**Importing Excel Files: Video Transcript**

R is capable of reading in various different types of file. We've already explored how to import csvs and in this video we're going to look at how to import another common type of file you'll most certainly have encountered - an Excel file. Excel is one of the most widely used data analysis tools around. And the good news is that R can easily read in Excel files, both the older .XLS type and the newer ‘.XLSx’. Even more conveniently, one of the best packages to do this readXL, is already included in the tidyverse set of packages, which we're familiar with. Let's give it a go.

First, we'll open up a new script and let's save it Excel import and say OK. So we need to start by loading the tidyverse as we have done previously, control and enter to run. Now, although the package ReadXL is part of the tidyverse set of packages and so was installed as part of a tidyverse, it's not one of the core packages, and so it's not automatically loaded. When we load the tidyverse, we have to do that separately. So we'll just add it in underneath - library, It's there, down and Tab to complete and then control enter to run. We're also going to make use of one more package in this video. And that's the *here* Package, which you've already come across. You might remember from the previous lesson on importing CSV files. The package here makes it much easier to locate files in our computer. So let's add that to library. Here.

I've already pre-saved an Excel file of cancelled operations dataset in the data folder within my project. So I'm going to read that and it's a good idea to check the data in Excel before you import it so that you know whether the import has worked or not. So let's take a look. We can access the file directly from within RStudio by going to the File tab, navigating to our data folder, then clicking on the file. Here it is, and selecting View File. We can see that our data is stored with the tab for each year. Now let's try reading the data in with an Excel import function. We want to save this data to our environment. So let's think of a useful name, Cancelled data. Then we call on our function readXL, then we're going to add in the command here so that there is no ambiguity about where our file is if we share this project with a colleague. And the first argument of the here function is the name of the subfolder 'data' within our project folder. Let's put that in. We have to put it in inverted commas. Then after this comes the name of our file, PHS\_cancelled. We have to remember the extension .XLSx. And that again is in inverted commas, then control enter to run -great!

Now let's take a look to check that it's worked. That looks good... Oh, but we've only got data for 2015. So by default out readxl command is only reading in the first sheet. But what if we want the second sheet? There's another argument within the readxl function which can help with this. Let's check the Help page to find out what that is. We can either go to our function and press F1, or we could also have typed the function in the search box here. Okay, so we can see that it takes a sheet argument, and that it's happy to have a sheet name or a number referring to the position of that sheet. Let’s try it.

Now, the whole of the here command is our first argument within the readxl function. So we need to put a comma after this for the next argument then sheet equals and we'll type the name of one of the other sheets. Let's try 201, then control and enter to run. Let's have a look at our data now and we can see that it's changed. Now, instead of showing data from the year 2015, we can see the second sheet with the 2016 data has been read in instead. Another cool feature of RStudio, which we used when importing the CSV file is that has a data import wizard which can help us if we forgotten the code to read in our data or if our file is a bit messy, it gives us a chance to preview it to see how R is going to cope with importing it. So let's try this option too. We can move over to the File tab, find our file, and click on Import Dataset. This opens up the wizard. Down here we can see the command we've just used, the readxl function. And this bit of code is not using the here package, so the file path is written out in full. As I mentioned, this makes it less compatible, but we can always change that later if you want to. It's a very similar layout to when we were using the wizard to import our CSV file. So we can choose to do similar things. We can skip columns if there are no values in them, just as we did with the CSV import. We also have a date option for our month variable. But, oh no! what's it done? It's not asked us what format our original data was in, it's just guessed, but it's not got it right. Let's change it to character and we can deal with that later. The CSV import wizard is clearly better coping with dates than the excel Import Wizard. At the moment, we're just seeing a preview of the first sheet. So that is data from our 2015 sheet. If you want to view and import another sheet, we can change that here. Let's try 2016. And you can see in the code that this adds the sheet argument that we're familiar with from having tried it out ourselves.

Okay, I think that is as much customization as we want at this stage. So if you remember, we don't want to click import, we want to copy the code into our R script so that it can be used again and again. And we can do this easily by clicking on the clipboard icon over here and then cancel. And in our script we can paste the copied code, control and V, we will delete the first line because we've already loaded the readxl package, we can delete the view function as it is not necessary. Now let's check that it runs, and there it is under data. If we click on it, we can see it has been successfully loaded without our empty columns as we asked it to. Now try loading and some of your own messy Excel files to see how they look and see how well R does at guessing what types of data you have.