Bi 410/510: Introduction to Programming for Biologists

Spring 2018

Project 7: Analysis Pipeline

The programs for this project each implement one of the "stages" of the 16S rRNA analysis pipeline (described in the lecture notes for May 21, in the file `pipeline.pdf`):

- the first stage combines paired end reads in FASTQ files and saves the results in a set of FASTA files
- the second stage removes duplicate sequences from the FASTA files, creating a new set of files
- the third stage uses a "clustering" algorithm to organize the data into groups of similar sequences

All three stages are Python scripts that run an application named `vsearch`. **You need to have `vsearch` installed on your computer in order to run your scripts.**

The program for the first stage was described in class. You can find a description of the program in lecture notes (`subprocess.html`) and can download a copy of the program, which is in a file named `merge_pairs.py` on Canvas.

Your job for this project is to write the programs for the next two stages:

- Write a script named `derep.py` that will run `vsearch` with a command that tells it to remove duplicate sequences from FASTA files
- Write a script name `cluster.py` that runs `vsearch` to produce OTUs from the dereplicated sequences.

Put both programs in a single folder named `pipeline`, then compress the folder:

```
$ zip -r pipeline.zip pipeline
```

Upload `pipeline.zip` to Canvas as your submission for Project 7.

Software Development Process

We strongly suggest you follow the guidelines described in class for developing new scripts:

- Create a "sandbox" directory where you can experiment with `vsearch`. Copy the `sim` folder to your sandbox, type shell commands that run `vsearch`, make sure you know what it is doing.
- Write a "spec document" that spells out in detail what your script will do. Specify the command line arguments (required or optional) and the data you expect the script to produce.
- Write a "design document" that summarizes the key steps and contains notes about how you will implement those steps.

The difference between the two documents is that a spec document describes **what** the script will do, the design document explains **how** it will do it.

Prepare the FASTA Files

To create the FASTA files that will be processed by your scripts download `merge_pairs.py` from Canvas. Create a project directory for this project, move `merge_pairs.py` to the new directory, and move your `sim` folder (or expand a fresh copy) to the same directory.

When you run `merge_pairs.py` it will create a new folder for the FASTA files it produces. You can name it anything you'd like, but we suggest calling it `fastas` and having `merge_pairs.py` create it in your project folder. This command will run `merge_pairs.py`, telling it to merge the FASTQ sequences in `sim` and put the resulting FASTA files in `fastas`:

```
$ python merge_pairs.py --source sim --output fastas
```

Project 1: Dereplication

To tell `vsearch` to identify duplicate sequences in a FASTA file use the `--derep_fulllength` option (notice the 3 l's in a row!).

This example shows how to process one of the files in the `fastas` directory and write the results to a file with the same name, but in a directory called `uniq`:

```
$ vsearch --derep_fulllength fastas/A.fasta --output uniq/A.fasta --sizeout
```

Your goal is to write a script named that automates this step. The script should identify all the FASTA sequences in a folder and run `vsearch` on each one.

Notes

- If the output is going to be written to a file in another directory the directory must exist already (this is the same rule `vsearch` uses for its other commands).
- The `--sizeout` option tells `vsearch` to include copy numbers in the output. If you look at the deflines for the first few sequences in the output you'll see something like this:

  ```
  >ART:0:0:810:412:436:962;size=14
  ```

  The "size=14" at the end of the line means `vsearch` found 14 identical copies of this sequence. They all had different sequence IDs in the input file, so `vsearch` just chose one to be the representative of the group and includes the group size in the defline.

Project 2: Clustering

To form clusters of similar sequences we want to collect all the sequences into a single FASTA file and then run `vsearch` on the combined file.

Combining the files is simple: just use the `cat` shell command. To combine (concatenate) all the unique FASTA files the command is

```
$ cat uniq/*.fasta
```

If you want to save the result in a file use redirection, *e.g.*

```
$ cat uniq/*.fasta > all.fasta
```

Now you can tell `vsearch` to form clusters:

```
$ vsearch --cluster_size all.fasta --uc X --id 0.97
```

where `X` is the name of the output file.

Your job is to write a script that automates this process. One way is to run two shell commands, one that concatenates the FASTA files and the second to run `vsearch` on the result. Another is to figure out how to "pipe" the output of `cat` directly into `vsearch`, without having to save it in a file first (read the "Input" section of the `vsearch` documentation to figure out how).

Notes: The `--uc` option tells `vsearch` to create a TSV file. Each line in the output will tell us how one of the input sequences was handled. Lines that start with S (for "seed") means a sequence was used to define the start of a new cluster. If a line starts with H (for "hit") it means the sequence is similar enough to one of the other sequences that it was combined into the same cluster as the other sequence.