PyPop User Guide

Release 0.0

Alexander K. Lancaster
Mark P. Nelson
Diogo Meyer
Richard M. Single
Owen D. Solberg

Jul 25, 2023

PyPop User Guide

Document release: 0.0.post2310+ga91d9e6

CONTENTS

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).

It is an object-oriented framework implemented in Python (http://www.pypop.org/), a language with powerful features for interfacing with other languages, such as C (in which we have already implemented many routines and which is particularly suited to computationally intensive tasks).

The output of the analyses are stored in XML. These output files can then be transformed using standard tools into many other data formats suitable for machine input (such as PHYLIP or input for spreadsheet programs such as Excel or statistical packages, such as R), plain text, or HTML for human-readable format. Storing the output in XML allows the final viewable output format to be redesigned at will, without requiring the (often time-consuming) re-running of the analyses themselves.

An outline of PyPop can be found in our 2007 *Tissue Antigens* and 2003 *PSB* papers.

How to cite PyPop

When citing PyPop, please cite the (2007) paper from *Tissue Antigens*:

- A. K. Lancaster, R. M. Single, O. D. Solberg, M. P. Nelson and G. Thomson (2007) "PyPop update - a software pipeline for large-scale multilocus population genomics" *Tissue Antigens* 69 (s1), 192-197. [journal page (http://dx.doi.org/10.1111/j.1399-0039.2006.00769.x), preprint PDF (112 kB) (http://pypop.org/tissue-antigens-lancaster-2007.pdf)].

In addition, you can also cite our 2003 Pacific Symposium on Biocomputing paper:

- Alex Lancaster, Mark P. Nelson, Richard M. Single, Diogo Meyer, and Glenys Thomson (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 Central (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3891851/), PDF (344 kB) (http://pypop.org/psb-pypop.pdf)].

PyPop was originally developed for the analysis of data for the 13th International Histocompatiblity Workshop and Conference (http://www.ihwg.org/) held in Seattle, Washington in 2002 ([Meyer:etal:2007], [Single:etal:2007a], [Single:etal:2007a]). For more details on the design and technical details of PyPop, please consult [Lancaster:etal:2003], [Lancaster:etal:2007a] and [Lancaster:etal:2007b].

Acknowlegements

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 four 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.

CHAPTER

ONE

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 (https://pypi.org). This is a working name and may change and is not yet the final package name until the package is released to PyPI.

1.1 Quickstart Guide

Installing `pypop-genomics`

If you already have Python and `pip` *installed*, install a test pre-releases using the following:

```
pip install pypop-genomics --extra-index-url https://test.pypi.org/simple/
```

Warning: These pre-release versions are being made available for initial testing, they are not intended to be used for production applications or analysis, and are not yet included in the main pypi.org index

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

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.

1.2.1 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 1.0.0a23
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.

1.2.2 Run a minimal dataset:

Download test `.ini` and `.pop` files: minimal.ini (https://github.com/alexlancaster/pypop/blob/main/tests/data/minimal.ini) and USAFEL-UchiTelle-small.pop (https://github.com/alexlancaster/pypop/blob/main/tests/data/USAFEL-UchiTelle-small.pop). 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 Test PyPI
3. adjusting your `PATH` variable after installation

1.3.1 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 (https://docs.python.org/3/using/mac.html) 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 (https://docs.python.org/3/using/windows.html) 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.

1.3.2 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` pre-releases, test packages for PyPop available directly on the Test PyPI (https://test.pypi.org/).

Warning: These pre-release versions are being made available for initial testing, they are not intended to be used for production applications or analysis, and are not yet included in the main pypi.org index

```
pip install pypop-genomics --extra-index-url https://test.pypi.org/simple/
```

Note: If, for whatever reason, you cannot use the these binaries (e.g. the pre-compiled binaries are not available for your platform), you may need to follow the developer installation instructions (http://pypop.org/docs/guide-chapter-contributing.html#installation-for-developers) 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 --extra-index-url https://test.pypi.org/simple/
```

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

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 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/`.

1.3.3 Install package from GitHub Releases (advanced)

We also sometimes make binary packages also available from the GitHub release page:

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

To install these is similar to installing via PyPI above, 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-a23`).

   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 (https://semver.org/), which Python internally normalizes and abbreviates. So the release with the `git` tag `v1.0.0-a23` is actually version `1.0.0a23` of the `pypop-genomics` package, and the version that `pip` "sees".

   ---

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-a23
   ```

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

   ```
   pip install pypop-genomics-1.0.0a23-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_
   ↪64.whl
   ```

1.3.4 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 (https://www.computerhope.com/issues/ch000549.htm), this may need additional help from your system administrator if your user doesn't have the right permissions, and also require a system reboot.

1.3.5 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 (http://pypop.org/docs/guide-chapter-contributing.html#reporting-and-requesting) 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

CHAPTER

TWO

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) (https://github.com/alexlancaster/pypop/blob/main/tests/data/USAFEL-UchiTelle-small.pop), and a population file (.pop) (https://github.com/alexlancaster/pypop/blob/main/tests/data/minimal.ini), can be found by clicking the respective links.

There are two ways to run PyPop:

- interactive mode (where the program will prompt you to directly type the input it needs); and
- batch mode (where you supply all the command line options the program needs).

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 batch 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).

2.1.1 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 (1.0.0a15)
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.

```

(continues on next page)

(continued from previous page)

```
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 (1.0.0a15)
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.

2.1.2 Batch mode

To run PyPop in the more common "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
```

Note: If your system administrator has installed PyPop the name of the script may be renamed to something different.

Batch mode assumes two things: that you have a file called `config.ini` in your current folder and that you also have your population file is 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
```

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
```

Please see *pypop usage* for the full list of command-line options.

2.1.3 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 batch 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 (`popmeta.bat` on Windows, `popmeta` on GNU/Linux) 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 `.dat` files, including `1-locus-allele.dat`.

Note: It's highly recommended to use the `-o` option to save the output in a separate subdirectory, as the output `.dat` 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.

2.3.1 pypop usage

```
usage: pypop [-h] [-o OUTPUTDIR] [-V] [-c CONFIG] [-m] [-d] [-x XSLFILE] [-t] [--enable-ihwg]␣
↪[--enable-phylip] [-i] [-f FILELIST]
            [POPFILE ...]
```

Options for pypop

-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-` flags only valid if `--enable-tsv/-t` selected

-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

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 (mutually exclusive with supplying POPFILEs)

POPFILE input population (`.pop`) file(s)

Default: []

2.3.2 `popmeta` usage

```
usage: popmeta [-h] [-o OUTPUTDIR] [-V] [--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

-o, --outputdir put output in directory OUTPUTDIR

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

--disable-tsv disable generation of `.dat` 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

2.4.1 Sample files

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

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 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 through to Listing 2.6.

Listing 2.1: Multi-locus allele-level genotype data

```
a_1   a_2   c_1   c_2   b_1   b_2
****  ****  0102  02025 1301  18012
0101  0201  0307  0605  1401  39021
0210  03012 0712  0102  1520  1301
0101  0218  0804  1202  35091 4005
2501  0201  1507  0307  51013 1401
0210  3204  1801  0102  78021 1301
03012 3204  1507  0605  51013 39021
```

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  ****  ****  0102  02025 1301  18012
UchiTelle  UT900-24  0101  0201  0307  0605  1401  39021
UchiTelle  UT900-25  0210  03012 0712  0102  1520  1301
UchiTelle  UT900-26  0101  0218  0804  1202  35091 4005
UchiTelle  UT910-01  2501  0201  1507  0307  51013 1401
UchiTelle  UT910-02  0210  3204  1801  0102  78021 1301
UchiTelle  UT910-03  03012 3204  1507  0605  51013 39021
```

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    ****    ****    0102    02025   1301    18012
UchiTelle   UT900-24    0101    0201    0307    0605    1401    39021
UchiTelle   UT900-25    0210    03012   0712    0102    1520    1301
UchiTelle   UT900-26    0101    0218    0804    1202    35091   4005
```

(continues on next page)

(continued from previous page)

```
UchiTelle   UT910-01   2501    0201    1507    0307    51013   1401
UchiTelle   UT910-02   0210    3204    1801    0102    78021   1301
UchiTelle   UT910-03   03012   3204    1507    0605    51013   39021
```

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      0301    0201     0501    249         249
UchiTelle  HJK_1   0301    0301    0201     0201    249         249
UchiTelle  HJK_3   01      0301    0201     0501    249         249
UchiTelle  HJK_4   01      0301    0201     0501    249         249
UchiTelle  MYU_2   02      0401    0302     0602    247         249
UchiTelle  MYU_1   0301    0301    0201     0201    247         249
UchiTelle  MYU_3   0301    0401    0201     0302    249         249
UchiTelle  MYU_4   0301    0401    0201     0302    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) TGFBhapl(1) TGFBhapl(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 TGFBhapl 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
0101  31
0102  37
0103  17
0201  21
0301  32
0401  9
```

(continues on next page)

(continued from previous page)

```
0501  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.

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.

2.4.2 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`)[1]

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**.

---

[1] These hardcoded numbers can be changed if you obtain the source code yourself and change the appropriate #define `emhaplofreq.h` and recompile the program.

2.5.1 A minimal configuration file

Here we present a minimal `.ini` file corresponding to Listing 2.1 A section by section review of this file follows. (Note comment lines have been omitted in the above example for clarity). A description of more advanced options is contained in *Advanced options*.

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
```

Configuration file sections (highlighted above)

- `[General]`

  This section contains variables that control the overall behavior of PyPop.

  - `debug=0`.

    This setting is for debugging. Setting it to 1 will set off a large amount of output of no interest to the general user. It should not be used unless you are running into trouble and need to communicate with the PyPop developers about the problems.

- 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]`.

  - `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
 *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 note_title in Listing 2.6). Listing multiple loci simply permits the same `.ini` file to be reused for each data file.

- `[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:Thompson:1992] code, using a Monte-Carlo Markov chain (MCMC). In addition, two measures (Chen and Diff) of the goodness of it of individual genotypes are reported under this option [Chen:etal: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 [Slatkin: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 estimation and calculation of linkage disequilibrium (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*).

2.5.2 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.

It also describes two extra sections related to using PyPop in conjunction with **`Arlequin`**: `[Arlequin]` and `[HardyWeinbergGuoThompsonArlequin]`.

`[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`.

  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)]**.

> **Warning:** If this 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.

`[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 [Guo:Thompson: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**.

`[Filters]` **extra section**

When this section is present, it allows you to specify succesive 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. Separate each filter name with a colon (`:`). 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.

`[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 but can be obtained via the IMGT ftp site (ftp://ftp.ebi.ac.uk/pub/databases/imgt/mhc/hla/). Invocation of this filter will produce a `popfile-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 (http://www.ebi.ac.uk/imgt/hla/download.html)) **[Default:** `msf` **]**

- `directory`.

  Specifies the path to the root of the sequence files. For `txt`: **[Default:** *prefix*`/share/PyPop/anthonynolan/HIG-seq-pep-text/` **]**. For `msf` files **[Default:** *prefix*`/share/PyPop/anthonynolan/msf/` **]**.

- `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 will 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`.

  Specifies the path to the root of the sequence files, 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 nth 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
 04/05/0306
 !06/1201/1301
 !07/0805
```

In the example above, `A*03` alleles will match to `01/02/03`, except for `A*0306`, which will match to `04/05/0306`. 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*0805` 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 sources[2]

[2] [Mack:etal:2007]; [Cano:etal:2007]; The Anthony Nolan list of deleted allele names (` <http://www.anthonynolan.com/HIG/lists/delnames.html>`__); and the Ambiguous Allele Combinations, release 2.18.0 (` <http://www.ebi.ac.uk/imgt/hla/ambig.html>`__).

CHAPTER

THREE

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.

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

3.2.1 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)
0201      0.21111   19        | 0101      0.13333   12
0301      0.15556   14        | 0201      0.21111   19
0101      0.13333   12        | 0210      0.10000   9
2501      0.12222   11        | 0218      0.10000   9
0210      0.10000   9         | 0301      0.15556   14
0218      0.10000   9         | 2501      0.12222   11
3204      0.08889   8         | 3204      0.08889   8
6901      0.04444   4         | 6814      0.03333   3
6814      0.03333   3         | 6901      0.04444   4
7403      0.01111   1         | 7403      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!
```

3.2.2 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.

0201 8/5.1
0301 4/4.0 1/0.8
0401 3/6.9 1/2.7 6/2.3
0501 8/9.9 5/3.8 5/6.7 6/4.8
      0201  0301  0401  0501
                            [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
              0201          15       20.78        1.61        0.2050
              0301          10       10.47        0.02        0.8850
              0401           9       16.31        3.28        0.0703
              0501          18       20.43        0.29        0.5915

-------------------------------------------------------------------------
Common genotypes
         0201:0201           8        5.11        1.63        0.2014
         0201:0401           3        6.93        2.23        0.1358
         0201:0501           8        9.89        0.36        0.5472
         0401:0501           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 situtation 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 by 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`.

3.2.3 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: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 [Schneider:etal: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
```

(continues on next page)

(continued from previous page)

```
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.

3.2.4 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:etal: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 ([Slatkin:1994]; [Slatkin: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:1977]; [Excoffier: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.

3.3.1 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
========================

Haplotype/ linkage disequlibrium (LD) statistics
------------------------------------------------

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

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

W_n
: Also known as Cramer's V Statistic [Cramer: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:

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

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.

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

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

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 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?*).

3.3.2 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
0101:1301:0402:   0.02222   2.0       | 0201:1401:0402:   0.03335   3.0
0101:1301:1101:   0.01111   1.0       | 3204:1401:0802:   0.03333   3.0
0101:1401:0901:   0.01111   1.0       | 0301:1401:0407:   0.03333   3.0
0101:1520:0802:   0.01111   1.0       | 0301:1301:0402:   0.03333   3.0
0101:1801:0407:   0.01111   1.0       | 0201:1401:1101:   0.03332   3.0
0101:3902:0404:   0.01111   1.0       | 0301:1520:0802:   0.02222   2.0
0101:3902:1602:   0.01111   1.0       | 0101:4005:0802:   0.02222   2.0
0101:4005:0802:   0.02222   2.0       | 0301:3902:0402:   0.02222   2.0
0101:8101:0802:   0.01111   1.0       | 0201:1301:1602:   0.02222   2.0
0101:8101:1602:   0.01111   1.0       | 0218:1401:0404:   0.02222   2.0
0201:1301:1602:   0.02222   2.0       | 0210:5101:1602:   0.02222   2.0
0201:1401:0402:   0.03335   3.0       | 0218:1401:1602:   0.02222   2.0
0201:1401:0404:   0.01111   1.0       | 0101:1301:0402:   0.02222   2.0
0201:1401:0407:   0.02222   2.0       | 2501:4005:0802:   0.02222   2.0
0201:1401:0802:   0.01111   1.0       | 2501:1301:0802:   0.02222   2.0

...
```

CHAPTER

FOUR

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

4.1.1 Did you find a bug?

When reporting a bug (https://github.com/alexlancaster/pypop/issues) please use one of the provided issue templates if applicable, otherwise just start a blank issue and describe your situation.

- Ensure the bug was not already reported by searching on GitHub under Issues (https://github.com/alexlancaster/pypop/issues).
- 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 demonstrating the expected behavior that is not occurring.
- If possible, use the relevant bug report templates to create the issue.
- When reporting bugs, especially during installation, please run the following and include the output:

```
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
```

4.1.2 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 (https://github.com/alexlancaster/pypop/issues/new) 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*.

4.1.3 Feature requests and feedback

The best way to send feedback is to file an issue using the feature template (https://github.com/alexlancaster/pypop/issues/new?assignees=&labels=&projects=&template=feature_request.md).

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. updating the changelog in `NEWS.rst`
7. checking unit tests pass
8. making a pull request

4.2.1 Fork this repository

Fork this repository before contributing (https://github.com/alexlancaster/pypop/network/members). Forks creates a cleaner representation of the contributions to the project (https://github.com/alexlancaster/pypop/network).

4.2.2 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
```

4.2.3 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.

4.2.4 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 can execute `pytest` as needed to run your unit tests. See *run unit tests with pytest*.

4.2.5 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 (https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github)

You can push to your fork now if you wish:

```
git push myfork new_branch
```

And, continue doing your developments are previously discussed.

4.2.6 Update `NEWS.rst`

Update the changelog file under `NEWS.rst` with an explanatory bullet list of your contribution. Add that list under the "Notes towards the next release" under the appropriate category, e.g. for a new feature you would add something like:

```
Notes towards next release
--------------------------
(unreleased)

New features
```

(continues on next page)

(continued from previous page)

```
^^^^^^^^^^^

* here goes my new additions
* explain them shortly and well
```

Also add your name to the authors list at `website/docs/AUTHORS.rst`.

4.2.7 Run unit tests with `pytest`

Once you have done your initial installation, you should first check that the build worked, by running the test suite, via `pytest`:

```
pytest tests
```

If `pytest` is not already installed, you can install via:

```
pip install pytest
```

If you run into errors during your initial installationg, 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

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

You should also continuously run `pytest` as you are developing your code, to ensure that you don't inadvertently break anything.

Also before creating a Pull Request from your branch, check that all the tests pass correctly, using the above.

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).

4.2.8 Make a Pull Request

Once you are finished, you can create a pull request to the main repository and engage with the developers. If you need some code review or feedback while you're developing the code just make a pull request.

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 finale merge, you should:

1. Make sure your PR passes all `pytest` tests.
2. Add unit tests if you are developing new features
3. Update documentation when there's new API, functionality etc.
4. Add a note to `NEWS.rst` about the changes.
5. Add yourself to `website/docs/AUTHORS.rst`.

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.

4.3.1 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 (https://mac.install.guide/commandlinetools/4.html), 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 developer command-line tools: https://developer.apple.com/downloads/ (includes `git`, `gcc`)
2. Visit http://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

(Currently untested in standalone-mode)

4.3.2 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.

1. 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).

2. 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]
```

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

Build-and-run-from-checkout (recommended for developers)

1. First manually install the dependencies via `pip`, note that if you are running on Python <= 3.8, you will need to also add `importlib-resources` to the list of packages, below.
   1. conda

   ```
   pip install numpy lxml psutil pytest
   ```

   2. system-wide

   ```
   pip install --user numpy lxml psutil pytest
   ```

2. Run the build

```
./setup.py build
```

3. You will be runnning PyPop, directly out of the `src/PyPop` subdirectory (e.g. `./src/PyPop/pypop.py` and `./src/PyPop/popmeta.py`). Note that you have to include the `.py` extension when you run from an uninstalled checkout, because the script is not installed.

Cleaning up build

If you installed using the approach in *Build-and-install (not recommended for developers)*, above, follow the end-user instructions on *Uninstalling PyPop*. In addition, to clean-up any compiled files and force a recompilation from scratch, run the `clean` command:

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

4.4 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.

4.4.1 Overview

All the documentation (including the website homepage) are maintained in this directory (and subdirectories) as reStructuredText (https://docutils.sourceforge.io/rst.html) (`.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 (https://www.sphinx-doc.org/en/master/) 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://alexlancaster.github.io/beta.pypop.org/

4.4.2 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.rst` and `AUTHORS.rst`, which is local to `website`)

  - `AUTHORS.rst`
  - `licenses.rst` (pulls in top-level `LICENSE`)
  - `biblio.rst`
- `html_root` (any files or directories commited 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 - https://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)

4.4.3 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:

> The commands in the "Sphinx build" section of the workflow .github/workflows/documentation.yaml (https://github.com/alexlancaster/pypop/blob/main/.github/workflows/documentation.yaml) which are used to run the GitHub Action that builds the documentation when it it deployed, is the best source for the most update-to-date commands to run, and should be consulted if the instructions in this document become out of date.

1. install sphinx and sphinx extensions

```
pip install setuptools_scm sphinx piccolo-theme sphinx_rtd_theme myst_parser rst2pdf␣
↪sphinx_togglebutton sphinx-argparse
```

2. make a fork of pypop if you haven't already (see *previous section*)
3. *clone the fork and add your fork as an upstream repository* on your local computer, and *make a new branch*.
4. make your changes to your `.rst` files and/or `conf.py`
5. build the HTML documentation:

```
sphinx-build website _build
```

6. view the local documention: you can open up browser and navigate to the `index.html` in the top-level of the newly-created `_build` directory
7. use `git commit` to commit your changes to your local fork.
8. 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
```

2. build the LaTeX and then compile the PDF:

```
sphinx-build -b latex website _latexbuild
make -C _latexbuild
```

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

CHAPTER

FIVE

AUTHORS AND HISTORY

5.1 Authors of software components

Alex Lancaster
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).

Mark P. Nelson
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
Author of haplotype frequency and linkage disequilibrium analysis module "emhaplofreq", author of R programs to do further statistical analysis and generate graphs and figures in PostScript.

Diogo Meyer
Contributed further statistical analysis code for the R programs.

Owen Solberg
Implemented filter modules, including conversion to allele name information to sequence data.

Yingssu Tsai
Implemented prototype of the allele names to sequence conversion filter module.

Glenys Thomson
Principal investigator and project lead.

gthwe
The Hardy-Weinberg "exact test" implementation is a modified version of Guo & Thompson's Guo:Thompson:1992 code. Dr. Sun-Wei Guo has kindly allowed us to release the code under the GNU General Public License (http://www.gnu.org/licenses/gpl.html). Original code available at http://www.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

5.2.1 Notes towards next release

(unreleased)

New features

- Updated to Python 3
- Implement new assymetric LD (ALD) measure
- New wrapper module `Haplostats`. This wraps a portion of the `haplo.stats` R package `haplo-stats` for haplotype estimation. [Implementation in alpha-phase].
- `popmeta`: now accepts the `-o`/`--outputdir` option for saving generated files.
- `pypop`: renamed `--generate-tsv` to `--enable-tsv`

5.2.2 Release Notes for PyPop 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.

5.2.3 Release Notes for PyPop 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 was 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.)

5.2.4 Release Notes for PyPop 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.

5.2.5 Release Notes for PyPop 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.

5.2.6 Release Notes for PyPop 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.

5.2.7 Release Notes for PyPop 0.4.3beta

Bug fixes

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

CHAPTER

SIX

LICENSES

6.1 License terms for PyPop

Except where noted in the source, PyPop is distributed under the terms of the GNU General Public License (http://www.gnu.org/licenses/gpl.html) (*GNU General Public License*). The list of authors and contributors is located in *Authors of software components*.

PyPop: Python for Population Genomics

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 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 (*GNU General Public License*) for more details.

6.1.1 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
```

(continues on next page)

(continued from previous page)

```
distribute copies of the software, or if you modify it.

  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.
```

(continues on next page)

(continued from previous page)

```
    b) You must cause any work that you distribute or publish, that in
    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
```

(continues on next page)

(continued from previous page)

```
operating system on which the executable runs, unless that component
itself accompanies the executable.

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.
```

(continues on next page)

(continued from previous page)

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

(continues on next page)

(continued from previous page)

```
    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 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

6.2.1 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. Secondarily, 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.

BIBLIOGRAPHY

[Cano:etal:2007] Pedro Cano. 2007. “Common and well-documented HLA alleles: report of the ad-hoc committee of the American Society for Histocompatiblity and Immunogenetics”. 68. 5. 392-417. *Human Immunology*.

[Chen:etal:1999] JJ Chen, JA Hollenbach, EA Trachtenberg, JJ Just, M Carrington, KS Ronningen, A Begovich, MC King, SK McWeeney, SJ Mack, HA Erlich, and G Thomson. 1999. “Hardy-Weinberg testing for HLA class II (DRB1, DQA1, DQB1 and DPB1) loci in 26 human ethnic groups”. 54. 533-542. *Tissue Antigens*.

[Cramer:1946] H Cramer. *Mathematical Models of Statistics*. 1946. Princeton University Press. Princeton NJ.

[Dempster:1977] A Dempster, N Laird, and D Rubin. 1977. “Maximum likelihood estimation from incomplete data using the EM algorithm”. 39. 1-38. *J Royal Stat Soc*.

[Ewens:1972] W Ewens. 1972. “The sampling theory of selectively neutral alleles.”. *Theor. Pop. Biol*. 3. 87-112.

[Excoffier:Slatkin:1995] Laurent Excoffier and Montgomery Slatkin. 1995. “Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population”. *Molecular Biology and Evolution*. 12. 5. 921-927.

[Guo:Thompson:1992] S W Guo and E A Thompson. 1992. “Performing the exact test of Hardy-Weinberg proportion for multiple alleles”. *Biometrics*. 48. 361-72.

[Hedrick:1987] P W Hedrick. 1987. “Gametic disequilibrium measures: proceed with caution”. 117. 2. 331-41. *Genetics*.

[Lancaster:etal:2003] Alex Lancaster, Mark P Nelson, Richard M Single, Diogo Meyer, and Glenys Thomson. 2003. “PyPop: a software framework for population genomics: analyzing large-scale multi-locus genotype data”. *Pacific Symposium on Biocomputing*. R B Altman. et al.. vol 8. 514-525. .. [PDF (344 kB)].

[Lancaster:etal:2007a] Alex Lancaster, Mark P Nelson, Richard M Single, Diogo Meyer, and Glenys Thomson. 2007a. *Software framework for Biostatistics Core*. JA Hansen. *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*. IHWG Press. Seattle, Washington. I. 510-517.

[Lancaster:etal:2007b] Alex Lancaster, Richard M Single, Owen D Solberg, Mark P Nelson, and Glenys Thomson. 2007b. “PyPop update - a software pipeline for large-scale multilocus population genomics.”. *Tissue Antigens*. 69 Suppl 1. 192-197. .. [PDF (150 kB)].

[Mack:etal:2007] Steven J. Mack, Alicia Sanchez-Mazas, Diogo Meyer, Richard M Single, Yingssu Tsai, and Henry Erlich. 2007. *Methods used in the generation and preparation of data for analysis in the 13th International Histocompatibility Workshop*. JA Hansen. *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*. IHWG Press. Seattle, Washington. I. 564-579.

[Meyer:etal:2007] Diogo Meyer, Richard M Single, Steven J. Mack, Alex Lancaster, Mark P Nelson, M. Fernánndez-Viñaa, Henry Erlich, and Glenys Thomson. 2007. *Haplotype frequencies and linkage disequilibrium among classical HLA genes*. JA Hansen. *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*. IHWG Press. Seattle, Washington. I. 705-746.

[Salamon:etal:1999] H Salamon, W Klitz, S Easteal, X Gao, HA Erlich, M Fernandez-Vina, and EA Trachtenberg. 1999. “Evolution of HLA class II molecules: Allelic and amino acid site variability across populations”. 152. 393-400. *Genetics*.

[Schneider:etal:2000] S Schneider, D Roessli, and L Excoffier. 2000. "Arlequin: A software for population genetics data analysis. http://lgb.unige.ch/arlequin/". Ver 2.000. Genetics and Biometry Lab, Dept. of Anthropology, University of Geneva.

[Single:etal:2007a] RM Single, D Meyer, and G Thomson. 2007a. *Statistical methods for analysis of population genetic data*. JA Hansen. *Immunobiology of the Human MHC: Proceedings of the 13th International Histocompatibility Workshop and Conference*. IHWG Press. Seattle, Washington. I. 518-522.

[Slatkin:1994] M Slatkin. 1994. "An exact test for neutrality based on the Ewens sampling distribution". *Genetical Research*. 64. 71-74.

[Slatkin:1996] M Slatkin. 1996. "A correction to the exact test based on the Ewens sampling distribution". *Genetical Research*. 68. 259-260.

[Solberg:etal:2008] Owen D. Solberg, Steven J. Mack, Alex K. Lancaster, Richard M. Single, Yingssu Tsai, Alicia Sanchez-Mazas, and Glenys Thomson Hum Immunol (2008), doi: 10.1016/j.humimm.2008.05.001

[Thomson:Single:2014] Glenys Thomson and Richard M. Single. 2014. "Conditional Asymmetric Linkage Disequilibrium (ALD): Extending the Biallelic r2 Measure" *Genetics*, 198, 321–331.

[Watterson:1978] G Watterson. 1978. "The homozygosity test of neutrality". *Genetics*. 88. 405-417.