CosmoABC Documentation

Release stable

January 14, 2015

Contents

CosmoABC is a package which enables parameter inference using an Approximate Bayesian Computation (ABC) algorithm, as described in Ishida et al., 2015 [LINK]. The code was originally designed for cosmological parameter inference from galaxy clusters number counts based on Sunyaev-Zel'dovich measurements. In this context, the cosmological simulations were performed using the NumCosmo library.

Nevertheless, the user can easily take advantadge of the ABC sampler along with his/her own simulator, as well as test personalized summary statistics and distance functions.

CHAPTER 1

Get it now!

The package can be installed using the PyPI and pip:

```
$ pip install CosmoABC
```

Or if the tarball or repository is downloaded, in the CosmoABC directory you can install and test it:

```
$ pyton setup.py install
$ python setup.py test
```

CHAPTER 2

Examples

Sample input in can be found in ~CosmoABC/examples. All example files mentioned in this section are available in that directory.

The user input file should contain all necessary variables for simulation as well as for the ABC sampler.

A simple example of user input file, using a simulator which takes 3 parameters as input (mean, std, n) but only two of them are considered free would look like this

```
path_to_obs         = data.dat         # path to observed data

param_to_fit        = mean  std        # parameters to fit
param_to_sim        = mean  std  n     # parameters needed for simulation

mean_lim            = -10.0  10.0      # extreme limits for parameters
std_lim             = 0.001   3.0


mean_prior_par      = -1.0  3.0         # parameters for prior distribution
std_prior_par       = 0.001  2.0

mean        = 1.0                       #fiducial parameters for simulation
std         = 0.1
n           = 1000

s           = 0                         # extra parameter for distance function
epsilon1    = 100                       # initial distance threshold for building first particle syst
M           = 100                       # number of particles in each particle system
delta       = 0.2                       # convergence criteria
qthreshold  = 0.75                      # quantile in distance threshold
file_root   = example_2par_PS           # root to output file names

simulation_func   = simulation              # simulation function provided by the user
prior_func        = flat_prior  flat_prior  # list of prior PDF, one for each fitted parameter
distance_func     = distance                # distance function
```

2.1 User defined simulation, distance and prior functions

The most important ingredients in an ABC analysis are:

- the prior probability distributions (PDF)
- the simulator

- the distance function

CosmoABC is able to handdle user defined functions for all three elements. You will find example files in the corresponding directory which will help you to taylor your functions for the ABC sampler.

Built-in options for priors PDF are:

- Gaussian
- flat
- beta

Supposing that the user defined functions for distance and simulation are all in file containing

```
import numpy

def simulation( v ):
    """
    Generates a Gaussian distributed catalog.
    """

    l1 = numpy.random.normal( loc=v['mean'], scale=v['std'], size=v['n'] )

    return numpy.atleast_2d( l1 ).T


def distance( dataset1, dataset2, s1=0 ):
    """
    Calculates distance between dataset1 and dataset2.
    """

    t1 = abs( numpy.mean( dataset1 ) - numpy.mean( dataset2 ) )
    t2 = abs( numpy.std( dataset1 ) - numpy.std( dataset2 ) )

    return t1 + t2
```

The ABC sampler can be called from the command line:

```
$ run_ABC.py -i <user_input_file>  -f <user_function_file>
```

This will run the ABC sampler until the convergence criteria is reached. A pdf file containing graphical representation of the results for each particle system is given as output.

If the achieved result is not satisfactory and we want torun the ABC sampler beginning from the last completed particle system *N*, this can be done in the command line as well:

```
$ continue_ABC.py -i <user_input_file> -f <user_function> -p N
```

If the sampler is running and we wish to take a look in the already calculated particle systems, we can generate the corresponding plots:

```
$ plot_ABC.py -i <user_input_file> -p N
```

It is also possible to use it interactively.

```
from CosmoABC.priors import flat_prior
from CosmoABC.ABC_sampler import ABC
from CosmoABC.plots import plot_2D
import numpy

def simulation( v ):
```

```
    """
    Generates a Gaussian distributed catalog.
    """

    l1 = numpy.random.normal( loc=v['mean'], scale=v['std'], size=v['n'] )

    return numpy.atleast_2d( l1 ).T


def distance( dataset1, dataset2, s1=0 ):
    """
    Calculates distance between dataset1 and dataset2.
    """

    t1 = abs( numpy.mean( dataset1 ) - numpy.mean( dataset2 ) )
    t2 = abs( numpy.std( dataset1 ) - numpy.std( dataset2 ) )

    return t1 + t2


#define fiducial model parameters
mean = 1.0
std  = 0.1
v1 = {'mean': mean, 'std': std, 'n':1000 }

#generate 'observed' catalog
data = simulation( v1 )

#create dictionary of required parameter values
params = {}
params['param_to_fit']=['mean', 'std' ]                    # parameters to fit
params['param_lim']=[[-10, 10], [0.001, 3.0]]              # extreme limits for parameters
params['prior_par'] = [[-1.0, 3.0], [0.001,2.0]]           # parameters for prior distribution
params['simulation_params'] = v1                           # parameters needed for simulation

params['mean'] = mean                                      # fiducial parameter value
params['std']  = std                                       # fiducial parameter value
params['s']=0                                              # extra parameter for distance function
params['epsilon1'] = 50.0                                  # initial distance threshold
params['M'] = 100                                          # number of particles in each particle sy
params['delta'] =0.2                                       # convergence criteria
params['qthreshold'] = 0.75                                # quantile in distance threshold

params['file_root'] = 'example_PS'                         # root to output file names
params['prior_func'] = [ flat_prior, flat_prior]           # prior functions
params['distance_func'] = distance                         # distance functions

#initiate ABC sampler
sampler_ABC = ABC( dataset1=data, params=params, simulation_func=simulation, prior_func=params['prior

#build first particle system
sys1 = sampler_ABC.BuildFirstPSystem( filename=params['file_root'] + '0.dat' )

#update particle system until convergence
sampler_ABC.fullABC(  params['file_root'] )


#plot results
```

```
#update parameter limits for plotting

params['param_lim'] = [[-1.0,3.0],[0.0001,0.2]]
plot_2D( sampler_ABC.T, 'results.pdf' , params)
```

2.2 NumCosmo simulations

In order to reproduce the results of Ishida *et al.* 2015, first you need to make sure the NumCosmo library is running smoothly. Instructions for complete instalation and tests can be found in <*http://www.nongnu.org/numcosmo/*>.

Once the simulator is installed run the complete ABC sampler + NumCosmo cluster simulations from the command line

```
$ run_ABC_NumCosmo.py -i <user_input_file>
```

This will run the complete analysis presented in Ishida *et al.*, 2015 as well as produce plots with the corresponding results.

** WARNING** : This might take a while! Be patient!

Analogously to what is available for the user defined simulations, we can also continue a NumCosmo calculation from particle system *N* with:

```
$ continue_ABC_NumCosmo.py -i <user_input_file> -p N
```

If we want to run the NumCosmo simulation with a different prior or distance function, we should define it in a separated file and run:

```
$ run_ABC_NumCosmo.py -i <user_input_file> -f <user_function_file>
```

CHAPTER 3

Documentation

The complete documentation can be found in [LINK].

CHAPTER 4

Requirements

- Python 2.7
- numpy >=1.8.2
- scipy >= 0.14.0
- statsmodels >= 0.5.0
- matplotlib >= 1.3.1
- argparse >= 1.1
- imp
- math
- argparse

4.1 Optional

- NumCosmo <*http://www.nongnu.org/numcosmo/*>

CHAPTER 5

License

- GNU General Public License (GPL>=3)