

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.

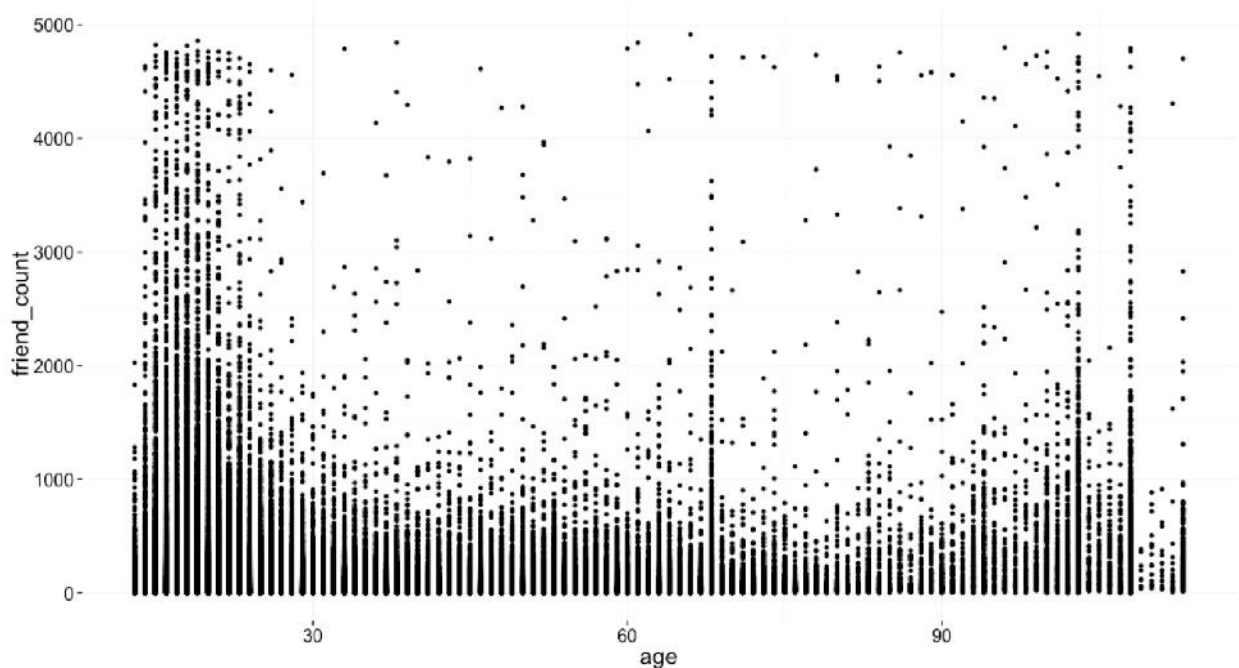
Scatterplots and Perceived Audience Size

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?

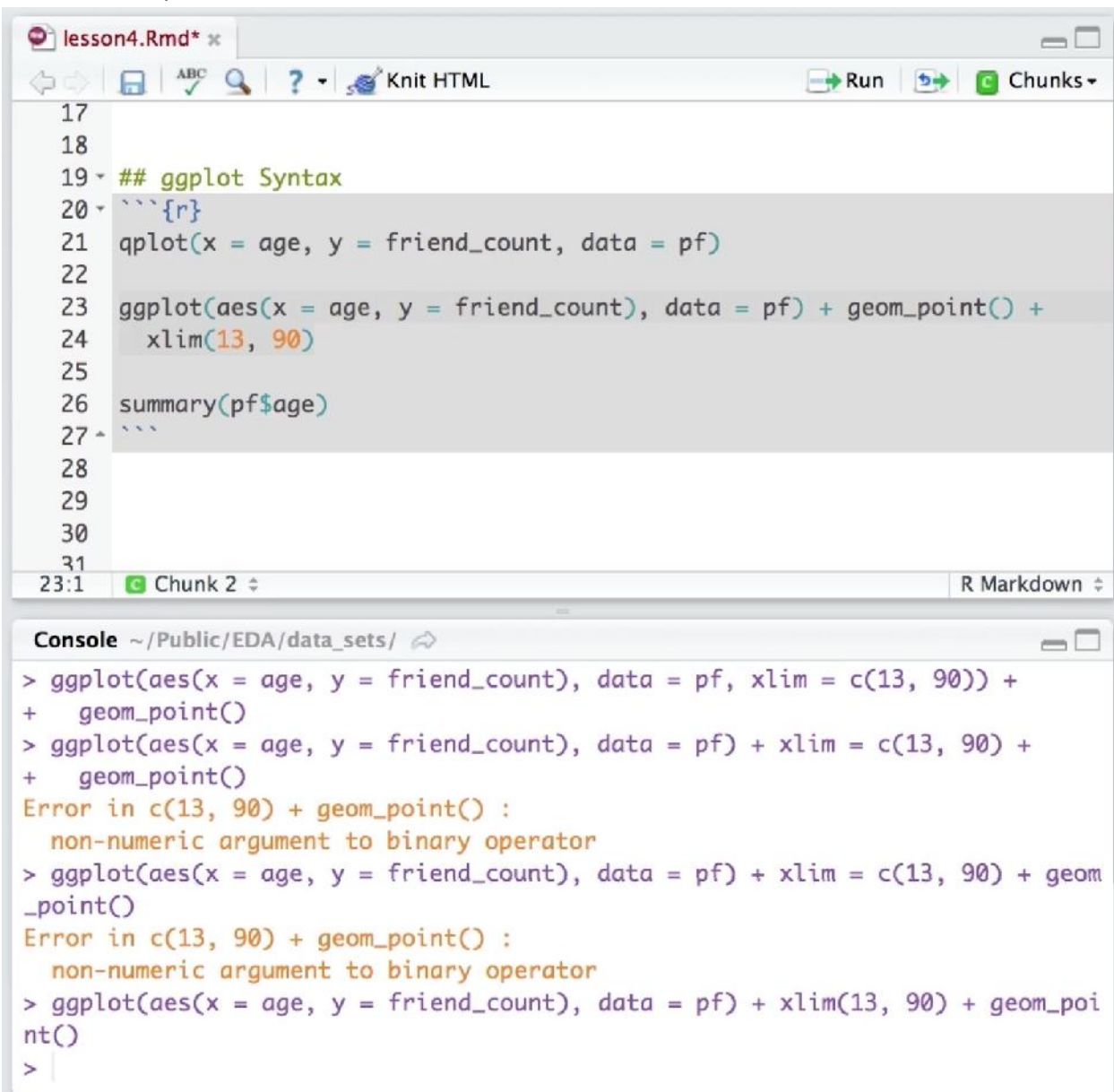


Answer:

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.



```
lesson4.Rmd* x
[Navigation icons] Knit HTML [Run] [Chunks]
17
18
19 ## ggplot Syntax
20 ```{r}
21 qplot(x = age, y = friend_count, data = pf)
22
23 ggplot(aes(x = age, y = friend_count), data = pf) + geom_point() +
24   xlim(13, 90)
25
26 summary(pf$age)
27 ```
28
29
30
31
23:1 [Chunk 2] R Markdown
```

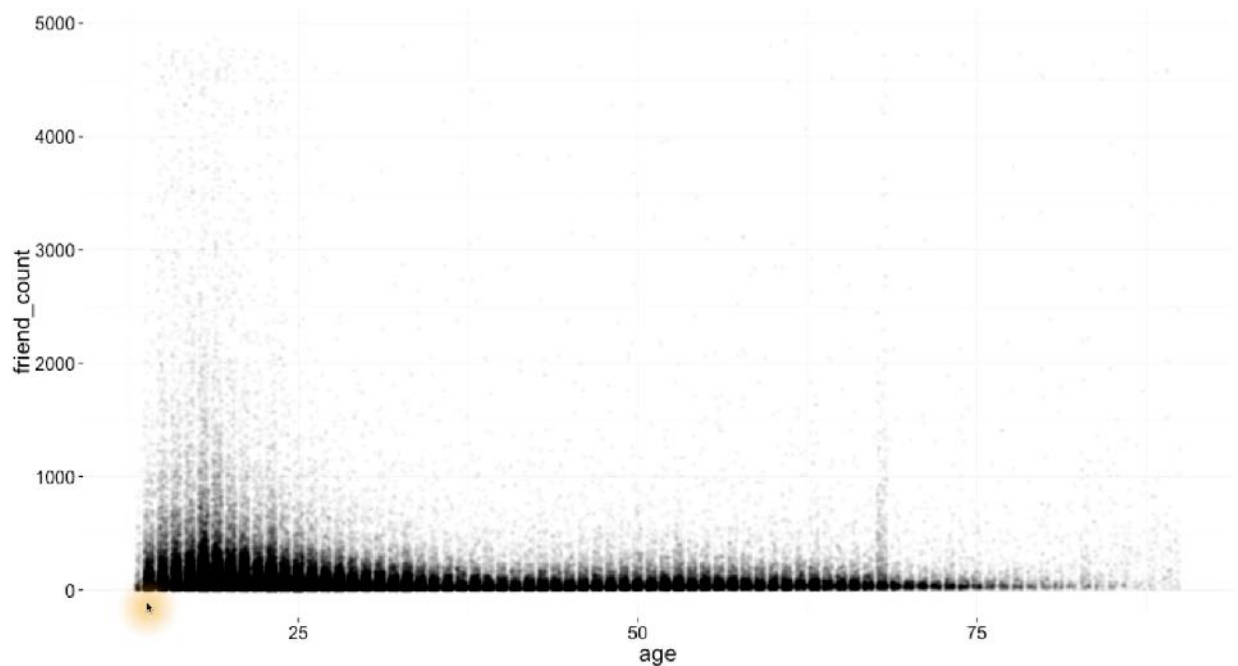
```
Console ~/Public/EDA/data_sets/
> ggplot(aes(x = age, y = friend_count), data = pf, xlim = c(13, 90)) +
+   geom_point()
> ggplot(aes(x = age, y = friend_count), data = pf) + xlim = c(13, 90) +
+   geom_point()
Error in c(13, 90) + geom_point() :
  non-numeric argument to binary operator
> ggplot(aes(x = age, y = friend_count), data = pf) + xlim = c(13, 90) + geom
_point()
Error in c(13, 90) + geom_point() :
  non-numeric argument to binary operator
> ggplot(aes(x = age, y = friend_count), data = pf) + xlim(13, 90) + geom_poi
nt()
>
```

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 `+` 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 $1/20$. 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.

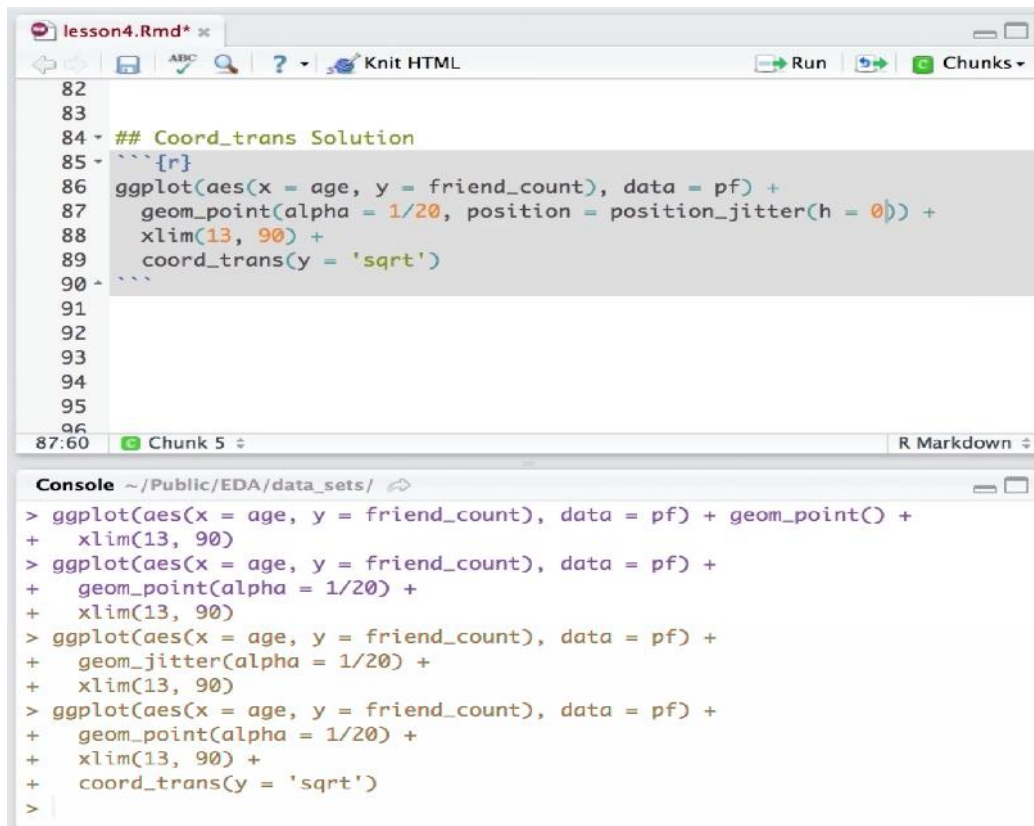


Answer:

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.



```
lesson4.Rmd* x
82
83
84 ## Coord_trans Solution
85 ```{r}
86 ggplot(aes(x = age, y = friend_count), data = pf) +
87   geom_point(alpha = 1/20, position = position_jitter(h = 0)) +
88   xlim(13, 90) +
89   coord_trans(y = 'sqrt')
90 ```
91
92
93
94
95
96
87:60 Chunk 5 R Markdown
```

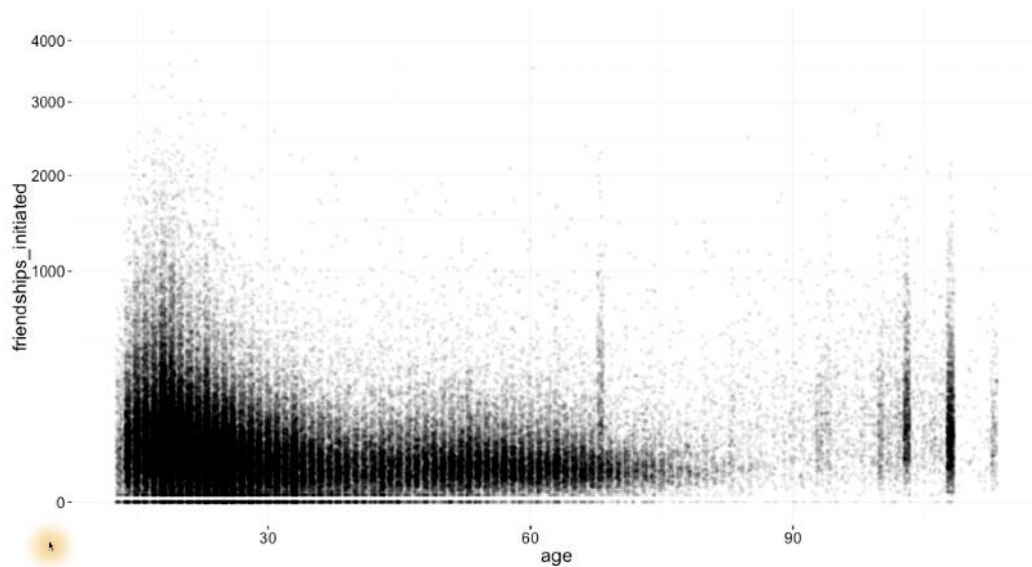
```
Console ~/Public/EDA/data_sets/
> ggplot(aes(x = age, y = friend_count), data = pf) + geom_point() +
+   xlim(13, 90)
> ggplot(aes(x = age, y = friend_count), data = pf) +
+   geom_point(alpha = 1/20) +
+   xlim(13, 90)
> ggplot(aes(x = age, y = friend_count), data = pf) +
+   geom_jitter(alpha = 1/20) +
+   xlim(13, 90)
> ggplot(aes(x = age, y = friend_count), data = pf) +
+   geom_point(alpha = 1/20) +
+   xlim(13, 90) +
+   coord_trans(y = 'sqrt')
>
```

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

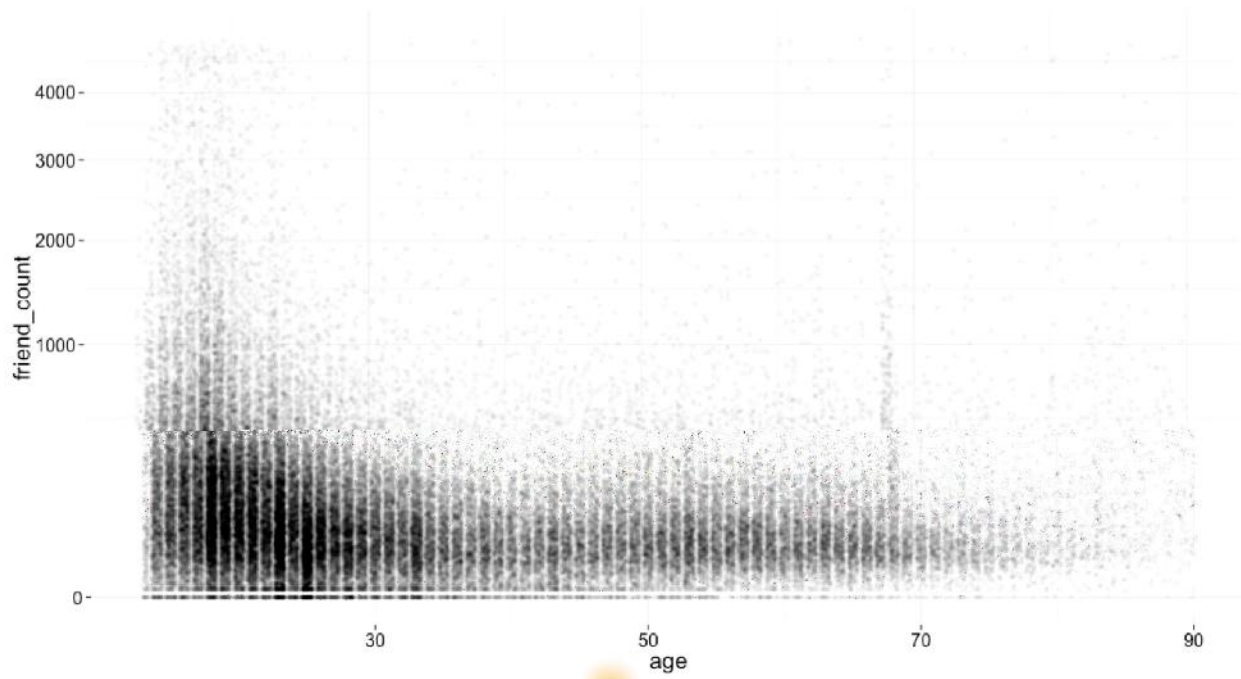
In the last exercise, we used alpha and jitter to reduce over plotting, but it turns out that there's more that we can do. Let's hear from Moira about how she used her domain knowledge and a transformation to make an adjustment to her scatter plot.

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

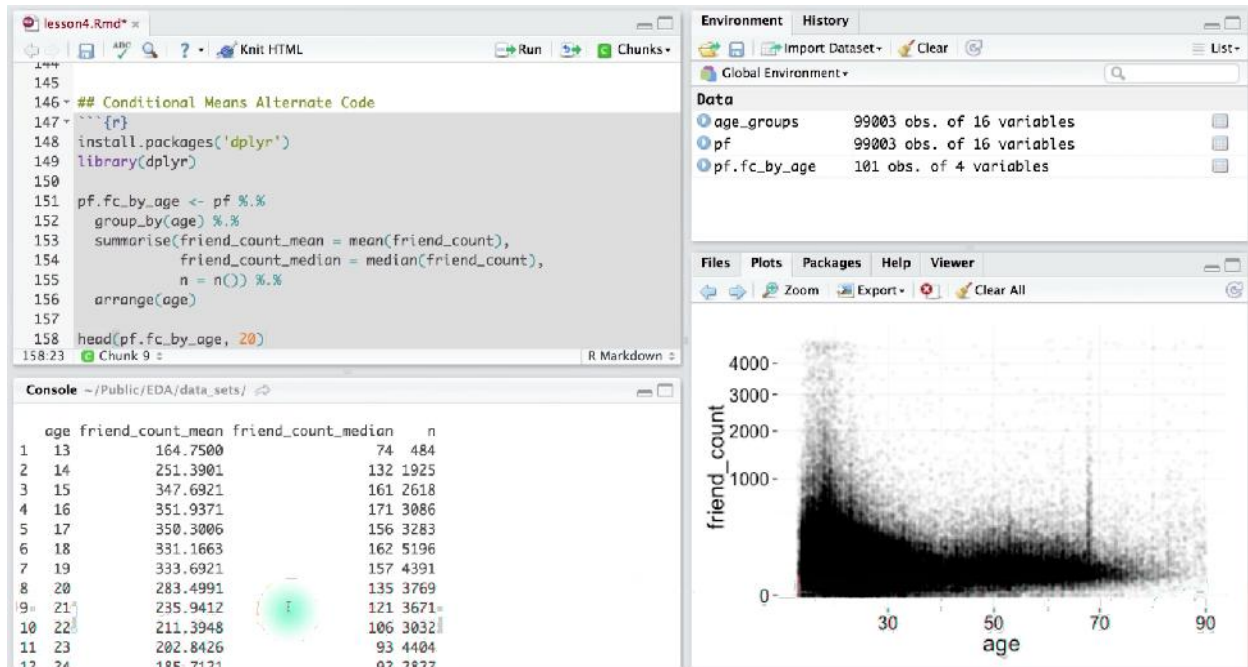
Let's take a step back and look at our scatter plot between friend count and age. This scatter plot keeps us really close to the data because we are visualizing every point in the data set. In general, it's not possible to judge important quantities from such a display. Sometimes you want to understand 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 fcbbyage, 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.

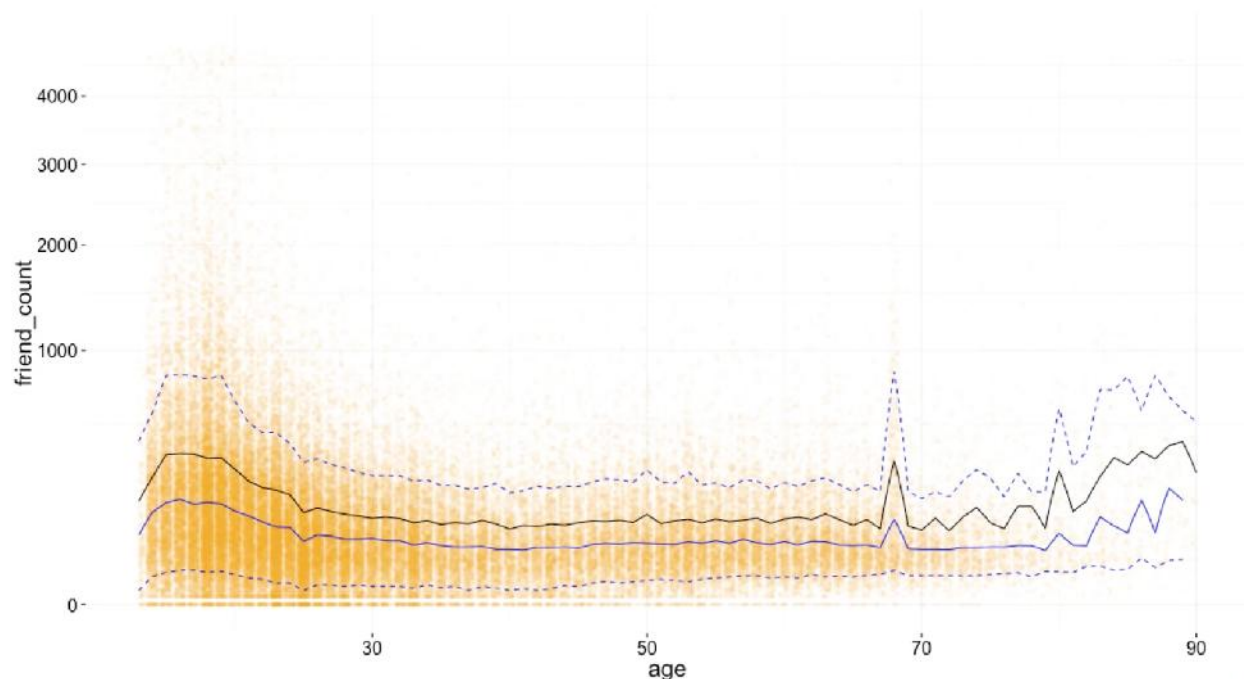
Answer:

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

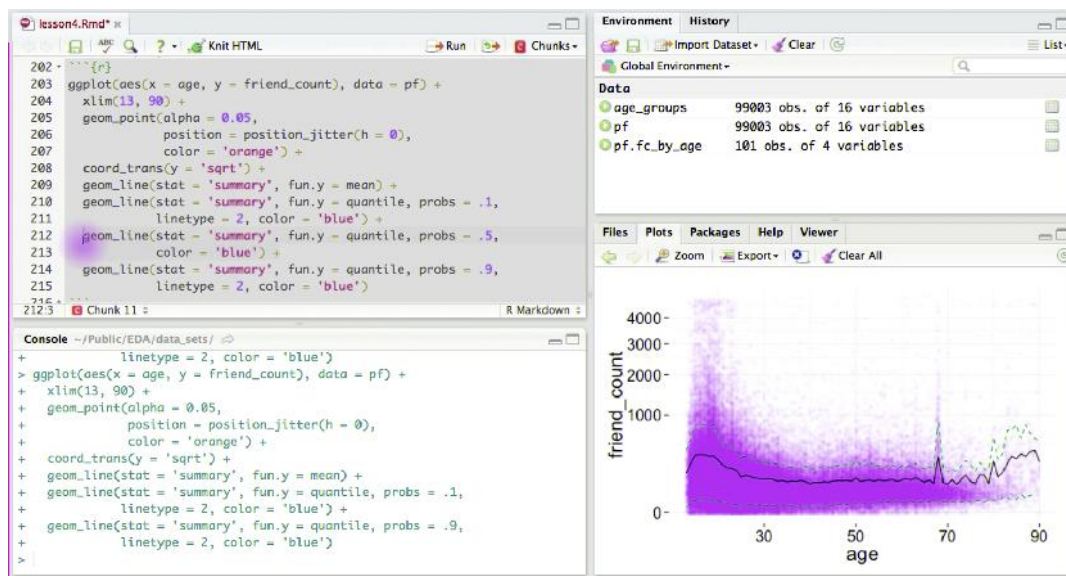
mean. And for the data variable, I want to pass it `pf.fc_by_age`, instead of `pf`, since this was the new data frame we're working with. Writing this code I see that I get an error, and that's because I forgot my `geom`. I needed to add `geom_point`. So adding that as another layer, I can run my code 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

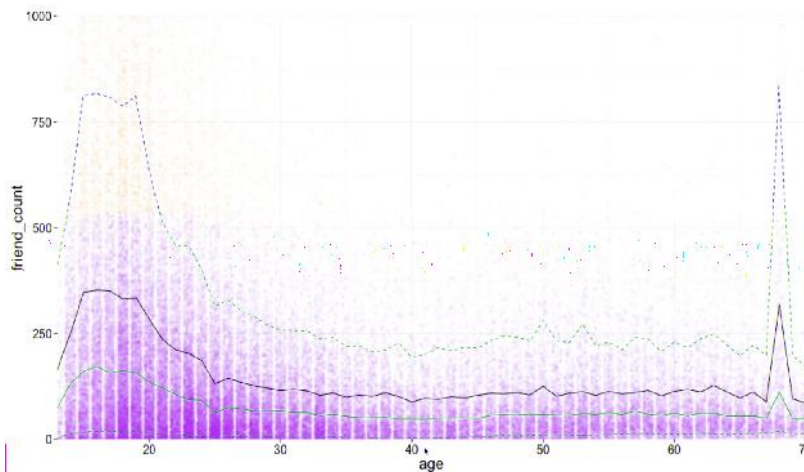
older crowd. And that the zoomed in on this plot using the `coord_cartesian` layer. To use it, we need to get rid of the `coord_trans` and the `xlim` layers first. So I'll delete those lines of code. Adding in the `coord_cartesian` layer, we can just set the `x` and `y` limits to whatever we think is 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

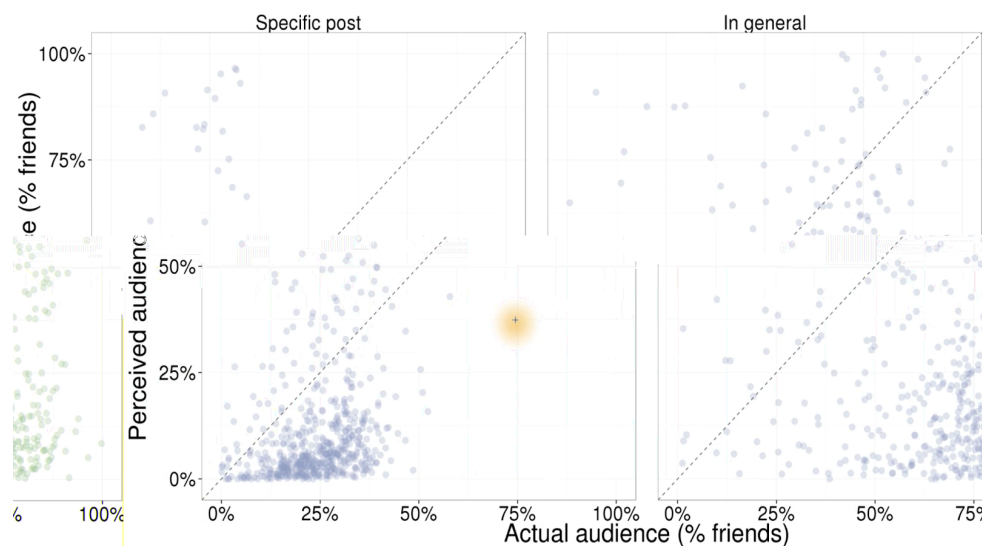
So, we've learned a little bit more about the relationship between age and friend count. And we did that using conditional means. We'll return to these two variables later and we'll see the relationship to a third variable in lesson five. But for now, let's see how more used a slightly different approach 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

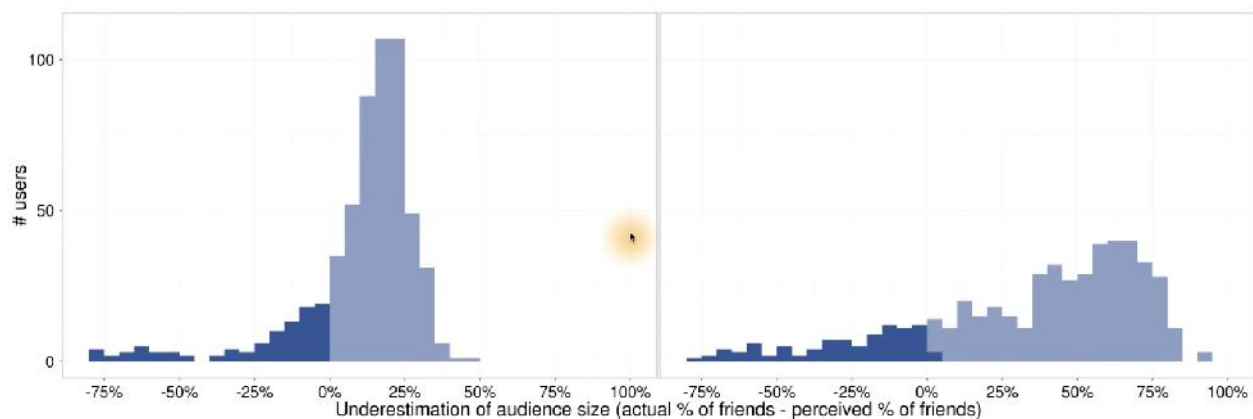
their actual audience size. And you can see they all fall below the perfect accuracy line. So people underestimate. But it's still kind of hard to get the big picture from these scatter plots and so, we have some histograms to capture more summary data. In this case, these two histograms show how much 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

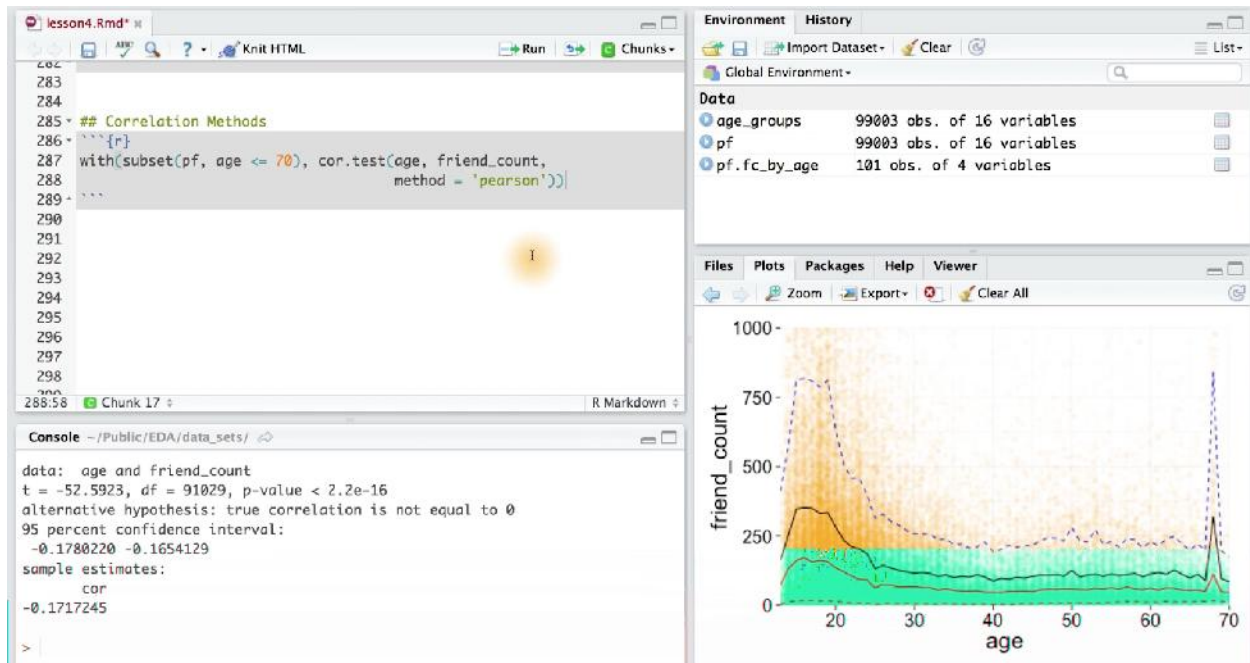
Based on the correlation coefficient in this plot, we just observed that the relationship between age and friend count is not linear. It isn't monotonic, either increasing or decreasing. Furthermore, based on the plot, we know that we maybe don't want to include the older ages in our correlation number. 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.

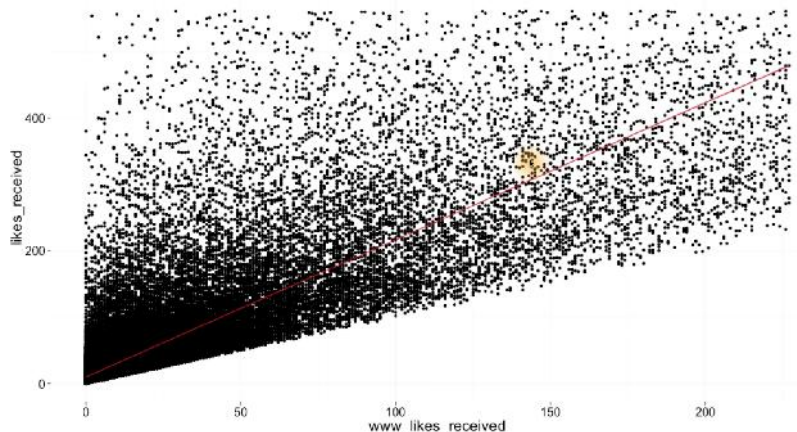
Answer:

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.

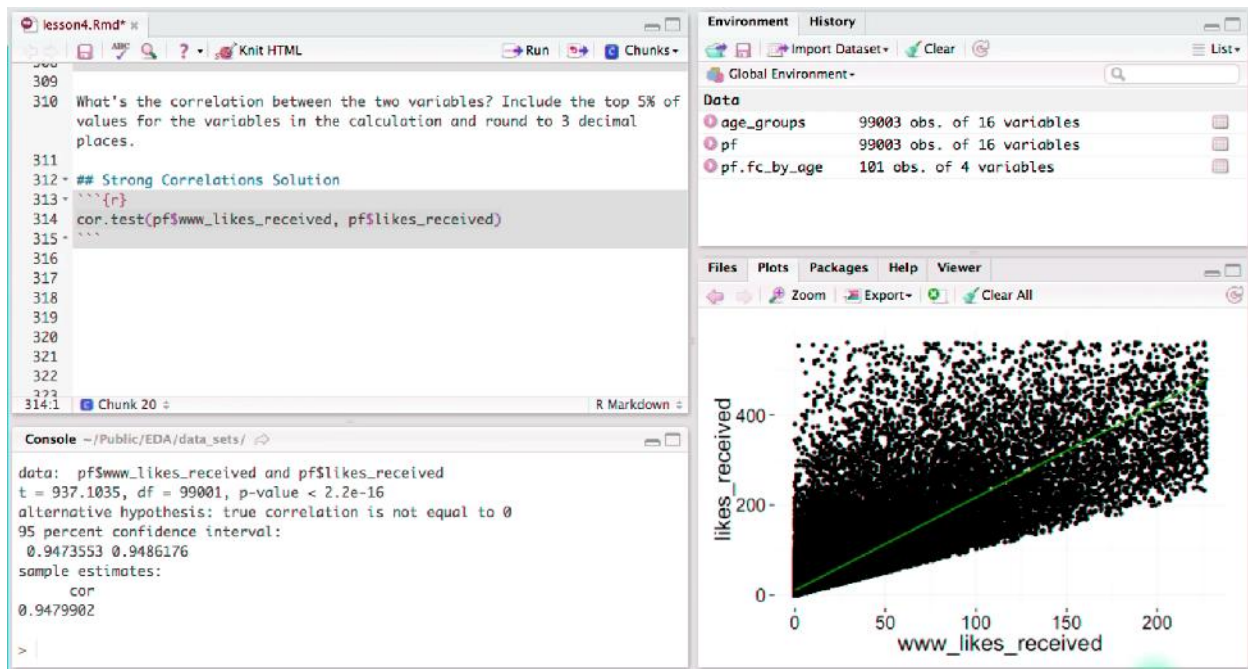
Answer:

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

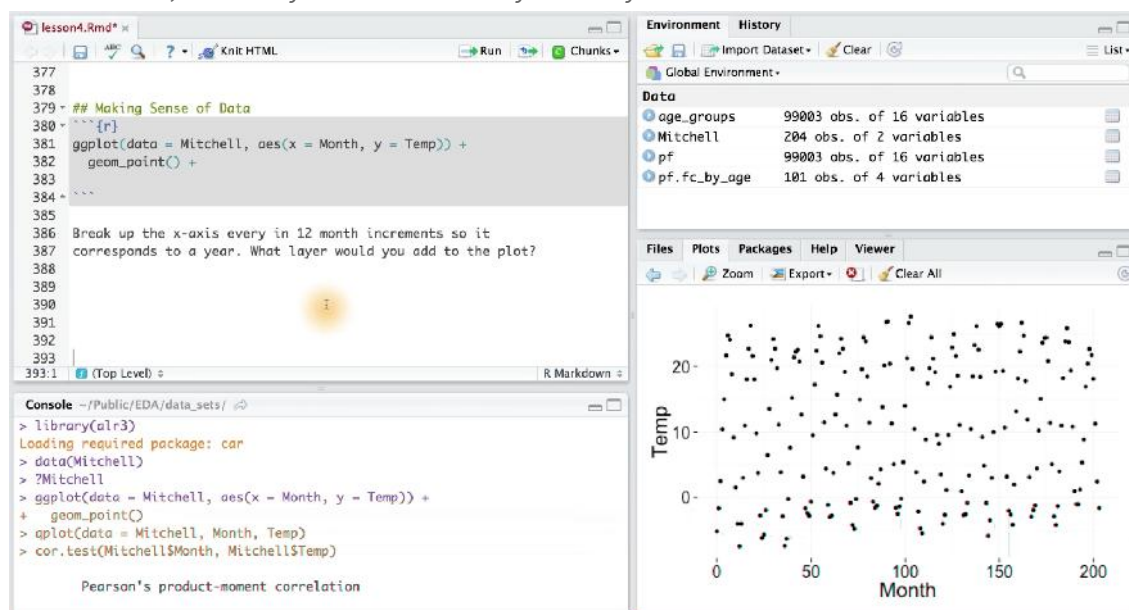
Now that we've got a scatter plot. I want you to take a guess as to what you think the correlation coefficient is, between these two variables. Write that answer in this box. After you've made your guess, I want you to calculate the actual correlation between the two variables and put that number here. 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 `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?

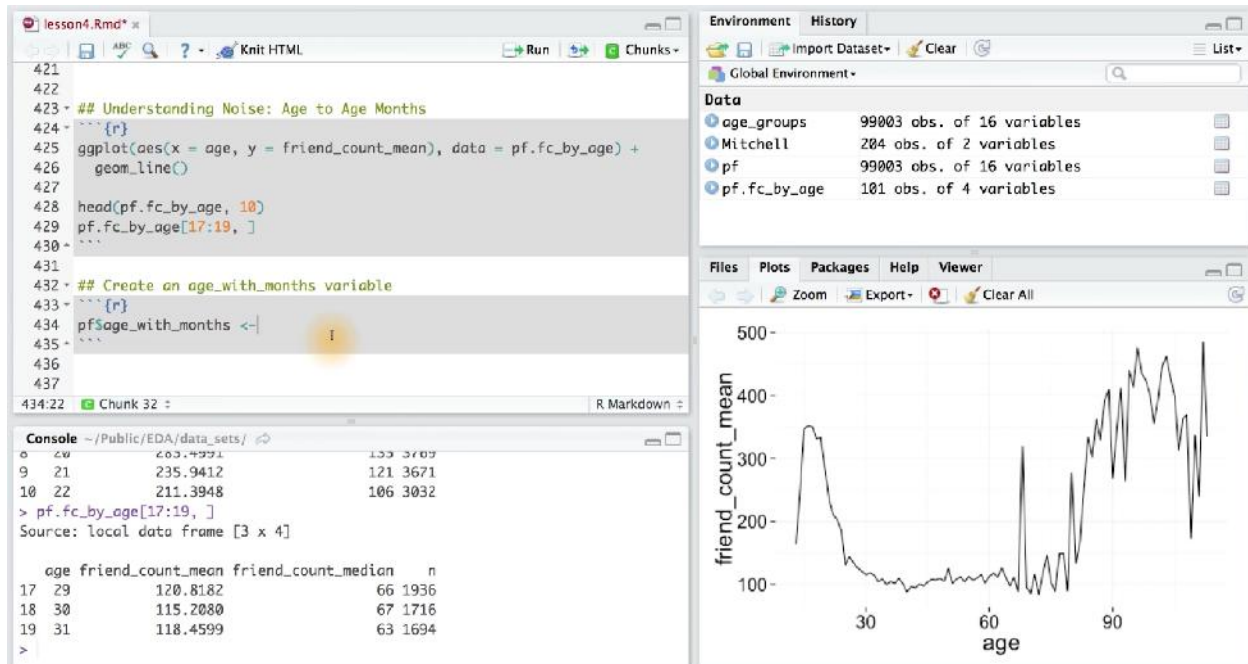
Answer:

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 `dplyr` 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 `n`, 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

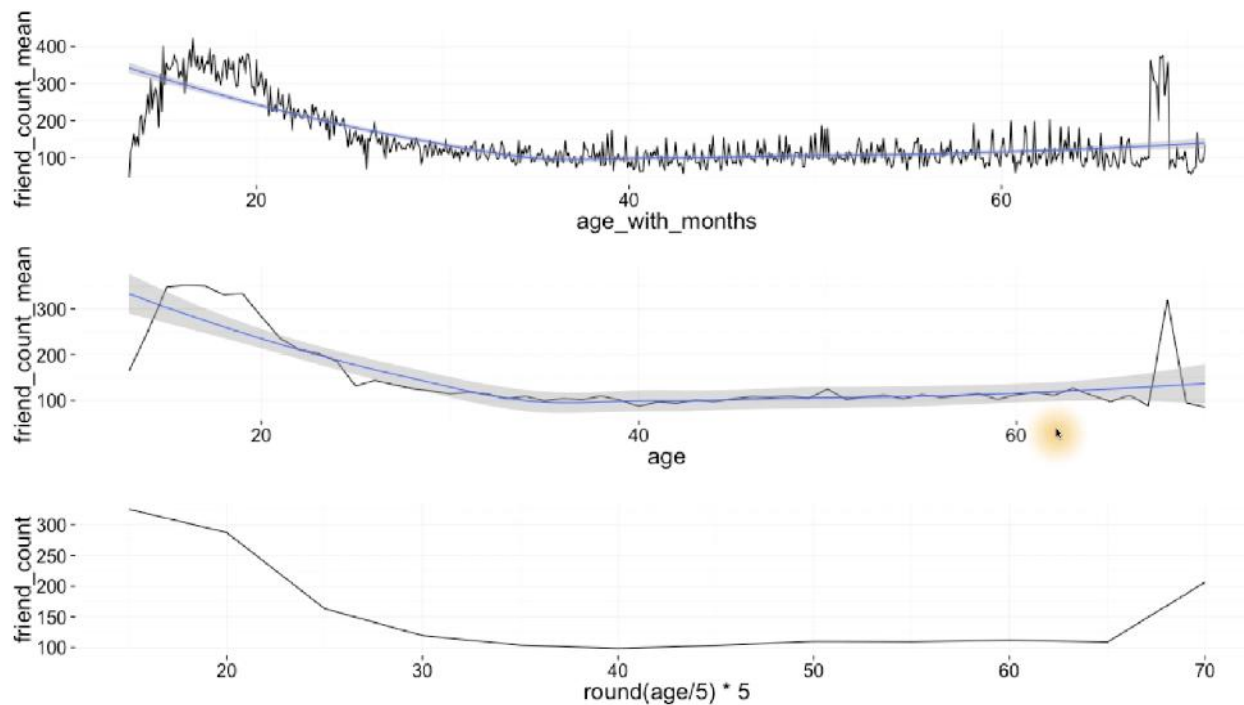
This is great. We've got our data frame, with our conditional means measured in months. Now it's just time to plot them. Your final programming task here is to make the plot of mean friend count versus age measured in months. Be sure that you use a line to connect all the points just like we did in this 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_line. So here's our much noisier plot, a friend count mean versus age with months.

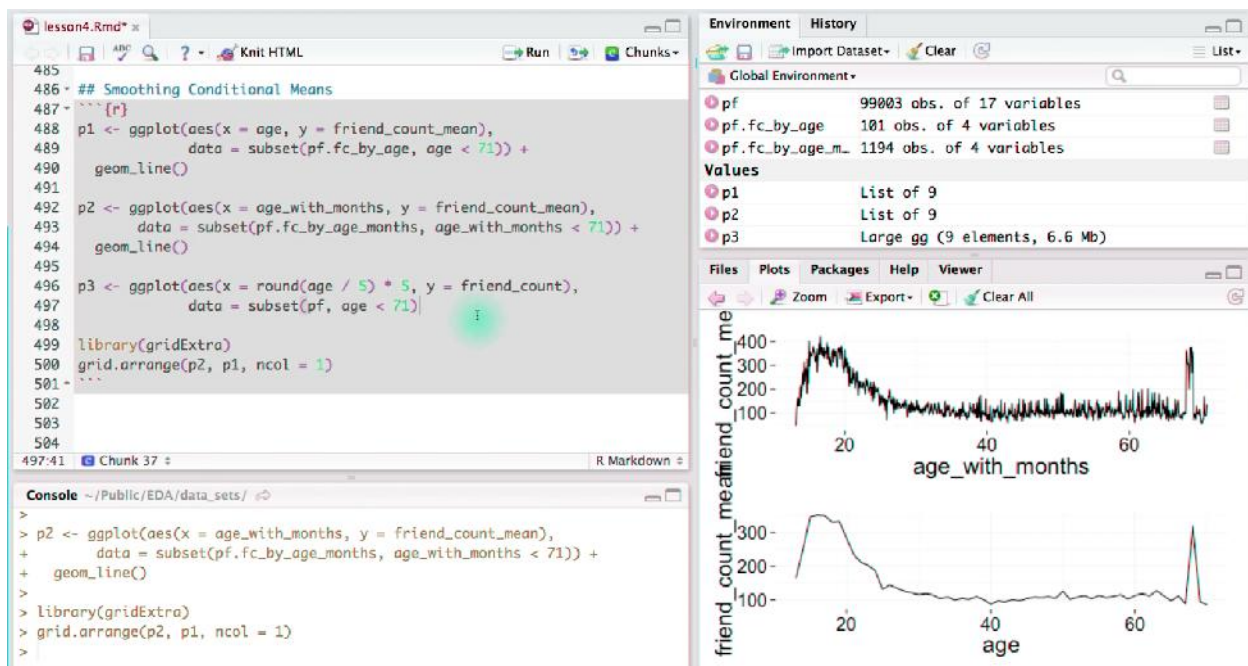
Smoothing Conditional Means

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 users 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 subsetting 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, gleaming different insights 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.