Error Analysis of PSMC’

Alex Lee Jackson

June 7, 2016

1 Miscellaneous

For further inquiries, I can be contacted at aj123@internode.on.net, or DM’d on the `rstat` Slack.
GitHub: `https://github.com/alex-jackson1994/errorAnalysis`

2 Objective

Determine if there is a relationship between the level/type of chromosomal fragmentation, population model (predictor variables) and the error (response variable) in PSMC’ (MSMC [6] with two haplotypes). This will probably be analysed using mixed effects models.

```
for Each population model (3) do
    for Each simulated genome (5) do
        for Each model of fragmentation (2) do
            for Each level of fragmentation (5) do
                Run MSMC, record output.
            end
        end
    end
end
Fit statistical model relating predictors to response.
```

Algorithm 1: Outline of the simulation and MSMC analysis process

3 Details

PSMC’ gives an estimate of how populations change over time, by analysing heterozygousity of sites across a genome (note: PSMC’/MSMC used interchangeably). However, real sequenced genomes will not always be nicely sorted into chromosomes. Often they will be assembled into smaller sections called contigs. We will simulate human genomes (as MSMC was originally written to analyse humans) under the following conditions:

- The length L of the genome will be taken from a real human genome. For simplicity, *only autosomes* will be considered.
- Three different population models will be considered. In this document, take time t as going from recent to ancient, i.e. $t = 0$ is the more recent than $t = 100$. These dynamics should be carefully considered, as we want them to fall into the time range where PSMC’ can actually pick them up.
  - Constant: $N_e(t) = N_0$.
  - Exponential decrease: $N_e(t) = N_0 e^{kt}$ for some realistic $k < 0$.

– Bottleneck:

$$N_e(t) = \begin{cases} N_0 & \text{if } 0 < t < t_1 \\ N_0 \exp\left\{\ln(\frac{N_1}{N_0})\frac{t-t_1}{t_2-t_1}\right\} & \text{if } t_1 < t < t_2 \\ N_2 & \text{if } t_2 < t < \infty \end{cases}$$

A quick outline of this potential "bottleneck" model follows. Going from ancient to present, the population sits at constant level N_2. At time t_2, there's a bottleneck, and the population instantaneously drops to N_1. There's exponential growth until the population reaches N_0 at time t_1. From t_1 until present, the population remains constant at N_0. Of course, a different bottleneck model may be desired.

3.1 Potential Whole Genome Simulators

We want a simulator with the following properties.

- Able to simulate exponential growth.
- Able to be converted to a format which can be taken by MSMC.
- Able to simulate linkage disequilibrium (maybe... this may be introducing too much complexity).
- Uses the coalescent model, not forward model (speed concerns?).
- Can handle human genome sizes (in the range of 2.6 Gbp).
- Can be converted to MSMC input format!

A number of simulators were considered.

- msHot-LITE (and ms, and msHot) (`https://github.com/lh3/foreign/tree/master/msHOT-lite`): This was used by Heng Li in his PSMC paper [2], and Schiffels, to simulate genomes. However, it has an upper size limit of roughly 10 Mbp.
- GENOME (`http://csg.sph.umich.edu//liang/genome/`): This doesn't appear to do exponential growth.
- GenomePop2 (`http://acraaj.webs.uvigo.es/GenomePop2.htm`): This can apparently output in ms-like format. Might go back to it later...
- **MaCS** (`https://github.com/gchen98/macs`): Looking at this right now. It CAN handle the 2.6 Gbp sizes needed, as well as the other things I believe. Also, it's very fast.
- **scrm** (`http://scrm.github.io/`): Recommended by Schiffels, but I haven't looked at it.

3.1.1 MaCS

For full detail, please refer to the MaCS README (`https://github.com/gchen98/macs/blob/master/README`), and the manual (`http://home.uchicago.edu/rhudson1/source/mksamples/msdir/msdoc.pdf`) for ms [1], from which MaCS is based upon. The MaCS README is fairly sparse, so I'd advise familiarising yourself with the relevant parts of the ms manual.

The MaCS README contains two examples of running MaCS. I'd advise taking the random seed, which is included in `stderr`, and storing it from each run. Alternatively, you could just specify the seed and store that each time. Reproducibility is important. From what I can gather, both `stderr` and `stdout` give useful things to use - simulated output goes to `stdout`, while tree statistics and other things go to `stderr`.

Further information can be found by running `./macs`. Note that:

- `<samplesize>` should always be 2, as we are simulating a diploid genome.
- `<region in base pairs>` would be of the order of 2.6 Gbp. You should get this exactly from an empirical genome.

Switches of interest are: `-s` (sets seed from the Mersenne Twister algorithm), `-t` (sets mutation rate in units of mutations per site per $4N_0$ generations), `-r` (sets recombination rate in units of recombinations per site per $4N_0$ generations), `-T` (outputs in Newick format), `-eG` (sets growth rate, presumably in N_0 units), and `-eN` (sets population size in N_0 unis, sets growth rate to 0).

MaCS output should be able to be converted into the ms format via the included script `msformatter`. I've run this without errors, but haven't confirmed the ms format can then be converted to MSMC input format.

To convert from ms format to MSMC input format, use the script `ms2multihetsep.py`, included in the GitHub `https://github.com/stschiff/msmc-tools`. I haven't tried to use this yet.

3.2 Simulation Parameters

We will need the following parameters for the simulation.

- Genome size: obtaining this from an empirical human genome would be ideal. Remember, we are *only considering autosomes.*
- Mutation rate: Julien suggests 1.25×10^{-8} mutations per bp per generation is the current best estimate.
- Recombination rate: I don't know a good estimate.
- Population models (this defines N_0 as well). We originally had Julien give us suggestions for times/effective sizes for a constant-bottleneck model. However, I suggest you guys all sit down and figure out exactly where you would like past demographic events to happen (as we want to make sure MSMC can pick them up).

Apart from the "chromosome-level" of fragmentation, fragmentation models will have to be discussed. This includes finding a relevant empirical contig distribution, and some mathematical model for chromosomal fragmentation (e.g. Poisson). Then you'll have to decide whether to do the fragmentation on the MaCS output, ms format, or the MSMC input.

3.3 File Names And Data Frames

This is up to you, but when we did the (unsuccessful) PSMC analysis we used the file names to store information about the simulation parameters. For example, say we had a MSMC output file corresponding to population demographic "Constant", simulated genome 2, fragmentation model "Empirical", and fragmentation level "3". Then the output file would be called something like `Constant_2_Empirical_3.txt`. We could then use regular expressions in R to extract "Constant" etc. for storing in a data frame. Of course, there are lots of other ways it could be done!

See Table 1 for an example of data frame we could use for storing results.

3.4 Error Analysis

3.4.1 Definition Of Error

We define the *error rate* $\Delta(s_1, s_2)$ between two functions $s_1(t)$ and $s_2(t)$ as

$$\Delta(s_1, s_2) = \int_{t_R}^{t_A} |s_1(t) - s_2(t)|\, dt, \tag{1}$$

Table 1: Example data frame for storing analysis results. You might also want to add some summary statistics e.g. mean/variance of contig lengths.

Pop. Dem.	**Sim. Number**	**Frag. Model**	**Frag. Level**	**Error**
Constant	1	Empirical	1	2303
Constant	1	Empirical	2	4054
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ExpDecr	5	Poisson	5	1406

for sensible limits of integration t_R (recent) and t_A (ancient). Remember, treat recent time as time near 0, and ancient time as time towards ∞. The interpretation of this is the absolute value of the area between the two curves, in the interval (t_R, t_A). Consult with Ben on choosing these values - a suggestion was the 95% confidence interval for the time of coalescence between two haplotypes.

The limits of integration T_R and t_A should definitely ignore any data from the first time interval, and the last time interval. This is because Schiffels says [6] "the limits on the inference are given by the second and second-to-last boundaries, which correspond to the 2.5% and 97.4% quantile boundaries for the distribution of first coalescence times."

Is error going to be evaluated on a log scale?

Are you going to compare fragmented results to the "truth" (simulation parameters of population which we specify), or to the "best you can do" (what MSMC recovers from the chromosomal level of fragmentation)?

3.4.2 Calculating Errors

Over the 2015/16 summer, the main script we used for calculating errors in PSMC was the following.

```
~/Dropbox/MAGenomics_2015/PSMC_Scholarship/alexScripts/PSMC_Regression/
    StringExtraction_ErrorAnalysis.R
```

While this is optimised for the specific task we were doing, parts of it could be adapted for a MSMC analysis.

The main two functions of interest in the script are `eval_popsize` and `absdifference`.

The function `eval_popsize(pos, x, y)` takes a set of discrete points e.g.

$$\{(t_1, N_e(t_1)), (t_2, N_e(t_2)), \cdots, (t_n, N_e(t_n))\}$$

where `x` is a vector of $\{t_i\}$ and `y` is a vector of $\{N_e(t_i)\}$. Given a time point `pos`, the function will return the corresponding population size, if the $\{(t_i, N_e(t_i))\}$ were connected by a step function. This allows us to construct the stepwise constant functions we see when MSMC graphs are plotted (see Figure 1).

The function `absdifference(xpos,d1,d2,d3,d4)` returns the absolute difference between two MSMC stepwise constant functions A and B, evaluated at point `xpos`. The other inputs to this function are `d1` and `d2` (vectors of $\{t_i\}$ and $\{N_e(t_i)\}$ respectively, for A), and `d3` and `d4` (vectors of $\{t_i\}$ and $\{N_e(t_i)\}$ respectively, for B). `absdifference` uses the `eval_popsize` function.

One could then use R's integration function on `absdifference` to compute the integral that defines the error rate (Equation 1).

Note that if you want to compare a "true" population model that involves curves (e.g. exponential growth) to MSMC output, the `absdifference` function may not be so useful. It might be better to write a new `absdifference` function which takes a MSMC stepwise constant function A as one input, and a curve function B as the other. However, this will be fine if comparing two MSMC curves (e.g. comparing the "gold standard" chromosomal fragmentation, with a much more fragmented genome).

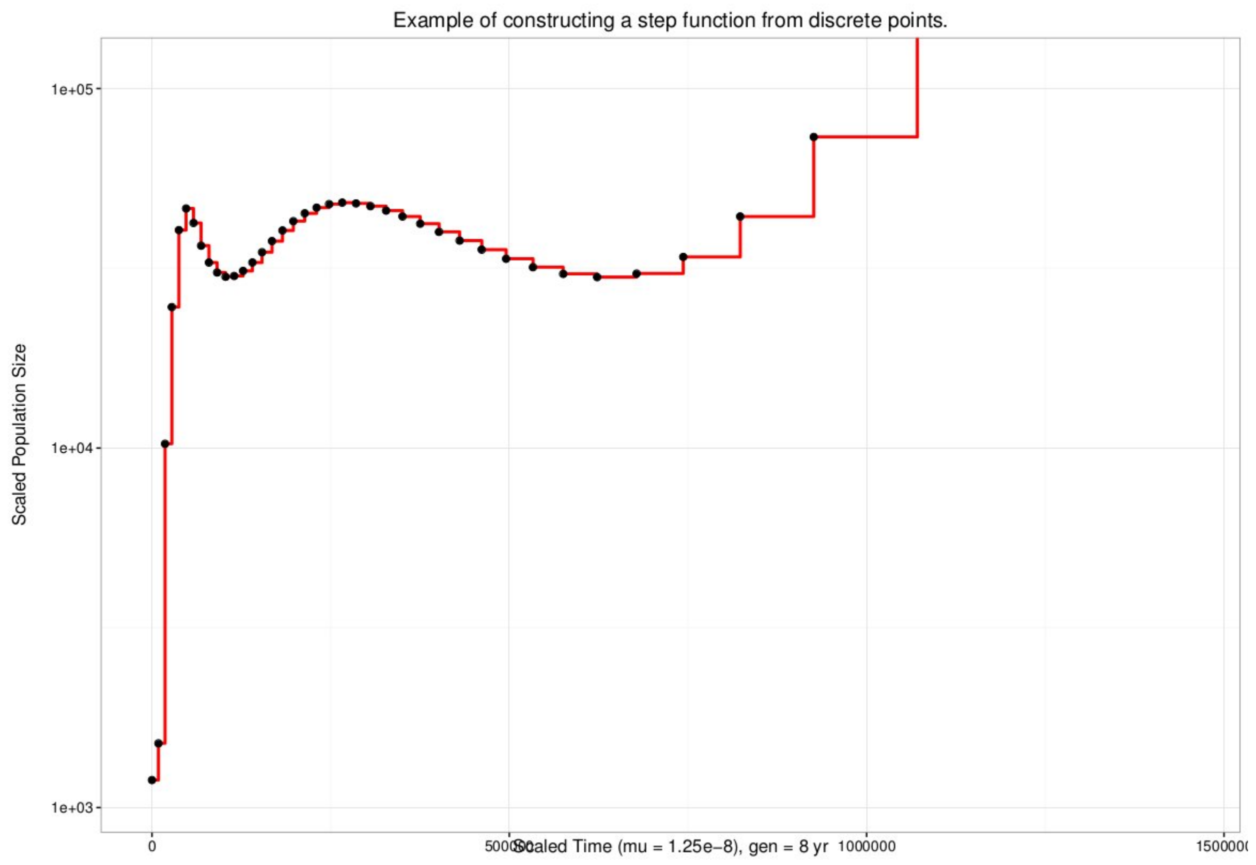

Figure 1: The function `eval_popsize` allows us to construct a step function (red) from discrete points (black).

3.5 Running MSMC

3.5.1 Theory

MSMC splits time from 0 (recent) to ∞ (ancient) into discrete unit time intervals. It then makes an estimate of population in each time interval. It then makes an estimate of the effective population in each interval (based on the heterozygousity of the genome).

The intervals are determined using the formula

$$t_i = \frac{-\ln(1 - \frac{i}{n})}{\binom{M}{2}} \tag{2}$$

for $i = 0, 1, \ldots, n$. Here, t_i is the time boundary of the particular interval, n is the number of intervals (which you can define when running `msmc`), and M is the number of haplotypes. As we will be looking at diploid individuals, the formula reduces to $t_i = -\ln(1 - \frac{i}{n})$.

I am still a bit confused about exactly what units MSMC outputs its times in. Sorry!

The unit intervals can be combined. For example, if you were running MSMC using the command

```
msmc -p 30*1 [...]
```

then this would be instructing the program to use 30 unit intervals ($n = 30$). If you used

```
msmc -p 10*1+15*2 [...]
```

then the program would use 10 unit intervals followed by 15 double-unit intervals ($n = 40$) (this also happens to be the default pattern). The left of `10*1+15*2` corresponds to recent time and the right corresponds to ancient time and the right corresponds to ancient time.

In the MSMC and PSMC papers, I have not found any justification for the use of particular combinations of time intervals.

I have found that if you scale MSMC time output such as in `plotMSMC_PSMC.R`, you will *not* get the same times as you do when you rescale the numbers you take from Equation 2. To get the same times, you need to rescale by MSMC's estimate of mutation rate (which you should be able to find as `mutationRate` in the `log` file outputted by MSMC).

If you are interested, the recombination rate estimate `recombinationRate` can also be found in the `log` file.

3.5.2 Using Real Data

See attached example `runMSMC.txt`, which uses the Tasmanian Devil data. Julien should also be able to help with this. This will give a quick run-through of what the different commands are doing.

```
awk '{print $1, 0, $2, $1, "0", "+"}' [...]
```

MSMC requires individual text input files per chromosome or contig. In this case, we look at the Devil `fasta` reference file to find the largest contigs to use. In this particular case, we only decided to use the contigs that were larger than 5 Mbp.

```
module load python/4.8.0/3.4.1 gnu/4.9.2 zlib/testing samtools/1.2 bcftools/1.2 msmc
    /20150413 htslib/1.2.1 parallel
```

Load the relevant modules.

```
parallel -j7 --nice 19 "samtools mpileup -EA -Q 20 -C 50 -u -r {} -f /localscratch/Refs/
    Sarcophilus_harrisii/Devil7_0_Raw [...]
```

You and Julien probably have a better idea of this than me. This creates the MSMC input files. For a different genome, obviously the names would have to be different. A `bam` genome and a `fasta` reference are required.

The next part is a copy from one of my files,

```
/localscratch/jsoubrier/Bison_Genomes/CowRef/AlexPSMCMSMC_Comparison/labBook
```

and should highlight relevant MSMC commands. Further information can be found on the GitHub guide (`https://github.com/stschiff/msmc/blob/master/guide.md`) or by running `msmc` on its own.

```
msmc -t 50 -p 40*1 -i 20 -o EuropeanBison.Cow_UMD3_1.realigned_AllAutosomes_MSMCOutput
    EuropeanBison.Cow_UMD3_1.realigned_chr1_MSMCInput.txt EuropeanBison.Cow_UMD3_1.
    realigned_chr2_MSMCInput.txt [...] | at now
```

The switches of interest are: `-t` (sets number of threads to run on), `-p` (set how MSMC chooses and combines its time intervals), `-i` (number of iterations... I don't now a good way to choose this, I've been using 20), and `-o` (set the name of the output file). The `--fixedRecombination` flag should be left out for running on two haplotypes (our situation), according to the guide.

Following the name of the output file are the MSMC input files (there were 29 in this case). This can make the commands long and messy. Finally, the `| at now` allowed me to leave this running on the server and then log off.

3.5.3 Using Simulated Data

I am not sure how to do this, but presumable it would go something like this:

1. Run MaCS (or similar) with the `-T` flag to output in the appropriate format.
2. Convert this to ms output format with the `msformatter` script which comes with MaCS. See A.1 for more detail on this file format.
3. Convert this to MSMC input format with `ms2multihetsep.py` included with Schiffels' MSMC tools.
4. (Fragment somewhere in above.)
5. Run MSMC as above.

3.6 Visualising MSMC

See the attached example file `plotMSMC_PSMC.R` for an annotated R file which runs through plotting MSMC (and PSMC). Just remove the parts relevant to PSMC if you only care about MSMC.

4 Other Suggestions

4.1 Reduction In Complexity

Julien suggested that we could do some trials to see if we recover the same dynamics with say 5 chromosomes, instead of the full 23. This would significantly decrease the run-time of the whole progress. However, fragmenting 5 chromosomes may be quite different to fragmenting 23 chromosomes.

4.2 Looking At Bias

We are more looking at the variability in the MSMC estimate. To look at bias, this would probably require a different definition of error rate.

4.3 Hybridisation

TBD.

5 PSMC

5.1 Background

The Pairwise Sequentially Markovian Coalescent (PSMC) model [2] is a method for estimating past population dynamics based on the genome of a single individual. PSMC uses the heterozygousity of the genome to estimate where recombination events have occurred in the genome. These estimates are found using a Hidden Markov Model (for an introduction to HMMs, see the paper by Rabiner [5]). A genome is split into many consecutive, non-overlapping 100 bp "bins". The observations are whether each of the bins contains a heterozygous pair ("1"), is homozygous ("0"), or data is missing ("."). The state space is the set of non-overlapping time intervals from the present back in time. The hidden state is the time interval into which the TMRCA for each bin falls. This allows PSMC to then construct a population estimate over a period of time.

For further detail, see the paper by Li [2] (the Methods section in particular gives a starting point) as well as the supplementary information for detail, and the package's README at `https://github.com/lh3/psmc`.

5.2 PSMC vs MSMC

The key differences between PSMC and MSMC are as follows:

- PSMC can only be run on a diploid genome, while MSMC can be run on more than two haplotypes.
- PSMC partitions the genome into 100 bp "bins". Each bin is then assigned either heterozygous or homozygous (or missing). MSMC does not need this, as its software is more efficient.
- MSMC allows for a "null recombination" event, which if I've interpreted Ben correctly, is a recombination which immediately coalesces back into the same branch... or something. PSMC ignores this.
- PSMC and MSMC have different ways of splitting up time from 0 to infinity. MSMC's formula is

  $$t_i = \frac{-\ln(1-\frac{i}{n})}{\binom{M}{2}}$$

  as mentioned above. PSMC's formula is

  $$t_i = 0.1 \exp\left\{\frac{i}{n}\ln(1+10T_{\max})\right\},$$

  for $i = 0, 1, \ldots, n$. The parameter $T_{\max}$ is the maximum TMRCA, which can be specified when running `psmc` using the `-t` switch (in units of $2N_0$, the default is 15). PSMC output is scaled to $2N_0$ - unfortunately, I'm not sure what the units are in the above equation.

  For an equivalent amount of time intervals, PSMC's time intervals reach more into recent time than MSMC's do. Nigel thinks this is because recent estimates are rubbish, so MSMC effectively ignores them.
- Apparently, MSMC estimates recombination rate better than PSMC.
- There are probably also some subtle mathematical differences which Ben and Jono will have to go over when they analyse the equations.
- The underlying model of PSMC is the SMC model [4], while the underlying model of PSMC' is the SMC' model [3].

5.3 Running PSMC On Real Data

Similarly to MSMC, we start with a `fasta` and a `bam`. We use samtools and related software to produce a `fastq` output file. This uses the kangaroo as an example.

```
samtools mpileup -EA -Q20 -C50 -u -f Macropus_eugenii.Meug_1.0.dna_rm.toplevel.fasta
    Kangaroo.Meug_1_0.realigned.bam | bcftools call -c | vcfutils.pl vcf2fq | gzip >
    mpileupedKangaroo.fq.gz
```

We then convert the `fastq` to the PSMC input format, `psmcfa` (see Appendix A.2 for detail on this format). The `fq2psmcfa` script is included with PSMC's utils.

```
fq2psmcfa redKangaroo.fq.gz > redKangaroo.psmcfa
```

We then run PSMC (here using the bison example). Switches used here are `-p` (specifies number of time intervals and how they're combined, like in MSMC) and `-N`, the number of iterations.

```
/localscratch/Programs/psmc/psmc -p 40*1 -N 20 EuropeanBisonFullGenome29Autosomes.psmcfa
    > EuropeanBison29Autosomes_Int40*1_Iter20.psmc | at now &
```

We still need to extract the data from PSMC if we wish to plot it, or analyse the error. Shaun wrote a script, `removeDataFromPSMC.sh`, which does this for us (in the `scripts` folder). It is run as follows. You will still need the `psmc` file to obtain θ_0, which is required for plotting PSMC output (see the example `plotMSMC_PSMC.R`).

```
./removeDataFromPSMC.sh PSMC/EuropeanBison29Autosomes_Int40*1_Iter20.psmc > PSMC/
    EuropeanBison29Autosomes_Int40*1_Iter20.psmcData
```

5.4 Running PSMC On Simulated Data

If you simulate your genome in the `ms` format, Heng Li has written the script `ms2psmcfa.pl` to convert to the `psmcfa` format. This is included in the PSMC utils.

If you use `msHOT-lite` (probably not recommended) to run your simulation, you need to use the `-l` switch to ensure the format is correct for PSMC. If you use MaCS or something similar, I'm not sure of what you'll have to do. MaCS' `msformatter` script might be adequate.

```
msHOT-lite 2 1 -t 60000 -r 10000 1000000 -eN 1 2 -eN 2 4 -l > output.ms
perl ms2psmcfa.pl input.ms > output.psmcfa
```

5.5 Goodness of Fit, And Correcting for False Negatives

Li [2] has devised methods for checking goodness of fit, but I don't understand how to test this. In Section 2.2 of the Supplementary Information, he gives detail. There does not appear to be a script for extracting the statistics though.

In the S.I., Li says that you can check $C_\sigma \pi_k$ for overfitting. If it's small (e.g. less than 20), you shouldn't trust the estimates of λ_k. I'm not sure how to get this value though.

In Appendix II of the PSMC README file, Li talks about correcting for false negatives in the case of low coverage (for example, the kangaroo only had 55% coverage). I'm not sure how to estimate this.

5.6 Scripts Which Might Be Useful

A File Types

Thanks to Shaun Barker, who wrote this over the summer.

Name	Description
`combinePsmcfa.sh`	Combines all contigs in a `.psmcfa` file to create one giant contig!
`findNLargestContigs.py` *	From a `psmcfa` file, creates a new `psmcfa`, only containing data from the `N` largest contigs (`N` user-specified) Uses `python3`.
`growthDataSlidingWindowError.R`	Does a sliding window-style error analysis.
`removeDataFromPSMC.sh` *	Extracts time, population estimates etc. from `psmc` files, in a tab-delimited format.
`removeRandomContigs.py`	Removes contigs randomly from a `.psmcfa` file.
`removeRandomPartsFromPsmcfa.py`	Removes lines randomly from a `.psmcfa` file.
`StringExtraction_ErrorAnalysis.R`	Extracts information about the specific simulation with regular expressions. Runs error analysis and creates data frame. Well commented, but very messy though.

A.1 MS (.ms)

MS (.ms) files are simulated output from msHOT-lite (as well as MS, SCRM and other MS derivatives). It was written by Heng Li (who also created the PSMC software and others software used in bioinformatics). It contains a list of genome fragments (e.g. chromosomes/contigs/scaffolds), and in each of those fragments it lists the positions of heterozygous pairs as well as the type. The file begins with the call that created the file then each chromosome generated is listed. Each chromosome is stored between a `@begin` and `@end` statement. An example of an MS file can be seen in Figure 2. For further detail, see the ms paper by Hudson REF and the msHOT-lite repository at .

```
./msHOT-lite 2 1 -t 30000 -r 6000 300000 -eN 0.01 0.1 -eN 0.06 1 -eN 0.2
    0.5 -eN 1 1 -eN 2 2 -l
//
@begin 11765
300000
37295  10
81727  01
104834 01
131283 01
191522 10
298343 01
@end
```

Figure 2: An example of a MS file.

A.2 PSMC Input Files (.psmcfa)

PSMC input files (.psmcfa) are plain text files (see Figure 3) generated by many of the utilities provided alongside PSMC. They are the only input files required by PSMC. The .psmcfa files contain a list of chromosomes/contigs/etc. and 60 character wide lines that denote where heterozygous pairs occur. Each character in the lines represent the bin $[100i, 100i + 100)$ along the sequence, for each i sequential non-overlapping 100bp bin along length of the sequence. The character is K if there is a heterozygous pair in that bin of the sequence, T if the bin is completely made of homozygous pairs, and N if the data is too low quality or missing.

```
>thisIsTheFirstSequenceName
TKTKTKKTKTKTKTKTKKTKTKTKKKKKKKKKTTTKKTKTKTKTKTKTKKTKTTKTKTKTK
KKKKTKKKKTKKKKKKKKTTTTTTTTTKKTKTKKKKKKKKKKKKKKKKKKKKKKKKKKKKTT
TTTTTTTTTTTTTKKKKKKKKKKTKKKKTKKKKKKKKKKKKKKKKKKKKKTTKKKKKKKKKK
TKKKTKTKKTKTKKTKTTTKTKTKTKTKTTKKTTKTKTKKKTKTKTKTKKTTTKTKTTKK
>thisIsTheSecondSequenceName
KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTKTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTKTTTTTTTTTTTTTTTTTKTTTTTTTTTTTTTTTTTTTTTTTTTTTKTTTT
```

Figure 3: An example of a .psmcfa file.

A.3 PSMC Output Files (.psmc)

The .psmc output files are plain text files outputted by PSMC. They contain details on the iterations performed by the program and the values. They all have a header describing the file itself (see Figure 4) and this should be read to attempt to understand the output. The final table in the .psmc file is the "final" result after 20 iterations of expectation-maximisation, and can be extracted with the `removeDataFromPSMC.sh` script.

Note: According to a comment by Heng Li at `https://hengli.uservoice.com/forums/152783-general/suggestions/6787405-shouldn-t-the-pi-k-column-in-the-psmc-output-be-be`, the `\pi_k` column is not actually `\pi_k`. It may be `\sigma_k` instead.

```
CC
CC   Brief Description of the file format:
CC     CC  comments
CC     MM  useful-messages
CC     RD  round-of-iterations
CC     LL  \log[P(sequence)]
CC     QD  Q-before-opt Q-after-opt
CC     TR  \theta_0 \rho_0
CC     RS  k t_k \lambda_k \pi_k \sum_{l\not=k}A_{kl} A_{kk}
CC     DC  begin end best-k t_k+\Delta_k max-prob
```

Figure 4: The header description of a .psmc file.

The output files generated by PSMC are a (roughly) tab delimited text format, where the first two characters on each line defines what values are on the line. The output files always contain a header which gives some information on the format, and then parameter estimates for each iteration of the Baum-Welch algorithm used in PSMC. The last table contains the estimates from the final iteration, this is the data we used for our analysis. The last table can be extracted as a tab delimited table using the script `removeDataFromPSMC.sh`. This script takes a `.psmc` output file as its input and then outputs the final table in the file as a tab delimited text file.

PSMC gives scaled output [2]. The time t_k is in terms of $2N_0$ generations. The population given is in terms of $\lambda_k N_0$, which is the proportion relative to N_0. N_0 is defined as "effective population size", but what that means specifically is unclear. N_0 can be obtained by the formula $N_0 = \theta/(4\mu s)$.

References

[1] Richard R Hudson. Generating samples under a wright–fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.

[2] Heng Li and Richard Durbin. Inference of human population history from individual whole-genome sequences. *Nature*, 475(7357):493–496, 2011.

[3] Paul Marjoram and Jeff D Wall. Fast" coalescent" simulation. *BMC genetics*, 7(1):1, 2006.

[4] Gilean AT McVean and Niall J Cardin. Approximating the coalescent with recombination. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 360(1459):1387–1393, 2005.

[5] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[6] Stephan Schiffels and Richard Durbin. Inferring human population size and separation history from multiple genome sequences. *Nature genetics*, 46(8):919–925, 2014.