**Preparing Data: Video Transcript**

In previous data exploration sessions, we started the process of examining our data more closely and learned how to convert it into a tidy format. Now we're ready to create a visualization or a plot to see what the values look like. We're going to cover plotting in much more detail in following sessions. But we've included it here to emphasize how important plotting and visualizing your data is right near the beginning of our project as part of the exploratory phase, not just during analysis and also to show you how cool it is that you can create a plot with only a few lines of code.

If you remember when we first started exploring the data, some suspiciously high-value values jumped out at us, and by plotting the data, it should make it much easier to spot any outliers or potential errors in the data. So let's try it. We're continuing from where we left off in the last session. So our data is now in a tidy format, but there are still errors in our values. I've already loaded the data into our environment and it's called cancelled tidy. The code to load the tidyverse already at the top here. So we just need to make sure that we run it again by pressing control and enter. Then for our next chunk of code, will include a quick comment to save what we're going to be doing, plot exploration.

To create a plot, we need to start with our data, just as we always do. So cancelled tidy. Then we're going to pipe that into the plotting function ggplot. This lets R know that we're about to create a plot. Next, we include AES, which stands for aesthetics. And inside these brackets, we're going to put information about our x and y axes. So we want time along the x-axis. Let's check our time variable. It's called Months. So x equals month. And let's look at a count of total operations on the Y-axis. Now R doesn't yet know what type of plot we want. It just knows that we want to plot. So let's add the information about the type we use. The plus sign at the end of each line, when we're writing plotting code, not the piping operator. Don't worry too much about this, just now, I will go over it in a future lesson. We just need to remember that every new line after the ggplot command needs a plus sign next to our plot type. This lets R know that we want a scatter plot, so that we can see each individual data point. Let's run that and see what we get. Control and enter.

You might notice there's a warning message down in our console that's just reminding us that eight of our rows had missing values. And that's the Grampian missing data from 2015. We can see a plot now over here in the plots tab. And we can certainly see that there is a mistake with a particularly high value in 2016. To examine this further, we can reuse the code for sorting we wrote when we first started exploring our data to see what our maximum value is for total operations. So first we're going to add a comment. Now let's copy our previous code, Control-C, Control-V. Then let's change this object name to check\_total\_op. We have to remember to change our data. It's now called cancelled tidy, and that's piped into our arrange function. But instead of looking at the variable Month, we want to look at total operations. So let's change that too. Control and enter to run. Then if we look at this object, we can see that it seems to just be this one data point, which is wrong in the total operations column. But let's put it in context with other currents from NHS Grampian and to see how it compares. To do this, we need to add in filter to our code. So we're going to pipe it into our filter command. Then the column that we want to filter by comes next. That's health board. Then double equals to let R know that we're carrying out a comparison between two values and NHS Grampian and in inverted commas. Let's run that and we'll look at the data again. This time we're just seeing Grampian and data. And we can see from the surrounding values, it looks likely that the six key was held down too long and that the value should just be 3566.

When we spot incorrect data, we have a few options for how to deal with it. And what we choose to do depends to some extent on our dataset, what the nature of our analysis is, and also how confident we are that the data is wrong. We could drop the incorrect value by filtering it out, or we could set it to missing. Or if you are confident of the true value, we could correct it. Let's say we recheck the original source and we're confident we know what the correct value should be.

We're going to practice using the mutate function to overwrite our data. So first let's include a comment, fixed mistaken high-value. Then we choose a name for our corrected data. Now we're hoping that this will be our final working data set. So let's call it cancelled data. Then our data next, cancelled tidy, which we're going to pipe into the mutate function. Remember, this is the function we used when we wanted to correct our spelling mistake in a previous session too. It's a useful function for adding columns to our data, but it's also useful for updating current columns in our data, which is what we're going to do here. Next. the mutate function wants to know the name of the column we want to create, or in our case, the column we want to overwrite. So that's total operations. Next, we want to use the Replace function. And this replace function behaves in a similar way to the Str\_replace function we use previously, but that was text and this is numbers. So we need to use replace instead. In the first argument of Replace, it wants to know what column to look at and we're interested in total operations. And the next argument we need to let R know what value we want to replace. And we can do this with a logical test. So total operations and our double equals. And let's check the number, the wrong number that we have, 356666. And finally, our last argument, what the replacement should be. So we're pretty sure that it should be 3566. We check with our original dataset, make sure that there are the correct number of closing brackets, and then control and enter to run.

We can check this has worked by rerunning our plot, but with our new dataset. So first let's copy our previous plotting code. Then will change cancelled tidy to cancelled data and re-run. That looks better. We no longer have a crazy high-value, but we're trying to observe change over time. So a scatterplot is not really ideal. Let's change this to a line graph. We can do this simply by changing geom\_point to geom\_line and then run that. So we definitely have lines rather than points now, but it's not looking great. That's because if we look at our data, we can see that for each month there are 15 different values corresponding to the 15 different health boards. And R just creates a line between each of these data points, which is why we end up with a lot of vertical lines.

If we make each line a separate health board, that will be a much better representation of the data. To do this, we need to add in group beside our x and y arguments. And we're going to group by health board then control and enter - much better, although it's still a bit of a jumble and looks very plain and black and white. Let's add color with a separate color for each health board. We can do this by adding color equals health board beside our group argument. And let's run that - much nicer.

Exploring our data with plots like these can really help us to check if the data look sensible. Here we can see an element of seasonal fluctuation, which we would expect. And also visible is the drop in planned elective operations due to the coronavirus. With the most recent data from the month of March, you might remember from our previous explorations that there was another column with suspiciously high value. So let's take a look at that too. It was the column nonclinical capacity reason. So let's change our y-axis then. And re-run it. We can see here a few large fluctuation. So we can't be sure just looking by eye, if these are correct or not.

Let's go back to our data and have a look to see the context. If we use the sorting arrows here, we can see that there are a few high values in this column, as we already discovered from the plot. But over here we have a column with aggregated data. It looks like this column total cancelled might be the sum of the following four columns which show counts of reasons for cancellation. Excellent, we can do some validation on this. And it's also another good opportunity to practice using the mutate function. But this time instead of overwriting a column, we're going to use mutate to add one to the end. We're going to create a new totals column, which we can check against our original totals column to see if there are any discrepancies.

Ok. So first a note to ourselves, check sum of total cancelled operations. The name of our new object is going to be checked totals. Next, our data, which we're going to pipe into the mutate function. We start with the name of our new column. Previously, this was the name of an existing column we were overrating. Now we want a new column and we're going to call it totals check. Then we're going to add up the total of all of our reasons columns. So let's check what these are - that's these four columns here. We can just add these together like a sum. Let's check to see what that is looking like. So we'll run it first and then check our object. Here you can see our new checking column at the end. And we can also see a lot of matching values. Now in our code, we want to compare the two columns and only return rows with values that are not equal. And we can do this with the filter function. So let's pipe our data into filter, the two columns we want to compare - our total cancelled and totals check. Now we've already come across the double equals when we want to compare two values to see if they're the same. This time we want to compare them to see if they are different. So we need to change one of our equals to an exclamation mark. This represents not equal to. Then when we run that - look at our result we've managed to extract our incorrect record and can be happy in the knowledge that all other totals match the sum of the individual values. Excellent.

So in this video, we continue to explore in our data set. This time through the use of plotting to help visualize and look for errors or outliers in our data. In the coming lessons, we'll explore the possibilities of plotting in a lot more depth. In the meantime, you can start exploring your own data. Good luck.