# Documentation for PopSyCLE

**Population Synthesis for Compact object Lensing Events**

Casey Lam

September 2019

# Contents

# 1 Installation

PopSyCLE has several dependencies. In addition to installing PopSyCLE, the user will need Galaxia, PopStar, and several Python libraries.

## 1.1 Installing Galaxia

### 1.1.1 Installation

Note there are also instructions from the creators of Galaxia at http://galaxia.sourceforge.net/Galaxia3pub.html; what follows is a super pedantic version of that.

1. Go to https://sourceforge.net/projects/galaxia/files/ and download Galaxia by clicking the big green button.

2. Go to your Downloads folder and untar the file by double-clicking it.

3. Move the untar'd folder (which should be called something like `galaxia-0.7.2`) to your home directory. (That's the directory where you get sent if you `cd` and don't put a location. On my laptop, it's `/Users/casey/`).

4. In your home directory, make a directory called `GalaxiaData` (i.e. `mkdir GalaxiaData`).

5. Move to the `galaxia-0.7.2` directory (i.e. `cd galaxia-0.7.2`), and in there, do the following (replacing `/Users/casey/` with your home directory as appropriate):

   - `./configure --datadir=/Users/casey/GalaxiaData/`
   - `make`
   - `sudo make install`
   - `cp -r GalaxiaData/ /Users/casey/GalaxiaData/`

6. Move back into your home directory (i.e. `cd`), then run the following:

   - `galaxia -s warp`

That should be it! You don't have to install `ebf` if you just download PopSyCLE!

NOTE: the instructions in step 5 are assuming a root install. If you want to do a local install, you need to have a folder for the software to be installed in. For example, in my home directory I made a `sw` directory (`/Users/casey/sw`) for `galaxia` to be installed in. Then I ran the following instead:

- `./configure --prefix=/Users/casey/sw --datadir=/Users/casey/GalaxiaData/`
- `make`
- `make install`
- `cp -r GalaxiaData/ /Users/casey/GalaxiaData/`

Also, you need to export `galaxia` to your path. In your `.bash_profile`, add the line `export PATH=$PATH:/Users/casey/sw/bin`. Then proceed with step 6 in the installation instructions.

#### 1.1.2 Uninstallation

You need to remove the compiled `galaxia` code (you can find where it is by typing `which galaxia` in the terminal), the `GalaxiaData` directory, and you might as well remove the `galaxia-0.7.2` directory also. When you do `which galaxia` nothing should be returned.

#### 1.1.3 Parameter modification

Suppose you want to change the pattern speed in `Galaxia`. To do this, follow the installation instructions up to and including step 4. Then do the following:

1. Move to the `galaxia-0.7.2/src` directory.
2. Open the `Population.h` file with your favorite text editor.
3. Find the pattern speed (in this case by searching for 71.62) and replace with your desired value (in this case 40.00).
4. Save the change.

Now return to step 5 in the installation instruction and proceed as instructed.

### 1.2 Installing `PopStar`

`PopStar` can be installed by cloning the repository from https://github.com/astropy/PopStar.

### 1.3 Installing Python libraries

We recommend the Anaconda distribution. In particular, numpy v1.13 or higher is required, along with Astropy and H5py.

# 2 Reading files

`PopSyCLE` use all sorts of different file formats. It can easily get confusing, so here is a short guide to the basics.

## 2.1 How to read HDF5 files

Within the HDF5 file are datasets that store the information. It is kind of like a dictionary in python– the dataset can be manipulated just like a numpy array.

First, go to the directory containing the HDF5 file you want to open. Next, start ipython. Then type the following:

```
import h5py

hf = h5py.File('filename.h5', 'r').
```

If you want to see the names of all the datasets in an HDF5 file, type the following:

```
list(hf.keys()).
```

Suppose you want to work with the dataset named `dname`. To access the dataset, type:

```
dset = hf['dname'].
```

Note that only one person at a time can work on an open HDF5 file. Thus, at the end, you need to close the file:

```
hf.close().
```

## 2.2 How to read EBF files

The EBF file is basically a dictionary in python. The output of `Galaxia` is in the EBF format.

First, go to the directory containing the EBF file you want to open. Next, start ipython. Then type the following:

```
import ebf

ef = ebf.read('filename.ebf', '/').
```

If you want to see the names of all the keys in the EBF file, type the following:

```
ef.keys().
```

Suppose you want to work with the key `xkey`. To access that part of the file, type:

```
x = ef['xkey'].
```

Now `x` is just a numpy array and can manipulated as such.

### 2.3 How to read FITS table files

First, go to the directory containing the fits file you want to open. Next, start ipython, Then type the following:

```
from astropy.table import Table

tab = Table.read('table.fits')
```

To view the entire table, just type `tab`. The table works similar to a python dictionary. The column names are the keys of the dictionary, and the dictionary name in this case is `tab`.

To view the header information/metadata, type

`tab.meta`.

# 3 Description of Pipeline

First, run `Galaxia` to create an EBF file, which produces a synthetic survey, i.e. a bunch of stars. Next, run population synthesis (perform_pop_syn) to inject compact objects into the synthetic survey; both the compact objects and stars are saved in an HDF5 file. Then run a synthetic survey (calc_events and refine_events) that will produce a list of microlensing events, which are listed in a FITS file.

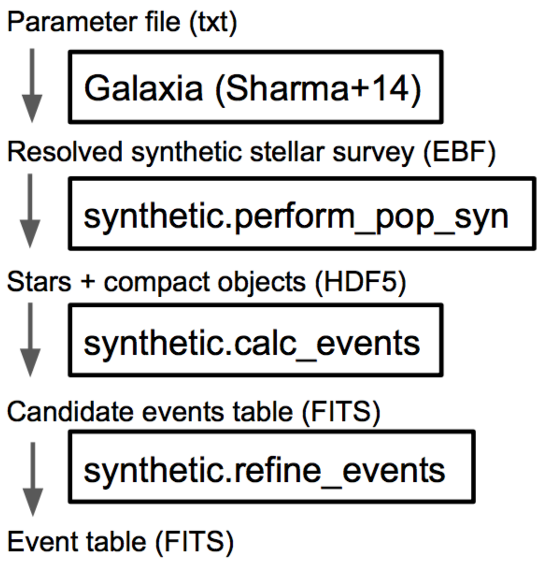


Figure 1: Flowchart of pipeline.

# 4 Outputs

In addition to the outputs about to be described, each function produces a text log file that lists the input parameters.

## 4.1 perform_pop_syn

### 4.1.1 Label/summary file (Astropy FITS table)

- file_name is the name of the dataset for the HDF5 file.
- long_start and long_end are the edges of the longitude bin.
- lat_start and lat_end are the edges of the latitude bin.
- objects is the number of objects in that latitude/longitude bin.
- N_stars, N_WD, N_NS, and N_BH are the number of stars, white dwarfs, neutron stars, and black holes, respectively, in that latitude/longitude bin. The sum of these should be equal to the total number of objects.

Total white dwarfs is equal to adding the numbers of WDs made from MIST and the IFMR. WDs from the MIST models have photometry (they're bright), while WDs from the IFMR and are dark (roughly 30th magnitude and below, so we assign them a value of `nan` for their magnitude.)

<Table length=36>

| file_name | long_start | long_end | lat_start | lat_end | objects | N_stars | N_WD | N_NS | N_BH |
|---|---|---|---|---|---|---|---|---|---|
| bytes4 | float64 | float64 | float64 | float64 | int64 | int64 | int64 | int64 | int64 |
| l0b0 | 0004.938 | 0004.959 | 001.938 | 001.959 | 0 | 0 | 0 | 0 | 0 |
| l0b1 | 0004.938 | 0004.959 | 001.959 | 001.979 | 38758 | 36313 | 2318 | 96 | 31 |
| l0b2 | 0004.938 | 0004.959 | 001.979 | 002.000 | 98252 | 92115 | 5761 | 240 | 136 |
| l0b3 | 0004.938 | 0004.959 | 002.000 | 002.021 | 98131 | 91989 | 5794 | 228 | 120 |
| l0b4 | 0004.938 | 0004.959 | 002.021 | 002.041 | 38538 | 36115 | 2274 | 100 | 49 |
| l0b5 | 0004.938 | 0004.959 | 002.041 | 002.062 | 0 | 0 | 0 | 0 | 0 |
| l1b0 | 0004.959 | 0004.979 | 001.938 | 001.959 | 37973 | 35635 | 2196 | 102 | 40 |
| l1b1 | 0004.959 | 0004.979 | 001.959 | 001.979 | 146343 | 137172 | 8605 | 388 | 178 |
| l1b2 | 0004.959 | 0004.979 | 001.979 | 002.000 | 146929 | 137769 | 8616 | 351 | 193 |
| l1b3 | 0004.959 | 0004.979 | 002.000 | 002.021 | 146109 | 136923 | 8587 | 395 | 204 |
| l1b4 | 0004.959 | 0004.979 | 002.021 | 002.041 | 143826 | 134865 | 8427 | 368 | 166 |
| l1b5 | 0004.959 | 0004.979 | 002.041 | 002.062 | 37739 | 35391 | 2206 | 83 | 59 |
| l2b0 | 0004.979 | 0005.000 | 001.938 | 001.959 | 97024 | 90831 | 5836 | 232 | 125 |
| l2b1 | 0004.979 | 0005.000 | 001.959 | 001.979 | 146781 | 137554 | 8635 | 392 | 200 |
| l2b2 | 0004.979 | 0005.000 | 001.979 | 002.000 | 146107 | 136876 | 8685 | 363 | 183 |
| l2b3 | 0004.979 | 0005.000 | 002.000 | 002.021 | 145358 | 136289 | 8503 | 366 | 200 |
| l2b4 | 0004.979 | 0005.000 | 002.021 | 002.041 | 144969 | 135597 | 8828 | 374 | 170 |
| l2b5 | 0004.979 | 0005.000 | 002.041 | 002.062 | 96090 | 89816 | 5904 | 249 | 121 |
| l3b0 | 0005.000 | 0005.021 | 001.938 | 001.959 | 97227 | 91158 | 5710 | 240 | 119 |
| l3b1 | 0005.000 | 0005.021 | 001.959 | 001.979 | 145371 | 136211 | 8609 | 363 | 188 |
| l3b2 | 0005.000 | 0005.021 | 001.979 | 002.000 | 145038 | 135760 | 8671 | 395 | 212 |
| l3b3 | 0005.000 | 0005.021 | 002.000 | 002.021 | 144262 | 135128 | 8564 | 368 | 202 |
| l3b4 | 0005.000 | 0005.021 | 002.021 | 002.041 | 143560 | 134530 | 8494 | 365 | 171 |
| l3b5 | 0005.000 | 0005.021 | 002.041 | 002.062 | 94873 | 88947 | 5598 | 217 | 111 |
| l4b0 | 0005.021 | 0005.041 | 001.938 | 001.959 | 37743 | 35294 | 2304 | 94 | 51 |
| l4b1 | 0005.021 | 0005.041 | 001.959 | 001.979 | 144179 | 135039 | 8618 | 332 | 190 |
| l4b2 | 0005.021 | 0005.041 | 001.979 | 002.000 | 143802 | 134786 | 8474 | 351 | 191 |
| l4b3 | 0005.021 | 0005.041 | 002.000 | 002.021 | 143790 | 134876 | 8383 | 355 | 176 |
| l4b4 | 0005.021 | 0005.041 | 002.021 | 002.041 | 141147 | 132257 | 8346 | 365 | 179 |
| l4b5 | 0005.021 | 0005.041 | 002.041 | 002.062 | 37238 | 34878 | 2232 | 90 | 38 |
| l5b0 | 0005.041 | 0005.062 | 001.938 | 001.959 | 0 | 0 | 0 | 0 | 0 |
| l5b1 | 0005.041 | 0005.062 | 001.959 | 001.979 | 37987 | 35590 | 2232 | 102 | 63 |
| l5b2 | 0005.041 | 0005.062 | 001.979 | 002.000 | 95712 | 89565 | 5769 | 247 | 131 |
| l5b3 | 0005.041 | 0005.062 | 002.000 | 002.021 | 95154 | 89039 | 5759 | 231 | 125 |
| l5b4 | 0005.041 | 0005.062 | 002.021 | 002.041 | 37353 | 35027 | 2174 | 99 | 53 |
| l5b5 | 0005.041 | 0005.062 | 002.041 | 002.062 | 0 | 0 | 0 | 0 | 0 |

### 4.1.2 Stars and compact objects (HDF5)

The data output contained in the HDF5 datasets are a combination of outputs that come directly from Galaxia, and outputs we ourselves have calculated or defined.

| Index | Tag name | Brief Description | Units |
|---|---|---|---|
| [0] | zams_mass | ZAMS mass | $M_\odot$ |
| [1] | rem_id | Integer indicating the remnant object type (more details in tag description) | N/A |
| [2] | mass | Current mass | $M_\odot$ |
| [3] | px | Heliocentric x position | kpc |
| [4] | py | Heliocentric y position | kpc |
| [5] | pz | Heliocentric z position | kpc |
| [6] | vx | Heliocentric x velocity | km/s |
| [7] | vy | Heliocentric y velocity | km/s |
| [8] | vz | Heliocentric z velocity | km/s |
| [9] | rad | Galactic radial distance | kpc |
| [10] | glat | Galactic latitude | deg |
| [11] | glon | Galactic longitude | deg |
| [12] | vr | Galactic radial velocity | km/s |
| [13] | mu_b | Galactic proper motion, b component | mas/yr |
| [14] | mu_lcosb | Galactic proper motion, l component | mas/yr |
| [15] | age | Age | log(age/yr) |
| [16] | popid | Population ID– integer indicating the population type ranging from 0 to 9 | N/A |
| [17] | ubv_k | UBV photometric system, K-band absolute magnitude | mag |
| [18] | ubv_i | UBV photometric system, I-band absolute magnitude | mag |
| [19] | exbv | Extinction E(B-V) at the location of star given by 3-D Schlegel extinction maps | mag |
| [20] | obj_id | Object ID– unique integer to identify star/compact object | N/A |
| [21] – [26] | ubv_j, u, r, b, h, v | UBV photometric system, J, U, R, B, H, V absolute magnitude (in this order) | mag |
| [27] | teff | Effective temperature | log(T/Kelvin) |
| [28] | grav | Surface gravity | log(gravity) |
| [29] | mbol | Bolometric magnitude | log(L/$L_\odot$) |
| [30] | feh | Metallicity | $[Fe/H]$ |

Table 1: Note that the tag names are NOT used in the HDF5 files. They are just listed here to show the direct correspondence between the HDF5 outputs and the Astropy table of candidate events (next section.)

For stars (which are generated by Galaxia), the following outputs are taken directly from Galaxia and just reformatted into the HDF5 format; parenthetical names correspond to the tag name from Galaxia, if different: zams_mass (smass), mass (mact), px, py, pz, vx, vy, vz, age, popid, ubv_k, ubv_i, ubv_u, ubv_b, ubv_v, ubv_r, ubv_j, ubv_h, exbv (exbv_schlegel), teff, grav, mbol (lum), feh. Note that the lum key from Galaxia is referred to as mbol in the Galaxia documentation. For compact objects (which we generated with our population synthesis code), we must assign these values ourselves. For both stars and compact objects, the following are things we have directly calculated or assigned ourselves: rem_id, rad, glat, glon, vr, mu_b, mu_lcosb, obj_id.[1]

[1]For reasons relating to managing RAM, we calculate rad, glat, and glon although they are an output given directly from Galaxia, and we could have just read in the value. However, it can be calculated directly from knowledge of px, py, and pz.

## 4.2 calc_events

### 4.2.1 Event candidates table (Astropy FITS table)

The event candidates table is very similar to the HDF5 file created in perform_pop_syn. (In fact, the top part is completely duplicated; it's here for completeness.) However, the main difference is that there is a LOT less of the output, so instead of writing it in arrays in an HDF5 file, we use an Astropy table.

| **Tag name** | **Brief Description** | **Units** |
|---|---|---|
| zams_mass (_L, _S) | ZAMS mass | $M_\odot$ |
| rem_id (_L, _S) | Remnant ID– integer indicating the remnant object type ranging from 0 to 3 | N/A |
| mass (_L, _S) | Current mass | $M_\odot$ |
| px (_L, _S) | Heliocentric x position | kpc |
| py (_L, _S) | Heliocentric y position | kpc |
| pz (_L, _S) | Heliocentric z position | kpc |
| vx (_L, _S) | Heliocentric x velocity | km/s |
| vy (_L, _S) | Heliocentric y velocity | km/s |
| vz (_L, _S) | Heliocentric z velocity | km/s |
| rad (_L, _S) | Galactic radial distance | kpc |
| glat (_L, _S) | Galactic latitude b | deg |
| glon (_L, _S) | Galactic longitude l | deg |
| vr (_L, _S) | Galactic radial velocity | km/s |
| mu_b (_L, _S) | Galactic proper motion, b component | mas/yr |
| mu_lcosb (_L, _S) | Galactic proper motion, l component | mas/yr |
| age (_L, _S) | Age | log(age/yr) |
| popid (_L, _S) | Population ID– integer indicating the population type ranging from 0 to 9 | N/A |
| ubv_u, b, v, i, r, j, h, k (_L, _S) | UBV photometric system absolute magnitudes | mag |
| exbv (_L, _S) | Extinction E(B-V) at the location of star given by 3-D Schlegel extinction maps | mag |
| obj_id (_L, _S) | Object ID– unique integer to identify star/compact object | N/A |
| teff (_L, _S) | Effective temperature | log(T/Kelvin) |
| grav (_L, _S) | Surface gravity | log(gravity) |
| mbol (_L, _S) | Bolometric magnitude | log(L/$L_\odot$) |
| feh (_L, _S) | Metallicity | $[Fe/H]$ |
| theta_E | (Angular) Einstein radius | mas |
| mu_rel | Relative source-lens proper motion | mas/yr |
| u0 | (Unitless) minimum source-lens separation, *during* the survey | dim'less (normalized to $\theta_E$) |
| t0 | Time at which minimum source-lens separation occurs | days |

Tag names ARE used for the Astropy table. You will see a lot of the tag names have a parenthetical after (_L, _S). That is to indicate there is one tag for the lens (L) and one for the source (S), since for a given event, you need to have both a lens and a source, and each of these things has a mass, a velocity, a position, etc. For example, zams_mass_L is the ZAMS mass of the lens, and age_S is the log(age/yr) of the source.

### 4.2.2 Blends table (Astropy FITS table)

For each candidate microlensing event, associated with it are blended stars, which we call neighbors. Given the blend radius chosen when running calc_events, the blend table saves all neighbor stars that fall within that distance from the lenses in the candidate events table. The blends table is again almost identical to the HDF5 output, but is has three additional items. For each neighbor star, it lists the object ID of the lens and source it is associated with, and the distance between itself and the lens. Note that there can be multiple neighbor stars associated with a single lens and source (microlensing event).

| **Tag name** | **Brief Description** | **Units** |
|---|---|---|
| zams_mass_N | ZAMS mass | $M_{\odot}$ |
| rem_id_N | Remnant ID– integer indicating the remnant object type ranging from 0 to 3 | N/A |
| mass_N | Current mass | $M_{\odot}$ |
| px_N | Heliocentric x position | kpc |
| py_N | Heliocentric y position | kpc |
| pz_N | Heliocentric z position | kpc |
| vx_N | Heliocentric x velocity | km/s |
| vy_N | Heliocentric y velocity | km/s |
| vz_N | Heliocentric z velocity | km/s |
| rad_N | Galactic radial distance | kpc |
| glat_N | Galactic latitude b | deg |
| glon_N | Galactic longitude l | deg |
| vr_N | Galactic radial velocity | km/s |
| mu_b_N | Galactic proper motion, b component | mas/yr |
| mu_lcosb_N | Galactic proper motion, l component | mas/yr |
| age_N | Age | log(age/yr) |
| popid_N | Population ID– integer indicating the population type ranging from 0 to 9 | N/A |
| ubv_u, b, v, i, r, j, h, k_N | UBV photometric system absolute magnitudes | mag |
| exbv_N | Extinction E(B-V) at the location of star given by 3-D Schlegel extinction maps | mag |
| obj_id_N | Object ID– unique integer to identify star/compact object | N/A |
| teff_N | Effective temperature | log(T/Kelvin) |
| grav_N | Surface gravity | log(gravity) |
| mbol_N | Bolometric magnitude | log(L/$L_{\odot}$) |
| feh_N | Metallicity | $[Fe/H]$ |
| obj_id_L | Object ID of the lens | N/A |
| obj_id_S | Object ID of the source | N/A |
| sep_LN | Separation between lens and neighbor | arcsec |

### 4.3 refine_events

#### 4.3.1 Events table (Astropy FITS table)

The output here is very similar to the candidate events table. In fact, part of it is completely duplicated. All tags listed in the event candidates table are also part of the events table. However, the following columns are also appended. NOTE: the entries for u0 and t0 are *overwritten*; the values for u0 and t0 returned from calc_events is different from that returned in refine_events. Each refine_events file requires you to choose a filter and extinction law; in this table we suppose filter $x$ is chosen.

| **Tag name** | **Brief Description** | **Units** |
|---|---|---|
| u0 | (Unitless) minimum source-lens separation, *during* the survey | dim'less |
| t0 | Time at which minimum source-lens separation occurs | days |
| delta_m_$x$ | Bump amplitude (difference in baseline and maximum magnification magnitude) in $x$-band | mag |
| pi_rel | Relative parallax | mas |
| pi_E | Microlensing parallax | dim'less |
| t_E | Einstein crossing time | days |
| ubv_$x$_app (_L, _S) | UBV photometric system, $x$-band apparent magnitude, with extinction | mag |
| ubv_$x$_LSN | Blended magnitude in $x$ band (Apparent magnitude of source + lens + neighbors) | mag |
| f_blend_$x$ | Source flux fraction (unlensed source flux divided by baseline) in $x$-band | dim'less |
| cent_glon_$x$_N | Galactic longitude l of neighbor stars' centroid | deg |
| cent_glat_$x$_N | Galactic latitude b of neighbor stars' centroid | deg |
| ubv_$x$_app_N | Apparent magnitude of neighbor stars, $x$-band apparent magnitude | mag |

## 4.4 Additional descriptions of tags

**rem_id** These label the different types of remnant objects (star, black hole, neutron star, or white dwarf.) They are identified as following:

- 0: Star
- 101: WD
- 102: NS
- 103: BH

**popid** Describes which population is generated.[2] They are identified as following:

- 0: Thin disk, $\leq$ 0.15 Gyr
- 1: Thin disk, 0.15-1 Gyr
- 2: Thin disk, 1-2 Gyr
- 3: Thin disk, 2-3 Gyr
- 4: Thin disk, 3-5 Gyr
- 5: Thin disk, 5-7 Gyr
- 6: Thin disk, 7-10 Gyr
- 7: Thick disk, 11 Gyr (single-age)
- 8: Stellar halo, 14 Gyr (single-age)
- 9: Bulge, 10 Gyr (single-age)

**px, py, pz; vx, vy, vz** These are given in heliocentric coordinates (i.e. Cartesian coordinates with the sun at the origin.) See subsection on coordinate systems for more information.

**rad, glat, glon; vr, mu_b, mu_lcosb** These are given in galactic coordinates (i.e. spherical coordinates with the sun at the origin.) See subsection on coordinate systems for more information.

**ubv_u, b, v, r, i, j, h, k; exbv** Photometry information is given in absolute magnitude. For NSs and BHs, all these values are `nan` to indicate they are dark.

**t0** Note that you can have a negative day (this just means time before the "zero" time, which is defined as the state of the system that is generated by Galaxia and the population synthesis. Since we are assuming everything moves in straight lines, we can propagate either forward or backwards.

[2] In `Galaxia` there is an option for a 10th population type; the Bullock and Johnston stellar halos. We have chosen not use it, and the code is not written to include it.

## 5 Coordinates Systems

There are two different coordinate systems used, Heliocentric and Galactic. Heliocentric coordinates are Cartesian coordinates with the sun at the origin. The positive $x$ axis is pointing toward the Galactic Center, and the positive $z$ axis is pointing toward the Galactic North Pole. Galactic coordinates are spherical coordinates with the sun at the origin. Longitude $l$ is measuring the angular distance of an object eastward along the galactic equator from the galactic center, and latitude $b$ is measuring the angle of an object north or south of the galactic equator (or midplane) as viewed from Earth; positive to the north, negative to the south. Radius $r$ is the distance from the sun to the object.

The conversion between Heliocentric and Galactic is just the same as converting between rectangular to spherical coordinates, where $\phi = l$ and $\theta = -b + 90°$. Going from Galactic to Heliocentric (units are degrees):

$$
\begin{aligned}
x &= r\sin(-b + 90°)\cos l = r\cos b\cos l \\
y &= r\sin(-b + 90°)\sin l = r\cos b\sin l \\
z &= r\cos(-b + 90°) = r\sin b
\end{aligned}
$$

Going from Heliocentric to Galactic (units are degrees):

$$
\begin{aligned}
r &= \sqrt{x^2 + y^2 + z^2} \\
b &= -\cos^{-1}(z/r) + 90° \\
l &= \tan^{-1}(y/x)
\end{aligned}
$$

Note: be careful with the branch of arctangent. Practically, use `numpy.arctan2` if using Python.

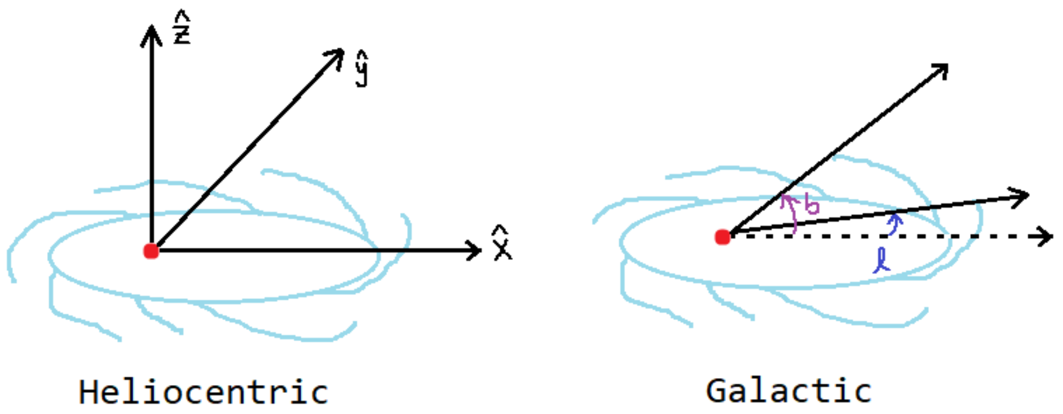


Figure 2: Diagram of Heliocentric and Galactic coordinate systems. The red dot is the sun.