# LightAssembler: fast and memory-efficient assembly algorithm for high-throughput sequencing reads

**Sara El-Metwally[1,*], Magdi Zakaria[1], and Taher Hamza[1]**

[1] Computer Science Department, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt.

*** Corresponding author**

Sara El-Metwally

Computer Science Department, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt, sarah_almetwally4@mans.edu.eg.

# Appendix A

## A.1 Simulation

We simulated ten different data sets with 101-bp single-end reads from *E. coli* reference genome (NC_000913.2) using Mason [194] v0.1.2 . These data sets are varying in average coverage (25x, 35x, 75x, 140x and 280x) and average error rate (1% and 3%). We will discuss our simulation steps in the following sections.

### A.1.1 Read Simulator

**Name:** Mason
**Version:** 0.1.2
**Availability**: http://packages.seqan.de/mason/
**Setup Steps:**

```
download mason from: http://packages.seqan.de/mason/
tar xvjf mason-0.1.2-Linux-x86_64.tar.bz2
cd mason-0.1.2-Linux-x86_64
cd bin
```

### A.1.2 Reference Genome for Simulated Data Sets

**Reference Genome:** Escherichia coli str. K-12

**NCBI Reference Sequence:** NC_000913.2

### A.1.3 Simulation Method

We simulated five data set varying the coverage from 25x to 280x with average error rate 1% and these errors are abundant toward the 3' end (depicted in the following simulation script):

```
#!/bin/bash
#cov: 25x, 35x,75x, 140x, 280x
#err: 1%
for i in 25 35 75 140 280
do

 let cnt=$i*5000000/101

./mason illumina -i -n 101 -N $cnt -pi 0 -pd 0 -pmm 0.01 -pmmb 0.005 -pmme 0.03 -
sq -hs 0 -hi 0 -o ecoli_reads_${i}_1.fq e_coli_k12.fa

done
```

The second five data sets are simulated with average error rate 3% (depicted in the following simulation script).

```
#!/bin/bash
#cov: 25x, 35x,75x, 140x, 280x
#err: 3%
for i in 25 35 75 140 280
do

  let cnt=$i*5000000/101

./mason illumina -i -n 101 -N $cnt -pi 0 -pd 0 -pmm 0.03 -pmmb 0.015 -pmme 0.09 -
sq -hs 0 -hi 0 -o ecoli_reads_${i}_3.fq e_coli_k12.fa

done
```

## A.2 Estimating Assembly Parameters

To estimate assembly parameters such as *k*-mer size and minimum abundance threshold using **KmerGenie** [190] :

**Name:** KmerGenie
**Version:** 1.6982
**Availability**: http://kmergenie.bx.psu.edu/
**Setup Steps:**

```
download KmerGenie from: http://kmergenie.bx.psu.edu/
tar xvzf kmergenie-1.6982.tar.gz
cd kmergenie-1.6982
make
```

- To run KmerGenie on the simulated datasets:

```
#!/bin/bash
#cov: 25x, 35x,75x, 140x, 280x
#err: 1% 3%
for c in 25 35 75 140 280
  do
    for e in 1 3
      do
         ./kmergenie ecoli_reads_${c}_${e}.fq -o assembly_info_${c}_${e}
     done
  done
```

# A.3 Assembly Programs

## A.3.1 Velvet

**Assembler name:** Velvet
**Version:** 1.2.10
**Availability**: https://www.ebi.ac.uk/~zerbino/velvet/
**Setup Steps:**

```
download Velvet from: https://www.ebi.ac.uk/~zerbino/velvet/
tar xvzf velvet_1.2.10.tgz
cd velvet_1.2.10
make MAXKMERLENGTH=31
```

## A.3.2 ABySS

**Assembler name:** ABySS
**Version:** 1.5.2
**Availability**: http://www.bcgsc.ca/platform/bioinfo/software/abyss
**Setup Steps:**

```
download ABySS
from: http://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.5.2
tar xvzf abyss-1.5.2.tar.gz
cd abyss-1.5.2
./configure && make
cd ABYSS
```

## A.3.3 Minia

**Assembler name:** Minia
**Version:** 2.0.3
**Availability**: http://minia.genouest.org/
**Setup Steps:**

```
download Minia from: http://minia.genouest.org/
tar xvzf minia-2.0.3-Linux.tar.gz
cd minia-2.0.3-Linux/bin
```

### A.3.4 SparseAssembler

**Assembler name:** SparseAssembler
**Availability**: http://sourceforge.net/projects/sparseassembler/
**Setup Steps:**

```
download SparseAssembler from: http://sourceforge.net/projects/sparseassembler/
unzip sparseassembler-code-..zip
cd sparseassembler-code-..
cd BetaVersion
cd SparseAssemblerCode
g++ SparseAssembler.cpp -o SparseAssembler
```

### A.3.5 LightAssembler

**Assembler name:** LightAssembler
**Version:** 1.0.0
**Availability**: https://github.com/SaraEl-Metwally/LightAssembler
**Setup Steps:**

```
download LightAssembler from: https://github.com/SaraEl-Metwally/LightAssembler
unzip LightAssembler-master.zip
cd LightAssembler-master
make k=57
```

### A.3.6 SSPACE

**Stand-alone scaffolding tool:** SSPACE
**Version:** 3.0.0
**Availability:** http://www.baseclear.com/genomics/bioinformatics/basetools/SSPACE
**Setup Steps**:

```
download SSPACE
from: http://www.baseclear.com/genomics/bioinformatics/basetools/SSPACE
tar xvzf SSPACE-STANDARD-3.0_linux-x86_64.tar.gz
cd SSPACE-STANDARD-3.0_linux-x86_64
```

## A.4 Assembly Evaluation Tools

### A.4.1 GAGE Evaluation Script

**Availability:** http://gage.cbcb.umd.edu/results/
**Setup Steps**:

```
download MUMmer3.23
from http://sourceforge.net/projects/mummer/files%2Fmummer%2F3.23/
tar xvzf MUMmer3.23.tar.gz
cd MUMmer3.23
make
download gage-validation from http://gage.cbcb.umd.edu/results/
tar xvzf gage-validation.tar.gz
```

### A.4.2 Assemblathon Evaluation Script

**Availability:** https://github.com/KorfLab/Assemblathon

**Setup Steps**:

```
download Assemblathon from https://github.com/KorfLab/Assemblathon
unzip assemblathon2-analysis-master.zip
cd assemblathon2-analysis-master
```

## A.5 Memory profiling Tool

### A.5.1 memusage.cpp script

**Availability:** https://github.com/SaraEl-Metwally/LightAssembler

**Setup Steps**:

```
download memusage.cpp from https://github.com/SaraEl-Metwally/LightAssembler
g++ memusage.cpp -o memusage
```

## A.6 Accuracy of LightAssembler *k*-mers classification

### A.6.1 ComputeCorrectKmers

**Availability**: https://github.com/SaraEl-Metwally/ComputeCorrectKmers

**Setup Steps:**

```
download ComputeCorrectKmers from:
            https://github.com/SaraEl-Metwally/ComputeCorrectKmers
unzip ComputeCorrectKmers-master.zip
cd ComputeCorrectKmers-master
make k=57
```

## A.7 Real Data sets

We used real sequencing reads from two evaluation studies: GAGE and Assemblathon.

### A.7.1 GAGE Dataset

The datasets for Staphylococcus aureus, Rhodobacter sphaeroides and Human chromosome 14 and their corrected version by ALLPATHS-LG [192] can be downloaded from GAGE website.

GAGE website: http://gage.cbcb.umd.edu/data/

The corrected version of **Staphylococcus aureus** dataset can be downloaded from the following:

http://gage.cbcb.umd.edu/data/Staphylococcus_aureus/

The corrected version of **Rhodobacter sphaeroides** dataset can be downloaded from the following:

http://gage.cbcb.umd.edu/data/Rhodobacter_sphaeroides/

The reference genomes for evaluation purpose can be downloaded from the following:

**Staphylococcus aureus** reference genome:

http://gage.cbcb.umd.edu/data/Staphylococcus_aureus/Data.original/genome.fasta

**Rhodobacter sphaeroides** reference genome:

http://gage.cbcb.umd.edu/data/Rhodobacter_sphaeroides/Data.original/genome.fasta

**Human chromosome 14** reference genome:

http://gage.cbcb.umd.edu/data/Hg_chr14/Data.original/genome.fasta

We removed the leading N's from the Human chromosome 14 reference genome before using it for evaluating assembly results.

### A.7.2 Assemblathon Dataset

The bird dataset (**Melopsittacus undulatus**) from the Assemblathon project has many sequencing libraries; we used the one with accession [EMBL: ERX218680]. The sequencing reads can be downloaded directly from the following website:

https://trace.ddbj.nig.ac.jp/DRASearch/experiment?acc=ERX218680

## A.8 Assembly and Evaluation Instructions

### A.8.1 Assembly and Evaluation Instructions for simulated dataset

We compared LightAssembler results with those from ABySS, Minia and SparseAssembler. We tried different values for $k$-mer size (23, 27, 31, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, and 63) for all assemblers. We also tried different values for minimum abundance threshold for Minia and different gap sizes for SparseAssembler.

#### A.8.1.1 ABySS

- Dataset with $25x$ and error rate 1%

```
#Assembly
./ABYSS -k49 ecoli_reads_25_1.fq -o ecoli_25_1.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_25_1.fasta
```

- Dataset with $25x$ and error rate 3%

```
#Assembly
./ABYSS -k23 ecoli_reads_25_3.fq -o ecoli_25_3.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_25_3.fasta
```

- Dataset with $35x$ and error rate 1%

```
#Assembly
./ABYSS -k51 ecoli_reads_35_1.fq -o ecoli_35_1.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_35_1.fasta
```

- Dataset with $35x$ and error rate 3%

```
#Assembly
./ABYSS -k23 ecoli_reads_35_3.fq -o ecoli_35_3.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_35_3.fasta
```

- Dataset with $75x$ and error rate 1%

```
#Assembly
./ABYSS -k63 ecoli_reads_75_1.fq -o ecoli_75_1.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_1.fasta
```

- Dataset with $75x$ and error rate 3%

```
#Assembly
./ABYSS -k49 ecoli_reads_75_3.fq -o ecoli_75_3.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_3.fasta
```

- Dataset with $140x$ and error rate 1%

```
#Assembly
./ABYSS -k63 ecoli_reads_140_1.fq -o ecoli_140_1.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_140_1.fasta
```

- Dataset with $140x$ and error rate 3%

```
#Assembly
./ABYSS -k57 ecoli_reads_140_3.fq -o ecoli_140_3.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_140_3.fasta
```

- Dataset with $280x$ and error rate 1%

```
#Assembly
./ABYSS -k63 ecoli_reads_280_1.fq -o ecoli_280_1.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_280_1.fasta
```

- Dataset with $280x$ and error rate 3%

```
#Assembly
./ABYSS -k63 ecoli_reads_280_3.fq -o ecoli_280_3.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_280_3.fasta
```

### A.8.1.2 Minia

- Dataset with $25x$ and error rate 1%

```
#Assembly
./minia -in ecoli_reads_25_1.fq -kmer-size 41 -abundance-min 2 -out ecoli_25_1

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_25_1.contigs.fa
```

- Dataset with $25x$ and error rate 3%

```
#Assembly
./minia -in ecoli_reads_25_3.fq -kmer-size 23 -abundance-min 2 -out ecoli_25_3

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_25_3.contigs.fa
```

- Dataset with $35x$ and error rate 1%

```
#Assembly
./minia -in ecoli_reads_35_1.fq -kmer-size 45 -abundance-min 2 -out ecoli_35_1

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_35_1.contigs.fa
```

- Dataset with $35x$ and error rate 3%

```
#Assembly
./minia -in ecoli_reads_35_3.fq -kmer-size 27 -abundance-min 3 -out ecoli_35_3

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_35_3.contigs.fa
```

- Dataset with $75x$ and error rate 1%

```
#Assembly
./minia -in ecoli_reads_75_1.fq -kmer-size 63 -abundance-min 2 -out ecoli_75_1

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_1.contigs.fa
```

- Dataset with $75x$ and error rate 3%

```
#Assembly
./minia -in ecoli_reads_75_3.fq -kmer-size 43 -abundance-min 3 -out ecoli_75_3

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_3.contigs.fa
```

- Dataset with $140x$ and error rate 1%

```
#Assembly
./minia -in ecoli_reads_140_1.fq -kmer-size 63 -abundance-min 4 -out ecoli_140_1

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_140_1.contigs.fa
```

- Dataset with $140x$ and error rate 3%

```
#Assembly
./minia -in ecoli_reads_140_3.fq -kmer-size 57 -abundance-min 3 -out ecoli_140_3

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_140_3.contigs.fa
```

- Dataset with $280x$ and error rate 1%

```
#Assembly
./minia -in ecoli_reads_280_1.fq -kmer-size 63 -abundance-min 12 -out ecoli_280_1

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_280_1.contigs.fa
```

- Dataset with $280x$ and error rate 3%

```
#Assembly
./minia -in ecoli_reads_280_3.fq -kmer-size 63 -abundance-min 4 -out ecoli_280_3

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_280_3.contigs.fa
```

#### A.8.1.3 SparseAssembler

The result of SparseAssembler is a contig file named *Contigs.txt*. This file should be converted to *Contigs.fasta* for evaluation step using the following command:

```
mv Contigs.txt Contigs.fasta
```

- Dataset with $25x$ and error rate 1%

```
#Assembly
./SparseAssembler g 7 k 41 LD 0 GS 4686137 f ecoli_reads_25_1.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $25x$ and error rate 3%

```
#Assembly
./SparseAssembler g 1 k 27 LD 0 GS 4686137 f ecoli_reads_25_3.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $35x$ and error rate 1%

```
#Assembly
./SparseAssembler g 10 k 45 LD 0 GS 4686137 f ecoli_reads_35_1.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $35x$ and error rate 3%

```
#Assembly
./SparseAssembler g 1 k 31 LD 0 GS 4686137 f ecoli_reads_35_3.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $75x$ and error rate 1%

```
#Assembly
./SparseAssembler g 8 k 63 LD 0 GS 4686137 f ecoli_reads_75_1.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $75x$ and error rate 3%

```
#Assembly
./SparseAssembler g 15 k 43 LD 0 GS 4686137 f ecoli_reads_75_3.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $140x$ and error rate 1%

```
#Assembly
./SparseAssembler g 25 k 63 LD 0 GS 4686137 f ecoli_reads_140_1.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $140x$ and error rate 3%

```
#Assembly
./SparseAssembler g 20 k 47 LD 0 GS 4686137 f ecoli_reads_140_3.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $280x$ and error rate 1%

```
#Assembly
./SparseAssembler g 22 k 63 LD 0 GS 4686137 f ecoli_reads_280_1.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

- Dataset with $280x$ and error rate 3%

```
#Assembly
./SparseAssembler g 25 k 57 LD 0 GS 4686137 f ecoli_reads_280_3.fq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa Contigs.fasta
```

### A.8.1.4 LightAssembler

- Dataset with $25x$ and error rate 1%

```
#Assembly
./LightAssembler -k 43 -g 29 -e 0.01 -G 4686137 -o ecoli_25_1 ecoli_reads_25_1.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_25_1.contigs.fasta
```

- Dataset with $25x$ and error rate 3%

```
#Assembly
./LightAssembler -k 23 -g 12 -e 0.03 -G 4686137 -o ecoli_25_3 ecoli_reads_25_3.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_25_3.contigs.fasta
```

- Dataset with $35x$ and error rate 1%

```
#Assembly
./LightAssembler -k 49 -g 26 -e 0.01 -G 4686137 -o ecoli_35_1 ecoli_reads_35_1.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_35_1.contigs.fasta
```

- Dataset with $35x$ and error rate 3%

```
#Assembly
./LightAssembler -k 27 -g 28 -e 0.03 -G 4686137 -o ecoli_35_3 ecoli_reads_35_3.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_35_3.contigs.fasta
```

- Dataset with $75x$ and error rate 1%

```
./LightAssembler -k 63 -g 35 -e 0.01 -G 4686137 -o ecoli_75_1 ecoli_reads_75_1.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_1.contigs.fasta
```

- Dataset with $75x$ and error rate 3%

```
#Assembly
./LightAssembler -k 43 -g 45 -e 0.03 -G 4686137 -o ecoli_75_3 ecoli_reads_75_3.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_3.contigs.fasta
```

- Dataset with $140x$ and error rate 1%

```
#Assembly
./LightAssembler -k 63 -g 39 -e 0.01 -G 4686137 -o ecoli_140_1 ecoli_reads_140_1.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_140_1.contigs.fasta
```

- Dataset with $140x$ and error rate 3%

```
#Assembly
./LightAssembler -k 49 -g 31 -e 0.03 -G 4686137 -o ecoli_140_3 ecoli_reads_140_3.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_140_3.contigs.fasta
```

- Dataset with $280x$ and error rate 1%

```
#Assembly
./LightAssembler -k 63 -g 29 -e 0.01 -G 4686137 -o ecoli_280_1 ecoli_reads_280_1.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_280_1.contigs.fasta
```

- Dataset with $280x$ and error rate 3%

```
#Assembly
./LightAssembler -k 57 -g 33 -e 0.03 -G 4686137 -o ecoli_280_3 ecoli_reads_280_3.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_280_3.contigs.fasta
```

### A.8.2 LightAssembler using different gap sizes

```
#!/bin/bash
#cov: 75x
#err: 1%
for i in 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40
do

#Assembly
./LightAssembler -k 63 -g ${i} -e 0.01 -G 4686137 -o ecoli_75_1_${i}
ecoli_reads_75_1.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_1_${i}.contigs.fasta

done

#!/bin/bash
#cov: 75x
#err: 3%
for i in 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59
do

#Assembly
./LightAssembler -k 43 -g ${i} -e 0.03 -G 4686137 -o ecoli_75_3_${i}
ecoli_reads_75_3.fq --verbose

#Evaluation
sh getCorrectnessStats.sh e_coli_k12.fa ecoli_75_3_${i}.contigs.fasta


done
```

### A.8.3 Assembly and Evaluation Instructions for real dataset

We compared LightAssembler results with those from Velvet, ABySS, Minia and SparseAssembler. We tried different values of $k$-mer size (31, 43, 45, 47, 49, 53, 57, 59, and 63) for GAGE datasets and (31, 47, 57 and 63) for Assemblathon bird dataset. We also tried different values for the minimum abundance threshold for Minia and different gap sizes for SparseAssembler.

#### A.8.3.1 Velvet

- Human chromosome 14 dataset:

```
#Assembly
./velveth velvet_H 59 -fmtAuto -separate -short frag_1.fastq.gz frag_2.fastq.gz
./velvetg velvet_H -exp_cov auto -scaffolding no

#Evaluation
sh getCorrectnessStats.sh genome.fasta velvet_H/contigs.fa
```

- Bird dataset:

```
#Assembly
./velveth velvet_B 47 -fmtAuto -separate -short ERR244146_1.fastq ERR244146_2.fastq
./velvetg velvet_B -exp_cov auto -scaffolding no

#Evaluation using GAGE
 java GetFastaStats -o -min 200 -genomeSize 1230000000 velvet_B/contigs.fa

#Evaluation using Assemblathon
./assemblathon_stats.pl velvet_B/contigs.fa -genome_size 1230000000
```

- Staphylococcus aureus corrected and uncorrected dataset:
  - Uncorrected dataset

```
#Assembly
./velveth velvet_S 31 -fmtAuto -separate -short frag_1.fastq.gz frag_2.fastq.gz
./velvetg velvet_S -exp_cov auto -scaffolding no

#Evaluation
sh getCorrectnessStats.sh genome.fasta velvet_S/contigs.fa
```

- Corrected dataset

```
#Assembly
./velveth velvet_S_Corr 31 -fmtAuto -separate -short Data/allpathsCor/frag_1.fastq
Data/allpathsCor/frag_2.fastq
./velvetg velvet_S_Corr -exp_cov auto -scaffolding no

#Evaluation
sh getCorrectnessStats.sh genome.fasta velvet_S_Corr/contigs.fa
```

- Rhodobacter sphaeroides corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
./velveth velvet_R 31 -fmtAuto -separate -short frag_1.fastq.gz frag_2.fastq.gz
./velvetg velvet_R -exp_cov auto -scaffolding no

#Evaluation
sh getCorrectnessStats.sh genome.fasta velvet_R/contigs.fa
```

- Corrected dataset

```
#Assembly
./velveth velvet_R_Corr 31 -fmtAuto -separate -short Data/allpathsCor/frag_1.fastq
Data/allpathsCor/frag_2.fastq
./velvetg velvet_R_Corr -exp_cov auto -scaffolding no

#Evaluation
sh getCorrectnessStats.sh genome.fasta velvet_R_Corr/contigs.fa
```

### A.8.3.2 ABySS

- Human chromosome 14 dataset:

```
#Assembly
./ABYSS -k59 frag_1.fastq frag_2.fastq -o H_contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta H_contigs.fasta
```

- Bird dataset:

```
#Assembly
./ABYSS -k63 ERR244146_1.fastq ERR244146_2.fastq -o B_contigs.fasta

#Evaluation using GAGE
 java GetFastaStats -o -min 200 -genomeSize 1230000000 B_contigs.fasta

#Evaluation using Assemblathon
./assemblathon_stats.pl B_contigs.fasta -genome_size 1230000000
```

- Staphylococcus aureus corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
./ABYSS -k31 frag_1.fastq frag_2.fastq -o S_contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta S_contigs.fasta
```

- Corrected dataset

```
#Assembly
./ABYSS -k31 Data/allpathsCor/frag_1.fastq Data/allpathsCor/frag_2.fastq -o S_contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta S_contigs.fasta
```

- Rhodobacter sphaeroides corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
./ABYSS -k31 frag_1.fastq frag_2.fastq -o R_contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta R_contigs.fasta
```

- Corrected dataset

```
#Assembly
./ABYSS -k31 Data/allpathsCor/frag_1.fastq Data/allpathsCor/frag_2.fastq -o R_contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta R_contigs.fasta
```

### A.8.3.3 Minia

- Human chromosome 14 dataset:

```
#Assembly
ls -1 frag_1.fastq frag_2.fastq > HD
./minia -in HD -kmer-size 57 -abundance-min 2 -out H

#Evaluation
sh getCorrectnessStats.sh genome.fasta H.contigs.fa
```

- Bird dataset:

```
#Assembly
ls -1 ERR244146_1.fastq ERR244146_2.fastq > BD
./minia -in BD -kmer-size 47 -abundance-min 3 -out B

#Evaluation using GAGE
 java GetFastaStats -o -min 200 -genomeSize 1230000000 B.contigs.fa

#Evaluation using Assemblathon
./assemblathon_stats.pl B.contigs.fa -genome_size 1230000000
```

- Staphylococcus aureus corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
 ls -1 frag_1.fastq frag_2.fastq > SD
./minia -in SD -kmer-size 31 -abundance-min 2 -out S

#Evaluation
sh getCorrectnessStats.sh genome.fasta S.contigs.fa
```

- Corrected dataset:

```
#Assembly
ls -1 Data/allpathsCor/frag_1.fastq Data/allpathsCor/frag_2.fastq > CSD
./minia -in CSD -kmer-size 31 -abundance-min 1 -out CS

#Evaluation
sh getCorrectnessStats.sh genome.fasta CS.contigs.fa
```

- Rhodobacter sphaeroides corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
ls -1 frag_1.fastq frag_2.fastq > RD
./minia -in RD -kmer-size 31 -abundance-min 2 -out R

#Evaluation
sh getCorrectnessStats.sh genome.fasta R.contigs.fa
```

- Corrected dataset

```
#Assembly
 ls -1 Data/allpathsCor/frag_1.fastq Data/allpathsCor/frag_2.fastq > CRD
./minia -in CRD -kmer-size 31 -abundance-min 1 -out CR

#Evaluation
sh getCorrectnessStats.sh genome.fasta CR.contigs.fa
```

### A.8.3.4 SparseAssembler

- Human chromosome 14 dataset:

```
#Assembly
./SparseAssembler g 15 k 53 LD 0 GS 88289540 i1 frag_1.fastq o1 frag_2.fastq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta Contigs.fasta
```

- Bird dataset:

```
#Assembly
./SparseAssembler g 25 k 57 LD 0 GS 1230000000 i1 ERR244146_1.fastq o1 ERR244146_2.fastq

mv Contigs.txt Contigs.fasta

#Evaluation using GAGE
 java GetFastaStats -o -min 200 -genomeSize 1230000000 Contigs.fasta

#Evaluation using Assemblathon
./assemblathon_stats.pl Contigs.fasta -genome_size 1230000000
```

- Staphylococcus aureus corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
./SparseAssembler g 15 k 31 LD 0 GS 2903081 i1 frag_1.fastq o1 frag_2.fastq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta Contigs.fasta
```

- Corrected dataset

```
#Assembly
./SparseAssembler g 21 k 31 LD 0 GS 2903081 i1 Data/allpathsCor/frag_1.fastq o1
Data/allpathsCor/frag_2.fastq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta Contigs.fasta
```

- Rhodobacter sphaeroides corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
./SparseAssembler g 8 k 31 LD 0 GS 4603060 i1 frag_1.fastq o1 frag_2.fastq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta Contigs.fasta
```

- Corrected dataset

```
#Assembly
./SparseAssembler g 22 k 31 LD 0 GS 4603060 i1 Data/allpathsCor/frag_1.fastq o1
Data/allpathsCor/frag_2.fastq

mv Contigs.txt Contigs.fasta

#Evaluation
sh getCorrectnessStats.sh genome.fasta Contigs.fasta
```

### A.8.3.5 LightAssembler

- Human chromosome 14 dataset:

```
#Assembly
./LightAssembler -k 57 -g 22 -e 0.01 -G 88289540 -o H frag_1.fastq frag_2.fastq --verbose

#Evaluation
sh getCorrectnessStats.sh genome.fasta H.contigs.fasta
```

- Human chromosome 14 dataset using different gap sizes:

```
#!/bin/bash
#different gap sizes
for i in 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
         27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
do

#Assembly
./LightAssembler -k 57 -g ${i} -e 0.01 -G 88289540 -o H_${i} frag_1.fastq frag_2.fastq --verbose

#Evaluation
sh getCorrectnessStats.sh genome.fasta H_${i}.contigs.fasta

done
```

- Bird dataset:

```
#Assembly
./LightAssembler -k 63 -g 4 -e 0.01 -G 1230000000 -o B ERR244146_1.fastq ERR244146_2.fastq
  --verbose

#Evaluation using GAGE
 java GetFastaStats -o -min 200 -genomeSize 1230000000 B.contigs.fasta

#Evaluation using Assemblathon
./assemblathon_stats.pl B.contigs.fasta -genome_size 1230000000
```

- Staphylococcus aureus corrected and uncorrected dataset:
  - Uncorrected dataset

```
#Assembly
./LightAssembler -k 31 -g 22 -e 0.01 -G 2903081 -o S frag_1.fastq frag_2.fastq --verbose

#Evaluation
sh getCorrectnessStats.sh genome.fasta S.contigs.fasta
```

- Corrected dataset

```
#Assembly
./LightAssembler -k 31 -g 1 -e 0.01 -G 2903081 -o S Data/allpathsCor/frag_1.fastq
Data/allpathsCor/frag_2.fastq --verbose

#Evaluation
sh getCorrectnessStats.sh genome.fasta S.contigs.fasta
```

- Rhodobacter sphaeroides corrected and uncorrected dataset:
- Uncorrected dataset

```
#Assembly
./LightAssembler -k 31 -g 4 -e 0.01 -G 4603060 -o R frag_1.fastq frag_2.fastq --verbose

#Evaluation
sh getCorrectnessStats.sh genome.fasta R.contigs.fasta
```

- Corrected dataset

```
#Assembly
./LightAssembler -k 31 -g 1 -e 0.01 -G 4603060 -o R Data/allpathsCor/frag_1.fastq
Data/allpathsCor/frag_2.fastq --verbose

#Evaluation
sh getCorrectnessStats.sh genome.fasta R.contigs.fasta
```

### A.8.4 SSPACE scaffolding results for GAGE human chromosome 14

We used the contigs resulted by different assemblers to perform scaffolding using short jump library downloaded from GAGE website.

To run SSPACE, we created *shortlib.txt* file that contains information about the short jump library and the mapping tool as shown in the following:

```
vi shortlib.txt
lib1 bowtie shortjump_1.fastq shortjump_2.fastq 2700 0.4 RF
```

Then, we ran SSPACE for each contig file as described on the following:

- Velvet

```
#Scaffolding
./SSPACE_Standard_v3.0.pl -l shortlib.txt -s velvet_H/contigs.fa -x 0 -m 32 -o 20
-k 5 -a 0.70 -n 15 -p 0 -v 0 -z 0 -g 3 -T 1 -S 0 -b SSPACE_H_Velvet

#Evaluation
sh getCorrectnessStats.sh genome.fasta velvet_H/contigs.fa
SSPACE_H_Velvet/SSPACE_H_Velvet.final.scaffolds.fasta
```

- ABySS

```
#Scaffolding
./SSPACE_Standard_v3.0.pl -l shortlib.txt -s H_contigs.fasta -x 0 -m 32 -o 20 -k 5
-a 0.70 -n 15 -p 0 -v 0 -z 0 -g 3 -T 1 -S 0 -b SSPACE_H_ABySS

#Evaluation
sh getCorrectnessStats.sh genome.fasta H_contigs.fasta
SSPACE_H_ABySS/SSPACE_H_ABySS.final.scaffolds.fasta
```

- Minia

```
#Scaffolding
./SSPACE_Standard_v3.0.pl -l shortlib.txt -s H.contigs.fa -x 0 -m 32 -o 20 -k 5 -a
0.70 -n 15 -p 0 -v 0 -z 0 -g 3 -T 1 -S 0 -b SSPACE_H_Minia

#Evaluation
sh getCorrectnessStats.sh genome.fasta H.contigs.fasta
SSPACE_H_Minia/SSPACE_H_Minia.final.scaffolds.fasta
```

- SparseAssembler

```
#Scaffolding
./SSPACE_Standard_v3.0.pl -l shortlib.txt -s Contigs.fasta -x 0 -m 32 -o 20 -k 5
-a 0.70 -n 15 -p 0 -v 0 -z 0 -g 3 -T 1 -S 0 -b SSPACE_H_SparseAssembler

#Evaluation
sh getCorrectnessStats.sh genome.fasta Contigs.fasta
SSPACE_H_SparseAssembler/SSPACE_H_SparseAssembler.final.scaffolds.fasta
```

- LightAssembler

```
#Scaffolding
./SSPACE_Standard_v3.0.pl -l shortlib.txt -s H.contigs.fasta -x 0 -m 32 -o 20 -k 5
-a 0.70 -n 15 -p 0 -v 0 -z 0 -g 3 -T 1 -S 0 -b SSPACE_H_LightAssembler

#Evaluation
sh getCorrectnessStats.sh genome.fasta H.contigs.fasta
SSPACE_H_LightAssembler/SSPACE_H_LightAssembler.final.scaffolds.fasta
```

### A.8.5 Light Assembler's scaffolding results using different gap sizes

```
#!/bin/bash
#different gap sizes
for i in 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
do

#Scaffolding
./SSPACE_Standard_v3.0.pl -l shortlib.txt -s H_${i}.contigs.fasta -x 0 -m 32 -o 20
-k 5 -a 0.70 -n 15 -p 0 -v 0 -z 0 -g 3 -T 1 -S 0 -b SSPACE_H_${i}_LightAssembler

#Evaluation
sh getCorrectnessStats.sh genome.fasta H_${i}.contigs.fasta
SSPACE_H_${i}_LightAssembler/SSPACE_H_${i}_LightAssembler.final.scaffolds.fasta
done
```

## A.9 Memory profiling

We used *memusage.cpp* script for measuring the memory peak for all assembly tools by preceding the assembly command line with:

```
./memusage <assembly command line>

#Example

#Assembly
./memusage ./LightAssembler -k 31 -g 1 -e 0.01 -G 4603060 -o R
Data/allpathsCor/frag_1.fastq Data/allpathsCor/frag_2.fastq --verbose
```

## A.10 Accuracy of LightAssembler *k*-mers classification

We used *ComputeCorrectKmers* program to count the number of correct classified *k*-mers, the number of missing *k*-mers and the number of incorrect *k*-mers that are kept in Bloom filter *B*.

```
# Staphylococcus_aureus genome

./ComputeCorrectKmers -k 31 Staphylococcus_aureus.fasta LightAssembler.solid_kmers

# Rhodobacter sphaeroides genome

./ComputeCorrectKmers -k 31 Rhodoba_sphaeroides.fasta LightAssembler.solid_kmers

# Human Chromosome 14

./ComputeCorrectKmers -k 57 Hch14.fasta LightAssembler.solid_kmers
```

# A.11 LightAssembler User Manual

## A.11.1 System requirements

**Operating System:** 64-bit Linux environment.
**Compiler:** g++ or GCC in general.
**Libraries dependency:** pthreads, and zlib.

## A.11.2 How to install LightAssembler

1. Clone the GitHub repo, e.g. with :
   `git clone https://github.com/SaraEl-Metwally/LightAssembler.git`

2. Run `make` in the repo directory for $k \leq 31$ or `make k=kmersize` for $k > 31$,
   e.g. `make k=57`

## A.11.2 Quick usage guide

The basic usage is:

```
./LightAssembler -k [kmer size] -g [gap size] -e [error rate] -G [genome size] -t
[threads] -o [output prefix] [input files] --verbose

#Example
./LightAssembler -k 31 -g 15 -e 0.01 -G 4686137 -o ecoli_35_1 -t 3
ecoli_reads_35_1.fq --verbose
```

- ☒ [**-k**] kmer size [default: 31]
- ☒ [**-g**] gap size [default: 25X:3 35X:4 75X:8 140X:15 280X:25]
- ☒ [**-e**] error rate [default: 0.01]
- ☒ [**-G**] genome size [default: 0]
- ☒ [**-t**] number of threads [default: 1]
- ☒ [**-o**] output prefix file name [default: LightAssembler]

### A.11.2.1 Input read files

LightAssembler assembles multiple input files of Illumina reads given in *FASTA/FASTQ* format. Also, LightAssembler can read directly the input files compressed with *gzip* [*FASTA.gz/FASTQ.gz*].

### A.11.2.2 Outputs

The output of LightAssembler is the set of assembled contigs in *FASTA* format, in the file:

```
[output prefix].contigs.fasta
```

LightAssembler also reports the following on the screen:

- ☒ Number of resulted contigs.
- ☒ Maximum contig length.
- ☒ Total Assembly size.
- ☒ Total genome coverage.
- ☒ Total Assembly time as well as the total time for each step.

Also, by using the *--verbose* option, LightAssembler reports the additional details for each step such as the number of *k*-mers, the false positive rate of Bloom filters and the number of branching *k*-mers in the dataset, the average read length and the average sequencing coverage.