



INSTITUTO DE  
CIENCIAS  
FÍSICAS

## A short introduction to: *MontePython*

---

Estimación de parámetros cosmológicos

Gabriela García Arroyo (arroyo@icf.unam.mx)

ICF-UNAM

# Contents

## 1 Referencias

## Referencias principales



[https://github.com/brinckmann/montepython\\_public](https://github.com/brinckmann/montepython_public)



<https://baudren.github.io/montepython.html>

# Descargar

[https://github.com/brinckmann/montepython\\_public](https://github.com/brinckmann/montepython_public)

The screenshot shows the GitHub repository page for `brinckmann/montepython_public`. The repository is public and forked from `baudren/montepython_public`. It has 13 watchers, 84 forks, and 95 stars. The repository is currently on the `3.6` branch, which is 486 commits ahead of and 1 commit behind the `baudren/montepython_public` branch. The repository contains several files and folders, including `bestff`, `chains`, `covmat`, `data`, `input`, `montepython`, `plot_files`, `sphinx-documentation`, and `tests`. A `Clone` dropdown menu is open, showing options for `Local`, `Codespaces`, and `Clone`. The `Clone` options include `HTTPS`, `SSH`, and `GitHub CLI`. The `HTTPS` URL is `https://github.com/brinckmann/montepython_public`. The `Clone` menu also includes a `Download ZIP` option. The repository is public and has 13 watchers, 84 forks, and 95 stars.

■ \$ git clone .....

# Documentación

Versiones anteriores, referencias a papers, cursos, etc...

The screenshot shows the homepage of the MontePython project. At the top, there is a navigation bar with links for Home, Monte Python, Note Organizer, Markup to Beamer, CV, and Github. The main heading is "Monte Python" with the subtitle "The Monte Carlo code for CLASS in Python". A prominent blue button says "Download the latest version". Below this, there are sections for "News", "Description", "Installation", "Documentation", and "Getting Help".

**News**  
 Latest release is 3.0, and lives in a different repository. Head over to there.  
 The full changelog is available on the [Github page](#).  
 To be informed by mail of the latest release, write to [brinckmann at physik.rwth-aachen.de](mailto:brinckmann@physik.rwth-aachen.de), and you will be added to the mailing list.

**Description**  
**MONTÉ PYTHON** is a Monte Carlo code for Cosmological Parameter extraction. It contains likelihood codes of most recent experiments, and interfaces with the Boltzmann code **CLASS** for computing the cosmological observables.  
 Several sampling methods are available: Metropolis-Hastings, Nested Sampling (through [MultiNest](#)), EMCEE (through [CosmoHammer](#)) and Importance Sampling.

**Installation**  
 The whole procedure is detailed [there](#). You will need Python 2 (from 2.5 to 2.7). The Python packages will be installed automatically by the installation script.  
 If you have `git`, the easiest way to install the code is to simply issue

```
git clone https://github.com/naudren/montepython_public
cd montepython_public
python setup.py install --user
```

**Documentation**  
 A complete documentation, alongside detailed instructions on installing the dependencies, notably the **CLASS** wrapper and the Planck likelihood is available on [Read the Docs](#). A less complete pdf version is also [available](#).

**Getting Help**  
 If the information contained in the documentation is not

<https://naudren.github.io/montepython.html>

# Instalación

## Después de descargar

- La parte de class ya la tenemos. (si les da problemas vayan desde la terminal a class\_public/python] \$ python setup.py -build )
- Los datos de Planck están fuera de este curso.
- Vamos a copiar el archivo: montepython\_public/default.conf.template a default.conf
- Allí debemos especificar la ruta a class\_public en:  
path['cosmo'] = 'PathToYour/class\_public', en mi caso (prefiero no usar root):

```

24 path['cosmo']           = '/home/gaby/Projects/Cosmo/class_public'
25 path['clik']            = '/home/gaby/Projects/code/plc_3.0/plc-3.01

```

Ya podemos hacer una prueba.

## param

Archivo de configuración que contiene los parámetros cosmológicos, sus priors y los likelihoods a utilizar, los ejemplos se encuentran en `montepython_public/input`

```

1 #-----Experiments to test (separated with commas)-----
2
3 # valid options are (so far): acbar, bicep, boomerang, cbl,
4 # hst, quad, sn, spt, fake_planck_bluebook,
5 # euclid_pk, euclid_lensing
6 # and clik_fake_planck (for Planck members)
7
8 data.experiments=['JLA']
9
10 #----- Parameter list -----
11 # data.parameters[class name] = [mean, min, max, 1-sigma, scale, role]
12 # - if min max irrelevant, put to -1 or None (if you want a boundary of -1, use -1.0)
13 # - if fixed, put 1-sigma to 0
14 # - if scale irrelevant, put to 1, otherwise to the appropriate factor
15 # - role is either 'cosmo', 'nuisance' or 'derived'
16
17
18 # Cosmological parameters list
19 data.parameters['Omega_cdm'] = [0.2562, None, None, 0.008, 1, 'cosmo']
20
21 # Nuisance
22 data.parameters['alpha'] = [0.15, None, None, 0.001, 1, 'nuisance']
23 data.parameters['beta'] = [3.559, None, None, 0.02, 1, 'nuisance']
24 data.parameters['M'] = [-19.02, None, None, 0.004, 1, 'nuisance']
25 data.parameters['Delta_M'] = [-0.10, None, None, 0.004, 1, 'nuisance']
26
27 # Derived parameter list
28 data.parameters['Omega_m'] = [0, -1, -1, 0,1, 'derived']
29
30
31 data.cosmo_arguments['Omega_b'] = 0.05
32 #----- Mcmc parameters ----
33 # Number of steps taken, by default (overwritten by the -N command)
34 data.N=10
35 # Number of accepted steps before writing to file the chain. Larger means less
36 # access to disc, but this is not so much time consuming.
37 data.write_step=5

```

## Algunos comentarios y ejemplos sobre .param

- Los nombres válidos para los likelihoods los encuentras en `montepython_public/montepython/likelihoods`  
`data.experiments=['hst', 'Pantheon']`
- Parámetros cosmológicos a variar, estos 'definen' tu cosmología:  
`data.parameters['omega_b'] = [2.249, 1.8, 3, 0.016, 0.01, 'cosmo']`
- Nuisance parameters dependen de tu conjunto de datos, revisa si es que tiene:  
`data.parameters['M'] = [0.1, -1., 1, 0.01, 1, 'nuisance']`
- Para parámetros derivados:  
`data.parameters['Omega_Lambda'] = [0, -1, -1, 0,1, 'derived']`
- Parámetros que no se varían y nos ayudan a definir la Cosmología:  
`data.cosmo_arguments['deg_ncdm'] = 3`  
`data.cosmo_arguments['fluid_equation_of_state'] = 'CLP'` En este último caso entonces ya puedes agregar  $w_0$  y  $w_a$  a los parámetros a variar y también puedes cambiar la parametrización´.

## Cómo se usa MontePython

Desde la terminal dentro de `montepython_public`:

- `montepython_public]$ python montepython/MontePython.py -help`
- `montepython_public]$ python montepython/MontePython.py run -help`
- `montepython_public]$ python montepython/MontePython.py info -help`

Con esto vas a conocer las opciones que tienes para correr cadenas (`run`) y para analizarlas (`info`).

Una primer corrida básica, para asegurarte que el `.param` está bien definido puede ser con 10 pasos (`-N 10`). Probemos `input/test.param`:

```
python montepython/MontePython.py run -p input/test.param -o testprueba -N 10
```

- después de `-p` indicas la ruta a tu `.param`
- después de `-o` la carpeta donde va a guardar las cadenas

Si la ejecución no fue exitosa hay que borrar la carpeta.- Antes de volver a ejecutar.- `montepython_public]$ rm -rf testprueba`

## Recomendaciones -personales-

- Aunque puedes usar `mpirun -np 4`, recomiendo mejor correr la versión estándar digamos 4 veces para evitar conflictos, (elige la misma carpeta de output)
- Realiza una primer corrida de prueba con algunos 500 pasos y analizala. Recomendable pedirle la matriz de covarianza (`-want-covmat`)  
`python montepython/MontePython.py info testprueba -want-covmat`
- Haz una corrida más 'enserio' y pásale la matriz de covarianza generada  
`python montepython/MontePython.py run -p input/test.param -o testprueba -N 1000 -c testprueba/test.covmat`
- Analiza los archivos de salida y las impresiones en la terminal. El comando `info` genera una triangle plot y una con las posteriors, también te escribe el bestfit, el valor a 1 y 2 sigma así como el valor del Gelman-Rubin (`testprueba/test.h_info`)
- Decide si tus cadenas ya convergieron, si no vuelve al paso anterior, también le puedes adicionar `-b testprueba/test.bestfit` esto va a reemplazar el punto donde inicia cada parámetro por el valor del mejor ajuste que llevas hasta este momento.



# Likelihoods

- Las likelihoods disponibles se encuentran en: `montepython_public/montepython/likelihoods`
- Sus datos correspondientes se encuentran en: `montepython_public/data`
- Como ejemplo veamos Pantheon



El archivo `__init__.py` tiene el código del Likelihood la clase se llama Pantheon -este es el nombre que escribimos en el `.param`-  
 El archivo `Pantheon.data` tiene especificaciones sobre el conjunto de datos; conflictos con otros datos, nuisance pars, ruta a los datos.

# Likelihoods

En `montepython_public/data`



Básicamente son las tablas con los datos, covarianzas, etc. Estas tablas son llamadas en el archivo `Pantheon.data` (previamente mencionado).