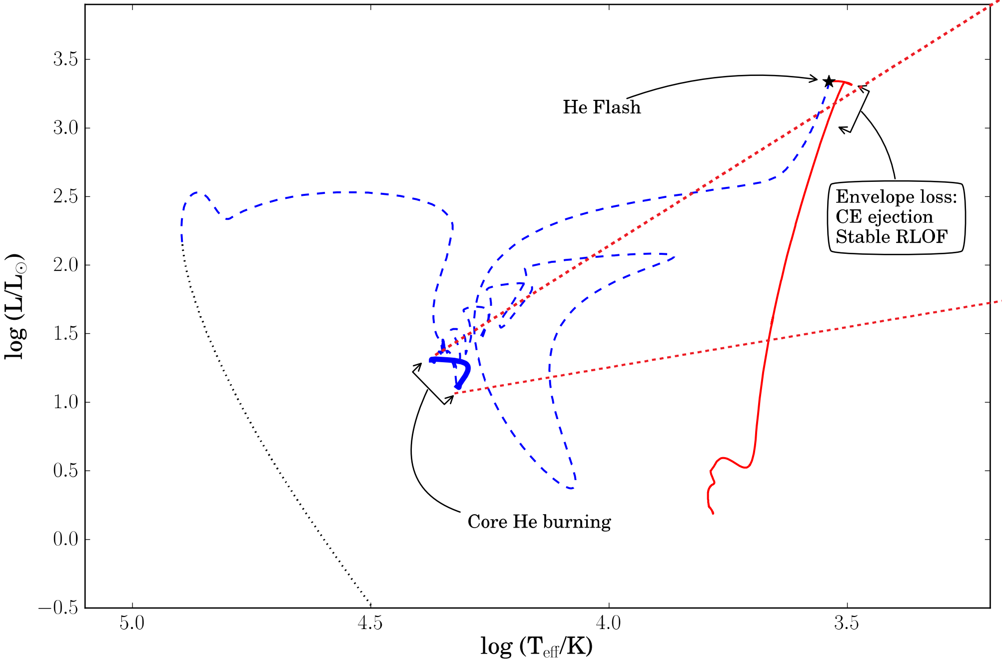
Neural Network assisted Population Synthesis studies

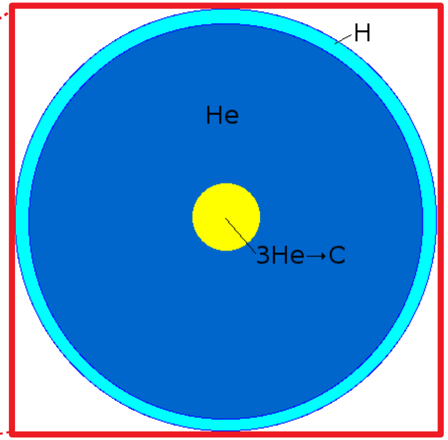
Annual meeting of the German Astronomical Society:
Machine learning methods in astronomy from solar systems to cosmology

23 September 2020

Joris Vos, Alexey Bobrick, Maja Vučković

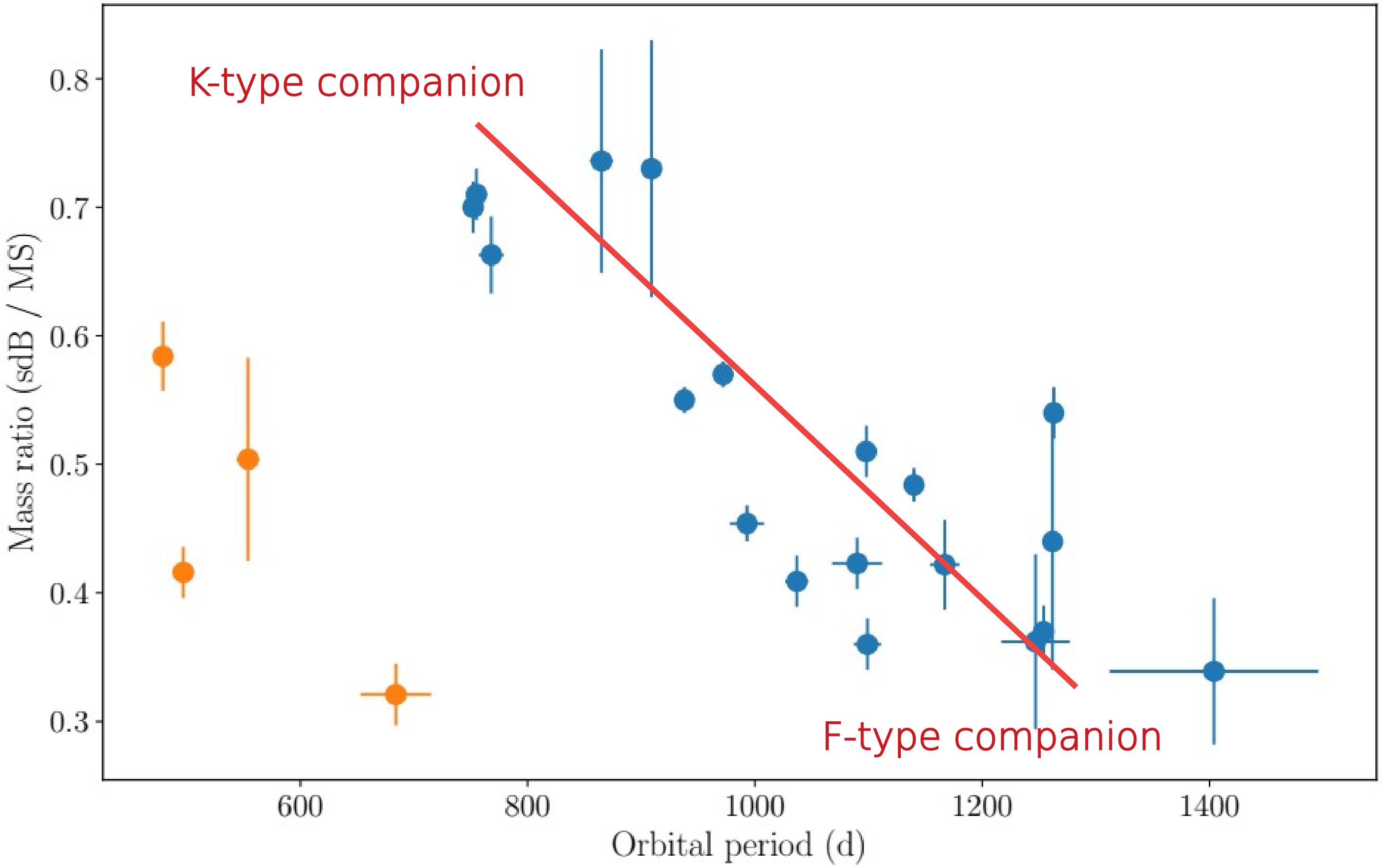
Hot subdwarf-B stars





- Look like B-type stars with high surface gravity
- Located at extreme blue end of the horizontal branch
- Evolved low mass stars, < 2 Msol
- Core He burning
- Lost the majority of their H envelope on the RGB

Observed period – mass ratio



23 systems with well defined P & q.

Two groups are visible with similar P-q correlation

Longer orbital period corresponds to heavier companion

Vos et al. 2019 MNRAS 482 459

Population synthesis studies

Population synthesis codes

e.g.: BSE (Hurley+2000 MNRAS 315 543)

- ✗ Approximated physics through fitting functions
- ✗ Low time resolution
- ✔ Very fast (<sec / model)

1D stellar evolution codes

e.g.: MESA (Paxton+2011 ApJS 192 3)

- ✔ Accurate physics, fully calculated models.
- ✔ High time resolution
- ✗ Very slow (~hours / model)

Problem: Many BPS codes could not produce hot subdwarf stars due to limited physics

Best case scenario:
A FAST and ACCURATE code that can produce sdBs

NNaPS

Create models for the systems under study

↓

extract observable parameters

↓

compare with real observations

↓

Interpret the difference and update the physics

Typical population synthesis codes are created by using pre-computed or fitted stellar evolution models together with some extra physics to determine the evolution of many systems.
Binary evolution is added on top of this.

Idea: Automatize this process using machine learning

NNaPS

Neural Network assisted Population Synthesis package

Python package to simplify using a 1D stellar evolution code as a population synthesis code

Easily extract aggregated parameters from a large grid of MESA models

Apply different stability criteria and CE formalisms

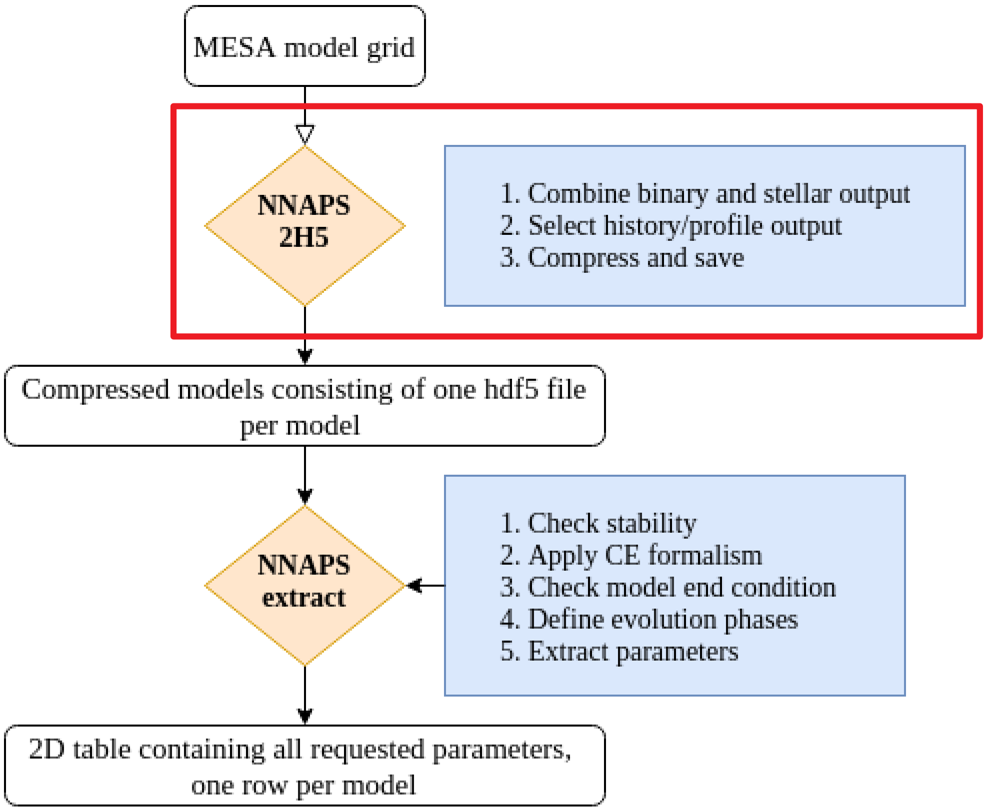
Train a Neural Network or a random forest to be used as a population synthesis code (non linear interpolation)

```
>>> pip install nnaps
```

https://nnaps.readthedocs.io

https://github.com/vosjo/nnaps

NNaPS-MESA



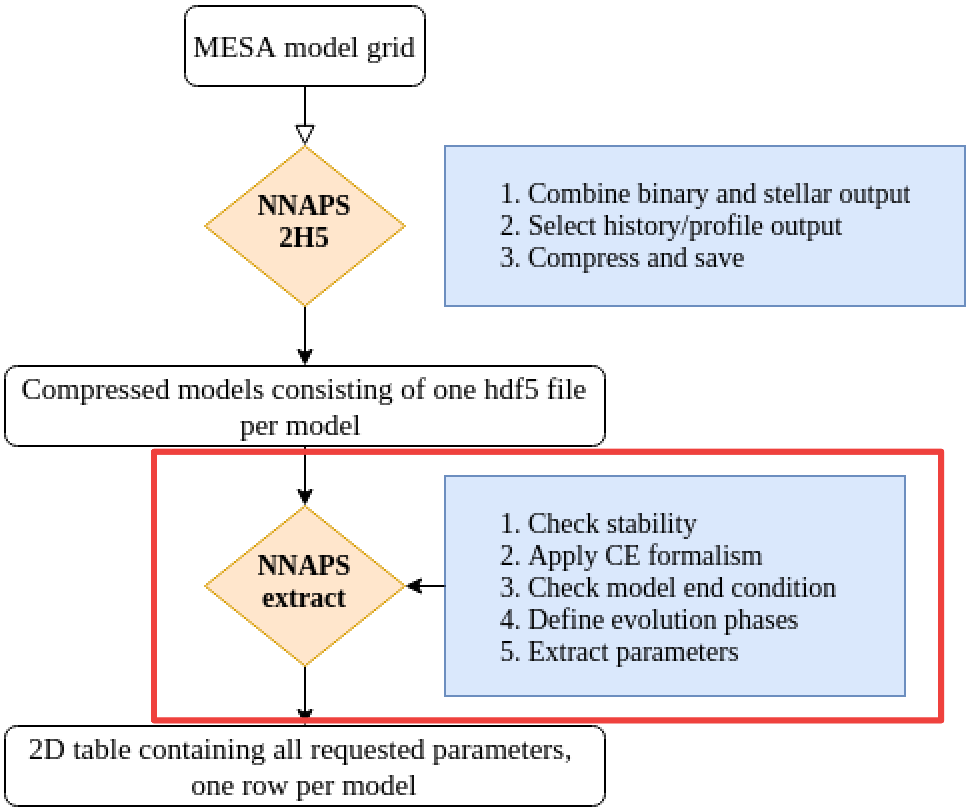
Compress MESA models

Select what output you really need and only keep that

Outputs in HDF5 format

Saves a lot of space (up to tenfold reduction)
→ useful when working on a laptop

NNaPS-MESA



Extract Parameters

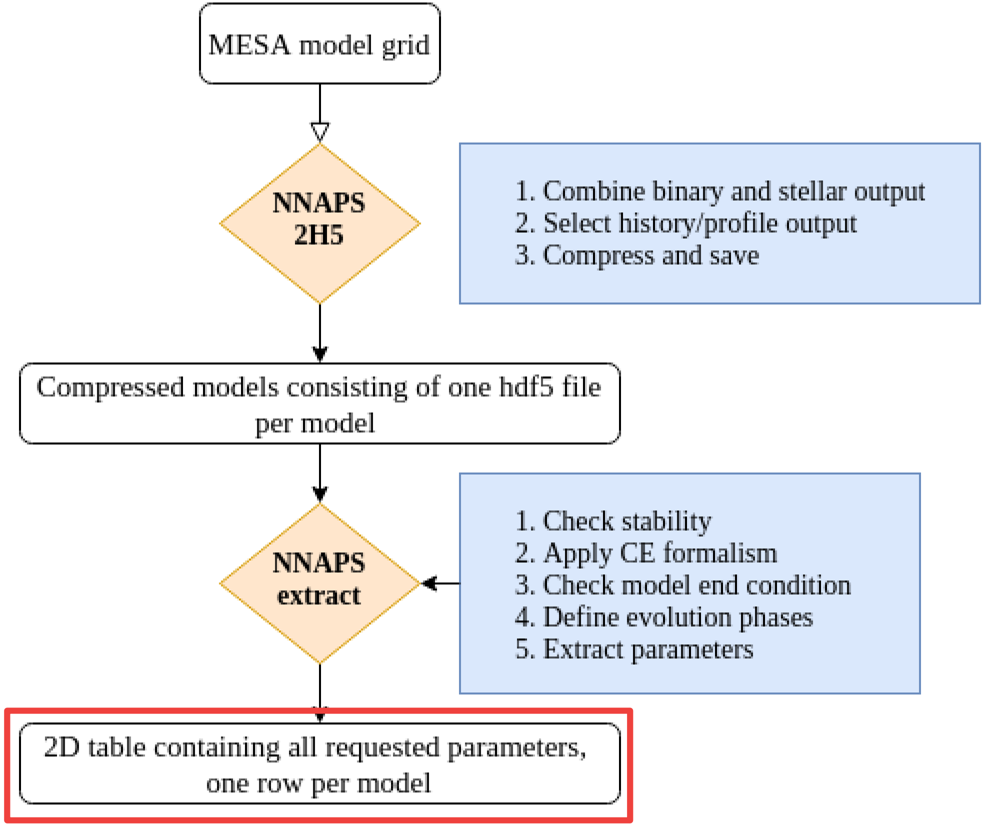
Define which stability criteria, CE formalism to use

Define aggregate parameters:

```
<parameter>__<phase>__<function>
```

- star_1_mass__init
  → initial mass
- he_core_mass__HeIgnition
  → core mass at He ignition phase
- age__HeCoreBurning__diff
  → duration of He core burning phase
- lg_mstar_dot_1__max
  → maximum mass loss rate

NNaPS-MESA



Results in a csv table with one row for each model and one column for each parameter

stability	P_init	P_He	a_He	M1_init	M1_He	M2_init
contact	116.330	0.0230	0.38802	1.5340	0.3972	0.272
stable	43.040	785.3573	573.62290	2.8940	0.5358	2.882
contact	131.820	0.2535	1.54305	1.9070	0.3615	0.925
stable	60.250	328.6860	227.38488	1.1210	0.4134	1.048
contact	56.480	0.1151	0.91781	1.8190	0.3325	0.931
contact	151.340	0.0292	0.47904	1.2180	0.3577	0.204
stable	188.960	1137.7345	590.74436	1.5460	0.5046	1.492
stable	32.380	525.4006	370.94182	2.7720	0.5020	1.981
stable	30.610	421.8050	346.38695	2.8640	0.6529	2.484
CE	433.360	0.5942	2.99529	1.4150	0.4543	0.476
...	...	...	...	...	...	...

NNaPS-MESA

Evolution phases:

MS, RGB, He ignition, He Core burning, He shell burning,
sdB, sdO, Horizontal Branch, He-WD
Mass loss phases, CE phase

Stability criteria based on:

mass ratio, mass loss rate, L3 mass loss, radius/separation,
mass lost per orbit, angular momentum lost per orbit

CE formalisms:

- Iben & Tutukov 1984, ApJ, 284, 719
- Webbink 1984, ApJ, 277, 355
- Dewi and Tauris 2000, A&A, 360, 1043 (profile integration)
- De Marco et al. 2011, MNRAS, 411, 2277

Actively being extended!

NNaPS predictions

Easy training of XG boosted trees and Neural Networks using the extracted parameters from nnaps-mesa.

Takes care of features scaling and encoding where necessary

Works out out of the box but also allows manually fine tuning parameters

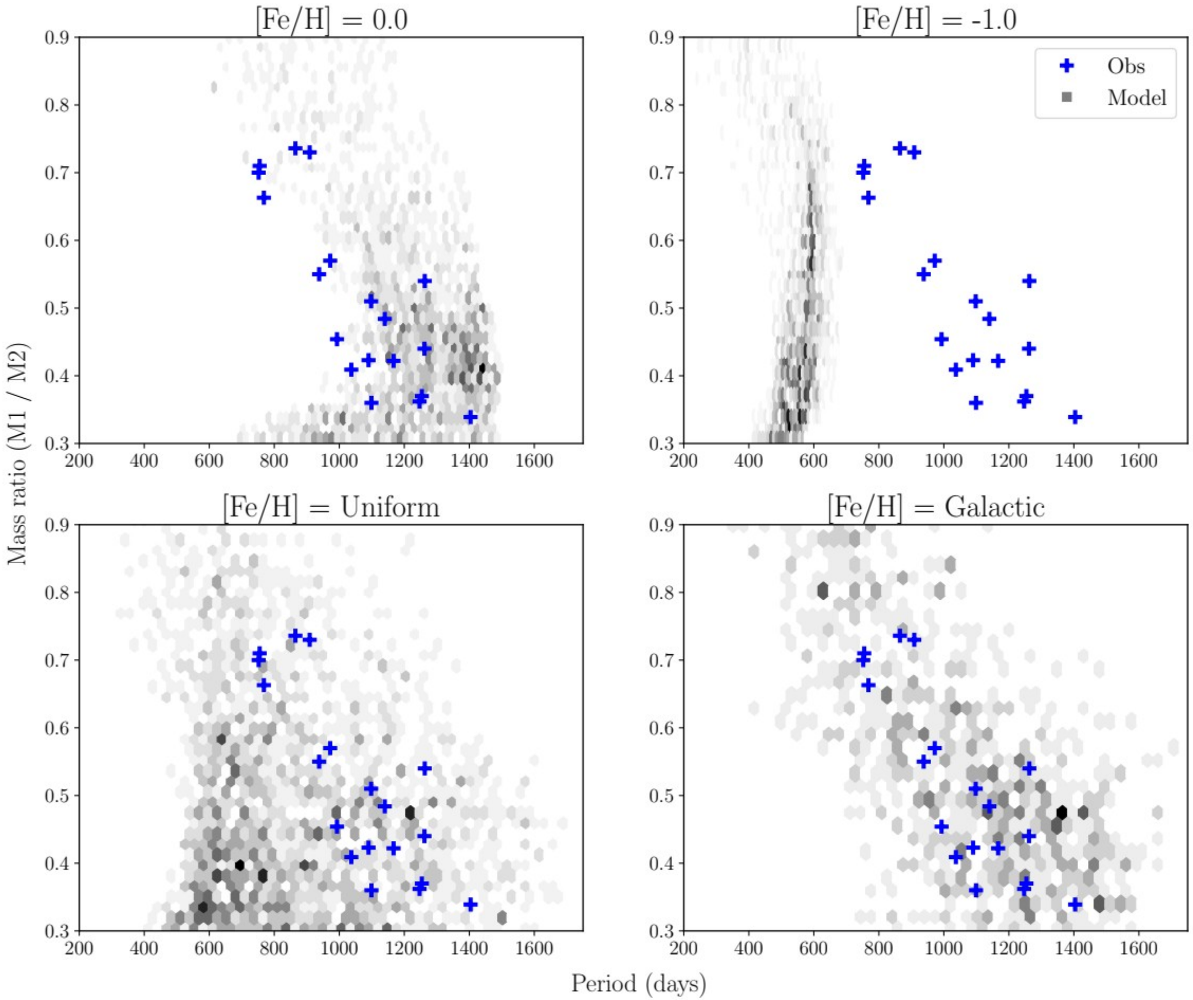
Build on top of Scikit learn and Keras/TensorFlow.

```
from nnaps import predictors

setup = {
    'datafile': <path to csvfile>,
    'features': ['donor_mass', 'initial_period', 'initial_q'],
    'regressors': ['final_period', 'final_q'],
    'classifiers': ['product_type']
}

predictor = predictors.FCPredictor(setup=setup)
predictor.fit()
new_predictions = predictor.predict(new_data)
```

Example: P-q relation in sdB binaries



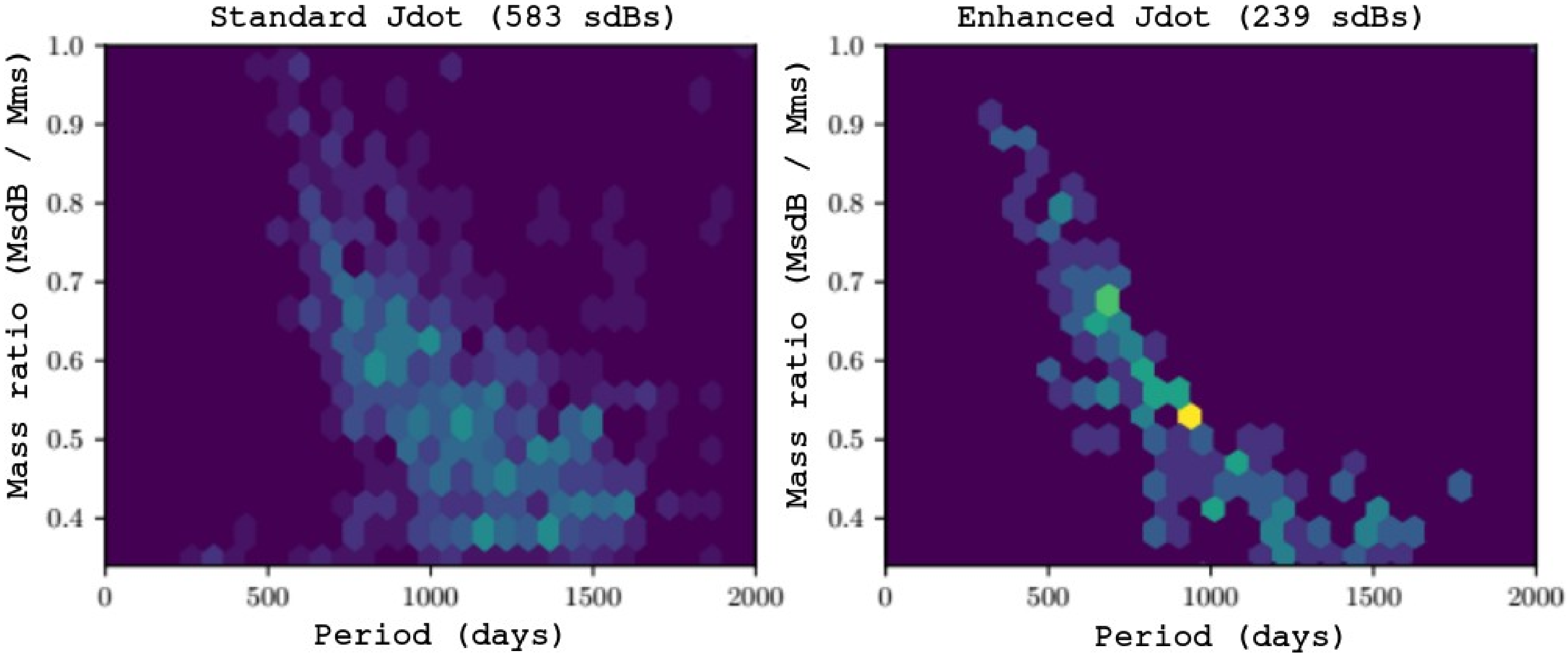
Orbital periods of hot subdwarfs are very strongly correlated with their mass ratio.

Why: combination of interaction physics and Galactic evolution.

→ 2000 MESA models randomly distributed in initial parameters
→ Used NNaPS to train a NN
→ 20000 models / case with NN

Vos et al. 2020, A&A, in press

Example: Jdot during RLOF



→ 1500 MESA models, half standard, half enhanced.
→ Same initial population (10000 systems) for each Jdot setting with NN

Angular momentum loss does not influence the final orbital period reached, but combined with an galactic initial distribution not all final orbits are possible

Summary

We developed a different approach to population synthesis studies

- Use a 1D hydrodynamical code (MESA) to calculated just enough models to span your parameter space
- Use NNaPS to extract parameters of interested and train a NN to act as interpolator in those parameters
- Combine with the required initial population.

Not a total replacement for Population synthesis codes, but has its use cases.

https://nnaps.readthedocs.io https://github.com/vosjo/nnaps