Minia-GATB — Short manual

R. Chikhi & G. Rizk & E. Drezen & D. Lavenier
rayan.chikhi@ens-cachan.org

April 18, 2018

Abstract

Minia is a software for ultra-low memory DNA sequence assembly. It takes as input a set of short genomic sequences (typically, data produced by the Illumina DNA sequencer). Its output is a set of contigs (assembled sequences), forming an approximation of the expected genome. Minia is based on a succinct representation of the de Bruijn graph. The computational resources required to run Minia are significantly lower than that of other assemblers.

Contents

1 Forewords

Minia 3 is a completely new version that no longer uses Bloom filters. In terms of goals, not much has changed: Minia remains a contigs assembler using very low memory. Notable changes since Minia 1/2 are:

- Faster k-mer counting step
- Different graph construction step (uses BCALM software to construct unitigs)
- Different graph representation (not a Bloom filter, but an indexed list of unitigs)
- SPAdes-style graph simplifications

2 Installation

See Github page.

3 Parameters

The basic usage is:

```
./minia -in [input file] -kmer-size [kmer size] -out [prefix]
```

Not much has changed since Minia 2 but we recommend that you still read the next section for an updated explanation of parameters.

An example command line is:

```
./minia -in reads.fastq -kmer-size 31 -out minia_assembly_k31
```

The main parameters are:

1. `in` – the input file(s) (see Section 5 for inputting multiple files)
2. `kmer-size` – k-mer length (integer)
3. `prefix` – any prefix string to store output contigs as well as temporary files for this assembly

4 Explanation of parameters

`kmer-size` The k-mer length is the length of the nodes in the de Bruijn graph. It strongly depends on the input dataset. For proper assembly, we recommend that you use the Minia-pipeline that runs Minia multiple times, with an iterative multi-k algorithm. That way, you won't need to choose k. If you insist on running with a single k value, the KmerGenie software can automatically find the best k for your dataset.

`abundance-min` The `abundance-min` is a hard cut-off to remove likely erroneous, low-abundance k-mers. In Minia 3, it is recommended to set it to 2 unless you have a good reason not to. Minia 3 also implements other ways to remove erroneous k-mers, using a more adequate relative cut-off, in the graph simplifications step.

Setting *abundance-min* to 1 is not recommended, as too many erroneous k-mer will be kept, which will likely result in a very large memory usage. If the dataset has high coverage, it may be worth it to try larger values.

`prefix` The `prefix` parameter is any arbitrary output files name, for example, `test_assembly`.

These are the main two parameters that control the quality of assembly. Minia command line has a few more parameters but they are either for advanced usage, or just minor tweaks. Some are described in the rest of this manual.

5 Input

Larger k-mer lengths

Minia supports arbitrary large k-mer lengths. To compile Minia from the source, to support k-mer lengths up to, say, 320, type this in the build folder:

```
rm -Rf CMake*
cmake -DKSIZE_LIST="32   64   96  128  160  192  224  256  320" ..
make -j 4
```

The list of kmers should only contain multiples of 32. Intermediate values are used to create optimized code for k values that are shorter than the max. The last k value specifies the highest kmer size that can effectively be used in this compiled version of Minia. Apart from that, whatever k values are in this list do not affect the assembly quality in any way.

FASTA/FASTQ

Minia assembles any type of Illumina reads, given in the FASTA or FASTQ format. Giving paired or mate-pairs reads as input is OK, but keep in mind that Minia won't use pairing information.

Multipe Files

Minia can assemble multiple input files. Just create a text file containing the list of read files, one file name per line, and pass this list as the first parameter of Minia (instead of a FASTA/FASTQ file). Therefore the parameter `input_file` can be either (i) the read file itself (FASTA/FASTQ/compressed), or (ii) a file containing a list of file names.

Line format

In FASTA files, each read can be split into multiple lines, whereas in FASTQ, each read sequence must be in a single line.

Gzip compression

Minia can direclty read files compressed with gzip. Compressed files should end with '.gz'. Input files of different types can be mixed (i.e. gzipped or not, in FASTA or FASTQ)

6 Output

6.1 Fasta

The main output of Minia is a set of contigs in the FASTA format, in the file `[prefix].contigs.fa`.

6.2 Assembly graph

The contigs FASTA file can be converted to GFA using BCALM's convertToGFA.py script.

Unitigs

Minia will also output unitigs, in the FASTA format. They correspond to non-branching paths in the de Bruijn graph prior to any graph simplification. File: `[prefix].unitigs.fa`.

7 Selection of graph simplification steps

By default, Minia performs 3 types of graph simplifications: tip removal, bulges removal, and erroneous connections removal. Those are similar to SPAdes. You

can selectively ask to not perform any (or all) of those operation through those command line arguments: `-no-tip-removal`, `-no-bulge-removal`, `-no-ec-removal`.

8 Fine-tuning of graph simplification steps

Most of the graph simplification steps are highly inspired by the SPAdes assembler.

A regular user needs not modify any of the graph simplifications parameters. Yet, for advanced users, it is possible to do so in order to increase or reduce the aggresiveness of variant/error removal.

Tip removal implements two modes: "short tips" removal and "longer tips" removal. Short tips are removed no matter what their coverage is. By default, a tip is considered short if it is of length $\leq 3.5k$. Longer tips are of length $\leq 10k$. They are removed if their coverage is smaller than the average coverage of the neighbor unitigs, multiplied by a constant factor.

To make tip clipping more conservative, I would recommend specifying a high `-tip-rctc-cutoff` cutoff, such as `-tip-rctc-cutoff 20`. This will make sure that the coverage of a tip is at least 20 times smaller than the coverage of its neighbors.

Explanation all the other simplifications steps would make this manual a bit longer. For now you can try to guess using slide 11 of `http://cristal.univ-lille.fr/~chikhi/pdf/2017-august-3-biata.pdf`. Some information can also be found in the SPAdes paper, where the term *bulge* is defined.

9 Memory usage

We estimate that the memory usage of Minia is in the order of 1 GB of RAM per gigabases in the target genome to assemble. It is independent of the coverage of the input dataset, provided that the `abundance-min` parameter is not too low.

10 Disk usage

Minia writes large temporary files during the k-mer counting phase. These files are written in the working directory when you launched Minia. For better performance, run Minia on a local hard drive, SSD, or (very large) ram disk, around 2-3x time the space of the input.

11 Reproducibility

Default Minia parameters do not ensure a deterministic execution: due to e.g. multi-threading, the results may change a little bit between executions (but should still give more or less the same assembly). To make sure that results are perfectly identical between runs on the same data and the same parameters, run with these options: `-nb-cores 1 -nb-glue-partitions 200`. Also make sure that the same compiler version was used.

12 Influences

This version of Minia could not exist without inspiration taken from these great pieces of software and methods:

- khmer `https://github.com/dib-lab/khmer`
- KMC2 `https://github.com/marekkokot/KMC`
- SPAdes `http://bioinf.spbau.ru/spades`
- MSP `http://www.vldb.org/pvldb/vol6/p169-li.pdf`
- Meraculous `https://jgi.doe.gov/data-and-tools/meraculous/`