CosmoSIS: A multilingual Python plugin architecture for cosmology

Joe Zuntz
University of Manchester

Sarah Bridle, Scott Dodelson, Elise Jennings, Jim Kowalkowski, Alessandro Manzotti, Marc Paterno, Doug Rudd, Saba Sehrish

URL

https://bitbucket.org/joezuntz/cosmosis

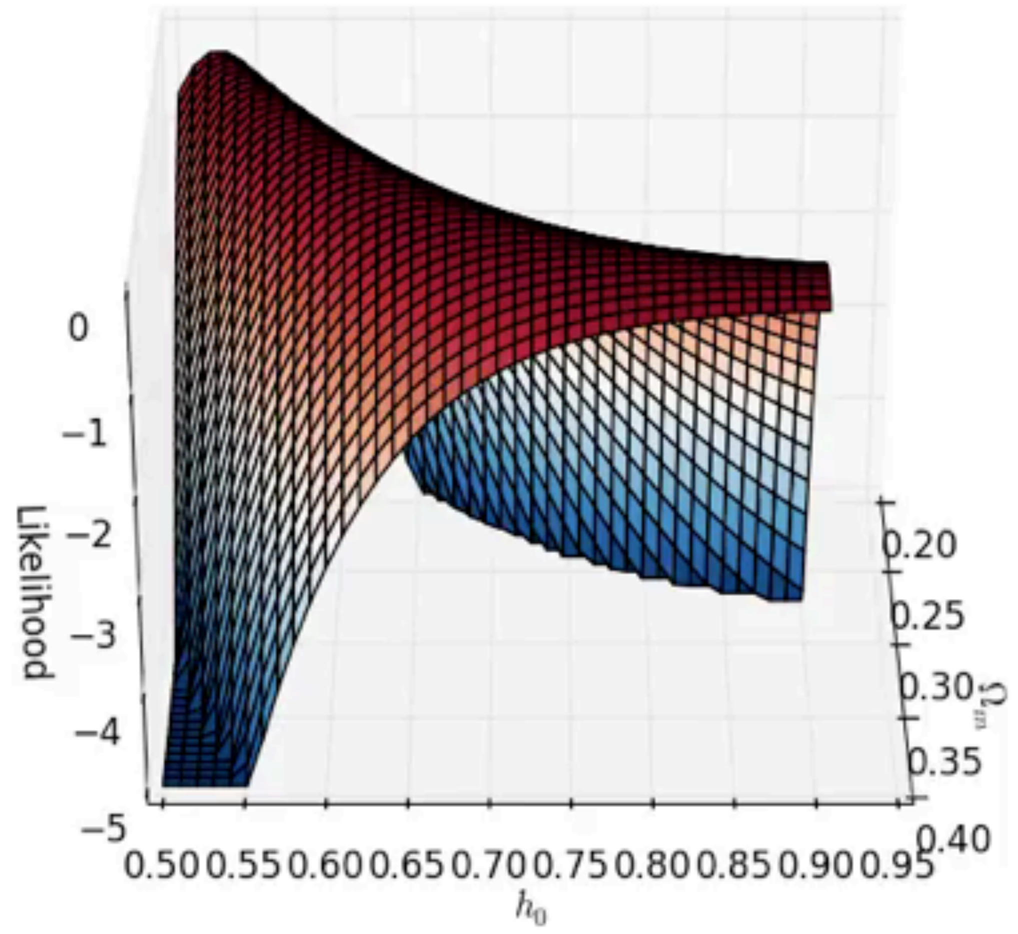
or google *cosmosis* and skip everything about the psychedelic trance band

Parameter Estimation

- Standard Bayesian approach to constraining models with data

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

$$\log P = \log \mathcal{L} + \log \pi + \text{const.}$$

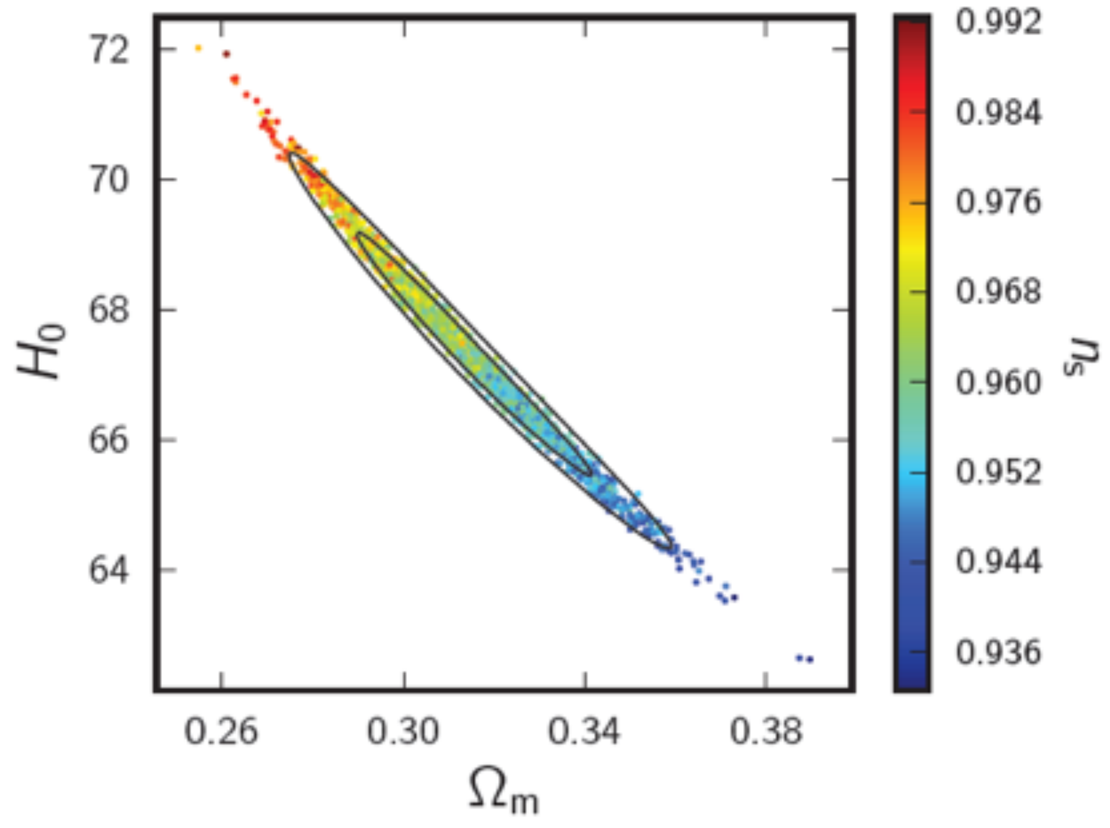


- Need to **evaluate likelihood** of particular model parameters given data

- And then **vary parameters** to explore space

Varying Parameters

- CosmoMC cosmology standard for 10 years
- Many other MCMC methods
    - emcee, multinest, PMC,
- Grid methods
- Maximum likelihood finders



CosmoSIS Samplers

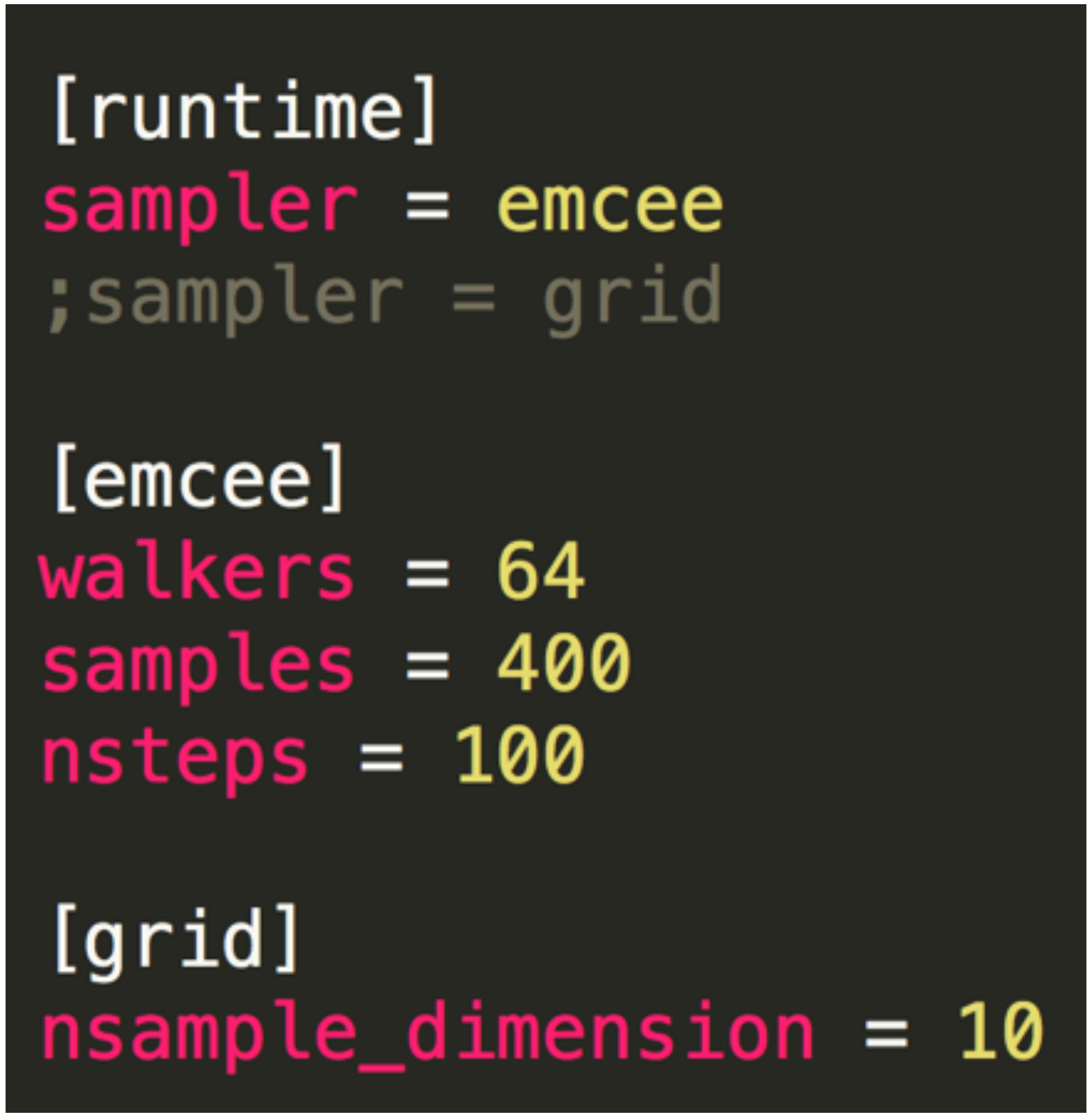
- Collect samplers, mostly python

- Provide a uniform interface to sampler

- Standard parallel sampler specification

- Shared output interface for samples

- Postprocessing

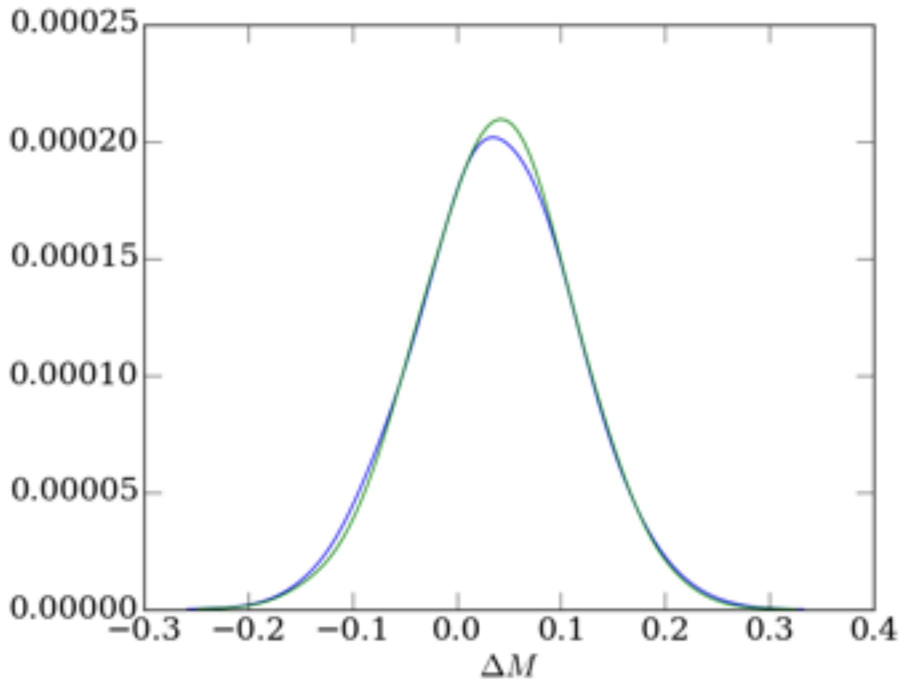


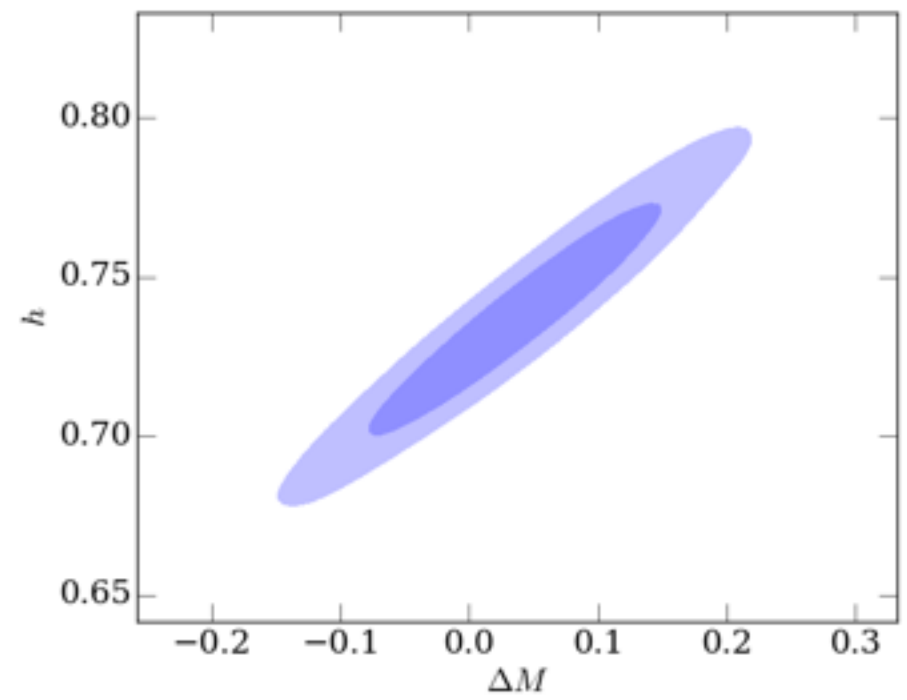
```
[runtime]
sampler = emcee
;sampler = grid

[emcee]
walkers = 64
samples = 400
nsteps = 100

[grid]
nsample_dimension = 10
```

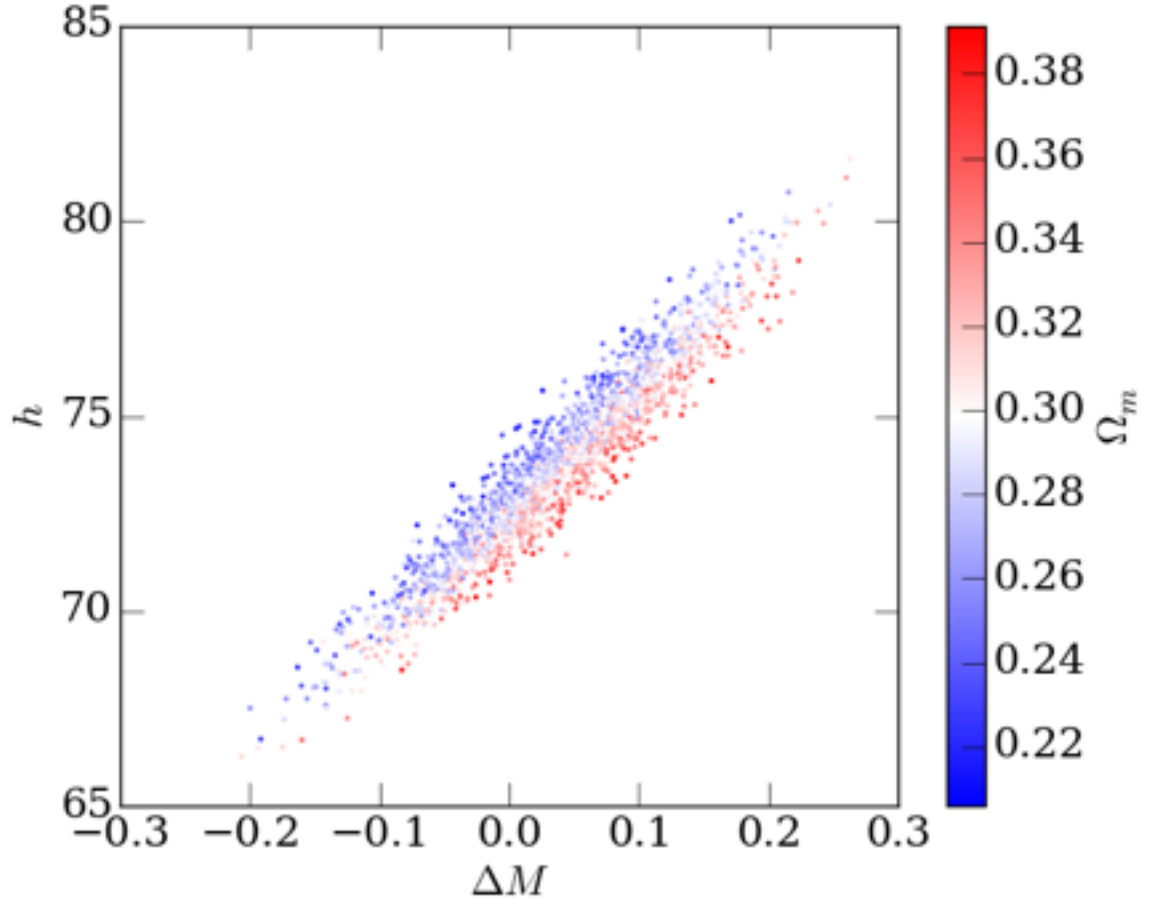
CosmoSIS Samplers

- Metropolis
- Emcee
- Maxlike
- Grid
- Multinest
- Population Monte Carlo
- Snake
- Test
- PyMC
- List *
- Minuit *
- Kombine *

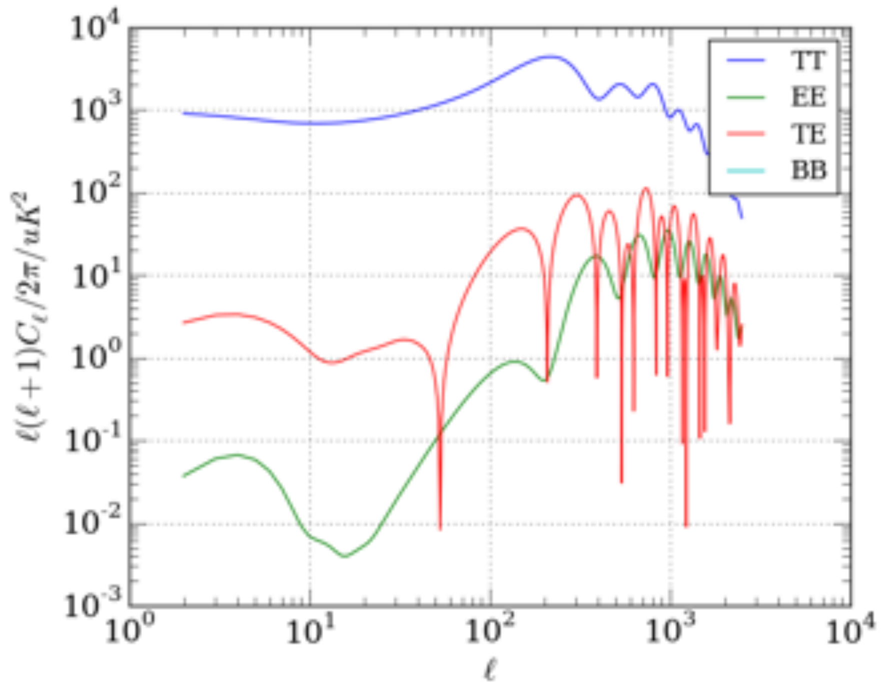
CosmoSIS Samplers

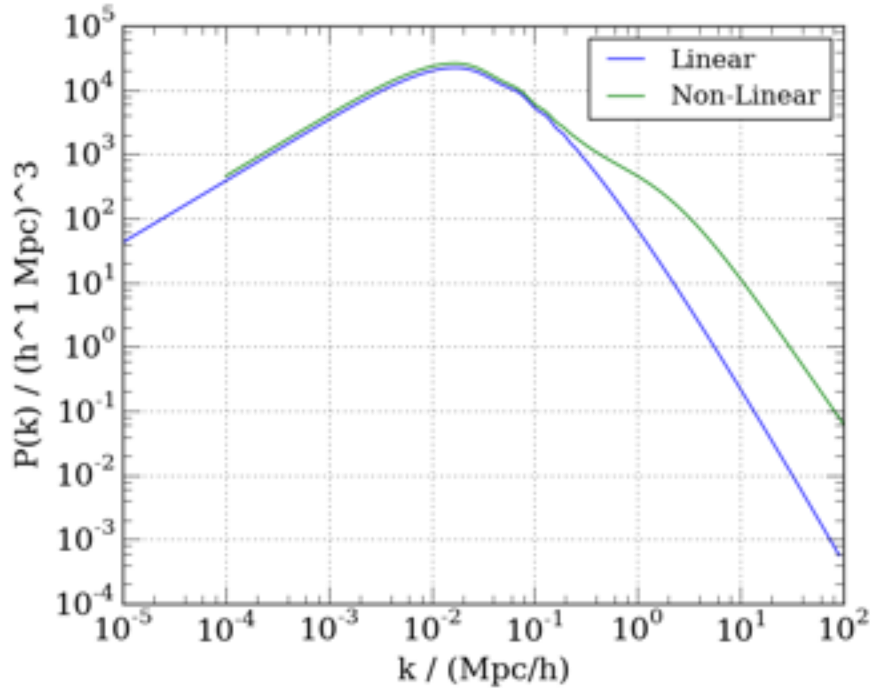






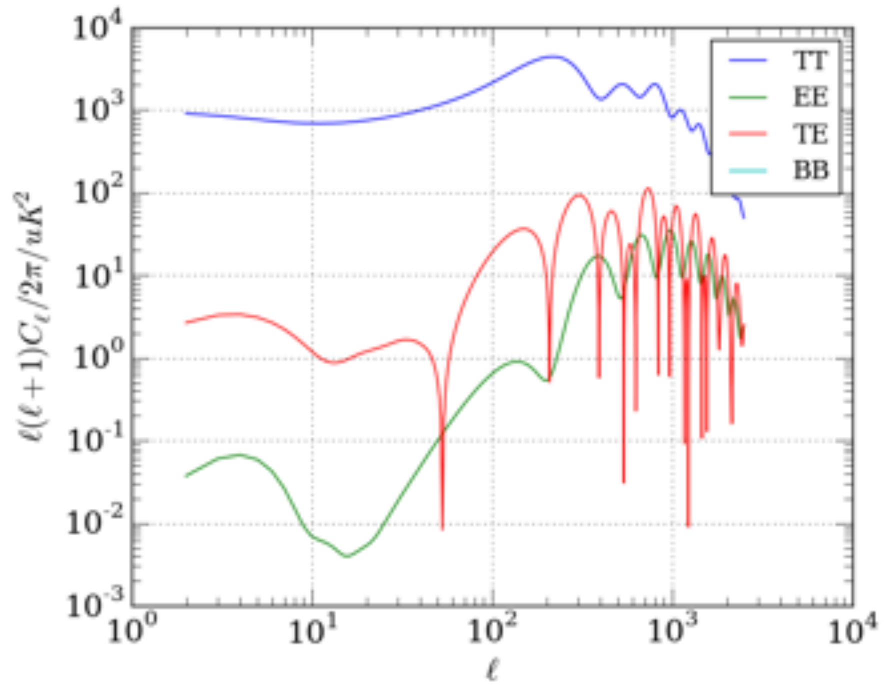
CosmoSIS Likelihood Pipelines

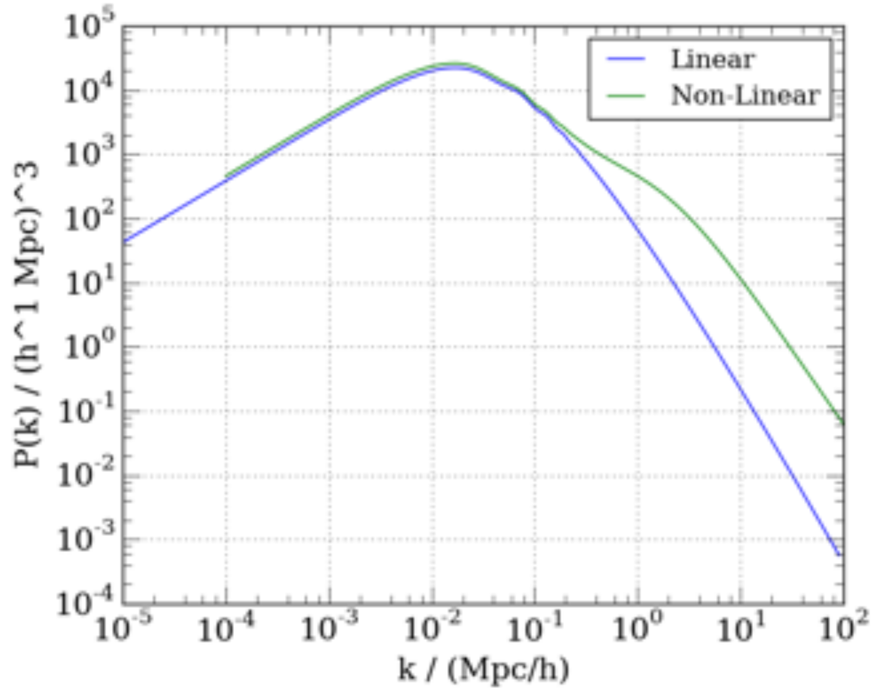




- Multiple theories and nuisance parameters models
- Painful shared systematic and statistical errors
- Strong legacy and community constraints

CosmoSIS Likelihood Pipelines

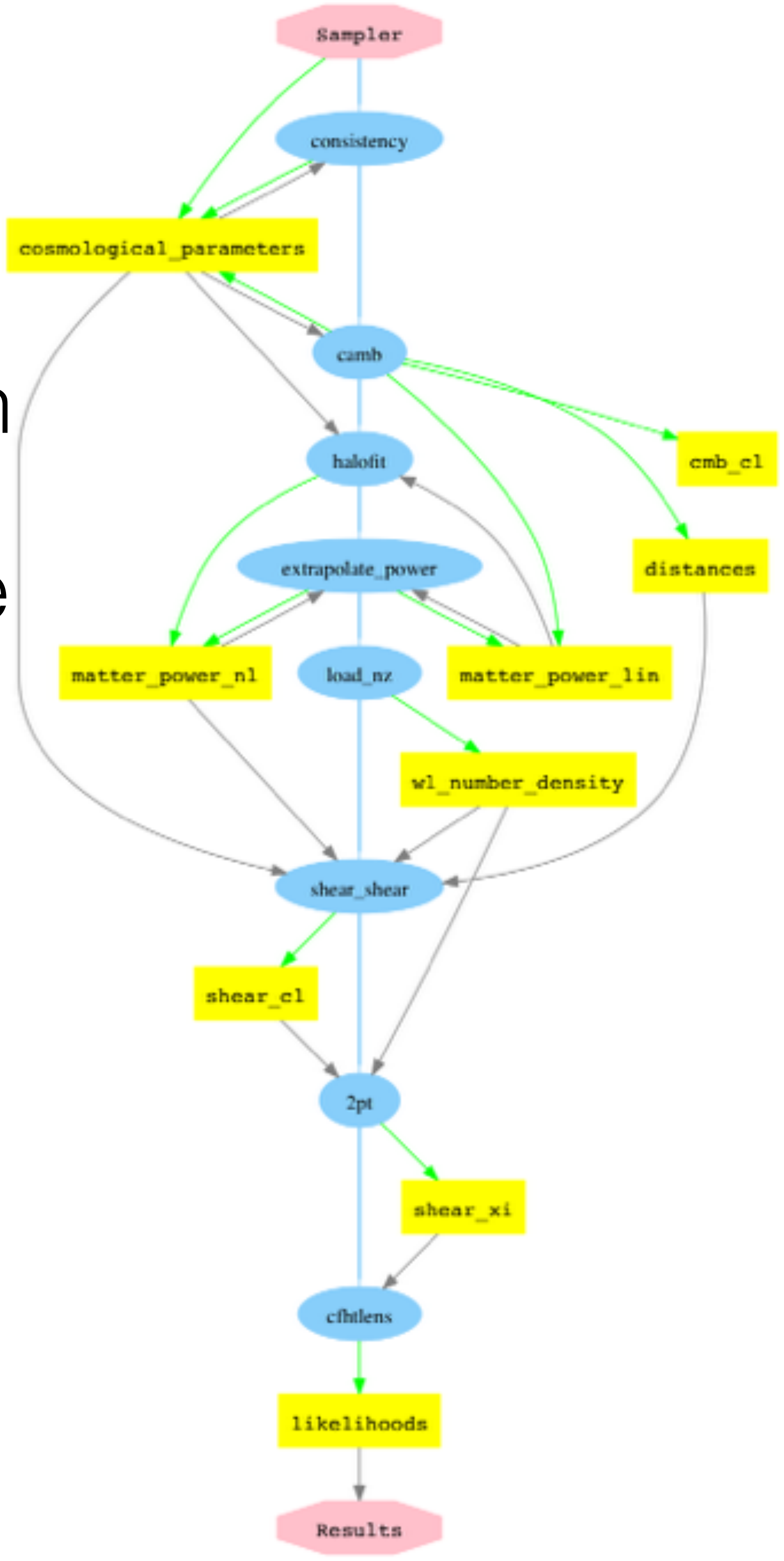




- Multiple theories and nuisance parameters models
- Painful shared systematic and statistical errors
- Strong legacy and community constraints

Modular Pipelines

- Chunk of theory / likelihood calculation
  - Becomes single cosmosis module shared lib or python module
  - Isolated from other modules All inputs/outputs via API
  - For legacy codes, simple-ish interface file connects to cosmosis



Advantages

- Compare & contrast models for data
- Verify & debug parts of code individually
- Build pipeline at runtime
- Mix languages
- Share code more easily
- Automate credit/citations

Implementation

- Tasks
  - Python needs to load and run modules from C/C++/Fortran
  - Python needs to call C API
- Choices
  - Cython, SWIG, Boost, CPython
  - None of the above: **ctypes & np.ctypeslib**

Running C/C++/Fortran modules from python

Standard form for cosmosis-module interface:

```
void * setup(c_datablock * options)
int execute(c_datablock * block, void * config)

function execute(block, config) result(status)
    use cosmosis_modules
    integer(cosmosis_status) :: status
    integer(cosmosis_block), value :: block
    integer(c_size_t), value :: config
```

and then native functions in F90/C to read from block

Running C/C++/Fortran modules from python

Compile each module into a shared library

Just add -shared and -fPIC to gcc/gfortran

CosmoSIS opens library with ctypes:

```
     lib = ctypes.cdll.LoadLibrary(filename)
```

and finds functions inside:

```
     exe = lib.execute
     exe.restype = ctypes.int
     exe.argtypes = [ctypes.c_size_t,
                     ctypes.c_voidp]
```

Installation

- Many modules in std lib =>
  Many dependencies
- CosmoSIS installer
  - OSX & SLF/Redhat
  - UPS package manager
  - down to compiler

No affiliation with
United Parcel Service

Outstanding Questions

- Do you have a cosmology likelihood? We would love to package and distribute it!
- Also physics calculations that can feed into likelihood super welcome!
- Open some issues!
- Can anyone make us a cool logo?

CosmoSIS
Enjoyable Parameter Estimation

Cosmosis