Introduction to COSMOMC

IFT SCHOOL ON COSMOLOGY TOOLS

Madrid, 13-17 March 2017

José Alberto Rubiño Martín

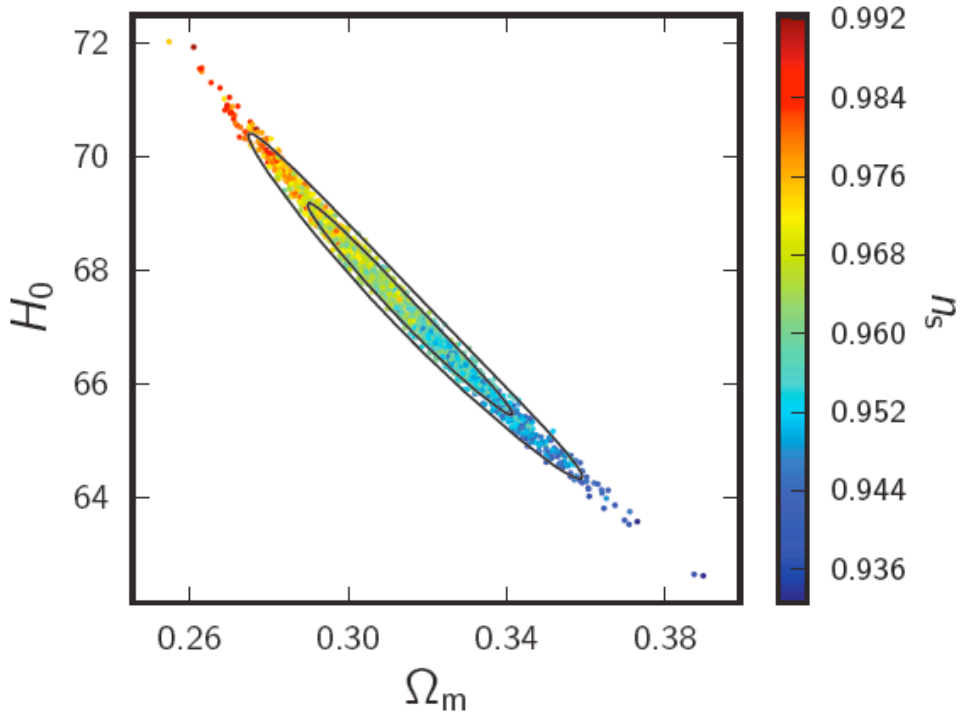
Marcos Pellejero Ibáñez

Emails: jalberto@iac.es, mpi@iac.es

Web: http://www.iac.es/galeria/jalberto

Hydra: /home/prof7/COSMOMC_material/

COSMOlogical Monte-Carlo



I. ***Introduction. Markov Chains and COSMOMC code.***

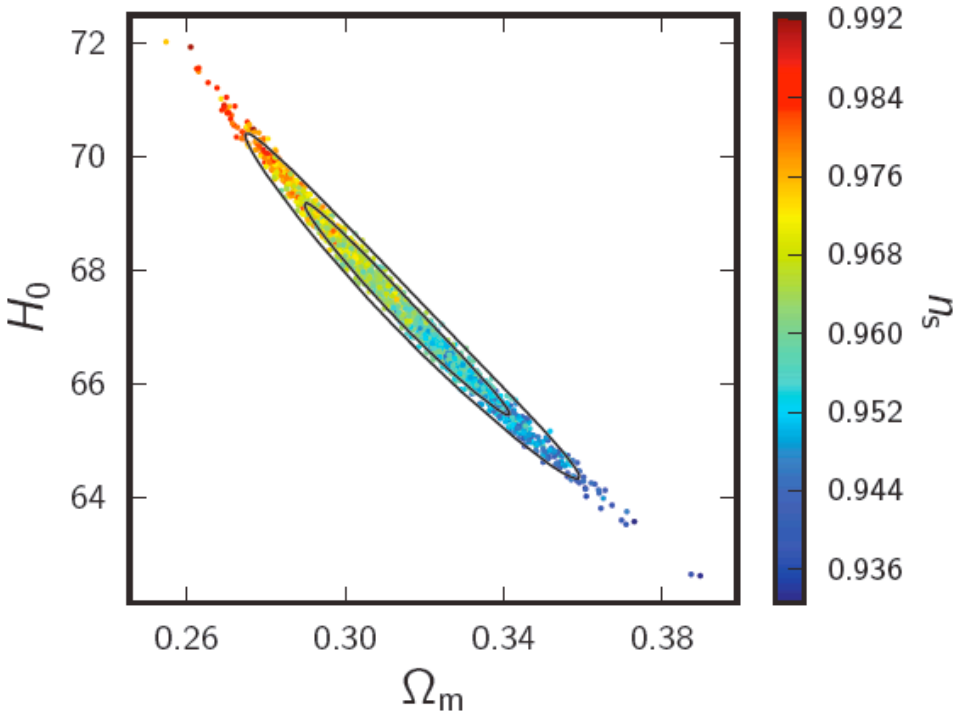
II. ***Structure of the code. Source files.***

III. ***How to run the code.***

- *I.* ***Parameterizations.***
- *II.* ***WMAP and Planck likelihood.***
- *III.* ***Forecasting***

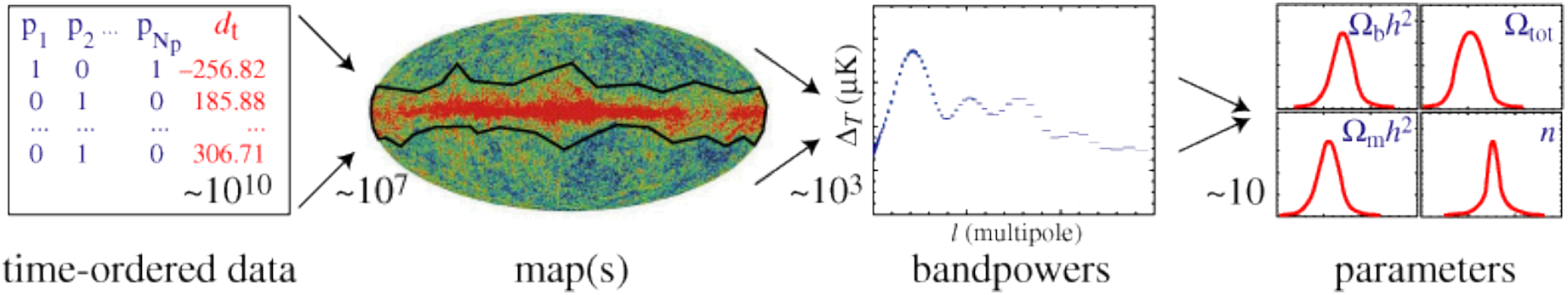
IV. ***Analysing the output. Convergence of the chains.***

COSMOlogical Monte-Carlo



- **CosmoMC** is a Fortran 2008 Markov-Chain Monte-Carlo (MCMC) engine for exploring cosmological parameter space, together with code for analysing Monte-Carlo samples and importance sampling. There are two programs supplied: **cosmomc** and **getdist**.
- Uses **CAMB** for the Boltzmann solver (http://camb.info)
- **Author**: Anthony Lewis.
- Code and documentation: http://cosmologist.info/cosmomc
- Specific forum for questions/tips and discussion: http://cosmocoffee.info/

Data compression in Cosmology



Statistical approach: Bayesian analysis

We have to deal with the inverse problem of probability. We use a Bayesian approach, which is based on the Bayes´ theorem:

$$Prob(\vec{p}|\vec{x}) = \frac{Prob(\vec{x}|\vec{p})\,Prob(\vec{p})}{Prob(\vec{x})} = \frac{Prob(\vec{x}|\vec{p})}{\int Prob(\vec{x}|\vec{p})d\vec{p}}$$

Cosmological parameter estimation

❖ The cosmological parameter estimation is done comparing the observed power spectra with theoretical ones, by means of the **likelihood function** $\mathcal{L}$.

$$\mathcal{L}(data|\{\vec{p}\}) \propto \mathcal{L}(data|C_\ell\{\vec{p}\})$$

❖The main difficulty now is not the size of the input dataset, but the dimensionality of the parameter space.

$$\chi^2 = \sum_{B,B'} (C_{B,o} - C_{B,p})(V)^{-1}_{BB'}(C_{B',o} - C_{B',p})$$

❖ **Dimensionality of the parameter space**: {Ω_b, Ω_m, Ω_Λ, Ω_v, H_0, τ, n, n_T, A_S, A_T,…, calibration, beam uncert.}

❖ **Posterior (maximum likelihood) evaluation**:

- ❖ Grid method
- ❖ Monte-Carlo Markov Chains

Example of application: grid method

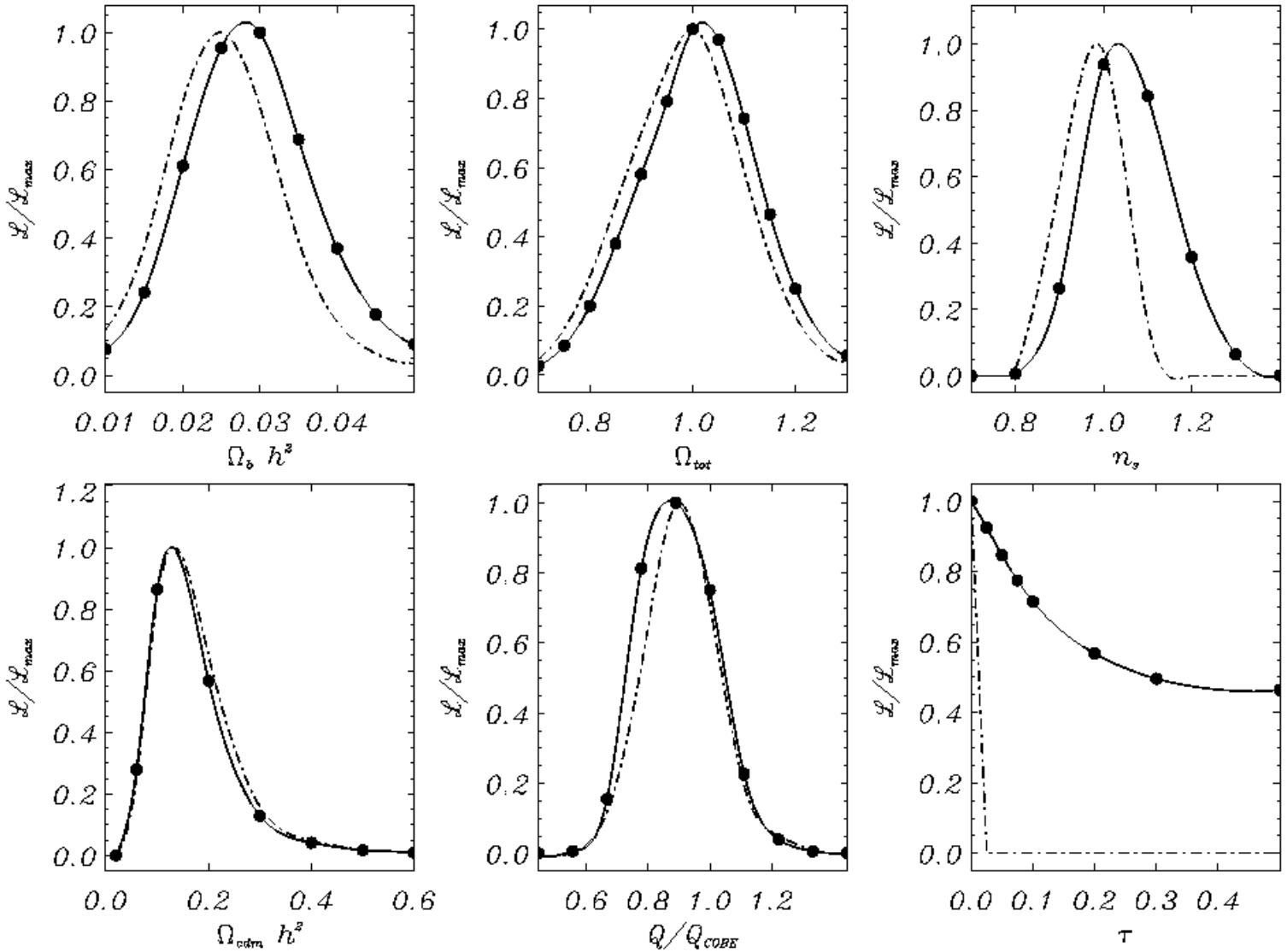
- Rubiño-Martín et al. (2002): Grid method in a 7-dimensional space.
- Family of models to explore:
  - Inflation with single scalar field
  - Adiabatic perturbations
  - No tensor modes.
  - No hot dark matter (neutrino contribution set to zero).
  - Normalization of the model was treated separately.
- A grid of 800,000 models was computed.

$$\vec{p} = \{\omega_b, \omega_{cdm}, \Omega_\Lambda, \Omega_{\rm tot}, n_{\rm s}, \tau, Q_{\rm rms-ps}\}$$

- $\Omega_b h^2 = [0.010,\ 0.015,\ 0.020,\ 0.025,\ 0.030,\ 0.035,\ 0.040,\ 0.045,\ 0.050]$,
- $\Omega_{cdm} h^2 = [0.02,\ 0.06,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1.0,\ 1.1]$,
- $\Omega_\Lambda = [0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8]$,
- $\Omega_{tot} = [0.7,\ 0.75,\ 0.80,\ 0.85,\ 0.90,\ 1.00,\ 1.05,\ 1.10,\ 1.15,\ 1.20,\ 1.30]$,
- $n_s = [0.7,\ 0.8,\ 0.9,\ 1.0,\ 1.1,\ 1.2,\ 1.3,\ 1.4]$,
- $\tau = [0.0,\ 0.025,\ 0.05,\ 0.075,\ 0.1,\ 0.2,\ 0.3,\ 0.5]$.

Example of application: grid method (II)

(pre-WMAP data)



Rubiño-Martín et al. (2002)

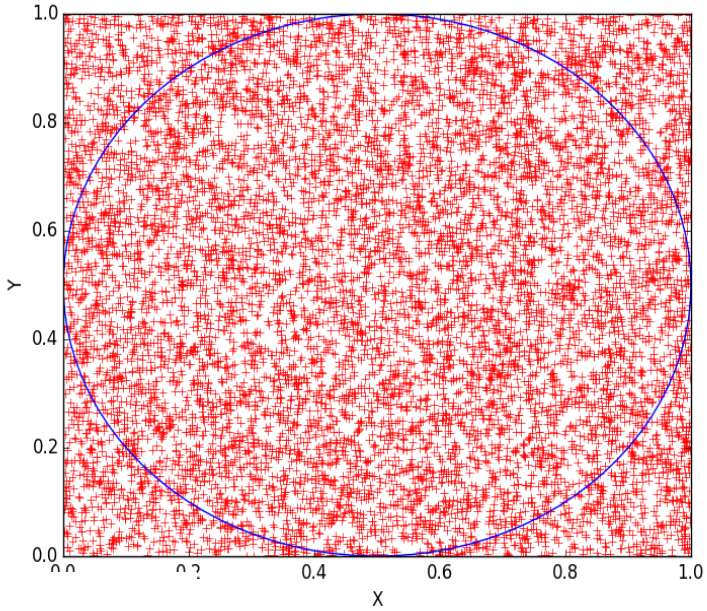
Monte Carlo Markov Chains

❖ A **Markov Chain** is is a discrete-time stochastic process with the Markov property.

❖ **Markov property:** a stochastic process has the Markov property if the conditional probability distribution of future states of the process, given the present state and all past states, depends only upon the present state and not on any past states, i.e. it is conditionally independent of the past states (the path of the process) given the present state.

❖ **Example**:

Estimate the value of PI using a Markov Chain. (see Python Code example_mc.py ; or IDL code example_mc.pro).




```
# Computing PI
nsamp = 10000
chain = np.random.uniform(0,1,[nsamp,2])

print chain

# Fraction of points inside the circle
dist = np.sqrt( (chain[:,0] - 0.5)**2. + (chain[:,1] - 0.5)**2. )
a, = np.where( dist < R )
pi_est = float(len(a)) / float(nsamp) * 4.0
```

Monte Carlo Markov Chains

❖ A **Markov Chain** is is a discrete-time stochastic process with the Markov property.

❖ **Markov property:** a stochastic process has the Markov property if the conditional probability distribution of future states of the process, given the present state and all past states, depends only upon the present state and not on any past states, i.e. it is conditionally independent of the past states (the path of the process) given the present state.

❖ It can be used to sample probability distributions. For example, using the **Metropolis-Hastings** algorithm (see Lewis & Bridle 2002; see also http://cosmologist.info/cosmomc/).

Chain moves from a position in parameter space θ_1 to the next position θ_2 with transition probability $T(\theta_1, \theta_2)$, where θ labels a vector of parameter values. The Metropolis-Hastings transition kernel $T(\theta_1, \theta_2)$ is chosen so that the Markov Chain has a stationary asymptotic distribution equal to $P(\theta)$, where $P(\theta)$ is the distribution we wish to sample from. This is done by using an arbitrary *proposal density* distribution $q(\theta_n, \theta_{n+1})$ to propose a new point θ_{n+1} given the chain is currently at θ_n. The proposed new point is then accepted with probability

$$\alpha(\theta_n, \theta_{n+1}) = \min\left\{1, \frac{P(\theta_{n+1})q(\theta_{n+1}, \theta_n)}{P(\theta_n)q(\theta_n, \theta_{n+1})}\right\}$$

so that $T(\theta_n, \theta_{n+1}) = \alpha(\theta_n, \theta_{n+1})q(\theta_n, \theta_{n+1})$. This construction ensures that detailed balance holds,

$$P(\theta_{n+1})T(\theta_{n+1}, \theta_n) = P(\theta_n)T(\theta_n, \theta_{n+1}),$$

and hence that $P(\theta)$ is the equilibrium distribution of the chain.

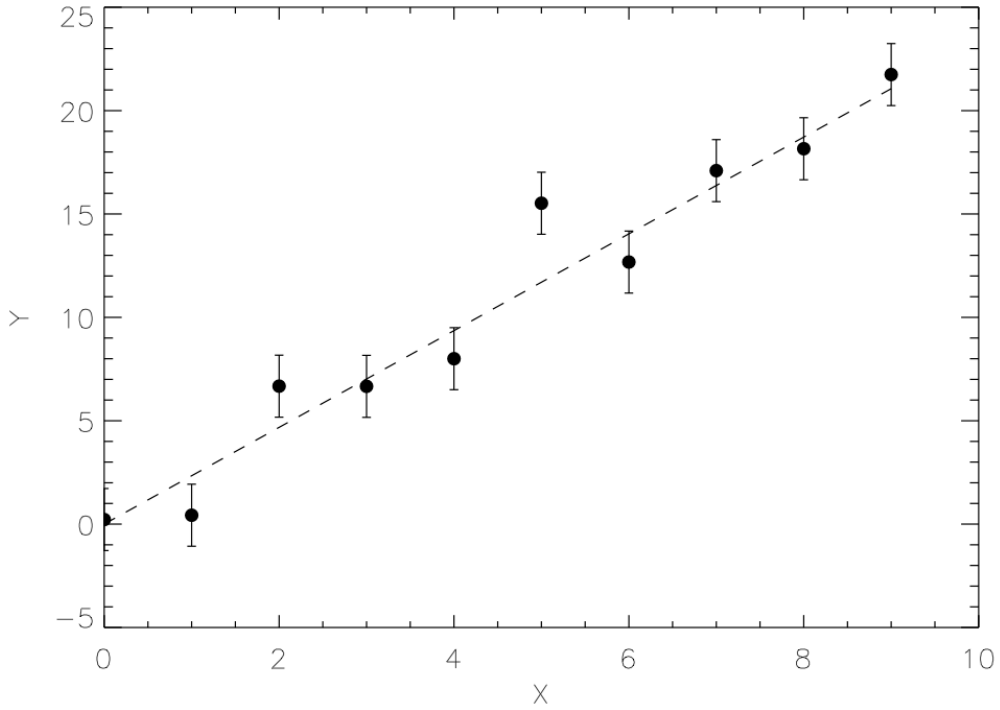
Monte Carlo Markov Chains: Metropolis-Hastings algorithm – Example.

❖ **Example**. Consider the following model, with a=0, b=2.34.

$$y = a + bx$$

❖ Assume that we measure in ten values of X = [0, 1, 2, 3, 4, .. 9], and we obtain then observed values of Y, each one with an error sigma=1.5. For example:

Y = [0.22096867 0.42906582 6.6721405 6.6635210 8.0009305 15.520682 12.675221 17.097184 18.156356 21.744441]



Monte Carlo Markov Chains: Metropolis-Hastings algorithm – Example.

❖ **Example**. Consider the following model, with a=0, b=2.34.

$$y = a + bx$$

❖ Assume that we measure in ten values of X = [0, 1, 2, 3, 4, .. 9], and we obtain then observed values of Y, each one with an error sigma=1.5. For example:

Y = [0.22096867 0.42906582 6.6721405 6.6635210 8.0009305 15.520682 12.675221 17.097184 18.156356 21.744441]

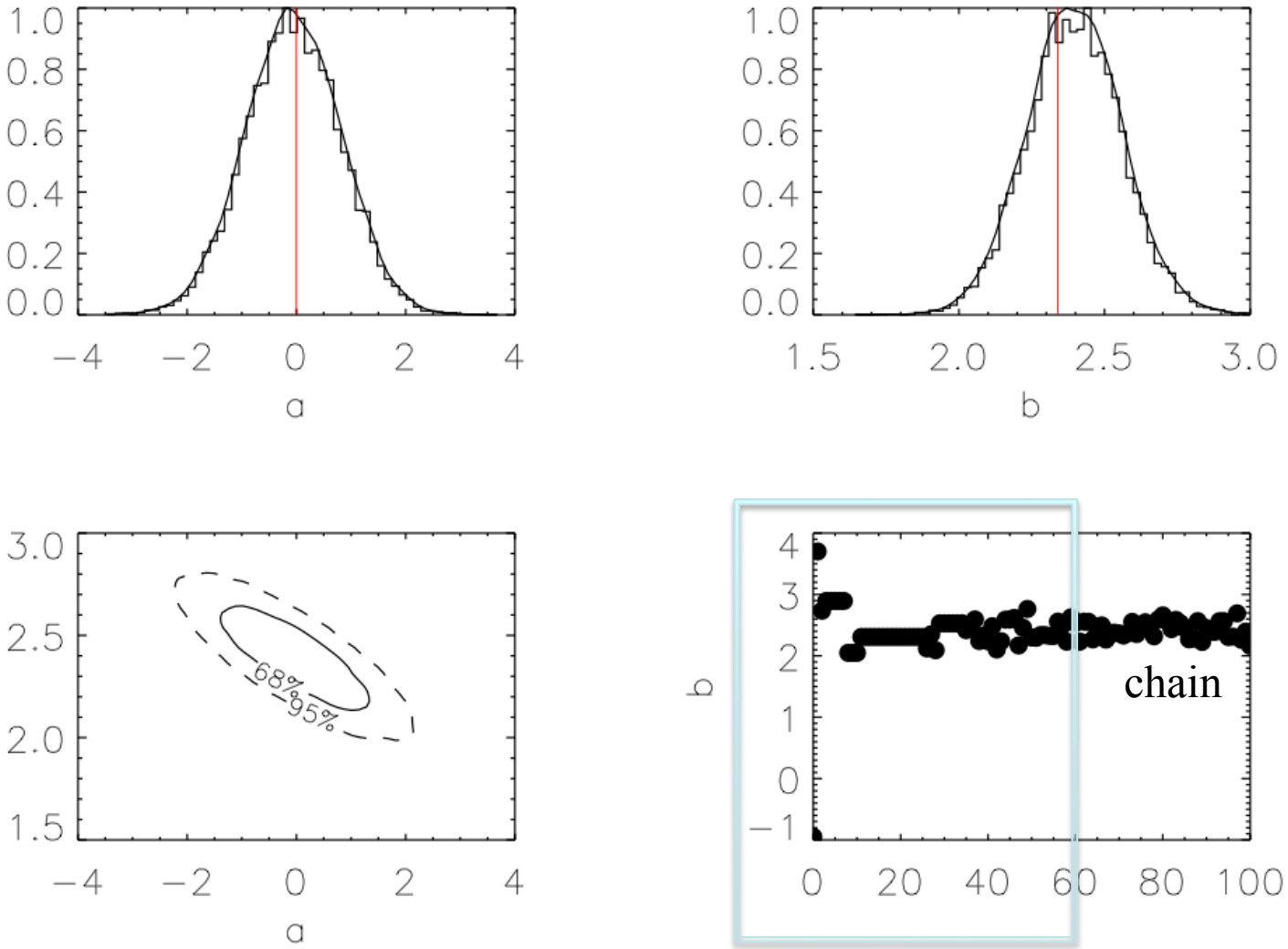
$$\chi^2 = \sum_{i=1}^{10} \frac{(y_i - a - bx_i)^2}{\sigma_i^2}$$

$$\mathcal{L} \propto \exp\left(-\frac{1}{2}\chi^2\right)$$

Monte Carlo Markov Chains: Metropolis-Hastings algorithm – Example.

❖ **Example**. Consider the following model, with a=0, b=2.34.

$$y = a + bx$$

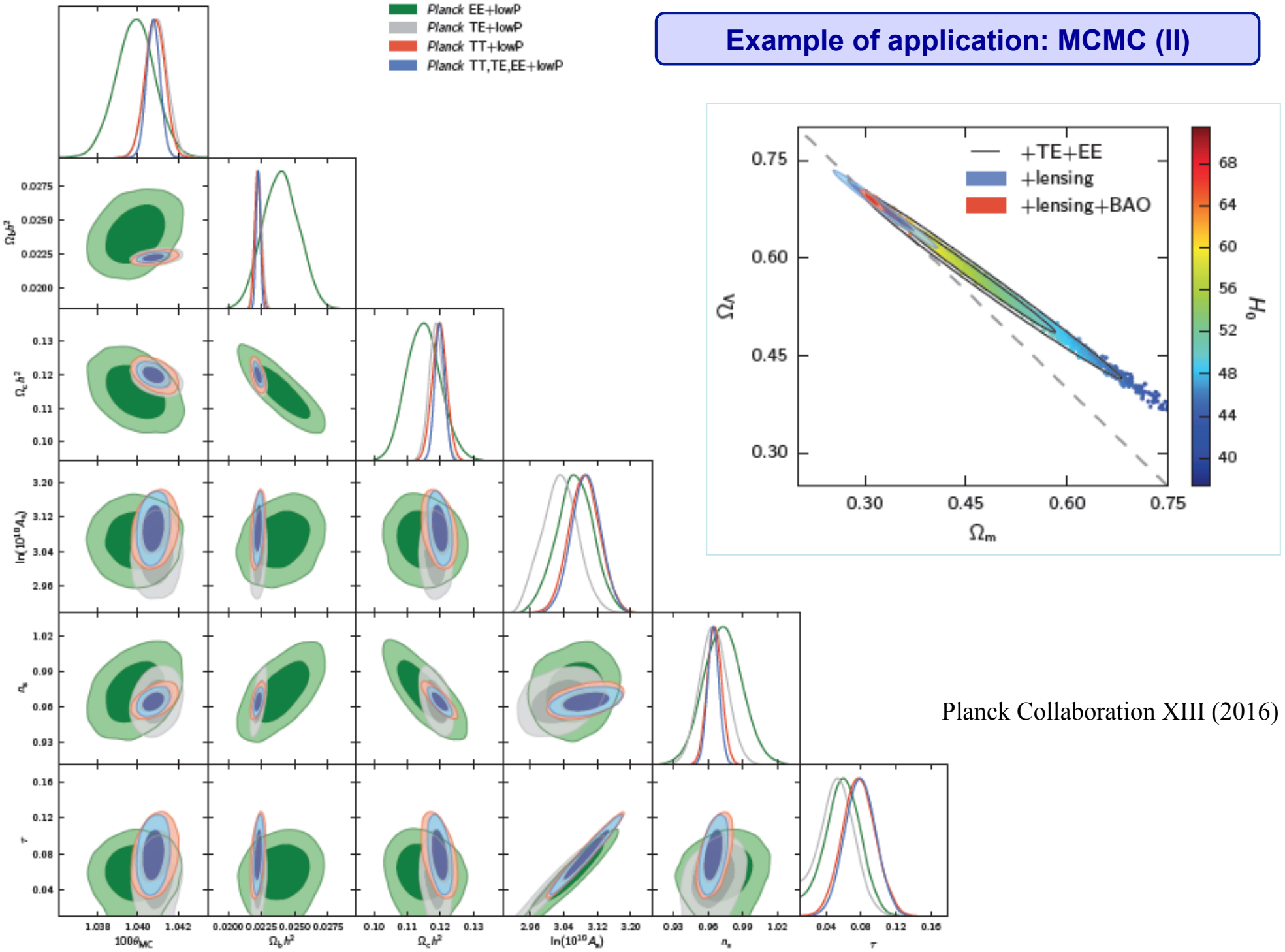


Example of application: MCMC

❖ Rebolo et al. (2004): MCMC in a 12-dimensional space.
❖ Input dataset: VSA data, WMAP, and other CMB experiments.
❖ Number of models explored is of the order of 250,000 for each case. → more efficient sampling.

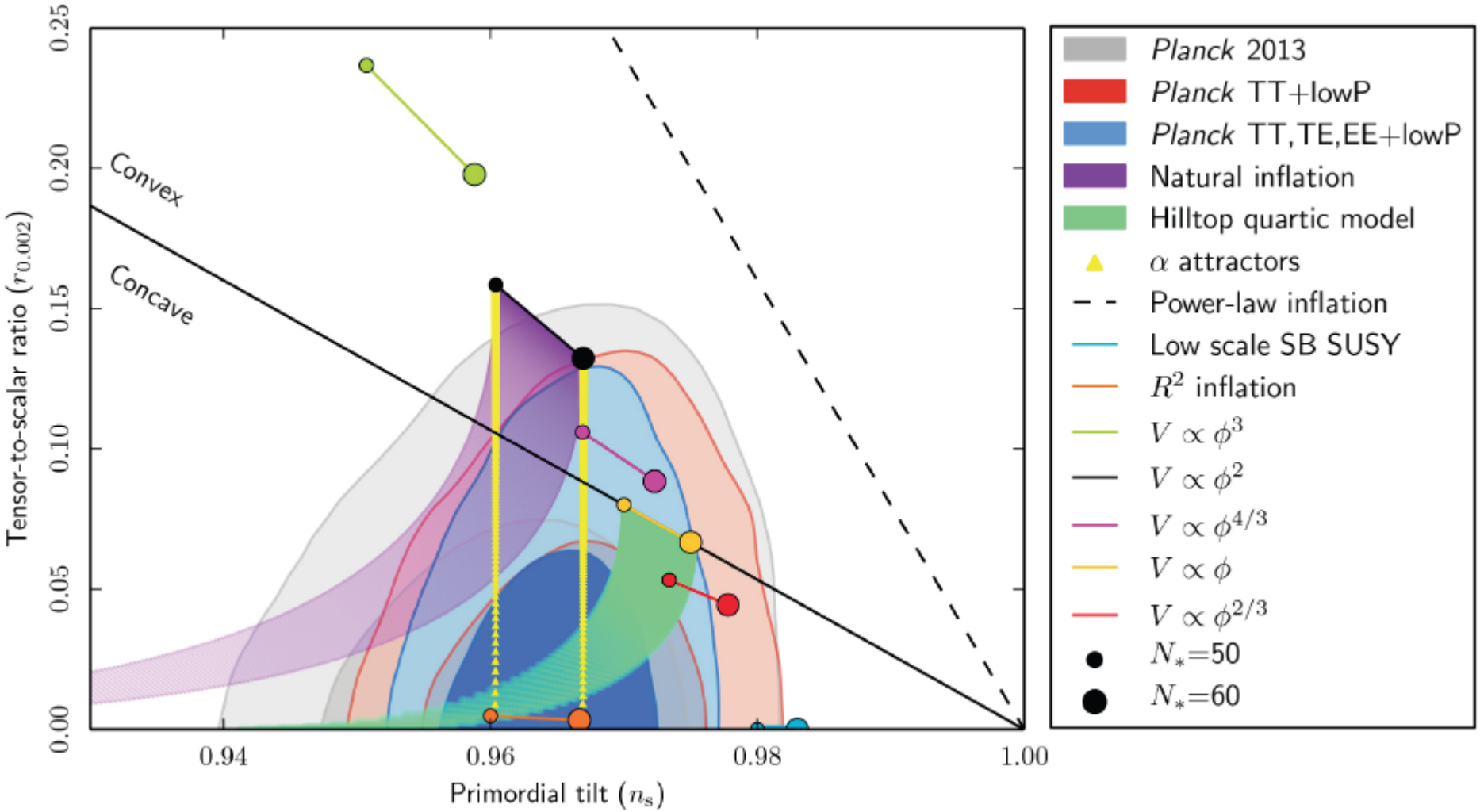
	WMAP	WMAP+VSA	AllCMB
$\Omega_{\rm b}h^2$	$0.025^{+0.003}_{-0.003}$	$0.024^{+0.003}_{-0.002}$	$0.023^{+0.002}_{-0.002}$
$\Omega_{\rm dm}h^2$	$0.108^{+0.022}_{-0.021}$	$0.111^{+0.021}_{-0.019}$	$0.113^{+0.017}_{-0.017}$
h	$0.66^{+0.07}_{-0.06}$	$0.66^{+0.06}_{-0.06}$	$0.65^{+0.07}_{-0.07}$
$z_{\rm re}$	18^{+7}_{-7}	19^{+7}_{-7}	17^{+7}_{-8}
$\Omega_{\rm k}$	$-0.02^{+0.03}_{-0.03}$	$-0.01^{+0.03}_{-0.03}$	$-0.02^{+0.03}_{-0.03}$
f_ν	< 0.093	< 0.083	< 0.083
w	$-1.00^{+0.24}_{-0.27}$	$-0.99^{+0.24}_{-0.27}$	$-1.06^{+0.24}_{-0.25}$
$n_{\rm S}$	$1.04^{+0.12}_{-0.11}$	$0.99^{+0.09}_{-0.09}$	$0.96^{+0.07}_{-0.07}$
$n_{\rm T}$	$0.26^{+0.53}_{-0.60}$	$0.13^{+0.49}_{-0.51}$	$0.12^{+0.48}_{-0.51}$
$n_{\rm run}$	$-0.02^{+0.07}_{-0.05}$	$-0.04^{+0.05}_{-0.04}$	$-0.04^{+0.04}_{-0.05}$
$10^{10}A_{\rm S}$	27^{+8}_{-5}	26^{+9}_{-5}	25^{+6}_{-5}
R	< 0.78	< 0.77	< 0.68
Ω_Λ	$0.71^{+0.07}_{-0.09}$	$0.70^{+0.06}_{-0.08}$	$0.69^{+0.07}_{-0.09}$
t_0	$14.1^{+1.4}_{-1.1}$	$14.1^{+1.3}_{-1.2}$	$14.4^{+1.4}_{-1.3}$
$\Omega_{\rm m}$	$0.31^{+0.09}_{-0.07}$	$0.31^{+0.08}_{-0.06}$	$0.33^{+0.10}_{-0.07}$
σ_8	$0.76^{+0.14}_{-0.14}$	$0.77^{+0.13}_{-0.13}$	$0.76^{+0.11}_{-0.12}$
τ	$0.20^{+0.13}_{-0.11}$	$0.20^{+0.15}_{-0.10}$	$0.17^{+0.12}_{-0.10}$

Example of application: MCMC (II)



Planck Collaboration XIII (2016)

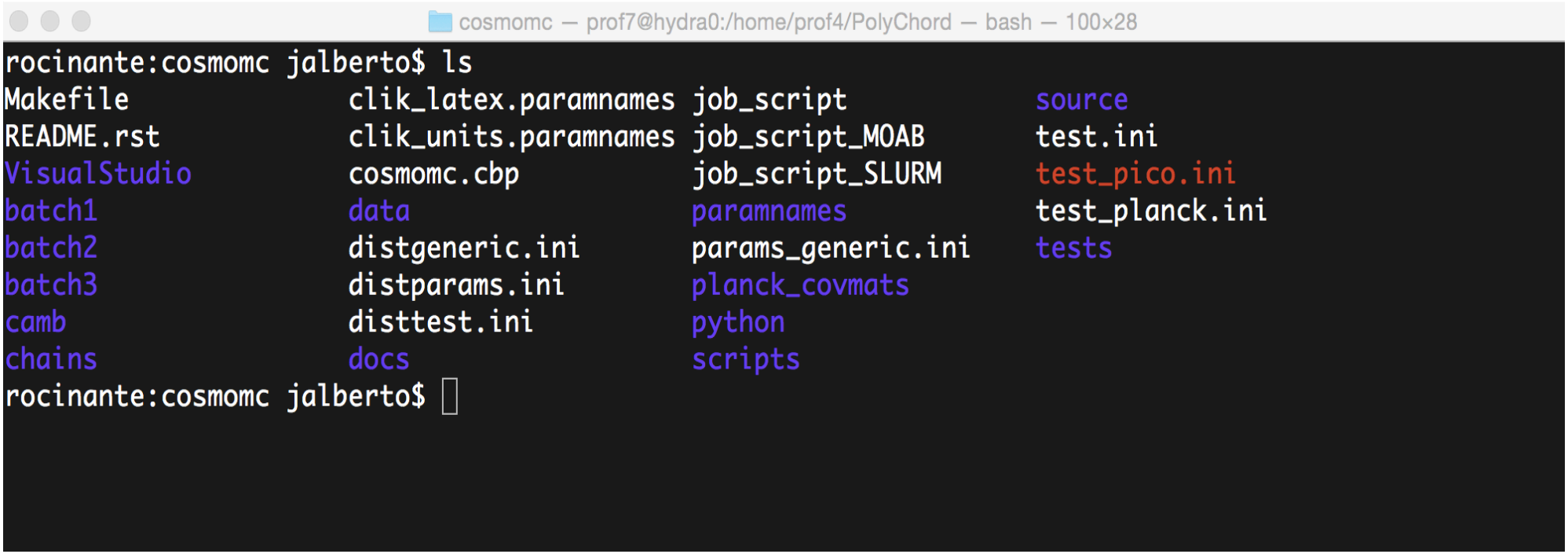
Example of application: MCMC (III)



Planck Collaboration XX (2016)

II. COSMOMC. Structure of the code

COSMOMC directory (Nov 2016)



test.ini → basic input file. See also test_planck.ini
batch1/ → old scripts
batch2/ → Planck 2015 and LSS scripts. Default.
batch3/ → other scripts (DR12 BAO, HST Riess et al.).

source/ → source code
data/ → reference data for all likelihoods.
camb/
chains/
paramnames/

Source files

ArrayUtils.f90
BaseParameters.f90
CMB.f90
CMB_BK_Planck.f90
CMBlikes.f90
CalcLike_Cosmology.f90
Calculator_CAMB.f90
Calculator_Cosmology.f90
Calculator_PICO.f90
CosmoTheory.f90
CosmologyConfig.f90
CosmologyParameterizations.f90
CosmologyTypes.f90
DataLikelihoods.f90
ElementAbundances.f90
EstCovmat.f90
FileUtils.f90
GeneralConfig.f90
GeneralSetup.f90
GeneralTypes.f90
GetDist.f90
HST.f90
IO.f90
ImportanceSampling.f90
IniObjects.f90
Interpolation.f90
Likelihood_Cosmology.f90
MCMC.f90
Makefile
Matrix_utils_new.f90
MiscUtils.f90
MpiUtils.f90
ObjectLists.f90
ObjectParamNames.f90
ParamSet.f90
PowellConstrainedMinimize.f90
RandUtils.f90
SampleCollector.f90
StringUtils.f90
bao.f90
bbn.f90
bsplinepk.c
calclike.f90
cliklike.f90
cmbdata.F90.old
driver.F90
likelihood.f90
lrggettheory.f90
minimize.f90
mpk.f90
propose.f90
samples.f90
settings.f90
supernovae.f90
supernovae_JLA.f90
supernovae_SNLS.f90
supernovae_Union2.f90
szcounts.f90
wigglez.f90
wl.f90

Source files

ArrayUtils.f90
BaseParameters.f90
CMB.f90
CMB_BK_Planck.f90
CMBlikes.f90
CalcLike_Cosmology.f90
Calculator_CAMB.f90
Calculator_Cosmology.f90
Calculator_PICO.f90
CosmoTheory.f90
CosmologyConfig.f90
CosmologyParameterizations.f90
CosmologyTypes.f90
DataLikelihoods.f90
ElementAbundances.f90
EstCovmat.f90
FileUtils.f90
GeneralConfig.f90
GeneralSetup.f90
GeneralTypes.f90
GetDist.f90
HST.f90
IO.f90
ImportanceSampling.f90
IniObjects.f90
Interpolation.f90
Likelihood_Cosmology.f90
MCMC.f90
Makefile
Matrix_utils_new.f90
MiscUtils.f90
MpiUtils.f90
ObjectLists.f90
ObjectParamNames.f90
ParamSet.f90
PowellConstrainedMinimize.f90
RandUtils.f90
SampleCollector.f90
StringUtils.f90
bao.f90
bbn.f90
bsplinepk.c
calclike.f90
cliklike.f90
cmbdata.F90.old
driver.F90
likelihood.f90
lrggettheory.f90
minimize.f90
mpk.f90
propose.f90
samples.f90
settings.f90
supernovae.f90
supernovae_JLA.f90
supernovae_SNLS.f90
supernovae_Union2.f90
szcounts.f90
wigglez.f90
wl.f90

CosmoMC documentation: http://cosmologist.info/cosmomc/doc/

settings.f90
This defines the maximum number of parameters and their types.

Calculator_Cosmology.f90 and **Calculator_CAMB.f90**
Routines for generating Cls, matter power spectra and sigma8 from CAMB. Override the calculator class in Calculator_Cosmology.f90 to implement cosmology using other calculators e.g. a fast approximator like PICO, or other Boltzmann code. etc.

DataLikelihoods.f90 This is where you can add in new likelihood functions

driver.F90 Main program that reads in parameters and calls MCMC or post-processing.

propose.f90 This is the proposal density and related constants and subroutines. The efficiency of MCMC is quite dependent on the proposal. Fast+slow and fast parameter subspaces are proposed separately. See Lewis (2013) for a discussion of the proposal density and use of fast and slow parameters.

III. Running the code

Serial mode:

> ./cosmomc test.ini 1

MPI mode:

> mpirun –np 2 ./cosmomc test.ini

Python script:

> python python/runMPI.py test

COSMOMC ini-file: test.ini

```
#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_TT.ini)
#DEFAULT(batch2/lowTEB.ini)
#DEFAULT(batch2/lowl.ini)
DEFAULT(batch2/lensing.ini)

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

#general settings
DEFAULT(batch2/common.ini)

#e.g. to vary r in addition to standard 6:
#(for r>0 also need compute_tensors=T)
#compute_tensors = T
#param[r] = 0.03 0 2 0.04 0.04

#high for new runs
MPI_Max_R_ProposeUpdate = 30

propose_matrix= planck_covmats/base_TT_lowTEB_plik.covmat

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

COSMOMC ini-file: test.ini

```
#Root name for files produced
file_root=test
#action= 0 runs chains, 1 importance samples, 2 minimizes
#use action=4 just to quickly test likelihoods
action = 4

#expected result for -(log like)
test_check_compare = 28.337

num_threads = 0

#if you want to get theory cl for test point
#test_output_root = output_cl_root

start_at_bestfit =F
feedback=1
use_fast_slow = T

checkpoint = F

#sampling_method=7 is a new fast-slow scheme good for Planck
sampling_method = 7
dragging_steps  = 3
propose_scale = 2

#Set >0 to make data files for importance sampling
indep_sample=10

#these are just small speedups for testing
get_sigma8=T
```

Action = 0 runs chains
= 1 importance sampling
= 4 test likelihoods

Cosmomc has 7 different sampling methods. Number 1 is the Metropolis-Hasting scheme

batch2/common.ini

```
##Sample file of common parameters for baseline Planck set of runs

batch_name = batch2

local_dir = %LOCALDIR%
#directory, e.g. window functions in directory windows under data_dir
data_dir = %LOCALDIR%data/

INCLUDE(likelihood.ini)
INCLUDE(params_CMB_defaults.ini)

#whether to include prior on a parameter if it has non-varying value
include_fixed_parameter_priors = F

#Feedback level ( 2=lots,1=chatty,0=none)
feedback = 1

#Force computation of sigma_8 even if use_mpk = F
get_sigma8 = T

#This only has a small effect at very high L
use_nonlinear_lensing = T
#if using non-linear lensing, better turn of power spectrum fast/slow since is now non-linear
block_semi_fast = F

#Temperature at which to Monte-Carlo
temperature = 1

#Maximum number of chain steps
samples = 4000000

#Scale of proposal relative to covariance; 2.4 is recommended by astro-ph/0405462 for Gaussians
#If propose_matrix is much broader than the new distribution, make proportionately smaller
#Generally make smaller if your acceptance rate is too low
propose_scale = 1.9

#Increase to oversample fast parameters more, e.g. if space is odd shape
oversample_fast = 1
```

To sample **from** P^(1/T) rather than P (good for the tails of P, the posterior distribution)

batch2/common.ini

```
#if non-zero number of steps between sample info dumped to file file_root.data
#WANT THIS ON so we can do importance sampling runs quickly later for likelihood updates
indep_sample = 10

#number of samples to disgard at start; usually set to zero and remove later
burn_in = 0

#If zero set automatically
num_threads = 0

#MPI mode multi-chain options (recommended)
#MPI_Converge_Stop is a (variance of chain means)/(mean of variances) parameter that can be used to stop the chains
#Set to a negative number not to use this feature. Does not guarantee good accuracy of confidence limits.
MPI_Converge_Stop = 0.01
```

(Gelman and Rubin R statistic)

```
#Do initial period of slice sampling; may be good idea if
#cov matrix or widths are likely to be very poor estimates
MPI_StartSliceSampling  = F

#Can optionally also check for convergence of confidence limits (after MPI_Converge_Stop reached)
#Can be good idea as small value of MPI_Converge_Stop does not (necessarily) imply good exploration of tails
MPI_Check_Limit_Converge = T

#if MPI_Check_Limit_Converge = T, give tail fraction to check (checks both tails):
MPI_Limit_Converge = 0.025
#permitted quantile chain variance in units of the standard deviation (small values v slow):
MPI_Limit_Converge_Err = 0.2
#which parameters tails to check. If zero, check all parameters:
MPI_Limit_Param = 0

#if MPI_LearnPropose = T, the proposal density is continally updated from the covariance of samples so far (since burn in)
MPI_LearnPropose = T
#can set a value of converge at which to stop updating covariance (so that it becomes rigorously Markovian)
#e.g. MPI_R_StopProposeUpdate = 0.4 will stop updating when (variance of chain means)/(mean of variances) < 0.4
MPI_R_StopProposeUpdate = 0
```

batch2/likelihood.ini

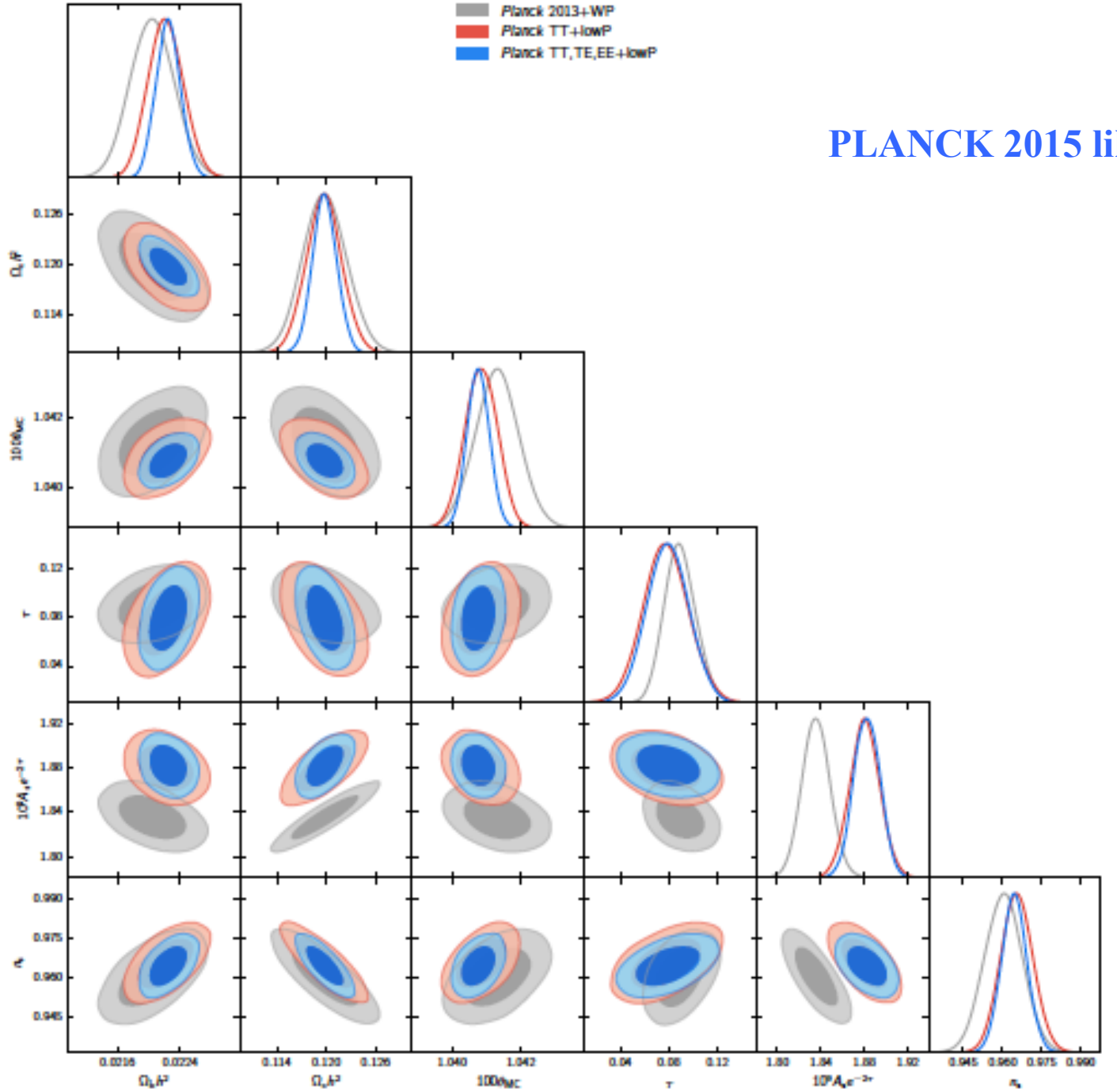
This controls the extra information that you want to put inside your likelihood function.

```
use_HST = F
use_mpk = F
use_SN = F
use_BAO=F

#if true, use HALOFIT for non-linear corrections (astro-ph/0207664).
nonlinear_pk = F

use_Age_Tophat_Prior = T
```

batch2/params_CMB_defaults.ini

```
#New for 2014
#no zre prior in chains, can do later by importance sampling
use_min_zre = 0

lmin_store_all_cmb = 2500


#CAMB parameters
#If we are including tensors
compute_tensors = F
#If using tensors, enforce n_T = -A_T/(8A_s)
inflation_consistency = T
#nt setting is then ignored
param[nt] = 0
param[ntrun] = 0

#Set Y_He from BBN constraint; if false set to fixed value of 0.24 by default.
bbn_consistency=T

H0_min=20
H0_max=100

#to vary parameters set param[name]= center, min, max, start width, propose width
#for fixed can just fix fixed value

param[omegabh2] = 0.0221 0.005 0.1 0.0001 0.0001
param[omegach2] = 0.12 0.001 0.99 0.001 0.0005
param[theta] = 1.0411 0.5 10 0.0004 0.0002
param[tau] = 0.09 0.01 0.8 0.01 0.005
```

Center min max start_width propose_width



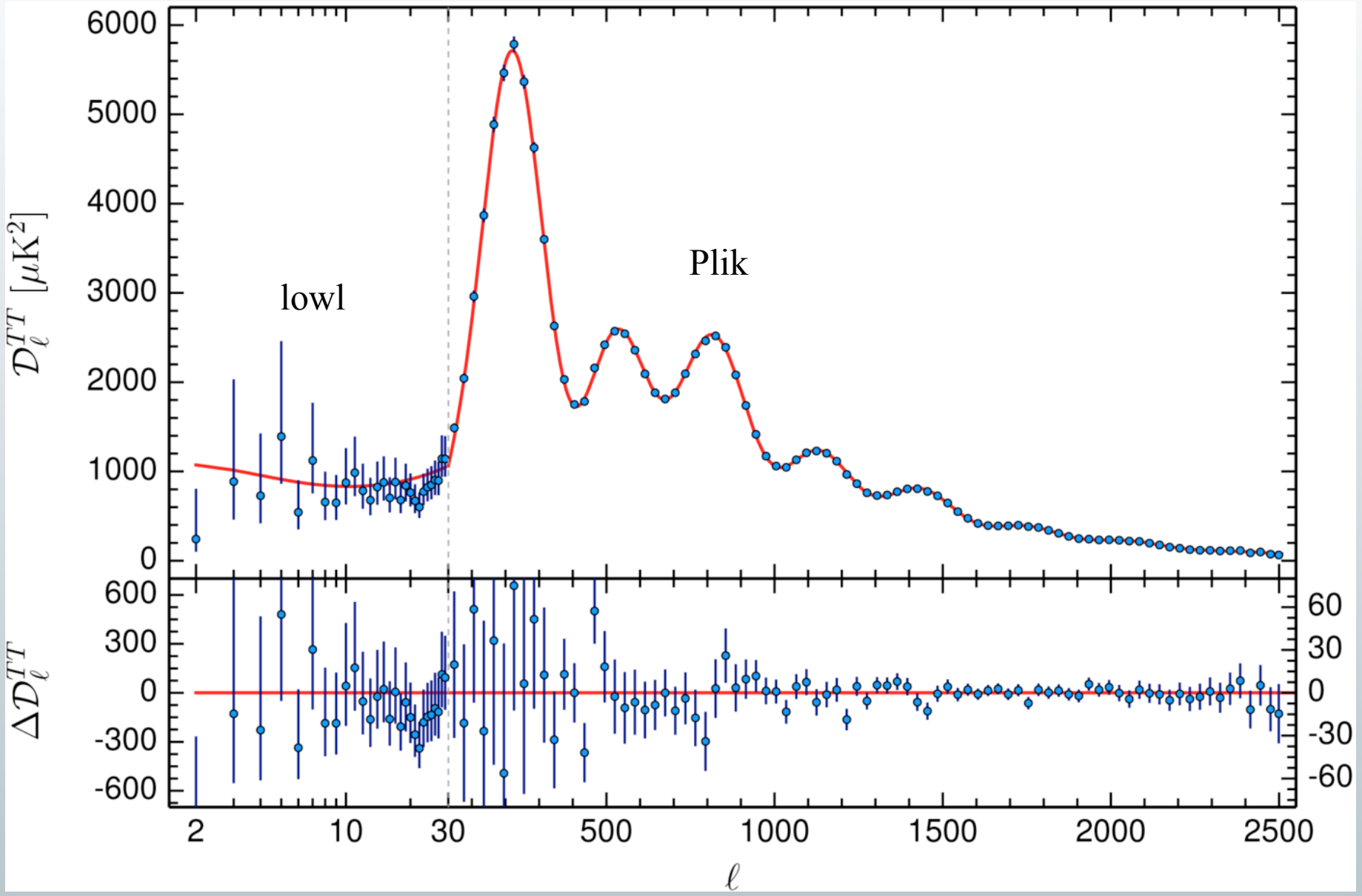
PLANCK 2015 likelihood

https://wiki.cosmos.esa.int/planckpla2015/index.php/CMB_spectrum_%26_Likelihood_Code

Likelihood Methodology

- Hybrid multi-frequency likelihood approach
  - *Large scales **(l<30)**: map-based Gaussian likelihood*
  - *Small scales **(l≥30)**: Gaussian likelihood approximation on spectra*
- Marginalization over foregrounds
  - *Large scales: Gibbs marginalization (map level)*
  - *Small scales: Parameterized at the spectrum level. **Plik***
- Validation
  - *Data selection*
  - *Null tests*
  - *Simulations*
  - *Foreground cleaned CMB maps*

Planck power spectrum (TT)



Parameterizations

The default parameters (which get implicit flat priors) are:

omegabh2 - the physical baryon density
omegach2 - the physical dark matter density
theta - 100*(the ratio of the [approx] sound horizon to the angular diameter distance)
tau - the reionization optical depth
omegak - omega_K
mnu - the sum of the neutrino masses (in eV)
nnu - the effective density parameter for neutrinos N_{eff}
w - the (assumed constant) equation of state of the dark energy (taken to be quintessence)
ns - the scale spectral index
nt - the tensor spectral index
nrun - the running of the scalar spectral index
logA - ln[10^10 A_s] where A_s is the primordial superhorizon power in the curvature perturbation on 0.05Mpc^{-1} scales (i.e. in this is an amplitude parameter)
r - the ratio A_t/A_s, where A_t is the primordial power in the transverse traceless part of the metric tensor

The list of parameter names and labels used in the default parameterization is listed in the supplied **params_CMB.paramnames** file, inside **paramnames/.**

III. Analysing the output of COSMOMC.

After running **COSMOMC**:

- Output files in *chains/*

- *.txt* files contain the chains. The format is:

  weight likelihood param1 param2 param3 ...

- *.data* binary data files for post-processing, contain the chains, power spectra, etc. Only generated if requested (*indep_sample*>0)

- *.log* files contain STDOUT from each chain.

- **.***paramnames* file, listing the names and labels of the parameters corresponding to the columns 3+ of the output chain files.

But you need to do post-processing of the chains (thinning, burn-in period) and study convergence → *getdist*.

> *./getdist distparams.ini*

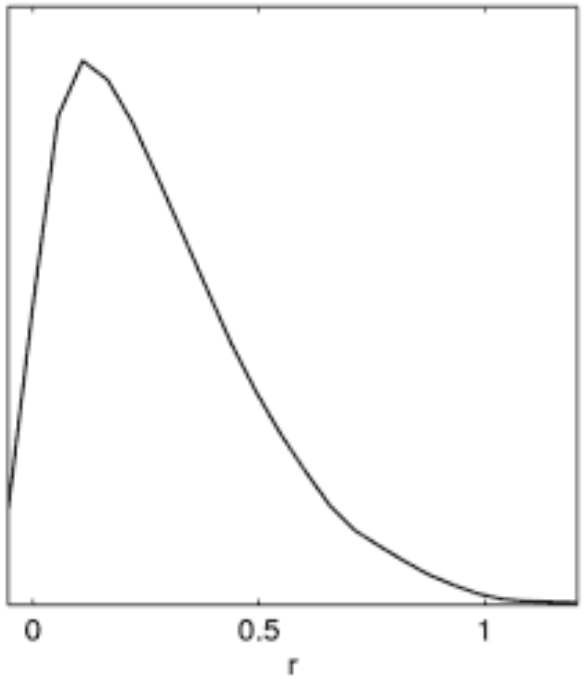
Or you can run *cosmomc* with *action=1* inside .ini file.

distparams.ini

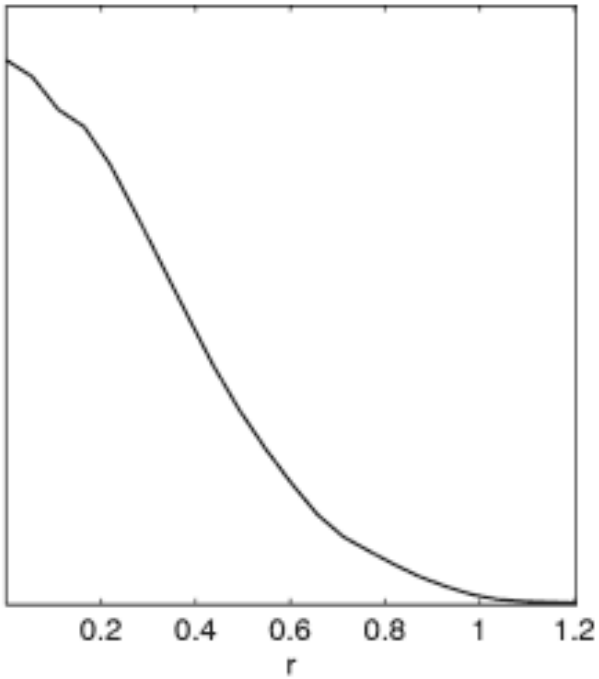
```
#sample params for "getdist" - for processing .txt chain information
#not required if you just want to run with defaults to produce .margestats etc

#set to true if you want to produce plot.py files for various standard plots
no_tests = F

file_root = chains/test
out_root =
out_dir =
plot_data_dir = ./plot_data/

#If 0 assume 1 and no chain filename prefixes; if -1 read as many as exist
chain_num = -1
first_chain =
exclude_chain =

#For disgarding burn-in if using raw chains. Set to zero if already removed or you have independent samples.
#if < 1 interpreted as a fraction of the total number of rows (0.3 ignores first 30% of lines)
ignore_rows = 0.3

#include defaults settings for kernel densitiy estimates etc, can also be specified in this file if you want to override
DEFAULT(python/getdist/analysis_defaults.ini)

#samples_are_chains = F can be useful for other samples when first two columns not present
samples_are_chains = T

#if no_plots =F, produce plot script files for specific plots producing ./plot_data/ density files
no_plots = T

#if we only want 2D plots agains a particular variable
plot_2D_param = 0
```

```
#if no_plots =F, produce plot script files for specific plots producing ./plot_data/ density files
no_plots = T

#if we only want 2D plots agains a particular variable
plot_2D_param = 0

#if above zero, instead plot just these combinations:
#if both zero it will plot most correlated variables
plot_2D_num = 0
plot1 = ns omegabh2
plot2 =

#number of sample plots, colored by third parameter
#if last parameter is 0 or -1 colored by the parameter most correlated
#with one of the eigenvector directions (e.g. parallel or orthogonal to degeneracy)
num_3D_plots = 1
3D_plot1 = H0 omegam tau

#Output 2D plots for param combos with 1D marginalized plots along the diagonal
triangle_plot = T
triangle_params = omegabh2 omegach2 tau omegak mnu nnu yhe Alens ns nrun logA r H0 omegam omegal sigma8 r02
```

```
#Need to give limits if prior cuts off distribution where not very
small
limits[r02]= 0 N
limits[r10]= 0 N
```



Incorrect result when **limits[r02]** is not set.



Correct result when setting **limits[r02]=0 N**.

- Convergence diagnostics are generated in the same directory, with files named **.converge*, **.likestats*, **.margestats*, **.covmat* , and **.corr*

file_root.margestats file contains the means, standard deviations and marginalized limits for the different parameters

file_root.likestats gives the best fit sample model, its likelihood, and limits from the extremal values of the N-dimensional distribution.

file_root.converge contains various convergence diagnostics

file_root.corr contains parameter correlations

file_root.covmat contains a covariance matrix you can use as a proposal matrix for generating future chains

Convergence diagnostics

- **Gelman and Rubin** "variance of chain means"/"mean of chain variances" **R statistic**. This statistic can be computed if you have multiple chains. Typically you want the value to be less than 0.2.

- For individual chains, *getdist* computes the **Raftery and Lewis** convergence diagnostics. This uses a binary chain derived from each parameter depending on whether the parameter is above or below a given percentile of its distribution. It is basically a criterion for accuracy of the estimation of a certain quantile q. It also assesses the thin factor needed for the binary chain to approximate an independence chain.

- There are other statistics included in *getdist* (see documentation for details).

COSMOMC output files (in *chains/*)

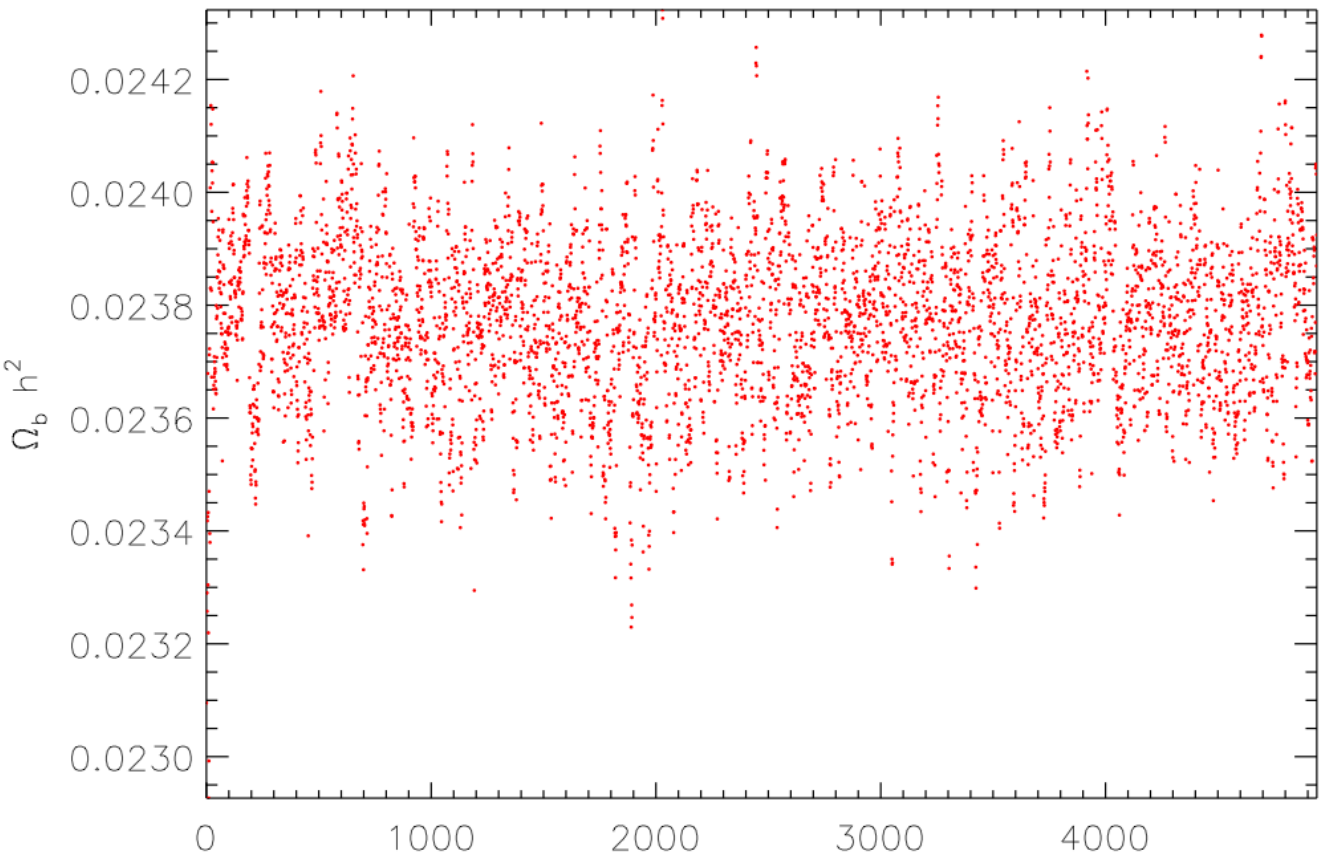
- *.txt* files contain the chains. The format is:

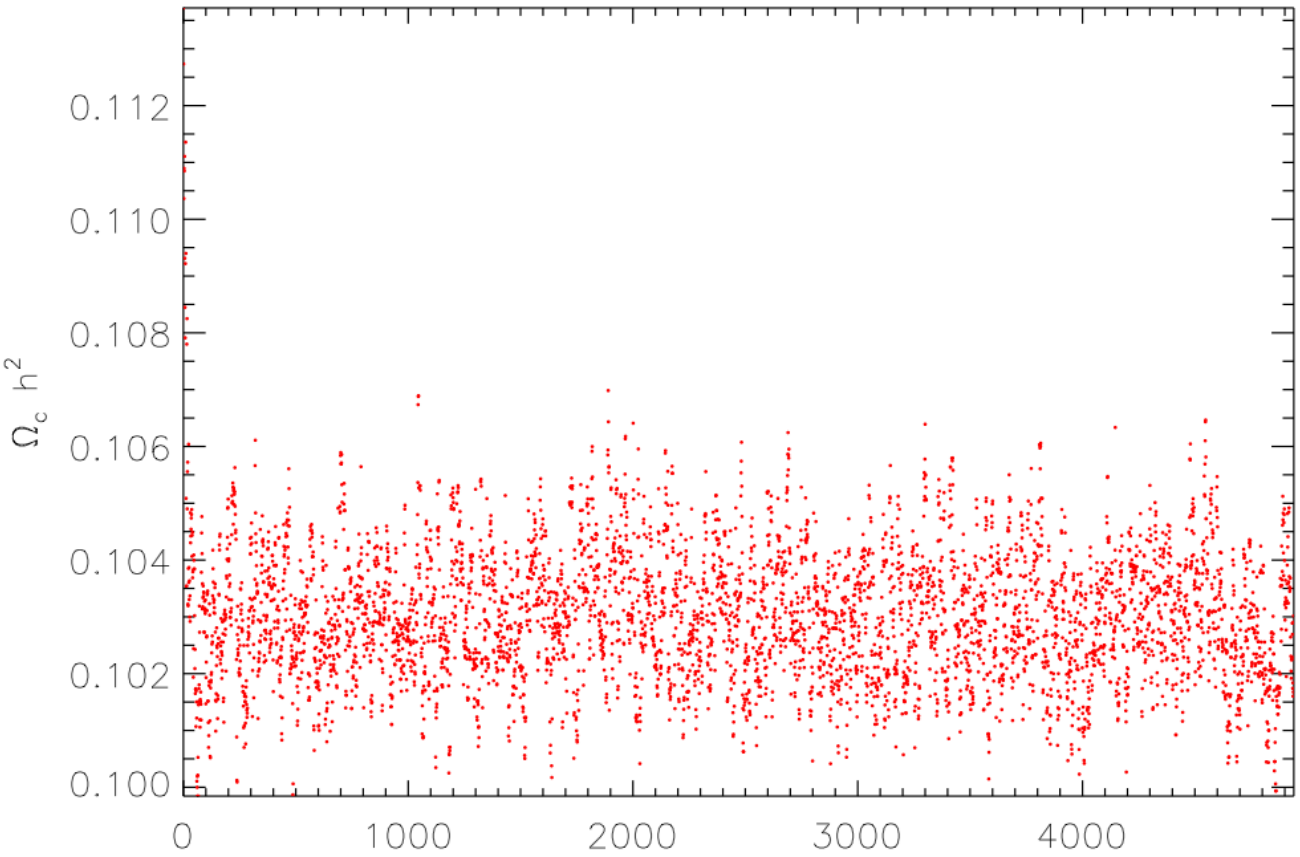
weight likelihood param1 param2 param3 ...

```
  0.2000000E+01  0.7496629E+03  0.2309515E-01  0.1137196E+00  0.1041837E+01  0.8840732E-01  0.9748397E+00  0.3051671E+01  0.7291811E+00  0.2708189E+00
0.0000000E+00  0.1051788E+02  0.0000000E+00  0.7107666E+02  0.0000000E+00  0.2115066E+01  0.1368148E+00  0.9724336E-01  0.2481444E+00  0.1772288E+01  0.0000000E+00
0.1363885E+02  0.1088588E+04  0.1455140E+03  0.1041860E+01  0.1061230E+04  0.1479632E+03  0.1403595E+00  0.1602554E+00  0.3269178E+04  0.8406736E+00  0.7401459E-01
0.3161102E-03  0.1340716E+04
  0.9000000E+01  0.6750891E+03  0.2329014E-01  0.1127334E+00  0.1042401E+01  0.7403696E-01  0.9745454E+00  0.3024771E+01  0.7362117E+00  0.2637883E+00
0.0000000E+00  0.9205627E+01  0.0000000E+00  0.7180908E+02  0.0000000E+00  0.2058929E+01  0.1360236E+00  0.9767728E-01  0.2482245E+00  0.1775553E+01  0.0000000E+00
0.1359757E+02  0.1088278E+04  0.1456248E+03  0.1042405E+01  0.1061573E+04  0.1480167E+03  0.1404365E+00  0.1601293E+00  0.3250267E+04  0.8453617E+00  0.7452110E-01
0.3175491E-03  0.1330808E+04
  0.1000000E+01  0.6276664E+03  0.2329036E-01  0.1103637E+00  0.1042806E+01  0.9711881E-01  0.9819501E+00  0.3065862E+01  0.7483236E+00  0.2516764E+00
0.0000000E+00  0.1109467E+02  0.0000000E+00  0.7287354E+02  0.0000000E+00  0.2145295E+01  0.1336540E+00  0.9739842E-01  0.2482246E+00  0.1766570E+01  0.0000000E+00
0.1357189E+02  0.1088071E+04  0.1462625E+03  0.1042825E+01  0.1061420E+04  0.1486659E+03  0.1397467E+00  0.1602826E+00  0.3193630E+04  0.8569429E+00  0.7545589E-01
0.3190523E-03  0.1317865E+04
  0.5000000E+01  0.6166291E+03  0.2325763E-01  0.1108959E+00  0.1042529E+01  0.9614229E-01  0.9827495E+00  0.3064756E+01  0.7450652E+00  0.2549348E+00
0.0000000E+00  0.1103973E+02  0.0000000E+00  0.7254150E+02  0.0000000E+00  0.2142924E+01  0.1341536E+00  0.9731702E-01  0.2482112E+00  0.1768067E+01  0.0000000E+00
0.1358637E+02  0.1088157E+04  0.1461439E+03  0.1042549E+01  0.1061382E+04  0.1485557E+03  0.1398435E+00  0.1602592E+00  0.3205569E+04  0.8542188E+00  0.7519120E-01
0.3184598E-03  0.1322129E+04
```

- *.paramnames* file, listing the names and labels of the parameters corresponding to the columns 3+ of the output chain files.

```
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 (approx)
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)
yhe             Y_{He}     #Helium mass fraction (not used if bbn_consistency)
alpha1          \alpha_{-1} #correlated CDM isocurvature
deltazrei       {\Delta}z_{\rm re} #width of reionization
Alens           A_{L}      #lensing potential scaled by sqrt(A_lens)
Alensf          A^f_L      #amplitude of smoothing power from fiducial models
fdm             \epsilon_0 f_d     #CosmoRec dark matter annihilation parameter, 0910.3663
```


0.0242
0.0240
0.0238
0.0236
0.0234
0.0232
0.0230
$\Omega_b\ h^2$
0
1000
2000
3000
4000

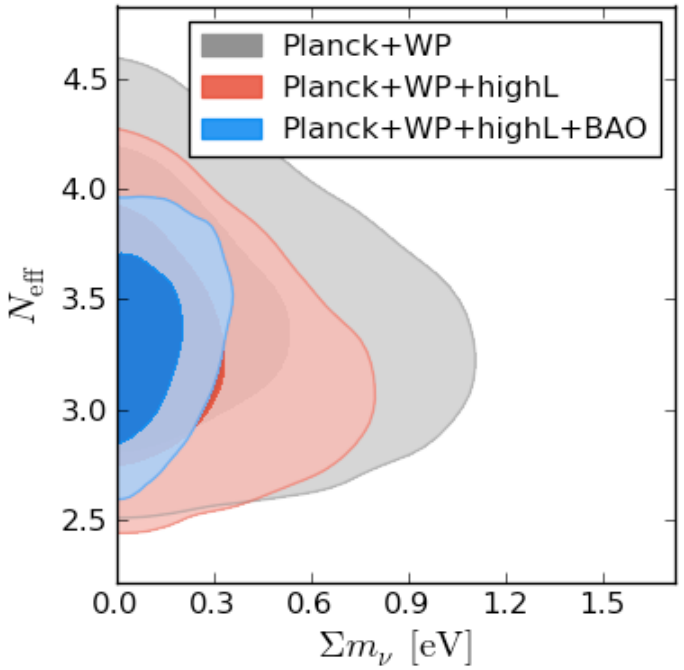

0.112
0.110
0.108
0.106
0.104
0.102
0.100
$\Omega_c\ h^2$
0
1000
2000
3000
4000

Weighted samples

All chains use weighted samples (first column). Properties of the chains have to be computed taking into account those weights.

$$\hat{P} = \frac{\sum w_i P_i}{\sum w_i}$$

For plotting the posterior distributions, you have to "weight" the samples in the chain and do histograms.



Importance sampling

(see Lewis & Bridle 2002, Phys.Rev.D66:103511,2002)

Given a set of samples from a distribution P, one can estimate quantities with respect to a different similar distribution P', by weighting the samples in proportion to the probability ratios. This effectively gives a collection of non-integer weighted samples for computing Monte-Carlo estimates. For example the expected value of a function $f(\boldsymbol{\theta})$ under P' is given by

$$\begin{aligned}\langle f(\boldsymbol{\theta})\rangle_{P'} &= \int \mathrm{d}\boldsymbol{\theta} P'(\boldsymbol{\theta}) f(\boldsymbol{\theta}) = \int \mathrm{d}\boldsymbol{\theta} \frac{P'(\boldsymbol{\theta})}{P(\boldsymbol{\theta})} P(\boldsymbol{\theta}) f(\boldsymbol{\theta}) \\ &= \left\langle \frac{P'(\boldsymbol{\theta})}{P(\boldsymbol{\theta})} f(\boldsymbol{\theta}) \right\rangle_P . \end{aligned} \tag{B1}$$

Given a set $\{\boldsymbol{\theta}_n\}$ of N samples from P a Monte-Carlo estimate is therefore

$$\langle f(\boldsymbol{\theta})\rangle_{P'} \approx \frac{1}{N} \sum_{n=1}^{N} \frac{P'(\boldsymbol{\theta}_n)}{P(\boldsymbol{\theta}_n)} f(\boldsymbol{\theta}_n). \tag{B2}$$

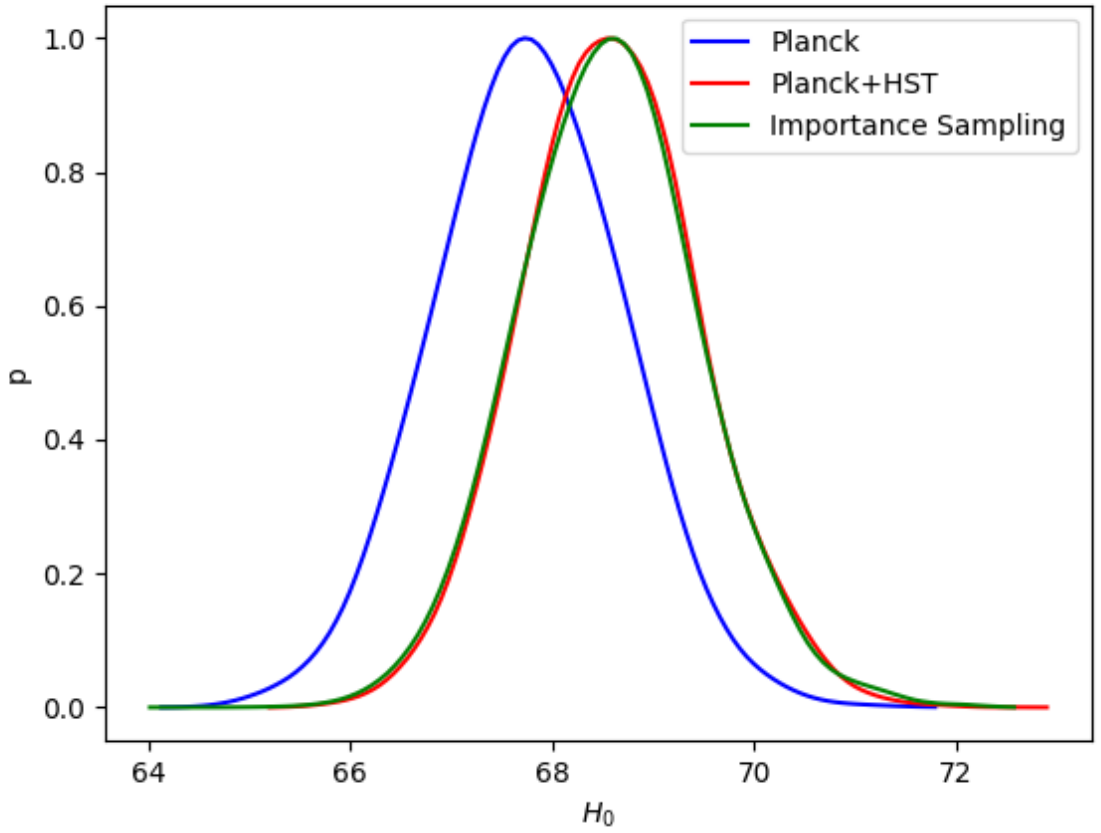
Exercise 1. Importance sampling

Do importance sampling of any pre-computed chain, adding a Gaussian likelihood for an additional parameter, as H_0.

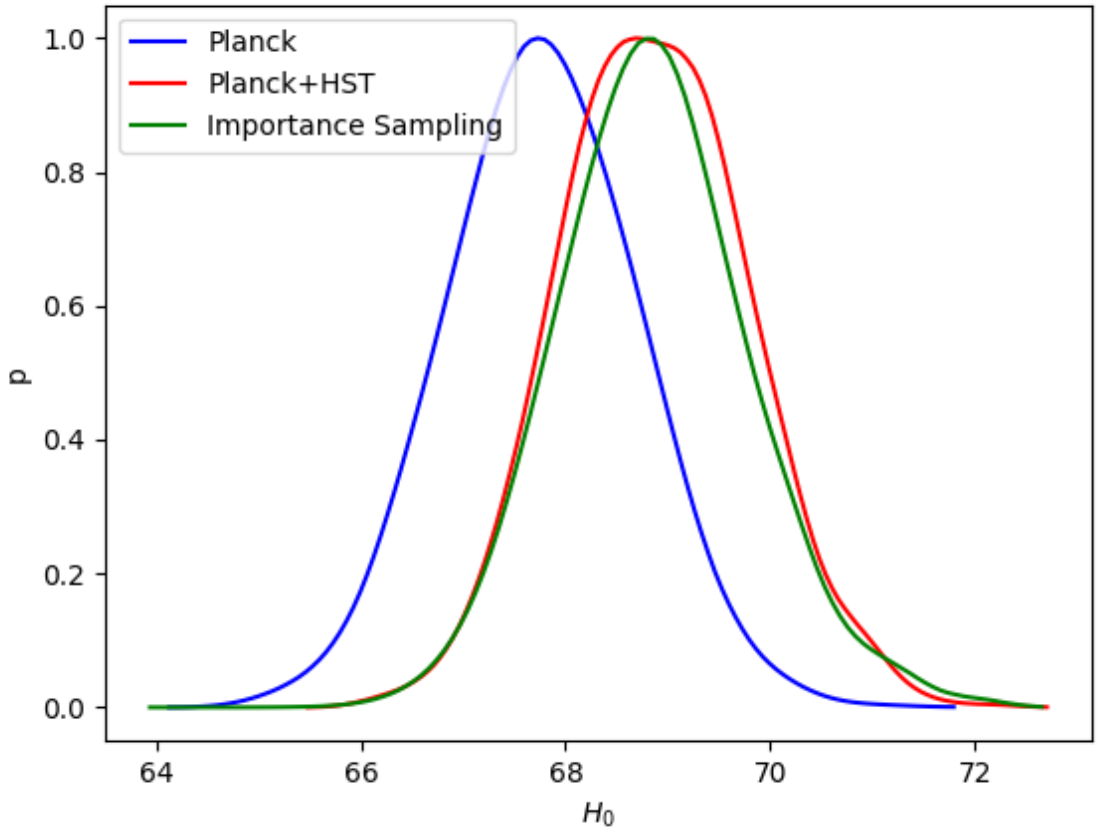
For example, c=73.8 km/s/Mpc, σ=2.4 km/s/Mpc.

$$\exp\left(-\frac{1}{2}\frac{(H_0 - c)^2}{\sigma^2}\right)$$

c=73.8 km/s/Mpc, σ=2.4 km/s/Mpc.



c=75.8 km/s/Mpc, σ=2.4 km/s/Mpc.



Exercise 2. Forecasting

Run a full case of a likelihood of an ideal experiment.

Note: You will need to use the *exact* approach for the likelihood. See this post: http://cosmocoffee.info/viewtopic.php?t=231 for detailed information on the likelihood. It uses the full-sky (exact) likelihood given by (Lewis 2005):

$$-2\log P(\hat{C}_l|C_l) = (2l+1)\left\{\mathrm{Tr}\left[\hat{C}_l C_l^{-1}\right] + \log|C_l|\right\}$$

Preparation of files: use python/makePerfectForecastDataset.py
Uses as an input an ideal or simulated files with Cls
(see slides on Healpix course – cosmic variance for an ideal experiment).

Including this realization inside .ini files:

cmb_dataset[MyForecast]=data/MyForecast/test_lensedCls_exactsim.dataset

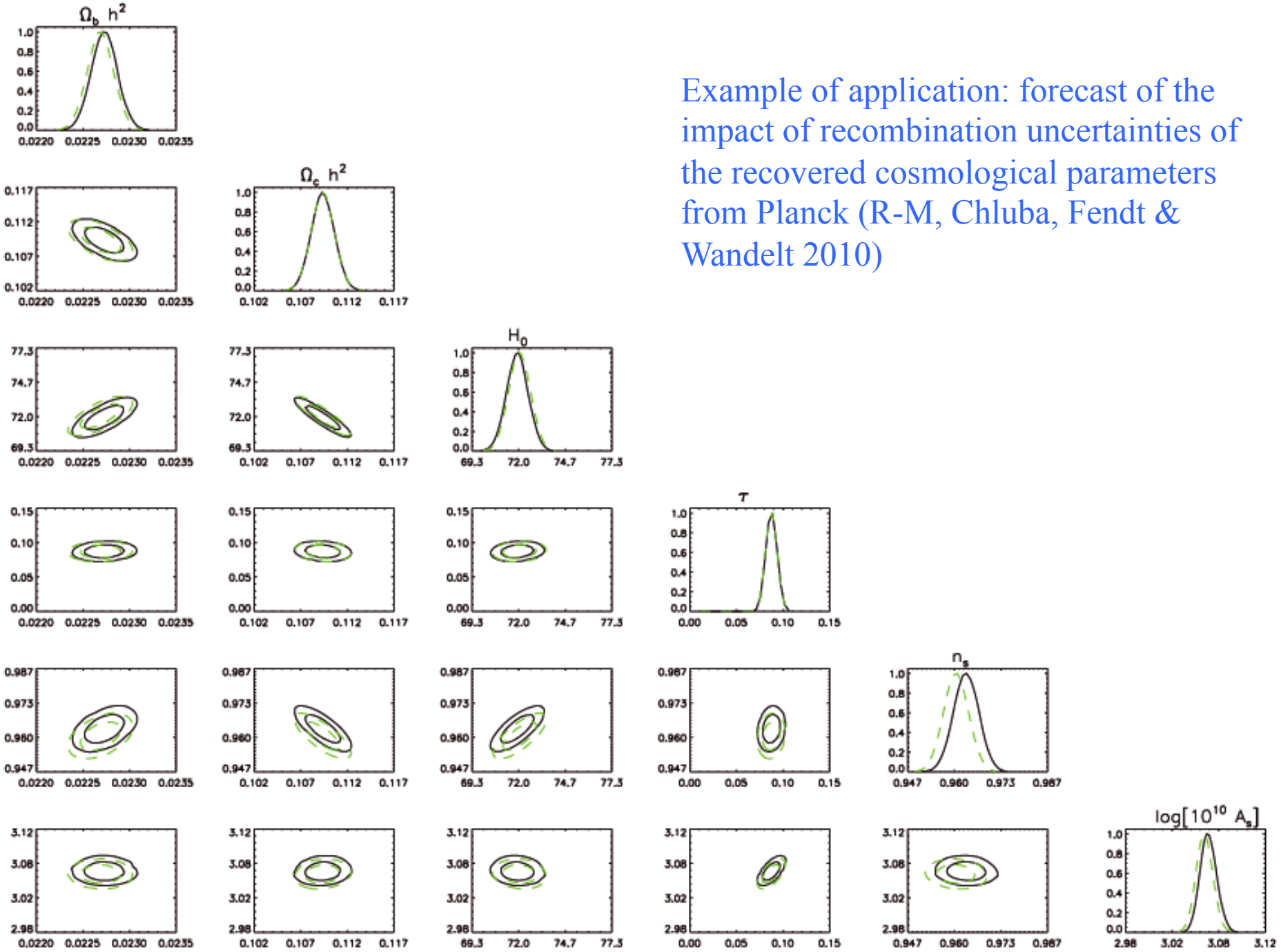
Cosmic variance (e.g. Knox 1995)

$$Var(C_\ell) = \frac{2}{2\ell+1} C_\ell^2$$

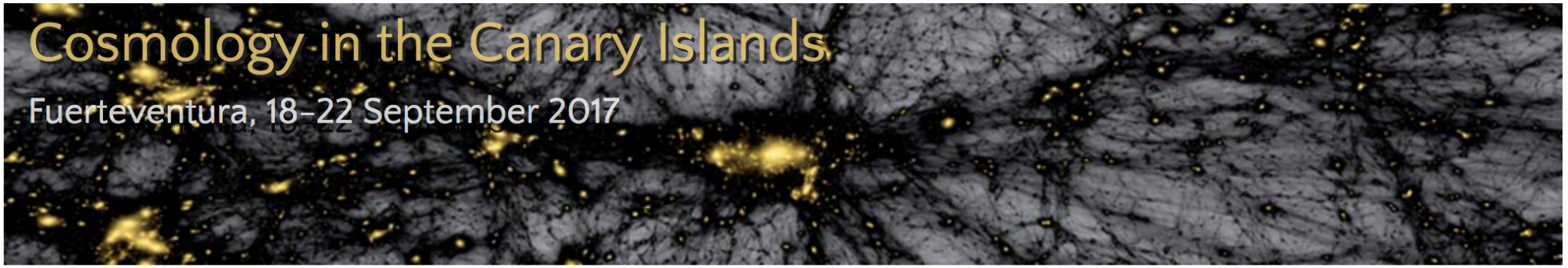
For the case of beam and instrumental noise, the expression changes to:

$$Var(C_\ell) = \frac{2}{2\ell+1} \left(C_\ell + \frac{1}{w B_\ell^2} \right)^2$$

$$w^{-1} = \sigma_{noise}^2 \Omega_{pix}$$



Example of application: forecast of the impact of recombination uncertainties of the recovered cosmological parameters from Planck (R-M, Chluba, Fendt & Wandelt 2010)



Presenta
Venue & Hotel Informa
Organizing Commit
Scientific Prog
Important D
Travel informa
Social Ev
Log in your acc

Search

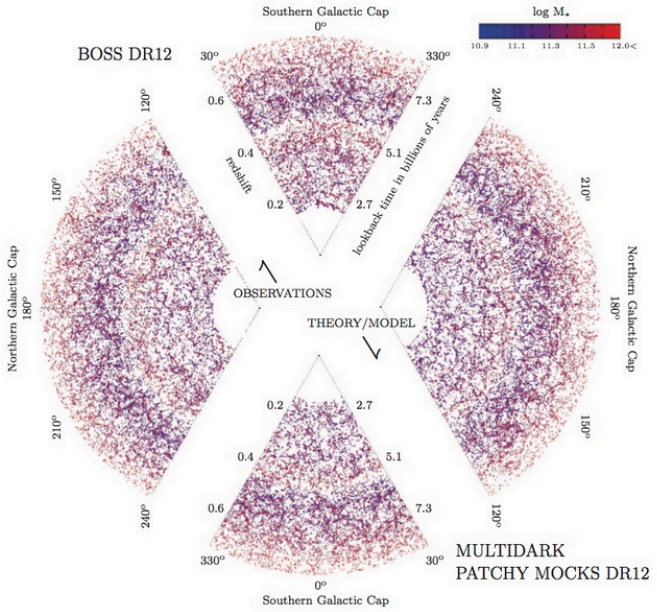
This summer school is especially addressed to PhD students and young postdocs and will have a workshop atmosphere with a large number of lecturers representing different branches of cosmology.

Confirmed lecturers:

- Nabila Aghanim
- Andrea Ferrara
- Will Percival
- Rien van de Weygaert
- Yun Wang

Focussed lectures:

- Raúl E. Angulo
- Jens Chluba
- Martín Crocce
- Bridget Falck
- Andreu Font Ribera
- Hector Gil Marín
- Carlos Hernández Monteagudo
- Francisco-Shu Kitaura
- Ruth Lazkoz
- Benton Metcalf
- Jenny Sorce
- Yi Zheng



http://www.iac.es/congreso/cosmo2017 - Registration will open soon!