Chapter 7

MSMC and MSMC2: The Multiple Sequentially Markovian Coalescent

Stephan Schiffels and Ke Wang

Abstract

The Multiple Sequentially Markovian Coalescent (MSMC) is a population genetic method and software for inferring demographic history and population structure through time from genome sequences. Here we describe the main program MSMC and its successor MSMC2. We go through all the necessary steps of processing genomic data from BAM files all the way to generating plots of inferred population size and separation histories. Some background on the methodology itself is provided, as well as bash scripts and python source code to run the necessary programs. The reader is also referred to community resources such as a mailing list and github repositories for further advice.

Key words Demographic inference, Complete genome sequencing, Phasing, Population structure, Coalescent modelling

1 Introduction

1.1 MSMC

MSMC [1] is an algorithm and program for analyzing genome sequence data to answer two basic questions: How did the effective population size of a population change through time? When and how did two populations separate from each other in the past? As input data, MSMC analyzes multiple phased genome sequences simultaneously (separated into haplotypes, i.e. maternal and paternal haploid chromosomes) to fit a demographic model to the data.

MSMC models an approximate version of the coalescent under recombination across the input sequences. Specifically, the coalescent under recombination is approximated by a Markov model along multiple sequences [2, 3], which describes how local genealogical trees change due to ancestral recombinations (Fig. 1).

These local genealogies as well as the recombination events are of course invisible and therefore act as latent variables that are to be integrated out of the joint probability distribution. Since it is infeasible to do this integration across the entire space of possible

Julien Y. Dutheil (ed.), *Statistical Population Genomics*, Methods in Molecular Biology, vol. 2090,
https://doi.org/10.1007/978-1-0716-0199-0_7,

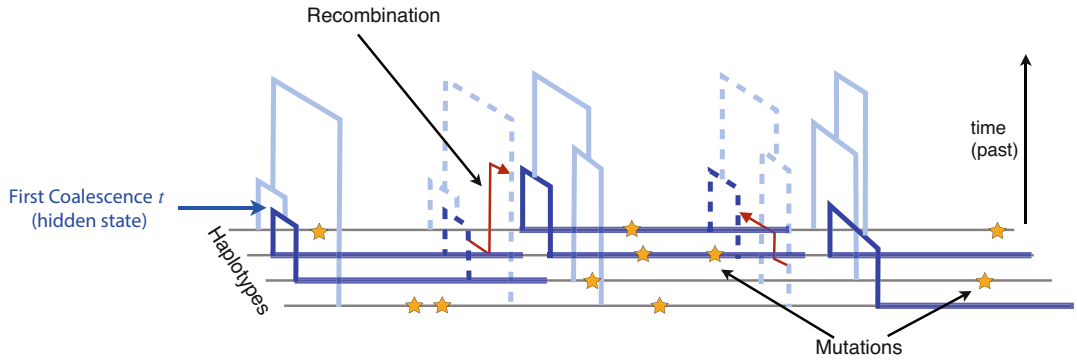

Fig. 1 Schematic description of MSMC as a hidden Markov model along multiple sequences. The sequences are related by local genealogical trees that change due to ancestral recombination events. The trees and recombination events are hidden states of the model and can be probabilistically inferred from the patterns of mutations

trees, MSMC focuses only on one particular aspect of those trees: the first coalescence event. This variable (dark blue in Fig. 1) acts as a hidden state in the Hidden Markov Model (HMM). Using standard HMM algorithms, the hidden state (trees and recombination events) can be integrated out efficiently using dynamic programming. We can thus efficiently compute the likelihood of the data given a demographic model, and iteratively find a demographic model that maximizes this likelihood.

The demographic model itself is—in the simplest case of just one population—parameterized by a sequence of piecewise constant coalescence rates, i.e. inverse effective population sizes. The time segments are chosen such that they cover the distribution of times to first coalescence. Therefore, the more sequences are analyzed, the more recent the window of analysis will be (Fig. 2).

If the input individuals come from two populations, the demographic model is parameterized by three coalescent rates through time: A coalescence rate between lineages sampled within the first population, a coalescence rate between lineages sampled within the second population, and a coalescence rate between lineages sampled across the two populations (Fig. 3a). As introduced in Schiffels and Durbin [1], to simplify interpretation of the three inferred rates, we can plot a simple summary by taking the ratio of the across-rate and the mean within-rate, which is termed the relative cross coalescence rate (rCCR) (Fig. 3b). This summary variable ranges between 0 and 1, and indicates when and how the two populations diverged. Values close to 1 indicate that the two populations were really one population at that time. At the time when the rCCR drops to zero, the two populations likely separated into two isolated populations. Heuristically, the mid-point of that

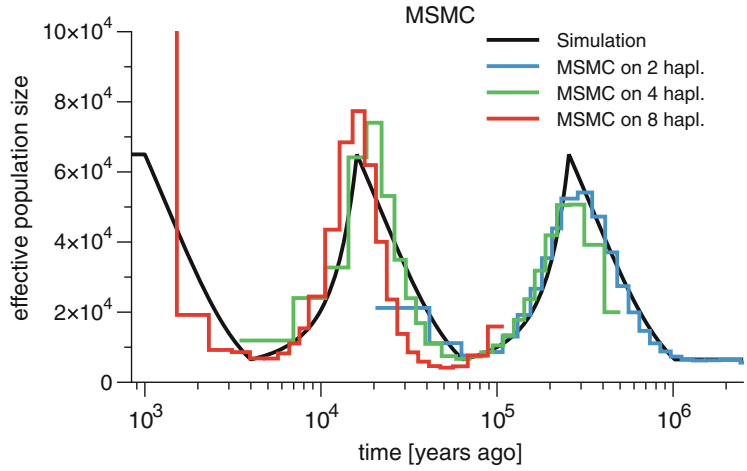

Fig. 2 Population size inference with MSMC from simulated data. Time segments are chosen to cover the distribution of first coalescence times. They cover younger time segments if more sequences are analyzed

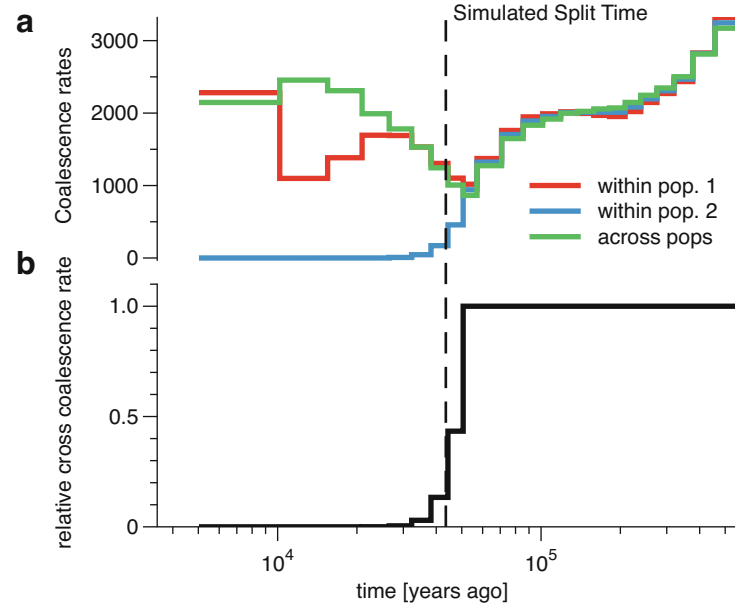

Fig. 3 Studying divergence processes through the cross-coalescence rate. When sequences are sampled from two different populations, MSMC estimates not one but three coalescence rates (two within and one across populations) over time (**a**), here for a simple scenario of two populations with a clean split about 43,500 years ago (1500 generations with generation time of 29 years). The split time is indicated by the dashed black line. The relative cross coalescence rate (**b**) is the cross coalescence rate divided by the mean within-rate. As it drops from 1 to zero (forward in time), it indicates when the two populations split. The drop agrees well with the simulated split time

decline (i.e., the time when the rCCR hits 0.5) is often taken to be an estimate for the split time between the two populations.

MSMC has been widely applied to human data (for example [4–8]) and non-human organisms (for example [9–13]).

1.2 MSMC2

MSMC2 is a newer algorithm, and the tool is still actively being developed. A first version was used in Malaspinas et al. [6] for analyzing Australian genomes. At the time of this writing, a manuscript that presents the new algorithm in more detail is in preparation. MSMC2 was developed to overcome some problems that we saw with MSMC. In particular, MSMC is computationally intensive, and for all practical purposes limited to analyzing eight haplotypes at most. But even within this scope, we see that coalescence rate estimates for more than four haplotypes are sometimes biased (see, for example, Fig. 2, red curve), with some systematic over- and underestimations of the true coalescence rates. These biases are in part caused by approximations in the emission rate of the HMM, which requires knowledge of the local lengths of leaf branches of trees. This variable is estimated by a separate HMM that is heuristic and cannot easily be improved, and which apparently performs poorly for larger trees. This means that even if we improved the computational aspects, we could not scale up this algorithm easily to more haplotypes.

MSMC2 takes a step back from these complications and approaches the problem of modelling multiple samples in a much simpler way: Instead of analyzing all input haplotypes simultaneously, it uses a much simpler pairwise HMM (very similar to PSMC) on all pairs of haplotypes. The likelihood of the data is then simply multiplied across all pairs as a composite likelihood. This has two interesting consequences: First, the pairwise model is—in contrast to the MSMC—an exact model under the Sequentially Markovian Coalescent, and does not suffer from biases with increasing number of genomes. Second, the pairwise model describes the entire distribution of pairwise coalescence times, not just the time to first coalescence. MSMC2 can therefore estimate coalescent rates across the entire distribution of pairwise coalescence times, with increasing resolution in more recent times, and importantly without biased estimates (Fig. 4). In contrast, MSMC loses power in ancient times with increasing numbers of input genomes (*see* Fig. 2).

MSMC2 can also analyze population separations via the relative cross coalescence rate, and gives similar results as MSMC, but with computational improvements, as we will point out further below.

We caution that at the time of writing, MSMC2 is still in beta and some aspects of the interface and algorithm may still change. Nevertheless, we will cover its use throughout this chapter alongside MSMC.

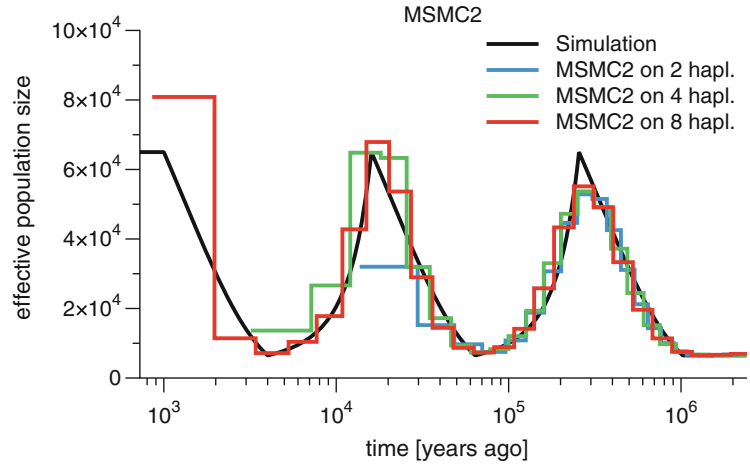

Fig. 4 MSMC2 population size estimates. Time segments are covering more recent times with increasing number of input haplotypes, without losing power in ancient times (compare with Fig. 2)

2 Software Overview

MSMC has been implemented in three open source software packages, summarized in the following. A mailing list for discussions around all three packages exists under https://groups.google.com/forum/#!forum/msmc-popgen

The main program is written in the D programming language (www.dlang.org).

A tutorial can be found at https://github.com/stschiff/msmc-tools/blob/master/msmc-tutorial/guide.md and general documentation can be found within each package.

2.1 MSMC

The main program used in the original publication [1] is accessible at http://www.github.com/stschiff/msmc. Pre-compiled packages for Mac and Linux can be found under the Releases tab. For compilation from source code, a D language compiler is needed (see www.dlang.org for details).

2.2 MSMC2

MSMC2 (*see* Subheading 1) can be accessed at http://www.github.com/stschiff/msmc2. MSMC2 is still under development, but has been used in a key publication [6], which can be used to cite this program. A publication describing the novel aspects and comparison to other state-of-the-art methods is in preparation at the time of this writing.

2.3 MSMC-Tools

Utilities for preparing input files for MSMC, as well as some other tasks, can be found in a separate repository at http://www.github.com/stschiff/msmc-tools and mainly contains python scripts that help with generating the input data and with processing the output data.

2.4 Data Requirements

MSMC normally operates on diploid, phased, complete, high coverage genomes. Here we discuss these conditions one by one.

2.4.1 Diploid Data

Technically, it is not a strict condition that input sequences be diploid. However, most populations/organisms that are not diploid do not follow a coalescent under recombination. For example, bacteria and viruses are asexual without recombination, which breaks several key assumptions that the MSMC model makes.

In some diploid model organisms, inbred lines are available and sequenced (for example, in Drosophila). Such inbred lines are effectively haploid, but originate from a diploid outbred population. In this case we think MSMC should work OK, by using each homozygous haploid input genome as a single "haplotype," although we lack explicit experience and overview of potential caveats in this case.

2.4.2 Phasing

When sequencing diploid genomes, modern sequencing platforms generate unphased data, which randomly permutes the association of heterozygous alleles to the paternal and maternal haplotypes. For MSMC, knowledge of the paternal vs. maternal allele is important when more than two haplotypes are analyzed. Note that for a single diploid genome as input (i.e., two haplotypes), no phasing is necessary.

Phasing can be a laborious preprocessing step, which requires external tools, such as shapeit (https://jmarchini.org/shapeit3/) or beagle (https://faculty.washington.edu/browning/beagle/beagle.html). As a general rule, what helps phasing quality a lot are:

- availability of a reference panel of phased populations
- presence of related individuals (e.g., parent–child duos or father–mother–child trios)
- long sequencing reads
- long-insert libraries in combination with paired-end sequencing.

Note that MSMC and MSMC2 can in principle handle unphased data within the input data format (see below), but for some analyses we recommend to exclude those sites from the analysis, which can be done within MSMC. Note also that MSMC2 now can optionally run on unphased genomes for population size analysis, but not for population separation analysis. As described below, this is achieved by running the MSMC2-HMM only within each diploid genome, but not across pairs of genomes. This will give lower resolution than with phased data, but may be a good compromise if phasing is not possible and only population sizes need to be estimated.

2.4.3 Complete Genomes

MSMC and MSMC2 cannot run on Array data, with selected SNPs, but require contiguous sequence segments. For many organisms, genomes are shorter than in humans, and from our experience, MSMC still works fine for much smaller genomes, but we recommend in these cases to run simulations with shorter genome length and specified heterozygosity to test performance of the program on shorter genomes.

For many non-model organisms, reference genomes are only available via assembly scaffolds, which are sometimes as short as a few hundred thousand basepairs (compared to hundred million basepairs for a human chromosome). In our experience, MSMC works still fine in many such cases, as long as scaffolds are not too short. Although the exact threshold depends on an organisms mean heterozygosity, in my experience scaffolds on the order of 500 kb and longer often work OK. We again recommend simulations of short chromosomes to assess the power in those cases.

2.4.4 High Coverage Data

MSMC requires good resolution of heterozygous vs. homozygous genotypes across the genome, which is only available with high coverage sequencing data. In our experience, 20-fold coverage and higher is sufficient. MSMC may work on lower coverage data as well, but detailed analyses of the effects of false negative/positives in genotype calling need to be assessed in these cases, ideally again through simulated data, into which sequencing errors are randomly introduced to test their effect on the estimates.

3 Input Data Format

MSMC/MSMC2 take several files as input, one for each chromosome, each with a list of segregating sites, including a column to denote how many sites have been called since the last segregating site. Note that here we use the term "chromosomes" to refer to coordinate blocks in a reference genome (which could also be an assembly scaffolds). We use the term "haplotypes," when we refer to the phased input sequences from multiple individuals. Here is an example part of an input file for chromosome 1 for four haplotypes (two diploid individuals):

```
1  58432   63     TCCC
1  58448   16     GAAA
1  68306   15     CTTT
1  68316   10     TCCC
1  69552   8      GCCC
1  69569   17     TCCC
1  801848  9730   CCCA
1  809876  1430   AAAG
1  825207  1971   CCCT,CCTC
1  833223  923    TCCC
```

The four (tab-separated) columns are:

1. The chromosome (can be an arbitrary string, but has to be the same for all rows in a file).
2. The position on the chromosome.
3. The number of called homozygous sites since the last segregating site, which includes the given location. This number must always be greater than zero and cannot be larger than the difference between the current position and the previous position.
4. The ordered and phased alleles of the multiple haplotypes. If the phasing is unknown or only partially known, multiple phasings can be given, separated by a comma to indicate the different possibilities (see the second-last line in the example). Unknown alleles can be indicated by "?", but they can also simply be left out and expressed by a reduced number of called sites in the line of the next heterozygous site.

The third column is needed to indicate where missing data is. For simulated data, without any missing data, this column should simply contain the distance in bp from the previous segregating site, indicating that all sites between segregating sites are called homozygous reference, without missing data. To the extent that this number is lower than the distance from the previous site do the input data contain missing data. Information about missing vs. homozygous reference calls is crucial for MSMC: If, for example, missing data is not correctly annotated, long distances between segregating sites may falsely be seen as long homozygous blocks, indicating a very recent time to the common ancestor between the lineages, thereby skewing model estimates.

The generation of such an input file follows three steps:

1. Generating VCF and mask files from individual BAM files.
2. Phasing the input.
3. Combining multiple phased individuals.

In the following, we describe these steps in order

3.1 Generating VCF and Mask Files from Individual BAM Files

Starting with a BAM file, `bamCaller.py` (included in the MSMC-Tools package) can be used for generating a sample-specific VCF file and a mask file. This script reads samtools mpileup data from stdin, so it has to be used in a pipe in which a reference file in fasta format is also required. Here is an example bash script using samtools 1.0 or higher for generating chromosome-specific VCF files (`sample1.chr*.vcf.gz`) and mask files (`sample1.mask.chr*.bed.gz`) from a human BAM file:

Listing 1.1. bash script to call genotypes and masks from a single BAM file

```
#!/bin/bash
BAM=sample1.bam

# estimating the average sequencing depth using all
    sites on chromosome 20
DEPTH=$(samtools depth -r 20 <in.bam> | awk '{sum +=
     $3} END {print sum / NR}')$

for CHR in {1..22}; do
   samtools mpileup -B -q 20 -Q 20 -C 50 -g -r $CHR
       -f <ref.fa> <in.bam> | bcftools call -c -V
       indels | ./bamCaller.py $DEPTH <out.mask.
       chr$CHR.bed.gz> | gzip -c > <out.chr$CHR.vcf.
       gz>
done
```

Further options of `bamCaller.py` are:

`--minMapQ` to set the minimum mapping quality, which defaults to 20.0

`--minConsQ` to set the minimum consensus quality, which defaults to 20.0

`--legend_file` If you aim to phase your data against a reference panel, e.g. from 1000 Genomes (*see* Subheading 3.2), you need your VCF to not only contain the variant sites of the sample, but also the genotypes at additional sites at which the panel is genotyped. This option takes a gzipped file of a format that is used in the IMPUTE and SHAPEIT reference panels. It is a simple tab-separated tabular file format with one header line which gets ignored. The only important columns for this purpose are: (1) the chromosome; (2) the position; (3) the reference allele; (4) the alternative allele; (5) the type of the variant, only sites of type SNP are considered here.

3.2 Phasing

If your samples are unrelated and you want to run MSMC on more than two haplotypes at a time, you would need to statistically phase the VCFs with a tool like shapeit. There are two different phasing strategies using shapeit, either with a reference panel or without a reference panel. If a good reference panel is available for your samples, shapeit phasing with a reference panel is recommended.

Here, as an example, we describe phasing a single human diploid sample against the 1000 Genomes Phase 3 reference panel. In the following, we assume that shapeit2 is installed, the 1000 Genomes (phase 3) reference panel is available locally (can be downloaded from https://mathgen.stats.ox.ac.uk/impute/

1000GP_Phase3.html), and that the unphased VCF file contains all variable positions in the sample, plus all variable positions in the 1000 Genomes reference panel. This can be achieved using the `--legend_file` option in `bamCaller.py`.

The script first removes multi-allelic sites in your VCF, generating .noMultiAllelicSites.vcf.gz with bcftools. Then it makes a list of sites to be excluded in the main run for phasing by `shapeit -check`, because shapeit can only phase SNPs that are in both the sample and the reference panel with the same allele type. Apart from the main log file per chromosome `sample1.chr$CHR.alignments.log`, the two following files will be generated from `shapeit -check`:

1. sample1.chr$CHR.alignments.strand: this file describes all sites in detail that either have incompatible allele types in the sample and the reference panel or found in the sample but not in the reference panel.
2. sample1.chr$CHR.alignments.strand.exclude: this file gives a simple list of physical positions of sites to be excluded from phasing.

Then the script runs shapeit with `--exclude-snp` and `-no-mcmc`, generating two output files including phased sites only `sample1.chr$CHR.phased.haps.gz` and `sample1.chr$CHR.phased.samples`. These two files can be converted into VCF format by `shapeit -convert`. Afterwards, we merge the phased VCF `sample1.chr$CHR.onlyPhased.vcf.gz` and the unphased (original) VCF `sample1.chr$CHR.fixedformat.vcf.gz`, keeping all unphased sites from the original VCF, but replacing the phased ones.

```
#!/bin/bash
for CHR in {1..22}; do
    UNPHASED_VCF=sample1.chr$CHR.vcf.gz
    UNPHASED_VCF_NOMAS=sample1.chr$CHR.
        noMultiAllelicSites.vcf.gz
    GEN_MAP=genetic_map_chr${CHR}_combined_b37.txt
    REF_HAPS=1000GP_Phase3_chr$CHR.hap.gz
    REF_LEGEND=1000GP_Phase3_chr$CHR.legend.gz
    REF_SAMPLE=1000GP_Phase3.sample
    LOG_ALIGN=sample1.chr$CHR.alignments
    EXCLUDE_LIST=$LOG_ALIGN.strand.exclude
    LOG_MAIN=sample1.chr$CHR.main

    PHASED_HAPS=sample1.chr$CHR.phased.haps.gz
    PHASED_SAMPLE=sample1.chr$CHR.phased.samples
    PHASED_VCF=sample1.chr$CHR.onlyPhased.vcf

    LOG_CONVERT=sample1.chr$CHR.convert
    FINAL_VCF=sample1.chr$CHR.phased.vcf.gz
```

```
        #Preparation
        bcftools view -M 2 -O z $UNPHASED_VCF >
            $UNPHASED_VCF_NOMAS
        shapeit -check -V $UNPHASED_VCF_NOMAS -M
            $GEN_MAP --input-ref $REF_HAPS
            $REF_LEGEND $REF_SAMPLE --output-log
            $LOG_ALIGN

        #Main run
        shapeit -V $UNPHASED_VCF_NOMAS -M $GEN_MAP
            --input-ref $REF_HAPS $REF_LEGEND
            $REF_SAMPLE -O $PHASED_HAPS
            $PHASED_SAMPLE --exclude-snp
            $EXCLUDE_LIST --no-mcmc --output-log
            $LOG_MAIN

    shapeit -convert --input-haps $PHASED_HAPS
        $PHASED_SAMPLE --output-vcf $PHASED_VCF --
        output-log $LOG_CONVERT

        #Zipping and indexing
     bcftools view -O z $PHASED_VCF > $PHASED_VCF
         .gz
     bcftools index -f $PHASED_VCF

     #Merging phased and unphased vcfs, keeping
         all unphased sites from the original vcf
         , but replacing the phased ones.
     bcftools merge --force-samples $UNPHASED_VCF
          $PHASED_VCF | awk 'BEGIN {ofs"=\"t}
     $0 ~ /^#CHROM/ {print $1, $2, $3, $4, $5, $6
         , $7, $8, $9, $10}
     $0 !~ /^#/ {
         if(substr($11, 1, 3) != "./.")
             $10 = $11
         print $1, $2, $3, $4, $5, $6, $7, $8, $9
             , $10
     }' | bcftools view -O z > $FINAL_VCF
done
```

Note that this script can also be found in the git repository accompanying this book chapter (https://github.com/StatisticalPopulationGenomics/MSMCandMSMC2).

3.3 Combining Multiple Individuals into One Input File

At this point, we assume that you have a phased VCF for each individual per chromosome (potentially containing some unphased sites not in the reference panel), and one mask file for each individual per chromosome. In addition, you will need one mappability mask file per chromosome, which is universal per chromosome and does not depend on the input individuals. Mappability masks ensure that only regions in the genome are included, which have

sufficiently high mappability, i.e. no repeat regions and other features that are hard to map with next-generation sequencing data. Mappability masks can be generated using the SNPable pipeline described at http://lh3lh3.users.sourceforge.net/snpable.shtml. For the human reference genome hs37d5, they can be downloaded here https://oc.gnz.mpg.de/owncloud/index.php/s/RNQAkHcNiXZz2fd.

For generating the input files for MSMC for one chromosome, the script `generate_multihetsep.py` from MSMC-tools is required, which merges VCF and mask files together, and also performs simple trio-phasing in case the data contains trios. Here is an example of generating multihetsep files for two (previously phased) diploid individuals on chromosome 1.

```
#!/bin/bash
generate_multihetsep.py --chr 1 --mask sample1.mask.
    chr1.bed.gz --mask sample2.mask.chr1.bed.gz --
    mask mappability_mask.chr1.bed sample1.chr1.
    phased.vcf.gz sample2.chr1.phased.vcf.gz >
    sample1_sample2.chr1.multihetsep.txt
```

Another useful option in generate_multihetsep.py is `--trio <child>,<father>,<mother>`, allowing the three members of a trio. All three fields must be integers specifying the index of the child/father/mother within the VCFs you gave as input, in order. So for example, if you had given three VCF files in the order of father, mother, child, you need to give –trio 2,0,1. This option will automatically apply a constraint for phasing and also strip the child genotypes from the result.

4 Running MSMC and MSMC2

4.1 Resource Requirements

Resource usage for MSMC and MSMC2 depend on the size of the dataset, the number of haplotypes analyzed, the number of time segments and on the number of CPUs used. The following numbers are example use cases and need to be somewhat extrapolated to other use cases. As a general rule of thumb, run time and number of CPUs are inversely proportional, and memory and number of CPUs are linearly proportional. Also, the number of haplotypes and the number of time segments affect both memory and run time quadratically.

Use cases for MSMC, assuming 22 human chromosomes and 11 CPUs, default time patterning:

- A single diploid genome: 30 min, 17Gb of RAM.
- Two diploid genomes, same population: 90 min, 32 Gb of RAM.

- Two diploid genomes, two populations: 11 h, 35 Gb of RAM.
- Four diploid genomes, two populations: 21 h, 170 Gb of RAM.

Use cases for MSMC2, assuming 22 human chromosome and 11 CPUs, default time patterning:

- A single diploid genome: 18 min, 7 Gb of RAM.
- Two diploid genomes, same population: 2 h, 36 Gb of RAM.
- Two diploid genomes, two populations: 90 min, 21 Gb of RAM.
- Four diploid genomes, two populations: 8 h, 100 Gb of RAM.

4.2 Test Data

We provide input files for MSMC and MSMC2 for four diploid human individuals, two Yoruba and two French individuals. The test input data consists of 22 text files for 22 autosomes in the MSMC input format described above. The test data can be accessed at https://github.com/StatisticalPopulationGenomics/MSMCandMSMC2.

4.3 Running MSMC

A typical command line to run MSMC on the test data is

```
msmc -t 11 -R -o out_prefix Yoruba_French.double.chr
    *.multihetsep.txt
```

which runs the program on 11 CPUs (option `-t`), keeps the recombination rate fixed at the initial value (option `-R`), and uses as output-prefix the file prefix `out_prefix`. The parallelization, here specified by the number of CPUs (`-t 11`), goes across input files. So when given 22 input chromosomes as in the test data, which is typical for human data, running on 11 CPUs means that the first 11 chromosomes can be run in parallel, and then the second 11. Using more CPUs will help a bit to make things even faster, but only to the extent that the number of chromosomes exceeds or equals the number of CPUs. The `-R` option is recommended for MSMC except when running on two haplotypes only. Additional options can be viewed by running `msmc -h`.

In order to run MSMC to obtain estimates of cross-population divergences, you need to prepare your input files to contain individuals from multiple populations. For example, in order to run MSMC on one Yoruba and one French individual from the test data, you run (here for chr1 only):

```
msmc -t 11 -R -s -I 0,1,4,5 -P 0,0,1,1 -o
    crosspop_out_prefix Yoruba_French.double.chr1.
    multihetsep.txt
```

There are two changes here with respect to the first run. First, we use the options `-I 0,1,4,5 -P 0,0,1,1`, which specifies that only the first two haplotypes in each subpopulation should be used

(indices 0,1 are the first Yoruba individual, indices 4,5 the first French), and that those selected four haplotypes belong to two subpopulations. Second, we set `-s`, which instructs MSMC to skip ambiguously phased sites. This is important if you have phased your samples against a reference panel and have private variants unphased. Empirically, we have found that MSMC is quite robust to unphased sites when analyzing population size changes in a single population, but that results on cross-population divergence are affected by unphased sites, and results are less biased if those sites are removed [1].

Upon running either of the two commands above, MSMC produces several output files. First, a file containing log output, called `prefix.log`. Second, a file containing the parameter estimates at each iteration step, called `prefix.loop.txt`. And third, a file containing the final results, called `prefix.final.txt`. This last file looks like this:

```
time_index    left_time_boundary    right_time_boundary    lambda_00
0             -0                    2.79218e-06            2605.47
1             2.79218e-06           5.68236e-06            6451.92
2             5.68236e-06           8.67766e-06            3152.31
3             8.67766e-06           1.1786e-05             2526.36
...
```

Each row of this output file lists one time segment, with scaled start and end time indicated by second and third column. The fourth column contains the scaled coalescent rate in each time segment. In case of cross-population analysis (using the `-P` flag), the output will contain two more columns, titled `lambda_01` and `lambda_11`, giving the coalescence rate estimates between populations and within the second population, respectively.

Times and rates are scaled. In order to convert to real values, you need a mutation rate μ per site per generation. All times can then be converted to generations by dividing the scaled time by μ. In order to convert generations into years, a generation time is needed (for humans we typically take 29 years). Population size estimates are obtained by first taking the inverse of the scaled coalescence rate, and then dividing that inverse rate by 2μ.

To get the relative cross coalescence rate (rCCR, *see* Fig. 3), you need to compute $2\lambda_{01}/(\lambda_{00}+\lambda_{11})$, without any additional scaling. It can then be informative to compute the time point at which the relative CCR hits 0.5, to reflect an estimate of the split time between two populations (provided that a clean-split scenario is appropriate).

4.4 Running MSMC2

Running MSMC2 is very similar to running MSMC if samples come from a single population. In that case, a typical command line may look like this:

```
msmc2 -t 11 -o out_prefix msmc_input_chr*.
    multihetsep.txt
```

Note that here we have omitted the option `-R`, since MSMC2 can robustly infer recombination rates simultaneously with population sizes, so there is no need to keep the recombination rate fixed. The output of the program is the same as in MSMC.

To analyze individuals from multiple populations, as in the provided test data the procedure is different from MSMC. In that case, MSMC2 needs to be run three times independently: Once each for estimating coalescence rates within population 1, within population 2, and across populations. This has two advantages: First, since runs can be parallelized, the combined running should be faster on computer clusters. Second, if many pairs of populations are analyzed, estimates of coalescence rates within populations need to be run only once and not co-estimated with each cross-coalescence rate estimates.

So taking the test data as an example, we have four diploid individuals from two populations in a single input file, and we can run on only one individual from each population like this:

```
msmc2 -t 11 -s -I 0,1 -o pop1_out_prefix Yoruba_French.double.
 chr*.multihetsep.txt
msmc2 -t 11 -s -I 4,5 -o pop2_out_prefix Yoruba_French.double.
 chr*.multihetsep.txt
msmc2 -t 11 -s -I 0-4,0-5,1-4,1-5 -o crosspop4hap_out_prefix
 Yoruba_French.double.chr*.multihetsep.txt
```

or if we want to run on all individuals:

```
msmc2 -t 11 -s -I 0,1,2,3 -o pop1_4hap_out_prefix
 Yoruba_French.double.chr*.multihetsep.txt
msmc2 -t 11 -s -I 4,5,6,7 -o pop2_4hap_out_prefix
 Yoruba_French.double.chr*.multihetsep.txt
msmc2 -t 11 -s -I
 0-4,0-5,0-6,0-7,1-4,1-5,1-6,1-7,2-4,2-5,2-6,2-7,3-4,3-5,3-6,3-7
  -o crosspop_8hap_out_prefix Yoruba_French.double.chr*.
 multihetsep.txt
```

Here, we have again used the option `-s` to remove unphased sites. A key difference to MSMC is how haplotype pairs in MSMC2 are specified using the `-I` option. In MSMC2, haplotype configurations passed via `-I` can be given in two flavors. First, you can enter a single comma-separated list, like this `-I 0,1,4,5`. In this case, MSMC2 will run over all pairs of haplotypes within this set of indices. This is useful for running on multiple phased diploid genomes

sampled from one population. In the second flavor, you can give a list of pairs, like above: `-I 0-4,0-5,1-4,1-5`. In this case, MSMC2 will run only those specified pairs, which are all pairs between the first Yoruba and first French individual in this case. Note that if you do not use this parameter altogether, MSMC2 will run on all pairs of input haplotypes and assume that they all belong to one population.

As a special feature in MSMC2, the option `-I` can be used also to run MSMC2 to get population size estimates from entirely unphased genomes, using the composite likelihood approach to run on all pairs of unphased diploids, but not across them. For example, if your input file contains four diploid unphased samples, you could use `-I 0-1,2-3,4-5,6-7` to instruct MSMC2 to estimate coalescence rates only within each diploid genome.

In order to simplify plotting and analysis of the relative cross coalescence rate from MSMC2, we provide a tool in the MSMC-tools repository called `combineCrossCoal.py`. This tool takes as input three result files from MSMC2, obtained by running within each population and across. It will then use interpolation to create a single joint output file with all three rates that can then be plotted exactly as in the MSMC case above. To use the script on the three estimates obtained with the three MSMC2 runs above, simply run

```
python3 combineCrossCoal.py crosspop_out_prefix.
    final.txt pop1_out_prefix.final.txt
    pop2_out_prefix.final.txt > combined_pop1_pop2.
    final.txt
```

and then use the combined file to proceed with plotting.

4.5 Plotting Results

Here is an example of plotting population sizes and relative CCR in python, as well as computing the midpoint of the rCCR curve, using the `numpy`, `pandas`, and `matplotlib` libraries. To try this out, we provide result files for MSMC2 within the book chapter repository (https://github.com/StatisticalPopulationGenomics/MSMCandMSMC2), and those result files are used in this script, which is also included in the same repository:

```
#!/usr/bin/env python3

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
mu = 1.25e-8
gen = 29
dir_ = "MSMC2_OUTPUT"
msmc_out=pd.read_csv("{}/Yoruba_French.8haps.combined.msmc2.
    final.txt".format(dir_), sep='\t', header=0)
t_years=gen * ((msmc_out.left_time_boundary + msmc_out.
    right_time_boundary)/2) / mu
```

```
plt.figure(figsize=(8, 10))
plt.subplot(211)
plt.semilogx(t_years, (1/msmc_out.lambda_00)/(2*mu), drawstyle
    ='steps',color='red', label='Yoruba')
plt.semilogx(t_years, (1/msmc_out.lambda_11)/(2*mu), drawstyle
    ='steps',color='blue', label='French')
plt.xlabel("years ago")
plt.ylabel("population Sizes")
plt.legend()
plt.subplot(212)
relativeCCR=2.0 * msmc_out.lambda_01 / (msmc_out.lambda_00 +
    msmc_out.lambda_11)
plt.semilogx(t_years,relativeCCR, drawstyle='steps')
plt.xlabel("years ago")
plt.ylabel("Relative CCR")
plt.savefig("MSMC_plot.pdf")

def getCCRintersect(df, val):
    xVec = gen * ((df.left_time_boundary + df.
        right_time_boundary)/2) / mu
    yVec = 2.0 * df.lambda_01 / (df.lambda_00 + df.lambda_11)
    i = 0
    while yVec[i] < val:
        i += 1
    assert i > 0 and i <= len(yVec), "CCR intersection index
        out of bounds: {}".format(i)
    assert yVec[i - 1] < val and yVec[i] >= val, "this should
        never happen"
    intersectDistance = (val - yVec[i - 1]) / (yVec[i] - yVec[
        i - 1])
    return xVec[i - 1] + intersectDistance * (xVec[i] - xVec[i
        - 1])
print(getCCRintersect(msmc_out, 0.5)) #Print out the time when
    relativeCCR=0.5
```

This script produces the plot shown in Fig. 5 and prints out the midpoint of the cross-coalescence rate, which is `69405.8165002096` for the test data, i.e. around 70,000 years ago for a rough estimate of the split time between French and Yoruba.

5 Tips and Tricks

5.1 Bootstrapping

It is often important to obtain confidence intervals around coalescence rate estimates (either for population size estimates or for rCCR estimates). This can be done using block-bootstrapping. We provide a script called `multihetsep_bootstrap.py` in the MSMC-tools repository. You can run `python3 multihetsep_bootstrap.py -h` to get some inline help. The program generates artificial "bootstrapped" datasets from an input dataset consisting of MSMC input files, by chopping up the input data into blocks (5 Mb long by default) and randomly sampling with replacement to create artificial 3 Gb long genomes out of these blocks. By default, 20 datasets are generated. You can run the tool via

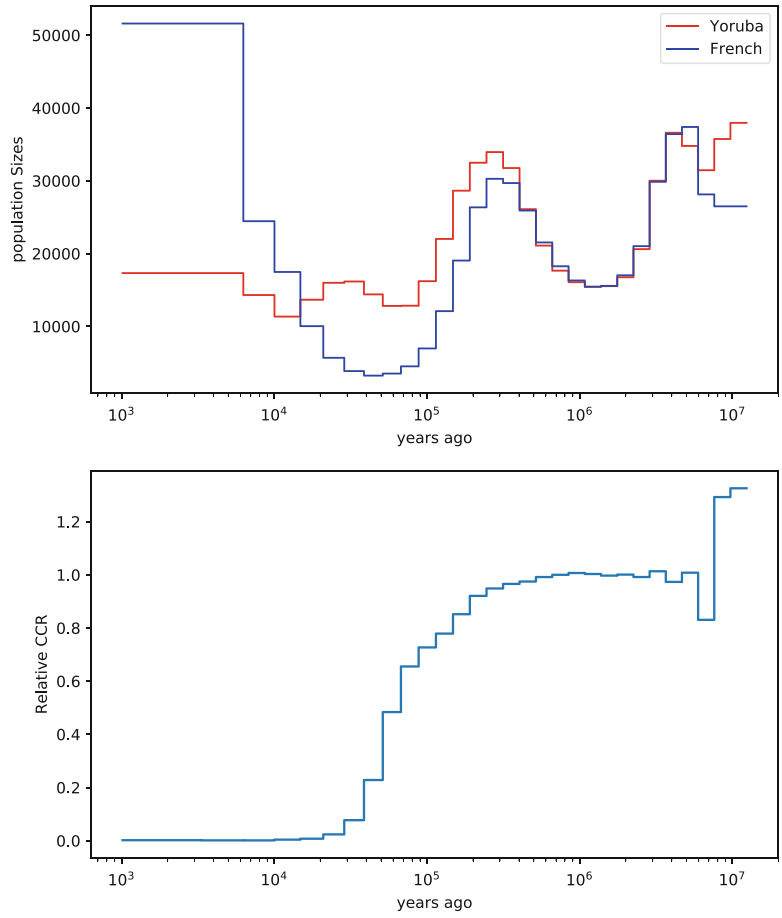

Fig. 5 The figure produced by the plotting script using the test data results

```
python3 multihetsep_bootstrap.py bootstrap_dir
    msmc_input_chr*.multihetsep.txt
```

which creates 20 subdirectories, here beginning with `bootstrap_dir`, each containing 30 multihetsep input files created with the block-sampling strategy described above. You should then run MSMC or MSMC2 on each of these datasets separately and plot all results together with the original estimates to visualize confidence intervals.

5.2 Controlling Time Patterning

Often, MSMC creates extremely large estimates in the most recent or the most ancient time intervals. This is a sign of overfitting, and can be mitigated by using fewer free parameters. By default, MSMC uses 40 time segments, with 25 free parameters (some neighboring time segments are forced to have the same coalescence rate). MSMC2 by default uses 32 time segments with 28 free parameters.

You can use the `-p` flag to control the time patterning in detail. For example, to change the patterning of MSMC2 from 32 to 20 time segments with 18 free parameters, you could try `-p 1*2+16*1 +1*2`, which would use 20 time segments, and merge together the first two and last two to have just one free coalescence rate parameter, respectively. We recommend to experiment with these settings, in particular when non-human data is analyzed, where sometimes the default settings in MSMC and MSMC2 are not appropriate because the genomes are substantially shorter and hence fewer parameters should be estimated.

References

1. Schiffels S, Durbin R (2014) Inferring human population size and separation history from multiple genome sequences. Nat Genet 46(8), 919–925
2. McVean GAT, Cardin NJ (2005) Approximating the coalescent with recombination. Philos Trans R Soc Lond B Biol Sci 360(1459), 1387–1393
3. Marjoram P, Wall JD (2006) Fast "coalescent" simulation. BMC Genet 7, 16
4. Pagani L, Schiffels S, Gurdasani D, Danecek P, Scally A, Chen Y, Xue Y, Haber M, Ekong R, Oljira T, Mekonnen E, Luiselli D, Bradman N, Bekele E, Zalloua P, Durbin R, Kivisild T, Tyler-Smith C (2015), Tracing the route of modern humans out of Africa by using 225 human genome sequences from Ethiopians and Egyptians. Am J Hum Genet 96(6), 986–991
5. Raghavan M, Steinrücken M, Harris K, Schiffels S, Rasmussen S, DeGiorgio M, Albrechtsen A, Valdiosera C, Ávila-Arcos MC, Malaspinas AS, Eriksson A, Moltke I, Metspalu M, Homburger JR, Wall J, Cornejo OE, Moreno-Mayar JV, Korneliussen TS, Pierre T, Rasmussen M, Campos PF, Damgaard PdB, Allentoft ME, Lindo J, Metspalu E, Rodríguez-Varela R, Mansilla J, Henrickson C, Seguin-Orlando A, Malmström H, Stafford T, Shringarpure SS, Moreno-Estrada A, Karmin M, Tambets K, Bergström A, Xue Y, Warmuth V, Friend AD, Singarayer J, Valdes P, Balloux F, Leboreiro I, Vera JL, Rangel-Villalobos H, Pettener D, Luiselli D, Davis LG, Heyer E, Zollikofer CPE, Ponce de León MS, Smith CI, Grimes V, Pike KA, Deal M, Fuller BT, Arriaza B, Standen V, Luz MF, Ricaut F, Guidon N, Osipova L, Voevoda MI, Posukh OL, Balanovsky O, Lavryashina M, Bogunov Y, Khusnutdinova E, Gubina M, Balanovska E, Fedorova S, Litvinov S, Malyarchuk B, Derenko M, Mosher MJ, Archer D, Cybulski J, Petzelt B, Mitchell J, Worl R, Norman PJ, Parham P, Kemp BM, Kivisild T, Tyler-Smith C, Sandhu MS, Crawford M, Villems R, Smith DG, Waters MR, Goebel T, Johnson JR, Malhi RS, Jakobsson M, Meltzer DJ, Manica A, Durbin R, Bustamante CD, Song YS, Nielsen R, Willerslev E (2015), Population Genetics. Genomic evidence for the pleistocene and recent population history of native Americans. Science 349(6250), aab3884–aab3884
6. Malaspinas AS, Westaway MC, Muller C, Sousa VC, Lao O, Alves I, Bergström A, Athanasiadis G, Cheng JY, Crawford JE, Heupink TH, Macholdt E, Peischl S, Rasmussen S, Schiffels S, Subramanian S, Wright JL, Albrechtsen A, Barbieri C, Dupanloup I, Eriksson A, Margaryan A, Moltke I, Pugach I, Korneliussen TS, Levkivskyi IP, Moreno-Mayar JV, Ni S, Racimo F, Sikora M, Xue Y, Aghakhanian FA, Brucato N, Brunak S, Campos PF, Clark W, Ellingvåg S, Fourmile G, Gerbault P, Injie D, Koki G, Leavesley M, Logan B, Lynch A, Matisoo-Smith EA, McAllister PJ, Mentzer AJ, Metspalu M, Migliano AB, Murgha L, Phipps ME, Pomat W, Reynolds D, Ricaut FX, Siba P, Thomas MG, Wales T, Wall CM, Oppenheimer SJ, Tyler-Smith C, Durbin R, Dortch J, Manica A, Schierup MH, Foley RA, Lahr MM, Bowern C, Wall JD, Mailund T, Stoneking M, Nielsen R, Sandhu MS, Excoffier L, Lambert DM, Willerslev E (2016) A genomic history of aboriginal Australia. Nature 538, 207–214
7. Mallick S, Li H, Lipson M, Mathieson I, Gymrek M, Racimo F, Zhao M, Chennagiri N, Nordenfelt S, Tandon A, Skoglund P, Lazaridis I, Sankararaman S, Fu Q, Rohland N, Renaud G, Erlich Y, Willems T, Gallo C, Spence JP, Song YS, Poletti G, Balloux F, van Driem G, de Knijff P, Romero IG, Jha AR, Behar DM, Bravi CM, Capelli C, Hervig T, Moreno-Estrada A, Posukh OL, Balanovska E,

Balanovsky O, Karachanak-Yankova S, Sahakyan H, Toncheva D, Yepiskoposyan L, Tyler-Smith C, Xue Y, Abdullah MS, Ruiz-Linares A, Beall CM, Di Rienzo A, Jeong C, Starikovskaya EB, Metspalu E, Parik J, Villems R, Henn BM, Hodoglugil U, Mahley R, Sajantila A, Stamatoyannopoulos G, Wee JTS, Khusainova R, Khusnutdinova E, Litvinov S, Ayodo G, Comas D, Hammer MF, Kivisild T, Klitz W, Winkler CA, Labuda D, Bamshad M, Jorde LB, Tishkoff SA, Watkins WS, Metspalu M, Dryomov S, Sukernik R, Singh L, Thangaraj K, Pääbo S, Kelso J, Patterson N, Reich DE (2016) The simons genome diversity project: 300 genomes from 142 diverse populations. Nature 538(7624), 201–206

8. Pagani L, Lawson DJ, Jagoda E, Mörseburg A, Eriksson A, Mitt M, Clemente F, Hudjashov G, DeGiorgio M, Saag L, Wall JD, Cardona A, Mägi R, Sayres MAW, Kaewert S, Inchley C, Scheib CL, Järve M, Karmin M, Jacobs GS, Antao T, Iliescu FM, Kushniarevich A, Ayub Q, Tyler-Smith C, Xue Y, Yunusbayev B, Tambets K, Mallick CB, Saag L, Pocheshkhova E, Andriadze G, Muller C, Westaway MC, Lambert DM, Zoraqi G, Turdikulova S, Dalimova D, Sabitov Z, Sultana GNN, Lachance J, Tishkoff S, Momynaliev K, Isakova J, Damba LD, Gubina M, Nymadawa P, Evseeva I, Atramentova L, Utevska O, Ricaut FX, Brucato N, Sudoyo H, Letellier T, Cox MP, Barashkov NA, Skaro V, Mulahasanovic L, Primorac D, Sahakyan H, Mormina M, Eichstaedt CA, Lichman DV, Abdullah S, Chaubey G, Wee JTS, Mihailov E, Karunas A, Litvinov S, Khusainova R, Ekomasova N, Akhmetova V, Khidiyatova I, Marjanović D, Yepiskoposyan L, Behar DM, Balanovska E, Metspalu A, Derenko M, Malyarchuk B, Voevoda M, Fedorova SA, Osipova LP, Lahr MM, Gerbault P, Leavesley M, Migliano AB, Petraglia M, Balanovsky O, Khusnutdinova EK, Metspalu E, Thomas MG, Manica A, Nielsen R, Villems R, Willerslev E, Kivisild T, Metspalu M (2016) Genomic analyses inform on migration events during the peopling of Eurasia. Nature 538(7624), 238–242
9. Malinsky M, Challis RJ, Tyers AM, Schiffels S, Terai Y, Ngatunga BP, Miska EA, Durbin R, Genner MJ, Turner GF (2015) Genomic islands of speciation separate cichlid ecomorphs in an East African crater lake. Science 350 (6267), 1493–1498
10. 1001 Genomes Consortium (2016), 1,135 genomes reveal the global pattern of polymorphism in Arabidopsis thaliana. Cell 166(2), 481–491
11. Frantz LAF, Mullin VE, Pionnier-Capitan M, Lebrasseur O, Ollivier M, Perri A, Linderholm A, Mattiangeli V, Teasdale MD, Dimopoulos EA, Tresset A, Duffraisse M, McCormick F, Bartosiewicz L, Gál E, Nyerges ÉA, Sablin MV, Bréhard S, Mashkour M, Bălăşescu A, Gillet B, Hughes S, Chassaing O, Hitte C, Vigne JD, Dobney K, Hänni C, Bradley DG, Larson G (2016) Genomic and archaeological evidence suggest a dual origin of domestic dogs. Science 352(6290), 1228–1231
12. Beissinger TM, Wang L, Crosby K, Durvasula A, Hufford MB, Ross-Ibarra J (2016) Recent demography drives changes in linked selection across the maize genome. Nat Plants 2, 16084
13. Hung CM, Shaner PJL, Zink RM, Liu WC, Chu TC, Huang WS, Li SH (2014) Drastic population fluctuations explain the rapid extinction of the passenger pigeon. Proc Natl Acad Sci USA 111(29), 10636–10641