Ancestry Inference in Low Dimensional Space Using Autoencoders

Sicheng Jia
Computer Science Department
University of California, Los Angeles
jsicheng@cs.ucla.edu

***Abstract*—Commonly in ancestry inference, PCA is applied over numerous SNPs to map an individual's genotype to a low dimensional space in order to aid in inference and visualization of populations. However, PCA is a linear transformation which may not capture all characteristics of the population data. Due to factors such as nonlinear SNP associations from linkage disequilibrium, more complex and nonlinear models for dimensionality reduction methods may be better suited to capture population structures. Here, we explore nonlinear dimensionality methods such as UMAP and Autoencoders, which are able to capture the nonlinear associations between the SNPs. We then improve upon this with Convolutional and Recurrent Autoencoders to capture additional sequential information from the SNP data in the low dimensional space. By incorporating this sequential information, we are able to produce a low dimensional mapping of data that better captures population structure information. This improved mapping is seen in the higher prediction accuracy in ancestry inference when using various classification models.**

I. Introduction

Dimensionality reduction is often performed on high dimensional data, mapping it to a low dimensional space while still preserving a large amount of the variance in the original data. This low dimensional representation of the data is useful for data exploration and visualization. It is also used to handle the *curse of dimensionality*, where a high number of features in the input data can lead to excessive computation time [1]. This is especially relevant in computational genomics, where an individual has millions of SNPs that can be used as features in a model.

In many past ancestry inference works, Principal Component Analysis (PCA) is performed to map high dimensional SNP data into a more manageable low dimensional space. An individual's genomic data can then be represented in this low dimensional space, making visualization and inference of population data much more feasible. An example of this can be seen in the works of Novembre et al., where the authors have shown that by performing PCA and rotating the axis in the low dimensional space, the mapping of the genome data of the individuals reflect the geographic origins of the individuals [2].

However, mappings produced by PCA are unable to capture many components of the genomic data. Linkage disequilibrium, or the nonrandom association between two or more alleles at different loci, can cause nearby SNPs to be associated with each other. More specifically, linkage disequilibrium causes nearby SNPs to be linearly dependent as well as nonlinearly dependent with each other [3]. While PCA is able to group linearly dependent components together, it is unable to catch these nonlinear associations due to PCA performing only a linear mapping of data. Additionally, since PCA only attempts to maximize the variance of the input data in the low dimensional space, it does not account for the sequentiality of the data. Both of these factors can cause important population information to be lost in the low dimensional space. In order to capture the nonlinear associations in the input SNPs, we explore nonlinear dimensionality reduction methods such as Uniform Manifold Approximation and Projection (UMAP) and Autoencoders. To capture the sequentiality of the input SNPs, we build on top of the Autoencoder model with Convolutional and Recurrent layers, creating Convolutional and Recurrent Autoencoders.

II. Models

In order to produce better low dimensional mappings of genomic data, we explore nonlinear dimensionality reduction methods and compare them against PCA as a benchmark.

A. PCA

Principal Component Analysis, or PCA, attempts to map the input data into a lower dimensional space such that the variance of the data is maximized in that space. This is done by computing the eigenvectors of the covariance matrix of the input SNPs through singular value decomposition. These computed eigenvectors are used as the principle components in the lower dimensional space, allowing for the mapping of high dimensional data into lower dimensions. Since the mapping of high dimensional variables into the lower dimensional space is done through matrix multiplication of the input variables and eigenvectors, PCA remains a linear model.

B. UMAP

Uniform Manifold Approximation and Projection, or UMAP, is a nonlinear dimensionality reduction method that uses manifold learning through topological structure modeling [4]. UMAP begins by building locally connected neighborhood graphs of the high dimensional data, which maps the topology of the input data. UMAP then searches in the low dimensional space in which the data in the manifold has the most similar topological structure as the data in the high dimensional space.

This learning is optimized through minimizing the following cross entropy function [4]:

$$\sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right) \quad (1)$$

Where $w_h(e)$ and $w_l(e)$ are the weights of edge e of the graph in the high dimensional low dimensional representation, respectively. This function balances out the edges of high and low weights in both the high dimensional and low dimensional case, pulling points that are close in the high dimensional representation close, while preserving the distance for far away points. For SNP data, UMAP produces low dimensional representations that result in local neighborhoods that group similar individuals together while still preserving global distances for distantly related individuals [5].

C. Autoencoders

Autoencoders are a family of neural network architectures that is composed of an encoder network and a decoder network. The encoder network attempts to project the input data x into a lower dimensional embedding z. This embedding z is then fed into the decoder network, which then attempts to reconstruct the original data x from the embedding, creating a reconstruction of the data x'. The goal of the autoencoder is to minimize the reconstruction error by minimizing a chosen loss function between x and x' [1]. To perform dimensionality reduction using autoencoders, the input data is fed into the encoder only. The encoder then maps the input data into low dimensional representation. Since the encoder-decoder network is trained to minimize reconstruction error, the encoder will learn to best represent the data in a low dimensional space for the decoder to later reconstruct. This allows the encoder to retain important information from the data in the lower dimensions.

An autoencoder that only uses a single fully connected layer and linear activation functions reduces down to PCA [6]. In order for our autoencoder to learn nonlinear mappings, we use multiple fully connected layers and use nonlinear activation functions between each layer. The use of multiple layers and nonlinear activation functions allow the autoencoder to produce more complex and nonlinear low dimensional mappings of the input data.

D. Convolutional Autoencoders

Standard autoencoders can be modified with more complex layers to learn better embeddings of the input data. One such modification is to place convolutional layers within the encoder to produce the embeddings, and to place deconvolutional layers within the decoder network. These convolutional layers help retain some sequential information from the SNPs by training a kernel that slides across the input. Since nearby input SNPs are taken into account by the kernel, the convolutional autoencoder is able to preserve some information from the ordering of the input SNPs [7].

E. Recurrent Autoencoders

As an alternative to convolutional layers, we can also add recurrent layers to an autoencoder. These recurrent layers contain feedback mechanisms where the previous output is fed back to the network with the current input. Each recurrent cell also has a hidden state that is used to learn the long term dependencies. However, standard recurrent networks suffer from the vanishing gradient problem and are often unable to capture long term dependencies. To solve this, we use two types of recurrent layers to create two variations of a recurrent autoencoder.

1) LSTM Autoencoder: Long Short Term Memory, or LSTM, is a type of recurrent layer that uses an input gate, output gate, forget gate and cell state to control the flow of information and hidden state updates. The input gate and forget gate control how much of the cell state is updated with the current information and the output gate controls how much information to propagate to the next input. The use of these gates allow some information to flow completely through, avoiding the vanishing gradient problem and allowing the cell to capture long term dependencies [8]. By capturing long term dependencies within the SNP data, these LSTM units can capture effects such as linkage disequilibrium over a longer range of input SNPs.

2) GRU Autoencoder: Gated Recurrent Units, or GRU, are a variation of the LSTM where an update gate controls how much information from the current input should be retained, and a reset gate controls how much information from the previous hidden state contributes to the current state [8]. GRUs have less parameters than LSTM units, which makes GRU networks more efficient to train. Similarly to the LSTM autoencoder, the GRU autoencoder is also able to capture linkage disequilibrium and sequential information from SNP data.

III. METHODS

A. Dataset

Genotype data and population annotations are obtained from the International Genome Sample Resource (1000 Genomes Project) [9]. 10,000 SNPs for each of the 2504 individuals were randomly sampled from chromosome 1. The 2504 individuals were split into a 80/20 train test split.

B. Model Setup

All dimensionality reduction methods are set to reduce the 10,000 input SNPs into two dimensions. For PCA and UMAP, the default settings from scikit-learn and umap-learn were used [10][4]. For the autoencoder methods, in order to keep the input size small, PCA is first run to reduce the input SNPs to 256 components. These components are then used as the input to each autoencoder. All autoencoders are implemented in PyTorch using the PyTorch layers and are trained for 1000 epochs [11]. The architecture of each autoencoder is listed in Table I.

TABLE I
TABLE OF AUTOENCODER ARCHITECTURES.

Autoencoder

	Layer	Activation	Dropout
Encoder	Linear(256, 64)	ReLU	0.1
	Linear(64, 32)	ReLU	0.1
	Linear(32, 2)	None	None
Decoder	Linear(2, 32)	ReLU	0.1
	Linear(32, 64)	ReLU	0.1
	Linear(64, 256)	None	None

Convolutional Autoencoder

	Layer	Activation	Dropout
Encoder	Conv1d(1, 4, 3)	ReLU	None
	Conv1d(4, 8, 3)	ReLU	None
	Linear(8*256, 256)	ReLU	0.1
	Linear(256, 2)	None	None
Decoder	Linear(2, 256)	ReLU	0.1
	Linear(256, 8*256)	ReLU	None
	ConvTranspose1d(8, 4, 3)	ReLU	None
	ConvTranspose1d(4, 1, 3)	None	None

LSTM Autoencoder

	Layer	Activation	Dropout
Encoder	Linear(256, 32)	ReLU	0.1
	LSTM(32, 8)	None	None
	Linear(8, 2)	None	None
Decoder	Linear(2, 8)	ReLU	0.1
	LSTM(8, 32)	None	None
	Linear(32, 256)	None	None

GRU Autoencoder

	Layer	Activation	Dropout
Encoder	Linear(256, 32)	ReLU	0.1
	GRU(32, 8)	None	None
	Linear(8, 2)	None	None
Decoder	Linear(2, 8)	ReLU	0.1
	GRU(8, 32)	None	None
	Linear(32, 256)	None	None

C. *Evaluation*

The dimensionality reduction methods are evaluated by training and testing the various classifiers on the population data in the 2-dimensional space. The classifiers include a logistic regression classifier, a random forest classifier, a SVM classifier, and a simple neural network classifier. Since evaluation of the dimensionality reduction methods is performed through comparing the accuracy of various classification models, the classifier parameters are kept as the default to keep the classifiers consistent. All classifiers are evaluated with an accuracy score.

IV. RESULTS

The classification accuracy of various classification models after performing the dimensionality methods are presented in Table II.

TABLE II
CLASSIFICATION ACCURACY AFTER VARIOUS DIMENSIONALITY REDUCTION METHODS.

	PCA	UMAP	AE	CAE	LSTMAE	GRUAE
Log. Reg.	0.888	0.936	0.934	0.858	0.946	**0.966**
Ran. Forest	0.882	0.902	0.954	**0.970**	**0.970**	0.964
SVM	0.908	0.934	0.960	**0.976**	0.972	0.972
MLP	0.916	0.934	0.956	**0.974**	0.970	**0.974**

Plots for the population data after running the various dimensionality reduction methods are presented in Fig 1 to Fig 6 for the training data. The plots of individuals for the test data are presented in the appendix.

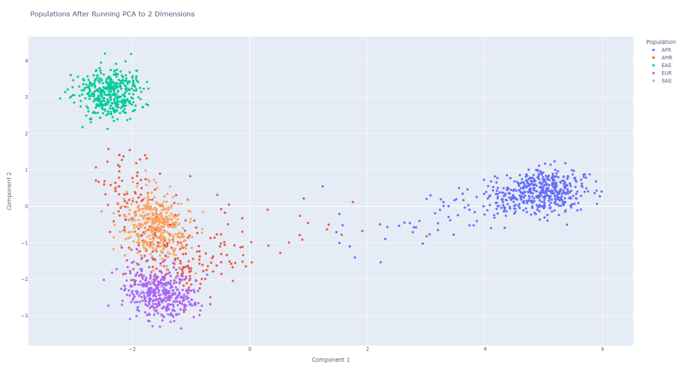

Fig. 1. Population data in 2 dimensions after running PCA.

From Figure 1, we see that PCA does well in separating the EAS and AFR populations, but fails to separate the admixed AMR population from the SAS and EUR populations.

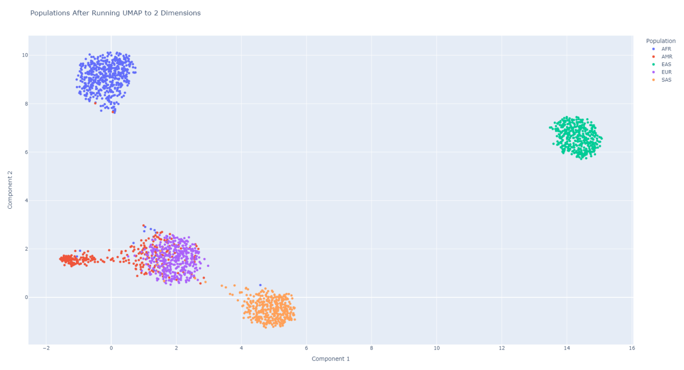

Fig. 2. Population data in 2 dimensions after running UMAP.

From Figure 2, we see that the nonlinearity of UMAP allows the AMR population to be nearly completely separated from the SAS population. However, much of the AMR population is still mixed with the EUR population.

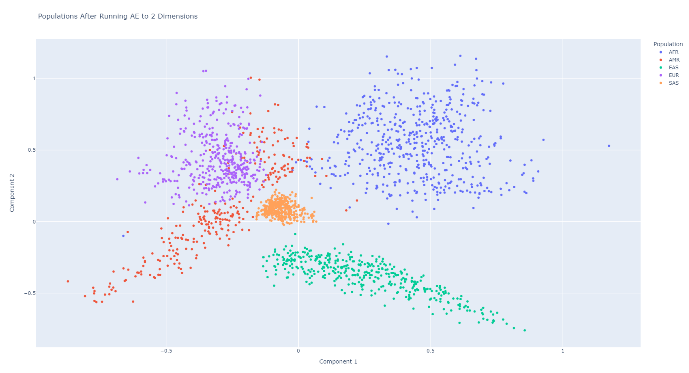

Fig. 3. Population data in 2 dimensions after running the Autoencoder.

From Figure 3, we see that the Autoencoder is able to better separate the AMR population from the EUR population compared to UMAP. However, the AMR population is split into two clusters in this separation.

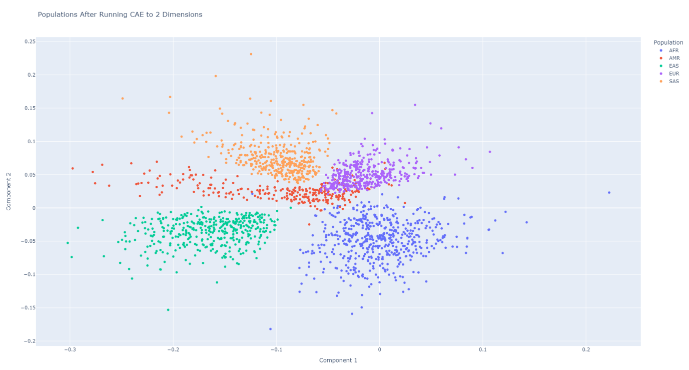

Fig. 4. Population data in 2 dimensions after running the Convolutional Autoencoder.

From Figure 4, we see that the Convolutional Autoencoder is able to mostly separate out the AMR population from the EUR population in addition to keeping the AMR population in one cluster.

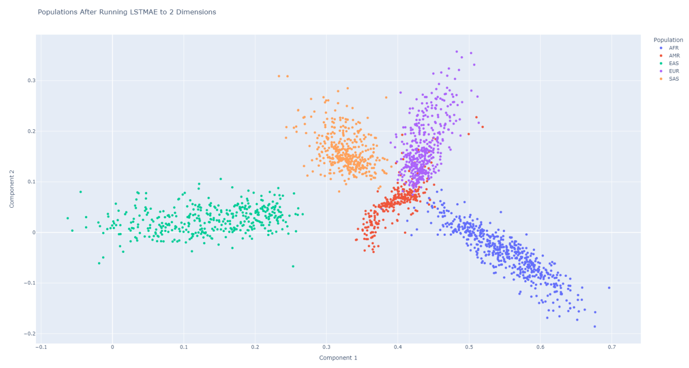

Fig. 5. Population data in 2 dimensions after running the LSTM Autoencoder.

From Figure 5, we see that similarly to the Convolutional Autoencoder, the LSTM Autoencoder is also able to mostly separate the AMR population from the EUR population and keep the AMR population in one cluster. We also notice that the AMR population is now more tightly packed within its own cluster.

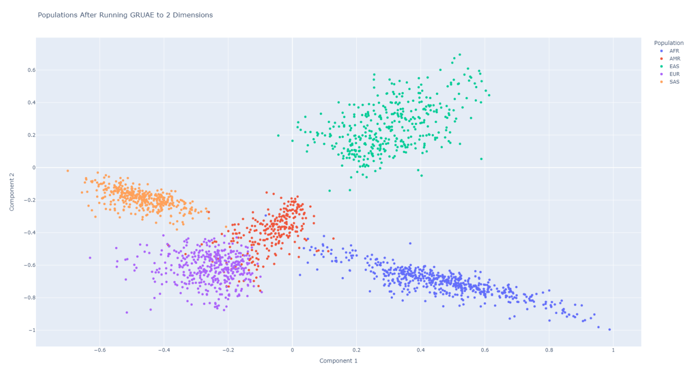

Fig. 6. Population data in 2 dimensions after running the GRU Autoencoder.

From Figure 6, we see that the GRU Autoencoder is able to separate the AMR population from the EUR population and keep the admixed population to a single cluster. We also notice that the AMR population forms a slightly more distinct cluster when compared to the other autoencoder models.

V. Discussion

From the classification accuracies, we see that the UMAP and the Autoencoder model significantly outperform the PCA model. This is due to the ability of UMAP and the Autoencoder to perform nonlinear low dimensional mappings with the topographical projections in UMAP and the ReLU activation function in the Autoencoder. This can be observed in the plots for each model where UMAP and the Autoencoder were both able to separate the AMR population from the SAS populations, while in the PCA, representation the populations remained mixed together.

We also see that the Convolutional Autoencoder, LSTM Autoencoder, and GRU Autoencoder outperformed the UMAP and the simple Autoencoder model. This improved performance is likely due to the preservation of sequential SNP data in the low dimensional space. The Convolutional Autoencoder achieves this with the kernel in the convolutional layer, where nearby SNPs are taken into account when extracting useful features. The LSTM and GRU Autoencoders preserve sequential information by using recurrent layers, where a hidden state and various gates filter how much information from previous input is factored into computing the current output. The effects of preserving sequential information can be seen in the population plots where the AMR population is not only separated from the SAS population, but is also increasingly separated from the EUR population. This separation of different populations allows the various classifiers perform better, which can be observed with the increase in classification accuracy.

The Convolutional Autoencoder, LSTM Autoencoder, and GRU Autoencoder performed similarly to each other, with an exception in the logistic regression model. The Convolutional Autoencoder performed relatively poorly on the logistic regression model while the LSTM and GRU Autoencoder have the highest score. This is likely due to the relative distance between each population cluster. In Figure 4, we see that the Convolutional Autoencoder maps the different populations groups closer while still having the AMR population spread out when compared to the LSTM and GRU Autoencoders in Figure 5 and 6. Thus, due to a more consistent classification accuracy in all models, the LSTM and GRU Autoencoder is suggested for dimensionality reduction. Furthermore, the GRU Autoencoder is suggested over the LSTM Autoencoder due to a faster training time and a minor overall improvement in accuracy.

VI. Future Work

The LSTM and GRU Autoencoders may be further improved by adding more layers and tuning the model parameters, however it should be noted that adding more recurrent layers may increase the training time significantly.

More advanced architectures can also be explored for even more information preservation. In the natural language processing domain, Transformers have seen success in preserving sequential text information [12]. These attention based models may be adapted for dimensionality reduction through learning latent SNP embeddings similarly to text embedding training for language tasks.

Lastly, since dimensionality reduction is a preprocessing step for many predictive models, these autoencoders may be applied to many downstream tasks beyond ancestry inference. These tasks may include disease prediction among many other phenotype inference tasks.

VII. Conclusion

While ancestry inference is typically performed by using PCA for dimensionality reduction, we show that using other dimensionality reduction methods can lead to better low dimensional representation of SNP data and result in better classification accuracies. By using nonlinear dimensionality reduction methods such as UMAP and a simple Autoencoder, we are able to capture nonlinear population structures such as linkage disequilibrium. We then extend upon the use of autoencoders to create more complex autoencoder architectures such as the Convolutional Autoencoder, LSTM Autoencoder and GRU Autoencoders. These advanced autoencoders are not only able to perform nonlinear mappings to a low dimensional space, but also able to capture sequential SNP data. This allows for more information to be encoded in the lower dimensional space, leading to higher classification accuracies during ancestry inference. Lastly, due to a low training time and consistent accuracies, the GRU Autoencoder is recommended for dimensionality reduction for SNP data.

References

[1] Q. Fournier and D. Aloise, *Empirical comparison between autoencoders and traditional dimensionality reduction methods*, Mar. 2021.

[2] J. Novembre, T. Johnson, K. Bryc, *et al.*, "Genes mirror geography within europe," *Nature*, vol. 456, p. 274, Dec. 2008. DOI: 10.1038/nature07566.

[3] R. D. Smith, "The nonlinear structure of linkage disequilibrium," *bioRxiv*, 2020. DOI: 10.1101/566208. eprint: https://www.biorxiv.org/content/early/2020/02/17/566208.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2020/02/17/566208.

[4] L. McInnes and J. Healy, "Umap: Uniform manifold approximation and projection for dimension reduction," *ArXiv*, vol. abs/1802.03426, 2018.

[5] A. Diaz-Papkovich, L. Anderson-Trocmé, C. Ben-Eghan, and S. Gravel, "Umap reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts," *PLoS Genetics*, vol. 15, 2019.

[6] E. Plaut, "From principal subspaces to principal components with linear autoencoders," *ArXiv*, vol. abs/1804.10253, 2018.

[7] K. Ausmees and C. Nettelblad, "A deep learning framework for characterization of genotype data," *bioRxiv*, 2020. DOI: 10.1101/2020.09.30.320994. eprint: https://www.biorxiv.org/content/early/2020/10/02/2020.09.30.320994.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2020/10/02/2020.09.30.320994.

[8] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *ArXiv*, vol. abs/1909.00590, 2019.

[9] S. Fairley, E. Lowy-Gallego, E. Perry, and P. Flicek, "The International Genome Sample Resource (IGSR) collection of open human genomic variation resources," *Nucleic Acids Research*, vol. 48, no. D1, pp. D941–D947, Oct. 2019, ISSN: 0305-1048. DOI: 10.1093/nar/gkz836. eprint: https://academic.oup.com/nar/article-pdf/48/D1/D941/31697918/gkz836.pdf. [Online]. Available: https://doi.org/10.1093/nar/gkz836.

[10] L. Buitinck, G. Louppe, M. Blondel, *et al.*, "API design for machine learning software: Experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[11] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[12] A. Vaswani, N. M. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *ArXiv*, vol. abs/1706.03762, 2017.

VIII. Appendix

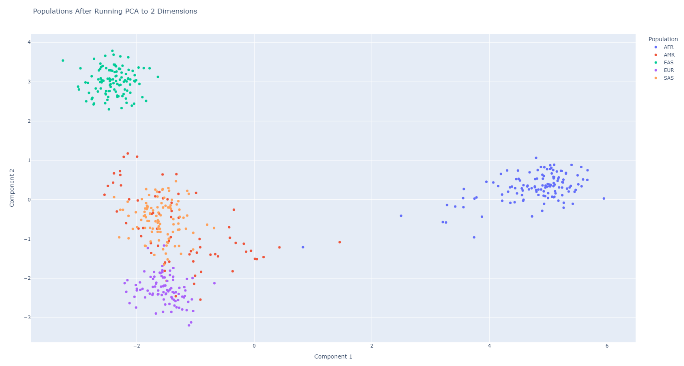

Fig. 7. Test population data in 2 dimensions after running PCA.

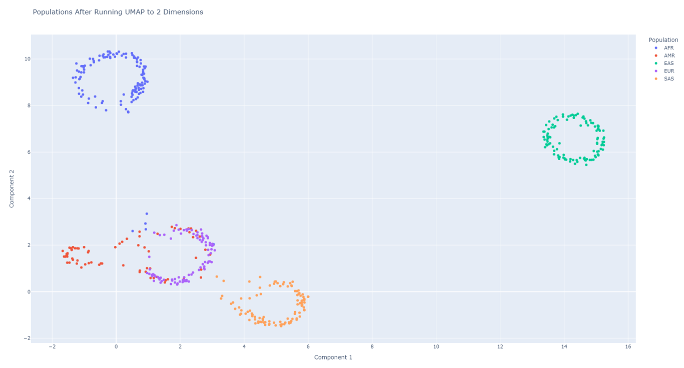

Fig. 8. Test population data in 2 dimensions after running UMAP.

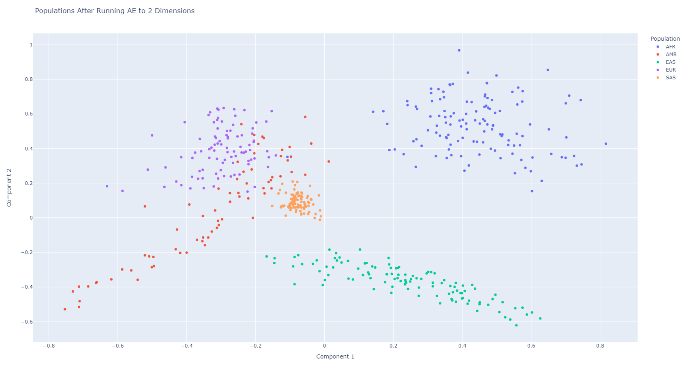

Fig. 9. Test population data in 2 dimensions after running the Autoencoder.

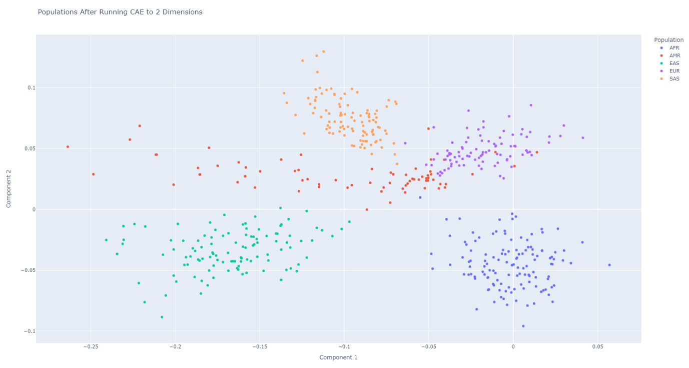

Fig. 10. Test population data in 2 dimensions after running the Convolutional Autoencoder.

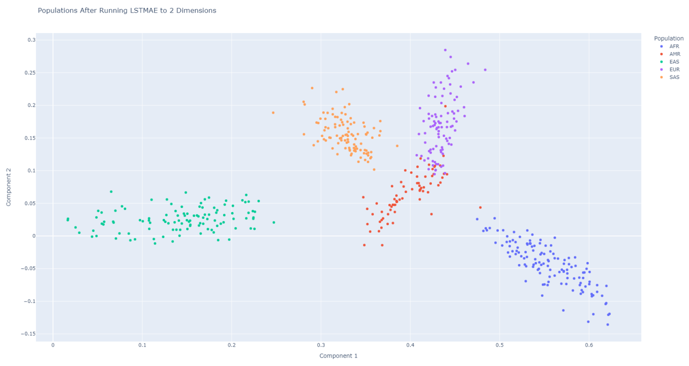

Fig. 11. Test population data in 2 dimensions after running the LSTM Autoencoder.

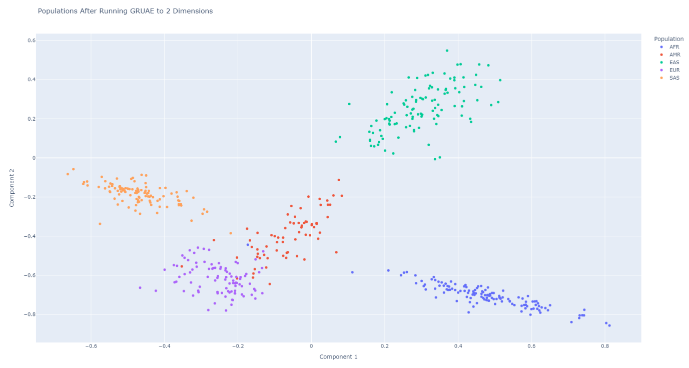

Fig. 12. Test population data in 2 dimensions after running the GRU Autoencoder.