



PyPop User Guide

User Guide for Python for Population Genomics

Release 1.3.1

Alexander K. Lancaster

Mark P. Nelson

Diogo Meyer

Richard M. Single

Owen D. Solberg

Oct 14, 2025

PyPop User Guide: User Guide for Python for Population Genomics

Copyright © 2003-2009 Regents of the University of California.

Copyright © 2025 PyPop contributors.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the License chapter.

Document revision: 1.3.1.post17+g1cf74ac58

CONTENTS

1	Installing PyPop	3
1.1	Quickstart Guide	3
1.2	Examples	3
	Checking version and installation	3
	Run a minimal dataset:	4
1.3	Detailed installation instructions	4
	Install Python 3 and <code>pip</code>	4
	Install package from PyPI	4
	Post-install PATH adjustments	5
	Uninstalling PyPop	6
1.4	Support and development	6
2	Getting started with PyPop	7
2.1	Introduction	7
	Interactive mode	7
	Command-line mode	8
	Programmatic access to PyPop	9
	What happens when you run PyPop?	10
2.2	Using <code>popmeta</code> to aggregate results	10
2.3	Command-line interfaces	10
	<code>pypop</code> usage	11
	<code>popmeta</code> usage	11
2.4	The data file	12
	Sample files	12
	Missing data	14
2.5	The configuration file	15
	A simple configuration file	15
	Specifying data formats	16
	Analysis options	17
	Advanced options	18
	Advanced filtering sections	20
2.6	PyPop API examples	23
3	Interpreting PyPop output	26
3.1	Population summary	26
3.2	Single locus analyses	27
	Basic allele count information	27
	Chi-square test for deviation from Hardy-Weinberg proportions (HWP).	27
	Exact test for deviation from HWP	29
	The Ewens-Watterson homozygosity test of neutrality	29
3.3	Multi-locus analyses	30

	All pairwise LD	30
	Haplotype frequency estimation	32
4	Contributing to PyPop	34
4.1	Reporting and requesting	34
	Did you find a bug?	34
	Documentation improvements	35
	Feature requests and feedback	35
4.2	Making a code contribution	35
	Fork this repository	36
	Clone the main repository	36
	Make a new branch	36
	Build locally and make your changes	36
	Keep your branch in sync with upstream	36
	Use nox for testing and code quality	37
	Update AUTHORS.rst	38
	Make a Pull Request	38
4.3	Installation for developers	39
	Install the build environment	40
	Build PyPop	42
4.4	Install package from GitHub Releases	43
4.5	Making a documentation or website contribution	43
	Overview	44
	Structure	44
	Modifying documentation	45
4.6	Crediting contributors	46
	AUTHORS.rst, NEWS.md and GitHub Release Notes	46
	Authorship on Zenodo archives of releases	46
	Scientific publications (papers)	47
5	Authors and history	48
5.1	PyPop contributors	48
	Third-party modules	49
5.2	PyPop Release History	49
	1.3.1- 2025-10-07	49
	1.3.0- 2025-09-08	49
	1.2.2- 2025-07-28	50
	1.2.1] - 2025-04-29	50
	1.2.0] - 2025-02-04	51
	1.1.2] - 2024-11-18	52
	1.1.1] - 2024-09-10	52
	1.1.0] - 2024-05-30	53
	1.0.2] - 2024-02-24	53
	1.0.1] - 2024-02-11	54
	1.0.0] - 2023-11-07	54
	0.7.0] - 2008-09-09	56
	0.6.0 - 2005-04-13	57
	0.5.2 (public beta) - 2004-03-09	58
	0.5.1 (public beta) - 2004-02-26	58
	0.5 (public beta) - 2003-12-31	59
	0.4.3beta	59
6	Licenses	63
6.1	License terms for PyPop	63
	GNU General Public License	63

6.2 License for PyPop documentation 68

GNU Free Documentation License 68

Bibliography **73**



PyPop (Python for Population Genomics) is an environment for doing large-scale population genetic analyses including:

- conformity to Hardy-Weinberg expectations
- tests for balancing or directional selection
- estimates of haplotype frequencies and measures and tests of significance for linkage disequilibrium (LD).

PyPop is an object-oriented framework implemented in [Python](#)¹, but also contains C extensions for some computationally intensive tasks. Output of analyses are stored in XML format for maximum downstream flexibility. PyPop also has an internal facility for additionally aggregating the output XML and generating output tab-separated (TSV) files, as well as well as generating a default plain text file summary for each population.

Although it can be run on any kind of genotype data, it has additional support for analyzing population genotype with allelic nomenclature from the human leukocyte antigen (HLA) region. An outline of PyPop can be found in our 2024 paper ([Lancaster et al., 2024](#)), and two previous papers.

How to cite PyPop

If you write a paper that uses PyPop in your analysis, please cite **both**:

- our [2024 article](#)² in *Frontiers in Immunology*:

Lancaster AK, Single RM, Mack SJ, Sochat V, Mariani MP, Webster GD. (2024) “PyPop: A mature open-source software pipeline for population genomics.” *Front. Immunol.* **15**:1378512 doi: [10.3389/fimmu.2024.1378512](https://doi.org/10.3389/fimmu.2024.1378512)³

- **and** a citation to the [Zenodo record](#)⁴ which includes a DOI for the version of the software you used in your analyses. Citing this record and DOI supports reproducibility by allowing researchers to determine the exact version of PyPop used in any particular analysis. In addition, it allows retrieval of long-term software source-code archives, independent of the original developers.

Here’s how to cite the correct version:

- If you have PyPop version 1.1.2 or later, currently installed, you can run:

```
pypop --citation
```

which outputs the Zenodo record citation in the simple “APA” format (you can also choose from BibTeX, EndNote, RIS and other formats, see the section on [command-line interfaces](#)⁵ in the *User Guide* for more details).

- If you do not have PyPop installed, have a release of PyPop earlier than 1.1.2, or otherwise want to obtain the DOI and citation for specific versions, follow these steps:
 - 1) First visit the DOI for the overall Zenodo record: [10.5281/zenodo.10080667](https://doi.org/10.5281/zenodo.10080667)⁶. This DOI represents **all versions**, and will always resolve to the latest one.
 - 2) When you are viewing the record, look for the **Versions** box in the right-sidebar. Here are listed all versions (including older versions).
 - 3) Select and click the version-specific DOI that matches the specific version of PyPop that you used for your analysis.

- 4) Once you are visiting the Zenodo record for the specific version, under the **Citation** box in the right-sidebar, select the citation format you wish to use and click to copy the citation. It will contain link to the version-specific DOI, and be of the form:

Lancaster, AK et al. (YYYY) “PyPop: Python for Population Genomics” (Version X.Y.Z) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.XXXXXX>

Note that citation metadata for the current Zenodo record is also stored in `CITATION.cff`⁷

Two previous papers are also available (but not necessary to cite):

- Lancaster AK, Single RM, Solberg OD, Nelson MP, Thomson G (2007) “PyPop update - a software pipeline for large-scale multilocus population genomics” *Tissue Antigens* 69 (s1), 192-197. [journal page⁸, preprint PDF (112 kB)⁹].
- Lancaster A, Nelson MP, Single RM, Meyer D, Thomson G (2003) “PyPop: a software framework for population genomics: analyzing large-scale multi-locus genotype data”, in *Pacific Symposium on Biocomputing* vol. 8:514-525 (edited by R B Altman. et al., World Scientific, Singapore, 2003) [PubMed Central¹⁰, PDF (344 kB)¹¹].

PyPop was originally developed for the analysis of data for the 13th [International Histocompatibility Workshop and Conference](#)¹² held in Seattle, Washington in 2002 (Meyer *et al.*, 2007, Single *et al.* 2007a, 2007b). For more details on the design and technical details of PyPop, please consult Lancaster *et al.* (2003, 2007a, 2007b, 2024).

Acknowledgements

This work has benefited from the support of NIH grant AI49213 (13th IHW) and NIH/NIAID Contract number HHSN266200400076C N01-AI-40076. Thanks to Steven J. Mack, Kristie A. Mather, Steve G.E. Marsh, Mark Grote and Leslie Louie for helpful comments and testing.

How to use this guide

This guide to PyPop contains five main parts:

- [Installing PyPop](#) describes how to install PyPop, including pre-release binaries.
- [Getting started with PyPop](#) describes how to run PyPop.
- [Interpreting PyPop output](#) details the population genetic methods and statistics that PyPop computes.
- [Contributing to PyPop](#) details how to contribute to ongoing PyPop code and documentation.
- [Authors and history](#) acknowledges our contributors and project history.

References to the *API Reference* can be found in the *PyPop API Reference*: [HTML](#)¹³ | [PDF](#)¹⁴

Notes

1. <https://www.python.org/>
2. <https://www.frontiersin.org/journals/immunology/articles/10.3389/fimmu.2024.1378512/full>
3. <https://doi.org/10.3389/fimmu.2024.1378512>
4. <https://zenodo.org/records/10080667>
5. <http://pypop.org/docs/guide-chapter-usage.html#command-line-interfaces>
6. <https://zenodo.org/doi/10.5281/zenodo.10080667>
7. <https://github.com/alexlancaster/pypop/blob/main/CITATION.cff>
8. <http://dx.doi.org/10.1111/j.1399-0039.2006.00769.x>
9. <http://pypop.org/tissue-antigens-lancaster-2007.pdf>
10. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3891851/>
11. <http://pypop.org/psb-pypop.pdf>
12. <http://www.ihwg.org/>
13. <http://pypop.org/api>
14. <http://pypop.org/pypop-api-1.3.1.pdf>

INSTALLING PYPOP

Attention

The package name for installation purposes is `pypop-genomics` - to avoid conflicting with an unrelated package with the name `pypop` already on [PyPI](#)¹.

1.1 Quickstart Guide

Installing `pypop-genomics`

If you already have Python and `pip` installed, install using the following:

```
pip install pypop-genomics
```

Otherwise, follow [these instructions](#) to install Python 3 and `pip`.

Once `pypop-genomics` is installed, depending on your platform, you may also need to [adjust](#) your `PATH` environment variable.

Upgrading `pypop-genomics`

```
pip install -U pypop-genomics
```

Uninstalling `pypop-genomics`

```
pip uninstall pypop-genomics
```

For more, including handling common installation issues, see the [detailed installation instructions](#) .

Once you have installed `pypop-genomics`, you can move on to try some [example runs](#).

1.2 Examples

These are examples of how to check that the program is installed and some minimal use cases.

Checking version and installation

```
pypop --version
```

This simply reports the version number and other information about PyPop, and indirectly checks that the program is installed. If all is well, you should see something like:

```
pypop 1.0.0
[Python 3.10.9 | Linux.x86_64-x86_64 | x86_64]
Copyright (C) 2003-2006 Regents of the University of California.
Copyright (C) 2007-2023 PyPop team.
This is free software. There is NO warranty; not even for
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

You can also run `pypop --help` to see a full list and explanation of all the options available.

Run a minimal dataset:

Download test `.ini` and `.pop` files: [minimal.ini](#)² and [USAFEL-UchiTelle-small.pop](#)³. You can then run them

```
pypop -c minimal.ini USAFEL-UchiTelle-small.pop
```

If you have already cloned the git repository and it is your working directory, you can simply run

```
pypop -c tests/data/minimal.ini tests/data/USAFEL-UchiTelle-small.pop
```

This will generate the following two files, an XML output file and a plain text version:

```
USAFEL-UchiTelle-small-out.xml
USAFEL-UchiTelle-small-out.txt
```

1.3 Detailed installation instructions

There are three main steps:

1. install Python and `pip`
2. install package from PyPI
3. adjusting your `PATH` variable after installation

Install Python 3 and `pip`

A full description of installing Python and `pip` on your system is beyond the scope of this guide, we recommend starting here:

<https://wiki.python.org/moin/BeginnersGuide/Download>

Here are some additional platform-specific notes that may be helpful:

- Most Linux distributions come with Python 3 preinstalled. On most modern systems, `pip` and `python` will default to Python 3.
- MacOS 10.9 (Jaguar) up until 12.3 (Catalina), used to ship with Python 2 pre-installed, but it now has to be manually installed. See the [MacOS quick-start guide](#)⁴ in the official documentation for how to install Python 3. (Note that if Python is installed on Mac via the MacOS developer tools, it may include the version 3 suffix on commands, e.g. `python3` and `pip3`, so modify the below, accordingly).
- For Windows, see also the [Windows quick-start guide](#)⁵ in the official documentation. Running `python` in the Windows command terminal in Windows 11 and later will launch the installer for the Microsoft-maintained Windows package of Python 3.

Install package from PyPI

Once you have both `python` and `pip` installed, you can use `pip` to install pre-compiled binary “wheels” of `pypop-genomics` directly from PyPI⁶.

```
pip install pypop-genomics
```

Note

If, for whatever reason, you cannot use these binaries (e.g. the pre-compiled binaries are not available for your platform), you may need to follow the [developer installation instructions](#)⁷ in the contributors guide.

Upgrade an existing PyPop installation

To update an existing installation to a newer version, use the same command as above, but add the `--upgrade` (short version: `-U`) flag, i.e.

```
pip install -U pypop-genomics
```

Installing from Test PyPI

From time to time, we may make available packages on the [Test PyPI](#)⁸ instance, rather than through the main instance. The above installation and updating instructions can be used, by appending the following:

```
--extra-index-url https://test.pypi.org/simple/
```

to the above `pip` commands.

Issues with installation permission

By default, `pip` will attempt to install the `pypop-genomics` package wherever the current Python installation is installed. This location may be a user-specific virtual environment (like `conda`, see below), or a system-wide installation. On many Unix-based systems, Python will generally already be pre-installed in a “system-wide” location (e.g. under `/usr/lib`) which is read-only for regular users. (This can also be true for system-installed versions of Python on Windows and MacOS.)

When `pip install` cannot install in a read-only system-wide location, `pip` will gracefully “fall-back” to installing just for you in your home directory (typically `~/.local/lib/python<VER>` where `<VER>` is the version number of your current Python). In general, this is what is wanted, so the above instructions are normally sufficient.

However, you can also explicitly set installation to be in the user directory, by adding the `--user` command-line option to the `pip install` command, i.e.:

```
pip install pypop-genomics --user
```

This may be necessary in certain cases where `pip install` doesn’t install into the expected user directory.

Installing within a conda environment

In the special case that you are installing from within an activated user-specific `conda` virtual environment that provides Python, then you should **not** add the `--user` because it will install it in `~/.local/lib/` rather than under the user-specific `conda` virtual environment in `~/.conda/envs/`.

Post-install PATH adjustments

You may need to adjust the `PATH` settings (especially on Windows) for the `pypop` scripts to be visible when run from your console application, without having to supply the full path to the `pypop` executable file.

Warning

Pay close attention to the “WARNINGS” that are shown during the pip installation, they will often note which directories need to be added to the PATH.

- On Linux and MacOS, systems this is normally fairly simple and only requires edit of the shell `.profile`, or similar and addition of the `$HOME/.local/bin` to the PATH variable, followed by a restart of the terminal.
- For Windows, however, as noted in most online [instructions](#)⁹, this may need additional help from your system administrator if your user doesn’t have the right permissions, and also require a system reboot.

Uninstalling PyPop

To uninstall the current version of `pypop-genomics`:

```
pip uninstall pypop-genomics
```

1.4 Support and development

Please submit any bug reports, feature requests or questions, via our GitHub issue tracker (see our [bug reporting guidelines](#)¹⁰ for more details on how to file a good bug report):

<https://github.com/alexlancaster/pypop/issues>

Please do not report bugs via private email to developers.

The development of the code for PyPop is via our GitHub project:

<https://github.com/alexlancaster/pypop>

Notes

1. <https://pypi.org>
2. <https://raw.githubusercontent.com/alexlancaster/pypop/main/tests/data/minimal.ini>
3. <https://raw.githubusercontent.com/alexlancaster/pypop/main/tests/data/USAFEL-UchiTelle-small.pop>
4. <https://docs.python.org/3/using/mac.html>
5. <https://docs.python.org/3/using/windows.html>
6. <https://pypi.org/>
7. <http://pypop.org/docs/guide-chapter-contributing.html#installation-for-developers>
8. <https://test.pypi.org/>
9. <https://www.computerhope.com/issues/ch000549.htm>
10. <http://pypop.org/docs/guide-chapter-contributing.html#reporting-and-requesting>

GETTING STARTED WITH PYPOP

2.1 Introduction

You may use **PyPop** to analyze many different kinds of data, including allele-level genotype data (as in [Listing 2.1](#)), allele-level frequency data (as in [Listing 2.6](#)), microsatellite data, SNP data, and nucleotide and amino acid sequence data.

As mentioned in the installation chapter, a minimal working example of a [configuration file \(.ini\)](#)¹, and a [population file \(.pop\)](#)², can be found by clicking the respective links.

There are three ways to run PyPop:

- interactive mode (where the program will prompt you to directly type the input it needs); and
- command-line (or “batch”) mode (where you supply all the command line options the program needs).
- programmatically, by writing a Python program that uses the *API Reference* (API).

For the most simplest application of PyPop, where you wish to analyze a single population, the interactive mode is the simplest to use. We will describe this mode first then describe command-line mode.

Note

The following assumes you have already *installed PyPop*, done any *post-install adjustments* needed for your platform, and verified that you can run the main commands (see the *Examples* section).

Interactive mode

To run PyPop in interactive mode, with a minimal “GUI”, on Windows or MacOS, you can directly click on the `pypop-interactive` file in the directory where the scripts were installed (see *post-install adjustments*).

You can also type `pypop-interactive` after starting a console application on all platforms (on MacOS and GNU/Linux, this is normally the **Terminal** program, on Windows, it's **Command prompt**).

In most cases, this will launch a console with the following:

```
PyPop: Python for Population Genomics (1.0.0)
[Python 3.10.9 | Linux.x86_64-x86_64 | x86_64]
Copyright (C) 2003-2006 Regents of the University of California
Copyright (C) 2007-2023 PyPop team.
This is free software. There is NO warranty; not even for
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

You may redistribute copies of PyPop under the terms of the GNU
General Public License. For more information about these
matters, see the file named COPYING.

Select both an '.ini' configuration file and a '.pop' file via the
system file dialog.
```

Following this:

1. the system file dialog will appear prompting you to select an `.ini` *configuration file*.
2. a second system file dialog will prompt you for a `.pop` *data file*.
3. after both files are selected the console will display the processing of the file:

```
PyPop is processing sample.pop ...
PyPop run complete!
XML output(s) can be found in: ['sample-out.xml']
Plain text output(s) can be found in: ['sample-out.txt']
Press Enter to continue...
```

4. when the run is completed, the last line will prompt you to press `Enter` to leave the console window (highlighted above).

If the system file GUI dialog does not appear (e.g. if you are running on a terminal without a display), it will fall-back to text-mode entry for the files, where you need to type the full (either relative or absolute) paths to the files. The output should resemble:

```
PyPop: Python for Population Genomics (1.0.0)
[Python 3.10.9 | Linux.x86_64-x86_64 | x86_64]
Copyright (C) 2003-2006 Regents of the University of California
Copyright (C) 2007-2023 PyPop team.
This is free software. There is NO warranty; not even for
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

You may redistribute copies of PyPop under the terms of the GNU
General Public License. For more information about these
matters, see the file named COPYING.

To accept the default in brackets for each filename, simply press
return for each prompt.

Please enter config filename [config.ini]: sample.ini
Please enter population filename [no default]: sample.pop
PyPop is processing sample.pop ...
PyPop run complete!
XML output(s) can be found in: ['sample-out.xml']
Plain text output(s) can be found in: ['sample-out.txt']
Press Enter to continue...
```

Note

Some messages with the prefix “LOG:” may appear during the console operation. They are informational only and do not indicate improper operation of the program.

In both cases you should substitute the names of your own configuration (e.g., `config.ini`) and population file (e.g., `Guatemalan.pop`) for `sample.ini` and `sample.pop` (highlighted above). The formats for these files are described in the sections on the *data file* and *configuration file*, below.

Command-line mode

To run PyPop in the more common command-line (or “batch”) mode, you can run PyPop from the console (as noted above, on Windows: open **Command prompt**, aka a “DOS shell”; on MacOS or GNU/Linux: open the **Terminal** application). Change to a directory where your `.pop` file is located, and type the command:

```
pypop Guatemalan.pop
```

Command-line mode assumes two things: that you have a file called `config.ini` in your current folder and that you also have your population file in the current folder, otherwise you will need to supply the full path to the file. You can specify a particular configuration file for PyPop to use, by supplying the `-c` option as follows:

```
pypop -c newconfig.ini Guatemalan.pop
```

Output to different directory

You may also redirect the output to a different directory (which must already exist) by using the `-o` option:

```
pypop -c newconfig.ini -o altdir Guatemalan.pop
```

Supplying multiple .pop files

If you have multiple `.pop` files with the same overall format (i.e. the same, or subset of, the loci listed in the `.ini` file), you can process those in one `pypop` invocation using a single `.ini` file. You either supply them directly on the command-line:

```
pypop -c config.ini Guatemalan.pop NorthAmerican.pop
```

or you use the `--filelist` command-line option to pass in a file containing a list of files, i.e.:

```
pypop -c config.ini --filelist popfilelist.txt
```

where the text file `popfilelist.txt` contains a list of the `.pop` files to be processed on separate lines, e.g.:

```
Guatemalan.pop
NorthAmerican.pop
```

Changed in version 1.3.0

Changed in version 1.3.0: New behavior: all files within `FILELIST` will be resolved relative to relative to the *parent* directory of `FILELIST`, **not** to the current working directory (the old behavior).

This ensures that files can be more straightforwardly located independently of where `pypop` is run from. For example, if your current working directory looked like the following:

```
data/popfilelist.txt
data/file1.pop
data/file2.pop
```

You would run `pypop` like:

```
pypop -c config.ini --filelist data/popfilelist.txt
```

and the contents of `popfilelist.txt` should be:

```
file1.pop
file2.pop
```

Full absolute paths will be processed as-is, and will not be treated as if they are relative to the filelist.

Please see [pypop usage](#) for the full list of command-line options.

Programmatic access to PyPop

It is also possible to use PyPop programmatically by writing a Python program that uses the Application Programmers Interface documented in the *API Reference* (API) directly. While the primary use-case for PyPop is as a standalone command-line script, and is it not fully optimized for use via a programmatic interface, much functionality is exposed as Python modules and classes. Examples of programmatic use can be found in [PyPop API examples](#).

What happens when you run PyPop?

The most common types of analysis will involve the editing of your `config.ini` file to suit your data (see [the configuration file](#)) followed by the selection of either the interactive or command-line mode described above. If your input configuration file is `configfilename` and your population file name is `popfilename.txt` the initial output will be generated quickly, but your the PyPop execution will not be finished until the text output file named `popfilename-out.txt` has been created. A successful run will produce two output files: `popfilename-out.xml`, `popfilename-out.txt`. A third output file will be created if you are using the Anthony Nolan HLA filter option for HLA data to check your input for valid/known HLA alleles: `popfilename-filter.xml`.

The `popfilename-out.xml` file is the primary output created by PyPop and the human-readable `popfilename-out.txt` file is a summary of the complete XML output. The XML output can be further transformed into plain text TSV files, either directly via `pypop` if invoked on multiple input files (using the `--enable-tsv` option, see [pypop usage](#)), or via the `popmeta` tool that aggregates results from different `pypop` runs (see [Using popmeta to aggregate results](#)).

A typical PyPop run might take anywhere from a few of minutes to a few hours, depending on how large your data set is and who else is using the system at the same time. Note that performing the `allPairwiseLDWithPermu` test may take several **days** if you have highly polymorphic loci in your data set.

2.2 Using popmeta to aggregate results

The `popmeta` script can aggregate results from a number of output XML files from individual populations into a set of tab-separated (TSV) files containing summary statistics via customized XSLT (eXtensible Stylesheet Language for Transformations) stylesheets. These TSV files can be directly imported into a spreadsheet or statistical software (e.g., **R**, **SAS**). In addition, there is some preliminary support for export into other formats, such as the population genetic software (e.g., **PHYLIP**).

Here is an example of a `popmeta` run, following on from the XML outputs generated in similar fashion in the previous `pypop` runs:

```
popmeta -o altdir Guatemalan-out.xml NorthAmerican-out.xml
```

This will generate a number of `.tsv` files, in the output directory `altdir`, of the form `1-locus-allele.tsv`, `1-locus-summary.tsv`, etc.

You can also supply a prefix to the command-line option `--prefix-tsv` so that all `.tsv` files are given a prefix, e.g.,

```
popmeta -o altdir --prefix-tsv myoutput Guatemalan-out.xml NorthAmerican-out.xml
```

Will result in files with a prefix, e.g. `myoutput-1-locus-allele.tsv`.

Note

It's highly recommended to use the `-o` option to save the output in a separate subdirectory, as the output `.tsv` files have fixed names, and will overwrite any files in the local directory with the same name. See [popmeta usage](#) for the full list of options.

Note that a similar effect can be achieved directly from a `pypop` run (assuming that the configuration file can be used for both `.pop` population files), by invoking `pypop` with the `--enable-tsv` option:

```
pypop -c newconfig.ini -o altdir Guatemalan.pop NorthAmerican.pop --enable-tsv
```

2.3 Command-line interfaces

Described below is the usage for both programs, including a full list of the current command-line options and arguments. Note that you can also view this full list of options from the program itself by supplying the `--help` option, i.e. `pypop --help`, or `popmeta --help`, respectively.

pypop usage

```
usage: pypop [-h]
             [--citation [{apalike,bibtex,endnote,ris,codemeta,cff,schema.org,zenodo}]]
             [-o OUTPUTDIR] [-V] [-c CONFIG] [-m] [-d] [-x XSLFILE] [-t]
             [--enable-ihwg] [--enable-phylip] [-p PREFIX_TSV] [-i]
             [-f FILELIST]
             [POPFILe ...]
```

Options for pypop

--citation	Possible choices: apalike, bibtex, endnote, ris, codemeta, cff, schema.org, zenodo generate citation to PyPop for this version of PyPop Default: 'apalike'
-o, --outputdir	put output in directory OUTPUTDIR
-V, --version	show program's version number and exit
-c, --config	select config file Default: 'config.ini'
-m, --testmode	run PyPop in test mode for unit testing
-d, --debug	enable debugging output (overrides config file setting)
-x, --xsl	override the default XSLT translation with XSLFILE

TSV output options

Note that `--enable-*` and `--prefix-tsv` options are only valid if `--enable-tsv/-t` is also supplied

-t, --enable-tsv	generate TSV output files (aka run 'popmeta')
--enable-ihwg	enable 13th IWHG workshop populationdata default headers
--enable-phylip	enable generation of PHYLIP .phy files
-p, --prefix-tsv	append PREFIX_TSV to the output TSV files

Mutually exclusive input options

-i, --interactive	run in interactive mode, prompting user for file names
-f, --filelist	file containing list of files (one per line) to process. files are resolved relative to FILELIST, unless absolute. mutually exclusive with supplying POPFILEs)
POPFILe	input population (.pop) file(s) Default: []

popmeta usage

```
usage: popmeta [-h]
               [--citation [{apalike,bibtex,endnote,ris,codemeta,cff,schema.org,zenodo}]]
               [-o OUTPUTDIR] [-V] [-p PREFIX_TSV] [--disable-tsv]
               [--output-meta] [-x XSLDIR] [--enable-ihwg]
               [--enable-phylip] [-b FACTOR]
               XMLFILE [XMLFILE ...]
```

Positional Arguments

XMLFILE XML (.xml) file(s) generated by pypop runs
Default: []

Options for popmeta

--citation Possible choices: apalike, bibtex, endnote, ris, codemeta, cff, schema.org, zenodo
generate citation to PyPop for this version of PyPop
Default: 'apalike'

-o, --outputdir put output in directory OUTPUTDIR

-V, --version show program's version number and exit

-p, --prefix-tsv append PREFIX_TSV to the output TSV files

--disable-tsv disable generation of .tsv TSV files

--output-meta dump the meta output file to stdout, ignore xslt file

-x, --xsldir use specified directory to find meta XSLT

--enable-ihwg enable 13th IWHG workshop populationdata default headers

Mutually exclusive popmeta options

--enable-phylip enable generation of PHYLIP .phy files

-b, --batchsize process in batches of size total/FACTOR rather than all at once, by default do separately (batchsize=0)
Default: 0

2.4 The data file

Sample files

Data can be input either as genotypes, or in an allele count format, depending on the format of your data.

Data files are tab-delimited

These population files are plain text files, such as you might save out of the **Notepad** application on Windows (or **Emacs**). The columns are all tab-delimited, so you can include spaces in your labels. If you have your data in a spreadsheet application, such as **Excel** or **LibreOffice**, export the file as tab-delimited text, in order to use it as PyPop data file.

Depending on how you are viewing this documentation, in some of the examples below, the columns may not appear to align with their headers, but that is purely due to how tabstops are rendered. If you copy-and-paste the data into a text editor you should be able to see that the columns are tab separated.

As you will see in the following examples, population files begin with header information. In the simplest case, the first line contains the column headers for the genotype, allele count, or, sequence information from the population. If the file contains a population data-block, then the first line consists of headers identifying the data on the second line, and the third line contains the column headers for the genotype or allele count information.

Note that for genotype data, each locus corresponds to two columns in the population file. The locus name must be repeated, with a suffix such as _1, _2 (the default) or _a, _b and must match the format defined in the `config.ini` (see [validSampleFields](#)). Although PyPop needs this distinction to be made, phase is NOT assumed, and if known it is ignored.

Listing 2.7 shows the relevant lines for the configuration to read in the data shown in Listing 2.1 and Listing 2.2.

Listing 2.1: Multi-locus allele-level genotype data

a_1	a_2	c_1	c_2	b_1	b_2
****	****	01:02	02:10:06	13:01	18:01:02
01:01	02:01	03:07	06:05	14:01	39:02:01
02:10	03:01:02	07:12	01:02	15:20	13:01
01:01	02:18	08:04	12:02	35:09:01	40:05
25:01	02:01	15:07	03:07	51:01:03	14:01
02:10	32:04	18:01	01:02	78:02:01	13:01
03:01:02	32:04	15:07	06:05	51:01:03	39:02:01

This is an example of the simplest kind of data file.

Listing 2.2: Multi-locus allele-level HLA genotype data with sample information

populat	id	a_1	a_2	c_1	c_2	b_1	b_2
UchiTelle	UT900-23	****	****	01:02	02:10:06	13:01	18:01:02
UchiTelle	UT900-24	01:01	02:01	03:07	06:05	14:01	39:02:01
UchiTelle	UT900-25	02:10	03:01:02	07:12	01:02	15:20	13:01
UchiTelle	UT900-26	01:01	02:18	08:04	12:02	35:09:01	40:05
UchiTelle	UT910-01	25:01	02:01	15:07	03:07	51:01:03	14:01
UchiTelle	UT910-02	02:10	32:04	18:01	01:02	78:02:01	13:01
UchiTelle	UT910-03	03:01:02	32:04	15:07	06:05	51:01:03	39:02:01

This example shows a data file which has non-allele data in some columns, here we have population (populat) and sample identifiers (id).

Listing 2.3: Multi-locus allele-level HLA genotype data with sample and header information

labcode	method	ethnic	contin	collect	latit	longit	
USAFEL	12th Workshop	SSOP	Telle	NW Asia	Targen Village	41 deg 12 min N	
↪	94 deg 7 min E						
populat	id	a_1	a_2	c_1	c_2	b_1	b_2
UchiTelle	UT900-23	****	****	01:02	02:10:06	13:01	18:01:02
UchiTelle	UT900-24	01:01	02:01	03:07	06:05	14:01	39:02:01
UchiTelle	UT900-25	02:10	03:01:02	07:12	01:02	15:20	13:01
UchiTelle	UT900-26	01:01	02:18	08:04	12:02	35:09:01	40:05
UchiTelle	UT910-01	25:01	02:01	15:07	03:07	51:01:03	14:01
UchiTelle	UT910-02	02:10	32:04	18:01	01:02	78:02:01	13:01
UchiTelle	UT910-03	03:01:02	32:04	15:07	06:05	51:01:03	39:02:01

This is an example of a data file which is identical to Listing 2.2, but which includes population level information.

Listing 2.4: Multi-locus allele-level HLA genotype and microsatellite genotype data with header information

labcode	ethnic	complex					
USAFEL	****	0					
populat	id	drb1_1	drb1_2	dqb1_1	dqb1_2	d6s2222_1	d6s2222_2
UchiTelle	HJK_2	01	03:01	02:01	05:01	249	249
UchiTelle	HJK_1	03:01	03:01	02:01	02:01	249	249
UchiTelle	HJK_3	01	03:01	02:01	05:01	249	249
UchiTelle	HJK_4	01	03:01	02:01	05:01	249	249
UchiTelle	MYU_2	02	04:01	03:02	06:02	247	249
UchiTelle	MYU_1	03:01	03:01	02:01	02:01	247	249
UchiTelle	MYU_3	03:01	04:01	02:01	03:02	249	249
UchiTelle	MYU_4	03:01	04:01	02:01	03:02	247	249

This example mixes different kinds of data: HLA allele data (from DRB1 and DQB1 loci) with microsatellite data (locus D6S2222).

Listing 2.5: Sequence genotype data with header information

labcode	file				
BLOGGS	C_New				
popName	ID	TGFB1cdn10(1)	TGFB1cdn10(2)	TGFBhap1(1)	TGFBhap1(2)
Urboro	XQ-1	C	T	CG	TG
Urboro	XQ-2	C	C	CG	CG
Urboro	XQ-5	C	T	CG	TG
Urboro	XQ-21	C	T	CG	TG
Urboro	XQ-7	C	T	CG	TG
Urboro	XQ-20	C	T	CG	TG
Urboro	XQ-6	T	T	TG	TG
Urboro	XQ-8	C	T	CG	TG
Urboro	XQ-9	T	T	TG	TG
Urboro	XQ-10	C	T	CG	TG

This example includes nucleotide sequence data: the TGFB1CDN10 locus consists of one nucleotide, the TGFBhap1 locus is actually haplotype data, but PyPop simply treats each combination as a separate “allele” for subsequent analysis.

Listing 2.6: Allele count data

populat	method	ethnic	country	latit	longit
UchiTelle	PCR-SSO	Klingon	QZ	052.81N	100.25E
dqa1	count				
01:01	31				
01:02	37				
01:03	17				
02:01	21				
03:01	32				
04:01	9				
05:01	35				

PyPop can also process allele count data. However, you cannot mix allele count data and genotype data together in the one file.

Note

Currently each `.pop` file can only contain allele count data for *one locus*. In order to process multiple loci for one population you must create a separate `.pop` for each locus.

Missing data

Untyped or missing data may be represented in a variety of ways. The default value for untyped or missing data is a series of four asterisks (****) as specified by the `config.ini`. You may not “represent” untyped data by leaving a column blank, nor may you represent a homozygote by leaving the second column blank. All cells for which you have data must include data, and all cells for which you do not have data must also be filled in, using a missing data value.

For individuals who were not typed at all loci, the data in loci for which they are typed will be used on all single-locus analyses for that individual and locus, so that you see the value of the number of individuals (n) vary from locus to locus in the output. These individuals’ data will also be used for multi-locus analyses. Only the loci that contain no missing data will be included in any multi-locus analysis.

If an individual is only partially typed at a locus, it will be treated as if it were completely untyped, and data for that individual for that locus will be dropped from ALL analyses.

Warning

- Do not leave trailing blank lines at the end of your data file, as this currently causes PyPop to terminate with an error message that takes experience to diagnose.
- For haplotype estimation and linkage disequilibrium calculations (i.e., the emhaplofreq part of the program) you are currently restricted to a maximum of seven loci per haplotype request. For haplotype estimation there is a limit of 5000 for the number of individuals (n) ³

2.5 The configuration file

The sets of population genetic analyses that are run on your population data file and the manner in which the data file is interpreted by PyPop is controlled by a configuration file, the default name for which is `config.ini`. This is another plain text file consisting of comments (which are lines that start with a semi-colon), sections (which are lines with labels in square brackets), and options (which are lines specifying settings relevant to that section in the `option=value` format).

Note

If any option runs over one line (such as `validSampleFields`) then the second and subsequent lines must be indented by exactly **one space**.

A simple configuration file

Here we present a simple `.ini` file corresponding to [Listing 2.1](#) (Note comment lines have been omitted in the above example for clarity). After this we review the sections that are highlighted in the example below, starting with general settings, followed by *how to specify data formats* and then *Analysis options*.

Descriptions of more advanced options for the previously described sections and additional filtering sections are contained in *Advanced options* and *Advanced filtering sections*, respectively.

Listing 2.7: Minimal config.ini file

```
[General]
debug=0

[ParseGenotypeFile]
untypedAllele=***
alleleDesignator=*
validSampleFields=*a_1
*a_2
*c_1
*c_2
*b_1
*b_2

[HardyWeinberg]
lumpBelow=5

[HardyWeinbergGuoThompson]
dememorizationSteps=2000
samplingNum=1000
samplingSize=1000

[HomozygosityEWSlatkinExact]
numReplicates=10000

[Emhaplofreq]
allPairwiseLD=1
allPairwiseLDWithPermu=0
;;numPermuInitCond=5
```

[General] settings

This section contains variables that control the overall behavior of PyPop. Additional variables are described in [\[General\] advanced options](#).

- debug=0

This setting enables verbose debugging messages. Setting it to 1 will generate output that can be useful in diagnosing problems. PyPop developers may ask you to enable it when reporting on problems on the issue tracker.

Specifying data formats

There are two possible formats: [ParseGenotypeFile] and [ParseAlleleCountFile]

[ParseGenotypeFile]

If your data is genotype data, you will want a section labeled: [ParseGenotypeFile] (as shown in the [Minimal config.ini file](#)).

- alleleDesignator

This option is used to tell PyPop what is allele data and what isn't. You must use this symbol in :ref:`validSampleFields` option. The default is * In general, you won't need to change it. **[Default: *]**

- untypedAllele

This option is used to tell PyPop what symbol you have used in your data files to represent untyped or unknown data fields. These fields MAY NOT BE LEFT BLANK. You must use something consistent that cannot be confused with real data here. **[Default: ****]**

- validSampleFields

This option should contain the names of the loci immediately preceding your genotype data (if it has three header lines, this information will be on the third line, otherwise it will be the first line of the file). **[There is no default, this option must always be present]**

The format is as follows, for each sample field (which may either be an identifying field for the sample such as populat, or contain allele data) create a new line where:

- The first line (validSampleFields=) consists of the name of your sample field (if it contains allele data, the name of the field should be preceded by the character designated in the alleleDesignator option above).
- All subsequent lines after the first *must* be preceded by *one space* (again if it contains allele data, the name of the field should be preceded by the character designated in the alleleDesignator option above).

Here is an example:

```
validSampleFields=*a_1
*a_2
*c_1
*c_2
*b_1
*b_2    # Note initial space at start of line.
```

Here is example that includes identifying (non-allele data) information such as sample id (id) and population name (populat):

```
validSampleFields=populat
id
*a_1
*a_2
*c_1
*c_2
```

(continues on next page)

(continued from previous page)

```
*b_1
*b_2
```

[ParseAlleleCountFile]

If your data is not genotype data, but rather, data of the allele-name count format, then you will want to use the [ParseAlleleCountFile] section **INSTEAD** of the [ParseGenotypeFile] section. The alleleDesignator and untypedAllele options work identically to that described for [ParseGenotypeFile].

- validSampleFields

This option should contain either a single locus name or a colon-separated list of all loci that will be in the data files you intend to analyze using a specific .ini file. The colon-separated list allows you to avoid changing the .ini file when running over a collection of data files containing different loci. e.g.,

```
validSampleFields=A:B:C:DQA1:DQB1:DRB1:DPB1:DPA1
count
```

Note that each .pop file must contain only one locus (see *the note* in Listing 2.6). Listing multiple loci simply permits the same .ini file to be reused for each data file.

Analysis options

These sections describe the primary analysis options that can be enabled for PyPop, as they are used in the simple example, above.

[HardyWeinberg]

Hardy-Weinberg analysis is enabled by the presence of this section.

- lumpBelow

This option value represents a cut-off value. Alleles with an expected value equal to or less than lumpBelow will be lumped together into a single category for the purpose of calculating the degrees of freedom and overall p-value for the chi-squared Hardy-Weinberg test.

[HardyWeinbergGuoThompson]

When this section is present, an implementation of the Hardy-Weinberg exact test is run using the original Guo and Thompson (1992) code, using a Monte-Carlo Markov chain (MCMC). In addition, two measures (Chen and Diff) of the goodness of fit of individual genotypes are reported under this option (Chen *et al.*, 1999). By default this section is not enabled. This is a different implementation to the **Arlequin** version listed in *Advanced options*, below.

- dememorizationSteps

Number of steps of to “burn-in” the Markov chain before statistics are collected. **[Default: 2000]**

- samplingNum

Number of Markov chain samples **[Default: 1000]**.

- samplingSize

Markov chain sample size **[Default: 1000]**.

Note that the **total** number of steps in the Monte-Carlo Markov chain is the product of samplingNum and samplingSize, so the default values described above would contain 1,000,000 (= 1000 x 1000) steps in the MCMC chain.

The default values for options described above have proved to be optimal for us and if the options are not provided these defaults will be used. If you change the values and have problems, please let us **know**.

[HomozygosityEWSlatkinExact]

The presence of this section enables Slatkin's (1994) implementation of the Ewens-Watterson exact test of neutrality.

- numReplicates

The default values have proved to be optimal for us. There is no reason to change them unless you are particularly curious. If you change the default values and have problems, please let us know.

[Emhaplofreq]

The presence of this section enables haplotype frequency estimation and calculation of linkage disequilibrium (LD) measures. *Please note that PyPop assumes that the genotype data is **unphased** when estimating haplotype frequencies and LD measures.*

- lociToEstHaplo

In this option you can list the multi-locus haplotypes for which you wish the program to estimate and to calculate the LD. It should be a comma-separated list of colon-joined loci. e.g.,

```
lociToEstHaplo=a:b:drb1,a:b:c,drb1:dqa1:dpb1,drb1:dqb1:dpb1
```

- allPairwiseLD

Set this to 1 (one) if you want the program to calculate all pairwise LD for your data, otherwise set this to 0 (zero).

- allPairwiseLDWithPermu

Set this to a positive integer greater than 1 if you need to determine the significance of the pairwise LD measures in the previous section. The number you use is the number of permutations that will be run to ascertain the significance (this should be at least 1000 or greater). (Note this is done via permutation testing performed after the pairwise LD test for all pairs of loci. Note also that this test can take *DAYS* if your data is highly polymorphic.)

- numPermuInitCond

Set this to change the number of initial conditions used per permutation. **[Default: 5]**. (Note: this parameter is only used if allPairwiseLDWithPermu is set and nonzero).

Advanced options

The following section describes additional options to previously described sections. Most of the time these options can be omitted and PyPop will choose defaults, however these advanced options do offer greater control over the application. In particular, customization will be required for data that has sample identifiers as in Listing 2.2 or header data block as in Listing 2.3 and both validSampleFields (described above) and validPopFields (described below) will need to be modified.

Deprecated since version 1.0.0

Deprecated since version 1.0.0: The sections [Arlequin] and [HardyWeinbergGuoThompsonArlequin] related to the **Arlequin** program as they are currently unmaintained.

[General] advanced options

- txtOutFilename and xmlOutFilename

If you wish to specify a particular name for the output file, which you want to remain identical over several runs, you can set these two items to particular values. The default is to have the program select the output filename, which can be controlled by the next variable. **[Default: not used]**

- outFilePrefixType

This option can either be omitted entirely (in which case the default will be filename) or be set in several ways. The default is set as filename, which will result in three output files named original-filename-minus-suffix-out.xml,

original-filename-minus-suffix-out.txt, and original-filename-minus-suffix-filter.xml. **[Default: filename]**

If you set the value to date instead of filename, you'll get the date incorporated in the filename as follows: original-filename-minus-suffix-YYYY-nn-dd-HH-MM-SS-out.xml,txt. e.g., USAFEL-UchiTelle-2003-09-21-01-29-35-out.xml (where Y, n, d, H, M, S refer to year, month, day, hour, minute and second, respectively).

- xslFilename

This option specifies where to find the XSLT file to use for transforming PyPop's xml output into human-readable form. Most users will not normally need to set this option, and the default is the system-installed text.xsl file.

[ParseGenotypeFile] advanced options

- fieldPairDesignator

This option allows you to override the coding for the headers for each pair of alleles at each locus; it must match the entry in the config file under validSampleFields and the entries in your population data file. If you want to use something other than _1 and _2, change this option, for instance, to use letters and parentheses, change it as follows: fieldPairDesignator=(a):(b) **[Default: _1:_2]**

- popNameDesignator

There is a special designator to mark the population name field, which is usually the first field in the data block. **[Default: +]**

If you are analyzing data that contains a population name for each sample, then the first entry in your validSampleFields section should have a prefixed +, as below:

```
validSampleFields=+populat
*a_1
*a_2
...
```

- validPopFields

If you are analyzing data with an initial two line population header block information as in *Multi-locus allele-level HLA genotype data with sample and header information*, then you will need to set this option. In this case, it should contain the field names in the first line of the header information of your file. **[Default: required when a population data-block is present in data file]**, e.g.:

```
validPopFields=labcode
method
ethnic
country
latit
longit
```

[Emhaplofreq] advanced options

- permutationPrintFlag

Warning

If permutationPrintFlag is enabled it can *drastically* increase the size of the output XML file on the order of the product of the number of possible pairwise comparisons and permutations. Machines with lower RAM and disk space may have difficulty coping with this.

Determines whether the likelihood ratio for each permutation will be logged to the XML output file, this is disabled by default. **[Default: 0 (i.e. OFF)]**.

Deprecated since version 1.0.0

Deprecated since version 1.0.0: currently unmaintained and untested.

[Arlequin] extra section

This section sets characteristics of the **Arlequin** application if it has been installed (it must be installed separately from PyPop as we cannot distribute it). The options in this section are only used when a test requiring **Arlequin**, such as it's implementation of Guo and Thompson's (1992) Hardy-Weinberg exact test is invoked (see below).

- **arlequinExec**

This option specifies where to find the **Arlequin** executable on your system. The default assumes it is on your system path. **[Default: arlecore.exe]**

[HardyWeinbergGuoThompsonArlequin] extra section

When this section is present, **Arlequin**'s implementation of the Hardy-Weinberg exact test is run, using a Monte-Carlo Markov Chain implementation. By default this section is not enabled.

- **markovChainStepsHW**

Length of steps in the Markov chain **[Default: 2500000]**.

- **markovChainDememorisationStepsHW**

Number of steps of to "burn-in" the Markov chain before statistics are collected. **[Default: 5000]**

The default values for options described above have proved to be optimal for us and if the options are not provided these defaults will be used. If you change the values and have problems, please let us **know**.

Advanced filtering sections

This section describes additional advanced sections that can be used for applying filtering to both the input and output of the population data.

[Filters] extra section

When this section is present, it allows you to specify successive filters to the data.

- **filtersToApply**

Here you specify which filters you want applied to the data and the order in which you want them applied. The format is ``FILTER[:FILTER]*`, i.e. a series of colon-delimited filters. Currently there are four predefined filter: **AnthonyNolan**, **Sequence**, **DigitBinning**, and **CustomBinning**. If you specify one or more of these filters, you will get the default behavior of the filter. If you wish to modify the default behavior, you should add a section with the same name as the specified filter(s). See next section for more on this. Please note that, while you are allowed to specify any ordering for the filters, some orderings may not make sense. For example, the ordering **Sequence:AnthonyNolan** would not make sense (because as far as PyPop is concerned, your alleles are now amino acid residues.) However, the reverse ordering, **AnthonyNolan:Sequence**, would be logical and perhaps even advisable.

- **makeNewPopFile**

This option creates intermediate population files (in the **.pop** format) before or after any filtering step. This allows the user to save and inspect the output, and for running the output through another PyPop process with potentially different parameters. The format of the argument is: `{all-loci|separate-loci}:<NUMBER>`. Where **separate-loci** generates separate **.pop** files for each of the loci, and **all-loci** generates a *single* **.pop** file with all loci. **<NUMBER>** is an integer representing the step in the filtering process at which the output files should be generated. So **0** represents the

original input data, i.e. the step before the filtering runs, whereas 1 would be the data after the first filter is run, and so on.

For example, with the following stanza in an `.ini` file

```
[Filters]
filtersToApply=Sequence
makeNewPopFile=all-loci:1

[Sequence]
directory=tests/data/anthonymolan/msf-2.18.0/
```

applied to an input file `MyPopulation.pop` with a single HLA locus A :

```
A_1      A_2
0101     0201
0210     03012
0101     0218
```

This would apply the Sequence filter options , generating a new file `MyPopulation-filtered.pop` where the original HLA A locus is translated into columns where each new locus would consist of the individual polymorphic amino acid residue position within HLA A, and then generate output files for that data at that point. For example, the first two columns showing the first two polymorphic positions at residues 9 (new locus `A_9`) and 44 (new locus `A_42`), the `NewPopulation-filtered.pop` might look something like this:

```
A_9_1  A_9_2  A_44_1  A_44_2  ...
F      F      K      R      ...
Y      F      R      R      ...
F      F      K      R      ...
```

[AnthonyNolan] filter section

This section is *only* useful for HLA data. Like all filter sections, it will only be used if present in the `filtersToApply` line specified above. If so enabled, your data will be filtered through the Anthony Nolan database of known HLA allele names before processing. The data files this filter relies on are *not* currently distributed with PyPop within the binary installable packages (“wheels”), but can be obtained as described in the `directory` section, below, or the [IMGT ftp site](#)⁴.

Warning

The current implementation of the AnthonyNolan filter only works with data in old-style (pre-2010) HLA nomenclature. However, the Sequence filter has been ported to use new nomenclature, but only by specifying a recent version of the Multiple Sequence Format ([MSF](#)⁵) files in the `remoteMSF` option

Invocation of this filter will produce a `<POPPFILE>-filter.xml` file output showing what was resolved and what could not be resolved.

- `alleleFileFormat`

This options specifies which of the formats the Anthony Nolan allele data will be used. The option can be set to either `txt` (for the plain free text format) or `msf` (for the [Multiple Sequence Format](#)⁶) [**Default: msf**]

- `directory` or `remoteMSF`

This section expects there to exist *either* `directory` option (which points to a local disk directory) or the `remoteMSF` option (which downloads MSF files on-the-fly) (but not both). **There are no defaults.**

To save space, sequence files are not distributed as part of the wheels, but are either downloaded on-the-fly, or (in the case of the versions using old sequence nomenclature) are distributed as part of the source distribution. They are then incorporated into the unit tests.

- **directory**: Specifies the path to the root of the sequence files. It can be either relative or absolute. If it is relative, the path will be resolved relative to the current working directory.

A version (2.18.0) of the sequence files that use old (pre-2010) nomenclature can be found in the GitHub repo here (you can either clone the repo or download the files manually):

* **txt** files: [tests/data/anthonymolan/HIG-seq-pep-text](#)⁷

* **msf** files: [tests/data/anthonymolan/msf-2.18.0](#)⁸

- **remoteMSF**: Specifies the version (tag) of the remote msf directory in the [IMGT-HLA GitHub repo](#)⁹. If present, the remote MSF files for the specified version will be downloaded on-demand, and cached for later reuse. For example:

```
[Sequence]
remoteMSF=3.59.0-alpha
```

would download the msf directory with v3.59.0-alpha tag, i.e.: <https://github.com/ANHIG/IMGTHLA/tree/v3.59.0-alpha/msf>, which at the time of writing, was the most recent release.

Note that this option is only available for MSF files, there is no equivalent for other file types.

- **preserve-ambiguous**

The default behavior of the **AnthonyNolan** filter is to ignore allele ambiguity (“slash”) notation. This notation, common in the literature, looks like: 010101/0102/010301. The default behavior will simply truncate this to 0101. If you want to preserve the notation, set the option to 1. This will result in a filtered allele “name” of 0101/0102/0103 in the above hypothetical example. **[Default: 0]**.

- **preserve-unknown**

The default behavior of the **AnthonyNolan** filter is to replace unknown alleles with the **untypedAllele** designator. If you want the filter to keep allele names it does not recognize, set the option to 1. **[Default: 0]**.

- **preserve-lowres**

This option is similar to **preserve-unknown**, but only applies to lowres alleles. If set to 1, PyPop will keep allele names that are shorter than the default allele name length, usually 4 digits long. But if the **preserve-unknown** flag is set, this one has no effect, because all unknown alleles are preserved. **[Default: 0]**.

[Sequence] filter section

This section allows configuration of the sequence filter. Like all filter sections, it will only be used if present in the **filtersToApply** line specified above. If so enabled, your allele names will be translated into sequences, and all ensuing analyses will consider each position in the sequence to be a distinct locus. This filter makes use of the same msf format alignment files as used above in the **AnthonyNolan** filter. **It does not work with the txt format alignment files.**

- **sequenceFileSuffix**

Determines the files that will be examined in order to read in a sequence for each allele. (ie, if the file for locus A is **A_prot.msf**, the value would be **_prot** whereas if you wanted to use the nucleotide sequence files, you might use **_nuc**.) **[Default: _prot]**.

- **directory or remoteMSF**

Specifies the either path to the root of the sequence files (**directory**), or the remote MSF version (**remoteMSF**) in the same manner as in the **AnthonyNolan** section, above.

[DigitBinning] filter section

This section allows configuration of the DigitBinning filter. Like all filter sections, it will be used if present in the `filtersToApply` line specified above. If so enabled, your allele names will be truncated after the *n*th digit.

- `binningDigits`

An integer that specifies how many digits to keep after the truncation. [Default: 4].

[CustomBinning] filter section

This section allows configuration of the CustomBinning filter. Like all filter sections, it will only be used if present in the `filtersToApply` line specified above.

You can provide a set of custom rules for replacing allele names. Allele names should be separated by / marks. This filter matches any allele names that are exactly the same as the ones you list here, and will also find “close matches” (but only if there are no exact matches.). Here is an example:

```
A=01/02/03
04/05/03:06
!06/12:01/13:01
!07/08:05
```

In the example above, `A*03` alleles will match to `01/02/03`, except for `A*03:06`, which will match to `04/05/03:06`. In the output file, `A*03:06` will be replaced with `04/05/03:06` and other `A*03` alleles will be replaced with `01/02/03`. If you place a `!` mark in front of the first allele name, that first name will be used as the “new name” for the binned group (for example, `A*08:05` will be called `07` in the custom-binned data.) Note that the space at the beginning of the lines (following the first line of each locus) is important. The above rules are just dummy examples, provided to illustrate how the filter works. PyPop is distributed with a biologically relevant set of CustomBinning rules that have been compiled from several (Cano *et al.*, 2007, Mack *et al.*, 2007) sources¹⁰

2.6 PyPop API examples

Here is a minimal example of using the API (documented in the *PyPop API Reference*) directly in your own Python program. This program reads a short `.pop data file` consisting of one locus with seven individuals, and rather than creating a *configuration file*, we create a configuration object with the file format details and selecting a single [HardyWeinberg] analysis. It then performs the equivalent of the *popmeta script* and generates output TSV files.

This is done by using `PyPop.Main.Main` to generate the initial XML output, and then using `PyPop.Meta.Meta` to process this XML to generate `.tsv` file output suitable for further analysis. Here is the process, step-by-step:

We first create the `configparser.ConfigParser`¹¹ instance (note that we specify the `untypedAllele` and `alleleDesignators` explicitly, even though they are the same as defaults, they must always match the input file):

```
>>> from configparser import ConfigParser
>>> config = ConfigParser()
>>> config.read_dict({"General": {"debug": "0"},
...   "ParseGenotypeFile": {"untypedAllele": "****",
...   "alleleDesignator": "*",
...   "validSampleFields": "*a_1\n*a_2"},
...   "HardyWeinberg": {"lumpBelow": "5"}}})
>>>
```

Next, for testing purposes, we create a `.pop` text file (note the tab-spaces inline). (You could replace this with your own input file, or generate `pop_contents` from an existing data structure in your program):

```
>>> pop_contents = '''a_1\ta_2
... ****\t****
... 01:01\t02:01
... 02:10\t03:01:02
```

(continues on next page)

(continued from previous page)

```
... 01:01\t02:18
... 25:01\t02:01
... 02:10\t32:04
... 03:01:02\t32:04'''
>>> with open("my.pop", "w") as f:
...     _ = f.write(pop_contents)
... 
```

Then we setup the XSLT transformation file that will generate a plain text output (`my-pop.txt`), along with the default XML output file: `my-out.xml`.¹²

Warning

Use of `importlib.resources.files()`¹³ requires Python version 3.9 or later.

```
>>> from PyPop.xslt import ns          # need for XSLT extensions
>>> from importlib.resources import files # get location from installation
>>> xslFilename = str(files("PyPop.xslt") / "text.xsl")
```

Now we can create the `PyPop.Main.Main` instance, using the config object to analyze the data in `my.pop` :

```
>>> from PyPop.Main import Main
>>> application = Main(
...     config=config,
...     fileName="my.pop",
...     version="fake",
...     xslFilename=xslFilename,
... )
...
LOG: Data file has no header data block
```

We can query the `Main` instance to get the name of output XML file: `my-out.xml`

```
>>> application.getXmlOutPath()
'my-out.xml'
```

Lastly, we pass this file to the `PyPop.Meta.Meta` to generate output TSV files (as described in *Using popmeta to aggregate results*):

```
>>> outXML = application.getXmlOutPath()
>>> from PyPop.Meta import Meta
>>> _ = Meta (TSV_output=True, xml_files=[outXML])
./1-locus-hardyweinberg.tsv
./1-locus-summary.tsv
./1-locus-allele.tsv
./1-locus-genotype.tsv
```

These `.tsv` files could then be read into another data structure (e.g. a `pandas dataframe`¹⁴) for further analysis.

Notes

1. <https://github.com/alexlancaster/pypop/blob/main/tests/data/minimal.ini>
2. <https://github.com/alexlancaster/pypop/blob/main/tests/data/USAFEL-UchiTelle-small.pop>
3. These hardcoded numbers can be changed if you obtain the source code yourself and change the appropriate `#define emhaplofreq.h` and recompile the program.
4. <ftp://ftp.ebi.ac.uk/pub/databases/imgt/mhc/hla/>
5. <https://tcoffee.org/Documentation/misc/Format%20Documentation.htm>
6. <https://www.ebi.ac.uk/ipd/imgt/hla/download/>
7. <https://github.com/alexlancaster/pypop/tree/main/tests/data/anthonymolan/HIG-seq-pep-text>

8. <https://github.com/alexlanaster/pypop/tree/main/tests/data/anthonymolan/msf-2.18.0>
9. <https://github.com/ANHIG/IMGTHLA/>
10. The Anthony Nolan list of deleted allele names (https://github.com/ANHIG/IMGTHLA/blob/Latest/Deleted/protect_alleles.txt); and the Ambiguous Allele Combinations, release 2.18.0 (<https://www.ebi.ac.uk/ipd/imgt/hla/ambiguity>).
11. <https://docs.python.org/3/library/configparser.html#configparser.ConfigParser>
12. The expectation that `PyPop.Main.Main` will generate a plain text output (and thus setting up an XSLT file for that transformation) will be removed in a future release, allowing for further simplification of this example, by eliminating the requirement that `xslFilename` is passed to the `Main` constructor.
13. <https://docs.python.org/3/library/importlib.resources.html#importlib.resources.files>
14. <https://pandas.pydata.org>

INTERPRETING PYPOP OUTPUT

As mentioned in *What happens when you run PyPop?*, The XML file is the primary output created by PyPop and contains the complete set of results. The text output, generated from the XML file via XSLT, contains a human-readable summary of the XML results. Below we discuss the output contained in this text file.

Warning

The text output we discuss below is strictly intended for consumption by an end-user, or incorporation into a paper. You should never extract information from this text file output to perform any downstream analyses (e.g. don't take the values in the output and paste them into another program). This is because the results are rounded for space, and you may lose a lot of precision if you use any floating-point output in further analyses.

You should use the *TSV outputs* for maximum precision (which, in turn, are derived from the raw XML output) for such analyses.

3.1 Population summary

A Population Summary is generated for each dataset analyzed. This summary provides basic demographic information and summarizes information about the sample size.

Sample output:

```
Population Summary
=====
Population Name: UchiTelle
  Lab code: USAFEL
  Typing method: 12th Workshop SSOP
  Ethnicity: Telle
  Continent: NW Asia
Collection site: Targen Village
  Latitude: 41 deg 12 min N
  Longitude: 94 deg 7 min E

Population Totals
=====
Sample Size (n): 47
Allele Count (2n): 94
Total Loci in file: 9
Total Loci with data: 8
```

3.2 Single locus analyses

Basic allele count information

Information relevant to individual loci is reported. Sample size and allele counts will differ among loci if not all individuals were typed at each locus. Untyped individuals are those for which one or two alleles were not reported. The alleles are listed in descending frequency (and count) in the left hand column, and are sorted numerically in the right column. The number of distinct alleles k is reported.

```
I. Single Locus Analyses
=====

1. Locus: A
-----

1.1. Allele Counts [A]
-----
Untyped individuals: 2
Sample Size (n): 45
Allele Count (2n): 90
Distinct alleles (k): 10

Counts ordered by frequency | Counts ordered by name
Name      Frequency (Count) | Name      Frequency (Count)
02:01     0.21111  19    | 01:01     0.13333  12
03:01     0.15556  14    | 02:01     0.21111  19
01:01     0.13333  12    | 02:10     0.10000  9
25:01     0.12222  11    | 02:18     0.10000  9
02:10     0.10000  9     | 03:01     0.15556  14
02:18     0.10000  9     | 25:01     0.12222  11
32:04     0.08889  8     | 32:04     0.08889  8
69:01     0.04444  4     | 68:14     0.03333  3
68:14     0.03333  3     | 69:01     0.04444  4
74:03     0.01111  1     | 74:03     0.01111  1
Total     1.00000  90    | Total     1.00000  90
```

In the cases where there is no information for a locus, a message is displayed indicating lack of data.

Sample output:

```
4. Locus: DRA
-----
No data for this locus!
```

Chi-square test for deviation from Hardy-Weinberg proportions (HWP).

For each locus, the observed genotype counts are compared to those expected under Hardy Weinberg proportions (HWP). A triangular matrix reports observed and expected genotype counts. If the matrix is more than 80 characters, the output is split into different sections. Each cell contains the observed and expected number for a given genotype in the format observed/expected.

```
6.2. HardyWeinberg [DQA1]
-----
Table of genotypes, format of each cell is: observed/expected.

02:01 8/5.1
03:01 4/4.0 1/0.8
04:01 3/6.9 1/2.7 6/2.3
05:01 8/9.9 5/3.8 5/6.7 6/4.8
      02:01 03:01 04:01 05:01
                        [Cols: 1 to 4]
```

The values in this matrix are used to test hypotheses of deviation from HWP. The output also includes the chi-square statistic, the number of degrees of freedom and associated p -value for a number of classes of genotypes and is summarized in the

following table:

	Observed	Expected	Chi-square	DoF	p-value
Common	N/A	N/A	4.65	1	0.0310*
Lumped genotypes	N/A	N/A	1.17	1	0.2797
Common + lumped	N/A	N/A	5.82	1	0.0158*
All homozygotes	21	13.01	4.91	1	0.0268*
All heterozygotes	26	33.99	1.88	1	0.1706
Common heterozygotes by allele					
02:01	15	20.78	1.61		0.2050
03:01	10	10.47	0.02		0.8850
04:01	9	16.31	3.28		0.0703
05:01	18	20.43	0.29		0.5915
Common genotypes					
02:01+02:01	8	5.11	1.63		0.2014
02:01+04:01	3	6.93	2.23		0.1358
02:01+05:01	8	9.89	0.36		0.5472
04:01+05:01	5	6.70	0.43		0.5109
Total	24	28.63			

- **Common.**

The result for goodness of fit to HWP using only the genotypes with at least `lumpBelow` expected counts (the common genotypes) (in the output shown throughout this example `lumpBelow` is equal to 5).

If the dataset contains no genotypes with expected counts equal or greater than `lumpBelow`, then there are no common genotypes and the following message is reported:

No common genotypes; chi-square cannot be calculated

The analysis of common genotypes may lead to a situation where there are fewer classes (genotypes) than allele frequencies to estimate. This means that the analysis cannot be performed (degrees of freedom < 1). In such a case the following message is reported, explaining why the analysis could not be performed:

Too many parameters for chi-square test.

To obviate this as much as possible, only alleles which occur in common genotypes are used in the calculation of degrees of freedom.

- **Lumped genotypes.**

The result for goodness of fit to HWP for the pooled set of genotypes that individually have less than `lumpBelow` expected counts.

The pooling procedure is designed to avoid carrying out the chi-square goodness of fit test in cases where there are low expected counts, which could lead to spurious rejection of HWP. However, in certain cases it may not be possible to carry out this pooling approach. The interpretation of results based on lumped genotypes will depend on the particular genotypes that are combined in this class.

If the sum of expected counts in the lumped class does not add up to `lumpBelow`, then the test for the lumped genotypes cannot be calculated and the following message is reported:

The total number of expected genotypes is less than 5

This may be remedied by combining rare alleles and recalculating overall chi-square value and degrees of freedom. (This would require appropriate manipulation of the data set by hand and is not a feature of PyPop).

- **Common + lumped.**

The result for goodness of fit to HWP for both the common and the lumped genotypes.

- **All homozygotes.**

The result for goodness of fit to HWP for the pooled set of homozygous genotypes.

- **All heterozygotes.**

The result for goodness of fit to HWP for the pooled set of heterozygous genotypes.

- **Common heterozygotes.**

The common heterozygotes by allele section summarizes the observed and expected number of counts of all heterozygotes carrying a specific allele with expected value `GE lumpBelow`.

- **Common genotypes.**

The common genotypes by genotype section lists observed, expected, chi-square and p -values for all observed genotypes with expected values `GE lumpBelow`.

Exact test for deviation from HWP

If enabled in the configuration file, the exact test for deviations from HWP will be output. The exact test uses the method of Guo and Thompson (1992). The p -value provided describes how probable the observed set of genotypes is, with respect to a large sample of other genotypic configurations (conditioned on the same allele frequencies and $2n$). p -values lower than 0.05 can be interpreted as evidence that the sample does not fit HWP. In addition, those individual genotypes deviating significantly ($p < 0.05$) from expected HWP as computed with the Chen and “diff” measures are reported.

There are two implementations for this test, the first using the gthwe implementation originally due to Guo & Thompson, but modified by John Chen, the second being Arlequin’s (Excoffier and Lischer, 2010, Schneider *et al.*, 2000) implementation.

6.3. Guo and Thompson HardyWeinberg output [DQA1]

```
-----
Total steps in MCMC: 1000000
Dememorization steps: 2000
Number of Markov chain samples: 1000
Markov chain sample size: 1000
Std. error: 0.0009431
p-value (overall): 0.0537
```

6.4. Guo and Thompson HardyWeinberg output(Arlequin's implementation) [DQA1]

```
-----
Observed heterozygosity: 0.553190
Expected heterozygosity: 0.763900
Std. deviation: 0.000630
Dememorization steps: 100172
p-value: 0.0518
```

Note that in the Arlequin implementation, the output is slightly different, and the only directly comparable value between the two implementation is the p -value. These p -values may be slightly different, but should agree to within one significant figure.

The Ewens-Watterson homozygosity test of neutrality

For each locus, we implement the Ewens-Watterson homozygosity test of neutrality (Ewens, 1972, Watterson, 1978). We use the term *observed homozygosity* to denote the homozygosity statistic (F), computed as the sum of the squared allele frequencies. This value is compared to the *expected homozygosity* which is computed by simulation under neutrality/equilibrium expectations, for the same sample size ($2n$) and number of unique alleles (k). Note that the homozygosity F statistic, $F = \sum_{i=1}^k p_i^2$, is often referred to as the *expected homozygosity* (with *expectation* referring to HWP) to distinguish it from the observed proportion of homozygotes. We avoid referring to the observed F statistic as the “*observed expected homozygosity*” (to simplify and hopefully avoid confusion) since the homozygosity test of neutrality is concerned with comparisons of

observed results to expectations under neutrality. Both the *observed* statistic (based on the actual data) and *expected* statistic (based on simulations under neutrality) used in this test are computed as the sum of the squared allele frequencies.

The *normalized deviate of the homozygosity* (F_{nd}) is the difference between the *observed homozygosity* and *expected homozygosity*, divided by the square root of the variance of the expected homozygosity (also obtained by simulations; (Salamon *et al.*, 1999)). Significant negative normalized deviates imply *observed homozygosity* values lower than *expected homozygosity*, in the direction of balancing selection. Significant positive values are in the direction of directional selection.

The p -value in the last row of the output is the probability of obtaining a homozygosity F statistic under neutral evolution that is less than or equal to the observed F statistic. It is computed based on the null distribution of homozygosity F values simulated under neutrality/equilibrium conditions for the same sample size ($2n$) and number of unique alleles (k). For a one-tailed test of the null hypothesis of neutrality against the alternative of balancing selection, p -values less than 0.05 are considered significant at the 0.05 level. For a two-tailed test against the alternative of either balancing or directional selection, p -values less than 0.025 or greater than 0.975 can be considered significant at the 0.05 level.

The standard implementation of the test uses a Monte-Carlo implementation of the exact test written by Slatkin (1994, 1996). A Markov-chain Monte Carlo method is used to obtain the null distribution of the homozygosity statistic under neutrality. The reported p -values are one-tailed (against the alternative of balancing selection), but can be interpreted for a two-tailed test by considering either extreme of the distribution (< 0.025 or > 0.975) at the 0.05 level.

1.6. Slatkin's implementation of EW homozygosity test of neutrality [A]

Observed F: 0.1326, Expected F: 0.2654, Variance in F: 0.0083
Normalized deviate of F (Fnd): -1.4603, p-value of F: 0.0029**

Warning

The version of this test based on tables of simulated percentiles of the Ewens-Watterson statistics is now disabled by default and its use is deprecated in preference to the Slatkin exact test described above, however some older PyPop runs may include output, so it is documented here for completeness. This version differs from the Monte-Carlo Markov Chain version described above in that the data is simulated under neutrality to obtain the required statistics.

1.4. Ewens-Watterson homozygosity test of neutrality [A]

Observed F: 0.1326, Expected F: 0.2651, Normalized deviate (Fnd): -1.4506
p-value range: 0.0000 < p <= 0.0100 *

3.3 Multi-locus analyses

Haplotype frequencies are estimated using the iterative Expectation-Maximization (EM) algorithm (Dempster *et al.*, 1977, Excoffier and Slatkin, 1995). Multiple starting conditions are used to minimize the possibility of local maxima being reached by the EM iterations. The haplotype frequencies reported are those that correspond to the highest logarithm of the sample likelihood found over the different starting conditions and are labeled as the maximum likelihood estimates (MLE).

The output provides the names of loci for which haplotype frequencies were estimated, the number of individual genotypes in the dataset (*before-filtering*), the number of genotypes that have data for all loci for which haplotype estimation will be performed (*after-filtering*), the number of unique phenotypes (unphased genotypes), the number of unique phased genotypes, the total number of possible haplotypes that are compatible with the genotypic data (many of these will have an estimated frequency of zero), and the log-likelihood of the observed genotypes under the assumption of linkage equilibrium.

All pairwise LD

A series of linkage disequilibrium (LD) measures are provided for each pair of loci, as shown in the sample output below.

II. Multi-locus Analyses

=====

(continues on next page)

(continued from previous page)

Haplotype/ linkage disequilibrium (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
A:B	0.01491	0.50895	0.40145	-293.47	-330.84	74.73	0.40726	0.39512
A:DRB1	0.01299	0.42896	0.38416	-282.00	-309.16	54.32	0.32934	0.38370
A:DQA1	0.01219	0.33413	0.36466	-269.57	-286.08	33.02	0.25803	0.34897
A:DQB1	0.01356	0.39266	0.37495	-275.58	-297.62	44.07	0.29931	0.37489
A:DPA1	0.01681	0.32397	0.36666	-219.78	-226.97	14.38	0.19446	0.35360
A:DPB1	0.01362	0.42240	0.40404	-237.85	-262.06	48.42	0.33848	0.41739
C:B	0.04125	0.88739	0.85752	-210.37	-342.68	264.63	0.84781	0.86104
C:DRB1	0.01698	0.48046	0.47513	-280.34	-317.66	74.62	0.32308	0.47691
C:DQA1	0.02072	0.47797	0.49368	-263.23	-293.74	61.01	0.31386	0.50338
C:DQB1	0.01766	0.45793	0.49879	-269.55	-305.28	71.46	0.30479	0.50122
C:DPA1	0.02039	0.41030	0.46438	-224.72	-236.52	23.61	0.21172	0.46433
C:DPB1	0.01898	0.46453	0.37002	-242.45	-268.46	52.01	0.33462	0.45327
B:DRB1	0.01723	0.50254	0.41712	-286.79	-320.50	67.42	0.32654	0.43913
B:DQA1	0.01845	0.44225	0.43582	-271.36	-296.59	50.45	0.28877	0.44993
B:DQB1	0.01958	0.49040	0.43654	-277.30	-308.13	61.65	0.31328	0.45679
B:DPA1	0.01875	0.37441	0.40117	-229.76	-239.16	18.80	0.20689	0.40443
B:DPB1	0.01898	0.46082	0.38001	-247.84	-272.77	49.86	0.32227	0.45680
DRB1:DQA1	0.06138	0.92556	0.92465	-164.06	-271.56	214.99	0.82051	0.93006
DRB1:DQB1	0.06058	1.00000	1.00000	-147.74	-283.10	270.72	0.93302	1.00000

...

For each locus pair, we report three measures of overall linkage disequilibrium. D' (Hedrick, 1987) weights the contribution to LD of specific allele pairs by the product of their allele frequencies (D' in the output); W_n (Cramér, 1946) is a re-expression of the chi-square statistic for deviations between observed and expected haplotype frequencies (W_n in the output)). $W_{A/B}$ and $W_{B/A}$ (ALD_1_2 and ALD_2_1, respectively in the output) are extensions of W_n that account for asymmetry when the number of alleles differs at two loci (Thomson and Single, 2014). Below we describe the measures, each of which is normalized to lie between zero and one.

 D'

Overall LD, summing contributions ($D'_{ij} = D_{ij}/D_{max}$) of all the haplotypes in a multi-allelic two-locus system, can be measured using Hedrick's D' statistic, using the products of allele frequencies at the loci, p_i and q_j , as weights.

$$D' = \sum_{i=1}^I \sum_{j=1}^J p_i q_j |D'_{ij}|$$

 W_n

Also known as Cramer's V Statistic (Cramér, 1946), W_n , is a second overall measure of LD between two loci. It is a re-expression of the Chi-square statistic, X^2_{LD} , normalized to be between zero and one. When there are only two alleles per locus, W_n is equivalent to the correlation coefficient between the two loci, defined as:

$$W_n = \left[\frac{\sum_{i=1}^I \sum_{j=1}^J D_{ij}^2 / p_i q_j}{\min(I-1, J-1)} \right]^{\frac{1}{2}} = \left[\frac{X^2_{LD} / 2N}{\min(I-1, J-1)} \right]^{\frac{1}{2}}$$

two alleles case

When there are only two alleles per locus, W_n is equivalent to the correlation coefficient between the two loci, defined as $r = \sqrt{D_{11}/p_1 p_2 q_1 q_2}$.

 $W_{A/B}$ and $W_{B/A}$

When there are different numbers of alleles at the two loci, the direct correlation property for the r correlation measure is not retained by W_n , its multi-allelic extension. The complementary pair of conditional asymmetric LD (ALD) measures, $W_{A/B}$ and $W_{B/A}$, were developed to extend the W_n measure. $W_{A/B}$ is (inversely) related to the degree of variation of A locus alleles on haplotypes conditioned on B locus alleles. If there is no variation of A locus alleles on haplotypes

conditioned on B locus alleles, then $W_{A/B} = 1$ $W_{A/B} = W_{B/A} = W_n$ when there is symmetry in the data and thus for bi-allelic SNPs.

$$W_{A/B} = \left[\frac{\sum_{i=1}^I \sum_{j=1}^J D_{ij}^2 / q_j}{1 - F_A} \right]^{\frac{1}{2}}$$

$$W_{B/A} = \left[\frac{\sum_{i=1}^I \sum_{j=1}^J D_{ij}^2 / p_i}{1 - F_B} \right]^{\frac{1}{2}}$$

In addition to the LD measures described above, for each locus pair, we describe three additional measures related to the log-likelihood that are displayed in the output above:

$\ln(L_1)$

the log-likelihood of obtaining the observed data given the inferred haplotype frequencies ($\ln(L_1)$ in the output)

$\ln(L_0)$

the log-likelihood of the data under the null hypothesis of linkage equilibrium ($\ln(L_0)$ in the output)

S

the statistic (S in the output) is defined as twice the difference between these likelihoods. S has an asymptotic chi-square distribution, but the null distribution of S is better approximated using a randomization procedure. If a permutation test is requested (by setting the option `allPairwiseLDWithPermu` to a number greater than zero in the *.ini file*), the empirical distribution of S is generated by shuffling genotypes among individuals, separately for each locus, thus creating linkage equilibrium. The additional column `# permu` that will be generated (not shown in the example output above) will indicate how many permutations were carried out. The p -value (also not shown) will be the fraction of permutations that results in values of S greater or equal to that observed. A $p < 0.05$ is indicative of overall significant LD.

Individual LD coefficients, D_{ij} , are stored in the XML output file, but are not printed in the default text output. They can be accessed in the summary text files created by the `popmeta` script (see *What happens when you run PyPop?*).

Haplotype frequency estimation

```
Haplotype frequency est. for loci: A:B:DRB1
-----
Number of individuals: 47 (before-filtering)
Number of individuals: 45 (after-filtering)
Unique phenotypes: 45
Unique genotypes: 113
Number of haplotypes: 188
Loglikelihood under linkage equilibrium [ln(L_0)]: -472.700542
Loglikelihood obtained via the EM algorithm [ln(L_1)]: -340.676530
Number of iterations before convergence: 67
```

The estimated haplotype frequencies are sorted alphanumerically by haplotype name (left side), or in decreasing frequency (right side). Only haplotypes estimated at a frequency of 0.00001 or larger are reported. The first column gives the allele names in each of the three loci, the second column provides the maximum likelihood estimate for their frequencies, (`frequency`), and the third column gives the corresponding approximate number of haplotypes (`# copies`).

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
01:01~14:01~09:01	0.01111	1.0	03:01~14:01~04:07	0.03333	3.0
01:01~15:20~08:02	0.01111	1.0	03:01~13:01~04:02	0.03333	3.0
01:01~18:01~04:07	0.01111	1.0	02:01~14:01~11:01	0.03332	3.0
01:01~39:02~04:04	0.01111	1.0	03:01~15:20~08:02	0.02222	2.0
01:01~39:02~16:02	0.01111	1.0	01:01~40:05~08:02	0.02222	2.0
01:01~40:05~08:02	0.02222	2.0	03:01~39:02~04:02	0.02222	2.0
01:01~81:01~08:02	0.01111	1.0	02:01~13:01~16:02	0.02222	2.0

(continues on next page)

(continued from previous page)

01:01~81:01~16:02	0.01111	1.0		02:18~14:01~04:04	0.02222	2.0
02:01~13:01~16:02	0.02222	2.0		02:10~51:01~16:02	0.02222	2.0
02:01~14:01~04:02	0.03335	3.0		02:18~14:01~16:02	0.02222	2.0
02:01~14:01~04:04	0.01111	1.0		01:01~13:01~04:02	0.02222	2.0
02:01~14:01~04:07	0.02222	2.0		25:01~40:05~08:02	0.02222	2.0
02:01~14:01~08:02	0.01111	1.0		25:01~13:01~08:02	0.02222	2.0
...						

CONTRIBUTING TO PYPOP

Contributions to PyPop are welcome, and they are greatly appreciated! Every little bit helps, and *credit will always be given*.

4.1 Reporting and requesting

Did you find a bug?

When [reporting a bug](#)¹ please use one of the provided issue templates if applicable, otherwise just start a blank issue and describe your situation. Here is a checklist:

- **Check previous issues.** Ensure the bug was not already reported by searching on GitHub under [Issues](#)².
- **Provide complete self-contained examples.** If you're unable to find an open issue addressing the problem, open a new one. Be sure to include a title and clear description, as much relevant information as possible, and a code sample or an executable test case (including any input files) demonstrating the expected behavior that is not occurring.
- **Use templates.** If possible, use the relevant bug report templates to create the issue. For a standard bug report (including installation issues), please use this: [bug report template](#)³, for feature requests or documentation issues, see below.
- **Provide full commands and errors as plaintext, not screenshots.** When you are including the output of an error in your bug report (whether an installation error, a build error, an error running pypop or an error building docs), please cut-and-paste from your console application or terminal, the entire set of commands leading up to the error, along with the **complete** error output as a **single plaintext** output. E.g. here is an example error from running pypop on a badly formed .ini file:

```
$ pypop -c minimal.ini USAFEL-UchiTelle-small.pop
Traceback (most recent call last):
  File "/home/user/.conda/envs/pypop/bin/pypop", line 8, in <module>
    sys.exit(main())
  File "/home/user/.conda/envs/pypop/lib/python3.10/site-packages/PyPop/pypop.py", line 250, in main
    config = getConfigInstance(configFilename, altpath)
  File "/home/user/.conda/envs/pypop/lib/python3.10/site-packages/PyPop/Main.py", line 62, in getConfigInstance
    config.read(configFilename)
  File "/home/user/.conda/envs/pypop/lib/python3.10/configparser.py", line 698, in read
    self._read(fp, filename)
  File "/home/user/.conda/envs/pypop/lib/python3.10/configparser.py", line 1086, in _read
    raise MissingSectionHeaderError(fpname, lineno, line)
configparser.MissingSectionHeaderError: File contains no section headers.
file: 'minimal.ini', line: 4
' j[General]\n'
```

Please do not just post screenshots of commands and error output. It's OK if you want to also include a screenshot as supplement, but be sure you also include the commands and output as plaintext as well. (If the output is too long for including inline as a comment on the issue, you can save it in a file, and drag-and-drop it into an issue comment).

- **Include environment.** When reporting bugs, especially during installation, please run the following and include the output of:

```
echo $CPATH
echo $LIBRARY_PATH
echo $PATH
which python
```

If you are running on MacOS, and you used the MacPorts installation method, please also run and include the output of:

```
port installed
```

- **Keep each issue focused on one specific problem.** Each issue should be focused on one problem. Don't use an issue for open-ended discussion, or as a place to collect all issues with pypop you run into. If, during the comments, you discover another bug, unrelated to the current issue, please open up a new issue and reference it in the current issue.
- **Run the test suite.** In many cases, especially if you are investigating a new platform (e.g. new architecture) developers may ask you run the full test suite via `pytest`, see [run unit tests](#). in “verbose” mode (i.e. `pytest -v`). If you do this, please supply the output of the resulting temporary directories on your issue (see the unit test section for more details). Note that you will likely need to [clone the main repository](#) as the unit tests are not distributed with the binary wheels.

Documentation improvements

pypop could always use more documentation, whether as part of the official docs, in docstrings, or even on the web in blog posts, articles, and such. Write us a [documentation issue](#)⁴ describing what you would like to see improved in here.

If you are able to contribute directly (e.g., via a pull request), please read our [website contribution guide](#).

Feature requests and feedback

The best way to send feedback is to file an issue using the [feature template](#)⁵.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome

4.2 Making a code contribution

To contribute new code that implement a feature, or fix a bug, this section provides a step-by-step guide to getting you set-up. The main steps are:

1. forking the repository (or “repo”)
2. cloning the main repo on to your local machine
3. making a new branch
4. [installing a development version](#) on your machine
5. updating your branch when “upstream” (the main repository) has changes to include those changes in your local branch
6. running code quality checks, like unit tests and `pre-commit` checks, using `nox` (or the tools directly)
7. updating `AUTHORS.rst`
8. making a pull request (including a description of your changes suitable for inclusion in `NEWS.md`)

Fork this repository

Fork this repository before contributing⁶. Forks creates a cleaner representation of the contributions to the project⁷.

Clone the main repository

Next, clone the main repository to your local machine:

```
git clone https://github.com/alexlancaster/pypop.git
cd pypop
```

Add your fork as an upstream repository:

```
git remote add myfork git://github.com/YOUR-USERNAME/pypop.git
git fetch myfork
```

Make a new branch

From the main branch create a new branch where to develop the new code.

```
git checkout main
git checkout -b new_branch
```

Note the main branch is from the main repository.

Build locally and make your changes

Now you are ready to make your changes. First, you need to build `pypop` locally on your machine, and ensure it works, see the separate section on *building and installing a development version*.

Once you have done the installation and have verified that it works, you can start to develop the feature, or make the bug fix, and keep regular pushes to your fork with comprehensible commit messages.

```
git status
git add # (the files you want)
git commit # (add a nice commit message)
git push myfork new_branch
```

While you are developing, you should use `nox` to frequently run unit tests and check for code quality (or temporary wheels), as described in *use nox for testing and code quality*.

Keep your branch in sync with upstream

You should keep your branch in sync with the upstream main branch. For that:

```
git checkout main # return to the main branch
git pull # retrieve the latest source from the main repository
git checkout new_branch # return to your devel branch
git merge --no-ff main # merge the new code to your branch
```

At this point you may need to solve merge conflicts if they exist. If you don't know how to do this, I suggest you start by reading the *official docs*⁸

You can push to your fork now if you wish:

```
git push myfork new_branch
```

And, continue doing your developments as previously discussed.

Use nox for testing and code quality

In addition to installing PyPop for development (see earlier sections), we have developed configurations for `nox`⁹ to simplify common developer tasks such as running tests, code formatting checks, and documentation builds. `nox` runs tasks in isolated Python environments, ensuring consistency across contributors and CI pipelines. Install `nox` (within the same environment as used for your build and install):

```
pip install nox
```

You can list the currently configured developer tasks available to run via `nox`, by calling `nox` itself:

```
nox --list
```

We describe some of the tasks that we recommend contributors run regularly during development of code contributions, below.

Run unit tests

To run the full (pytest-based) unit test suite:

```
nox --sessions tests
```

You should also frequently use `nox` to run the unit tests as you are developing your code, to ensure that you don't inadvertently break anything, especially before commits, or creating a Pull Request from your branch. (`-s` is the short option `--sessions`).

These are exactly the same tests that will be performed online via Github Actions continuous integration (CI). This project follows CI good practices (let us know if something can be improved).

Preserving output from unit tests

Running `nox -s tests` uses `pytest` within an isolated environment. To debug tests or preserve output from temporary directories, you may prefer to run `pytest` directly:

```
pytest -s -v tests
```

Supplying the `-v` verbose option will preserve the run-time output of unit tests that write files to disk in temporary directories unique for each run. The format of the output directories is `run_test_<test-name>_<unique_id>`, e.g. the directories created will look similar to the following:

```
run_test_AlleleColon_HardyWeinberg_u3dnf99y
run_test_USAFEL_49h_exhg
```

These directories store per-test outputs but are normally deleted unless output is preserved.

If you run into errors during your initial installation, please first carefully repeat and/or check your installation. If you still get errors, file a bug, and include the output of `pytest` run in verbose mode and capturing the output, as described above.

Run pre-commit checks

All pull requests (PRs) submitted to PyPop will be automatically run through a series pre-configured `pre-commit checks`¹⁰(called “hooks”), configured in the `.pre-commit-config.yaml` [YAML file](#)¹¹.

To run these checks locally:

```
nox -s precommit
```

These checks reformat code and catch errors in:

- Python code (uses `ruff` and, `ruff-format` hooks)
- C extension code (uses `clang-format` to format code according to the LLVM style)

- Common formatting errors in documentation, including Markdown and RST
- Check code and documentation for spelling errors (via codespell)

Running the `precommit` checks will result in either:

1. All checks passing (no action needed)
2. Some checks fail, this can be due either to:
 - Code being reformatted to coding standards (use `git diff` to see the additional changes), but are otherwise OK. Generally, all you need to do then is to re-run the `nox -s precommit` command, and it will proceed according to (1)
 - An error is detected in the code that requires manual intervention (e.g. non-standard Python construct, formatting issue, spelling error). Please fix this and re-run `nox -s precommit` until it passes.

`pre-commit` checks are the same checks that will be enforced automatically in your PR via GitHub Actions CI. Running them locally before submitting a PR will speed up acceptance. The automated `pre-commit` checks will be run just once upon initial PR submission, and results posted on the PR. This may also result in changes to the code (mainly reformatting that can be applied automatically). You will need to ensure that you pull these changes back to your local checkout before applying new changes.

`pre-commit` Git Hook

We highly recommend you enable `pre-commit` Git hook in your *local checkout*, **before** you commit to your PR branch, so you can catch errors early. Ensuring your code passes `pre-commit` checks will speed the merging of your PR into the main branch, as the code will already be in a good state for merging.

To enable checks, first ensure that `pre-commit` is installed, and then install the hooks:

```
pip install pre-commit
pre-commit install --install-hooks
```

You can then manually trigger checks using `pre-commit` (i.e. outside `nox`):

```
pre-commit run
```

If you attempt to commit to the repo, e.g. using a command like `git commit -a`, `pre-commit` checks will run on your changed files, and behave as if `pre-commit run` had been called directly. Once all checks pass the `git commit` command will commit to the repository and you can `git push` your changes.

Build distribution packages

To locally build sdist and wheel packages for testing, you can run the `build` command:

```
nox -s build
```

Update `AUTHORS.rst`

Also add your name to the author table at `AUTHORS.rst`, so you will also be included in the periodic Zenodo software releases (see also the section on [Crediting contributors](#)).

Make a Pull Request

Once you are finished, create a pull request to the main repository and engage with the developers.

When you create the pull request in the initial submission box, you should create a description of your changes with an explanatory bullet list of the contributions. Please note if any of your changes will break existing behaviour or anything else that would be important for an end-user to know. This description should be in Markdown format. Here is an example:

New features

- here goes my new additions, explain them shortly and well
- this feature will require an update to your `.ini` file

This will be used to populate the Release Notes and eventually be included in the `NEWS.md` file.

If you need some code review or feedback while you're developing the code, you can also make a pull request, even if you're not fully finished.

However, before submitting a Pull Request, verify your development branch passes all tests as *described above* . If you are developing new code you should also implement new test cases.

Pull Request checklist

Before requesting a final merge, you should:

1. Make sure your PR passes all existing `pytest` tests.
2. Add unit tests if you are developing new features and make sure these also pass.
3. Run and address the *pre-commit checks as described above*.
4. Update documentation when there's new API, functionality etc.
5. In the submission for the PR, include a description of the changes, in markdown format, suitable for eventual inclusion in `NEWS.md`.
6. Add yourself to `AUTHORS.rst`.

Note

Note that the `pre-commit` checks are automatically run on all new PRs, and this may result in changes to your code, please approve or otherwise ensure these changes make it back into your development branch.

4.3 Installation for developers

Once you have setup your branch as described in *making a code contribution*, above, you are ready for the four main steps of the developer installation:

1. install a build environment
2. build
3. run tests

Note

Note that you if you need to install PyPop from source, but do not intend to contribute code, you can skip creating your own forking and making an additional branch, and clone the main upstream repository directly:

```
git clone https://github.com/alexlancaster/pypop.git
cd pypop
```

For most developers, we recommend using the miniconda approach described below.

Install the build environment

To install the build environment, you should choose either conda or system packages. Once you have chosen and installed the build environment, you should follow the instructions related to the option you chose here in all subsequent steps.

Install build environment via miniconda (recommended)

1. Visit <https://docs.conda.io/en/latest/miniconda.html> to download the miniconda installer for your platform, and follow the instructions to install.

In principle, the rest of the PyPop miniconda installation process should work on any platform that is supported by miniconda, but only Linux and MacOS have been tested in standalone mode, at this time.

2. Once miniconda is installed, create a new conda environment, using the following commands:

```
conda create -n pypop3 gsl swig python=3
```

This will download and create a self-contained build-environment that uses of Python to the system-installed one, along with other requirements. You will need to use this this environment for the build, installation and running of PyPop. The conda environment name, above, `pypop3`, can be replaced with your own name.

When installing on MacOS, before installing conda, you should first to ensure that the Apple Command Line Developer Tools (XCode) are [installed](#)¹², so you have the compiler (`clang`, the drop-in replacement for `gcc`), `git` etc. conda is unable to include the full development environment for `clang` as a conda package for legal reasons.

3. Activate the environment, and set environments variables needed for compilation:

```
conda activate pypop3
conda env config vars set CPATH=${CONDA_PREFIX}/include:${CPATH}
conda env config vars set LIBRARY_PATH=${CONDA_PREFIX}/lib:${LIBRARY_PATH}
conda env config vars set LD_LIBRARY_PATH=${CONDA_PREFIX}/lib:${LD_LIBRARY_PATH}
```

4. To ensure that the environment variables are saved, reactivate the environment:

```
conda activate pypop3
```

5. Skip ahead to *Build PyPop*.

Install build environment via system packages (advanced)

Unix/Linux:

1. Ensure Python 3 version of `pip` is installed:

```
python3 -m ensurepip --user --no-default-pip
```

Note the use of the `python3` - you may find this to be necessary on systems which parallel-install both Python 2 and 3, which is typically the case. On newer systems you may find that `python` and `pip` are, by default, the Python 3 version of those tools.

2. Install packages system-wide:

1. Fedora/Centos/RHEL

```
sudo dnf install git swig gsl-devel python3-devel
```

2. Ubuntu

```
sudo apt install git swig libgsl-dev python-setuptools
```

MacOS X

1. Install the developer command-line tools: <https://developer.apple.com/downloads/> (includes git, gcc). (Note that you may have to sign-in/create a developer account with Apple using your Apple ID to access this link.). You may also be able to install via the terminal and skip the above step by running `xcode-select --install` (but first check to see if you already have a version installed, see <https://mac.install.guide/commandlinetools/4.html> for more details).
2. Visit <https://www.macports.org> and follow the instructions there to install the latest version of MacPorts for your version of MacOS X.
3. Set environment variables to use macports version of Python and other packages, packages add the following to `~/ .bash_profile`

```
export PATH=/opt/local/bin:$PATH
export LIBRARY_PATH=/opt/local/lib:$LIBRARY_PATH
export CPATH=/opt/local/include:$CPATH
```

4. Rerun your bash shell login in order to make these new exports active in your environment. At the command line type:

```
exec bash -login
```

5. Install dependencies via MacPorts and set Python version to use (FIXME: currently untested!)

```
sudo port install swig-python gsl py39-numpy py39-lxml py39-setuptools py39-pip py39-pytest
sudo port select --set python python39
sudo port select --set pip pip39
```

6. Check that the MacPorts version of Python is active by typing: `which python`, if it is working correctly you should see `/opt/local/bin/python`.

Windows

You will need a compiler installed, the GitHub Action is tested using [Microsoft Visual Studio 17 2022](#)¹³ (you can also download [2019 and earlier versions](#)¹⁴). We also recommend that you setup the [NuGet package repository](#)¹⁵ to install following build-time dependencies.

Note

Please note that we have not directly tested building on standalone Windows machines, only via the GitHub runner workflows. In addition, the ARM64 wheels are cross-compiled on the GitHub runner, which cannot run the resulting wheels, therefore all unit tests are skipped.

1. Install swig: when compiled on a GitHub runner, the swig package is part of the default image. If compiled on a standalone-mode Windows machine, swig may be available as NuGet package, and installable (untested):

```
nuget install swig
```

2. Install gsl.

- X64: install the gsl package:

```
nuget install gsl-msvc14-x64 -Version 2.3.0.2779
```

- ARM64: The NuGet repository doesn't have an ARM64 version of gsl, it is necessary to build a .nupkg from source (see details in [DEV_NOTES.md](#)¹⁶). We have made this available in [vendor-binaries](#)¹⁷ directory of the repo. To install the package from top-level repository run:

```
nuget install gsl-msvc14-arm64 -Source "%CD%\vendor-binaries
```

- Before starting the build process, you need to modify the environment variables CPATH and LIBRARY_PATH to point to the installed gsl package, e.g. for X64:

```
CPATH="gsl-msvc14-x64.2.3.0.2779\\build\\native"
LIBRARY_PATH="gsl-msvc14-x64.2.3.0.2779\\build\\native\\static"
```

Build PyPop

You should choose *either* of the following two approaches. Don't try to mix-and-match the two. The build-and-install approach is only recommended if don't plan to make any modifications to the code locally.

Build-and-install (not recommended for developers)

Once you have setup your environment and cloned the repo, you can use the following one-liner to examine the `setup.py` and pull all the required dependencies from `pypi.org` and build and install the package.

Note that if you use this method and install the package, it will be available to run anywhere on your system, by running `pypop`.

If you use this installation method, changes you make to the code, locally, or via subsequent `git pull` requests will not be available in the installed version until you repeat the `pip install` command.

- if you installed the conda development environment, use:

```
pip install .[test]
```

(the `[test]` keyword is included to make sure that any package requirements for the test suite are installed as well).

- if you installed a system-wide environment, the process is slightly different, because we install into the user's `$HOME/.local` rather than the conda environment:

```
pip install --user .[test]
```

- PyPop is ready-to-use, you should *run unit tests*.
- if you later decide you want to switch to using the developer approach, below, follow the *cleaning up build* before starting.

Build-and-install developer-mode (recommended for developers)

Installing in “developer” or “edit” mode¹⁸ should be used by developers, or anyone who wants to make changes to PyPop code. It is almost identical to the regular installation above (e.g. it will pull down all required dependencies automatically), but instead you will add the `--editable` option (`-e` is the short version) to the `pip install` command. In edit mode, any changes you make in your local code will be reflected in the installed version.

- conda

```
pip install --editable .[test]
```

- system-wide

```
pip install --user --editable .[test]
```

- The scripts `pypop` and `popmeta` will operate the same way, and any changes in the underlying Python `.py` files will be picked up by the scripts.

Cleaning up build

To clean up, first uninstall PyPop (whether you installed in editable mode or not):

```
pip uninstall pypop-genomics
```

In addition, to clean-up any compiled files and force a recompilation from scratch, run the `clean` command:

```
./setup clean --all
```

4.4 Install package from GitHub Releases

Packages that are released to PyPI, are also available via the releases on the GitHub release page:

<https://github.com/alexlancaster/pypop/releases>

Warning

We recommend installing binary packages using the main PyPI repository, **not** via the GitHub release packages. However from time to time, we also sometimes make binary packages that are not necessarily also released via PyPI. In addition, if PyPI is unavailable, you may want to install directly from the GitHub release. These instructions will help you do that.

Installing these packages is similar to installing via PyPI, except that you need to explicitly provide a URL to the release page.

1. First, visit the release page, and choose the release version you wish to install (usually the most recent), and note the release tag (e.g. `v1.0.0`).

Release version numbers

Note that version of the release is slightly different to the `git` tag. This is because the `git` tag follows [Semantic Versioning](#)¹⁹, which Python internally normalizes and abbreviates. So the release with the `git` tag `v1.0.0` is actually version `1.0.0` of the `pypop-genomics` package, and the version that `pip` “sees” (the difference is more notable with prereleases which might have a `git` tag of `v1.0.0-rc2` but the PyPI version will be `1.0.0rc2`).

2. Next, use `pip` to install the package by running a command of the form (this will select and install the correct wheel for your Python version and operating system automatically):

```
pip install pypop-genomics -f https://github.com/alexlancaster/pypop/releases/expanded_assets/<TAG_NAME>
```

where `<TAG_NAME>` is replaced with a specific tag, e.g. for the example given above, you would run:

```
pip install pypop-genomics -f https://github.com/alexlancaster/pypop/releases/expanded_assets/v1.0.0
```

You can also manually download the specific wheel from the github release webpage and install directly, e.g.:

```
pip install pypop_genomics-1.0.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
```

4.5 Making a documentation or website contribution

Interested in maintaining the PyPop website and/or documentation, such as the *PyPop User Guide*? Here are ways to help.

Overview

All the documentation (including the website homepage) are maintained in this directory (and subdirectories) as `reStructuredText`²⁰ (.rst) documents. `reStructuredText` is very similar to GitHub markdown (.md) and should be fairly self-explanatory to edit (especially for pure text changes). From the .rst “source” files which are maintained here on github, we use `sphinx`²¹ to generate (aka “compile”) the HTML for both the pypop.org user guide and and PDF (via LaTeX) output. We have setup a GitHub action, so that as soon as a documentation source file is changed, it will automatically recompile all the documentation, update the `gh-pages` branch (which is synced to the GitHub pages) and update the files on the website.

Here’s an overview of the process:

```
.rst files -> sphinx -> HTML / PDF -> push to gh-pages branch -> publish on pypop.org
```

This means that any changes to the source will automatically update both website home page the documentation.

Once any changes are pushed to a branch (as described below), the GitHub action will automatically rebuild the website, and the results will be synced to a “staging” version of the website at:

- <https://alexlanaster.github.io/beta.pypop.org/>

Structure

Here’s an overview of the source files for the website/documentation located in the `website` subdirectory at the time of writing. Note that some of the documentation and website files, use the `include::` directive to include some “top-level” files, located outside `website` like `README.rst` and `CONTRIBUTING.rst`:

- `index.rst` (this is the source for the homepage at <http://pypop.org/>)
- `conf.py` (Sphinx configuration file - project name and other global settings are stored here)
- `docs` (directory containing the source for the *PyPop User Guide*, which will eventually live at <http://pypop.org/docs>).
 - `index.rst` (source for the top-level of the *PyPop User Guide*)
 - `guide-chapter-install.rst` (pulls in parts of the top-level `README.rst`)
 - `guide-chapter-usage.rst`
 - `guide-chapter-instructions.rst`
 - `guide-chapter-contributing.rst` (pulls in top-level `CONTRIBUTING.rst` that contains the source of the text that you are reading right now)
 - `guide-chapter-changes.rst` (pulls in top-level `NEWS.md` and `AUTHORS.rst`)
 - `licenses.rst` (pulls in top-level `LICENSE`)
 - `biblio.rst`
 - `pypop.bib` (BibTeX source file for bibliography)
- `html_root` (any files or directories committed in this directory will appear at the top-level of the website)
 - `psb-pypop.pdf` (e.g. this resides at <http://pypop.org/psb-pypop.pdf>)
 - `tissue-antigens-lancaster-2007.pdf`
 - `PyPopLinux-0.7.0.tar.gz` (old binaries - will be removed soon)
 - `PyPopWin32-0.7.0.zip`
 - `popdata` (directory - Suppl. data for Solberg et. al 2018 - <http://pypop.org/popdata/>)
- `reference` (directory containing the old DocBook-based documentation, preserved to allow for unconverted files to be converted later, this directory is ignored by the build process)

Modifying documentation

Minor modifications

For small typo fixes, moderate copyedits at the paragraph level (e.g. adding or modifying paragraphs with little or no embedded markup), you can make changes directly on the github website.

1. navigate to the `.rst` file you want to modify in the GitHub code directory, you'll see a preview of how most of the `.rst` will be rendered
2. hover over the edit button - you'll see an **"Edit the file in a fork in your project"** (if you are already a project collaborator, you may also have the optional of creating a branch directly in the main repository).
3. click it and it will open up a window where you can make your changes
4. make your edits (it's a good idea to look at the preview tab periodically as you make modifications)
5. once you've finished with the modifications, click **"Commit changes"**
6. put in an a commit message, and click **"Propose changes"**
7. this will automatically create a new branch in your local fork, and you can immediately open up a pull-request by clicking **"Create pull request"**
8. open up a pull-request and submit - new documentation will be automatically built and reviewed. if all is good, it will be merged by the maintainer and made live on the site.

Major modifications

For larger structural changes involving restructuring documentation or other major changes across multiple `.rst` files, **it is highly recommended** that you should make all changes in your own local fork, by cloning the repository on your computer and then building the documentation locally. Here's an overview of how to do that:

1. make a fork of pypop if you haven't already (see [previous section](#))
2. *clone the fork and add your fork as an upstream repository* on your local computer, and *make a new branch*. Note that you do not have to build the PyPop software first in order to build the documentation, you can build them separately.
3. make your changes to your `.rst` files and/or `conf.py`
4. build the HTML documentation, using `nox`, which creates a temporary virtual environment with `sphinx` and `sphinx` extensions (see [nox installation](#)):

```
nox -s docs
```

5. view the local documentation: you can open up browser and navigate to the `index.html` in the top-level of the newly-created `_htmlbuild` directory
6. use `git commit` to commit your changes to your local fork.
7. open up a pull-request against the upstream repository

Building the PDF for the *PyPop User Guide* is a bit more involved, as you will need to have various TeX packages installed.

1. install the LaTeX packages (these are packages needed for Ubuntu, they may be different on your distribution):

```
sudo apt-get install -y latexmk texlive-latex-recommended texlive-latex-extra texlive-fonts-recommended texlive-
→ fonts-extra texlive-luatex texlive-xetex
```

Ubuntu LaTeX packages

The most up to date list of LaTeX packages for Ubuntu are in the “Sphinx build” section of the workflow [.github/workflows/documentation.yaml](#)²² that builds the documentation upon deployment. Consult this if the list of packages, becomes out of date.

2. build the LaTeX and then compile the PDF, using the nox command:

```
nox -s docs_pdf
```

3. the user guide will be generated in `_latexbuild/pypop-guide.pdf`

Note

To change the target output directory of either the HTML or PDF documentation, you can supply the directory name on the nox command-line, after a `--` e.g.

```
nox -s docs -- output_dir
```

4.6 Crediting contributors

Note

These guidelines were heavily adapted from [similar guidelines](#)²³ in the PyGMT project.

We define *contributions* in a broad way: including both writing code as well as documentation, and reviewing issues and PRs etc. Here are some ways we credit contributors:

AUTHORS.rst, NEWS.md and GitHub Release Notes

Anyone who has contributed a pull request to the project is welcome to add themselves (or request to be added) to `AUTHORS.rst`, which is part of the repository and included with distributions.

Every time we make a release, everyone who has made a commit to the repository since the previous release will be mentioned in either the `NEWS.md` or in the GitHub Release Notes.

Authorship on Zenodo archives of releases

Anyone who has contributed to the repository (i.e., appears on `git log`) will be invited to be an author on the [Zenodo](#)²⁴ archive of new releases.

To be included as an author, you *must* add the following to the `AUTHORS.rst` file of the repository:

1. Full name (and optional link to your website or GitHub page)
2. [ORCID](#)²⁵ (optional)
3. Affiliation (optional)

The order of authors is generally defined by the number of commits to the repository (`git shortlog -sne`). The order can also be changed on a case-by-case basis, such as contributions to PyPop project that due not relate to commit numbers, such as writing grants/proposals, and other programming efforts (including reviewing PRs).

If you have contributed and *do not* wish to be included in Zenodo archives, either don't add yourself to `AUTHORS.rst`, or open an issue or file a PR that:

1. Removes yourself from `AUTHORS.rst`, or;

2. Indicates next to your name on `AUTHORS.rst` that you do not wish to be included with something like (not included in Zenodo).

Note that authors included in the Zenodo archive will also have their name listed in the `CITATION.cff` file. This is a machine (and human) readable file that enables citation of PyPop easily.

Scientific publications (papers)

From time to time we may write academic papers for PyPop, e.g., for major changes or significant new components of the package.

To be included as an author on the paper, you *must* have

1. either made multiple and regular contributions to the PyPop repository; or, have made other non-coding contributions (or both);
2. have participated in the writing and reviewing of the paper.
3. added your full name, affiliation, and (optionally) ORCID to the paper.
4. written and/or read and review the manuscript in a timely manner and provide comments on the paper

Notes

1. <https://github.com/alexlancaster/pypop/issues>
2. <https://github.com/alexlancaster/pypop/issues>
3. https://github.com/alexlancaster/pypop/issues/new?assignees=&labels=bug&projects=&template=bug\protect_report.yml
4. <https://github.com/alexlancaster/pypop/issues/new?assignees=&labels=documentation&projects=&template=documentation.yml>
5. https://github.com/alexlancaster/pypop/issues/new?assignees=&labels=enhancement&projects=&template=feature\protect_request.yml
6. <https://github.com/alexlancaster/pypop/network/members>
7. <https://github.com/alexlancaster/pypop/network>
8. <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github>
9. <https://nox.thea.codes/>
10. <https://pre-commit.com/>
11. <https://github.com/alexlancaster/pypop/blob/main/.pre-commit-config.yaml>
12. <https://mac.install.guide/commandlinetools/4.html>
13. <https://visualstudio.microsoft.com/downloads/>
14. <https://visualstudio.microsoft.com/vs/older-downloads/>
15. <https://www.nuget.org/packages>
16. https://github.com/alexlancaster/pypop/blob/main/DEV\protect_NOTES.md
17. <https://github.com/alexlancaster/pypop/tree/main/vendor-binaries>
18. https://setuptools.pypa.io/en/latest/userguide/development\protect_mode.html
19. <https://semver.org/>
20. <https://docutils.sourceforge.io/rst.html>
21. <https://www.sphinx-doc.org/en/master/>
22. <https://github.com/alexlancaster/pypop/blob/main/.github/workflows/documentation.yaml>
23. <https://github.com/GenericMappingTools/pygmt/blob/main/AUTHORSHIP.md>
24. <https://zenodo.org/>
25. <https://orcid.org>

AUTHORS AND HISTORY

5.1 PyPop contributors

(Listed in alphabetical order)

Author	ORCID	Affiliation	Contribution
Karl Kornel	0000-0001-5847-5330 ¹	Stanford Research Computing Center	Contributed containerization
Alex Lancaster	0000-0002-0002-9263 ²	Amber Biology LLC, Ronin Institute	Lead developer. Co-designer of Python framework: author of main engine, text file parser, Python extension module framework using SWIG, XML output and XSLT post-processing framework (to generate plain text and HTML output).
Steven J. Mack	0000-0001-9820-9547 ³	University of California, San Francisco	Contributed bug reports, documentation, reviewed PRs.
Michael P. Mariani	0000-0001-5852-0517 ⁴	Mariani Systems LLC and University of Vermont	Contributed bug reports, documentation, reviewed PRs.
Diogo Meyer	0000-0002-7155-5674 ⁵	University of São Paulo	Reviewed and tested PyPop, contributed some statistical analysis code.
Mark P. Nelson		University of California, Berkeley	Co-designer of Python framework: implemented and maintained Python modules, particularly the module for Hardy-Weinberg analysis. Updated and maintained XSLT code.
Richard M. Single	0000-0001-6054-6505 ⁶	University of Vermont	Author of haplotype frequency and linkage disequilibrium analysis module <code>emhaplofreq</code> . Contributed documentation and testing/reviewing PRs.
Vanessa Sochat	0000-0002-4387-3819 ⁷	Lawrence Livermore National Laboratory	Contributed to the Python 3 port.
Owen Solberg	0000-0003-3060-9709 ⁸		Implemented filter modules, including conversion to allele name information to sequence data.
Jurriaan H. Spaaks	0000-0002-7064-4069 ⁹	Netherlands eScience Center	Contributed to Zenodo upload GitHub action

continues on next page

Table 5.1 – continued from previous page

Author	ORCID	Affiliation	Contribution
Glenys Thomson	0000-0001-5235-4159 ¹⁰	University of California, Berkeley	Principal investigator
Yingssu Tsai ¹¹	0009-0006-0162-6066 ¹²	University of California, Berkeley	Implemented prototype of the allele names to sequence conversion filter module.
Gordon Webster	0009-0009-2862-0467 ¹³	Amber Biology LLC	Contributed documentation and testing framework.

Third-party modules

Included with permission, or via GPL-compatible licenses.

gthwe

The Hardy-Weinberg “exact test” implementation is a modified version of Guo & Thompson’s (Guo and Thompson, 1992) code. Dr. Sun-Wei Guo has kindly allowed us to release the code under the GNU General Public License¹⁴. Original code available at <https://sites.stat.washington.edu/thompson/Genepi/Hardy.shtml>

slatkin-exact/monte-carlo.c

Montgomery Slatkin’s implementation of a Monte Carlo approximation of the Ewens-Watterson exact test of neutrality (Slatkin, 1994, Slatkin, 1996). Original code can be found at: http://ib.berkeley.edu/labs/slatkin/monty/Ewens_exact.program.

pval

The code in the ‘pval’ directory (with the exception of ‘pval.c’ the SWIG wrapper, ‘pval_wrap.i’ and the Makefile) is part of the R project’s ‘nmath’ numerical library <http://www.r-project.org/> and is also licensed under the GNU General Public License (GPL). Minor modifications have been made to allow the module to build correctly.

5.2 PyPop Release History

1.3.1 - 2025-10-07

Internal

- Update lxml requirement from <=6.0.1 to <=6.0.2 (#326¹⁵)
- python(CI deps): bump cibuildwheel from 3.1.4 to 3.2.0 in /.github in the build-wheel-deps group (#328¹⁶)
- Update numpy requirement from <=2.3.2 to <=2.3.3 (#324¹⁷)

Documentation

- Handle code change directives like changed, added, removed in PDF output (#322¹⁸)
- Update theme to pydata-sphinx-theme, add logo + website refresh (#321¹⁹)

1.3.0 - 2025-09-08

Features

- Change: filenames resolved relative to supplied --filename + enable Python 3.14 (#318²⁰)

Internal

- Bump actions/setup-python from 5 to 6 in the actions group (#319²¹)
- Update MacOS cibuildwheel config to handle brew changes (#317²²)
- Update lxml requirement from <=6.0.0 to <=6.0.1 (#314²³)
- python(CI deps): bump cibuildwheel from 3.1.3 to 3.1.4 in /.github in the build-wheel-deps group (#315²⁴)
- Bump setuptools-scm from 9.1.1 to 9.2.0 (#312²⁵)
- Bump the actions group with 2 updates: download-artifact and checkout (#309²⁶)
- Bump setuptools-scm from 8.3.1 to 9.1.1 (#310²⁷)
- python(CI deps): bump cibuildwheel from 3.0.1 to 3.1.3 in /.github in the build-wheel-deps group (#307²⁸)
- Update numpy requirement from <=2.3.1 to <=2.3.2 (#306²⁹)

1.2.2 - 2025-07-28

Internal

- Add developer tasks using nox and overhaul developer documentation (#303³⁰)
- Update numpy requirement from <=2.2.4 to <=2.3.1 (#299³¹)
- python(CI deps): bump sphinx from 8.0.2 to 8.2.3 in /.github in the documentation-deps group (#300³²)
- python(CI deps): Bump cibuildwheel from 3.0.0 to 3.0.1 in /.github in the build-wheel-deps group (#298³³)
- Update cibuildwheel to 3.0.0 (#291³⁴)
- Update GSL build for new windows-2022 runners and GH actions (#295³⁵)
- coding convention and pre-commit updates: move imports to top where possible (#292³⁶)
- Update Windows image for CI: windows-2019 -> windows-2022 (#294³⁷)
- Update lxml requirement from <=5.4.0 to <=6.0.0 (#293³⁸)

Documentation

- Add developer tasks using nox and overhaul developer documentation (#303³⁹)
- version pip install dependencies in documentation workflow to manage by dependabot (#296⁴⁰)
- fix PDF generation and other website upgrades (#286⁴¹)
- Styling the homepage (#283⁴²)
- Add missing citations and fix display, cleanup news (#282⁴³)

1.2.1] - 2025-04-29

Features

- Add support for optional computation of pval using scipy and benchmarking tests (#257⁴⁴)

Bug Fixes

- update all Linux images: musllinux_1_2, manylinux_2_28 + fix musllinux segfaults (#272⁴⁵)
- Ensure long allele names in HW genotype tables don't overlap (#256⁴⁶)

Internal

- Update lxml requirement from `<=5.3.2` to `<=5.4.0` (#280⁴⁷)
- Bump pypa/cibuildwheel from 2.23.2 to 2.23.3 and update runner to ubuntu-22.04 (#279⁴⁸)
- Update lxml requirement from `<=5.3.1` to `<=5.3.2` (#275⁴⁹)
- Bump pypa/cibuildwheel from 2.23.1 to 2.23.2 in the actions group (#273⁵⁰)
- update all Linux images: musllinux_1_2, manylinux_2_28 + fix musllinux segfaults (#272⁵¹)
- Bump pypa/cibuildwheel from 2.23.0 to 2.23.1 in the actions group (#270⁵²)
- Update numpy requirement from `<=2.2.3` to `<=2.2.4` (#269⁵³)
- use pathlib rather than os + update pre-commit hooks (#268⁵⁴)
- Simplify swig installation via PyPI package, other build cleanups (#265⁵⁵)
- Bump pypa/cibuildwheel from 2.22.0 to 2.23.0 and pin swig on Linux (#264⁵⁶)
- Update numpy requirement from `<=2.2.2` to `<=2.2.3` (#261⁵⁷)
- Update lxml requirement from `<=5.3.0` to `<=5.3.1` (#259⁵⁸)
- Tidy workflow and PR labeler (#260⁵⁹)

Contributors:: @alexlancaaster

1.2.0] - 2025-02-04

Features

- Add support for new HLA nomenclature including remote MSF sequence retrieval (#252⁶⁰)
- Build Linux (aarch64) using new ARM64 runners (#250⁶¹)
- [EXPERIMENTAL] Build Windows ARM64-compatible wheels (#235⁶²)

Bug Fixes

- Update all HLA nomenclature to correct version 3 allele names in unit tests and documentation (#251⁶³)
- Restore makeNewPops and fix ParseAlleleCount, major doc upgrade (#247⁶⁴)
- Restore functionality for [Sequence] filters: Python 3 upgrades (#244⁶⁵)
- fix generation of schema.org citation format (#233⁶⁶)

Internal

- Generate .zenodo.json metadata from CITATION.cff (#253⁶⁷)
- Update numpy requirement from `<=2.2.1` to `<=2.2.2` by @dependabot[bot]⁶⁸ (#245⁶⁹)
- Port metadata from setup.py to pyproject.toml (#242⁷⁰)
- Overhaul C extension code: pre-commit and fix compiler warnings (#239⁷¹)
- Major cleanup of codebase: add and enforce pre-commit checks (#238⁷²)
- Bump stefanzweifel/git-auto-commit-action from 4 to 5 in the actions group by @dependabot[bot]⁷³ (#237⁷⁴)
- Code cleanups: address warnings and add pre-commit checks (#236⁷⁵)
- Move codeql tests to ubuntu-latest (#234⁷⁶)
- Update numpy requirement from `<=2.2.0` to `<=2.2.1` by @dependabot[bot]⁷⁷ (#232⁷⁸)

- Update numpy requirement from $\leq 2.1.3$ to $\leq 2.2.0$ by @dependabot[bot]⁷⁹(#231⁸⁰)
- Bump pypa/cibuildwheel from 2.21.3 to 2.22.0 in the actions group by @dependabot[bot]⁸¹(#230⁸²)
- Bump actions/setup-python from 4 to 5 in the actions group by @dependabot[bot]⁸³(#229⁸⁴)

Documentation

- Update all HLA nomenclature to correct version 3 allele names in unit tests and documentation (#251⁸⁵)
- Update documentation for makeNewPops filter (#247⁸⁶)
- Documentation updates for [Sequence] filters: (#244⁸⁷)
- Add information on pre-commit checks (#240⁸⁸)

Contributors:: @alexlanaster

1.1.2] - 2024-11-18

Features

- Add new command-line option --citation for installed version (#228⁸⁹)
- enable PyPy 3.10 as numpy wheels now exist (#219⁹⁰)

Internal

- Update numpy requirement from $\leq 2.1.2$ to $\leq 2.1.3$ by @dependabot (#226⁹¹)
- update x86 runner to macos-13, pin swig version (#227⁹²)
- Bump pypa/cibuildwheel from 2.21.2 to 2.21.3 in the actions group by @dependabot (#225⁹³)
- Bump pypa/cibuildwheel from 2.21.1 to 2.21.2 in the actions group by @dependabot (#224⁹⁴)
- Update numpy requirement from $\leq 2.1.1$ to $\leq 2.1.2$ by @dependabot (#223⁹⁵)
- Bump pypa/cibuildwheel from 2.20.0 to 2.21.1 in the actions group across 1 directory by @dependabot (#222⁹⁶)

Documentation

- Add new command-line option --citation for installed version (#228⁹⁷)
- Update to latest sphinx and add subtitle (#220⁹⁸)

Contributors:: @alexlanaster

1.1.1] - 2024-09-10

Features

- Enable support for Python 3.13 (#217⁹⁹)

Bug Fixes

- Pin gsl bottles to 2.7.1 on macOS to preserve 10.15/Catalina on x86 and 11.0/Big Sur compatibility + update cibuildwheels (#212¹⁰⁰)

Internal

- Update numpy requirement from $\leq 2.1.0$ to $\leq 2.1.1$ by @dependabot (#218¹⁰¹)
- Update numpy requirement from $\leq 2.0.1$ to $\leq 2.1.0$ by @dependabot (#216¹⁰²)
- Update lxml requirement from $\leq 5.2.2$ to $\leq 5.3.0$ by @dependabot (#215¹⁰³)
- Bump pypa/cibuildwheel from 2.19.2 to 2.20.0 in the actions group by @dependabot (#214¹⁰⁴)
- Update numpy requirement from $\leq 2.0.0$ to $\leq 2.0.1$ by @dependabot (#213¹⁰⁵)
- Update numpy requirement from $\leq 1.26.4$ to $\leq 2.0.0$ by @dependabot (#209¹⁰⁶)
- Bump pypa/cibuildwheel from 2.19.0 to 2.19.1 in the actions group by @dependabot (#210¹⁰⁷)
- Bump pypa/cibuildwheel from 2.18.1 to 2.19.0 in the actions group by @dependabot (#208¹⁰⁸)

1.1.0] - 2024-05-30

This release increases the minimum macOS requirements to Catalina (Intel) and Big Sur (Silicon) to ensure binary compatibility with the GNU Scientific Library (gs1) on those platforms. Thanks to @sjmack for testing.

Internal

- Bump cibuildwheel from 2.17.0 to 2.18.1, increase minimum macOS requirements by @dependabot (#206¹⁰⁹)
- Update lxml requirement from $\leq 5.2.1$ to $\leq 5.2.2$ by @dependabot (#205¹¹⁰)
- Bump peaceiris/actions-gh-pages from 3 to 4 in the actions group by @dependabot (#201¹¹¹)
- Update lxml requirement from $\leq 5.2.0$ to $\leq 5.2.1$ by @dependabot (#202¹¹²)
- Update lxml requirement from $\leq 5.1.0$ to $\leq 5.2.0$ by @dependabot (#200¹¹³)
- Enable native MacOS Apple Silicon runners (#199¹¹⁴)
- Update cibuildwheel GitHub action to 2.17.0 by @dependabot (#198¹¹⁵)
- Update softprops/action-gh-release github action that uploads builds to releases by @dependabot (#197¹¹⁶)

Documentation

- fix reversed links (#203) (#204¹¹⁷)

Contributors:: @alexlanaster

1.0.2] - 2024-02-24

Bug Fixes

- Synchronize with upstream haplo.stats, fix some redundant checks (#196¹¹⁸)

Internal

- customize code security scanning for C extensions (#195¹¹⁹)
- Update numpy requirement from $\leq 1.26.3$ to $\leq 1.26.4$ by @dependabot (#193¹²⁰)

Documentation

- Documentation updates including security policy (#194¹²¹)

Contributors:: @alexlanaster

1.0.1] - 2024-02-11

Features

- Add [CustomBinning] filtering unit tests for G and P-codes (#186¹²²)

Bug Fixes

- switch to scientific notation when frequencies can't be displayed as decimals (#192¹²³)
- Port [RandomBinningFilter] to Python 3, include more complex filtering tests (#187¹²⁴)

Internal

- Bump the cibuildwheel version from 2.16.4 to 2.16.5: fixes Windows CI builds by @dependabot (#189¹²⁵)
- Bump the version of cibuildwheel from 2.16.2 to 2.16.4 by @dependabot (#188¹²⁶)
- increase test strictness: make test warnings into errors (#185¹²⁷)
- Enable wheels on aarch64 architecture (#184¹²⁸)
- Update actions/upload-artifact from 3 to 4 in Build on ARM64 by @dependabot (#183¹²⁹)
- Streamline continuous integration: reduce number of wheels, concurrency (#182¹³⁰)
- Parallelize wheel builds, re-enable musllinux wheels for Python 3.9+ (#181¹³¹)
- Update lxml requirement from <=5.0.0 to <=5.1.0; disable PyPy 3.7 on Linux by @dependabot (#178¹³²)
- Update numpy requirement from <=1.26.2 to <=1.26.3 by @dependabot (#177¹³³)
- Update lxml requirement from <=4.9.4 to <=5.0.0 by @dependabot (#174¹³⁴)
- Update lxml requirement from <=4.9.3 to <=4.9.4 by @dependabot (#173¹³⁵)
- update to v4 of download-artifact / upload-artifact (#172¹³⁶)
- Bump actions/setup-python from 4 to 5 by @dependabot (#168¹³⁷)
- Update numpy requirement from <=1.26.1 to <=1.26.2 by @dependabot (#167¹³⁸)

Documentation

- Link to new preprint in docs (#190¹³⁹)
- Convert bibliography to bibtex (#176¹⁴⁰)
- Convert NEWS.rst to NEWS.md, improve PDF documentation output (#175¹⁴¹)

Contributors:: @alexlanaster

1.0.0] - 2023-11-07

PyPop 1.0.0 is the first official release of PyPop using Python 3, and the first release to be included on PyPI¹⁴². In addition to using modern libraries, there are some new features, such as the new asymmetric LD measures, and better handling of TSV files, along with the typical slew of bug fixes. Many more changes are of an “under the hood” nature, such as a new unit testing and documentation framework, and are detailed below. Many people contributed to this latest release, which has been a while in coming. Thanks especially to all new contributors including Vanessa Sochat, Gordon Webster, Jurriaan H. Spaaks, Karl Kornel and Michael Mariani. Thanks also to all of our bug reporters, and ongoing contributors, especially Richard Single, Owen Solberg and Steve Mack.

New features

- PyPop now fully ported to run under Python 3 (thanks to Vanessa Sochat for major patch)
- Added new asymmetric linkage disequilibrium (ALD) calculations (thanks to Richard Single), see [Thomson & Single, 2014](#)¹⁴³ for more details. Added in both the plain text (.txt) as well as the 2-locus-summary.tsv TSV file outputs.
- Improved tab-separated values (TSV) output file handling:
 - old IHWG headers are disabled by default, so the `-disable-ihwg` option has been replaced by the `--enable-ihwg` option, which will re-enable them.
 - `popmeta`: allow TSV files to be put in separate directory with `-o/ --outputdir` command-line option for saving generated files.
 - `pypop`: renamed `--generate-tsv` to `--enable-tsv`
 - dynamic generation of TSV files based on XML input, so the list of files is no longer hard-coded (thanks to Steve Mack for suggestion), this also adds support for haplotypes involving more than 4 loci
- Preliminary support for Genotype List (GL) String (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3715123/>)
- Added unit-tests using `pytest` testing framework.
- New documentation system using `sphinx`, replacing the old DocBook XML, to generate both the website and the *PyPop User Guide* (HTML and PDF):
 - `sphinx`-based documentation is now written in ReStructuredText (.rst)
 - improve `popmeta` and other documentation for command-line programs
 - documentation additions and improvements from Richard Single, Michael Mariani, Gordon Webster and Steve Mack
 - overhaul release process and add a contribution-guide to the *PyPop User Guide*.
 - update documentation to use new HLA nomenclature throughout
- PyPop now uses `numpy` in place of the old Numeric library `Numpy`, and `lxml` in place of `libxml_mod`

Bug fixes

- TSV file fixes:
 - fix missing columns in TSV files (thanks to Steve Mack for report)
 - fix headers in 3 and 4 locus TSV files
 - output 2 locus haplotypes in TSV if they are explicitly specified (thanks to Steve Mack)
 - 2-locus-haplo.tsv: fixed missing output in `ld.d`, `ld.dprime`, `ld.chisq`, and `exp` columns (thanks to Nabil M for the report)
 - rename, remove and add some columns headers, including the new ALD measures:
 - * 2-locus-haplo.tsv: rename columns: `allele` -> `haplotype`, `exp` -> `haplotype.no-ld.count`
 - * 2-locus-haplo.tsv: remove `obs` and `obs.freq` columns which were duplicative of `haplotype.count` and `haplotype.freq`, respectively
 - * 2-locus-summary.tsv: add two new ALD measure columns: `ald.1_2` and `ald.2_1`
 - * *-locus-summary.tsv: rename columns for multilocus haplotypes for 3 or more loci, `allele` -> `haplotype` and `locus` -> `loci`
- Fix `DigitBinning` and `CustomBinning` filters [Filters] (report from Steve Mack)
- Fix issues with using colons in alleles, and other separation issues (thanks to Steve Mack)

- Use ~ as the genotype terminator rather than | (fixes some haplotype estimation bugs)
- Round all haplotype frequencies to 5 decimal places to avoid truncation issues (thanks to Steve Mack for report)
- Restore semi-GUI interactive mode by using built-in TkInter file dialog, and use more informative default “placeholder” file names
- Fix warnings generated by `numpy` and `re` libraries.
- Windows fixes:
 - `Emhaplofreq` will now give identical results on Windows as all other platform (needed to port POSIX-version of `drand48()` to Windows).
 - Fixed `CustomBinning` filters that were failing on Windows.
 - Enable all unit tests for Windows.

Internal

- Replace old `getopt` with `argparse` library
- Major code refactoring, including moving code into `src` directory, and using packages in `setup.py`
- Added continuous integration via GitHub Actions for releases and website updates
- Prepare package for inclusion in PyPI
- Add code examples used in documentation as unit tests
- Create GitHub action for upload to Zenodo (thanks to Jurriaan H. Spaaks)
- Support for arm64 builds on MacOS, e.g. M1-based Macs (thanks to Owen Solberg for report and extensive testing).
- Remove dependency on `psutil`, rarely needed.
- Only build wheels on platforms for which binary wheels are available for all dependencies.

0.7.0] - 2008-09-09

New features

- `makeNewPopFile` option has been changed. This option allows user to generate intermediate output of filtered files. Now option should be of the format: `type:order` where `type` is one of `separate-loci` or `all-loci` so that the user can specify whether a separate file should be generated for each locus (`separate-loci`) or a single file with all loci (`all-loci`). `order` should be the order in the filtering chain where the matrix is generated, there is no default, for example, for generating files after the first filter operation use 1.
- New command-line option `--generate-tsv`, will generate the `.dat` tab-separated values (TSV) files on the the generated `-out.xml` files (aka “popmeta”) directly from `pypop` without needing to run additional script. Now output from `pypop` can be directly read into spreadsheet.
- New feature: add individual genotype tests to Hardy-Weinberg module (`gthwe`), now computes statistics based on individual genotypes in the HWP table. The `[HardyWeinbergGuoThompson]` or `[HardyWeinbergGuoThompsonMonteCarlo]` options must be enabled in the configuration “.ini” file in order for these tests to be carried out.
- Major improvements to custom and random binning filters (Owen Solberg).
- New feature: generate homozygosity values using the Ewens-Watterson test for all pairwise loci, or all sites within a gene for sequence data (`[homozygosityEWSlatkinExactPairwise]` in .ini file). Note: this really only works for sequence data where the phase for sites within an allele are known.
- Haplotype and LD estimation module `emhaplofreq` improvements
 - improved memory usage and speed for `emhaplofreq` module.

- maximum sample size for emhaplofreq module increased from 1023 to 5000 individuals.
- maximum length of allele names increased to 20

Bug fixes

- Support Python 2.4 on GCC 4.0 platforms.
- Add missing initialisation for non-sequence data when processing haplotypes. Thanks to Jill Hollenbach for the report.
- Fix memory leak in xslt translation.
- Various fixes relating to parsing XML output.
- Fixed an incorrect parameter name.
- Handle some missing sections in .ini better. Thanks to Owen Solberg for report.
- Various build and installation fixes (SWIG, compilation flags)
- Make name of source package be lowercase “pypop”.
- Change data directory: /usr/share/pypop/ to /usr/share/PyPop/
- Print out warning when maximum length of allele exceeded, rather than crashing. Thanks to Steve Mack for report.

Other issues

- Sequence filter
 - In the Sequence filter, add special case for Anthony Nolan HLA data: mark null alleles ending in “N” (e.g. HLA-B*5127N) as “missing data” (****).
 - Also in Sequence, keep track of unsequenced sites separately (via unsequencedSites variable) from “untyped” (aka “missing data”). Treat unsequencedSite as a unique allele to make sure that those sites don’t get treated as having a consensus sequence if only one of the sequences in the the set of matches is typed.
 - If no matching sequence is found in the MSF files, then return a sequence of * symbols (ie, will be treated as truly missing data, not untyped alleles.
 - Add another special case for HLA data: test for 7 digits in allele names (e.g. if 2402101 is not found insert a zero after the first 4 digits to form 24020101, and check for that). This is to cope with yet-another HLA nomenclature change.
- Change semantics of batchsize, make “0” (default) process files separately if only R dat files is enabled. If batchsize not set explicitly (and therefore 0) set batchsize to 1 is PHYLIP mode is enabled.

0.6.0 - 2005-04-13

New features

- Allow for odd allele counts when processing an allele count data (i.e “semi”-typing). When PyPop is dealing with data that is originally genotyped, the current default is preserved i.e. we dis-allow individuals that are typed at only allele, and set allowSemiTyped to false.
- New command-line option -f (long version --filelist) which accepts a file containing a list of files (one per line) to process (note that this is mutually exclusive with supplying INPUTFILES, and will abort with an error message if you supply both simultaneously).
- In batch version, handle multiple INPUTFILES supplied as command-line arguments and support Unix shell-globbing syntax (e.g. pypop.py -c config.ini *.pop). (NOTE: This is supported *only* in batch version, not in the interactive version, which expects one and only one file supplied by user.

- Allele count files can now be filtered through the filter apparatus (particularly the Sequence and AnthonyNolan) in the same way as genotype files transparently. [This has been enabled via a code refactor that treats allele count files as pseudo-genotype files for the purpose of filtering]. This change also resulted in the removal of the obsolete lookup-table-based homozygosity test.
- Add `--disable-ihwg` option to `popmeta` script to disable hardcoded generation of the IHWG header output, and use the output as defined in the header in the original `.pop` input text file. This is disabled by default to preserve backwards compatibility.
- Add `--batchsize` (`-b` short version) option for `popmeta`. Does the processing in “batches”. If set and greater than one, list of XML files is split into batchsize group. For example, if there are 20 XML files and option is via using (“`-b 2`” or “`--batchsize=2`”) then the files will be processed in two batches, each consisting of 10 files. If the number does not divide evenly, the last list will contain all the “left-over” files. This option is particularly useful with large XML files that may not fit in memory all at once. Note this option is mutually exclusive with the `--enable-PHYLIP` option because the PHYLIP output needs to calculate allele frequencies across all populations before generating files.
- New `.ini` file option: [`HardyWeinbergGuoThompsonMonteCarlo`]: add a plain Monte-Carlo (randomization, without the Markov chain test) test for the HardyWeinberg “exact test”. Add code for Guo & Thompson test to distribution (now under GNU GPL).

Bug fixes

- HardyWeinbergGuoThompson overall p-value test was numerically unstable because it attempted to check for equality in greater than or equal to constructs (“`<=`”) which is not reliable in C. Replaced this with a GNU Scientific Library (GSL) function `gsl_fcmp()` which compares floats to within an EPSILON (defaults to `1e-6`).
- Allow HardyWeinbergGuoThompson test to be run if at least two alleles present (test was originally failing with a `too-few-alleles` message if there were not at least 3 alleles). Thanks to Kristie Mather for the report.
- Checks to see if a locus is monomorphic, if it is, it generates an allele summary report, but skips the rest of the single locus analyses which do not make sense for monomorphic locus. Thanks to Steve Mack and Owen Solberg for the bug report(s).
- Now builds against recent versions of SWIG (no longer stuck at version 1.3.9), should be compatible with versions of SWIG > 1.3.10. (Tested against SWIG 1.3.21).
- Homozygosity module: Prevent math errors by in Slatkin’s exact test by forcing the homozygosity to be positive (only a problem for rare cases, when the result is so close to zero that the floating point algorithms cause a negative result.)

0.5.2 (public beta) - 2004-03-09

Bug fixes

- Add missing `RandomBinning.py` file to source distribution Thanks to Hazael Maldonado Torres for the bug report.
- Fixed line endings for `.bat` scripts for Win32 so they work under Windows 98 thanks to Wendy Hartogensis for the bug report.

0.5.1 (public beta) - 2004-02-26

Changes

- New parameter `numInitCond`, number of initial conditions by the haplotype estimation and LD algorithm used before performing permutations. Defaults to 50.
- Remove some LOG messages/diagnostics that were erroneously implying an error to the user (if nothing is wrong, don’t say anything). Add some more useful messages for what is being done in haplo/LD estimation step.
- Add `popmeta.py` to the distribution: this is undocumented and unsupported as yet, it is at alpha stage only, use at your own risk!

Bug fixes

- Remember to output plaintext version of LD for specified loci.

0.5 (public beta) - 2003-12-31

Changes

- All Linux wrapper scripts no longer have .sh file suffixes for consistency with DOS (all DOS bat files can be executed without specifying the .bat extension).

Bug fixes

- Add wrapper scripts for interactive and batch mode for both DOS and Linux so that correct shared libraries are called.
- Pause and wait for user to press a key at end of DOS .bat file so that output can be viewed before window close.
- Set PYTHONHOME in wrapper scripts to prevent messages about missing <prefix> being displayed.

0.4.3beta

Bug fixes

- Fixed bug in processing of popname field. Thanks to Richard Single for the report.

Notes

1. <https://orcid.org/0000-0001-5847-5330>
2. <https://orcid.org/0000-0002-0002-9263>
3. <https://orcid.org/0000-0001-9820-9547>
4. <https://orcid.org/0000-0001-5852-0517>
5. <https://orcid.org/0000-0002-7155-5674>
6. <https://orcid.org/0000-0001-6054-6505>
7. <https://orcid.org/0000-0002-4387-3819>
8. <https://orcid.org/0000-0003-3060-9709>
9. <https://orcid.org/0000-0002-7064-4069>
10. <https://orcid.org/0000-0001-5235-4159>
11. <https://github.com/yystsai>
12. <https://orcid.org/0009-0006-0162-6066>
13. <https://orcid.org/0009-0009-2862-0467>
14. <http://www.gnu.org/licenses/gpl.html>
15. <https://github.com/alexlancaster/pypop/pull/326>
16. <https://github.com/alexlancaster/pypop/pull/328>
17. <https://github.com/alexlancaster/pypop/pull/324>
18. <https://github.com/alexlancaster/pypop/pull/322>
19. <https://github.com/alexlancaster/pypop/pull/321>
20. <https://github.com/alexlancaster/pypop/pull/318>
21. <https://github.com/alexlancaster/pypop/pull/319>
22. <https://github.com/alexlancaster/pypop/pull/317>
23. <https://github.com/alexlancaster/pypop/pull/314>
24. <https://github.com/alexlancaster/pypop/pull/315>
25. <https://github.com/alexlancaster/pypop/pull/312>
26. <https://github.com/alexlancaster/pypop/pull/309>
27. <https://github.com/alexlancaster/pypop/pull/310>

28. <https://github.com/alexlancaster/pypop/pull/307>
29. <https://github.com/alexlancaster/pypop/pull/306>
30. <https://github.com/alexlancaster/pypop/pull/303>
31. <https://github.com/alexlancaster/pypop/pull/299>
32. <https://github.com/alexlancaster/pypop/pull/300>
33. <https://github.com/alexlancaster/pypop/pull/298>
34. <https://github.com/alexlancaster/pypop/pull/291>
35. <https://github.com/alexlancaster/pypop/pull/295>
36. <https://github.com/alexlancaster/pypop/pull/292>
37.] <https://github.com/alexlancaster/pypop/pull/294>
38.] <https://github.com/alexlancaster/pypop/pull/293>
39.] <https://github.com/alexlancaster/pypop/pull/303>
40.] <https://github.com/alexlancaster/pypop/pull/296>
41.] <https://github.com/alexlancaster/pypop/pull/286>
42.] <https://github.com/alexlancaster/pypop/pull/283>
43.] <https://github.com/alexlancaster/pypop/pull/282>
44.] <https://github.com/alexlancaster/pypop/pull/257>
45.] <https://github.com/alexlancaster/pypop/pull/272>
46.] <https://github.com/alexlancaster/pypop/pull/256>
47.] <https://github.com/alexlancaster/pypop/pull/280>
48.] <https://github.com/alexlancaster/pypop/pull/279>
49.] <https://github.com/alexlancaster/pypop/pull/275>
50.] <https://github.com/alexlancaster/pypop/pull/273>
51.] <https://github.com/alexlancaster/pypop/pull/272>
52.] <https://github.com/alexlancaster/pypop/pull/270>
53.] <https://github.com/alexlancaster/pypop/pull/269>
54.] <https://github.com/alexlancaster/pypop/pull/268>
55.] <https://github.com/alexlancaster/pypop/pull/265>
56.] <https://github.com/alexlancaster/pypop/pull/264>
57.] <https://github.com/alexlancaster/pypop/pull/261>
58.] <https://github.com/alexlancaster/pypop/pull/259>
59.] <https://github.com/alexlancaster/pypop/pull/260>
60.] <https://github.com/alexlancaster/pypop/pull/252>
61.] <https://github.com/alexlancaster/pypop/pull/250>
62.] <https://github.com/alexlancaster/pypop/pull/235>
63.] <https://github.com/alexlancaster/pypop/pull/251>
64.] <https://github.com/alexlancaster/pypop/pull/247>
65.] <https://github.com/alexlancaster/pypop/pull/244>
66.] <https://github.com/alexlancaster/pypop/pull/233>
67.] <https://github.com/alexlancaster/pypop/pull/253>
68.] <https://github.com/apps/dependabot>
69.] <https://github.com/alexlancaster/pypop/pull/245>
70.] <https://github.com/alexlancaster/pypop/pull/242>
71.] <https://github.com/alexlancaster/pypop/pull/239>
72.] <https://github.com/alexlancaster/pypop/pull/238>
73.] <https://github.com/apps/dependabot>
74.] <https://github.com/alexlancaster/pypop/pull/237>

- 75.] <https://github.com/alexlancaster/pypop/pull/236>
- 76.] <https://github.com/alexlancaster/pypop/pull/234>
- 77.] <https://github.com/apps/dependabot>
- 78.] <https://github.com/alexlancaster/pypop/pull/232>
- 79.] <https://github.com/apps/dependabot>
- 80.] <https://github.com/alexlancaster/pypop/pull/231>
- 81.] <https://github.com/apps/dependabot>
- 82.] <https://github.com/alexlancaster/pypop/pull/230>
- 83.] <https://github.com/apps/dependabot>
- 84.] <https://github.com/alexlancaster/pypop/pull/229>
- 85.] <https://github.com/alexlancaster/pypop/pull/251>
- 86.] <https://github.com/alexlancaster/pypop/pull/247>
- 87.] <https://github.com/alexlancaster/pypop/pull/244>
- 88.] <https://github.com/alexlancaster/pypop/pull/240>
- 89.] <https://github.com/alexlancaster/pypop/pull/228>
- 90.] <https://github.com/alexlancaster/pypop/pull/219>
- 91.] <https://github.com/alexlancaster/pypop/pull/226>
- 92.] <https://github.com/alexlancaster/pypop/pull/227>
- 93.] <https://github.com/alexlancaster/pypop/pull/225>
- 94.] <https://github.com/alexlancaster/pypop/pull/224>
- 95.] <https://github.com/alexlancaster/pypop/pull/223>
- 96.] <https://github.com/alexlancaster/pypop/pull/222>
- 97.] <https://github.com/alexlancaster/pypop/pull/228>
- 98.] <https://github.com/alexlancaster/pypop/pull/220>
- 99.] <https://github.com/alexlancaster/pypop/pull/217>
- 100.] <https://github.com/alexlancaster/pypop/pull/212>
- 101.] <https://github.com/alexlancaster/pypop/pull/218>
- 102.] <https://github.com/alexlancaster/pypop/pull/216>
- 103.] <https://github.com/alexlancaster/pypop/pull/215>
- 104.] <https://github.com/alexlancaster/pypop/pull/214>
- 105.] <https://github.com/alexlancaster/pypop/pull/213>
- 106.] <https://github.com/alexlancaster/pypop/pull/209>
- 107.] <https://github.com/alexlancaster/pypop/pull/210>
- 108.] <https://github.com/alexlancaster/pypop/pull/208>
- 109.] <https://github.com/alexlancaster/pypop/pull/206>
- 110.] <https://github.com/alexlancaster/pypop/pull/205>
- 111.] <https://github.com/alexlancaster/pypop/pull/201>
- 112.] <https://github.com/alexlancaster/pypop/pull/202>
- 113.] <https://github.com/alexlancaster/pypop/pull/200>
- 114.] <https://github.com/alexlancaster/pypop/pull/199>
- 115.] <https://github.com/alexlancaster/pypop/pull/198>
- 116.] <https://github.com/alexlancaster/pypop/pull/197>
- 117.] <https://github.com/alexlancaster/pypop/pull/204>
- 118.] <https://github.com/alexlancaster/pypop/pull/196>
- 119.] <https://github.com/alexlancaster/pypop/pull/195>
- 120.] <https://github.com/alexlancaster/pypop/pull/193>
- 121.] <https://github.com/alexlancaster/pypop/pull/194>

- 122.] <https://github.com/alexlancaster/pypop/pull/186>
- 123.] <https://github.com/alexlancaster/pypop/pull/192>
- 124.] <https://github.com/alexlancaster/pypop/pull/187>
- 125.] <https://github.com/alexlancaster/pypop/pull/189>
- 126.] <https://github.com/alexlancaster/pypop/pull/188>
- 127.] <https://github.com/alexlancaster/pypop/pull/185>
- 128.] <https://github.com/alexlancaster/pypop/pull/184>
- 129.] <https://github.com/alexlancaster/pypop/pull/183>
- 130.] <https://github.com/alexlancaster/pypop/pull/182>
- 131.] <https://github.com/alexlancaster/pypop/pull/181>
- 132.] <https://github.com/alexlancaster/pypop/pull/178>
- 133.] <https://github.com/alexlancaster/pypop/pull/177>
- 134.] <https://github.com/alexlancaster/pypop/pull/174>
- 135.] <https://github.com/alexlancaster/pypop/pull/173>
- 136.] <https://github.com/alexlancaster/pypop/pull/172>
- 137.] <https://github.com/alexlancaster/pypop/pull/168>
- 138.] <https://github.com/alexlancaster/pypop/pull/167>
- 139.] <https://github.com/alexlancaster/pypop/pull/190>
- 140.] <https://github.com/alexlancaster/pypop/pull/176>
- 141.] <https://github.com/alexlancaster/pypop/pull/175>
- 142.] <https://pypi.org/project/pypop-genomics/>
- 143.] <https://doi.org/10.1534/genetics.114.165266>

LICENSES

6.1 License terms for PyPop

Except where noted in the source, PyPop is distributed under the terms of the GNU General Public License¹(*GNU General Public License*). The list of authors and contributors is located in *PyPop contributors*.

PyPop: Python for Population Genomics

Copyright © 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009 by The Regents of the University of California (Regents). All rights reserved.

All code in this program is copyright the Regents (except where otherwise explicitly mentioned).

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2² of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License (*GNU General Public License*) for more details.

GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

(continues on next page)

(continued from previous page)

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in

(continues on next page)

(continued from previous page)

whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

(continues on next page)

(continued from previous page)

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the

(continues on next page)

(continued from previous page)

original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This program is free software; you can redistribute it and/or modify
```

(continues on next page)

(continued from previous page)

it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

6.2 License for PyPop documentation

GNU Free Documentation License

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4. above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Notes

1.] <https://www.gnu.org/licenses/gpl.html>
2.] <https://www.gnu.org/licenses/gpl-2.0.html>

BIBLIOGRAPHY

- [CKM+07] Cano, P., Klitz, W., Mack, S. J., Maiers, M., Marsh, S. G. E., Noreen, H., Reed, E. F., Senitzer, D., Setterholm, M., Smith, A., and Fernández-Viña, M. Common and well-documented HLA alleles: report of the Ad-Hoc committee of the American Society for Histocompatibility and Immunogenetics. *Human Immunology*, 68(5):392–417, 2007. doi:10.1016/j.humimm.2007.01.014³.
- [CHT+99] Chen, J. J., Hollenbach, J. A., Trachtenberg, E. A., Just, J. J., Carrington, M., Ronningen, K. S., Begovich, A. B., King, M.-C., McWeeney, S. K., Mack, S. J., Erlich, H. A., and Thomson, G. Hardy-Weinberg testing for HLA class II (DRB1, DQA1, DQB1, and DPB1) loci in 26 human ethnic groups. *Tissue Antigens*, 54(6):533–542, 1999. doi:10.1034/j.1399-0039.1999.540601.x⁴.
- [Cramer46] Cramér, H. *Mathematical Methods of Statistics*. Princeton University Press, Princeton, NJ, 1946. ISBN 978-0-691-00547-8.
- [DLR77] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–38, 1977. doi:10.1111/j.2517-6161.1977.tb01600.x⁵.
- [Ewe72] Ewens, W. J. The sampling theory of selectively neutral alleles. *Theoretical Population Biology*, 3(1):87–112, 1972. doi:10.1016/0040-5809(72)90035-4⁶.
- [EL10] Excoffier, L. and Lischer, H. E. L. Arlequin suite ver 3.5: a new series of programs to perform population genetics analyses under Linux and Windows. *Molecular Ecology Resources*, 10(3):564–567, 2010. doi:10.1111/j.1755-0998.2010.02847.x⁷.
- [ES95] Excoffier, L. and Slatkin, M. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–927, 1995. doi:10.1093/oxfordjournals.molbev.a040269⁸.
- [GT92] Guo, S. W. and Thompson, E. A. Performing the exact test of Hardy-Weinberg proportion for multiple alleles. *Biometrics*, 48(2):361–372, 1992. arXiv:2532296⁹, doi:10.2307/2532296¹⁰.
- [Hed87] Hedrick, P. W. Gametic disequilibrium measures: proceed with caution. *Genetics*, 117(2):331–341, 1987. doi:10.1093/genetics/117.2.331¹¹.
- [LNM+03] Lancaster, A., Nelson, M. P., Meyer, D., Single, R. M., and Thomson, G. PyPop: a software framework for population genomics: analyzing large-scale multi-locus genotype data. *Pacific Symposium on Biocomputing*, pages 514–525, 2003. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3891851/> (accessed on 2024-01-04)
- [LNS+07] Lancaster, A. K., Nelson, M. P., Single, R. M., Meyer, D., and Thomson, G. Software framework for the Biostatistics Core of the International Histocompatibility Working Group. In Hansen, J. A., editor, *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*, volume I, pages 510–517. IHWG Press, Seattle, WA, 2007.

- [LSS+07] Lancaster, A. K., Single, R. M., Solberg, O. D., Nelson, M. P., and Thomson, G. PyPop update – a software pipeline for large-scale multilocus population genomics. *Tissue Antigens*, 69(s1):192–197, 2007. doi:10.1111/j.1399-0039.2006.00769.x¹².
- [LSM+24] Lancaster, A. K., Single, R. M., Mack, S. J., Sochat, V., Mariani, M. P., and Webster, G. D. PyPop: A mature open-source software pipeline for population genomics. *Frontiers in Immunology*, 2024. doi:10.3389/fimmu.2024.1378512¹³.
- [MSanchezMazasM+07] Mack, S. J., Sanchez-Mazas, A., Meyer, D., Single, R. M., Tsai, Y., and Erlich, H. A. Methods used in the generation and preparation of data for analysis in the 13th International Histocompatibility Workshop. In Hansen, J. A., editor, *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*, volume I, pages 564–579. IHWG Press, Seattle, WA, 2007.
- [MSM+07] Meyer, D., Single, R. M., Mack, S. J., Lancaster, A. K., Nelson, M. P., Erlich, H. A., Fernández-Viña, M., and Thomson, G. Single locus polymorphism of classical HLA genes. In Hansen, J. A., editor, *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*, volume I, pages 653–704. IHWG Press, Seattle, WA, 2007.
- [SKE+99] Salamon, H., Klitz, W., Eastel, S., Gao, X., Erlich, H. A., Fernandez-Viña, M., Trachtenberg, E. A., McWeeney, S. K., Nelson, M. P., and Thomson, G. Evolution of HLA class II molecules: Allelic and amino acid site variability across populations. *Genetics*, 152(1):393–400, 1999. doi:10.1093/genetics/152.1.393¹⁴.
- [SKRE00] Schneider, S., Kueffer, J.-M., Roessli, D., and Excoffier, L. Arlequin: A software for population genetics data analysis. Ver 2.000. Genetics and Biometry Lab, Dept. of Anthropology, University of Geneva, 2000. URL: <http://cmpg.unibe.ch/software/arlequin/> (accessed on 2024-01-06)
- [SMM+07] Single, R. M., Meyer, D., Mack, S. J., Lancaster, A. K., Nelson, M. P., Erlich, H. A., Fernández-Viña, M., and Thomson, G. Haplotype frequencies and linkage disequilibrium among classical HLA genes. In Hansen, J. A., editor, *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*, volume I, pages 705–746. IHWG Press, Seattle, WA, 2007.
- [SMT07] Single, R. M., Meyer, D., and Thomson, G. Statistical methods for analysis of population genetic data. In Hansen, J. A., editor, *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*, volume I, pages 518–522. IHWG Press, Seattle, WA, 2007.
- [Sla94] Slatkin, M. An exact test for neutrality based on the Ewens sampling distribution. *Genetics Research*, 64(1):71–74, 1994. doi:10.1017/S0016672300032560¹⁵.
- [Sla96] Slatkin, M. A correction to the exact test based on the Ewens sampling distribution. *Genetics Research*, 68(3):259–260, 1996. doi:10.1017/S0016672300034236¹⁶.
- [SML+08] Solberg, O. D., Mack, S. J., Lancaster, A. K., Single, R. M., Tsai, Y., Sanchez-Mazas, A., and Thomson, G. Balancing selection and heterogeneity across the classical human leukocyte antigen loci: A meta-analytic review of 497 population studies. *Human Immunology*, 69(7):443–464, 2008. doi:10.1016/j.humimm.2008.05.001¹⁷.
- [TS14] Thomson, G. and Single, R. M. Conditional asymmetric linkage disequilibrium (ALD): extending the bi-allelic r^2 measure. *Genetics*, 198(1):321–331, 2014. doi:10.1534/genetics.114.165266¹⁸.
- [Wat78] Watterson, G. A. The homozygosity test of neutrality. *Genetics*, 88(2):405–417, 1978. doi:10.1093/genetics/88.2.405¹⁹.