[SQUEAKING] [RUSTLING] [CLICKING]

HELENA VALLICROSA: Now is where things get interesting and where we're going to use the power of the computer to make our lives easier and not prevent us from doing mechanical stuff over and over. So we're just going to set some commands, and the computer is going to run them for us without us being there repeating the stuff. So there are two different types of loops. We have the for loops and the while loops.

The main difference between them is that for loops are static, so they are going to be repeated for the amount of iterations that we set at the beginning. But then we have the while loops, which are the ones that are going to stop once a command is reached. So whenever there's a constant that has overpassed or whatever is a different that has been fulfilled. You'll see that further when I'll explain them a little bit more in detail.

So let's just go with for loop at the beginning. I'm just going to create a vector here that we're going to name here. I'm going to go from 2010 and 2015. Just run it so we have it. Also just as a reminder, we could do it like this and it would work either way. We have the same thing. So what are we going to do with this? There's a very fixed structure for for loops. It starts with the command for so the R session understands that this is a for loop. You'll see that the font color changes here.

We're going to set initially the amount of times this is going to be repeated. So it's going to be throughout the whole year vector. So it's going to be repeated 1, 2, 3, 4, 5, 6 times. And then this is going to be the iteration, so every time this is going to change. So when we go to the expression we are creating or as we want for loop to repeat over and over, this is the command that it's going to change. So in the first iteration, this part of the code is going to be 2010.

But in the second iteration, it's going to be 2011, and so forth until the end. So let's just see how it goes. Just click in here and you can see that we have been repeating the same thing for six times, but every time it's changed the command referring to year. All right. So let's just move into the while loop. In this case, I'm going to create the iteration that's going to be the number one.

So it starts kind of the same way as for loop. So we said we're going to create a while loop. So here the font changes again, and we set the parameters about this loop. So initially we want this to go until all the times that this value here is lower than six. So as long as this value is lower than six, just keep going and it's going to stop when this is not fulfilled. So we're going to print the value and then take the value and add one every time. So I'm just going to run this.

And what we have been doing is to add one at every iteration, and a loop stopped when this condition was not fulfilled. So at the time when we reached the six, six is obviously no smaller than six so here is where the loop stopped. Also there is something else it could become handy. Even it's not a loop, it's something that could be introduced inside a loop, which is the if else condition. So in this case, I'm just going to set x as a 2. And what we're going to do is say, OK, if this value is bigger than two, fulfill this command right here.

But if this is not the case, just do something else. So just going to run this. So we printed positive number because x is bigger than zero, so this is the command that has been executed. But if I change the 2 for a minus 1, for example, just run the command and then just run the whole if else. And now instead of positive number, we have a negative number because the route that has been followed is the second one.

So far, we've been working with functions that have already been defined or incorporated in some packages, but we can also create our own functions So they can help us into our purposes. So here I'm going to be creating a function called Celsius to Kelvin, which is going to transform Celsius temperatures into Kelvin temperatures automatically.

First of all, I have to name the function and tell R that I'm going to be creating a function. Inside the parentheses I have to specify what are the variables that this function would need. In this case, we would need an input of Celsius temperatures, and here is what we want the function to do. So we open the key. And these are the mathematical calculations we want to set up.

And finally, we need a return. So it's like, OK, once you've done all these, give me back this object here, which is the one that we have created right there. And obviously closing the thing. OK. This is the function that we've created. We went directly here in the subset of functions. We could go any time and see how the function looks like.

And now it's time to execute that function and obtain the results. So first of all, we create this sequence that goes from 1 to 30 and it's stored in the E. Mhm. And I'm just going to store an object. Let's say Kelvin to make it more elegant.

And we apply the function right here, the one that we created, the name that we gave to our function, Celsius to Kelvin. And then we-- you remember that we ask the function to work with this input, so we are saying this input that you need, take it from E. So if we run the thing, we have these results where all the temperatures have been automatically converted from Celsius to Kelvin, and here is how they look.

During these years of being an R user, I came across different good manners and tricks while coding. Those good manners and tricks made my life easier, so I'm translating them to you. Hopefully you incorporate them now that you are starting your journey with R so you don't pass through the different hassles that I passed through. The first one is to format according to the subprocess. What do I mean with that? So I'm just going to go through a complex function.

Here you can see that I indented the process, so I put more spaces or less depending on the subprocess I'm currently at. So at this stage, we have whatever is related to the function. But then if I get in a little bit-- I get into the for loop, and this is the different commands inside for loop. So this helps us visually keeping our code tidy and be able to reference where are we working on and what part corresponds to what section of the code.

Then it's also useful to follow the same policy with spaces. The reason for that is when you use the search tool. So if you are strict with your policy on spaces, then you will be able to find all the variables at the same time, and so you're not going to get any behind. Then we have a good tool to convert text into a comment and a comment to a text.

This is useful because when you are experimenting in your code and you don't know which one is going to be the good code, you keep converting them into comment or real code. So let's say this is something that I tried and I want to recover again, so you click on Command-Shift and C. So it's no longer a comment. So here you are saying, OK, this is the good code.

But then you keep elaborating your code. At some point you're like, OK, this code, I think I'm not going to use it anymore. I don't want to erase it because I might want to recover it, but I just can keep converting them into comment or into real code just by clicking this. There's also click on a key just to know where this key ends. So going back, again, to that function here. If you click on that key, you're going to get this one highlighted.

So this one that is highlighted is the one that's closing this key. So the function starts here and it's closed here. This is pretty obvious, but if you go to for loop here in the middle of the function, you might get lost. And it's like, OK, what is closing this key here? So this is the key that's closing, so the for loop is comprehended only here. So this is very useful as well. And here we have that you can hide the whole section just clicking on the arrow next to its title.

So I've been creating here different title sections, which are the same titles that appear here on the side. So this is something that I follow to keep my script tidy and jump from one place to the other pretty easily. So if I click in any of these names, I'm going to be teleported to the place that I just clicked, dataframe, go to dataframe. Let's just go back to where we were. So if you click on this arrow, we can make the whole section disappear.

It's not that I'm erasing the code. It's just that I'm folding the thing in so it's not occupying visual space. I can just recover things pretty easily just clicking here again. And finally, here is a way to free some memory. When you are working with very heavy databases, let's say maps or huge data frames, this is going to occupy a lot of space in your computer. And sometimes even if you clean your environment, this memory is not freed. So just to force the computer to free these memory, you just click on this command and run it and you'll see that your memory will be significantly cleaned.

And finally, to wrap up this session about introduction to R, I planned this practical example, which is a real example that I have used in my papers or in exercises that I do with R. We put together everything we've learned, which is to charge packages, to take these databases that are part of R, and combine everything in a function that also includes a for loop. And at the end, just make some rearrangements on the product and save the data.

So what I am doing here-- let's just first of all remind you how we charge packages and how we install them. This package, I already have it installed in my device, but you can install yours by going to tool install packages, again, and type the name of the package and then install. And then you charge it to your session using library. So I'm going to charge this to the session and I'm going to store this NPK database that's inside R, name it the same way.

So this is the database, and what I'm going to do in this function that I'm going to be creating is to create univariate models. So just as a reminder, we had this database where we have different blogs right here and three different treatments. So nitrogen fertilization, phosphorus fertilization, potassium fertilization, and finally, we have the product which is the amount of yield that has been produced after the fertilization.

So what we want to do in this function is to create univariate models. So yield as a response of nitrogen addition, yield as a response of phosphorus addition, and yield as a response of potassium addition one by one. So first of all, just this is going to be the name of the function. It stands for mixed models.

This is the function that we're going to be creating and these are the two variables that this function is going to need. First LM, which stands for elements, and this is the data. The data is going to be the NPK database and the elements are going to be the names of the different elements, so N, P, K.

What I'm going to be doing first is to create an empty dataframe.You can see that I'm going to be naming it out. So to create an empty dataframe you just set dataframe here and put nothing inside. Just going to see how it looks like. I created the object and it has zero observations and zero variables, so it's an empty database. You click on here. There's nothing there. So here is where we are going to be storing the information that we're going to be creating. What do I do?

So inside of this function, I created a for loop. This for loop, how long is it going to run? So it's going to run as long as the element is. So in this case, I'm going to create the element, which is which includes the column names from NPK that goes from two to four, which in this case is going to be N, P, and K. Let's just check it out. N, P, and K. So it is going to run for three times, three iterations. one for N, one for P, and one for K.

What I'm doing here is to create the function itself, so the model that I'm going to be creating. This is going to be LMA, the yield, which is going to be the y variable, and every iteration I'm going to change the independent variable. So the first run is going to be the nitrogen and so forth. And then I also include the random factor because this is going to be a mixed model. The random factor is going to be the block. Then we have the data and the inaction requirements.

OK. Then on the next line we have the evaluation of this form. So let's just click-- well, I'm just going to set the y as one, so let's just mimic the first iteration. I'm just going to check line by line what happens right here. So we have the form. This form looks like this. And now we want to run this text that we have. So here is where we include a value and this here, and the text is going to be the form.

So this line of code is going to execute this model, so we just click here. Oh, it's missing something there. Let's see what is the complaint. Oh, yeah. Sure. I didn't say that data is NPK. So now I'm going to name data as NPK. Now it runs. So it shows we have that model, as we should.

And then I want to store the summary of this model in an object called sum and see what we have. So this is the summary that we would have in a normal model. And then I'm going to select the items inside this model that I'm particularly interested. So I'm going to get the T table. What is the T table? Let's just see.

Just the p values and the estimates and standard error and differential square and T values. Then I also want to store the Akaike values to know how good is the model. And finally, just to have a reference what's the element that we run in that model. So N. So I'm taking everything and I'm building another data frame that's going to be called sum mods, sum of models. You click here, this is what we're going to have.

And I'm just going to put it beautiful inside this output that we created at the beginning. So this result that we just saw is going to be stored there. And now the first iteration would be done and we would go to the second one. So we would repeat the same process but instead of the N, we would use the P. Then we would store this outcome again down to the-- we would have the outcome of the N, then the outcome of the P, and finally, the outcome of the K.

And once we would have that created, the for loop would stop and we would keep going in our function. So we would change the column names just to make them prettier so it's going to be named like this element and AIC. And we're going to have this as a return. So we want the function to give us back this out now that it's going to be filled. So let's just try it out.

Just click the function. Now the function is created. Let's just run the function. I want the outcome to be stored in treatment and I say the elements in this case are going to be N, P, and K, and the data is going to be PK. You could be able to change, obviously, each of these two variables. So if you would like to try that with another database, you could do that. That's the magic of the functions. And just run this and see what we have.

See? Here we have the outcome that we had in the first iteration, that's the second, and that's the third. Just as a reminder, R doesn't allow us to have a row name repeated. So in the second time it repeats intercept it has to put a one, and the second-- the third time it includes intercept it has to put two. So that's something that R does automatically to avoid repeated names and get confused on the orders. And just I decided to change the row names to make them more understandable. Again, I had to name them differently to avoid this confusion. So row names.

Now we have it nice and pretty to be understood, and that could be a nice table to publish in a paper where easily the results of the models could be seen. And also if you don't want to see-- oh, that I already have activated. But sometimes the numbers are shown in scientific notation. So if you want to change that and make them with commas, you just run options, type in 999, and the scientific notation is going to be changed to decimal.

And finally, we want to store this outcome that we created, this treatment, into a file. So in this case, I'm going to be creating a CSV file. And you can change the route here to install it or to create this in your laptop, whatever you want. And just run this, and you're going to have a beautiful outcome of this function.