Exercise 3 – Classification with R

**Day 4**

1. **Running commands:**

As per the lecture, R commands are run in Rstudio. In contrast to the Bash terminal, you have an editor to write in before you execute the commands. A couple of tips:

## Running commands:

You can run one line or several at a time depending on what you want to do. If you press  
 **Ctrl+Enter**  
Rstudio will run the line you are currently at. The clever thing is that you can be wherever you want to in your script, line 1 or 500. Similarly, you can select several lines and run them at the same time. If you want to run your entire script, the easiest way is to use  
 **Ctrl+A**and then  
 **Ctrl+Enter**Which will select all and then run it.

Exiting unfinished commands:  
Sometimes, you might get stuck and your command wont get executed no matter how many times you push Ctrl+Enter. In your console, it should have this (i.e. the “>”):

A screenshot of a computer

Description automatically generated  
If you instead have this (i.e. the “+”):  
A screenshot of a computer

Description automatically generated  
R is waiting for you to finish a command, like closing a paranthesis or similar. Either finish the command if you can, or exit by clicking the console window and press **Escape.**

1. **Everything must be in an R-project**

The R-project is a need way to keep track of what you do for a particular project. Technically, it is just a folder with a bit of info for Rstudio, but it has a couple of neat features, such as keeping your path where you want it and allowing you to interface with various version control systems and websites. **These exercises wont work if you don’t use projects.**

1. **Make an R project and set it up correctly**

Make a new R-project as described in the lecture. Put it on your Desktop and call it something easily recognized, like PlayingWithR. We will make a new project for the real stuff.

Make sure you can find the folder with your operating system, then make 3 folders there:

* Hello
* Professor
* Mikael

Make sure you can now see those 3 folders in the right side of Rstudio

1. **Make the real project**

You know how 😊 Call it something that makes sense.

1. **Downloading data**

The data we use is from Pernilles paper (https://www.nature.com/articles/s41522-023-00379-3) . I have made a subset of a couple of samples, which we will download from my github.

In the project, run the following two commands:

**download.file(url = "https://github.com/mikaells/SlovSchool23/raw/main/Biofilm.zip", destfile = "Biofilm.zip")**

**unzip(zipfile = "Biofilm.zip" )**

**? What happened?**

**? Can you find the data you just downloaded?**

1. **Set up paths, packages and data**

Download and open the DADA2.R script in R.

**download.file(url = "https://raw.githubusercontent.com/mikaells/SlovSchool23/main/DADA2.R", destfile = "DADA2.R")**

**The list below correspond to the headlines in the script!**

**Again, the numbers in the list below correspond to the numbers in the script!**

This script has mostly been filed out but requires a couple of additions. Whenever you see XXXX, you need to fill in yourself

1. Install the dada2-package and read it in. dada2 is a Bioconductor package, so you need to use the BiocManager::install()-function.
   1. How do you install BiocManager in the first place?
   2. What text to you need to replace the “XXXX” with?
   3. Remember to remove the “#” from the installation command!
   4. Remember to read in the dada2 package! It is now a library and can be read as such.
2. We will first create variables which contain the paths of our files. Make sure that the variable ‘path’ contains the path of the data you just downloaded.
   1. Remember that since we are working in a Project, the folder with the files are right where you are.
3. This is a tricky little set of code to get the correct paths of the individual files, and has already been filled out.
4. Plotting quality profiles. These plots are useful to investigate if something is wrong with the data and if/where sequences should be trimmed. See if you can plot the 3rd and 4th forward reads.
   1. What does 1:2 mean in the R-language?
   2. Are forward reads of better quality than reverse reads?
   3. Should the reads be trimmed and if so, where?
5. Quality filtering and trimming. Now we do the actual filtering, and here we write new (filtered) files to a folder called ‘filtered’ which DADA2 will create itself. Note that we will also save information of the filtering in an object called ‘out’ in order to fish out the trimming statistics.
   1. The truncLen parameter takes a vector with two values, i.e. the lengths of the forward and reverse reads after trimming. The forward read is fine at 240 bp, but what should the reverse be trimmed to?
   2. Be careful with the multithreading option! Make sure it is set to fewer threads than your CPU has.
6. Error models. Next we learn the distribution of errors in our reads, so that the DADA2 error-correction model can be started. These models are basically how likely one base is erroneously measures as another. Have a look at the plots.
   1. See if you can make sense of the plots. We will talk about them in class later
7. Run the denoising algorithm. Here, dada2 corrects individual bases if they are likely to be incorrect, and produces corrected, or denoised, reads
8. Merge the corrected reads. Here we stitch together each read-pairs by matching the overlapping regions.
   1. Are all the sequences equally long? Why/why not?
9. Remove chimeras/bimeras. These are PCR artifacts, from when a sequence acts as a primer on another sequence and produces a resulting sequence from two different parents. These should be removed.
10. Processing statistics. Now we are ready to summarize the effects of all our preprocessing. A neat table is found as the variable ‘track’.
    1. It would be helpful to get the table as a percentage of starting reads. Two hints – you can divide all elements of a table with a single division, and the input number of reads was 6500 in all samples.
    2. How many reads are left in the end? Which step was the worst for losing reads?
11. Getting the classifier data. To classify the sequences, we need a database to match them with. We will download a preformed database here. It is somewhat likely that this download will fail, but we will see in class.
12. Lastly, we classify. The taxa-object now contains all the taxonomies of each of our sequences!