ASAFE (Ancestry Specific Allele Frequency Estimation)

Qian Zhang

2016-06-05

Contents

1 Introduction: What ASAFE does **1**

1.1 ASAFE in the Context of a Larger Genetic Analysis Workflow 1

2 Input Files **2**

2.1 Ancestry File 2

2.2 Genotype File 2

3 Functions **3**

4 Reproducibility **3**

4.1 Reproducing Table 1 of the ASAFE paper 4

4.2 Reproducing Supplementary Tables of the ASAFE paper 6

5 Try ASAFE Out on a Small Data Set **9**

6 Citation **11**

1 Introduction: What ASAFE does

The ASAFE (Ancestry Specific Allele Frequency Estimation) package contains a collection of functions that can be used to carry out an EM algorithm to estimate ancestry-specific allele frequencies for a bi-allelic genetic marker (e.g. a SNP) from genotypes and ancestry pairs, when each diploid individual's genotype phase relative to ancestry pair is unknown. If there are three ancestries, a = 0, 1, or 2 (e.g. African, European, or Native American), ASAFE functions can be used to estimate three probabilities, P(Allele 1|Ancestry a), a $\in \{0, 1, 2\}$, at each marker. ASAFE function algorithm_1snp_wrapper() can be applied to a matrix of ancestries and a matrix of genotypes for 3-way admixed diploid individuals at bi-allelic markers. Ancestries at different markers need not be phased with respect to each other, and genotypes at different markers need not be phased with respect to each other. Denoting each marker's alleles 0 and 1, algorithm_1snp_wrapper() outputs estimates of three ancestry-specific allele 1 frequencies for each marker.

1.1 ASAFE in the Context of a Larger Genetic Analysis Workflow

The following file in the ASAFE R package gives a diagram illustrating a genetic analysis workflow involving ASAFE: inst/ASAFE_Visual.pdf. An example script that performs two steps in the workflow, involving phasing of admixed genotypes with BEAGLE and then obtainment of local ancestry estimates and re-phased genotypes via RFMIX, is here: inst/scripts/bgl_then_rfmix.sh.

2 Input Files

2.1 Ancestry File

Your ancestry file should give admixed individuals' phased ancestries as a rectangular matrix with the following rows, columns, and entries:

2.1.1 Rows

- 1st row: Header line with column names
- Subsequent rows: 1 row corresponds to 1 marker

2.1.2 Columns

- 1st column: Marker ID
- 1 column per chromosome, with two consecutive columns per individual, corresponding to the individual's pair of homologous chromosomes. For example, if there are 3 admixed individuals, then the first row that gives column names might be

Marker ADM1 ADM1 ADM2 ADM2 ADM3 ADM3

- Columns should be separated by whitespace (i.e. spaces or tabs)

2.1.3 Entries

- For an entry that is not in the Marker ID column, an entry can take value 0, 1, or 2, which are arbitrary labels for three ancestries

To read your file into R, use a command like this:

```
ancestries <- read.table(file = "your_ancestry_file.txt", header = TRUE)
```

We have provided the full data set of simulated ancestries that was used in the ASAFE paper. [1] This data set is stored as a matrix adm_ancestries. It contains ancestries at 56,003 markers for 250 admixed individuals.

2.2 Genotype File

Your genotype file should give unphased admixed individuals' genotypes as a rectangular matrix with the following rows, columns, and entries:

2.2.1 Rows

- 1st row: Header line with column names
- Subsequent rows: 1 row corresponds to 1 marker

2.2.2 Columns

- 1st column: Marker ID
- 1 column per person. For example, if there are 3 admixed individuals, then the first row that gives column names might be

ID ADM1 ADM2 ADM3

- Columns should be separated by whitespace (i.e. spaces or tabs)
- Individuals must be listed in the same order in the genotype file as in the ancestry file

2.2.3 Entries

- For an entry that is not in the Marker ID column, an entry can take value 0/0, 0/1, 1/0, or 1/1, where 0 and 1 are arbitrary labels for a bi-allelic SNP's two alleles
- A slash "/" indicates an unphased genotype, so 0/1 and 1/0 are the same unphased genotype

To read your file into R, use a command like this:

```r
genotypes <- read.table(file = "your_genotypes_file.txt", header = TRUE)
```

We have provided the full data set of simulated genotypes that was used in the ASAFE paper. [1] This data set is called adm_genotypes. It contains genotypes at 56,003 markers for 250 individuals.

For details about data matrices that come with the ASAFE package (e.g. adm_ancestries, adm_genotypes), load the ASAFE package into R with the command "library(ASAFE)" and type "?" followed immediately by the name of the matrix, for example "?adm_ancestries".

3 Functions

These are the functions you might want to use:

- algorithm_1snp()
- algorithm_1snp_wrapper()

For information about these functions (e.g. their inputs, outputs, and examples for usage), see the function's man page by doing the following: Load the ASAFE package into R with the command "library(ASAFE)" and type "?" followed immediately by the name of the function, for example "?algorithm_1snp".

If you are interested in other functions that are not listed above, for instance functions that the above functions call or functions used in the Reproducibility section of this vignette, see the .R file that implements the function you're interested in for comments describing the function.

4 Reproducibility

We demonstrate how ASAFE package functions can be used in an analysis, by repeating the analysis in the ASAFE paper.

The following is code that

- Generates ancestry-specific allele 1 frequency estimates, for all ancestries 0, 1, and 2, and for all 56,003 markers
- Reproduces Table 1 of the ASAFE paper
- Reproduces supplementary tables of the ASAFE paper

4.1 Reproducing Table 1 of the ASAFE paper

Review the data matrices adm_ancestries and adm_genotypes that come with ASAFE.

```r
# Clear workspace and load ASAFE
rm(list=ls())
library(ASAFE)

# Rows: Marker ID
# Cols: Marker ID, 2 consecutive columns for each individual's 2 chromosomes
# (i.e. ADM1, ADM1, ADM2, ADM2, ..., ADM250, ADM250)
dim(adm_ancestries) # 56,003 x 501
```

```
## [1] 56003   501
```

```r
# Rows: Marker ID
# Cols: Marker ID, 1 column for each individual
# (i.e. ADM1, ADM2, ..., ADM250)
dim(adm_genotypes) # 56003 x 251
```

```
## [1] 56003   251
```

Re-format the data matrices.

```r
# Making the rsID column row names

row.names(adm_ancestries) <- adm_ancestries[,1]
row.names(adm_genotypes) <- adm_genotypes[,1]

adm_ancestries <- adm_ancestries[,-1]
adm_genotypes <- adm_genotypes[,-1]
```

Manipulate the genotypes matrix adm_genotypes to create the alleles matrix, which has individuals' chromosomes as rows, with homologous chromosomes belonging to the same individual listed consecutively, and SNP markers as columns.

```r
# apply() works on each row (SNP) of the genotypes matrix, and returns a list,
# with each element of the list corresponding to a SNP.
#
# alleles_list is a list of lists.
# Elements of the outer list correspond to SNPs.
# Elements of the inner list correspond to 250
# individual's alleles with no delimiter "/" separating alleles.
alleles_list = apply(X = adm_genotypes, MARGIN = 1, FUN = strsplit, split = "/")
```

```r
# Creates a matrix: Number of alleles (ADM1, ADM1, ..., ADM250, ADM250) x (SNPs)
alleles_unlisted = sapply(alleles_list, unlist)

# Change elements of the matrix to numeric:
# Number of alleles (ADM1, ADM1, ..., ADM250, ADM250) x (SNPs).
alleles = apply(X = alleles_unlisted, MARGIN = 2, as.numeric)
```

The vector 1:ncol(alleles) indexes SNPs in the alleles and adm_ancestries matrices. For these SNPs, we input alleles from admixed individuals' genotypes and ancestries to the function algorithm_1snp_wrapper(), to obtain ancestry-specific allele frequency estimates. We then store these estimates in the matrix adm_estimates, which has rows corresponding to SNPs and columns corresponding to marker, and then ancestry-specific allele frequencies P(Allele 1 | Anc 0), P(Allele 1 | Anc 1), and P(Allele 1 | Anc 2). Because the call to algorithm_1snp_wrapper() can take a long time on a laptop, we have included in ASAFE the adm_estimates matrix that would result from evaluating the following chunk of code.

```r
# adm_estimates has
# Columns: SNPs.
# Rows: First row is Marker ID. Following rows are ancestries 0, 1, and 2.
# Entries: Ancestry-specific allele frequency estimates.

adm_estimates = sapply(X = 1:ncol(alleles), FUN = algorithm_1snp_wrapper,
                       alleles = alleles, ancestries = adm_ancestries)

# Transpose adm_estimates so that
# Row: SNPs
# Columns: Marker ID, then ancestries 0, 1, and 2.

adm_estimates = t(adm_estimates)

# Add column names to the matrix adm_estimates

colnames(adm_estimates) = c("Marker", "Est_A1freq_Anc0",
                            "Est_A1freq_Anc1", "Est_A1freq_Anc2")
```

Now we calculate Table 1 of the ASAFE paper, which gives summary statistics on ASAFE's errors in estimating ancestry-specific allele frequencies. [1] For each ancestry and each marker with true ancestry-specific allele 1 frequency falling into a certain bin, we calculate error (estimated frequency - true frequency). Table 1 reports the mean of the errors and the SD of the errors, for each ancestry (0, 1, or 2) and each true allele frequency bin.

```r
# Make Table 1 of the paper
#
# - Rows:
# 1) Anc 0 Mean Error, 2) Anc 0 SD Error
# 3) Anc 1 Mean Error, 4) Anc 1 SD Error
# 5) Anc 2 Mean Error, 6) Anc 2 SD Error
#
# - Cols: True Allele 1 frequency bins
# 1) (0, 0.2], 2) (0.2, 0.4], 3) (0.4, 0.6],
# 4) (0.6, 0.8], 5) (0.8, 1.0]

results = matrix(nrow = 6, ncol = 5)
```

```
# Get mean, sd errors for ancestry 0, and all 5 true allele freuqency bins
results[1:2,] = get_mean_sd_error_anc(ancestry = 0,
                                      estimates = adm_estimates,
                                      truth = ancestral_freqs)

# Get mean, sd errors for ancestry 1, and all 5 true allele freuqency bins
results[3:4,] = get_mean_sd_error_anc(ancestry = 1,
                                      estimates = adm_estimates,
                                      truth = ancestral_freqs)

# Get mean, sd errors for ancestry 2, and all 5 true allele freuqency bins
results[5:6,] = get_mean_sd_error_anc(ancestry = 2,
                                      estimates = adm_estimates,
                                      truth = ancestral_freqs)

results
```

```
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,] -0.0011187898 -0.0003076899 -0.0003727417  0.0004320703 -4.303504e-04
## [2,]  0.0065161633  0.0185322424  0.0233157145  0.0185924874  1.177260e-02
## [3,] -0.0015222919 -0.0004365006 -0.0007248265 -0.0010224213  5.804149e-06
## [4,]  0.0076957998  0.0208800239  0.0249106468  0.0220425769  1.224611e-02
## [5,] -0.0004061659 -0.0016625745  0.0021160926  0.0047794938  7.237515e-04
## [6,]  0.0082948791  0.0235352670  0.0237762153  0.0256700423  1.181338e-02
```

The results matrix should match up with Table 1 of the ASAFE paper. [1]

4.2 Reproducing Supplementary Tables of the ASAFE paper

The original R script used to produce supplementary tables is called simulate.R, and is provided in the inst/scripts directory of the ASAFE package. It can be run from the command line of a terminal window with the command "Rscript simulate.R 1 250 5000".

Alternatively, to retain usage of your command line while simulate.R is running, you can submit simulate.R as a job to a cluster, with the shell script run.sh, also provided in the inst/scripts directory of the ASAFE package.

Output files error_summary_stats_error_0.07_jobID_30146.txt and error_summary_stats_error_0_jobID_30146.txt obtained using run.sh are provided in the inst/extdata directory of the ASAFE R package.
Numbers in Supplementary Tables 1 and 2 came from these two files.

The following is R code that can be used to get numbers close to those in Supplementary Tables 1 and 2 of the ASAFE paper. Differences between numbers in the supplementary tables and numbers generated here can be attributed to differences in random number generation.

The original simulate.R code used to generate Supplementary Tables 1 and 2 has extraneous comments and code in it, with function tests mixed in with function definitions. To obtain tidier code below, simulate.R was cleaned up, separating function tests from function definitions. This separation resulted in different random numbers being generated by the original code and by the code below.

```
n_ind <- 250
n_markers <- 5000

# Load in functions needed to use the ASAFE EM algorithm
```

```r
library(ASAFE)

# Source in all functions needed to assess error
# when [p0, p1, p2] takes different values, and
# when there is error in local ancestry calls.

source("../R/draw_allele_given_anc.R")
source("../R/get_true_freqs_1snp.R")
source("../R/get_errors_1_scenario.R")
source("../R/get_errors_summary_stats_1_scenario.R")
source("../R/get_scenario_errors.R")
source("../R/sample_ancestry.R")
source("../R/change_ancestry.R")
source("../R/get_results_error.R")

# Ancestry-specific allele frequencies.
# Rows: Scenarios.
# Cols: P(Allele 1 | Anc 0), P(Allele 1 | Anc 1), P(Allele 1 | Anc 2).
# Each row is a [P(Allele 1 | Anc 0, P(Allele 1 | Anc 1, P(Allele 1 | Anc 2)]
# combination that we will consider.

anc_spec_freqs <- matrix(c(0, 0.1, 0.4, 0.48, 0.1, 0.4,
                 0.5, 0.5, 0.5, 0.5, 0.6, 0.5,
             0.9, 0.8, 0.6, 0.52, 0.6, 0.5),
             nrow = 6, ncol = 3)

#### Draw ancestries

set.seed(1)

# Ancestries matrix.
# Rows: Individuals' alleles. Cols: Markers.

ancestries_matrix <- matrix(NA, nrow = 2 * n_ind, ncol = n_markers)

# Randomly generate ancestries
# for each individual's allele at every marker.
# Ancestries are 0, 1, or 2.

for(snp in 1:n_markers){

    ancestries_matrix[,snp] <- replicate(n = 2 * n_ind,
                                         sample(x = 0:2, size = 1))

}

# Throw out the draws (i.e. columns i.e. markers)
# where there are not all three ancestries (0,1, or 2).

col_all_3_ancs <- apply(  X = ancestries_matrix,
                          MARGIN = 2,
                          FUN = function(col_vector){
                              ind_all_3 <- (length(unique(col_vector)) == 3)
                              return(ind_all_3)

                          })

# This fixed ancestries matrix will be used for multiple
# ancestral allele frequency scenarios.

ancestries_matrix_original <- ancestries_matrix[, col_all_3_ancs]

# Error rate = 0. out_0.0 should be similar to Supplementary Table 1.

error_rate <- 0

out_0.0 <- get_results_error(error_rate = error_rate,
                             anc_spec_freqs = anc_spec_freqs,
                             ancestries_matrix_true = ancestries_matrix_original)

out_0.0
```

```
##                     p0  p1   p2          Mean           SD
## Abs_Error_Afr_Freq 0.00 0.5 0.90  1.282718e-09 1.128781e-09
## Abs_Error_Eur_Freq 0.00 0.5 0.90 -6.762604e-05 1.087293e-02
## Abs_Error_NA_Freq  0.00 0.5 0.90  8.700814e-05 1.084779e-02
## Abs_Error_Afr_Freq 0.10 0.5 0.80  7.298124e-05 1.355683e-02
## Abs_Error_Eur_Freq 0.10 0.5 0.80 -1.417433e-04 1.935734e-02
## Abs_Error_NA_Freq  0.10 0.5 0.80  9.117781e-05 1.718271e-02
## Abs_Error_Afr_Freq 0.40 0.5 0.60  4.909904e-04 3.004370e-02
## Abs_Error_Eur_Freq 0.40 0.5 0.60 -2.915598e-04 3.094985e-02
## Abs_Error_NA_Freq  0.40 0.5 0.60 -1.048205e-04 2.948988e-02
## Abs_Error_Afr_Freq 0.48 0.5 0.52 -8.241337e-04 3.226393e-02
## Abs_Error_Eur_Freq 0.48 0.5 0.52  6.081167e-04 3.235874e-02
## Abs_Error_NA_Freq  0.48 0.5 0.52  2.360481e-04 3.239321e-02
## Abs_Error_Afr_Freq 0.10 0.6 0.60 -8.457614e-05 1.509251e-02
## Abs_Error_Eur_Freq 0.10 0.6 0.60  2.795426e-04 2.286121e-02
## Abs_Error_NA_Freq  0.10 0.6 0.60 -2.138383e-04 2.304949e-02
## Abs_Error_Afr_Freq 0.40 0.5 0.50  5.777319e-04 3.174236e-02
## Abs_Error_Eur_Freq 0.40 0.5 0.50 -1.174642e-03 3.189592e-02
## Abs_Error_NA_Freq  0.40 0.5 0.50  6.853494e-04 3.176509e-02
```

```
# write.table(x = out_0.0, file = paste("error_summary_stats_error_", error_rate,
#             "_jobID_", jobID,
#             ".txt", sep = ""),
#                           quote = FALSE,
#                           col.names = TRUE, row.names = TRUE, append = FALSE)

# Error rate = 0.07. out_0.07 should be similar to Supplementary Table 2.

error_rate <- 0.07

out_0.07 <- get_results_error(error_rate = error_rate,
                         anc_spec_freqs = anc_spec_freqs,
                         ancestries_matrix_true = ancestries_matrix_original)
```

```
out_0.07
```

```
##                        p0  p1   p2          Mean         SD
## Abs_Error_Afr_Freq 0.00 0.5 0.90  0.0486326246 0.01881265
## Abs_Error_Eur_Freq 0.00 0.5 0.90 -0.0033589709 0.02116062
## Abs_Error_NA_Freq  0.00 0.5 0.90 -0.0454656953 0.02170835
## Abs_Error_Afr_Freq 0.10 0.5 0.80  0.0385872442 0.02299693
## Abs_Error_Eur_Freq 0.10 0.5 0.80 -0.0033608804 0.02620828
## Abs_Error_NA_Freq  0.10 0.5 0.80 -0.0354141313 0.02484491
## Abs_Error_Afr_Freq 0.40 0.5 0.60  0.0108874142 0.03458431
## Abs_Error_Eur_Freq 0.40 0.5 0.60  0.0000986558 0.03470471
## Abs_Error_NA_Freq  0.40 0.5 0.60 -0.0109981666 0.03379990
## Abs_Error_Afr_Freq 0.48 0.5 0.52  0.0022911243 0.03534967
## Abs_Error_Eur_Freq 0.48 0.5 0.52  0.0001933984 0.03577480
## Abs_Error_NA_Freq  0.48 0.5 0.52 -0.0024236081 0.03525088
## Abs_Error_Afr_Freq 0.10 0.6 0.60  0.0353676128 0.02384481
## Abs_Error_Eur_Freq 0.10 0.6 0.60 -0.0178135851 0.02904330
## Abs_Error_NA_Freq  0.10 0.6 0.60 -0.0178437820 0.02879976
## Abs_Error_Afr_Freq 0.40 0.5 0.50  0.0070023044 0.03526576
## Abs_Error_Eur_Freq 0.40 0.5 0.50 -0.0036141513 0.03460109
## Abs_Error_NA_Freq  0.40 0.5 0.50 -0.0034272089 0.03460115
```

```
# write.table(x = out_0.07, file = paste("error_summary_stats_error_", error_rate,
#                          "_jobID_", jobID,
#                          ".txt", sep = ""),
#                          quote = FALSE,
#                          col.names = TRUE, row.names = TRUE, append = FALSE)
```

5 Try ASAFE Out on a Small Data Set

To quickly try ASAFE's EM algorithm out, we can do the same analysis given in the Reproducibility section, except on subsets of simulated genotype and ancestry data contained in adm_ancestries and adm_genotypes. We extract information for the first 9 SNPs in adm_ancestries and adm_genotypes to respectively create matrices adm_ancestries_test and adm_genotypes_test. With these subsetted matrices, we then repeat the same analysis to generate Table 1, given in the Reproducibility section.

```
# Clear workspace and load ASAFE
rm(list=ls())
library(ASAFE)

# adm_ancestries_test is a matrix with
# Rows: Markers
# Columns: Marker ID, individuals' chromosomes' ancestries
# (e.g. ADM1, ADM1, ADM2, ADM2, and etc.)

# adm_genotypes_test is a matrix with
# Rows: Markers
# Columns: Marker ID, individuals' genotypes (a1/a2)
# (e.g. ADM1, ADM2, ADM3, and etc.)
```

```r
adm_ancestries_test <- head(adm_ancestries)
adm_genotypes_test <- head(adm_genotypes)

# Making the rsID column row names

row.names(adm_ancestries_test) <- adm_ancestries_test[,1]
row.names(adm_genotypes_test) <- adm_genotypes_test[,1]

adm_ancestries_test <- adm_ancestries_test[,-1]
adm_genotypes_test <- adm_genotypes_test[,-1]

# alleles_list is a list of lists.
# Outer list elements correspond to SNPs.
# Inner list elements correspond to 250 people's alleles
# with no delimiter separating alleles.
alleles_list <- apply(X = adm_genotypes_test, MARGIN = 1,
                      FUN = strsplit, split = "/")

# Creates a matrix:
# Alleles for chromosomes (ADM1, ADM1, ..., ADM250, ADM250) x (SNPs)
alleles_unlisted <- sapply(alleles_list, unlist)

# Change elements of the matrix to numeric
alleles <- apply(X = alleles_unlisted, MARGIN = 2, as.numeric)

# Apply the EM algorithm to each SNP to obtain
# ancestry-specific allele frequency estimates for all SNPs in
# matrices alleles and adm_ancestries_test.
#
# Columns correspond to markers.
# Rows correspond to ancestries 0, 1, and then 2.
# Entries in rows 2 through 4
# give P(Allele 1 | Ancestry a), a = 0, 1, or 2 for a marker.

adm_estimates_test <- sapply(X = 1:ncol(alleles), FUN = algorithm_1snp_wrapper,
                        alleles = alleles, ancestries = adm_ancestries_test)

adm_estimates_test
```

```
##      [,1]                   [,2]                   [,3]
## [1,] "rs0"                  "rs1"                  "rs4"
## [2,] "2.17620769780578e-08" "0.105263153761591"    "1.18131694231919e-09"
## [3,] "0.168278615987883"    "3.83154290109415e-09" "0.057142851651153"
## [4,] "0.334748301823412"    "2.35375994316977e-10" "4.92885078071704e-09"
##      [,4]                [,5]                   [,6]
## [1,] "rs7"               "rs14"                 "rs19"
## [2,] "0.815381149051769" "2.92397652269074e-08" "5.79951848126169e-22"
## [3,] "0.728970436943625" "8.78469964260191e-08" "0.017142818073981"
## [4,] "0.999999980824761" "0.00649340666737419"  "4.43964502125075e-08"
```

6 Citation

ASAFE: Ancestry-Specific Allele Frequency Estimation Qian S. Zhang; Brian L. Browning; Sharon R. Browning Bioinformatics 2016; doi: 10.1093/bioinformatics/btw220