**Tidying Data: Video Transcript**

So you've had a first look and explored your data, discovered some of its flaws, and now you want to know how to tidy it up to get it into a more useful format for analyzing and plotting. That's what we're going to cover in this video, we're going to be using the public health Scotland cancelled operations dataset again. But as I mentioned in the previous session it's too tidy to practice tidying on. So I've messed it up a bit and preloaded it into our environment, it's called cancelled messy. Here it is. This is definitely not what we would call a tidy data set. And we need it to be tidy so that we can plot our data easily.

In a tidy data set, each variable must have its own column, each observation it's own row, and each value its own cell. Here we can see that the Health Board column names are actually a list of related values, not individual variables. These values are a complete list of all the different health boards in Scotland. So we should make a new variable called Health Board to tidy these values into. And we also have the opposite problem too. We have variable names represented as values. That's the categories in the Measure column. Total operations, total cancelled operations, cancelled reasons. These are actually variables, not values, so these should each have their own column. Instead, if we try to plot this dataset to check if the values looked okay, we immediately run into some issues.

So how can we transform our data? The package which can help you fix your messy data is aptly named tidyr. And you'll be glad to know that it's part of the tidyverse. Here's our script with the code from our previous session when we were starting to explore our data. So let's carry on from where we left off. The command for loading the tidyverse is already there. So let's just run that line - control and enter. Now we are going to tackle the issue of the column names. First, the individual health boards being values. We want to bring them into one column. And we can do this using the function pivot longer. A nice way of remembering this function is to think of a table of data at the moment which is quite wide due to each individual health board having its own column. And we're going to make it longer instead.

First, a note to ourselves, create a tidy data set. Then we'll call our new object cancelled tidy. So our name goes on the left-hand side of an equal sign. And could we could've called it anything. And then we're going to start with our current messy data, so cancelled messy, which we're going to pipe into our tidying function, Pivot longer. I'm forever forgetting which arguments are needed for which functions and in which order. So let's just check the Help tab. We need to make sure our cursor is within the name of our function. And then press F1. So the first argument is data, and the pipe has taken care of that by piping the object called cancelled messy into the first position. Next, we have the argument calls, which we can see refers to the columns we want to put into the longer format. So let's add that argument. Now I am going to have to put the names of the health boards in inverted commas because there are spaces and the names, and I will get confused. This is similar to the issue we had previously with the select function. NHS Ayrshire and Arran. And I don't want to have to write out every health board. So instead I'm going to use the colon to signify a range. This tells R that I want to include the values on either side of the colon and all those in-betweens. Well then let's check the range that we want. We're starting with Ayrshire and Arran and ending with the Golden Jubilee national hospital. Control and enter to run. Let's check our data. That looks good. Although the name of our column is not a very informative name, it's been generated automatically by R. Let's see if we can improve on this. Let's check the Help tab again. Okay, the next argument after calls the 'names to' argument allows us to specify the name of the column. So let's add that in comma then names to and health board. Now when we run it, let's have a look... That's better. We can see our new variable name here.

Next we need to fix the opposite problem where we have variable names listed in the Measures column as values. We're going to use the function pivot wider for this, which again is easy to remember because by creating new variables from the Measure column, we're going to make a table of data wider. So let's try that. First. We need to add a pipe at the end of the last line that we did. And we're piping this partially tidy dataset into our new function, pivot wider. Let's take a quick look at the help page for this function to remind ourselves of the arguments. So again, the first argument data has been taken care of due to the pipe, which has already sent the data to the right place. Then we have ID calls, which we don't need to worry about in this dataset because the default will correctly identify how to define our unique observations. If we had more variables in the dataset, we might need to use this argument. Next, we see the pair of arguments 'names from' and 'values from' And these are the ones that we're interested in. We want to take the variable names from the Measure column names from measure and for values from we need to let R know where our values are for the new columns. And this is easy, they're in the Value column values from, let's see what we get. Now that's definitely looking better. But at the end here we've got the column with lots of NAs. There are only a few values. That's because it's the spelling mistake we discovered when we were exploring our data and we've not corrected it yet, we can run this code again from our previous session. And there it is. Let's correct it.

Now we'll add in our new code before our pivoting code. Let's write a quick comment. Fix a mistake. We're going to overwrite the mistaken value in our data set. So this time, instead of creating a new object, will just write the name of our current dataset, canceled messy. And then we'll take cancelled messy again, which we are going to pipe into a function called the mutate function. The mutate function is extremely useful and you'll find yourself using it again and again. It can be used to add a column to your dataset, or it can be used to overwrite an existing column, which is what we want to do here. In order to correct the mistakes in our data, first we need to specify the variable or column that we're overwriting. So let's check that this is the Measure column there was mutate is going to update our column. We also need the command to update only the incorrect values. And for this we can use the function str\_replace. The 'str' or str stands for string, and this function comes from the stringr package. We call variables which contain text or words strings. And knowing how to manipulate these variables is crucial when cleaning your data. The package stringr is excellent at this, and has many useful functions like trimming off whitespace around your text. You should definitely explore these on your own time. So let's add in our string replace function. In the first argument, it wants to know what column to look at. So that's measure. The next argument is the thing that you want to replace. So operation, with our incorrect spelling. Next, we specify what a replacement should be. So operation. Make sure that the correct number of brackets is at the end, then control and enter to run. Now we can check that it's worked by rerunning 'typo', which previously picked up on the mistake. There it is from before. Let's run it again. And we can already see there are no observations. Now we can also rerun our pivoting code and know when we look at our columns, we can see that our last incorrect column has disappeared. Much tidier now. Brilliant!

We're going to have a lot more practice using mutate in the following lessons as we continue to tidy data. But for now, we have succeeded in getting our data ready for plotting.