Comparing Parametric and Nonparametric Methods of Genetic Population Structure Detection

Henry Wittich 6 May 2022

Abstract

Non-random inheritance of genetic markers across a population creates population structure, and these patterns of genetic variation across a population can be used to infer the ancestry of individuals within it. Tools for detecting genetic population structure can be divided into two categories: parametric approaches, which use statistical models to infer ancestry of individuals and assign them to clusters, and nonparametric approaches, which group individuals into clusters based on genetic similarity metrics.(1) Here, I perform a benchmarking study to compare the run-time, memory usage, and accuracy of three different methods for population structure: PCAclust (2009), a nonparametric approach, and Snapclust (2018) and Fastbaps (2019), two parametric approaches. The tools were run on 48 simulated datasets of varying sizes and levels of population structure, as well as genotype data from the 1000 Genomes Project. While PCAclust ran the quickest, Fastbaps was almost as fast as PCAclust and it also used less memory and was considerably more consistent and accurate than PCAclust. Snapclust did not run the fastest, did not use the least memory, and did not produce the most accurate clusters. This suggests that, while nonparametric approaches to population structure detection are still the fastest and most flexible in terms of dataset size, speed optimizations in parametric approaches like Fastbaps have greatly improved their viability.

Introduction

Genetic markers are inherited traits that, due to forces such as geographic separation, gene flow, and evolutionary pressure, tend to get passed down in non-random ways, creating genetic population structure. These patterns of genetic variation across a population can be used to infer the ancestry of individuals within the population. The presence of population structure can have wide-ranging implications on the genetic study of a population, thus detecting and accounting for population structure has become an integral part of most genetic analysis pipelines.(1) For example, in genome-wide association studies (GWAS), which aim to identify genomic loci associated with different phenotypic traits, both local and global ancestry can create spurious correlations in the dataset, polluting the results with false positives.(2) Thus, for analyses like GWAS, as well as other phylogenetic, conservation ecology, microbiology, and even forensic studies, tools for detecting population structure must be implemented either to produce covariate information to adjust for the underlying genetic structure or to cluster the individuals into population groups.(3,4)

Tools for detecting genetic population structure can be divided into two distinct approaches: there are parametric methods, which use statistical models to assign individuals to subpopulations, and nonparametric methods, which cluster individuals by some measure of genetic similarity or dissimilarity.(1) Initially, parametric approaches were most commonly used for population structure detection due to their statistical power and ability to calculate the likelihood that an individual belonged to a subpopulation. Early methods like STRUCTURE and PARTITION, which use Bayesian inferencing approaches to estimate population allele

frequencies and estimate proportions of ancestry for each individual, defined the field, and were used widely to explore the genetic structure of different human as well as non-human populations.(5,6) ADMIXTURE is another early method that incorporates a maximum-likelihood (ML) model.(7) The main drawback of these methods is that they are computationally intensive and thus not very efficient at processing larger datasets. Recent parametric approaches to detecting population structure attempt to address this by expanding the framework of older methods like STRUCTURE and ADMIXTURE with speed-optimizing alterations. For example, FRAPPE and Snapclust are Bayesian and ML methods, respectively, that implement an expectation-maximization (EM) algorithm to improve on speed.(4,8) The BAPs family of tools, including BAPS, hierBAPS, and Fastbaps, is a series of methods each expanding upon the Bayesian framework of STRUCTURE with alterations for improved speed on large and complex datasets.(9–11) Fastbaps is a newer method that optimizes speed by combining three different approaches: first, it performs fast, low-level clustering with nonparametric methods, then it uses the Dirichlet process mixture model (DPM) implemented in hierBAPS to optimize *K*, the number of clusters in the population being studied, and finally it performs Bayesian hierarchical clustering (BHC) to assign individuals into subpopulations.(11)

However, with the rapidly improving ability to generate genetic sequence data thanks to next-generation sequencing methods, the speed and dataset size limitations of parametric approaches have led to a decrease in popularity.(1) As study populations get larger and the amount of sequencing information per individual increases, nonparametric approaches have increasingly been used because of their improved efficiency on large datasets. The industry standard for nonparametric approaches is principal component analysis (PCA), a dimension reduction-based method that can detect underlying structure within highly dimensional data.(1) PCA can be used in conjunction with any clustering method, but has been implemented successfully on genetic data with Gaussian mixture model (GMM) clustering because PCs are normally distributed.(12) Other nonparametric methods take a distance-based approach, such as AWclust, which clusters individuals based on allele-sharing distance.(13)

Here, I perform a benchmarking study to compare the run-time, memory usage, and accuracy of three different population structure detection methods: PCAclust (2009), which was implemented using PLINK and the Scikit-learn Python library, Snapclust (2018), and Fastbaps (2019).(14,15)

Methods

Benchmark datasets

To compare the performance of the three tools, genomic datasets were simulated using scrm and population structure was simulated using models of coalescent evolution from the R package, Coala.(16,17) The number of underlying populations and the size of each subpopulation were varied to alter the complexity and size of the datasets. Full parameter configurations are listed in Table 1. Each parameter combination was replicated three times, creating a total of 48 test datasets.

Parameter	Value(s)
Number of replicates	3
Number of clusters	5, 10, 15, 20
Cluster population size	100, 250, 350, 500
Sequence length	100,000
Mutation rate	0.05
Recombination rate	10
Migration rate	0.05
Proportion of population sampled	0.1

Table 1. Parameter combinations used for coalescent evolution models built using Coala R package.

To test the tools on non-simulated data, genotype data from phase three of the 1000 Genomes Project was used.(18) This contains whole genome variant-level phased genotype data from 2,504 individuals from 26 different global populations (5 super-populations), including over 88 million variants. Due to size limitations of the tools, a subset of this data – the variants from chromosome 5 – was used to perform the analysis.

Data pre-processing

The human genotype data was retrieved in variant call format (VCF) and had to be converted to PLINK binary format for input into the PCAclust algorithm, and a FASTA multiple sequence alignment (MSA) format for input into Fastbaps and Snapclust. The chromosome 5 VCF file was converted into PLINK2 binary files with PLINK in order to preserve the phased genotypes for constructing haplotypes downstream in the analysis.(14) Non-SNPs and variants with more than one allele were filtered out of the dataset. Next, to reduce the number of variants included in analysis, linkage disequilibrium (LD) pruning was performed with PLINK by first calculating pairwise r^2 correlation values between every SNP and then filtering out SNPs with an r^2 values greater than 0.2.(14) Before LD pruning, there were about 5,037,955 variants and after, there were 4,224,328. The LD pruned PLINK2 binary files were converted into PLINK1 binary files with PLINK for input into PCAclust and a VCF file with PLINK for converting into the input format of Fastbaps and Snapclust.(14) A Python script was written to convert the phased genotypes in VCF format into a fasta MSA by concatenating all of the SNPs from each haplotype of every individual into a sequence string.

The simulated genotype data was produced in fasta MSA format, and thus prepared to be run through Fastbaps and Snapclust. To prepare the datasets for input into PCAclust, the fasta files were first converted into VCF files using the tool SNP-sites.(19) Then, the VCF files were converted into PLINK binary files using PLINK.(14)

Running every tool

To run each tool, a Python wrapper script was written to run each command and track the run-time and memory usage.

PCAclust was performed in two steps. First, PCA was performed on the input PLINK binary files using PLINK, outputting an eigenvec and eigenval file.(14) For each dataset, the first 20 PCs were calculated. Then, a Python script written to perform GMM clustering with the Scikit-learn Python library.(15) The eigenval output file was used to calculate the proportion of variance explained (PVE) by each of the first 20 PCs. Only the first few PCs were selected to perform clustering on (95% cumulative PVE threshold). A GMM model was fit to the significant PCs and then the individuals in the dataset were assigned to clusters according to the model. The GMM hyperparameter, K, which represents the number of clusters to create, was set to the known number of populations for the simulated genomes and to 5 for the human genomes.

Fastbaps is an R package, and an R script was written to perform the analysis. The R package ape was also required to perform the analysis.(20) First, the fasta files were read into sparse matrix format.(11) The optimized BAPs prior was selected for the analysis and used to generate the initial fast clustering with Fastbap's optimise_prior() and fast_baps() functions.(11) Finally, the individuals were assigned to clusters with Fastbap's best_baps_partition() function.(11)

Snapclust is a method from the adegenet R package, and an R script was written to perform the analysis.(4,21) The R package ape was also required to perform the analysis.(20) The maximum-likelihood clustering was performed on the fasta sequences using adegenet's snap_clust() function.(4,21) A maximum k number of cluster of 30 was used. The prediction error was calculated using the Akaike information criterion (AIC) metric with adegenet's AIC.snapclust() function, and then this information was used to select the partition with the best k, number of clusters.(4,21)

Benchmarking time, memory, and accuracy

To record the run-time of every program, Python's time module was used. Run-time was measured as the time it takes to go from reading in the input files to writing the output files for each tool. Memory usage was recorded using the psutil Python library. The peak memory usage over the course of running each tool was recorded and used to compare their memory usage. To assess the accuracy of each tool when clustering the simulated datasets, the Fowlkes-Mallows index was calculating according to the following equation:

$$FM = \sqrt{\frac{TP}{TP + FP} * \frac{TP}{TP + FN}}$$

Where TP is the number of pairs of individuals that are present in the same cluster in both the simulated population and the tool output, FP is the number of pairs of individuals that are present in the same cluster in the tool output but not the simulated population, and FN is the number of pairs of individuals that are present in the same cluster in the simulated population but not in the tool output.

Results

PCAclust runs fastest, uses most memory

Across the 48 different simulated datasets, PCAclust ran the fastest on average, followed closely by Fastbaps. Snapclust was noticeably slower (Figure 1a). To see how the time

complexity of each tool scales with the size of the dataset, datasets were simulated with different numbers of subpopulations of different sizes, resulting in overall populations of different sizes (Table 1). The distribution of run-times for running each tool on datasets of different-sized populations are shown in Figure 1. Fastbaps had the least consistent run-time, and the run-time of the software didn't seem to scale with the size of the dataset (Figure 1c). Likewise, the run-time of both PCAclust and Snapclust increased with the size of the dataset, though the run-time of Snapclust increased more strongly than that of PCAclust (Figure 1b,d). This is consistent with the finding that on average Snapclust ran the slowest and PCAclust ran the fastest.

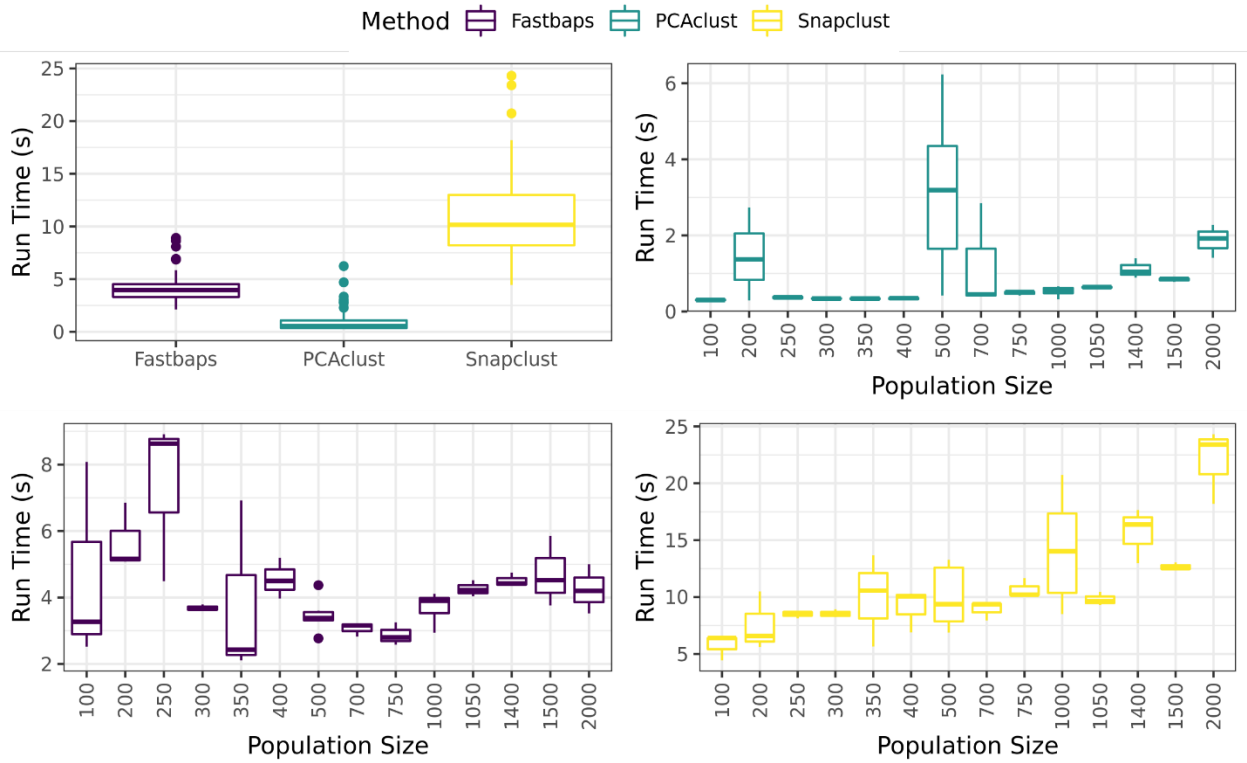

Figure 1. Run-time comparison of the three tools in seconds. (a) Distribution of run-times of each method on every simulated dataset. (b-d) Distribution of run-times of each program by size of the simulated population (number of sequences).

Across the 48 simulated datasets, PCAclust also used the most memory; Snapclust and Fastbaps used considerably less and were about even with each other (Figure 2a). For all three methods, memory usage didn't appear to scale with the size of the dataset, and it remained mostly constant for every population size (Figure 2b-d).

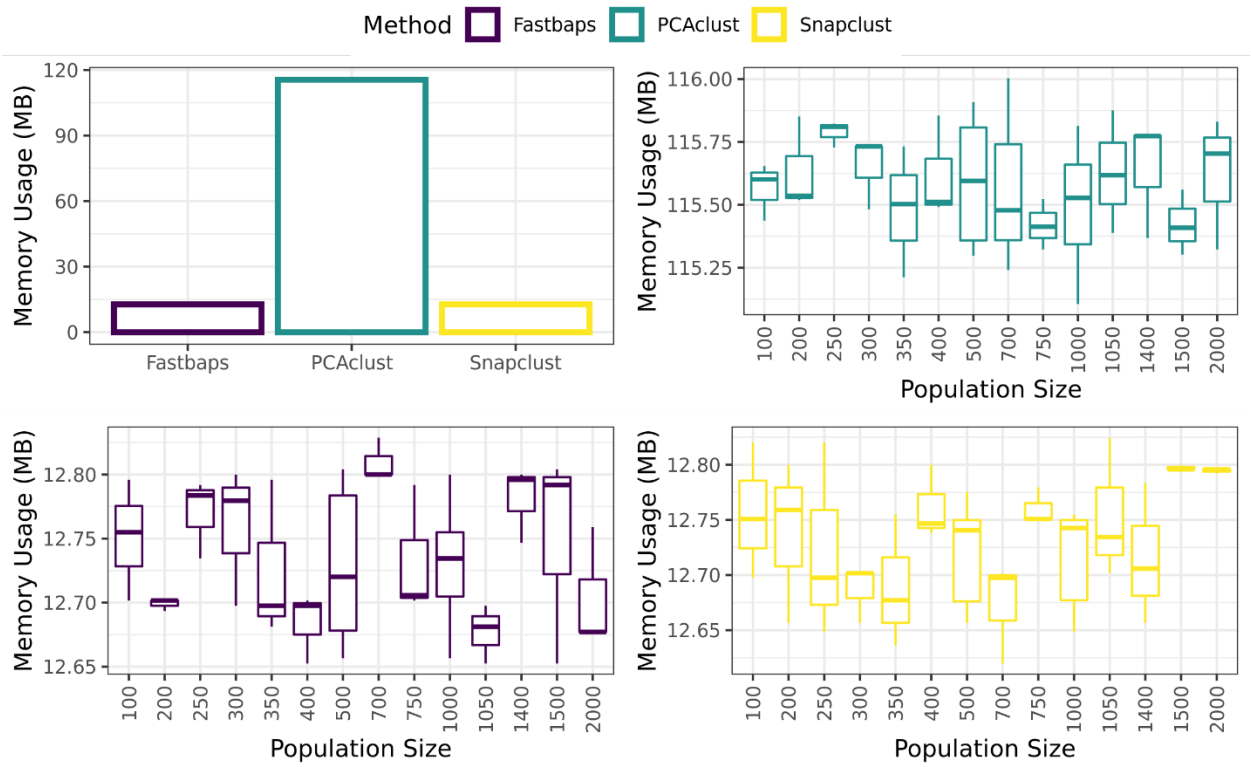

Figure 2. Memory usage comparison of the three tools in megabytes. (a) Average memory usage of each method on every simulated dataset. (b-d) Distribution of memory usage of each program by size of the simulated population (number of sequences).

Fastbaps produces most accurate clusters

All three tools were mostly accurate when clustering all 48 simulated datasets, with only a few runs scoring below an FM index of 0.8. To see how the accuracy of each tool was affected by the complexity of the dataset, datasets with different numbers of subpopulations were simulated to produce varying levels of population structure. The accuracy of each tool decreased somewhat equally as the complexity of the dataset increased (Figure 3.) In general, Fastbaps was the most accurate of the three tools, followed closely by Snapclust, and then PCAclust, which had a wide range of FM scores (Figure 3).

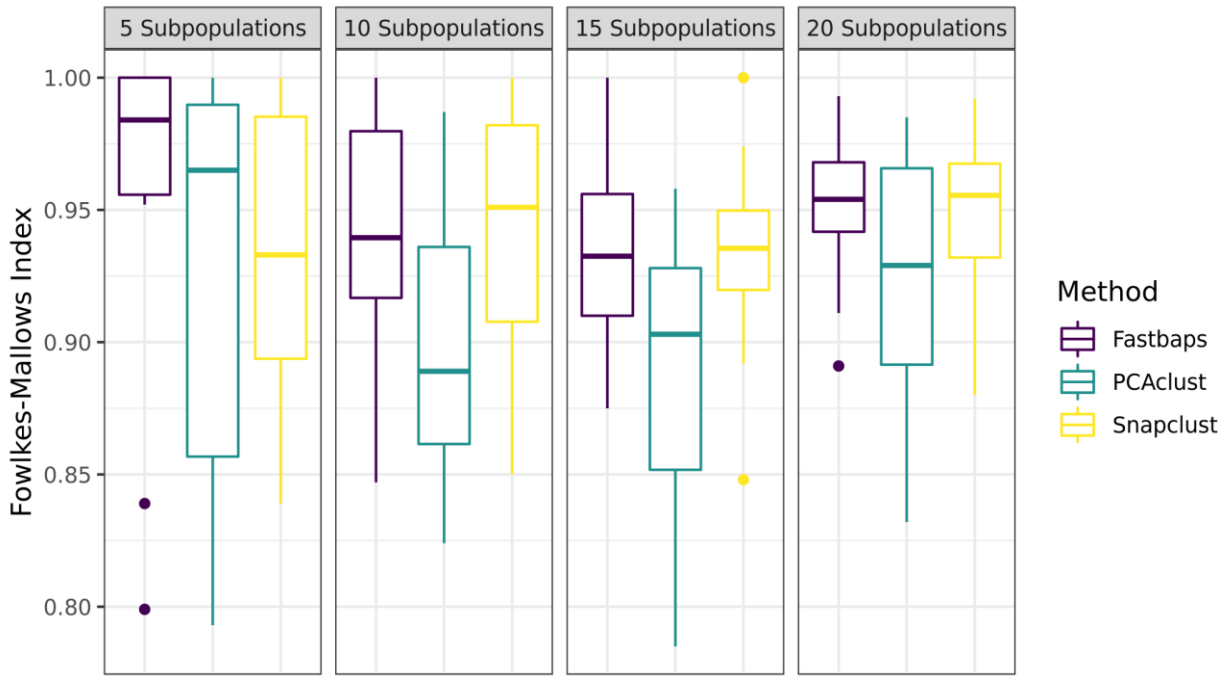

Figure 3. Accuracy comparison of the three tools by different numbers of subpopulations in the dataset. Accuracy is estimated by the Fowlkes-Mallows index.

Ability to produce biologically informative clusters

I also ran each tool on human genotype data to assess how they performed on non-simulated data. The individuals genotyped for the 1000 Genomes Project come from 26 different populations that can be grouped into five super-populations: Africans (AFR), Admixed Americans (AMR), Asians (EAS), Europeans (EUR), and South Asians (SAS).(18) Two of these populations have recently mixed ancestry, the AFR and AMR populations, making this dataset complex and difficult to cluster. On a PCA plot, these admixed populations don't cluster cohesively, and instead are spread across the plot between the population clusters (Figure 4).

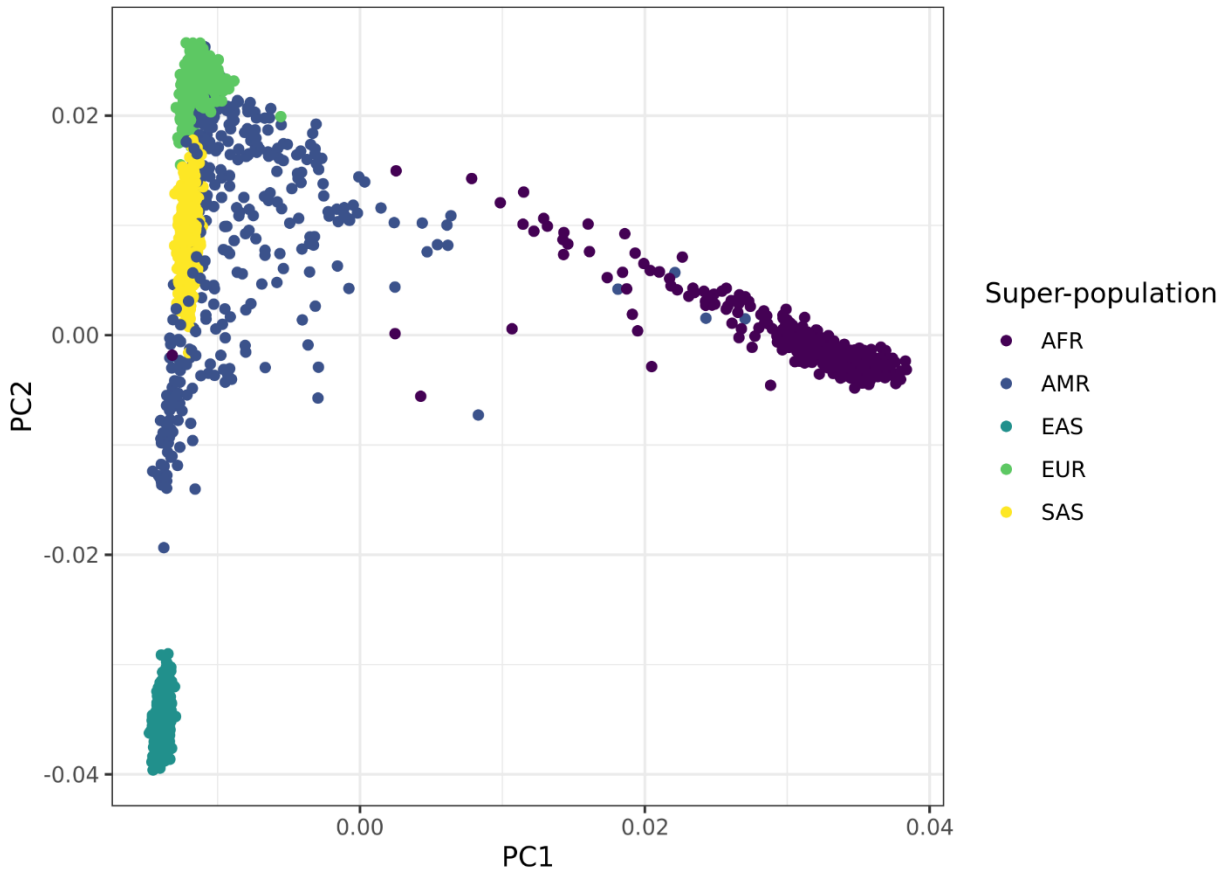

Figure 4. PC1 vs. PC2 plot of the 1000 Genomes Individuals, colored by super-population.

To compare how each of the three methods clustered the 1000 Genomes data, a PCA plot was produced using the first two PCs calculated during the PCAclust algorithm, and the individuals were colored according to the different clusters that each method assigned them to. Unfortunately, Snapclust never finished running on the human genotype data because the dataset was too large. Nevertheless, the PCA plots for PCAclust and Fastbaps are shown in Figure 5. Both methods essentially created four population clusters, though PCAclust was told there were five subpopulations. Both methods were able to cluster the EUR, EAS, and AFR populations. PCAclust divided the AFR population into 2 clusters, perhaps dividing them into East and West Africans. Both methods also struggled to cluster the more admixed populations; Fastbaps clustered the SAS and part of the AMR population with the EUR population and PCAclust clustered the entire AMR population with the EUR and the SAS population with the EAS. Notably, across replications, the clusters created by Fastbaps remained mostly consistent, while the clusters created by PCAclust were highly variable. Overall, both Fastbaps and PCAclust performed well given the complexity of the dataset and were able to generate biologically informative clusters.

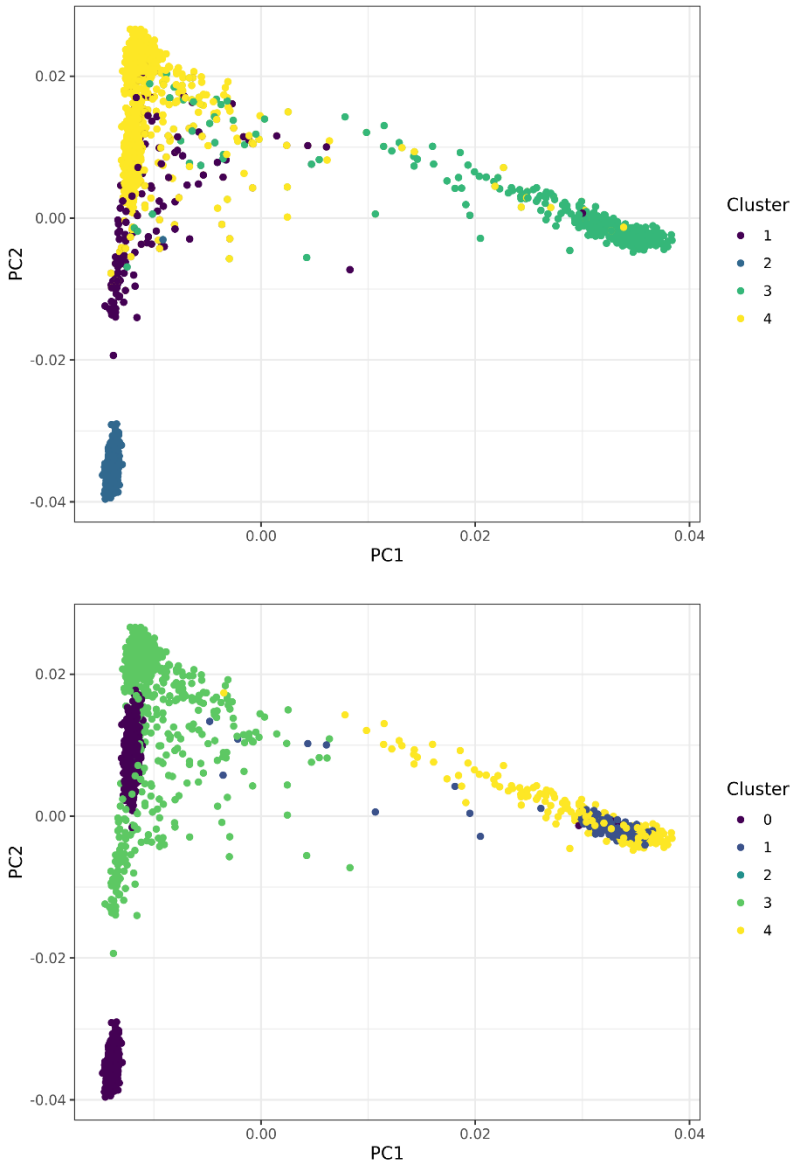

Figure 5. PC1 vs. PC2 plot of the 1000 Genomes Individuals, colored by clusters assigned by Fastbaps (a) and PCAclust (b).

Discussion

The run-time, memory usage, and accuracy of three methods for genetic population structure detection, PCAclust (2009), Snapclust (2018), and Fastbaps (2019) were benchmarked on 48 simulated populations and the 1000 Genomes phase 3 genotype data. As expected, PCAclust, the nonparametric method ran the fastest, though Fastbaps was not much slower despite being a parametric approach. The run-time of Fastbaps was less consistent than that of PCAclust and Snapclust, which is consistent with previous findings that the prior optimization step of Fastbaps can be highly variable depending on the dataset and can sometimes be slower than using a set prior.(11) Nevertheless, Fastbaps still ran faster than Snapclust, especially for larger datasets, which is consistent with the finding that Snapclust's run-time increases with k.(11)

Interestingly, while previous studies have demonstrated that Fastbaps at times uses significantly less memory than Snapclust, that was not replicated in this study; the memory usage was almost equal between the two.(11) Finally, Fastbaps was on average the most accurate tool of the three, which is consistent with findings that Fastbaps is more accurate than Snapclust.(11) Nevertheless, all three methods were fairly accurate, even at high levels of population structure. However, PCAclust, while accurate, was not very consistent, producing a high range of FM scores across its runs and clustering the 1000 Genomes individuals in dramatically different ways between replications. Fastbaps was additionally effective at consistently producing biologically informative clusters on human genotype data, which has not been shown previously.

In terms of usability, the three tools have different strengths. The input format of Fastbaps and Snapclust, a FASTA MSA, is not very compatible with variant-level genotype data and, as a result, datasets with a lot of variants. Likewise, the input format of PCAclust, a PLINK binary file, is a lot more receptive of different types of data; it is easier to convert sequence-level data into variant-level data and there are also many sophisticated tools such as PLINK that are able to manipulate and work with variant-level genotype data. All three of the tools were easy to install; Fastbaps and Snapclust are R packages that can be installed from CRAN. They were all also easy to run once the data was in the correct format; I was able to directly run the commands from the tools' manuals with little to no alterations.

For all of the above reasons, I would recommend either using PCAclust or Fastbaps, depending on whether or not you have variant-level or sequence-level genotype data. PCAclust was the fastest tool, though just marginally, and it only had a small increase in memory usage as a trade-off. While Fastbaps performed well on the large, simulated datasets as well as the human genotype data from chromosome 5, it wouldn't have been able to run on all 22 autosomes but PCAclust could, so for extremely large dataset, PCAclust is still a better option. Yet, Fastbaps was able to make accurate and biologically informative clusters on relatively little genotype information – there were only a couple of hundred SNPs generated in each of the simulations – while PCAclust was considerably less accurate. Furthermore, PCAclust was supplied the number of subpopulations in the dataset, which is not practical on a non-simulated dataset, while Fastbaps calculates the optimal number of clusters. In any case, I wouldn't recommend Snapclust, as it was slower and less accurate than Fastbaps and they both can be run on the same input data and with the same level of ease.

One major limitation in this study is the number of genomes simulated; at only 48 datasets and 3 replications for every parameter set, it is hard to generate results for accuracy, run-time, and memory usage that are representative of how the tools run. Additionally, it would be good to test variations of different parameters that I kept constant for my study, such as the migration rate, recombination rate, and sequence length. I think a better method of measuring the memory usage of the tools might be needed as well, as there was very little variation in the memory used for different sized datasets, and the amount of memory used by Snapclust and Fastbaps was nearly equal, which isn't consistent with prior findings.

References

1. Alhusain L, Hafez AM. Nonparametric approaches for population structure analysis. Hum Genomics. 2018 Dec;12(1):25.

2. Pritchard JK, Donnelly P. Case–Control Studies of Association in Structured or Admixed Populations. Theor Popul Biol. 2001 Nov;60(3):227–37.

3. Sethuraman A. On inferring and interpreting genetic population structure - applications to conservation, and the estimation of pairwise genetic relatedness [Internet] [Doctor of Philosophy]. [Ames]: Iowa State University, Digital Repository; 2013 [cited 2022 May 6]. p. 4615831. Available from: https://lib.dr.iastate.edu/etd/13332/

4. Beugin M, Gayet T, Pontier D, Devillard S, Jombart T. A fast likelihood solution to the genetic clustering problem. Hansen T, editor. Methods Ecol Evol. 2018 Apr;9(4):1006–16.

5. Porras-Hurtado L, Ruiz Y, Santos C, Phillips C, Carracedo Á, Lareu MV. An overview of STRUCTURE: applications, parameter settings, and supporting software. Front Genet [Internet]. 2013 [cited 2022 May 6];4. Available from: http://journal.frontiersin.org/article/10.3389/fgene.2013.00098/abstract

6. Dawson KJ, Belkhir K. A Bayesian approach to the identification of panmictic populations and the assignment of individuals. Genet Res. 2001 Aug;78(1):59–77.

7. Alexander DH, Novembre J, Lange K. Fast model-based estimation of ancestry in unrelated individuals. Genome Res. 2009 Sep;19(9):1655–64.

8. Tang H, Peng J, Wang P, Risch NJ. Estimation of individual admixture: Analytical and study design considerations. Genet Epidemiol. 2005 May;28(4):289–301.

9. Corander J, Marttinen P, Sirén J, Tang J. Enhanced Bayesian modelling in BAPS software for learning genetic structures of populations. BMC Bioinformatics. 2008 Dec;9(1):539.

10. Tonkin-Hill G, Lees JA, Bentley SD, Frost SDW, Corander J. RhierBAPS: An R implementation of the population clustering algorithm hierBAPS. Wellcome Open Res. 2018 Jul 30;3:93.

11. Tonkin-Hill G, Lees JA, Bentley SD, Frost SDW, Corander J. Fast hierarchical Bayesian analysis of population structure. Nucleic Acids Res. 2019 Jun 20;47(11):5539–49.

12. Lee C, Abdool A, Huang CH. PCA-based population structure inference with generic clustering algorithms. BMC Bioinformatics. 2009 Jan;10(S1):S73.

13. Gao X, Starmer JD. AWclust: point-and-click software for non-parametric population structure analysis. BMC Bioinformatics. 2008 Dec;9(1):77.

14. Chang CC, Chow CC, Tellier LC, Vattikuti S, Purcell SM, Lee JJ. Second-generation PLINK: rising to the challenge of larger and richer datasets. GigaScience. 2015 Dec;4(1):7.

15. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Mach Learn PYTHON. :6.

16. Staab PR, Zhu S, Metzler D, Lunter G. scrm: efficiently simulating long sequences using the approximated coalescent with recombination. Bioinformatics. 2015 May 15;31(10):1680–2.

17. Staab PR, Metzler D. Coala: an R framework for coalescent simulation. Bioinformatics. 2016 Jun 15;32(12):1903–4.

18. The 1000 Genomes Project Consortium, Corresponding authors, Auton A, Abecasis GR, Steering committee, Altshuler DM, et al. A global reference for human genetic variation. Nature. 2015 Oct 1;526(7571):68–74.

19. Page AJ, Taylor B, Delaney AJ, Soares J, Seemann T, Keane JA, et al. SNP-sites: rapid efficient extraction of SNPs from multi-FASTA alignments. Microb Genomics [Internet]. 2016 Apr 29 [cited 2022 May 6];2(4). Available from: https://www.microbiologyresearch.org/content/journal/mgen/10.1099/mgen.0.000056

20. Paradis E, Schliep K. ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. Schwartz R, editor. Bioinformatics. 2019 Feb 1;35(3):526–8.

21. Jombart T, Ahmed I. adegenet 1.3-1: new tools for the analysis of genome-wide SNP data. Bioinformatics. 2011 Nov 1;27(21):3070–1.