledge. But often, we might want visualizations or summaries to help us identify such auxiliary variables. In some analyses, we may plan to make use of a large number of variables. Perhaps, we are planning on predicting one variable with ten, 20, or hundreds of others. Or maybe we want to summarize a large set of variables into a smaller set of dimensions. Or perhaps, we're looking for interesting relationships among a large set of variables. In such cases, we can help speed up our exploratory data analysis by producing many plots or comparisons at once. This could be one way to let the data set as a whole speak in part by drawing our attention to variables we didn't have a preexisting interest in.



Programming Quiz: Scatterplot Matrices

As Dean described, we should let the data speak to determine variables of interest. There's a tool that we can use to create a number of scatter plots automatically. It's called a scatter plot matrix. In a scatter plot matrix. There's a grid of scatter plots between every pair of variables. As we've seen, scatter plots are great, but not necessarily suited for all types of variables. For example, categorical ones. So there are other types of visualizations that can be created instead of scatter plots. Like box plots or histograms when the variables are categorical. Let's produce the scatter plot matrix for our pseudo Facebook data set. We're going to use the GGally package to do so. So make sure you've installed it and then go ahead and load it using the library command. Now, I'm also going to set the theme here too. Now, there's two other things that we want to do. First we want to set the seed so we get reproducible results. Now, you might be wondering why we set the seed in the first place. And it's because we're going to sample from our data set. Our data set contains all these variables and I

actually don't want all the variables. I don't want user ID, year joined, or year joined.bucket. So what I can do is subset my data frame and then sample from that subset. If I check out the variables in my subset data frame these are the ones of interest.



Now I didn't use year joined or year joined.bucket, because this one's a categorical variable and really these were derived from tenure. Now I'm ready to use the GG pairs function inside of GGally to create this scatter plot matrix. Now, I've already run this piece of code and I do want to warn you that it takes a long time for this to generate. It might even take over an hour. Feel free to run the command and if its taking a long time for your plot to generate just come back to your computer at another time. We've also included a pdf of the scatter plot in the instructor notes so you can check that out as well. Here's our scatter plot matrix, and notice in the upper part of the matrix we can see the correlation coefficients for the pairs of variables. For age and date of birth year, the correlation is actually negative one. And we can see that on the scatter plot. Sometimes we may want to produce these types of matrices so that way we can produce one number summaries of the different relationships of our variables. This is just like the correlation work that we did in lesson four. So, I've described the plots above the diagonal for the scatter plot matrix, but what do you notice in the lower left of the scatter plot matrix? Write a few sentences about what you see in this next quiz. And pay careful attention to the variable of gender. What types of plots do you think these are?

Answer:

The GG pairs function uses a different plot type for different types of combinations of variables. Hence, we have histograms here and we have scatter plots here. Many of these plots aren't quite as nice as they would be if we fine-tuned them for the particular variables. For example, for all the counts of likes, we might want to work on a logarithmic scale. But, ggpairs doesn't do this for us. At the very

least, a scatter plot matrix can be a useful starting point in many analyses.

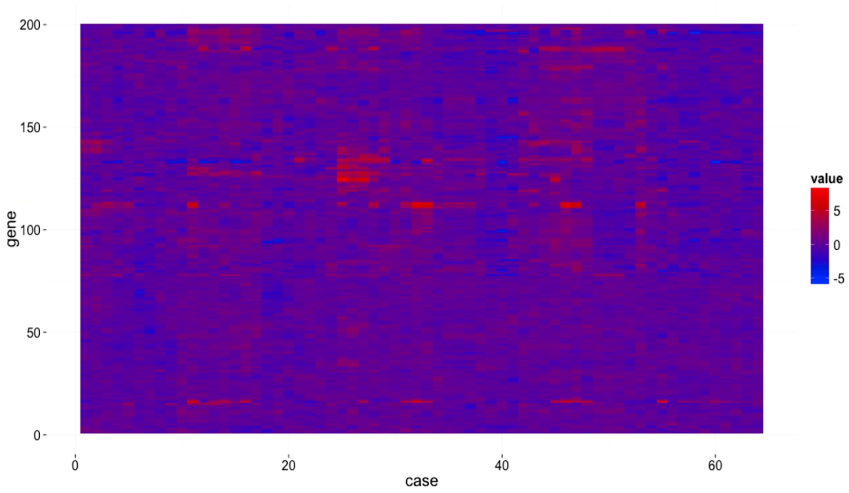
Even More Variables

A matrix such as this one will be extremely helpful when we have even more variables than those in the pseudo-Facebook data set. Examples arise in many areas, but one that has attracted the attention of statisticians is genomic data. In these data sets, they're often thousands of genetic measurements for each of a small number of samples. In some cases, some of these samples have a disease, and so we'd like to identify genes that are associated with the disease. In the instructor notes, you'll find a data set of gene expression in tumors. The data contains the expression of 6,830 genes, compared with a larger baseline reference sample. Now, this is a ton of data. So let's go ahead and read in the data set, and then I'll change the color names of the data set to be the numbers from one to 64. Now, I'm just doing this so that way the plot that I create is going to be a little bit nicer with the labeling on the x axis.

Heat Maps

The last plot that we'll make for this course is called a Heat Map. For our data set we want to display each combination of gene and sample case, the difference in gene expression and the sample from the base line. We want to display combinations where a gene is overexpressed in red. in combinations where it is under expressed in blue. Here's the code to make that Heat Map. First, we'll run all of this in order to melt our data to a long format. And then we just run our ggplot code using the geom, geom tile. Now, this last line is going to give us a scale gradient. And we're going to use the colors from blue to red. So, let's see what the output looks like. And, there's our Heat Map. Even with such a dense display, we aren't looking at all the data. In particular, we're just showing the first 200 genes. That's 200 genes of over Genomic data sets of these kind, sometimes called micro data are only getting larger, and more complex. What's most interesting, is that other data sets also look like this. For example, internet companies run lots of randomized experiments. Where in the simplest versions, users are randomly assigned to a treatment like a new version of a website or some sort of new feature or

product or a control condition. Then the difference in outcome between the treatment and control can be computed for a number of metrics of interest. In many situations, there might have been hundreds or thousands of experiments and hundreds of metrics. This data looks very similar to the genomic data in some ways. And this is why the useful maxim plot all the data might not always apply to a data set as it did to most of this course.



Reflection Quiz: Analyzing Three or More Variables

Congratulations. You've made it a long way and you've reached the end of lesson five. I want to end this lesson with another reflection. Take a few moments and write down some of the things you've learned in this lesson in the text box that will appear. Once you've done that, submit your answer and then hear from Tim and Moira about what they thought was important.

Analyzing Three or More Variables

Congratulations on finishing lesson five. Here, we took many of the basic techniques you learned in previous lessons and extended them to look at patterns across many variables at once. >> We started with simple extensions to the scatter plot, and plots of conditional summaries that you worked with in lesson four, such as adding summaries for multiple groups. Then, we tried some techniques for examining a large number of variables at once, such as scatter-plot matrices and heat maps. >> We also learned how to reshape data, moving from broad data with one row per case, to aggregate data with one row per combination of variables, and we moved back and forth between long and wide formats for our data. >> Next up in lesson six, we'll hear from our colleague Solomon. He's going to do an in-depth analysis of the diamond data set, so you can see how an expert does it. Lesson six also covers a larger part of the data analysis process, highlighting the role of EDA. Solomon did a lot of reading about the diamond industry. He wrote codes to scrape and format a new data set, and he used the results of his exploratory data analysis to guide, interpret, and evaluate a statistical model of diamond prices.

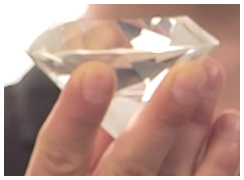
Exploratory Data Analysis

Lesson 6: Diamonds & Price Predictions

Quick links

Welcome to Lesson 6 of EDA!

Chris: Congratulations on making it to lesson six. This is the final lesson for the course, and we brought in Facebook data scientist Solomon Messing to share with you some of his own work in EDA.



Solomon: In this lesson, I'll walk you through an analysis of diamonds. You'll learn about the rich history behind the diamond market, and use the EDA techniques that you've learned to develop a quantitative understanding of it. The ultimate goal is to build a predictive model of diamonds that's going to help you figure out whether a given diamond is a good deal, or a rip-off.

Chris: If you're in the market for a diamond, exploring this data set can help you understand what goes into the price of a diamond, and even if you're not looking to buy, the socioeconomic and political history of the diamond industry is fascinating.

Solomon: Diamonds gave rise to the mining industry in South Africa, which is now the most advanced economy in the region. Diamonds also drove the British and the Dutch to colonize southern Africa in the first place, and have driven conflicts ranging from the Boer wars to modern day civil strife across the region.

Chris: Now, you should have some experience working with the diamonds data based on the problem sets in this course. I'll let Solomon guide you through his exploration.

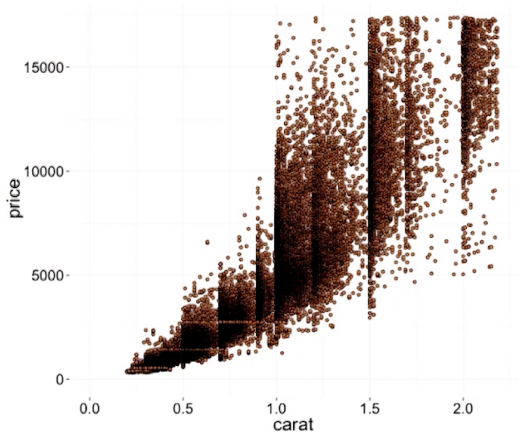
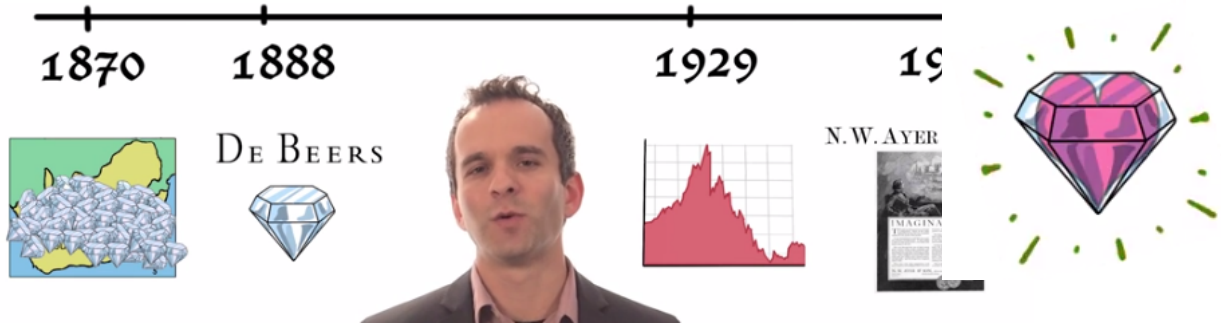
Scatterplot Review

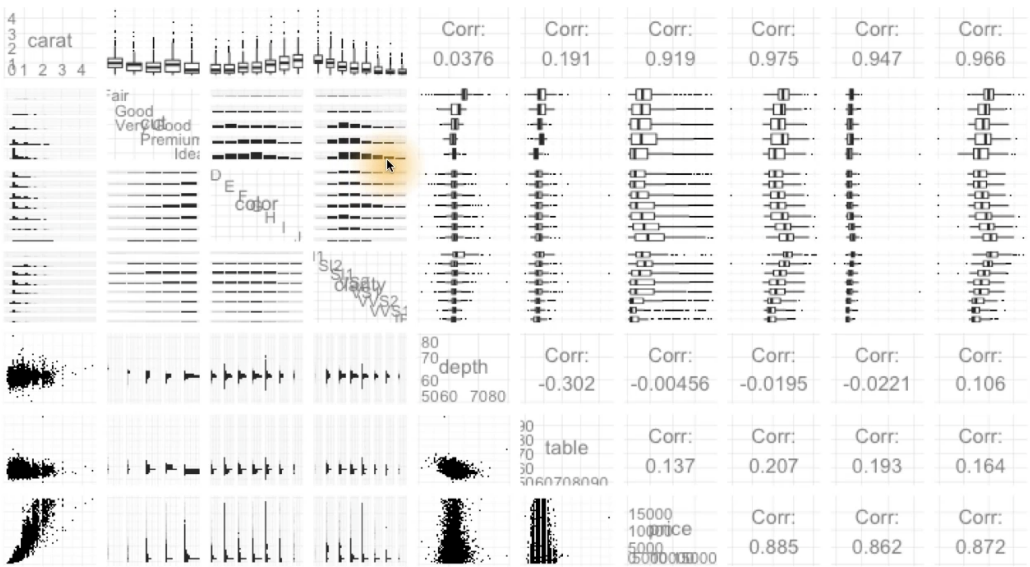
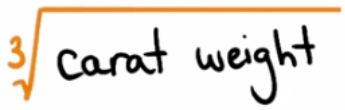
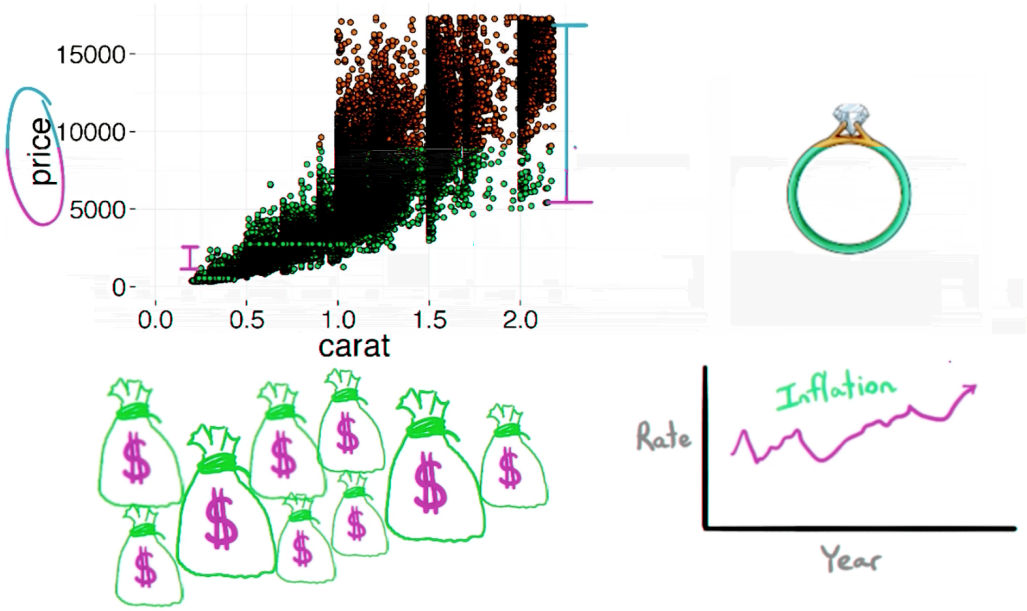
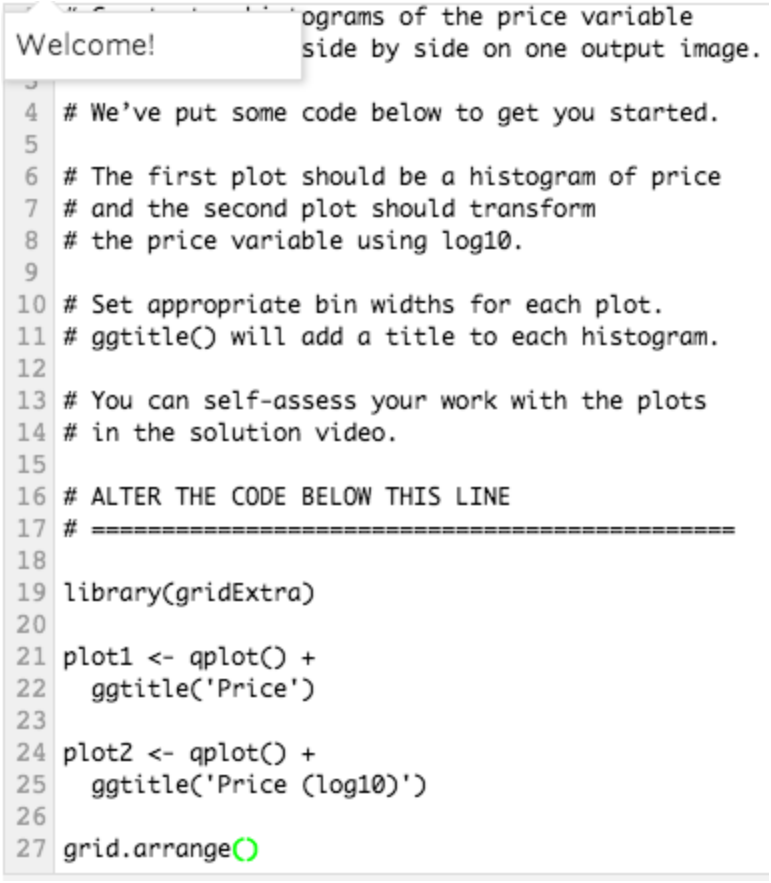
Solomon: Let's run our usual code to load in the data side. For review, let's have you create a scatter plot in the next exercise. (Lesson 6 RMD File)

```r
# Let's start by examining two variables in the data set.
# The scatterplot is a powerful tool to help you understand
# the relationship between two continuous variables.

# We can quickly see if the relationship is linear or not.
# In this case, we can use a variety of diamond
# characteristics to help us figure out whether
# the price advertised for any given diamond is
# reasonable or a rip-off.

# Let's consider the price of a diamond and it's carat weight.
# Create a scatterplot of price (y) vs carat weight (x).

# Limit the x-axis and y-axis to omit the top 1% of values.

# ENTER YOUR CODE BELOW THIS LINE
# ==========================================================

```

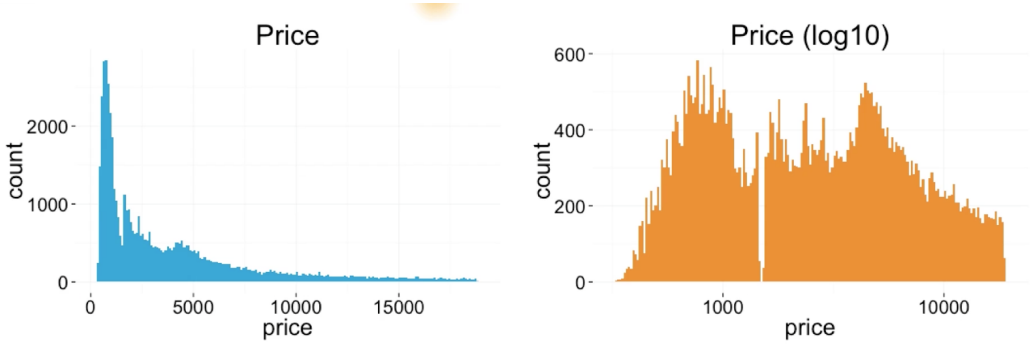
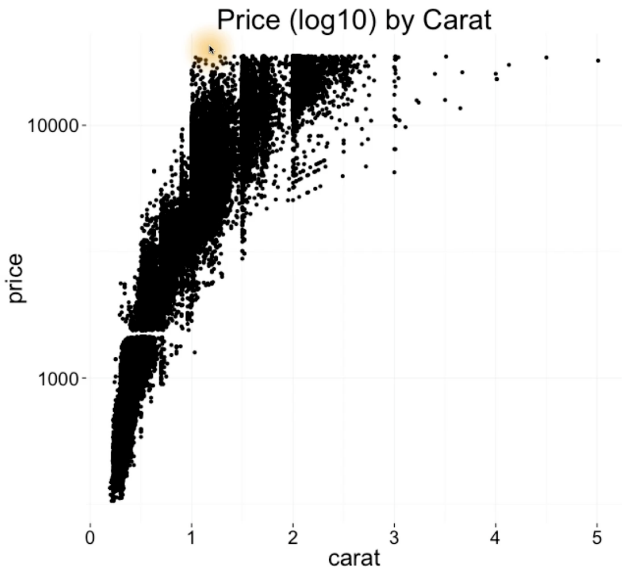
Programming Quiz

```r
## Scatter Plot Review
```{r Scatter Plot Review}
qplot(data = diamonds, x = carat, y = price,
      xlim = c(0, quantile(diamonds$carat, 0.99)),
      ylim = c(0, quantile(diamonds$price, 0.99))) +
  geom_point(fill = I('#F79420'), color = I('black'), shape = 21)
```

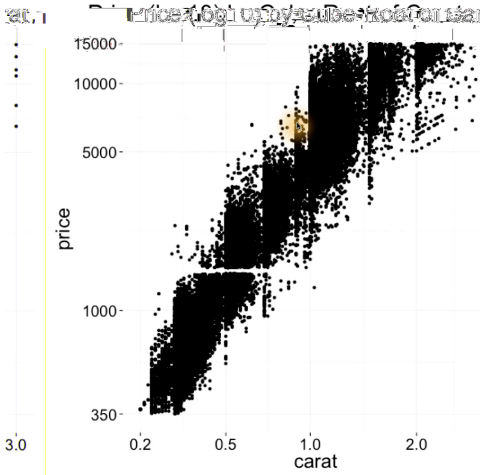
ggplot equivalent
ggplot(diamonds, aes(x = carat, y = price)) +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99))) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99))) +
  geom_point(fill = I('#F79420'), color = I('black'), shape = 21)
```

Answer

We've seen this kind of code before. All we're doing is we're using **qplot** to plot the price of diamond against its carat weight. And we're going to trim the top percentile off of both carat and price. If you use the full ggplot syntax, your code will look something like this.

Price and Carat Relationship

Alright, so let me ask you this question. What do you notice about the relationship here between price and carat? Go ahead and enter your response.

Answer



When we look at the plot, a few things pop out right away. We can see a nonlinear relationship. Maybe it's exponential or maybe it's something else. We can see that the dispersion or variance of the relationship also increases as carat size increases. With just a quick look at the data we've learned two important things about the functional relationship between price and carat size. We can add a linear trim line to the plot by using the stat smooth function with method equals lm. We can see that the linear trend line doesn't go through the center of the data at some key places. It misses it here. It should curve a little bit in the center of the relationship, and it should slope up more toward the end. And if we tried to use this to make predictions, we might be off for some key places inside and outside of the existing data that we have displayed.

Frances Gerety

So far, we've only considered the bivariate relationship between price and carat weight. But there's much more to this data set and I'd like to give it a proper introduction. The diamonds data set ships with **ggplot2** and contains the prices and the specs for more than 50,000 diamonds collected in 2008 from diamondse.info. Now, analyzing this data is particularly useful. Because diamonds are unique in a way that just isn't true of most manufacture products that we're used to buying. You can't just plug in the model number and just look up the price. Though the diamonds data set is full of prices and fairly esoteric certification ratings. Hidden in the data are reflections of how a legendary marketing campaign permeated and was subsumed by our culture. Hints about how different social strata responded and how the diamond market functions today as a result. The story starts in 1870. When many tons of diamonds were discovered in South Africa near the Orange River. Until then the diamond market had been small, only a few pounds of diamonds were mined each year from India and Brazil. At the time, there was no use for diamonds outside of jewelry, so price depended only on scarce supply. Hence, the project's investors formed the De Beers Cartel in By most accounts, this has been the most successful cartel in history. But World War I and the Great Depression saw diamond sales

Quiz

Finish the famous DeBeers slogan, a diamond is: _____.

Answer

A diamond is forever. This tag along has appeared in every De Beers ad since 1948. In 1999, two weeks before Mrs. Geary died, Ayer signed Age named it the slogan of the century.

plummet. In 1938, the De Beers Cartel contacted Philadelphia ad agency N.W. Ayer & Son to inquire whether, “The use of propaganda in various forms might help jump start diamond sales in the U.S.” Whi for diamonds at the time. Surveys showed, however, that among couples contemplating marriage, diamonds were low on the list of priorities. A luxury for the rich mostly down the chain. Frances Gerety took on the DeBeers account at N.W. Ayer & Son. And worked towards the company's goal to “create a situation in which every couple contemplating marriage feels the need to acquire a diamond engagement rings.” A few years later, she would coin a famous slogan.



The Rise of Diamonds

Many argue that this campaign gave birth to modern demand advertising. The objective here was not demand generation nor branch strengthening. But simply to impress the glamor, the sentiment and the emotional charge contained in the product, itself. The company gave diamonds to movie stars. They sent out press packets emphasizing the size of diamonds that celebrities gave each other. They loaned diamonds to prominent socialites attending events like the Kentucky Derby or the Academy Awards. And even persuaded the British royal family to wear diamonds over other gems. Later, De Beers sought to market diamond rings to couples as a status symbol. To reflect quote, a man's success in life. A 1980's add introduced the famous two month bench mark. Isn't two months salary a small price to pay for something that lasts forever? By any reasonable measure, Francis Geary succeeded. Getting engaged in America means getting a diamond ring. Can you think of a movie where two people get engaged with out a diamond ?

When you get engaged on Facebook, what icon does the site display? Still think this might not be the most successful mass persuasion effort in history? I present to you a James Bond film whose title bears the diamond cartel's trademark. Awe-inspiring and a little terrifying. Let's open the data set.

ggpairs Function

Keep this history in mind as we work through our exploratory analysis. The first thing you might consider doing is plotting key variables against each other using the **ggpairs** function. This function plots each variable against each other variable, pairwise. You may want to sample first, otherwise the function will take a long time to render the plots. Also, if your data set has more than about 10 columns, there will be too many plotting windows. So, subset on your columns first if that's the case. The first thing we want to do is make sure you have these packages installed. We use **ggally** for this particular plot. We use **scales** for a variety of things. **memisc** to summarize the regression. **lattice** for few other things. **mass** for various functions. **car** to recode variables. **reshape** to reshape and wrangle your data. **plyr** to create interesting summaries and transmissions that you've done. We'll go ahead and load these packages, and then set the seed for randomization purposes and sample happening is that **ggpairs** is plotting each variable against the other in a pretty smart way. In the lower triangle of the plot matrix, it uses grouped histograms for qualitative, qualitative pairs and scatter plots for quantitative, quantitative pairs. In

the upper triangle, it plots grouped histograms for qualitative, qualitative pairs, this time using the x instead of the y variable as the grouping factor. Box plots for qualitative, quantitative pairs, and it provides the correlation for quantitative quantitative pairs. Remember that our goal is to understand the price of diamonds.

```
# install these if necessary
install.packages('GGally')
install.packages('scales')
install.packages('memisc')
install.packages('lattice')
install.packages('MASS')
install.packages('car')
install.packages('reshape')
install.packages('plyr')

# load the ggplot graphics package and the others
library(ggplot2)
library(GGally)
library(scales)
library(memisc)
```

```
# sample 10,000 diamonds from the data set
set.seed(20022012)
diamond_samp <- diamonds[sample(1:length(diamonds$price), 10000), ]
ggpairs(diamond_samp, params = c(shape = I('.'), outlier.shape = I('.'
)))
```

So, let's focus on that.



Quiz

The price variable is here. So, let's look at the relationships that correspond to price. What are some things you notice? There's no right or wrong answer here, but think deeply about the plots and the associations you see. Write a few sentences about what stands out to you.

Answer

We can see what might be relationships between price and clarity and price and color, which we'll keep in mind for later when start modelling our data. You might remember this when you create the box plots in problem set three. Be that as it may, the critical factor driving price is the size, or the carat weight of the diamond. As we saw at the start of the lesson, the relationship between price and diamond size is nonlinear. What might explain this pattern? On the supply side, larger continuous chunks of diamonds without significant flaws are probably harder to find than smaller ones. This might help explain the sort of exponential looking curve, and I thought I noticed this when I was shopping for a diamond for my soon to be wife. Of course, this is related to the fact that the weight of a diamond is a function of volume, and volume is a function of the



length times thewidth times the height of a diamond, and this suggests that we might be especially interested in the cube root of carat weight. It's often the case, that leveraging substantive knowledge about your data like this can lead to especially fruitful transformations, and we'll see that in a second.

The Demand of Diamonds

On the demand side, customers in the market for a less expensive, smaller diamond are probably more sensitive to price than more well-to-do buyers. Many less than one carat customers would surely never buy a diamond. Diamond were not for the social norm of presenting one when proposing. And there are fewer customers who can afford a bigger diamond that is one that is larger than than one carat, hence we shouldn't expect the market for bigger diamonds to be as competitive as the one for smaller diamonds. So it makes sense that the variants As well as the price would increase with carat size. Now often the distribution of any monetary variable like dollars will be highly skewed and vary over orders of magnitude. Now this can result from path dependence for example the rich getting richer, or multiplicative processes



like year on year inflation, or some combination of both. Hence it's a good idea to look into compressing any such variable by putting it on a log scale. For more tips on when to use the log scale and how to transform your variables check out the link. Let's examine the distribution of price again. In this next programming exercise you complete the code to generate two price histograms.

Programming Quiz



Answer

```
plot1 <- qplot(data = diamonds, x = price, binwidth = 100, fill = I
('#099DD9')) +
  ggtitle('Price')


plot2 <- qplot(data = diamonds, x = price, binwidth = 0.01, fill = I
('#F79420')) +
  ggtitle('Price (log10)') +
  scale_x_log10()


library(gridExtra)
library(grid)
grid.arrange(plot1, plot2, ncol = 2)
```

Here we're just using **qplot** to produce two histograms of price. One on just a regular scale and one on the **log10** scale. We're to use this **gridExtra** library to plot both



plots in the same window and here's what the result looks like.

Connecting Demand and Price Distribution

When looking at these plots, what do you notice? Think specifically about the two peaks in the transformed plot and how they relate to the demand for diamonds. Alright let me know what you think here.

Answer

Indeed we can see that the prices for diamonds are pretty heavily skewed, but when you put those prices on a log ten scale, they seem much better behaved. They're much closer to the bell curve of a normal distribution. We can even see a little bit of evidence of bimodality on this $\log_{10}$ scale. Which is consistent with our two class rich buyer poor buyer speculation about the nature of customers for diamonds. You actually saw a sneak peek of this in problem set set three when you looked at the $\log_{10}$ of price by cut.

Scatterplot Transformation



Now that we have a better understanding of our variables, and the overall demand for diamonds, let's replot the data. This time we'll put price on a $\log_{10}$ scale, and here's what it looks like. This plot looks better than before. On the log scale, the prices look less dispersed at the high end of Carat size and price, but actually we can do better. Let's try using the cube root of Carat in light of our speculation about flaws being exponentially more likely in diamonds with more volume. Remember, volume is on a cubic scale. First, we need a function to transform the Carat variable. If you'd like to learn more about writing your own functions in R, check out [this link about the basic structure of functions] and [this blog post about using defining new transformations in ggplot2 and the scales package]

This may seem like a lot of code, but really, there's only one new piece here. It's this **cuberoot_trans** function. It's a function that takes the cube root of any input variable, and it also has an inverse function to undo that operation, which we need to display the plot correctly. Then when we get to our actual **ggplot** command. What

```
## Use the cuberoot_trans function
```{r}
ggplot(aes(carat, price), data = diamonds) +
  geom_point() +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat')
```
```

we'll do is we'll use the **scale_x_continuous** argument to transform the x-axis with this cube root transformation function. Keep in mind we're also transforming the y axis with this $\log_{10}$ transformation that we discussed previously. And, let's see what this plot looks like. Taking a look at the plot, we can actually see that with these transformations that we used to get our data on this nice scale. Things look almost linear. We can now move forward and see about modelling our data using just a linear model.



Overplotting Revisited

Until now we haven't really done anything about over plotting. That's when multiple points take on the same value. This is often due to rounding. So let's look at this by

running some code. We're going to run the **table** command on both carat, and price. Then we're going to sort that so that the highest values appear first. We're just going to look at the top six which is the default for the head function. Of course, what's happening here is that the first line is the price or the carat, and the second line is the count for each one of these values. We can see that these are really high numbers which is going to result in a substantial amount of overplotting. When you have this much data, you're going to have serious overplotting, even when you're plotting the variables against each other, and this can really obscure some of the density and the sparsity of our data at really key points. Okay, so as we discussed in lesson four, you can deal with this by making your points smaller by jittering your points and by adding transparency. In **ggplot**, this is done with the **alpha** parameter. In the next coding exercise, you'll do just that.

Programming Quiz

```
# Add a layer to adjust the features of the
# scatterplot. Set the transparency to one half,
# the size to three-fourths, and jitter the points.

# If you need hints, see the Instructor Notes.
# There are three hints so scroll down slowly if
# you don't want all the hints at once.

# ALTER THE CODE BELOW THIS LINE
# ==========================================================================

ggplot(aes(carat, price), data = diamonds) +
  geom_point() +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat')
```

Answer

```
# Add a layer to adjust the features of the
# scatterplot. Set the transparency to one half,
# the size to three-fourths, and jitter the points.

# If you need hints, see the Instructor Notes.
# There are three hints so scroll down slowly if
# you don't want all the hints at once.

# ALTER THE CODE BELOW THIS LINE
# ==========================================================================

ggplot(aes(carat, price), data = diamonds) +
  geom_point() +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat')
```

Your code should look something like this. After making these 3 adjustments, you can see that the plot gives us a better sense of how dense and how sparse our data is at key places. See?

Plot Colors for Qualitative Factors

We can see what looks like a almost linear relationship between carat weight and price after doing some transformations, but surely there are other factors that influence the price of a diamond. When I was looking around at diamonds I noticed that clarity seemed to factor into price. Of course, many consumers are looking for a diamond of a certain minimum size. So we shouldn't expect clarity to be as strong a factor as carat weight. And I must admit that even though my grandparents were jewelers. I initially had a hard time discerning a diamond rated VVS1 from one rated SI2. Surely, most people need a jeweler's loupe to tell the difference. What makes a diamond sparkle anyway? According to blue nile, the cut of a diamond has a much more consequential impact on that fiery quality that jewelers describe when they talk about diamonds. On clarity, the website states many of these imperfections are microscopic and do not affect the diamonds beauty in any discernible way.

Price vs. Carat and Clarity

Let's see if clarity, cut, or color can explain some of the variants in price when we visualize it on our plot using color. We'll start by examining clarity. This code visualizes price by carat. Adjust it to color the points by clarity.

```
# Adjust the code below to color the points by clarity.

# A layer called scale_color_brewer() has
# been added to adjust the legend and
# provide custom colors.

# See if you can figure out what it does.
# Links to resources are in the Instructor Notes.

# You will need to install the package RColorBrewer
# in R to get the same colors and color palettes.

# ========================================
library(RColorBrewer)

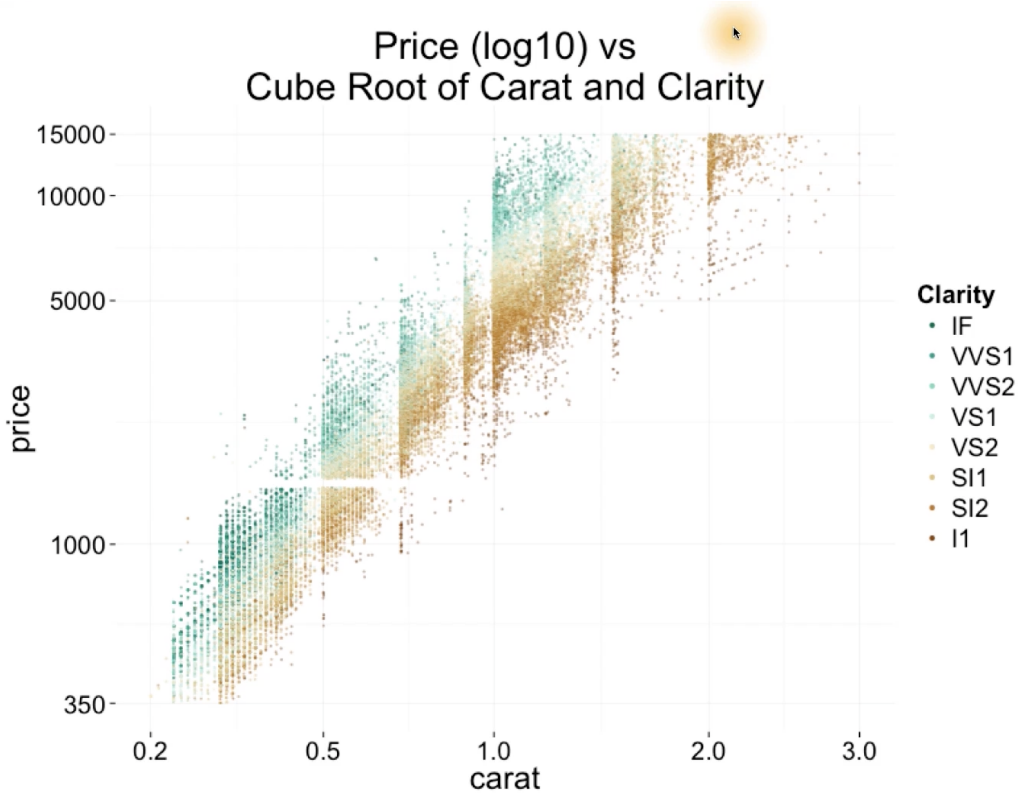
ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
    guide = guide_legend(title = 'Clarity', reverse = T,
    override.aes = list(alpha = 1, size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
    breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
    breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Clarity')
```

Programming Quiz

Answer

```{r}
# install.packages(RColorBrewer)
library(RColorBrewer)

ggplot(aes(x = carat, y = price, colour = clarity), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
      guide = guide_legend(title = 'Clarity', reverse = TRUE,
                           override.aes = list(alpha = 1, size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) vs\nCube Root of Carat and Clarity')
```

We just need to add one parameter to our aesthetic wrapper in ggplot like so. Set the color parameter for equal to clarity. And when we run it, here's what it looks like.



Clarity and Price

Based on the plot, do you think clarity explains some of the change in price? Why? Enter your response here.

Answer

Clarity does seem to explain an awful lot of the remaining variance in price, after adding color to our plot. Holding carat weight constant, we're looking at one part of the plot. We see the diamonds with lower clarity are almost always cheaper than diamonds with better clarity.

Price vs. Carat and Cut

Now let's look at cut. Change the code such that you're coloring the points by cut, and don't forget to change the titles.

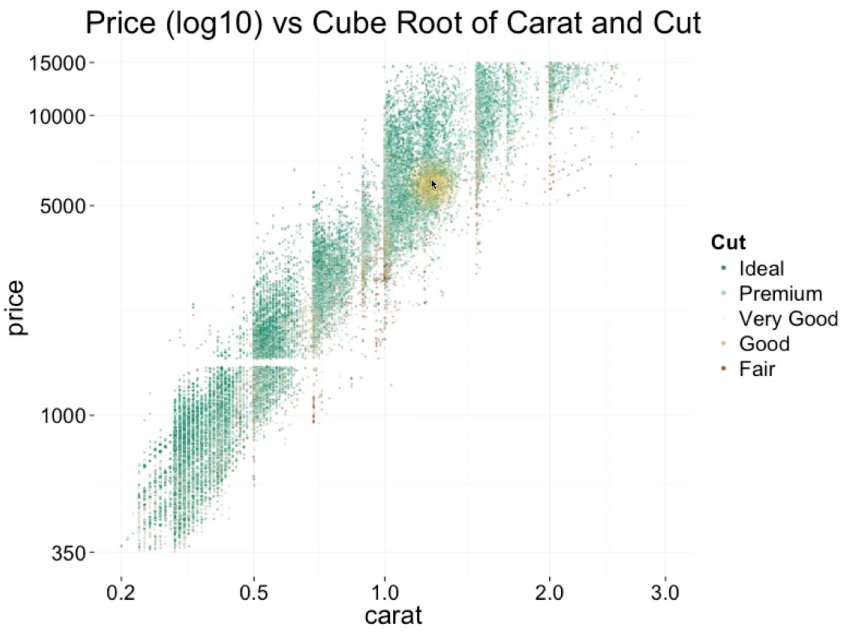
Programming Quiz

```r
# Let's look at cut and see if we find a similar result.

# Adjust the code below to color the points by cut.
# Change any other parts of the code as needed.

# ALTER THE CODE BELOW THIS LINE
# ==========================================================================================

ggplot(aes(x = carat, y = price, color = clarity), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Clarity', reverse = T,
                     override.aes = list(alpha = 1, size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-root of Carat and Clarity')
```

Answer

```r
```{r}
ggplot(aes(x = carat, y = price, color = cut), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
    guide = guide_legend(title = 'Cut',
                reverse = TRUE,
                override.aes = list(alpha = 1, size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) vs Cube Root of Carat and Cut')
```
```

You might have made just one change to the code by swapping out the cut variable with the Clarity variable, but I hope you changed your titles as well. Running the code, here is the plot that we get.



Cut and Price

Based on the plot, do you think that cut accounts for some of the variance in price, why? Enter your response here.

Answer

Despite what Blue Nile says, we don't see much variation on cut. Most of the diamonds in the data are ideal cut anyway, so we've lost the color pattern that we saw before.

Price vs. Carat and Color

Finally, let's use diamond color as a color in our plot. Adjust the code below, to color the points by diamond color.

Programming Quiz

```
# Finally, let's use diamond color to color our plot.

# Adjust the code below to color the points by diamond colors
# and change the titles.

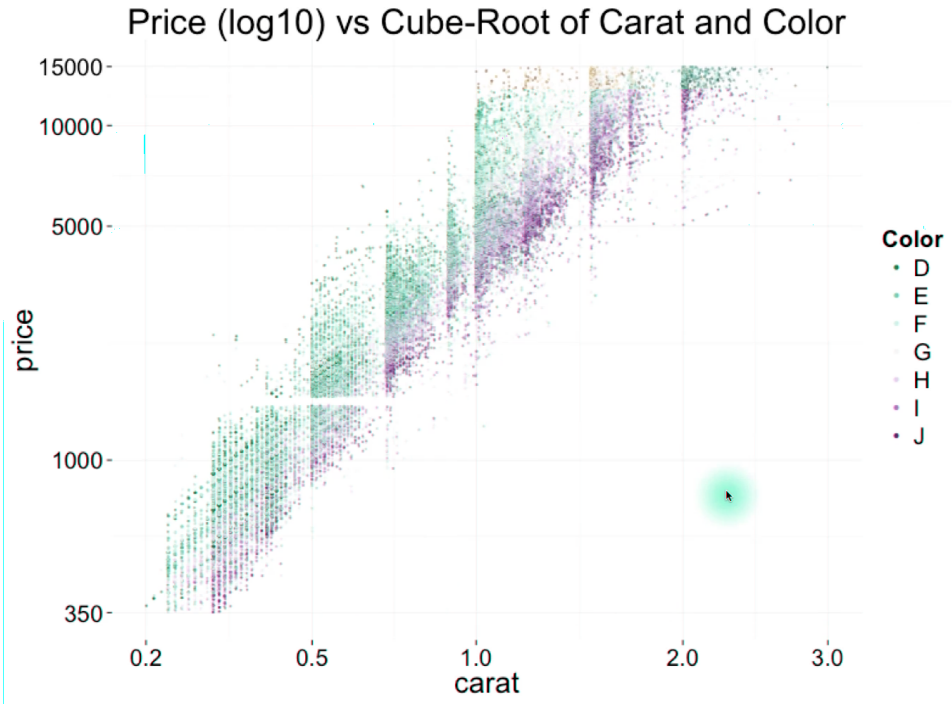
# ALTER THE CODE BELOW THIS LINE
# ==========================================================================================

ggplot(aes(x = carat, y = price, color = cut), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Cut', reverse = T,
                                          override.aes = list(alpha = 1, size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Cut')
```

Answer

```r
ggplot(aes(x = carat, y = price, color = color), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
      guide = guide_legend(title = 'Color', reverse = FALSE,
                  override.aes = list(alpha = 1, size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) vs Cube-Root of Carat and Color')
```

Here, I made four changes to the code. I replaced cut with color in the aesthetic wrapper. I also changed the legend title and the plot title, to use the word color. And finally, I removed the reverse parameter in the legend, so that the best color would be at the top of the list in the legend. Now that we've run our code, this is the output.



Color and Price

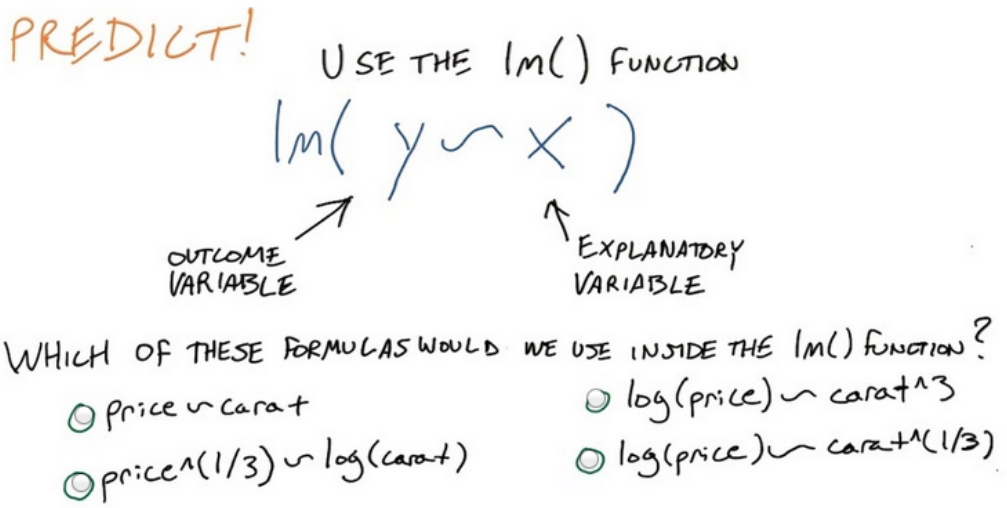
Based on the plot, do you think that the color of the diamond influences price? Why? Enter your answer in the box.

Answer

Color does seem to explain some of the variance in price. Just like we saw with the clarity variable. Blue now however, states that the difference between all color grades from D to J are basically not noticeable to the naked eye. Yet, we do see the color difference in the price tag.

Linear Models in R

In R we can create models using the lm function. We need to supply a formula in the form of y~x. Here, y is the outcome variable and x is the explanatory variable. Which of the following formulas would we use inside the lm function? Price~^. Price to the cube root ~log^ log price ~^ carat to the third, log price ~^ to the 1 3rd, select one.



Answer

Remember we applied the log transformation to our long tailed dollar variable, and we speculated that the flawless diamond should become exponentially rarer as diamond volume increases. So we should be interested in the cube root of carat weight.

Building the Linear Model

Let's build up our linear model for price. I'm going to store the first model in a variable called **m1**. You probably also noticed how I used the I wrapper around each of the variables. The "I" stands for as is. In this case, it tells R to use the expression inside the "**I**" function to transform a variable before using it in the regression. This is instead of instructing R to interpret these symbols as part of the formula to construct the design matrix for the regression. You can read more about the syntax for linear models online. I can also update the previous model to add the carat variable in the regression, using the syntax. The real functional relationship is surely not as simple as the cubed root of carat, so we add a simple linear function of carat in our model predicting price. And we can continue to make more complex models by adding more variables. We add cut even though we don't expect it to have much influence

on price. Next, we add color to a fourth model and clarity to a fifth. When we run the code, we can see that we're getting some very nice R square values. We're accounting

```r
m1 <- lm(I(log(price)) ~ I(carat^(1/3)), data = diamonds)
m2 <- update(m1, ~ . + carat)
m3 <- update(m2, ~ . + cut)
m4 <- update(m3, ~ . + color)
m5 <- update(m4, ~ . + clarity)
mtable(m1, m2, m3, m4, m5)
```

for almost all of the variance in price using the four “C”s. If we want to know whether the price for, a diamond is reasonable, we might now use this model.

Console ~/Public/EDA/data_sets/

```
                                                                  (0.002)
clarity: ^6                                                       -0.002
                                                                  (0.002)
clarity: ^7                                                        0.032***
                                                                  (0.001)
-------------------------------------------------------------------------
R-squared            0.924      0.935      0.939      0.951      0.984
adj. R-squared       0.924      0.935      0.939      0.951      0.984
sigma                0.280      0.259      0.250      0.224      0.129
F               652012.063 387489.366 138654.523 87959.467 173791.084
p                    0.000      0.000      0.000      0.000      0.000
Log-likelihood   -7962.499  -3631.319  -1837.416  4235.240  34091.272
Deviance          4242.831   3613.360   3380.837  2699.212    892.214
```

$$\ln(\text{price}) = 0.415 + 9.144 * \sqrt[3]{\text{carat}} - 1.093 * \text{carat} + (\ldots * \text{cut} + \ldots * \text{color} + \ldots * \text{clarity}) + \varepsilon$$

Model Problems

Our model is the log of price equals 0.415 plus 9.144 times the cube root of carat, minus 1.093 times carat plus a series of coefficient times each factor in cut another series of coefficient times each factor in color. And another series of coefficients times each factor in clarity plus an error term.

Quiz

What could be some problems when using this model. What else should we think about when we're using it. Go ahead and enter your answer here.

Answer

There was so much you might have said here. And, if you have any ideas that I don't mention, please share them in the forum. To start, our data's from 2008. So, not only do we need to account for inflation, but the diamond market is quite different now than it was. In fact, when I fit models to this data and predicted the price of the diamonds that I found off a market, I kept getting predictions that were way too low. After some additional digging, I found that global diamonds were poor. It turns out that prices plummeted in 2008 due to the global financial crisis. And since then prices, at least for wholesale polished diamonds, have grown at about of couples in China buying diamond engagement rings might also explain this increase. You can click on the links in the instructor notes to learn more about that. And finally, after looking at the data on price scope, I realize that diamond prices grew unevenly across different carat sizes since 2008, meaning that the model I initially estimated couldn't simply be adjusted by inflation.

A Bigger, Better Data Set

Thankfully, I was able to put together a Python script to get the current diamond price data, similar to the original diamonds data set, from diamondse.info without too much trouble. This data set is about ten times the size of the 2008 diamonds data set, and features diamonds from all over the world certified by an array of authorities, besides just the Gemological Institute Of America, or the GIA. You can read in this data set as follows, but make sure you've installed the RCurl and bitops libraries before you do. This might take a while as a data set contains over about

500,000 cases. The code used to obtain the data is available at this link. Let's fit the model again to the big data set. We'll only use GIA certified diamonds in this log. I look only at diamonds under $10,000 because these are the type of diamonds sold at most retailers, and hence, the kind we care most about. By trimming the most expensive diamonds from the data set, our model will also be less likely to be thrown

off by outliers, and the higher variants at the high-end of price and carat. You can learn how to scrape data from the web by taking Data Wrangling with MongoDB: Data Manipulation and Retrieval

```
# Your task is to build five linear models like Solomon
# did for the diamonds data set only this
# time you'll use a sample of diamonds from the
# diamondsbig data set.

# Be sure to make use of the same variables
# (logprice, carat, etc.) and model
# names (m1, m2, m3, m4, m5).

# To get the diamondsbig data into RStudio
# on your machine, copy, paste, and run the
# code in the Instructor Notes. There's
# 598,024 diamonds in this data set!

# Since the data set is so large,
# you are going to use a sample of the
# data set to compute the models. You can use
# the entire data set on your machine which
# will produce slightly different coefficients
# and statistics for the models.

# This exercise WILL BE automatically graded.

# You can leave off the code to load in the data.
# We've sampled the data for you.
# You also don't need code to create the table output of the models.
# We'll do that for you and check your model summaries (R^2 values, AIC, etc.)
```

Programming Quiz

```
28
29 # Your task is to write the code to create the models.
30
31 # DO NOT ALTER THE CODE BELOW THIS LINE (Reads in a sample of the diamondsbig data set)
32 #=============================================================================================
33 diamondsBigSample <- read.csv('diamondsBigSample.csv')
34
35
36 # ENTER YOUR CODE BELOW THIS LINE. (Create the five models)
37 #=============================================================================================
38
39
40
41
42 # DO NOT ALTER THE CODE BELOW THIS LINE (Tables your models and pulls out the statistics)
43 #=============================================================================================
44 suppressMessages(library(lattice))
45 suppressMessages(library(MASS))
46 suppressMessages(library(memisc))
47 models <- mtable(m1, m2, m3, m4, m5)
48
```

Answer

```
## Building a Model Using the Better Data Set
```{r}
diamondsbig$logprice = log(diamondsbig$price)
m1 <- lm(logprice ~  I(carat^(1/3)),
    data=diamondsbig[diamondsbig$price < 10000 &
                     diamondsbig$cert == "GIA",])
m2 <- update(m1, ~ . + carat)
m3 <- update(m2, ~ . + cut )
m4 <- update(m3, ~ . + color)
m5 <- update(m4, ~ . + clarity)
mtable(m1, m2, m3, m4, m5)
```
```

What we will do first is make a variable called logged price, the log of diamond price. And then we will build our model similarly to the way that we did for the small diamond status set. Notice that we're setting price by diamonds whose price is less than $10,000 and whose certificate is G.I.A.. Thankfully our models look quite a bit like they did for the smaller diamond status set. Although our R squared values are just a touch weaker.

```
---------------------------------------------------------------------------
R-squared             0.888       0.892      0.899       0.937       0.969
adj. R-squared        0.888       0.892      0.899       0.937       0.969
sigma                 0.289       0.284      0.275       0.216       0.154
F               2700903.714 1406538.330 754405.425  423311.488  521161.443
p                     0.000       0.000      0.000       0.000       0.000
Log-likelihood   -60137.791  -53996.269 -43339.818   37830.414  154124.270
Deviance          28298.689   27291.534  25628.285   15874.910    7992.720
AIC              120281.582  108000.539  86691.636  -75632.827 -308204.540
BIC              120313.783  108043.473  86756.037  -75482.557 -307968.400
N                338946      338946     338946      338946      338946
===========================================================================
```

Predictions

After all this work, let's use our model to make a prediction. We need to exponentiate the result of our model. Since we took the log of price. Let's take a look at an example from Blue Nile. We'll use the full model m5, to predict the value of the diamond. Try to understand what this code does. If you see something you don't understand, look it up in the documentation, then answer the question. Evaluate how well the model predicts the Blue Nile diamond price. Think about the fitted point estimate. As well as, the lower and upper bound of the 95% confidence interval.

Quiz

```
## Predictions
Example Diamond from BlueNile:
Round 1.00 Very Good I VS1 $5,601
```{r}
#Be sure you've loaded the library memisc and have m5 saved as an
object in your workspace.
thisDiamond = data.frame(carat = 1.00, cut = "V.Good",
                         color = "I", clarity="VS1")
modelEstimate = predict(m5, newdata = thisDiamond,
                        interval="prediction", level = .95)
```


Evaluate how well the model predicts the BlueNile diamond's price.
Think about the fitted point estimate as well as the upper and lower
bound of the 95% CI.
```

Answer

The results yield an expected value for price given the characteristics of our diamond, and the upper and lower bounds of a 95% confidence level. Note, because this is a linear model, predict is just multiplying each model coefficient by each value in our data. It turns out that this diamond is a touch pricier than the expected value under the full model, though it is by no means outside of the 95% confidence interval. Blue now has by most accounts a better reputation than diamond SE.info however. And reputation is worth a lot in a business that relies on easy to forge certificates in which the non-expert can be easily fooled. So, while this model might give you a sense of whether your diamond is a ripoff against diamondSE.info diamonds, it's not clear that diamondSE.info should be regarded as the universal source of truth over whether the price of a diamond is reasonable. Nonetheless, to have the expected price and diamondassay.info with a 95% interval, is a lot more information than we had about the price we should be willing to pay for a diamond before we started this exercise.

Final Thoughts

An important point. Even though we can predict the price of a diamond based on a function for c's. One thing you should not conclude with this exercise is that where you buy your diamond is irrelevant. You almost surely pay more for the same diamond at Tiffany's compared to Costco. Regardless you can use a model like this to get a sense of whether you were overpaying. One last thing, data and models are never infallible and you can still get taken even equipped with this kind of analysis. There's no substitute for establishing a personal connection and lasting business relationship with a jeweler you can trust.

Congratulations and Next Steps

Solomon: Congratulations on completing lesson six.

Moira: You've also reached the end of the course. We hope you feel prepared to explore data on your own, and answer questions that are interesting to you.

Dean: One next step is for you to find or collect some data, and start exploring.

Chris: Good luck in your future explorations, and thanks for joining us.