Minia-GATB — Short manual

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

February 1, 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-GATB is the codename of a new version of Minia using the GATB library. We will drop the "-GATB" suffix and just refer to it as "Minia". In terms of features, not much has changed: Minia remains a contigs assembler using very low memory. Notable changes are:

- Different command line format

- Much faster *k*-mer counting step
- Different graph construction step (uses BCALM software to construct unitigs)
- Different graph representation (no longer a Bloom filter, but an indexed list of unitigs)
- SPAdes-style graph simplifications

2 Installation

Downlad the binary and run Minia using the comand `./minia`.

To install Minia from the sources, just type:

```
mkdir build && cd build && cmake .. && make -j 4
```

Minia has been tested on Linux and MacOS systems.

3 Parameters

The basic usage is:

```
./minia -in [input file] -kmer-size [kmer size] \
        -abundance-min [abundance value] -out [prefix]
```

An example command line is:

```
./minia -in reads.fastq -kmer-size 31 -abundance-min 3 -out minia_assembly_k31_m3
```

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. `abundance-min` – specifies how many times a *k*-mer must be seen in the reads to be considered correct (integer)
4. `prefix` – any prefix string to store output contigs as well as temporary files for this assembly

Minia now uses the Cascading Bloom filters improvement (http://arxiv.org/abs/1302.7278) by default, thanks to Gustavo Sacomoto for the implementation in Minia. Launch Minia with the `-debloom original` option to revert to the original data structure.

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. We recommend that you use the KmerGenie software to automatically find the best k for your dataset.

`abundance-min` The `abundance-min` is used to remove erroneous, low-abundance k-mers. This parameter also strongly depends on the dataset. Any k-mer that appears strictly less than *abundance-min* times in the reads will be discarded. A typical value is 3. It is also called "coverage cut-off" in other software (e.g. Velvet).

KmerGenie has an experimental feature (in the output of the command, not in the HTML report) that reports an optimal coverage cut-off (synonymous to abundance) parameter that exactly corresponds to the abundance-min parameter of Minia.

Setting *abundance-min* to 1 **is not recommended**, as no erroneous k-mer will be discarded, which will likely result in a very large memory usage. In particular, if KmerGenie tells you to set abundance to 1, then still, set it to 2. If the dataset has high coverage, try larger values.

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

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)

Graph input

Minia can take as input a graph in the HDF5 format, constructed using the dbgh5 program, or a different GATB tool, or from a previous run of Minia (useful when one wants to tweak assembly parameters). Use the `-graph` parameter to specify the graph, instead of the `-in` parameter.

6 Output

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

Creating unitigs

Minia supports the creation of unitigs. Use the following command line:

```
-starter simple -no-length-cutoff -traversal unitig
```

The `-starter simple` option is to turn off Minia's heuristics for starting node selection (which avoid starting inside a bubble). Note that there is a known bug in the `-starter simple` behavior: some of the unitigs of length k won't be returned (technically: the branching nodes which are neighbors of branching nodes are not returned).

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 some constant times the average coverage of the neighbor unitigs.

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.

An explanation of the other simplifications steps is coming later. For now you can try to guess using slide 11 of `http://cristal.univ-lille.fr/~chikhi/pdf/2017-august-3-biata.pdf`.

9 Memory usage

We estimate that the memory usage of Minia is roughly 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 correctly set. For example, a human genome was assembled in 5.7 GB of RAM. This was using the original data structure; the current implementation relies on Cascading Bloom filters and uses ≈ 4 GB. A better estimation of the memory usage can be found in the Minia article.

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.

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`.

12 Influences

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

- khmer `https://github.com/dib-lab/khmer`
- KMC2 `https://github.com/marekkokot/KMC`
- SPAdes `http://bioinf.spbau.ru/spades`