Relate Practical 1: Estimating genealogies using Relate

Leo Speidel[1,*], Simon R. Myers[1,2]

[1] Department of Statistics, University of Oxford

[2] Wellcome Centre for Human Genetics, University of Oxford

[*] Contact: `leo.speidel@outlook.com`

Relate estimates the joint genealogies of many thousands of modern individuals genome-wide. These genealogies describe how individuals are related through their most-recent common ancestors back in time and can be seen as the genetic analogue of a family tree for unrelated individuals.

The output of Relate is a sequence of binary trees, each describing the genealogical relationships locally in that part of the genome. Neighbouring genealogical trees differ because of recombination events that change the genetic relationships of individuals.

The method is published in L. Speidel, S. Shi, M. Forest, S. R. Myers. A method for genome-wide genealogy estimation for thousands of samples. Nature Genetics **51**, 1321–1329 (2019).
`https://doi.org/10.1038/s41588-019-0484-x`

Note 1

A detailed documentation for **Relate** is available at `https://myersgroup.github.io/relate`.

For this practical, we will use **bash scripts, python3, and R**.
Example files used in this practical can be found in the `data/` subdirectory and example scripts can be found in the `scripts/` subdirectory.

The Relate binaries can be found under `relate_v1.0.17_*/` or downloaded from `https://myersgroup.github.io/relate`. It will be convenient to define a variable storing the location of the Relate binaries, e.g.,

```
PATH_TO_RELATE="${PWD}/relate_v1.0.17_MacOSX"
```

Overview

1 Preparing input data

Relate uses the **haps/sample** file format (output file format of SHAPEIT2) as input (see `https://myersgroup.github.io/relate/input_data.html#FileFormat`).

- You can convert from **vcf** to haps/sample and from **hap/legend/sample** to haps/sample using functions provided by Relate. (see code below or `https://myersgroup.github.io/relate/input_data.html#ConvertToHapsSample`)

Note 2: Data requirements

- Please separate your data by chromosome.
- Relate assumes whole-genome sequencing data as input.
- The data needs to be phased. This can be done, for instance, using SHAPEIT.
- Ancestral and derived states at single-nucleotide polymorphisms (SNPs) need to be known.

We provide a script to prepare your data under

```
${PATH_TO_RELATE}/scripts/PrepareInputFiles/PrepareInputFiles.sh
```

(see `https://myersgroup.github.io/relate/input_data.html#Prepare`).
This script will make sure that,

- For every SNP, the ancestral allele is denoted by 0 and the derived allele is denoted by 1.
- Remove any non-biallelic SNPs
- Optional: filter SNPs (and adjust distances between SNPs) using a genomic mask.
- Optional: subset the samples in the data set.
- Optional: generate additional annotation of SNPs that can be appended to the output of Relate.

```
cd data/
# Convert vcf to haps/sample
${PATH_TO_RELATE}/bin/RelateFileFormats \
                  --mode ConvertFromVcf \
                  --haps "example.haps" \
                  --sample "example.sample" \
                  -i "example"
gzip example.haps
gzip example.sample
```

```
# Set the ancestral allele to 0 and filter genome using a mask.
${PATH_TO_RELATE}/scripts/PrepareInputFiles/PrepareInputFiles.sh \
                --haps "example.haps.gz" \
                --sample "example.sample.gz" \
                --ancestor "ancestor.fa.gz" \
                --mask "mask.fa.gz" \
                -o "example_input"

# Rename/overwrite input for convenience
mv example_input.haps.gz example.haps.gz
mv example_input.sample.gz example.sample.gz
mv example_input.dist.gz example.dist.gz
```

This will generate `example.haps.gz`, `example.sample.gz`, and `example.dist.gz`, with the *.dist file storing distances between SNPs, adjusted for regions excluded by mask.

1.1 Simulating genetic variation datasets and genealogies using msprime

Msprime is a very efficient and flexible coalescent simulator. Please consult the msprime documentation for how to install msprime and its usage: `https://msprime.readthedocs.io/en/stable/`. (On my Mac, I first installed gsl using `brew install gsl` and then msprime using `pip3 install msprime`.)

See below for an example script simulating a single population with constant population size history:

```
import msprime
# simulate 50 samples of 1Mb length with specified parameters
tree_sequence = msprime.simulate(sample_size=50,
        Ne=10000,
        length = 1e6,
        recombination_rate = 1e-8,
        mutation_rate = 1e-8)
# save simulation in vcf
with open("msprime_simulation.vcf", "w") as vcf_file:
        tree_sequence.write_vcf(vcf_file)
# save simulation in tree sequence format
tree_sequence.dump("msprime_simulation.trees")
```

Now you can run Relate on the output vcf (see next Section). The .trees file contains the simulated (true) genealogies and can be converted into Relate format as follows:

```
./scripts/Convert_static \
        --mode ConvertFromTreeSequence \
        -i "msprime_simulation.trees" \
        --anc "msprime_simulation.anc" \
        --mut "msprime_simulation.mut"
```

The source code for this binary is available at `https://github.com/leospeidel/relate_lib`.

2 Running Relate on a small example

2.1 The arguments

Required arguments

--mode	Mode in which to run Relate. Mode "All" will execute all stages of the algorithm and other modes can be used to execute individual stages (see here). This is useful, for instance, when parallelizing Relate.
-m,--mutation_rate	Mutation rate per base per generation.
-N,--effective_size	Haploid effective population size. (For diploid organisms, multiply the effective population size of individuals by 2)
--haps	Filename of haps file.
--sample	Filename of sample file.
--map	Filename of genetic map.
-o,--output	Filename of output files without file extension.

Note 3: Optional arguments

There are other optional arguments for Relate, please consult the documentation for these (`https://myersgroup.github.io/relate/getting_started.html#GettingStarted`).

Note 4: Warning

Relate creates temporary files and directories.
Therefore, please do not run more than one instance of Relate in the same directory. (You can run Relate in different subdirectories.)

Now let's run Relate on the example file provided.

```
${PATH_TO_RELATE}/bin/Relate \
      --mode All \
      -m 1.25e-8 \
      -N 30000 \
      --haps "data/example.haps.gz" \
      --sample "data/example.sample.gz" \
      --map "data/genetic_map_Relate.txt" \
      -o "example"
```

This will generate files `example.anc` and `example.mut`, which store the estimated genealogies. In the following, we will explore how to use these files.

For the precise file format see
`https://myersgroup.github.io/relate/getting_started.html#Output`.

Note 5: Parallelising Relate

For larger datasets, it is necessary to parallelise Relate. To see how, please consult `https://myersgroup.github.io/relate/parallelise.html`.

This is **not necessary** for this practical.

2.2 Plotting trees

We can now plot the trees, e.g., using the TreeView.sh script provided. For this, we need an additional file storing assignment of individuals to populations. This file has four columns, named "sample", "population", "group", and "sex". For us, only the second column is of interest. The order in which individuals are listed has to be consistent with the order in the samples file.

Example:

```
sample population group sex
UNR1 PJB SAS 1
UNR2 JPT EAS 2
UNR3 GBR EUR 2
UNR4 YRI AFR 2
```

We can now run the TreeView.sh script as follows (this will use R and requires ggplot2 and cowplot - these will be installed if missing):

```
${PATH_TO_RELATE}/scripts/TreeView/TreeView.sh \
                 --haps "data/example.haps.gz" \
                 --sample "data/example.sample.gz" \
                 --anc "example.anc.gz" \
                 --mut "example.mut.gz" \
                 --poplabels "data/example.poplabels" \
                 --bp_of_interest 3000000 \
                 --years_per_gen 28 \
                 -o "example"
```

This will produce a pdf named `example.pdf`.

Exercise 1: Optional

Plot the corresponding true tree using example_truth.anc.gz, example_truth.mut.gz, located in `./precomp_results/` and compare!

2.3 Converting output into tree sequences and Newick strings

Sometimes it is useful to convert Relate output into a different file format. Newick strings are one of the most commonly used file formats (although inefficient!) and tree sequences are a very efficient format used by msprime, with great python and C APIs available.

Tree sequences

```
${PATH_TO_RELATE}/bin/RelateFileFormats \
                  --mode ConvertToTreeSequence \
                  -i "example" \
                  -o "example"
```

This will produce a file named `example.trees`. You can now load this file into python3 using the tskit package (`https://tskit.readthedocs.io/en/latest/`).

Tskit allows you to do many things with the trees, including plotting:

```
import tskit

ts = tskit.load("./msprime_simulation.trees")

# print the first tree
tree = ts.first()
print(tree.draw(format="unicode"))

# print all trees
for tree in ts.trees():
    print("-" * 20)
    print("tree {}: interval = {}".format(tree.index, tree.interval))
    print(tree.draw(format="unicode"))
```

Newick strings

```
${PATH_TO_RELATE}/bin/RelateExtract \
                  --mode TreeAtSNPAsNewick \
                  --anc "example.anc.gz" \
                  --mut "example.mut.gz" \
                  --bp_of_interest 3000000 \
                  -o "example"
```

This will output a file named `example_at_3000000.newick`. Now you can load the newick string into R (e.g., for plotting)

```
library(ggtree)
tree <- read.newick("example_at_3000000.newick")
ggtree(tree)
```

See `https://yulab-smu.github.io/treedata-book/` for a documentation of ggtree.

2.4 Relater

Relater is an R package for parsing input and output files of Relate, and implements functions for their manipulation.

To install, run the following commands in R:

```
install.packages("remotes")
remotes::install_github("leospeidel/relater")

library(relater)
# list all functions defined in this package
ls("package:relater")
```

To read a haps file, you can use

```
haps <- read.haps(filename)
```

To read a mut file, you can use

```
mut <- read.mut(filename)
```

2.5 Dating mutations

A great feature of genealogies is that we can now date mutations essentially "for free". To do this, let's open R and load relater and read a mut file.

```
library(relater)
mut <- read.mut(filename)
```

Assuming neutrality of a mutation, it is equally likely to have arisen anywhere between `age_begin` and `age_end`, the two ends of the branch it is mapped to. Therefore, we estimate its expected age as

```
age <- (mut$age_begin + mut$age_end)/2
```

To obtain the age of a mutation in years, multiply age by some factor (e.g., 28 for humans).

Exercise 2: Optional

Try plotting these mutation ages, e.g. against true mutation ages (using true genealogies) against derived-allele frequency.

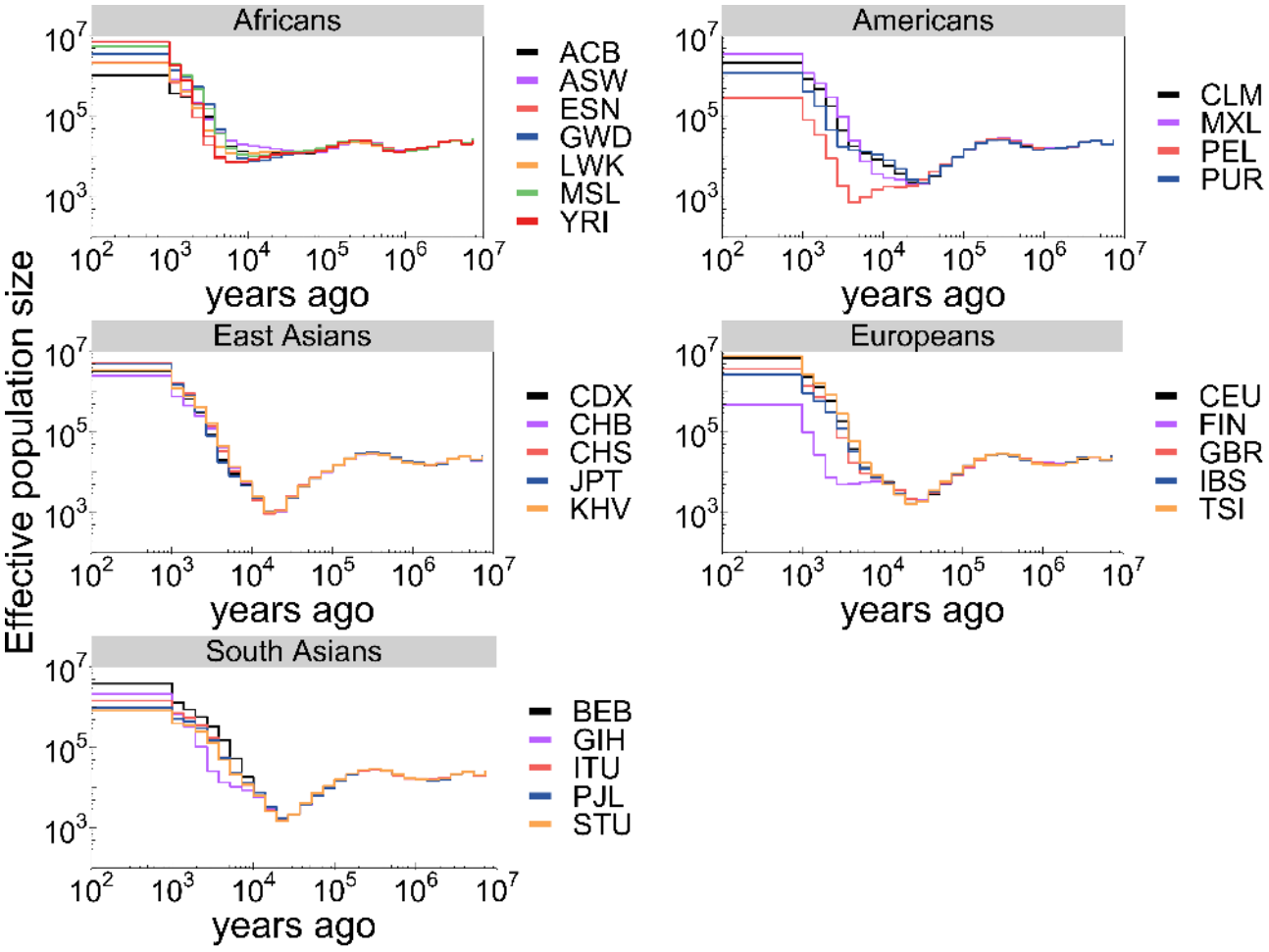

Figure 1: Effective population sizes for 26 human populations of the 1000 Genomes Project dataset estimated using Relate.

3 Jointly estimating coalescence rates and branch lengths

Demographic histories are rarely constant and vary across groups (see Fig. 1 for Relate-estimated human demographic histories.). With Relate, we can jointly fit a variable demographic history and genealogies. Downstream analyses will now be consistent with this estimated demographic history (which will e.g., improve accuracy of mutation ages).

This is done using the EstimatePopulationSize.sh script.

```
${PATH_TO_RELATE}/scripts/EstimatePopulationSize/EstimatePopulationSize.sh \
             -i "example" \
             -m 1.25e-8 \
             --poplabels "data/example.poplabels" \
             --seed 1 \
             -o "example_popsize"
```

This script will produce updated anc/mut files, as well as a .coal file storing the coalescence rates within and across groups (defined in the poplabels file). For the file format of the coal file, see `https://myersgroup.github.io/relate/modules.html#PopulationSizeScript_FileFormats`. A coal file can be loaded into R using relater (Section 2.4) as follows:

```
library(relater)
coal <- read.coal(filename)
head(coal)
```

Once the coal file is loaded into R, we can calculate diploid effective population sizes, which are defined as the inverse of the coalescence rates (and divided by two for diploid organisms):

```
coal$eff <- 0.5/coal$haploid.coalescence.rate
```

Exercise 3

Apply this to the provided example. Also, you may want to plot trees and mutation ages and compare to previous plots that assume a constant population size.

Note 6

Sometimes effective population sizes are known (e.g., estimated using another method). Then, you can specify a coal file in Relate when estimating genealogies. This is done using the –coal argument and will overwrite the -N argument. Please only specify a coal file that treats all samples as one group.

Example:

```
${PATH_TO_RELATE}/bin/Relate \
      --mode All \
      -m 1.25e-8 \
      --haps "data/example.haps.gz" \
      --sample "data/example.sample.gz" \
      --map "data/genetic_map.txt" \
      --coal "data/example.coal" \
      -o "example"
```

3.1 Estimating effective population sizes from genealogies

For a given anc/mut file, you can extract coalescence rates without reestimating branch lengths.

```
${PATH_TO_RELATE}/bin/RelateCoalescenceRate \
                --mode EstimatePopulationSize \
                --poplabels "data/example.poplabels" \
                -i "example" \
                -o "example"
```

Here, the poplabels argument is optional and the output is `example.coal`. If the poplabels file specifies more than one population, then coalescence rates will be calculated within and across all populations.

Exercise 4

For the provided example, plot estimated diploid effective population sizes. Using a log-log-scale might be useful here. Compare to the true effective population size, e.g., by estimating the coalescence rates from the true trees.

Simulating variable population size histories

We use msprime to simulate a variable population size history. Please find the script `simulate_msprime_vareff.py` under `scripts/`, which can be used as follows:

```
output="msprime_example" #output filename
mu="1e-8" #mutation rate per base per generation
recmap="./data/genetic_map_const_msprime.txt" #recombination map
popsize="./data/popsize.txt"

# run simulation script
python3 scripts/simulate_msprime_vareff.py ${output} ${mu} ${recmap} ${popsize}
```

Here, popsize.txt is a file in which we specify the demographic history of each population. The first line specifies the sample size of each population, the second line lists epoch boundaries in generations, and subsequent lines specify the diploid effective population sizes, where -1 means that this population has merged with the population of the previous line.

```
25 25
0 1e4 1.0001e4 8e4 8.0001e4 2e6 2.0001e6 1e7
2e5 1e5 5e3 5e3 3e4 3e4 1e5 1e5
5e4 5e4 1e4 1e4 -1 -1 -1 -1
```

Exercise 5

Simulate your own example and estimate the effective population sizes using Relate! Please note that for accurate estimation, you need a lot of data (on the order of a whole genome, minimum 100Mb).