1 MSMC Tutorial

MSMC2 Workflow and Code, v1.2 (Nov 2019)

This guide is intended to get you started on MSMC analyses, and to provide scripts for you to work off of. These scripts may not be perfect for your analyses, however, so please also read the manuals for the different programs being used (e.g. guides to PSMC/MSMC/MSMC2) so that you understand and can make good choices about parameter settings and the workflow you're using.

To start with, you need (1) a reference genome, (2) whole genome resequencing data, (3) the MSMC2 program (already installed as a module on Teton), and (4) the MSMC helper tools. The more complete your genome is, the better your plots will look (in my experience). Theoretically you could do this with *de novo* WGS assembly, but I don't think it's recommended, as the analyses are performed per chromosome. If your genome is in a bunch of pieces, one way to fix this for MSMC is to concatenate groups of the scaffolds together, separated by a bunch of 'N's. Theoretically, this shouldn't mess up your estimates, as long as you space your scaffolds far enough apart. Here's how you could do that:

```
echo '> Genome_name' > genome_new.fasta
cat genome.fa | sed 's/^>.*$/\n\(N\)\{100\}/g' | tr -d '\n' |
    fold -w 80 >> genome_new.fasta
```

Another (probably better) thing to do could be to use Chromosomer or a similar program to assemble your draft genome to a more-complete published genome. If you're interested in how the quality of your reference genome may affect your inferences, here's a neat paper using simulations to test this: Patton et al. 2019, MBE.

Once you have a good genome, you'll also need the following scripts:

1. `msmc_params` control file, which allows you to specify parameters in common for all of the scripts. It is important to change this to match your specfic data and run parameters.
2. `run_snpable.sh` – this generates a "mappability mask" for your reference genome using Heng Li's SNPable Regions program
3. `submit_1.txt` and `msmc_1_call.sh` – this generates `vcf` and `mask` files for each individual and each chromosome. The submission script loops this script over all individuals and all chromosomes.
4. `submit_2.txt` and `msmc_2_generateInput_singleInd.sh` (if working with one indv) – this uses the `vcf` and `mask` files to generate a msmc input file when you will be analyzing each individual separately.
5. `submit_2_multi.txt` and `msmc_2_generateInput_multiInd.sh` (if multiple ind per pop), which will generate input files for all individuals and all chromosomes for runs where you intend to combine multiple individuals.
6. `msmc_3_runMSMC_onepop.sh`
7. bootstrap scripts: `msmc_4_bootstraps.sh`, `submit_bootstrap.sh`

These scripts are appended to this guide and can be found on Github at https://github.com/jessicarick/msmc2_scripts.

1.1 Alignment

The MSMC process starts with sorted `.bam` files, so if you haven't already, you'll need to align your `fastq` reads to your reference genome. You can do this in whatever way you fancy, such as with a `bwa` or `bowtie` pipeline.

1.2 Step 0 - Create Mappability Mask

One more preparation step before you begin (which can also be run concurrently with `Step 1` below) is to create a mappability mask for your genome. To do this, you need the scripts for SNPable, which you'll find linked on Heng Li's website. You'll also need the `makeMappabilityMask.py` script from the msmc-tools-master set of scripts.

This process involves first extracting all k-mer subsequences from the genome as read sequences, then aligning them back to the genome to get an estimate of how "mappable" different regions are. This is done using the `run_snpable2.sh` script, which requires `samtools` and `bwa`, in addition to the `splitfa`, `gen_raw_mask.pl`, and `gen_mask` scripts from `SNPable` and `msmc-tools-master`.

```
# running on command line
./run_snpable2.sh
# submitting as a SLURM job
sbatch run_snpable2_slurm.sh
```

1.3 Step 1 - Call Variants

From here, you will need to edit the `msmc_params` control file to match your data. The `OUTDIR` is the folder where you want to be working; *GENOME* is the (full) path to the reference genome that you aligned your reads to; `BAMDIR` is the directory containing all of your sorted bamfiles; `BAMFILE` indicates the syntax for how your bamfiles are named (e.g. `$IND.bowtie2.sorted.bam`).

The script uses the `samtools-bcftools-vcftools` pipeline for calling variants, but could be modified if you'd prefer a different pipeline. You'll also need a file called `SCAFFOLDS.txt` with all of the scaffold names in your reference genome and a file called `INDS.txt` with all of your individual IDs. If you don't already have the list of scaffolds, you can create one using this simple one-liner:

```
grep '^>' reference.fa | sed 's/>//' > SCAFFOLDS.txt
```

Once you have this file and you've edited both the `msmc_1_call.sh` and `submit_1.txt` scripts to correspond to your files, you can run this first step using:

```
source submit_1.txt
```

1.3.1 Step 1.5 - Phasing

If you are planning to phase your data, this is the time! I won't be explaining how to do it, but common phasing programs to use are Beagle (Browning & Browning 2007), fastPHASE (Scheet & Stephens 2006), and SHAPEIT2 (Delaneau et al. 2013).

Alternatively, you can generate VCF files however you'd like and then jump into the MSMC process with `Step 2`.

1.4 Step 2 - Generate Input

Now that you have `vcf` files and `mask` files for each individual and each chromosome/scaffold, you can use these to generate the input for MSMC2 using `msmc_2_generateInput_singleInd.sh`. This script is for analyzing a single genome/individual separately; if you have multiple individuals per population/species that you want to combine in analyses, then follow the instructions in "MSMC on Multiple Individuals" below. **NOTE**: I would recommend doing a first-pass analysis with all individuals separate to make sure that they're all reasonably similar, and then combining multiple individuals in a later run for greater statistical power.

The workhorse of this script is the line

```
generate_multihetsep.py --mask=${MASK_INDIV} --mask=${MASK_GENOME} $VCF > $MSMC_INPUT
```

The output of this script, the MSMC input, has a list of heterozygous sites in the genome, and the distance between these heterozygous sites.

1.5 Step 3 - Run MSMC

Now that we have input files for all chromosomes for an individual, we can run MSMC on these. You will need to edit the `msmc_params` variables to work with your data and the parameters youâd like to use for running the analysis. The things you might want to change are `P_PAR` (the timesteps used for binning the data for anlaysis) and `-i` (number of iterations). I recommend going with the default `P_PAR` to start, and then potentially changing it later depending on how your data look. The scripts for running MSMC can be called as follows:

```
# for a single individual
sbatch msmc_3_runMSMC_slurm.sh
# or without using slurm
./msmc_3_runMSMC.sh
```

The outputs will be stored as `*.loop.text`, `*.log`, and `*.final.txt`. This last file can be imported in to R to plot the results! You'll want to have the mutation rate and generation time ready so that you can scale Ne and time to individuals and years, respectively, like so:

```
### R Script for plotting MSMC Results ###
data <- msmc_output.final.txt

mu <- [mutation rate]
gen <- [generation time]

time <- (data$left_time_boundary/mu*gen)
pop.size <- (1/data$lambda)/(2*mu)
```

```
plot(time, pop.size, type="s",
    xlab="log Years before present",
    ylab="Effective Population Size",
    log="x")
########################################
```

From here, you may need to adjust the time steps and iterations for your MSMC run. There are more details on what these mean, and why you might want to adjust them, in the MSMC guide.

1.5.1 MSMC on Multiple Genomes

To combine multiple individuals, the steps generally follow what is detailed above. However, in step 2, you will instead use the script `msmc_2_generateInput_multiInd.sh`, which will combine all of the individuals from a given population. For this script, which is submitted using `submit_2_multi.txt`, you will need a file with the names of the individuals you want combined in your MSMC run (named POP_IND.txt, where "POP" is your population name).

In step 3, you will then need to change `-I`, which specifies which haplotypes to analyze. For one individual, this is `0,1`, for two individuals `0,1,2,3`, and so on. This will be done automatically in the "multiInd" script, which can simply be run as follows:

```
# for multiple individuals in a population
sbatch msmc_3_runMSMC_slurm.sh numInd popID
# or without using slurm
./msmc_3_runMSMC.sh numInd popID
```

1.5.2 Cross-Coalescence

If you're interested in looking at the split between two populations or species, then you'll want to do things slightly differently, and will need to actually make three MSMC runs. You should generate one combined input file with all of the individuals you want to include in analyses. Then, you'll run MSMC three times, using different indices: once for each of the two populations separately, and once for the two populations combined. The command for running MSMC on two individuals from each of two populations would look something like this:

```
msmc2 -I 0,1,2,3 -o within_pop1 `cat INPUT_LIST.txt`
msmc2 -I 4,5,6,7 -o within_pop2 `cat INPUT_LIST.txt`
msmc2 -P 0,0,0,0,1,1,1,1 -o between_pop1-2 `cat INPUT_LIST.txt`
```

You then need to combine all three of these runs into a combined msmc output file using `combineCrossCoal.py` (from `msmc-tools-master`):

```
python combineCrossCoal.py \
    between_pop1-2.final.txt \
    within_pop1.final.txt \
    within_pop2.final.txt > combined_pop1-2.final.txt
```

This output can then be plotted in `R`, using the same transformations as described above.

1.6 Step 4 - Bootstraps

Once your MSMC plots are looking reasonable, then you can create bootstraps to determine your confidence in your findings. This requires randomly subsampling your data and running MSMC on those subsampled data. To do this, you'll use the `generate_bootstrap.sh` and `msmc_bootstrap.sh` scripts. The first will generate the bootstrap data, and the second will actually run the MSMC program on those subsampled data. NOTE: As written, this is currently tailored for single individuals, and will need to be modified to get bootstraps for multi-individual runs.