Apr 23, 2019

Working

02: Parsing FASTA files

Version 3

Frank Aylward[1]

[1]Virginia Tech

dx.doi.org/10.17504/protocols.io.z9mf946

Frank O. Aylward
Virginia Tech

ABSTRACT

Week 1
Introduction to parsing FASTA files.

Commands to be entered into the command line are in bold.

Here we will be using various base Unix commands such as head, tail, sort, wget, and others.

We will also be using the seqkit tool to process FASTA files. The main page for seqkit is here:
https://github.com/shenwei356/seqkit

1 Today we will be looking at the genome of Yersinia pestis, which can be found on NCBI at this location
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/009/065/GCA_000009065.1_ASM906v1

Copy this URL into your browser and take a look at the files. These are publicly-available files that are made available from the National Center for Biotechnology Information (NCBI).

Note that many of the files are in a compressed .gz format.

.fna files are Fasta Nucleic Acid (chromosome or gene sequences)
.faa files are Fasta Amino Acid (protein sequences)

Today we will be most interested in the gene and chromosome files.

2 To get started let's download the main genome FASTA file for Yersinia Pestis CO92

wget
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/009/065/GCA_000009065.1_ASM906v1/GCA_000009065.1_ASM906v1_cds_from_genomic.fna.gz

This command uses the common Unix utility "wget", which will download a file directy to the folder in which you are located. After doing this you should see the .fna.gz file in your folder. You can check this with the "ls" command.

3 Because this file is compressed, we must first uncompress it with the "gunzip" command (another common Unix utility).

gunzip GCA_000009065.1_ASM906v1_cds_from_genomic.fna.gz

After this you should see the exact same file, only without the .gz ending. You can check this with the "ls" command.

4 Now to start analyzing this FASTA file we first want to check on the formatting.
Sometimes genome files can be quite large, so we don't want to open the entire file with a text editor. Instead we can just check the first and last few lines to see what the format looks like. For this we can use the "head" and "tail" Unix commands.

head GCA_000009065.1_ASM906v1_cds_from_genomic.fna

and

tail GCA_000009065.1_ASM906v1_cds_from_genomic.fna

You should see a pretty typical FASTA format. Header lines start with a ">" and provide names and descriptions, and subsequent lines have the actual sequence information (in this case ATGCs since the sequence is DNA).

5 Now that we have confirmed this is a typical FASTA file, we can start analyzing it with the "seqkit" tool.

The home page for seqkit with instructions for use is here: https://github.com/shenwei356/seqkit

You can also get a list of instructions by typing:

seqkit --help

seqkit is a very versatile tools and it has a large number of sub-commands. We will primarily be using the "stats" and "fx2tab" commands, so check out the help menu for those with:

seqkit stats --help

and

seqkit fx2tab --help

6

Now let's get some basic stats about the genes with the "stats" command.

seqkit stats GCA_000009065.1_ASM906v1_cds_from_genomic.fna

How does the total length of the protein-coding genes compare to the total length of the whole genome?
What is the range of gene lengths?

7 Let's look at some stats from individual genes using the "fx2tab" command:

seqkit fx2tab -a -i -g -l -n GCA_000009065.1_ASM906v1_cds_from_genomic.fna | head

Note that we are piping the command to the "head" command here, so that only the first 10 lines are shown. Otherwise thousands of entries would flood our terminal, which is always difficult to interpret (and may cause it to crash).

8 Now let's try to sort the genes based on their length, so that we can find the names of the longest and shortest genes:

seqkit fx2tab -a -i -g -l -n GCA_000009065.1_ASM906v1_cds_from_genomic.fna | sort -r -n -k 2,2 | head

This should return the longest 10 genes. The "sort" command uses several flags.
-r indicates a reverse sort (default is from low to high.
-n indicates a numeric sort (default is alphabetical).
-k denotes the columns to sort by. The "2,2" means we are sorting only by the second column. Note that all whitespace counts as a single tab here, so the second column is the length (it would be the 4th if we exported the results to a file).

We can do the same with "tail" instead of head to find the names of the shortest genes.

9 We can use the same logic as above to find the genes with the highest and lowest %GC content. For this we need to sort by the third column.

seqkit fx2tab -H -a -i -g -l -n GCA_000009065.1_ASM906v1_cds_from_genomic.fna | sort -rn -k 3,3 | head

10 If we want to do something more complex and find the average GC content of the genes, we can pipe the result to "datamash" a nice program for simple math.

seqkit fx2tab -H -a -i -g -l -n GCA_000009065.1_ASM906v1_cds_from_genomic.fna | sort -rn -k 3,3 | datamash mean 5

Datamash needs the operation as the second item in the command (sum, mean, etc) and the column ID as the third item. Note that whitespace is sometimes interpreted strangely by these commands, so here we actually need to use the 5th column (just by looking at the output we would assume %GC would be the third column, and that is how "sort" interprets it, but every tool is different).

11 Now let's say we want to retrieve the actual DNA sequence of the gene with the highest %GC content. We can do this by using a "seqkit fx2tab" command and piping the results to a "grep" command.

seqkit fx2tab -H -a -i -g -l GCA_000009065.1_ASM906v1_cds_from_genomic.fna | head -n 1

Note that we did not use the "-n" flag in the seqkit fx2tab command, since this time we wanted the sequence (before we just wanted the statistics).

12 We can also search the actual gene sequences to find certain motifs with the "locate" subcommand.

Lets see how many genes have a string of 11 G's:
seqkit locate -p "GGGGGGGGGGG" GCA_000009065.1_ASM906v1_cds_from_genomic.fna

Note that some genes come up multiple times, since they have multiple matches (in this case, genes with a very high GC content will come up several times).
To return only one line per gene, we can pipe the result to a "sort" command using the "-u" option, to specify we only want unique matches.

seqkit locate -p "GGGGGGGGGGG" GCA_000009065.1_ASM906v1_cds_from_genomic.fna | sort -u -k 1,1

Note that we still have to use the -k 1,1 flag for sort, since we only want to sort by the first column.
Now let's do the same, but allow for one mismatch:

seqkit locate -m 1 -p "GGGGGGGGGG" GCA_000009065.1_ASM906v1_cds_from_genomic.fna | sort -u -k1,1

Note that we get many more matches when allowing for even just a single mismatch, since that mismatch can occur anywhere on the sequence. |

13 Lastly, let's say we want to examine the actual sequences a bit more. Perhaps we are interested in the start codons used by these genes. We can identify them by using the "subseq" command.

seqkit subseq -r 1:3 GCA_000009065.1_ASM906v1_cds_from_genomic.fna | grep -v ">" | head

Here "-r" tells the command that we want only the first 3 bases in the sequences returned. The command also returns the FASTA headers, so we can remove them by piping the result to "grep -v ">"", since this will only match lines that do not have the FASTA > header character.
If we wanted to get a summary of how often different start codons are used, we can sort and summarize the results with "sort" and then "uniq".

seqkit subseq -r 1:3 GCA_000009065.1_ASM906v1_cds_from_genomic.fna | grep -v ">" | sort | uniq -c

Here we can see that although ATG is the canonical and most frequently used start codon, there are actually a variety of different start codons used throughout the genome.

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited