PyPop: A mature open-source software pipeline for population genomics

Alexander K. Lancaster [1,2,*], Richard M. Single[3,*], Steven J. Mack[4,*], Vanessa Sochat[5], Michael P. Mariani[3,6], Gordon D. Webster[1,2]

[1]Amber Biology LLC, Cambridge MA, United States

[2]Ronin Institute, Montclair, NJ, United States

[3]University of Vermont, Burlington, VT, United States

[4]University of California, San Francisco, Oakland, CA, United States

[5]Lawrence Livermore National Laboratory, Livermore, CA, United States

[6]Mariani Systems LLC, Hanover, NH, United States

*** Correspondence:**

Alexander K. Lancaster alancaster@amberbiology.com
Richard M. Single richard.single@uvm.edu
Steven J. Mack steven.mack@ucsf.edu

Keywords: HLA, MHC, population genomics, software, bioinformatics

Abstract

Python for Population Genomics (PyPop) is a software package that processes genotype and allele data and performs large-scale population genetic analyses on highly polymorphic multi-locus genotype data. In particular, PyPop tests data conformity to Hardy-Weinberg equilibrium expectations, performs Ewens-Watterson tests for selection, estimates haplotype frequencies, measures linkage disequilibrium, and tests significance. Standardized means of performing these tests is key for contemporary studies of evolutionary biology and population genetics, and these tests are central to genetic studies of disease association as well. Here, we present PyPop 1.0.0, a new major release of the package, which implements new features using the more robust infrastructure of GitHub, and is distributed via the industry-standard Python Package Index. New features include implementation of the asymmetric linkage disequilibrium measures and, of particular interest to the immunogenetics research communities, support for modern nomenclature, including colon-delimited allele names, and improvements to meta-analysis features for aggregating outputs for multiple populations.

Code available at: https://zenodo.org/records/10080668 and https://github.com/alexlancaster/pypop

1 Introduction

Since its principles were established a century ago (1–5), population genetics has been a computational science. The advent of electronic computing, and its widespread adoption for academic research in the 1980s and 1990s, fostered the development of computational genetics software (e.g., 6,7) that could perform multiple analyses and return results in standardized, human and machine-readable formats. PyPop (Python for Population Genomics) was initially developed between 2002 and 2007 (8,9) as a Python 2-based framework that performed multiple population genetic analyses on highly-polymorphic, multilocus genotype data, and generated both standardized, "publication ready" text-formatted outputs and machine-readable XML outputs, allowing for further downstream analyses and meta-analyses.

A standard PyPop analysis is initiated by running the "`pypop`" command-line program that is supplied with one or more plainText input "population" or "dataset" files (with the suffix ".pop"), along with a plainText input configuration file (with the suffix ".ini"). The input configuration file defines both the expected input format, as well as the specific analyses that will be run, including tests of Hardy-Weinberg equilibrium expectations, Ewens-Watterson tests of selection,and estimation of haplotype frequencies and linkage disequilibrium (a full list of the configuration options is available in the *PyPop User Guide* (10)). Each input file results in a corresponding set of output files: a machine-readable XML file, and a human readable plain-text file. These primary analyses can be aggregated to generate cross-dataset meta analyses using "`popmeta`", another tool in the PyPop suite . Here, we describe PyPop version 1.0.0, which is built using Python 3 and includes new features and improvements as well as a new development platform.

We first document the ongoing use of PyPop in the immunogenetics and other research communities in the years since the last release of PyPop (version 0.7.0). Next we describe new features and analytical methods, including measure of asymmetric linkage disequilibrium (ALD), and updates to support the current nomenclatures for major histocompatibility complex (MHC) and human leukocyte antigen (HLA) genes. We also note the streamlining and improvement of existing features such as the custom grouping of alleles and output of tab-separated value (TSV) files. We close by describing features in development, as well the porting of the project to GitHub to support future Python versions and new machine architectures, providing a stable home for PyPop to evolve as a community resource.

2 Methods and Results

2.1 PyPop in the human immunogenetics community and beyond

Since the first public release of the software in 2003 and the subsequent publication of descriptions in 2003 (8) and 2007 (9), PyPop has been in regular and continuous use within the HLA and the larger genomics communities, as shown in an analysis of Google Scholar citations (Figure 1). This analysis estimates that there have been 433 unique citations of PyPop since its inception (134 for the 2003 paper alone, 220 for the 2007 paper, and 79 for both). Of those unique citations, 367 are from 2007 or later. PyPop has been applied extensively within the immunogenetics community since its first release, as expected given its origins as part of the 13th International Histocompatibility Workshop (IHWS) in 2002 (11). A notable early meta-analysis of the action of natural selection on HLA polymorphism across 497 populations (12), relied heavily on PyPop 0.7.0 analyses and has 360 citations in Google Scholar at the time of writing.

Many of these citations are from researchers studying human immune system genes. However, PyPop has been used in many studies, far from its home research community. These include studies that are both taxonomically distinct (genetic heterogeneity of urban foxes (13)) and genetically distinct (population genetics of cytochrome enzyme proteins (14)) from human immunogenetics. These two examples illustrate the wide utility of PyPop as a computational population genomics resource.

2.2 New features and improvements

2.2.1 Asymmetric linkage disequilibrium measures

The conditional asymmetric linkage disequilibrium (ALD) measures, first described by Thomson and Single (15), are the major new analytic feature of PyPop 1.0.0. Previous PyPop versions computed two measures of overall linkage disequilibrium: *D'* (16), which uses the product of pairwise allele frequencies to weight the individual haplotype-level coefficients of LD, and W_n (17), which is a multi-allelic extension of the "*r*" correlation measure commonly used for LD with bi-allelic SNPs. ALD, further extends the W_n measure, accounting for asymmetries that arise from different numbers of alleles at different loci. The two measures, W_{12} and W_{21}, assess LD conditional on the second and first locus, respectively, and are both equal to the usual *r* statistic for SNPs (Table 1).

ALD is particularly useful when investigating LD in highly polymorphic gene-systems, where each locus displays large and very different numbers of alleles in a population. These ALD measures, computed using PyPop, have been used in anthropological studies dissecting LD in human populations (18,19); studies of permissible mismatches in HLA donor registries (20); and studies of HLA haplotypes and amino acid motifs that predispose for disease (21). Additional publications, using different implementations of the ALD, include studies of the impact of anti-malarial drugs on parasite populations among individuals with complex infection status (22,23). ALD measures allow one to condition on known disease genes in association studies when searching for additional genetic effects in a region. Similarly, by conditioning on putative targets of selection ALD measures can help characterize other potentially selected variants.

2.2.2 Support for modern HLA/MHC nomenclature

Since the major release of PyPop 0.7.0 in 2008, the allele-name nomenclatures for MHC and HLA genes have changed significantly. In 2010 (24) the format of HLA and MHC allele names was changed to include colon-delimited fields, where previous formats had relied on 'digit-based' fields. An allele denoted as `0101` before 2010 is now denoted as `01:01`. This nomenclature change also means that much longer HLA allele names (eg., A*02:01:01:134Q or DPB1*1372:01:01:02) are now valid, and PyPop can continue to process such data. In addition, the ~ operator, defined in the text-based Genotype List (GL) String grammar for describing HLA and Killer-cell Immunoglobulin-like receptor (KIR) genotyping results (25,26), has been the standard for delimiting alleles in multi-locus haplotypes with the immunogenetics community. In PyPop 1.0.0, a two locus haplotype of alleles at two loci, `A` and `B` respectively, is represented as `A~B`, where this haplotype had been represented as `A:B` in earlier PyPop releases.

Although previously there was nothing actively preventing a user of PyPop from using the 2010 HLA/MHC nomenclature for PyPop input data, PyPop 0.7.0's separation of haplotype elements with colons, meant that a ":" *within* an allele name could lead to ambiguous output. We introduced changes in version 1.0.0 to seamlessly handle the 2010 nomenclature, and now PyPop output includes the GL String '~' separator by default, facilitating use of the output in publications or further downstream analyses (Table 2). We have updated all documentation, examples and unit tests to reflect these changes.

2.2.3 Cross-platform support for custom grouping ("binning") filters

PyPop's capacity for "custom binning", which combines allele-names into specific categories for analysis, is now available on all platforms. This capacity extends to commonly used allele groupings (e.g., G- and P-groups (24), supertype groups (27), HLA T-cell epitope (TCE) groups (28,29), and National Marrow Donor Program [NMDP] allele codes (30,31)) that group distinct variants by common aspects. For example, as of January 2024, the A*01:01:01G G-group designation represents 240 HLA-A alleles that share identical exon 2 and exon 3 nucleotide sequences. Supertypes are groups of alleles with similar peptide-binding features; for example DPB1 alleles with identical peptide sequences for amino-acid positions 11, 69 and 84 are sorted into eight supertypes groups (27).

TCE groups identify sets of DPB1 alleles with shared amino acid motifs that result in permissive mismatches in the context of hematopoietic stem cell transplantation (29). NMDP allele codes identify groups of alleles that cannot be distinguished by genotyping methods that do not sequence the entire HLA gene. For example, the DRB1*11AD allele code is used to represent a genotyping result that could be either DRB1*1101 or DRB1*1104 (31).

PyPop custom binning is not restricted to these specific, community-defined examples; variant names can be combined into any user-defined category for PyPop analysis. An example custom binning filter for converting alleles to a G-group designation is presented in Figure 2. Additional examples are provided in Supplementary File 1.

2.2.4 Improved support for downstream analyses: enhancements to TSV output

PyPop analyses are always output as machine-readable XML files, with one XML file per population or dataset. Previous versions of PyPop included a feature to aggregate these individual dataset or population-level XML files into a set of files in tab-separated value (TSV) format, suitable for input into spreadsheets or other downstream software (Table 3). However, this feature was originally tuned to the needs of the 13th IHWS (11), and required adaptation for use outside this context. In PyPop 1.0.0, we have overhauled and re-tooled the output mechanism for general use. The changes include:

1. Previously the list of output TSV files was hardcoded, and this set of files was generated regardless of whether the analysis created any relevant data. For example, a `3-locus-haplo.tsv` file was generated even if estimation of 3 locus haplotypes was not requested by the user - resulting in a file with headers, but no data. The output files are now dynamically generated based on the analyses that were requested by the user (ultimately based on aggregating the contents of the separate XML outputs generated by each input `.pop` dataset). In addition, we have also enabled generation of TSV output for haplotype estimation involving five or more loci, e.g. `5-locus-haplo.tsv`, `6-locus-haplo.tsv`, etc. (see the last two rows of Table 3).

2. Output files now use the standard "`.tsv`" suffix (rather than "`.dat`") so they are more easily identified as tab separated value files that are parsable by other software. We have also renamed the command-line options accordingly (e.g. `--generate-dat` to `--enable-tsv`).

3. Previous versions included fixed metadata columns that were only relevant for the analyses performed for the 13th IHWS. These additional columns are now disabled by default (we have added a new "`--enable-ihwg`" option which will re-enable them).

4. We have added new options to enable TSV files to be saved in a separate directory (`--outputdir`) and include a prefix (`--prefix-tsv`).

These changes should increase the utility of PyPop for meta-analyses in a wider range of research use-cases, particularly for studies that need to aggregate analyses where haplotypes were estimated at more than four loci.

2.3 Development updates

When PyPop development started in late 2001, Python was at version 2. Soon after the last release of PyPop (0.7.0) in 2008, Python 3 was released. Python 3 unfortunately introduced breaking changes (breaking the existing PyPop code). With the end-of-life of Python 2 in 2020, migration from PyPop to Python 3 became an imperative. In addition to the new scientific features described above, and the desired transition to Python 3, other major goals of the PyPop 1.0.0 release were (a) to improve ease of installation and the overall experience for end-users, (b) to make it easier to contribute to PyPop, and (c) reduce "technical debt" (32) and thus improve overall project longevity. In this section, we discuss these changes to the development process, the Python 3 migration, improvements in packaging, deployment, provenance, and documentation to further these end-goals.

Development moved to the GitHub platform
In 2013 we migrated the source code version control system of PyPop from an internal Concurrent Versions System (CVS) repository to Git, and subsequently imported it as a public project on the GitHub platform. GitHub supports advanced features for developers including issue and milestone tracking, discussions, collaborative code review (pull requests), security scanning, and automation of testing via continuous integration (CI). With this change, the development process became more open to the community. Updates that added support for codon-delimited alleles and increased capacity for multi-locus analyses were made as part of the 17th International HLA & Immunogenetics Workshop, which was held in 2017 (33) and made available via GitHub, although no formal release was made at this time.

Migration to Python 3
Migration commenced in 2017, by an author of this paper (34) - outside the original development team - via a "pull-request", illustrating the benefits of moving to the GitHub platform. Initially the process was largely manual, including fixing of print statements, addition of modules, and rearranging of module imports. We included Singularity (35), an upcoming container technology for high performance computing, and a pull request to update from the deprecated "Numeric" to the "numpy" library was merged later in 2017 (36). In early 2023, we merged a modified version of the pull request, including additional changes, back into the main branch, which finalized the conversion to Python 3.

New test suite and continuous integration
During the port, we created a test suite that included both unit tests, and end-to-end "pipeline" tests, emulating end-user runs. As a result of this process, we refactored code, and removed obsolete or out-dated code, helping to reduce technical debt. Apart from its direct utility in detecting regressions introduced during development, this test suite has resulted in a wider set of configuration (".ini") and data (".pop") files that provide examples for end-users of PyPop to emulate. We also leveraged GitHub's CI feature, known as GitHub Actions, so that these tests are automatically run upon a commit to the repository.

Generating source distributions and binary wheels for Windows, MacOS X and Linux
The `cibuildwheel` system (37) generates "wheels" (architecture-specific installable versions of a Python package containing pre-compiled extensions), installs each wheel in a virtual environment, and then runs unit tests within the virtual environment with that installed wheel. Key to this process

is that `cibuildwheel` automates the process of compiling and testing wheels across multiple operating systems and Python versions, ensuring that they will work on each of those end-user systems. We deployed `cibuildwheel` as part of our GitHub Action workflow, resulting in over 40 different tested wheels on a wider range of architectures and Python versions (Supplementary Table 1) - compared with only two binary packages available previously (one for Linux, and one for Windows). These wheels include, for the first time, an official pre-compiled MacOS X version of PyPop, on both Intel (x86) and Apple Silicon (arm64) architectures. In addition to the automated CI testing, we did manual testing on several Windows, Linux and Android platforms (Supplementary Table 2).

Deploying releases via the Python Package Index (PyPI)

When a release is made via GitHub's "tag-and-release" interface, our workflow triggers a build of all binary wheels and source distribution via GitHub's CI system, as described above, but includes an additional step in the workflow of uploading a versioned release to the PyPI repository. This vastly simplifies installation for end users who can install PyPop directly from PyPI via a single "`pip install pypop-genomics`" command.

Provenance via Zenodo DOI

We configured the workflow so that, upon a production release via GitHub, it will deposit the source and metadata about the release to the Zenodo repository (38). This generates version-specific archives of the source code, together with a unique Digital Object Identifier [DOI]. Users can then cite the specific version used for their analyses as a DOI in their paper to enable more effective reproducibility (39). For example, the DOI for the 1.0.0 release being described in this paper is 10.5281/zenodo.1008066 (40).

Maintainable documentation

The previous version of the *PyPop User Guide* (10) was written using DocBook XML (41), which, while powerful, has a steep learning curve. For this new release, we converted all documentation to reStructuredText (42) which, as a simple plaintext-like language, is more intuitive for contributors. We created another GitHub Action workflow that runs the sphinx documentation generator (43) to generate both HTML and PDF versions of the *User Guide* and the website from the reStructuredText documents. This GitHub workflow ensures that all changes are automatically deployed to the pypop.org website with each commit to the repository. In addition, some of the documentation (e.g. command-line options) is either generated directly from the code, or pulled in from configuration and data files in the unit tests, further ensuring that documentation is always kept in sync with the current codebase.

3 Discussion

PyPop development continues beyond this 1.0.0 release. A set of features in development related to the estimation of haplotype frequencies and LD include a reworking of the existing implementation of the Expectation-Maximization algorithm; computing LD between loci when allelic phase is known; and moving less computationally-intensive aspects of code currently implemented in C extensions into Python. This will allow for an increase in the number of loci for which haplotypes can be estimated, relative to the existing implementation, because the new implementation doesn't require retention of all possible haplotype combinations. A preliminary, but incomplete implementation is already contained within PyPop 1.0.0 for alpha testing, but should not be used for production analyses.

Since the last release 16 years ago, PyPop has been in active and continuous use across a range of research communities. Despite a relative stasis in development during that period, PyPop has illustrated its durability as a framework for producing standardized reports for population genomics analyses. With the updated development platform, unit testing, packaging and deployment system in place, we have set a foundation to allow for more frequent, and well-tested releases, in addition to improving maintainability and encouraging contributions.

4 Software information

- **Project links**: http://pypop.org (home page), https://github.com/alexlancaster/pypop/ (development page)
- **Operating systems**: Linux, MacOS X, Android, Windows
- **Programming languages**: Python and C
- **License**: GNU GPLv2: https://www.gnu.org/licenses/gpl
- **Any restrictions for non-academic use**? None
- **Zenodo record**: https://zenodo.org/records/10080668

5 Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

6 Author Contributions

AKL, RMS and SJM conceived and designed the methodologies of PyPop. AKL and RMS developed new modules for the current version. AKL migrated the platform to GitHub, set up the continuous integration system and maintained the releases. AKL and VS carried out the Python 3 migration. AKL and GDW implemented the test suite. AKL, RMS, SJM, MPM and GDW contributed to the documentation. RMS, MPM and GDW performed software testing. AKL, SJM and VS wrote the first manuscript version. RMS, MPM and GDW contributed to manuscript review and editing. All authors read and approved the final manuscript. No artificial intelligence systems were applied in the writing of the paper or for the work described.

7 Funding

The work described here was supported in part by National Institutes of Health (NIH) National Institute of Allergy and Infectious Disease (NIAID) grant R01AI128775 (SJM), NIH National Institute of General Medical Sciences (NIGMS) grant R01GM109030 (SJM), and NIH Contract HHSN272201200028C (RMS). The content is solely the responsibility of the authors and does not necessarily reflect the official views of the NIAID, NIGMS, NIH, or the United States Government.

8 Acknowledgments

We would like to acknowledge Owen D. Solberg, Mark P. Nelson, Diogo Meyer, Yingssu Tsai, Karl Kornel and Jurriaan H. Spaaks, for contributions to the PyPop code base, software testing and documentation. We also thank original project lead, Glenys Thomson. We thank the Histocompatibility, Immunogenetics and Disease Profiling Laboratory in the Department of Pathology at Stanford University Medical School for support in upgrading PyPop to enable analysis

of colon-delimited allele names and increase its multi-locus analysis capacity as part of the 17th International HLA & Immunogenetics Workshop.

Figures and Tables

Figure 1: Number of unique citations over time to the two previous PyPop publications: *Pacific Symposium in Biocomputing* (*PSB*) (8) and *Tissue Antigens* (9). Some publications cited both PyPop papers. Google Scholar was used for the counts.

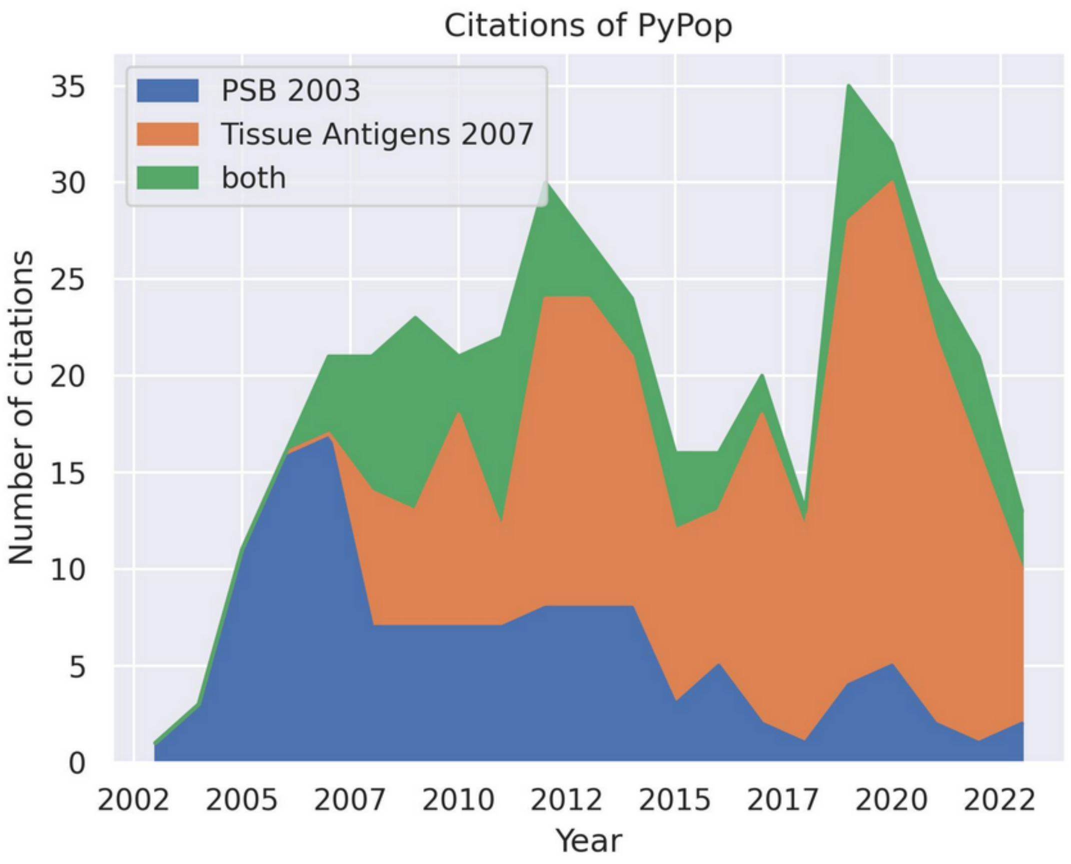
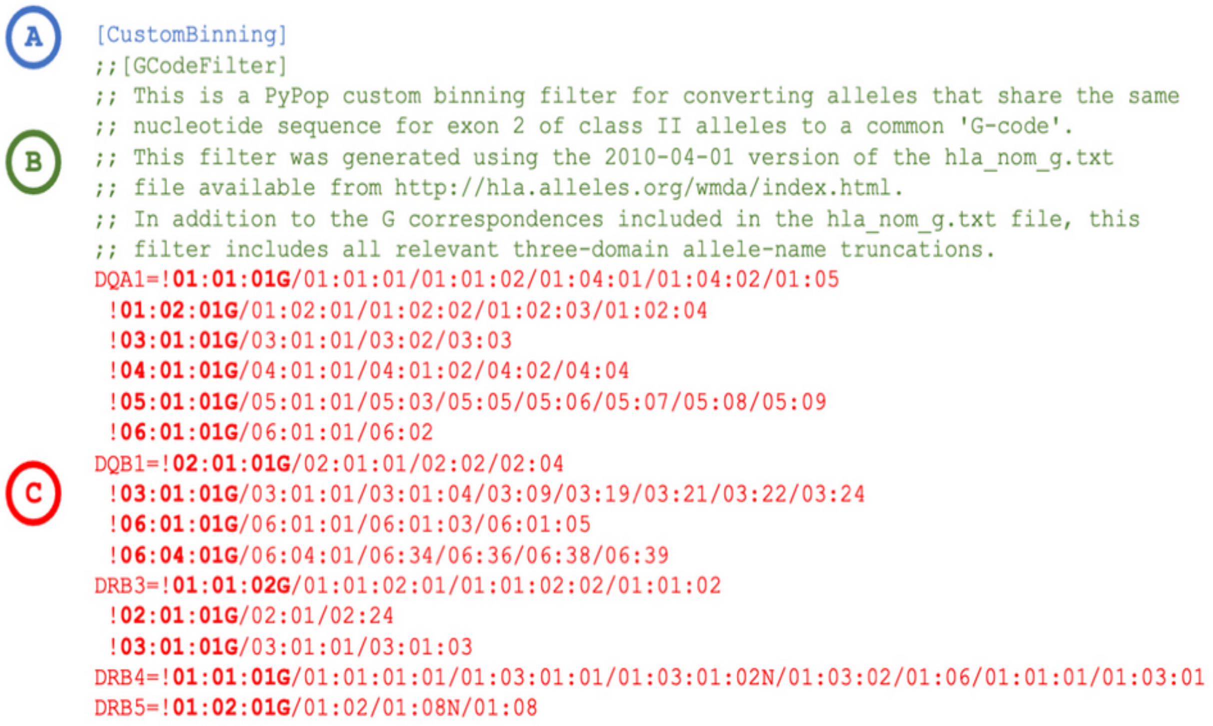


Figure 2: Example PyPop "CustomBinning" filter that would be included within the configuration ".ini" file for a PyPop run. The three elements of a custom binning filter for five HLA loci are shown. **(A): Header block**. Every custom binning filter begins with the `[CustomBinning]` keyword. **(B): Comment block (optional)**. Comments are indicated with double semicolons. This comment block identifies the type of filter (here, "GCode") and includes specific details about the source of the data used to inform the filter. (**C**): **Filters block**. Filters for DQA1, DQB1, DRB3, DRB4 and DRB5 are shown. Each filter starts with an exclamation point, which is followed by the group identifier (shown in **bold**). The group identifier and its constituent alleles are delimited by forward slashes. Multiple groups for a locus are defined on separate lines, and all groups after the first start with a whitespace. When the filter is applied, any alleles in the dataset that are in a group will be converted to the group identifier for PyPop analysis.



```
[CustomBinning]
;;[GCodeFilter]
;; This is a PyPop custom binning filter for converting alleles that share the same
;; nucleotide sequence for exon 2 of class II alleles to a common 'G-code'.
;; This filter was generated using the 2010-04-01 version of the hla_nom_g.txt
;; file available from http://hla.alleles.org/wmda/index.html.
;; In addition to the G correspondences included in the hla_nom_g.txt file, this
;; filter includes all relevant three-domain allele-name truncations.
DQA1=!01:01:01G/01:01:01/01:01:02/01:04:01/01:04:02/01:05
 !01:02:01G/01:02:01/01:02:02/01:02:03/01:02:04
 !03:01:01G/03:01:01/03:02/03:03
 !04:01:01G/04:01:01/04:01:02/04:02/04:04
 !05:01:01G/05:01:01/05:03/05:05/05:06/05:07/05:08/05:09
 !06:01:01G/06:01:01/06:02
DQB1=!02:01:01G/02:01:01/02:02/02:04
 !03:01:01G/03:01:01/03:01:04/03:09/03:19/03:21/03:22/03:24
 !06:01:01G/06:01:01/06:01:03/06:01:05
 !06:04:01G/06:04:01/06:34/06:36/06:38/06:39
DRB3=!01:01:02G/01:01:02:01/01:01:02:02/01:01:02
 !02:01:01G/02:01/02:24
 !03:01:01G/03:01:01/03:01:03
DRB4=!01:01:01G/01:01:01:01/01:03:01:01/01:03:01:02N/01:03:02/01:06/01:01:01/01:03:01
DRB5=!01:02:01G/01:02/01:08N/01:08
```

Table 1. Comparison of the default text-based output for a single two-locus pairwise LD measures for a pre-1.0.0 version (a) and 1.0.0 version (b) of PyPop, which include the new ALD measures, W_{12} and W_{21}, denoted by `ALD_1_2` and `ALD_2_1` in the output, respectively. Note that the `# permu` and `p-value` columns are now only displayed if a permutation test is run.

```
II. Multi-locus Analyses
========================

Haplotype/ linkage disequlibrium (LD) statistics
________________________________________________

Pairwise LD estimates
---------------------
Locus pair        D          D'        Wn    ln(L_1) ln(L_0)       S  # permu  p-value
A:C         0.01465     0.49229   0.39472    -289.09 -326.81   75.44  --       --
```

(a) 0.7.0 output

```
II. Multi-locus Analyses
========================

Haplotype/ linkage disequlibrium (LD) statistics
________________________________________________

Pairwise LD estimates
---------------------
Locus pair        D          D'        Wn    ln(L_1) ln(L_0)       S   ALD_1_2  ALD_2_1
A:C         0.01465    0.49229   0.39472     -289.09 -326.81   75.44   0.41435  0.37525
```

(b) 1.0.0 and later output including new ALD measure

Table 2. Comparison of haplotype estimation output indicating use of both the new nomenclature and the GL String haplotype separator.

```
Haplotypes sorted by name                | Haplotypes sorted by frequency
haplotype       frequency   # copies     | haplotype          frequency # copies
0101:1301:0402:   0.02222        2.0     | 0201:1401:0402:      0.03335      3.0
0101:1301:1101:   0.01111        1.0     | 3204:1401:0802:      0.03333      3.0
```

(b) 0.7.0 output with old nomenclature and separator

```
Haplotypes sorted by name                 | Haplotypes sorted by frequency
haplotype           frequency  # copies   | haplotype          frequency # copies
01:01~13:01~04:02   0.02222       2.0     | 02:01~14:01~04:02   0.03335      3.0
01:01~13:01~11:01   0.01111       1.0     | 32:04~14:01~08:02   0.03333      3.0
```

(b) 1.0.0 and later output using new nomenclature and GL String '~' operator

Table 3. List of possible types of TSV files, their row data type and a brief description, including the generation of files containing multi-locus analyses with an arbitrary number of *n* loci.

Default file name suffix	**Row data**	**Description**
`1-locus-summary.tsv`	locus	Consists of a line for population and locus, with fields for number of gametes, number of distinct alleles, HWP p-value for the Chi-square test and all other single locus statistics.
`1-locus-allele.tsv`	allele	Consists of a line for each combination of population, locus and allele. The line of data contains the allele frequency (allele.freq) and count (allele.count)
`1-locus-genotype.tsv`	genotype	Consists of a line for each combination of population, locus and genotype, with individual genotypes statistics (only output if individual statistics are selected by the user)
`1-locus-hardyweinberg.tsv`	locus	Consists of a line for each population and locus, with fields for number of distinct alleles and several versions of computing p-values for HWP (Guo and Thompson original and monte-carlo method, full enumeration when possible, heterozygotes, homozygotes)
`2-locus-summary.tsv`	locus	Consists of a line for each combination of population, and locus group. Columns representing locus-level statistics. If a pairwise analysis has been requested, it will also include the pairwise LD statistics discussed above, D’, Wn and ALD12, ALD21.
`2-locus-haplo.tsv`	haplotype	This is analogous to the 1-locus-allele.tsv, except with information for each population’s haplotype, such as the estimated haplotype count and frequency. If pairwise analysis has been selected, it will also include individual haplotype D’ and Wn measures.
n`-locus-summary.tsv`	locus	Analogous to the 2-locus-summary.tsv output, but no pairwise statistics
n`-locus-haplo.tsv`	haplotype	Analogous to the 2-locus-haplo.tsv output, but omits the individual pairwise LD measurements

References

1. Wright S. Systems of mating. I. The biometric relations between parent and offspring. *Genetics* (1921) 6:111–123. doi: 10.1093/genetics/6.2.111
2. Wright S. Systems of mating. II. The effects of inbreeding on the genetic composition of a population. *Genetics* (1921) 6:124–143. doi: 10.1093/genetics/6.2.124
3. Fisher RA. On the Dominance Ratio. *Proc R Soc Edinb* (1923) 42:321–341. doi: 10.1017/S0370164600023993
4. Haldane JBS. A mathematical theory of natural and artificial selection. Part I. *Trans Camb Philos Soc* (1924) 23:19–41.
5. Haldane JBS. A Mathematical Theory of Natural and Artificial Selection. Part Ii the Influence of Partial Self-Fertilisation, Inbreeding, Assortative Mating, and Selective Fertilisation on the Composition of Mendelian Populations, and on Natural Selection. *Proc Camb Philos Soc* (1924) 1:158–163. doi: 10.1111/j.1469-185X.1924.tb00546.x
6. Felsenstein J. PHYLIP-Phylogeny Inference Package (Version 3.2). *Cladistics* (1989) 5:164–166. doi: 10.1111/j.1096-0031.1989.tb00562.x
7. Kumar S, Tamura K, Nei M. MEGA: Molecular Evolutionary Genetics Analysis software for microcomputers. *Comput Appl Biosci* (1994) 10:189–191. doi: 10.1093/bioinformatics/10.2.189
8. Lancaster AK, Nelson MP, Meyer D, Single RM, Thomson G. PyPop: a software framework for population genomics: analyzing large-scale multi-locus genotype data. *Pac Symp Biocomput* (2003)514–525.
9. Lancaster AK, Single RM, Solberg OD, Nelson MP, Thomson G. PyPop update – a software pipeline for large-scale multilocus population genomics. *Tissue Antigens* (2007) 69:192–197. doi: 10.1111/j.1399-0039.2006.00769.x
10. Lancaster AK, Nelson MP, Meyer D, Single RM, Solberg O. *PyPop User Guide: User Guide for Python for Population Genomics*. 1.0.0. (2024). http://pypop.org/docs/ [Accessed January 21, 2024]
11. Lancaster AK, Nelson MP, Single RM, Meyer D, Thomson G. "Software framework for the Biostatistics Core of the International Histocompatibility Working Group.," In: Hansen JA, editor. *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*. Seattle, WA: IHWG Press (2007). p. 510–517
12. Solberg OD, Mack SJ, Lancaster AK, Single RM, Tsai Y, Sanchez-Mazas A, Thomson G. Balancing selection and heterogeneity across the classical human leukocyte antigen loci: A meta-analytic review of 497 population studies. *Hum Immunol* (2008) 69:443–464. doi: 10.1016/j.humimm.2008.05.001
13. DeCandia AL, Brzeski KE, Heppenheimer E, Caro CV, Camenisch G, Wandeler P, Driscoll C, vonHoldt BM. Urban colonization through multiple genetic lenses: The city-fox phenomenon revisited. *Ecol Evol* (2019) 9:2046–2060. doi: 10.1002/ece3.4898
14. Hernández-Verdin E, Ganelón-Ríos A, Pettet-Ruiz G, Sánchez-Garza M, Reinoso-Reyes J, López-Revilla R. CYP2C9, CYP2D6, G6PD, GCLC, GSTM1 and NAT2 gene polymorphisms and risk of adverse reactions to sulfamethoxazole and ciprofloxacin in San Luis Potosí, Mexico. *Meta Gene* (2019) 21:100574. doi: 10.1016/j.mgene.2019.100574
15. Thomson G, Single RM. Conditional asymmetric linkage disequilibrium (ALD): extending the bi-allelic r^2 measure. *Genetics* (2014) 198:321–331. doi: 10.1534/genetics.114.165266
16. Hedrick PW. Gametic disequilibrium measures: proceed with caution. *Genetics* (1987) 117:331–341. doi: 10.1093/genetics/117.2.331
17. Cramér H. *Mathematical Methods of Statistics*. Princeton, NJ: Princeton University Press (1946). 592 p.
18. Kulski JK, Mawart A, Marie K, Tay GK, AlSafar HS. MHC class I polymorphic *Alu* insertion

(POALIN) allele and haplotype frequencies in the Arabs of the United Arab Emirates and other world populations. *Int J Immunogenet* (2019) 46:247–262. doi: 10.1111/iji.12426

19. Nunes K, Maia MHT, Dos Santos EJM, Dos Santos SEB, Guerreiro JF, Petzl-Erler ML, Bedoya G, Gallo C, Poletti G, Llop E, et al. How natural selection shapes genetic differentiation in the MHC region: A case study with Native Americans. *Hum Immunol* (2021) 82:523–531. doi: 10.1016/j.humimm.2021.03.005
20. Gragert L, Spellman SR, Shaw BE, Maiers M. Unrelated Stem Cell Donor HLA Match Likelihood in the US Registry Incorporating HLA-DPB1 Permissive Mismatching. *Transplant Cell Ther* (2023) 29:244–252. doi: 10.1016/j.jtct.2022.12.027
21. Mack SJ, Udell J, Cohen F, Osoegawa K, Hawbecker SK, Noonan DA, Ladner MB, Goodridge D, Trachtenberg EA, Oksenberg JR, et al. High resolution HLA analysis reveals independent class I haplotypes and amino-acid motifs protective for multiple sclerosis. *Genes Immun* (2019) 20:308–326. doi: 10.1038/s41435-017-0006-8
22. Pacheco MA, Schneider KA, Céspedes N, Herrera S, Arévalo-Herrera M, Escalante AA. Limited differentiation among Plasmodium vivax populations from the northwest and to the south Pacific Coast of Colombia: A malaria corridor? *PLoS Negl Trop Dis* (2019) 13:e0007310. doi: 10.1371/journal.pntd.0007310
23. Pacheco MA, Forero-Peña DA, Schneider KA, Chavero M, Gamardo A, Figuera L, Kadakia ER, Grillet ME, Oliveira-Ferreira J, Escalante AA. Malaria in Venezuela: changes in the complexity of infection reflects the increment in transmission intensity. *Malar J* (2020) 19:176. doi: 10.1186/s12936-020-03247-z
24. Marsh SGE, Albert ED, Bodmer WF, Bontrop RE, Dupont B, Erlich HA, Fernández-Viña M, Geraghty DE, Holdsworth R, Hurley CK, et al. Nomenclature for factors of the HLA system, 2010. *Tissue Antigens* (2010) 75:291–455. doi: 10.1111/j.1399-0039.2010.01466.x
25. Milius RP, Mack SJ, Hollenbach JA, Pollack J, Heuer ML, Gragert L, Spellman S, Guethlein LA, Trachtenberg EA, Cooley S, et al. Genotype List String: a grammar for describing HLA and KIR genotyping results in a text string. *Tissue Antigens* (2013) 82:106–112. doi: 10.1111/tan.12150
26. Mack SJ, Schefzyk D, Millius RP, Maiers M, Hollenbach JA, Pollack J, Heuer ML, Gragert L, Spellman SR, Guethlein LA, et al. Genotype List String 1.1: Extending the Genotype List String grammar for describing HLA and Killer-cell Immunoglobulin-like Receptor genotypes. *HLA* (2023) 102:206–212. doi: 10.1111/tan.15126
27. Taylor GM, Hussain A, Lightfoot TJ, Birch JM, Eden TOB, Greaves MF. HLA-associated susceptibility to childhood B-cell precursor ALL: definition and role of HLA-DPB1 supertypes. *Br J Cancer* (2008) 98:1125–1131. doi: 10.1038/sj.bjc.6604257
28. Zino E. A T-cell epitope encoded by a subset of HLA-DPB1 alleles determines nonpermissive mismatches for hematologic stem cell transplantation. *Blood* (2003) 103:1417–1424. doi: 10.1182/blood-2003-04-1279
29. Sizzano F, Zito L, Crivello P, Crocchiolo R, Vago L, Zino E, Fleischhauer K. Significantly higher frequencies of alloreactive CD4+ T cells responding to nonpermissive than to permissive HLA-DPB1 T-cell epitope disparities. *Blood* (2010) 116:1991–1992. doi: 10.1182/blood-2010-05-284687
30. Maiers M, Hurley CK, Perlee L, Fernandez-Vina M, Baisch J, Cook D, Fraser P, Heine U, Hsu S, Leffell MS, et al. Maintaining updated DNA-based HLA assignments in the National Marrow Donor Program Bone Marrow Registry. *Rev Immunogenet* (2000) 2:449–460.
31. Hurley CK, Setterholm M, Lau M, Pollack MS, Noreen H, Howard A, Fernandez-Vina M, Kukuruga D, Müller CR, Venance M, et al. Hematopoietic stem cell donor registry strategies for assigning search determinants and matching relationships. *Bone Marrow Transplant* (2004)

33:443–450. doi: 10.1038/sj.bmt.1704365

32. Hinsen K. Technical Debt in Computational Science. *Comput Sci Eng* (2015) 17:103–107. doi: 10.1109/MCSE.2015.113
33. Chang C-J, Osoegawa K, Milius RP, Maiers M, Xiao W, Fernandez-Viňa M, Mack SJ. Collection and storage of HLA NGS genotyping data for the 17th International HLA and Immunogenetics Workshop. *Hum Immunol* (2018) 79:77–86. doi: 10.1016/j.humimm.2017.12.004
34. Sochat V. Remove/numeric, update to python 3.0 by vsoch · Pull Request #12 · alexlancaster/pypop. *GitHub* (2017) https://github.com/alexlancaster/pypop/pull/12 [Accessed December 15, 2023]
35. Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. *PLOS ONE* (2017) 12:e0177459. doi: 10.1371/journal.pone.0177459
36. Lancaster AK. Port PyPop to Numpy by alexlancaster · Pull Request #24 · alexlancaster/pypop. *GitHub* (2017) https://github.com/alexlancaster/pypop/pull/24 [Accessed December 15, 2023]
37. Rickerby J, Jadoul Y, Schreiner H, Darbois M, Bokota G, Dower S, Kuczys J, Lessa H, Crall J, Treinish M, et al. pypa/cibuildwheel: v2.12.1. (2023) doi: 10.5281/zenodo.7722899
38. European Organization For Nuclear Research, OpenAIRE. Zenodo. (2013) doi: 10.25495/7GXK-RD71
39. McKiernan EC, Bourne PE, Brown CT, Buck S, Kenall A, Lin J, McDougall D, Nosek BA, Ram K, Soderberg CK, et al. How open science helps researchers succeed. *eLife* (2016) 5:e16800. doi: 10.7554/eLife.16800
40. Lancaster AK, Nelson MP, Single RM, Solberg OD, Tsai Y, Meyer D, Mariani M, Sochat V, Kornel K, Spaaks JH, et al. PyPop: Python for Population Genomics. (2023) doi: 10.5281/zenodo.10080667
41. Walsh N. *DocBook 5: The Definitive Guide: The Official Documentation for DocBook*. 1st edition. Hamilton RL, editor. Beijing ; Sebastopol: O'Reilly Media (2010). 548 p. https://tdg.docbook.org/tdg/5.0/docbook
42. reStructuredText. (2021) https://docutils.sourceforge.io/rst.html [Accessed January 12, 2024]
43. Brandl G. Sphinx: Python Documentation Generator. (2010) https://www.sphinx-doc.org/ [Accessed January 17, 2024]