Hapsembler version 2.12 (+ Encore & Scarpa) Manual

Nilgun Donmez
Department of Computer Science
University of Toronto

June 15, 2013

Contents

1 Introduction

We are proud to release the second major version of Hapsembler. Hapsembler version 2.12 features automated setting of most parameters and added options such as estimation of library statistics. The new Hapsembler package also includes the scaffolding tool Scarpa, which can be used as a stand-alone scaffolder.

If you are only interested in scaffolding or error correction, Scarpa and the read correction module Encore are also released separately. See http://compbio.cs.toronto.edu/hapsembler for details.

2 Installation

To install Hapsembler download the package available at http://compbio.cs.toronto.edu/hapsembler and run:

```
tar -xzvf hapsembler-2.12.tar.gz
```

Once the files are extracted, change your directory to hapsembler-2.12 and type:

```
make
```

This will create the binaries and place them under hapsembler-2.12/bin. In order to access the individual programs from any location, you should add this directory to your system path.

Please note that Hapsembler is mainly tested in 64-bit Linux-based environments. You may install Hapsembler on a 32-bit environment however this is not recommended due to memory limitations. Please see the Frequently Asked Questions (section 11) for platforms other than Linux.

3 Requirements

The Hapsembler package requires the following to install and run successfully. Listed in parenthesis are the versions used to test the current Hapsembler package. These utilities must be accessible via the system path:

- make (GNU make 3.81)
- g++ (GNU gcc 4.4.6)
- bash (GNU bash 3.2.25)
- perl (v5.8.8)

You will need OpenMP to run Hapsembler in multi-threaded mode. You can get the list of compilers supporting OpenMP at http://openmp.org/wp/. Note that you can control the number of threads used by Hapsembler at runtime so there is no downside to compiling Hapsembler with this option. If you are unable to obtain a compiler that supports OpenMP you can still compile the binaries for single-threaded use with the following directive:

```
make NO_OMP=1
```

In addition, you will need an efficient read mapping software (e.g. Bowtie http://bowtie-bio.sourceforge.net/index.shtml , BWA http://bio-bwa.sourceforge.net/ , etc) in order to use the scaffolding module Scarpa. You may use any mapping software available for your sequencing platform provided that it can report its output in SAM format.

Note that Hapsembler comes bundled with the lp_solve library (version 5.5.2) for 64-bit Linux. If you have a 32-bit system, you will have to download the appropriate files from http://sourceforge.net/projects/lpsolve/ and change the makefile accordingly in order to compile Scarpa. Nonetheless, we strongly recommend that you compile and run all Hapsembler tools in a 64-bit architecture.

If you do not intend to perform scaffolding, use the directive:

```
make custom NO_LP=1
```

This will allow you to compile the rest of the binaries without having to obtain the appropriate lp_solve library for your system.

Make sure you have sufficient memory and disk space available to Hapsembler before running. A minimum of 16GB memory is recommended. Typically, Hapsembler will require around 3-4 times the memory that is required to store the reads. Keep in mind that actual requirements will vary with sequence coverage depth and repeat structure of the genome.

4 Changes from the previous release

In addition to changes to the code base, Hapsembler 2.12 has undergone various user interface changes. Some parameters are now automatically set for convenience and more robust results. The default values of some of the parameters have also changed. Although we include a(n incomplete) list of major changes from the previous releases below, due to the high volume of modification and additions, we kindly request that our old users read this documentation in full.

4.1 From Hapsembler-1.1:

- `correctr` has been replaced by a new tool named `encore`.
- `hapsemble` wrapper script now only runs `overlappr`, `hapsemblr` and `consensr`.
- The advanced parameters `--KMER_SIZE`, `--MIN_OVERLAP` and `--COVERAGE_CUTOFF` are now set automatically. Other advanced parameters are integrated into dividual tools.
- `--diploid` and `--sdwidth` arguments are deprecated. Instead, a new option named `--calibrate` is added to `hapsemblr`. If toggled on, this option lets `hapsemblr` re-estimate the mean and standard deviation of each paired-end library.

5 Data preperation

Hapsembler requires all the reads - even if they are generated from different libraries - to be in a single fastq formatted file. If you have paired read libraries, you will also need to create a library info file where each line obeys the following format:

```
<start> <end> <mean insert size> <standard deviation> <flag>
```

For instance, the following library info file:

```
1 24000000 300 30 1
24000001 34000000 700 70 1
```

indicates that the data consists of 2 libraries (possibly followed by some single-end reads). The first library includes the first 24m reads in the fastq file and has a mean insert size and standard deviation of 300bp and 30bp respectively. The second library contains the next 10m reads and has a mean and standard deviation of 700bp and 70bp respectively. The last column (i.e 1) is a flag reserved for future use. The rest of the reads (if any) are taken to be single reads.

Note that paired reads must come BEFORE any single reads in the input file and each pair must be consecutive in the file. The paired reads should be sampled from opposite strands and face towards each other:

```
Genome:

5' ------------------------------------------------------------> 3'
3' <------------------------------------------------------------ 5'

       Read_A:  5' |-----> 3'
                                              3' <-----| 5'  : Read_B
```

If your reads are in an orientation different from above, you have to reverse complement them. For your convenience, the utility tool `preprocr` included in this package has an option to reverse complement either of the reads. See section 7.1 for further information.

5.1 A note about Illumina mate-pairs

According to some recent studies, mate-pair production reads may have a high percentage of chimeric reads. For this reason, we do not recommend using mate-pairs during the contig assembly stage. Instead, we suggest that these reads are included during the scaffolding phase as described in section 9.

6 Hapsembler Pipeline

The Hapsembler pipeline is distributed among several executables under 3 main stages: 1) error correction 2) genome assembly and 3) scaffolding. The following sections describe each stage in detail and how to use the individual tools. If you forget the options available to any program in the Hapsembler package, simply run it with option `--help` and a help message will be displayed.

7 Error Correction

This stage is performed to boost the quality of your sequencing reads by quality trimming and/or correction of sequencing errors. If you have reasons to believe you have high quality reads (e.g. simulated error-free reads) you may skip this stage and proceed with assembly (see section 8).

7.1 preprocr

`preprocr` is a utility program that helps prepare your data for further steps in the assembly. We recommend using this tool if you have long (>150bp) Illumina reads or Roche/454 reads in order to quality trim the reads. If you have separate files for different libraries, you can run this tool separately on each file. If you have paired reads in two complementary files you can also merge them using `preprocr` (see options below). Note that `preprocr` does not trim for vector sequences. If your data contain any barcodes or vector sequences, please remove these before using `preprocr`.

SYNOPSIS

```
./preprocr -p <platform> -f <file> -o <file>
```

OPTIONS

```
--platform|-p [illumina|fourfivefour]
    Define the type of the platform the reads are produced from
       (required)

--fastq|-f fastq_filename
    Fastq formatted input file (required)

--fastq2|-x fastq_filename
    The second reads in fastq format for paired libraries. The
       order of the reads should match the order of the reads in
        the file given with --fastq option.

--output|-o output_filename
    Output filename. Default is standard output.

--revcomp|-n [1|2|3]
    Reverse complement the first (1), the second (2) or both (3)
        reads in a read pair. If this option is set all reads
       are taken to be paired.

--phred|-d N
    Set the phred offset for the quality values to N. Default
       value is 33 for fourfivefour and 64 for illumina.

--threshold|-s E
    Set the threshold for trimming. E must be a real number
       between 0.0 (no trimming) and 0.5 (most trimming).
```

```
Default values are 0.05 for illumina and 0.1 for
fourfivefour.
```

7.2 encore

Once you prepare your reads, you can run the read error correction program `encore`. You can use this program even if you do not intend to perform *de novo* assembly on your reads, for instance as preprocessing for SNP detection, reference-guided assembly, etc.

SYNOPSIS

```
./encore -p <platform> -f <file> -o <file> -g <genome>
```

OPTIONS

```
--platform|-p [illumina|fourfivefour]
    Define the type of the platform the reads are produced from
      (required)

--fastq|-f fastq_filename
    Fastq formatted input file (required)

--output|-o output_filename
    Output filename for corrected reads (required)

--genome|-g V
    Estimated genome size in KILO base pairs. (required)

--nthreads|-t K
    Use K number of threads (ignored if program is compiled
      without OpenMP)

--onestrand|-a [yes|no]
    If set to yes, the reads are treated as single stranded.
      Default value is no.

--epsilon|-e X (real number)
    Set the expected discrepancy (mismatches+indels) rate. X
      must be a real number between 0.01 and 0.09. Default
      values for illumina and fourfivefour are 0.04 and 0.06
      respectively.

--phred|-d N
    Set the phred offset for the quality values to N. Default
      value is 33 for fourfivefour and 64 for illumina.
```

7.3 Performance

Suppose you have n reads totalling to M base pairs, t threads are used and let $k = 14$ if the genome size to be assembled is less than 270mbp, and $k = 15$ otherwise and F is

the size of the input fastq file in bytes. Then the approximate memory requirement for `encore` is:

$$F + 14(M/k) + 10(2^{2k}) + n(8t + 60) + C \quad (1)$$

where C is a small (usually less than 2GB) overhead. The memory requirement for `preprocr` is constant and its running time is linear.

We note that the actual requirements may vary slightly. The speed-up gained from multiple-threads is almost linear in the number of threads used as long as the machine is not overcrowded. Thus, we strongly recommend enabling multi-threading if you have sufficient memory available.

8 Genome Assembly

The main pipeline of contig assembly consists of the following programs: `overlappr`, `hapsemblr` and `consensr`. These can either be run individually, or through the wrapper shell script `hapsemble`. We first give details of the individual programs and then explain how to use the wrapper script.

8.1 overlappr

`overlappr` is a tool that computes the overlaps between the reads, which form the basis of the overlap graph. It takes the reads in fastq format and outputs two files containing the reads and overlaps. These two files are required by `hapsemblr` (discussed below).

SYNOPSIS

```
./overlappr -p <platform> -f <file> -o <prefix> -g <genome>
```

OPTIONS

```
--platform|-p [illumina|fourfivefour]
    Define the type of the platform the reads are produced from
       (required)

--fastq|-f fastq_filename
    Fastq formatted input file (required)

--output|-o prefix
    Prefix for output files

--genome|-g V
    Estimated genome size in KILO base pairs. (required)

--nthreads|-t K
    Use K number of threads (ignored if program is compiled
       without OpenMP)

--onestrand|-a [yes|no]
```

```
    If set to yes, the reads are treated as single stranded.
       Default value is no.

--epsilon|-e X (real number)
    Set the expected discrepancy (mismatches+indels) rate. X
       must be a real number between 0.01 and 0.09. Default
       values for illumina and fourfivefour are 0.04 and 0.06
       respectively.
```

8.2 hapsemblr

`hapsemblr` is the main assembly program implementing the overlap graph based assembly. It takes the two files produced by `overlappr` as input and outputs a contig information file required by `consensr`, which is then converted to a fasta file containing the contigs.

SYNOPSIS

```
hapsemblr -r <file> -c <file> -g <genome size>
```

OPTIONS

```
--prefix|-r reads_filename
    Prefix of the files produced by overlappr (required)

--contigs|-c contigs_filename
    Output filename for contigs (required)

--genome|-g genome_size
    Estimated genome size in KILO base pairs (required)

--library|-l library_filename
    File containing information about libraries. If unset, all
       reads are taken to be single production.

--onestrand|-a [yes|no]
    If set to yes, performs strand specific assembly. This
       option can not be turned on if -l option is set. Default
       value is no.

--calibrate|-b [yes|no]
    If set to yes, re-calculates the mean and standard deviation
        of each library. The default is yes.
```

8.3 consensr

`consensr` computes a consensus alignment for each contig produced by `hapsemblr` and writes the results into a single fasta formatted file. As input, it takes a fastq file containing the reads and a file containing the contigs as output by `hapsemblr`. Note that the fastq file given as input to `consensr` should be identical to the one given as input to `overlappr`.

SYNOPSIS

```
./consensr -p <platform> -f <file> -c <file> -o <file>
```

OPTIONS

```
--platform|-p [illumina|fourfivefour]
    Define the type of the platform the reads are produced from
       (required)

--fastq|-f fastq_filename
    Fastq formatted input file (required)

--contigs|-c contigs_filename
    Contig file produced by hapsemblr (required)

--output|-o output_filename
    Output file for contigs (required)

--min-size|-m N (integer)
    Set the minimum size of a contig to be reported to N (in bp)
       . Default value is 200.

--phred|-d N
    Set the phred offset for the quality values to N. Default
       value is 33 for fourfivefour and 64 for illumina.
```

8.4 hapsemble

`hapsemble` is a wrapper script that executes the programs in the core Hapsembler pipeline in a logical order. Note that this script will not work unless the full path of the Hapsembler bin directory is added to your system path. The synopsis of the script and the options are described below.

SYNOPSIS

```
./hapsemble -p <platform> -f <input file> -g <genome size>
```

OPTIONS

```
--platform|-p [illumina|fourfivefour]
    Defines the type of the platform the reads are produced from
        (required)

--fastq|-f fastq_filename
    Fastq formatted input file (required)

--genome|-g genome_size
    Estimated genome size in KILO base pairs. (required)

--output|-o output_filename
```

```
    Output filename. Default is 'output.fasta'.

--library|-l library_filename
    File containing information about libraries. If unset, all
       reads are taken to be single production.

--nthreads|-t N (integer)
    Use N number of threads (ignored if compiled without OpenMP)

--onestrand|-a [yes|no]
    If set to yes, the reads are treated as single stranded.
       Default is no.

--calibrate|-b [yes|no]
    If set to yes, re-calculates the mean and standard deviation
        of each library. The default is yes.

--epsilon|-e E (real number)
    Set the expected (mismatches+indels) rate. E must be a real
       number between 0.01 and 0.09. Default values for illumina
        and fourfivefour are 0.04 and 0.06 respectively.

--phred|-d N (integer)
    Set the phred offset for the quality values to N. Default
       values for illumina and fourfivefour are 64 and 33
       respectively.

--min-size|-m N (integer)
    Set the minimum size of a contig to be reported to N (in bp)
       . Default value is 200.

--stage-start|-y [--stage-end|-z] N (integer)
    Set the starting/ending stage to N, where N is an integer
       between 1 and 3. Defaults are 1 and 3 respectively.
       Stages are as follows:
        1 -> overlappr
        2 -> hapsemblr
        3 -> consensr
```

You can use the `stage-start` and `stage-end` options to re-evaluate one or more stages due to previous failures or due to option changes. For instance, if you would like to re-run `hapsemblr` but this time without library calibration, you can change the corresponding option and set the `stage-start` to 2 and only the last two steps of the pipeline will be executed.

8.5 Performance

Suppose you have n reads totalling to M base pairs, t threads are used and let $k = 14$ if the genome size to be assembled is less than 270mbp, and $k = 15$ otherwise. Then the

approximate memory requirements for `overlappr` and `consensr` are:

$$M + 14(M/k) + 10(2^{2k}) + n(8t + 52) + C \quad (2)$$

$$2M + 37n + C \quad (3)$$

where C is a small (usually less than 2GB) overhead. Note that, the actual memory used by these programs might vary slightly in different environments. The memory requirement of `hapsemblr` is harder to estimate since it depends on factors that can not be immediately measured from the raw input. Typically, `hapsemblr` will take longer and require more memory when reads are paired. If your machine does not have enough memory for paired assembly, you may still be able to perform unpaired assembly by omitting the `--library` argument.

9 Scaffolding

Although `hapsemblr` uses paired reads to resolve repeats and polymorphism, it does not perform "gapped" scaffolding: if there is a coverage gap between a read pair, this pair is not utilized. To leverage such pairs and to make better use of long insert libraries, from version 2.12 onwards, Hapsembler is bundled with the scaffolding tool Scarpa.

Below we give information on how to use Scarpa. Since Scarpa is designed as a stand-alone tool, the discussion in this section is valid on the output of any assembler unless stated otherwise.

9.1 scarpa_process

As a first step, the reads and contigs have to be prepared using the perl script `scarpa_process`:

SYNOPSIS

```
scarpa_process -c <contigs> -f <reads> -i <insert size>
```

OPTIONS

```
<contigs> -> is a fasta file containing the contigs.
<reads> -> is a fastq file containing each read pair in a
    consecutive order.
<insert size> -> is the insert size of the library in base pairs
    .
```

Above, the orientation of the reads are expected to be forward-reverse. If you have more than one library, you can use the -f and -i options repeatedly. For example, if you have two libraries in files lib1.fq and lib2.fq with insert sizes 300bp and 600bp, respectively:

```
scarpa_process -c contigs.fasta -f lib1.fq -i 300 -f lib2.fq -i 600
```

Note that you should always provide the libraries in order of increasing insert size. This command will produce the following files:

```
contigs.fasta.scarpa.fa
contigs.fasta.scarpa.info
lib1.fq.scarpa.fq
lib2.fq.scarpa.fq
```

You should map the file(s) *.scarpa.fq to contigs.fasta.scarpa.fa using a read mapper and get the results in a single SAM formatted file. The mapping should be performed in single-end mode. The *.scarpa.fq files are not needed after the SAM file is created, you may delete them subsequently to free disk space.

Make sure to tell your mapper to report the header section; the lines starting with "@SQ" are required by scarpa_parser. In addition, if possible, use a mapper that can report the optional "NM:i" tag, which is used for edit distance.

9.2 scarpa_parser

Before running Scarpa, you need to process the mappings using the script `scarpa_parser` using the following command:

```
cat <sam> | scarpa_parser [-e <max # of mismatches>] > <map>
```

The argument -e is optional and by default set to 0. "map" file will be needed in the next step. Note that this script takes the SAM file from standard input, so if you are using a mapper that writes the SAM file to standard output you can pipe its output directly to scarpa_parser. This can save significant time and disk space for large datasets.

9.3 scarpa

In this last step, simply run the following command:

```
scarpa -c <contigs>.scarpa.fa -i <map file> -l <contigs>.scarpa.info
    -o <output>
```

Above "output" is the name for the output file and it will contain the scaffolds in fasta format. The other arguments are the files you obtained by running `scarpa_process` and `scarpa_parser`. In addition to the mandatory arguments above, the `scarpa` executable takes several advanced options as listed below:

```
SYNOPSIS
    ./scarpa -c <file> -l <file> -i <file> -o <file>

OPTIONS
    --contigs|-c contigs_filename
        Fasta formatted file containing the contigs (required)

    --library|-l libraries_filename
        File containing the paired read library information (
           required)
```

```
--mappings|-i mappings_filename
    File containing the read mappings (required)

--output|-o output_filename
    Output file for scaffolds (required)

--calibrate|-b [yes|no]
    If set to yes, re-calculates the mean and standard deviation
        of each library. Default is yes.

--min_support N
    Sets the minimum number of mate links to connect two contigs
        to N. Default value is 2.

--max_removal N
    Sets the maximum number of contigs that can be removed
       during the orientation step to N. Default value is 6.

--min_contig N
    Sets the minimum contig size (in bp) to be used in
       scaffolding to N. Default value is 100bp.
```

9.4 Performance

The perl scripts `scarpa_process` and `scarpa_parser` use constant memory and run in linear time. The memory requirement of `scarpa` is difficult to estimate from the raw input, though it grows roughly linearly with the number of contigs. The running time will largely depend on the quality of the reads and the contigs, as well as the complexity of the genome. If it is taking more time than tolerable for your project, you may try increasing the `min_support` value.

If you are running `scarpa` with contigs generated by a third-party assembler, please be aware that some NGS assemblers tend to report a large number of very short contigs (<75bp). Such contigs can not be reliably scaffolded, yet they increase the running time significantly. You can discard small contigs by setting the `--min_contig` option as described above, however, we highly recommend removing these contigs before scaffolding. This will not only speed up the mapping process but also decrease the number of multi-mapped reads.

10 Testing Hapsembler

The Hapsembler package contains a small dataset and a demo script that can be used to test the scripts and binaries included in the package. Instructions on how to run the demo is given in the `SAMPLE.README` file that is located under hapsembler-2.12/sample.

11 Frequently Asked Questions

11.1 What is Hapsembler?

Hapsembler[1] is a *de novo* genome assembler built for highly polymorphic organisms for use with capillary, Roche/454 and Illumina reads. For examples of highly polymorphic organisms see [5] or [6].

11.2 What is polymorphism?

In genomics, polymorphism refers to any type of genetic variation that is present between the individuals of a species. Examples of genomic polymorphism include Single Nucleotide Polymorphisms (SNPs), Copy Number Variations (CNVs) and Structural Variations (SVs).

11.3 Why is assembling a highly polymorphic genome difficult?

For multiploid organisms, polymorphism is also present between the haplotypes of a single individual. On one hand, a high degree of polymorphism means that reads from different haplotypes can not be assembled together. On the other hand, since the level of polymorphism is also highly variable across the genome, some regions of the genome are inevitably homozygous and reads from these regions are difficult to separate. Such regions will be collapsed while others are separated, thus creating tangles in the assembly. Moreover, sequencing errors may be mistaken for SNPs and small indels further complicating the assembly.

11.4 Can Hapsembler assemble haploid genomes?

Hapsembler can assemble genomes with low/no polymorphism (e.g. bacteria) however it is designed for highly polymorphic genomes and therefore may not be the best choice for all organisms. If you have a large genome (>1gbp) with low polymorphism (e.g. mouse, human) and high coverage of short reads, we suggest that you use assemblers optimized for such data (e.g. [4]).

11.5 Can Hapsembler assemble human genomes?

See question 11.4.

11.6 Can Hapsembler be used for metagenomics?

Although metagenomics and polymorphic genome assembly share certain aspects, we believe they have distinct goals and call for different strategies. Hapsembler is not tested on any metagenomics data and some of the methods are not suitable for highly variable coverage. For such projects, we recommend tools tailored for metagenomics (e.g. [3]).

11.7 Can I use color space reads with Hapsembler?

Unfortunately, color space reads (i.e. ABI/SOLiD platform) are not supported by Hapsembler.

11.8 Does Hapsembler perform scaffolding?

Yes! The Hapsembler package now includes a stand-alone scaffolding tool called Scarpa. See section 9 for details. If you are only interested in scaffolding, you can also download Scarpa from `http://compbio.cs.toronto.edu/hapsembler/scarpa.html` instead of installing the entire Hapsembler package.

11.9 Can I use Hapsembler just to do error correction?

Yes! See section 7 for details. If you only intend to do error correction on your reads, you can also download the error correction module Encore from `http://compbio.cs.toronto.edu/hapsembler/encore.html` instead of installing the entire Hapsembler package.

11.10 Can I run Hapsembler on platforms other than Linux?

It is possible to install and run Hapsembler on Windows XP/Vista via Cygwin or on MAC OSX if the appropriate compiler and utility packages are present. However, these environments will not be supported.

11.11 How do I cite Hapsembler?

Please cite Hapsembler and Encore as: [1]. Scarpa is currently in print, however you can cite the advanced online publication as: [2].

11.12 I found a bug, what do I do?

Send any suggestions or bug reports to: nild@cs.toronto.edu. Please try to give as much detail as possible to trace the bug: the computing environment, sample data (if possible), parameters used, etc.

References

[1] N. Donmez and M. Brudno, *Hapsembler: An assembler for highly polymorphic genomes*, in RECOMB, V. Bafna and C. S. Sahinalp, eds., vol. 6577 of LNBI, Springer-Verlag Berlin Heidelberg, 2011, pp. 38–52.

[2] N. Donmez and M. Brudno, *Scarpa: Scaffolding reads with practical algorithms*, Bioinformatics, doi:10.1093/bioinformatics/bts716 (2012).

[3] J. Laserson, V. Jojic, and D. Koller, *Genovo: De novo assembly for metagenomes*, in RECOMB, B. Berger, ed., vol. 6044 of Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2010, pp. 341–356.

[4] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and I. Birol, *ABySS: A parallel assembler for short read sequence data*, Genome Research, 19 (2009), pp. 1117–1123.

[5] K. S. Small, M. Brudno, M. M. Hill, and A. Sidow, *Extreme genomic variation in a natural population*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 5698–5703.

[6] E. Sodergren, *The genome of the sea urchin Strongylocentrotus purpuratus*, Science, 314 (2006), pp. 941–952.