hifiasm Documentation

Release 0.16.0

Haoyu Cheng, Heng Li

Aug 18, 2023

CONTENTS

1 HiFi-only Assembly **1**
- 1.1 Produce two partially phased assemblies . . . 1
- 1.2 Produce primary/alternate assemblies . . . 2

2 Trio-binning Assembly **3**

3 Hi-C Integrated Assembly **5**

4 Hifiasm Output **7**
- 4.1 Output files . . . 7
- 4.2 Output file formats . . . 8
- 4.3 Hifiasm log interpretation . . . 8

5 Hifiasm FAQ **9**
- 5.1 How do I get contigs in FASTA? . . . 10
- 5.2 Which types of assemblies should I use? . . . 10
- 5.3 Are inbred/homozygous genomes supported? . . . 10
- 5.4 Are diploid genomes supported? . . . 10
- 5.5 Are polyploid genomes supported? . . . 10
- 5.6 Why one Hi-C integrated assembly is larger than another one? . . . 10
- 5.7 For Hi-C integrated assembly, why the assembly size of both haplotypes are much larger than the estimated genome size? . . . 11
- 5.8 How can I tweak parameters to improve Hi-C integrated assembly? . . . 11
- 5.9 Why the size of primary assembly or partially phased assembly is much larger than the estimated genome size? . . . 11
- 5.10 Why the hamming error rate or the swith error rate of trio-binning assembly is very high? . . . 12
- 5.11 Why does hifiasm stuck or crash? . . . 12
- 5.12 What's the usage of different bin files in hifiasm? . . . 12
- 5.13 Can I generate HiFi-only assembly first, and then add Hi-C or trio data later? . . . 12
- 5.14 What is the minimum read coverage required for hifiasm? . . . 12
- 5.15 Why the primary assembly is more contiguous than the fully-phased assemblies and the partially phased assemblies (i.e. `*.hap*.p_ctg.gfa`)? . . . 13
- 5.16 My assembly is fragmented or not contiguous enough, how do I improve it? . . . 13
- 5.17 How do I avoid misassemblies? . . . 13

6 Hifiasm Parameter Reference **15**
- 6.1 Synopsis . . . 15
- 6.2 General options . . . 15
- 6.3 Error correction options . . . 16
- 6.4 Assembly options . . . 17
- 6.5 Trio-binning options . . . 18

6.6 Purge duplication options 18
6.7 Hi-C integration options 19

7 Publications **21**

8 Install **23**

9 Assembly Concepts **25**

10 Why Hifiasm? **27**

11 Learn **29**

CHAPTER

ONE

HIFI-ONLY ASSEMBLY

A typical hifiasm command line looks like:

```
hifiasm -o NA12878.asm -t 32 NA12878.fq.gz
```

where `NA12878.fq.gz` provides the input reads, `-t` sets the number of CPUs in use and `-o` specifies the prefix of output files. Input sequences should be FASTA or FASTQ format, uncompressed or compressed with gzip (.gz). The quality scores of reads in FASTQ are ignored by hifiasm. Hifiasm outputs assemblies in GFA format.

At the first run, hifiasm saves corrected reads and overlaps to disk as `NA12878.asm.*.bin`. It reuses the saved results to avoid the time-consuming all-vs-all overlap calculation next time. You may specify `-i` to ignore precomputed overlaps and redo overlapping from raw reads. You can also dump error corrected reads in FASTA and read overlaps in PAF with:

```
hifiasm -o NA12878.asm -t 32 --write-paf --write-ec /dev/null
```

Hifiasm purges haplotig duplications by default. For inbred or homozygous genomes, you may disable purging with option `-l0`. Old HiFi reads may contain short adapter sequences at the ends of reads. You can specify `-z20` to trim both ends of reads by 20bp. For small genomes, use `-f0` to disable the initial bloom filter which takes 16GB memory at the beginning. For genomes much larger than human, applying `-f38` or even `-f39` is preferred to save memory on k-mer counting.

1.1 Produce two partially phased assemblies

Since v0.15, hifiasm produces two sets of partially phased contigs in default like:

```
hifiasm -o NA12878.asm -t 32 NA12878.fq.gz
```

In this example, the partially phased contigs are written to `NA12878.asm.bp.hap*.p_ctg.gfa`. This pair of files can be thought to represent the two haplotypes in a diploid genome, though with occasional switch errors. The frequency of switches is determined by the heterozygosity of the input sample. Hifiasm also writes the primary contigs to `NA12878.asm.bp.p_ctg.gfa`.

For samples with high heterozygosity rate, a common issue is that one set of partially phased contigs is much larger than another set. To fix this issue, please set smaller value for `-s` (default: 0.55). Another possibility is that hifiasm misidentifies coverage threshold for homozygous reads. In this case, please set `--hom-cov` to homozygous coverage. See *Why the size of primary assembly or partially phased assembly is much larger than the estimated genome size?* for more details.

1.2 Produce primary/alternate assemblies

To get primary/alternate assemblies, the option `--primary` should be set:

```
hifiasm -o NA12878.asm --primary -t 32 NA12878.fq.gz
```

The primary contigs and the alternate contigs are written to `NA12878.asm.p_ctg.gfa` and `NA12878.asm.a_ctg.gfa`, respectively. For inbred or homozygous genomes, the primary/alternate assemblies can be also produced by `-l0`. Similarly, turning `-s` or `--hom-cov` should be helpful if the primary assembly is too large. See *Why the size of primary assembly or partially phased assembly is much larger than the estimated genome size?* for more details.

CHAPTER

TWO

TRIO-BINNING ASSEMBLY

When parental short reads are available, hifiasm can also generate a pair of haplotype-resolved assemblies with trio binning. To perform such assembly, you need to count k-mers first with yak and then do assembly:

```
yak count -k31 -b37 -t16 -o pat.yak paternal.fq.gz
yak count -k31 -b37 -t16 -o mat.yak maternal.fq.gz
hifiasm -o NA12878.asm -t 32 -1 pat.yak -2 mat.yak NA12878.fq.gz
```

Here `NA12878.asm.hap1.p_ctg.gfa` and `NA12878.asm.hap2.p_ctg.gfa` give the assemblies for two haplotypes. In the binning mode, hifiasm does not purge haplotig duplicates by default. Because hifiasm reuses saved overlaps, you can generate both primary/alternate assemblies and trio binning assemblies with:

```
hifiasm -o NA12878.asm --primary -t 32 NA12878.fq.gz 2> NA12878.asm.pri.log
hifiasm -o NA12878.asm -t 32 -1 pat.yak -2 mat.yak /dev/null 2> NA12878.asm.trio.log
```

The second command line will run much faster than the first. The phasing switch error rate and hamming error rate are able to be evaluated quickly by yak:

```
yak trioeval -t16 pat.yak mat.yak assembly.fa
```

The W-line and H-line reported by `yak trioeval` indicate switch error rate and hamming error rate respectively:

```
W       26714   3029448 0.008818
H       24315   3029885 0.008025
```

For this example, the switch error rate is 0.8818% and the hamming error rate is 0.8025%. If the hamming error rate or the swith error rate of trio-binning assembly is very high, it might be caused by hifiasm or the incorrect parental data. To fix it, see *Why the hamming error rate or the swith error rate of trio-binning assembly is very high?* for more details.

CHAPTER

THREE

HI-C INTEGRATED ASSEMBLY

Hifiasm can generate a pair of haplotype-resolved assemblies with paired-end Hi-C reads:

```
hifiasm -o NA12878.asm -t32 --h1 read1.fq.gz --h2 read2.fq.gz HiFi-reads.fq.gz
```

In this mode, each contig is supposed to be a haplotig, which by definition comes from one parental haplotype only. Hifiasm often puts all contigs from the same parental chromosome in one assembly. It has cleanly separated chrX and chrY for a human male dataset. Nonetheless, phasing across centromeres is challenging. Hifiasm is often able to phase entire chromosomes but it may fail in rare cases. Also, contigs from different parental chromosomes are randomly mixed as it is just not possible to phase across chromosomes with Hi-C. Hifiasm does not perform scaffolding for now. You need to run a standalone scaffolder such as SALSA or 3D-DNA to scaffold phased haplotigs.

For samples with high heterozygosity rate, a common issue is that one assembly is much larger than another one. To fix this issue, please set smaller value for `-s` (default: 0.55). Another possibility is that hifiasm misidentifies coverage threshold for homozygous reads. In this case, please set `--hom-cov` to homozygous coverage peak. See *How can I tweak parameters to improve Hi-C integrated assembly?* for more details.

At the first run, hifiasm saves the alignment of Hi-C reads to disk as `*hic*.bin`. It reuses the saved results to avoid Hi-C alignment next time. Please note that `*hic*.bin` should be deleted when tuning any parameters affecting `*p_utg*gfa`. Since v0.15.5, hifiasm can detect such changes and renew Hi-C bin files automatically. There are several parameters which do not change `*p_utg*gfa`, including `-s`, `--seed`, `--n-weight`, `--n-perturb`, `--f-perturb` and `--l-msjoin`.

CHAPTER

FOUR

HIFIASM OUTPUT

4.1 Output files

In general, hifiasm generates the following assembly graphs in the GFA format:

- `prefix`.r_utg.gfa: haplotype-resolved raw unitig graph. This graph keeps all haplotype information.
- `prefix`.p_utg.gfa: haplotype-resolved processed unitig graph without small bubbles. Small bubbles might be caused by somatic mutations or noise in data, which are not the real haplotype information. Hifiasm automatically pops such small bubbles based on coverage. The option --hom-cov affects the result. See *homozygous coverage setting* for more details. In addition, the option -p forcedly pops bubbles.
- `prefix`.p_ctg.gfa: assembly graph of primary contigs. This graph includes a complete assembly with long stretches of phased blocks.
- `prefix`.a_ctg.gfa: assembly graph of alternate contigs. This graph consists of all contigs that are discarded in primary contig graph.
- `prefix`.*hap*.p_ctg.gfa: phased contig graph. This graph keeps the phased contigs.

Hifiasm outputs *.r_utg.gfa and *.p_utg.gfa in any cases. Specifically, hifiasm outputs the following assembly graphs in trio-binning mode:

- `prefix`.dip.hap1.p_ctg.gfa: fully phased paternal/haplotype1 contig graph keeping the phased paternal/haplotype1 assembly.
- `prefix`.dip.hap2.p_ctg.gfa: fully phased maternal/haplotype2 contig graph keeping the phased maternal/haplotype2 assembly.

With Hi-C partition options, hifiasm outputs:

- `prefix`.hic.p_ctg.gfa: assembly graph of primary contigs.
- `prefix`.hic.hap1.p_ctg.gfa: fully phased contig graph of haplotype1 where each contig is fully phased.
- `prefix`.hic.hap2.p_ctg.gfa: fully phased contig graph of haplotype2 where each contig is fully phased.
- `prefix`.hic.a_ctg.gfa (optional with --primary): assembly graph of alternate contigs.

Hifiasm generates the following assembly graphs only with HiFi reads in default:

- `prefix`.bp.p_ctg.gfa: assembly graph of primary contigs.
- `prefix`.bp.hap1.p_ctg.gfa: partially phased contig graph of haplotype1.
- `prefix`.bp.hap2.p_ctg.gfa: partially phased contig graph of haplotype2.

If the option --primary or -l0 is specified, hifiasm outputs:

- `prefix`.p_ctg.gfa: assembly graph of primary contigs.

- `prefix`.a_ctg.gfa: assembly graph of alternate contigs.

For each graph, hifiasm also outputs a simplified version (*noseq*gfa) without sequences for the ease of visualization. The coordinates of low quality regions are written to *lowQ.bed in BED format. The concepts of different types of assemblies can be found here.

4.2 Output file formats

Hifiasm broadly follows the specification for GFA 1.0. There are several fields that are specifically used by hifiasm. For S segment line:

- rd:i:: read coverage. It is calculated by the reads coming from the same contig/unitig.

Hifiasm outputs A lines including the information of reads which are used to construct contig/unitig. Each A line is plain-text, tab-separated, and the columns appear in the following order:

Col	Type	Description
1	string	Should be always A
2	string	Contig/unitig name
3	int	Contig/unitig start coordinate of subregion constructed by read
4	char	Read strand: "+" or "-"
5	string	Read name
6	int	Read start coordinate of subregion which is used to construct contig/unitig
7	int	Read end coordinate of subregion which is used to construct contig/unitig
8	id:i:int	Read ID
9	HG:A:char	Haplotype status of read. HG:A:a, HG:A:p, HG:A:m indicate read is non-binnable, father/hap1-specific and mother/hap2-specific, respectively.

4.3 Hifiasm log interpretation

Hifiasm prints several information for quick debugging, including:

- k-mer plot: showing how many k-mers appear a certain number of times. For homozygous samples, there should be one peak around read coverage. For heterozygous samples, there should two peaks, where the smaller peak is around the heterozygous read coverage and the larger peak is around the homozygous read coverage. For example, issue10 indicates the heterozygous read coverage and the homozygous read coverage are 28 and 57, respectively. Issue49 is another good example. Weird k-mer plot like issue93 is often caused by insufficient coverage or presence of contaminants.
- homozygous coverage: coverage threshold for homozygous reads. Hifiasm prints it as: [M::purge_dups] homozygous read coverage threshold: X. If it is not around homozygous coverage, the final assembly might be either too large or too small. To fix this issue, please set --hom-cov to homozygous coverage.
- number of het/hom bases: how many bases in unitig graph are heterozygous and homozygous during Hi-C phased assembly. Hifiasm prints it as: [M::stat] # heterozygous bases: X; # homozygous bases: Y. Given a heterozygous sample, if there are much more homozygous bases than heterozygous bases, hifiasm fails to identify correct coverage threshold for homozygous reads. In this case, please set --hom-cov to homozygous coverage.

CHAPTER

FIVE

HIFIASM FAQ

- *How do I get contigs in FASTA?*
- *Which types of assemblies should I use?*
- *Are inbred/homozygous genomes supported?*
- *Are diploid genomes supported?*
- *Are polyploid genomes supported?*
- *Why one Hi-C integrated assembly is larger than another one?*
- *For Hi-C integrated assembly, why the assembly size of both haplotypes are much larger than the estimated genome size?*
- *How can I tweak parameters to improve Hi-C integrated assembly?*
- *Why the size of primary assembly or partially phased assembly is much larger than the estimated genome size?*
- *Why the hamming error rate or the swith error rate of trio-binning assembly is very high?*
- *Why does hifiasm stuck or crash?*
- *What's the usage of different bin files in hifiasm?*
- *Can I generate HiFi-only assembly first, and then add Hi-C or trio data later?*
- *What is the minimum read coverage required for hifiasm?*
- *Why the primary assembly is more contiguous than the fully-phased assemblies and the partially phased assemblies (i.e. `*.hap*.p_ctg.gfa`)?*
- *My assembly is fragmented or not contiguous enough, how do I improve it?*
- *How do I avoid misassemblies?*

5.1 How do I get contigs in FASTA?

The FASTA file can be produced from GFA as follows:

```
awk '/^S/{print ">"$2;print $3}' test.p_ctg.gfa > test.p_ctg.fa
```

5.2 Which types of assemblies should I use?

If parental data is available, `*dip.hap*.p_ctg.gfa` produced in trio-binning mode should be always preferred. Otherwise if Hi-C data is available, `*hic.hap*.p_ctg.gfa` produced in Hi-C mode is the best choice. Both trio-binning mode and Hi-C mode generate fully-phased assemblies.

If you only have HiFi reads, hifiasm in default outputs `*bp.hap*.p_ctg.gfa`. The primary/alternate assemblies can be also produced by using `--primary`. All these HiFi-only assemblies are not fully-phased. See blog here for more details.

5.3 Are inbred/homozygous genomes supported?

Yes, please use the `-l0` option to disable purge duplication step.

5.4 Are diploid genomes supported?

Yes, most modules of hifiasm are designed for diploid samples, including purge duplication step, partially phased assembly and fully-phased assembly with trio-binning or Hi-C.

5.5 Are polyploid genomes supported?

The `*r_utg.gfa` and `*p_utg.gfa` are lossless so that they also work for polyploid genomes. However, currently the contig-generation modules of hifiasm are designed for diploid samples, which means both the partially phased assembly and the fully-phased assembly does not directly support polyploid genomes. If it is set to >2, the quality of primary assembly for polyploid genomes might be improved. Please use primary assembly for polyploid samples and run multiple rounds of purging steps using third-party tools such as purge_dups.

5.6 Why one Hi-C integrated assembly is larger than another one?

For some samples like human male, the paternal haplotype should be larger than the maternal haplotype. However, if one assembly is much larger than another one, it should be the issues of hifiasm. To fix it, please set smaller value for `-s` (default: 0.55).

Another possibility is that hifiasm misidentifies coverage threshold for homozygous reads. For instance, hifiasm prints the following information during assembly:

```
[M::purge_dups] homozygous read coverage threshold: 36
```

In this example, hifiasm identifies the coverage threshold for homozygous reads as 36. If it is significantly smaller than the homozygous coverage peak, hifiasm will generate two unbalanced assemblies. In this case, please set `--hom-cov` to homozygous coverage peak. Please note that tuning `--hom-cov` may affect `*p_utg*gfa` so that `*hic*.bin` should be deleted. Since v0.15.5, hifiasm can detect such changes and renew Hi-C bin files automatically.

5.7 For Hi-C integrated assembly, why the assembly size of both haplotypes are much larger than the estimated genome size?

It is likely that hifiasm misidentifies coverage threshold for homozygous reads. Hifiasm prints the following information for debugging:

```
[M::stat] # heterozygous bases: 645155110; # homozygous bases: 1495396634
```

If most bases of a diploid sample are homozygous, the coverage threshold is wrongly determined by hifiasm. For instance, hifiasm prints the following information during assembly:

```
[M::purge_dups] homozygous read coverage threshold: 36
```

In this example, hifiasm identifies the coverage threshold for homozygous reads as 36. If it is much smaller than homozygous coverage peak, hifiasm thinks most reads are homozygous and assign them to both assemblies, making both of them much larger than the estimated haploid genome size. In this case, please set `--hom-cov` to homozygous coverage peak. Please note that tuning `--hom-cov` may affect `*p_utg*gfa` so that `*hic*.bin` should be deleted. Since v0.15.5, hifiasm can detect such changes and renew Hi-C bin files automatically.

5.8 How can I tweak parameters to improve Hi-C integrated assembly?

Compared with the HiFi-only assembly or the trio-binning assembly, the Hi-C integrated assembly is a little bit more complex so that you need to take care of the results. See *Why one Hi-C integrated assembly is larger than another one?* and *For Hi-C integrated assembly, why the assembly size of both haplotypes are much larger than the estimated genome size?* for details on how to fix potential issues.

There are several other options that may affect the Hi-C integrated assembly. Increasing the values of `--n-weight`, `--n-perturb` and `--f-perturb` may improve phasing results but takes longer time. However, tuning `--l-msjoin` is tricky. All these options do not affect `*p_utg*gfa` so that `*hic*.bin` can be reused.

5.9 Why the size of primary assembly or partially phased assembly is much larger than the estimated genome size?

It could be because the estimated genome size is incorrect. Another possibility is that hifiasm does not perform enough purging. Setting smaller value for `-s` (default: 0.55) or turning `--hom-cov` should be helpful. See *Hifiasm log interpretation* for more details.

5.10 Why the hamming error rate or the swith error rate of trio-binning assembly is very high?

In rare cases, a potential issue is that a few contigs may misjoin two haplotypes. For example, half of a contig come from mother while another half come from father. Such misjoined contigs can be fixed by manually breaking. The coordinates of problematic regions can be found by A-lines in GFA file or `yak trioeval -e` (see issue 37 for more details). However, if there are many misjoined contigs or the switch/hamming error rate reported by `yak trioeval` is very high, users should check if the parental data is correct (see issue 130 for more details).

Another possibility is that there are some unitigs in unitig graph misjoining two haplotypes. Such problematic unitigs might be ignored by the graph-binning strategy. Set smaller value for `--t-occ` forcedly remove unitig including unexpected haplotype-specific reads.

5.11 Why does hifiasm stuck or crash?

In most cases, it is caused by the low quality HiFi reads. A good HiFi dataset should have a k-mer plot like issue10 or issue49. In contrast, low quality HiFi data often lead to weird k-mer plot like issue93. Such weird k-mer plots usually indicate insufficient coverage or presence of contaminants. See *Hifiasm log interpretation* for more details. If the HiFi data look fine, please raise an issue at the issue page.

5.12 What's the usage of different bin files in hifiasm?

`*ec.bin`, `*ovlp.reverse.bin` and `*ovlp.source.bin` save the results of error correction step. `*hic*bin` saves the results of Hi-C alignment. Please note that `*hic*.bin` should be deleted when tuning any parameters affecting `*p_utg*gfa`. There are several parameters which does not change `*p_utg*gfa`, including `-s`, `--seed`, `--n-weight`, `--n-perturb`, `--f-perturb` and `--l-msjoin`. Since v0.15.5, hifiasm can detect such changes and renew Hi-C bin files automatically.

5.13 Can I generate HiFi-only assembly first, and then add Hi-C or trio data later?

Yes, the HiFi-only assembly, Hi-C phased assembly and trio-binning assembly share the same `*ec.bin`, `*ovlp.reverse.bin` and `*ovlp.source.bin`.

5.14 What is the minimum read coverage required for hifiasm?

Usually >=13x HiFi reads per haplotype. Higher coverage might be able to improve the contiguity of assembly.

5.15 Why the primary assembly is more contiguous than the fully-phased assemblies and the partially phased assemblies (i.e. `*.hap*.p_ctg.gfa`)?

For diploid samples, primary assembly usually has greater N50 but at the expense of highly fragmented alternate assembly. From the method view, the primary assembly has an extra joining step, which joins two haplotypes to make primary assembly more contiguous.

When producing fully-phased assemblies and partially phased assemblies, hifiasm is designed to keep both haplotypes contiguous. It is important for many downstream applications like SV calling.

5.16 My assembly is fragmented or not contiguous enough, how do I improve it?

Raising `-D` or `-N` may improve the resolution of repetitive regions but takes longer time. These two options affect all types of assemblies and usually do not have a negative impact on the assembly quality. In contrast, `--purge-max` only affects primary assembly. Setting larger value for `--purge-max` makes primary assembly more contiguous but may collapse repeats or segmental duplications.

If the assembly is too fragmented, users should check if HiFi data is good enough. See *Why does hifiasm stuck or crash?* for details.

5.17 How do I avoid misassemblies?

Set smaller value for `--purge-max`, `-s` and `-O`, or use the `-u` option.

CHAPTER

SIX

HIFIASM PARAMETER REFERENCE

6.1 Synopsis

Assembly only with HiFi reads:

```
hifiasm -o [prefix] -t [nThreads] [options] input1.fq [input2.fq [...]]
```

Trio binning assembly with yak dumps:

```
yak count -o paternal.yak -b37 [-t nThreads] [-k kmerLen] paternal.fq.gz
yak count -o maternal.yak -b37 [-t nThreads] [-k kmerLen] maternal.fq.gz
hifiasm [-o prefix] [-t nThreads] [options] -1 paternal.yak -2 maternal.yak child.hifi.
↪fq.gz
```

Hi-C integrated assembly:

```
hifiasm -o [prefix] -t [nThreads] --h1 [hic_r1.fq.gz,...] --h2 [hic_r2.fq.gz,...]␣
↪[options] HiFi.read.fq.gz
```

To get detailed description of options, run:

```
hifiasm -h
```

or:

```
man ./hifiasm.1
```

6.2 General options

-o <FILE=hifiasm.asm>
Prefix of output files. See *Output files* and *Output file formats* for more details.

-t <INT=1>
Number of CPU threads used by hifiasm.

-h
Show help information.

--version
Show version number.

6.3 Error correction options

-k <INT=51>
K-mer length. This option must be less than 64.

-w <INT=51>
Minimizer window size.

-f <INT=37>
Number of bits for bloom filter; 0 to disable. This bloom filter is used to filter out singleton k-mers when counting all k-mers. It takes $2^{(INT-3)}$ bytes of memory. A proper setting saves memory. `-f37` is recommended for human assembly. For small genomes, use `-f0` to disable the initial bloom filter which takes 16GB memory at the beginning. For genomes much larger than human, applying `-f38` or even `-f39` is preferred to save memory on k-mer counting.

-D <FLOAT=5.0>
Drop k-mers occurring >`FLOAT*coverage` times. Hifiasm discards these high-frequency k-mers during error correction to reduce running time. The `coverage` is determined automatically by hifiasm based on k-mer plot, representing homozygous read coverage. Raising this option may improve the resolution of repetitive regions but takes longer time.

-N <INT=100>
Consider up to `max(-D*coverage,-N)` overlaps for each oriented read. The `coverage` is determined automatically by hifiasm based on k-mer plot, representing homozygous read coverage. Raising this option may improve the resolution of repetitive regions but takes longer time.

-r <INT=3>
Rounds of haplotype-aware error correction. This option affects all outputs of hifiasm. Odd rounds of correction are preferred in practice.

-z <INT=0>
Length of adapters that should be removed. This option remove `INT` bases from both ends of each read. Some old HiFi reads may consist of short adapters (e.g. 20bp adapter at one end). For such data, trimming short adapters would significantly improve the assembly quality.

--max-kocc <INT=2000>
Employ k-mers occurring < `INT` times to rescue repetitive overlaps. This option may improve the resolution of repeats.

--hg-size <INT(k/m/g)>
Estimated haploid genome size used for inferring read coverage. This option is used to get accurate homozygous read coverage during error correction. Common suffices are required, for example, 100m or 3g.

--min-hist-cnt <INT=5>
When analyzing the k-mer spectrum, ignore counts below `INT`. For very low coverage of HiFi data, set smaller value for this option. See issue 45 for example.

6.4 Assembly options

-a <INT=4>
Rounds of assembly graph cleaning. This option is used with `-x` and `-y`. Note that unlike -r, this option does not affect error corrected reads and all-to-all overlaps.

-m <INT=10000000>
Maximal probing distance for bubble popping when generating primary/alternate contig graphs. Bubbles longer than `INT` bases will not be popped.

-p <INT=0>
Maximal probing distance for bubble popping when generating haplotype-resolved processed unitig graph without small bubbles. Bubbles longer than `INT` bases will not be popped. Small bubbles might be caused by somatic mutations or noise in data. Please note that hifiasm automatically pops small bubbles based on coverage, which can be tweaked by `--hom-cov`.

-n <INT=3>
A unitig is considered small if it is composed of less than `INT` reads. Hifiasm may try to remove small unitigs at various steps.

-x <FLOAT1=0.8>, -y <FLOAT2=0.2>
Max and min overlap drop ratio. This option is used with `-a`. Given a node N in the assembly graph, let max(N) be the length of the longest overlap of N. Hifiasm iteratively drops overlaps of N if their length/max(N) is below a threshold controlled by `-x` and `-y`. Hifiasm applies `-a` rounds of short overlap removal with an increasing threshold between `FLOAT1` and `FLOAT2`.

-i
Ignore all bin files so that hifiasm will start again from scratch.

-u
Disable post-join step for contigs which may improve N50. The post-join step of hifiasm improves contig N50 but may introduce misassemblies.

--hom-cov <INT>
Homozygous read coverage inferred automatically in default. This option affects different types of outputs, including Hi-C phased assembly and HiFi-only assembly. For more details, see *How can I tweak parameters to improve Hi-C integrated assembly?*, *Why the size of primary assembly or partially phased assembly is much larger than the estimated genome size?* and *Hifiasm log interpretation*.

--pri-range <INT1[,INT2]>
Min and max coverage cutoffs of primary contigs. Keep contigs with coverage in this range at p_ctg.gfa. Inferred automatically in default. If `INT2` is not specified, it is set to infinity. Set -1 to disable.

--lowQ <INT=70>
Output contig regions with >=`INT`% inconsistency to the bed file with suffix lowQ.bed. Set 0 to disable.

--b-cov <INT=0>
Break contigs at potential misassemblies with <`INT`-fold coverage. Work with `--m-rate`. Set 0 to disable.

--h-cov <INT=-1>
Break contigs at potential misassemblies with >`INT`-fold coverage. Work with `--m-rate`. Set -1 to disable.

--m-rate <FLOAT=0.75>
Break contigs with <=`FLOAT`*`coverage` exact overlaps. Only work when `--b-cov` and `--h-cov` are specified.

--primary
Output a primary assembly and an alternate assembly. Enable this option or `-l0` outputs a primary assembly and an alternate assembly.

6.5 Trio-binning options

-1 <FILE>
K-mer dump generated by yak count from the paternal/haplotype1 reads.

-2 <FILE>
K-mer dump generated by yak count from the maternal/haplotype2 reads.

-3 <FILE>
List of paternal/haplotype1 read names.

-4 <FILE>
List of maternal/haplotype2 read names.

-c <INT1=2>, -d <INT2=5>
Lower bound and upper bound of the binned k-mer's frequency. When doing trio binning, a k-mer is said to be differentiating if it occurs `>= INT2` times in one sample but occurs `< INT1` times in the other sample.

--t-occ <INT=60>
Forcedly remove unitig including `>INT` unexpected haplotype-specific reads without considering graph topology. For more details, see *Why the hamming error rate or the swith error rate of trio-binning assembly is very high?*.

6.6 Purge duplication options

-l <INT=3>
Level of purge duplication. 0 to disable, 1 to only purge contained haplotigs, 2 to purge all types of haplotigs, 3 to purge all types of haplotigs in the most aggressive way. In default, 3 for non-trio assembly, 0 for trio-binning assembly. For trio-binning assembly, only level 0 and level 1 are allowed.

-s <FLOAT=0.55>
Similarity threshold for duplicate haplotigs that should be purged. In default, 0.75 for `-l1`/`-l2`, 0.55 for `-l3`. This option affects both HiFi-only assembly and Hi-C phased assembly. For more details, see *How can I tweak parameters to improve Hi-C integrated assembly?* and *Why the size of primary assembly or partially phased assembly is much larger than the estimated genome size?*.

-O <INT=1>
Min number of overlapped reads for duplicate haplotigs that should be purged.

--purge-max <INT>
Coverage upper bound of purge duplication, which is inferred automatically in default. If the coverage of a contig is higher than this bound, don't apply purge duplication. Larger value makes assembly more contiguous but may collapse repeats or segmental duplications.

--n-hap <INT=2>
Assumption of haplotype number. If it is set to >2, the quality of primary assembly for polyploid genomes might be improved.

6.7 Hi-C integration options

--h1 <FILEs>
File names of input Hi-C R1 `[r1_1.fq,r1_2.fq,...]`.

--h2 <FILEs>
File names of input Hi-C R2 `[r2_1.fq,r2_2.fq,...]`.

--n-weight <INT=3>
Rounds of reweighting Hi-C links. Raising this option may improve phasing results but takes longer time.

--n-perturb <INT=10000>
Rounds of perturbation. Increasing this option may improve phasing results but takes longer time.

--f-perturb <FLOAT=0.1>
Fraction to flip for perturbation. Increasing this option may improve phasing results but takes longer time.

--seed <INT=11>
RNG seed.

--l-msjoin <INT=500000>
Detect misjoined unitigs of >=`INT` in size; 0 to disable.

Hifiasm is a fast haplotype-resolved de novo assembler for PacBio HiFi reads. It can assemble a human genome in several hours and assemble a ~30Gb California redwood genome in a few days. Hifiasm emits partially phased assemblies of quality competitive with the best assemblers. Given parental short reads or Hi-C data, it produces arguably the best haplotype-resolved assemblies so far.

CHAPTER

SEVEN

PUBLICATIONS

Hifiasm

Haoyu Cheng, Gregory T. Concepcion, Xiaowen Feng, Haowen Zhang & Heng Li. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. Nature Methods. (2021).

CHAPTER

EIGHT

INSTALL

The easiest way to get started is to download a release. Please report any issues on github issues page.

In addition, the latest unreleased version can be found from github:

```
git clone https://github.com/chhylp123/hifiasm
cd hifiasm && make
```

Another way is to install hifiasm via bioconda:

```
conda install -c bioconda hifiasm
```

CHAPTER

NINE

ASSEMBLY CONCEPTS

There are different types of assemblies which are commonly used in practice (see details). Hifiasm produces primary/alternate assemblies or partially phased assemblies only with HiFi reads. Given Hi-C data or trio-binning data, hifiasm produces contiguous fully-phased assemblies, i.e. haplotype-resolved assemblies.

CHAPTER

TEN

WHY HIFIASM?

- Hifiasm delivers high-quality assemblies. It tends to generate longer contigs and resolve more segmental duplications than other assemblers.
- Given Hi-C reads or short reads from the parents, hifiasm can produce overall the best haplotype-resolved assembly so far. It is the assembler of choice by the Human Pangenome Project for the first batch of samples.
- Hifiasm can purge duplications between haplotigs without relying on third-party tools such as purge_dups. Hifiasm does not need polishing tools like pilon or racon, either. This simplifies the assembly pipeline and saves running time.
- Hifiasm is fast. It can assemble a human genome in half a day and assemble a ~30Gb redwood genome in three days. No genome is too large for hifiasm.
- Hifiasm is trivial to install and easy to use. It does not required Python, R or C++11 compilers, and can be compiled into a single executable. The default setting works well with a variety of genomes.

CHAPTER

ELEVEN

LEARN

- *HiFi-only Assembly* - Assembling HiFi reads without additional data types
- *Trio-binning Assembly* - Producing fully phased assemblies with HiFi and trio-binning data
- *Hi-C Integrated Assembly* - Producing fully phased assemblies with HiFi and Hi-C data
- *Hifiasm Output* - Interpreting results
- *Hifiasm FAQ* - Frequently asked questions
- *Hifiasm Parameters* - Parameter reference of hifiasm