



UNIVERSIDAD DE CÓRDOBA



Bayesian estimation of cosmological parameters with CosmoMC & MontePython

Antonio J. Cuesta

Universidad de Córdoba, Spain

Cosmology School in the Canary Islands, Fuerteventura 18/09/2017

Outline

(keeping it simple...)

- Motivation & Initial remarks
- Generic steps when running MCMC's
- Introduction to MontePython
- Introduction to CosmoMC

Introduction & Motivation

What is all this about?

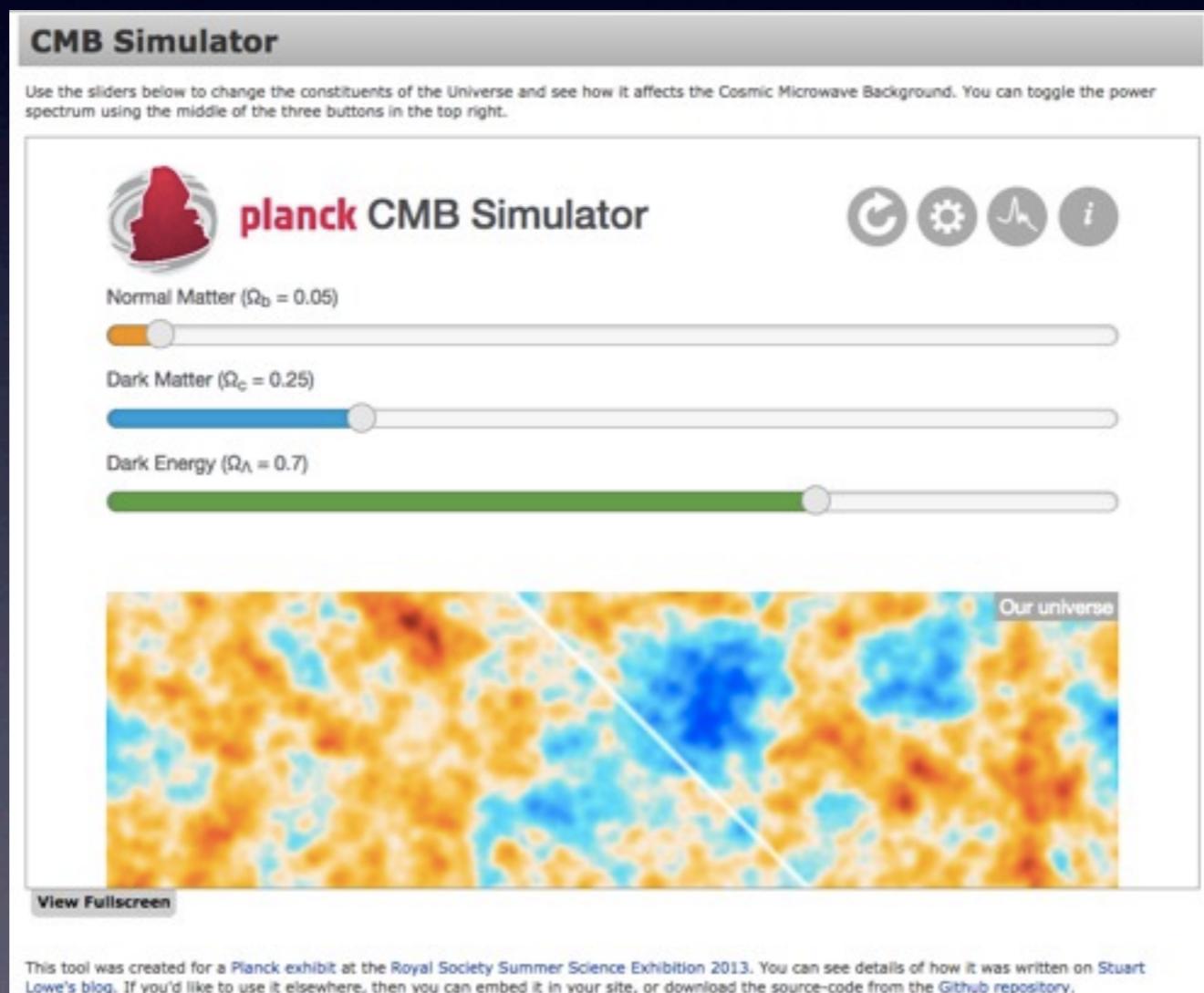
- Cosmological models have **parameters** (matter density Ω_m , Hubble constant H_0 , ...) For example, $\Lambda\text{CDM} = \{ \Omega_b h^2, \Omega_c h^2, H_0 \text{ (or } \theta_s), A_s \text{ (or } \ln 10^{10} A_s), n_s, T_{\text{reio}} \}$
- We want to know the **value** and the **error** of those parameters (mean and rms).
- Ideally, we want the **full probability distribution function** (pdf) for each parameter, and even the full joint distribution for all the parameters together $P(\theta_1, \theta_2, \theta_3, \dots)$
- The **limits** on the possible values of those parameters ($1\sigma=68.3\%$ confidence, $2\sigma=95.4\%$, $3\sigma=99.7\%$...) can be obtained in a Bayesian way, in which the prior knowledge of the parameters will play an important role $P(\theta|\mathbf{x}) = P(\theta)\mathcal{L}(\mathbf{x}|\theta)$

What is all this about?

- The resulting constraints on each parameter will depend on these two ingredients: the DATA you input (each experiment targets a different cosmological probe) the MODEL studied (each model contains a different number of free parameters)
- In general, a model with *more* free parameters will usually provide weaker constraints, but also provides a better fit: more parameters can accommodate more features, (hence returning a lower χ^2) although not necessarily will improve the *bayesian* evidence
- Furthermore, the constraints from two different datasets might appear to be inconsistent (“in tension”, i.e. not overlap) in one model but not in another model

Cosmological parameter estimation

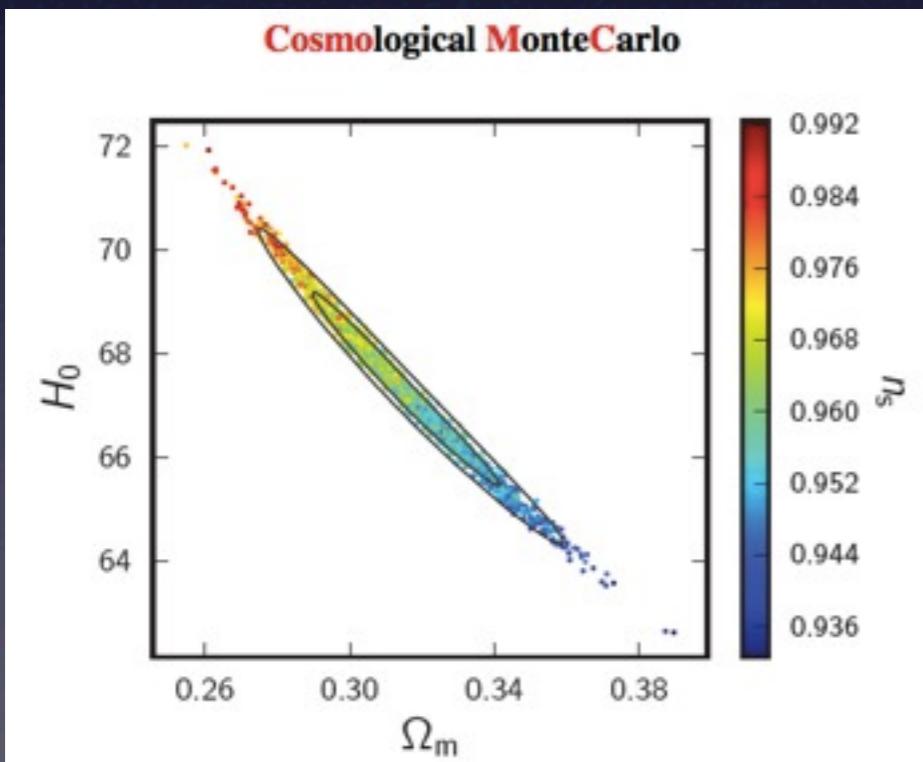
You can estimate parameters by eye (not recommended)...



<http://planck.cf.ac.uk/cmb-sim>

Cosmological parameter estimation

Or we can estimate parameters *quantitatively*
In cosmology there are two major codes for this:
CosmoMC and **MontePython** (& Cosmossis, PICO,...)



<http://cosmologist.info/cosmomc/>

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

Why CosmoMC?

- It has been used in many papers!
(2000+ citations to Lewis & Bridle 2002)
- Usually new data (likelihood codes) are typically available first in CosmoMC
- It has a GUI to analyze the chains!!
(but you can use it for MontePython chains too)
- Its Boltzmann solver, CAMB (based on CMBFAST) has been extensively used for a long time
- Lots of tools to run this code in a cluster
Capabilities to run batches of models, MPI/OpenMP parallelization

Why MontePython?

- It is **very easy** to get started!
(we'll see that in the afternoon hands-on session)
- It is **written in Python**:
the code is easy to read and easy to modify
- Its Boltzmann solver (the code to compute
the evolution of perturbations) is **CLASS**: (or hi-class)
a direct transcript of the equations in [astro-ph/9506072](#) (Ma & Bertschinger 1995)
very well commented, so less pain to implement new physics (if not already built-in)
- It is completely **agnostic about cosmology**:
if you define a new parameter in CLASS, you don't need to modify MontePython

Running Markov Chain Monte Carlo (MCMC)

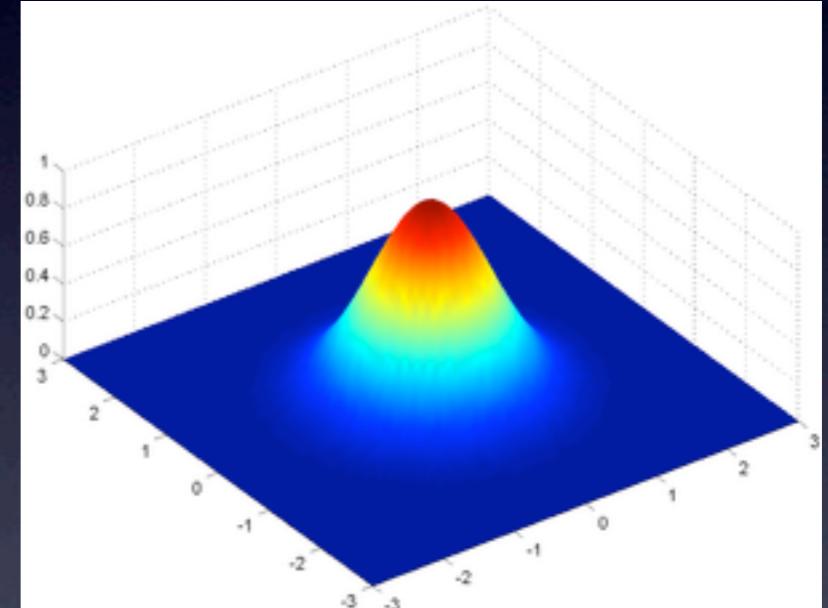
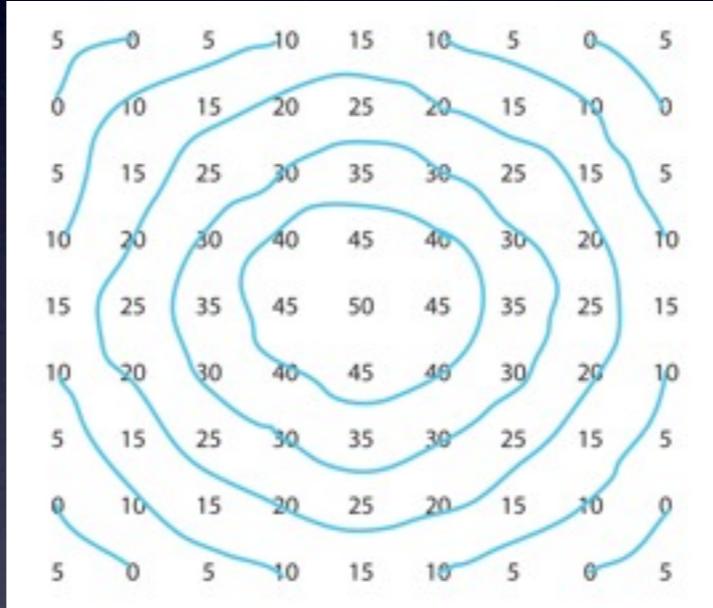
MCMC: How-To?

Breaking the code into individual tasks

- **Step 1:** Choose a point in the parameter space (*walk* the parameter space)
- **Step 2:** Solve the Boltzmann equations for those values of the parameters which are translated into the observables $P(k)$, $D(z)$, C_ℓ ...
- **Step 3:** Evaluate the likelihood of this point for each dataset (CMB, BAO, SNe...) by comparing observables with the measurements (evaluation of χ^2)
- **Step 4:** Compute the total likelihood, and Accept the point if the likelihood is larger (-lnlike smaller), or reject it otherwise, then move to another point (step 1)
- **Step 5:** After you have RUN the chains (i.e. after getting enough points to sample the posterior distribution) the final step is to do an ANALYSIS of the results

Step I: walk a parameter space (low- d case)

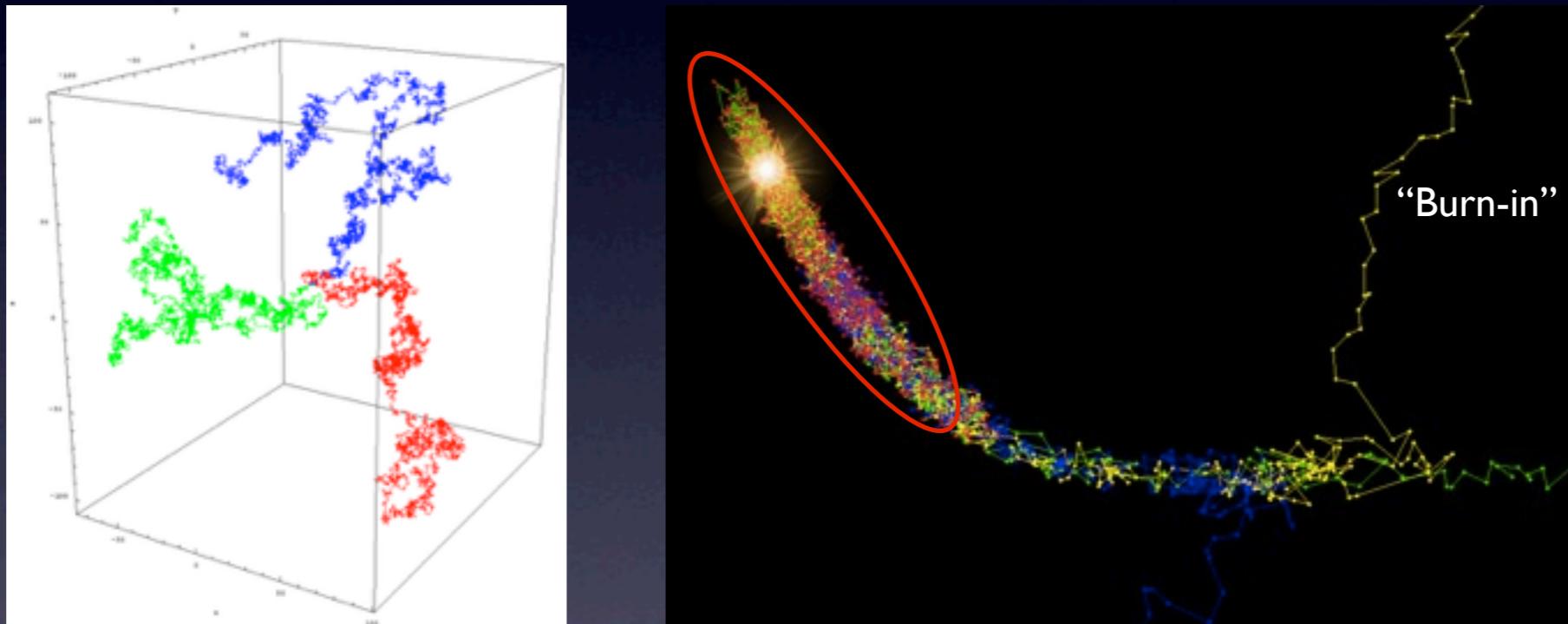
- If the number of dimensions d is small (1D or 2D), then a grid (discretized) would suffice, but this is not true in cosmology (even Λ CDM has already 6 parameters)



- Number of (potentially) visited points: N^d (grows exponentially with d). N depends on how fine the spatial discretization (resolution) you want it to be. Usually one computes the distribution in all the points (even away from the peak).

Step I: walk a parameter space (high- d case)

- If the number of dimensions is large, it is not feasible nor efficient to do a grid



unimodal distribution → different initial conditions will converge to the same region

- It is better to do a *random walk*, in which there is some kind of “dragging force” towards the high likelihood region plus a thermal bath to move around the peak (the goal is not to find the maximum -- it is to sample points from the distribution)

The Metropolis-Hastings algorithm

This is exactly what the **Metropolis-Hastings algorithm** does:

If you are at location x , pick a location y (within some search radius from x):

- If $P(y) > P(x)$, jump there, and store y in the list of visited points
- If $P(y) < P(x)$, then pick a uniform random number r between 0 and 1:
 - If $r < P(y)/P(x)$, jump there, and store y to the list of visited points
 - If $r > P(y)/P(x)$, do not jump there, and store x (again) to the list of visited points

Cycle over this loop until you get enough points ($\sim 10^5$), or better, until a *convergence criterion* is achieved (see next slide)

Remember: In general x and y will be d -dimensional vectors, with as many components as parameters you want to determine



When MCMC is considered “converged”?

- The most common criterion is the R diagnostic by Gelman & Rubin (1992). This diagnostic is defined for each parameter, so we focus on a single parameter θ .
- Suppose all chains have length N (each chain has N steps) and we have run M chains.
(M initial conditions)
- Each chain “ m ” will return a different mean μ_m and variance σ^2_m for the parameter θ .
- The variance of the means μ_m (times N) is called the variance “between chains” B .
- The mean of the variances σ^2_m is called the variance “within chains” W .
- The R parameter is then defined as (see Brooks & Gelman 2007 for exact formulae):

$$\hat{R} = \sqrt{1 + (1/N)(B/W)}$$

Typically, a $R-1$ value of <0.03 (ideally <0.01) for all parameters is considered good enough

So when all chains return roughly the same μ_m & σ_m if the Central Limit Theorem applies (so $B \approx W$)

we have that $R-1$ goes to zero as N goes to ∞

Step 2: Solve the Boltzmann equations

(already covered in the previous sessions...)

- CAMB and CLASS are written to solve efficiently the evolution of each component (baryons, neutrinos, photons, dark matter, dark energy, metric...)

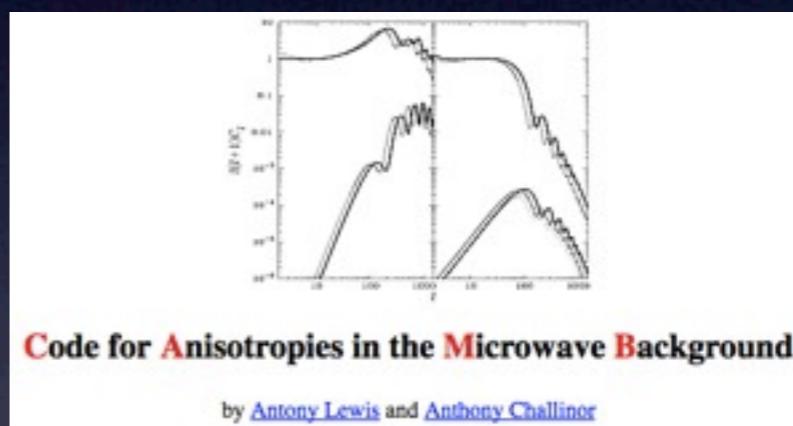
Synchronous gauge —

$$\begin{aligned}\dot{\delta} &= -(1+w) \left(\theta + \frac{\dot{h}}{2} \right) - 3\frac{\dot{a}}{a} \left(\frac{\delta P}{\delta \rho} - w \right) \delta, \\ \dot{\theta} &= -\frac{\dot{a}}{a} (1-3w)\theta - \frac{\dot{w}}{1+w}\theta + \frac{\delta P/\delta \rho}{1+w} k^2 \delta - k^2 \sigma,\end{aligned}$$

Conformal Newtonian gauge —

$$\begin{aligned}\dot{\delta} &= -(1+w) \left(\theta - 3\dot{\phi} \right) - 3\frac{\dot{a}}{a} \left(\frac{\delta P}{\delta \rho} - w \right) \delta, \\ \dot{\theta} &= -\frac{\dot{a}}{a} (1-3w)\theta - \frac{\dot{w}}{1+w}\theta + \frac{\delta P/\delta \rho}{1+w} k^2 \delta - k^2 \sigma + k^2 \psi.\end{aligned}$$

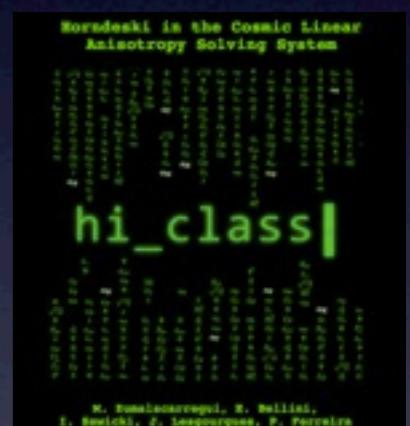
and many others...



Code for Anisotropies in the Microwave Background

by [Antony Lewis](#) and [Anthony Challinor](#)

<http://camb.info/>



<http://www.hiclass-code.net/>



CLASS

the Cosmic Linear Anisotropy Solving System

<http://class-code.net/>



- This is the part of the code you do not have to worry about (UNLESS you want to constrain non-standard cosmologies, which YOU will have to implement)

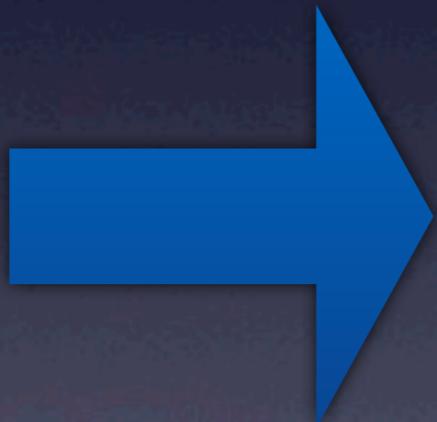
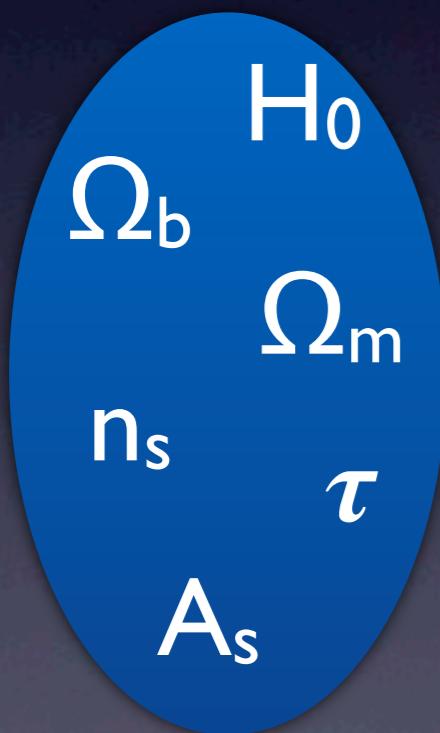
Step 3: Evaluate the likelihood(s)

parameter values
at this MCMC step

Likelihood
code

likelihood value or
chi-squared value

$$\vec{x} \xrightarrow{\mathcal{L}} \chi^2$$



$$\chi^2 = \left(\frac{\text{theory value} - \text{measured value}}{\text{measured error}} \right)^2$$

In general, if the data is a vector of values:

$$\chi^2 = (\vec{T} - \vec{D}) C^{-1} (\vec{T} - \vec{D})$$

C is the data (not parameter!) covariance matrix

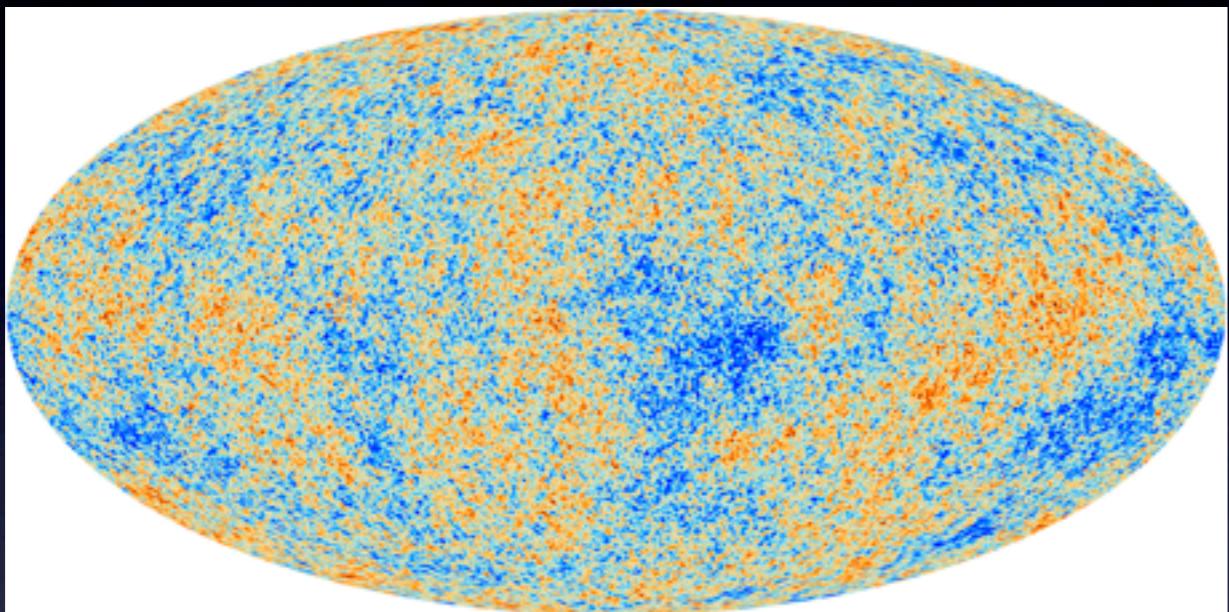
CosmoMC and MontePython will report $-\ln(\text{likelihood}) = \chi^2/2$ (Gaussian case, Wick's Theorem)

Here “value” can be a MCMC parameter (e.g. H_0) or a function of them (an observable, like $P(k)$ or σ_8)

WARNING: If “value” is NOT a MCMC parameter, it might depend on the (cosmological) model!

Cosmological probes

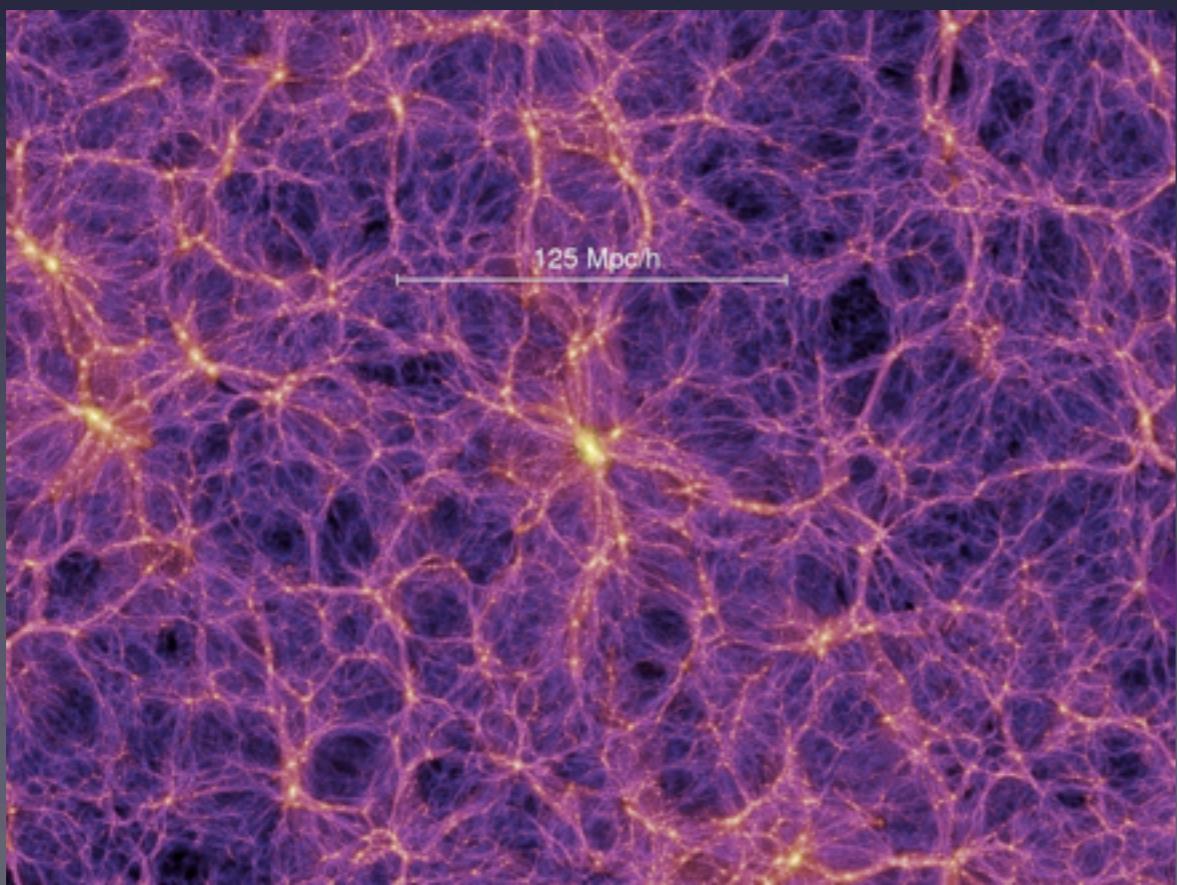
(and most commonly used *observables*)



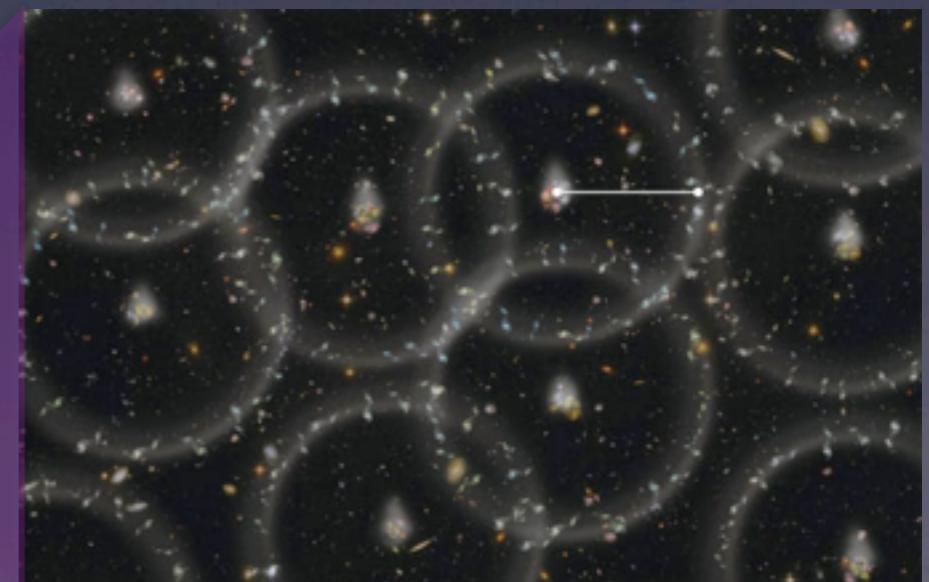
CMB



SNe

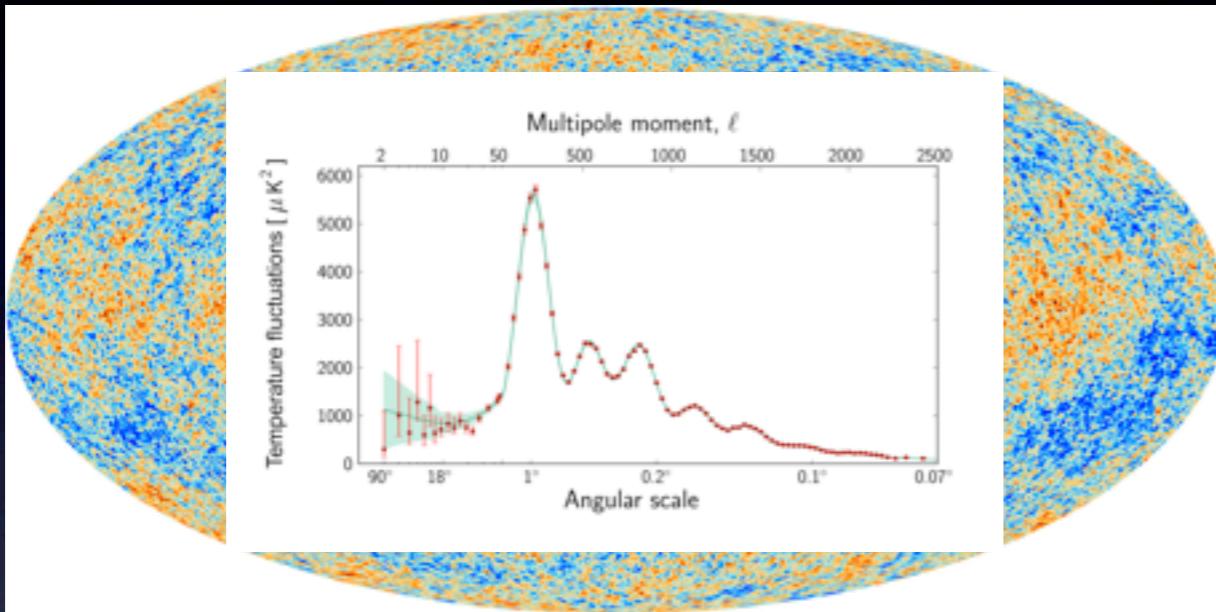


LSS

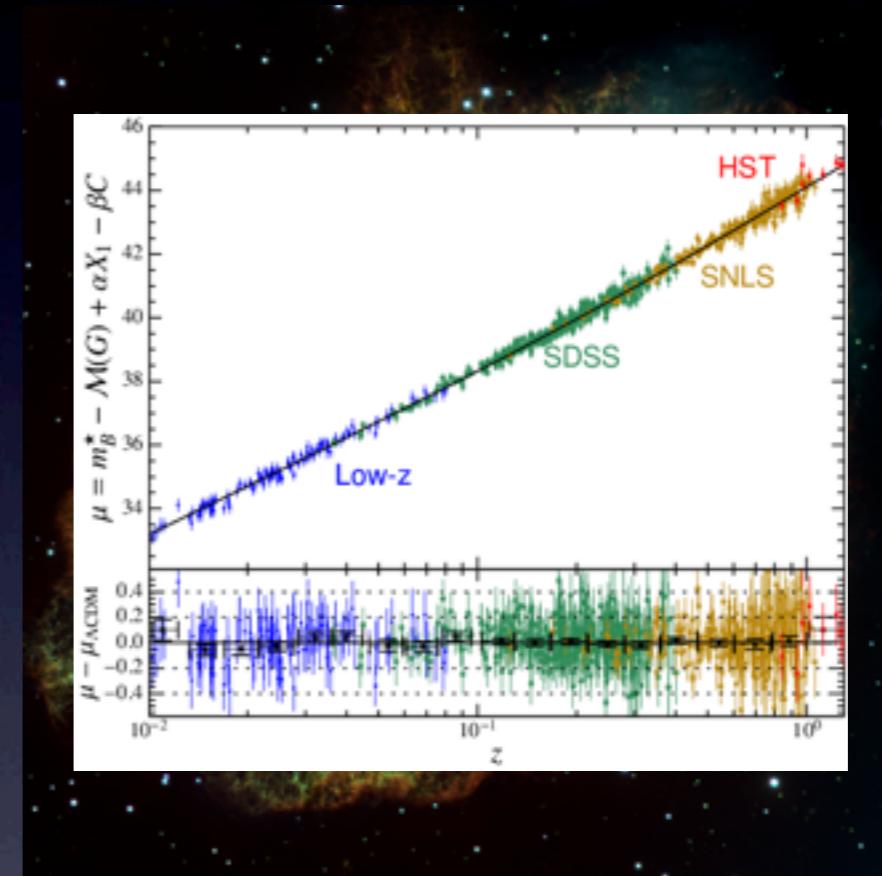


Cosmological probes (and most commonly used observables)

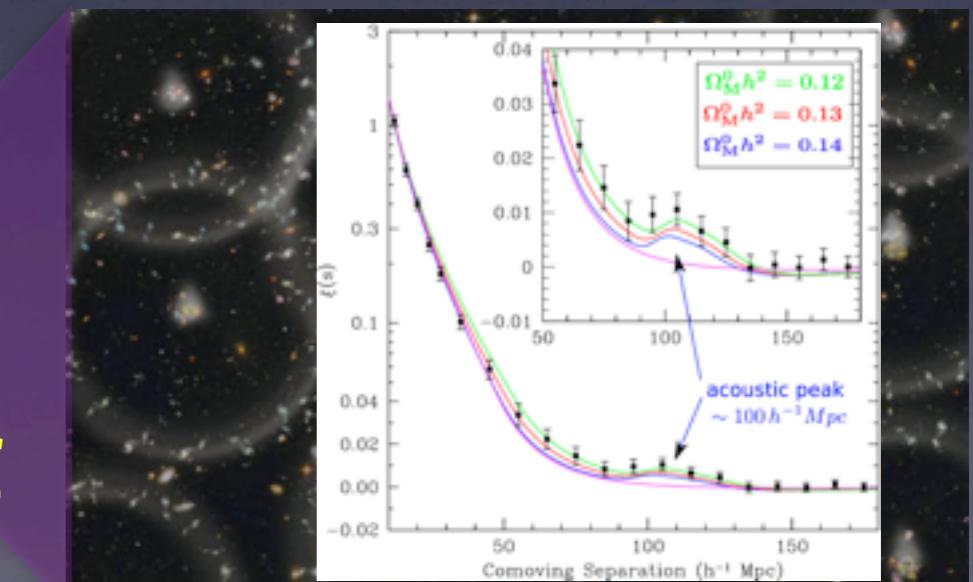
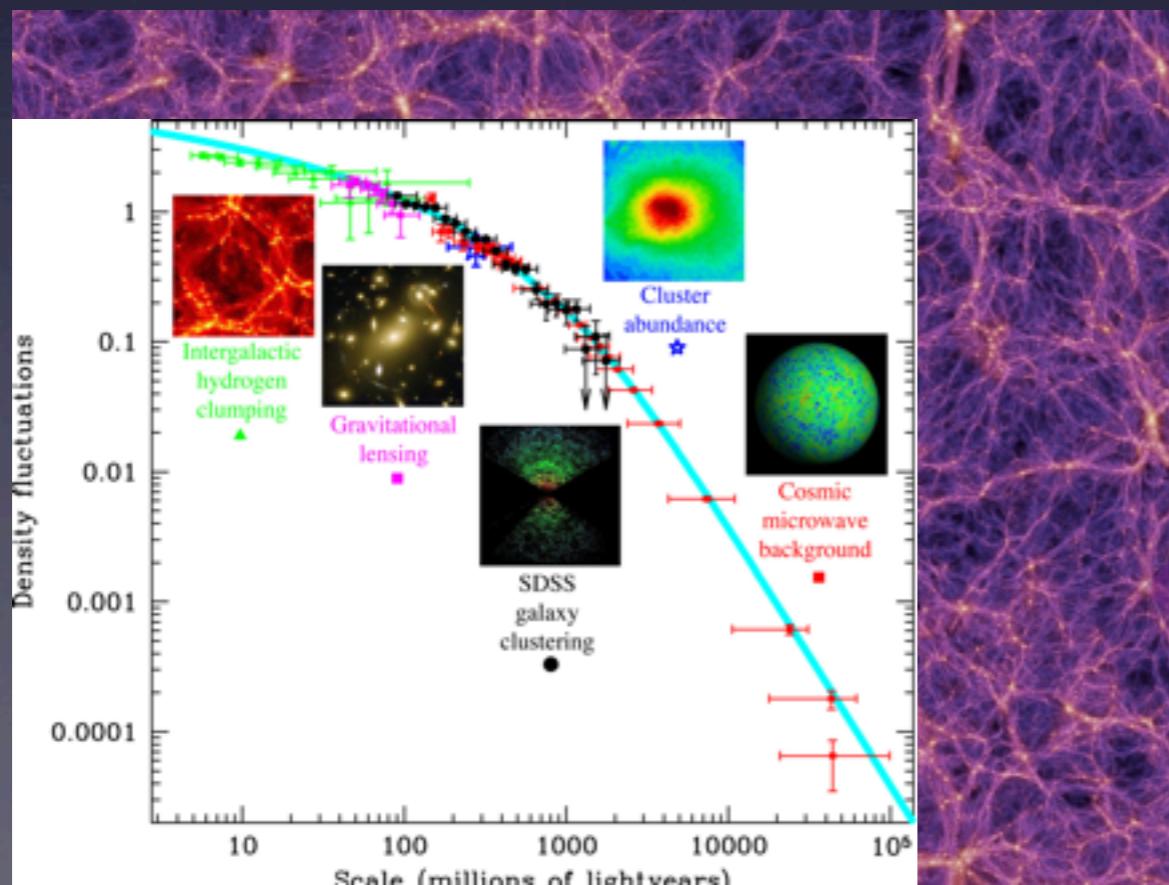
CMB



SNe



LSS



Step 3: Evaluate the likelihood(s)

<http://pla.esac.esa.int/pla/#home>

<http://www.cfhtlens.org/astronomers/cosmological-data-products>

We also deliver presentation material.

Since March 2014, the JLA likelihood plugin is included in the official release of cosmomc. For older versions, the plugin is still available (see below: Installation of the cosmomc plugin).

To analyze the JLA sample with SNANA, see \$SNANALYSIS_ROOT/sample_input_files/JLA2014/AAA_README.

http://supernovae.in2p3.fr/sdss_snls_jla/ReadMe.html

http://www.sdss3.org/science/boss_publications.php

Built-in likelihoods in CosmoMC & MontePython

Lots of likelihoods to choose from!
(you can also write your own, if you have the data)

```
AJCuestaMacBookAir:~ ajcuesta$ ls ~/montepython_public/montepython/likelihoods/
BK14                                bao_boss          fake_planck_bluebook
BK14priors                           bao_boss_aniso    gunn_peterson
CFHTLens                             bao_boss_aniso_gauss_approx hst
CFHTLens_correlation                 bao_known_rs      hst_riess
JLA                                  bicep            igm_temperature
JLA_simple                           bicep2           lowlike
Planck_SZ                            boomerang        polarbear
Planck_actsp                         cbi              quad
Planck_highl                          clik_fake_planck sdss_lrgDR4
Planck_highl_TTTEEE                   clik_wmap_full   sn
Planck_highl_lite                     clik_wmap_lowl  spt
Planck_lensing                        cmb_baryon       spt_2500
Planck_lowl                           cosmic_clocks_BC03 test_gaussian
WiggleZ                               cosmic_clocks_BC03_all test_nuisance1
WiggleZ_bao                           cosmic_clocks_MaStro test_nuisance2
__init__.py                            da_rec            timedelay
__init__.pyc                           euclid_lensing    wmap
acbar                                 euclid_pk         wmap_9yr
bao                                  fake_desi
```

```
AJCuestaMacBookAir:CosmoMC ajcuesta$ ls batch2/
BAO.ini                               WLHeymans.ini
BAODR11.ini                           WLonly.ini
BAO_RSD.ini                           WLonlyHeymans.ini
BK14                                  WMAP.ini
BK14.ini                             WMAPTEB.ini
BK14_README.txt                      WMAP_lowl.ini
BK14only.ini                         WMAP_tauprior.ini
BKPlanck                             WiggleZ_MPk.ini
BKPlanck.ini                         abundances.ini
BKPlanck_README.txt                  accuracy.ini
BKPlanckonly.ini                     common.ini
HST.ini                               fix_params.ini
HST_Freeman12.ini                    getdist_background.ini
HST_GPE70p6.ini                      getdist_base.ini
HST_GPE72p5.ini                      getdist_common.ini
HST_high.ini                         importance_sampling.ini
JLA.ini                               lensing.ini
JLA_marge.ini                        lensing_aggressive.ini
MPK.ini                               lensonly.ini
SZ_plus_CMB.ini                      likelihood.ini
SZ_plus_priors.ini                   lowEB.ini
WL.ini                               lowLike.ini
```

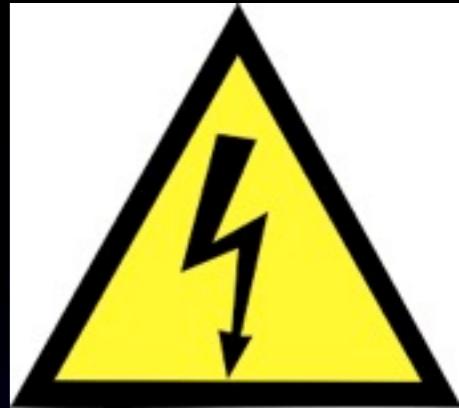
MontePython likelihoods

CosmoMC likelihoods

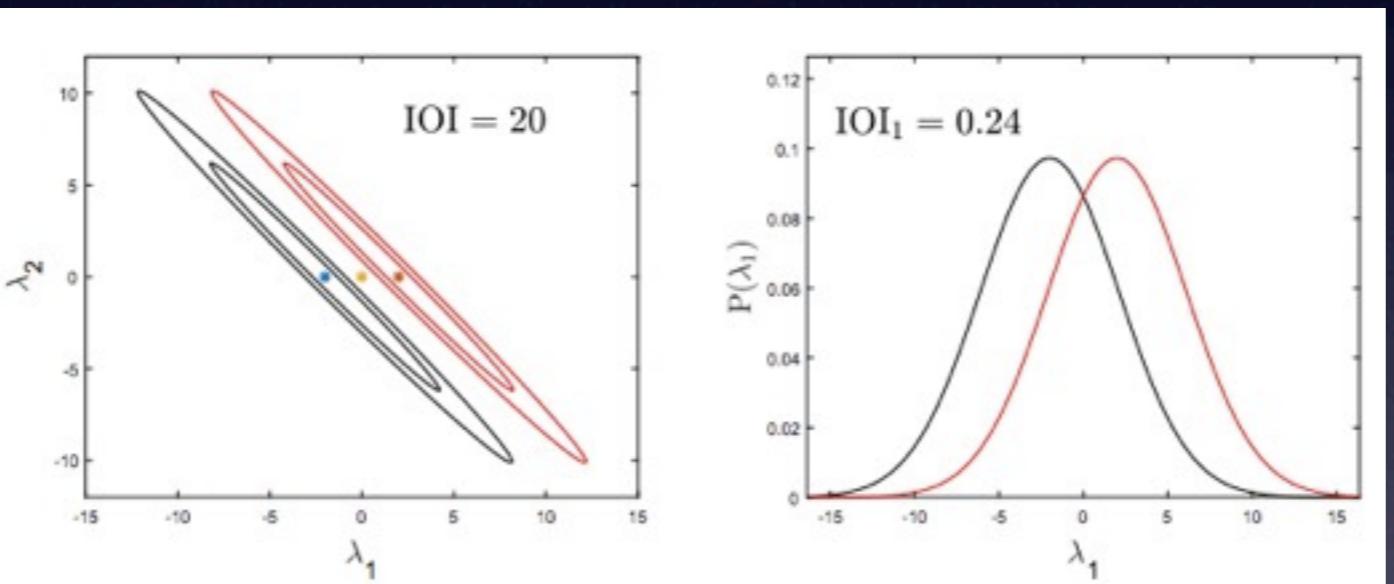
CMB from Planck (TT,TE,EE) and others, BAO (eg. BOSS), SNe (JLA), matter power spectrum, Hubble constant, Bicep/Keck CMB B-modes, Euclid/DESI mock likelihoods (for forecasts),....

Tensions!

why not use everything?



- When combining different datasets, one should always keep in mind that some pair of datasets can produce constraints that do not overlap in the d -dimensional parameter space (they might overlap in lower-dimensional plots)

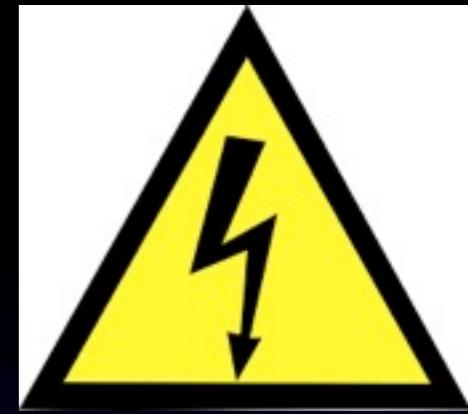


arXiv:1705.05303

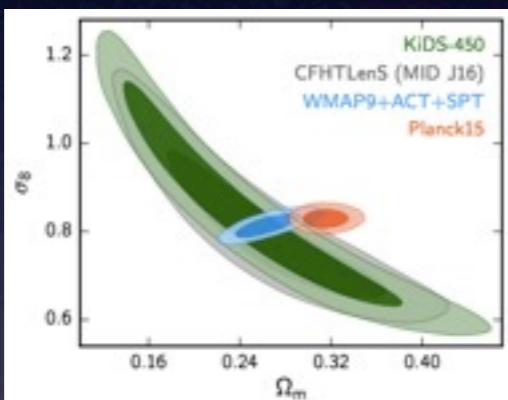
- This can lead to multi-modal posteriors, (potentially) spurious constraints (e.g. claims of non-zero neutrino masses, dynamical dark energy, couplings in the dark sector,...) and to probably too many astro-ph.CO postings every day

try to be minimalistic!

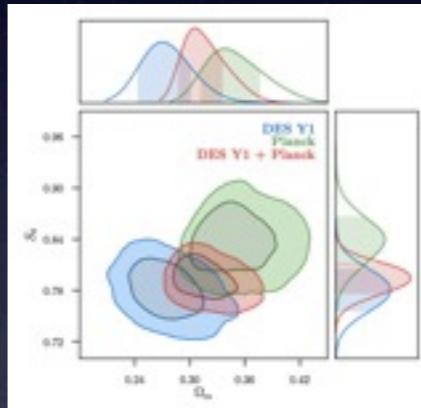
Tensions!



- Well-known examples of dataset tensions are:
 - The Large-Scale Structure (LSS) measurements of matter fluctuation amplitude σ_8 via weak lensing and those derived from the Cosmic Microwave Background (CMB)

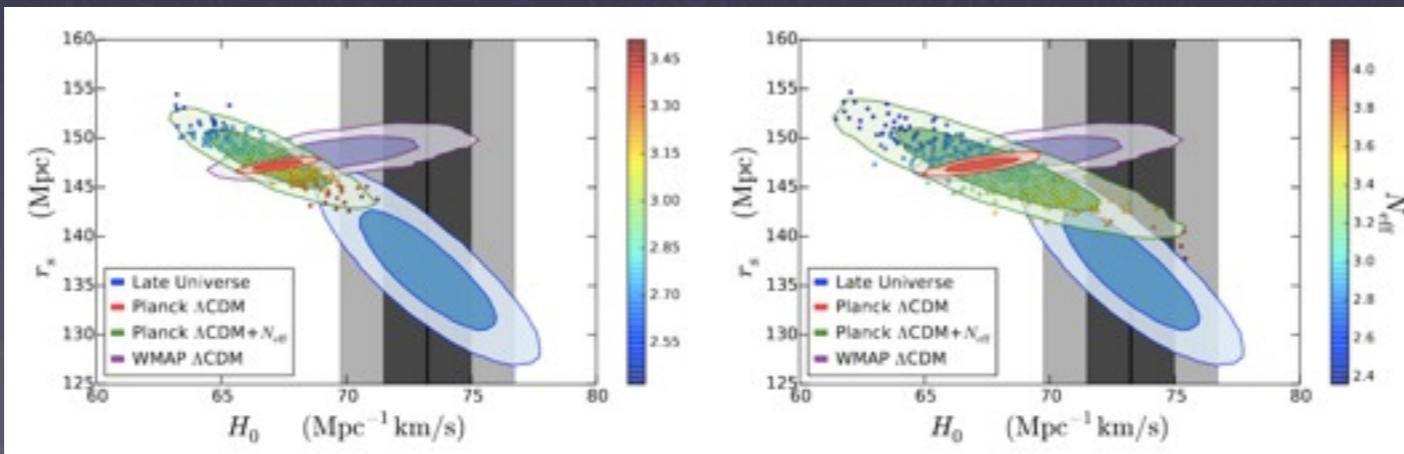


KIDS-450, arXiv: 1606.05338



DES Y1, arXiv: 1708.01530

- The local measurement of H_0 and its extrapolation from CMB assuming Λ CDM

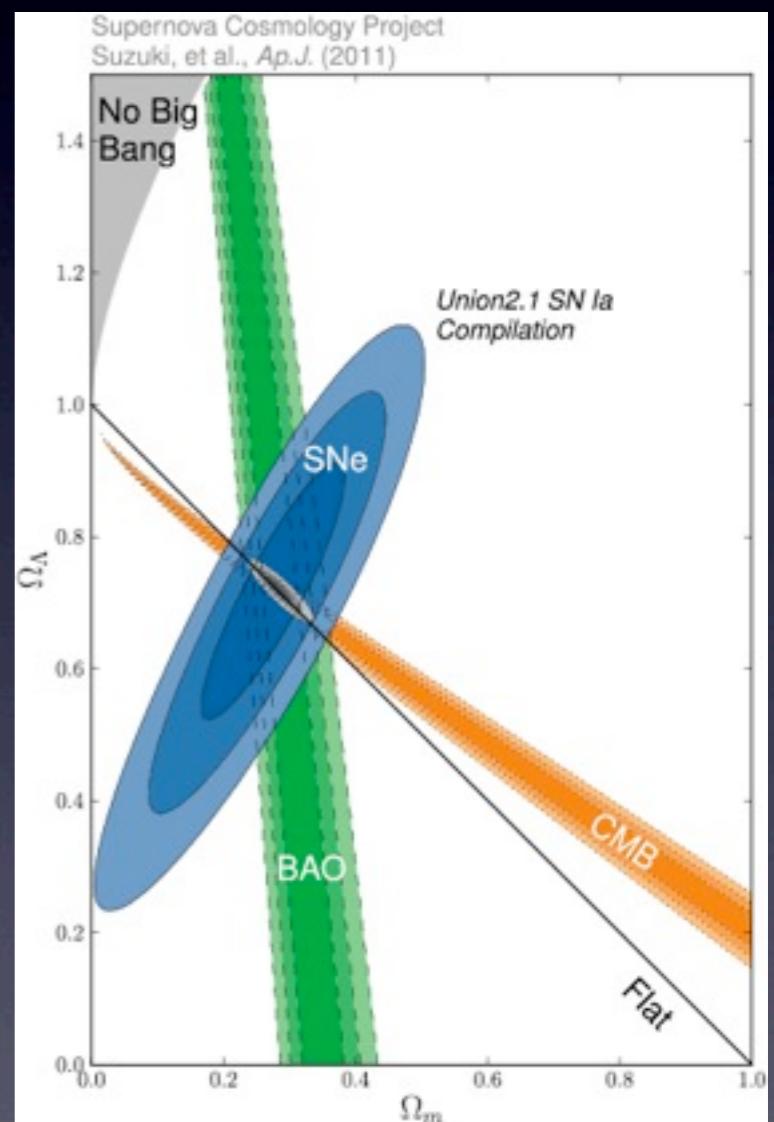


Bernal et al. arXiv:1607.05617

- Did we under-estimate uncertainties? Is Λ CDM wrong? Is GR/FLRW wrong? Is Planck wrong?
(need for massive neutrinos / dark radiation / non- Λ dark energy / other extensions ?)

Step 4: Decide whether to accept the point & Write to file

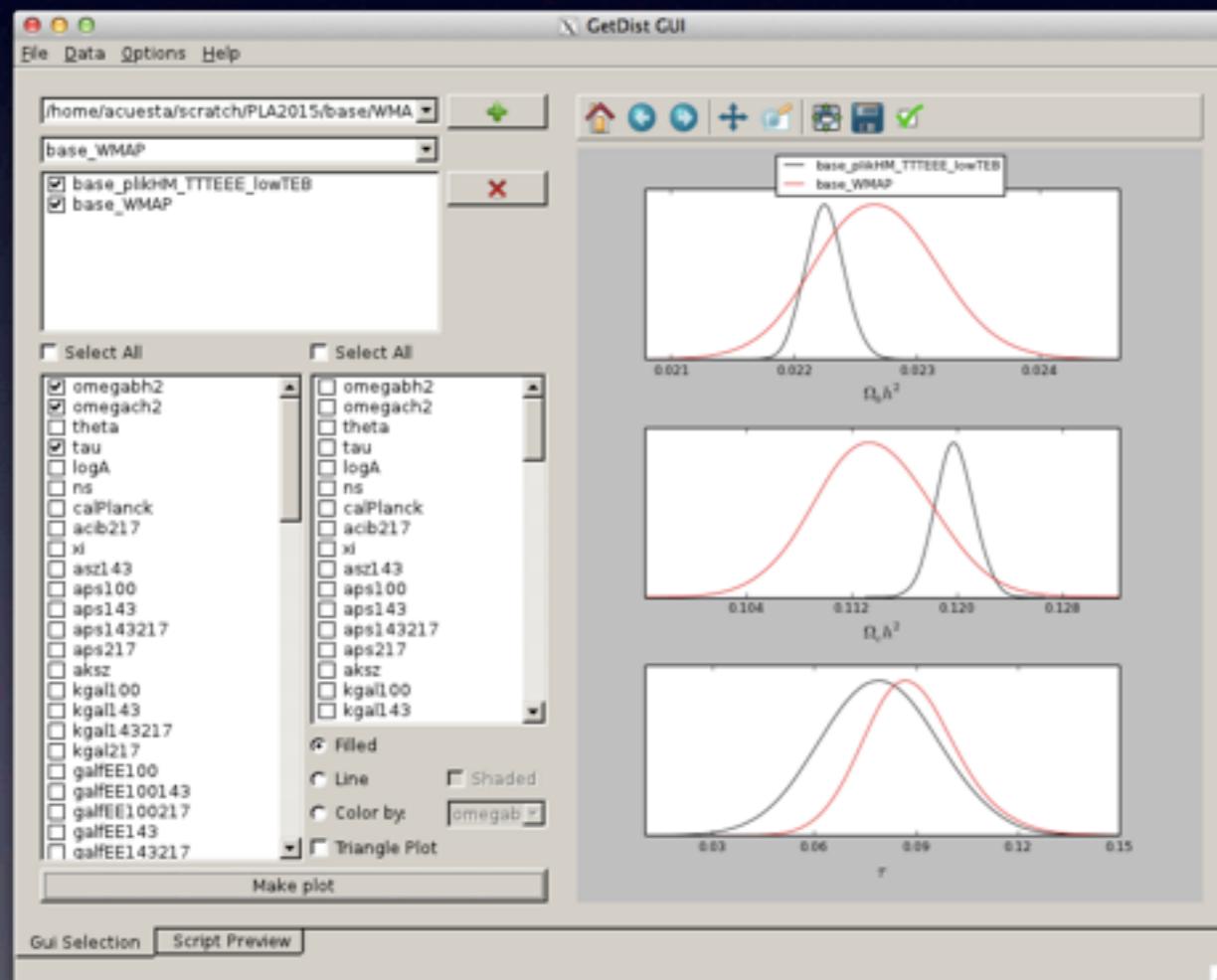
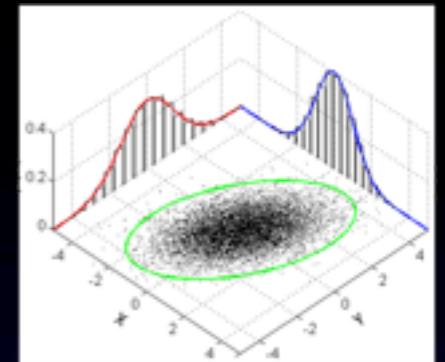
- Assuming all datasets are independent, the total likelihood is the *product* of the individual likelihoods, so the -log(likelihoods) are *added*
- For a *fixed* model, different datasets will allow different regions of the parameter space. If we combine all datasets, the allowed region will be the *intersection* of the individual allowed regions
- Following the Metropolis-Hastings algorithm, we compare the total likelihood at the new point with the likelihood at the previous point.
- Depending on this comparison, it will move to the new point or otherwise, it will count the previous point twice (the multiplicity of the point will be >1)



Step 5: Analysis of the results

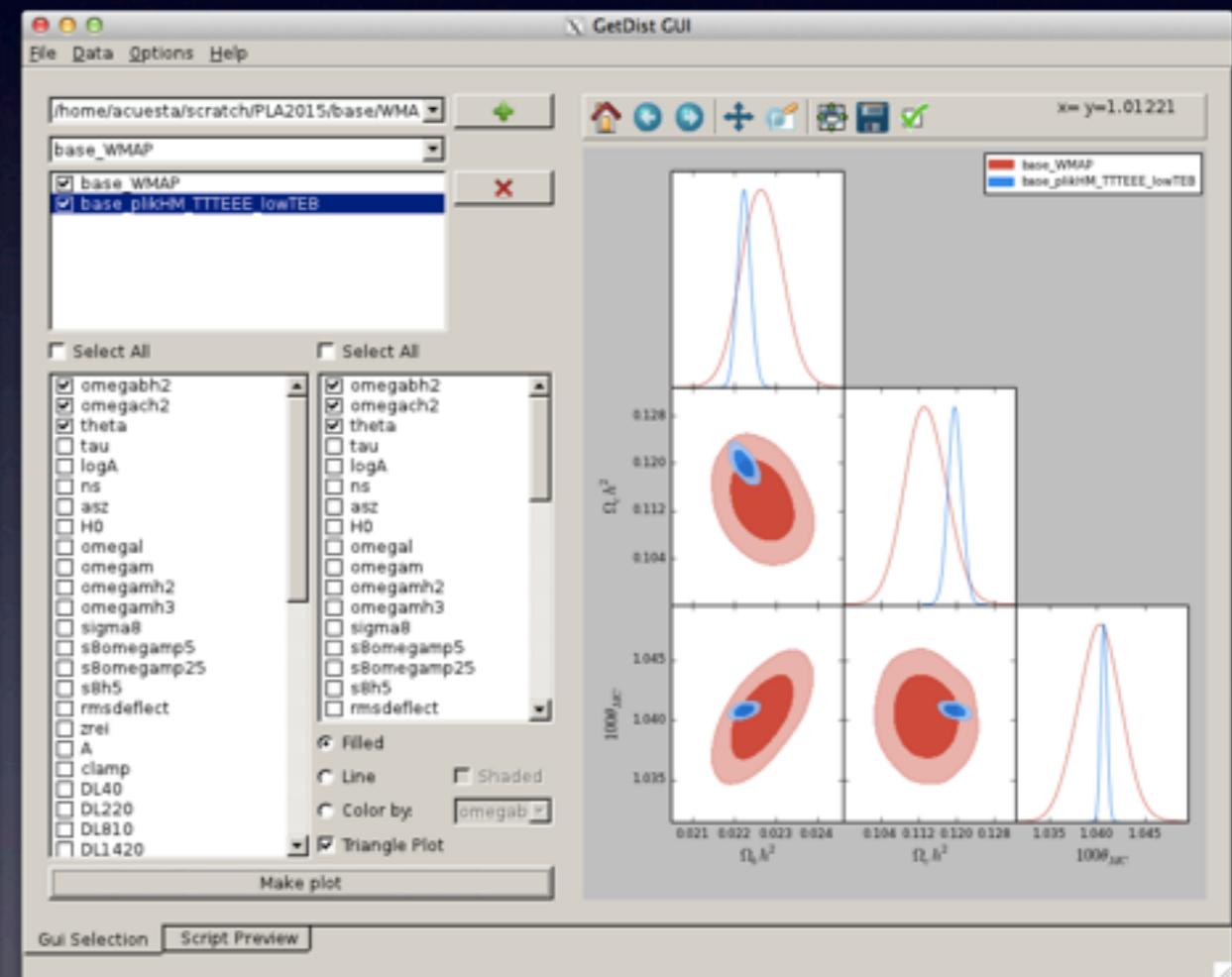
Once the chains have converged, we can make different types of summary **figures or tables** to convey the resulting information

This requires **marginalization**: integrate the d -dim PDF over all other parameters



1D Plot

(mean values and standard deviations)



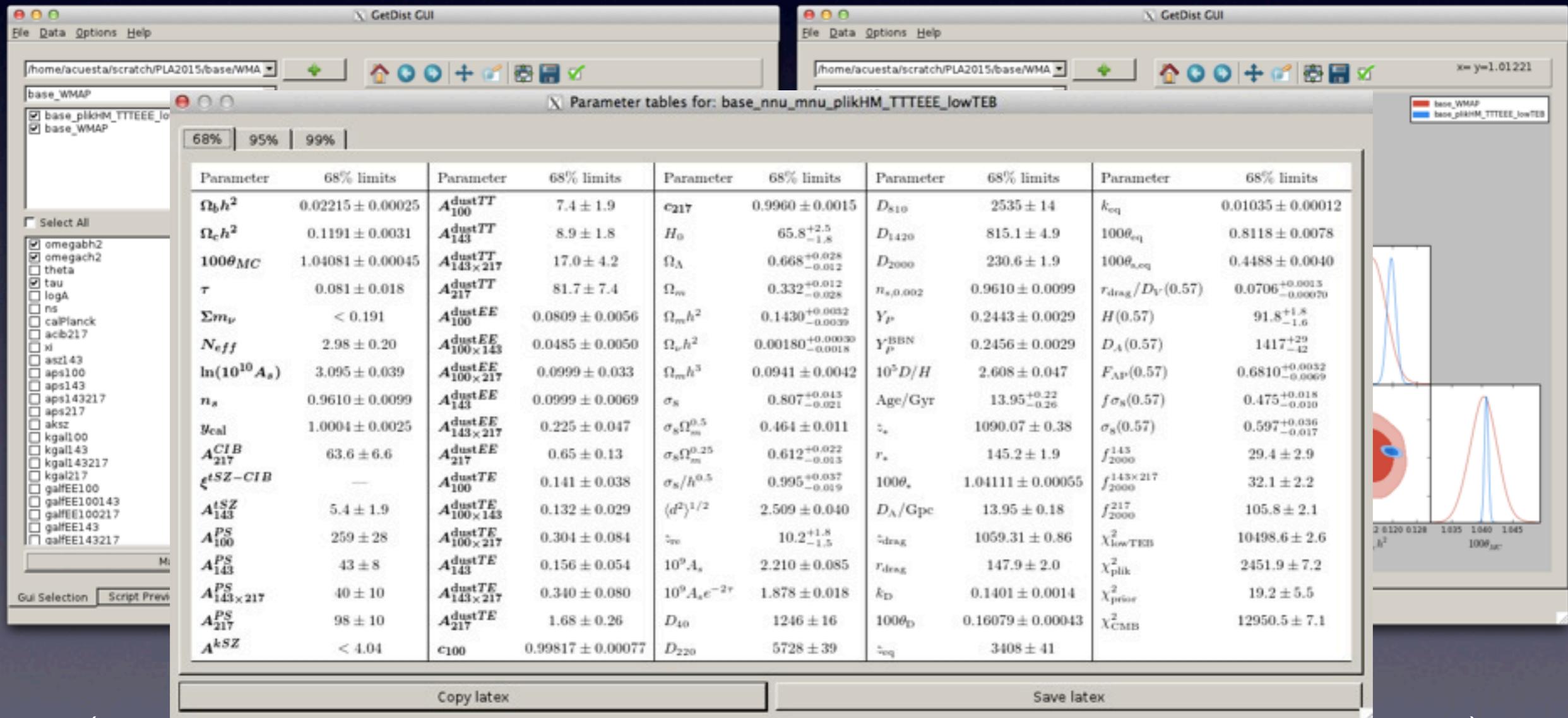
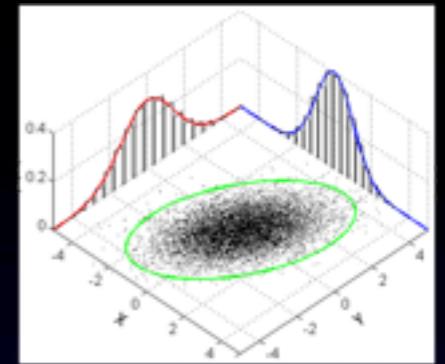
2D Plot

(correlations, degeneracy directions)

Step 5: Analysis of the results

Once the chains have converged, we can make different types of summary **figures or tables** to convey the resulting information

This requires **marginalization**: integrate the d -dim PDF over all other parameters



Planck's public chains

- Planck released their **full analysis** (done with CosmoMC) for a comprehensive **combination of cosmological models and datasets**
- They can be downloaded at <http://pla.esac.esa.int/pla> (Go to “Cosmology” and then click on “Cosmological parameters”)
- The **full grid** shown in Planck 2015 Paper XIII can be found in the file `COM_CosmoParams_base-plikHM_R2.00.tar.gz` (3.6GB)
- More information can be found at the **ESA/Planck wiki**: http://wiki.cosmos.esa.int/planckpla2015/index.php/Cosmological_Parameters

The image shows two screenshots of the Planck Legacy Archive website. The left screenshot displays the 'PR2 - 2015 COSMOLOGY PRODUCTS' page, featuring a navigation bar with links for 'Cosmological parameters', 'CMB angular power spectra', 'Likelihood', 'Lensing products', and 'Noise covariance matrices'. Below this is a 'RESULTS' section with a table header for 'Description', 'File name', and 'Size'. A note at the bottom states: 'Planck Likelihood Code. It provides C and Fourier binning that allows users to compute'.

The right screenshot shows a detailed view of the 'RESULTS' section titled 'Cosmology products (5)'. The table lists five files:

File name	Size	Product type	Release
COM_CosmoParams_base-lensonly_R2.00.tar.gz	13.5 MB	Cosmological Parameters	PR2
COM_CosmoParams_base-plikHM_R2.00.tar.gz	415.3 MB	Cosmological Parameters	PR2
COM_CosmoParams_base-plikHM-TT-lowTEB_R2.00.tar.gz	46.7 MB	Cosmological Parameters	PR2
COM_CosmoParams_base-r-plikHM-BKP_R2.00.tar.gz	173.4 MB	Cosmological Parameters	PR2
COM_CosmoParams_fullGrid_R2.00.tar	3.6 GB	Cosmological Parameters	PR2

Text labels 'Likelihood Code' and 'Grid of chains' are overlaid on the bottom left and right respectively.

Introduction to MontePython



The CLASSY interface: CLASS Python wrapper

- Since MontePython and CLASS are written in **different languages** (Python and C) MontePython needs a **wrapper to communicate with CLASS** at each MCMC step (there is also a pycamb interface to CAMB)
- Much like Montepython does to run, **this wrapper can also be used stand-alone**:

Terminal: write `your_parameter_file.ini`

```
h = 0.6774
omega_b = 0.02230
Omega_cdm = 0.2603
Omega_fld = 0
Omega_smg = 0 #GR in hi_class
background_verbose = 1 #info
output = tCl,mPk #what to compute
write background = y
root = output/your_model_ #future files
```

Python: write a dictionary

```
params = {
    "h": 0.6774,
    "omega_b": 0.02230,
    "Omega_cdm": 0.2603,
    "Omega_fld": 0,
    "Omega_smg": 0, #GR in hi_class
    "background_verbose": 1, #info
    "output": "tCl,mPk" #observables
}
```

Terminal:

From the to the base directory run

```
./class your_parameter_file.ini
```

(plus an optional `.pre` precision file)

Your output will be ready in the `root` address.

Python:

```
from classy import Class
cosmos = Class() #create universe
cosmos.set(params) #feed params to cosmos
cosmos.compute() #duh...
...               #play with the output
cosmo.struct_cleanup() #free memory
cosmo.empty() #start over
```

(from Miguel Zumalacarregui's notes)

- To check if your installation works, open a python terminal and type:
`import classy`
- More details can be found at: https://github.com/lesgourg/class_public/wiki/Python-wrapper

Running Montepython

The procedure is well described in the [official documentation website](#):
<http://monte-python.readthedocs.io/en/latest/example.html>

Monte Python 2.2.0 documentation »

Previous topic

Getting Started

Next topic

Using MultiNest with Monte Python

Quick search

Go

Enter search terms or a module, class or function name.

Example of a complete work session

I just downloaded and installed *Monte Python*, read the previous pages, and I wish to launch and analyse my first run.

I can first create a few folders in order to keep my `montepython` directory tidy in the future. I do a

```
$ mkdir chains  
for storing all my chains
```

\$ mkdir chains/planck
if the first run I want to launch is based on the fake planck likelihood proposed in the `example.param` file

```
$ mkdir input  
for storing all my input files
```

\$ mkdir scripts
for storing all my scripts for running the code in batch mode

I then copy `example.param` in my input folder, with a name of my choice, e.g. `lcdm.param`, and edit it if needed:

```
$ cp example.param input/lcdm.param
```

I then launch a short chain with

```
$ montepython/Montepython.py run -p input/lcdm.param -o chains/planck/lcdm -N 5
```

MontePython input file

The input file is where we specify:

- the experiments (cosmological datasets) we want to use (in folder montepython/likelihoods)
- the values of the parameters that we want fixed (“cosmo_arguments”)
- the parameters we want to measure (“cosmo”)
- the parameters required to be marginalized by some likelihoods (“nuisance”)
- the extra parameters we want to be computed and written to file (“derived”)
- other defaults like number of steps (override if specified), etc.

```
data.experiments = ['experiment1', 'experiment2', ...]

data.parameters['cosmo_name']      = [mean, min, max, sigma, scale, 'cosmo']
...
data.parameters['nuisance_name']   = [mean, min, max, sigma, scale, 'nuisance']
...
data.parameters['cosmo_name']      = [mean, min, max, sigma, scale, 'derived']
...
data.cosmo_arguments['cosmo_name'] = value

data.N = 10
data.write_step = 5
```

Running chains with MontePython run

- Each instance of MontePython run will run only one chain, unless you run the command with MPI (requires the python module mpi4py)
- You can specify: an input file, an output folder, the number of steps (the code will not stop at convergence), the best fit point and a proposal covariance matrix (these two are needed for faster convergence, but you will have these files only if you did a previous shorter run or a similar run)
- You can also specify a step size (default -f 2.4, target should be ~20% to 25% acceptance rate), sampling method (e.g. MultiNest), and not use fast/slow sampling
- You can update the proposal covariance matrix every 500 steps with the option --update 500 (it will use all chains to recompute it, even without MPI)
- Example: if you want to run 4 chains with 10^5 points each:

```
for n $(seq 1 4); do python montepython/Montepython run \
    -o output_directory -N 100000 [other_options]; done
```

The first run creates a log.param file in the output folder, so that if you (mistakenly) try to run a chain in that folder with a different input file, the old configuration overrides the input file (prevents mix-ups)

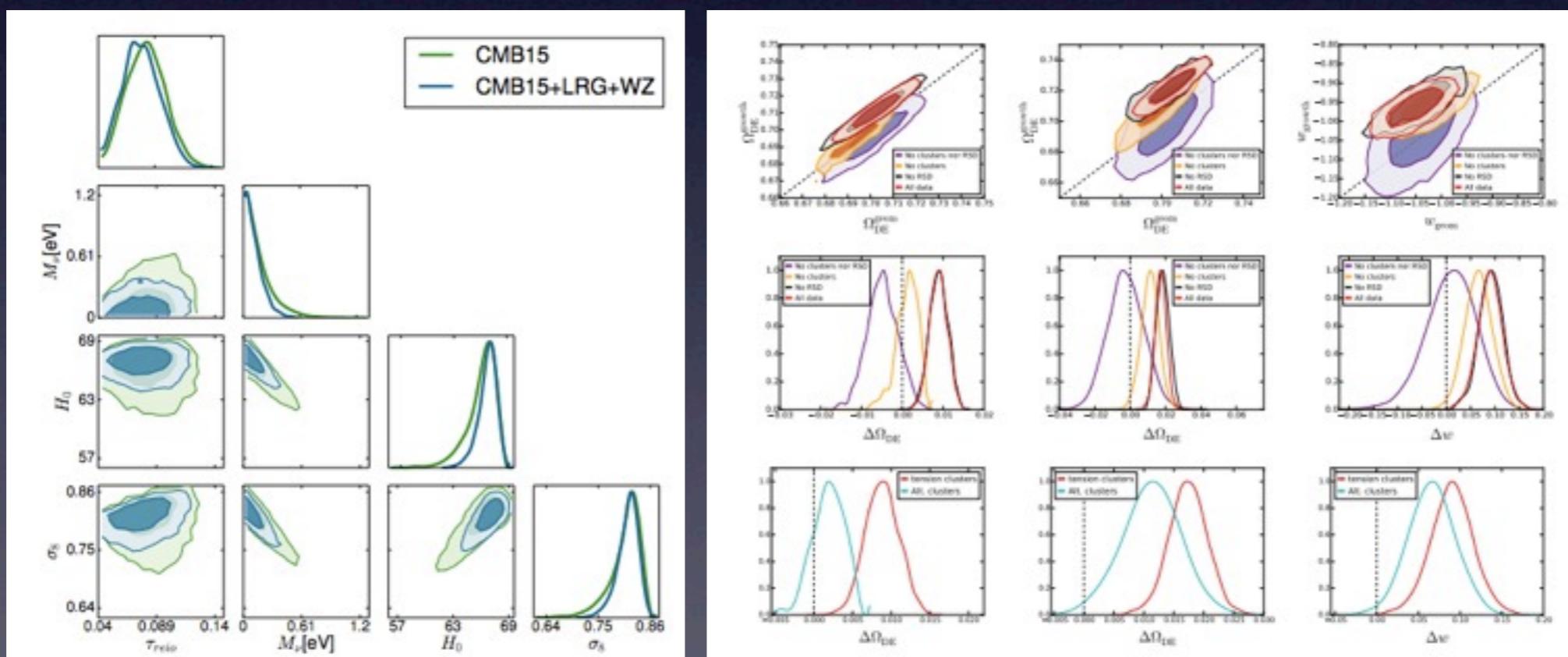
Plotting your results with MontePython info

- Customize your plots!! (use the **--extra** option if you have saved these options into a file)
 - Lots of **layout options**: font size, legend style, line width, no-mean, ...
 - Change the **number of bins** for credible intervals (less bins = less resolution)
 - Do Gaussian **smoothing** or increase **interpolation** (for smoother plots)
 - To do **operations with the parameters**:
`info.redefine = {'new_param': '5*old_param+10'}`
 - To **rename a parameter** in the output plots:
`info.to_change={'old_name': 'new_name'}`
 - To plot and compute credible intervals for **only some specific parameters**:
`info.to_plot=['new_param', 'new_name', 'old_param2', ...]`

Plotting your results with MontePython info

To compare the constraints on two different models from a fixed dataset
(or on a single cosmological model from two different datasets)

```
python montepython/MontePython.py info experiment_1/ experiment_2/ ...
```



arXiv:1511.05983

arXiv:1511.03049

Creating tables from MontePython info

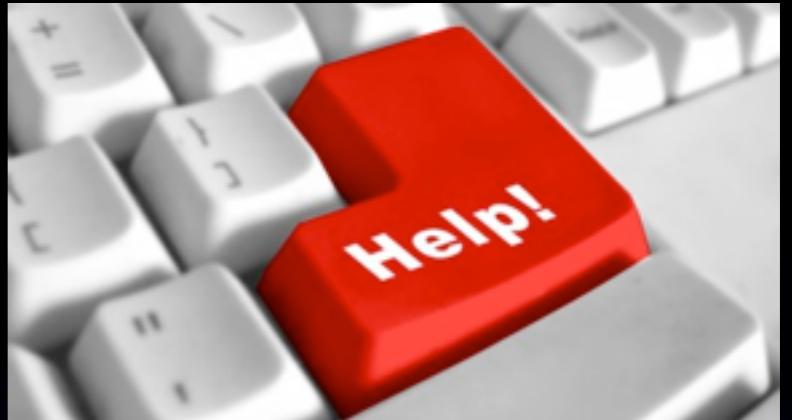
You can also use the output *tex files to compile *LaTeX* tables, including:

- All the parameters ('cosmo', 'derived', 'nuisance', ...) especified in to_plot
- Their best-fit values and their mean values
- Their 1-sigma and 2-sigma credible intervals (sigma and 95%)
- Other info like the maximum likelihood found and minimum value of χ^2

Param	best-fit	mean $\pm\sigma$	95% lower	95% upper
Ω_k	0.1984	$0.1619^{+0.26}_{-0.14}$	-0.2234	0.4994
Ω_m	0.2115	$0.2315^{+0.072}_{-0.1}$	0.07255	0.4076
Ω_Λ	0.59	$0.6065^{+0.098}_{-0.16}$	0.3697	0.8639

$-\ln \mathcal{L}_{\min} = 341.105$, minimum $\chi^2 = 682.2$

HELP!!!



- MontePython is well documented:
<http://monte-python.readthedocs.io/en/latest/>
- Quick access to all the options in each mode:
 - The options to run chains are:
`python montepython/MontePython.py run --help`
 - The options to plot and analyze chains are:
`python montepython/MontePython.py info --help`

Introduction to CosmoMC



Running CosmoMC

- Running a set of N chains is as simple as executing the compiled code `./cosmomc` with an input file `params.ini` in the command line (but you need to compile it first! Needs Intel Fortran 14 or GCC 6)
- CosmoMC also offers a simple way to generate job scripts and submit them to run the code in a computing cluster (e.g. Torque, MOAB, SLURM)

```
python python/runMPI.py --nodes 1 --chainsPerNode 8 --coresPerNode 16 --mem_per_node 40000  
--walltime 480:00:00 --job_template job_script_hipatia --program ./cosmomc --queue batch ow0wacdmabaosn.ini
```

- When running in a cluster, each chain will be parallelized (using OpenMP) over a number of processors equal to `coresPerNode` over `chainsPerNode`. The communication between chains will be done via MPI communication.

CosmoMC input file

The input file for CosmoMC is named `params.ini` where we can set up:

```
#Folder where files (chains, checkpoints, etc.) are stored
root_dir = chains/

#Root name for files produced
file_root=planck15dr12

#action= 0 runs chains, 1 importance samples, 2 minimizes
#use action=4 just to quickly test likelihoods
action = 0

#general settings
#Bicep-Keck-Planck, varying cosmological parameters
#DEFAULT(batch2/BKPlanck.ini)

#Planck 2015, default just include native likelihoods (others require clik)
DEFAULT(batch2/plik_dx11dr2_HM_v18_TTTEEE.ini)
DEFAULT(batch2/lowTEB.ini)
#DEFAULT(batch2/lowl.ini)
#DEFAULT(batch2/lensing.ini)

#Other Likelihoods
#DEFAULT(batch2/BA0new.ini)
#DEFAULT(batch2/WiggleZ_MPK.ini)
#DEFAULT(batch2/MPK.ini)
#DEFAULT(batch2/WL.ini)

#to vary parameters set param[name]= center, min, max, start width, propose width
#param[mnu] = 0 0 0 0 0

param[omegak] = 0 -0.1 0.1 0.005 0.005
param[w] = -1 -3 1 0.05 0.05
param[wa] = 0 -3 3 0.3 0.3

prior[omegabh2] = 0.0223 0.0009
```

Output folder and chain file names

Execution mode (0=run chains)

Likelihoods
(datasets)
you want
to include
(each one
is in turn a
settings file)

Planck2015 highL temperature+polarization
Planck2015 lowL temperature+polarization
Planck2015 lowL temperature only
Planck2015 CMB lensing

BAO from BOSS+MGS+6dF(+WiggleZ)
WiggleZ galaxy power spectrum
SDSS galaxy power spectrum
CFHTLens weak lensing

Also, SNe from JLA, etc.

Model parameters:
Values for fixed ones
and ranges for those
to be measured

Easily implement Gaussian priors

GetDist: Analysis of CosmoMC chains

CosmoMC's traditional (text-only) analysis tool is **GetDist**. Besides outputting info to the screen, it creates files with **parameter bounds** (margestats), **covariance matrices**, and **python scripts** to generate plots with matplotlib (in the `plot_data` folder). The analysis settings can be modified in the `distparams.ini` file

```
[acuesta@nova0 cosmomc2015jul]$ ./getdist distparams.ini chains/lcdmbaosn
skipped unused params: omegak mnu nnu yhe Alens nrun r r02
reading chains/lcdmbaosn_1.txt
reading chains/lcdmbaosn_2.txt
reading chains/lcdmbaosn_3.txt
reading chains/lcdmbaosn_4.txt
reading chains/lcdmbaosn_5.txt
reading chains/lcdmbaosn_6.txt
reading chains/lcdmbaosn_7.txt
reading chains/lcdmbaosn_8.txt
Number of chains used =          8
var(mean)/mean(var), remaining chains, worst e-value: R-1 =      0.00666
RL: Thin for Markov:            30
RL: Thin for indep samples:    31
RL: Estimated burn in steps: 120 (50 rows)
mean input multiplicity = 2.39603247549020
using      65280 rows, processing      92 parameters
Approx indep samples:           5046
Best fit sample -log(Like) =   6822.39800000000
mean(-Ln(like)) =   6835.30173439549
-Ln(mean like) =   6830.18592659466
Warning: sharp edge in parameter chi2_JLA - check limits[chi2_JLA] or limits86
Warning: sharp edge in parameter chi2_6DF - check limits[chi2_6DF] or limits87
```

Syntax: `./getdist distparams.ini chains/root`
parameters defined in `distparams.ini` but not in the chains

number of chains read (check that these are all your chains)

MCMC Convergence status

number of parameters (cosmo+nuisance+der)

warnings about non-converged parameters

GetDist: Analysis of CosmoMC chains

```
Marginalized limits: 0.68; 0.95; 0.99

parameter    mean      sddev     lower1     upper1     limit1    lower2     upper2     limit2    lower3     upper3     limit3
omegabh2   0.2227594E-01 0.1409804E-03 0.2213470E-01 0.2241652E-01 two      0.2200101E-01 0.2255144E-01 two      0.2191544E-01 0.2264190E-01 two      \Omega_b h^2
omegach2   0.1192956E+00 0.1068506E-02 0.1182545E+00 0.1203491E+00 two      0.1171635E+00 0.1213484E+00 two      0.1164438E+00 0.1221176E+00 two      \Omega_c h^2
theta       0.1040831E+01 0.2995374E-03 0.1040534E+01 0.1041130E+01 two      0.1040247E+01 0.1041420E+01 two      0.1040057E+01 0.1041614E+01 two      100\theta_{MC}
tau        0.8200741E-01 0.1643758E-01 0.6557802E-01 0.9862151E-01 two      0.4956239E-01 0.1138723E+00 two      0.4020595E-01 0.1240224E+00 two      \tau
logA        0.3097935E+01 0.3246072E-01 0.3065291E+01 0.3130707E+01 two      0.3034228E+01 0.3160888E+01 two      0.3013767E+01 0.3180444E+01 two      {\rm ln}](10^{10} A_s)
ns          0.9658500E+00 0.4063676E-02 0.9618321E+00 0.9698272E+00 two      0.9579955E+00 0.9740912E+00 two      0.9555469E+00 0.9768459E+00 two      n_s
calPlanck  0.1000465E+01 0.2516048E-02 0.9979804E+00 0.1002947E+01 two      0.9954956E+00 0.1005439E+01 two      0.9939398E+00 0.1007128E+01 two      y_{\rm cal}
acib217    0.6359261E+02 0.6577819E+01 0.5697960E+02 0.7011801E+02 two      0.5088221E+02 0.7649609E+02 two      0.4702681E+02 0.8075525E+02 two      A^{CIB}_{(217)}
xi          0.5220534E+00 0.2840235E+00 0.0000000E+00 0.1000000E+01 none    0.0000000E+00 0.1000000E+01 none    0.0000000E+00 0.1000000E+01 none    \xi^{tSZ-CIB}
asz143     0.5399273E+01 0.1871846E+01 0.3556131E+01 0.7442447E+01 two      0.1661532E+01 0.8952850E+01 two      0.7504497E+00 0.9658879E+01 two      A^{tSZ}_{(143)}
aps100     0.2590954E+03 0.2746669E+02 0.2316349E+03 0.2864320E+03 two      0.2053967E+03 0.3134392E+03 two      0.1890985E+03 0.3311659E+03 two      A^{PS}_{(100)}
aps143     0.4325286E+02 0.7687977E+01 0.3552913E+02 0.5095578E+02 two      0.2797599E+02 0.5791646E+02 two      0.2349064E+02 0.6198630E+02 two      A^{PS}_{(143)}
aps143217  0.4026165E+02 0.9948208E+01 0.3006222E+02 0.5063956E+02 two      0.2158706E+02 0.5957445E+02 two      0.1655117E+02 0.6414078E+02 two      A^{PS}_{(143\times217)}
aps217     0.9815534E+02 0.1100181E+02 0.8722243E+02 0.1092326E+03 two      0.7626975E+02 0.1192058E+03 two      0.6946944E+02 0.1255847E+03 two      A^{PS}_{(217)}
aksz       0.3160785E+01 0.2332050E+01 0.0000000E+00 0.4034485E+01 >    0.0000000E+00 0.7749367E+01 >    0.0000000E+00 0.1000000E+02 none    A^{kSZ}

[acuesta@nova0 cosmomc2015jul]$ ./getdist distparams.ini chains/lcdmbaosn
```

Syntax: ./getdist distparams.ini chains/root
 skipped unused params: omegak mnu nnu yhe Alens nrun r r02 parameters defined in distparams.ini but not in the chains

number of chains read (check that these are all your chains)

MCMC Convergence status

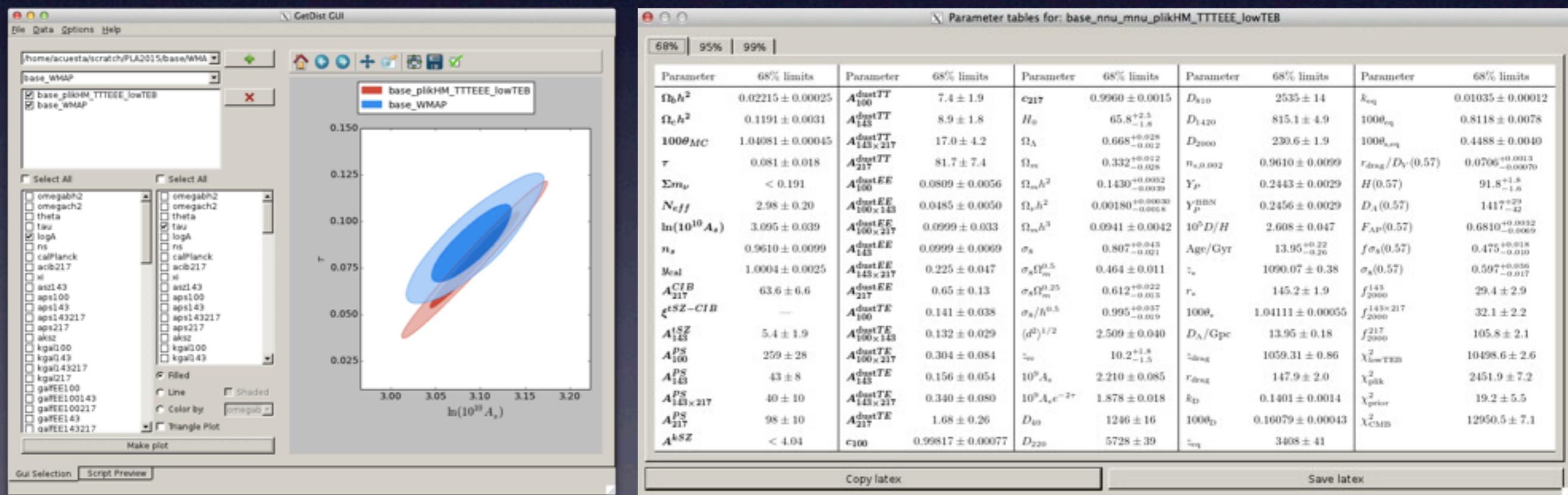
number of parameters (cosmo+nuisance+der)

warnings about non-converged parameters

GetDist GUI:

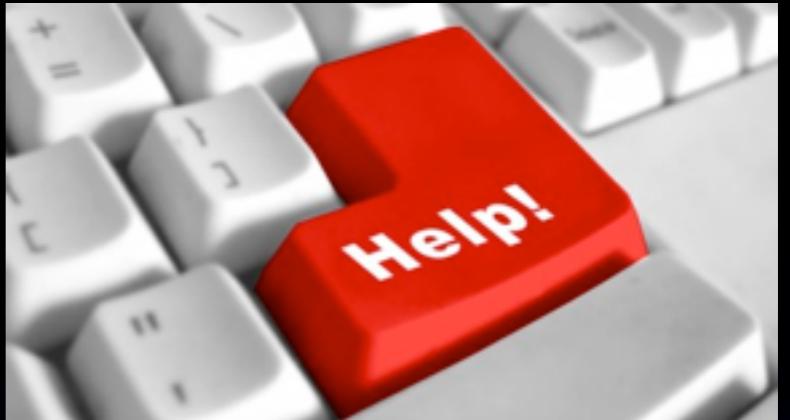
MCMC graphical analysis

- Getdist's graphical interface allows the interactive analysis of CosmoMC but also MontePython chains, making it easy to visually inspect the results
- It has an option to create tables with parameter bounds and export to LaTeX



More examples are shown at http://getdist.readthedocs.org/en/latest/plot_gallery.html

HELP!!!



- CosmoMC has a complete README website:
<http://cosmologist.info/cosmomc/readme.html>
- It contains individual pages for many topics:
Planck likelihood, Python modules, GetDist GUI, running grids of models/datasets,...
- See also the CosmoCoffee forum:
<http://cosmocoffee.info/>

Thank you

ajcuesta@uco.es

Supplementary slides

Structure of a chain file

- A **chain file** is just a **text file**, you can open it with any editor or manipulate it through any **python script** (or the way that is most convenient for you)
 - Each **row** is a **MCMC step**. Each **column** is each one of the **parameters** in this order: cosmological, nuisance, and derived. Check the *** .paramnames** file to find out the ordering
 - The first two columns are the **multiplicity** (or number of steps spent in that point) and the **-log(likelihood)**, or “**loglike**”, which is just 0.5 times the total **χ^2** value.

mult & loglike

LCDM parameters

0.1000000E+01	0.7588278E+04	0.2217817E-01	0.1210163E+00	0.1040737E+01	0.9666598E-01	0.3097666E+01	0.9647925E+00	0.1001027E+01	0.1340781E+00	0.3073859E+01	0.5418683E+02	0.16258	
50E+00	0.2454537E+01	0.2765384E+03	0.4889315E+02	0.2765156E+02	0.1165507E+03	0.9301511E+00	0.4311650E+01	0.8843642E+01	0.2329630E+02	0.7097032E+02	0.6636701E-01	0.7642297E-01	
1.240745E+00	0.1117157E+00	0.3490495E+00	0.5666420E+00	0.1334311E+00	0.1757612E+00	0.3807932E+00	0.2035620E+00	0.6760326E+00	0.1314555E+01	0.998682E+00	0.9941414E+00	0.6677979E+00	
0.6	0.6774565E+00	0.3225435E+00	0.1438396E+00	0.6451439E-03	0.9605576E-01	0.8370669E+00	0.4753947E+00	0.6308226E+00	0.1824325E+01	0.2523461E+01	0.1168064E+02	0.2214488E+01	0.182
5188E+01	0.1206639E+04	0.5520434E+04	0.2454898E+04	0.7891323E+03	0.2233522E+03	0.9647925E+00	0.2453066E+00	0.2466328E+00	0.2627639E+01	0.1382655E+02	0.1090254E+04	0.1443158E+03	
0.1040935E+01	0.1386495E+02	0.1059551E+04	0.1470385E+03	0.1407766E+00	0.1609642E+00	0.3421912E+04	0.1044440E+01	0.8891325E+00	0.4473433E+00	0.1147113E+00	0.7956448E+02	0.1001239	
E+04	0.3507600E+00	0.7098143E-01	0.9268106E+02	0.1398548E+04	0.6788807E+00	0.4943056E+00	0.7053466E+00	0.4894761E+00	0.6200091E+00	0.3953857E+04	0.1049665E+05	0.6969622E+03	0.2
200932E+00	0.5372448E+00	0.7533748E+01	0.7409717E+00	0.2050706E+02	0.1445050E+05	0.9032058E+01							
0.2000000E+01	0.7432273E+04	0.2205772E-01	0.1223525E+00	0.1040409E+01	0.9243788E-01	0.3091427E+01	0.9585111E+00	0.9999859E+00	0.1381403E+00	0.3085730E+01	0.6719501E+02	0.16074	
88E+01	0.2326204E+00	0.3234647E+03	0.4471673E+02	0.1465753E+02	0.8721921E+02	0.5519028E+01	0.4840853E+01	0.6603848E+01	0.2843156E+02	0.8216884E+02	0.6988734E+01	0.8529992E+01	
1.473713E+00	0.1212628E+00	0.4138422E+00	0.5325758E+00	0.2066227E+00	0.2024037E+00	0.3101168E+00	0.2384299E+00	0.6598682E+00	0.1248762E+01	0.9968187E+00	0.9981919E+00	0.6611572E+00	
02	0.6681633E+00	0.3318367E+00	0.1459554E+00	0.6451439E-03	0.9590442E-01	0.8370717E+00	0.4821974E+00	0.6353218E+00	0.1029462E+01	0.2543836E+01	0.1138763E+02	0.2208845E+01	0.182
9359E+01	0.1218436E+04	0.5524315E+04	0.2451456E+04	0.7855802E+03	0.2219278E+03	0.9585111E+00	0.24552473E+00	0.2465732E+00	0.2650759E+01	0.1385349E+02	0.1090526E+04	0.1440652E+03	
0.1040627E+01	0.1384408E+02	0.1059361E+04	0.1468229E+03	0.1409909E+00	0.1610448E+00	0.3459975E+04	0.1053267E+01	0.8034788E+00	0.4444655E+00	0.1137244E+00	0.7910798E+02	0.1009162	
E+04	0.3515068E+00	0.7049756E+01	0.9239276E+02	0.1407670E+04	0.6811109E+00	0.4986407E+00	0.7037144E+00	0.4916913E+00	0.6178100E+00	0.3612763E+04	0.1049756E+05	0.6964181E+03	0.4
869998E+00	0.1972903E+00	0.1102911E+02	0.1891880E+01	0.4420137E+02	0.1411032E+05	0.1360520E+02							
0.4000000E+01	0.7284871E+04	0.2198426E+01	0.1221917E+00	0.1040366E+01	0.1098112E+00	0.3125897E+01	0.9607862E+00	0.1000165E+01	0.1525372E+00	0.3207994E+01	0.8596888E+02	0.10553	
51E+00	0.3467409E+01	0.2893997E+03	0.4493034E+02	0.2553329E+02	0.7202911E+02	0.6357391E+01	0.12509035E+02	0.8057744E+01	0.2133231E+02	0.6452248E+02	0.6871896E-01	0.7897106E-01	
1.537515E+00	0.1198039E+00	0.4199892E+00	0.5379834E+00	0.1532049E+00	0.1571962E+00	0.4808829E+00	0.2579597E+00	0.7759683E+00	0.1462243E+01	0.9989428E+00	0.6600619E+00		
02	0.6685035E+00	0.3314965E+00	0.1448211E+00	0.6451439E-03	0.9572121E-01	0.85206334E+00	0.4905817E+00	0.6465344E+00	0.1048054E+01	0.2582354E+01	0.1288380E+02	0.2278031E+01	0.182
8854E+01	0.1221446E+04	0.5509919E+04	0.2451231E+04	0.7856988E+03	0.2223143E+03	0.9607862E+00	0.2452129E+00	0.2465387E+00	0.2665088E+01	0.1386161E+02	0.1090605E+04	0.1441621E+03	
0.1040593E+01	0.1385383E+02	0.1059208E+04	0.1469439E+03	0.1407185E+00	0.1611494E+00	0.3445374E+04	0.1051558E+01	0.8842171E+00	0.4449052E+00	0.1137728E+00	0.7907232E+02	0.1009537	
E+04	0.35154796E+00	0.7052297E+00	0.9234238E+02	0.1408265E+04	0.6810272E+00	0.5074146E+00	0.7163785E+00	0.5004196E+00	0.62809566E+00	0.3312784E+04	0.1050069E+05	0.7015079E+03	0.4
721827E+00	0.2234986E+00	0.1081429E+02	0.1822413E+01	0.4143067E+02	0.1381347E+05	0.1333238E+02							

1 MCMC step

List of CosmoMC's valid parameter names

You can see a complete list in the file `paramnames/params_CMB.paramnames` and also in Planck's documentation http://wiki.cosmos.esa.int/planckpla2015/images/b/b9/Parameter_tag_definitions_2015.pdf

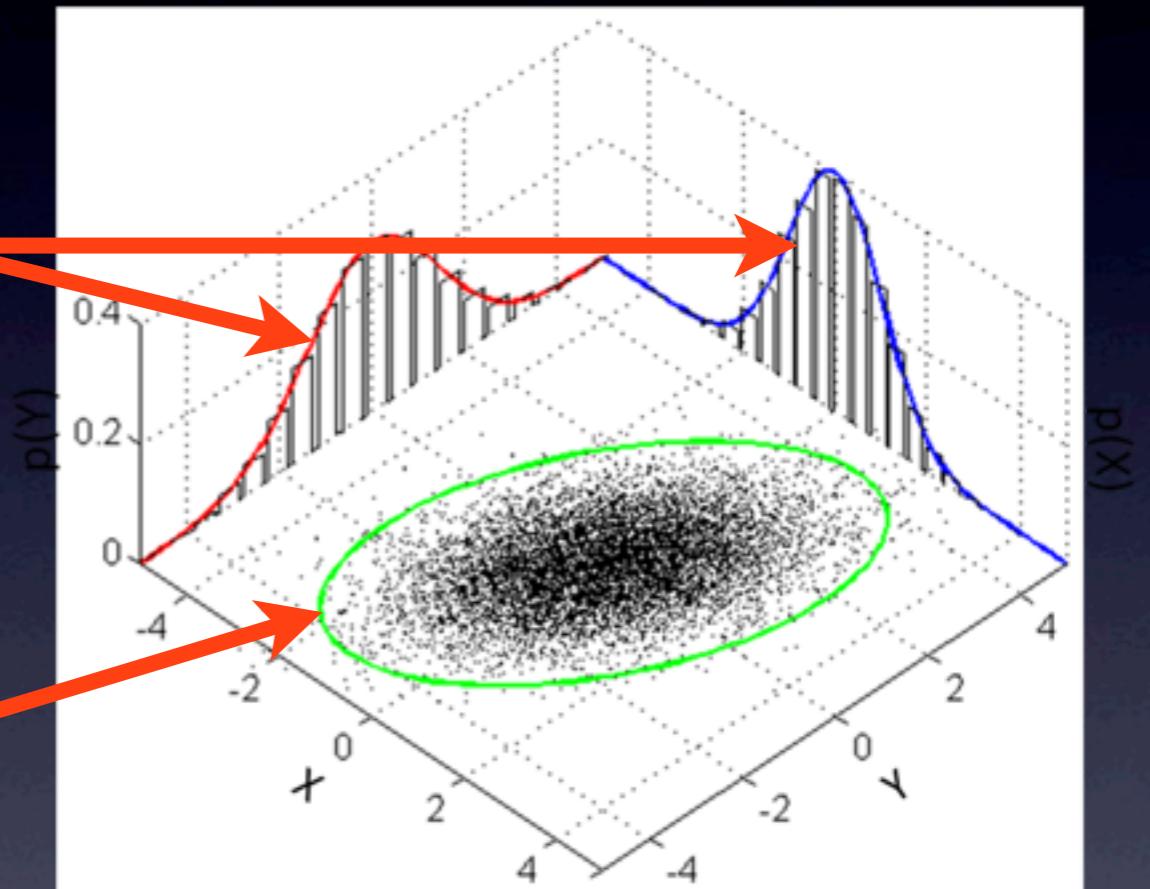
```
omegabh2      \Omega_b h^2      #physical baryon density
omegach2      \Omega_c h^2      #physical CDM matter density
theta          100\theta_{MC} #100 times the ratio of the angular diameter distance to the LSS sound horizon
tau            \tau
omegak         \Omega_K
mnu            \Sigma m_\nu   #sum of physical masses of standard neutrinos
meffsterile   m_{\nu,\{\rm{sterile}\}}^{\rm{eff}} #effective mass of sterile neutrino, \approx omeganuh2*94
w              w              #equation of state parameter for scalar field dark energy today
wa             w_a            #w_a variation
nnu            N_{eff}        #effective number of neutrinos (only clearly defined for massless)
H0*            H_0            #hubble parameter is H0 km/s/Mpc
omegal*        \Omega_\Lambda
omegam*        \Omega_m
```

Parameter Covariance Matrix

- Choosing a good **covariance matrix** is important because it **HELPS** the chain to **converge** much faster

- It gives you the optimal **sampling step size** in each direction of the parameter space (**diagonal elements** of the matrix)

- But also, if the parameters are correlated, it also gives the **direction** where to move when we change one of the parameters (**off-diagonal elements**)



$$C_{ij} = \begin{pmatrix} \sigma_{\theta_1 \theta_1} & \sigma_{\theta_1 \theta_2} \\ \sigma_{\theta_1 \theta_2} & \sigma_{\theta_2 \theta_2} \end{pmatrix}$$

Covariance matrix

$$\sigma_{\theta_i \theta_i} = \sigma_{\theta_i}^2$$

$$R_{ij} = \frac{C_{ij}}{\sigma_{\theta_i} \sigma_{\theta_j}} = \begin{pmatrix} 1 & \rho_{\theta_1 \theta_2} \\ \rho_{\theta_1 \theta_2} & 1 \end{pmatrix}$$

Correlation matrix

Modifying a (simple) CosmoMC likelihood

- Let's make a copy of this likelihood to batch2/HSTnew.ini because we want to use the new 2.4% determination of H_0 by Riess et al. 2016 <http://arxiv.org/abs/1604.01424>

```
# Riess et al (2011) value of H0 = 73.8 +/- 2.4 km/s/Mpc
# Riess et al: 1103.2976
Hubble_zeff = 0.04
Hubble_angconversion = 11425.8
#angconversion converts inverse of the angular diameter distance at z = zeff to H0
#for the fiducial cosmology (omega_k = 0, omega_lambda = 0.7, w = -1)
#likelihood is in terms of inverse of the angular diameter distance, so includes the tiny cosmological
#dependence of the measurement (primarily on w) correctly.
Hubble_H0 = 73.8 ← 73.24
Hubble_H0_err = 2.4 ← 1.74
Hubble_name = HST ← HSTnew
```

$$P(H_0) \propto e^{-0.5 \left(\frac{H_0 - 73.24}{1.74} \right)^2}$$

- Now you can INCLUDE in your parameter file this new likelihood batch2/HSTnew.ini