**Simplified workflow for short-read genome assembly using paired-end Illumina reads and MaSuRCA**

© Shweta V. Pipaliya

Last Updated – July 28, 2020

**Typographical notes about this document**: *Italics denote additional information or comments (please read those ☺* ), actual instructions in Arial 10pt, and command prompts denoted as $ mycommand. Additionally, there are embedded links to original program manuals or papers that may be of help.

**1. INTRODUCTION**

Genomes are an invaluable source for investigation of many biological questions. Often genome assembly and genomics in general can feel like a daunting task but it doesn’t have to be. It goes without saying, read plenty of papers to familiarize yourself with the general workflow of the genome assembly process you have chosen to embark on, whether it is short-read, long-read, hybrid de novo, or reference-based assembly. Understanding the basics and the steps involved will significantly cut down on the amount of troubleshooting or failed experiments you’ll run into later. In the most summarized manner, these are the steps you can expect from start to finish for a draft genome assembly, gene predictions, and annotation.

**

**2. BEFORE YOU BEGIN**

*Before you start the assembly process, it is extremely important that you know your data. Growth conditions (i.e. were the organisms grown in axenic conditions, in vivo, bacterial monoculture etc.), expected genome size, sequencing protocol, paired vs. mated paired reads, and genome ploidy.* All of these factors will determine what assembler you use, whether you need to filter your reads **prior** to the assembly process, or what to specify in your assembly parameters. Following are some examples of things for considerations:

* Sequencer of choice
* Approximate/Expected Genome Size
* Genome ploidy
  + Haploid
  + Diploid/Polyploid – Expected allelic variation
* Single cell vs. culture genomics
* Sequencing depth and coverage
* Number of sequencing runs (one vs. multiple)
* Bacterial or other contamination expected?
* Computational Resources available
  + Memory, CPU, RAM, and Disk Space
  + Access to clusters and grid environments (Compute Canada or UAlberta Bioinfor)

**3. RETRIEVING RAW READS**

*First step with any genome assembly projects is retrieving raw sequencing reads. These maybe delivered to you in form of a USB or maybe deposited in a public File Transfer Protocol (FTP) or Galaxy based repositories. I recommend the European Nucleotide Database over NCBI’s Short Read Archive Database to retrieve reads if you are working with a dataset published on the NCBI Short Read Archive (Unless of course you want to waste your time with the SRA ToolKit). You can use the same project accession to search ENA and it should pull out all experimental runs from that project.*

*It is inefficient to download those directly onto your computer unless you have a computer that allows you to store large amounts of data locally. But even then, this is not recommended. Best to make use of the large amounts of allocated storage space we get through the UAlberta’s bioinfor or Compute Canada accounts storage. If you have reads from a sequencing run that were given to you on a USB or a drive by the sequencing facility transfer these onto an FTP server using Globus Filesystems. It is also good practice to make deposit these to a repository for ease of access by others in your group or field. If you are working with reads stored on cloud or an external repository, below is the best way to retrieve and store them.*

1. Navigate to the folder where you wish to store the reads. Identify, copy, and save the **ftp** or **galaxy** URL address for your single or forward and reverse reads linked to a specific experimental run

*If you are downloading multiple runs OR doing a population genomics analyses, navigate to the project page and the corresponding XML report with all of the sample accessions for that project and build a csv metafile that parses out project ID, sample, run, and fwd and rev read ftps into separate columns*.

2. Instead of downloading a large dataset, like a reads file, using a download button, use curl or wget to pull and unzip the fastq files from the appropriate repository directly into your bioinfor/compute canada folder (replace the ftp link with yours and unzipping optional/dependent on downstream programs):

$ curl ftp.sra.ebi.ac.uk/vol1/fastq/SRR195/007/SRR1957167/SRR1957167\_1.fastq.gz --/output SRR1957167\_1.fastq.gz

$ gunzip SRR1957167\_1.fastq.gz

*If you are downloading multiple reads files, best to use the shell script I have written (scripts > batch\_curl\_reads.sh), replace your links that you parsed out in your csv file, and submit it to be batch processed on a cluster.*

**4. PREPROCESSING AND READ QUALITY ASSESSMENT**

*As mentioned above, it is very important that you assess the quality of your dataset. Often times, organisms are not grown in xenobiotic conditions or samples are retrieved from in vivo systems. It is therefore very easy to sequence cells from that environment that are not your organism of interest. In this case, the first step then becomes the coverage of your reads from the organism of interest and contamination.*

*To do this, you need to use a tool specifically developed for taxonomic classification of sequences (reads or assembled fasta). Popular ones include Kraken, Kraken2, and Centrifuge (all three developed by the same group over the years). Kraken is available as a module on Compute Canada but is no longer maintained therefore recommended to choose between Kraken2 and Centrifuge. There are pros and cons to each (for a full list and explanation, visit* [*this*](http://ccb.jhu.edu/software/choosing-a-metagenomics-classifier/) *webpage). TLDR, Kraken2 has several advantages over Centrifuge, especially reducing redundancy and ambiguity in classification.*

***Kraken2*** *is a good tool to map short genomic sequences (i.e. reads) to a database such as the NCBI RefSeq/nr or custom-built databases containing sequences with taxonomic assignments. Building of the program is resource intensive, therefore it is recommended that you instead use Kraken or Kraken2 modules are available on Compute Canada clusters. If you chose to install Kraken2 locally, there are numerous dependencies you must also have installed, therefore please check the Kraken2 GitHub for specific instructions. In any case, before you begin your analyses, you must build a database for taxonomic classification of your reads or fasta sequences. Following are three different options for doing this.*

1.1 Option 1- Nick Loman’s pre-built database and retrieval of microbial-fat-free (2018) hash.k2d, opts.k2d, taxo.k2d libraries that are 30GB in total size. Simply navigate to a folder where you wish to install these libraries and run the following:

$ mkdir kraken2-microbial-fatfree/

$ cd kraken2-microbial-fatfree/

$ wget -c https://refdb.s3.climb.ac.uk/kraken2-microbial/hash.k2d

$ wget https://refdb.s3.climb.ac.uk/kraken2-microbial/opts.k2d

$ wget https://refdb.s3.climb.ac.uk/kraken2-microbial/taxo.k2d

* + 1. If your download halts simply use wget -c to restart where it stopped

*Visit* [*https://lomanlab.github.io/mockcommunity/mc\_databases.html*](https://lomanlab.github.io/mockcommunity/mc_databases.html) *if you wish to install the larger maxikraken2 that additionally has the Human RefSeq database and updated libraries for archaea, bacteria, fungi, viral, and protozoa (March 2019). The size of this one is much larger – 140GB.*

1.2 Option 2. Building a custom database using select NCBI RefSeq Databases plus custom addition of specific genomes including closest available genome to the genome of interest. Following is an example of a library (kraken2-microbial-soup-giardia) build with NCBI RefSeq Bacteria, Archaea, Human, Protist, Viral, and UniVec + addition of *Giardia intestinalis* AWB reference genome:

$ kraken2-build --download-taxonomy --db kraken2-microbial-soup-giardia --use-ftp

$ kraken2-build --download-library archaea --db kraken2-microbial-soup-giardia --use-ftp

$ kraken2-build --download-library bacteria --db kraken2-microbial-soup-giardia --use-ftp

$ kraken2-build --download-library viral --db kraken2-microbial-soup-giardia --use-ftp

$ kraken2-build --download-library protozoa --db kraken2-microbial-soup-giardia --use-ftp

$ kraken2-build --download-library UniVec\_Core --db kraken2-microbial-soup-giardia --use-ftp

$ kraken2-build --add-to-library GL2\_ref\_2019\_chr5.fa --db kraken2-microbial-soup-giardia

$ kraken2-build --add-to-library GL2\_ref\_2019\_chr4.fa --db kraken2-microbial-soup-giardia

$ kraken2-build --add-to-library GL2\_ref\_2019\_chr3.fa --db kraken2-microbial-soup-giardia

$ kraken2-build --add-to-library GL2\_ref\_2019\_chr2.fa --db kraken2-microbial-soup-giardia

$ kraken2-build --add-to-library GL2\_ref\_2019\_chr1.fa --db kraken2-microbial-soup-giardia

$ kraken2-build --build --db kraken2-microbial-soup-giardia

$ kraken2-build --clean --db kraken2-microbial-soup-giardia

*Last line is optional but recommended: It removes intermediate build files and caches to free up memory.*

Note: Sometimes there are issues with installation of NCBI RefSeq databases via Rsync. Specify the --use-ftp or --use-wget option to force Kraken to not use Rsync. IMPORTANT: Downloaded genomes you wish to add to your custom database should ideally have an NCBI accession associated to the contigs. If not, you will need to assign taxid ID in Kraken format as specified below:

>sequence16|kraken:taxid|32630 sequence\_name

CAAGCAGAAGACGGCATACGAGATCTTCGAGTGACTGGAGTTCCTTGGCACCCGAGAATTCCA

1.2.1 If you are downloading a ton of genomes belonging to specific genus or taxid, it is recommended that you do this efficiently by batch retrieval. One option (and my preferred way) is downloading of NCBI RefSeq genomes is using ncbi-genome-download python package by Kai Blin (requires >Python 3.5)

$ pip install ncbi-genome-download

$ ncbi-genome-download name\_of\_lineage

Refer to the GitHub page for more options (<https://github.com/svpipaliya/ncbi-genome-download>) like installation and usage via Conda or python virtual environment.

1.2.2 Another option is using the ‘datasets’ tools by NCBI. Following is an example for installation on MacOS. Refer to NCBI page for Linux commands.

$ curl -o datasets 'https://ftp.ncbi.nlm.nih.gov/pub/datasets/command-line/LATEST/mac/datasets'

$ chmod +x datasets

Simple - Download all datasets associated with the mouse genome (*mus musculus)*

$ ./datasets assembly-descriptors tax-name 'mus musculus'

Batch - Download a compact, dehydrated bag, containing metadata and file locations for 29 primate RefSeq genomes, then retrieve the data when needed using rehydrate:

# First download the dehydrated bag (<10 MB) for 29 primate (Taxonomy ID: 9443) RefSeq genomes

$ ./datasets download assembly tax-id 9443 --refseq --dehydrated --filename primates\_refseq\_dehydrated.zip

Use --help if you want to explore other options

*Building a custom database is very resource intensive (both RAM and Disk Space). The estimated build time can also be anywhere from 6-18 hours. For these reasons, it is therefore highly recommended you do this on a cluster node using multiple threads.*

1.3. Option 3. Building and using Kraken2’s default database MiniKraken

$ kraken2-build --standard --db $DBNAME

*Regardless of which option you chose to work with, it is important you always check your databases after they have finished building to inspect the contents and to make sure your taxa of interest are represented. Remember, classification of sequences is only as good as your database!*

2. Inspecting Kraken2 databases relies on the kraken2-inspect script:

$ kraken2-inspect --db path/to/db

This will produce a report file with all taxids present in the database. Use the –help option if you want more options for inspect usage.

3. Running your paired-end reads through Kraken2 database you finished building. If you chose to install Kraken2 locally, you will need to ensure that the program path is specified in your bash\_profile. I personally recommend you use the Kraken2/2.0.8 module available on Compute Canada clusters (Cedar, Graham, and Beluga). In any case, below are commands for running Kraken2

$ module load gcc/7.3.0

$ module load nixpkgs/16.09

$ module load kraken2/2.0.8-beta

$ kraken2 -db $DBNAME --threads 16 --report output\_name.fastq.report --unclassified-out/ output\_name.fastqunclassified#.fq --classified-out output\_name.fastq.fastqclassified#.fq --/paired /path/to/forward/reads/r1.fastq /path/to/reverse/reads/r2.fastq > /output\_name.fastq.Kraken.out

*Note: These set of commands will also parse out the unclassified from classified reads. If you wish to omit this step, simply remove the option* --unclassified-out and --classified-out. *If you additionally choose to run this on CC, apart from the Kraken2 module, you will also need to call gcc/7.3.0 and nixpkgs/16.09. Important: you must specify* --report *and* *kraken.out. The former will produce a tabulated report of % of fragments that were classified into specific taxa that are ranked as following: R – root, D – domain, K- kingdom, P – phylum, C – Class, F – family, G – Genus, and S – species. It is recommended you parse this into a csv file. The latter*

*If you are working with multiple genomes and paired-end read files and wish to repeat this task recursively, it is recommended you loop this command on each seq file in specified reads folder or make use of SLURM Job Array. Below is a very simple example of job-array script making use of a config file looks like this:*

# get file name and call SLURM\_Array

$ echo "Starting task $SLURM\_ARRAY\_TASK\_ID"

$ DIR=$(sed -n "${SLURM\_ARRAY\_TASK\_ID}p" job.conf)

$ cd $DIR

# run some task on all files with same extension in subfolders specified in job.conf

$ ./program\_name -l some\_option -f \*.extension -o output

$ pwd

$ ls

*Apart from assessing for microbial contamination, you will also need to examine for sequencing read quality and adaptor sequence contamination. A popular tool for this is* ***FASTQC*** *that will generate a report file consisting of per base and sequence quality scores, GC content, ambiguous bases (N), sequence length distribution, and overrepresented sequences and kmers. Below are instructions for FASTQC installation and usage:*

4*.*1To install FASTQC on MacOS (See FASTQC website for installation for Linux or Windows):

$ wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc\_v0.11.9.dmg

*Since this is a java package, you will be required to install or update java on your system.*

4.2 To examine the quality of your individual read files (either zipped or unzipped), simply open FASTQC > file > open > path/to/ r1andr2.fastq.gz

*Examine for flags. You will most likely have adapter sequences and poor base quality scores that will required to be trimmed.* ***Important: If you are working with MaSuRCA, do not perform any processing of the reads as MaSuRCA has an in-built trimming and reads QC tool QuorUM. Pre-processing of reads may result in a deteriorated assembly!!*** *If you are not using MaSuRCA, you will need to trim out poor quality sequences. Popular tools include Trimmomatic and Trimal. Trimmomatic is easy to install and use*:

5.1. Download Trimmomatic binary file that contains trimmomatic-0.36.jar and folder containing adapter sequences in fasta format. Add your paired-end read files in zipped or unzipped format and run the code below with the specified trimming parameters

$ wget <http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.39.zip>

$ gunzip \*.zip

$ java -jar /path/to/trimmomatic-0.39.jar PE $r1.fastq.gz $r2.fastq.gz/ "${prefix}"\_1\_output\_forward\_paired.fq.gz "${prefix}"\_1\_output\_forward\_unpaired.fq.gz/ "${prefix}"\_2\_output\_reverse\_paired.fq.gz "${prefix}"\_2\_output\_reverse\_unpaired.fq.gz/ ILLUMINACLIP:NexteraPE-PE.fa:1:30:11:8:true LEADING:3 TRAILING:3 MINLEN:50

*This will perform the following:*

*Remove adapters but keep PE reads even if adapters are found (ILLUMINACLIP:NexteraPE-PE.fa:* *1:30:11:8:true) \*\* important to specify true after 8 if you wish to keep all reads*

*Remove leading low quality or N bases (below quality 3) (LEADING:3)*

*Remove trailing low quality or N bases (below quality 3) (TRAILING:3)*

*Scan the read with a 4-base wide sliding window, cutting when the average quality per base drops below 15 (SLIDINGWINDOW:4:15)*

*Drop reads below the 50 bases long (MINLEN:50) \*\*Minlen can be adjusted to be longer or shorter depending on your data. I like to keep mine at 50. Default is 36 but this in my opinion is too short. >90 can get too stringent. Everything I leave as is.*

**5. CONTAMINANT FILTERING**

*Once you have finished inspecting your reads, you will need to decide whether there is enough contamination that can be removed by the assembler during assembly process or if filtering will have to be performed prior to. In either instance, filtering of genomes is a crucial part of assembly process whether or not you decide to do this with the reads or with the assembled contigs. There are different ways of performing this. Common practices include graphing of GC content in proportion to sequence length or BLAST based approaches. There are drawbacks to each of these and therefore I prefer to use a taxonomy-based parsing using KrakenTools developed by Jen Lu which makes use of the Kraken2 database and kraken.out and .report output files.*

1.1 Download the KrakenTools python package from Jen Lu’s GitHub containing many useful scripts for metagenomic and post-Kraken2 analyses. For our purpose, we only need the extract\_kraken\_reads.py script to filter the paired end reads.

$ git clone <https://github.com/jenniferlu717/KrakenTools.git>

$ chmod +x \*.py

$ ./extract\_kraken\_reads.py -k reads\_kraken\_output.kraken -s1 r1.fastq.gz -s2 r2.fastq.gz -t taxidID\_to\_extract --include-children output\_kraken.report -o1 r1\_filtered.fastq.gz -o2 r2\_filtered.fastq.gz

*You will be required to input the output .kraken and .report files for the Kraken run corresponding the specific PE reads . You will also be required to specify the taxidID that you wish to extract from the contaminants.* --include-children *parses out all taxid falling within that parent taxid specified (Example: All species within a specific genus or all lineages within a kingdom).* *Alternatively, if you wish to parse specific reads from different taxa falling within different lineages, you can specify those individually or a combination of parent and children taxid. Use* --h *for these additional options or refer to the README.md.*

1.2. To work with a fasta file, the variation of above command:

$ ./extract\_kraken\_reads.py -k SRR1957168.kraken -U SRR1957168.final.genome.scf.fasta -t 0 5741 941442 1394984 658858 598745 -o SRR1957169.filtered.fasta

*Note: Taxid 0 refers to inclusion of unclassified sequences with the classified sequences belonging to the specified list of taxids.*

**6. GENOME ASSEMBLY USING MASURCA**

*Now that you have dealt with the pre-processing of the reads, we can finally move onto the genome assembly process which entails reads mapping via de Bruijin graphing or overlap consensus methods (or hybrid), consensus sequence assembly into contigs, contig stitching into scaffolds, and polishing. For the most part, these steps are in-built into the assemblers using various tools. Choosing an appropriate assembler become important depending on what type of data you are working with including considerations for dealing with genome ploidy or heterozygosity. We will be using MaSuRCA, a ploidy aware assembler, that is compatible with many different types of data (single reads, paired end reads, mate-paired reads, and long reads from ONT or PacBio). Apart from flexible data compatibility, MaSuRCA uses a combination of de bruijn graphing and OLC to generate superreads. For details on MaSuRCA performance against other populat assemblers and how superreads are generated, read the paper published by* [*Aleksey Zimin et al.*](https://academic.oup.com/bioinformatics/article/29/21/2669/195975)

*Compilation of MaSuRCA is resource intensive and requires several dependencies. If you wish to install this locally, please read the MaSuRCA instruction manual available on <https://github.com/alekseyzimin/masurca>. Assembly should generally be run on clusters if they are available it computationally intensive process. MaSuRCA is available as a module on Compute Canada clusters. Unlike many assemblers, parameters have to be specified in a separate config file (there are a lot!).*

1.1 Copy the parameters below into a text file named masurca.config

# DATA is specified as type {PE,JUMP,OTHER,PACBIO} and 5 fields:

# 1)two\_letter\_prefix 2)mean 3)stdev 4)fastq(.gz)\_fwd\_reads

# 5)fastq(.gz)\_rev\_reads. The 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. Reverse 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

#Illumina paired end reads supplied as <two-character prefix> <fragment mean> <fragment stdev> <forward\_reads> <reverse\_reads>

#if single-end, do not specify <reverse\_reads>

#MUST HAVE Illumina paired end reads to use MaSuRCA

PE= pe 500 50 /FULL\_PATH/frag\_1.fastq /FULL\_PATH/frag\_2.fastq

#Illumina mate pair reads supplied as <two-character prefix> <fragment mean> <fragment stdev> <forward\_reads> <reverse\_reads>

JUMP= sh 3600 200 /FULL\_PATH/short\_1.fastq /FULL\_PATH/short\_2.fastq

#pacbio OR nanopore reads must be in a single fasta or fastq file with absolute path, can be gzipped

#if you have both types of reads supply them both as NANOPORE type

#PACBIO=/FULL\_PATH/pacbio.fa

#NANOPORE=/FULL\_PATH/nanopore.fa

#Other reads (Sanger, 454, etc) one frg file, concatenate your frg files into one if you have many

#OTHER=/FULL\_PATH/file.frg

#synteny-assisted assembly, concatenate all reference genomes into one reference.fa; works for Illumina-only data

#REFERENCE=/FULL\_PATH/nanopore.fa

END

PARAMETERS

#PLEASE READ all comments to essential parameters below, and set the parameters according to your project

#set this to 1 if your Illumina jumping library reads are shorter than 100bp

EXTEND\_JUMP\_READS=0

#this is k-mer size for deBruijn graph values between 25 and 127 are supported, auto will compute the optimal size based on the read data and GC content

GRAPH\_KMER\_SIZE = auto

#set this to 1 for all Illumina-only assemblies

#set this to 0 if you have more than 15x coverage by long reads (Pacbio or Nanopore) or any other long reads/mate pairs (Illumina MP, Sanger, 454, etc)

USE\_LINKING\_MATES = 0

#specifies whether to run the assembly on the grid

USE\_GRID=0

#specifies grid engine to use SGE or SLURM

GRID\_ENGINE=SLURM

#specifies queue (for SGE) or partition (for SLURM) to use when running on the grid MANDATORY

GRID\_QUEUE=all.q

#batch size in the amount of long read sequence for each batch on the grid

GRID\_BATCH\_SIZE=500000000

#use at most this much coverage by the longest Pacbio or Nanopore reads, discard the rest of the reads

#can increase this to 30 or 35 if your reads are short (N50<7000bp)

LHE\_COVERAGE=25

#set to 0 (default) to do two passes of mega-reads for slower, but higher quality assembly, otherwise set to 1

MEGA\_READS\_ONE\_PASS=0

#this parameter is useful if you have too many Illumina jumping library mates. Typically set it to 60 for bacteria and 300 for the other organisms

LIMIT\_JUMP\_COVERAGE = 300

#these are the additional parameters to Celera Assembler. do not worry about performance, number or processors or batch sizes -- these are computed automatically.

#CABOG ASSEMBLY ONLY: set cgwErrorRate=0.25 for bacteria and 0.1<=cgwErrorRate<=0.15 for other organisms.

CA\_PARAMETERS = cgwErrorRate=0.15

#CABOG ASSEMBLY ONLY: whether to attempt to close gaps in scaffolds with Illumina or long read data

CLOSE\_GAPS=1

#number of cpus to use, set this to the number of CPUs/threads per node you will be using

NUM\_THREADS = 16

#this is mandatory jellyfish hash size -- a safe value is estimated\_genome\_size\*20

JF\_SIZE = 200000000

#ILLUMINA ONLY. Set this to 1 to use SOAPdenovo contigging/scaffolding module.

#Assembly will be worse but will run faster. Useful for very large (>=8Gbp) genomes from Illumina-only data

SOAP\_ASSEMBLY=0

#If you are doing Hybrid Illumina paired end + Nanopore/PacBio assembly ONLY (no Illumina mate pairs or OTHER frg files).

#Set this to 1 to use Flye assembler for final assembly of corrected mega-reads.

#A lot faster than CABOG, AND QUALITY IS THE SAME OR BETTER.

#Works well even when MEGA\_READS\_ONE\_PASS is set to 1.

#DO NOT use if you have less than 15x coverage by long reads.

FLYE\_ASSEMBLY=0

END

*IMPORTANT NOTE: Unless you are performing hybrid or long read assembly, many of these parameters can be commented out (DO NOT DELETE THEM). For* ***PE Illumina-only assembly*** *the following will be necessary and changed according to your dataset:*

1.2. Specify the average insert length and the stdev of read length. This information can be determined through the FASTQC report or any additional report files that may have been provided to you post-sequencing run. If you don’t know the stdev, set it to approx. 15% of the average fragment length.

#MUST HAVE Illumina paired end reads to use MaSuRCA

PE= pe 250 38 /FULL\_PATH/frag\_1.fastq /FULL\_PATH/frag\_2.fastq

1.3 I prefer to let MaSuRCA auto compute the Kmer sizes but if you want to specify a specific value, you can go ahead and do that.

#this is k-mer size for deBruijn graph values between 25 and 127 are supported, auto will compute the optimal size based on the read data and GC content

GRAPH\_KMER\_SIZE = auto

1.4 Change USE\_LINKING\_MATES value to 1

#set this to 1 for all Illumina-only assemblies

#set this to 0 if you have more than 15x coverage by long reads (Pacbio or Nanopore) or any other long reads/mate pairs (Illumina MP, Sanger, 454, etc)

USE\_LINKING\_MATES = 1

1.5 Specify GRID engine to SLURM (Compute Canada makes use of SLURM clusters)

#specifies grid engine to use SGE or SLURM

GRID\_ENGINE=SLURM

1.6 Specify number of threads for assembly run

#number of cpus to use, set this to the number of CPUs/threads per node you will be using

NUM\_THREADS = 16

1.7 Specify JellyFish hash size. Estimated genome size \* 20

#this is mandatory jellyfish hash size -- a safe value is estimated\_genome\_size\*20

JF\_SIZE = 100000000

1.8 Specify whether you wish to use the Soap de novo contigging module.

#ILLUMINA ONLY. Set this to 1 to use SOAPdenovo contigging/scaffolding module.

#Assembly will be worse but will run faster. Useful for very large (>=8Gbp) genomes from Illumina-only data

SOAP\_ASSEMBLY=0

1.9 Upload masurca.config specifying read paths and assembly parameters to either projects or scratch (DO NOT RUN FROM HOME DIRECTORY). Generate a SLURM script calling the following necessary modules to run MaSuRCA on Compute Canada cluster and executing ./assemble.sh generated by MaSuRCA

$ module load gcc/7.3.0

$ module load nixpkgs/16.09

$ module load masurca/3.3.0

$ masurca masurca.config

$ ./assemble.sh

*It is recommended you run this on atleast 16 CPUs each with 10GB of memory allocated for genomes that are around 10MB run. Larger genomes will likely require 32 cores. Run times vary on genome size and allocated computational resources. Small genomes with minimal resource specifications will take 1-3 hours. Large genomes will take multiple days. Plant genomes will take the longest to assemble (up to three months). In case you do run out memory or time, checkpoint files are generated to be able to restart your run.*

**INSTRUCTION SECTIONS ON POST-ASSEMBLY STATISTICS ANALYSES (KRAKEN2 INSPECTION OF CONTIGS, ASSESSMENT OF ASSEMBLY COMPLETENESS/N50, QUAST) PENDING**

**INSTRUCTIONS FOR BUSCO INSTALLATION AND ANALYSES PENDING**

**INSTRUCTIONS ON GENERATING KRONA PLOTS (ADD TO PRE-ASSEMBLY KRAKEN2 SECTION)**

**INSTRUCTION SECTIONS ON GENE AND PROTEIN PREDICTION (AUGUSTUS AND LIFTOFF) PENDING**