

---

# **BAGGInS: Bayesian Analysis of Galaxy-Galaxy Interactions in Simulations**

***Release 1.0***

**Alex Rawlings**

**Oct 21, 2024**



## CONTENTS:

<b>1</b>	<b>Python API</b>	<b>1</b>
1.1	BAGGInS . . . . .	1
1.1.1	_backend package . . . . .	1
1.1.1.1	Submodules . . . . .	1
1.1.1.2	_backend.Logging module . . . . .	1
1.1.1.3	_backend.States module . . . . .	1
1.1.1.4	Module contents . . . . .	2
1.1.2	analysis package . . . . .	2
1.1.2.1	Subpackages . . . . .	2
1.1.2.2	Submodules . . . . .	3
1.1.2.3	analysis.analyse_ketju module . . . . .	3
1.1.2.4	analysis.analyse_snap module . . . . .	3
1.1.2.5	analysis.general module . . . . .	3
1.1.2.6	analysis.masks module . . . . .	3
1.1.2.7	analysis.orbits module . . . . .	5
1.1.2.8	analysis.voronoi module . . . . .	5
1.1.2.9	Module contents . . . . .	5
1.1.3	cosmology package . . . . .	5
1.1.3.1	Submodules . . . . .	5
1.1.3.2	cosmology.conversions module . . . . .	5
1.1.3.3	cosmology.cosmology module . . . . .	5
1.1.3.4	Module contents . . . . .	8
1.1.4	general package . . . . .	8
1.1.4.1	Submodules . . . . .	8
1.1.4.2	general.general module . . . . .	8
1.1.4.3	general.pygad_helper module . . . . .	8
1.1.4.4	general.units module . . . . .	8
1.1.4.5	Module contents . . . . .	8
1.1.5	initialise package . . . . .	8
1.1.5.1	Submodules . . . . .	8
1.1.5.2	initialise.GalaxyIC module . . . . .	8
1.1.5.3	initialise.MergerIC module . . . . .	8
1.1.5.4	initialise.galaxy_components module . . . . .	8
1.1.5.5	initialise.mergers module . . . . .	8
1.1.5.6	Module contents . . . . .	8
1.1.6	literature package . . . . .	8
1.1.6.1	Submodules . . . . .	8
1.1.6.2	literature.LiteratureTables module . . . . .	8
1.1.6.3	literature.bh_bulge module . . . . .	8
1.1.6.4	literature.density_profiles module . . . . .	8

1.1.6.5	literature.dm_bulge module . . . . .	8
1.1.6.6	literature.radial_relations module . . . . .	8
1.1.6.7	literature.smbh_recoil module . . . . .	8
1.1.6.8	literature.smbh_spins module . . . . .	8
1.1.6.9	Module contents . . . . .	8
1.1.7	mathematics package . . . . .	8
1.1.7.1	Submodules . . . . .	8
1.1.7.2	mathematics.array_operations module . . . . .	8
1.1.7.3	mathematics.coordinates module . . . . .	8
1.1.7.4	mathematics.geometry module . . . . .	8
1.1.7.5	mathematics.statistics module . . . . .	8
1.1.7.6	Module contents . . . . .	8
1.1.8	plotting package . . . . .	8
1.1.8.1	Submodules . . . . .	8
1.1.8.2	plotting.PlotClasses module . . . . .	8
1.1.8.3	plotting.custom_cmaps module . . . . .	8
1.1.8.4	plotting.general module . . . . .	8
1.1.8.5	plotting.plot_utils module . . . . .	8
1.1.8.6	plotting.specific_plots module . . . . .	8
1.1.8.7	Module contents . . . . .	8
1.1.9	utils package . . . . .	8
1.1.9.1	utils.data_handling module . . . . .	8
1.1.9.2	utils.dataset_operations module . . . . .	8
1.1.9.3	utils.get_cl_args module . . . . .	8
1.1.9.4	utils.os_operations module . . . . .	8
1.1.9.5	utils.param_io module . . . . .	8
1.1.9.6	Module contents . . . . .	8
1.1.10	visualisation package . . . . .	8
1.1.10.1	Submodules . . . . .	8
1.1.10.2	visualisation.animation module . . . . .	8
1.1.10.3	Module contents . . . . .	8

**Python Module Index** 9

**Index** 11

## PYTHON API

### 1.1 BAGGInS

Here you can find a list of available functions and classes in the BAGGInS package.

#### 1.1.1 \_backend package

##### 1.1.1.1 Submodules

##### 1.1.1.2 \_backend.Logging module

```
_backend.Logging.setup_logger(name, console_level='WARNING', logfile=None, file_level='WARNING', capture_warnings=True)
```

Set up a logger instance

##### Parameters

- **name** (*str*) – name of logger
- **console\_level** (*str, optional*) – logging level for console printing, by default “WARNING”
- **logfile** (*path-like, optional*) – file to print logs to, by default None
- **file\_level** (*str, optional*) – logging level for file printing, by default “WARNING”
- **capture\_warnings** (*bool, optional*) – redirect all warnings to the logger, by default True

##### Returns

*CustomLogger* – logger

##### 1.1.1.3 \_backend.States module

```
class _backend.States.PUBLISHINGSTATE
```

Bases: *\_State*

##### \_\_init\_\_()

Simple object to store the state of a plotting-style setting

```
class _backend.States.TMPDIRREGISTER(basename)
```

Bases: *object*

##### \_\_init\_\_(basename)

Class to hold a register of temporary directories. This means that temporary directories are particular to the given invocation of boggins, and can be cleaned up at exit without affecting the temporary directories of other invocations running concurrently.

**Parameters**

**basename** (*str, path-like*) – base temporary directory path to which a timestamp will be appended

**make\_new\_dir()**

Make a new temporary directory, and add it to the register

**property register**

#### 1.1.1.4 Module contents

`_backend.setup_logger(name, console_level='WARNING', logfile=None, file_level='WARNING', capture_warnings=True)`

Set up a logger instance

**Parameters**

- **name** (*str*) – name of logger
- **console\_level** (*str, optional*) – logging level for console printing, by default “WARNING”
- **logfile** (*path-like, optional*) – file to print logs to, by default None
- **file\_level** (*str, optional*) – logging level for file printing, by default “WARNING”
- **capture\_warnings** (*bool, optional*) – redirect all warnings to the logger, by default True

**Returns**

*CustomLogger* – logger

## 1.1.2 analysis package

### 1.1.2.1 Subpackages

#### analysis.analysis\_classes package

##### Submodules

`analysis.analysis_classes.BHBinary module`

`analysis.analysis_classes.BHBinaryData module`

`analysis.analysis_classes.Brownian module`

`analysis.analysis_classes.BrownianData module`

`analysis.analysis_classes.ChildSim module`

`analysis.analysis_classes.ChildSimData module`

`analysis.analysis_classes.CoreKick module`

`analysis.analysis_classes.GaussianProcesses module`

`analysis.analysis_classes.GrahamModels module`

`analysis.analysis_classes.HDF5Base module`

`analysis.analysis_classes.HMQuantitiesBinary module`

`analysis.analysis_classes.HMQuantitiesBinaryData module`

`analysis.analysis_classes.HMQuantitiesSingle module`

`analysis.analysis_classes.HMQuantitiesSingleData module`

`analysis.analysis_classes.KeplerModels module`

`analysis.analysis_classes.PQModels module`

`analysis.analysis_classes.QuinlanModels module`

`analysis.analysis_classes.StanModel module`

## Module contents

### 1.1.2.2 Submodules

`1.1.2.3 analysis.analyse_ketju module`

`1.1.2.4 analysis.analyse_snap module`

`1.1.2.5 analysis.general module`

`1.1.2.6 analysis.masks module`

`analysis.masks.get_all_id_masks(snap, family='stars')`

Return a dict of masks that mask the chosen particles by ID number to a specific galaxy, assuming each galaxy has a single SMBH in it.

#### Parameters

- `snap (pygad.Snapshot)` – snapshot to mask
- `family (str, optional)` – particle type we want to mask, by default ‘stars’

#### Returns

- `masks (dict)`
- `keys corresponding to the host galaxy SMBH particle ID, values to pygad.`
- `snapshot.masks.IDMask objects`

`analysis.masks.get_all_radial_masks(snap, radius, centre=None, id_masks=None, family='stars')`

Return a dict of masks that mask the chosen particles by radial distance to the desired centre. Either a ball mask or a shell mask may be constructed. Additionally, the option to initially filter particles by ID to constrain the mask to those particles belonging to a given galaxy is possible.

#### Parameters

- `snap (pygad.Snapshot)` – snapshot to mask
- `radius (float, tuple)` – radius to constrain the particles to - can be either a number to construct a ball mask, or a tuple of (inner\_radius, outer\_radius) to construct a shell mask
- `centre (pygad.UnitArr, optional)` – centre from which the radial measurements should be made, by default None
- `id_mask (dict, optional)` – ID masks to constrain particles to a given galaxy, by default None

- **family** (*str, optional*) – particle type to construct the mask for, by default None (all particle used)

**Returns**

**masks** (*dict*) – pygad mask objects, with keys corresponding to the host galaxy SMBH particle ID number

**Raises**

**ValueError** – centre method is invalid

**analysis.masks.get\_binding\_energy\_mask(snap, energy=None, id\_mask=None, family=None)**

Mask particles into bins of binding energy. Assumes that we are in centre of mass coordinates (including velocity): no centring is done.

**Parameters**

- **snap** (*pygad.Snapshot*) – snapshot to mask
- **energy** (*array-like, list of tuples, optional*) – energy bins to mask particles into - can either be a regular array, in which a “ball” mask is created, or a list of tuples, in which a “shell” mask is created. By default (None) creates a “shell” type mask from the 5% to 95% quantile of binding energy in 20 bins evenly spaced in logscale
- **id\_mask** (*pygad.snapshot.masks.IDMask, optional*) – ID mask to constrain particles to a given galaxy, by default None
- **family** (*\_type\_, optional*) – particle type we want to mask, by default None (all)

**Yields**

- **mask** (*pygad.snapshots.mask.IDMask*) – mask for that energy interval
- **energy** (*array-like, list of tuples*) – energy sequence
- *str* – energy units

**Raises**

**ValueError** – invalid energy input

**analysis.masks.get\_id\_mask(snap, bhid, family='stars')**

Obtain a mask that allows filtering of particular particles. Particles are first filtered to the host galaxy depending on their particle ID number, where it is necessarily assumed that the particle ID ordering follows:

galaxy 1 IDs → galaxy 2 IDs

To obtain a mask of all particles in a system, the routine can be run for each particle family, and the masks combined as e.g.:

```
all_id_mask = star_id_mask & dm_id_mask & bh_id_mask
```

#TODO should this be a separate method??

**Parameters**

- **snap** (*pygad.Snapshot*) – snapshot to mask
- **bhid** (*int*) – SMBH id number we want to find particles around
- **family** (*str, optional*) – particle type we want to mask, by default ‘stars’

**Returns**

*pygad.snapshot.masks.IDMask* – object to mask future snapshots of the same simulation

**Raises**

- **RuntimeError** – ID ordering is not divided by galaxy

- **RuntimeError** – only galaxy in the system

```
analysis.masks.get_radial_mask(snap, radius, centre=None, id_mask=None, family=None)
```

Create a radial-based mask, can be either a ball or a shell. The default function arguments are designed for compatibility with get\_all\_radial\_masks.

#### Parameters

- **snap** (*pygad.Snapshot*) – snapshot to mask
- **radius** (*float, tuple*) – radius to constrain the particles to - can be either a number to construct a ball mask, or a tuple of (inner\_radius, outer\_radius) to construct a shell mask
- **centre** (*pygad.UnitArr, optional*) – centre from which the radial measurements should be made, by default None
- **id\_mask** (*pygad.snapshot.masks.IDMask, optional*) – ID mask to constrain particles to a given galaxy, by default None
- **family** (*str, optional*) – particle type to construct the mask for, by default None (all particle used)

#### Returns

**mask** (*pygad.snapshot.masks.BallMask*) – radial mask of particles

#### Raises

**ValueError** – radius input is invalid

### 1.1.2.7 analysis.orbits module

### 1.1.2.8 analysis.voronoi module

### 1.1.2.9 Module contents

## 1.1.3 cosmology package

### 1.1.3.1 Submodules

### 1.1.3.2 cosmology.conversions module

```
cosmology.conversions.time2z(t, H0=67.36, pres=False)
```

Estimate the redshift for a corresponding cosmic time from Carmeli 2008

#### Parameters

- **t** (*float*) – cosmic time since big bang [Gyr] -> pres = False time before present [Gyr] -> pres = True
- **H0** (*float, optional*) – Hubble constant in km/s/Mpc, by default 100\*cosmology['h']
- **pres** (*bool, optional*) – t is given from today not Big Bang?, by default False

#### Returns

*float* – redshift

### 1.1.3.3 cosmology.cosmology module

```
cosmology.cosmology.angular_diameter_distance(z, cosmology={'h': 0.6736, 'omega_L': 0.6847, 'omega_M': 0.3153, 'zeq': 3402})
```

Determine the angular diameter distance for a flat universe

#### Parameters

- **z** (*float*) – redshift

- **cosmology** (*dict, optional*) – cosmological parameters, by default cosmology

**Returns**

*float* – angular diameter distance [kpc]

```
cosmology.cosmology.get_a0r(z, cosmology={'h': 0.6736, 'omega_L': 0.6847, 'omega_M': 0.3153, 'zeq': 3402})
```

Determine the value  $a_0 \cdot r$  from MBW eq. 3.106, for use in cosmological distance calculations

**Parameters**

- **z** (*float*) – redshift
- **cosmology** (*dict, optional*) – cosmological parameters, by default cosmology

**Returns**

*float* –  $a_0 \cdot r$  [kpc]

**Raises**

**AssertionError** – redshift larger than redshift of matter radiation equality

```
cosmology.cosmology.luminosity_distance(z, cosmology={'h': 0.6736, 'omega_L': 0.6847, 'omega_M': 0.3153, 'zeq': 3402})
```

Determine the luminosity distance for a flat universe

**Parameters**

- **z** (*float*) – redshift
- **cosmology** (*dict, optional*) – cosmological parameters, by default cosmology

**Returns**

*float* – luminosity distance [kpc]



**1.1.3.4 Module contents**

**1.1.4 general package**

**1.1.4.1 Submodules**

**1.1.4.2 general.general module**

**1.1.4.3 general.pygad\_helper module**

**1.1.4.4 general.units module**

**1.1.4.5 Module contents**

**1.1.5 initialise package**

**1.1.5.1 Submodules**

**1.1.5.2 initialise.GalaxyIC module**

**1.1.5.3 initialise.MergerIC module**

**1.1.5.4 initialise.galaxy\_components module**

**1.1.5.5 initialise.mergers module**

**1.1.5.6 Module contents**

**1.1.6 literature package**

**1.1.6.1 Submodules**

**1.1.6.2 literature.LiteratureTables module**

**1.1.6.3 literature.bh\_bulge module**

**1.1.6.4 literature.density\_profiles module**

**1.1.6.5 literature.dm\_bulge module**

**1.1.6.6 literature.radial\_relations module**

**1.1.6.7 literature.smbh\_recoil module**

**1.1.6.8 literature.smbh\_spins module**

**1.1.6.9 Module contents**

**1.1.7 mathematics package**

**1.1.7.1 Submodules**

**1.1.7.2 mathematics.array\_operations module**

**1.1.7.3 mathematics.coordinates module**

**1.1.7.4 mathematics.geometry module**

**1.1.7.5 mathematics.statistics module**

**1.1.7.6 Module contents**

**1.1.8 plotting package**

**1.1.8.1 Submodules**

**1.1.8.2 plotting.PlotClasses module**

**1.1.8.3 plotting.custom\_cmaps module**

**1.1.8.4 plotting.general module**

**1.1.8.5 plotting.plot\_utils module**

**1.1.8.6 plotting.specific\_plots module**

**1.1.8.7 Module contents**

## PYTHON MODULE INDEX

—  
\_backend, 2  
\_backend.Logging, 1  
\_backend.States, 1

### a

analysis.masks, 3

### c

cosmology, 8  
cosmology.conversions, 5  
cosmology.cosmology, 5



# INDEX

## Symbols

`__init__()` (`_backend.States.PublishingState` method), 1  
`__init__()` (`_backend.States.TmpDirRegister` method), 1  
`_backend`  
    `module`, 2  
`_backend.Logging`  
    `module`, 1  
`_backend.States`  
    `module`, 1

`make_new_dir()` (`_backend.States.TmpDirRegister` method), 2

`module`  
    `_backend`, 2  
    `_backend.Logging`, 1  
    `_backend.States`, 1  
    `analysis.masks`, 3  
    `cosmology`, 8  
    `cosmology.conversions`, 5  
    `cosmology.cosmology`, 5

## A

`analysis.masks`  
    `module`, 3  
`angular_diameter_distance()` (in module `cosmology.cosmology`), 5

## C

`cosmology`  
    `module`, 8  
`cosmology.conversions`  
    `module`, 5  
`cosmology.cosmology`  
    `module`, 5

## G

`get_a0r()` (in module `cosmology.cosmology`), 6  
`get_all_id_masks()` (in module `analysis.masks`), 3  
`get_all_radial_masks()` (in module `analysis.masks`), 3  
`get_binding_energy_mask()` (in module `analysis.masks`), 4  
`get_id_mask()` (in module `analysis.masks`), 4  
`get_radial_mask()` (in module `analysis.masks`), 5

## L

`luminosity_distance()` (in module `cosmology.cosmology`), 6

## M

`make_new_dir()` (`_backend.States.TmpDirRegister` method), 2

`module`  
    `_backend`, 2  
    `_backend.Logging`, 1  
    `_backend.States`, 1  
    `analysis.masks`, 3  
    `cosmology`, 8  
    `cosmology.conversions`, 5  
    `cosmology.cosmology`, 5

## P

`PublishingState` (class in `_backend.States`), 1

## R

`register` (`_backend.States.TmpDirRegister` property), 2

## S

`setup_logger()` (in module `_backend`), 2  
`setup_logger()` (in module `_backend.Logging`), 1

## T

`time2z()` (in module `cosmology.conversions`), 5  
`TmpDirRegister` (class in `_backend.States`), 1