TeraPCA: a fast and scalable software package to study genetic variation in tera-scale genotypes

Aritra Bose [1,†], Vassilis Kalantzis [2,†], Eugenia Kontopoulou [1,†], Mai Elkady [1], Peristera Paschou [3,*] and Petros Drineas [1]

[1] Computer Science Department, Purdue University, West Lafayette, IN, 47907, USA and
[2] IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA and
[3] Department of Biological Sciences, Purdue University, West Lafayette, IN, 47907, USA.

[*] To whom correspondence should be addressed. [†] Equal Contribution.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Principal Component Analysis (PCA) is a key tool in the study of population structure in human genetics. As modern datasets become increasingly larger in size, traditional approaches based on loading the entire dataset in the system memory (RAM) become impractical and out-of-core implementations are the only viable alternative.
Results: We present TeraPCA, a C++ implementation of the Randomized Subspace Iteration method to perform PCA of large-scale datasets. TeraPCA can be applied both in-core and out-of-core and is able to successfully operate even on commodity hardware with a system memory of just a few gigabytes. Moreover, TeraPCA has minimal dependencies on external libraries and only requires a working installation of the BLAS and LAPACK libraries. When applied to a dataset containing a million individuals genotyped on a million markers, TeraPCA requires less than five hours (in multi-threaded mode) to accurately compute the ten leading principal components. An extensive experimental analysis shows that TeraPCA is both fast and accurate and is competitive with current state-of-the-art software for the same task.
Availability: Source code and documentation are both available at https://github.com/aritra90/TeraPCA
Contact: ppaschou@purdue.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Principal Component Analysis (PCA) is perhaps the most fundamental unsupervised linear dimensionality reduction technique. It was invented by Pearson in the early 1900s (Pearson, 1901); and later reinvented and named by Hotelling in the 1930s (Hotelling, 1933, 1936). In statistical parlance, PCA converts a set of observations of possibly correlated variables into a set of linearly uncorrelated (orthogonal) variables called principal components (PCs). The seminal work of Luca Cavalli-Sforza and collaborators in the late 1970s (Menozzi *et al.*, 1978; Chisholm *et al.*, 1995) pioneered the application of PCA for the study of human genetic variation.

PCA analyses and plots appear in virtually *every single paper* that analyzes human genetic variation in order to make inferences about population structures. Given m samples genotyped on n genetic loci, it is well-known that applying PCA on the $m \times m$ covariance matrix that emerges by computing any reasonable notion of genotypic distance between every pair of samples using the n genotyped loci results in the observation that the leading PCs mirror geography, e.g. see (Novembre *et al.*, 2008; Wang *et al.*, 2010; Paschou *et al.*, 2014) for detailed discussions and examples. This observation was leveraged by (Price *et al.*, 2006; Patterson *et al.*, 2006; Price *et al.*, 2010) to derive one of the most established methods to account (and correct) for the confounding effects of population stratification in genome-wide association studies (GWAS). The method in (Price *et al.*, 2006; Patterson *et al.*, 2006; Price *et al.*, 2010) is essentially equivalent to using a small number of leading PCs as covariates in order to check for associations between genetic loci and affection status in statistical tests, and is implemented in the EIGENSTRAT software package which is routinely used in GWAS analyses to correct for population stratification. Other applications of PCA include the identification of sets of genetic loci that are ancestry-informative or are under selective pressure (Paschou *et al.*, 2007; Price *et al.*, 2006; Paschou *et al.*, 2008); and, when combined with other lines of evidence such as social structure and linguistics, the extraction of complex population histories and demographic structures (Bose *et al.*, 2017). We also note that PCA extracts the fundamental features of a dataset without complex computational modeling. Interestingly, even the output of model-based, more complex, methods to detect population structure (such as ADMIXTURE (Alexander and Novembre, 2009)) typically exhibits high correlation with the output of PCA, rendering further support to the significance of PCA in the analysis of human genetics data.

From a computational viewpoint, PCA essentially amounts to computing eigenvectors of the $m \times m$ (normalized) covariance matrix associated with the dataset at hand. When m does not exceed a few thousands, all eigenvectors can be computed by appropriate dense linear algebra routines in LAPACK, a Fortran 90 matrix factorization-based library which is widely used for solving systems of linear equations, least-squares problems, eigenvalue problems, and singular value problems (Anderson *et al.*, 1999). Matrix factorization-based dense eigenvalue solvers return all m eigenvectors with a time complexity in the order of $O(m^3)$, which becomes impractical as m, the number of samples, increases. Practical applications of PCA in population genetics only require the computation of those principal components (PCs) determined by the eigenvectors associated with only a few (say 10-20) of the largest eigenvalues. Computing a few of the leading eigenvalues and associated eigenvectors of large (sparse or dense) matrices is typically achieved by first projecting the original eigenvalue problem onto a low-dimensional subspace which includes an invariant subspace associated with the relevant eigenvectors. This low-dimensional subspace can be formed in many different ways, e.g., by means of subspace iteration or Krylov projection schemes and much work in the Numerical Analysis community has been devoted in understanding the theoretical properties of such approaches (Parlett, 1998; Saad, 2011). In particular, a variant of the family of Krylov projection schemes, the so-called Implicitly Restarted Arnoldi method (IRA), is the projection scheme of choice in FlashPCA2 (Abraham *et al.*, 2017), a software package which has been shown to outperform other PCA software packages, both in terms of memory usage and wall-clock time. On the other hand, recent advances in the design and analysis of Randomized Numerical Linear Algebra (RandNLA) (Drineas and Mahoney, 2016) algorithms have yielded novel insights as well as fast and efficient alternatives to approximate the leading principal components of large matrices (Halko *et al.*, 2011; Musco and Musco, 2015; Drineas and Mahoney, 2018; Drineas *et al.*, 2018). Indeed, FastPCA (Galinsky *et al.*, 2016) applied such randomized algorithms to perform PCA analyses in population genetics data.

This paper presents TeraPCA, a C++ software package to perform PCA of tera-scale genotypic datasets that can not fully reside in the system memory. TeraPCA is essentially an out-of-core implementation of the Randomized Subspace Iteration method (Rokhlin *et al.*, 2010; Halko *et al.*, 2011) and features minimal dependencies to external[1] libraries. As the amount of time spent on I/O typically dominates the wall-clock time in out-of-core scenarios, TeraPCA builds a high-dimensional initial approximation subspace by loading the dataset from secondary storage exactly once. The dimension of this initial approximation subspace can be controlled directly by the user. Each subsequent iteration of Randomized Subspace Iteration "corrects" the initial subspace so that an invariant subspace associated with the leading target eigenvectors is computed. The dataset needs to be accessed twice in each iteration, but, fortunately, a few steps of Randomized Subspace Iteration are typically sufficient in practice in order to get highly accurate approximations to the leading eigenvectors. Note here that the above idea is somewhat orthogonal to the ideas underlying IRA, which builds the approximation subspace in a vector-by-vector manner, thus necessitating a large number of dataset fetches from secondary storage to even form an approximation subspace whose dimension is equal to or slightly larger than the number of PCs that we seek to approximate.

TeraPCA was tested extensively on both real (Human Genome Diversity Panel, 1000 Genomes, etc.) and synthetic datasets. Our synthetic datasets were generated via the Pritchard-Stephens-Donelly (PSD) model (Pritchard *et al.*, 2000; Gopalan *et al.*, 2016). Our results suggest that TeraPCA is both fast and accurate and in most cases outperforms other out-of-core PCA libraries such as FlashPCA2. Specific highlights include the computation of the ten leading principal components of a dataset of one million samples genotyped on one million genetic markers (this dataset exceeds 3.5 TBs in uncompressed format) in about 13 hours (using a single thread) and in less than 4.5 hours (using 12 threads).

2 Methods

Simulated Datasets

The first group of the datasets used for our experiments was generated using the Pritchard-Stephens-Donelly's (PSD) model of simulating genotypes. In particular, a recent study (Gopalan *et al.*, 2016) simulated genotypic data by obtaining individual ancestry proportions from the PSD model to fit the 1000 Genomes dataset and then modelling the per-population allele frequencies using Wright's F_{ST} and the Weir & Cockerham estimate (Weir and Cockerham, 1984). We developed a multi-threaded C++ package which is essentially an efficient implementation of the R code developed in Tera-Structure (Gopalan *et al.*, 2016). We generated various datasets in order to evaluate TeraPCA's performance, with the number of markers ranging from 100,000 to 1,000,000 and the number of samples ranging from 5,000 to 1,000,000.

Table 1. Our data sets (simulated and real)

Dataset	Size (.PED file)	Size (.BED file)	# Samples	# SNPs
S_1 (simulated)	19 GB	120 MB	5,000	1,000,000
S_2 (simulated)	38 GB	239 MB	10,000	1,000,000
S_3 (simulated)	373 GB	24 GB	100,000	1,000,000
S_4 (simulated)	1.9 TB	117 GB	500,000	1,000,000
S_5 (simulated)	3.7 TB	233 GB	1,000,000	1,000,000
S_6 (simulated)	38 GB	2.4 GB	100,000	100,000
S_7 (simulated)	150 GB	9.4 GB	2,000	20,000,000
HGDP	615 MB	39 MB	1,043	154,417
1000 Genomes	8.4 GB	483 MB	2,504	808,704
PRK	2 GB	126 MB	4,706	111,831
T2D	1.8 GB	111 MB	6,370	72,457

Real Datasets

The Human Genome Diversity Panel (HGDP) dataset consists of 1,043 individuals genotyped at 660,734 SNPs, across 51 populations across Africa, Europe, Middle East, South and Central Asia, East Asia, Oceania, and the Americas (Cann *et al.*, 2002). We ran Quality Control (QC) on the data by filtering SNPs with minor allele frequency below 0.01 and subsequently pruning for LD using a window size of 1000 kb. Moreover, we set the variance inflation factor to 50 and set $r^2 > 0.2$, thus retaining 154,471 variants. We applied the same parameters for LD pruning on

[1] In contrast to FlashPCA2 which relies on the IRA implementation on the Spectra C++ library, TeraPCA comes with an in-house implementation of the Randomized Subspace Iteration algorithm.

the 1000 Genomes dataset which has 2,504 individuals sampled from 26 different populations across all continents genotyped at 39 million SNPs. After QC, we retained approximately 808,704 SNPs and ran our experiments on the pruned dataset.

We also tested the performance of TeraPCA on case-control data, which are ubiquitous in population genetics. We used the Wellcome Trust Case Control Consortium's (WTCCC) Type 2 Diabetes (T2D) and Parkinson's (PRK) datasets. The T2D dataset had 6,371 individuals (1,816 cases and 4,555 controls) genotyped on 313,654 SNPs and the PRK dataset had 5,000 individuals (2,000 cases and 3,000 controls) genotyped on 500,000 SNPs. We removed related samples from these datasets and pruned them using the aforementioned QC parameters resulting in datasets with 6,370 individuals genotyped on 72,457 SNPs for T2D and 4,706 individuals genotyped on 111,831 SNPs for Parkinson's.

TeraPCA

TeraPCA first normalizes the genotypes using the same procedure that was used by both FlashPCA (Abraham and Inouye, 2014) and FastPCA (Galinsky *et al.*, 2016) (see our supplementary material for details) and then applies Randomized Subspace Iteration in an out-of-core fashion.

The main parameters of TeraPCA are as follows (see our supplementary material for more details and our code release for full documentation):

1. Number of PCs to be computed (denoted by k). Default value is set to $k := 10$.
2. Number of contiguous rows of the SNP-major input matrix fetched from the secondary storage at each time unit (denoted by β). This can be user-defined or automatically determined based on the available system memory.
3. Dimension of the initial approximation subspace (denoted by s). Default value is set to $s := 2k$.
4. Convergence tolerance (denoted by tol). Default value is set to $\mathrm{tol} := 1e - 3$.

The wall-clock time of TeraPCA is affected by all of the above parameters. Clearly, reducing tol or increasing k results in an increase of the wall-clock time. Using a higher-dimensional approximation subspace, i.e., increasing s, might reduce the corresponding wall-clock time as it typically enhances convergence towards the k-leading eigenvectors. On the other hand, increasing the value of s also increases the amount of floating-point operations performed. Finally, since only a part of the dataset can fit in the system memory at any time unit, the choice of β is typically determined automatically by TeraPCA based on the size of the system memory. The total amount of time spent on I/O is largely independent of the value of β but we have observed that the value of β has an effect on the wall-clock time of the LAPACK routines.

3 Implementation and Discussion

The performance of TeraPCA was tested on both simulated and real-world genotypic datasets. All our experiments were performed at Purdue's `Brown` cluster on a dedicated node which features an Intel Xeon Gold 6126 processor running at 2.6 GHz with 96 GB of RAM and a 64-bit CentOS Linux 7 operating system. Table 1 lists the number of samples, number of SNPs, and size of each dataset. Datasets S_1 through S_7 are synthetic datasets and the remaining ones are real-world datasets. This section provides comparisons between TeraPCA and FlashPCA2. The latter has already been shown to be faster than previous methods such as FlashPCA (Abraham and Inouye, 2014), FastPCA (Galinsky *et al.*, 2016), etc. The results reported throughout the remainder of this section were obtained by setting the amount of system memory made available to TeraPCA (as well as FlashPCA2) to 2 GBs. This is precisely the amount of memory allowed to FlashPCA2 in prior work.

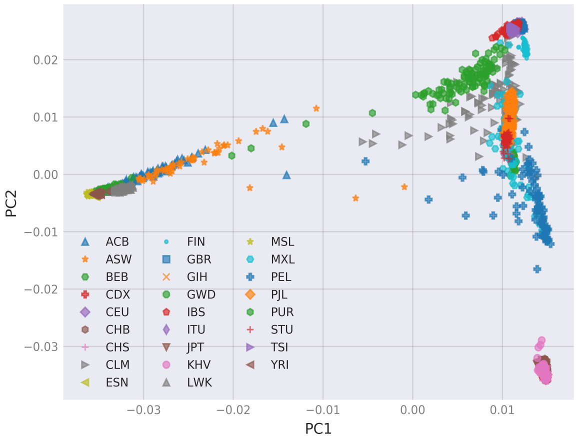

Fig. 1. Projection of the samples of the 1000 Genomes dataset on the top two left singular vectors (PC1 and PC2), as computed by TeraPCA.

3.1 Synthetic datasets

Datasets S_1 through S_5 in Table 1 have a fixed number of SNPs (equal to one million) and a varying number of samples (from 5,000 to one million). On the other hand, dataset S_6 was used to fine-tune prior state-of-the-art methods and contains 100,000 samples genotyped on 100,000 SNPs. S_7 was used to test the performance of TeraPCA on extremely rectangular matrices, where the number of SNPs heavily outnumbers the number of individuals.

We first consider the plots of the three leading principal components returned by both TeraPCA and FlashPCA2 for dataset S_6 (see Figure 1 in supplementary material). TeraPCA and FlashPCA2 show a complete visual agreement with each other and both libraries agree with the expected outcome of the PSD model. For this particular example, TeraPCA terminated in just under 40 minutes, while FlashPCA2 required 141 minutes[2].

Table 2 lists the wall-clock times achieved by TeraPCA when applied on datasets S_1 through S_7. For datasets S_4 and S_5, which were the largest ones in our collection, TeraPCA terminated after 7.3 and 13.2 hours respectively. On the other hand, FlashPCA2 did not terminate within the 50 hours limit that we imposed. TeraPCA outperformed FlashPCA2 on all synthetic datasets, with a speedup that ranged between 1.3 and 4.5, at least for those datasets where FlashPCA2 terminated within our 50 hour limit. We note that for all synthetic datasets the leading PCs returned by TeraPCA and FlashPCA2 showed perfect correlation as measured by the Pearson correlation coefficient (equal to one in all cases). To further test TeraPCA's performance on datasets where the number of SNPs heavily outnumbers the number of individuals, we applied it to S_7 and observed

[2] To be fair in our comparisons between TeraPCA and FlashPCA2, we performed multiple runs of FlashPCA2 on dataset S_6 in order to explore and understand its properties. In particular, we varied the convergence criterion in FlashPCA2 and recorded the resulting trade-off between wall-clock time and digits of accuracy for the top ten computed eigenvalues. Fixing the convergence tolerance in FlashPCA2 to three digits of accuracy and the maximum number of iterations of FlashPCA2 to 100 was the best choice in terms of the tradeoff between running time and accuracy (see Supplementary text for more details)

that even in a heavily under-determined system, TeraPCA outperformed FlashPCA2 by a factor of 2.9, with similar accuracy guarantees.

3.2 Real datasets

We first considered the Human Genome Diversity Panel (HGDP) dataset (Cann *et al.*, 2002). TeraPCA was marginally faster than FlashPCA2 and both libraries required about seven seconds. A plot of the projection of the HGDP dataset along the two leading PCs computed by TeraPCA is shown in Figure 2 in the supplementary material.Given the relatively small size of this dataset, we were able to compute the exact ten leading eigenvectors using LAPACK. Figure 2 reports the entry-wise error of the ten leading eigenvectors returned by TeraPCA. As expected, eigenvectors associated with the largest eigenvalues are captured more accurately since they converge faster.

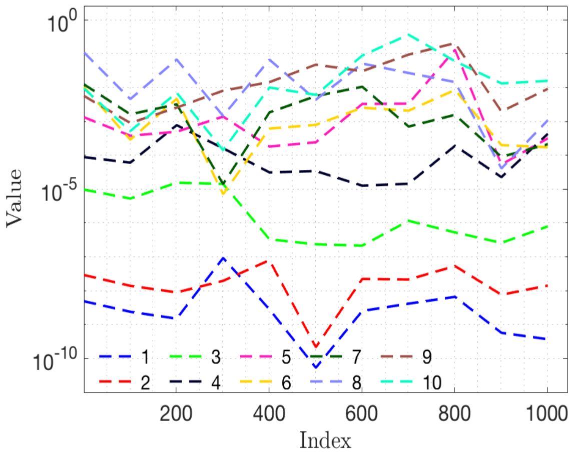

Fig. 2. Entry-wise relative error of the top ten leading eigenvectors returned by TeraPCA for the HGDP dataset, compared to the eigenvectors returned by LAPACK. The y-axis shows the relative error; recall that each eigenvector has 1,043 entries. We observe that the relative error is roughly the same for each entry of a specific eigenvector.

In addition, Supplementary Table 1 reports the relative and absolute errors of the ten leading eigenvalues returned by TeraPCA and FlashPCA2. For TeraPCA, the (much) higher accuracy in the approximation of the three-four leading eigenvalues is due to the fact that these approximate eigenvalues kept improving as Randomized Subspace Iteration kept iterating to approximate the trailing eigenvalues and eigenvectors. On the other hand, the accuracy in the approximation of the eigenvalues returned by FlashPCA2 was somewhat uniform for all eigenvalues.

TeraPCA and FlashPCA2 showed similar qualitative and computational performance on the pruned 1000 Genomes dataset (see Figure 1), with FlashPCA2 terminating slightly faster than TeraPCA. Notice that this dataset is also the one in which the number of SNPs outnumbered the number of individuals by the largest factor.

PCA is an essential tool to detect population stratication in GWAS. In order to evaluate TeraPCA's performance on real-world case-control studies, we applied it on WTCCC's T2D and PRK datasets. Like other real-world datasets, both FlashPCA2 and TeraPCA performed similarly, needing roughly the same wall-clock time. Execution of TeraPCA on these datasets can also be done in-core, as they fit in the system memory, leading to comparatively faster computation.

Table 2. Wall-clock running times comparisons for the datasets of Table 1 using a single thread and 2 GBs of system memory (* indicates no convergence after 50 hrs).

Dataset	TeraPCA	FlashPCA2	Speed-up
S_1	26.2 mins	33.3 mins	1.27
S_2	39.3 mins	87.5 mins	2.22
S_3	7.9 hrs	35.6 hrs	4.50
S_4	7.3 hrs	n/a*	∞
S_5	13.2 hrs	n/a*	∞
S_6	39.5 mins	141.1 mins	3.57
S_7	37.3 mins	106.5 mins	2.86
HGDP	6.5 secs	7.7 secs	1.22
1000 Genomes	4.3 mins	3.5 mins	0.81
T2D	96 secs	119 secs	1.24
PRK	76 secs	73 secs	0.96

3.3 Multithreading

The wall-clock times of TeraPCA and FlashPCA2 can significantly improve by executing the associated linear algebra computations using more than one threads. This is indeed the most obvious way to speed up software such as ours. To test the performance of TeraPCA as a function of the number of threads, we focused on datasets S_1, S_2, S_4, S_6, S_7, and the 1000 Genomes dataset. The number of threads was set to 4, 8, and 12 and the speedups reported in Figure 3 are against the single-thread execution of TeraPCA. Generally speaking, we observed a 1.6x-2.8x speedup, which is somewhat sub-optimal. The reason underlying this non-optimality is that we used multithreading only for the linear algebraic operations. However, much of the wall-clock time is spent on I/O operations in order to load the dataset from secondary memory, a procedure that cannot be multithreaded. We emphasize that FlashPCA2 did not demonstrate comparable improvements when multi-threading was enabled. In particular, when applied to the dataset S_6, the wall-clock time of FlashPCA2 reduced only by two minutes, i.e., from 141 minutes to 139 minutes.

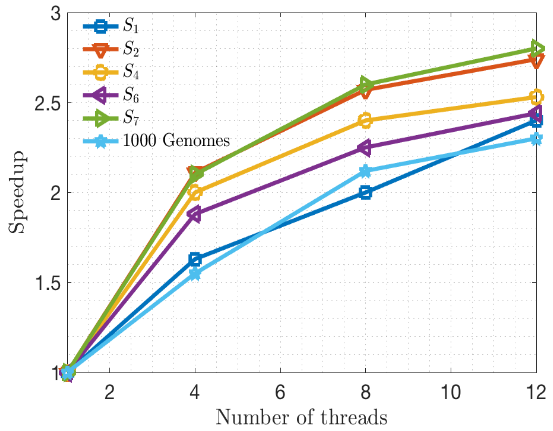

Fig. 3. Speedup of TeraPCA over single-threaded execution.

In all of the above experiments we set $s := 2k$ and $k := 10$. Finally, Supplementary Figure 6 reports the amount of time required to multiply the (normalized) covariance matrix by a set of s vectors using the DGEMM BLAS routine of MKL and a varying number of threads for different values of s and β for datasets S_6 and HGDP. It is worth noting that while an exhaustive analysis lies outside the goals of this paper, it is easy to verify that doubling the value of s does not double the amount of time required to perform the multiplication, while larger values of s also lead to higher speedups when multiple threads are used. Similarly, very small values of β are likely to penalize the performance of DGEMM due to non-optimal cache utilization.

4 Summary and future work

In this paper we presented TeraPCA, a C++ library to perform out-of-core PCA analysis of massive genomic datasets. It is based on Randomized Subspace Iteration, building upon principled and theoretically sound methods to approximate the top principal components of massive covariance matrices. TeraPCA returns highly accurate approximations to the top principal components, while taking advantage of modern computer architectures that support multi-threading and it has minimal dependencies to external libraries. TeraPCA can be applied both in-core and out-of-core and is able to successfully operate even on personal workstations with a system memory of just a few gigabytes. Numerical experiments performed on synthetic and real datasets demonstrate that TeraPCA performs similarly or better when compared to state-of-the-art software packages such as FlashPCA2, on a single thread and significantly better with multi-threading.

Future work will focus on implementing a distributed memory version of TeraPCA using the Message Passing Interface (MPI) standard. Another interesting research direction would be to combine TeraPCA with block Krylov subspace techniques.

5 Author's contributions

AB, VK, EK, and PD conceived and designed the work. AB, VK, and EK developed the TeraPCA C++ package. EK, ME, and AB developed the C++ package to generate the simulated data sets from the PSD model. PD and PP participated in and discussed analyses. AB and VK ran the experiments. AB, VK, EK, PD, and PP wrote and revised the manuscript.

6 Funding

This work has been partially supported by NSF [IIS-1661760, IIS-1661756, IIS-1715202] to PP and PD.

7 Acknowledgements

The authors are grateful to P. Gopalan and W. Hao for sharing their R script to generate the simulated data sets as well as for their valuable comments. The authors are also thankful to P. Anappindi for contributing to the code in the pilot phase.

References

Abraham, G. and Inouye, M. (2014). Fast principal component analysis of large-scale genome-wide data. *PLOS ONE*, **9**(4), 1–5.

Abraham, G., Qiu, Y., and Inouye, M. (2017). Flashpca2: principal component analysis of biobank-scale genotype datasets. *Bioinformatics*, **33**(17), 2776–2778.

Alexander, D. H. and Novembre, J. & Lange, K. (2009). Fast model-based estimation of ancestry in unrelated individuals. *Genome Res.*

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition.

Bose, A., Platt, D. E., Parida, L., Paschou, P., and Drineas, P. (2017). Dissecting population substructure in india via correlation optimization of genetics and geodemographics. *bioRxiv*.

Cann, H. M., de Toma, C., Cazes, L., Legrand, M.-F., Morel, V., Piouffre, L., Bodmer, J., Bodmer, W. F., Bonne-Tamir, B., Cambon-Thomsen, A., Chen, Z., Chu, J., Carcassi, C., Contu, L., Du, R., Excoffier, L., Ferrara, G. B., Friedlaender, J. S., Groot, H., Gurwitz, D., Jenkins, T., Herrera, R. J., Huang, X., Kidd, J., Kidd, K. K., Langaney, A., Lin, A. A., Mehdi, S. Q., Parham, P., Piazza, A., Pistillo, M. P., Qian, Y., Shu, Q., Xu, J., Zhu, S., Weber, J. L., Greely, H. T., Feldman, M. W., Thomas, G., Dausset, J., and Cavalli-Sforza, L. L. (2002). A human genome diversity cell line panel. *Science*, **296**(5566), 261–262.

Chisholm, B., Cavalli-Sforza, L. L., Menozzi, P., and Piazza, A. (1995). The History and Geography of Human Genes. *The Journal of Asian Studies*, **54**(2), 490.

Drineas, P. and Mahoney, M. W. (2016). RandNLA: Randomized Numerical Linear Algebra. *Communications of the ACM*, **59**(6), 80–90.

Drineas, P. and Mahoney, M. W. (2018). *Lectures on Randomized Numerical Linear Algebra, The Mathematics of Data, IAS/Park City Math. Ser.*, volume 25, pages 1–45. Amer. Math. Soc., Providence, RI.

Drineas, P., Ipsen, I. C. F., Kontopoulou, E., and Magdon-Ismail, M. (2018). Structural convergence results for low-rank approximations from block krylov spaces. *SIAM Journal of Matrix Analysis and Applications, to appear*.

Galinsky, K. J., Bhatia, G., Loh, P.-R., Georgiev, S., Mukherjee, S., Patterson, N. J., and Price, A. L. (2016). Fast principal-component analysis reveals convergent evolution of <em>adh1b</em> in europe and east asia. *The American Journal of Human Genetics*, **98**(3), 456–472.

Gopalan, P., Hao, W., Blei, D. M., and Storey, J. D. (2016). Scaling probabilistic models of genetic variation to millions of humans. *Nat Genet*, **48**(12), 1587–1590. 27819665[pmid].

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, **53**(2), 217–288.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, **24**(6), 417–441.

Hotelling, H. (1936). Relations Between Two Sets of Variates. *Biometrika*, **28**(3/4), 321–377.

Menozzi, P., Piazza, A., and Cavalli-Sforza, L. (1978). Synthetic maps of human gene frequencies in europeans. *Science*, **201**(4358), 786–792.

Musco, C. and Musco, C. (2015). Randomized block krylov methods for stronger and faster approximate singular value decomposition. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1396–1404. Curran Associates, Inc.

Novembre, J., Johnson, T., Bryc, K., Kutalik, Z., Boyko, A. R., Auton, A., Indap, A., King, K. S., Bergmann, S., Nelson, M. R., Stephens, M., and Bustamante, C. D. (2008). Genes mirror geography within europe. *Nature*, **456**, 98 EP –.

Parlett, B. (1998). *The Symmetric Eigenvalue Problem*. Society for Industrial and Applied Mathematics.

Paschou, P., Ziv, E., Burchard, E. G., Choudhry, S., Rodriguez-Cintron, W., Mahoney, M. W., and Drineas, P. (2007). Pca-correlated snps for structure identification in worldwide human populations. *PLOS Genetics*, **3**(9), 1–15.

Paschou, P., Drineas, P., Lewis, J., Nievergelt, C. M., Nickerson, D. A., Smith, J. D., Ridker, P. M., Chasman, D. I., Krauss, R. M., and Ziv, E. (2008). Tracing sub-structure in the european american population with pca-informative markers. *PLOS Genetics*, **4**(7), 1–13.

Paschou, P., Drineas, P., Yannaki, E., Razou, A., Kanaki, K., Tsetsos, F., Padmanabhuni, S. S., Michalodimitrakis, M., Renda, M. C., Pavlovic, S., Anagnostopoulos, A., Stamatoyannopoulos, J. a., Kidd, K. K., and Stamatoyannopoulos, G. (2014). Maritime route of colonization of Europe. *Proceedings of the National Academy of Sciences of the United States of America*, **111**(25), 9211–9216.

Patterson, N., Price, A. L., and Reich, D. (2006). Population structure and eigenanalysis. *PLOS Genetics*, **2**(12), 1–20.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**(11), 559–572.

Price, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A., and Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, **38**, 904 EP –. Article.

Price, A. L., Zaitlen, N. A., Reich, D., and Patterson, N. (2010). New approaches to population stratification in genome-wide association studies. *Nat Rev Genet*, **11**(7), 459–463. 20548291[pmid].

Pritchard, J. K., Stephens, M., and Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, **155**(2), 945–959. 10835412[pmid].

Rokhlin, V., Szlam, A., and Tygert, M. (2010). A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, **31**(3), 1100–1124.

Saad, Y. (2011). *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics.

Wang, C., A Szpiech, Z., Degnan, J., Jakobsson, M., J Pemberton, T., Hardy, J., B Singleton, A., and A Rosenberg, N. (2010). *Comparing Spatial Maps of Human Population-Genetic Variation Using Procrustes Analysis*, volume 9. Statistical applications in genetics and molecular biology.