Draft version September 7, 2024
Typeset using LATEX twocolumn style in AASTeX631

cogsworth: A Gala of COSMIC proportions combining binary stellar evolution and galactic dynamics

Tom Wagg [1,2], Katelyn Breivik [3], Mathieu Renzo [4] and Adrian M. Price-Whelan [2]

[1] Department of Astronomy, University of Washington, Seattle, WA, 98195
[2] Center for Computational Astrophysics, Flatiron Institute, 162 Fifth Ave, New York, NY, 10010, USA
[3] McWilliams Center for Cosmology and Astrophysics, Department of Physics, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[4] University of Arizona, Department of Astronomy & Steward Observatory, 933 N. Cherry Ave., Tucson, AZ 85721, USA

(Received September 7, 2024)

ABSTRACT

We present cogsworth, an open-source Python tool for producing self-consistent population synthesis and galactic dynamics simulations. With cogsworth one can (1) sample a population of binaries and star formation history, (2) perform rapid (binary) stellar evolution, (3) integrate orbits through the galaxy and (4) inspect the full evolutionary history of each star or compact object, as well as their positions and kinematics. We include the functionality for post-processing hydrodynamical zoom-in simulations as a basis for galactic potentials and star formation histories to better account for initial spatial stellar clustering and more complex potentials. Alternatively, several analytic models are available for both the potential and star formation history. cogsworth can transform the intrinsic simulated population to an observed population through the joint application of dust maps, bolometric correction functions and survey selection functions.

We provide a detailed explanation of the functionality of cogsworth and demonstrate its capabilities through a series of use cases: (1) We predict the spatial distribution of compact objects and runaways in both dwarf and Milky-Way-like galaxies, (2) using a star cluster from a hydrodynamical simulation, we show how supernovae can change the orbits of stars in several ways, and (3) we predict the separation of disrupted binary stellar companions on the sky and create a synthetic *Gaia* colour-magnitude diagram. We also discuss some current limitations and plans for future developments. We designed cogsworth and its online documentation to provide a powerful tool for constraining binary evolution, but also a flexible and accessible resource for the entire community.

1. INTRODUCTION

The majority of stars are born in binaries and multiple star systems (e.g., Duchêne & Kraus 2013; Moe & Di Stefano 2017; Offner et al. 2023), a large subset of which will exchange mass at some point in their lifetime (e.g Podsiadlowski et al. 1992; Sana et al. 2012; de Mink et al. 2014). These massive stars play a critical role in the formation and evolution of galaxies as a result of their feedback (e.g., Dekel & Silk 1986; Hopkins et al. 2012; Nomoto et al. 2013; Somerville & Davé 2015; Naab & Ostriker 2017).

However, binary evolution remains uncertain, with many parameters such as common-envelope efficiency, mass transfer efficiency, angular momentum loss due to mass transfer and the mean magnitude of supernova natal kicks unconstrained across several orders of magnitude (e.g., Janka 2012; Ivanova et al. 2013; Katsuda et al. 2018; Ivanova et al. 2020; Röpke & De Marco 2023; Marchant & Bodensteiner 2023).

Single massive stars are not expected to migrate far from their birth location before reaching core-collapse due to their short lifetimes ($\lesssim 50\,$Myr, e.g., Zapartas et al. 2017). However, binary stars may disrupt after an initial supernova event, ejecting the secondary star from the system with its orbital velocity (e.g., Blaauw 1961; Eldridge et al. 2011; Renzo et al. 2019). Thus, close massive binaries that disrupt can lead to the displacement of secondary stars significantly farther from star-forming regions. The present-day positions and kinematics of massive stars and binary products are therefore strongly impacted by changes in binary physics that alter the

Corresponding author: Tom Wagg
tomjwagg@gmail.com

separation prior to supernova. This means that comparing simulations of positions and kinematics of stars and compact objects to observations will enable constraints on binary stellar evolution parameters.

The use of positions and kinematics as tracers of binary evolution has been considered in the past. Recent work has shown the importance of accounting for the galactic potential, which can change the velocity of kicked objects (e.g., Disberg et al. 2024). It is also important to consider the inclination or timing of a supernova kick relative to the galactic orbit, since, for example, a kick out of the galactic plane at an object's highest galactic vertical position will have a strong effect on its final position. Failing to consider impacts from both a galactic potential and kicks (i.e. velocity impulses) will lead to misleading conclusions regarding the final spatial distributions of the population. Some studies have considered using the Galactic potential at the present-day position of objects to place a lower limit on the peculiar velocity at birth and constrain supernova kicks (Repetto et al. 2012; Repetto & Nelemans 2015; Repetto et al. 2017; Atri et al. 2019), but the accuracy of this method is debated (Mandel 2016). Other work has considered the impact of the Galactic potential for individual special cases, rather than at a population level. For example, Evans et al. (2020) considered the orbits of hyper-runaway candidates evolving through the Milky Way potential, whilst Neuhäuser et al. (2020) developed software for tracing the motion of stars to investigate the recent nearby supernovae that ejected ζ Ophiuchi.

In this paper we present `cogsworth`, a new open-source tool for self-consistent population synthesis and galactic dynamics simulations. `cogsworth` provides the theoretical infrastructure for making predictions for the positions and kinematics of massive stars and compact objects, placing these systems in the context of their host galaxy and its gravitational potential. The code is applicable to a wide range of binary products, both common and rare, from walkaway and runaway stars to X-ray binaries, as well as gravitational-wave and gamma-ray burst progenitors.

The paper is structured as follows: in Section 2 we explain the functionality of `cogsworth` and describe its primary features and capabilities. We demonstrate these capabilities in a series of example use cases in Section 3. We use `cogsworth` to predict the spatial distribution of compact objects and runaways in both dwarf and Milky-Way-like galaxies. Using a cluster from a hydrodynamical simulation, we show how supernovae can change the orbits of stars in several ways. We predict the separation of disrupted binary stellar companions on the sky, as well as create a synthetic *Gaia* colour-magnitude diagram. In Section 4, we discuss the current limitations of the package and we outline planned additional future developments in Section 5.

2. COGSWORTH

`cogsworth` is a code that combines binary population synthesis simulations (via `COSMIC`, Breivik et al. 2020a) with galactic dynamics (via `Gala`, Price-Whelan 2017) to self-consistently use stellar and orbital evolution to rapidly derive present day positions, kinematics and demographics for complete populations of binary stars and their descendants.

Our code is fully open-source and openly-developed (available on GitHub[1]), pip installable (`pip install cogsworth`) and indexed on Zenodo. In this paper we describe v2.0.0 of the code. We wrote `cogsworth` in Python to make it convenient and accessible, but its core dependencies are written in Fortran and C (via `COSMIC` and `Gala` respectively) for efficiency. We use automated testing via a detailed suite of unit tests, with full code coverage. Additionally, we have written thorough documentation of `cogsworth`, including ∼20 tutorials covering full usage of the code, several short examples and a series of longer in-depth case studies, all of which is available online[2].

We describe the specific capabilities in the following subsections, and illustrate an overview of the code in Figure 1. The first subsections focus on core functionality of `cogsworth`, which is accessed via a `Population`, with which one can:

- §2.1 Sample initial galactic positions, velocities, birth times and metallicities from a star formation history model
- §2.2 Sample and evolve a (binary) stellar population until present day
- §2.3 Integrate the orbits of each binary through the galaxy, accounting for supernova kicks and disruptions
- §2.4 Identify observable constituents of the present day intrinsic population

Each of these features are flexible and can be tuned to a particular use case. In addition, in §2.5 we describe how one can alternatively use `cogsworth` to initialise and evolve populations based on hydrodynamical

[1] https://github.com/TomWagg/cogsworth
[2] https://cogsworth.readthedocs.io

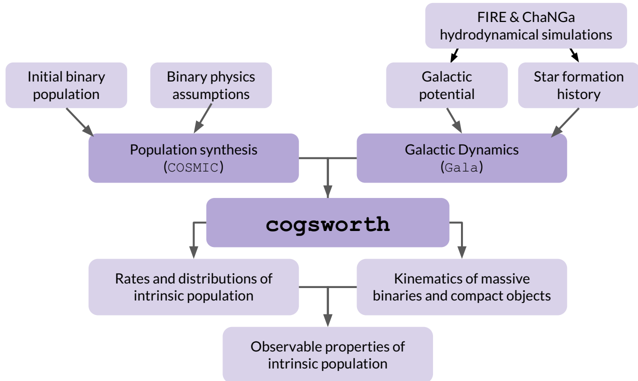

Figure 1. `cogsworth` combines population synthesis and galactic dynamics self-consistently to produce rates, distributions, observables and kinematics of massive binaries and compact objects. A schematic of the input options and outputs of `cogsworth` simulations.

zoom-in simulations. In the later subsections we detail `cogsworth`'s visualisation functionalities ($2.6), details of its typical runtime (§2.7) and data storage (§2.8), and its ability to create custom citation statements for a given simulation (§2.9).

2.1. *Galactic star formation histories*

A galactic star formation history (SFH) defines the distributions of times, locations and metallicities at which stars are formed in a galaxy. In `cogsworth`, one can flexibly adjust the SFH of a simulation with a `StarFormationHistory` Python class. By default, `cogsworth` uses `Wagg2022`, an empirically motivated analytic model for the Milky Way (Wagg et al. 2022c). This model contains a low-[α/Fe] (a.k.a. "thin") disc, high-[α/Fe] (a.k.a. "thick") disc, and bulge, each with their own spatial and temporal distributions. Metallicity is calculated as a function of birth time and galactocentric radius. This SFH accounts for effects such as the inside-out growth growth of the galaxy and radial migration (Frankel et al. 2018, 2019). We explore this SFH further in Section 3.4 and the full details of the model are given in Section 2.2.1 of Wagg et al. (2022c).

Beyond the default SFH, we include simpler parameterised SFHs, such as `BurstUniformDisc`, in which stars are formed in a single burst of star formation with a fixed metallicity in a uniform disc. Additionally, we include action-based SFHs using `Agama` (Vasiliev 2019), such as `QuasiIsothermalDisc`, which represents a quasi-isothermal distribution function for the Milky Way disc as described in Sanders & Binney (2015).

Each SFH class is designed to be modular and flexible and as such, they can be entirely customised. Users can overwrite individual distributions in a given SFH (such as changing the birth time distribution of the bulge component), or define their own entirely custom SFH. We explain how to accomplish this in a tutorial.

For each binary or single star, i, in a population, we use the SFH to draw the initial galactic parameters

$$g_i = \{\tau, R, z, \phi, Z\}, \tag{1}$$

where τ is the lookback time (the time before the present day when the system formed), R is the initial galactocentric radius, z is the initial height above the plane, ϕ is the azimuthal angle, and Z is the metallicity. For an SFH that doesn't explicitly define a distribution for galactocentric velocities, we assign the initial galactocentric velocity of a system, i, as follows

$$\vec{\mathbf{v}}_i = \vec{\mathbf{v}}_{\rm circ}(R_i) + \vec{\mathbf{v}}_{\rm disp}, \tag{2}$$

where $\vec{\mathbf{v}}_{\rm circ}$ is the circular velocity for the population's galactic potential and $\vec{\mathbf{v}}_{\rm disp}$ is an isotropic velocity dispersion, which is an input option that by default has a magnitude of $5\,{\rm km\,s^{-1}}$.

2.2. *Stellar population sampling and evolution*

`cogsworth` uses the open-source and community-driven rapid binary population synthesis code `COSMIC` to perform the sampling of the initial (binary) stellar population and the (binary) stellar evolution (Breivik et al. 2020a). `COSMIC` uses fitting formulae based on single stellar tracks originally developed for the Binary Stellar Evolution (`BSE`) code (Tout et al. 1997; Pols et al. 1998; Hurley et al. 2000, 2002) and allows the user to rapidly sample and evolve populations of binaries with a variety of physics assumptions. With `COSMIC` a user can specify dynamic time resolution conditions for its outputs based on binary parameters which, for example, could be used to increase the number of outputted timesteps once a star is stripped, or during mass transfer onto a compact object to investigate X-ray binaries. One can also easily access the initial conditions of a population evolved with `COSMIC`, allowing for convenient reproducibility of simulations. This is additionally useful for re-running identical initial populations with alternative binary physics settings to ascertain how the evolution changes with different settings. Each of these features are directly inherited by `cogsworth`.

When sampling an initial binary stellar population, a user can specify their choice of initial mass function (IMF) for drawing primary masses, a binary fraction and distributions of initial orbital period, eccentricity and mass ratio for a given `Population` of binaries (see Section 2.1.1 of Breivik et al. 2020a for all available options). Metallicities are set based on the chosen SFH model (see Section 2.1). Using these distributions, one can either draw a fixed number of systems, or specify a total mass to sample. This tutorial explains in detail how to change these settings.

`cogsworth` evolves this initial binary population from their individual birth times until present day with a user-specified choice of binary physics. Any binary stellar evolution parameter that can be supplied to `COSMIC` can also be specified in a `Population` in `cogsworth`. These parameters cover a range of binary physics including stellar wind mass and accretion, mass transfer through Roche-lobe overflow, common-envelopes, supernova kicks, remnant mass prescriptions and tides. For a full list of the available parameters see the `COSMIC` documentation and learn about changing them in `cogsworth` with this tutorial.

2.3. *Galactic orbit integration*

`cogsworth` applies the galactic dynamics package `Gala` for the galactic orbit integration of binaries (Price-Whelan 2017). This package allows users to integrate the orbits of sources rapidly with user-friendly functions wrapped on low-level code (primarily C) for fast computations. One can choose from numerous, flexible potentials (or even define custom potentials) through which to integrate orbits. `cogsworth` uses `Gala` to integrate the full orbit of each binary in a population through a given galactic potential. By default, `cogsworth` uses the `MilkyWayPotential2022` potential, which is fit to observations of the Milky way rotation curve, the shape of the phase-space spiral in the solar neighbourhood and a compilation of recent mass measurements of the Milky Way (Eilers et al. 2019; Darragh-Ford et al. 2023).

For systems that experience supernovae, `cogsworth` accounts for the resulting changes in velocity. `COSMIC` logs the velocities imparted by Blaauw (Blaauw 1961) and natal kicks (Katz 1975; Janka 2012, 2017) and whether a binary is disrupted by a supernova (e.g., Renzo et al. 2019). In each case, `cogsworth` transforms the resulting velocities to galactocentric coordinates (uniformly sampling a random orbital phase, θ, of the binary and inclination, ι, of the binary relative to the galaxy) and updating the orbit of the system. In the case of disruptions, a second orbit is produced for the secondary, tracking the binary position until the disruption and then the subsequent motion of the secondary.

Overall, this allows users to track the location of either star in a binary system at *any* point in its evolutionary history. This can be used to, for instance, predict the location of supernovae or track the sites of r-process enrichment from binary mergers.

2.4. *Observables estimation*

Key to applying `cogsworth` to realistic problems and constraining our models is being able to compare simulations to observations. In this Section, we explain how users can transform intrinsic `cogsworth` populations into observables.

2.4.1. *Electromagnetic observations*

We have implemented functionality to translate intrinsic stellar parameters in `cogsworth` populations (such as mass, luminosity and galactic position) into observables (such as fluxes and colours). Currently, `cogsworth` focuses on producing predictions for *Gaia* observables, but we intend to build on this with other instruments in future (see Section 5.3).

`cogsworth` can compute the magnitude of sources in arbitrary filters by applying bolometric corrections and dust extinctions, achieved through a combination of the `dustmaps` and `isochrones` packages (Green 2018; Morton 2015). You can follow this tutorial to learn how to do this. Within the Milky Way, the interplay between distance, the 3-dimensional dust distribution, and the *Gaia* scanning pattern leads to a complex selection function, but one that can be captured through the empirical selection function made available through `gaiaunlimited` (Cantat-Gaudin et al. 2023). `cogsworth` is therefore capable of predicting whether a given source (either a bound binary or star from a disrupted binary) would be detectable by *Gaia*. This tutorial explains how to make predictions about the observable *Gaia* population.

2.4.2. *Gravitational waves*

In addition to electromagnetic observations, we consider gravitational wave detections from the inspiral of double compact objects. Stellar-mass binaries in the Milky Way will be detectable by LISA via millihertz gravitational wave emission (Amaro-Seoane et al. 2017). The `LEGWORK` package allows users to compute gravitational-wave strains and SNRs for binaries, and calculate the evolution of binary separations and eccentricities due to gravitational wave emission (Wagg et al. 2022a,b). We connect `cogsworth` to `LEGWORK`, allowing users to quickly calculate the LISA SNR of each binary in a population, as well as the time until its merger with a single, simple function call. This tutorial shows an example of calculating LISA SNRs for a `cogsworth` population of double white dwarfs.

2.5. *Building off hydrodynamical zoom-in simulations*

Using an analytic model for a galactic SFH can work well for longer-lived populations, which can be expected to have erased all memory of their initial positions. However, it is unrealistic for younger stars, and in particular short-lived massive ones, which should retain significant initial spatial clustering and correlations with the surrounding ISM (e.g., Sarbadhicary et al. 2023). Such correlations are particularly important when attempting to constrain aspects of binary physics by comparing predicted present-day locations from `cogsworth` models to observations of recently formed binaries.

Motivated by a need for more detailed initial spatial clustering, we include the option to initialise a `cogsworth` population using hydrodynamical zoom-in simulations. These simulations are not only used to set the locations and times of star formation, but also the galactic gravitational potential.

2.5.1. *Compatible simulations*

We currently support the post-processing for two different suites of hydrodynamical zoom-in simulations. One can use any of the public `FIRE` simulations (Wetzel et al. 2016; Hopkins et al. 2018; Sanderson et al. 2020), which have been connected to population synthesis and cluster models successfully in the past (e.g., Lamberts et al. 2018; Chawla et al. 2022; Grudić et al. 2023; Rodriguez et al. 2023; Thiele et al. 2023). Additionally, one can use simulations from `ChaNGa`, such as the MARVEL-ous dwarfs and DC Justice League simulations (Applebaum et al. 2021; Christensen et al. 2023). These simulation suites directly resolve the formation of giant molecular clouds and the interstellar medium (ISM), and thus capture the characteristic spatial clustering of star formation. The simulations additionally explicitly account for the feedback from stars, following predictions laid out by stellar population synthesis models - though neglecting the impact of binary evolution on the timing and positioning of supernovae (see Wagg et al. in prep).

2.5.2. *Snapshot preparation*

Hydrodynamical simulations record snapshots of the state of the simulation a specific times. `cogsworth` provides a wrapper over `pynbody` (Pontzen et al. 2013) in order to prepare simulation snapshots for use as initial conditions to simulations. This functionality centres snapshots on the primary halo, using either an automatically detected halo catalogue or applying a shrinking sphere method to iteratively refine an estimate of the centre of the mass of the simulation. It additionally rotates the halo to be edge-on and then face-on, and converts data to physical units.

Galactic potential—As with a regular `Population`, before initialisation one needs to know the galactic potential and SFH of the galaxy. We provide functionality for computing a galactic potential from a simulation snapshot, accounting for stars, gas and dark matter, using the self-consistent field method implemented in `Gala` based on Hernquist & Ostriker (1992) and Lowing et al. (2011). This method fits the galactic mass distribution using a basis function expansion in spherical harmonics.

Initial stellar positions—The formation locations of star particles are necessary for sampling the initial positions of binary stellar populations. `cogsworth` can identify these formation locations by backwards integrating the orbits of star particles through the galactic potential derived from the simulation. Note that this step is only necessary for `FIRE` simulations, since `ChaNGa` simulations store formation locations.

For more information on processing simulation snapshots in `cogsworth` we refer interested readers to this tutorial.

2.5.3. *Population initialisation and evolution*

A `cogsworth` `Population` based on the star particles and galactic potential of a hydrodynamical zoom-in is called a `HydroPopulation`. Each star particle in a hydrodynamical simulation can represent many 100–1000s of solar masses. Given this, we use `COSMIC` to sample binary stellar populations from each star particle, assigning each system the same formation time and metallicity as the star particle. Based on a user-defined star particle radius, each sampled system is assigned a random position, $\vec{\mathbf{p}}_i$, from a Gaussian centred on the parent star particle such that

$$\vec{\mathbf{p}}_i = \mathcal{N}(\vec{\mathbf{p}}_{\mathrm{sp},i}, r), \tag{3}$$

where $\vec{\mathbf{p}}_{\mathrm{sp},i}$ is the position of star particle from which system i was sampled and r is the user choice of radius. Similarly, based on a user's choice of virial parameter, α_{vir} (the ratio of kinetic and gravitational energy of a cluster, as defined in Bertoldi & McKee 1992) a velocity dispersion of each star particle is determined and used for sampling initial velocities, $\vec{\mathbf{v}}_i$, such that

$$\vec{\mathbf{v}}_i = \vec{\mathbf{v}}_{\mathrm{sp},i} + \vec{\mathbf{v}}_{\mathrm{disp},i}, \tag{4}$$

where $\vec{\mathbf{v}}_{\mathrm{sp},i}$ is the velocity of the star particle from which system i was sampled and

$$\vec{\mathbf{v}}_{\mathrm{disp},i} = \sqrt{\frac{\alpha_{\mathrm{vir}} G M_{\mathrm{cl},i}}{5r}}, \tag{5}$$

where $M_{\mathrm{cl},i}$ is user-defined mass of the star cluster from which system i was formed. Beyond this initial sampling, a `HydroPopulation` has the exact same functionality and methods available as a regular `Population`. A demonstration of evolving a population sampled from a snapshot is given in this tutorial.

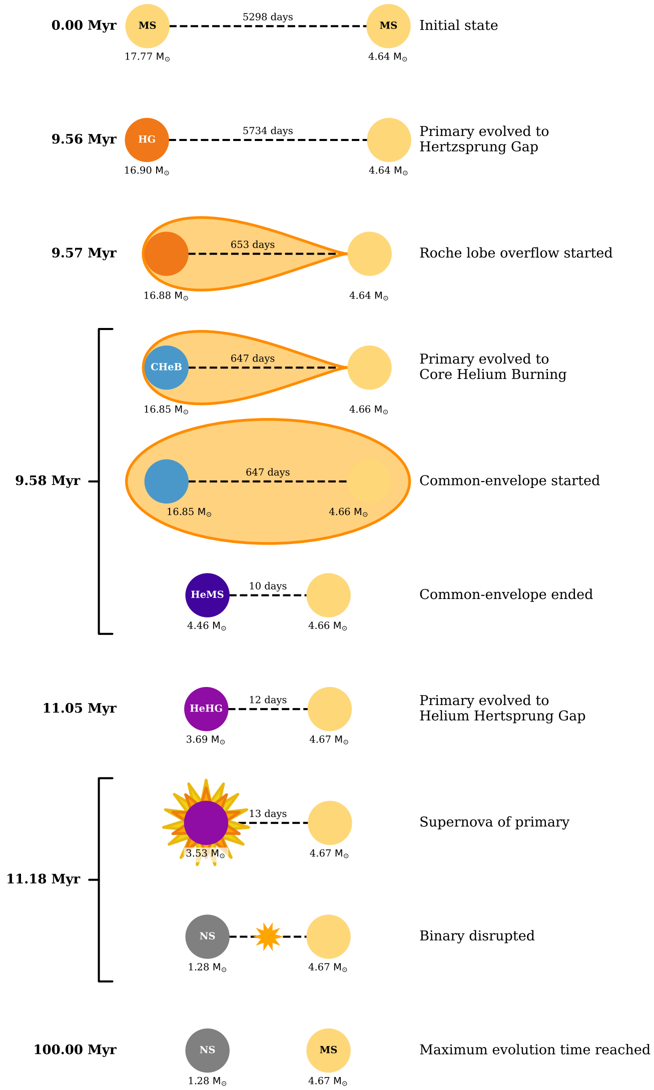

Figure 2. A dynamically generated cartoon binary evolution timeline. Each row shows an evolutionary step, labelled with its time and the event that occurred. Circles are shown for each star, annotated with their masses and the binary's orbital period. A dashed line indicates a bound binary, whilst a lack thereof indicates two unbound stars that were previously in a binary.

2.6. *Visualisation*

`cogsworth` offers several methods for visualising the evolution and end-states of binaries evolved in populations. `COSMIC` and `Gala` already provide useful tools for investigating the binary evolution history and galactic orbits, but `cogsworth` expands these to aid in the interpretation of simulation data.

Binary evolution—For each binary evolved in `cogsworth`, one can dynamically generate a cartoon timeline of its evolution, sometimes called a Van den Heuvel diagram (van den Heuvel 1976), with the function `plot_cartoon_binary()`. This timeline will show the evolutionary history and is capable of illustrating: the masses of each star, orbital period, mass transfer, common-envelope events, contact phases, mergers and supernovae. The distance between each star in the plot is directly scaled by the orbital separation of the binary (on a log-scale). This functionality provides a simple way to interpret the evolution of a binary without needing to know the meanings of each number representing stellar types and evolutionary stages in `COSMIC` output tables. We show an example of this in Figure 2 for a randomly sampled binary that we evolved for 100 Myr. Initially, as the primary loses mass to stellar winds the orbit widens very slightly. However, once the primary ends its main sequence and expands across the Hertzspung gap, it initiates mass transfer. The star continues to evolve during mass transfer, eventually expanding at such a rate to make the mass transfer unstable, leading to a common-envelope and the stripping of the primary star. The primary star reaches supernova after around 11 Myr and the resulting kick disrupts the binary, ejecting the newly formed neutron star and its (prior) companion across the galaxy.

Galactic orbits—`cogsworth` provides a wrapper, `plot_orbit()`, on the `Gala` orbit plotting functionality to allow users to plot projections of a given binary's orbit in galactocentric coordinates. This allows users to plot the orbit of any evolved system, with markers indicating the location of each supernova. For a disrupted system, an additional line will be plotted for the secondary star after the binary disrupts.

Sky maps—For Milky Way simulations, users can plot their simulated populations on the sky with `cogsworth`. This is possible either with a simple scatter plot of right ascension and declination, or with a HEALPix Molleweide heatmap via `healpy` (Zonca et al. 2019; Górski et al. 2005). For HEALPix maps, one can customise the plots in several ways, including choosing the coordinates to plot (celestial, galactic, or equatorial) and the resolution of the map. See Section 3.6 for a demonstration of plotting a simulated population on the sky.

2.7. *Multiprocessing scalability*

Each binary in `cogsworth` is assumed to evolve independently of all other binaries in the simulation (we do not account for dynamical N-body interactions, see Section 4). An advantage of this is that each system can be efficiently parallelised. Leveraging this fact, `cogsworth` uses a multiprocessing pool for the evolution of each binary system. This means that the runtime of simulations scales well with the number of processes used.

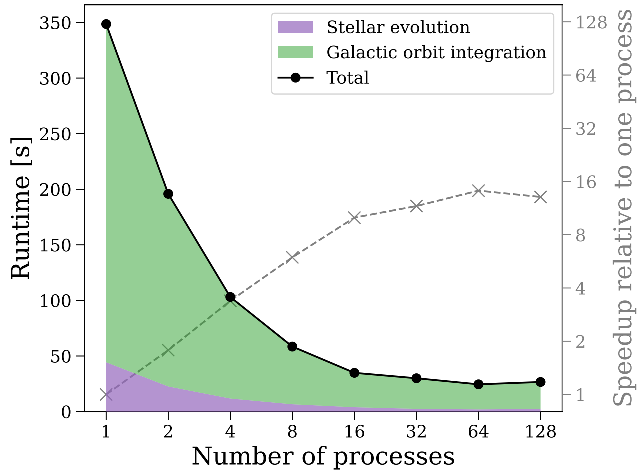

Figure 3. The scaling of `cogsworth`'s runtime with the number of processes used for a fixed population size of 10,000 binaries. The black points indicate the total runtime, whilst the shaded areas show the relative contribution from the stellar evolution and galactic orbit integration. The right y-axis and grey dashed line shows the relative speedup compared to the simulation using a single process.

We demonstrate the scaling of `cogsworth`'s runtime with the number of processes used in Figure 3. We first sample a fixed population of 10,000 binaries to ensure consistency across runs. We do not include this sampling in the runtime analysis since it is typically negligible relative to the evolution. We computed the runtime for a `cogsworth` simulation when using our default settings and varying the number of processes used.

Figure 3 shows that increasing the number of processes can significantly decrease the runtime of a simulation. However, this only continues up to a point, beyond which increasing the number of processes yields diminishing returns. The reason that adding processes does not always increase the runtime is that small subsets of the population take longer to run, such as binaries with multiple interactions between the two stars, or orbits that pass close to the galactic centre requiring finer time-sampling for the orbital evolution. Since we do not spread these equally among processes, additional resources are left idle while the complicated subset runs on a limited number of cores. In the case of this test, using more than 16 processes does not improve the performance, though we note that this threshold is dependent on the population (both its size and demographics) and the settings chosen by a user. We recommend that users perform similar tests to ascertain the optimal number of processes for their use case.

In Figure 3 we additionally show the relative contributions to the runtime from performing the stellar evolution and galactic orbit integration. For this example, the galactic orbit integration typically dominates the runtime by around a factor of $\sim$5. The relatively higher cost of the orbit integration is expected given that `COSMIC` relies on pre-computed stellar models for stellar evolution, whilst `Gala` fully integrates galactic orbits. The relative contributions from the two phases depend on the simulation that is run. One with more binaries that have many interactions between companions (e.g. a population focused on massive stars) may increase the runtime of the stellar evolution. Conversely, a simulation using a more complex galactic potential than the smooth `MilkyWayPotential2022` (such as one computed from a hydrodynamical zoom-in simulation) may increase the runtime of the galactic orbit integration. We highlight that the overhead added by `cogsworth` in connecting these two aspects is negligible.

2.8. *Data storage*

`cogsworth` includes functions for saving and loading an evolved `Population`. Simulations are efficiently stored in a single HDF5 file using `h5py` (Collette 2013), which contains: simulation input settings, a SFH and sampled initial galactic variables, the chosen galactic potential, the initial state, final state and full evolutionary history of the binary stellar population, and the orbits of each system through the galaxy. This provides all the information necessary for reproducing a simulation and analysing its outputs.

We implement lazy-loading for `cogsworth` simulations. This means that not all data is immediately loaded and is instead only loaded as it is needed. For example, one can load a population without the full galactic orbits of the binaries, select a subpopulation of interest (runaway stars, for instance) and plot their orbits, at which point `cogsworth` will load only the data necessary for these systems on the fly.

The file size of a simulation depends on various factors. As one would expect, increasing the number of binaries or single stars simulated, increases the size of the output file. Additionally, since the output log includes a row for each significant evolutionary stage, binaries with more interactions between companions result in larger files. The length of time simulated is also an important factor, since it not only allows more time for binaries to experience complex evolution, but also requires more integration timesteps for galactic orbits. Users can reduce the file size of a simulation by specifying that `cogsworth` use larger integration timesteps, or even that `cogsworth` should only retain the final position of each star rather than its full galactic orbit. As some examples, the population used in Section 3.3 consists of $\sim$500 binaries, which is evolved for $\sim$50 million years, results in a file size of ∼5Mb. The population used in Section 3.1 is larger, containing 30,000 massive binaries and 30,000 massive single stars, and is evolved for 100 Myr, producing a file of ∼300Mb.

2.9. *Citations*

Given the breadth of work upon which `cogsworth` depends, we make it simple for users to ensure they fully credit the work that went into a given simulation. `cogsworth` is capable of creating a custom citation statement for any given `Population` using the `get_citations()` function.

For example, the simplest `cogsworth` simulation may cite only `cogsworth` itself, `COSMIC` and `Gala`. But if you compute the observable features of your population, `cogsworth` will add citations for the dust maps, isochrones, and selection function that you use. This is similarly true for citations regarding star formation histories and hydrodynamical zoom-in simulations.

3. USE CASES

The following subsections each demonstrate a particular use case of `cogsworth`, showcasing its capabilities in binary stellar evolution, galactic dynamics, observables estimation, and integration with hydrodynamical simulations. Each icon links directly to a page in our online documentation that guides users through using `cogsworth` to reproduce a given figure.

Unless otherwise specified in the individual use cases, each `cogsworth` simulation uses the `Wagg2022` SFH, the `MilkyWayPotential2022` Galactic potential and the default binary physics settings from `COSMIC` v3.4.16. Additionally, primary masses are sampled using the Kroupa (2001) IMF, a uniform mass ratio distribution is assumed and the initial orbital period and eccentricity distributions follow Sana et al. (2012). These assumptions are appropriate for massive stars, which can experience core-collapse events and be kicked and disrupted.

3.1. *The importance of binary evolution and the galactic potential*

In this use case, we demonstrate the need for accounting for binary interactions and a galactic potential simultaneously. We use `cogsworth` to simulate a population of massive stars formed in the most recent 100 Myr in the Milky Way assuming a 50% binary fraction. We then repeat the orbital integration for the binaries in this population, but without a galactic potential. In this way, binaries with no core-collapse events remain in their birth locations, whilst those that receive kicks continue at their ejection velocity indefinitely.

We compare the present-day positions of single stars and binary stars in Figure 4, additionally showing the

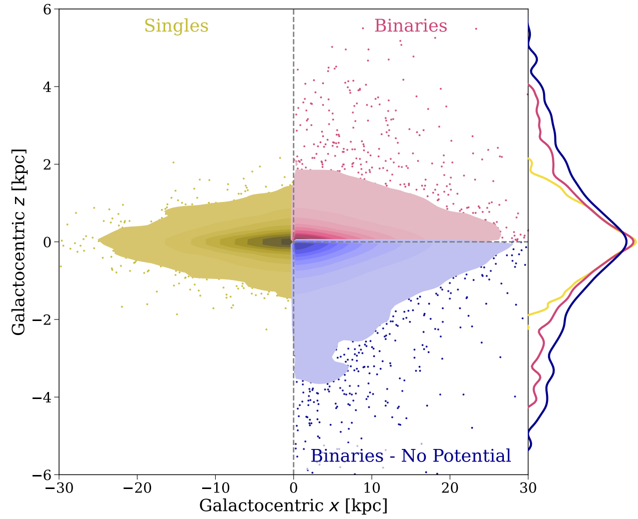

Figure 4. Binary evolution and the presence of a galactic potential both have a significant impact on the spatial distribution of stars. Each panel shows the spatial distributions of massive stars at present day from a `cogsworth` population of single stars (left), binaries (right, top) and binaries evolved without a galactic potential (right, bottom). Density distributions are computed using kernel density estimators (KDE) and shown (in linearly spaced isolevels) up to the 98th percentiles, with the remaining stars plotted as scatter points. Marginal KDEs are shown on the right with a log scale.

distribution of binary stars when neglecting the Galactic potential. First, comparing the single and binary stars in the presence of a potential, we note that the tails of the binary distribution are significantly extended. For single stars, the fraction of the population at $|z| > 1\,\mathrm{kpc}$ is only 1.8%, whereas for binaries it increases by a factor of 2.5x to 4.5%. Previous work has investigated the spatial distribution of compact objects without accounting for binary interactions (Sweeney et al. 2022). However, Figure 4 demonstrates that binary interactions can significantly alter the spatial distributions of massive stars and compact objects.

Moreover, comparing the binary population with and without the Galactic potential, it is clear that neglecting the Galactic potential results in misleading conclusions regarding spatial distributions. In particular, the population without a potential is broader in both width and height, and the fraction at $|z| > 1\,\mathrm{kpc}$ is 10.0%, more than doubling the fraction when accounting for the potential.

3.2. *Comparing the impact of supernova kicks and galactic potentials on spatial distributions*

A key feature of `cogsworth` is its ability to self-consistently account for the effect of binary interactions and galactic potentials. We explore this capability by

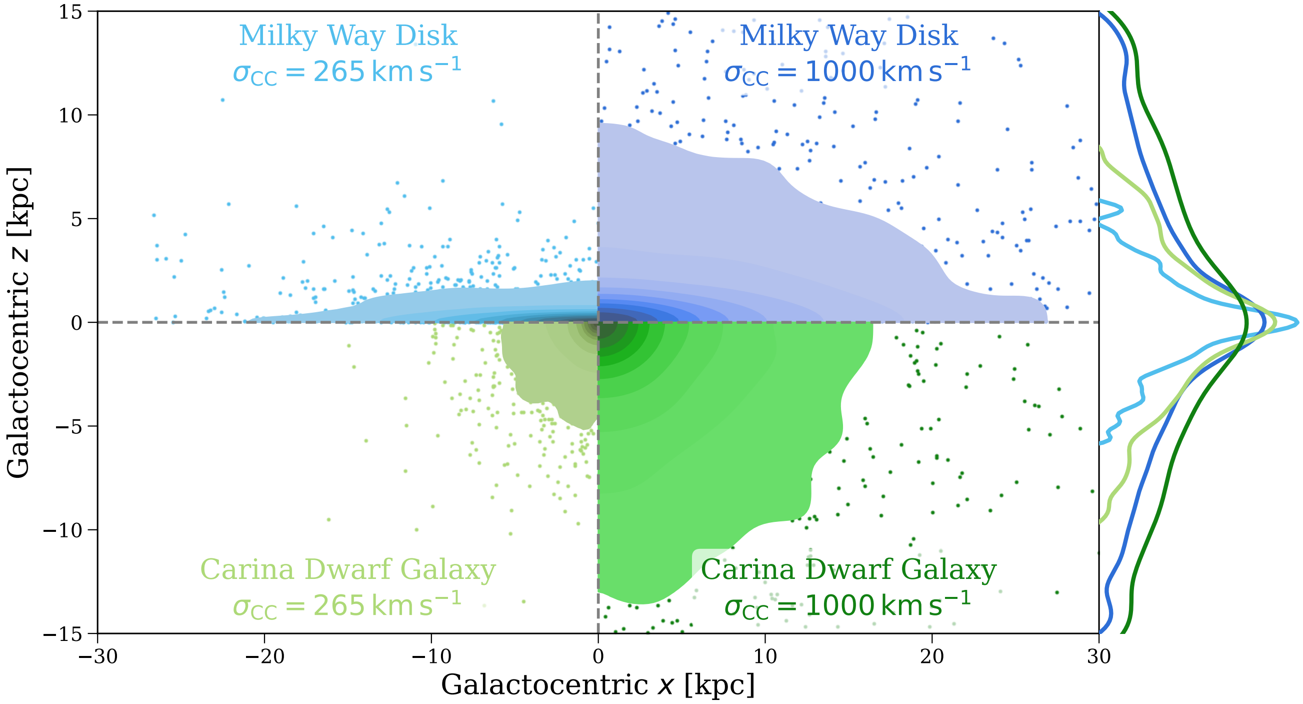

Figure 5. Similar to Figure 4, but comparing the impact of varying binary physics and the galactic potential. Each quadrant corresponds to a different combination of supernova natal kick magnitudes and galactic potential (annotated in each).

determining the relative impact of varying supernova natal kicks and galactic potentials on the spatial distribution of a population of binary stars.

We vary the supernova natal kick distribution from our default choice of a Maxwellian with $\sigma = 265\,\mathrm{km\,s^{-1}}$ for core-collapse supernovae and $\sigma = 20\,\mathrm{km\,s^{-1}}$ for electron-capture and ultra-stripped supernovae (e.g., Hobbs et al. 2005; Igoshev 2020) to an extreme choice of a Maxwellian with $\sigma = 1000\,\mathrm{km\,s^{-1}}$ for all supernovae. For each case we assume fallback-modulated black hole kicks following Fryer et al. (2012), though these are more rare as a result of the initial mass function. For galactic potentials, we compare the Milky Way's disk (using parameters from Sanders & Binney 2015) to a spheroidal dwarf galaxy with parameters matching the Carina dwarf galaxy (Pascale et al. 2019). The latter is assumed to be in isolation, such that there is no tidal stripping from the Milky Way as is observed in the Carina galaxy.

Since here we are only interested in stars that can end their life with a core-collapse event, we sample an initial binary population (assuming a binary fraction of 100%) from the most recent 100 Myr of star formation, retaining only stars more massive than $7\,\mathrm{M_\odot}$. `COSMIC` reports the precise initial conditions of sampled populations, making them easily reproducible with different evolution settings. Using this feature, we evolve identical initial populations with the two different supernova natal kick distributions, then integrate the orbits of each variation through the two different potentials. This results in 4 different populations of present-day positions.

In Figure 5, we compare these populations, showing the positions of all massive stars that experienced a core-collapse event, or were a companion to a star that did. Given the variety of results from the four panels, we highlight that both binary physics and the choice of galactic potential can have a strong effect on the resulting spatial distribution of massive stars. In particular, stronger natal kicks result in large galactic scale heights for massive stars. For the Milky Way disc populations, increasing the natal kick magnitude increases the full-width half maximum (FWHM) of the galactocentric heights from 0.24 kpc to 0.30 kpc. The same is true for the Carina dwarf galaxy population, which for default kicks has larger FWHM than the disc of 0.36 kpc, that increases with stronger kicks to 0.42 kpc. In addition to the width of the overall distribution, the tails of the distribution are particularly strongly affected. The fraction of objects at $|z| > 2\,\mathrm{kpc}$ increases from 2% to 10% for the disc population, and from 6% to 18% for the dwarf galaxy population. Therefore, observing the outliers in a galactic height distribution can be more informative than the FWHM for inferring the strength of natal kicks.

3.3. *Evolution of binary orbits in a star cluster*

`cogsworth` is capable of producing populations of binaries based on hydrodynamical simulations (Section 2.5). We use `cogsworth` to post-process the FIRE m11h simulation (El-Badry et al. 2018; Wetzel et al. 2023), an intermediate-mass halo with a strong disc component, fitting a galactic potential and rewinding all star particles formed in the past 150 Myr to their formation locations. We then sample a binary population from each star particle, matching its formation time, metallicity and mass. For this example, we examine one random star particle and the orbits of its constituents. This star particle was formed ~43 Myr before present-day, with a mass of $\sim 6000\,\mathrm{M_\odot}$ and a metallicity of $Z \approx 0.0137$. The sampled population consists of ~ 7600 systems, split evenly between single stars and binary stars (as one would expect given our assumption of a 50% binary fraction). Note that we neglect the self-gravity of the cluster.

In Figure 6, we plot the orbits of a representative subset of 500 of the binaries sampled from the star particle. In grey, we show the orbits of binaries that experienced no supernovae events, which are by far the majority since the IMF favours low mass stars. The cluster is formed in the lower left (at $\rho = 6.6\,\mathrm{kpc}, z = -0.42\,\mathrm{kpc}$) and evolves to larger ρ. One can note the dissolution of the cluster over time in the grey orbits, which occurs as a result of the initial velocity dispersion (Eq. 5).

The coloured lines show the more eventful orbits of binaries that experienced supernovae. In many cases this leads to the disruption of the binary orbit and so we show the orbit of the subsequent evolution of the ejected companion with a dashed line. With `cogsworth`, one can examine the detailed evolution of each binary to understand its orbit. The examples shown include several scenarios involving bound, disrupted and merged binaries - we discuss each in detail below.

The earliest core-collapse event occurs for the dark blue binary after 4 Myr, which is indicated by closest scatter point to the cluster origin in the lower left. This forms a $12\,\mathrm{M_\odot}$ black hole and, due to a fallback fraction of 92% for the black hole, much of the explosion asymmetry is negated, resulting in a relatively weak natal kick of $11\,\mathrm{km\,s^{-1}}$, which allows the binary the remain bound. The companion to this binary reaches core-collapse 1.5 Myr later, forming a slightly less massive black hole of $6\,\mathrm{M_\odot}$, with a stronger natal kick of $70\,\mathrm{km\,s^{-1}}$. Yet the binary is much tighter at this point (with a separation of $45\,\mathrm{R_\odot}$) and thus has a higher binding energy. This means that it remains bound and is ejected from the cluster as a binary black hole.

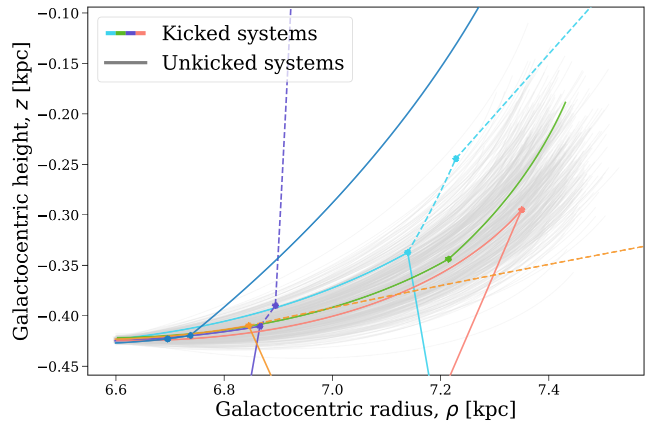

Figure 6. An example star cluster, sampled from a single star particle in the FIRE m11h hydrodynamical zoom-in simulation, evolved within a `cogsworth` simulation. Each line shows the orbit of a star in the cluster. Coloured lines are for stars that had supernova events (highlighted by scatter points), while the rest are shown in grey. For binaries that disrupt, an additional dashed line is shown for the subsequent orbit of the ejected companion.

The first supernova in the orange binary occurs 9 Myr after the cluster birth and forms a neutron star with a natal kick of $447\,\mathrm{km\,s^{-1}}$. This kick disrupts the binary orbit, such that both stars are ejected from the cluster. The secondary is a lower mass star of $3.6\,\mathrm{M_\odot}$ and so experiences no supernova, but it is ejected from the cluster at $160\,\mathrm{km\,s^{-1}}$ and as such is now a runaway star.

The purple binary experiences its first supernova at 9.3 Myr and, similar to the orange binary, this forms a $1.6\,\mathrm{M_\odot}$ NS with a natal kick of $415\,\mathrm{km\,s^{-1}}$ that unbinds the binary. Interestingly, the mass ratio of this system is inverted, such that the companion forms a more massive $5.3\,\mathrm{M_\odot}$ black hole after its core-collapse 2 Myr later. This inversion occurred as a result of significant, near conservative mass transfer from the primary star during its Hertzsprung gap phase 1.2 Myr before it reached core-collapse. Both supernovae for the light blue binary form neutron stars (of 1.3 and $2.2\,\mathrm{M_\odot}$ respectively) with strong kicks (of 406 and $728\,\mathrm{km\,s^{-1}}$ respectively), which disrupt the binary and eject both neutron stars rapidly from the cluster.

The primary star in the green binary reaches core-collapse 26 Myr after the cluster birth, forming a neutron star of $1.27\,\mathrm{M_\odot}$. The star explodes as an electron-capture supernova and thus its kick is assumed to be weaker, in this case the drawn natal kick is only $35\,\mathrm{km\,s^{-1}}$ and the binary remains bound. In addition, as a result of the angle of the kick relative to the binary's orbit, this only results in a $3.4\,\mathrm{km\,s^{-1}}$ change to the systemic velocity of the binary. As a result, the

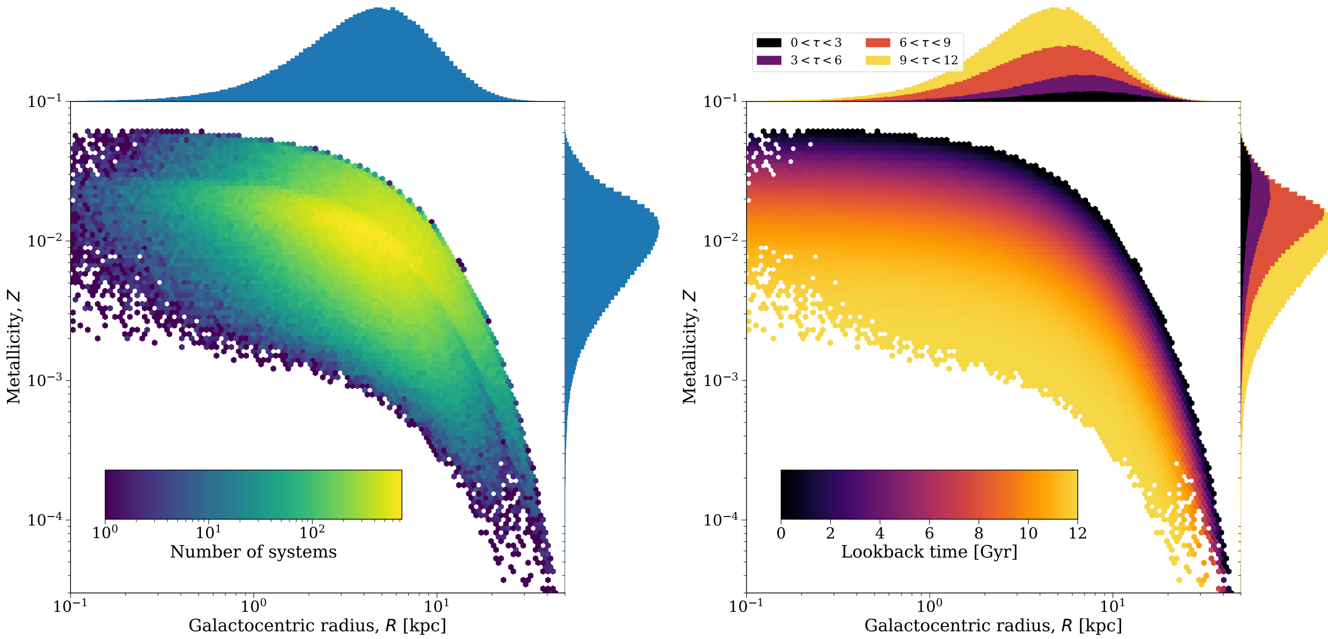

Figure 7. The relationship between galactocentric radius, metallicity and lookback time in the Wagg2022 SFH. **Left:** The main panel shows a 2D histogram of the number of stars formed at a given radius and metallicity, with marginal distributions for each parameter shown as histograms on the side panels. **Right:** The main panel shows a 2D histogram in which each bin is coloured by the average lookback time of associated stars. Marginal distributions are now shown as stacked histograms, grouped by lookback time.

binary remains bound to the cluster for its subsequent evolution.

Finally, after $\sim$33 Myr, the primary star in the red binary finishes its main sequence, As it expands during its Hertzsprung gap phase, it overflows its Roche lobe, causing unstable mass transfer which leads to a merger. The merged star then reaches supernova 4 Myr later, forming a neutron star which is ejected by its strong natal kick of 819 $\mathrm{km\,s^{-1}}$, in almost the opposite direction to the cluster's centre of mass motion.

3.4. *Examining metallicity-radius-time relations in star formation histories*

`cogsworth` can be used to sample detailed star formation histories independently of evolving binary stars or performing galactic orbit integration (see Section 2.1). In this use case, we explore the Wagg2022 SFH model in more detail. We sample 500,000 points (which could be designated as a single or binary star) from the SFH, which each have an associated position, lookback time and metallicity.

In the left panel of Figure 7, we plot the distribution of Galactocentric radii and metallicities for each sampled point. As a general trend, one can note that stars closer to the centre of the Galaxy are more metal-rich than those on the outer edges. This is due to the inside-out growth of the Galaxy (e.g., Fall & Efstathiou 1980; Frankel et al. 2019), which is accounted for in this SFH following the model of Frankel et al. (2018). Additionally, the discontinuity in the distribution (occurring at inner radii at $Z \approx 0.03$) is a result of the multi-component nature of the model. The upper right portion above the discontinuity comes is primarily from the low-[α/Fe] disc component, which forms stars from 8 Gyr ago until present-day, while the lower portion is primarily from the high-[α/Fe] disc, which form stars from 12 Gyr ago until 8 Gyr ago. The bulge component contributes to both portions, though only at small radii.

For a given radius, there is a wide variation in the metallicity of sampled stars. This is because of the birth time of each star, which we demonstrate in the right panel of Figure 7. The 2D histogram now shows the average lookback time, τ, of the stars in each bin (where $\tau = 0$ corresponds to present-day). The clear gradient shows that over time the Galaxy as a whole becomes more metal-rich as it is enriched by stellar evolution. This is additionally visible in the marginal distribution of metallicities, where no high metallicity stars are formed at early birth times. The marginal distribution of radii again demonstrates the inside-out growth, as older stars were formed closer to the Galactic centre.

3.5. *Simulating a Gaia colour-magnitude diagram*

In this use case, we highlight `cogsworth`'s ability to transform an intrinsic population into a simulated observable population (see Section 2.4). We sample 2500

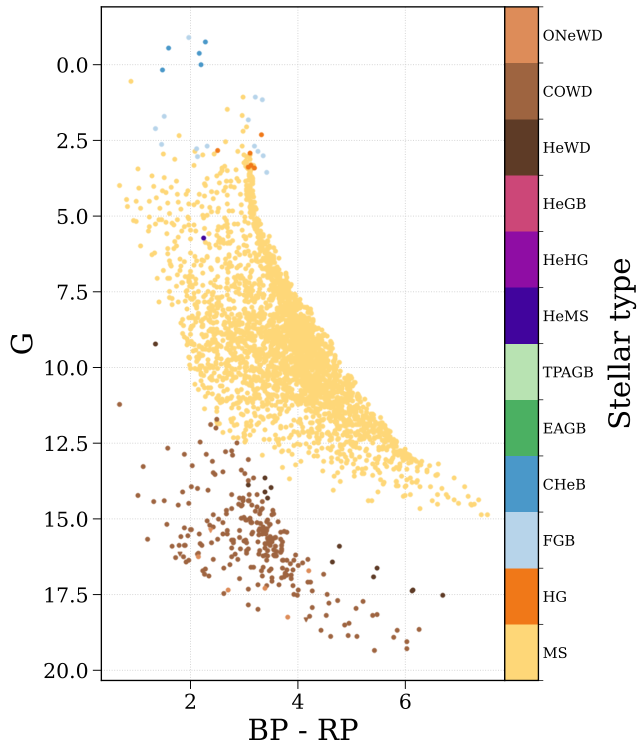

Figure 8. An example simulated *Gaia* colour-magnitude diagram. Each scatter point corresponds to a binary or disrupted star, coloured by its stellar type (using the stellar type of the brighter component for binaries). The stellar types abbreviated in the colourbar follow those defined in BSE (see Section 4 of Hurley et al. 2000).

binary systems over the full SFH of the Milky Way and evolve them until present day.

We compute observables for this population using the `Population.get_observables()` function in `cogsworth`. We first calculate the absolute magnitude of each star and determine which star is brighter in a binary. `cogsworth` then converts these magnitudes to the *Gaia* filters G, BP and RP using the MIST isochrones to apply bolometric correction with the `isochrones` package (Morton 2015; Dotter 2016; Choi et al. 2016; Paxton et al. 2011, 2013, 2015). Finally, `cogsworth` uses the `dustmaps` package to account for dust extinction through the application of the Bayestar 2019 dust maps (Green 2018; Green et al. 2019).

We plot the resulting colour-magnitude diagram (CMD) in Figure 8, colouring each bound system by the stellar type of the brighter component in the G band. These systems cover a range of metallicities, distances and ages and hence have a wide spread in the CMD. In addition to these systems, 59 isolated (either from mergers or binary disruptions) neutron stars and black holes are present in the evolved population. This same simulation could be easily repeated with a starburst localised in one specific place to model a specific cluster CMD instead.

3.6. *Comparing present-day sky locations with and without supernova kicks*

`cogsworth` can report the present-day sky position of each source, in addition to its evolutionary history. In this use case we demonstrate how one can track the relative positions of binary companions after they disrupt, as well as consider where they would be found if no supernova kick had occurred.

We sample and evolve 100 random binaries in the Milky Way with our default assumptions, except we set the minimum mass of the IMF to $3\,\mathrm{M}_\odot$. This preferentially samples more massive stars, which are more likely to reach core-collapse and cause a binary disruption. From this population, we subselect five binaries that are disrupted. These binaries experienced at least one core-collapse event which disrupted the orbit and led to the separation of the two unbound companions. We use `cogsworth` to compute the present-day sky location of both companions for each binary, before re-integrating the binary's orbit without accounting for the impact of supernova kicks. `cogsworth` returns the final coordinates of stars as an `Astropy SkyCoord` (Astropy Collaboration et al. 2013, 2018, 2022), which allows for simple transformation between coordinate frames.

In Figure 9, we compare the present-day sky locations (in galactic coordinates) of each compact object from the disrupted binaries to the location of the binary had it experienced no supernova kicks. As expected, when neglecting supernova kicks, binaries are generally concentrated close to the galactic midplane and centre. However, when accounting for the effect of supernova kicks on the internal and galactic orbits, the present-day sky locations are often significantly different. In many cases, companions are located not only far from one another, but also far from the position of the binary had no kicks occurred. Several of the compact objects from the disrupted binaries are found well beyond the typical sky locations of galactic sources (shown as a histogram in the background), though all remain bound to the galaxy. In particular, the secondary from the blue binary is first ejected from the binary as a runaway star (travelling at $\sim$35 km s^{-1}) after the primary reaches core-collapse. After spending 10 Myr as an O-type runaway star, the secondary reaches core-collapse and is kicked onto an orbit that takes it even further from the typical galactic population. The green binary starts on a relatively wide initial orbital period of $\sim$30,000 days. The binary widens by nearly 10% due to the stellar winds of the

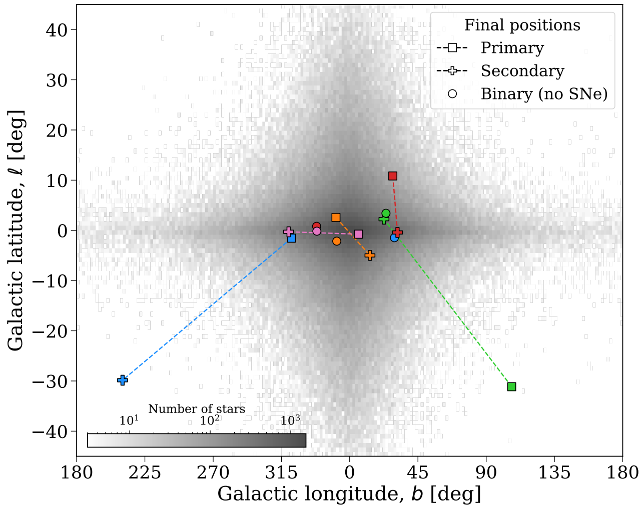

Figure 9. Present-day sky positions of disrupted binaries, with and without supernova kicks. A square (plus) marker shows the location of the primary (secondary) star from a disrupted binary, connected to its companion by a dashed line. Circular markers indicate the location of the binary had no supernova kicks occurred. The background density shows a 2D histogram of 500,000 stars sampled from the same SFH (Wagg2022) for comparison. Note that disk appears extended in latitude because we limit the axes to the relevant region.

primary before this star's core-collapse 32 Myr after the birth of the binary. The primary forms a neutron star that receives a natal kick of $325\,\mathrm{km\,s^{-1}}$, such that it takes large excursions from the galactic plane during its orbit. However, due to the large orbital period at the supernova, the secondary star is only ejected at $16\,\mathrm{km\,s^{-1}}$. This star eventually forms a white dwarf and receives no further kick, hence its present-day location is relatively similar to that of the binary had no kick occurred.

4. LIMITATIONS

Although `cogsworth` has a wide range of features and capabilities, there are still some limitations to the code that users should be aware of.

Dynamical interactions—We do not implement dynamical interactions between systems or account for any N-body dynamics. As mentioned in Section 2.7, each binary in `cogsworth` is evolved independent of all others. This means that dynamical formation channels for different populations (such as interactions in dense clusters forming gravitational-wave progenitors) cannot be simulated in `cogsworth`. However, we do intend to implement a prescription for emulating dynamical cluster ejections (see Section 5.2), such that investigations into runaway stars could consider both channels.

Impact of galactic potential on internal orbits—`cogsworth` accounts for the impact of stellar evolution on the galactic orbits of binaries (i.e. as a result of supernovae). Yet it does not currently account for the inverse case, in which a galactic potential alters the orbit of the binary. For wide binaries the potential can significantly affect the orbit, causing large-amplitude oscillations and potentially drive systems to disrupt or merge (e.g., Weinberg et al. 1987; Heisler & Tremaine 1986; Jiang & Tremaine 2010; Modak & Hamilton 2023; Stegmann et al. 2024). However, for closer binaries the effect is negligible and as such as we do not currently account for it in `cogsworth`.

Population synthesis model uncertainties—`cogsworth` uses `COSMIC` for binary population synthesis, which is a code based on `BSE`. The `BSE` code relies on approximate parametric prescriptions for a limited set of evolutionary tracks of single stars (Pols et al. 1998; Hurley et al. 2000, 2002). Although many of the original prescriptions used in the `BSE` code have been improved in `COSMIC` (Breivik et al. 2020a), the core of the code still relies on the same methodology. In particular, the treatment of mass loss and the stability of mass transfer, as well as the reliability of the most massive progenitor models, is uncertain. However, some of these uncertainties can be alleviated by incorporating information on the internal structure of stars (e.g., Kruckow et al. 2018; Fragos et al. 2023). `COSMIC` is in the process of being integrated with `METISSE`, MEthod of Interpolation for Single-Star Evolution. `METISSE` is an alternative to fitting formulae that allows for the interpolation between pre-computed detailed one-dimensional stellar evolution tracks, while maintaining the same code interfaces as the previously implemented prescriptions of SSE (Agrawal et al. 2020, 2023). By working with updated libraries of pre-computed single star tracks from `MESA` (Paxton et al. 2011, 2013, 2015, 2018, 2019; Jermyn et al. 2023), `METISSE` enables a wide range of investigations of the impact of uncertainties in single-star evolution like convection, rotation, and nuclear reaction rates and how these uncertainties interface with uncertainties in binary interaction physics. Once `METISSE` is fully integrated into `COSMIC`, `cogsworth` will be able to immediately leverage these new improvements.

5. FUTURE DEVELOPMENTS

We intend to complete further development on `cogsworth` beyond this initial release. In the following subsections we highlight some areas in which we aim to focus.

5.1. *Time-evolving galactic potentials*

Traditional models using static galactic potentials are not capable of describing the dynamically complex evolutionary history of galaxies, and can lead to misleading results (e.g., Arora et al. 2022). Although this is less relevant for shorter lived populations (such as massive runaway stars), it could have important implications for longer lasting tracers of binary endpoints (e.g., gravitational wave mergers). Thus we also intend to leverage our integration with hydrodynamical simulations to implement a `cogsworth` option for a time-evolving gravitational potential that accounts for the mass growth of a galaxy over time.

5.2. *Dynamical cluster ejections*

`cogsworth` does not currently account for the impact of dynamical interactions between binaries in dense environments, as noted in Section 4. The interactions can change the initial architecture of binaries (e.g., Fujii & Portegies Zwart 2011) and create alternate formation channels for binary products. For instance, runaway stars are thought to be formed in two main channels: the disruption of binaries as a result of supernovae (Blaauw 1961; Eldridge et al. 2011; Renzo et al. 2019) and dynamical ejections from stellar clusters (Poveda et al. 1967). Fully modelling the latter channel would require more complex N-body dynamics that are currently beyond the scope of the code. Instead, we intend to create an approximation in which we will give a fraction of massive stars kicks shortly after their formation. The mass-dependent rate, kick velocity and timing will follow distributions modelled in N-body simulations (e.g., Oh & Kroupa 2016; Schoettler et al. 2022).

5.3. *Other observables*

For high-energy, degenerate, and/or accreting sources formed through binary channels (e.g., X-ray binaries, cataclysmic variables, short gamma-ray bursts, type Ia supernovae), the mapping between binary physical parameters and flux is naturally more complex (and sometimes uncertain) than it is for most stars. This means that predictions for other observables (beyond those current implemented) are more complicated, though not out of reach in many cases. For example, prescriptions for the X-ray luminosity of a given binary exist (Misra et al. 2023), and we intend to add this feature to `COSMIC` (and thus also to `cogsworth`) to make predictions for the X-ray binary populations that have been widely observed with *Chandra* in nearby galaxies. In the future, we will implement mappings for other missions and observables based on their own selection functions.

6. CONCLUSIONS & SUMMARY

In this paper we have presented `cogsworth`, a new open-source code for performing self-consistent population synthesis and galactic dynamics simulations. `cogsworth` provides the theoretical infrastructure necessary to make predictions about the positions and velocities of stars and compact objects. We have demonstrated several use cases of the code, showcasing its capabilities to investigate the impact of binary interactions and galactic potentials on the evolution of stars and compact objects - both for intrinsic and observable populations. `cogsworth` could be applied to a plethora of investigations on a wide-range of populations, including runaway stars, supernova remnants, X-ray binaries, short gamma-ray bursts and double compact objects.

Given its accessibility and flexibility, we hope that `cogsworth` will be a useful tool for the community, enabling and accelerating future studies into binary stars and compact objects.

ACKNOWLEDGEMENTS

We gratefully acknowledge many fruitful discussions with Julianne Dalcanton and Eric Bellm that resulted in several helpful suggestions. TW acknowledges valuable conversations with Matt Orr and Chris Hayward regarding the `FIRE` simulations, and with Alyson Brooks and Akaxia Cruz regarding the `ChaNGa` simulations. TW thanks the Simons Foundation, Flatiron Institute and Center for Computational Astrophysics for running the pre-doctoral program during which much of this work was completed. The Flatiron Institute is supported by the Simons Foundation. TW and KB acknowledge support from NASA ATP grant 80NSSC24K0768.

Software: `astropy` (Astropy Collaboration et al. 2013, 2018, 2022), `Jupyter` (Perez & Granger 2007; Kluyver et al. 2016), `matplotlib` (Hunter 2007), `numpy` (Harris et al. 2020), `pandas` (Wes McKinney 2010; pandas development team 2024), `python` (Van Rossum & Drake 2009), `scipy` (Virtanen et al. 2020; Gommers et al. 2024), `Agama` (Vasiliev 2019), `astroquery` (Ginsburg et al. 2019; Ginsburg et al. 2024), `ChaNGa` (Jetley et al. 2008, 2010; Menon et al. 2015), `COSMIC` (Breivik et al. 2020b; Coughlin et al. 2024), `Cython` (Behnel et al. 2011), `dustmaps` (Green 2018; Green et al. 2024), `gaiaunlimited` (Cantat-Gaudin et al. 2023), `gala` (Price-Whelan 2017; Price-Whelan et al. 2024), `h5py` (Collette 2013; Collette et al. 2023), `isochrones` (Morton 2015), `legwork` (Wagg et al. 2022a,b; Wagg & Breivik 2024), `Numba` (Lam et al. 2015; Lam et al. 2024), `pynbody` (Pontzen et al. 2013; Pontzen et al. 2023), `PyTables` (Team 2002–), `schwimmbad` (Price-Whelan & Foreman-Mackey 2017), `seaborn` (Waskom 2021), and `tqdm` (da Costa-Luis et al. 2024). Some of the results in this paper have been derived using `healpy` and the

HEALPix package[3] (Zonca et al. 2019; Górski et al. 2005; Zonca et al. 2024). This research has made use of NASA's Astrophysics Data System. Software citation information aggregated using `The Software Citation Station` (Wagg & Broekgaarden 2024; Wagg et al. 2024).

REFERENCES

Agrawal, P., Hurley, J., Stevenson, S., et al. 2023, MNRAS, 525, 933, doi: 10.1093/mnras/stad2334

Agrawal, P., Hurley, J., Stevenson, S., Szécsi, D., & Flynn, C. 2020, MNRAS, 497, 4549, doi: 10.1093/mnras/staa2264

Amaro-Seoane, P., Audley, H., Babak, S., et al. 2017, arXiv e-prints, arXiv:1702.00786, doi: 10.48550/arXiv.1702.00786

Applebaum, E., Brooks, A. M., Christensen, C. R., et al. 2021, ApJ, 906, 96, doi: 10.3847/1538-4357/abcafa

Arora, A., Sanderson, R. E., Panithanpaisal, N., et al. 2022, ApJ, 939, 2, doi: 10.3847/1538-4357/ac93fb

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33, doi: 10.1051/0004-6361/201322068

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123, doi: 10.3847/1538-3881/aabc4f

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, ApJ, 935, 167, doi: 10.3847/1538-4357/ac7c74

Atri, P., Miller-Jones, J. C. A., Bahramian, A., et al. 2019, MNRAS, 489, 3116, doi: 10.1093/mnras/stz2335

Behnel, S., Bradshaw, R., Citro, C., et al. 2011, Computing in Science Engineering, 13, 31, doi: 10.1109/MCSE.2010.118

Bertoldi, F., & McKee, C. F. 1992, ApJ, 395, 140, doi: 10.1086/171638

Blaauw, A. 1961, BAN, 15, 265

Breivik, K., Coughlin, S., Zevin, M., et al. 2020a, ApJ, 898, 71, doi: 10.3847/1538-4357/ab9d85

—. 2020b, ApJ, 898, 71, doi: 10.3847/1538-4357/ab9d85

Cantat-Gaudin, T., Fouesneau, M., Rix, H.-W., et al. 2023, A&A, 669, A55, doi: 10.1051/0004-6361/202244784

Chawla, C., Chatterjee, S., Breivik, K., et al. 2022, ApJ, 931, 107, doi: 10.3847/1538-4357/ac60a5

Choi, J., Dotter, A., Conroy, C., et al. 2016, ApJ, 823, 102, doi: 10.3847/0004-637X/823/2/102

Christensen, C., Munshi, F., & Brooks, A. 2023, in American Astronomical Society Meeting Abstracts, Vol. 55, American Astronomical Society Meeting Abstracts, 407.04

Collette, A. 2013, Python and HDF5 (O'Reilly)

Collette, A., Kluyver, T., Caswell, T. A., et al. 2023, h5py/h5py: 3.8.0, 3.8.0, Zenodo, doi: 10.5281/zenodo.7560547

Coughlin, S., Breivik, K., Zevin, M., et al. 2024, COSMIC-PopSynth/COSMIC: v3.4.16, v3.4.16, Zenodo, doi: 10.5281/zenodo.13351205

da Costa-Luis, C., Larroque, S. K., Altendorf, K., et al. 2024, tqdm: A fast, Extensible Progress Bar for Python and CLI, v4.66.2, Zenodo, doi: 10.5281/zenodo.3551211

Darragh-Ford, E., Hunt, J. A. S., Price-Whelan, A. M., & Johnston, K. V. 2023, ApJ, 955, 74, doi: 10.3847/1538-4357/acf1fc

de Mink, S. E., Sana, H., Langer, N., Izzard, R. G., & Schneider, F. R. N. 2014, ApJ, 782, 7, doi: 10.1088/0004-637X/782/1/7

Dekel, A., & Silk, J. 1986, ApJ, 303, 39, doi: 10.1086/164050

Disberg, P., Gaspari, N., & Levan, A. J. 2024, A&A, 687, A272, doi: 10.1051/0004-6361/202449996

Dotter, A. 2016, ApJS, 222, 8, doi: 10.3847/0067-0049/222/1/8

Duchêne, G., & Kraus, A. 2013, ARA&A, 51, 269, doi: 10.1146/annurev-astro-081710-102602

Eilers, A.-C., Hogg, D. W., Rix, H.-W., & Ness, M. K. 2019, ApJ, 871, 120, doi: 10.3847/1538-4357/aaf648

El-Badry, K., Quataert, E., Wetzel, A., et al. 2018, MNRAS, 473, 1930, doi: 10.1093/mnras/stx2482

Eldridge, J. J., Langer, N., & Tout, C. A. 2011, MNRAS, 414, 3501, doi: 10.1111/j.1365-2966.2011.18650.x

Evans, F. A., Renzo, M., & Rossi, E. M. 2020, MNRAS, 497, 5344, doi: 10.1093/mnras/staa2334

Fall, S. M., & Efstathiou, G. 1980, MNRAS, 193, 189, doi: 10.1093/mnras/193.2.189

Fragos, T., Andrews, J. J., Bavera, S. S., et al. 2023, ApJS, 264, 45, doi: 10.3847/1538-4365/ac90c1

Frankel, N., Rix, H.-W., Ting, Y.-S., Ness, M., & Hogg, D. W. 2018, ApJ, 865, 96, doi: 10.3847/1538-4357/aadba5

Frankel, N., Sanders, J., Rix, H.-W., Ting, Y.-S., & Ness, M. 2019, ApJ, 884, 99, doi: 10.3847/1538-4357/ab4254

Fryer, C. L., Belczynski, K., Wiktorowicz, G., et al. 2012, ApJ, 749, 91, doi: 10.1088/0004-637X/749/1/91

Fujii, M. S., & Portegies Zwart, S. 2011, Science, 334, 1380, doi: 10.1126/science.1211927

[3]http://healpix.sourceforge.net

Ginsburg, A., Sipőcz, B. M., Brasseur, C. E., et al. 2019, AJ, 157, 98, doi: 10.3847/1538-3881/aafc33

Ginsburg, A., Sipőcz, B., Brasseur, C. E., et al. 2024, astropy/astroquery: v0.4.7, v0.4.7, Zenodo, doi: 10.5281/zenodo.10799414

Gommers, R., Virtanen, P., Haberland, M., et al. 2024, scipy/scipy: SciPy 1.14.0rc2, v1.14.0rc2, Zenodo, doi: 10.5281/zenodo.11702230

Górski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ, 622, 759, doi: 10.1086/427976

Green, G. 2018, The Journal of Open Source Software, 3, 695, doi: 10.21105/joss.00695

Green, G., Edenhofer, G., Krughoff, S., et al. 2024, gregreen/dustmaps: v1.0.13, v1.0.13, Zenodo, doi: 10.5281/zenodo.10517733

Green, G. M., Schlafly, E., Zucker, C., Speagle, J. S., & Finkbeiner, D. 2019, ApJ, 887, 93, doi: 10.3847/1538-4357/ab5362

Grudić, M. Y., Hafen, Z., Rodriguez, C. L., et al. 2023, MNRAS, 519, 1366, doi: 10.1093/mnras/stac3573

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357, doi: 10.1038/s41586-020-2649-2

Heisler, J., & Tremaine, S. 1986, Icarus, 65, 13, doi: 10.1016/0019-1035(86)90060-6

Hernquist, L., & Ostriker, J. P. 1992, ApJ, 386, 375, doi: 10.1086/171025

Hobbs, G., Lorimer, D. R., Lyne, A. G., & Kramer, M. 2005, MNRAS, 360, 974, doi: 10.1111/j.1365-2966.2005.09087.x

Hopkins, P. F., Quataert, E., & Murray, N. 2012, MNRAS, 421, 3522, doi: 10.1111/j.1365-2966.2012.20593.x

Hopkins, P. F., Wetzel, A., Kereš, D., et al. 2018, MNRAS, 480, 800, doi: 10.1093/mnras/sty1690

Hunter, J. D. 2007, Computing in Science & Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

Hurley, J. R., Pols, O. R., & Tout, C. A. 2000, MNRAS, 315, 543, doi: 10.1046/j.1365-8711.2000.03426.x

Hurley, J. R., Tout, C. A., & Pols, O. R. 2002, MNRAS, 329, 897, doi: 10.1046/j.1365-8711.2002.05038.x

Igoshev, A. P. 2020, MNRAS, 494, 3663, doi: 10.1093/mnras/staa958

Ivanova, N., Justham, S., & Ricker, P. 2020, Common Envelope Evolution, doi: 10.1088/2514-3433/abb6f0

Ivanova, N., Justham, S., Chen, X., et al. 2013, A&A Rv, 21, 59, doi: 10.1007/s00159-013-0059-2

Janka, H.-T. 2012, Annual Review of Nuclear and Particle Science, 62, 407, doi: 10.1146/annurev-nucl-102711-094901

—. 2017, ApJ, 837, 84, doi: 10.3847/1538-4357/aa618e

Jermyn, A. S., Bauer, E. B., Schwab, J., et al. 2023, ApJS, 265, 15, doi: 10.3847/1538-4365/acae8d

Jetley, P., Gioachin, F., Mendes, C., Kale, L. V., & Quinn, T. 2008, in 2008 IEEE International Symposium on Parallel and Distributed Processing, 1–12, doi: 10.1109/IPDPS.2008.4536319

Jetley, P., Wesolowski, L., Gioachin, F., Kalé, L. V., & Quinn, T. R. 2010, in SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 1–11, doi: 10.1109/SC.2010.49

Jiang, Y.-F., & Tremaine, S. 2010, MNRAS, 401, 977, doi: 10.1111/j.1365-2966.2009.15744.x

Katsuda, S., Morii, M., Janka, H.-T., et al. 2018, ApJ, 856, 18, doi: 10.3847/1538-4357/aab092

Katz, J. I. 1975, Nature, 253, 698, doi: 10.1038/253698a0

Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in ELPUB, 87–90

Kroupa, P. 2001, MNRAS, 322, 231, doi: 10.1046/j.1365-8711.2001.04022.x

Kruckow, M. U., Tauris, T. M., Langer, N., Kramer, M., & Izzard, R. G. 2018, MNRAS, 481, 1908, doi: 10.1093/mnras/sty2190

Lam, S. K., Pitrou, A., & Seibert, S. 2015, in Proc. Second Workshop on the LLVM Compiler Infrastructure in HPC, 1–6, doi: 10.1145/2833157.2833162

Lam, S. K., stuartarchibald, Pitrou, A., et al. 2024, numba/numba: 0.60.0, 0.60.0, Zenodo, doi: 10.5281/zenodo.11642058

Lamberts, A., Garrison-Kimmel, S., Hopkins, P. F., et al. 2018, MNRAS, 480, 2704, doi: 10.1093/mnras/sty2035

Lowing, B., Jenkins, A., Eke, V., & Frenk, C. 2011, MNRAS, 416, 2697, doi: 10.1111/j.1365-2966.2011.19222.x

Mandel, I. 2016, MNRAS, 456, 578, doi: 10.1093/mnras/stv2733

Marchant, P., & Bodensteiner, J. 2023, arXiv e-prints, arXiv:2311.01865, doi: 10.48550/arXiv.2311.01865

Menon, H., Wesolowski, L., Zheng, G., et al. 2015, Computational Astrophysics and Cosmology, 2, 1, doi: 10.1186/s40668-015-0007-9

Misra, D., Kovlakas, K., Fragos, T., et al. 2023, A&A, 672, A99, doi: 10.1051/0004-6361/202244929

Modak, S., & Hamilton, C. 2023, MNRAS, 524, 3102, doi: 10.1093/mnras/stad2073

Moe, M., & Di Stefano, R. 2017, ApJS, 230, 15, doi: 10.3847/1538-4365/aa6fb6

Morton, T. D. 2015, isochrones: Stellar model grid package, Astrophysics Source Code Library, record ascl:1503.010

Naab, T., & Ostriker, J. P. 2017, ARA&A, 55, 59, doi: 10.1146/annurev-astro-081913-040019

Neuhäuser, R., Gießler, F., & Hambaryan, V. V. 2020, MNRAS, 498, 899, doi: 10.1093/mnras/stz2629

Nomoto, K., Kobayashi, C., & Tominaga, N. 2013, ARA&A, 51, 457, doi: 10.1146/annurev-astro-082812-140956

Offner, S. S. R., Moe, M., Kratter, K. M., et al. 2023, in Astronomical Society of the Pacific Conference Series, Vol. 534, Protostars and Planets VII, ed. S. Inutsuka, Y. Aikawa, T. Muto, K. Tomida, & M. Tamura, 275, doi: 10.48550/arXiv.2203.10066

Oh, S., & Kroupa, P. 2016, A&A, 590, A107, doi: 10.1051/0004-6361/201628233

pandas development team, T. 2024, pandas-dev/pandas: Pandas, v2.2.2, Zenodo, doi: 10.5281/zenodo.10957263

Pascale, R., Binney, J., Nipoti, C., & Posti, L. 2019, MNRAS, 488, 2423, doi: 10.1093/mnras/stz1617

Paxton, B., Bildsten, L., Dotter, A., et al. 2011, ApJS, 192, 3, doi: 10.1088/0067-0049/192/1/3

Paxton, B., Cantiello, M., Arras, P., et al. 2013, ApJS, 208, 4, doi: 10.1088/0067-0049/208/1/4

Paxton, B., Marchant, P., Schwab, J., et al. 2015, ApJS, 220, 15, doi: 10.1088/0067-0049/220/1/15

Paxton, B., Schwab, J., Bauer, E. B., et al. 2018, ApJS, 234, 34, doi: 10.3847/1538-4365/aaa5a8

Paxton, B., Smolec, R., Schwab, J., et al. 2019, ApJS, 243, 10, doi: 10.3847/1538-4365/ab2241

Perez, F., & Granger, B. E. 2007, Computing in Science and Engineering, 9, 21, doi: 10.1109/MCSE.2007.53

Podsiadlowski, P., Joss, P. C., & Hsu, J. J. L. 1992, ApJ, 391, 246, doi: 10.1086/171341

Pols, O. R., Schröder, K.-P., Hurley, J. R., Tout, C., & Eggleton, P. 1998, MNRAS, 298, 525, doi: 10.1046/j.1365-8711.1998.01658.x

Pontzen, A., Roškar, R., Stinson, G. S., et al. 2013, pynbody: Astrophysics Simulation Analysis for Python

Pontzen, A., Roškar, R., Cadiou, C., et al. 2023, pynbody/pynbody: Version 1.5.2, v1.5.2, Zenodo, doi: 10.5281/zenodo.10276404

Poveda, A., Ruiz, J., & Allen, C. 1967, Boletin de los Observatorios Tonantzintla y Tacubaya, 4, 86

Price-Whelan, A., Souchereau, H., Wagg, T., et al. 2024, adrn/gala: v1.9.1, v1.9.1, Zenodo, doi: 10.5281/zenodo.13377376

Price-Whelan, A. M. 2017, The Journal of Open Source Software, 2, 388, doi: 10.21105/joss.00388

Price-Whelan, A. M. 2017, The Journal of Open Source Software, 2, doi: 10.21105/joss.00388

Price-Whelan, A. M., & Foreman-Mackey, D. 2017, The Journal of Open Source Software, 2, doi: 10.21105/joss.00357

Renzo, M., Zapartas, E., de Mink, S. E., et al. 2019, A&A, 624, A66, doi: 10.1051/0004-6361/201833297

Repetto, S., Davies, M. B., & Sigurdsson, S. 2012, MNRAS, 425, 2799, doi: 10.1111/j.1365-2966.2012.21549.x

Repetto, S., Igoshev, A. P., & Nelemans, G. 2017, MNRAS, 467, 298, doi: 10.1093/mnras/stx027

Repetto, S., & Nelemans, G. 2015, MNRAS, 453, 3341, doi: 10.1093/mnras/stv1753

Rodriguez, C. L., Hafen, Z., Grudić, M. Y., et al. 2023, MNRAS, 521, 124, doi: 10.1093/mnras/stad578

Röpke, F. K., & De Marco, O. 2023, Living Reviews in Computational Astrophysics, 9, 2, doi: 10.1007/s41115-023-00017-x

Sana, H., de Mink, S. E., de Koter, A., et al. 2012, Science, 337, 444, doi: 10.1126/science.1223344

Sanders, J. L., & Binney, J. 2015, MNRAS, 449, 3479, doi: 10.1093/mnras/stv578

Sanderson, R. E., Wetzel, A., Loebman, S., et al. 2020, ApJS, 246, 6, doi: 10.3847/1538-4365/ab5b9d

Sarbadhicary, S. K., Wagner, J., Koch, E. W., et al. 2023, arXiv e-prints, arXiv:2310.17694, doi: 10.48550/arXiv.2310.17694

Schoettler, C., Parker, R. J., & de Bruijne, J. 2022, MNRAS, 510, 3178, doi: 10.1093/mnras/stab3529

Somerville, R. S., & Davé, R. 2015, ARA&A, 53, 51, doi: 10.1146/annurev-astro-082812-140951

Stegmann, J., Vigna-Gómez, A., Rantala, A., et al. 2024, arXiv e-prints, arXiv:2405.02912, doi: 10.48550/arXiv.2405.02912

Sweeney, D., Tuthill, P., Sharma, S., & Hirai, R. 2022, MNRAS, 516, 4971, doi: 10.1093/mnras/stac2092

Team, P. D. 2002–, PyTables: Hierarchical Datasets in Python. http://www.pytables.org/

Thiele, S., Breivik, K., Sanderson, R. E., & Luger, R. 2023, ApJ, 945, 162, doi: 10.3847/1538-4357/aca7be

Tout, C. A., Aarseth, S. J., Pols, O. R., & Eggleton, P. P. 1997, MNRAS, 291, 732, doi: 10.1093/mnras/291.4.732

van den Heuvel, E. P. J. 1976, in IAU Symposium, Vol. 73, Structure and Evolution of Close Binary Systems, ed. P. Eggleton, S. Mitton, & J. Whelan, 35

Van Rossum, G., & Drake, F. L. 2009, Python 3 Reference Manual (Scotts Valley, CA: CreateSpace)

Vasiliev, E. 2019, MNRAS, 482, 1525, doi: 10.1093/mnras/sty2672

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/s41592-019-0686-2

Wagg, T., & Breivik, K. 2024, TeamLEGWORK/LEGWORK: LEGWORK v0.5.0, v0.5.0, Zenodo, doi: 10.5281/zenodo.12476977

Wagg, T., Breivik, K., & de Mink, S. 2022a, The Journal of Open Source Software, 7, 3998, doi: 10.21105/joss.03998

Wagg, T., Breivik, K., & de Mink, S. E. 2022b, ApJS, 260, 52, doi: 10.3847/1538-4365/ac5c52

Wagg, T., Broekgaarden, F., & Gültekin, K. 2024, TomWagg/software-citation-station: v1.2, v1.2, Zenodo, doi: 10.5281/zenodo.13225824

Wagg, T., & Broekgaarden, F. S. 2024, arXiv e-prints, arXiv:2406.04405. https://arxiv.org/abs/2406.04405

Wagg, T., Broekgaarden, F. S., de Mink, S. E., et al. 2022c, ApJ, 937, 118, doi: 10.3847/1538-4357/ac8675

Waskom, M. L. 2021, Journal of Open Source Software, 6, 3021, doi: 10.21105/joss.03021

Weinberg, M. D., Shapiro, S. L., & Wasserman, I. 1987, ApJ, 312, 367, doi: 10.1086/164883

Wes McKinney. 2010, in Proceedings of the 9th Python in Science Conference, ed. Stéfan van der Walt & Jarrod Millman, 56 – 61, doi: 10.25080/Majora-92bf1922-00a

Wetzel, A., Hayward, C. C., Sanderson, R. E., et al. 2023, ApJS, 265, 44, doi: 10.3847/1538-4365/acb99a

Wetzel, A. R., Hopkins, P. F., Kim, J.-h., et al. 2016, ApJL, 827, L23, doi: 10.3847/2041-8205/827/2/L23

Zapartas, E., de Mink, S. E., Izzard, R. G., et al. 2017, A&A, 601, A29, doi: 10.1051/0004-6361/201629685

Zonca, A., Singer, L., Lenz, D., et al. 2019, Journal of Open Source Software, 4, 1298, doi: 10.21105/joss.01298

Zonca, A., Singer, L., crosset, et al. 2024, healpy/healpy: 1.17.3, 1.17.3, Zenodo, doi: 10.5281/zenodo.12746571

APPENDIX

A. TYPICAL SIMULATION CODE

In this Section, we demonstrate the code for a typical `cogsworth` simulation, to illustrate its ease-of-use and flexibility. The following code block shows how one can run a basic `cogsworth` simulation and access and interpret a variety of results.

```
import cogsworth
import gala.potential as gp
import astropy.units as u

# run the simulation
p = cogsworth.pop.Population(
    n_binaries=1000,
    processes=6,
    sfh_model=cogsworth.sfh.Wagg2022,
    galactic_potential=gp.MilkyWayPotential2022(),
    v_dispersion=5 * u.km / u.s,
    max_ev_time=12 * u.Gyr,
    BSE_settings={
        # adjust binary stellar evolution settings here
    },
    sampling_params={
        # adjust initial condition sampling settings here
    }
)
p.create_population()

# access DataFrames of initial conditions + evolution
p.initC, p.bpp

# explore Gala orbits, final positions/velocities
p.orbits, p.final_pos, p.final_vel

# convert to observables (e.g. flux, colour)
p.get_observables(filters=["G", "BP", "RP"],
              assume_mw_galactocentric=True)

# make some plots
p.plot_cartoon_binary(bin_num=42)
p.plot_orbit(bin_num=42)
p.plot_sky_locations()
cogsworth.plot.plot_cmd(p, "G", "BP", "RP")

# save population for later
p.save("population.h5")
```

In relatively few lines of code, this simulation allows users to sample binaries from a SFH, evolve the stars until present day with `COSMIC`, integrate their orbits through a galactic potential with `Gala`, convert the intrinsic population to observables and create a series of plots for interpreting the result (including similar plots to Figures 2 and 8). `cogsworth` will use the default choices for binary stellar evolution and sampling settings when `BSE_settings` and `sampling_params` are left empty respectively. Each settings that is individually added to the input dictionary will override the default, such that `BSE_settings = {'alpha1': 0.5}` would change the efficiency of common-envelope events to 0.5 to leave the other defaults unchanged. For a full list of the settings one case change via `BSE_settings` and `sampling_params`, see the `COSMIC` documentation.