Population Genetics

GPyS: A python implementation and web application of the Geographic Population Structure analysis algorithm

Felicia Schulz

Department of Biology, Box 118, 221 00, Lund University, Sweden

Project supervisor: Eran Elhaik

Abstract

Motivation: In this project, the python web application GPyS was created, which provides a user-friendly interface for the Geographic Population Structure Analysis (GPS) algorithm. The GPS algorithm was written from R into python and a web application was created with the python library Streamlit. This was done with the aim of facilitating the use of the GPS algorithm for all users, and for spreading accessibility so that more valuable research can be carried out on genetic origin localization.

Results: The application GPyS is user-friendly and easily understandable. It is robust and error-proof. As of now, it only works with a specific type of input data, so this is something to look into in further research.

Availability: The GPyS web application code and the data used are available at https://github.com/feliciaschulz/PopGen_Project.git

Contact: fe5502sc-s@student.lu.se

1 Introduction

One of the main focus areas of genetics research nowadays is biogeography and finding out where someone is from based on their DNA.

The Geographic Population Structure (GPS) algorithm created by Elhaik et al. (2014) does just that, by taking in DNA data and predicting the individual's origin population from it. In this project, the GPS algorithm was translated from its original source code in R to python. Additionally, a web application interface was created to facilitate usage of this algorithm and functionality for whoever may need it.

By creating a user interface, this GPS algorithm is made much more accessible for all users regardless of their programming skills and their general bioinformatics proficiency.

Because the GPS algorithm was hereby translated to python, this application is called GPyS. GPyS has the same functionality as the original GPS R code, but with its novel web application implementation, it elevates the algorithm to a new level.

2 Methods

2.1 Data origin

The original GPS algorithm, as well as the data used in this project, were taken from the GPS GitHub repository by Homologous in 2017 (https://github.com/homologus/GPS) that contains the files and resources described in Elhaik et al. (2014).

In the research and code development of Elhaik et al. (2014), the pool of reference populations was created by carrying out unsupervised ADMIXTURE on a wide range of different populations from all over the world and choosing those that proved to have high genetic diversity as well as that had lived in their respective geographic area for multiple centuries. A supervised ADMIXTURE was then used to analyze the admixture proportions of the chosen populations in regards to their presumptive ancestral populations, based on which two distance matrices were created: GEN, which is the distance matrix of the mean genetic admixture fractions of all reference populations, and GEO, the matrix of their geographic distances. The distance matrices GEN and GEO are required as input files in GPyS. The third input file for GPyS is the file containing the sample admixture data. In Elhaik et al.'s (2014) research, this was created by first converting the sample genotype file to plink files. Next, the files were merged and an admixture analysis was run on the merged file. Finally, the second output file after the admixture step will made up the third GPyS input file. This file contains sample name, origin of the sample and admixture fractions.

2.2 Algorithm and web application

Overall, the process of translating the code from R into Python consisted of careful studying of each component of GPS and testing the functions and output of each step in order to understand exactly what is

computed. Of course, the programming language that was used was Python (v. 3.10.8), in combination with several different libraries.

For the actual GPS algorithm, numpy (v. 1.24.2) and pandas (v. 1.5.3) were used to handle the data frames in a similar way as R can. Scipy (v. 1.10.1) was used to compute distance matrices like the function dist() in R does, and to calculate the slope of the linear regression of the distance matrices, which in R is done by simply using the lm() function. In addition to data handling, numpy was also used for its function allclose(), which calculates the closeness of two arrays based on a similarity threshold and returns True if the overall distance is below that threshold. This functionality was required to make up for the function all.equal() in R by using the same distance threshold which is the default for all.equal().

For the web application, the library Streamlit (v. 1.19.0) was used. Streamlit is an open-source python library which allows for seamless integration of python algorithms into a web application interface. It also supports many other libraries within it, such as plotly (v. 2.18.1), which is the library that the data was plotted with.

3 Features

3.1 Summary, usage and input

The GPyS web application takes GEN and GEO distance matrices and the sample admixture proportion data as input and creates the GPS output file, saves it to the current working directory and displays it to the screen by means of an interactive table as well as an interactive map.

GPyS is run by typing "streamlit run GPyS.py" in the command line. The web app opens automatically in the browser. An image of the GPyS application can be seen in Figure 1. The application can be re-run by simply typing "r", or by re-loading the site. In the application, the user is asked to upload the three data files. This is done by simply clicking the "Browse files" button, which opens the user's directories so that they can manually select the appropriate file to upload. The files do not need to be in the same directory as GPyS.py. Additionally, the user is asked to specify *N_best*. This is a variable that is necessary for the GPS algorithm. It determines the number of closest Euclidean distances out of which the geographic radius for each sample is calculated.

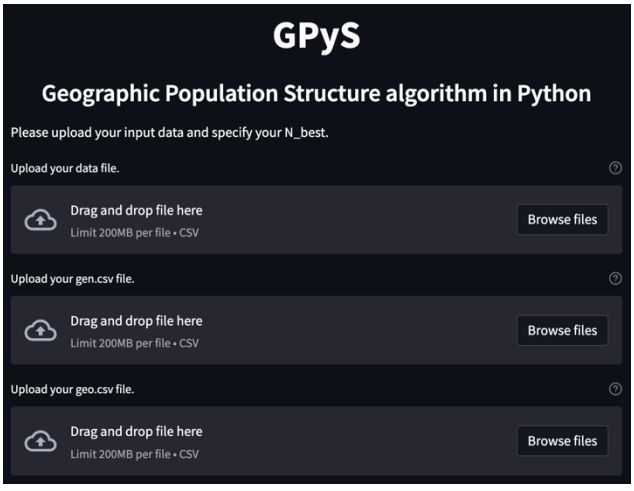

Figure 1: GPyS user interface.

3.2 Algorithm and output

Once the user has fulfilled all four input requirements, a loading circle appears with the label "Computing the GPS Analysis…", which loads until the analysis is done. Due to the fact that the GPS algorithm is very complex, the analysis can take a while to run; hence, the loading circle was implemented in order to make the waiting process more transparent and clearer for the user. This way, any confusion as to whether the program is running or not is completely avoided.

Once GPS is finished, the text "Done! Your results file has been saved to your current working directory" is displayed in a green box, clearly indicating the success of the procedure. The output file is indeed saved to the user's current working directory. It is important to note here that this may not be the directory that stores the data, but rather the directory in which GPyS.py is located. Additionally, the output data is displayed in an interactive table. The data contains the sample's population, sample ID, its predicted ancestral population as well as latitude and longitude for each sample. The screen shows only a proportion of the data and the user can scroll through and select cells, which highlights them. This decreased view was chosen because of the fact that the data frames the user might input, and hence get as output, could be very large, which if fully displayed would make the application less clear and cause problems. The table can also be enlarged by clicking a button on the top right so that it takes up the full screen.

Furthermore, the percentage of individuals whose origin was "correctly predicted", meaning that the ancestral population predicted by the GPS algorithm was the same as their actual origin, is displayed on screen. The table as well as the accuracy score can be seen in Figure 2.

	Population	Sample_no	Sample_id	Prediction	lat	lon
0	Abkhazians	1	GRC12076288	Abkhazians_0	39.5307	46.885
1	Abkhazians	2	GRC12076300	Abkhazians_0	41.3675	47.0145
2	Abkhazians	3	GRC12076312	Ingush_3	42.9137	39.0599
3	Abkhazians	4	GRC12076324	Abkhazians_0	42.9251	45.9893
4	Altaians	1	GRC12076232	Altaians_4	46.099	65.9615
5	Altaians	2	GRC12076245	Tatars_2	54.2356	55.1234
6	Altaians	3	GRC12076244	Altaians_3	50.3202	87.4742
7	Altaians	4	GRC12076256	Altaians_3	50.594	86.2797
8	Altaians	5	GRC12076268	Altaians_3	50.566	86.3992
9	Altaians	6	GRC12076280	Altaians_3	50.5969	86.2671

Percentage of individuals predicted to come from their region of origin: 0.8024475524475524 %

Figure 2: GPyS results table and accuracy score

Finally, the application ends with an interactive map showing each sample data point on a world map. The location information was of course taken from the latitude and longitude data calculated by GPS. The user is able to scroll in and out of the map and move it by holding down the mouse. When hovering over a data point, its sample ID is displayed in a box as well as latitude, longitude, their population of origin and their origin prediction. There is also a bar on top of the map which allows the user to save the map to their current working directory, scroll in and out, reset map position or show regions of the map by drawing a circle or a square around them. Just like the table, the map can also be enlarged and viewed full-screen. The map can be seen in Figure 3.

4 Discussion

The web application GPyS infers geographic origin location from genetic admixture proportions. It presents a simple way of not only running the GPS algorithm in python but also visualizing the data immediately and clearly in an interactive and concise way.

There are several advantages to GPyS, such as aspects of the user interface which facilitate use and smoothen the process of running the GPS

Figure 3: Interactive map

analysis. For example, the buttons that allow users to upload files by browsing through their directories make it easy to find the correct files, especially for those who are not very familiar with the command line. Rather than going through code and finding the appropriate variable to change in the right function, the user can just enter their preferred *N_best* in a text box. Furthermore, the loading circle creates a more pleasant user experience as it makes the process more transparent. The green message which pops up when the algorithm has run immediately conveys the feeling of success, which also improves user experience. The interactive table and map allow the user to make sense of their results in a more comprehensive way, really linking the data back to the biogeography and the real-world findings behind it. The application also shows unambiguous error messages, for example if the input is incorrect. Overall, GPyS presents a clear and fool-proof workflow.

One weakness of the current version of GPyS is that it only works for a very specific input data format, specifically the format used by Elhaik et al. (2014). With the aim of making the application as robust as possible, there is no room for deviation in the data structure, and data files which will not work with the algorithm are rejected by GPyS. Additionally, the loading circle does not give any indication as to how long the user will have to wait. It would be a better user experience to for example have a loading bar, however, this was not possible to implement as the program does not know how long the algorithm will take to compute the output.

The highly complex GPS algorithm achieves high accuracy in the difficult process of predicting individuals' origin (Elhaik et al., 2014). This has valuable applications in biogeography because the problem of locating DNA has been a focus of genetics research for many years.

With this novel, easy-to-use interface, the GPS algorithm will be accessible to everyone, no matter how proficient users are in coding.

Acknowledgement

We are grateful to all reviewers for their valuable comments on the manuscript and to Lund University for their financial support.

References

Elhaik, E., Tatarinova, T., Chebotarev, D., Piras, I. S., Maria Calò, C., De Montis, A., ... & Wells, R. S. (2014). Geographic population structure analysis of worldwide human populations infers their biogeographical origins. *Nature communications*, ***5***(1), 3513.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Homologous, GPS (2017), GitHub repository, https://github.com/homologus/GPS

Plotly Technologies Inc. Collaborative data science. Montréal, QC, 2015. https://plot.ly.

The Pandas Development Team, (2020). pandas-dev/pandas: Pandas (1.5.3). doi:10.5281/zenodo.3509134

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi:10.1038/s41592-019-0686-2