CosmoSIS

Joe Zuntz

Outline

- Understanding CosmoSIS
- Using CosmoSIS
- Modifying CosmoSIS

Purpose

- **Connect** other codes together to form Likelihood Pipeline
- Run many **sampling methods**
- Package **standard code library**

Relationships to other codes

- Connects to Camb & Class
- Alternative to Montepython & CosmoMC
    - Not specific to cosmology
    - More flexible
    - Easier to implement ranges of new models
    - Harder for simple projects

Likelihoods

- Likelihood = P(data | parameters)
- Final step (prediction vs observable) sometimes simple function
- Prediction as function of parameters usually complicated
    - Sequence of calculations
    - CosmoSIS = framework for building and sampling likelihood function pipelines

Philosophy

- Many pipelines start with common components like CAMB, CLASS, Astropy
    - Calculate b/g evolution, matter power, growth function, others
- Much easier to **build on** these than **modify/extend** them
    - Use as CosmoSIS modules & do calculations from their output

Components

- Replaceable parts with explicit inputs and outputs
    - Multi-lingual plugin framework
    - Simple data-passing API
- Unified configuration
- Unified interface to many sampling (and other) methods

Philosophy

- Make conceptually separate calculations actually separate in code
- Example: Weak Lensing Likelihood

Example

1. CAMB
2. Halofit
3. Extrapolate P(k)
4. Load n(z)
5. n(z) Biases
6. NLA model
7. 3D P(k) -> 2D C_ℓ
8. Add IA
9. Calibration Biases
10. Correlation Functions
11. Likelihood

If you are looking at the PDF slides you'll miss the cool animation here

Modules

- Each one of these is wrapped as a **CosmoSIS Module**
- Self-contained chunk of code, python, c, c++, f90
- Inputs & outputs from & to CosmoSIS

Modules

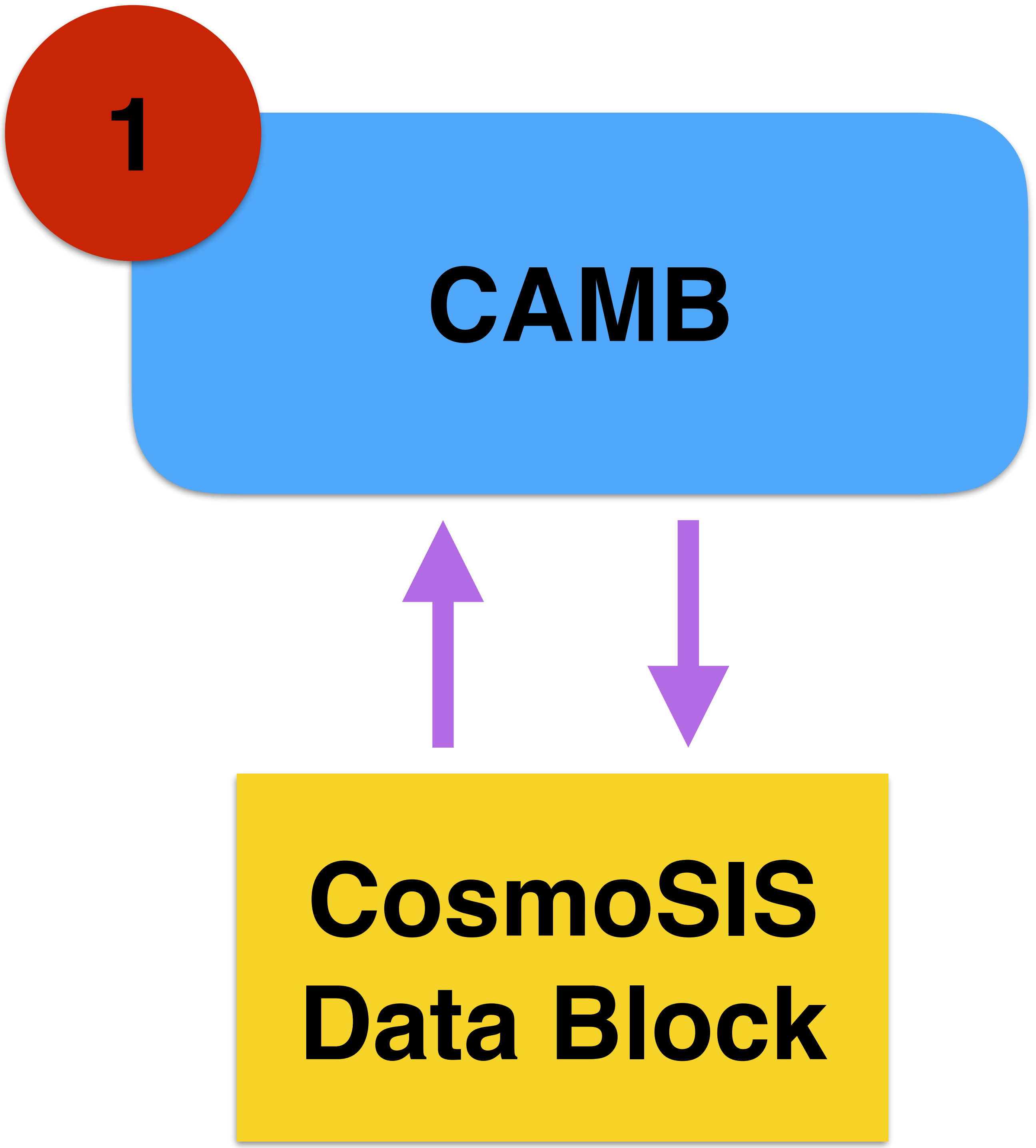
- Large collection provided with CosmoSIS
- You can add your own new ones very easily
- Typically last modules in pipeline compute final likelihood (e.g Gaussian)

Connections

- Two things connect modules into CosmoSIS
  - Module functions:
    `setup` - configure module, at start
    `execute` - run module, for each param set
  - Use collection of functions (API) to
    read inputs from CosmoSIS
    & write results to CosmoSIS

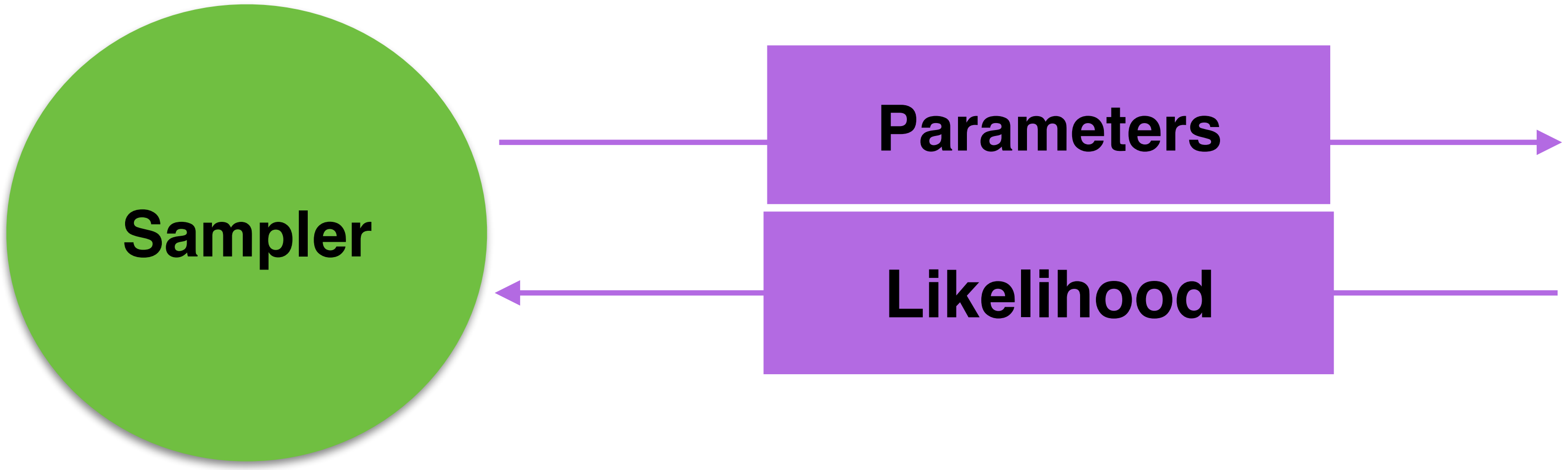
Connections

- Cosmology theory is stored in a *Data Block*
- CAMB:
    - reads cosmological parameters from block
    - writes power spectra (CMB & matter) and distance measures to block
- Block stores numbers, strings, and arrays with two keys, *section* & *name*
    - e.g section=cosmological_parameters name=omega_m



Samplers

- “Sampler” = Anything that generates 1+ sets of sample parameters
    - e.g MCMC, grids, maximum likelihood
    - always adding new samplers



Unified Sampler Interface

Samplers

- Emcee
- Metropolis
- Importance
- Kombine
- Multinest
- Polychord
- PMC

Other Posterior Explorers

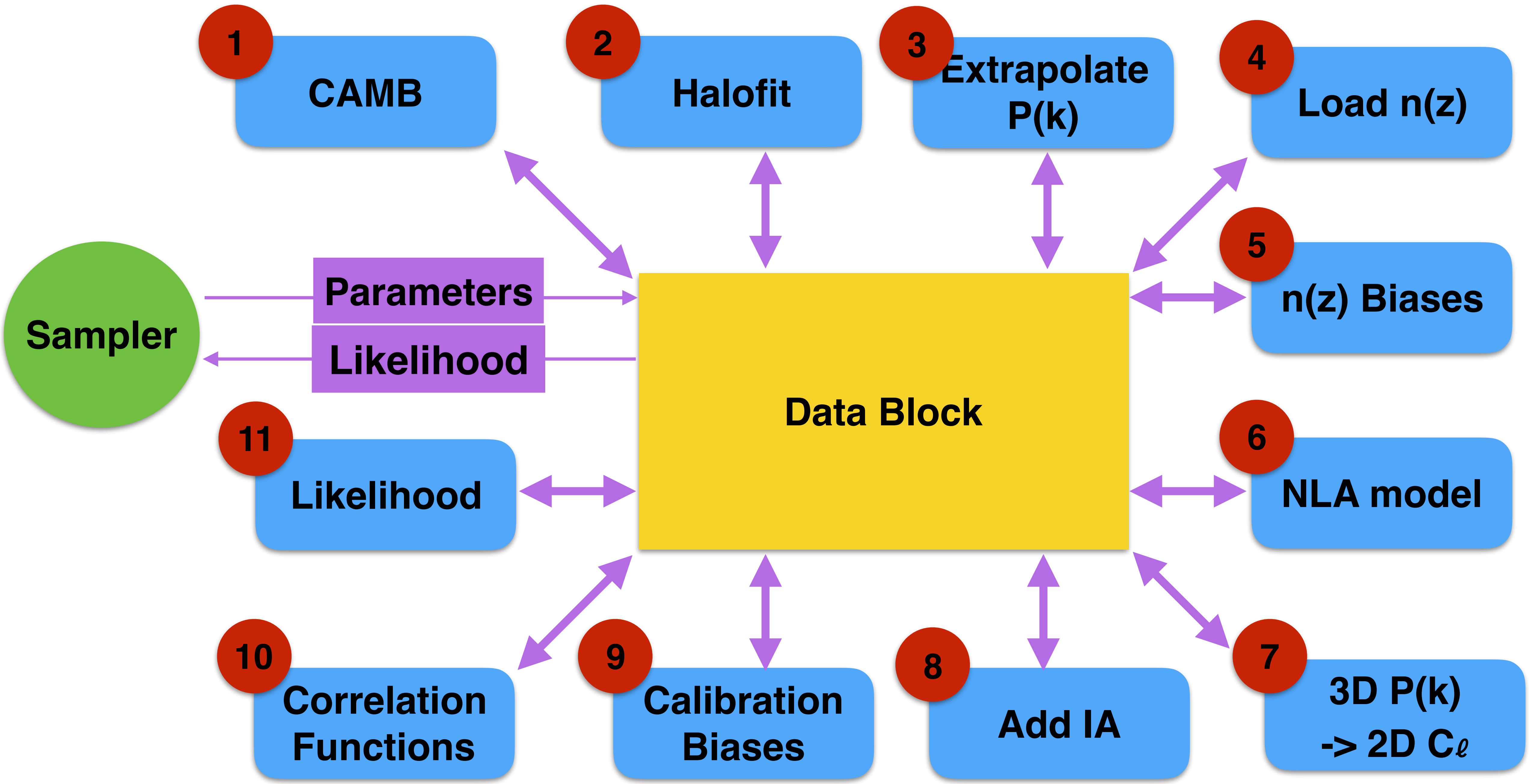
- Grid
- Fisher Matrix
- Snake
- Minuit

Maximum-Likelihood

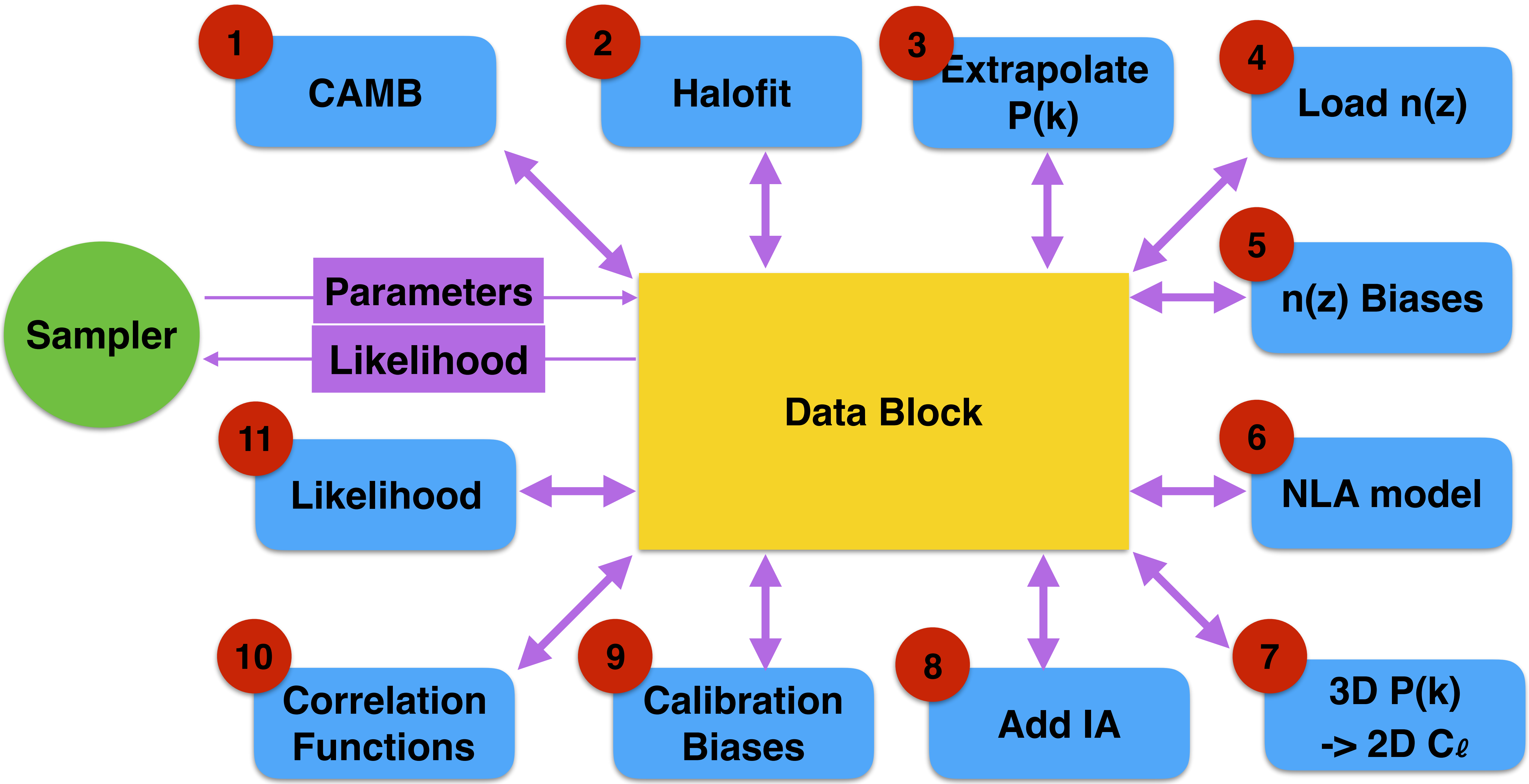
- Scipy Max-like
- Grid-search
- Parallel ML

Other Tools

- Test - single point
- List of points
- Star - 1D lines through point



If you are looking at the PDF slides you'll miss the cool animation here



1
CAMB
2
Halofit
3
Extrapolate P(k)
4
Load n(z)
5
n(z) Biases
Sampler
Parameters
Likelihood
Data Block
11
Likelihood
6
NLA model
10
Correlation Functions
9
Calibration Biases
8
Add IA
7
3D P(k) -> 2D Cℓ

Using CosmoSIS

Documentation

- See the nearly adequate docs on our wiki:

  https://bitbucket.org/joezuntz/cosmosis/

Process

- Figure out your pipeline
- Write configuration files
- Run `cosmosis` to make chains
- Run `postprocess` to make plots

Figuring out pipeline

- Combine existing modules
  https://bitbucket.org/joezuntz/cosmosis/wiki/default_modules
- Write new modules
- Think how they connect together to form a pipeline
- Maybe start from existing pipeline, adding new modules to start, middle, or end

Three Configuration Files

- Parameters - configure modules and sampler
- Values - configure input (fixed and varying) (cosmological, nuisance, etc) params
- Priors (optional) - set priors on Values

Parameter File

```
[runtime]
sampler = metropolis
```

Choose & configure samplers

```
[metropolis]
nsteps=10
random_start=F
samples=100000
covmat=examples/covmat_a.txt
Rconverge = 0.02
```

Output file

```
[output]
format=text
filename=example_output_a.txt
verbosity=debug
```

Pipeline to run

```
[pipeline]
modules = consistency camb wmap
values = examples/values_a.ini
likelihoods = wmap9
extra_output = cosmological_parameters/omega_m cosmological_parameters/omega_b
quiet=F
debug=F
timing=F
```

Parameter File

```
[runtime]
sampler = metropolis
```

Choose & configure samplers

```
[metropolis]
nsteps=10
random_start=F
samples=100000
covmat=examples/covmat_a.txt
Rconverge = 0.02
```

```
[output]
format=text
filename=example_output_a.txt
verbosity=debug
```

Output file

These indicate section [names] within this same file

```
[pipeline]
modules = consistency camb wmap
values = examples/values_a.ini
likelihoods = wmap9
extra_output = cosmological_parameters/omega_m cosmological_parameters/omega_b
quiet=F
debug=F
timing=F
```

Pipeline to run

Parameter File

Configure individual modules

```
[consistency]
file=cosmosis-standard-library/utility/consistency/consistency_interface.py
verbose=F

[camb]
file = cosmosis-standard-library/boltzmann/camb/camb.so
mode=cmb
lmax=1300
feedback=0

[wmap]
file = cosmosis-standard-library/likelihood/wmap9/wmap_interface.so
```

Values File

```
[cosmological_parameters]
ommh2 =  0.05        0.129     0.25
h0    =  0.5         0.726     0.9
ombh2 =  0.01        0.0224    0.04
tau   =  0.05        0.085     0.11
n_s   =  0.8         0.975     1.1
a_s   =  2.0e-09     2.12-09   2.4e-09
```

Varying parameters

min, start, max

```
omega_k   =  0.0
w  =  -1.0
wa  =  0.0
```

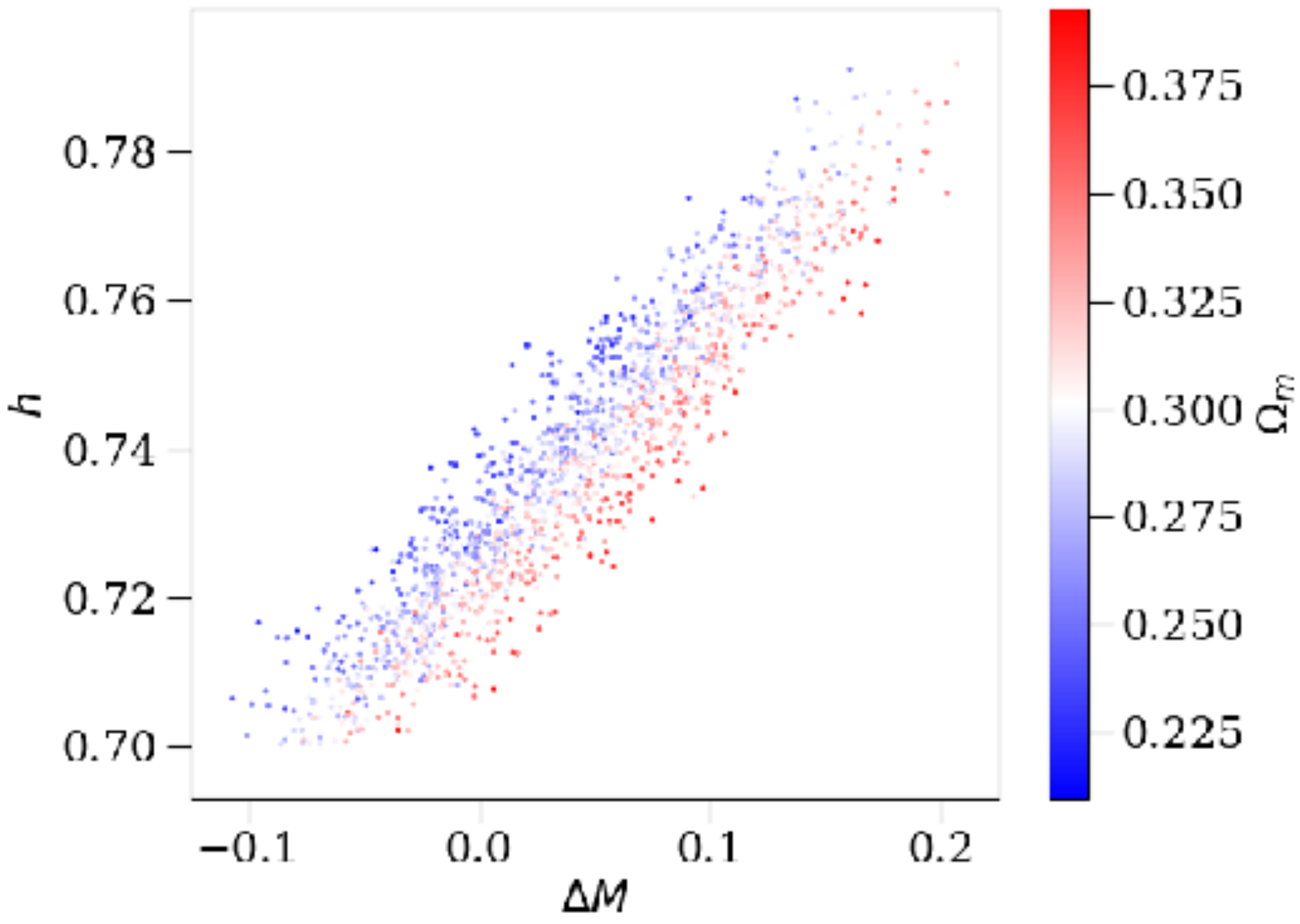
Fixed parameters

Priors File

```
#examples only!
[cosmological_parameters]
h0 = gaussian 0.738  0.024
omnhu2 = exponential 0.005
omega_b = uniform 0.04 0.05
```

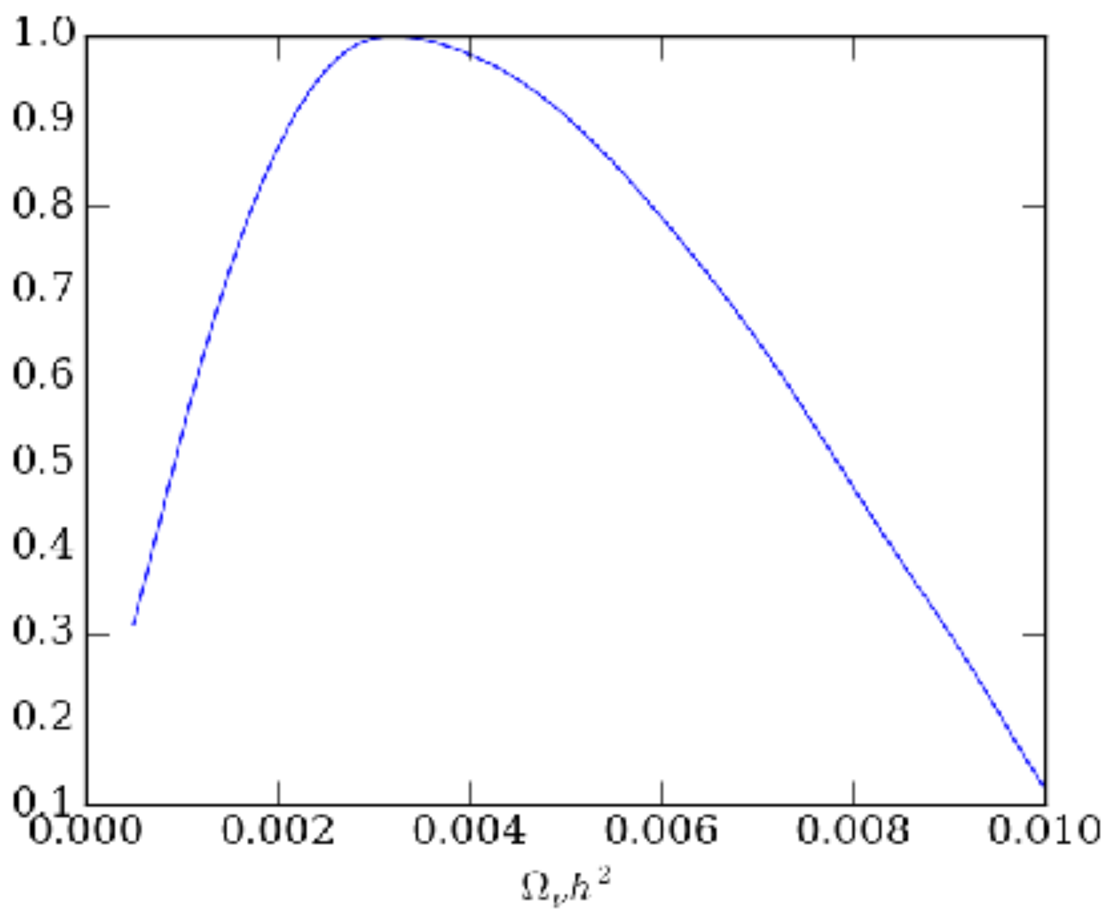
Running

```
> cosmosis params.ini
      (Single process)

> mpirun -n 8 cosmosis --mpi params.ini
      (Parallel)



> postprocess params.ini

> postprocess chain.txt
```

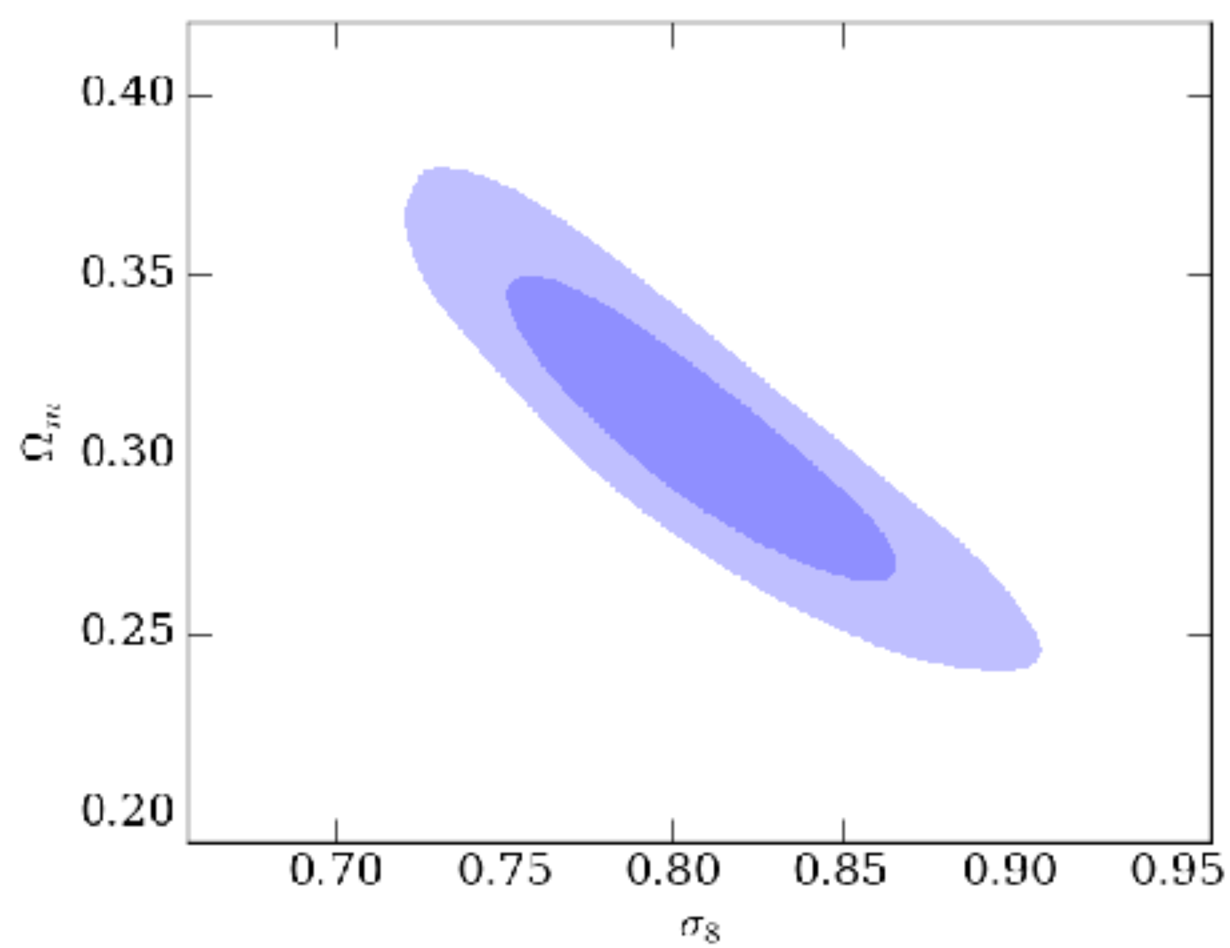


0.78
0.76
0.74
0.72
0.70
h
−0.1
0.0
0.1
0.2
ΔM
0.375
0.350
0.325
0.300
0.275
0.250
0.225
Ω_m

Plots



1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.000
0.002
0.004
0.006
0.008
0.010
$\Omega_\nu h^2$



0.40
0.35
0.30
0.25
0.20
Ω_m
0.70
0.75
0.80
0.85
0.90
0.95
σ_8

Modifying CosmoSIS

Setup Function

- Argument
  - `options` (datablock, contents of parameter file)
- Returns
  - C-pointer or python object

Execute Function

- Argument
  - `block` (datablock, output from sampler and prev. pipeline)
  - `config` (pointer/object, output of setup function)
- Returns
  - Status Integer (success = 0)

API
Used by setup, execute

- In C, C++, Python, and Fortran

- Get (read) and Put (write) functions/methods:

```
c_datablock_get_double(c_datablock* s,
    const char* section, const char* name,
    double* val);
```

C

```
DataBlock::get_val(std::string section, std::string name, T& val);
```

C++

```
datablock_put_logical(block, section, name, value) result(status)
        integer(cosmosis_status) :: status
        integer(cosmosis_block) :: block
        character(len=*) :: section
        character(len=*) :: name
```

Fortran

```
def get_double_array_1d(self, section, name):
```

Python

- Lookup functions, other introspection

Overview

- Break up your calculation into conceptual modules

- Make modules by wrapping calculation
  setup & execute functions calling API to get
  inputs/outputs from from/to pipeline

- Write parameter files to link and configure modules and choose a sampler

- Run the sampler and post-process the output chain

- Learn more by looking at the demos and existing modules

Additional Cool Features

- Sampler Chaining: ML -> Fisher -> MCMC
- Very flexible parameter files: include other files, used default args, use environment vars
- Scripting: import cosmosis directly in python
- Fault handling gives you a traceback on e.g. segfault
- Debug mode gives you python debugger on error
- Automatic blinding
- File locking detects multiple runs trying to write to same output file

Getting Started

- Set up the shared cosmosis libraries:

  source /net/software/cosmo/cosmosis/setup-cosmosis

- Copy the demos into your own directory:

  cp -r $COSMOSIS_SRC_DIR/demos ./demos

- Try running a few of the demos - explanations here: https://bitbucket.org/joezuntz/cosmosis/

CosmoSIS Exercises

Exercise 1

- Run and understand demo 2
- https://bitbucket.org/joezuntz/cosmosis/wiki/Demo2
- Look at the plots of the CMB spectra

Exercise 2

- Run and understand demo 5, a supernova example
- https://bitbucket.org/joezuntz/cosmosis/wiki/Demo5
- Try reducing the number of emcee walkers and see how it affects things

Exercise 3

- Run demo 11 - a lensing example with CFHTLenS
- Modify it to:
  - switch off saving the full cosmo data at each step
  - grid over w_0 as well as modified gravity parameters

Exercise 4

- Modify demo 2 to:
    - remove the BICEP 2 likelihood
    - run a Fisher matrix with the Planck likelihood
    - make plots of your Fisher matrix

Exercise 5

- Run demo 17, the Dark Energy Survey Science Verification Fisher matrix (try doing it in parallel)
- Free the nuisance DES-SV nuisance parameters and run a new Fisher matrix
- Make a plot comparing the Fisher matrices in the two cases