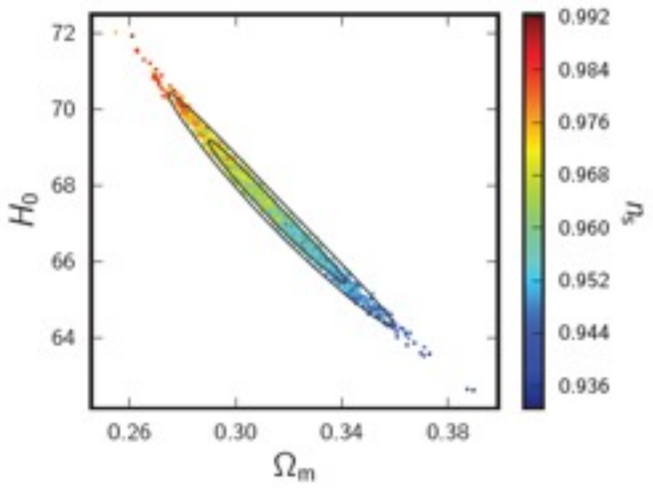
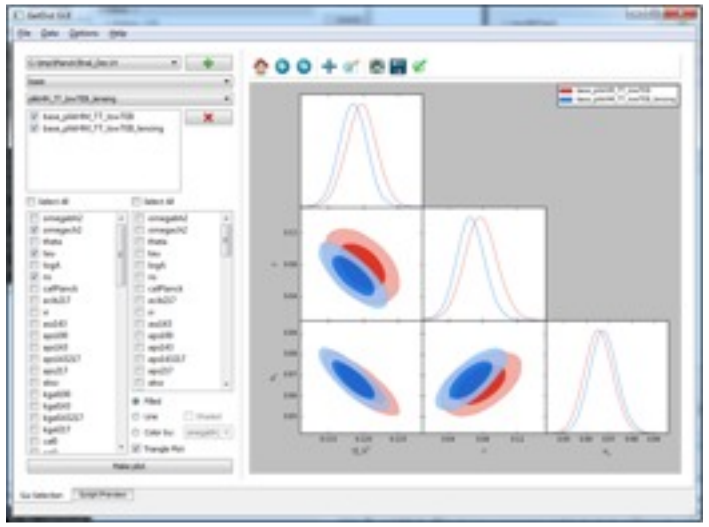




Introduction to CosmoMC

Part II: Installation and Execution

Antonio J. Cuesta

Institut de Ciències del Cosmos - Universitat de Barcelona

Institut de Ciències del Cosmos

EXCELENCIA MARÍA DE MAEZTU

UNIVERSITAT DE BARCELONA

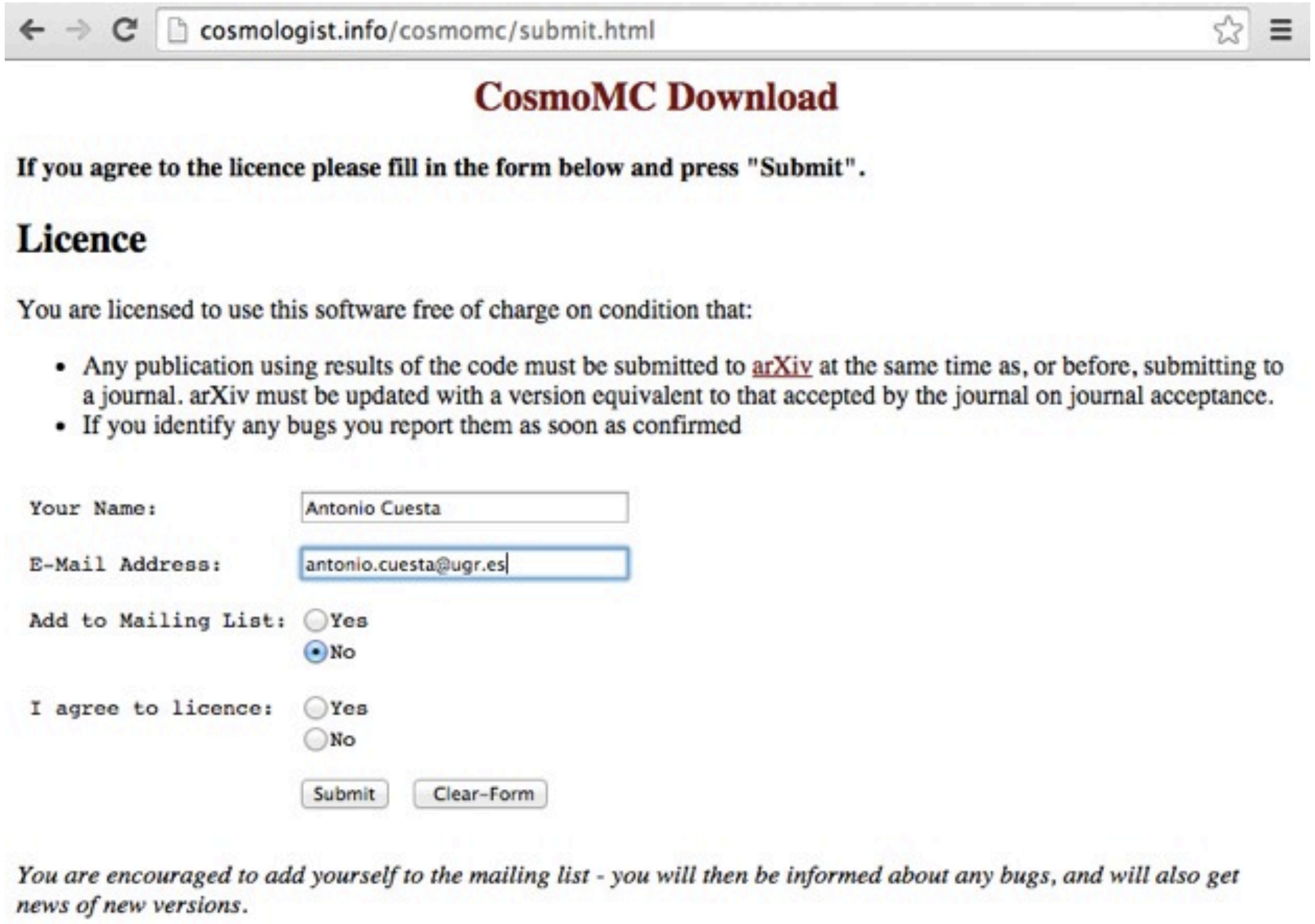
Dept. de Física Teórica y del Cosmos, Universidad de Granada, 1-3 Marzo 2016

Outline of Part II

- Downloading and Installing CosmoMC
- Customizing Params.ini: datasets and model parameters
- Running the code: the script runMPI.py
- Testing for chain convergence
- Finding the maximum likelihood values
- Post-processing: Importance sampling

Downloading CosmoMC

- Go to http://cosmologist.info/cosmomc/submit.html
  You will receive an email with links to download current and previous versions


cosmologist.info/cosmomc/submit.html

CosmoMC Download

If you agree to the licence please fill in the form below and press "Submit".

Licence

You are licensed to use this software free of charge on condition that:

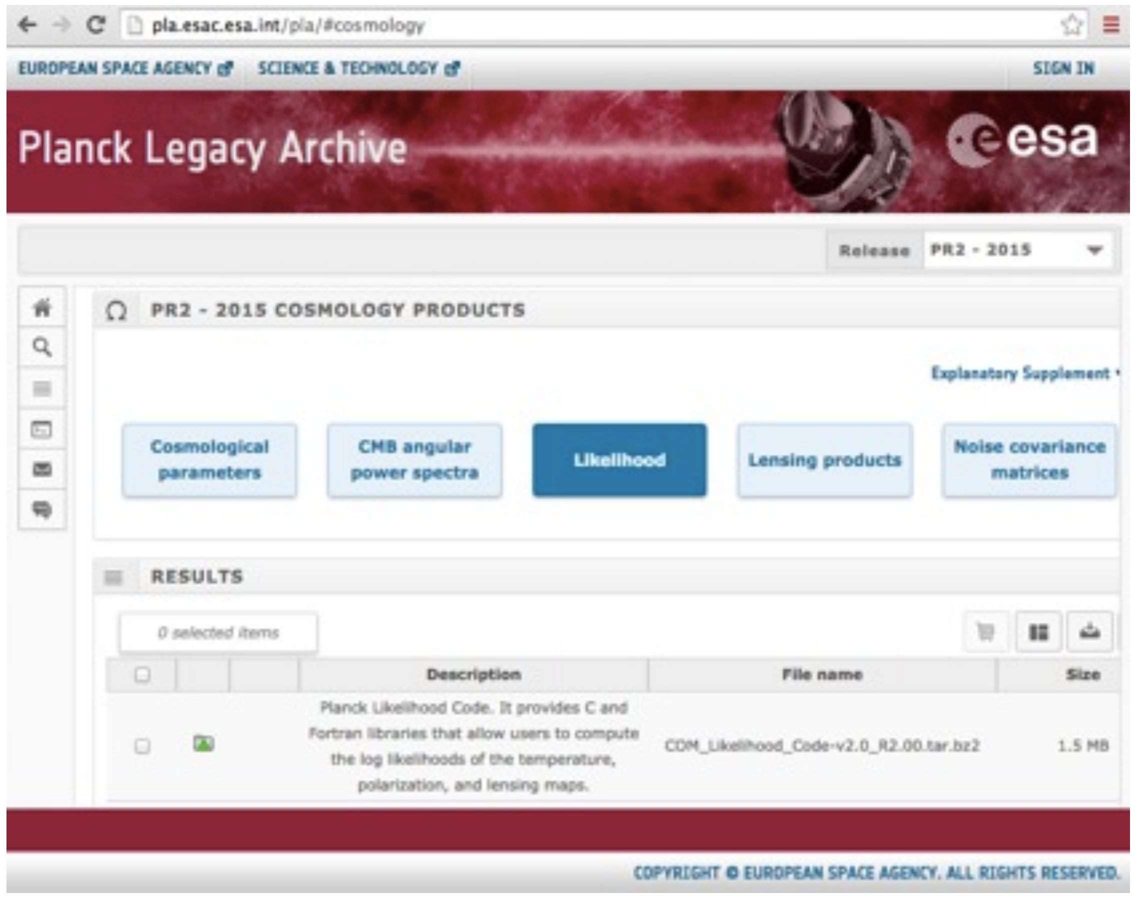
- Any publication using results of the code must be submitted to arXiv at the same time as, or before, submitting to a journal. arXiv must be updated with a version equivalent to that accepted by the journal on journal acceptance.
- If you identify any bugs you report them as soon as confirmed

Your Name:	Antonio Cuesta
E-Mail Address:	antonio.cuesta@ugr.es
Add to Mailing List:	Yes [✓ No]
I agree to licence:	Yes No

You are encouraged to add yourself to the mailing list - you will then be informed about any bugs, and will also get news of new versions.

Installing Planck likelihood

- This is technically NOT part of CosmoMC, but **you will need it** every time, so when you install CosmoMC, you want to install this all together.



			Description	File name	Size
			Planck Likelihood Code. It provides C and Fortran libraries that allow users to compute the log likelihoods of the temperature, polarization, and lensing maps.	COM_Likelihood_Code-v2.0_R2.00.tar.bz2	1.5 MB

- You have to download at least two files (300MB): the **likelihood** code `COM_Likelihood_Code-v2.0_R2.00.tar.bz2` and the **data** files `COM_Likelihood_Data-baseline_R2.00.tar.gz` (that contains hi_l, low_l, lensing)

Installing Planck likelihood

- Installation is done by running these two commands:

```
./waf configure --lapack_mkl=${MKLROOT} --install_all_deps
./waf install
```

- We will not need to do this, since it is already installed in *ftaecluster*, but we do need to set up the environment so that CosmoMC can find the files:

```
source ./bin/clik_profile.sh
```

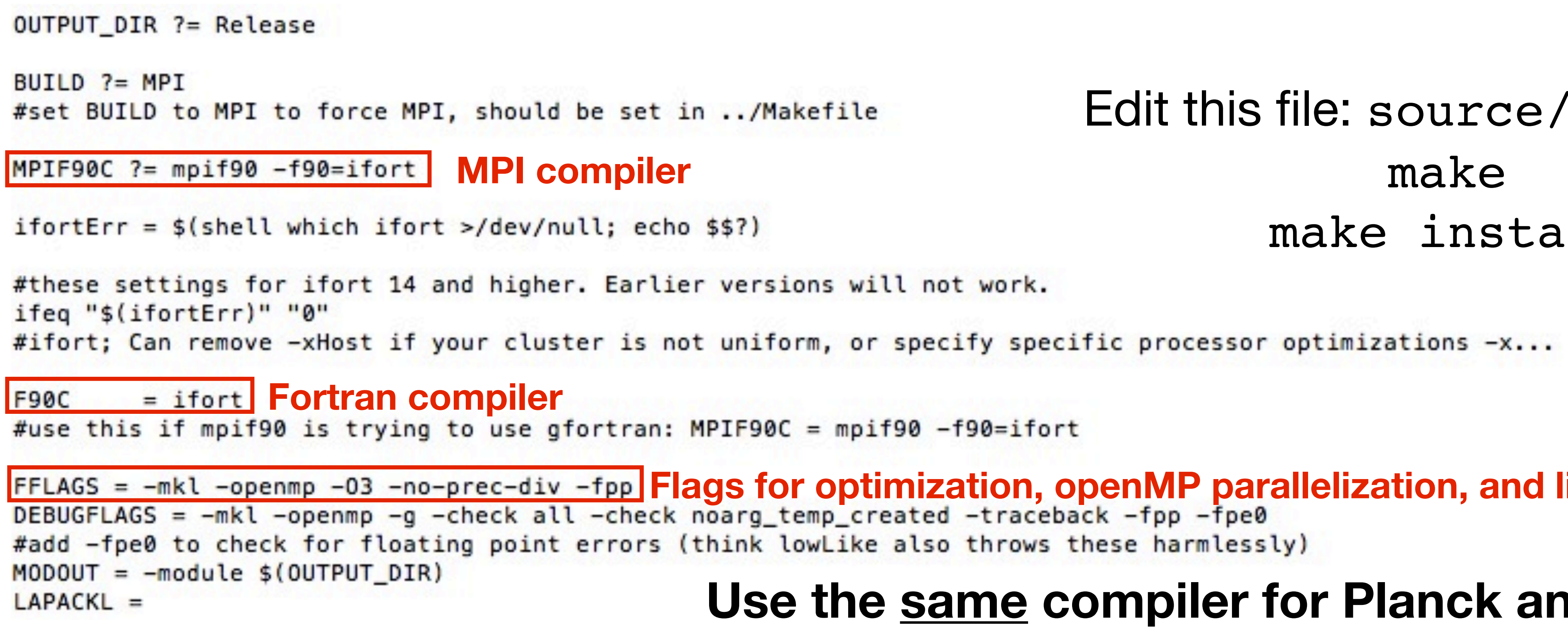
- You just need to create a symbolic link to your Planck2015 likelihood directory (plc_2.0) to a folder named '**clik**' inside **cosmomc/data/**

```
ln -s /path/to/planck/2015/plc_2.0 /path/to/cosmomc/data/clik
```

- NOW you can compile!

Installing CosmoMC

- Unfortunately current version of CosmoMC requires a commercial compiler (Intel Fortran Compiler, part of Intel Parallel Studio XE), a.k.a. “**ifort**”
- The (free) GNU Fortran compiler “**gfortran**” could in principle be used, but current released versions (including v5.3.0) do not work, but dev-v6.0 does. Performance is not expected to match ifort, but at least it is free



```
OUTPUT_DIR ?= Release

BUILD ?= MPI
#set BUILD to MPI to force MPI, should be set in ../Makefile

MPIF90C ?= mpif90 -f90=ifort
```

MPI compiler

```
ifortErr = $(shell which ifort >/dev/null; echo $$?)

#these settings for ifort 14 and higher. Earlier versions will not work.
ifeq "$(ifortErr)" "0"
#ifort; Can remove -xHost if your cluster is not uniform, or specify specific processor optimizations -x...

F90C     = ifort
```

Fortran compiler

```
#use this if mpif90 is trying to use gfortran: MPIF90C = mpif90 -f90=ifort

FFLAGS = -mkl -openmp -O3 -no-prec-div -fpp
```

Flags for optimization, openMP parallelization, and link to Intel MKL library

```
DEBUGFLAGS = -mkl -openmp -g -check all -check noarg_temp_created -traceback -fpp -fpe0
#add -fpe0 to check for floating point errors (think lowLike also throws these harmlessly)
MODOUT = -module $(OUTPUT_DIR)
LAPACKL =
```

Edit this file: `source/Makefile`

```
make
make install
```

Use the <u>same</u> compiler for Planck and CosmoMC !!

Parameter files

- This code has **two** main parameter files:

  - **params.ini**, the parameter file for **CosmoMC**,
    which sets up the *chains* (cosmological model and datasets)

  - **distparams.ini**, the parameter file for **GetDist**,
    which sets up the *analysis* (figures and statistics)

  - There are many other (minor) settings contained in other files. If you have a specific need, it will most likely be contained in those other files!

params.ini : File structure

- This file contains all the settings for your MCMC chains. It can have any name, but we will refer to it as **params.ini**.

- The settings can be (and actually are) nested, i.e. inside params.ini you can `INCLUDE` or `DEFAULT` **another** settings file. This is called **nesting**

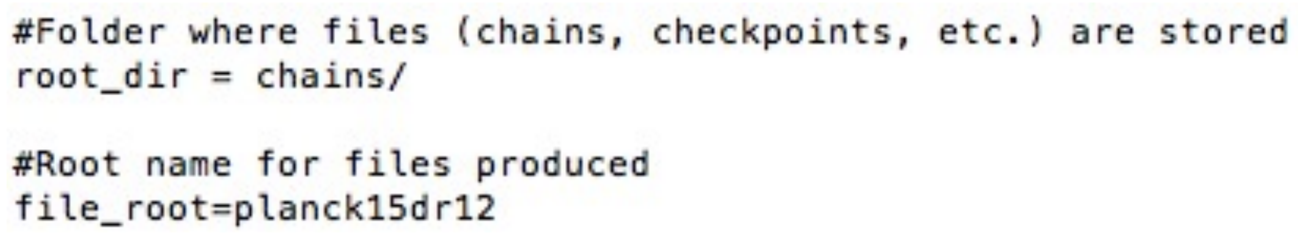
- `INCLUDE` will add the new settings if they are **not previously defined**

- `DEFAULT` will **override** previously defined settings if there is a conflict

- Note that the main file which is nested inside params.ini is **batch2/common.ini**, where the variable `MPI_Converge_Stop` is set.

params.ini : Basic settings

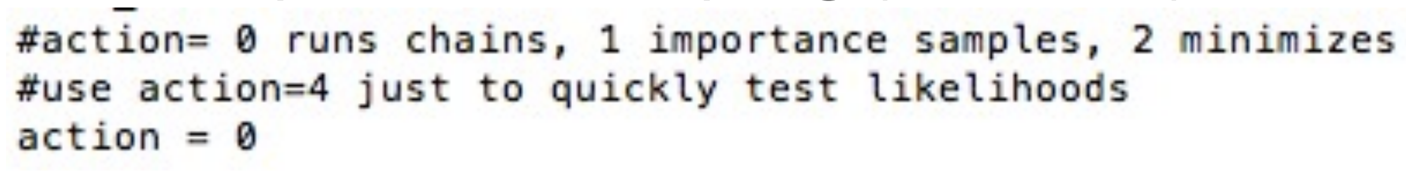
- First, you want to name your chain. Tip: the name should contain information about the **cosmological model**, and about the **datasets** used (e.g. ‘OLCDM_Planck2015_BAO’)



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

#Root name for files produced
file_root=planck15dr12
```

- Second, you want to **run** a MCMC chain. So make sure that the setting *action* in params.ini is set to zero (**action=0**). By default it is set to test likelihoods (**action=4**). Note that you can also find the best-fitting values of your cosmological parameters given the datasets (**action=2**) or add a new dataset to existing chains via importance sampling (**action=1**)



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

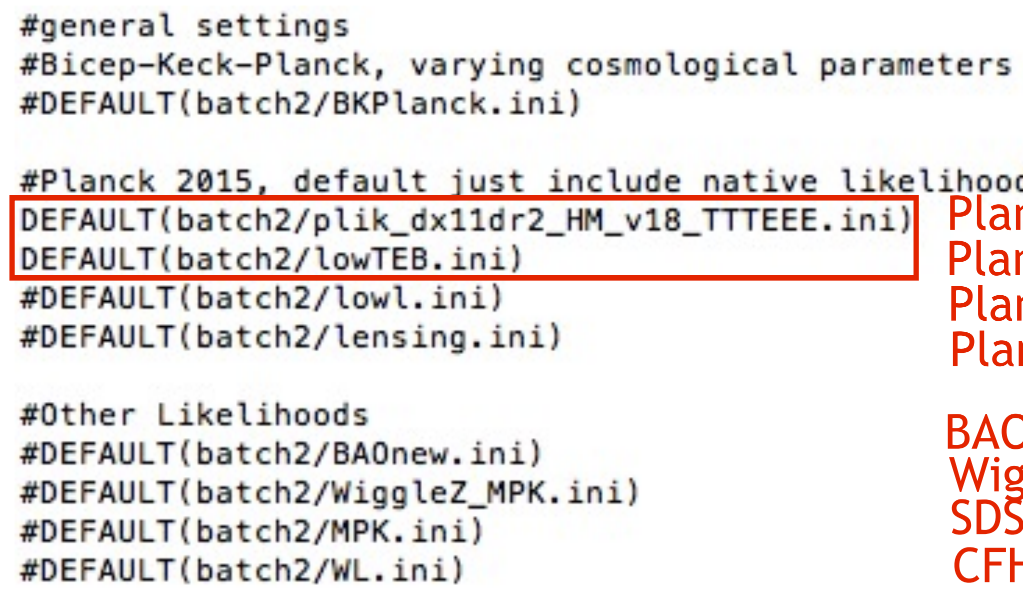
You can also change the **temperature** in your chains, i.e. sample the distribution **$P^{1/T}(x)$** rather than **P(x)**.

High temperature will explore tails better.

params.ini : Choosing your datasets

- Select the **datasets (likelihoods)** to be used. Comment out (using '#' symbol) the ones you don't need. Include the ones you want with `DEFAULT`

  All the built-in likelihoods are installed in the ***batch2/*** directory
  You can define yours too (take ***batch2/HST.ini*** as a good starting example)



```
#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/BAOnew.ini)
#DEFAULT(batch2/WiggleZ_MPK.ini)
#DEFAULT(batch2/MPK.ini)
#DEFAULT(batch2/WL.ini)
```

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.

for example, for CMB from PLANCK, you typically want to include a low-ℓ likelihood, a high-ℓ likelihood, and (sometimes) a CMB lensing likelihood.

params.ini : Setting your parameters

- In CosmoMC there are **three** types of parameters:

- **Cosmological** parameters: these are the ones you **want** to vary, as part of the cosmological model you choose. The default is the 6 ΛCDM base parameters

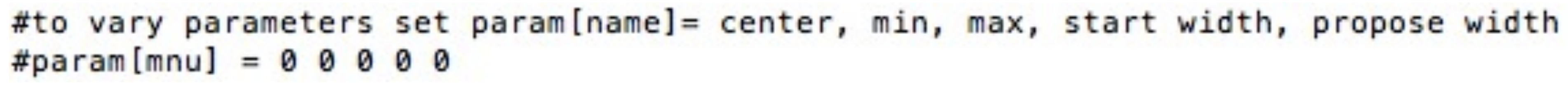
- **Nuisance** parameters: these you are forced to vary when you include some particular datasets (e.g. Planck, JLA) because this is the way to marginalize over systematics effects. They are added automatically when you use that data

- **Derived** parameters: these are not varied themselves, but they depend on cosmological parameters ($\Omega_m^{0.3}\sigma_8$), so you indirectly obtain constraints on them They are marked with a star (*), see `paramnames/derived_.paramnames`

They will all included in your chains explicitly, so you can check that everything is OK by looking at your chains or the *paramnames file generated in the directory cosmomc/chains

params.ini : Setting your parameters

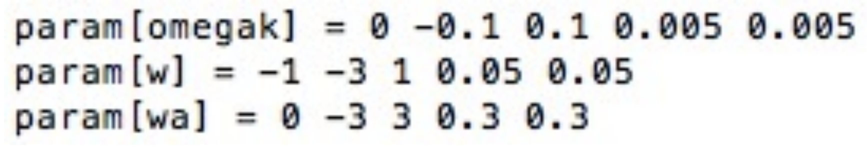
- The priors of the 6 LCDM parameters are defined in the file ***batch2/params_CMB_defaults.ini***
  Change them here if you need to, or (better) re-define them again in your params.ini, which has higher priority. Follow this syntax:



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

- Basically write a **fiducial** (a guess) starting value, a **left and a right bound** (hard priors), and two numbers you guess for the **uncertainties**

- You can of course include additional non-LCDM parameters defined in CAMB
  A complete list is in the file `paramnames/params_CMB.paramnames`



```
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
```

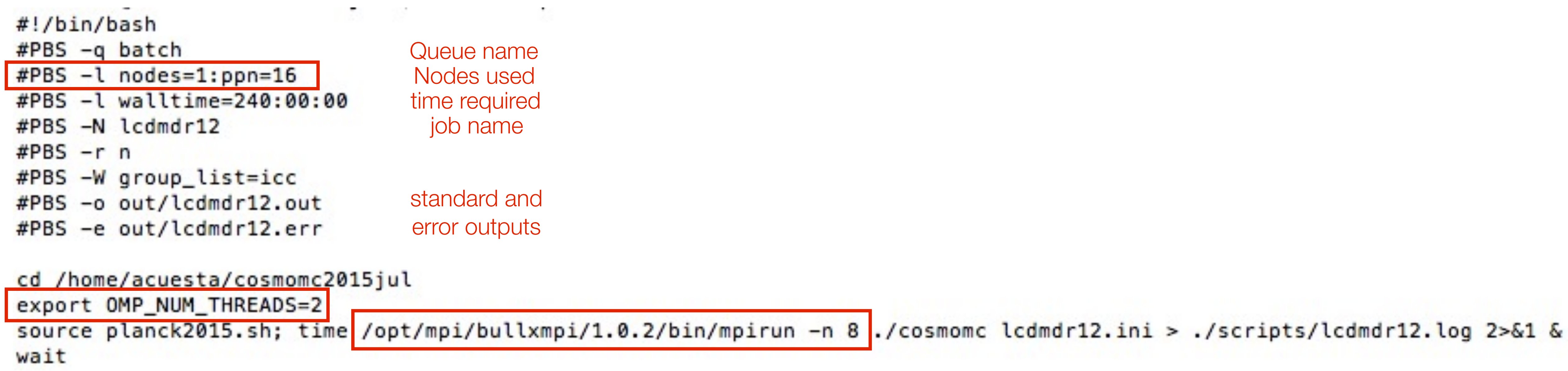
Running the code

- Remember that this is a ***PARALLEL*** code! Although it can be run in any machine (provided that it meets the compiling requirements), it only makes sense to run it in multi-processor machines (not on your laptop)
- The code has **OpenMP/MPI** parallelization implemented. That basically means that you can run the code in parallel using processors in the same node (OpenMP) or in different nodes (MPI). Or, better, combine both
- The typical workload distribution to get the best of both parallelization schemes: communication between chains uses **MPI** and each chain uses as many processors as possible = **#processors_per_node / #chains_per_node**
- In the practice session, we will use 1 node per person, so MPI communication will be INSIDE the same node

The script python/runMPI.py

- OpenMP is controlled by the environment (bash) variable `OMP_NUM_THREADS` MPI is set up with the option `-n` of `mpirun` / `mpiexec`

- In a computing cluster like *ftaecluster*, we typically need to write a **job file** for the torque/moab/slurm scheduler to set up the **resources** the code needs and the **command** to be run



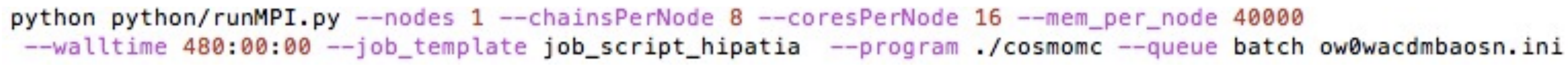
```
#!/bin/bash
#PBS -q batch
#PBS -l nodes=1:ppn=16
#PBS -l walltime=240:00:00
#PBS -N lcdmdr12
#PBS -r n
#PBS -W group_list=icc
#PBS -o out/lcdmdr12.out
#PBS -e out/lcdmdr12.err

cd /home/acuesta/cosmomc2015jul
export OMP_NUM_THREADS=2
source planck2015.sh; time /opt/mpi/bullxmpi/1.0.2/bin/mpirun -n 8 ./cosmomc lcdmdr12.ini > ./scripts/lcdmdr12.log 2>&1 &
wait
```

- This is what you'd always need to write in order to run a code in *ftaecluster*. But fortunately, CosmoMC simplifies this task by using the script **runMPI.py**

The script python/runMPI.py

- The script **python/runMPI.py** uses an alternative way to submit runs to the computing cluster by just using a 1-line command, e.g.:



```
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 ow0wacdmbaosn.ini
```

- This basically gives the same information as before, but you don't need to write a file to submit a run -- you just write a **command**

- However, this requires that the cluster configuration is correctly specified in the **job_script** file, which you have to customize (only once!) to your cluster

- In the case of ftaecluster, you can use the file **job_script_ftaecluster** provided. Remember to change this if the cluster configuration changes!

How to minimize runtime

- This is usually given by your own experience, but there are general guidelines:

- **OpenMP** speeds up **a chain** by computing in parallel each call to **CAMB** (‘*slow*’ steps, when the transfer function needs to be recomputed).

- **MPI** is used to communicate **between chains**. This happens rarely, when the code decides to update the covariance matrix of your parameters from the information in the already running chains, adapting the step size

- Remember also to include in your parameter file a `propose_matrix` as close as possible to your model+dataset combination

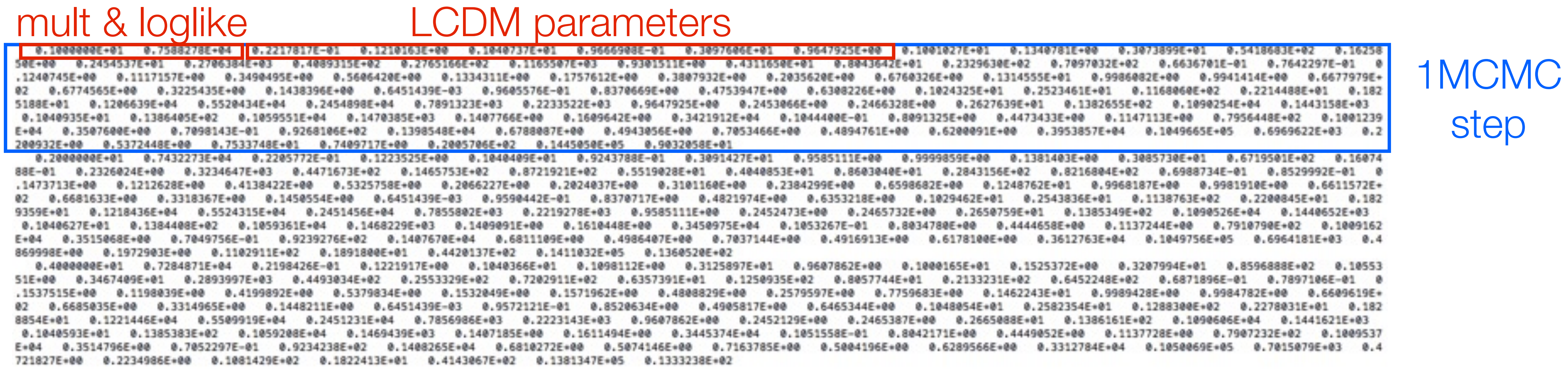
How to minimize runtime

- I would recommend this **typical set-up**:
- **at least 8 chains** per code execution (therefore you will start with 8 different initial conditions to sample your parameter space). If the node has 16 processors (ftae, nodes 1-7), that means `OMP_NUM_THREADS=2`. If the node has 32 processors (ftae, nodes 8-9) that means `OMP_NUM_THREADS=4`
- If there are many nodes available, in principle you can run **1 chain per node**
  ```
  #PBS -l nodes=8:ppn=16
  ```
- In practice, you want to run several cosmological models or several dataset combinations at the same time, so **all chains in the same node**
  ```
  #PBS -l nodes=1:ppn=16
  ```

MCMC Chain files

- A **chain file** is just a **text** file, you can open it with any editor or use it through any python **script** (or the way that is most convenient for you)

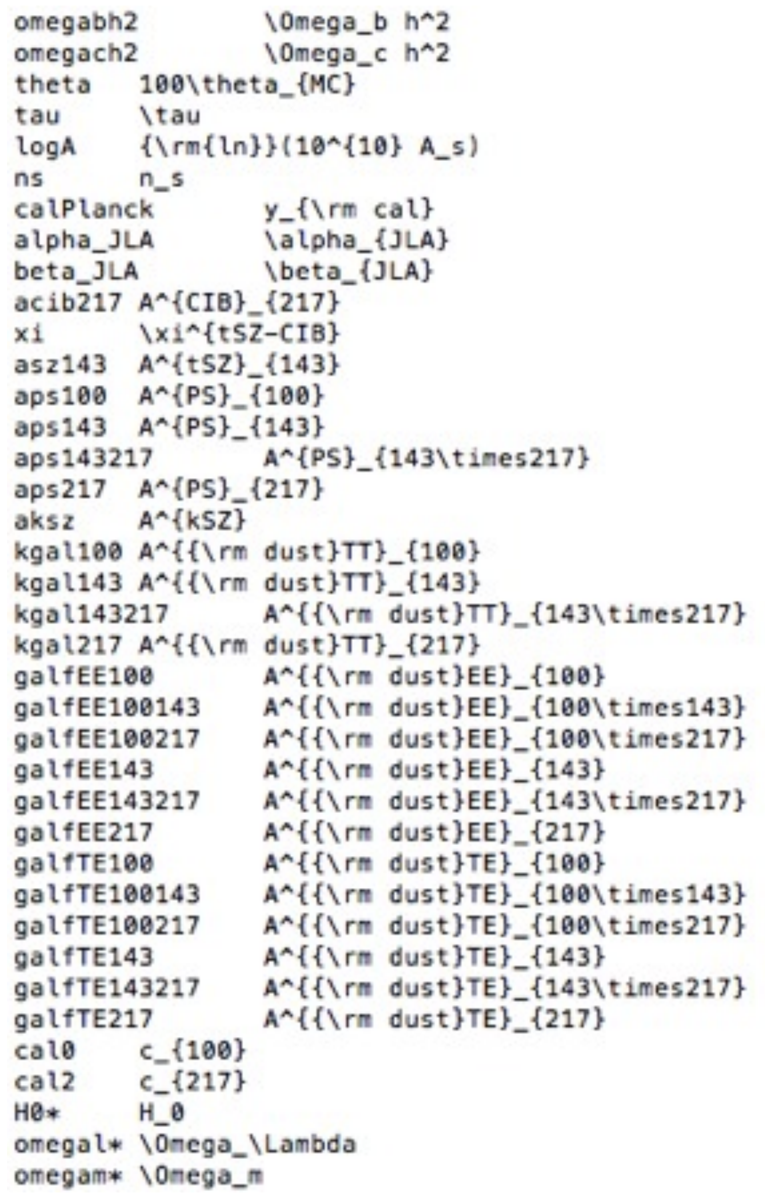
- Each **row** is a **MCMC step**. Each **column** is each one of the **parameters** in this order: cosmological, nuisance, and derived. The first two columns are the **multiplicity** (or number of steps spent in that point) and the **-log(likelihood)**, or “loglike”, which is just 0.5 times the total χ^2 value.



```
   0.1000000E+01   0.7588278E+04   0.2217817E-01   0.1210163E+00   0.1040737E+01   0.9666908E-01   0.3097606E+01   0.9647925E+00   0.1001027E+01   0.1340781E+00   0.3073899E+01   0.5418683E+02   0.16258
50E+00   0.2454537E+01   0.2706384E+03   0.4089315E+02   0.2765166E+02   0.1165507E+03   0.9301511E+00   0.4311650E+01   0.8043642E+01   0.2329630E+02   0.7097032E+02   0.6636701E-01   0.7642297E-01   0
.1240745E+00   0.1117157E+00   0.3490495E+00   0.5606420E+00   0.1334311E+00   0.1757612E+00   0.3807932E+00   0.2035620E+00   0.6760326E+00   0.1314555E+01   0.9986082E+00   0.9941414E+00   0.6677979E+
02   0.6774565E+00   0.3225435E+00   0.1438396E+00   0.6451439E-03   0.9605576E-01   0.8370669E+00   0.4753947E+00   0.6308226E+00   0.1024325E+01   0.2523461E+01   0.1168060E+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   0.1386405E+02   0.1059551E+04   0.1470385E+03   0.1407766E+00   0.1609642E+00   0.3421912E+04   0.1044400E-01   0.8091325E+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   0.6788087E+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   0.2005706E+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   0.2326024E+00   0.3234647E+03   0.4471673E+02   0.1465753E+02   0.8721921E+02   0.5519028E+01   0.4040853E+01   0.8603040E+01   0.2843156E+02   0.8216804E+02   0.6988734E-01   0.8529992E-01   0
.1473713E+00   0.1212628E+00   0.4138422E+00   0.5325758E+00   0.2066227E+00   0.2024037E+00   0.3101160E+00   0.2384299E+00   0.6598682E+00   0.1248762E+01   0.9968187E+00   0.9981910E+00   0.6611572E+
02   0.6681633E+00   0.3318367E+00   0.1450554E+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   0.2200845E+01   0.182
9359E+01   0.1218436E+04   0.5524315E+04   0.2451456E+04   0.7855802E+03   0.2219278E+03   0.9585111E+00   0.2452473E+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   0.1409091E+00   0.1610448E+00   0.3450975E+04   0.1053267E-01   0.8034780E+00   0.4444658E+00   0.1137244E+00   0.7910790E+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   0.1891800E+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   0.1250935E+02   0.8057744E+01   0.2133231E+02   0.6452248E+02   0.6871896E-01   0.7897106E-01   0
.1537515E+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   0.9984782E+00   0.6609619E+
02   0.6685035E+00   0.3314965E+00   0.1448211E+00   0.6451439E-03   0.9572121E-01   0.8520634E+00   0.4905817E+00   0.6465344E+00   0.1048054E+01   0.2582354E+01   0.1288300E+02   0.2278031E+01   0.182
8854E+01   0.1221446E+04   0.5509919E+04   0.2451231E+04   0.7856986E+03   0.2223143E+03   0.9607862E+00   0.2452129E+00   0.2465387E+00   0.2665088E+01   0.1386161E+02   0.1090606E+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   0.8042171E+00   0.4449052E+00   0.1137728E+00   0.7907232E+02   0.1009537
E+04   0.3514796E+00   0.7052297E-01   0.9234238E+02   0.1408265E+04   0.6810272E+00   0.5074146E+00   0.7163785E+00   0.5004196E+00   0.6289566E+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
```

MCMC Chain files

- The column ordering can be found in the file **chains/root.paramnames**



```
omegabh2        \Omega_b h^2
omegach2        \Omega_c h^2
theta   100\theta_{MC}
tau     \tau
logA    {\rm{ln}}(10^{10} A_s)
ns      n_s
calPlanck       y_{\rm cal}
alpha_JLA       \alpha_{JLA}
beta_JLA        \beta_{JLA}
acib217 A^{CIB}_{217}
xi      \xi^{tSZ-CIB}
asz143  A^{tSZ}_{143}
aps100  A^{PS}_{100}
aps143  A^{PS}_{143}
aps143217       A^{PS}_{143\times217}
aps217  A^{PS}_{217}
aksz    A^{kSZ}
kgal100 A^{{\rm dust}TT}_{100}
kgal143 A^{{\rm dust}TT}_{143}
kgal143217      A^{{\rm dust}TT}_{143\times217}
kgal217 A^{{\rm dust}TT}_{217}
galfEE100       A^{{\rm dust}EE}_{100}
galfEE100143    A^{{\rm dust}EE}_{100\times143}
galfEE100217    A^{{\rm dust}EE}_{100\times217}
galfEE143       A^{{\rm dust}EE}_{143}
galfEE143217    A^{{\rm dust}EE}_{143\times217}
galfEE217       A^{{\rm dust}EE}_{217}
galfTE100       A^{{\rm dust}TE}_{100}
galfTE100143    A^{{\rm dust}TE}_{100\times143}
galfTE100217    A^{{\rm dust}TE}_{100\times217}
galfTE143       A^{{\rm dust}TE}_{143}
galfTE143217    A^{{\rm dust}TE}_{143\times217}
galfTE217       A^{{\rm dust}TE}_{217}
cal0    c_{100}
cal2    c_{217}
H0*     H_0
omegal* \Omega_\Lambda
omegam* \Omega_m
```

This file has two columns: the internal name of the parameter and a LaTeX script so that the plots display their name properly (edit this file to your convenience)

Convergence: Gelman-Rubin criterion

- The ***R-1* estimator** (Gelman and Rubin 1992) is defined roughly as "the variance of the chain means divided by the mean of the variances". A set of chains is declared **converged** when this estimator is **small** enough (typically 0.01)
- This is computed for **EACH** parameter. Typically the value you quote to assess the convergence of your chain is the worst (**largest**) value over all parameters
- Each chain will have a (slightly) different mean but the dispersion between chains is small compared to the mean error bar when converged

$$\hat{R} = \sqrt{\frac{\hat{\mathrm{Var}}(\theta)}{W}}$$

$$\hat{\mathrm{Var}}(\theta) = (1 - \frac{1}{n})W + \frac{1}{n}B$$

where W is the "within chain" variance and B is the "between chains" variance of the means

if the chain is more or less converged R will be a little bit over 1.0
convergence is measured by the value of R-1

How to check progress of your chains

- In the directory **cosmomc/scripts** there will be a file constantly updated named **root.log**, where *root* is the name you gave to your chains in *params.ini*
- Open that file and search the word "***convergence***". You will find several instances like this:

```
Current convergence R-1 =  0.1990366 chain steps = 1370
```

- The last occurrence gives you an idea of the current chain convergence and how far you are from the target value `MPI_Converge_Stop` (default 0.01)
- Another (better) way to obtain R-1 is to run **GetDist** (session 3)

Find maximum likelihood (action=2)

- MCMC is **not optimized** to find a maximum, but to sample parameter spaces (Metropolis-Hastings is NOT a gradient method like downhill simplex etc). The bestfit value given at the **likestats** file (see session 3) will be **approximate**

- This uses Powell 2009 BOBYQA minimization. The code starts at the "center" parameter values in the .ini file, and requires that the likelihood does not return NaN in the region between those values

- It stops at a given value given by `minimize_loglike_tolerance` given that the true minimum is within `max_like_radius` (in units of the parameter error). All these settings are found at **batch2/common.ini**

- The output is then given in **root.minimum**

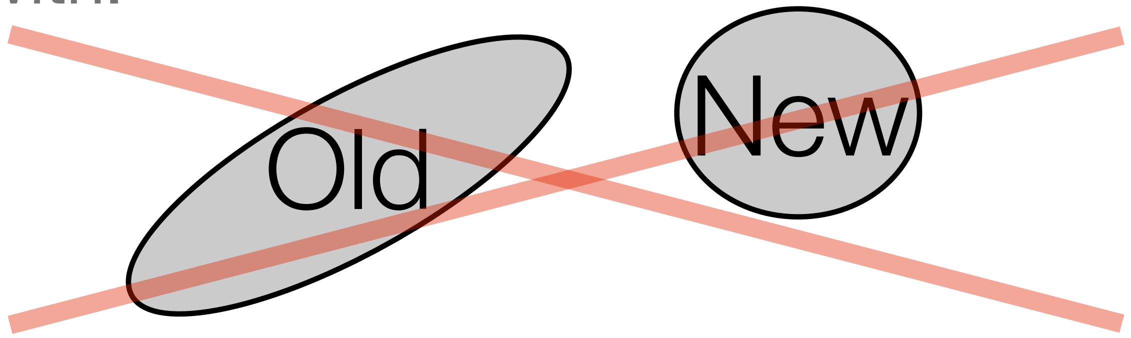
Importance sampling (action=1)

- Sometimes you want to explore the effect of an **additional** dataset on your chains that you **already** run with your favorite model+dataset combination. (For example: you want to see the effect of a recent measurement of H0 on your chains that you ran for CMB+BAO+SN in the ΛCDM model)

- In principle you would need to **run a new chain**, which is slow.
  But this can be done much faster by **re-weighting** each step of your chains (the re-weighting factor depends on the new data),
  only if the following two requirements are met:

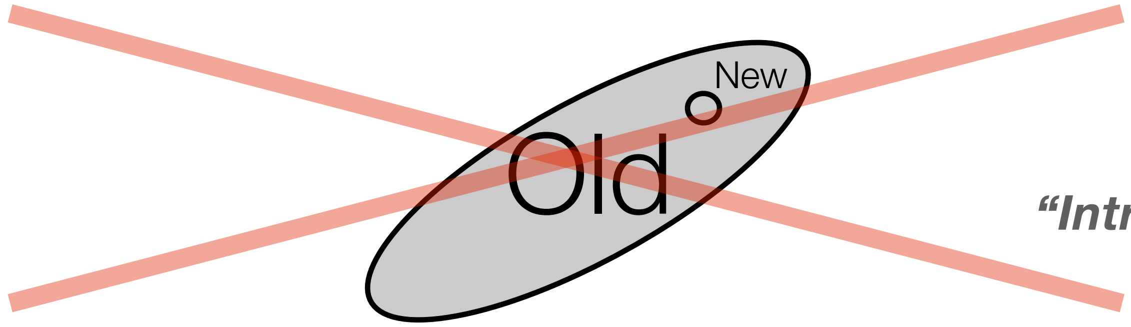
Importance sampling (action=1)

- 1) The new measurement is **consistent** with the constraints from the chain. That means, if your new data falls away from the allowed parameter space, your resulting constraints will not be reliable since you did not have points to reweight to begin with!



- 2) The new data is **not too constraining**. If the error bar of the new data is small compared with the allowed region in your chains, most points will be re-weighted to zero except for a small subset, and your constraints will be noisy.

The opposite case (new data has a very large error bar) can be applied though! but all the points will be reweighted by the same value, so that will have no effect whatsoever



Importance sampling (action=1)

- If you have ***.data** files this is **fast** and easy (~10 min). In your parameter file set `action=1` and `redo_add=T` and include the likelihoods you want to add. Or if you set `redo_add=F` just include all the likelihoods you want included (if a likelihood was included in the old chains, it will not be included twice)

- To create ***.data** files set the variable `indep_sample` to some number where the steps are not correlated (GetDist will tell us this), also because *.data files will contain C_ℓ and P(k,z) and occupy disk space. Set it to more than 10.

- If you don't have .data files you can still use `redo_from_text` to do the sampling from the chain files (*.txt), but this is slow and does not work well

Background parametrization

- When no CMB likelihoods are going to be used, then a good choice is the `parametrization=background` option. This is useful to constrain only parameters **not** related to perturbations (e.g. not {A_s n_s τ} but {H_0 Ω_m Ω_k Ω_v w})

- `parametrization=background` is included in **batch2/common.ini**. Also useful would be to create a file similar to **params_CMB_defaults.ini** in which the perturbation parameters {A_s n_s τ} are set to constant values, and force `bbn_consistency` to F

- The current version of the file **likelihood_paramsCMB.f90** sets $\Omega_c h^2=0$, and $\tau=0$, and $\Omega_b=\Omega_m-\Omega_v$. This is obviously a bug, so it needs to be replaced and **recompiled**. Also note that Ω_b needs to be constrained by some dataset, or just fixed to some standard value.