

Starcode tutorial

Guillaume Filion

November 17, 2015

1 Build instructions

Starcode runs on Mac and Linux as a command line application. You can download the source code of Starcode from Github with the following command on a standard terminal.

```
git clone git@github.com:guillaume/starcode
```

Note that this requires that you already have a Github account and that the computer you are working on has an SSH key registered on Github. If this is not the case, follow the instructions from <https://help.github.com/articles/generating-ssh-keys/>.

This should download a directory named **starcode**. To build Starcode, execute the following.

```
cd starcode
make
```

This should succeed on most Linux systems because **make** is available by default. If this is not the case, you can obtain it by typing **sudo apt-get install make** on Ubuntu. On Mac, you need to install XCode, which may take some time. First, you will need an Apple ID, then you will need to download it from the developer website of Apple <https://developer.apple.com/xcode/downloads/>. Then, you may need to follow the instructions shown on the following link to install the command line version of **make** <http://stackoverflow.com/q/10265742/1248687>.

Calling **make** should create an executable called **starcode**. To check that the building is successful, execute the following command in the same directory.

`./starcode`

If you obtain the output shown below, then everything went fine and you are done with the build. If not, then something went wrong. In this case, you can explain how to reproduce the problem on <https://github.com/guillaume/starcode/issues>.

Usage: `starcode [options]`

general options:

- `-d --dist`: maximum Levenshtein distance (default auto)
- `-t --threads`: number of concurrent threads (default 1)
- `-r --cluster-ratio`: minimum cluster size ratio (default 5)
- `-s --sphere`: sphere clustering (default message passing)
- `-q --quiet`: quiet output (default verbose)
- `-v --version`: display version and exit

input/output options (single file, default)

- `-i --input`: input file (default stdin)
- `-o --output`: output file (default stdout)

input options (paired-end fastq files)

- `-1 --input1`: input file 1
- `-2 --input2`: input file 2

output format options

- `--print-clusters`: outputs cluster compositions
- `--non-redundant`: remove redundant sequences
- `--seq-id`: also print sequence id numbers (1-based)

2 Starcode basics

Starcode is a sequence clustering algorithm. It takes a bunch of sequences, finds which are similar and pools them together. Starcode considers that two sequences are similar if their edit distance (also known as the Levenshtein distance) is low, *i.e.* if they are almost identical, except for a few mismatches, insertions and deletions.

Let us get started with a simple example. At the command line, issue the command below.

```
./starcode test/test_file.txt
```

In principle, you should obtain the following output.

```
running starcode with 1 thread
reading input files
raw format detected
sorting
setting dist to 2
progress: 100.00%
AGGGCTTACAAGTATAGGCC 7
CCTCATTATTTGTCGCAATG 7
GGGAGCCCACAGTAAGCGAA 7
TAGCCTGGTGCGACTGTCAT 7
TGCGCCAAGTACGATTCCG 7
```

By default, Starcode is verbose and gives some information on how the run is performed before it outputs the clusters. In this case, we learn that Starcode used a single CPU, that the input file `test/test_file.txt` is in raw format (*i.e.* plain text), and that the threshold distance for similarity search was set to 2. The output proper starts after the line **progress: 100.00%**. It consists of five lines (one per cluster) where the sequence is the centroid of the cluster (the irepresentative sequence) and the number is its size (the number of sequences in the cluster).

The input file `test/test_file.txt` contains 35 sequences that Starcode arranged in 5 clusters of 7 sequences each.

We can make Starcode quiet (non verbose) with the `-q` option, and we can set the clustering distance to 0 with `-d0`.

```
./starcode -q -d0 test/test_file.txt
```

The output is the following.

```
AGGGCTTACAAGTATAGGCC 6
CCTCATTATTTGTCGCAATG 6
GGGAGCCCACAGTAAGCGAA 6
```

```

TAGCCTGGTGCGACTGTCAT  6
TGCGCCAAGTACGATTTCCG  6
AGGGGTTACAAGTCTAGGCC  1
CCTCATTATTTACCGCAATG  1
GGAAGCCCACAGCAAGCGAA  1
TAACCTGGTGCGACTGTTAT  1
TGCGCCAAGTAAGAATTCCG  1

```

This time we obtain five clusters of size 6 and five clusters of size 1. Each centroid of the clusters of size 1 has 2 mismatches with one of the centroids of the clusters of size 6, which is why they form separate clusters when the distance is less than 2.

As a side note, this example output illustrates that the clusters are sorted first by size and then by alphabetical order of the centroid.

3 Clustering options

Coming soon...

4 Documenting clusters

Starcode also allows you to track the sequences that end up in a given cluster. Here, ‘sequences’ can be considered as either input sequences, or distinct nucleotide sequences. The option `--print-clusters` shows the distinct nucleotide sequences that constitute the cluster. Below is an example.

```

./starcode -q --print-clusters test/test_file.txt
AGGGCTTACAAGTATAGGCC  7 AGGGCTTACAAGTATAGGCC,AGGGGTTACAAGTCTAGGCC
CCTCATTATTTGTCGCAATG  7 CCTCATTATTTACCGCAATG,CCTCATTATTTGTCGCAATG
GGGAGCCCACAGTAAGCGAA  7 GGAAGCCCACAGCAAGCGAA,GGGAGCCCACAGTAAGCGAA
TAGCCTGGTGCGACTGTCAT  7 TAACCTGGTGCGACTGTTAT,TAGCCTGGTGCGACTGTCAT
TGCGCCAAGTACGATTTCCG  7 TGCGCCAAGTAAGAATTCCG,TGCGCCAAGTACGATTTCCG

```

With this type of output, we know that the first cluster consists of two distinct sequences, namely the centroid itself and `AGGGGTTACAAGTCTAGGCC`, which together amount to 7 sequences. This kind of output is useful when you want to convert sequences to their centroid.

As mentioned above, ‘sequences’ can also be viewed as input sequences in which case they represent different entities, even when their nucleotide sequences are the same. These sequences are typically different reads from a sequencing experiment. The option `--seq-id` prints the ID numbers of the input sequences that end up in each cluster. Here is an example.

```
./starcode -q --seq-id test/test_file.txt
AGGGCTTACAAGTATAGGCC 7 1,4,5,13,17,20,22
CCTCATTATTTGTCGCAATG 7 3,14,19,27,30,31,32
GGGAGCCACAGTAAGCGAA 7 6,7,10,12,15,24,25
TAGCCTGGTGGACTGTCAT 7 8,9,16,21,28,33,35
TGCGCCAAGTACGATTCCG 7 2,11,18,23,26,29,34
```

With this type of output, we know that the first cluster consists of input sequences number 1, 4, 5, 13, 17,20 and 22. This kind of input is useful when sequences have different origins and you want to know the composition of a cluster relative to the different sources. Note that the first input sequence is given ID number 1 and not 0.

Note that options `--print-clusters` and `--seq-id` are not mutually exclusive, and they may be used together. These options have only mild effects on the speed and the memory footprint of Starcode.

5 Input formats

Starcode accepts FASTA files, FASTQ files and plain text files. The directory `test` contains some example files. We have already seen what happens on plain text files in the previous examples. For FASTA files, Starcode is run as before. The FASTA headers are simply ignored.

```
./starcode -q test/test_file.fasta
AGGGCTTACAAGTATAGGCC 3
CCTCATTATTTGTCGCAATG 1
TGCGCCAAGTACGATTCCG 1
```

FASTQ files present a complication because they can be paired. In the case of a single FASTQ file, Starcode is run as before. The FASTQ headers and the quality are ignored.

```
./starcode -q test/test_file1.fastq
AGGGCTTACAAGTATAGGCC 3
CCTCATTATTTGTCGCAATG 1
TGCGCCAAGTACGATTTCG 1
```

When there are two paired FASTQ files, you must tell Starcode to expect two input files. The first way is to use the `-1` flag for the first file and the `-2` for the second. For more clarity, you can use the longer form `--input1` and `input2` instead. Below is an example.

```
./starcode -q -1 test/test_file1.fastq -2 test/test_file2.fastq
AGGGCTTACAAGTATAGGCC/AAGGGCTTACAAGTATAGGC 3
CCTCATTATTTGTCGCAATG/ACCTCATTATTTGTCGCAAT 1
TGCGCCAAGTACGATTTCG/ATGCGCCAAGTACGATTTC 1
```

As you can see, the output also changes. The two paired sequences are lumped next to each other and each centroid now consists of a pair of sequences. It is important to mention that in this mode, the edit distance between two pairs of sequences is the sum of the pairwise edit distances. For instance, if the edit distance is set to 3 with `-d3`, the pair `AAA/TTT` is a neighbor of the pair `CAA/GGT`, but not of the pair `CCC/GGG`.

6 Getting a non redundant sample

By default, Starcode clusters the input sequences and outputs the clusters. In some application, it is interesting to obtain a non redundant sample of the input. In general, it is easy to do so by collecting the centroid sequences, but because Starcode discards extra information from FASTA and FASTQ files, we provide the option `--non-redundant` for cases that the format has to be respected.

```
./starcode -q --non-redundant test/test_file1.fastq
@seq1/1
AGGGCTTACAAGTATAGGCC
+
BBBBBBBBBBBBBBBBBBBB
@seq3/1
CCTCATTATTTGTCGCAATG
```

```
+  
BBBBBBBBBBBBBBBBBBBB  
@seq2/1  
TGCGCCAAGTACGATTTCG  
+  
BBBBBBBBBBBBBBBBBBBB
```

When two FASTQ files are used with this option, nothing is printed on the screen. Instead, two new FASTQ files are created. Note that the sequences are still sorted in alphabetical order, so their order will usually not be the same as the original input.