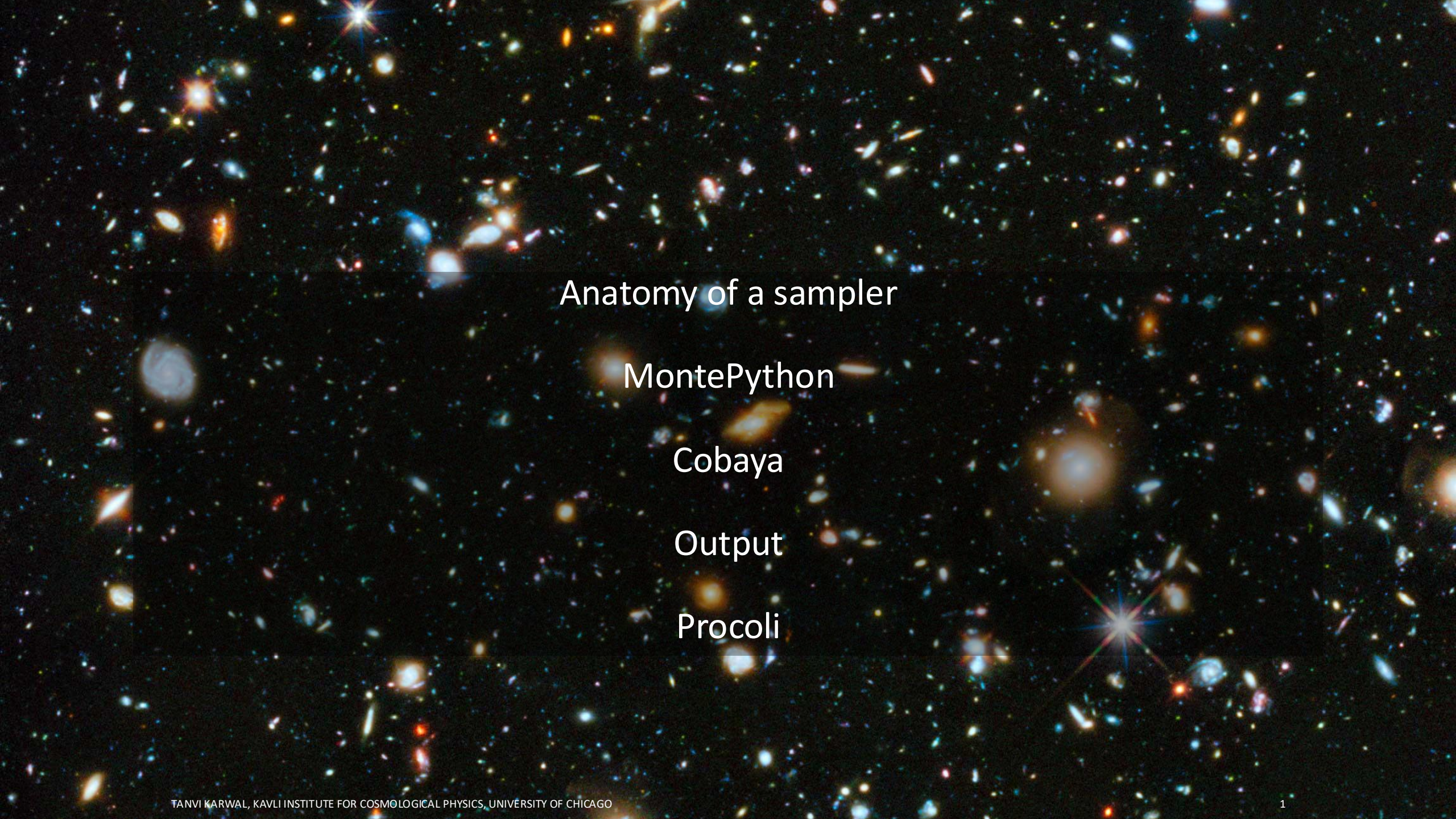
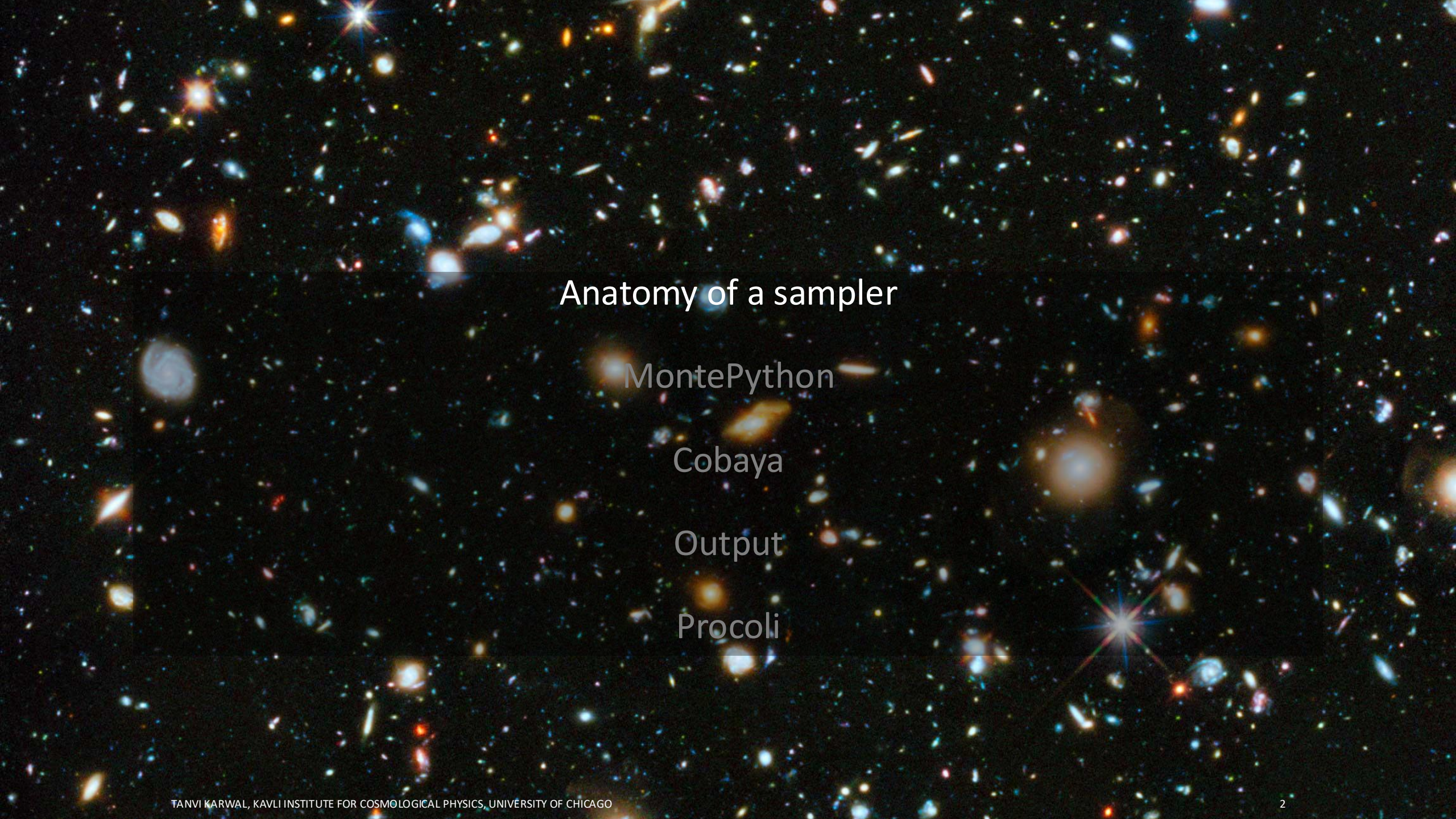
Cosmological parameter-estimation existing infrastructure

Tanvi Karwal

THE UNIVERSITY OF CHICAGO KICP

GGI workshop: Exploring New Frontiers in Cosmology
Parameter estimation in cosmology I
9th July 2026


Anatomy of a sampler
MontePython
Cobaya
Output
Procoli



Anatomy of a sampler

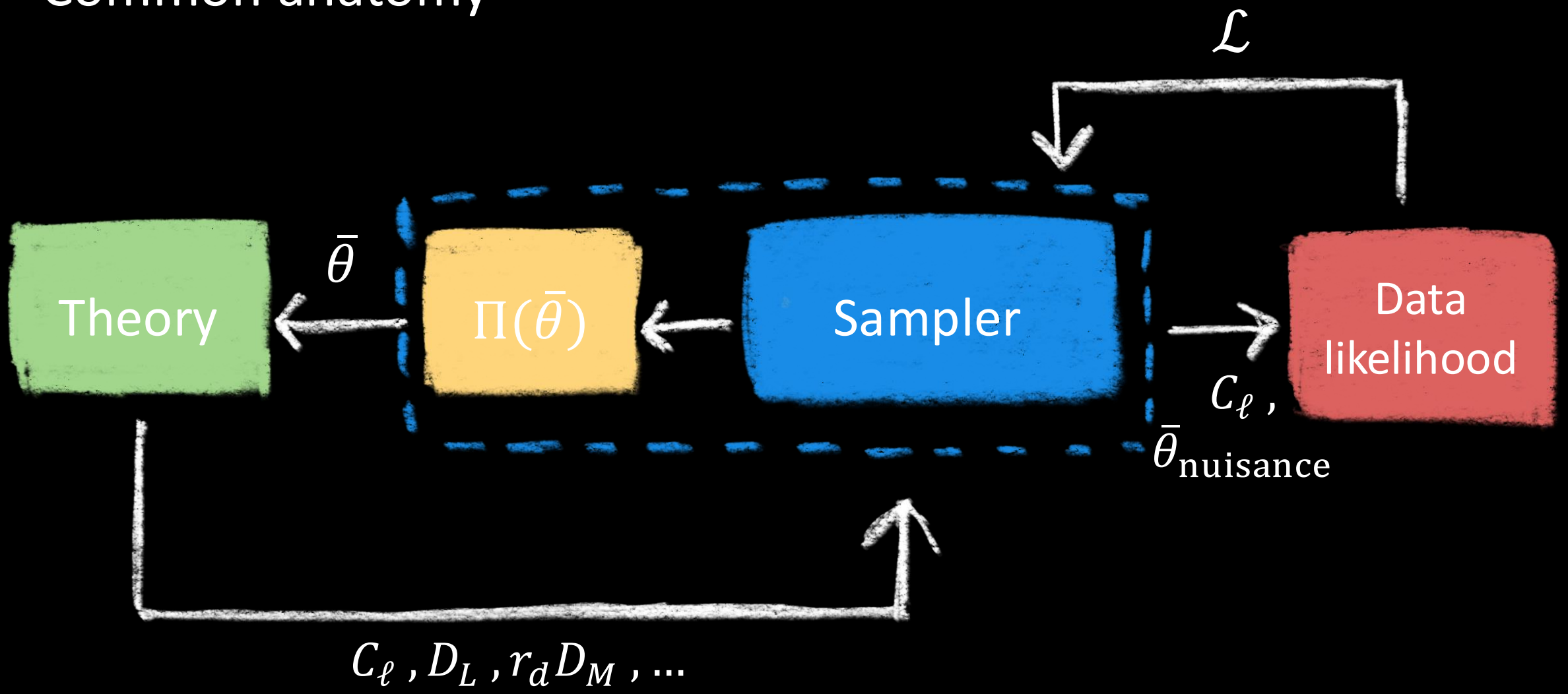
MontePython

Cobaya

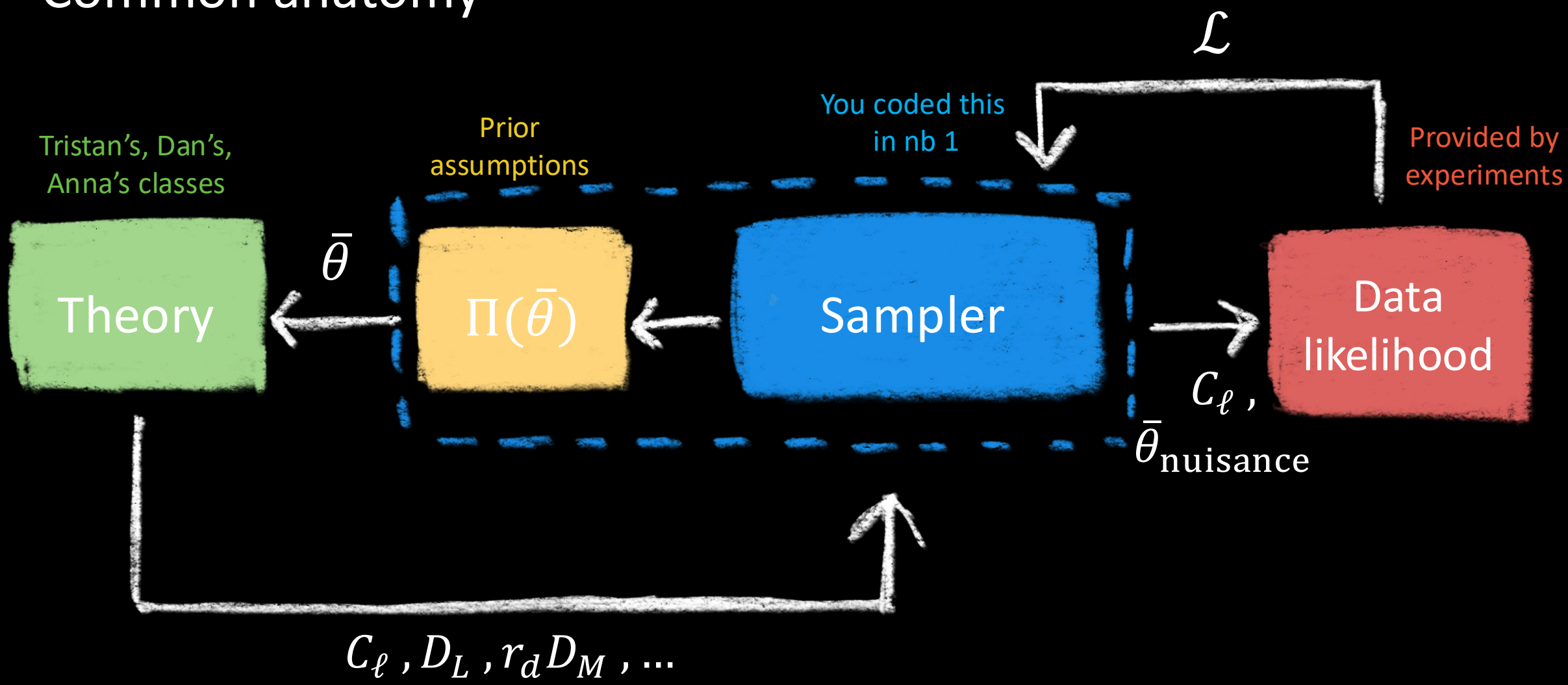
Output

Procoli

Common anatomy



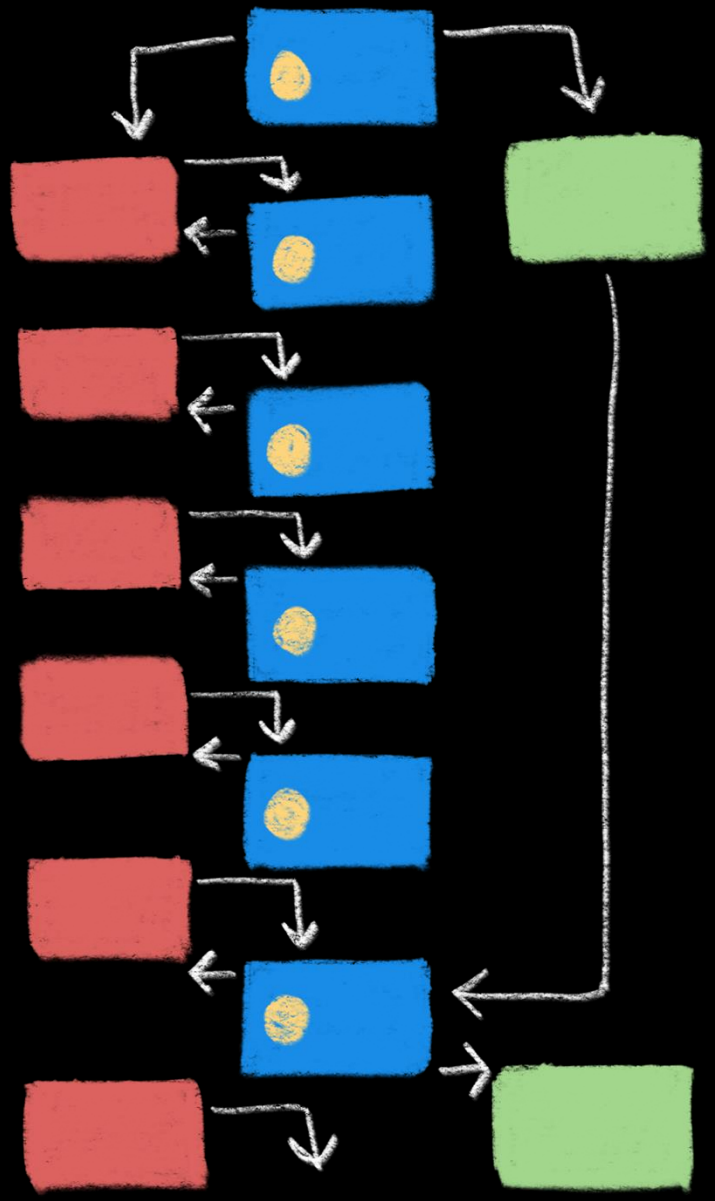
Common anatomy



Slow vs fast parameters

- Cosmological param changes → full CLASS/CAMB recompute - slow
- Nuisance param changes → likelihood re-evaluated only (fast)
- This one split drives fast-slow decorrelation, oversampling, dragging
- The single most important performance concept in both codes

Sampler
Prior
Data likelihood
Theory code



Efficient sampling of fast and slow cosmological parameters

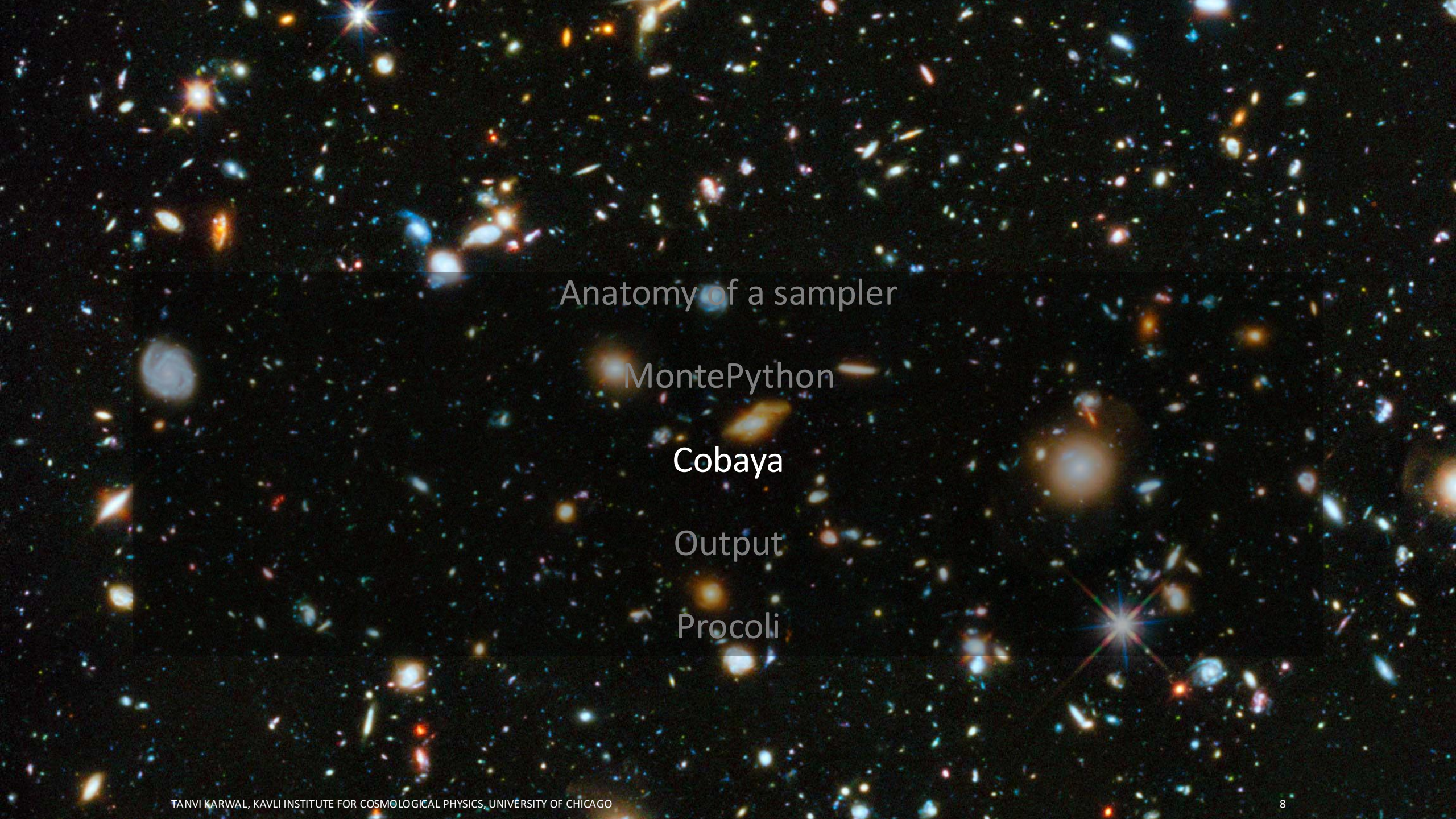
Antony Lewis[1,*]

[1]*Department of Physics & Astronomy, University of Sussex, Brighton BN1 9QH, UK*


Anatomy of a sampler
MontePython
Cobaya
Output
Procoli

MontePython

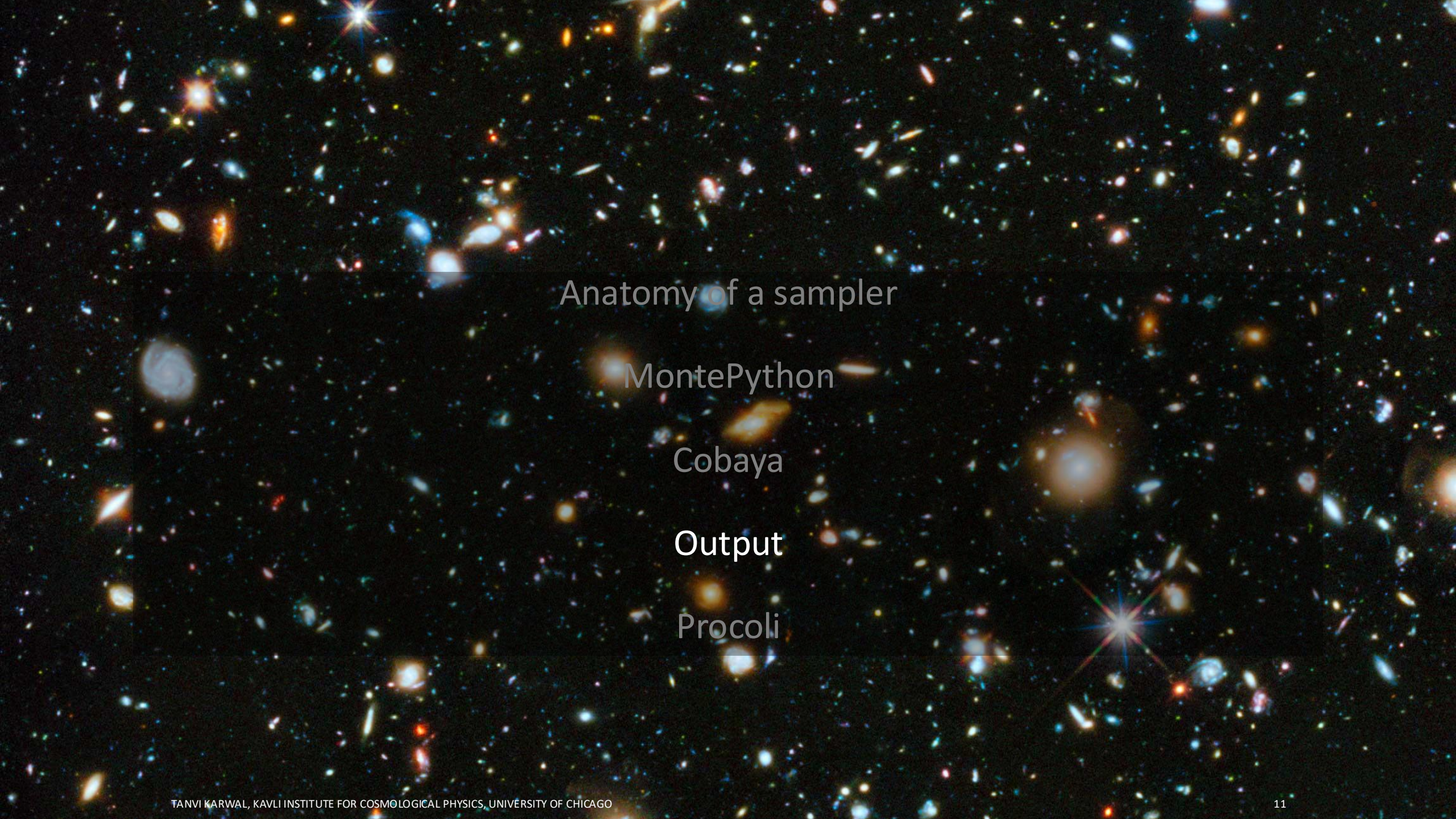
- MP set up, callable from anywhere, `.conf` file
- File structure - inputs, bestfits, covmats
- Likelihoods that are recognised, writing likelihoods, PLC
- One `.param` file = the whole run
- `parameters['x_class_readable'] = [mean, min, max, sigma, scale, role]`
  - `MP/data.py`
- `role = cosmo / nuisance / derived / cosmo_arguments` → the slow/fast split, explicit
  - Oversampling
  - Must pass nuisance parameters
- Run: `mpirun -np 6 MontePython.py run -o ./ -N 100000000 --update 50 --superupdate 20 -c ../w0wa_bao_sne.covmat`
- Run end point
- Output format – job output, files
- Analyse: `MontePython.py info chains/` → covmat, bestfit, triangle PDFs
- Once a chain folder exists, MP reads `log.param`, not your edited `.param`
- Rerunning / continuing runs


Anatomy of a sampler
MontePython
Cobaya
Output
Procoli

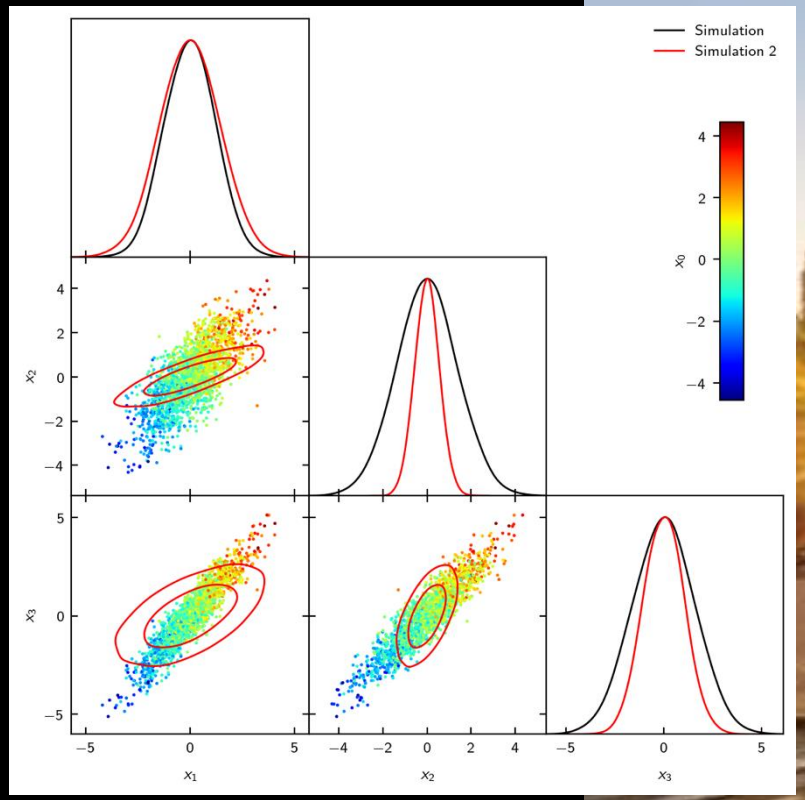
cobaya 

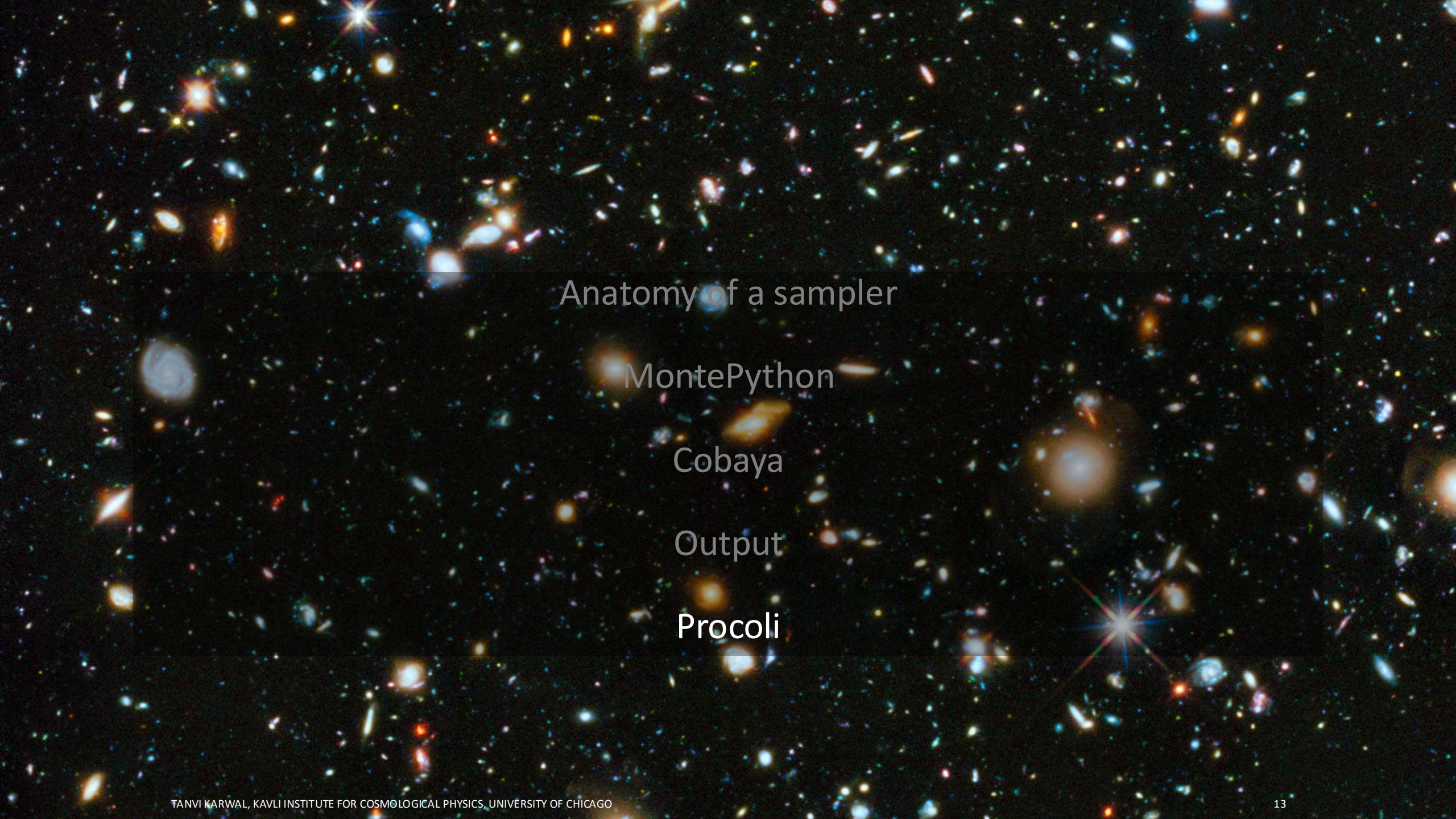
- Cobaya installation, documentation
- One `.yaml`, four explicit blocks: `theory` · `likelihood` · `params` · `sampler`
  - Dropping parameters, defining composite parameters
- Swap `classy` ↔ `camb` on one line
- `cobaya-install`
  - Auto-installs `CLASS/CAMB/Planck`
  - More likelihoods released
- Run: `cobaya-run base.yaml` → writes a resolved `.updated.yaml`
- Run end point
- Output format – job output, debug output, files
- Swap samplers: `mcmc`, minimize (MAP), `polychord` (evidence), `evaluate` (tests / χ^2)
- Rerunning / continuing runs

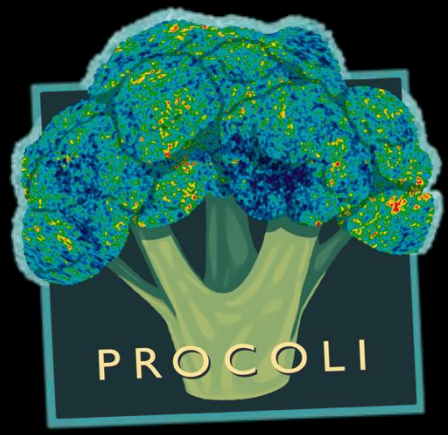
	MontePython	cobaya
Config format	.param, .conf	.yaml
Boltzmann code	CLASS	CLASS / CAMB
Likelihood installation	Manual	cobaya-install
Samplers	MH MCMC, nested sampling, PolyChord, importance sampling, Fisher, minimisation, “evaluate”	MH MCMC, PolyChord, importance sampling, minimisation, evaluate
Run record	log.param	.updated.yaml
Best for	Hackability, CLASS world	Auto-installs, intuitive use, complicated priors


Anatomy of a sampler
MontePython
Cobaya
Output
Procoli

All roads lead to ~~Rome~~ GetDist




Anatomy of a sampler
MontePython
Cobaya
Output
Procoli



- Currently wraps MP
  - Same likelihoods, same parameter files
  - Frequentist instead of Bayesian
- Input MP run `log.param, .covmat ,`
- Run via python script, submit a job
- Simulated-annealing optimiser + sequential profiling (what you built in notebook 2)
  - Robust global optimiser run. Faster more efficient profile runs
- Output $\chi^2_{\min}(\theta_i)$ profile – record of $\bar{\theta}_i$ + χ^2 per experiment
- Prior-independent**: the check on the prior-volume effects from the last lecture
- **priors on nuisance parameters are included