Astronomy & Astrophysics manuscript no. output
January 28, 2021

P-pop user documentation

J. Kammerer[1,2], S. P. Quanz[3], and E. Fontanet[3]

[1] European Southern Observatory, Karl-Schwarzschild-Str 2, 85748, Garching, Germany
e-mail: jens.kammerer@eso.org
[2] Research School of Astronomy & Astrophysics, Australian National University, ACT 2611, Australia
[3] ETH Zurich, Institute for Particle Physics and Astrophysics, Wolfgang-Pauli-Strasse 27, 8093 Zurich, Switzerland

Received today; accepted tomorrow

ABSTRACT

Aims. This user documentation aims to help the reader to get the planet population synthesis tool P-pop running on their own machine. It also defines the format of its output, the planet population table.
Methods. P-pop is a publicly distributed Python tool. It uses planet occurrence statistics from the *Kepler* mission and model distributions for the exoplanet system parameters in order to simulate synthetic planet populations.
Results. P-pop allows to simulate millions of synthetic planets within timescales of hours. These can then be used in order to make yield predictions for future instruments and space missions for example.

Key words. Planets and satellites: detection – Planets and satellites: terrestrial planets – Methods: numerical

1. Introduction

P-pop is a Monte-Carlo tool to simulate exoplanet populations based on planet occurrence statistics from the *Kepler* mission. It consists of two Python libraries, P-pop and P-pop Photometry.

P-pop is used to simulate exoplanet populations which are saved in a planet population table which can then be used by other tools. P-pop Photometry is used to compute the photometry for a given planet population table which can then be used to estimate the expected exoplanet yield for future instruments or space missions for example.

P-pop has been used in Kammerer & Quanz (2018), Quanz et al. (2018), and Quanz et al. (2019) to make yield predictions for the LIFE[1] space mission.

2. Installation

P-pop and P-pop Photometry can be downloaded from GitHub[2]. Except for a Python distribution with the numpy, scipy, and astropy packages there is no installation required.

The provided example only works if P-pop and P-pop Photometry are cloned into the same parent directory. If not, the paths of the planet population table and its photometry might have to be updated.

P-pop and P-pop Photometry have been developed in Python 2.7, but should also be compatible with Python 3.7. Please help to improve the code by reporting any issues to the authors.

3. Usage

3.1. P-pop

In order to simulate an exoplanet population with P-pop run the script `P-pop.py`. This will first display some informative plots about the star catalog and the different distributions for the physical parameters of the planets. Then, it will simulate an exoplanet population consisting of 100 universes and write it to a file called `TestPlanetPopulation.txt`.

Various settings can be changed in the „SETUP“ section of the script `P-pop.py` (cf. Figure 1). More detailed information on the available settings can be found in the comments.

3.2. P-pop Photometry

In order to compute the photometry for a given planet population table with *P-pop Photometry* run the script `P-pop_Photometry.py`. This will first display some informative plots about the filters. Then it will compute the photometry for the `TestPlanetPopulation.txt` planet population table and write it to a file called `TestPlanetPopulation_filter.txt`.

Various settings can be changed in the „SETUP“ section of the script `P-pop_Photometry.py` (cf. Figure 3). More detailed information on the available settings can be found in the comments.

4. Planet population table

4.1. Description and format

The output of P-pop is a so-called planet population table. This table contains all information about the simulated planets and its content is described in Table 1.

In general, each line in the planet population table corresponds to one simulated planet. The columns are split by tabs (`\t`) and the column headers are unique and permanent. Hence, you can find a specific column by looking for its permanent header, as it is done in the script `ReadPlanetPopulation.py`. Note that the first two rows of the planet population table are devoted to headers. The second row contains the column

[1] `https://www.life-space-mission.com/`
[2] `https://github.com/kammerje/P-pop` and `https://github.com/kammerje/P-pop-Photometry`

```
"""
# ============================================================================
# P-POP
# A Monte-Carlo tool to simulate exoplanet populations
#
# Authors: Jens Kammerer, Sascha Quanz, Emile Fontanet
# Version: 5.0.0
# Last edited: 15.04.2020
# ============================================================================
#
# P-pop is introduced in Kammerer & Quanz 2018
# (https://ui.adsabs.harvard.edu/abs/2018A%26A...609A...4K/abstract). Please
# cite this paper if you use P-pop for your research.
#
# P-pop makes use of forecaster from Chen & Kipping 2017
# (https://ui.adsabs.harvard.edu/abs/2017ApJ...834...17C/abstract).
"""


# Don't print annoying warnings. Comment out if you want to see them.
import warnings
warnings.filterwarnings('ignore')


# ============================================================================
# IMPORTS
# ============================================================================

# Import your own catalogs, distributions and models here.
import SystemGenerator
from StarCatalogs import CrossfieldBrightSample, ExoCat_1, LTC_2
from PlanetDistributions import Weiss2018KDE, Weiss2018, SAG13, Dressing2015, Burke2015,
    Fressin2013
from MassModels import Chen2017
from EccentricityModels import Circular
from StabilityModels import He2019
from OrbitModels import Random
from AlbedoModels import Uniform
from ExozodiModels import Ertel2020, Ertel2018


# ============================================================================
# SETUP
# ============================================================================

# Select the catalogs, distributions and models which you want to use here.

# Select the star catalog, the spectral types, the distance range and the
# declination range which should be included here.
StarCatalog = CrossfieldBrightSample # used in Kammerer & Quanz 2018
#StarCatalog = ExoCat_1 # used by NASA
#StarCatalog = LTC_2 # LIFE Target Catalog
Stypes = ['A', 'F', 'G', 'K', 'M'] # list of str
Dist_range = [0., 20.] # pc, list of float, [min, max]
Dec_range = [-90., 90.] # deg, list of float, [min, max]

# Select the planet distributions and the scenario which should be used here.
# A different planet distribution can be assigned to each spectral type.
# dict, StarCatalog.Stype as keys, PlanetDistribution as data
#StypeToModel = {'F': Fressin2013, 'G': Fressin2013, 'K': Fressin2013, 'M': Fressin2013}
#StypeToModel = {'F': Burke2015, 'G': Burke2015, 'K': Burke2015, 'M': Dressing2015}
StypeToModel = {'F': SAG13, 'G': SAG13, 'K': SAG13, 'M': Dressing2015} # used in the ESA Voyage
    2050 White Paper
#StypeToModel = {'F': SAG13, 'G': SAG13, 'K': SAG13, 'M': SAG13}
#StypeToModel = {'F': Weiss2018, 'G': Weiss2018, 'K': Weiss2018, 'M': Weiss2018}
#StypeToModel = {'F': Weiss2018KDE, 'G': Weiss2018KDE, 'K': Weiss2018KDE, 'M': Weiss2018KDE}
```

Fig. 1. Continued on next page...

```
Scenario = 'baseline'
#Scenario = 'pessimistic' # for 1-sigma lower error bars on planet distribution
#Scenario = 'optimistic' # for 1-sigma upper error bars on planet distribution

# Select the mass model, the eccentricity model, the stability model, the orbit
# model, the albedo model and the exozodiacal dust model which should be used
# here.
MassModel = Chen2017 # Forecaster
EccentricityModel = Circular
StabilityModel = None
#StabilityModel = He2019
OrbitModel = Random
AlbedoModel = Uniform
ExozodiModel = Ertel2018
#ExozodiModel = Ertel2020 # handle with care!

# Select whether you want to display summary plots after loading the catalogs,
# distributions and models selected above and how many test draws should be
# done for generating these plots.
SummaryPlots = True
#SummaryPlots = False
Ntest = 100000 # int
#Ntest = 10000 # int

# Select a name for the output planet population table and how many universes
# should be simulated.
Name = 'TestPlanetPopulation' # str
Nuniverses = 100 # int


# ============================================================================
# P-POP
# ============================================================================

# Don't modify the following code.
SysGen = SystemGenerator.SystemGenerator(StarCatalog,
                                         StypeToModel,
                                         MassModel,
                                         EccentricityModel,
                                         StabilityModel,
                                         OrbitModel,
                                         AlbedoModel,
                                         ExozodiModel,
                                         Stypes,
                                         Dist_range, # pc
                                         Dec_range, # deg
                                         Scenario,
                                         SummaryPlots,
                                         Ntest)
SysGen.SimulateUniverses(Name,
                         Nuniverses)
```

Fig. 2. The script `P-pop.py` which can be used to simulate an exoplanet population.

names of the new P-pop while the first line contains the column names of the old P-pop (`https://github.com/kammerje/P-pop-old-version`).

4.2. Reading a planet population table

P-pop comes with the class `PlanetPopulation` which is implemented in the script `ReadPlanetPopulation.py` and can be used to read a planet population table including its photometry.

An example on how to use the class `PlanetPopulation` is given in the script `TestPlanetPopulation.py` which reads the test planet population table including its photometry which are provided as part of the P-pop library (cf. Figure 5).

Acknowledgements. Acknowledgements.

References

Kammerer, J. & Quanz, S. P. 2018, A&A, 609, A4
Quanz, S. P., Absil, O., Angerhausen, D., et al. 2019, arXiv e-prints, arXiv:1908.01316
Quanz, S. P., Kammerer, J., Defrère, D., et al. 2018, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 10701, Proc. SPIE, 107011I

```
"""
# ============================================================================
# P-POP PHOTOMETRY
# A photometry tool for P-POP
#
# Authors: Jens Kammerer, Sascha Quanz, Emile Fontanet
# Version: 5.0.0
# Last edited: 15.04.2020
# ============================================================================
#
# P-pop is introduced in Kammerer & Quanz 2018
# (https://ui.adsabs.harvard.edu/abs/2018A%26A...609A...4K/abstract). Please
# cite this paper if you use P-pop for your research.
#
# P-pop makes use of forecaster from Chen & Kipping 2017
# (https://ui.adsabs.harvard.edu/abs/2017ApJ...834...17C/abstract).
"""


# Don't print annoying warnings. Comment out if you want to see them.
import warnings
warnings.filterwarnings('ignore')


# ============================================================================
# IMPORTS
# ============================================================================

# Import your own filters and photometry tools here.
import PhotometryComputer
from Filters import SVO
from Star import Blackbody
from Planet import Thermal, Reflected


# ============================================================================
# SETUP
# ============================================================================

# Select the filters and photometry tools which you want to use here.

# Select the name of the planet population table for which the photometry
# should be computed here.
PathPlanetTable = '../P-pop/TestPlanetPopulation.txt' # str

# Select the filters for which the photometry should be computed here. You can
# simply use the filter names from the Spanish Virtual Observatory
# (http://svo2.cab.inta-csic.es/theory/fps/).
# list of str
SVOids = ['JWST/MIRI.F560W',\
          'JWST/MIRI.F1000W',\
          'JWST/MIRI.F1500W'] # used for LIFE
#SVOids = ['Paranal/SPHERE.ZIMPOL_V',\
#          'Paranal/SPHERE.IRDIS_B_J',\
#          'Paranal/SPHERE.IRDIS_B_H'] # used for HabEx/LUVOIR

# Select the photometry tools to compute the fluxes from the stars and the
# planets as well as their unit and the wavelength range in which the mission
# is operating here.
Sstar = [Blackbody] # list of Star
Splanet = [Thermal, Reflected] # list of Planet
Unit = 'uJy' # micro-Jansky
#Unit = 'ph' # photons per second per square meter
Mission = 'MIR' # use AgeomMIR for reflected light (used for LIFE)
#Mission = 'VIS' # use AgeomVIS for reflected light (used for HabEx/LUVOIR)
```

Fig. 3. Continued on next page...

```
# Select whether you want to display summary plots after loading the filters
# and models selected above.
SummaryPlots = True
#SummaryPlots = False


# ============================================================================
# P-POP PHOTOMETRY
# ============================================================================

# Don't modify the following code.
Filters = []
for i in range(len(SVOids)):
    Filters += [SVO.getFilter(SVOids[i], SummaryPlots)]

PhotComp = PhotometryComputer.PhotometryComputer(PathPlanetTable,
                                                 Filters,
                                                 Sstar,
                                                 Splanet,
                                                 Unit,
                                                 Mission,
                                                 SummaryPlots)
PhotComp.Run()
```

Fig. 4. The script `P-pop_Photometry.py` which can be used to compute the photometry for a given planet population table.

Table 1. Content of the planet population table.

Header	Header (old)	Content	Unit
Nuniverse	nMC	Number of the universe to which the planet belongs to	–
Rp	Rp	Planet radius	$R_{\rm Earth}$
Porb	Porb	Planet orbital period	d
Mp	Mp	Planet mass	$M_{\rm Earth}$
ep	ecc	Planet eccentricity	–
ip	inc	Planet inclination	rad
Omegap	Omega	Planet longitude of ascending node	rad
omegap	omega	Planet argument of periapsis	rad
thetap	theta	Planet true anomaly	rad
Abond	Abond	Planet Bond albedo	–
AgeomVIS	AgeomVIS	Planet geometric albedo in the visible	–
AgeomMIR	AgeomMIR	Planet geometric albedo in the mid-infrared	–
z	zodis	Exozodiacal dust level	–
ap	a	Planet semi-major axis	au
rp	rp	Planet physical separation	au
AngSep	ang_sep	Planet projected angular separation	arcsec
maxAngSep	max_ang_sep	Max planet projected angular separation	arcsec
Fp	Finc	Planet incident host star flux	$S_{\rm Earth}$
fp	f	Planet Lambertian reflectance	–
Tp	Tp	Planet equilibrium temperature	K
Nstar	nstar	Number of the star	–
Rs	Rs	Host star radius	$R_{\rm Sun}$
Ms	Ms	Host star mass	$M_{\rm Sun}$
Ts	Ts	Host star effective temperature	K
Ds	dist	Host star distance	pc
Stype	stype	Host star spectral type	–
RA	ra	Host star right ascension	deg
Dec	dec	Host star declination	deg
lGal	lGal	Host star galactic longitude	deg
bGal	bGal	Host star galactic latitude	deg

```
"""
# ============================================================================
# P-POP
# A Monte-Carlo tool to simulate exoplanet populations
# ============================================================================
"""


# ============================================================================
# IMPORTS
# ============================================================================

import ReadPlanetPopulation as RPP


# ============================================================================
# SETUP
# ============================================================================

# Select the name of the planet population table to be read.
PathPlanetTable = 'TestPlanetPopulation.txt' # str

# Select a model for the computation of the habitable zone.
Model = 'MS'
#Model = 'POST-MS'

# Select the name of the photometry tables to be read and give a tag to each
# of them (leave empty if you don't want to read any photometry tables).
PathPhotometryTable = ['TestPlanetPopulation_MIRI.F560W.txt',
                       'TestPlanetPopulation_MIRI.F1000W.txt',
                       'TestPlanetPopulation_MIRI.F1500W.txt'] # list of str
Tag = ['F560W', 'F1000W', 'F1500W'] # list of str


# ============================================================================
# READ PLANET POPULATION
# ============================================================================

# The next five lines read the planet population table and its photometry.
PP = RPP.PlanetPopulation(PathPlanetTable)
PP.ComputeHZ(Model)
for i in range(len(PathPhotometryTable)):
    PP.appendPhotometry(PathPhotometryTable[i],
                        Tag[i])

print('Number of planets in the planet population table')
print(len(PP.Rp))

print('Planet radius (Rearth) of the first planet in the planet population table')
print(PP.Rp[0])

print('Host star radius (Rsun) of the first planet in the planet population table')
print(PP.Rs[0])

print('Display the header of the '+Tag[0]+' photometry')
print(PP.Phot[Tag[0]]['HEAD'])

print('Display the data of the '+Tag[0]+' photometry of the first planet in the planet
    population table')
for i in range(len(PP.Phot[Tag[0]]['HEAD'])):
    print(PP.Phot[Tag[0]]['HEAD'][i]+': '+str(PP.Phot[Tag[0]]['DATA'][i][0]))

import pdb; pdb.set_trace()
```

Fig. 5. The script `TestPlanetPopulation.py` which reads the test planet population table including its photometry which are provided as part of the P-pop library.