Willard Ford
Group 27

Project Report: wf-admixture

Introduction:

My tool, wf-admixture, generates a global ancestry measure for each sample given a set of individuals and their corresponding genotypes across a set of variants. The global ancestry measure is simply a proportion of each individual that can be attributed to each of the k ancestral genotypes. The commonly used tool of this type, and what inspired wf-admixture, is called Admixture. These outputs can be used on both population or individual scales. For populations, admixture measures can inform the relative makeup of each population as a sum or combination of other populations and even be used with other genetic information to determine historical migrations that correspond with genetic influxes of new populations. For individuals, admixture measures can show what percentage of each ancestral population composes their genome. However, there are several caveats to all admixture methods including wf-admixture. While the ancestral may seem to correspond to specific population groups, this is not guaranteed and what could look like a “European” population group actually represents a set of allele frequencies that are most common in the people who make up that population. Because of this caveat, especially on an individual level, many qualifications must be made and efforts to verify your output should be considered before widely sharing your data.

Methods:

We structure the problem with 2 matrices: F, a set of J minor allele frequencies for each of the K ancestral populations, and Q, the contribution of each ancestral population to each of the I individuals. Then our problem can be interpreted as what set of allele frequencies and population contributions is most likely given our input set of genotypes.

This problem has been shown impossible to solve analytically therefore we must use numerical methods. Additionally, we must assume that each individual is independent from every other individual in our dataset. By filtering out families and closely related individuals using standard plink commands we can achieve a close enough approximation to generate useful results.

Given our assumptions we can generate a Hardy Weinberg equilibrium shown below.

$$
\begin{aligned}
\Pr(1/1 \text{ for i at SNP j}) &= \Big[\sum_k q_{ik} f_{kj}\Big]^2 \\
\Pr(1/2 \text{ for i at SNP j}) &= 2\Big[\sum_k q_{ik} f_{kj}\Big]\Big[\sum_k q_{ik}(1-f_{kj})\Big] \qquad (1) \\
\Pr(2/2 \text{ for i at SNP j}) &= \Big[\sum_k q_{ik}(1-f_{kj})\Big]^2.
\end{aligned}
$$

From this we'd like to use the Method of Likelihood Estimators to generate our matrices F and Q. The log likelihood function, which when maximized will give our desired output, can be directly calculated from our probabilities by summing over our genotype inputs. It is shown below:

$$
L(Q,F) = \sum_i \sum_j \Big\{ g_{ij} \ln\Big[\sum_k q_{ik} f_{kj}\Big] + (2-g_{ij}) \ln\Big[\sum_k q_{ik}(1-f_{kj})\Big]\Big\}, \qquad (2)
$$

As previously stated, this function is impossible to calculate analytically and we must resort to numerical methods. wf-admixture uses an Expectation-Maximization (EM) algorithm to iteratively calculate subsequent steps of each matrix F and Q. The derivation for the EM formulas for F and Q can be derived by assuming only 1 matrix is being updated at a time and that the other contains constant values of 0 or 1. The method is described further in Tang et al.

$$
\begin{aligned}
f_{kj}^{n+1} &= \frac{\sum_i g_{ij} a_{ijk}^n}{\sum_i g_{ij} a_{ijk}^n + \sum_i (2-g_{ij}) b_{ijk}^n}, \qquad (3) \\
q_{ik}^{n+1} &= \frac{1}{2J} \sum_j \Big[g_{ij} a_{ijk}^n + (2-g_{ij}) b_{ijk}^n \Big], \qquad (4)
\end{aligned}
$$

$$
a_{ijk}^n = \frac{q_{ik}^n f_{kj}^n}{\sum_m q_{im}^n f_{mj}^n}, \quad b_{ijk}^n = \frac{q_{ik}^n (1-f_{kj}^n)}{\sum_m q_{im}^n (1-f_{mj}^n)}.
$$

Each iteration takes O(I*J*K^2) time to run. Leading to an asymptotic runtime of O(n*I*J*K^2) where n corresponds to the number of total iterations required. This is impacted by the choice of stopping threshold where a looser stopping threshold will take less iterations but may produce a less useful result.

Given this background the method of wf-admixture can be simplified to the following steps:

1. Filter samples for families and closely related individuals.
2. Initialize each sample in Q to a uniform distribution across all but 1 ancestral population and assign outsized proportion to the final, randomly chosen k'th population.
3. Initialize F from Q and the known genotypes of each individual
4. Use the EM algorithm to iteratively update F and Q until the difference between subsequent log-likelihoods falls below some threshold.

Results:

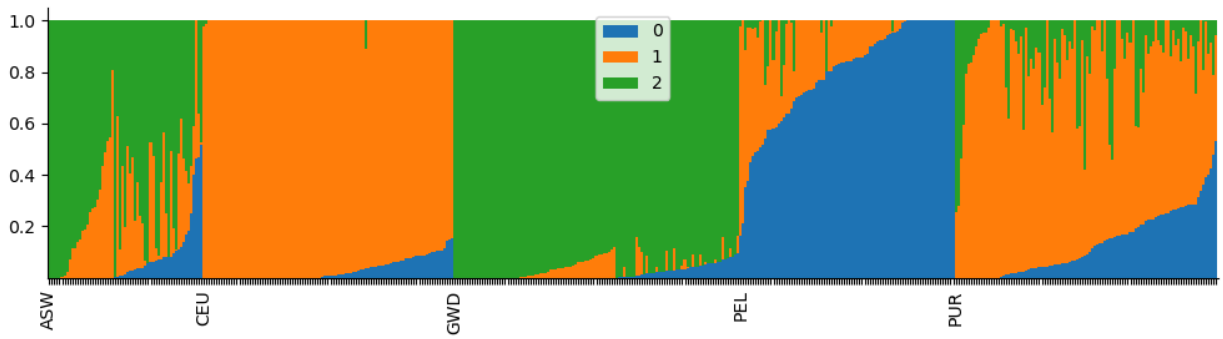
I benchmarked wf-admixture against admixture on a subset of the 1000 Genomes publicly available dataset. I first downloaded the VCF file for chromosome 21 and filtered it down to a set of 462 individuals with known population groups from assignment 2. Then I further filtered down the number of variants to eliminate any linkage disequilibrium and then to a random set of 1000 variants. Then I ran wf-admixture and admixture on this dataset with k = 3. No parallelization was used in either tool.

Tool	Runtime	# Iterations	Time/Iteration
wf-admixture	8 hours, 24 minutes, 29 seconds	3463 iterations	8.74 seconds per iteration
admixture	21 seconds	13 iterations	1.62 seconds per iteration

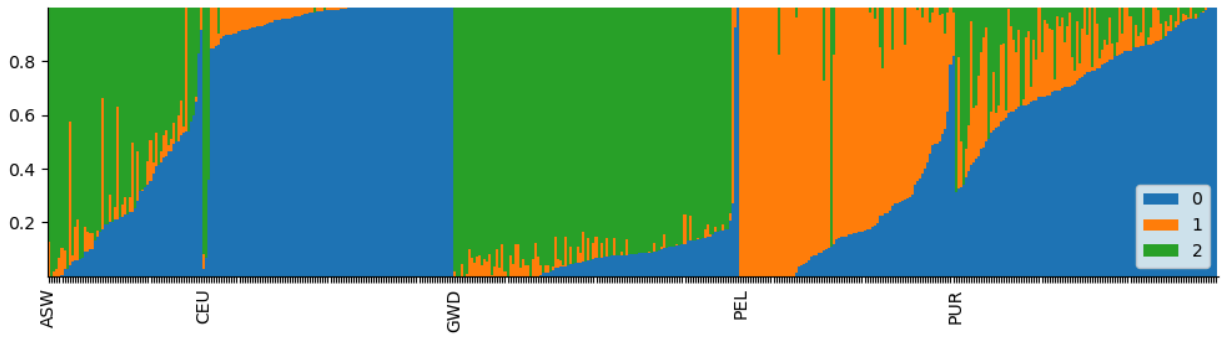
We can see that wf-admixture took significantly longer than admixture by several orders of magnitude. However this difference cannot be attributed to the time-cost of each iteration alone. Instead we see that the largest difference lies in the number of iterations, where my EM algorithm approached the stable point much slower than admixture. This will be further explained in the discussion.

Despite these differences in runtime, the generated admixture proportions align rather closely, indicating that wf-admixture was able to correctly discern real information.

admixture output:



wf-admixture output:



Although the percentages inside each population group of wf-admixture don't exactly match admixture, by eye, they are well within the range where the output can lead to useful insights. And, as mentioned previously, the ancestral populations do not actually exist and as such neither output can purport to be *correct*. However, agreement from two different tools indicates that real genetic patterns were likely uncovered.

Discussion:

Challenges:

In building wf-admixture, I faced difficulty understanding many of the algorithms used by more refined tools. Not only have I never taken a linear programming or numerical optimization class, I had never even built any large scale iterative algorithms. Simply understanding the meaning and derivation of the log likelihood functions and the EM algorithm well enough to implement was not trivial.

There was also a learning curve when reading in our genotype data. The standard input from plink is a .bed file which contains information about each sample bitwise. Learning how to efficiently read in this data was a small challenge.

Lastly, I tried several different Q initialization methods before settling on the one noted above. I found that a pure uniform distribution too often led to a steady state where each

iteration of Q and F was identical to the last and failed to produce real results. I then tried assigning each sample to an ancestral population at random however the abundance of 0's in the Q matrix similarly caused the EM algorithm to fail. I eventually settled on choosing one ancestral population at random to contribute 0.9 to each individual's genotypes and the remaining K-1 ancestral populations would have a uniform non-zero distribution summing to 0.1.
Improvements:

Several improvements can still be made to further optimize wf-admixture. Primarily by implementing a block relaxation algorithm akin to what admixture uses. Block relaxation involves only updating Q or F, but not both, on each iteration which allows the assumption of convex linearity. This allows for the use of much more efficient algorithms which both run faster per iteration and require less iterations overall. admixture also uses a form of acceleration where it heuristically approximates the distance from the maximum log-likelihood and increases its rate of change accordingly. I'd like to implement both improvements.

I would also like to parallelize more steps. While I did add the option for parallelizing each iteration in wf-admixture, it does not parallelize all operations and requires copying large matrices across multiple threads which might ultimately slow down the process for large datasets. It should be possible to avoid this data copying in python but the method is not yet clear to me, but when attempting to reach these levels of speed it would be far more efficient to rewrite the program in a lower level language instead.

Code Availability:

All code is freely available for any use: https://github.com/WillardFord/wf-admixture

References and Research:

Alexander et al. https://genome.cshlp.org/content/19/9/1655

Moreno-Estrada et al.
https://journals.plos.org/plosgenetics/article%3Fid=10.1371%2Fjournal.pgen.1003925

Tang et al. https://pubmed.ncbi.nlm.nih.gov/15712363/