Monte Python

the way back from cosmology to Fundamental Physics

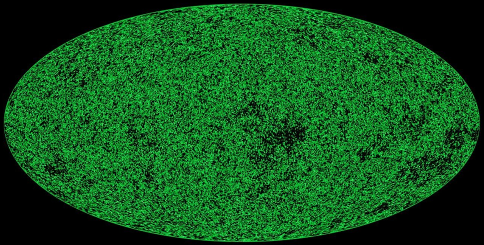
Miguel Zumalacárregui

Nordic Institute for Theoretical Physics and UC Berkeley

IFT School on Cosmology Tools

March 2017

Related Tools

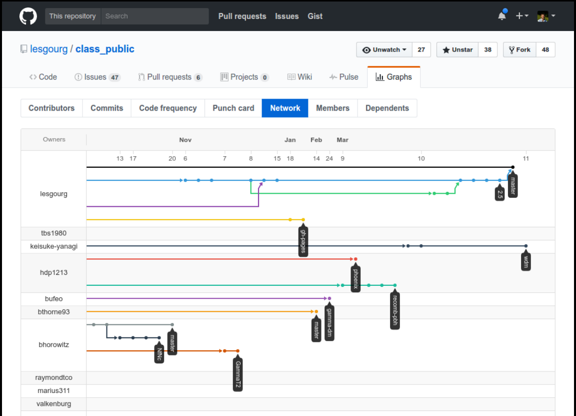


Version control and Python interfacing

PLANCK

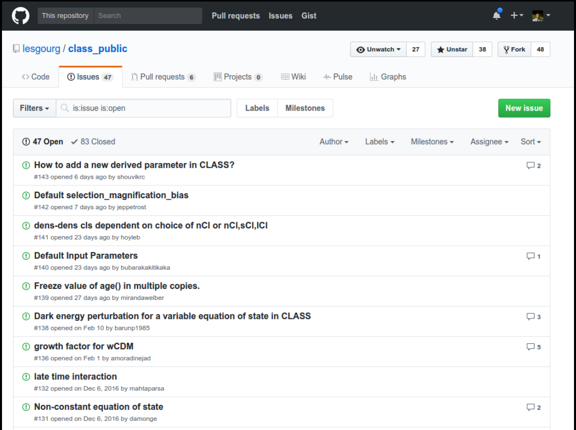
Version control: GIT + github

- GIT: access all previous versions, restore, compare, branch...
- Github: website + interface
  - Network: users, branches intermediate versions



Version control: GIT + github

- GIT: access all previous versions, restore, compare, branch...
- Github: website + interface
  - Network: users, branches intermediate versions
  - Issues: troubleshooting, forum/improvements



Python interfacing with classy

- classy → use CLASS as a Python module
  - Required for MCMC (tomorrow!)
  - Useful for plotting

```
from classy import Class
import numpy as np
import matplotlib.pyplot as plt
cosmo = Class ()
cosmo.set ({'output': 'tCl, pCl, lCl', 'lensing': 'yes'})
cosmo.compute ()
l = np.array ( range (2 ,2501) )
factor = l*(l +1) /(2*np.pi )
lensed_cl = cosmo.lensed_cl (2500)
#then just plot lensed_cl...
```

Python interfacing with `classy`

- `classy` $\rightarrow$ use CLASS as a Python module
  - Required for MCMC (tomorrow!)
  - Useful for plotting
- `IPython` $\rightarrow$ Interactive Python frontend
  - TAB auto-completion

```
miguel@Goedel:~$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from classy import Class

In [2]: c = Class()

In [3]: c.
         c.age                             c.h                      c.Omega_nu
         c.angular_distance                c.Hubble                 c.pars
         c.baryon_temperature              c.ionization_fraction    c.pk
         c.compute                         c.lensed_cl              c.raw_cl
         c.density_cl                      c.luminosity_distance    c.rs_drag
         c.empty                           c.n_s                    c.set
         c.get_background                  c.Neff                   c.set_default
         c.get_current_derived_parameters  c.nonlinear_method       c.sigma8
         c.get_perturbations               c.nonlinear_scale        c.state
         c.get_pk                          c.Omega0_m               c.struct_cleanup
         c.get_primordial                  c.Omega_b                c.T_cmb
         c.get_thermodynamics              c.omega_b                c.z_of_r
         c.get_transfer                    c.Omega_m
```

Python interfacing with `classy`

- `classy` → use CLASS as a Python module
  - Required for MCMC (tomorrow!)
  - Useful for plotting
- `IPython` → Interactive Python frontend
  - TAB auto-completion
- `Jupyter` → Notebook interface (Julia+Pyton+R)

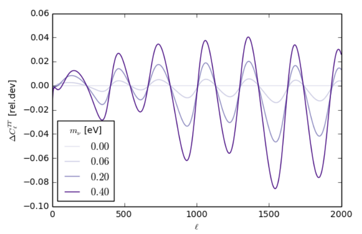
Jupyter example

You can write $\LaTeX$ and awesome equations like $m_\nu \gtrsim 60$ meV. **And plot the effect of m_ν in few lines**

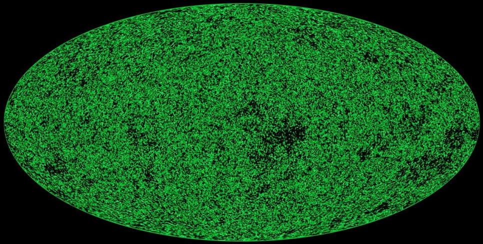
```
cm = plt.get_cmap('Purples')
c = Class()
cl = {} #dictionary for output
for m in [0.0, 0.06, 0.2, 0.4]:
    c.set({'N_ncdm':1, 'N_ur':2.033,'m_ncdm':m,'output':'tCl'})
    c.compute()
    cl[m] = c.raw_cl(2000)
    plt.plot(cl[m]['ell'][2:],cl[m]['tt'][2:]/cl[0]['tt'][2:]-1.,
             color=cm((m+0.1)/0.5),label=r'$%.2f$'%(m))
    c.empty()

plt.xlabel(r'$\ell$')
plt.ylabel(r'$\Delta C_\ell^{TT}$ [rel.dev]')
plt.legend(loc='lower left',fontsize = 12,title= r'$m_\nu$ [eV]')
```

```
<matplotlib.legend.Legend at 0x7fc7b75d5810>
```



Monte Python



from cosmology back to fundamental physics

PLANCK

DISCLAIMER: Short time!

$\lesssim 3h$ course $\Rightarrow$ overview and basic usage

Learn further:

- MontePython slides by Sebastien Clesse ($\ll 1h$?)

  https://lesgourg.github.io/class-tour/16.06.02_Lisbon_intro.pdf

- MontePython course ($\sim 5h$)

  https://lesgourg.github.io/class-tour-Tokyo.html

- Links to extra resources in exercise sheet

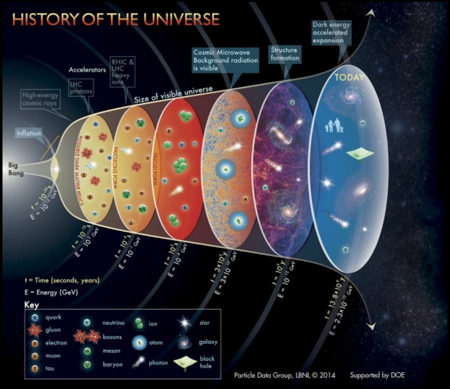
IMPORTANT DISCLAIMER:

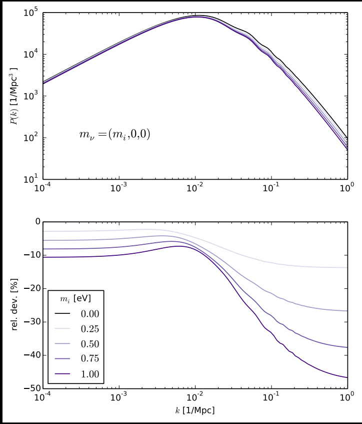
$\downarrow$ I'm mainly a user with little experience developing!

$\uparrow$ Help from experts:

Thejs Brinckmann, Carlos Garcia, Deanna Hooper & Vivian Poulin

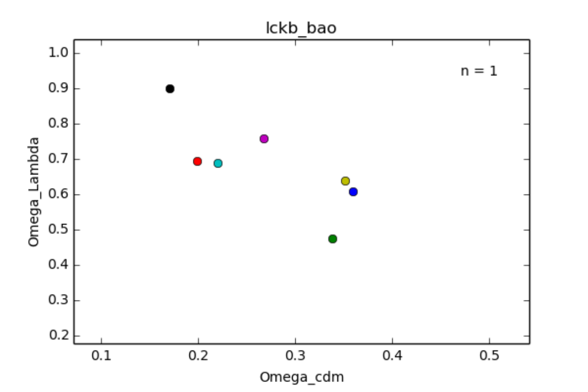
Fundamental physics and cosmology





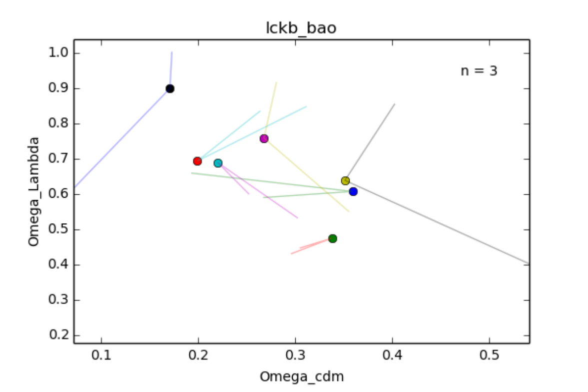
Initial conditions, Dark Matter, Neutrinos, Dark Energy, Gravity...

Scan space of parameters



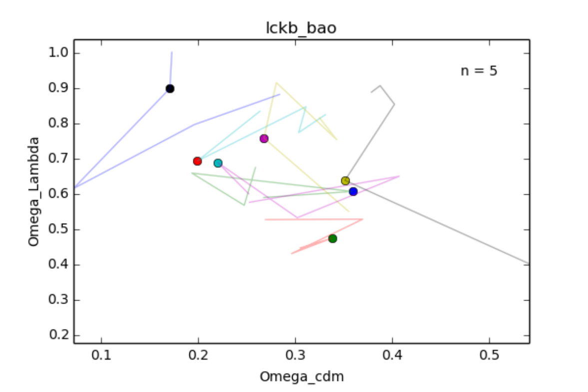
(recall lecture by Will Handley)

Scan space of parameters



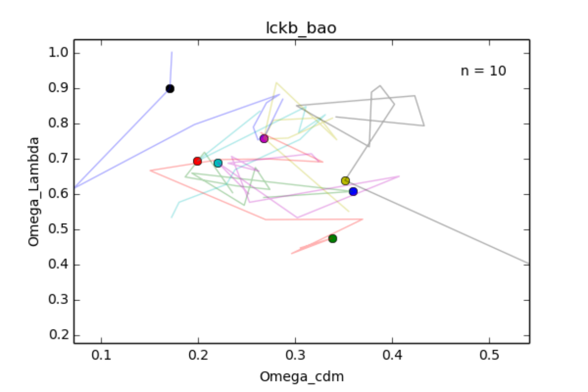
(recall lecture by Will Handley)

Scan space of parameters



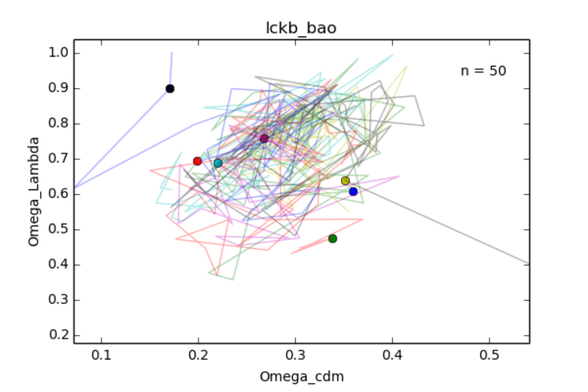
(recall lecture by Will Handley)

Scan space of parameters



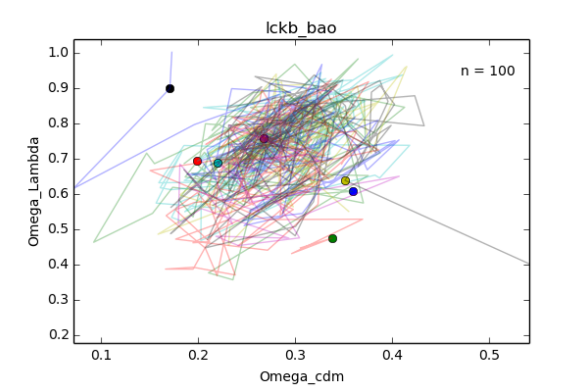
(recall lecture by Will Handley)

Scan space of parameters



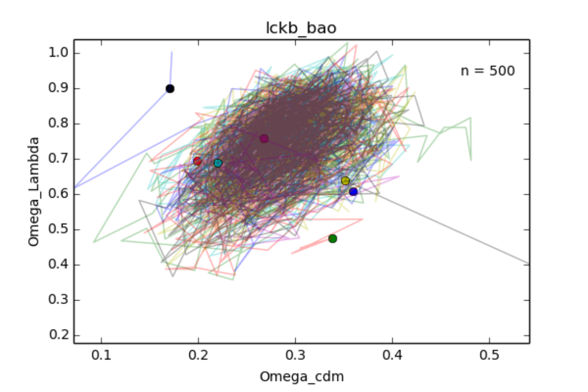
(recall lecture by Will Handley)

Scan space of parameters



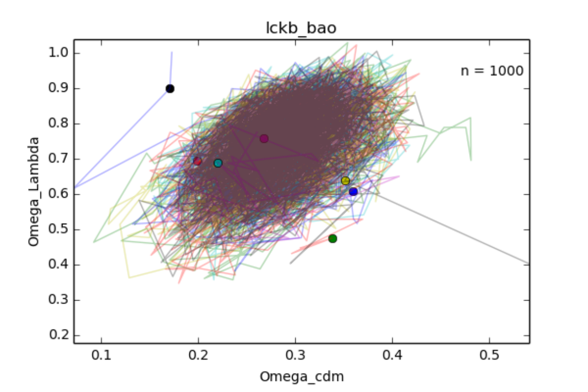
(recall lecture by Will Handley)

Scan space of parameters



(recall lecture by Will Handley)

Scan space of parameters



(recall lecture by Will Handley)

MontePython

An Markov Chain Monte Carlo engine for parameter extraction:

Features

- Written in Python
  - Python is practically magic!
  - imports routines from `numpy` and `scipy`
  - useful outside academia, standard for Big Data

MontePython

An Markov Chain Monte Carlo engine for parameter extraction:

Features

- Written in Python
  - Python is practically magic!
  - imports routines from `numpy` and `scipy`
  - useful outside academia, standard for Big Data
- Uses CLASS through the *classy* wrapper

MontePython

An Markov Chain Monte Carlo engine for parameter extraction:

Features

- Written in Python
  - Python is practically magic!
  - imports routines from `numpy` and `scipy`
  - useful outside academia, standard for Big Data
- Uses CLASS through the *classy* wrapper
- Modular, easy to add
  - likelihoods for new experiments
  - features for sampling, plotting...

MontePython

An Markov Chain Monte Carlo engine for parameter extraction:

Features

- Written in Python
  - Python is practically magic!
  - imports routines from `numpy` and `scipy`
  - useful outside academia, standard for Big Data
- Uses CLASS through the *classy* wrapper
- Modular, easy to add
  - likelihoods for new experiments
  - features for sampling, plotting...
- Easy to use, intensively documented

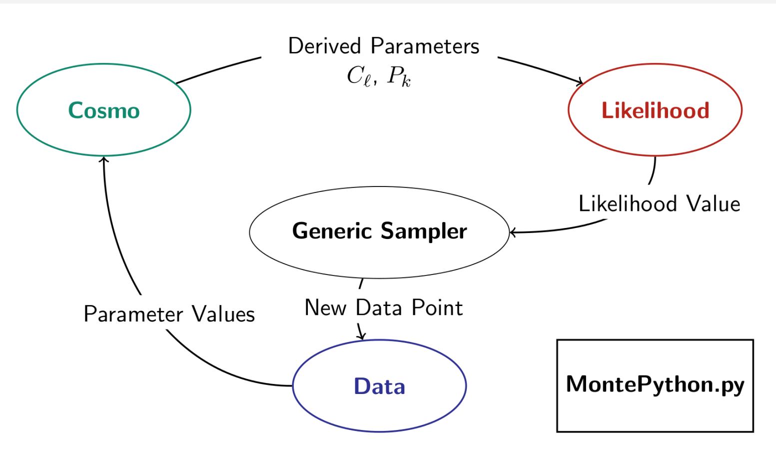
MontePython

An Markov Chain Monte Carlo engine for parameter extraction:

Features

- Written in Python
  - Python is practically magic!
  - imports routines from `numpy` and `scipy`
  - useful outside academia, standard for Big Data
- Uses CLASS through the *classy* wrapper
- Modular, easy to add
  - likelihoods for new experiments
  - features for sampling, plotting...
- Easy to use, intensively documented
- Parallelization is optional
  - simpler to install
  - runs in old/separate computers, short queue

Modular Structure



(from B. Audren's Monte Python slides)

Defining a run (model.param)

- List of experiments

```
data.experiments=['Planck_highl','Planck_lowl','Planck_lensing']
```

Collected in montepython/likelihoods:

```
miguel@Goedel:~/code/montepython_zuma/montepython/likelihoods$ ls
acbar                       clik_wmap_full          __init__.py          quad
bao                         clik_wmap_lowl          __init__.pyc         sdss_lrgDR4
bao_boss                    cosmic_clocks_BC03      JLA                  sn
bao_boss_aniso              cosmic_clocks_BC03_all  JLA_simple           spt
bao_boss_aniso_gauss_approx cosmic_clocks_MaStro    lowlike              spt_2500
bao_known_rs                da_rec                  Planck_actspt        test_gaussian
bicep                       euclid_lensing          Planck_highl         test_nuisance1
bicep2                      euclid_pk               Planck_highl_lite    test_nuisance2
boomerang                   fake_desi               Planck_highl_TTTEEE  timedelay
cbi                         fake_planck_bluebook    Planck_lensing       WiggleZ
CFHTLens                    gunn_peterson           Planck_lowl          WiggleZ_bao
CFHTLens_correlation        hst                     Planck_SZ            wmap
clik_fake_planck            igm_temperature         polarbear            wmap_9yr
```

Defining a run (model.param)

- List of experiments

```
data.experiments=['Planck_highl','Planck_lowl','Planck_lensing']
```

- Cosmological parameters

```
                      #  [mean, (bounds) , SIGMA,scale, type ]
data.parameters['n_s']=[0.96, None,None, 0.008, 1 , 'cosmo']
```

Defining a run (`model.param`)

- List of experiments

```
data.experiments=['Planck_highl','Planck_lowl','Planck_lensing']
```

- Cosmological parameters

```
                        #  [mean,  (bounds)  , SIGMA,scale, type ]
data.parameters['n_s']=[0.96, None,None, 0.008, 1 , 'cosmo']
```

Fixed values:

- set SIGMA $= 0$
- `data.cosmo_arguments['N_ncdm'] = 1`

Defining a run (`model.param`)

- List of experiments

```
data.experiments=['Planck_highl','Planck_lowl','Planck_lensing']
```

- Cosmological parameters

```
                         #  [mean,  (bounds) , SIGMA,scale, type ]
data.parameters['n_s']=[0.96, None,None, 0.008, 1 , 'cosmo']
```

Fixed values:

- set SIGMA $= 0$
- `data.cosmo_arguments['N_ncdm'] = 1`

- Derived and Nuisance parameters

```
data.parameters['sigma8'] = [0, None, None, 0, 1, 'derived']
data.parameters['A_cib_217'] = [61,0,200,7,1,'nuisance']
```

Defining a run (model.param)

- List of experiments

```
data.experiments=['Planck_highl','Planck_lowl','Planck_lensing']
```

- Cosmological parameters

```
                        #  [mean, (bounds) , SIGMA,scale, type ]
data.parameters['n_s']=[0.96, None,None, 0.008, 1 , 'cosmo']
```

Fixed values:

- set SIGMA = 0
- `data.cosmo_arguments['N_ncdm'] = 1`

- Derived and Nuisance parameters

```
data.parameters['sigma8'] = [0, None, None, 0, 1, 'derived']
data.parameters['A_cib_217'] = [61,0,200,7,1,'nuisance']
```

- MCMC parameters: `data.N=10`, `data.write_step=5` ...

Running Monte Python (run)

Single chain:

```
python montepython/MontePython.py run \
       -p model.param \
       -o output_directory  (...)
```

Parallel run (4 chains):

```
mpirun -nc 4 python (...)
```

Some options:

```
 /!\ Appending to an existing folder: using the log.param instead of
     input/lambda_bao_b.param
Running Monte Python v2.2.2

with CLASS v2.4.5

Testing likelihoods for:
 -> bao_boss, bao_boss_aniso

 /!\ excluding isotropic CMASS measurement
Creating chains/lcdmk_bao_b/2017-03-12_100__2.txt

Creating error file chains/lcdmk_bao_b/error_log.txt


Deduced starting covariance matrix:
['omega_b', 'Omega_cdm', 'Omega_k']
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]

#  -LogLkl      1e+02omega_b    Omega_cdm       Omega_k         Omega_Lambda
3  11.9558      2.221160e+00    3.132032e-01    -1.692095e-02   6.553103e-01
1  12.7946      2.262336e+00    3.271086e-01    2.742679e-02    5.961614e-01
1  12.0403      2.256136e+00    2.494162e-01    1.271974e-01    5.742182e-01
3  11.6619      2.286805e+00    2.308487e-01    1.014554e-01    6.178604e-01
2  11.4992      2.275734e+00    2.108346e-01    5.439644e-02    6.851744e-01
1  11.1477      2.309970e+00    2.013242e-01    3.477642e-02    7.135600e-01
3  7.83276      2.297757e+00    2.751741e-01    9.733703e-02    5.774153e-01
14  6.96653     2.302347e+00    2.748664e-01    5.921469e-02    6.157454e-01
4  6.46121      2.372939e+00    2.523288e-01    2.276382e-02    6.731983e-01
13  6.9573      2.334726e+00    2.363992e-01    -3.071722e-02   7.434402e-01
4  8.883        2.350200e+00    2.572473e-01    -3.956319e-02   7.311015e-01
14  7.19446     2.380873e+00    2.210523e-01    -2.615132e-02   7.532173e-01
2  6.95177      2.359364e+00    2.629244e-01    7.590971e-02    6.097521e-01
6  7.18606      2.339764e+00    2.595452e-01    8.159985e-02    6.078676e-01
1  6.44771      2.353985e+00    2.392110e-01    3.775342e-02    6.717389e-01
4  8.22755      2.328585e+00    2.075475e-01    -2.491768e-03   7.442001e-01
1  7.89432      2.390601e+00    2.076928e-01    -2.671562e-02   7.669296e-01
6  7.21255      2.388188e+00    2.145373e-01    7.295908e-03    7.261260e-01
2  7.07413      2.397083e+00    2.685047e-01    6.233775e-02    6.169233e-01
1  7.37498      2.406336e+00    2.207678e-01    4.592705e-02    6.808696e-01
2  10.6892      2.439808e+00    1.946383e-01    5.075931e-02    7.014388e-01
3  11.7196      2.476394e+00    2.298767e-01    1.535277e-01    5.626362e-01
1  12.3995      2.441832e+00    3.239238e-01    2.005879e-01    4.222807e-01
3  10.2062      2.420720e+00    2.880782e-01    1.751385e-01    4.840349e-01
1  7.3809       2.439488e+00    2.699004e-01    7.482511e-02    6.021178e-01
3  7.25137      2.437977e+00    2.578430e-01    2.764030e-02    6.613929e-01
1  6.83719      2.459588e+00    2.214344e-01    4.100809e-02    6.839635e-01

#  100 steps done, acceptance rate: 0.26
```

Running Monte Python (run)

Single chain:

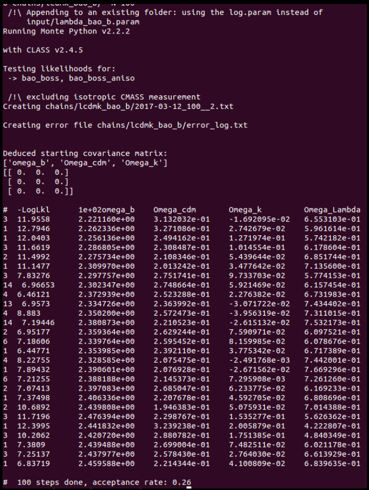
```
python montepython/MontePython.py run \
       -p model.param \
       -o output_directory  (...)
```

Parallel run (4 chains):

```
 mpirun -nc 4 python (...)
```

Some options:

- `-N` → # points
- `-C` → covariance matrix
- `-r` → restart from last point of chain
- `--update` → update sampling + covariance



```
 /!\ Appending to an existing folder: using the log.param instead of
     input/lambda_bao_b.param
Running Monte Python v2.2.2

with CLASS v2.4.5

Testing likelihoods for:
 -> bao_boss, bao_boss_aniso

 /!\ excluding isotropic CMASS measurement
Creating chains/lcdmk_bao_b/2017-03-12_100__2.txt

Creating error file chains/lcdmk_bao_b/error_log.txt


Deduced starting covariance matrix:
['omega_b', 'Omega_cdm', 'Omega_k']
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]

#  -LogLkl      1e+02omega_b    Omega_cdm       Omega_k         Omega_Lambda
3  11.9558      2.221160e+00    3.132032e-01    -1.692095e-02   6.553103e-01
1  12.7946      2.262336e+00    3.271086e-01    2.742679e-02    5.961614e-01
1  12.0403      2.256136e+00    2.494162e-01    1.271974e-01    5.742182e-01
3  11.6619      2.286805e+00    2.308487e-01    1.014554e-01    6.178604e-01
2  11.4992      2.275734e+00    2.108346e-01    5.439644e-02    6.851744e-01
1  11.1477      2.309970e+00    2.013242e-01    3.477642e-02    7.135600e-01
3  7.83276      2.297757e+00    2.751741e-01    9.733703e-02    5.774153e-01
14  6.96653     2.302347e+00    2.748664e-01    5.921469e-02    6.157454e-01
4  6.46121      2.372939e+00    2.523288e-01    2.276382e-02    6.731983e-01
13  6.9573      2.334726e+00    2.363992e-01    -3.071722e-02   7.434402e-01
4  8.883        2.350200e+00    2.572473e-01    -3.956319e-02   7.311015e-01
14  7.19446     2.380873e+00    2.210523e-01    -2.615132e-02   7.532173e-01
2  6.95177      2.359364e+00    2.629244e-01    7.590971e-02    6.097521e-01
6  7.18606      2.339764e+00    2.595452e-01    8.159985e-02    6.078676e-01
1  6.44771      2.353985e+00    2.392110e-01    3.775342e-02    6.717389e-01
4  8.22755      2.328585e+00    2.075475e-01    -2.491768e-03   7.442001e-01
1  7.89432      2.390601e+00    2.076928e-01    -2.671562e-02   7.669296e-01
6  7.21255      2.388188e+00    2.145373e-01    7.295908e-03    7.261260e-01
2  7.07413      2.397083e+00    2.685047e-01    6.233775e-02    6.169233e-01
1  7.37498      2.406336e+00    2.207678e-01    4.592705e-02    6.808696e-01
2  10.6892      2.439808e+00    1.946383e-01    5.075931e-02    7.014388e-01
3  11.7196      2.476394e+00    2.298767e-01    1.535277e-01    5.626362e-01
1  12.3995      2.441832e+00    3.239238e-01    2.005879e-01    4.222807e-01
3  10.2062      2.420720e+00    2.880782e-01    1.751385e-01    4.840349e-01
1  7.3809       2.439488e+00    2.699004e-01    7.482511e-02    6.021178e-01
3  7.25137      2.437977e+00    2.578430e-01    2.764030e-02    6.613929e-01
1  6.83719      2.459588e+00    2.214344e-01    4.100809e-02    6.839635e-01

#  100 steps done, acceptance rate: 0.26
```

All options explained

```
python montepython/MontePython.py info --help
```

Analyzing results (`info`)

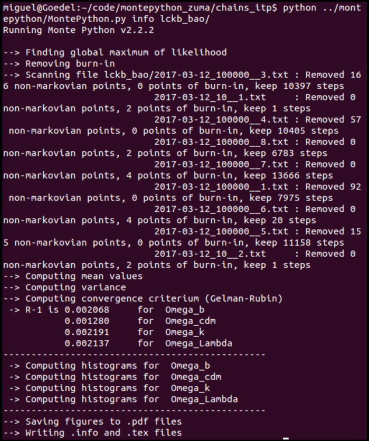
Single model/experiment:

```
python montepython/MontePython.py info \
       output_directory  (...)
```

Comparing several runs:

```
python montepython/MontePython.py info \
       output_1 output_2 output_3 (...)
```

Configuring the output/analysis



Analyzing results (info)

Single model/experiment:

```
python montepython/MontePython.py info \
       output_directory  (...)
```

Comparing several runs:

```
python montepython/MontePython.py info \
       output_1 output_2 output_3 (...)
```

Configuring the output/analysis

- --extra → file with plot options
- --bins → # bins for posterior
- --all → plot every subplot separately
- --no-mean → only marginalized in 1D

```
miguel@Goedel:~/code/montepython_zuma/chains_itp$ python ../mont
epython/MontePython.py info lckb_bao/
Running Monte Python v2.2.2

--> Finding global maximum of likelihood
--> Removing burn-in
--> Scanning file lckb_bao/2017-03-12_100000__3.txt : Removed 16
6 non-markovian points, 0 points of burn-in, keep 10397 steps
                           2017-03-12_10__1.txt     : Removed 0
non-markovian points, 2 points of burn-in, keep 1 steps
                           2017-03-12_100000__4.txt : Removed 57
 non-markovian points, 0 points of burn-in, keep 10405 steps
                           2017-03-12_100000__8.txt : Removed 0
non-markovian points, 2 points of burn-in, keep 6783 steps
                           2017-03-12_100000__7.txt : Removed 0
non-markovian points, 4 points of burn-in, keep 13666 steps
                           2017-03-12_100000__1.txt : Removed 92
 non-markovian points, 0 points of burn-in, keep 7975 steps
                           2017-03-12_100000__6.txt : Removed 0
non-markovian points, 4 points of burn-in, keep 20 steps
                           2017-03-12_100000__5.txt : Removed 15
5 non-markovian points, 0 points of burn-in, keep 11158 steps
                           2017-03-12_10__2.txt     : Removed 0
non-markovian points, 2 points of burn-in, keep 1 steps
--> Computing mean values
--> Computing variance
--> Computing convergence criterium (Gelman-Rubin)
 -> R-1 is 0.002068     for  Omega_b
           0.001280     for  Omega_cdm
           0.002191     for  Omega_k
           0.002137     for  Omega_Lambda
----------------------------------------------
 -> Computing histograms for  Omega_b
 -> Computing histograms for  Omega_cdm
 -> Computing histograms for  Omega_k
 -> Computing histograms for  Omega_Lambda
----------------------------------------------
--> Saving figures to .pdf files
--> Writing .info and .tex files
```

All options explained

```
python montepython/MontePython.py info --help
```

A very minimal run

Write `lckb.param`:

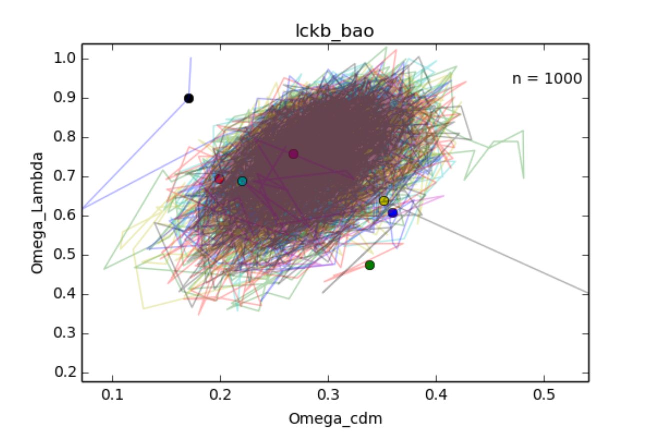
```
data.experiments=['bao_boss','bao_boss_aniso']
#Cosmo parameteress          [mean, min, max, sigma, scale, type]
data.parameters['Omega_b'] = [0.045,0.01, None,0.01,1,'cosmo']
data.parameters['Omega_cdm'] = [0.3, 0, None, 0.1, 1, 'cosmo']
data.parameters['Omega_k'] = [0.0, -0.5, 0.5, 0.1, 1, 'cosmo']
#Fixed parameters (sigma = 0)
data.parameters['H0'] = [67.8, None, None, 0, 1, 'cosmo']
data.cosmo_arguments['YHe'] = 0.24
#derived parameters
data.parameters['Omega_Lambda'] = [1,None,None,0,1,'derived']
#mcmc parameters
data.N=10
data.write_step=5
```

Run ~ 7 chains with

```
python montepython/MontePython.py run -o chains/lckb_bao \
                     -p lckb_param --update 300 -N 100000
```

A very minimal run

The 7 chains explore the parameter space



Chains named `yyyy-mm-dd_N_n.txt (date_points_id)`

A very minimal run

Analyze:

```
python montepython/MontePython.py info chains/lckb_bao
```

- lckb_bao.tex → table with MCMC results

Param	best-fit	mean$\pm\sigma$	95% lower	95% upper
Ω_b	0.03595	$0.03977^{+0.0095}_{-0.015}$	0.01662	0.06547
Ω_{cdm}	0.2931	$0.2872^{+0.049}_{-0.048}$	0.1892	0.3847
Ω_k	-0.1183	$-0.08087^{+0.11}_{-0.14}$	-0.3182	0.1755
Ω_Λ	0.7891	$0.7538^{+0.12}_{-0.091}$	0.542	0.9564

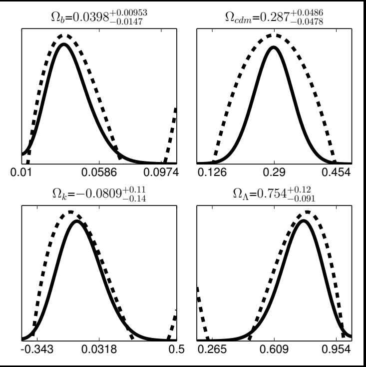
$-\ln\mathcal{L}_{\min} = 5.57269$, minimum $\chi^2 = 11.15$

- `lckb_bao.covmat` → covariance matrix
- `lckb_bao.bestfit` → best fit values

→ arguments for another run

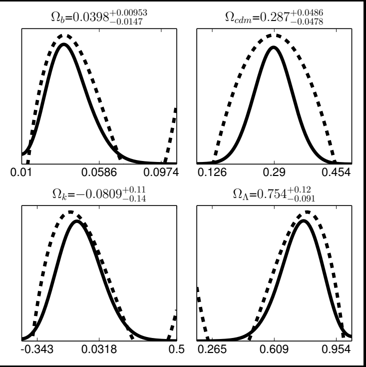
A very minimal run

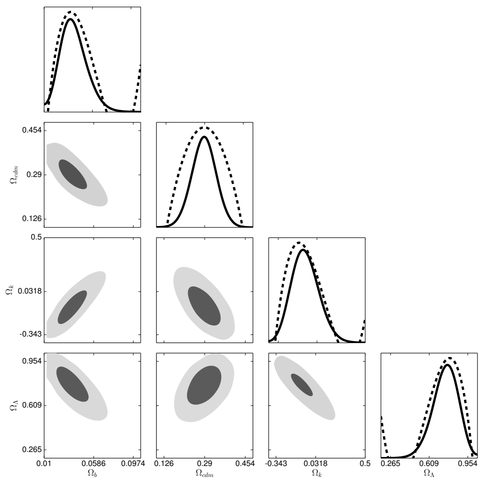
In `lckb_bao/plots`:



A very minimal run

In `lckb_bao/plots`:



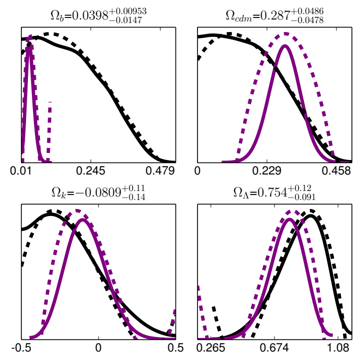


Comparing several runs

Run chains `lckb_sne` with `data.experiments=['sne']`

Analyze: `python ... info chains/lckb_sne chains/lckb_bao`

In `lckb_sne/plots`:

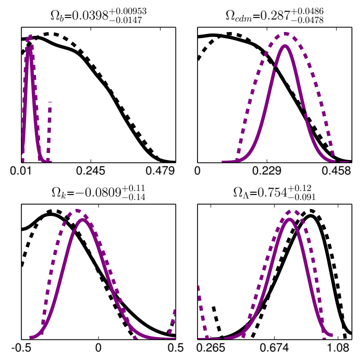


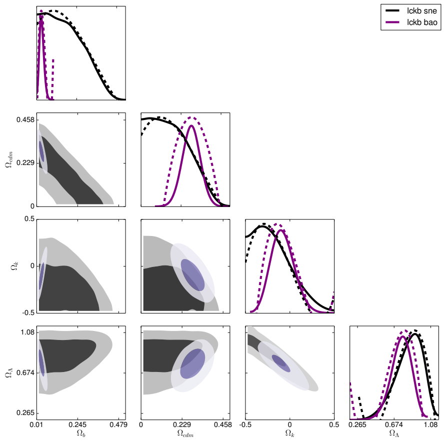
Comparing several runs

Run chains `lckb_sne` with `data.experiments=['sne']`

Analyze: `python ... info chains/lckb_sne chains/lckb_bao`

In `lckb_sne/plots`:





Experiments and Likelihoods

Simplest example ever: Prior on H_0 (1103.2976)

The Hubble Space Telescope measured $h_{obs} = 0.738 \pm 0.024$

$$\log(\mathcal{L}) = -\frac{1}{2}\frac{(h_{th} - h_{obs})^2}{\sigma_h^2}$$

Likelihood (`montepython/likelihoods/hst/__init__`):

```
from montepython.likelihood_class import Likelihood_prior
class hst(Likelihood_prior):
    def loglkl(self, cosmo, data):
        h = cosmo.h()
        loglkl = -0.5 * (h - self.h) ** 2 / (self.sigma ** 2)
        return loglkl
```

Data (`montepython/likelihoods/hst/hst.data`):

```
# Values for Hubble Space Telescope (following 1103.2976)
hst.h     = 0.738
hst.sigma = 0.024
```

Likelihood rules

- Likelihoods in directory `montepython/likelihoods/l_name`
- Needed files: `__init__.py` and `l_name.data`
- `__init__.py` defines a class, inheriting from `Likelihood`
- Contains function `loglkl` $\rightarrow \log(\mathcal{L})$

Introducing your own Likelihoods

- Follow the above rules
- Inspire yourself with the examples
- $\exists$ similar likelihood? $\rightarrow$ you can inherit its methods!
- You can use additional python packages

(See also B. Audren's lecture on likelihoods)

Conclusions

- Brings all the power of CLASS to Python
- Easy to run chains and analyze likelihoods
- Many available experiments
- Advantages from object oriented features in python
  - Add likelihoods
  - Add samplers or other features
- This just scratches the surface, many more options!

(See also B. Audren's slides)

The hi_class academy

Coming soon!

- Set of interrelated projects:
  - ⋆ Theory & model building
  - ⋆ Implementation and phenomenology
  - ⋆ Compare with data
- Collaboration → Publishable results
  - ⋆ Review of models
  - ⋆ Observational constaints
- Stay tuned for more info!

www.hiclass-code.net