Mini-Admixfrog

Stephan Schiffels

August 2021

Introduction

We want to create a simple HMM that is able to paint an unphased diploid target genome into local ancestry states, according to multiple source genomes.

Observations are structured along SNPs at which i) the derived allele frequencies *differ* between the sources, and ii) there is no missing data in the target and at least one non-missing genotype in each source genome.

At every such SNP, an observation is given by a tuple $(G^i, (a_1^i, d_1^i), (a_2^i, d_2^i), \ldots)$, where i denotes the SNP index, $G^i = \{0, 1, 2\}$ denotes the nr of derived alleles in the target genotype, and a_k^i and d_k^i are the numbers of ancestral and derived alleles in source k at SNP i.

A diploid *state* is a tuple of sources. For example, with 2 sources, we have states $S^i = 11$, $S^i = 12$ and $S^i = 22$. With three sources, there are 6 states, and so on.

We are for now not concered with parameter optimization, but only in posterior decoding, that is, informing about local state probabilities given data and a model consisting of given transition- and emission-probabilities.

Emission probabilities

We will omit the SNP indices i in the following.

The emission probability

$$e(G|\{a_k, d_k\}, S)$$

is the probability of a target genotype, given the local state S^i and source allele counts (a_k^i and d_k^i). We follow the original admixfrog model and write the emission probabilities as a betabinomial sampling probability (Peter 2020).

Homozygous states

Specifically, for homozygous states (i.e. $S = 11, 22, \ldots$), we have

$$e(G|a, d, S) = \binom{2}{G} \frac{B(G + d + d', 2 - G + a + a')}{B(d + d' + a + a')} \tag{1}$$

where we have omitted the k index for the respective homozygous state. There are prior parameters a' and d' which control the sampling uncertainty in the source genotypes. A simple choice for the priors is $a' = d' = 1$, but since we here deal mostly with the case of very few source genomes, often only one per source, it is more advisable to choose values much smaller than 1 to mimic the expected site frequency spectrum. In practice, we can fit it from data, or simply guess around values such as 0.1 or 0.01, since arguably the results won't depend too much on it.

Heterozygous states

For heterozygous states, without loss of generality we here write a_1, d_1, a_2, d_2 for the two respective sources, and have:

$$e(G = 0|a_1, d_1, a_2, d_2, S) = \frac{(a_1 + a')(a_2 + a')}{(d_1 + d' + a_1 + a')(d_2 + d' + a_2 + a')} \tag{2}$$

$$e(G = 1|a_1, d_1, a_2, d_2, S) = \frac{(a_1 + a')(d_2 + d') + (a_2 + a')(d_1 + d')}{(d_1 + d' + a_1 + a')(d_2 + d' + a_2 + a')} \tag{3}$$

$$e(G = 2|a_1, d_1, a_2, d_2, S) = \frac{(d_1 + d')(d_2 + d')}{(d_1 + d' + a_1 + a')(d_2 + d' + a_2 + a')} \tag{4}$$

Inspecting emission probabilities

```
emission_dat <- tidyr::crossing(
  target = c(0,1,2),
  source1 = c(0,1,2),
  source2 = c(0,1,2),
)
get_emission_probs <- function(state_tuple) {
  purrr::pmap_dbl(emission_dat,
                  function(target, source1, source2)
                    emission_prob(target, state_tuple, c(2 - source1, 2 - source2), c(source1, source2))
}
emission_dat <- dplyr::mutate(emission_dat,
               emission_probs_11 = get_emission_probs(c(1, 1)),
               emission_probs_12 = get_emission_probs(c(1, 2)),
               emission_probs_22 = get_emission_probs(c(2, 2))
)
emission_dat %>% knitr::kable()
```

target	source1	source2	emission_probs_11	emission_probs_12	emission_probs_22
0	0	0	0.9247159	0.9111570	0.9247159
0	0	1	0.9247159	0.4772727	0.3281250
0	0	2	0.9247159	0.0433884	0.0156250
0	1	0	0.3281250	0.4772727	0.9247159
0	1	1	0.3281250	0.2500000	0.3281250
0	1	2	0.3281250	0.0227273	0.0156250
0	2	0	0.0156250	0.0433884	0.9247159
0	2	1	0.0156250	0.0227273	0.3281250
0	2	2	0.0156250	0.0020661	0.0156250
1	0	0	0.0596591	0.0867769	0.0596591
1	0	1	0.0596591	0.5000000	0.3437500
1	0	2	0.0596591	0.9132231	0.0596591
1	1	0	0.3437500	0.5000000	0.0596591
1	1	1	0.3437500	0.5000000	0.3437500
1	1	2	0.3437500	0.5000000	0.0596591
1	2	0	0.0596591	0.9132231	0.0596591
1	2	1	0.0596591	0.5000000	0.3437500

target	source1	source2	emission_probs_11	emission_probs_12	emission_probs_22
1	2	2	0.0596591	0.0867769	0.0596591
2	0	0	0.0156250	0.0020661	0.0156250
2	0	1	0.0156250	0.0227273	0.3281250
2	0	2	0.0156250	0.0433884	0.9247159
2	1	0	0.3281250	0.0227273	0.0156250
2	1	1	0.3281250	0.2500000	0.3281250
2	1	2	0.3281250	0.4772727	0.9247159
2	2	0	0.9247159	0.0433884	0.0156250
2	2	1	0.9247159	0.4772727	0.3281250
2	2	2	0.9247159	0.9111570	0.9247159

We can get a bit more overview by listing those observations which are most likely under a given state. Starting with state 11:

```
emission_dat %>%
  dplyr::filter(emission_probs_11 > emission_probs_12 & emission_probs_11 > emission_probs_22) %>%
  knitr::kable()
```

target	source1	source2	emission_probs_11	emission_probs_12	emission_probs_22
0	0	1	0.9247159	0.4772727	0.328125
0	0	2	0.9247159	0.0433884	0.015625
0	1	2	0.3281250	0.0227273	0.015625
2	1	0	0.3281250	0.0227273	0.015625
2	2	0	0.9247159	0.0433884	0.015625
2	2	1	0.9247159	0.4772727	0.328125

This makes sense, as these are homozygous target states for which source1 has at least one of the target alleles, and source2 has an opposing homozygote. We don't have to check for state 22, as it's going to be just the opposite symmetric.

For the heterozygous state 12 we have:

```
emission_dat %>%
  dplyr::filter(emission_probs_12 > emission_probs_11 & emission_probs_12 > emission_probs_22) %>%
  knitr::kable()
```

target	source1	source2	emission_probs_11	emission_probs_12	emission_probs_22
1	0	0	0.0596591	0.0867769	0.0596591
1	0	1	0.0596591	0.5000000	0.3437500
1	0	2	0.0596591	0.9132231	0.0596591
1	1	0	0.3437500	0.5000000	0.0596591
1	1	1	0.3437500	0.5000000	0.3437500
1	1	2	0.3437500	0.5000000	0.0596591
1	2	0	0.0596591	0.9132231	0.0596591
1	2	1	0.0596591	0.5000000	0.3437500
1	2	2	0.0596591	0.0867769	0.0596591

We see that these are all possible heterozygous target genotypes, interestingly even those for which the two sources have the same genotype.

An important insight here is that for a diploid model such as this one, we should not filter our observations at which both sources have the same genotype, or even two similar homozygotes. The key is that even at those sites, the three states have differing probabilities which may help with decoding. An exception are cases where either all three genomes are hom-ref, or all three genomes are hom-alt. While in these cases, there is minimal advantage for homozygous states (see complete table above), but it's so minimal that we can skip those.

Equilibrium states

The diploid equilibrium states follow Hardy-Weinberg equilibrium given certain ancestral source proportions. In general, given source proportions p_1 and $p_2 = 1 - p_1$, we have equilibrium probabilities

$$p_{11} = (p_1)^2 \tag{5}$$
$$p_{12} = 2p_1p_2 \tag{6}$$
$$p_{22} = (p_2)^2 \tag{7}$$

By construction, the equilibrium probabilities sum to 1.

Let's take the example where proportions are 0.97 and 0.03 for the two haploid states, respectively:

```
equilibrium_probs(0.03)
```

```
## [1] 0.9409 0.0582 0.0009
```

Transition probabilities

Transition probabilities are described from one state (index `i`) to another state (index `j`), and are considered *conditional* on state `i`, and then normalised across index `j` (i.e. rows).

Given the constraints set by the equilibrium probabilities, there is a single parameter which controls the transition probabilities, which is the expected length (in units of SNPs) of one of the states (we pick state 2 here). We denote the expected length of (haploid) state 2 by l_2. The expected length of state 1 is then determined by the haploid state proportions, following detailed balance:

$$\frac{l_1}{l_2} = \frac{p_1}{p_2} \tag{8}$$

which means that $l_1 = p_1 * l_2/p_2$.

So now, were we to describe transition probabilities in haploid states, then the transitions would be simple:

$$\begin{pmatrix} 1 - 1/l_1 & 1/l_1 \\ 1/l_2 & 1 - 1/l_2 \end{pmatrix} \tag{9}$$

This translates to the diploid state space as follows:

$$(A_{ij}) = \begin{pmatrix} \cdot & 2/l_1 & 1/(l_1)^2 \\ 1/l_2 & \cdot & 1/l_1 \\ 1/(l_2)^2 & 2/l_2 & \cdot \end{pmatrix} \tag{10}$$

where the dots on the diagonal simply are fixed by the row-normalization. The factor 2 on two of the middle off-diagonals reflect the fact that from a homozygous state, there are two ways to switch to a heterozygous state, and the squared values on the corner off-diagonals reflect the very small probability to switch from one homozygous state to another, by simultaneously switching both haplotypes.

We can look at an example, using a proportion of 0.03 of the minor component, and an expected length of 100 SNPs for the minor component:

```
transition_probs(0.03, 100)
```

```
##            [,1]         [,2]         [,3]
## [1,] 0.9993813 0.0006185567 9.565310e-08
## [2,] 0.0100000 0.9896907216 3.092784e-04
## [3,] 0.0001000 0.0200000000 9.799000e-01
```

which shows the large numbers on the diagonal, reflecting the fact that states don't switch all the time, and the extremely low number on the top right, which gives the almost non-existent probability for both haplotypes to switch simultaneously from state 1 to state 2. Note also the highly asymmetrical structure reflecting the very different lengths of the two states (reflecting of course the very different proportions).

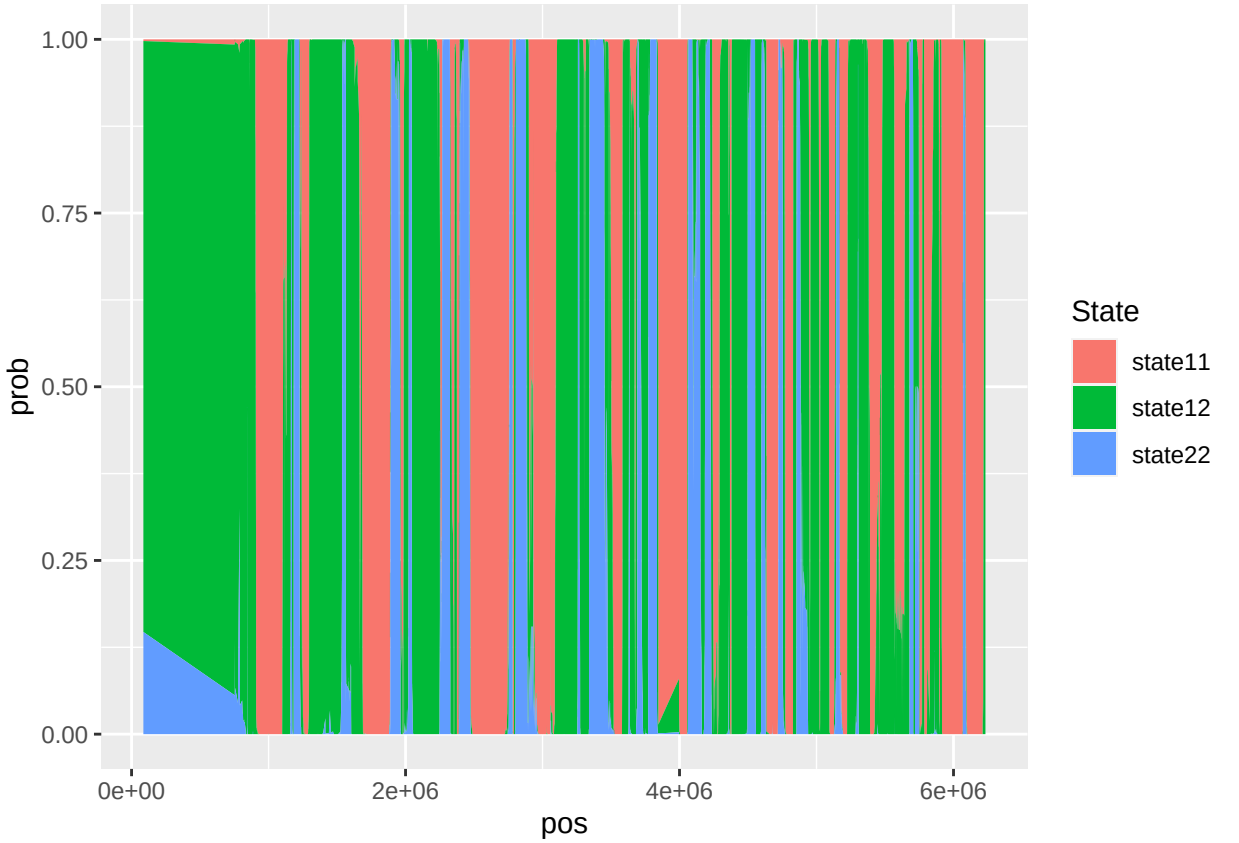
Posterior State Decoding

We can use the forward-backward algorithm to compute posterior state probabilities given the data. For the data, we here load real data from the Iceman, and two reference genomes, Stuttgart (Source 1) and Loeschbour (Source 2).

```
# I have prepared a test file with the first 10,000 SNPs of the input file
dat <- readr::read_tsv("~/Data/merge_vcf.iceman_obs.wgs_short.tsv") %>%
  dplyr::transmute(
    chrom = chr,
    pos = pos,
    genotype = target_dr,
    source_a = purrr::map2(s1_anc, s2_anc, c),
    source_d = purrr::map2(s1_dr, s2_dr, c)
  )
```

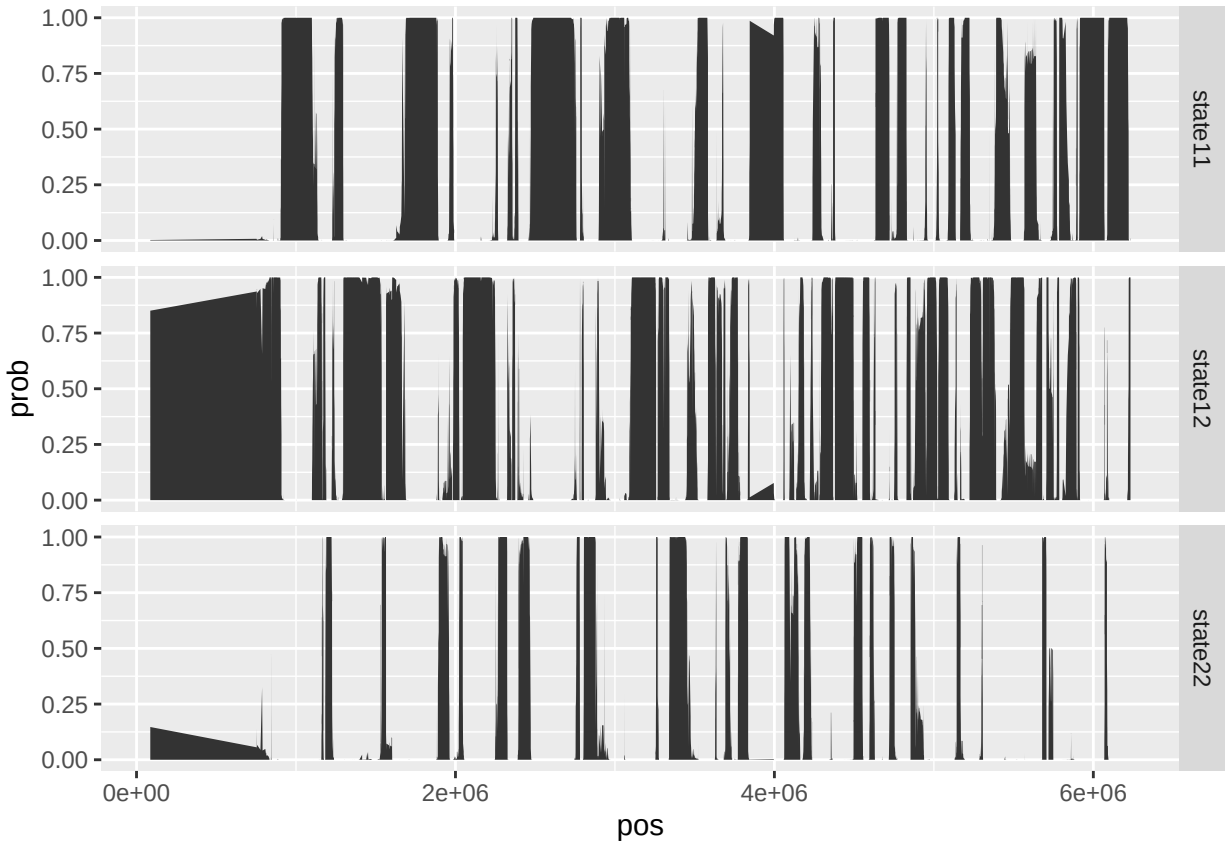
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   chr = col_double(),
##   pos = col_double(),
##   target_dr = col_double(),
##   s1_dr = col_double(),
##   s1_anc = col_double(),
##   s2_dr = col_double(),
##   s2_anc = col_double()
## )
```

```r
# three states in Hardy-Weinberg eq.
eq_probs <- equilibrium_probs(0.03)

# transition probabilities from state i to state j are conditional on state i,
# so normalised across the second index
trans_matrix <- transition_probs(0.03, 1000)

fwd_dat <- run_forward(dat, trans_matrix, 0.1, 0.1)
fwd_vec <- fwd_dat[[1]]
scaling_facs <- fwd_dat[[2]]
bwd_vec <- run_backward(dat, scaling_facs, eq_probs, trans_matrix, 0.1, 0.1)

# posterior_normalisations:
post_norms <- colSums(fwd_vec * bwd_vec)

post_norm_matrix <- matrix(rep(post_norms, 3), nrow=3, byrow = TRUE)

posterior_probs <- fwd_vec * bwd_vec / post_norm_matrix

for_plotting <- dat %>%
  dplyr::mutate(state11 = posterior_probs[1,],
                state12 = posterior_probs[2,],
                state22 = posterior_probs[3,]) %>%
  tidyr::pivot_longer(c(state11, state12, state22), names_to = "State", values_to = "prob")

for_plotting %>%
  ggplot() + geom_area(aes(x = pos, y = prob, fill = State))
```



We can also break down the resulting states into facets, perhaps showing more clearly the differnet length scales of the three states:

```
for_plotting %>%
  ggplot() + geom_area(aes(x = pos, y = prob)) + facet_grid(rows = vars(State))
```



References

Peter, Benjamin M. 2020. "100,000 Years of Gene Flow Between Neandertals and Denisovans in the Altai Mountains." *bioRxiv*. bioRxiv. https://doi.org/10.1101/2020.03.13.990523.