Data Storing Upload script

Hello! In this short video, we will look at ways of creating and storing data, and then we'll see how you can upload and look at your data in R.
I will show you an example dataset in a spreadsheet, and then I will do a little bit of live coding in R, to show you some useful functions for loading data into R and having a look at the dataset.
So, as we think about creating and storing data, I wanted you to think back to the little exercise that you completed as you were learning about data types. The exercise asked you to imagine that you had a Research Assistant collecting some simple data from patients in a walk-in clinic. While these data are perhaps not very complicated, it's still worth thinking about the format that the data will be in, and ways of storing them. Would you give your RA a clipboard and ask them to write things down? Would you give them a tablet to write the data straight into a computer file? What software would you use to create that file?
A common way to create and store data is to use a spreadsheet. You're probably familiar with spreadsheet software, such as MS Excel or Google Sheets. A spreadsheet is basically a table for putting data in.
Let's imagine that we gave our RA a tablet with access to Excel, and they came back to us at the end of the day with a neat file. Let's have a look at that file in Excel. (open Excel)
You'll notice that in the spreadsheet we've got rows and columns. Every row represents an observation - in our case, a patient. Every column represents a variable - one bit of information about the patient. We have as many rows as patients, plus one that contains column names. You'll notice that this format makes it pretty clear what the structure of data is, and we've given our columns meaningful names, so that we know what each variable stands for.
Now, a quick word about file types, or extensions. The default extension for Excel files these days is .xlsx, which saves a lot of information other than the data - it saves information about colours, fonts, formatting, etc. Most of the time we are only interested in the data themselves, which means that data scientists often prefer to work with a simpler file, with the extension .csv. CSV stands for comma separated values, and you can choose it when you're saving your file (SHOW THIS). Once you have saved your file as csv, you can check what it looks like. When you open it with Excel, it looks just like any other spreadsheet (SHOW). But, if we open it with a text editor (in my case, TextEdit, because I'm working on a Mac), we'll see that the comma separated values name is very meaningful - the values are organised in rows, and are indeed separated with commas. There is no other information apart from the values and commas - this makes the file very easy to load into R.
Ok, at this point we are ready to move into R - I will do a little bit of live coding to show you how you can upload and check your data in R. Let's open RStudio and have a look. (OPEN RStudio)
Right, let's start a new script for our exercise today. I'll start by making a comment to describe what we'll be doing.
Next, let's load the required libraries. We'll be needing the 'tidyverse' for some of our operations today. You'll see that R tells us in the console which packages from the tidyverse have been loaded.
Now, we'll use the read_csv function to read our data in. The file is in the same working directory as our script, so we can simply use the file name in the function. We save the dataset as an R object - we can call it "data" for simplicity. Once we've run this line of code, we see that

the object “data” has now appeared in our environment - this means that R has it in memory and we’ll be able to call it when we want to perform other operations.

Now that we have the data inside R, I would like to show you a few ways of checking them. Whenever you get a new dataset, it’s a good idea to eyeball it - familiarise yourself with its dimensions, the variables it contains, its structure, and so on. One way of doing this is by pressing the little spreadsheet button next to the “data” object we’ve created (in the environment tab). If we press this button, we’ll see a new tab open in the script area, and this is a spreadsheet view of our data. It’s great for a simple check and works well with our small dataset - it becomes less useful when you’re working with data containing thousands of records and variables. Clicking buttons, however, is not the most reproducible way of doing things. If you wanted to view your data in this way and leave a reproducible trace of this in your script, you can use the “View” command (remember the capital V). It produces exactly the same effect as clicking the button, only more reproducible.

If you had a large dataset and wanted to have a look at the first few records, you can use the “head” function. This will show you the first 6 rows of your data - 6 is the default number here. If you wanted fewer or more rows, you can change the number by specifying the n argument within the head function.
Apart from the data, now you can also see the type of data present in each column - dbl stands for double, or number, and chr stands for character, or text data.
The head function produces a new object - in this case, it’s a tibble with 6 rows and 7 columns.

Another useful function to getting to know your data is “str”. Let’s try it. As you can see, it gives you information about the object, then the first few records from each column, and then the column types. The output is less neat than what we saw in head, and this is because the output of head was another tibble, and the output of str simply gets printed into the console.

And finally, I wanted to show you the “glimpse” function, which presents the same information as str, only in a different format. Let’s try and type it in. As you can see, we get the dimensions of data, followed by column names, types of data in each column, and a few records from each column (as many as could fit on the screen). The advantage of printing each column as a line is that even if you have many columns, you will see them all in the console.

Ok, we will stop coding for now and recap what we have done so far.
In this video, we looked at creating and storing data in a spreadsheet, and we also talked about the advantages of using csv, or comma separated values, files.
We then learned how to upload data into R, and how to inspect data using a few different functions. Thank you for watching, and see you in the next video!