mond_project Documentation

Release 0.0.1

Erika Wagoner, Michael Schott, John-Paul Mann

May 08, 2018

Contents:

Release 0.0.1

Date May 08, 2018

CHAPTER 1

Installation

First, clone the repo from github:

```
git clone https://github.com/wagoner47/mond_project.git
```

for HTTPS or:

```
git clone git@github.com:wagoner47/mond_project.git
```

for SSH.

Now change to the root `mond_project` directory:

```
cd mond_project
```

Finally, use the `setup.py` script:

```
python setup.py install
```

Standard command line options are available, such as `--prefix=`. However, an additional option should also be given, `--api-key=`, followed by the Illustris API key to use (you must first register with Illustris for this). This will create/modify a configuration file which is used by the code with the API key stored. This ensures that the user does not need to continually input the API key or have an environment variable which may be different on different operating systems.

CHAPTER 2

Reference/API

2.1 Data utilities

Utilities for reading and utilizing the data are provided through the `data_utils` subpackage. The bulk of this subpackage is contained within the *data_read_utils* module, which contains the function for getting the data from the the Illustris simulations.

2.1.1 Utilities for reading data (`data_read_utils`)

The `data_read_utils` module is the main interface point for accessing the data from the Illustris simulations. The primary function `get()` can be used recursively to access deeper levels of the data from the Illustris API, or can be used to access a table directly if the URL is already known. Once a table level page is reached, the data is stored in an HDF5 table, which can then be read in by the user using their favorite HDF5 reader in python, such as h5py.

A slightly higher level function for accessing the Illustris data can also be used, `save_halos()`. This function is built on the `get()` function, but it does the recursive calls for the user, and also only stores the relevant entries for the MOND calculations from the Illustris API. With this function, the user specifies a simulation (either the full name or the number for the base Illustris simulations), a snapshot number or redshift, and a directory in which to save the data. Any subhalo within the snapshot that qualifies as a galaxy is then queried for coordinates, velocities, and masses of all gas and star particles. The coordinates are used to calculate a radius within the galaxy, and the velocities are used to calculate a velocity dispersion (with respect to the galaxy), and the results are then stored into a single file per galaxy, with tags identifying each entry as "gas" or "star". The files are compressed pickle files which can be read with `pandas`, with names based upon the simulation, snapshot/redshift, and subhalo ID. File names are also stored in a "list file" in the same directory, saved as a numpy compressed binary file, and the file path for this list file is returned for future use. The columns and units for each subhalo file are as follows:

Column Name	Unit	Description
r	kpc	Radius, in physical units
M	$M_\odot$	Mass
v	km/s	Speed relative to galaxy, in physical units
type	n/a	Type of particle, either "gas" or "star"

Todo: Make sure we like our galaxy definition!

Todo: Do we need anything else to be saved for each subhalo?

One potential gotcha for using this is that the user must have an environment variable with the API key they will use to access the data from Illustris. To get this key, you must complete the registration through Illustris. The API key environment variable should be named "ILL_KEY", and can be set by adding to the appropriate user profile file for the system or by using the appropriate command line utility. For Linux/Unix based systems, for instance, to set and check the value:

```
$ export ILL_KEY=insert-your-assigned-api-key
$ echo $ILL_KEY
```

2.2 Calculate

The calculation part of this code gives functions that can be used when testing data with MOND (**Mo**dified **N**ewtonian **D**ynamics). This includes functions for calculating the observed and expected accelerations from data as well as a function for the prediction assuming MOND as found in Lelli et al. (2017).

2.2.1 Calculate accelerations in data (`calc_accel`)

The `calc_accel` module contains the functions needed to find accelerations in the Illustris data. The data is assumed to have been read already using `data_read_utils.save_halos()`, although the user could also get the data themselves and store with the same assumed structure. There are two accelerations that need to be calculated, and they are found very differently.

Observed acceleration

The observed acceleration is calculated from the velocities of stars and gas in a given galaxy or subhalo:

$$g_{obs}(R) = \frac{V_{obs}^2(R)}{R}$$

This can be calculated in the data rather simply by averaging over the value of $g_{obs}(R)$ for all star/gas particles within a bin at R.

Baryonic acceleration

The expected (or baryonic) acceleration is found by calculating the expected gravitational force given the mass of stars and gas within some radius, or alternatively by solving for the gradient of the gravitational potential due to stars and gas from the Poisson equation. As we have masses of stars and gas at different radii, we here use the force due to the sum of masses within the radius of interest:

$$M_{bar}(r < R) = \sum_{r_i < R} M_{stars}(r_i) + M_{gas}(r_i)$$

$$g_{bar}(R) = \frac{GM_{bar}(r < R)}{R^2}$$

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

M