sdss$_{c}atl_{u}tils Documentation

Release 0.0.32

Victor Calderon

Feb 11, 2019

Contents

I Getting Started 3

II What's New? 13

III Project Details 21

IV Reference 59

`SDSS_Catl_Utils` is a specialized python package for downloading and analyzing catalogues of galaxies and dark matter haloes. The core functionality of the package includes:

- Easy interface to download desired galaxy catalogues from **Calderon et al. (2019)**
- A modular, object-oriented framework for handling the galaxy catalogues.
- End-to-end support for reducing galaxy catalogues and caching them as fast-loading hdf5 files.

The source code is freely available at https://github.com/vcalderon2009/sdss_catl_utils

Part I

Getting Started

CHAPTER 1

Package Installation

Ther are different ways to install `SDSS_Catl_Utils`. The preferred way to install it is through pip. Another possibility is to install the package through the source code by cloning the repository from Github and build from the source code.

1.1 `pip` Installation

The simplest way to install `SDSS_Catl_Utils` is with `pip`. To install it with `pip`

```
pip install sdss_catl_utils
```

This will install the latest official release of the code.

1.2 Upgrading via `pip`

Whenever there is a new release, you can upgrade your current version by running

```
pip install --upgrade sdss_catl_utils
```

This will ensure that you have the most up-to-date version of `SDSS_Catl_Utils`.

Note: Consider installing `sdss_catl_utils` into a virtual environment. Setting this up is completely straightforward and takes less than a minute, even if this is your first time using a virtual environment. Using a virtual environment simplifies not just the current installation but also package upgrades and your subsequent workflow. If you use conda to manage your python distribution, you can find explicit instructions in the installing_sdss_catl_utils_with_virtualenv section of the documentation.

1.3 Building from Source

If you don't install the latest release using `pip`, you can instead `clone` the source code and call the setup file. This is the most common way to install `SDSS_Catl_Utils` if you want versions of the code that have been updated since the last latest official release. In this case, after installation, it is particularly important that you follow the instructions in *Verifying your installation* section bellow.

The first step is to clone the `SDSS_Catl_Utils` repository

```
git clone https://github.com/vcalderon2009/sdss_catl_utils
cd sdss_catl_utils
```

1.3.1 Installing one of the official releases

All oficial releases of the code are tagged with their version name, e.g. `v0.1.0` pertains to the `0.1.0` release. To install a particular release

```
git checkout v0.1.0
python setup.py install
```

This will install the `v0.1.0` release of the `SDSS_Catl_Utils`. Other official release version can be installed similarly.

1.3.2 Installing the most recent master branch

If you prefer to use the most recent version of the code

```
git checkout master
python setup.py install
```

This will install the `master` branch of the code, which is the branch currently under development. While the features in the officiail releases have a stable API, new features being developed in the `master` branch may not. However, the master branch may have *new* features and/or perfomance enhancements that you may wish to use for your science applications. A concerted effort is made to ensure that only thoroughly tested and documented code appears in the public `master` branch, though `SDSS_Catl_Utils` users should be awayre of the distinction between the bleeding edge version in master and the official release version available through `pip`.

1.4 Dependencies

If your install `sdss_catl_utils` using pip, then most of your dependencies will be handled for you automatically. The only additional dependency you may need is:

- h5py : 2.5 or later

The h5py package is used for fast I/O of galaxy and group catalogues.

If you did not use pip, then you should be aware of the following strict requirements:

- Python: 3x or higher
- Numpy
- Astropy
- Pandas

- h5py
- GitPython
- Cython
- requests
- numexpr
- Scipy
- Scikit-Learn
- BeautifulSoup
- wget
- tqdm
- cosmo-utils

Any of the above can be installed with either `pip` or `conda`.

1.5 Verifying your installation

After installing the code and its dependencies, fire up a Python interpreter and check that the version number matches what you expect:

```
>>> import sdss_catl_utils
>>> print(sdss_catl_utils.__version__)
```

If the version number is not what it should be, this likely means you have a previous installation that is superseding the version you tried to install. This *should* be accomplished by doing `pip uninstall sdss_catl_utils` before your new installation, but you may need to uninstall the previous build "manually". Like all python packages, you can find the installation location as follows:

```
>>> import sdss_catl_utils
>>> print(sdss_catl_utils.__file__)
```

This wil show where your active version is located on your machine. You can manually delete this copy of `SDSS_Catl_Utils` prior to your new installation to avoid version conflicts. (There may be multiple copies of `SDSS_Catl_Utils` in this location, depending on how many times you have previously installed the code - all such copies my be deleted prior to reinstallation).

CHAPTER 2

Importing SDSS_Catl_Utils

In order to encourage consistency amongst users in importing and using `SDSS_Catl_Utils` functionality, we have put together the following guidelines.

Since most of the functionality in `SDSS_Catl_Utils` resides in sub-packages, importing astropy as:

```
>>> import sdss_catl_utils
```

is not very useful. Instead, it is best to import the desired sub-package with the syntax:

```
>>> from sdss_catl_utils import subpackage
```

CHAPTER 3

Quickstart Guides and Tutorials

This section of the documentation contains both quickstart guides and detailed usage-tutorials. If you are instead looking for a quick reference to the API of some class or function, try looking in *Comprehensive SDSS_Catl_Utils Reference/API*.

3.1 Quickstart Guides

3.1.1 Getting started with SDSS_Catl_Utils

This 10-minute guide gives an overview of the functionality of `SDSS_Catl_Utils` and each of its subpackages.

3.1.2 Downloading suite of galaxy catalogues

Downloading and caching Galaxy- and Group-galaxy catalogues

This section of the documentation describes how to get up and running with the galaxy and group-galaxy catalogues provided by `SDSS_Catl_Utils`. To see if `SDSS_Catl_Utils` provides the catalogues that you need, check Calderon et al. (2018).

`SDSS_Catl_Utils` provides a handful of homogeneously processed galaxy and group galaxy catalogues and associated group membership data. These catalogues have been prepared into a standard form, and so once they are downloaded they will be directly added to your cache and can immediately be used for your science applications.

The class responsible for downloading and caching these catalogues is `DownloadManager`.

Part II

What's New?

CHAPTER 4

What's new in SDSS_Catl_Utils 0.0.1

This was the initial version of `SDSS_Catl_Utils` released on December 11, 2018.

`SDSS_Catl_Utils` 0.0.1 integrated functions and submodules under a single `SDSS_Catl_Utils` package with a unified installer.

CHAPTER 5

Major Release History

5.1 What's new in SDSS_Catl_Utils 0.0.2

`SDSS_Catl_Utils` `v0.0.2` is now available for installation with `pip`. New features are summarized below. See *Full Changelog* for details on smaller issues and bug-fixes. See *Major Release History* for full release history information.

5.1.1 New Utility Functions

Models

A new module `models` contains the set of `models` pertaining to previous/current analysis performed that have used the set of galaxy and group catalogues presented here.

There set of new functions include:

- `DownloadManager`
- `CatlUtils`
- `SDSSConformity`
- `SDSSCatlAnalysis`
- `SDSSMLAnalysis`

These functions allow for the **downloading** and **handling** of the galaxy and group catalogues.

Catalogue Utils

- New functionality was added to the `catl_utils` that allows for an recovery and extraction of the data in the galaxy/group catalogues.

5.2 What's new in (unreleased) SDSS_Catl_Utils 0.0.3

`SDSS_Catl_Utils` `v0.0.3` is currently under development. The latest release is `v0.0.2`, which can now be installed with pip. New features currently be developed for future release `v0.0.3` are summarized below. See *Major Release History* for full release history information.

CHAPTER 6

Full Changelog

6.1 0.0.32 (2019-02-11)

- Updated function `CatlUtils` and a problem with its `catl_merge` function

6.2 0.0.31 (2019-02-08)

- Added new functions to `catl_utils`

6.3 0.0.3 (2019-01-21)

- Added new parameter `sigma_clf_c`, which corresponds to the scatter in `log(L)` for central galaxies in the `conditional luminosity function` (CLF). The affected functions were: - `DownloadManager` - `catl_utils` - `check_input_params` - and more.

6.4 0.0.2 (2018-12-29)

- Added new functions to download catalogues to `DownloadManager`. It allows for the download of catalogues of specific characteristics.
- Added new function `CatlUtils` that acts as a tool for handling the synthetic catalogues.
- Added functions `catl_keys` and `catl_keys_prop` to facilitate the handling of the catalogues.
- Added unit-testing for `catl_utils` module.
- Added more functions for `catl_utils` module (`catl_prefix_main`, `check_input_params`)
- Added classes for the Conformity (`SDSSConformity`), ML (`SDSSMLAnalysis`), and Catl (`SDSSCatlAnalysis`) analyses.

6.5 0.0.1 (2018-12-11)

- Initial release

Part III

Project Details

CHAPTER 7

Authors and Credits

7.1 `SDSS_Catl_Utils` Coordinators

- Victor Calderon (victor.calderon@vanderbilt.edu)

CHAPTER 8

Licenses

8.1 SDSS_Catl_Utils License

`SDSS_Catl_Utils` is licensed under a 3-clause BSD style license:

Copyright (c) 2018, Victor Calderon All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Astropy Team nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8.2 Other Licenses

Full licenses for third-party software `SDSS_Catl_Utils` is derived from or included with `SDSS_Catl_Utils` can be found in the `'licenses/'` directory of the source code distribution.

CHAPTER 9

Comprehensive SDSS_Catl_Utils Reference/API

9.1 sdss_catl_utils Package

9.1.1 Functions

test(**kwargs)	Run the tests for the package.

test

sdss_catl_utils.**test**(***kwargs*)

Run the tests for the package.

This method builds arguments for and then calls `pytest.main`.

Parameters

package
[str, optional] The name of a specific package to test, e.g. 'io.fits' or 'utils'. Accepts comma separated string to specify multiple packages. If nothing is specified all default tests are run.

args
[str, optional] Additional arguments to be passed to `pytest.main` in the `args` keyword argument.

docs_path
[str, optional] The path to the documentation .rst files.

open_files
[bool, optional] Fail when any tests leave files open. Off by default, because this adds extra run time to the test suite. Requires the `psutil` package.

parallel
[int or 'auto', optional] When provided, run the tests in parallel on the specified number

of CPUs. If parallel is `'auto'`, it will use the all the cores on the machine. Requires the `pytest-xdist` plugin.

pastebin
[('failed', 'all', None), optional] Convenience option for turning on py.test pastebin output. Set to 'failed' to upload info for failed tests, or 'all' to upload info for all tests.

pdb
[bool, optional] Turn on PDB post-mortem analysis for failing tests. Same as specifying `--pdb` in `args`.

pep8
[bool, optional] Turn on PEP8 checking via the pytest-pep8 plugin and disable normal tests. Same as specifying `--pep8 -k pep8` in `args`.

plugins
[list, optional] Plugins to be passed to `pytest.main` in the `plugins` keyword argument.

remote_data
[{'none', 'astropy', 'any'}, optional] Controls whether to run tests marked with @pytest.mark.remote_data. This can be set to run no tests with remote data (`none`), only ones that use data from http://data.astropy.org (`astropy`), or all tests that use remote data (`any`). The default is `none`.

repeat
[`int`, optional] If set, specifies how many times each test should be run. This is useful for diagnosing sporadic failures.

skip_docs
[`bool`, optional] When `True`, skips running the doctests in the .rst files.

test_path
[str, optional] Specify location to test by path. May be a single file or directory. Must be specified absolutely or relative to the calling directory.

verbose
[bool, optional] Convenience option to turn on verbose output from py.test. Passing True is the same as specifying `-v` in `args`.

9.1.2 Classes

`SDSSCatlUtils_Error`(message)	Base class of all LSS_Utils-specific exceptions
`UnsupportedPythonError`	

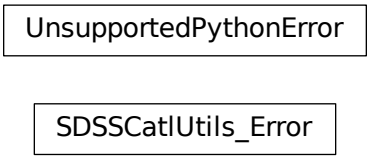
SDSSCatlUtils_Error

exception sdss_catl_utils.**SDSSCatlUtils_Error**(*message*)
Base class of all LSS_Utils-specific exceptions

UnsupportedPythonError

exception sdss_catl_utils.**UnsupportedPythonError**

9.1.3 Class Inheritance Diagram



9.2 sdss_catl_utils.mocks_manager Package

The mocks_manager sub-package is responsible for downloading galaxy and group galaxy catalogue data, storing hdf5 binaries and keeping a persistent memory of their location on disk and associated metadata.

9.3 sdss_catl_utils.mocks_manager.catl_utils Module

9.3.1 Functions

catl_keys([catl_kind, perf_opt, return_type])	Provides a dictionary/list with the corresponding keys for 1) halo mass, 2) Haloid/GroupID, 3) Group/Halo Galaxy-Type.
catl_keys_prop([catl_kind, catl_info, ...])	Provides a dictionary/list with the corresponding keys for 1) specific star formation rate (sSFR) and 2) stellar mass.
catl_clean(catl_pd, catl_kind[, catl_info, ...])	Cleans and removes the bad rows, i.e.
catl_clean_nmin(catl_pd, catl_kind[, ...])	Cleans and removed the bad rows with failed values, i.e.
catl_prefix_str([catl_kind, hod_n, ...])	Prefix string for the combination of input parameters that describe a set of catalog(s).
catl_prefix_path([catl_kind, hod_n, ...])	Prefix of the paths based on the type of catalogues and input parameters chosen.
catl_prefix_main([catl_type, catl_kind, ...])	Prefix of the paths based on the type of catalogues and input parameters chosen.
check_input_params(input_var, var_name[, ...])	Checks the type and/or values for different variables.

catl_keys

sdss_catl_utils.mocks_manager.catl_utils.**catl_keys**(*catl_kind='data'*, *perf_opt=False*, *return_type='list'*)

Provides a dictionary/list with the corresponding keys for 1) halo mass, 2) Haloid/GroupID, 3) Group/Halo Galaxy-Type.

Parameters

catl_kind
[{`data`, `mocks`} `str`, optional] Type of the catalogue being analyzed. This variable corresponds to whether a `real` or `synthetic/mock` catalogue is being read/analyzed. This variable is set to `data` by default.

Options:

- `data`: Catalogue(s) from the SDSS `real` catalogues
- `mocks`: Catalogue(s) from the `mock` catalogues.

perf_opt
[`bool`, optional] If `True`, it returns the corresponding keys for a `perfect` SDSS catalogue. This option only applies when `catl_kind == 'mocks'`.

return_type
[{`list`, `dict`} `str`, optional] Type of output to be returned. This variable is set to `list` by default.

Options:

- **`list`: Returns the output as part of a list. The order of** the elements are: `'group mass'`, `'groupid/haloid'`, and `Group/Halo ID`.

Returns

catl_objs
[`dict` or `list`] Dictionary or list of keys for `group/halo mass`, `group ID` and `galaxy type` columns in catalogues. This variable depends on the choice of `return_type`.

Raises

SDSSCatlUtils_Error
[Exception from `SDSSCatlUtils_Error`] Program exception if input parameters are `not` accepted.

Examples

This function can be used for several different combinations of parameters. For example, if we wanted to analyze a `data` catalogue, and have this function return a `list` as the output, one could write:

```
>>> catl_keys(catl_kind='data', return_type='list')
['M_h', 'groupid', 'galtype']
```

This list corresponds to the 1) Group estimated mass, 2) Galaxy's group ID, and 3) Galaxy's group galaxy type.

If instead, we wanted to analyze `perfect mock catalogues`, we could write:

```
>>> catl_keys(catl_kind='mocks', perf_opt=True, return_type='list')
['M_h', 'haloid', 'galtype']
```

For more information and examples, please refer to *Getting started with SDSS_Catl_Utils*.

catl_keys_prop

sdss_catl_utils.mocks_manager.catl_utils.**catl_keys_prop**(*catl_kind='data'*, *catl_info='memb'*, *return_type='list'*)

Provides a dictionary/list with the corresponding keys for 1) specific star formation rate (sSFR) and 2) stellar mass.

Parameters

catl_kind
[{`data`, `mocks`} `str`, optional] Type of the catalogue being analyzed. This variable corresponds to whether a `real` or `synthetic/mock` catalogue is being read/analyzed. This variable is set to `data` by default.

Options:

- `data`: Catalogue(s) from the SDSS `real` catalogues
- `mocks`: Catalogue(s) from the `mock` catalogues.

catl_info
[{`memb`, `groups`} `bool`, optional] Option for which type of catalogue is being analyzed. This variable correspondos to whether a `galaxy`-catalogue or a `group`-catalogue is being analyzed. This variable is set to `memb` by default.

Options:

- `memb`: Galaxy catalogue with the `member` galaxies of groups.
- `groups`: Catalogues with `group` information.

return_type
[{`list`, `dict`} `str`, optional] Type of output to be returned. This variable is set to `list` by default.

Options:

- **`list`: Returns the output as part of a list. The order of** the elements are: `'group mass'`, `'groupid/haloid'`, and `Group/Halo ID`.

Returns

---

catl_objs
[`dict` or `list`] Dictionary or list of keys for `logssfr` and `logmstar` columns in catalogues. This variable depends on the choice of `return_type`.

Raises

SDSSCatlUtils_Error
[Exception from `SDSSCatlUtils_Error`] Program exception if input parameters are `not` accepted.

Examples

This function can be used for several different combinations of parameters. For example, if we wanted to analyze a data catalogue, and have this function return a list as the output, one could write:

```
>>> catl_keys_prop(catl_kind='data', return_type='list')
['logssfr', 'logMstar_JHU']
```

This list corresponds to the 1) specific star formation rate key and 2) stellar mass key.

If instead, we wanted to analyze a mock catalogue and return the appropriate keys for a group catalogue, we could write:

```
>>> catl_keys_prop(catl_kind='mocks', catl_info='groups')
['logssfr', 'logMstar']
```

For more information and examples, please refer to *Getting started with SDSS_Catl_Utils*.

catl_clean

sdss_catl_utils.mocks_manager.catl_utils.**catl_clean**(*catl_pd*, *catl_kind*, *catl_info='memb'*, *reindex=True*)

Cleans and removes the bad rows, i.e. those that contain failed entries for sSFR and Mstar.

Parameters

catl_pd
[pandas.DataFrame] DataFrame containing the information about galaxies or galaxy groups.

catl_kind
[{data, mocks} str, optional] Type of the catalogue being analyzed. This variable corresponds to whether a real or synthetic/mock catalogue is being read/analyzed.

Options:

- data: Catalogue(s) from the SDSS real catalogues
- mocks: Catalogue(s) from the mock catalogues.

catl_info
[{memb, groups} bool, optional] Option for which type of catalogue is being analyzed. This variable correspondos to whether a galaxy-catalogue or a group-catalogue is being analyzed. This variable is set to memb by default.

Options:

- memb: Galaxy catalogue with the member galaxies of groups.
- groups: Catalogues with group information.

reindex
[bool, optional] If True, the output catalogue is reindexed from the original dataframe catl_pd. This variable is set to True by default.

Returns

catl_pd_mod
[pandas.DataFrame] Modified clean version of `catl_pd`. It removes the `failed` values.

Raises

SDSSCatlUtils_Error
[Exception from `SDSSCatlUtils_Error`] Program exception if input parameters are `not` accepted.

catl_clean_nmin

sdss_catl_utils.mocks_manager.catl_utils.**catl_clean_nmin**(*catl_pd*, *catl_kind*, *catl_info='memb'*, *reindex=True*, *nmin=1*, *perf_opt=False*)

Cleans and removed the `bad` rows with `failed` values, i.e. those that contain `failed` entries. This method also includes galaxies from groups above the `nmin` galaxy number threshold.

Parameters

catl_pd
[pandas.DataFrame] DataFrame containing the information about galaxies or galaxy groups.

catl_kind
[{`data`, `mocks`} `str`, optional] Type of the catalogue being analyzed. This variable corresponds to whether a `real` or `synthetic/mock` catalogue is being read/analyzed.

Options:

- `data`: Catalogue(s) from the SDSS `real` catalogues
- `mocks`: Catalogue(s) from the `mock` catalogues.

catl_info
[{`memb`, `groups`} `bool`, optional] Option for which type of catalogue is being analyzed. This variable correspondos to whether a `galaxy`-catalogue or a `group`-catalogue is being analyzed. This variable is set to `memb` by default.

Options:

- `memb`: Galaxy catalogue with the `member` galaxies of groups.
- `groups`: Catalogues with `group` information.

reindex
[`bool`, optional] If `True`, the output catalogue is reindexed from the original dataframe `catl_pd`. This variable is set to `True` by default.

nmin
[`int`, optional] Minimum group richness to have in the (galaxy) group catalogue. This variable is set to `1` by default, and must be larger than `1`.

perf_opt
[`bool`, optional] Option for using a `perfect` mock catalogue. This variable is set to `False` by default.

Returns

catl_pd_mod
[pandas.DataFrame] Version of `catl_pd` after having removed the `failed` values of `sSFR` and `Mstar`, and also after having chosen only galaxies and groups with group richnesses larger than `nmin`.

Raises

SDSSCatlUtils_Error
[Exception from `SDSSCatlUtils_Error`] Program exception if input parameters are `not` accepted.

Examples

Before using this function, one needs to have read one of the (galaxy) group catalogues. If for example, one wants to create a new object from the `data` `real` SDSS catalogue with galaxies from groups with `n > 10`, one can do:

```
>>> from cosmo_utils.utils import file_readers as cfr
>>> from sdss_catl_utils.mocks_manager.catl_utils import catl_clean_nmin
>>> nmin = 10 # Minimum number of galaxies in file
>>> catl_pd = cfr.read_hdf5_file_to_pandas_DF('/path/to/file') # doctest: +SKIP
>>> catl_mod = catl_clean_nmin(catl_pd, 'data', nmin=nmin) # doctest: +SKIP
```

Now, the resulting catalogue will only include galaxies from groups with `n > 10`.

catl_prefix_str

sdss_catl_utils.mocks_manager.catl_utils.**catl_prefix_str**(*catl_kind='mocks'*, *hod_n=0*, *halotype='fof'*, *clf_method=1*, *clf_seed=1235*, *dv=1.0*, *sigma_clf_c=0.1417*, *sample='19'*, *type_am='mr'*, *perf_opt=False*)

Prefix string for the combination of input parameters that describe a set of catalog(s).

Parameters

catl_kind
[{`data`, `mocks`} `str`] Kind of catalogues to download. This variable is set to `mocks` by default.

Options:

- `data`: Downloads the SDSS DR7 real catalogues.
- `mocks`: Downloads the synthetic catalogues of SDSS DR7.

hod_n
[`int`, optional] Number of the HOD model to use. This value is set to `0` by default.

halotype
[{'so', 'fof'}, `str`, optional] Type of dark matter definition to use. This value is set to `so` by default.

Options:

- `so`: Spherical Overdensity halo definition.
- `fof`: Friends-of-Friends halo definition.

clf_method
[{1, 2, 3}, `int`, optional] Method for assigning galaxy properties to mock galaxies. This variable dictates how galaxies are assigned luminosities or stellar masses based on their galaxy type and host halo's mass. This variable is set to `1` by default.

Options:

- `1`: Independent assignment of (g-r) colour, sersic, and specific star formation rate (`logssfr`)
- `2`: (g-r) colour dictates active/passive designation and draws values independently.
- `3`: (g-r) colour dictates active/passive designation, and assigns other galaxy properties for that given galaxy.

clf_seed
[`int`, optional] Value of the random seed used for the conditional luminosity function. This variable is set to `1235` default.

dv
[`float`, optional] Value for the `velocity bias` parameter. It is the difference between the galaxy and matter velocity profiles.

$$dv = \frac{v_g - v_c}{v_m - v_c}$$

where v_g is the galaxy's velocity; v_m, the matter velocity.

sigma_clf_c
[`float`, optional] Value of the scatter in log(L) for central galaxies, when being assigned during the `conditional luminosity function` (CLF). This variable is set to `0.1417` by default.

sample
[{'19', '20', '21'}, `str`, optional] Luminosity of the SDSS volume-limited sample to analyze. This variable is set to `'19'` by default.

Options:

- `'19'`: $M_r = 19$ volume-limited sample
- `'20'`: $M_r = 20$ volume-limited sample
- `'21'`: $M_r = 21$ volume-limited sample

type_am
[{'mr', 'mstar'}, `str`, optional] Type of Abundance matching used in the catalogue. This variable is set to `'mr'` by default.

Options:

- `'mr'`: Luminosity-based abundance matching used
- `'mstar'`: Stellar-mass-based abundance matching used.

perf_opt
[bool, optional] If True, it chooses to analyze the perfect version of the synthetic galaxy/group galaxy catalogues. Otherwise, it downloads the catalogues with group-finding errors included. This variable is set to False by default.

Returns

catl_pre_str
[str] String of the prefix for each file based on input parameters.

catl_prefix_path

sdss_catl_utils.mocks_manager.catl_utils.**catl_prefix_path**(*catl_kind='mocks'*, *hod_n=0*, *halotype='fof'*, *clf_method=1*, *clf_seed=1235*, *dv=1.0*, *sigma_clf_c=0.1417*, *sample='19'*, *type_am='mr'*, *perf_opt=False*)

Prefix of the paths based on the type of catalogues and input parameters chosen. It returns the typical path to the galaxy/group catalogues.

Parameters

catl_kind
[{data, mocks} str] Kind of catalogues to download. This variable is set to mocks by default.

Options:

- data: Downloads the SDSS DR7 real catalogues.
- mocks: Downloads the synthetic catalogues of SDSS DR7.

hod_n
[int, optional] Number of the HOD model to use. This value is set to 0 by default.

halotype
[{'so', 'fof'}, str, optional] Type of dark matter definition to use. This value is set to so by default.

Options:

- so: Spherical Overdensity halo definition.
- fof: Friends-of-Friends halo definition.

clf_method
[{1, 2, 3}, int, optional] Method for assigning galaxy properties to mock galaxies. This variable dictates how galaxies are assigned luminosities or stellar masses based on their galaxy type and host halo's mass. This variable is set to 1 by default.

Options:

- 1: Independent assignment of (g-r) colour, sersic, and specific star formation rate (logssfr)
- 2: (g-r) colour dictates active/passive designation and draws values independently.

- `3`: (g-r) colour dictates active/passive designation, and assigns other galaxy properties for that given galaxy.

clf_seed
[`int`, optional] Value of the random seed used for the conditional luminosity function. This variable is set to `1235` default.

dv
[`float`, optional] Value for the `velocity bias` parameter. It is the difference between the galaxy and matter velocity profiles.

$$dv = \frac{v_g - v_c}{v_m - v_c}$$

where v_g is the galaxy's velocity; v_m, the matter velocity.

sigma_clf_c
[`float`, optional] Value of the scatter in log(L) for central galaxies, when being assigned during the `conditional luminosity function` (CLF). This variable is set to `0.1417` by default.

sample
[{'19', '20', '21'}, `str`, optional] Luminosity of the SDSS volume-limited sample to analyze. This variable is set to `'19'` by default.

Options:

- `'19'`: $M_r = 19$ volume-limited sample
- `'20'`: $M_r = 20$ volume-limited sample
- `'21'`: $M_r = 21$ volume-limited sample

type_am
[{'mr', 'mstar'}, `str`, optional] Type of Abundance matching used in the catalogue. This variable is set to `'mr'` by default.

Options:

- `'mr'`: Luminosity-based abundance matching used
- `'mstar'`: Stellar-mass-based abundance matching used.

perf_opt
[`bool`, optional] If `True`, it chooses to analyze the `perfect` version of the synthetic galaxy/group galaxy catalogues. Otherwise, it downloads the catalogues with group-finding errors included. This variable is set to `False` by default.

Returns

catl_prefix
[`str`] Prefix of the paths based on the type of catalogues and input parameters.

catl_prefix_main

sdss_catl_utils.mocks_manager.catl_utils.**catl_prefix_main**(*catl_type='memb'*, *catl_kind='mocks'*, *hod_n=0*, *halotype='fof'*, *clf_method=1*, *clf_seed=1235*, *dv=1.0*, *sigma_clf_c=0.1417*, *sample='19'*, *type_am='mr'*, *perf_opt=False*)

Prefix of the paths based on the type of catalogues and input parameters chosen.

Parameters

catl_type
[{`memb`, `gal`, `group`} `str`, optional] Type of catalog to analyze. This option is set to `memb` by default.

Options:

- `memb`: Analyzes the member galaxy catalogues of a group catalog
- `gal`: Analyzes a simple galaxy catalogue
- `group`: Analyzes a `group` galaxy catalogues with galaxy groups.

catl_kind
[{`data`, `mocks`} `str`] Kind of catalogues to download. This variable is set to `mocks` by default.

Options:

- `data`: Downloads the SDSS DR7 real catalogues.
- `mocks`: Downloads the synthetic catalogues of SDSS DR7.

hod_n
[`int`, optional] Number of the HOD model to use. This value is set to `0` by default.

halotype
[{'so', 'fof'}, `str`, optional] Type of dark matter definition to use. This value is set to `so` by default.

Options:

- `so`: Spherical Overdensity halo definition.
- `fof`: Friends-of-Friends halo definition.

clf_method
[{1, 2, 3}, `int`, optional] Method for assigning galaxy properties to mock galaxies. This variable dictates how galaxies are assigned luminosities or stellar masses based on their galaxy type and host halo's mass. This variable is set to `1` by default.

Options:

- `1`: Independent assignment of (g-r) colour, sersic, and specific star formation rate (`logssfr`)
- `2`: (g-r) colour dictates active/passive designation and draws values independently.

- `3`: (g-r) colour dictates active/passive designation, and assigns other galaxy properties for that given galaxy.

clf_seed
[`int`, optional] Value of the random seed used for the conditional luminosity function. This variable is set to `1235` default.

dv
[`float`, optional] Value for the `velocity bias` parameter. It is the difference between the galaxy and matter velocity profiles.

$$dv = \frac{v_g - v_c}{v_m - v_c}$$

where v_g is the galaxy's velocity; v_m, the matter velocity.

sigma_clf_c
[`float`, optional] Value of the scatter in log(L) for central galaxies, when being assigned during the `conditional luminosity function` (CLF). This variable is set to `0.1417` by default.

sample
[{'19', '20', '21'}, `str`, optional] Luminosity of the SDSS volume-limited sample to analyze. This variable is set to `'19'` by default.

Options:

- `'19'`: $M_r = 19$ volume-limited sample
- `'20'`: $M_r = 20$ volume-limited sample
- `'21'`: $M_r = 21$ volume-limited sample

type_am
[{'mr', 'mstar'}, `str`, optional] Type of Abundance matching used in the catalogue. This variable is set to `'mr'` by default.

Options:

- `'mr'`: Luminosity-based abundance matching used
- `'mstar'`: Stellar-mass-based abundance matching used.

perf_opt
[`bool`, optional] If `True`, it chooses to analyze the `perfect` version of the synthetic galaxy/group galaxy catalogues. Otherwise, it downloads the catalogues with group-finding errors included. This variable is set to `False` by default.

Returns

catl_prefix
[`str`] Prefix of the paths based on the type of catalogues and input parameters.

check_input_params

sdss_catl_utils.mocks_manager.catl_utils.**check_input_params**(*input_var*, *var_name*, *check_type='type'*)

Checks the type and/or values for different variables.

Parameters

input_var
[int, float, bool, str] Input variable to be evaluated.

var_name
[str] Name of the input parameter being evaluated. This variable name must correspond to one of the keys in the type or vals dictionaries.

check_type
[{type, vals} str] Type of check to perform. This variable is set to type by default.

Options:

- type: It checks for the type of input_var.
- vals: It checks for the value of input_var.

Raises

SDSSCatlUtils_Error
[Exception from SDSSCatlUtils_Error] Program exception if input parameters are not accepted.

9.4 sdss_catl_utils.models Package

The models sub-package is responsible for downloading galaxy and group galaxy catalogue data, storing hdf5 binaries and keeping a persistent memory of their location on disk and associated metadata.

9.5 sdss_catl_utils.models.catl_models Module

9.5.1 Classes

DownloadManager(**kwargs)	Class used to scrape the web for galaxy and group galaxy catalogue data and cache the downloaded catalogues.
CatlUtils(**kwargs)	Class used to handle the galaxy/group galaxy/group catalogues, *after* the catalogues have been downloaded.
SDSSConformity()	Class used to handle the galaxy/group catalogues for the Calderon et al.(2018) analysis.
SDSSCatlAnalysis()	Class used to handle the galaxy/group catalogues for the Calderon et al.(2019) analysis.
SDSSMLAnalysis()	Class used to handle the galaxy/group catalogues for the Calderon et al.(2019) analysis.

DownloadManager

class sdss_catl_utils.models.catl_models.**DownloadManager**(***kwargs*)
Bases: sdss_catl_utils.models.catl_models_template.CatlClassTemplate

Class used to scrape the web for galaxy and group galaxy catalogue data and cache the downloaded catalogues.

For list of available pre-processed galaxy- and group-galaxy catalogues provided by `sdss_catl_utils`, see

Parameters

catl_kind
[{`data`, `mocks`} `str`] Kind of catalogues to download. This variable is set to `mocks` by default.

Options:

- `data`: Downloads the SDSS DR7 real catalogues.
- `mocks`: Downloads the synthetic catalogues of SDSS DR7.

hod_n
[`int`, optional] Number of the HOD model to use. This value is set to `0` by default.

halotype
[{'so', 'fof'}, `str`, optional] Type of dark matter definition to use. This value is set to `so` by default.

Options:

- `so`: Spherical Overdensity halo definition.
- `fof`: Friends-of-Friends halo definition.

clf_method
[{1, 2, 3}, `int`, optional] Method for assigning galaxy properties to mock galaxies. This variable dictates how galaxies are assigned luminosities or stellar masses based on their galaxy type and host halo's mass. This variable is set to `1` by default.

Options:

- `1`: Independent assignment of (g-r) colour, sersic, and specific star formation rate (`logssfr`)
- `2`: (g-r) colour dictates active/passive designation and draws values independently.
- `3`: (g-r) colour dictates active/passive designation, and assigns other galaxy properties for that given galaxy.

clf_seed
[`int`, optional] Value of the random seed used for the conditional luminosity function. This variable is set to `1235` default.

dv
[`float`, optional] Value for the `velocity bias` parameter. It is the difference between the galaxy and matter velocity profiles.

$$dv = \frac{v_g - v_c}{v_m - v_c}$$

where v_g is the galaxy's velocity; v_m, the matter velocity.

sigma_clf_c
[`float`, optional] Value of the scatter in log(L) for central galaxies, when being assigned during the `conditional luminosity function` (CLF). This variable is set to `0.1417` by default.

sample
[{'19', '20', '21'}, str, optional] Luminosity of the SDSS volume-limited sample to analyze. This variable is set to '19' by default.

Options:

- '19': $M_r = 19$ volume-limited sample
- '20': $M_r = 20$ volume-limited sample
- '21': $M_r = 21$ volume-limited sample

catl_type
[{'mr', 'mstar'}, str, optional] Type of Abundance matching used in the catalogue. This variable is set to 'mr' by default.

Options:

- 'mr': Luminosity-based abundance matching used
- 'mstar': Stellar-mass-based abundance matching used.

cosmo_choice
[{ 'LasDamas', 'Planck'} str, optional] Choice of cosmology to use. This variable is set to LasDamas by default.

Options:

- **LasDamas**
  [Uses the cosmological parameters from the] LasDamas simulations.
- Planck : Uses the Planck 2015 cosmology.

perf_opt
[bool, optional] If True, it chooses to analyze the perfect version of the synthetic galaxy/group galaxy catalogues. Otherwise, it downloads the catalogues with group-finding errors included. This variable is set to False by default.

environ_name
[str] Name of the environment variable to assign to outdir. This variable is set to the default environ_name from mocks_default

Examples

```
>>> from sdss_catl_utils.models.catl_models import DownloadManager
```

Methods Summary

catl_download([outdir, download_type, ext, ...])	Downloads the corresponding catalogues to the designated folder.
catl_web_params_check([catl_kind, ...])	Checks that the combination of parameters is valid and that there are catalogues available to download.
directory_tree_check([loc])	Checks the file structure of the current directory, and determines if it is a git repository or not.

Continued on next page

Table 5 – continued from previous page

local_outdir_path([loc, catl_type, ...])	Output directory for local copies of the catalogues.

Methods Documentation

catl_download(*outdir='./'*, *download_type='all'*, *ext='hdf5'*, *print_outdir=False*)

Downloads the corresponding catalogues to the designated folder.

Parameters

outdir
[str, optional] Output directory, to which catalogues will be saved. This variable is set to the default location of directory_tree_check. This variable is set to ./ by default.

download_type
[{all, data, mocks} str, optional] Type of catalogues to download. This variable is set to all by default.

Options:

- 'data': Downloads the SDSS data catalogues.
- 'mocks': Downloads the SDSS synthetic catalogues.
- 'all': Downloads both data and mocks catalogues.

ext
[{'hdf5'} str] File extension used for the catalogues. This variable is set to hdf5 by default.

print_outdir
[bool, optional] If True, it prints out the path to the output directories of the catalogues.

Examples

```
>>> # To download the synthetic catalogues
>>> from sdss_catl_utils.models.catl_models import DownloadManager
>>>
>>> # Downloading catalogues
>>> A = DownloadManager()
>>> A.catl_download(outdir='./', download_type='all') # doctest: +SKIP
>>>
```

catl_web_params_check(*catl_kind='mocks'*, *catl_type='memb'*, *ext='hdf5'*, *return_files_url=False*, *perf_opt=False*)

Checks that the combination of parameters is valid and that there are catalogues available to download.

Parameters

catl_kind
[{data, mocks} str] Kind of catalogues to download. This variable is set to mocks by default.

Options:

- data: Downloads the SDSS DR7 real catalogues.

- `mocks`: Downloads the synthetic catalogues of SDSS DR7.

catl_type
[{'gal', 'memb', 'group'}, `str`] Option for which kind of catalogue is being analyzed. This variable is set to `memb` by default.

Options:

- `'gal'` : Galaxy catalogue
- `'memb'` : Group Member galaxy catalogue
- `'group'` : Group galaxy catalogue

ext
[{'hdf5'} `str`] File extension used for the catalogues. This variable is set to `hdf5` by default.

return_files_url
[`bool`, optional] If True, it returns an array of files from the desired combination of parameters, and with a file extension `ext`. This variable is set to `False` by default.

perf_opt
[`bool`, optional] If True, it chooses to analyze the `perfect` version of the synthetic galaxy/group galaxy catalogues. Otherwise, it downloads the catalogues with group-finding errors included. This variable is set to `False` by default.

Returns

catl_url
[`str`] URL of the files being downloaded.

catl_files_url
[`numpy.ndarray`, optional]

directory_tree_check(*loc='./'*)
Checks the file structure of the current directory, and determines if it is a `git` repository or not. If not, it will save the catalogues to a location in the user's home directory, under `~/user/.sdss_catls`. It also adds the location of the output directory as an environment variable.

Parameters

loc
[`str`] Path to the directory being analyzed for possible git initialization directory. This variable is set to `./` by default.

Returns

outdir
[`str`] Path to the output directory, to which catalogues can be saved and stored.

local_outdir_path(*loc='./'*, *catl_type='memb'*, *catl_kind='data'*, *check_exist=False*, *create_dir=False*, *perf_opt=False*)
Output directory for `local` copies of the catalogues.

Parameters

loc
[`str`] Path to the directory being analyzed for possible git initialization directory.

catl_type
[{'gal', 'memb', 'group'}, str] Option for which kind of catalogue is being analyzed. This variable is set to memb by default.

Options:

- 'gal' : Galaxy catalogue
- 'memb' : Group Member galaxy catalogue
- 'group' : Group galaxy catalogue

catl_kind
[{data, mocks} str] Kind of catalogues to download. This variable is set to mocks by default.

Options:

- data: Downloads the SDSS DR7 real catalogues.
- mocks: Downloads the synthetic catalogues of SDSS DR7.

check_exist
[bool, optional] If True, it checks for whether or not the file exists. This variable is set to False by default.

create_dir
[bool, optional] If True, it creates the directory if it does not exist. This variable is set to False by default.

perf_opt
[bool, optional] If True, it chooses to analyze the perfect version of the synthetic galaxy/group galaxy catalogues. Otherwise, it downloads the catalogues with group-finding errors included. This variable is set to False by default.

Returns

outdir_local
[str] Path to the local output directory, based on the type of catalogues and input parameters.

CatlUtils

class sdss_catl_utils.models.catl_models.**CatlUtils**(***kwargs*)
Bases: sdss_catl_utils.models.catl_models_template.CatlClassTemplate

Class used to handle the galaxy/group galaxy/group catalogues, *after* the catalogues have been downloaded. This class has functions to read, modify, and analyze the galaxy/group catalogues.

For a list of available pre-processed galaxy- and group-galaxy catalogues provided by sdss_catl_utils, see ...

Parameters

catl_kind
[{data, mocks} str] Kind of catalogues to download. This variable is set to mocks by default.

Options:

- `data`: Downloads the SDSS DR7 real catalogues.
- `mocks`: Downloads the synthetic catalogues of SDSS DR7.

hod_n
[`int`, optional] Number of the HOD model to use. This value is set to `0` by default.

halotype
[{'so', 'fof'}, `str`, optional] Type of dark matter definition to use. This value is set to `so` by default.

Options:

- `so`: Spherical Overdensity halo definition.
- `fof`: Friends-of-Friends halo definition.

clf_method
[{1, 2, 3}, `int`, optional] Method for assigning galaxy properties to mock galaxies. This variable dictates how galaxies are assigned luminosities or stellar masses based on their galaxy type and host halo's mass. This variable is set to `1` by default.

Options:

- `1`: Independent assignment of (g-r) colour, sersic, and specific star formation rate (`logssfr`)
- `2`: (g-r) colour dictates active/passive designation and draws values independently.
- `3`: (g-r) colour dictates active/passive designation, and assigns other galaxy properties for that given galaxy.

clf_seed
[`int`, optional] Value of the random seed used for the conditional luminosity function. This variable is set to `1235` default.

dv
[`float`, optional] Value for the `velocity bias` parameter. It is the difference between the galaxy and matter velocity profiles.

$$dv = fracv_g - v_c v_m - v_c$$

where v_g is the galaxy's velocity; v_m, the matter velocity.

sigma_clf_c
[`float`, optional] Value of the scatter in log(L) for central galaxies, when being assigned during the `conditional luminosity function` (CLF). This variable is set to `0.1417` by default.

sample
[{'19', '20', '21'}, `str`, optional] Luminosity of the SDSS volume-limited sample to analyze. This variable is set to `'19'` by default.

Options:

- `'19'`: $M_r = 19$ volume-limited sample

- `'20'`: $M_r = 20$ volume-limited sample
- `'21'`: $M_r = 21$ volume-limited sample

type_am
[{'mr', 'mstar'}, `str`, optional] Type of Abundance matching used in the catalogue. This variable is set to `'mr'` by default.

Options:

- `'mr'`: Luminosity-based abundance matching used
- `'mstar'`: Stellar-mass-based abundance matching used.

cosmo_choice
[{ `'LasDamas'`, `'Planck'`} `str`, optional] Choice of cosmology to use. This variable is set to `LasDamas` by default.

Options:

- **`LasDamas`**
  [Uses the cosmological parameters from the] LasDamas simulations.
- `Planck` : Uses the Planck 2015 cosmology.

perf_opt
[`bool`, optional] If `True`, it chooses to analyze the `perfect` version of the synthetic galaxy/group galaxy catalogues. Otherwise, it downloads the catalogues with group-finding errors included. This variable is set to `False` by default.

environ_name
[`str`] Name of the environment variable to assign to `outdir`. This variable is set to the default `environ_name` from `mocks_default`

Notes

There are many combinations of parameters that could be performed. `However, note that not all variations exist and would not be available for download/analysis!`

Examples

```
>>> from sdss_catl_utils.models.catl_models import CatlUtils
```

If one wants to initialize an `catalogue` object with the default input parameters, one could write:

```
>>> catl_obj = CatlUtils() # Initialzing catalogue object
```

However, if one wants to have a `modified` version of `catl_obj`, one can pass a dictionary with the proper set of input parameters:

```
>>> params_d = {'catl_kind': 'mocks', 'clf_method': 3, 'clf_seed': 123}
>>> catl_obj = CatlUtils(**params_d)
```

This will create an `catalogue object` that corresponds to `mocks` catalogues with `clf_method = 3` and `clf_seed = 123`.

Methods Summary

catl_arr_extract([catl_type, catl_kind, ...])	Extracts the list of galaxy/group catalogues.
catl_merge([catl_idx, return_memb_group, ...])	Merges the member and group catalogues for a given set of input parameters, and returns a modified version of the galaxy group catalogues with added info about the galaxy groups.
catls_dir([catl_type, catl_kind, print_filedir])	Location of the group and group galaxy catalogues with the specified parameters.
get_params_dict()	Gets the dictionary of input parameters for the given model.
main_dir()	Path to the main directory, in which all catalogues are stored.

Methods Documentation

catl_arr_extract(*catl_type='memb'*, *catl_kind='mocks'*, *ext='hdf5'*, *print_filedir=False*, *return_len=False*)

Extracts the list of galaxy/group catalogues.

Parameters

catl_type
[{gal, memb, group}, str] Option for which kind of catalogue is being analyzed. This variable is set to memb by default.

Options:

- 'gal' : Galaxy catalogue
- 'memb' : Group Member galaxy catalogue
- 'group' : Group galaxy catalogue

catl_kind
[{data, mocks} str] Kind of catalogues to download. This variable is set to mocks by default.

Options:

- data: Downloads the SDSS DR7 real catalogues.
- mocks: Downloads the synthetic catalogues of SDSS DR7.

ext
[{'hdf5'} str] File extension used for the catalogues. This variable is set to hdf5 by default.

print_filedir
[bool, optional] If True, the path of the catalogue directory is printed onto the screen. This variable is set to False by default.

Returns

catls_arr
[numpy.ndarray, shape (N,)] Array of the paths of the galaxy/group catalogues with specified parameters. The shape of the variable is (N,), where N, is the number of catalogues in the directory.

Notes

If no files are present or exist in the given directory, it will return an empty array of files.

Examples

This function is able to return the path to individual kinds of galaxy and group catalogues that meet some criteria based on the different combinations of input parameters.

For example, if one wants to get the paths to the mocks member galaxy catalogues with a halotype = 'fof' prescription, and following a clf_method = 2 methodology, one could retrieve them by:

```
>>> params_dict = {'catl_kind': 'mocks', 'halotype': 'fof', 'clf_method': 2}
>>> catl_obj = CatlUtils(**params_dict) # doctest: +SKIP
>>> catl_paths = catl_arr_extract(catl_type='memb', catl_kind='mocks', halotype='fof') #␣
↪doctest: +SKIP
```

This will return the paths to the member galaxy catalogues that meet those prescribed parameters.

catl_merge(*catl_idx=0*, *return_memb_group=False*, *print_filedir=False*)
Merges the member and group catalogues for a given set of input parameters, and returns a modified version of the galaxy group catalogues with added info about the galaxy groups.

Parameters

catl_idx
[int, optional] Index of the catalogue to match. The index must be smaller than the number of files returned by function catl_arr_extract for the given combination of parameters. This variable is set to 0 by default.

return_memb_group
[bool, optional] If True, the function returns the member and group catalogues along with the merged catalogue. This variable is set to False by default.

print_filedir
[bool, optional] If True, the output directory is printed onto the screen. This variable is set to False by default.

Returns

merged_pd
[pandas.DataFrame] Combined DataFrame containing both galaxy and group information for the given combination of parameters and the catl_idx-th catalogue(s).

memb_pd
[pandas.DataFrame] Member galaxy catalogue of the catl_idx-th catalogue. This catalogue contains the information about the member galaxies. This object is only returned if return_memb_group == True,

group_pd
[pandas.DataFrame] Group galaxy catalogue of the catl_idx-th catalogue. This catalogue contains the information about the galaxy groups This object is only returned if return_memb_group == True,

Raises

SDSSCatlUtils_Error
[Exception from SDSSCatlUtils_Error] Program exception if input parameters are not accepted.

Examples

This function will merge the member and group catalogues into a single pandas.DataFrame object, and it can be used to merge two existing catalogues. For example, to merge the member and group catalogues using the *default* parameters, one can easily write:

```
>>> merged_pd = catl_merge() # doctest: +SKIP
```

The resulting object will consist of a merge between the member and group catalogues, with the columns pertaining to **group** properties having a GG_ attached to their names.

However, if one wants to merge two **mock** catalogues with a clf_method = 2 and halotype = 'fof' prescription, one could easily write this:

```
>>> params_dict = {'catl_kind': 'mocks', 'halotype': 'fof', 'clf_method': 2}
>>> catl_obj = CatlUtils(**params_dict) # doctest: +SKIP
>>> merged_pd = catl_obj.catl_merge() # doctest: +SKIP
```

Additionally, one could recover the member and group catalogues as well:

```
>>> merged_pd, memb_pd, group_pd = catl_obj.catl_merge(return_memb_group=True) # doctest:␣
↪+SKIP
```

For more information and examples, please refer to *Getting started with SDSS_Catl_Utils*.

catls_dir(*catl_type='memb'*, *catl_kind='mocks'*, *print_filedir=False*)
Location of the group and group galaxy catalogues with the specified parameters.

Parameters

catl_type
[{'gal', 'memb', 'group'}, str] Option for which kind of catalogue is being analyzed. This variable is set to memb by default.

Options:

- 'gal' : Galaxy catalogue
- 'memb' : Group Member galaxy catalogue
- 'group' : Group galaxy catalogue

catl_kind
[{data, mocks} str] Kind of catalogues to download. This variable is set to mocks by default.

Options:

- `data`: Downloads the SDSS DR7 real catalogues.
- `mocks`: Downloads the synthetic catalogues of SDSS DR7.

print_filedir
[`bool`, optional] If `True`, the path of the catalogue directory is printed onto the screen. This variable is set to `False` by default.

Returns

catls_dirpath
[`str`] Path to the location of the group and galaxy catalogues with the specified parameters.

Notes

This method can be used to access the different `kinds` of catalogues, i.e. galaxy-, member-, and group-catalogues for different combinations of input parameters.

Examples

`catls_dir` can be used to explore the directories that host different `kinds` of catalogues, i.e. galaxy-, member-, and group-catalogues for various combinations of input parameters.

For example, if one wants to read the `member` galaxy catalogue, i.e. a catalogue with both `galaxy` and `group` information, for `mocks` catalogues, one could easily write:

```
>>> memb_dir = catls_dir(catl_type='memb', catl_kind='mocks') # doctest: +SKIP
```

This will return the path to the `member` galaxy catalogues directory with the synthetic catalogues in it. This method only returns the path of the directory, and not the path to the actual files in the directory. To accomplish this, one would need to use the function `catl_arr_extract`.

get_params_dict()
Gets the dictionary of input parameters for the given model.

Returns

param_dict
[`dict`] Dictionary of input parameters for the chosen combination of parameters.

Examples

After having initializing an `CatlUtils` object, one can easily recover the set of input parameters used.

```
>>> from sdss_catl_utils.models.catl_models import CatlUtils
>>> catl_params_dict = {'catl_kind': 'mocks', 'clf_seed': 3} # Catalogue parameters
>>> catl_obj = CatlUtils(**catl_params_dict) # Initialing object
```

The dictionary of parameters is saved as `self.param_dict` for the class object. It can be easily accessed by:

```
>>> catl_obj.param_dict # doctest: +SKIP
```

Or, it can also be accessed as:

```
>>> catl_obj.get_params_dict() # doctest: +SKIP
```

For example, in order to print out the list of input parameters, one can do the following:

```
>>> param_dict = catl_obj.param_dict # doctest: +SKIP
>>> param_dict # doctest: +SKIP
{'catl_kind': 'mocks',
 'hod_n': 0,
 'halotype': 'fof',
 'clf_method': 1,
 'clf_seed': 3,
 'dv': 1.0,
 'sigma_clf_c': 0.1417,
 'sample': '19',
 'type_am': 'mr',
 'cosmo_choice': 'LasDamas',
 'perf_opt': False,
 'remove_files': False,
 'environ_name': 'sdss_catl_path',
 'sample_Mr': 'Mr19',
 'sample_s': '19'}
```

This dictionary can now be used in other parts of one's analysis, or as a reference of the parameters used.

main_dir()

Path to the main directory, in which all catalogues are stored. This directory depends on your installation of `SDSS_Catl_Utils` and on your current working space.

Returns

main_dirpath
[`str`] Path to the main directory, in which all catalogues are stored. It uses the environment variable `environ_name` from the class.

Examples

The main directory of the catalogues can be easily accessed after having created an object for the combination of input parameters for the catalogues.

```
>>> from sdss_catl_utils.models.catl_models import CatlUtils
>>> catl_params_dict = {'catl_kind': 'mocks', 'clf_seed': 3} # Catalogue parameters
>>> catl_obj = CatlUtils(**catl_params_dict) # Initialing object
```

One can access the directory, under which the galaxy- and group-catalogues are saved, by typing:

```
>>> catl_obj.main_dir() # doctest: +SKIP
```

This will return the path of the directory.

SDSSConformity

class sdss_catl_utils.models.catl_models.**SDSSConformity**

Bases: sdss_catl_utils.models.catl_models.CatlUtils, sdss_catl_utils.models.catl_models.DownloadManager

Class used to handle the galaxy/group catalogues for the `Calderon et al. (2018)` analysis. This class has functions to read, modify, and analyze the galaxy/group catalogues for this analysis.

The scripts and the rest of the codes for `SDSS Conformity` analysis can be found here

For a list of available pre-processed galaxy- and group-galaxy catalogues provided by `sdss_catl_utils`, see ...

Examples

This class serves as the gateway for downloading and handling the catalogues used in the Calderon et al. (2018) analysis.

One can easily *initialize* `SDSSConformity`:

```
>>> from sdss_catl_utils.models.catl_models import SDSSConformity
>>> catl_obj = SDSSConformity()
```

One can also get the parameters used in the Conformity analysis:

```
>>> catl_obj.param_dict # doctest: +SKIP
```

The catalogues used for the analysis can be easily downloaded via the `catl_download` method. If one wants to download the `data` catalogues, i.e. both `memb` and `group` catalogues, one can do it by:

```
>>> catl_obj.catl_download(download_type='data') # doctest: +SKIP
```

And if one wants to print out the paths of the URLs and output directories, one can do that by setting `print_outdir = True`:

```
>>> catl_obj.catl_download(download_type='data', print_outdir = True) # doctest: +SKIP
```

Additionally, one could also `extract` the paths to the catalogues. For example, to recover the path to the `data` `group` catalogue, you could do that by:

```
>>> catl_obj.catl_arr_extract(catl_kind='data', catl_type='group') # doctest: +SKIP
```

This would return an array with the path(s) of the catalogues in question.

SDSSCatlAnalysis

class sdss_catl_utils.models.catl_models.**SDSSCatlAnalysis**

Bases: sdss_catl_utils.models.catl_models.CatlUtils, sdss_catl_utils.models.catl_models.DownloadManager

Class used to handle the galaxy/group catalogues for the `Calderon et al. (2019)` analysis. This class has functions to read, modify, and analyze the galaxy/group catalogues for this analysis.

The scripts and the rest of the codes for `SDSS Conformity` analysis can be found here

For a list of available pre-processed galaxy- and group-galaxy catalogues provided by `sdss_catl_utils`, see ...

Examples

This class serves as the gateway for downloading and handling the catalogues used in the Calderon et al. (2018) analysis.

One can easily *initialize* SDSSCatlAnalysis:

```
>>> from sdss_catl_utils.models.catl_models import SDSSCatlAnalysis
>>> catl_obj = SDSSCatlAnalysis()
```

One can also get the parameters used in the Conformity analysis:

```
>>> catl_obj.param_dict # doctest: +SKIP
```

The catalogues used for the analysis can be easily downloaded via the catl_download method. If one wants to download the data catalogues, i.e. both memb and group catalogues, one can do it by:

```
>>> catl_obj.catl_download(download_type='data') # doctest: +SKIP
```

And if one wants to print out the paths of the URLs and output directories, one can do that by setting print_outdir = True:

```
>>> catl_obj.catl_download(download_type='data', print_outdir = True) # doctest: +SKIP
```

Additionally, one could also extract the paths to the catalogues. For example, to recover the path to the data group catalogue, you could do that by:

```
>>> catl_obj.catl_arr_extract(catl_kind='data', catl_type='group') # doctest: +SKIP
```

This would return an array with the path(s) of the catalogues in question.

SDSSMLAnalysis

class sdss_catl_utils.models.catl_models.**SDSSMLAnalysis**

Bases: sdss_catl_utils.models.catl_models.CatlUtils, sdss_catl_utils.models.catl_models.DownloadManager

Class used to handle the galaxy/group catalogues for the Calderon et al. (2019) analysis. This class has functions to read, modify, and analyze the galaxy/group catalogues for this analysis.

The scripts and the rest of the codes for SDSS Conformity analysis can be found here

For a list of available pre-processed galaxy- and group-galaxy catalogues provided by sdss_catl_utils, see ...

Examples

This class serves as the gateway for downloading and handling the catalogues used in the Calderon et al. (2019) analysis.

One can easily *initialize* SDSSMLAnalysis:

```
>>> from sdss_catl_utils.models.catl_models import SDSSMLAnalysis
>>> catl_obj = SDSSMLAnalysis()
```

One can also get the parameters used in the Conformity analysis:

```
>>> catl_obj.param_dict # doctest: +SKIP
```

The catalogues used for the analysis can be easily downloaded via the `catl_download` method. If one wants to download the `data` catalogues, i.e. both `memb` and `group` catalogues, one can do it by:

```
>>> catl_obj.catl_download(download_type='data') # doctest: +SKIP
```

And if one wants to print out the paths of the URLs and output directories, one can do that by setting `print_outdir = True`:

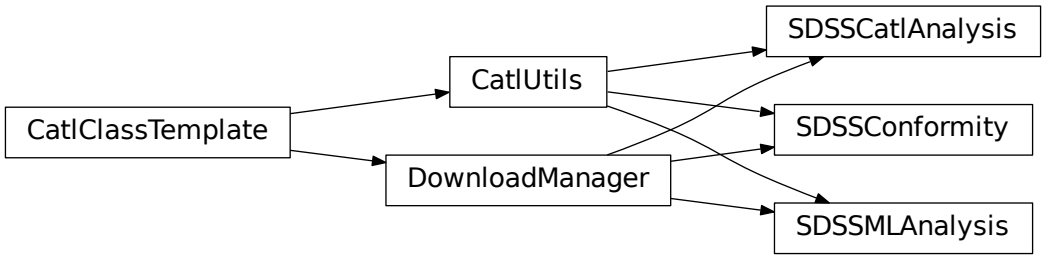
```
>>> catl_obj.catl_download(download_type='data', print_outdir = True) # doctest: +SKIP
```

Additionally, one could also `extract` the paths to the catalogues. For example, to recover the path to the `data` `group` catalogue, you could do that by:

```
>>> catl_obj.catl_arr_extract(catl_kind='data', catl_type='group') # doctest: +SKIP
```

This would return an array with the path(s) of the catalogues in question.

9.5.2 Class Inheritance Diagram



CHAPTER 10

sdss_catl_utils Documentation

This is the documentation for sdss_catl_utils.

10.1 Reference/API

10.1.1 sdss_catl_utils Package

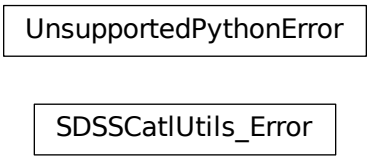
Functions

test(**kwargs)	Run the tests for the package.

Classes

SDSSCatlUtils_Error(message)	Base class of all LSS_Utils-specific exceptions
UnsupportedPythonError	

Class Inheritance Diagram



Part IV

Reference

- genindex
- modindex
- search

Python Module Index

S

sdss_catl_utils, 57
sdss_catl_utils.mocks_manager, 29
sdss_catl_utils.mocks_manager.catl_utils, 29
sdss_catl_utils.models, 40
sdss_catl_utils.models.catl_models, 40

Index

C

catl_arr_extract() (*sdss_catl_utils.models.catl_models.CatlUtils method*), 48
catl_clean() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 32
catl_clean_nmin() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 33
catl_download() (*sdss_catl_utils.models.catl_models.DownloadManager method*), 43
catl_keys() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 29
catl_keys_prop() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 31
catl_merge() (*sdss_catl_utils.models.catl_models.CatlUtils method*), 49
catl_prefix_main() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 38
catl_prefix_path() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 36
catl_prefix_str() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 34
catl_web_params_check() (*sdss_catl_utils.models.catl_models.DownloadManager method*), 43
catls_dir() (*sdss_catl_utils.models.catl_models.CatlUtils method*), 50
CatlUtils (*class in sdss_catl_utils.models.catl_models*), 45
check_input_params() (*in module sdss_catl_utils.mocks_manager.catl_utils*), 39

D

directory_tree_check() (*sdss_catl_utils.models.catl_models.DownloadManager method*), 44
DownloadManager (*class in sdss_catl_utils.models.catl_models*), 40

G

get_params_dict() (*sdss_catl_utils.models.catl_models.CatlUtils method*), 51

L

local_outdir_path() (*sdss_catl_utils.models.catl_models.DownloadMan method*), 44

M

main_dir() (*sdss_catl_utils.models.catl_models.CatlUtils method*), 52

S

sdss_catl_utils (*module*), 27, 57
sdss_catl_utils.mocks_manager (*module*), 29
sdss_catl_utils.mocks_manager.catl_utils (*module*), 29
sdss_catl_utils.models (*module*), 40
sdss_catl_utils.models.catl_models (*module*), 40
SDSSCatlAnalysis (*class in sdss_catl_utils.models.catl_models*), 53
SDSSCatlUtils_Error, 28
SDSSConformity (*class in sdss_catl_utils.models.catl_models*), 53
SDSSMLAnalysis (*class in sdss_catl_utils.models.catl_models*), 54

T

test() (*in module sdss_catl_utils*), 27

U

UnsupportedPythonError, 28