Record matching of identical individuals

Jaehee Kim

05/28/2021

Overview

This vignette demonstrates a step-by-step guide for record-matching between SNP profiles to STR profiles when two samples are from the identical individual using linkage disequilibrium between samples. The method is based on Kim et al. 2018.

Setup

External software requirements

In order to run the pipeline, there are two external software requires.

- BEAGLE 4.1 for phasing and imputation. Please note that `BEAGLE` requires Java version 8. See BEAGLE manual for details.
- VCFtools for extracting necessary information from vcf files. If building from the source fail, binary executible can be found here.

Set environment variables

We first need to set the base directory where the output files of the record matching pipeline will be saved. We also set the full path to the `BEAGLE` jar file and `vcftools` executable. The following environment variables need to be set.

- `base.dir`: This is a directory where all the files generated by the pipeline will be saved.
- `bgl.jar`: Path to `BEAGLE` jar file. Format is `(full path to dir)/beagle.(version).jar`. `(version)` is the Beagle version code (eg. "01Oct15.6a3").
- `vcf.exe`: Path to `vcftools` executable. Format is `(full path to dir)/vcftools`.

The above variables will be used in all functions in the record-matching pipeline and is set by `setup` function.

```
setup(base.dir='(path to local save directory)',
      bgl.jar='(full path to dir)/beagle.(version).jar',
      vcf.exe='(full path to dir)/vcftools')
```

To run this vignette, set the environment variables in the `setup` function below:

```
setup(base.dir='/Users/Shared/RM/twin_test',
      bgl.jar='/Users/Shared/RM/beagle.27Jan18.7e1.jar',
      vcf.exe='/usr/local/bin/vcftools')
#> Loading required package: clue
#> Loading required package: plyr
#>
#> ============================================================
#> ---- Base path setup for Record Matching ----
#>
#>     Data directory and save directory for outputs:
```

```
#>          /Users/Shared/RM/twin_test
#>     BEAGLE jar file (including full path):
#>          /Users/Shared/RM/beagle.27Jan18.7e1.jar
#>     vcftools executable (including full path):
#>          /usr/local/bin/vcftools
#> ============================================================
```

Input files requirements

Each file should contain information about single STR locus and/or its surrounding SNPs only.

- `ref.f`: A file containing STR-SNP genotypes in standard vcf format. If it's unphased, it must be phased first using `phase.ref` before applying it to `impute.str` function.
- `snp.f`: A file containing SNP genotypes in 1-Mb SNP windows extending 500 kb in each direction from the STR midpoint. Must be in standard vcf format.
- `str.f`: A file containing STR genotypes at a STR locus. Must be a a `GT` file format of `vcftools`. The first column should be a chromosome and the second column should be a marker genomic position. The sample data starts from the third column. The first row should be a header and the second row should be data. The genotype can be either a phased or unphased format.

In addition, HapMap GrCh36, GrCh37, or GrCh38 genetic maps with cM units in PLINK format are required. Each file (`map.f`) should contain a map file of a single chromosome. They can be downloaded from the BEAGLE web.

In this vignette, all the above files are provided as a package data.

Markers

In this example, we will use the following 13 CODIS loci.

```
marker.f <- system.file("extdata", "marker_positions.txt",
                        package="RecordMatching", mustWork=TRUE)
marker.info <- read.table(marker.f, header=TRUE, as.is=TRUE)
```

name	chr	start	end
D13S317	13	81620034	81620315
CSF1PO	5	149436014	149436257
TH01	11	2148853	2149098
TPOX	2	1472375	1472488
VWA	12	5963365	5963514
D3S1358	3	45557209	45557339
D5S818	5	123139024	123139301
D7S820	7	83627317	83627654
D8S1179	8	125976245	125976586
D16S539	16	84943535	84943929
D18S51	18	59099658	59100344
D21S11	21	19476134	19476354
FGA	4	155728298	155728493

One can include specific loci only by subsetting the `marker.info` as shown below. For a quick testing of the pipeline, start with one locus.

```
loci.include <- c(3,5)
marker.info <- marker.info[loci.include,]
```

In general, including more loci results in the better record-matching accuracy. In this vignette, all 13 available CODIS loci are used.

Phase SNP-STR reference panel

If the reference panel is unphased, it must be phased first using `BEAGLE`. Any missing values are also imputed in this step. `phase.ref` function saves the output phased reference panel to `(base.dir)/ref_phased` directory. The output file name is the same as `ref.f` but with `_phs` postfix.

From the STR-SNP reference panel, each STR marker's allele frequency is computed using `ref.al.freq` function. The output is saved to `(base.dir)/ref_alfrq` directory. Each marker's allele frequency file has `ref_(marker name).frq` format.

```
for (m in marker.info$name) {
    ref.f <- system.file("extdata", "twin_test", "reference",
                         paste0("ref_", m, ".vcf"),
                         package="RecordMatching", mustWork=TRUE)

    # if reference panel is unphased, phase them.
    phase.ref(ref.f)

    # otherwise, go straight to allele frequency
    ref.al.freq(ref.f, m)
}
```

Impute STR genotypes and compute compute genotype probabilities

We now impute genotypes of individuals at a STR `marker` locus from its surrounding SNPs `snp.f` using a reference panel `ref.f` and a genetic map `map.f`by running `impute.str` function.

Note that **the reference panel must be phased and have no missing values** for imputation with `BEAGLE`.

The imputed and phased SNP-STR vcf file is stored at `(base.dir)/imputed_str` where the `base.dir` was set by running `setup` at the beginning of the pipeline. From the imputed SNP-STR haplotype, the `impute.str` function also extracts imputed genotypes and genotype probabilities at the STR locus and save it in the `(base.dir)/imputed_str` folder. The output file name containing imputed genotypes probabilities is `imp_str_(marker name).GP.FORMAT`.

```
for (i in 1:dim(marker.info)[1]) {
    m <- marker.info$name[i]
    chr <- marker.info$chr[i]
    snp.f <- system.file("extdata", "twin_test", "SNP",
                         paste("snp_", m, ".vcf", sep=''),
                         package="RecordMatching", mustWork=TRUE)
    ref.f <- system.file("extdata", "twin_test", "reference",
                         paste("ref_", m, ".vcf", sep=''),
                         package="RecordMatching", mustWork=TRUE)
    map.f <- system.file("extdata", "twin_test", "plink.GRCh36.map",
                         paste("plink.chr", chr, '.GRCh36.map', sep=''),
                         package="RecordMatching", mustWork=TRUE)

    impute.str(snp.f=snp.f, ref.f=ref.f, map.f=map.f, marker=m)
}
```

Compute match score matrix

We characterize the familial relationship between two individuals A and B by a vector of the three Cotterman identity coefficients, $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \Delta_2)$, respectively representing the probabilities of three identity states C_0, C_1, C_2. Each Δ_k represents the probability that two diploid individuals share k alleles identically by descent at an autosomal locus .

For two non-inbred individuals, individual A with a STR profile R_A and individual B with a SNP profile S_B, we test a hypothesis that A and B are related with a relationship defined by $\boldsymbol{\Delta}_{\text{test}}$, against the null hypothesis in which the two individuals are unrelated. To test the hypothesis, we use the log-likelihood-ratio match score:

$$\lambda(R_A, S_B) = \ln[\mathbb{P}(R_A \mid S_B, \boldsymbol{\Delta}_{\text{test}})] - \ln[\mathbb{P}(R_A)].$$

Assuming independence of the STR loci, so that genotypes at separate STRs are independent, the match score can be expressed as a sum of log-likelihood ratios across STR loci:

$$\lambda(R_A, S_B) = \sum_{\ell=1}^{L} \Big[\ln[\mathbb{P}(R_{A\ell} \mid S_{B\ell}, \boldsymbol{\Delta}_{\text{test}})] - \ln[\mathbb{P}(R_{A\ell})] \Big].$$

Three relatedness scenarios considered in Kim et al. 2018 are

- Same individual $\boldsymbol{\Delta} = (0, 0, 1)$
- Parent-offspring $\boldsymbol{\Delta} = (0, 1, 0)$
- Sib pairs $\boldsymbol{\Delta} = (0.25, 0.5, 0.25)$

Here, we demonstrate the case $\boldsymbol{\Delta}_{\text{test}} = (0, 0, 1)$ when the true relationship is $\boldsymbol{\Delta}_{\text{true}} = (0, 0, 1)$. Other test hypothesis can be examined by updating `cott.vec` below.

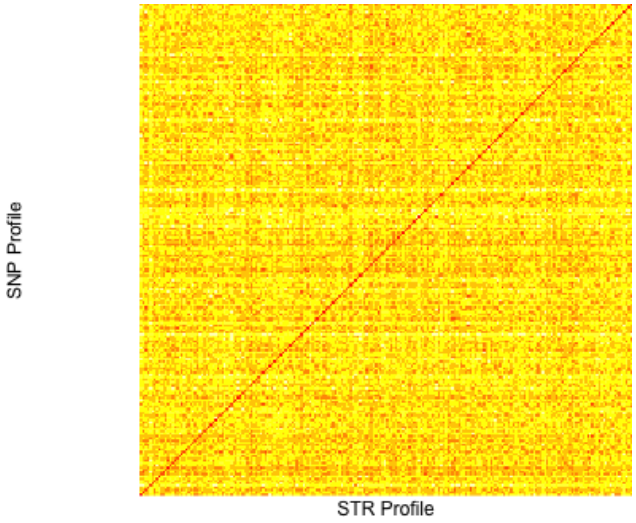
```
cott.vec <- c(0,0,1) # same individual
# cott.vec <- c(0,1,0) # parent-offspring
# cott.vec <- c(0.25, 0.5, 0.25) # sib pairs

llr.single.irt <- list()

for (i in 1:dim(marker.info)[1]) {
    m <- marker.info$name[i]
    str.f <- system.file("extdata", "twin_test", "STR",
                         paste("str_", m, ".GT.FORMAT", sep=''),
                         package="RecordMatching", mustWork=TRUE)
    llr.single.irt[[i]] <- comp.match.mat(cott.vec=cott.vec, marker=m, str.f=str.f)
    names(llr.single.irt)[i] <- m
}

mat <- Reduce('+', llr.single.irt)
```

The final match score matrix using 13 loci is shown below. The following figure corresponds to Figure 2(A) in Kim et al. 2018.



Compute match accuracies

From the match score matrix, we now compute the record-matching accuracy. We consider four different match-assignment scenarios: one-to-one matching, one-to-many matching with a query SNP profile, one-to-many matching with a query STR profile, and needle-in-haystack matching.

In one-to-one matching, we assume that it is already known that each profile in the SNP dataset has exactly one true relative in the STR dataset and vice versa. To find the pairing of profiles that maximizes the sum of match scores across all paired profiles, we use the Hungarian algorithm. Record-matching accuracy is defined as the fraction of STR profiles matched to the correct SNP profiles.

The match accuracies under all four scenarios can be computed using `comp.match.acc` function.

```
match.acc <- comp.match.acc(mat)
```

hungarian.acc	pickSTR.acc	pickSNP.acc	onematch.acc
1	0.8990826	0.8899083	0.4770642