21 May 2018

Analysis Pipeline

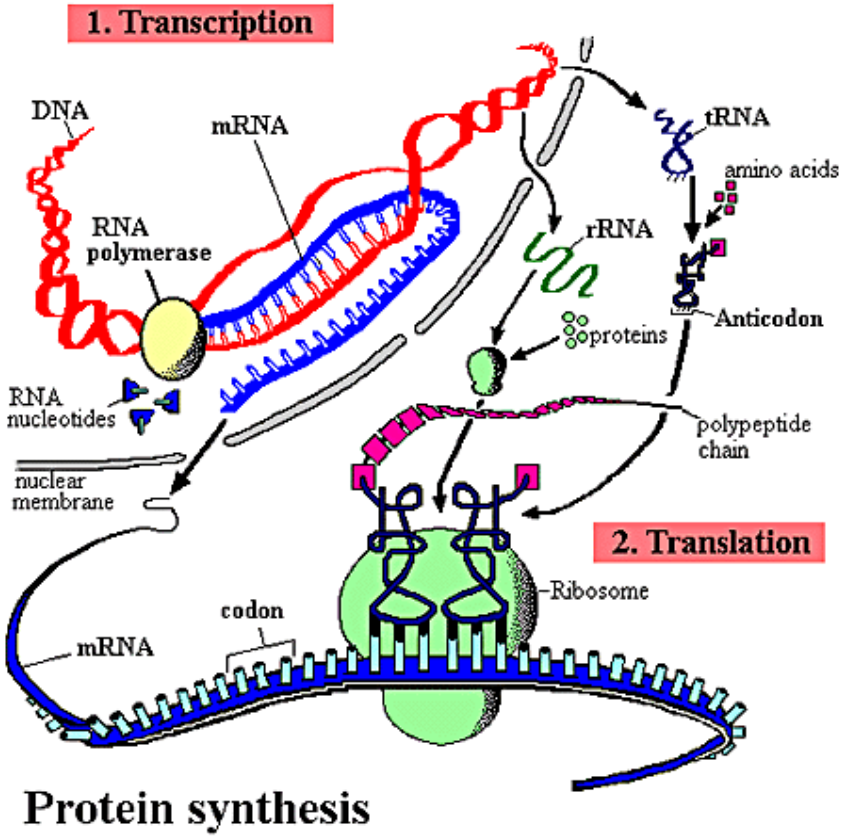
Unit 7: Python scripts to control an analysis pipeline

Topics this week:

- overview of 16S analysis pipeline
- how to write Python programs that run other programs
- examples: generate artificial reads used to test the pipeline, run `vsearch` to "merge" paired end reads
- project: write Python scripts to run `vsearch` to do the next two steps (dereplicate, cluster)

16S rRNA

Ribosomes are an essential part of the “machinery of life”



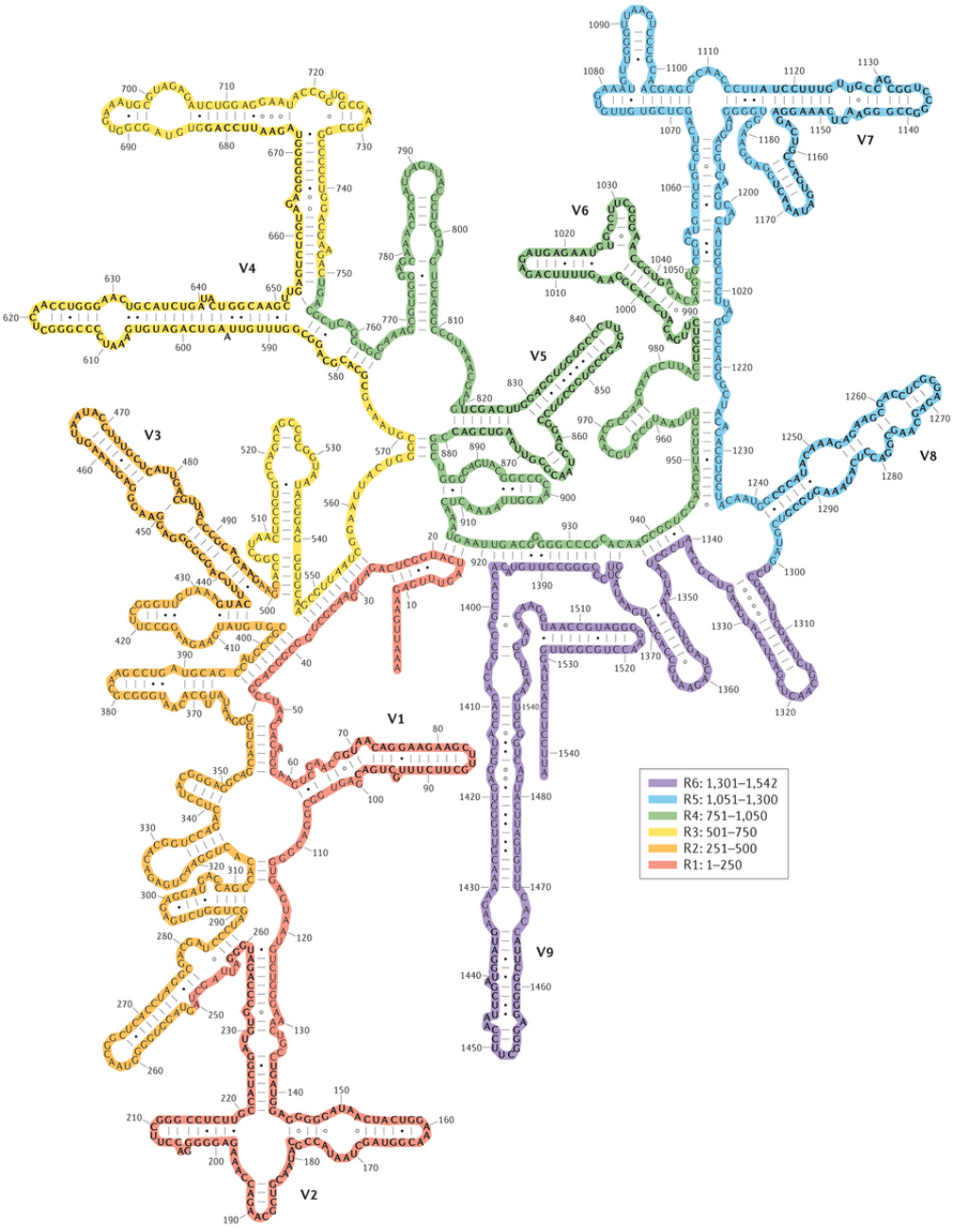
The ribosome consists of several strands of RNA, which are themselves the result of transcribing an RNA gene

Our data comes from the DNA sequences of genes that encode for the “16S sub-unit” in bacteria

Variable Regions

Within the 16S gene are variable regions, labeled V1, V2, etc; our data come from the V4 region

- the other parts of the gene are highly conserved
- the V4 regions act as a “fingerprint” -- they have enough differences that we can use them to distinguish between bacteria

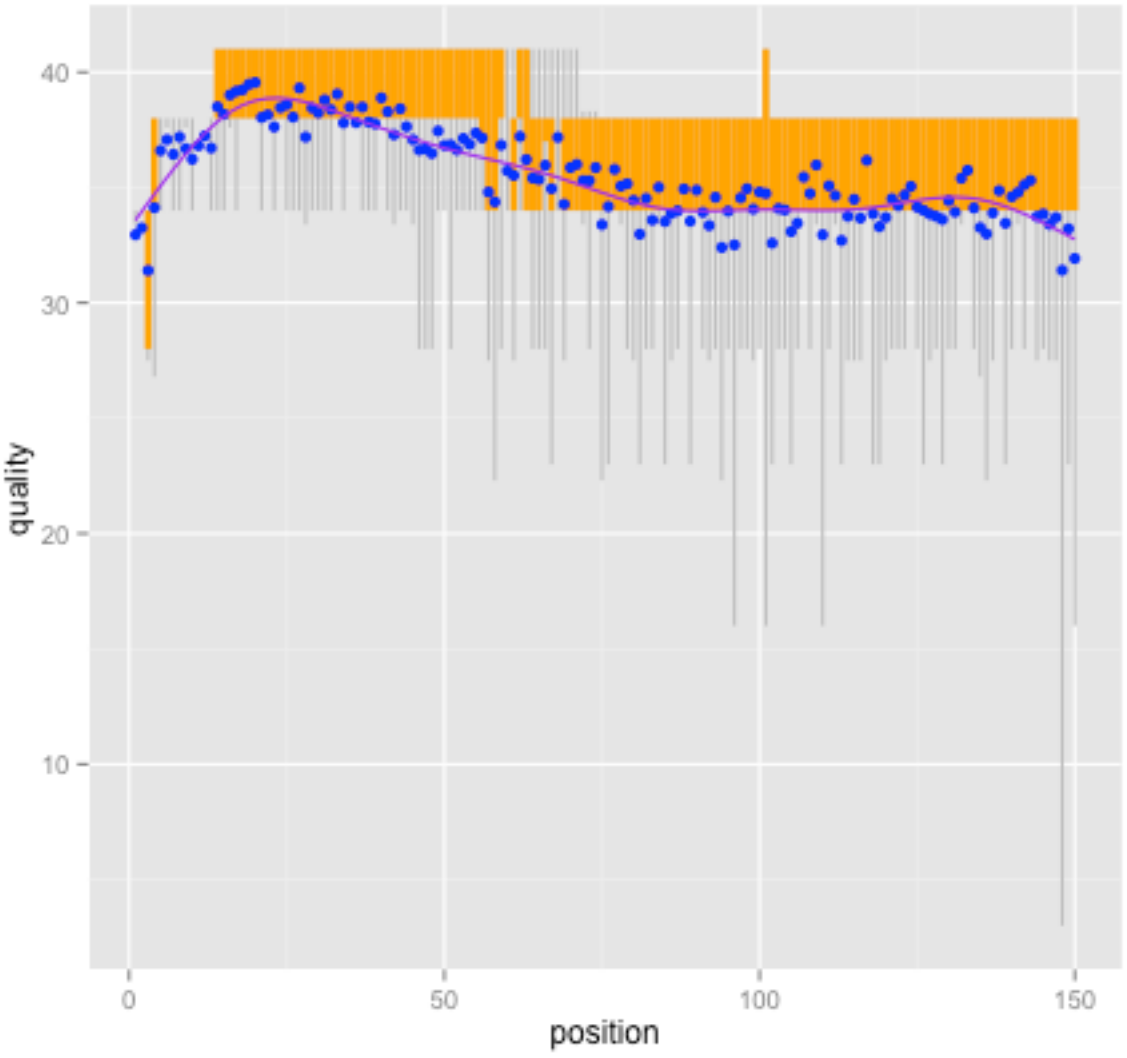


Illumina Data

The V4 regions are about 253bp long.

The Illumina sequencer can reliably sequence up to 150bp before the reliability of the data start to drop off

- each sequence in the data set is called a **read**



FASTQ

The most common data format for short sequence reads is called FASTQ

> "FASTA with quality scores"

Instead of two lines per sequence we have four:

- sequence description (line starts with @)
- sequence letters
- a line that starts with + (may have further information)
- quality letters

Example (first two sequences in `A_R1.fastq`):

```
@ART:0:0:810:412:436:984 1:N:0:0
TACGTAGGTGGCAAGCGTTGTCCGGAATAATTGGGCGTAAAGC...
+
CC=GGGCGGGGGGJJGGJCJJJJJGJJG$JJJGJ=JGJJGJCJ...
@ART:0:0:2358:123:2061:196 1:N:0:0
TACGGAGGGTGCGAGCGTTAATCGGAATAACTGGGCGTAAAGG...
+
CC=GGGCGGGGGGJJGGJCJJJJJGJJG$JJJGJ=JGJJGJCJ...
```

Quality Scores

The quality characters tell us how confident the sequencer is that a letter is correct

- lots of things can go wrong during the sequencing process
- we want to filter out sequences that may have sequencing errors

The character at location i in the quality line is related to the probability the character i in the sequence line is correct

Scores range from 0 to 41

- see the Wikipedia entry for FASTQ to see how to convert these integers into probabilities

Instead of using numbers we can use 41 consecutive characters starting with ! (ASCII code 33).

Character:

```
!  "  #  $  %   ...   @  A  B  C  D  E  F  G  H  I  J
```

ASCII:

```
33 34 35 36 37        64 65 66 67 68 69 70 71 72 73 74
```

Quality Score:

```
0  1  2  3  4   ...   31 32 33 34 35 36 37 38 39 40 41
```

Paired End Reads

We can sequence an entire V4 region if we used paired end reads

- use restriction enzymes to cut out the V4 region (around 250 bp)
- sequence each strand of DNA (one in the forward direction, one in the reverse direction)

We'll end up with two data files:

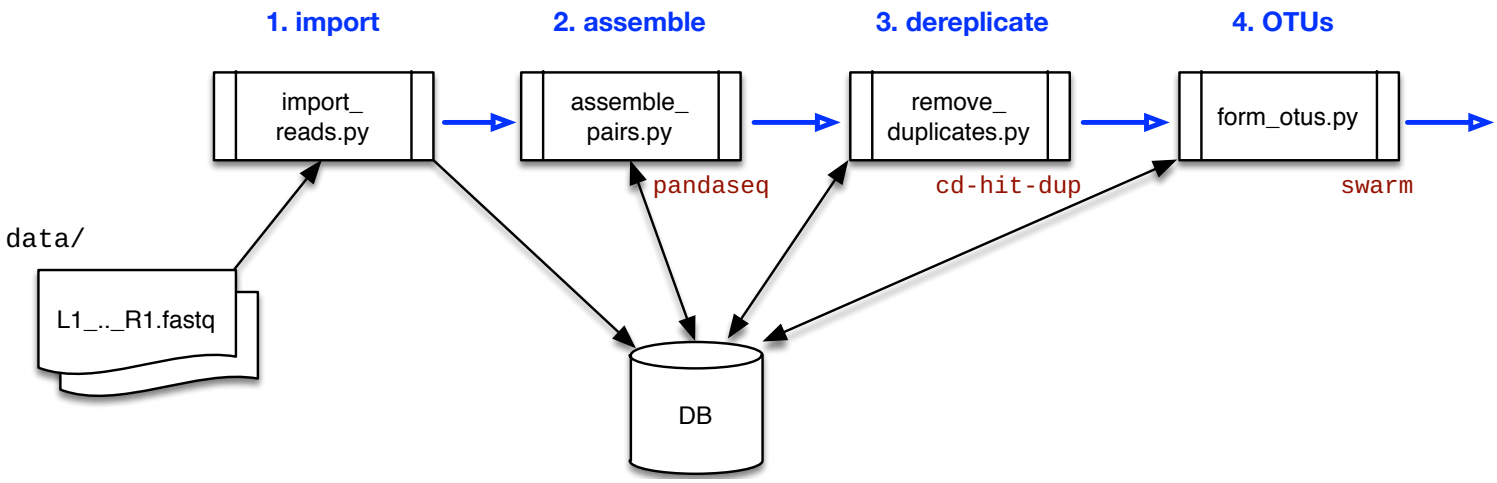
```
X_R1.fastq
X_R2.fastq
```

where X is typically the sample name

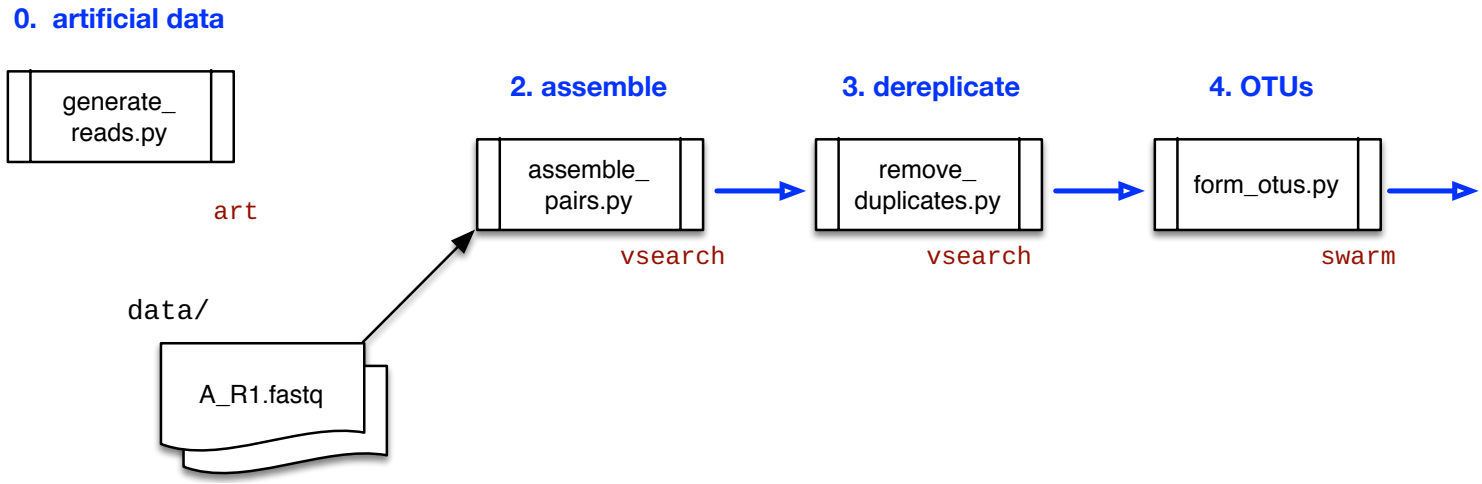
Analysis Pipeline

As part of the META project I helped develop a set of scripts to manage the data analysis

- we use SQLite databases to keep track of the results of each step



For our class project we'll use an artificial data set created to test the accuracy of the pipeline (and skip the database)



Pipeline Scripts

To generate the data in the sim folder I ran a Python script named `generate_reads.py`

- tell it how many samples to make (we have three: A, B and C), how many species to put in each sample, and the expected abundances

Then `cd` to the resulting directory and:

```
$ assemble_pairs.py --datadir . --dbname sim.db

$ remove_duplicates.py --load_seqs --dbname sim.db

$ form_otus.py --dbname sim.db

$ map_otus.py --dbname sim.db
```

Script Options

Each of these scripts has several different command line options

Example:

```
$ form_otus.py --help
usage: form_otus.py [-h] -d DB [--force] [--preview]
 [--workspace dir] [--singletons] [--application APP]
Create de novo OTUs.
optional arguments:
  --help              show this help message and exit
  --dbname DB         name of project database
  --force             replace existing data
  --preview           preview shell commands
  --workspace dir     working directory
  --singletons        include singletons
  --application APP   clustering application
```

```
$ form_otus.py --app foo
```

```
form_otus.py: error: argument --application:
invalid choice: 'foo' (choose from 'usearch',
'vsearch', 'swarm')
```