Fine-scale subpopulation detection via an SNP-based unsupervised method: A case study on the 1000 Genomes Project resources

Kridsadakorn Chaichoompu[1]
GIGA-R, BIO3, University of Liege, Avenue de l'Hôpital 11, 4000 Liège, Belgium

Alisa Wilantho, Pongsakorn Wangkumhang, and Sissades Tongsima[2]
National Biobank of Thailand, 111 Thailand Science Park, Phahonyothin Road, Khlong Neung, Khlong Luang, Pathum Thani, 12120, Thailand

Bruno Cavadas and Luísa Pereira[3]
Instituto de Investigação e Inovação em Saúde, Universidade do Porto, Rua Alfredo Allen, 208 | 4200-135 Porto, Portugal
Instituto de Patologia e Imunologia Molecular da Universidade do Porto, Rua Júlio Amaral de Carvalho, 45 | 4200-135 Porto, Portugal

Kristel Van Steen[1]
GIGA-R, BIO3, University of Liege, Avenue de l'Hôpital 11, 4000 Liège, Belgium
Email: kristel.vansteen@uliege.be[4]

SNP-based information is used in several existing clustering methods to detect shared genetic ancestry or to identify population substructure. Here, we present a methodology, called IPCAPS for unsupervised population analysis using iterative pruning. Our method, which can capture fine-level structure in populations, supports ordinal data, and thus can readily be applied to SNP data. Although haplotypes may be more informative than SNPs, especially in fine-level substructure detection contexts, the haplotype inference process often remains too computationally intensive. In this work, we investigate the scale of the structure we can detect in populations without knowledge about haplotypes; our simulated data do not assume the availability of haplotype information while comparing our method to existing tools for detecting fine-level population substructures. We demonstrate experimentally that IPCAPS can achieve high accuracy and can outperform existing tools in several simulated scenarios. The fine-level structure detected by IPCAPS on an application to the 1000 Genomes Project data underlines its subject heterogeneity.

Keywords: fine-level population structure; SNP-based population clustering; iterative clustering; 1000 Genome project.

[1] Supported by Fonds de la Recherche Scientifique (FNRS PDR T.0180.13)
[2] Supported by National Science and Technology Development Agency research grant (P18507439)
[3] Supported by European Regional Development Fund (COMPETE 2020 and Portugal 2020), and by Portuguese funds through Fundação para a Ciência e a Tecnologia (POCI-01-0145-FEDER-007274)
[4] Corresponding author

© 2022 The Authors. Open Access chapter published by World Scientific Publishing Company and distributed under the terms of the Creative Commons Attribution Non-Commercial (CC BY-NC) 4.0 License.

1. Introduction

Over time, evolutionary processes, for instance, natural selection, migration, and genetic drift, are the key factors that have contributed to the variations in genetic makeup and admixture of populations. Populations living in the same geographical area share a similar genetic background. Generally, genetic population substructure is analyzed using single nucleotide polymorphisms (SNPs) or haplotypes (derived from SNPs). Haplotype-based methods usually involve making haplotype inferences, which are computationally more cumbersome than direct allele frequency processing in SNP-based methods. Subpopulation detection has an important role in precision medicine and is beneficial for downstream analyses, especially for genome-wide association studies (*1*) and drug target identification for homogenized groups of individuals (*2*). Hence, the accuracy of population subtyping methods is crucial to generate sufficient power in these efforts.

Several SNP-based clustering algorithms exist, each leading to different clustering results and accuracy. Promising results in the context of SNP-based fine-scale population subtyping have been demonstrated for the following iterative algorithms. The iterative pruning Principal Component Analysis (ipPCA) method classifies individuals into groups without prior assumptions (*3*, *4*). The idea of iteratively creating latent Principal Component (PC) spaces can be applied to estimating the number of clusters. Spectral Hierarchical clustering for the Inference of Population Structure (SHIPS) is based on determining the number of clusters in the post-process (*5*). This method incorporates a divisive hierarchical clustering, which allows a progressive investigation of population structure. The SHIPS method estimates the number of clusters in a dataset via the gap statistic (*6*). The method produces a promising solution to infer fine-scale genetic patterns and has a low computational cost when applied to genome-wide SNP data. The graph-based method iNJclust is an alternative unsupervised clustering method (*7*). It operates iteratively on the Neighbor-Joining (NJ) tree. This framework uses the allele-sharing distance to build up the neighbor-joining tree. The behavior of the fixation index (F_{ST}) is utilized as a stopping criterion for this algorithm.

This paper presents detailed information on IPCAPS (implemented as R package (*8*)), including its underlying methodology, its performance via large-scale simulations, and a real-life application. IPCAPS is based on the ipPCA mechanism and relies on iterative PCA estimation from SNP data. However, all known limitations of ipPCA are addressed by IPCAPS. These include: resolving limitations of inflated type I errors caused by a 2-means algorithm, advancing on ipPCA's stopping criteria based on Tracy-Widom statistics (*3*) and the EigenDev heuristic (*4*), mowing away from a commercial implementation environment (MATLAB) toward a widely accessible environment (R environment (*9*)), and identifying outliers interfering with robust clustering. Most importantly, although ipPCA can capture general population structure, it cannot detect fine-level structure when F_{ST} is close to 0.001, such as is the case for Swedish-Norwegian samples (F_{ST}=0.001) or Polish-German samples (F_{ST}=0.0012) (*10*). A proof of concept for IPCAPS' ability to identify fine-scale structure was given before on a relatively small African dataset (*8*). In this paper, we not only showcase IPCAPS on 1000 Genomes data, using populations from African, American, European, East Asian, and South Asian ancestries, but we also demonstrate IPCAPS performance in a variety of theoretical scenarios via an extensive simulation study.

2. IPCAPS methodology for SNP-based subtyping

The current implementation of IPCAPS takes GWAS SNPs as input and iteratively creates PCA spaces to identify substructures in populations. Hence, prior to IPCAPS applications, Quality Control (QC) pre-processing steps largely coincide with standard practices to GWAS QC or SNP-based PCA analyses for population structure evaluations. In particular, data QC may include missing genotype filtering (missingness < 0.02), Hardy–Weinberg equilibrium (HWE) testing ($p<0.001$), and linkage disequilibrium (LD) pruning ($r^2<0.1$). Each missing genotype is replaced by the most common value (*11*); see also supplementary section S1. QC processing steps are followed by a data matrix construction for IPCAPS analysis: rows of this data matrix represent individuals, and the columns represent SNPs. SNPs are encoded as 0, 1, and 2, reflecting the number of minor alleles present at the corresponding loci. As a consequence, the encoded data matrix contains numeric values suitable for standard PCA. The data matrix is subsequently normalized by a zero-mean and unit variance procedure. In case all individuals contain only a single genotype at some loci, the normalized value is zero representing no variation. In practical applications, this issue frequently occurs for common alleles.

IPCAPS' core methodology can be broken down into the following steps (see Supplementary Figure S1 for a graphical workflow):

Step 1: In each iteration, select genotype data X according to the remaining individuals from the previous iteration; however, the whole data are used for the first iteration. The matrix X contains N rows and M columns representing a number of individuals and a number of SNPs, respectively. The SNP matrix is normalized using zero-mean and unit variance methods.

Step2: Construct a covariance matrix from matrix multiplication XX^T in order to reduce complexity for computation.

Step 3: Extract principal components (PCs) from the matrix XX^T as $XX^T = UDU^T$, where U represents eigenvectors and D is a diagonal matrix of positive eigenvalues of XX^T eigenvalues. A matrix of eigenvectors is used as PCs. For faster computation, PCs are calculated partially (*12*) according to the estimated high-impact number of PCs (P) in *Step 4* from the previous iteration. The matrix PCs contains N rows and P columns representing a number of individuals and a number of PCs, respectively.

Step 4: Calculate the EigenFit value from the matrix D (Step 3), defined as in Eq. (1):

$$EigenFit = max(D), \quad (1)$$

where D is a vector of differences defined as in Eq. (2)

$$D = (|L_1 - L_2|, |L_2 - L_3|, \dots , |L_{N-1} - L_N|) \ , \quad (2)$$

with L_i the eigenvector corresponding to the logarithm of eigenvector i (*i=1,...,N*). If the EigenFit is less than a user-specified threshold, then stop the iteration and define the current set of individuals as a subgroup. If not, continue to the next step. In this step, the P high-impact PCs is also estimated to be used in *Step 3* in the next iteration, where P is derived from P' as defined in Eq. (4) and (5):

$$P' = \sum_{i=1}^{N-1} I_{[i \le m]} \tag{4}$$

$$I_{[i \le m]} = \begin{cases} 1 & i \le m \\ 0 & i > m \end{cases} \tag{5}$$

with $i = 1, 2, 3, \ldots, N$, and m is an order of EigenFit in vector D. To have at least a 3D PC space for clustering, if P' is less than or equal to 3, then $P = 3$, otherwise $P = P'$.

Step 5: Apply RubikClust (*13*) on P high-impact PCs. Higher-dimensional outlier detection via RubikClust is used to enhance the stability of IPCAPS clustering, which identified outliers are allocated to separate clusters.

Step 6: Check if the number of subgroups obtained from RubikClust equals 1. If so, submit PCs to MIXMOD clustering in *Step 7*. If not, skip to *Step 9*.

Step 7: Apply MIXMOD clustering on PCs via the function mixmodCluster in the R package Rmixmod (*14*) and the Bayesian information criterion (BIC) (*15*) to determine the optimal number of subtypes (clusters). Additional details are provided in supplementary section S2.

Step 8: Check if the number of subgroups obtained from MIXMOD clustering equals 1. If so, then stop this iteration and define the current set of individuals as a subgroup. If not, continue to the next step.

Step 9: Check if the number of individuals in obtained subgroups is less than a pre-specified cutoff by users. If so, then stop this iteration and define the current set of individuals as a subgroup (defined as a group of outliers). If not, continue to the next step.

Step 10: Calculate pairwise F_{ST} for all pairs of subgroups. If F_{ST} is more than a user-specified threshold, then continue to the next iteration in step 1. If not, the pairs with F_{ST} less than the threshold are combined and defined as a single cluster.

Note that IPCAPS methodology combines three stopping criteria: 1) checking whether the EigenFit is lower than a prespecified threshold (*Step 4*), 2) determining whether the fixation index (F_{ST}) value between two clusters is lower than a threshold (*Step 10*), 3) checking whether a number of individuals in each cluster is lower than a customizable cutoff. Regarding the latter, if the minimum cluster size is low (i.e., 3-5), then too many sparse subgroups may be obtained. Hence, a balance needs to be sought between many potentially small, highly homogeneous, clusters or a few less homogeneous clusters that have sufficient power to be followed up for characterization in downstream analyses. When analyzing a dataset with a small number of individuals, for example, <100 individuals, it may be more practical to allow for minimally 5 to 10 individuals per final IPCAPS group. Our findings from an application on 1000 Genomes data (see Section 3) motivates 0.18 as maximum for the EigenFit threshold and the simulation scenario I (supplementary section S4) suggests 0.03 as minimum. The choice of F_{ST} threshold depends on the number of markers and samples available for subtyping. However, we motivate in supplementary section S3 the default of 0.0008.

3. Datasets

To assess the performance of IPCAPS we generated synthetic data with FILEST (*16*) according to 3 main scenarios. Simulation scenario I aimed to investigate a type I error, simulation scenario II aimed to assess the accuracy of IPCAPS, and simulation scenario III targeted quantifying scalability and speed. Cluster agreement between two clustering results was assessed via the Adjusted Rand Index (ARI) using the R Package mclust (*17*). The maximal attainable ARI value is 1, representing an identical match between 2 groups, while a negative value represents a mismatch.

3.1. *Simulation scenario I: test for Type I error*

The objective of simulation scenario I is to examine the type I error rate of our method. Ideally, there should not be any error if we apply IPCAPS to a single homogeneous population. In other words, in this case, the IPCAPS algorithm should only reveal 1 group; the initial population. We compared our method to other iterative pruning-based clustering methods such as ipPCA (*3*, *4*), iNJclust (*7*), and SHIPS (*5*). Moreover, we simulated one population with 500 individuals and 10,000 SNPs without any outliers and did so 100 times (i.e., 100 replicates). The parameter settings for FILEST are listed in Table 1 and the supplementary section S4.

3.2. *Simulation scenario II: test for accuracy*

The objective of simulation scenario II is to determine the accuracy of IPCAPS. Comparative iterative pruning-based methods for clustering are the same as for simulation scenario I: ipPCA, iNJclust, and SHIPS. For scenario II, we simulated 100 replicates of 10,000 SNPs and 500 individuals per population. For the settings SII-1, SII-3 and SII-5, two populations were simulated. We added an additional population in the settings SII-2, SII-4 and SII-6. The adopted F_{ST} values represented pairwise genetic distances as before (Hudson's fixation index) and ranged from 0.0008 to 0.005. We selected the lowest F_{ST} as 0.0008 according to the result in the supplementary section S3, and the highest F_{ST} as 0.005 according to the genetic distance among clearly distinct European populations. To assess the impact of outliers, we added three outliers in the settings SII-3 and SII-4, and five outliers in the settings SII-5 and SII-6. An outlier was considered when it was separated into its own group or grouped with other outliers. All setting parameters are summarized in Table 1 and the supplementary section S5.

3.3. *Simulation scenario III: test for scalability and speed*

The objective of simulation scenario III is to check the scalability and speed of IPCAPS. In particular, we wanted to investigate which of the two factors, the number of individuals or the number of SNPs, has the most impact on computation time. We chose to compare IPCAPS to ipPCA only. The parameter settings for this simulation scenario were as follows. We simulated 100 replicates of two populations while fixing F_{ST} at 0.005. This single fixed value of F_{ST} was motivated by the fact that IPCAPS was able to accurately separate two populations with F_{ST}=0.005. For setting SIII-1, we fixed the number of input SNP to 10,000 and varied the number of individuals from 100 to 10,000. For setting SIII-2, we considered 1,000 individuals and varied the number of SNPs from 25,000 to 100,000. To measure the performance of IPCAPS in terms of computation time, we performed all experiments on the same 64-bit Linux cluster with the 2.2 GHz Intel Xeon 8-core processor and 128 GB of memory per node. Since the cluster was working routinely and we could not control other running processes, we reported the median of execution times from 100 replicates instead of the mean. All parameter settings are summarized in Table 1 and the supplementary section S6.

Table 1. Parameter settings for simulation studies of scenarios I, II, and III. Scenario I contains one setting. Scenario II contains six settings, and Scenario III contains two settings.

Parameters	Settings								
	SI	SII-1	SII-2	SII-3	SII-4	SII-5	SII-6	SIII-1	SIII-2
No. populations	1	2	3	2	3	2	3	2	2
No. outliers	0	0	0	3	3	5	5	0	0
Population Distance (F_{ST})	-	0.0008, 0.0009, 0.001, 0.002, 0.003, 0.004, 0.005						0.005	0.005
No. individuals per population	500	500						100, 2.5k, 5k, 7.5k, 10k	1k
No. SNPs	10k	10k						10k	25k, 50k, 75k, 100k
No. replications	100								

3.4. *Real-life scenario: application of genome-wide data using the 1000 Genomes Project*

The aim of this experiment is to check the performance of IPCAPS in a big real-life dataset (large matrix calculation usually causes a computational error) and to potentially refine the genetic structure of the 1000 Genomes data in view of the obtained ADMIXTURE profiles (version 1.3.0) (*18*). Initially, we obtained the 1000 Genomes dataset (*19*), which consisted of 3,609 individuals from 26 populations and 78,136,341 SNPs in total. All quality control (QC) steps were carried out in PLINK version 1.9 (*20*). The QC steps consisted in selecting only founders or unrelated individuals (--filter-founders), selecting only autosomal chromosomes 1-22 (--not-chr 0,x,y,xy,mt), filtering out SNPs in linkage disequilibrium (LD) blocks (--indep-pairwise 50 5 0.1), removing SNPs that disagree with the Hardy–Weinberg equilibrium (HWE) testing (--hwe 0.001), allowing individuals with call rate at least 95% (--mind 0.05), filtering out missing genotypes >2% (--geno 0.02), and removing SNPs with low minor allele frequency (MAF) (--maf 0.05). After data QC, there were 2,504 individuals and 127,526 SNPs left. We then performed a population clustering analysis using IPCAPS on the filtered data set after QC steps. Finding actual SNP-based discriminators between IPCAPS clusters was beyond the scope of this study. Since this dataset was huge, it required a lot of memory to perform the analysis. Therefore, all analyses were performed on the 64-bit Linux cluster with the 2.3 GHz Intel Xeon 24-core processor and 512 GB of memory per node.

4. Results

4.1. *Type I error (simulation scenario I)*

From the considered clustering methods, only IPCAPS and SHIPS did not split up the single population into subgroups (average ARI=1). Notably, ipPCA and iNJclust enforced two subgroups (average ARI=0) and 174.79 groups (average ARI=0), respectively. (see supplementary section S4)

Apart from testing for Type I error, we also estimated the minimum EigenFit value from the results of ipPCA because ipPCA forced to split the data into 2 clusters. The average EigenFit value was 0.03, therefore we could use this value as the minimum threshold of EigenFit.

4.2. *Accuracy (simulation scenario II)*

Overall, IPCAPS (red curve in Fig. 1) had optimal performance compared to ipPCA (blue), SHIPS (green), and iNJclust (yellow) in terms of accuracy expressed by average ARI estimated over 100 replicates when comparing observed and expected clustering methods. In particular, IPCAPS performed well with 100% accuracy when F_{ST}=0.002 for all simulation scenarios, while the other strategies performed less for the same F_{ST}=0.002. As for the other strategies, the performance of IPCAPS decreased for decreasing F_{ST}<0.002, although the accuracy reduction was least dramatic for IPCAPS compared to ipPCA, SHIPS, and iNJclust.

In the case of two populations without outliers (setting SII-1, Fig. 1A), IPCAPS and ipPCA gave similar results, but ipPCA performed slightly better for F_{ST}=0.0008. The average ARI of both methods increased to 0.8 when F_{ST}=0.001 and reached 1 when F_{ST}=0.002. SHIPS became highly accurate (ARI=1) only when F_{ST}>=0.004, while iNJclust performed poorly (ARI=0) in this setting.

In the case of three populations without outliers (setting SII-2, Fig. 1B), IPCAPS was more accurate than the other considered methods. The average ARI of IPCAPS reached 1 when F_{ST}=0.002, while the performance of ipPCA dropped in this setting, with ARI reaching 1 from F_{ST}=0.003 onwards. SHIPS showed similar performance in setting SII-2 as in the previous setting, SII-1. The average ARI of iNJclust started to increase when F_{ST}=0.003 and increased up to 0.8 but never reached 1.

In the case of two populations with three and five outliers (settings SII-3 and SII-5, Fig. 1C and 1E), IPCAPS maintained its good performance, similar to the simulation setting SII-1 with two populations and no outliers. The performances of ipPCA and SHIPS dropped in comparison to setting SII-1; iNJclust consistently performed poorly for all F_{ST} in this setting (ARI=0).

In the case of three populations with three and five outliers (settings SII-4 and SII-6, Fig. 1D and 1F), IPCAPS still performed similarly to the corresponding settings without outliers. The performances of ipPCA and SHIPS dropped in comparison to setting SII-2. Interestingly, iNJclust showed increased accuracy for F_{ST}>0.003. However, the average values of ARI of ipPCA, SHIPS, and iNJclust stayed lower than 1 in settings SII-4 and SII-6.

Focusing on simulation settings with outliers and visualizing the number of outliers detected versus F_{ST}, IPCAPS clearly detected the largest number of outliers in comparison to ipPCA, SHIPS, and iNJclust (Fig. 1G, 1H, 1I, and 1J). Recall that an outlier was considered when it was separated into its own group or was grouped with other outliers. Particularly in the settings SII-4 and SII-6 (Fig. 1H and 1J), iNJclust had a hard time identifying any outliers at all. Although ipPCA was able to identify outliers, for all scenarios, it could detect approximately 2 out of 3 in the settings SII-3 and SII-4 (Fig. 1G and 1H), and 4 out of 5 in the settings SII-5 and SII-6 (Fig. 1I and 1J). SHIPS could not detect outliers in the settings SII-3 and SII-4 (Fig. 1G and 1H), but it was able to identify approximately 1 out of 5 in the settings SII-5 and SII-6 (Fig. 1I and 1J).

4.3. *Scalability and speed (simulation scenario III)*

The average execution time of ipPCA (Fig. 1K – blue curve) exponentially grew according to the number of individuals, reaching >24,000 seconds for 10,000 individuals (setting SIII-1). In contrast, the average execution time of IPCAPS (Fig. 1K – red curve) was lower than ipPCA; it reached approximately 2,000 seconds for 10,000 individuals (setting SIII-1). For setting SIII-2 (Fig. 1L), the average execution time of IPCAPS and ipPCA was much lower than for setting SIII-1. The average execution time of ipPCA reached 150 seconds for 100K SNPs, while the average execution time of IPCAPS was slightly lower and less than 150 seconds for 100K SNPs.

4.4. *Real-life scenario: the 1000 Genome Project*

IPCAPS subtyping resulted in 24 groups (excluding the outliers) as shown in Fig. 2. There were five selected populations of East Asian, and IPCAPS could detect four groups (groups 1 to 4) since two closely related Chinese populations were in the same groups (CHB and CHS). Interestingly, the selected four admixed American populations were clustered into five groups (groups 5 to 9) because the Peruvian population (PEL) was mainly separated into two clusters (groups 8 and 9). This was due to the complex admixture found in most of the American populations in that one cluster of PEL had an ancestral background from European (cyan) (group 9), while another cluster of PEL did not (group 8). Five South Asian populations were rather clustered into five clusters. Group 11 was mainly mixed and mainly driven by ITU and STU. Group 12 (BEB) was slightly mixed with East Asian ancestry (light green), and this evidence agreed with what was found in Changmai et al. 2022 (*21*). Groups 10, 13, and 14 (PJL and GIH) were differentiated according to the European ancestry (cyan). Seven African populations were clustered into six groups, and the results agreed with what was described in Chaichoompu et al. 2019 (*22*). Groups 15, 16, and 17 were differentiated by the different admixed proportions of two ancestors (pink and brown). ESN and YRI were clustered in the same groups. ABC and ASW were clustered together but separated into two groups 18 and 19, and they were differentiated by the cyan European ancestor. Group 20, LWK, rather had a unique admixture profile. Five European populations were separated into four clusters (groups 21-24). FIN had a unique ancestral profile (red), as suggested in Wangkumhang et al. 2022 that they were the most distinct group amongst Europeans (*23*). GBR and CEU were in the same group (group 22), unlike IBS and TSI, which had similar ancestral patterns and were differentiated by the other small ancestral parts.

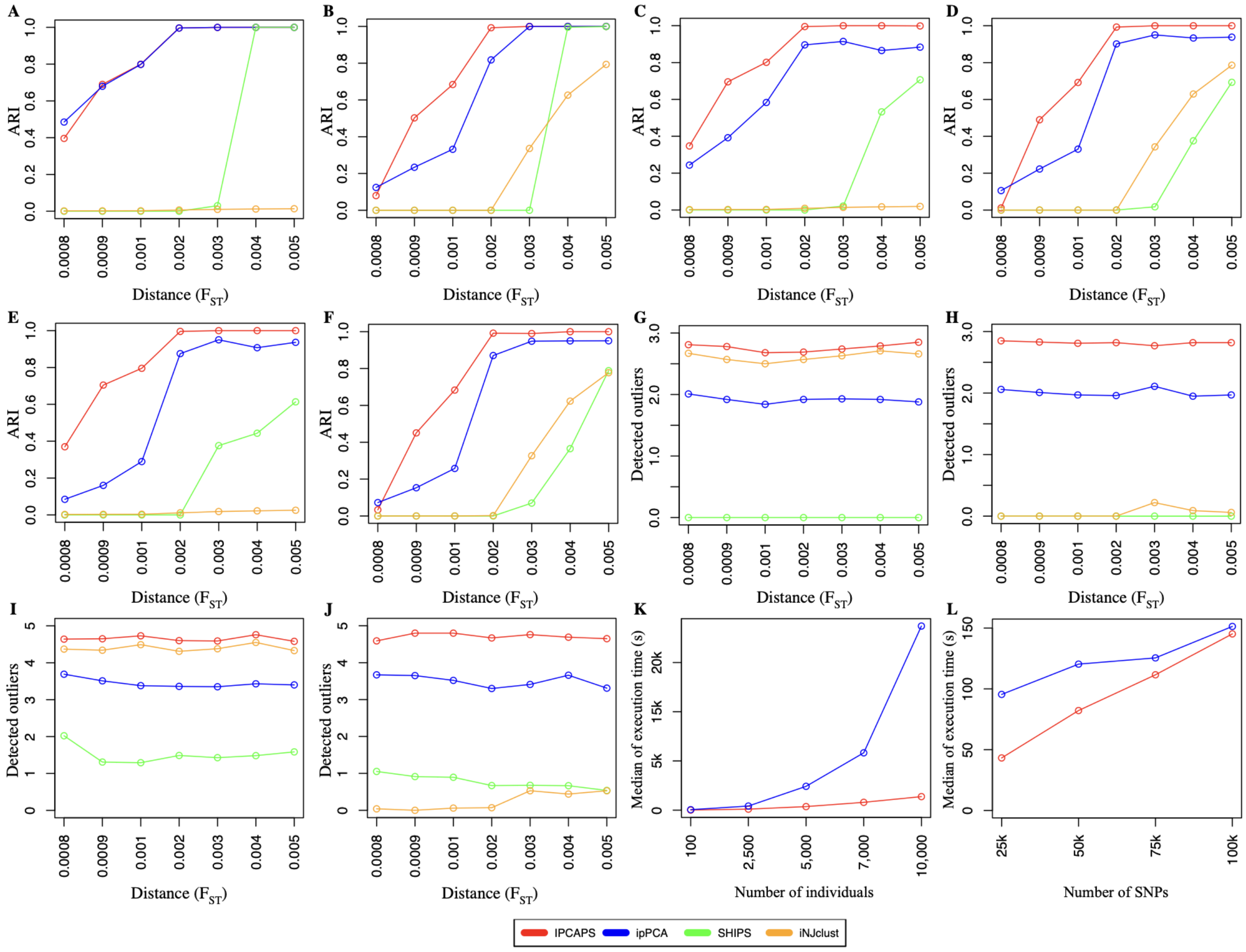

Fig. 1. The simulation results from different scenarios. The clustering accuracy for the simulated datasets without outlier is shown in A (2 populations) and B (3 populations). The clustering accuracy for the simulated datasets with 3 outliers is shown in C (2 populations) and D (3 populations). The clustering accuracy for the simulated datasets with 5 outliers is shown in E (2 populations) and F (3 populations). The number of detected outliers for the simulated datasets with 3 outliers are shown in G (2 populations) and H (3 populations). The number of detected outliers for the simulated datasets with 5 outliers are shown in I (2 populations) and J (3 populations). The median execution time when the number of individuals is scaled is shown in K, and when the number of SNPs is scaled is shown in L. See Table 1 for detailed information for all simulation settings.

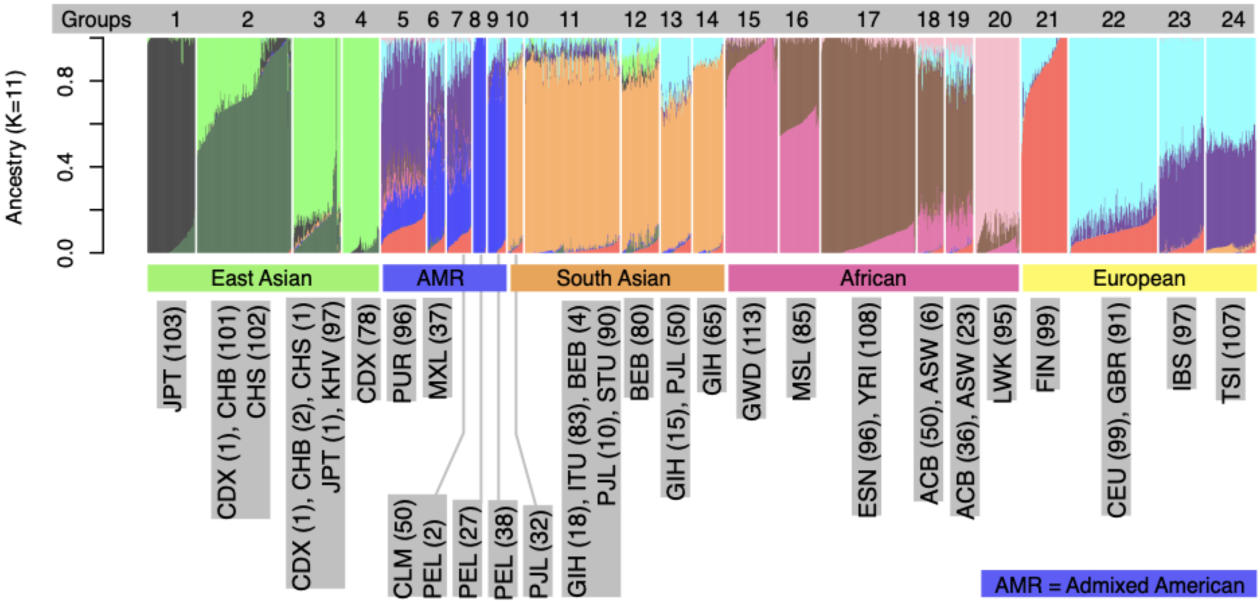

Fig. 2. IPCAPS results of the 1000 Genomes dataset depicted via ADMIXTURE (*18*) profiles (default options). The 24 identified IPCAPS groups are shown in the grey bar (without outliers).

5. Discussions

IPCAPS was generated in the quest for efficient subtyping of individuals at fine scale, using SNPs rather than multilocus haplotypes. It is an iterative clustering algorithm that was templated on ipPCA, yet combines three stopping criteria. As part of the IPCAPS development RubikClust provides a first rough assessment of substructure detection, identifying multi-dimensional outliers. Here, an outlier is an observation that is isolated from other observations in a PC-space. Outliers are collectively removed from further analysis but can be analyzed separately in a sensitivity analysis. The impact of outliers on clustering results with IPCAPS was assessed via several simulation studies, which entailed increasingly complex data structures (simulation scenario II). Among several explored clustering algorithms available from the R environment, MIXMOD served our purposes best, exhibiting excellent performance on complex datasets (see supplementary text S2). Notably, to further increase robustness of final conclusions, rather than choosing one clustering algorithm, multiple ones can be chosen, and a consensus clustering may be derived. The computational burden of IPCAPS depends on the number of individuals due to the dimensionality reduction prior to PCA via the XX^T technique, in which a dimension of the matrix becomes smaller.

EigenFit, which is used as one of the IPCAPS stopping criteria, is motivated by the drawback of EigenDev (*4*). The EigenDev value for a diverse and small subgroup is high and causes the failure in stopping clustering iteration in ipPCA. Hence, we have adjusted the calculation and provided the empirical simulation to check for the minimum threshold of EigenFit (see the section 5.1).

Estimated type I error rates for IPCAPS in our simulation scenario are zero due to the fact that IPCAPS has adopted the F_{ST} threshold of 0.0008 according to sample size and the number of SNPs. Accuracy is defined as the ability to retrieve existing substructure; the adjusted rand index method (ARI) is used to measure accuracy. In all simulation scenarios, IPCAPS generally outperforms ipPCA, SHIPS, and iNJclust. The ipPCA method has a higher average ARI than IPCAPS for F_{ST}=0.008 because ipPCA over separates data as it is observed that Type I error of ipPCA is high (100%). When testing for speed, it is observed that IPCAPS is faster than ipPCA. When dealing with complex population structures in our simulation scenario, IPCAPS delivers satisfactory results. In general, IPCAPS has excellent performance for both fine-level and large-scale settings as supported by experimental results. Initially, the objective of this paper was to develop a tool that deals with fine-level structure. IPCAPS accurately deals with the rough structures by splitting off bigger groups and then zooming in to find additional subtle structures.

Finally, in principle, it is possible to apply IPCAPS methodology to other data types then SNPs, as long as it makes sense to derive PCs, and distance-based stopping criteria (for instance F_{ST}) are adapted to the nature of the data at hand.

6. Conclusions

In this paper, we explained and motivated the components underlying IPCAPS subtyping and for the first time showed extensive simulation studies that underpin its outperformance compared to other iterative subtyping algorithms, namely ipPCA, iNJclust, and SHIPS. The simulated datasets used in all experiments were generated using our own tool FILEST. It allows simulating samples and complex data structures with and without outliers. Furthermore, IPCAPS was applied to big data from the 1000 Genomes project, and it revealed the potential of IPCAPS to detect general population structure, possibly non-linear in nature. Especially for populations in geographically confined regions, IPCAPS was shown to detect meaningful subgroups, which are otherwise hard to detect with classic PCA or ADMIXTURE. We recommend the use of ADMIXTURE or similar software tools to assist in interpreting obtained population subtypes.

7. Supplementary information and availability of software

The supplementary information, the datasets, the experimental scripts, and the results from this paper are publicly available on Zenodo (DOI: 10.5281/zenodo.7141144). IPCAPS is implemented as an R package that is publicly available on the CRAN repository (https://CRAN.R-project.org/package=IPCAPS), and the GitHub repository (https://github.com/kridsadakorn/ipcaps).

Acknowledgments

The authors thank Raphaël Philippart and Alain Empain for critical help with computing clusters at Consortium des Équipements de Calcul Intensif (CÉCI).

References

1. C. Medina-Gomez *et al.*, *Eur. J. Epidemiol.* **30**, 317–330 (2015).
2. C. Finan *et al.*, *Sci. Transl. Med.* **9**, eaag1166 (2017).
3. A. Intarapanich *et al.*, *BMC Bioinformatics*. **10**, 382 (2009).

4. T. Limpiti *et al.*, *BMC Bioinformatics*. **12**, 255 (2011).
5. M. Bouaziz, C. Paccard, M. Guedj, C. Ambroise, *PLOS ONE*. **7**, e45685 (2012).
6. R. Tibshirani, G. Walther, T. Hastie, *J. R. Stat. Soc. Ser. B Stat. Methodol.* **63**, 411–423 (2001).
7. T. Limpiti, C. Amornbunchornvej, A. Intarapanich, A. Assawamakin, S. Tongsima, *IEEE/ACM Trans. Comput. Biol. Bioinform.* **11**, 903–914 (2014).
8. K. Chaichoompu *et al.*, *Source Code Biol. Med.* **14** (2019), doi:10.1186/s13029-019-0072-6.
9. R Core Team, R: A Language and Environment for Statistical Computing (2020), (available at https://www.R-project.org).
10. S. C. Heath *et al.*, *Eur. J. Hum. Genet. EJHG*. **16**, 1413–1429 (2008).
11. D. Clayton, *snpStats: SnpMatrix and XSnpMatrix classes and methods* (2015).
12. Y. Qiu, J. Mei, authors of the A. library S. file A. for details, *rARPACK: Solvers for Large Scale Eigenvalue and SVD Problems* (2016; https://CRAN.R-project.org/package=rARPACK).
13. K. Chaichoompu *et al.*, KRIS: Keen and Reliable Interface Subroutines for Bioinformatic Analysis (2021), (available at https://CRAN.R-project.org/package=KRIS).
14. R. Lebret *et al.*, *J. Stat. Softw.* **67** (2015), doi:10.18637/jss.v067.i06.
15. G. Schwaziz, *Ann. Stat.* **6**, 461–464 (1978).
16. K. Chaichoompu, F. Abegaz, K. V. Steen, FILEST: Fine-Level Structure Simulator (2018), (available at https://CRAN.R-project.org/package=FILEST).
17. C. Fraley, A. E. Raftery, T. B. Murphy, L. Scrucca, *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation* (Department of Statistics, University of Washington, 2012).
18. D. H. Alexander, J. Novembre, K. Lange, *Genome Res.* **19**, 1655–1664 (2009).
19. A. Auton *et al.*, *Nature*. **526**, 68–74 (2015).
20. S. Purcell, C. Chang, PLINK 1.9. *BGI Cogn. Genomics*, (available at www.cog-genomics.org/plink2).
21. P. Changmai *et al.*, *PLOS Genet.* **18**, e1010036 (2022).
22. K. Chaichoompu *et al.*, *Hum. Genet.* (2019), doi:10.1007/s00439-019-02069-7.
23. P. Wangkumhang, M. Greenfield, G. Hellenthal, *Genome Res.*, in press, doi:10.1101/gr.275994.121.