

# Basic structure of Monte Python

Benjamin Audren

École Polytechnique Fédérale de Lausanne

01/04/2014

- 1 Code Structure
- 2 Important Modules in (some) detail
- 3 Usage
- 4 Practice with Monte Python

# Outline

- 1 Code Structure
  - Wrapper
  - Flow
- 2 Important Modules in (some) detail
- 3 Usage
- 4 Practice with Monte Python

# classy - wrapper around CLASS

## Cython

- language to **interface** Python with C
- can be used to **speed up** a bottleneck in Python or...
- to wrap an **existing C code** and call it from Python as a normal function

# classy - wrapper around CLASS

## Cython

- language to **interface** Python with C
- can be used to **speed up** a bottleneck in Python or...
- to wrap an **existing C code** and call it from Python as a normal function

## Classy

- classy.pyx **wraps** CLASS modules and some functions (only the ones needed from within Monte Python).
- uses a **dictionary** in place of the **.ini file** to give parameters (file content)
- if no arguments are specified, CLASS **default values** will be used.

# classy - wrapper around CLASS

## Cython

- language to **interface** Python with C
- can be used to **speed up** a bottleneck in Python or ...
- to wrap an **existing C code** and call it from Python as a normal function

## Classy

- `classy.pyx` **wraps** CLASS modules and some functions (only the ones needed from within Monte Python).
- uses a **dictionary** in place of the **.ini file** to give parameters (file content)
- if no arguments are specified, CLASS **default values** will be used.

## Last word

You (most probably) don't need to know how `classy.pyx` is written. The only important thing is **its interface**.

# General structure of the modules

## Files

### Input Files

- configuration file: **path to codes on machine**
- parameter file: **parameters, prior range, proposal, experiments**
- opt** covariance matrix, bestfit file

### Ouptut

- a folder: **stores every information concerning the run**
- a chain per run: **Markov Chain of a given length**

# General structure of the modules

A drawing

**Cosmo**

**Likelihood**

**Generic Sampler**

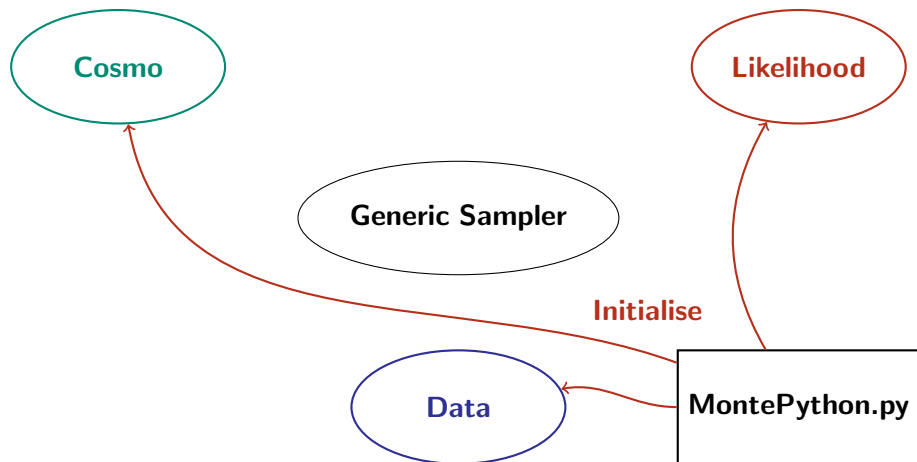
**Data**

**MontePython.py**



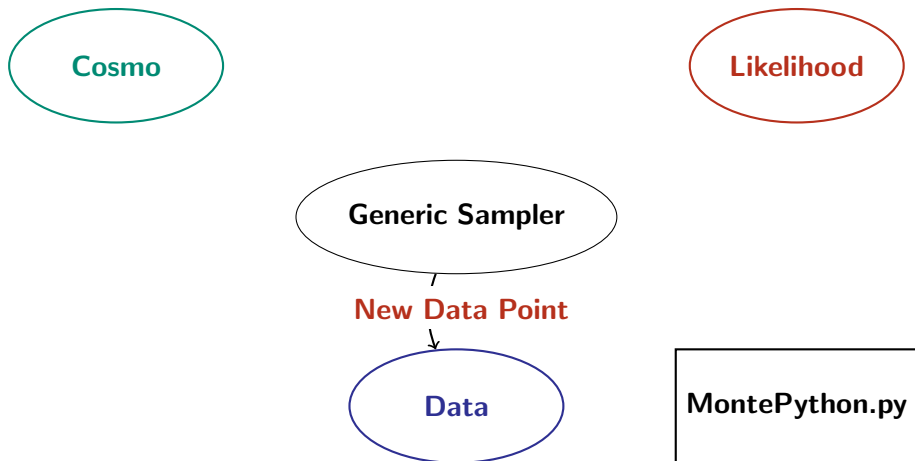
# General structure of the modules

A drawing



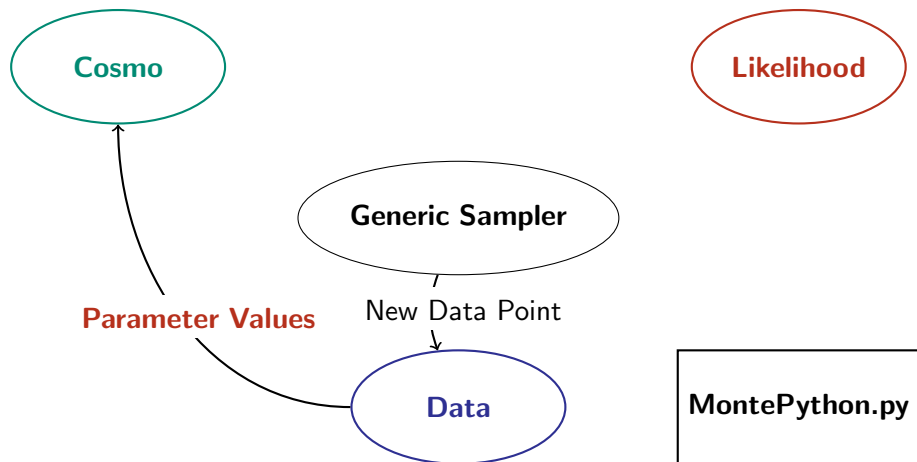
# General structure of the modules

A drawing



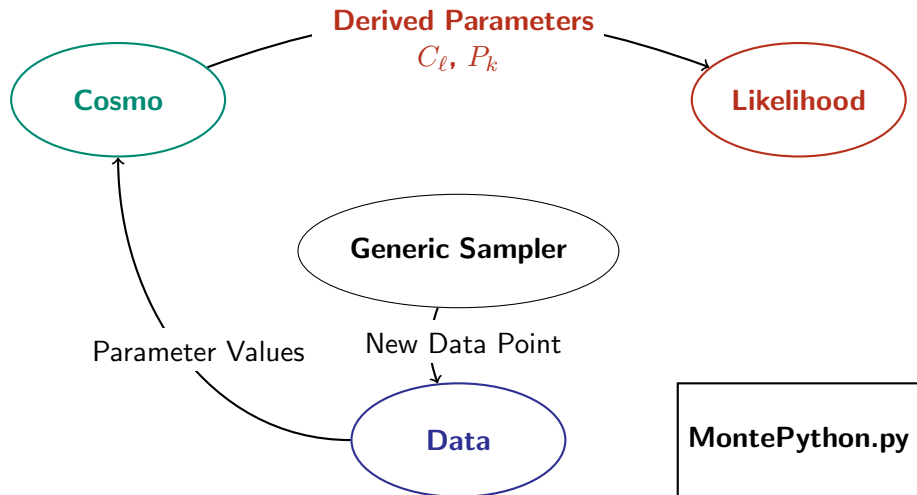
# General structure of the modules

A drawing



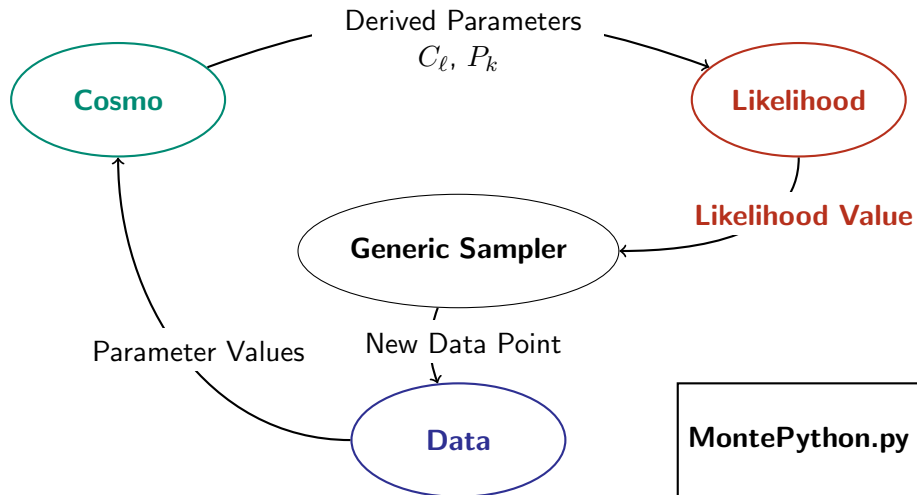
# General structure of the modules

A drawing



# General structure of the modules

A drawing



# Information containers

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

## instances

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



# General structure of the modules

## Main Modules

- **MontePython** Simple script launching the code
- **run call initialise, and launch a sampler session**

# 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**

# 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**

# 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**

# 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**

# 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**

# 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**



# Outline

- 1 Code Structure
- 2 Important Modules in (some) detail
  - MontePython.py
  - Initialise.py
  - data.py
  - sampler.py
  - likelihood class
  - analyze.py
- 3 Usage
- 4 Practice with Monte Python

# Note on the documentation

## Note on the documentation

### Use it!

Monte Python uses an automatic documentation tool, *sphinx*, that generates a website automatically. <http://baudren.web.cern.ch/baudren/documentation/index.html>

# Note on the documentation

## Use it!

Monte Python uses an automatic documentation tool, *sphinx*, that generates a website automatically. <http://baudren.web.cern.ch/baudren/documentation/index.html>

## Use the forum!

Go to the issues page

[https://github.com/baudren/montepython\\_public/issues](https://github.com/baudren/montepython_public/issues), to require a feature, report a bug.

# Note on the documentation

## Use it!

Monte Python uses an automatic documentation tool, *sphinx*, that generates a website automatically. <http://baudren.web.cern.ch/baudren/documentation/index.html>

## Use the forum!

Go to the issues page

[https://github.com/baudren/montepython\\_public/issues](https://github.com/baudren/montepython_public/issues), to require a feature, report a bug.

## Use the wiki!

[https://github.com/baudren/montepython\\_public/wiki](https://github.com/baudren/montepython_public/wiki), or [https://github.com/lesgourg/class\\_public/wiki](https://github.com/lesgourg/class_public/wiki) for classy business.

# MontePython.py

## Role

Convenience script that calls the Monte Python run function.

# Initialise I

## Main

- Reads command line, configuration file

```
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**

```
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
38     modules.
39     """
40
41     def __init__(self, command_line, path):
42
43
44
136         self.cosmo_arguments = {}
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

# 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**

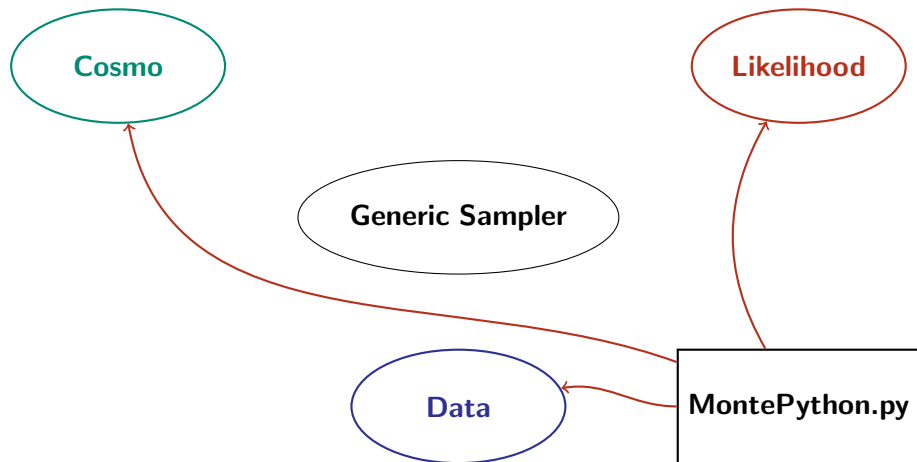
# Data III

## 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)`

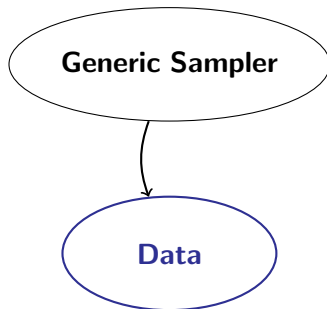
## Role

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

# Choosing a new point

**Cosmo**

**Likelihood**



**MontePython.py**

# Sampler II

Get new position

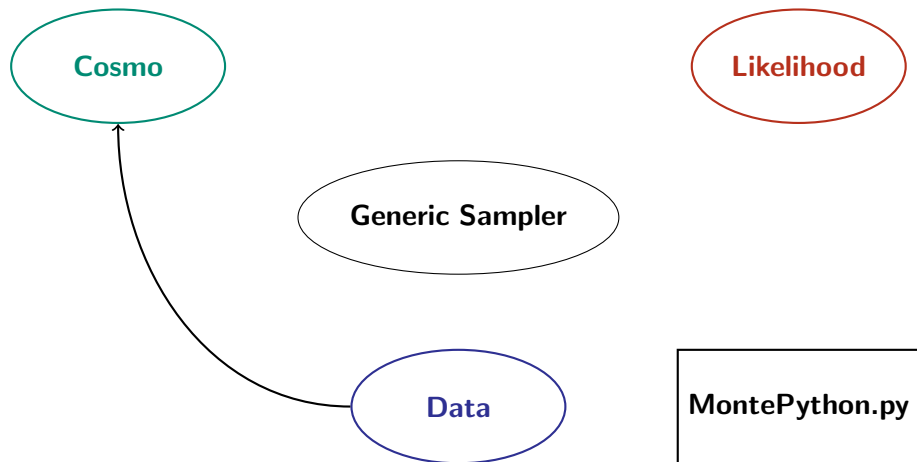
## Sampler

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



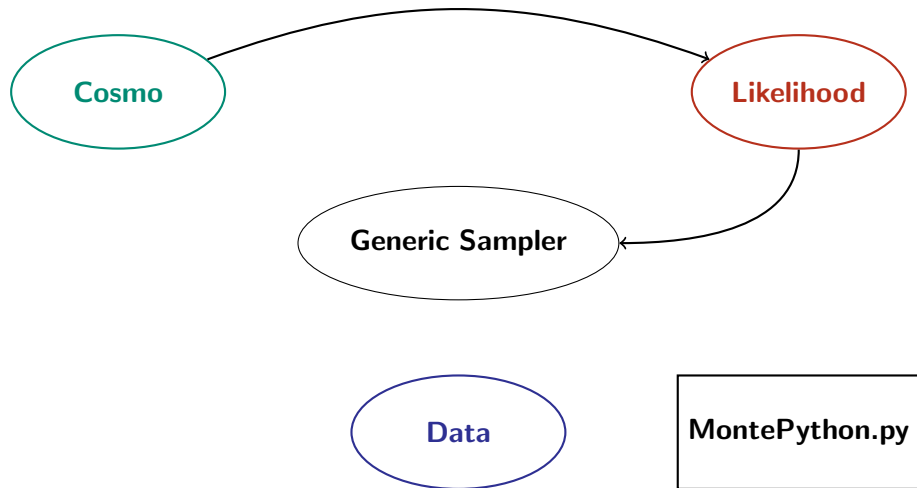
# Sampler III

## 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  
375         parameters  
        cosmo.set(data.cosmo_arguments)  
  
390     try:  
391         cosmo.compute(["lensing"])  
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.intervals():
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 without scale factors for numerical reason**



# Sampler IV

Get the covariance matrix

## Main ideas

- stores values without scale factors for numerical reason
- **automatic handling of parameters**

# Sampler IV

Get the covariance matrix

## Main ideas

- stores values without 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`
- always **inherit** at least from: `Likelihood`

# Analyze

## Convergence Computation

Gelman-Rubin Diagnostic:

variance between chains = variance within chains

This gives a number  $R$ , that must be  $< 0.01$ .

Beware of this number computed for a single file!

# Analyze

## Convergence Computation

Gelman-Rubin Diagnostic:

variance between chains = variance within chains

This gives a number  $R$ , that must be  $< 0.01$ .

Beware of this number computed for a single file!

## Plotting

pdf output (or png): triangle plot and 1-dimensional  
**marginalized posterior** and **Mean likelihood** (visual  
indication of convergence)

# Analyze

## Convergence Computation

Gelman-Rubin Diagnostic:

variance between chains = variance within chains

This gives a number  $R$ , that must be  $< 0.01$ .

Beware of this number computed for a single file!

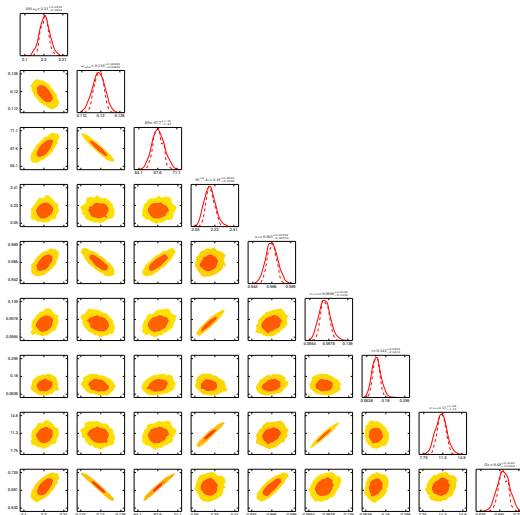
## Plotting

pdf output (or png): triangle plot and 1-dimensional **marginalized posterior** and **Mean likelihood** (visual indication of convergence)

## Output Files

`.covmat`, `.v_info`, `h_info`, `.bestfit`, `.log`

## Analyze



# Analyze

## horizontal info

```

param names      :  Omega_L          h
R-1 values      :  0.000055    0.000015
Best Fit        :  7.918934e-01  7.339652e-01
mean            :  7.781465e-01  7.308724e-01
sigma           :  6.793778e-02  2.382595e-02

1-sigma -      :  -5.799105e-02 -2.310380e-02
1-sigma +      :  7.788452e-02  2.454811e-02
2-sigma -      :  -1.418414e-01 -4.808685e-02
2-sigma +      :  1.333294e-01  4.764994e-02
3-sigma -      :  -2.379013e-01 -7.257030e-02
3-sigma +      :  1.779086e-01  6.790438e-02

1-sigma >      :  7.201555e-01  7.077686e-01
1-sigma <      :  8.560310e-01  7.554205e-01
2-sigma >      :  6.363051e-01  6.827856e-01
2-sigma <      :  9.114759e-01  7.785224e-01
3-sigma >      :  5.402452e-01  6.583021e-01
3-sigma <      :  9.560552e-01  7.987768e-01

```

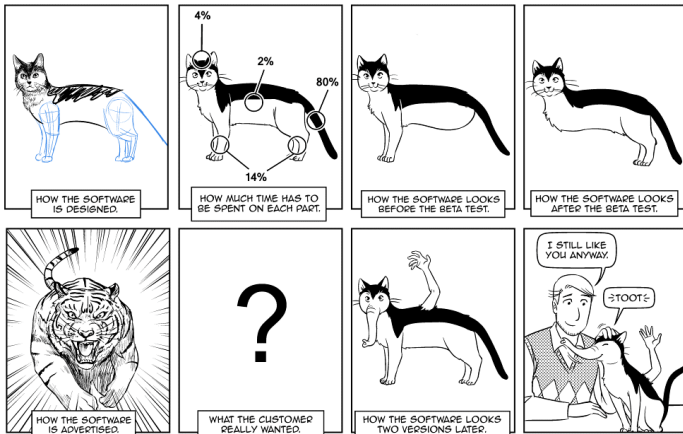
# Analyze

other files

- run.log
- run.covmat
- run.tex

# Conclusion on Design

## Richard's guide to software development



Sandra and Woo by Oliver Knörzer (writer) and Powree (artist) – [www.sandraandwoo.com](http://www.sandraandwoo.com)



# Outline

- 1 Code Structure
- 2 Important Modules in (some) detail
- 3 Usage
  - Organizing the directory
  - Complete work session example
  - Analyzing and plotting the results
- 4 Practice with Monte Python

# Organizing the directory

## Installing CLASS Python wrapper

```
]$ cd class/  
class]$ make; cd python; python setup.py install --user
```

## Configuration

- `cp default.conf.template default.conf` and **edit it**
- `python montepython/MontePython.py -o chains/test \`  
`-p example.param -N 10`

This last line will create the file `chains/test/2014-03-18_10.txt`

# Work session example

Test a model with running tilt with (fake) Planck data

- **Copy** `example.param` into `example_alpha.param`

```
data.experiments=['fake_planck_bluebook']
```

```
# Cosmological parameters list
```

```
data.parameters['omega_b'] = [2.249, None, None, 0.016, 0.01, 'cosmo']
data.parameters['omega_cdm'] = [0.1120, None, None, 0.0016, 1, 'cosmo']
data.parameters['n_s'] = [0.963, None, None, 0.004, 1, 'cosmo']
data.parameters['A_s'] = [2.42, None, None, 0.038, 1e-9, 'cosmo']
data.parameters['h'] = [0.703, None, None, 0.0065, 1, 'cosmo']
data.parameters['tau_reio'] = [0.085, None, None, 0.0044, 1, 'cosmo']
```

# Work session example

Test a model with running tilt with (fake) Planck data

- Copy `example.param` into `example_alpha.param`
- **Edit** `example_alpha.param`, add  $\Omega_k$

```
data.experiments=['fake_planck_bluebook']
```

```
# Cosmological parameters list
```

```
data.parameters['omega_b'] = [2.249, None, None, 0.016, 0.01, 'cosmo']
data.parameters['omega_cdm'] = [0.1120, None, None, 0.0016, 1, 'cosmo']
data.parameters['n_s'] = [0.963, None, None, 0.004, 1, 'cosmo']
data.parameters['A_s'] = [2.42, None, None, 0.038, 1e-9, 'cosmo']
data.parameters['h'] = [0.703, None, None, 0.0065, 1, 'cosmo']
data.parameters['tau_reio'] = [0.085, None, None, 0.0044, 1, 'cosmo']
data.parameters['alpha_s'] = [0, None, None, 0.01, 1, 'cosmo']
```

# Work session example

Test a model with running tilt with (fake) Planck data

- Copy `example.param` into `example_alpha.param`
- Edit `example_alpha.param`, add  $\Omega_k$

- **Launch a short run to see if it works:**

```
python montepython/MontePython.py -o chains/running \  
-p example_alpha.param -N 10
```

# Analyzing and Plotting

## After running longer chains

- do: `python montepython/MontePython.py -info chains/planck/Omega_k -bins 10`
- use the output covariance matrix as an input (new chains!)

# Outline

- 1 Code Structure
- 2 Important Modules in (some) detail
- 3 Usage
- 4 Practice with Monte Python

# Exercices

## I) Hst, SN, BAO

Make a  $\Lambda$ CDM run with Hubble Space Telescope, Super Novae and BAO data.

## II) Analyze the result

Using chains in the dropbox folder

## III) Have fun with classy

Use the classy wrapper in a Python interpreter (notebook) to redo yesterday's exercices (see example in the dropbox folder).

## IV) Installing Planck likelihood

and trying it with `base.param`

<http://pla.esac.esa.int/pla/aio/planckProducts.html>