Monte Python code structure

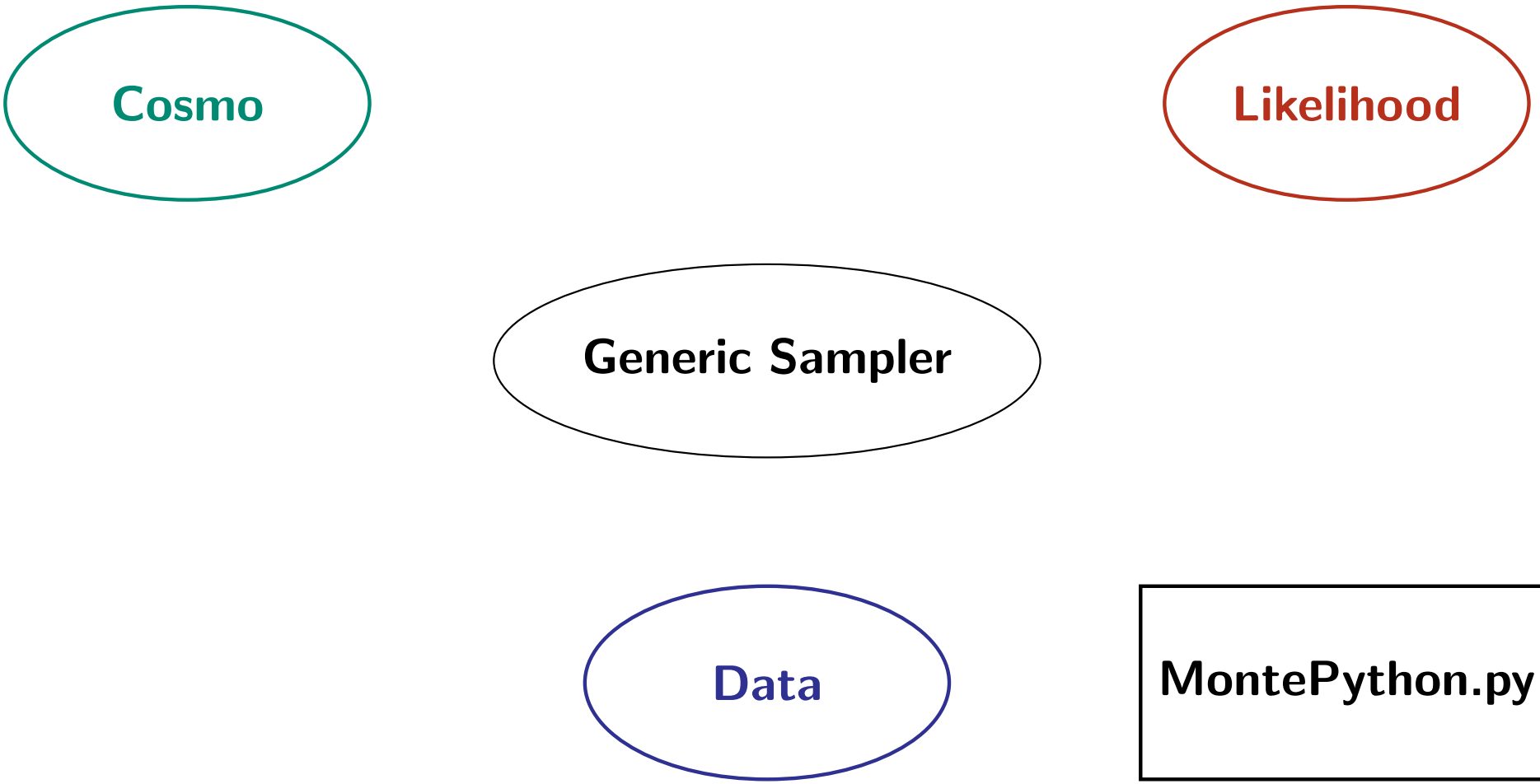
Benjamin Audren

École Polytechnique Fédérale de Lausanne

30/10/2014

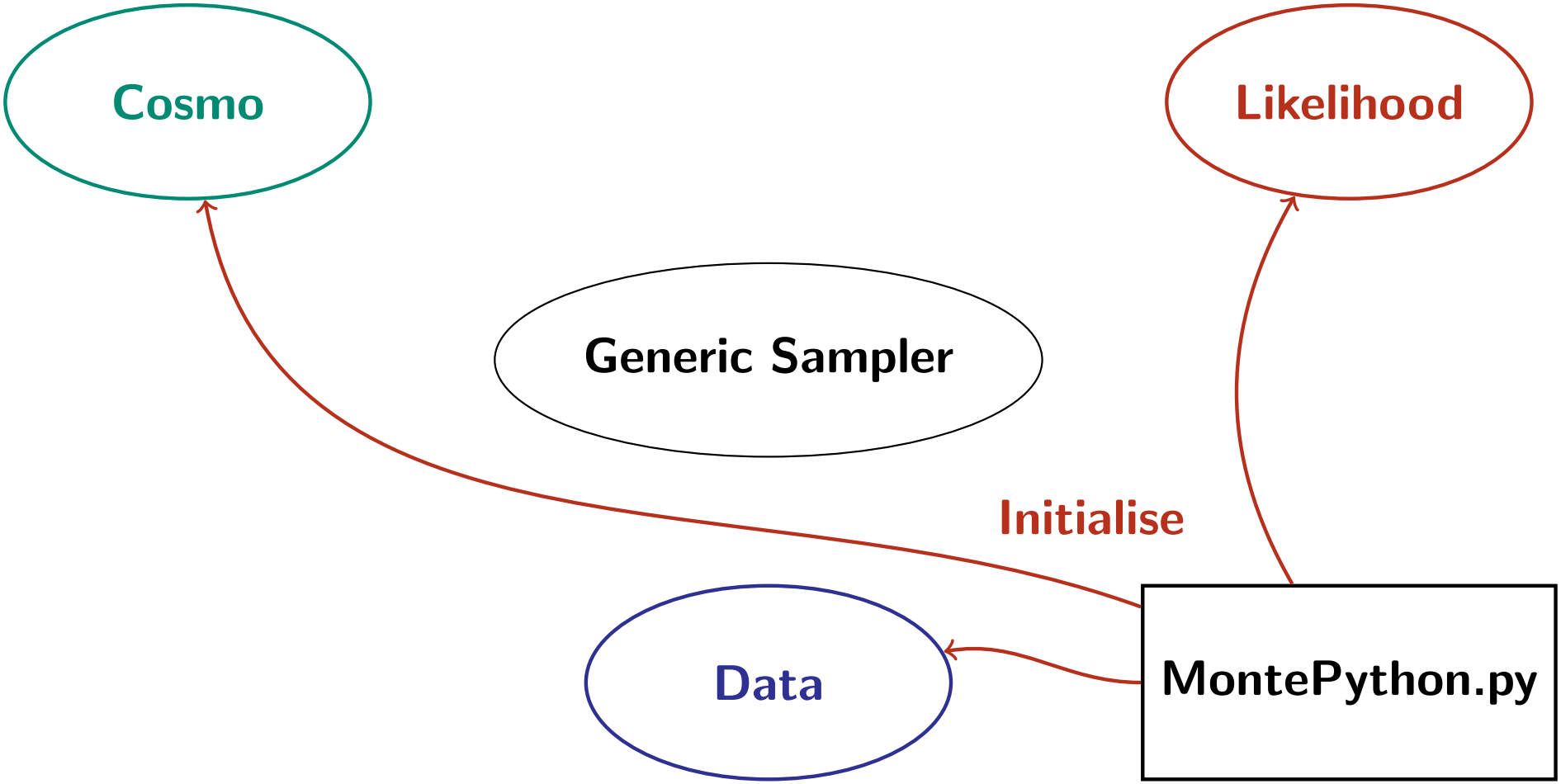
General structure of the modules

A drawing



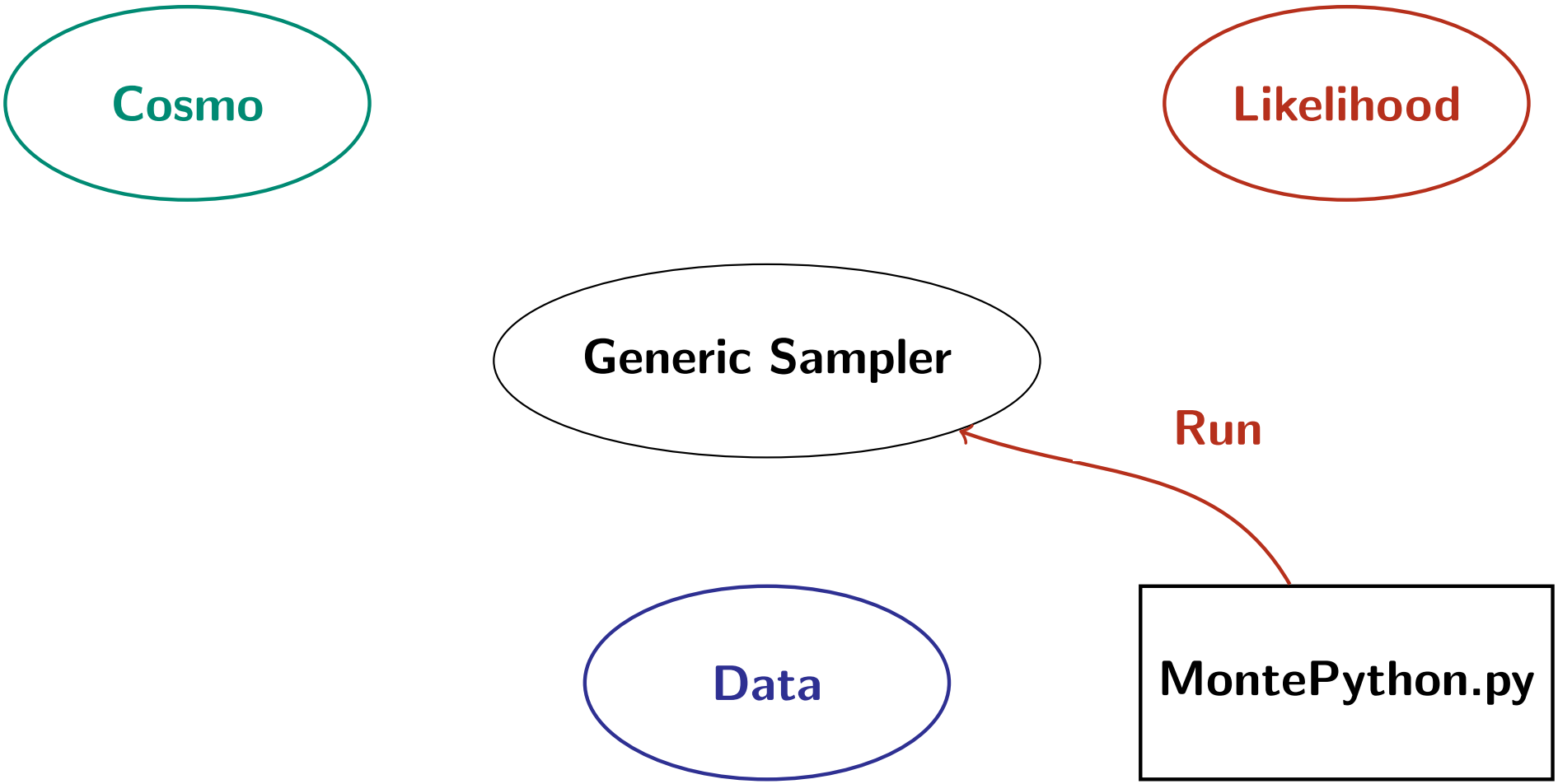
General structure of the modules

A drawing



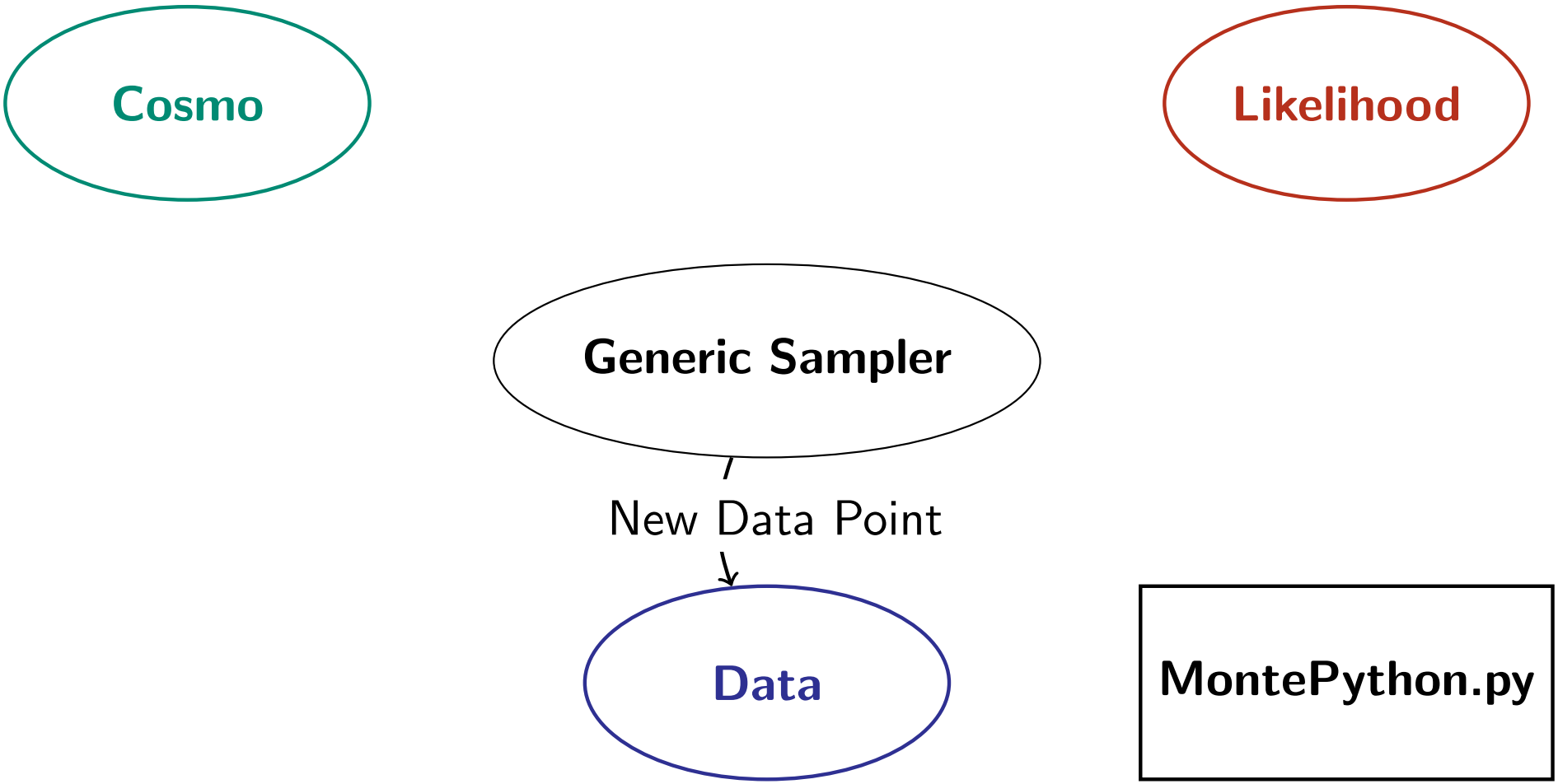
General structure of the modules

A drawing



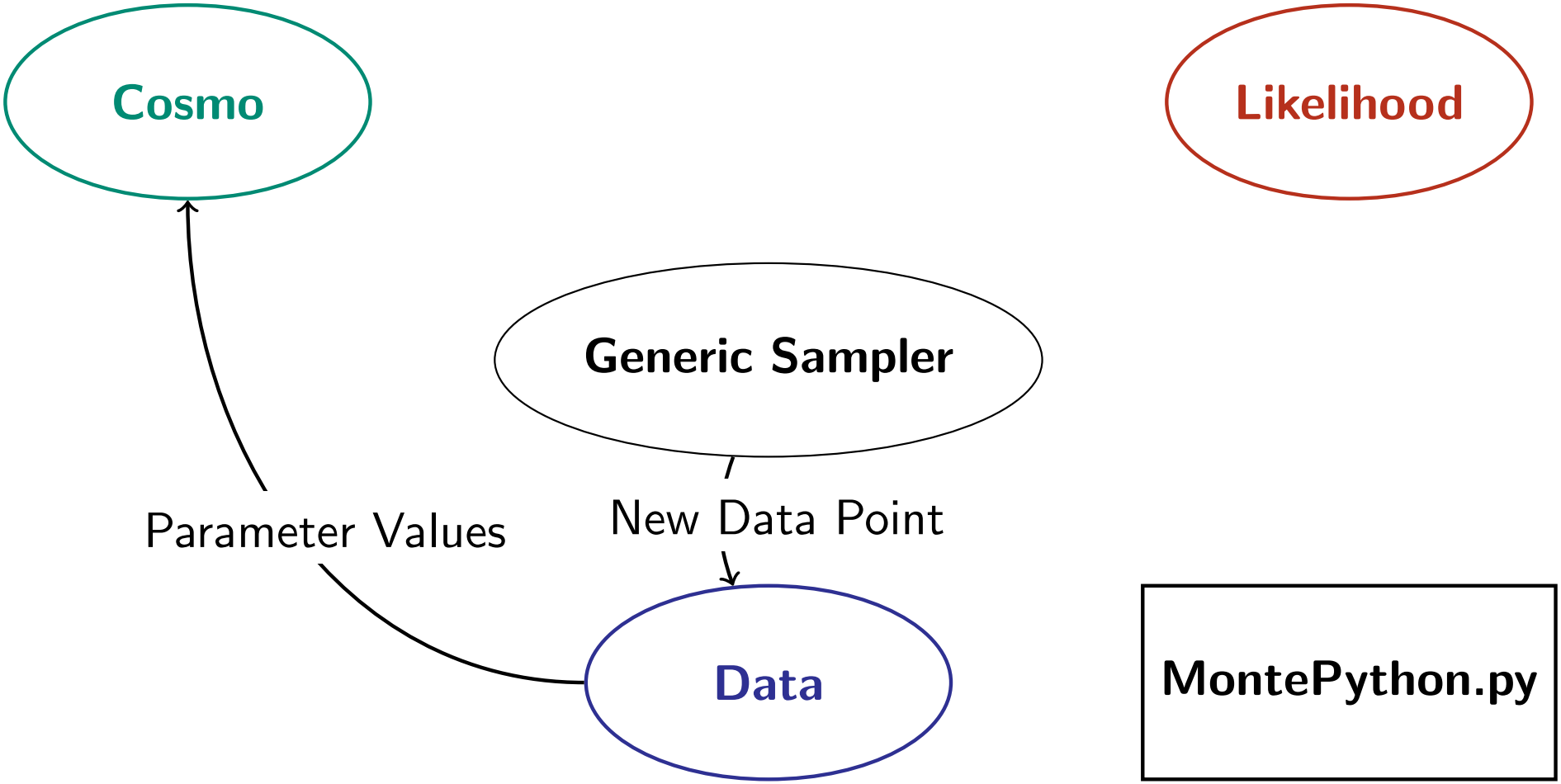
General structure of the modules

A drawing



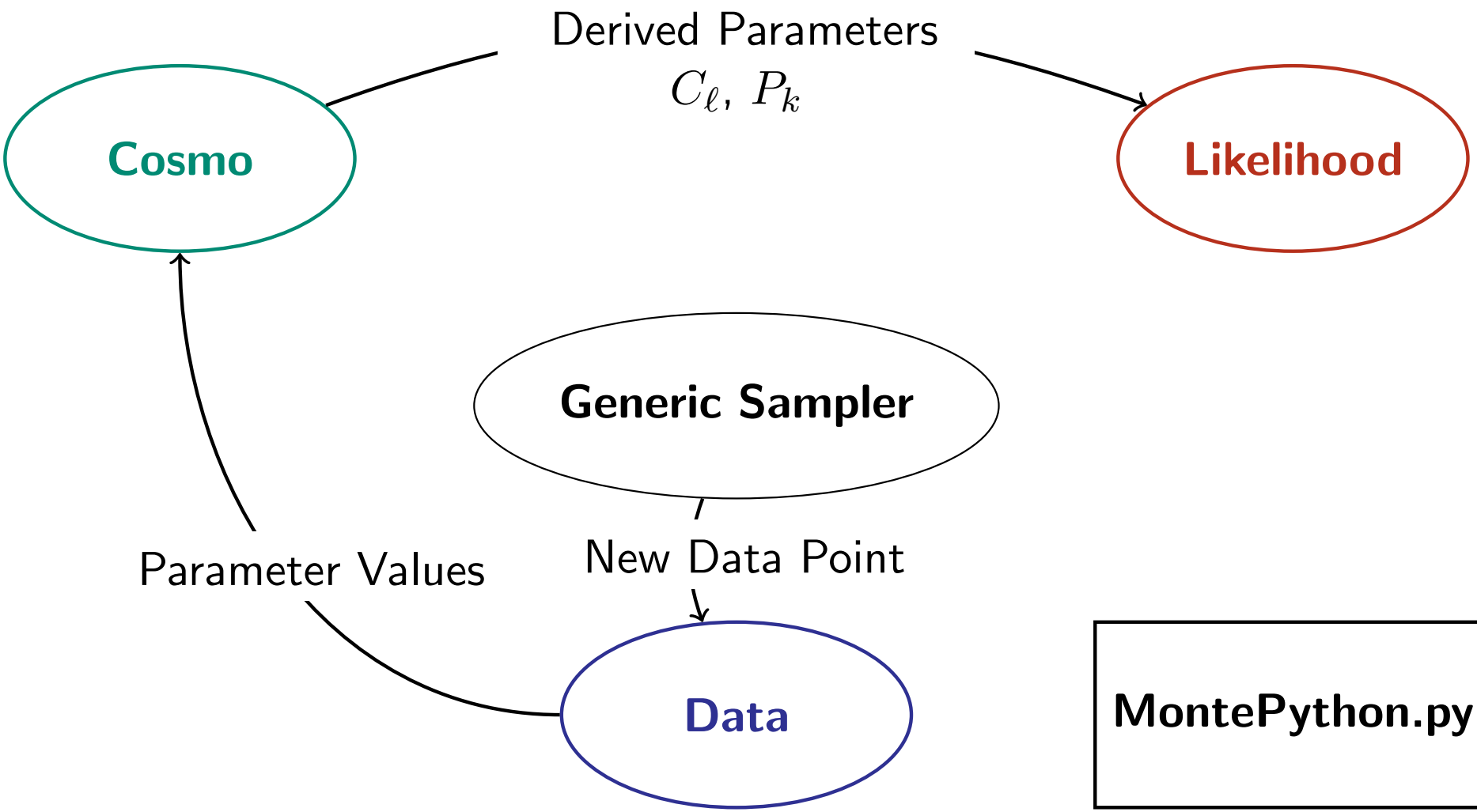
General structure of the modules

A drawing



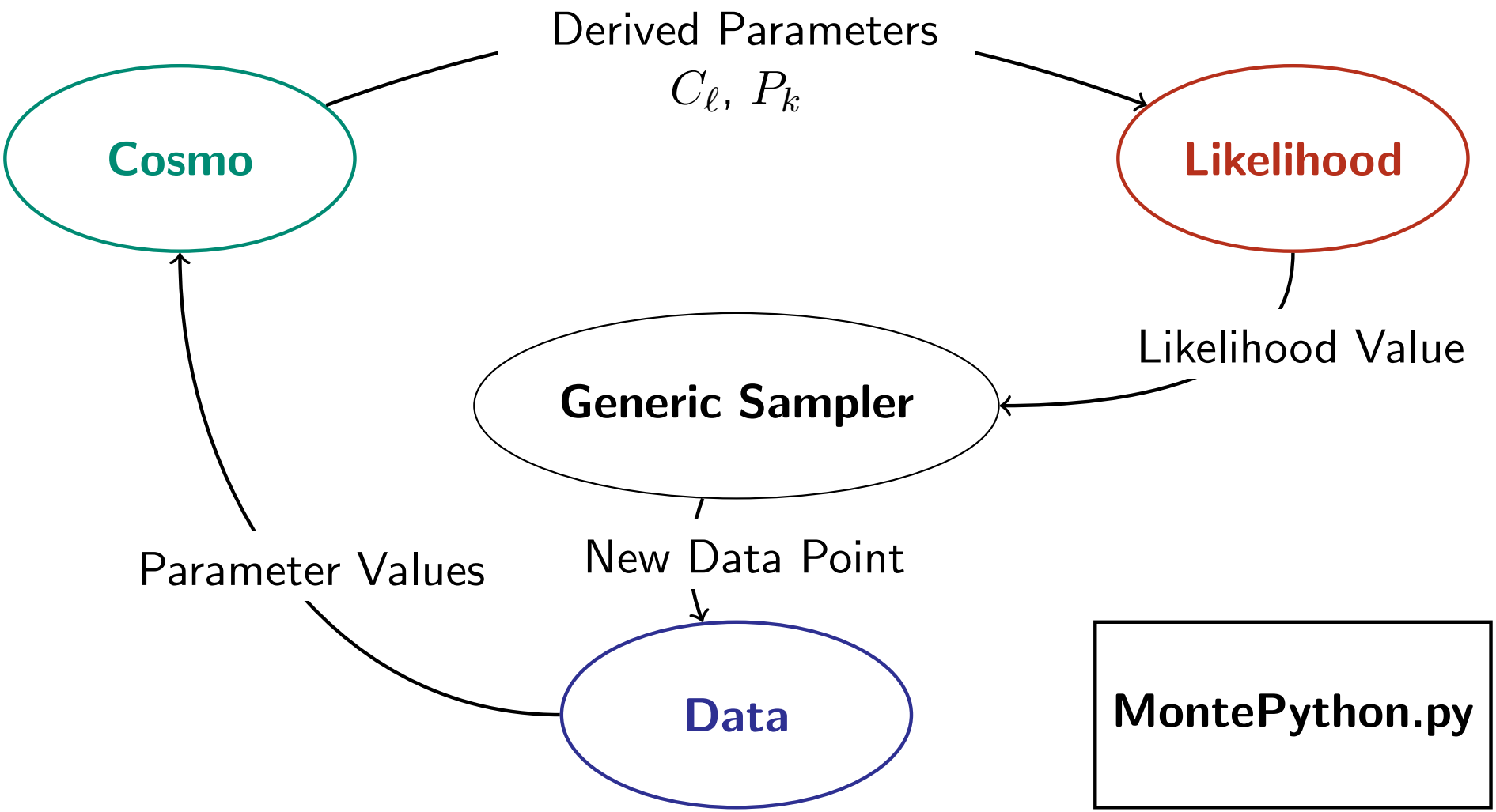
General structure of the modules

A drawing



General structure of the modules

A drawing



Information containers

Foreword: class definitions in capital letters, instances in small.

Classes

- **Data** defined in `data.py`
- **Class** defined in `classy.pyx`
- **Likelihood** defined in `likelihood_class.py`

Information containers

Foreword: class definitions in capital letters, instances in small.

Classes

- **Data** defined in `data.py`
- **Class** defined in `classy.pyx`
- **Likelihood** defined in `likelihood_class.py`

instances

- **data** initialized in `initialise.py`
- **cosmo** initialized in `initialise.py`
- **hst, bicep2, ...** initialized in `initialise.py`

General structure of the modules

Main Modules

- `MontePython` Simple script launching the code
- `run` call initialise, and launch a sampler session
- `parser_mp` reads the command line arguments
- `initialise` creates a cosmological code, Data and likelihoods instances
- `data` defines the Data class, where Parameters are initialized
- `sampler` Generic Sampler calling MCMC, or MultiNest, or CosmoHammer
- `likelihood_class` Likelihood computation for generic ones

General structure of the modules

Main Modules

- `MontePython` Simple script launching the code
- `run` call initialise, and launch a sampler session
- `parser_mp` reads the command line arguments
- `initialise` creates a cosmological code, Data and likelihoods instances
- `data` defines the Data class, where Parameters are initialized
- `sampler` Generic Sampler calling MCMC, or MultiNest, or CosmoHammer
- `likelihood_class` Likelihood computation for generic ones

Helper Modules

- `analyze` Computes convergence, posterior from chains
- `io_mp` Handles I/O, error message

MontePython.py

Role

Convenience script that calls the Monte Python run function.

Initialise I

Main

- **Reads command line, configuration file**
- Creates a data instance
- Initializes the cosmological module

```
29    # Parsing line argument
30    command_line = parser_mp.parse(custom_command)
31
32    # Recovering the local configuration
33    path = recover_local_path(command_line)
```

Initialise I

Main

- Reads command line, configuration file
- **Creates a data instance**
- Initializes the cosmological module

```
56          data = Data(command_line, path)
```

Initialise I

Main

- Reads command line, configuration file
- Creates a data instance
- **Initializes the cosmological module**

```
72    # Loading up the cosmological backbone. For the moment, only
          CLASS has been
73    # wrapped.
74    cosmo = recover_cosmological_module(data)
```

Data I

Defining a data class

- **Initialization**

```
35 class Data(object):
36     """
37     Store all relevant data to communicate between the different
           modules.
38
39     """
40
41     def __init__(self, command_line, path):

136         self.cosmo_arguments = {}

145         self.mcmc_parameters = od()
```

Data I

Defining a data class

- Initialization
- **Fill in parameter information**

```
195         # Read from the parameter file to fill properly the
                mcmc_parameters
196         # dictionary.
197         self.fill_mcmc_parameters()
```

Data I

Defining a data class

- Initialization
- Fill in parameter information
- **Log parameter file if needed**

Data II

Defining a data class

- **Initialization of likelihood (dynamical)**

```
338         for elem in self.experiments:

343             # ... import easily the likelihood.py program
344             exec "from likelihoods.%s import %s" % (
345                 elem, elem)

350             exec "self.lkl['%s'] = %s('%s/%s.data',\
351                 self,command_line)" % (
352                 elem, elem, folder, elem)
```

Data II

Defining a data class

- Initialization of likelihood (dynamical)

Why so complicated

No hard coded likelihood! The code does not know the names: **no need to modify the core code to add a new likelihood**

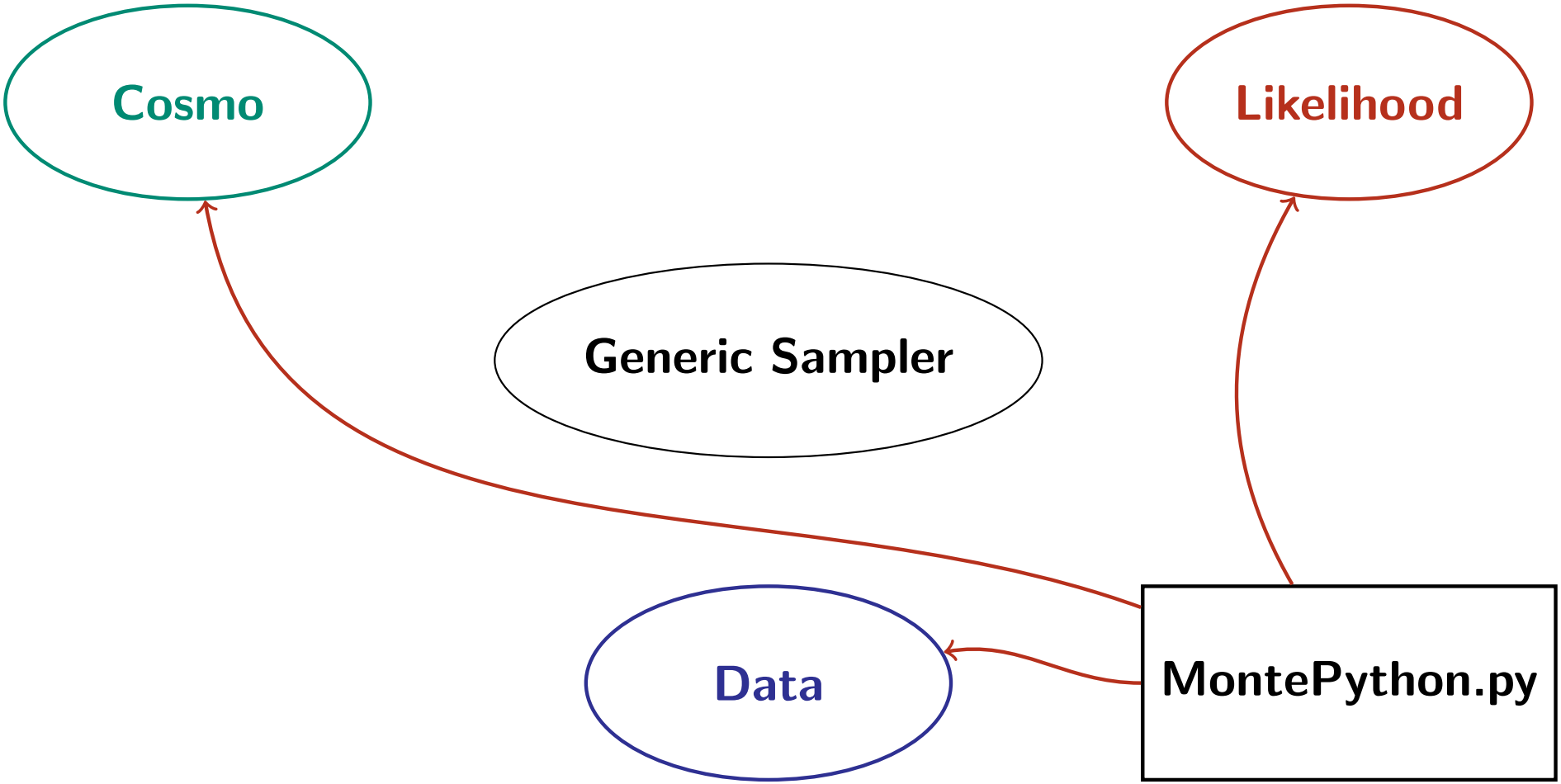
Data III

essential function

`get_mcmc_parameters` returns the list of desired parameters.

- `get_mcmc_parameters(['varying'])`
- `get_mcmc_parameters(['cosmo', 'nuisance'])`
- `get_mcmc_parameters(['cosmo', 'varying'])`

Recap Initialisation



Sampler I

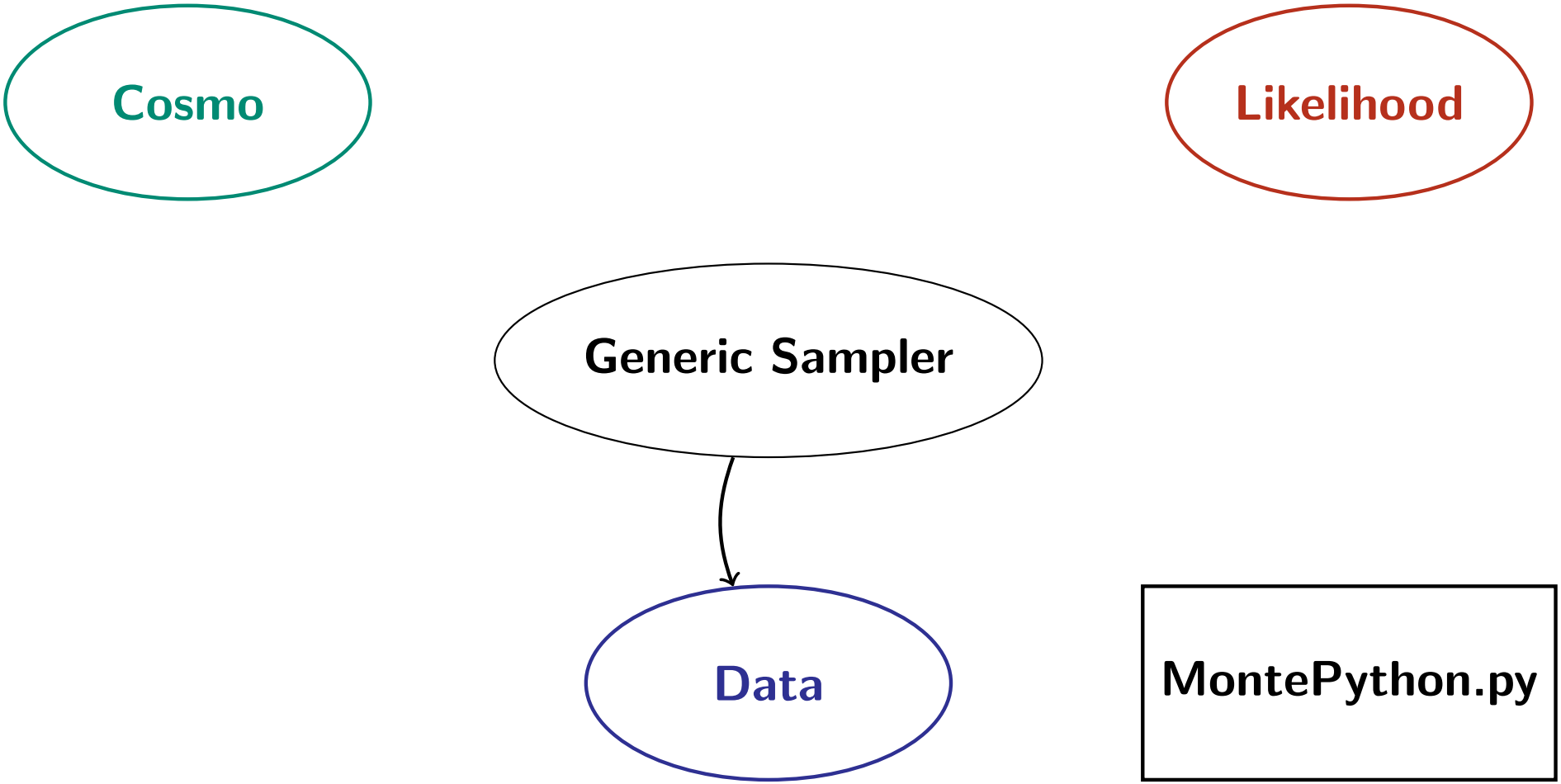
Generic helper functions

- `compute_lkl(cosmo, data)`
- `get_covariance_matrix(data, command_line)`

Role

calling the sampler specified via the **command line**

Choosing a new point



Sampler II

Get new position

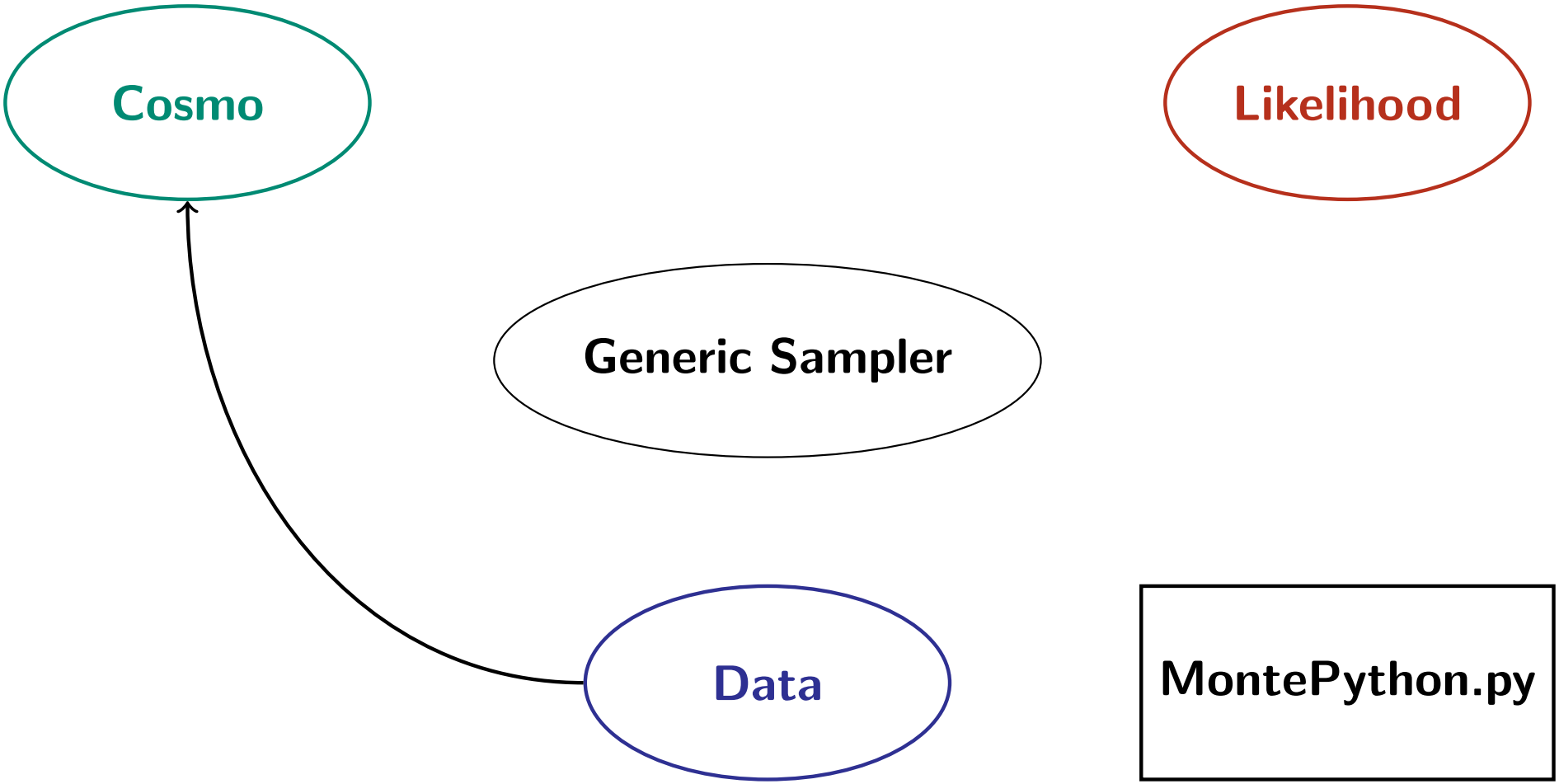
Sampler

How to choose a new point?

- basic eigen-values/vector decomposition
- Cholesky decomposition (Planck) (`-j fast`)
- Nested Sampling with MultiNest (`-m NS`)
- Emcee with Cosmo Hammer (`-m CH`)

Compute Likelihood

Set the cosmo



Sampler III

Compute likelihood

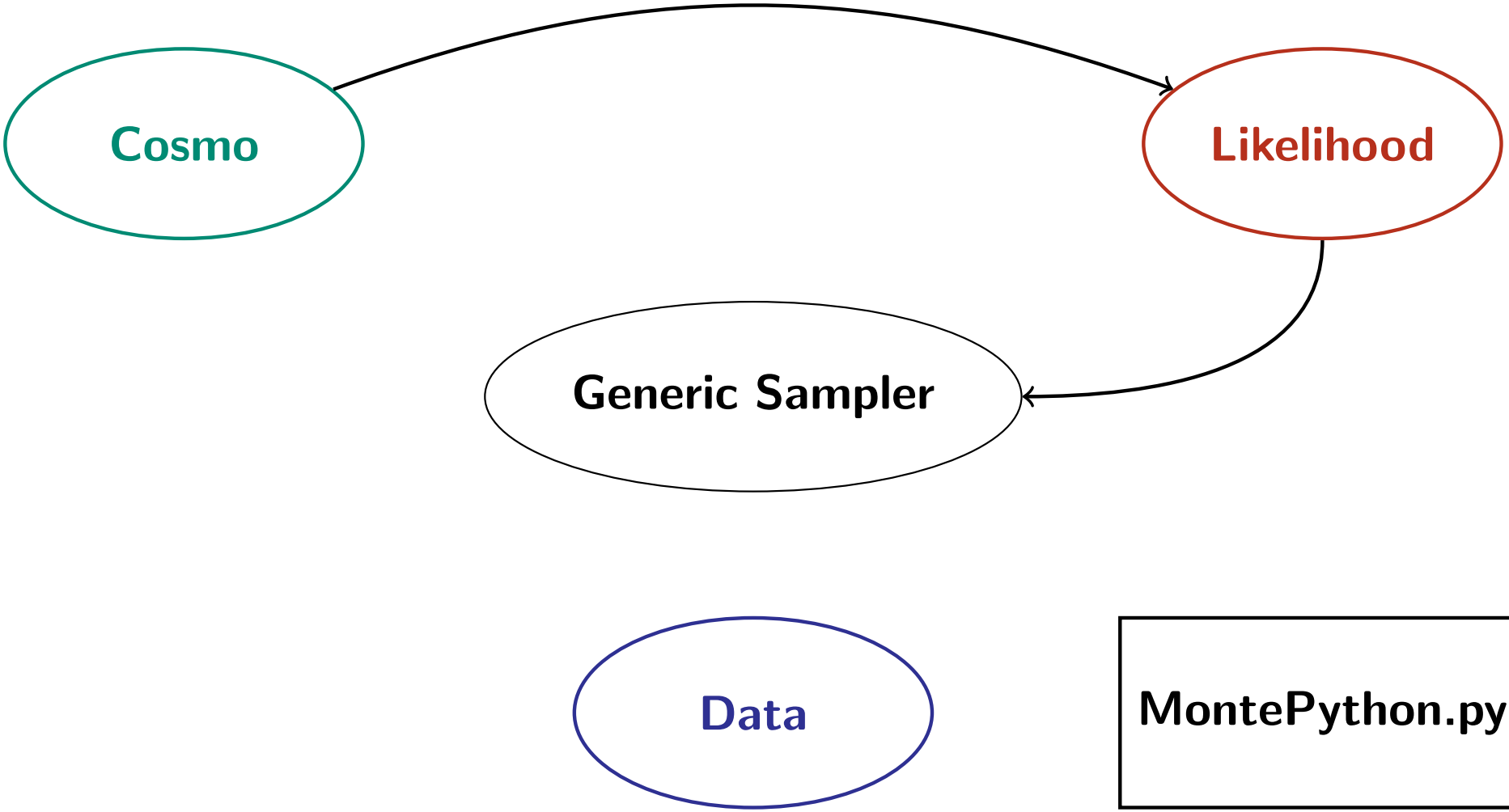
```
334 def compute_lkl(cosmo, data):

370     if ((data.need_cosmo_update) or
371             (not cosmo.state) or
372             (data.jumping_factor == 0)):
373
374         # Prepare the cosmological module with the new set of
                parameters
375         cosmo.set(data.cosmo_arguments)

390         try:
391             cosmo.compute()
392         except CosmoComputationError:
393             return data.boundary_loglike
394         except CosmoSevereError, message:
395             print str(message)
396             raise io_mp.CosmologicalModuleError(
397                 "Something went wrong when calling CLASS")
```

Compute Likelihood

For each likelihood



Sampler III

Compute likelihood

```
404     loglike = 0
```

```
410     for likelihood in data.lkl.itervalues():
411         if likelihood.need_update is True:
412             value = likelihood.loglkl(cosmo, data)
413             # Storing the result
414             likelihood.backup_value = value
415         # Otherwise, take the existing value
416         else:
417             value = likelihood.backup_value
418         loglike += value
```

. . . fiducial . . .

```
446     return loglike
```

Sampler IV

Get the covariance matrix

Main ideas

- **stores values with scale factors on the disk, but uses a matrix without the scale factors for numerical reason**

Sampler IV

Get the covariance matrix

Main ideas

- stores values with scale factors on the disk, but uses a matrix without the scale factors for numerical reason
- **automatic handling of parameters**

Sampler IV

Get the covariance matrix

Main ideas

- stores values with scale factors on the disk, but uses a matrix without the scale factors for numerical reason
- automatic handling of parameters
- **computes eigen vectors, values, and Cholesky decomposition**

Likelihood class

Heavily object oriented

in `likelihood_class.py` are defined:

- the basic `Likelihood` class (parent of all others)

Likelihood class

Heavily object oriented

in `likelihood_class.py` are defined:

- the basic `Likelihood` class (parent of all others)
- `Likelihood_newdat` (standard format)

Likelihood class

Heavily object oriented

in `likelihood_class.py` are defined:

- the basic `Likelihood` class (parent of all others)
- `Likelihood_newdat` (standard format)
- `Likelihood_clik` (Planck, WMAP)

Likelihood class

Heavily object oriented

in `likelihood_class.py` are defined:

- the basic `Likelihood` class (parent of all others)
- `Likelihood_newdat` (standard format)
- `Likelihood_clik` (Planck, WMAP)
- `Likelihood_mpk` (WiggleZ, Euclid)

Likelihoods

Implementation

in the `likelihoods` folder, always the following structure:

- `likelihoods/something/__init__.py` and

Likelihoods

Implementation

in the `likelihoods` folder, always the following structure:

- `likelihoods/something/__init__.py` and
- `likelihoods/something/something.data`

Likelihoods

Implementation

in the `likelihoods` folder, always the following structure:

- `likelihoods/something/__init__.py` and
- `likelihoods/something/something.data`
- alway **inherit** at least from: `Likelihood`