University of Maryland
Institute for Physical Science and Technology

MaSuRCA-2.1.0 Genome Assembler Quick Start Guide

The MaSuRCA (Maryland Super Read Cabog Assembler) assembler combines the benefits of deBruijn graph and Overlap-Layout-Consensus assembly approaches.

Citation: Aleksey Zimin*, Guillaume Marçais, Daniela Puiu, Michael Roberts, Steven L. Salzberg and James A. Yorke, The MaSuRCa genome assembler. *Bioinformatics* (2013) doi: 10.1093/bioinformatics/btt476

1. System requirements/run rimes

Compile/Install. To compile the assembler we require gcc version 4.4 or newer to be installed on the system.
Only Linux is supported (May or may not compile under gcc for MacOS or Cygwin, Windows, etc). The assembler has been tested on the following distributions:

- Fedora 12 and up
- RedHat 5 and 6 (requires installation of gcc 4.4 from an RPM)
- CentOS 5 and 6
- Ubuntu 12 LTS
- SUSE Linux 16

Hardware requirements. The hardware requirements vary with the size of the genome project. Both Intel and AMD x64 architectures are supported. The general guidelines for hardware configuration are as follows:

- Bacteria (up to 10Mb): 16Gb RAM, 8+ cores, 10Gb disk space
- Insect (up to 500Mb): 128Gb RAM, 16+ cores, 1Tb disk space
- Avian/small plant genomes (up to 1Gb): 256Gb RAM, 32+ cores, 1Tb disk space
- Mammalian genomes (up to 3Gb): 512Gb RAM, 32+ cores, 3Tb disk space
- Plant genomes (up to 30Gb): 1Tb RAM, 64+cores, 10Tb disk space

Expected run times. The expected run times depend of course on the cpu speed/number of cores used for the assembly. The following lists the expected run times for the minimum configurations outlined above for Illumina-only data sets. Adding long reads (454, Sanger, etc. makes the assembly run about 50-100% longer:

- Bacteria (up to 10Mb): <1 hour
- Insect (up to 500Mb): 1-2 days
- Avian/small plant genomes (up to 1Gb): 4-5 days
- Mammalian genomes (up to 3Gb): 15-20 days
- Plant genomes (up to 30Gb): 60-90 days

2. Installation instructions

To install, first download the latest distribution from ftp://ftp.genome.umd.edu/pub/MaSuRCA/. Then untar/unzip the package MaSuRCA-X.X.X.tgz, cd to the resulting folder and run './install.sh'. The installation script will configure and make all necessary packages.

In the rest of this document, '/install_path' refers to a path to the directory in which './install.sh' was run.

3. Running the assembler

Overview. The general steps to run the MaSuRCA assemblers are as follows, and will be covered in details in later sections. It is advised to create a new directory for each assembly project.

First, create a configuration file which contains the location of the compiled assembler, the location of the data and some parameters. Copy in your assembly directory the template configuration file '/install_path/sr_config_example.txt' which was created by the installer with the correct paths to the freshly compiled software and with reasonable parameters. Many assembly projects should only need to set the path to the input data.

Second, run the 'masurca' script which will generate from the configuration file a shell script 'assemble.sh'. This last script is the main driver of the assembly.

Finally, run the script 'assemble.sh' to assemble the data.

Configuration. To run the assembler, one must first create a configuration file that specifies the location of the executables, data and assembly parameters for the assembler. The installation script will create a sample config file 'sr_config_example.txt'. Lines starting with a pound sign ('#') are comments and ignored. The sample configuration file looks like this:

```
#example configuration file for rhodobacter sphaeroides assembly from
GAGE project (http://gage.cbcb.umd.edu)
#DATA is specified as type {PE,JUMP,OTHER}= two_letter_prefix mean
stdev fastq(.gz)_fwd_reads fastq(.gz)_rev_reads
#NOTE that PE reads are always assumed to be  innies, i.e. --->  <---,
and JUMP are assumed to be outties <---    --->; if there are any jump
libraries that are innies, such as longjump, specify them as JUMP and
specify NEGATIVE mean
#IT IS MANDATORY to supply some Illumina paired end or single end
reads
#reverse (R2) reads are optional for PE libraries and mandatory for
JUMP libraries
```

```
#any OTHER sequence data (454, Sanger, Ion torrent, etc) must be first
converted into Celera Assembler compatible .frg files (see http://wgs-
assembler.sourceforge.com)
DATA
PE= pe 180 20  /FULL_PATH/frag_1.fastq  /FULL_PATH/frag_2.fastq
JUMP= sh 3600 200  /FULL_PATH/short_1.fastq  /FULL_PATH/short_2.fastq
OTHER=/FULL_PATH/file.frg
END

PARAMETERS
#this is k-mer size for deBruijn graph values between 25 and 101 are
supported, auto will compute the optimal size based on the read data
and GC content.  Do not set this longer than PE read length!!!
GRAPH_KMER_SIZE=auto
#set this to 1 for Illumina-only assemblies and to 0 if you have 2x or
more long (Sanger, 454) reads
USE_LINKING_MATES=1
#this parameter is useful if you have too many jumping library mates.
Typically set it to 60 for bacteria and something large (300) for
mammals
LIMIT_JUMP_COVERAGE = 60
#these are the additional parameters to Celera Assembler.  do not
worry about performance, number or processors or batch sizes -- these
are computed automatically. for mammals do not set cgwErrorRate above
0.15!!!
CA_PARAMETERS = ovlMerSize=30 cgwErrorRate=0.25 ovlMemory=4GB
#auto-detected number of cpus to use
NUM_THREADS= 64
#this is mandatory jellyfish hash size – 10x the genome size is a good
starting value
JF_SIZE=100000000
#this specifies if we do (1) or do not (0) want to trim long runs of
homopolymers (e.g. GGGGGGGG) from 3' read ends, use it for high GC
genomes
DO_HOMOPOLYMER_TRIM=0
END
```

The config file consists of two sections: DATA and PARAMETERS. Each section concludes with END statement. User should copy the sample config file to the directory of choice for running the assembly and then modify it according to the specifications of the assembly project. Here are brief descriptions of the sections.

DATA – in this section the user must specify the types of data available for the assembly. Each line represent a library and must start with PE=, JUMP= or OTHER= for the 3 different type of input read library (Paired Ends, Jumping or other). There can be multiple lines starting with 'PE=' (or JUMP=), one line per library. PE and JUMP data must be in fastq format while the other data is in provided as a Celera Assembler frag format ('.frg'). Every PE or JUMP library is named by a unique two letter prefix. No two library can have the same prefix and a prefix should be

made of two printable characters or number (no space or control characters), e.g. 'aa', 'ZZ', 'l5', or 'J2'.

The following types of data are supported:

- Illumina paired end (or single end) reads -- MANDATORY:
  PE = two_letter_prefix mean stdev /PATH/fwd_reads.fastq /PATH/rev_reads.fastq
  example:
  PE = aa 180 20 /data/fwd_reads.fastq /data/rev_reads.fastq
  The 'mean' and 'stdev' parameters are the library insert average length and standard deviation. If the standard deviation is not known, set it to approximately 15% of the mean.If the second (reverse) read set is not available, do not specify it and just specify the forward reads.
- Illumina jumping/DiTag/other circularization protocol-based library mate pair reads:
  JUMP = two_letter_prefix mean stdev /PATH/fwd_reads.fastq /PATH/rev_reads.fastq
  example:
  JUMP = cc 3500 500 /data/jump_fwd_reads.fastq /data/jump_rev_reads.fastq
  By default, the assembler assumes that the jumping library pairs are “outties” (<-- -->). Some protocols (DiTag) use double-circularization which results in “innie” pairs (--> <--). In this case please specify negative mean.
- Other types of data (454, Sanger, etc) must be converted to CABOG format FRG files (see CABOG documentation at http://sourceforge.net/apps/mediawiki/wgs-assembler/index.php?title=Main_Page):
  OTHER = data.frg

More than one entry for each data type/set of files is allowed. That is if you have several pairs of PE fastq files, specify each pair on a separate line with a different two-letter prefix.

PARAMETERS. The following parameters are mandatory:

- NUM_THREADS=16
  set it to the number of cores in the computer to be used for assembly
- JF_SIZE=2000000000
  jellyfish hash size, set this to about 10x the genome size.

Optional parameters:

- USE_LINKING_MATES=1
  most of the paired end reads end up in the same super read and thus are not passed to the assembler. Those that do not end up in the same super read are called ”linking mates” . The best assembly results are achieved by setting this parameter to 1 for Illumina-only assemblies. If you have more than 2x coverage by long (454, Sanger, etc) reads, set this to 0.
- GRAPH_KMER_SIZE=auto
  this is the kmer size to be used for super reads. “auto” is the safest choice. Only advanced users should modify this parameter.

- LIMIT_JUMP_COVERAGE = 60
  in some cases (especially for bacterial assemblies) the jumping library has too much coverage which confuses the assembler. By setting this parameter you can have assembler down-sample the jumping library to 60x (from above) coverage. For bigger eukaryotic genomes you can set this parameter to 1000.
- CA_PARAMETERS = ovlMerSize=30 cgwErrorRate=0.25 ovlMemory=4GB
  these are the additional parameters to Celera Assembler, and they should only be supplied/modified by advanced users

The masurca and the assemble.sh script. Once you've created a configuration file, use the 'masurca' script from the MaSuRCA bin directory to generate the 'assemble.sh' shell script that executes the assembly:

$ /install_path/ MaSuRCA-X.X.X/bin/masurca config.txt

To run the assembly, execute 'assemble.sh'.
Typically upon completion of the successful assembly, the current directory, where 'assemble.sh' was generated, will contain the following files, in reverse chronological order:

$ ls –lth

FILE	CONTAINS
gapClose.err	STDOUT/STDERR for gap filling
CA	CABOG folder – the final assembly ends up in CA/9-terminator or CA/10-gapclose (if gapClose succeeded, most of the time)
runCA2.out	CABOG stdout for scaffolder
tigStore.err	Stderr for tigStore
unitig_cov.txt	CABOG coverage statistics
global_arrival_rate.txt	CABOG coverage statistics, global arrival rate
unitig_layout.txt	Unitig layout/sequences used to recomputed the coverage statistics
genome.uid	File relating UID to read name for CABOG
runCA1.out	CABOG stdout for unitig consensus
runCA0.out	CABOG stdout for initial stages: store building, overlapping, unitigging

superReadSequences_shr.frg	FRG file for super reads, super reads >2047br are shredded with an overlap of 1500bp
renamed_sr.txt	Deprecated file, ignore
pe.linking.frg	FRG file of PE pairs where the two reads ended up in different super reads
pe.linking.fa	Fasta file of PE pairs where the two reads ended up in different super reads
work1	Working directory of the super reads code that generates the super reads from PE reads
super1.err	STDERR output of the super reads code that generates the super reads from PE reads
sj.cor.clean.frg	CABOG FRG file with corrected jumping library pairs, redundant and non-junction removed, coverage limited
sj.cor.ext.reduced.fa	Fasta file with corrected jumping library pairs, redundant and non-junction removed, coverage limited
mates_to_break.txt	File that lists the jumping library mates that are to be removed, if the jumping library clone coverage exceeds the LIMIT_JUMP_COVERAGE parameter
compute_jump_coverage.txt	Supplementary file used to compute clone coverage of the jumping library, post-filtering
sj.cor.clean.fa	Fasta file with corrected jumping library pairs, redundant and non-junction removed
redundant_sj.txt	Text file with names of the redundant jumping library mate pairs
chimeric_sj.txt	text file with names of the non-junction jumping library mate pairs
work2	working directory for the super reads code used to filter the JUMP libraries for non-junction/redundant pairs
work2.1	working directory for secondary jumping filter based on the variable k-mer size

super2.err	STDERR output of the super reads code used to filter the JUMP libraries for non-junction/redundant pairs
guillaumeKUnitigsAtLeast32bases_all.fasta	fasta file for k-unitigs (see the paper referred above)
k_u_0	jellyfish hash created from all error corrected reads, and used to estimate the genome size
??.cor.fa	error corrected JUMP reads, one such file for each library with '??' being the prefix. The ordering of the reads is arbitrary, but the pairs are guaranteed to appear together. No quality scores.
error_correct.log	log file for error correction
pe.cor.fa	error corrected PE reads. The ordering of the reads is arbitrary, but the pairs are guaranteed to appear together. No quality scores
combined_0	special combined Jellyfish hash for error correction
pe_data.tmp	supplementary information to figure out the GC content and lengths of PE reads
sj.renamed.fastq	the ??.renamed.fastq file is created for each "JUMP= …" entry in the configuration file. These file(s) contain renamed reads in the fastq format
meanAndStdevByPrefix.sj.txt	auto-generated file of "fake" mean and stdev for non-junction jumping library pairs. The Illumina protocol states that these are about 200-700bp long
pe.renamed.fastq	the ??.renamed.fastq file is created for each "PE= …" entry in the configuration file. These file(s) contain renamed reads in the fastq format.
meanAndStdevByPrefix.pe.txt	auto-generated file of means and stdevs for PE reads
assemble.sh	the original assemble.sh script

Restarting a failed assembly. If something fails or goes wrong, or you noticed a mistake made in configuration, you can stop and re-start the assembly as follows.

- Terminate the assembly by Control-C or by killing the 'assemble.sh' process

- Examine the assembly folder and delete all files that contain incorrect/failed contents (see table above for file designations).
- Run $/install_path/MaSuRCA-X.X.X/bin/masurca config.txt in the assembly directory. This will create a new 'assemble.sh' script accounting for the files that are already present and checking for all dependencies to only run the steps that need to be run
- Run ./assemble.sh

For example

- if you noticed that CABOG failed due to lack of disk space, then, after freeing some space, simply run $/install_path/MaSuRCA-X.X.X/bin/masurca config.txt and execute 'assemble.sh'
- if you noticed that you omitted or misspecified one of the jumping library files, add the files to the DATA section of config.txt and run $/install_path/ MaSuRCA-X.X.X/bin/masurca config.txt and execute 'assemble.sh'
- if error correction failed then remove the files named '??.cor.fa' and then run $/install_path/ MaSuRCA-X.X.X/bin/masurca config.txt and execute 'assemble.sh'

Assembly result. The final assembly files are under CA/10-gapclose and named 'genome.ctg.fasta' for the contig sequences and 'genome.scf.fasta' for the scaffold sequences.