Nils Schöneberg Tuesday 06.06.2023 **nils.science@gmail.com**

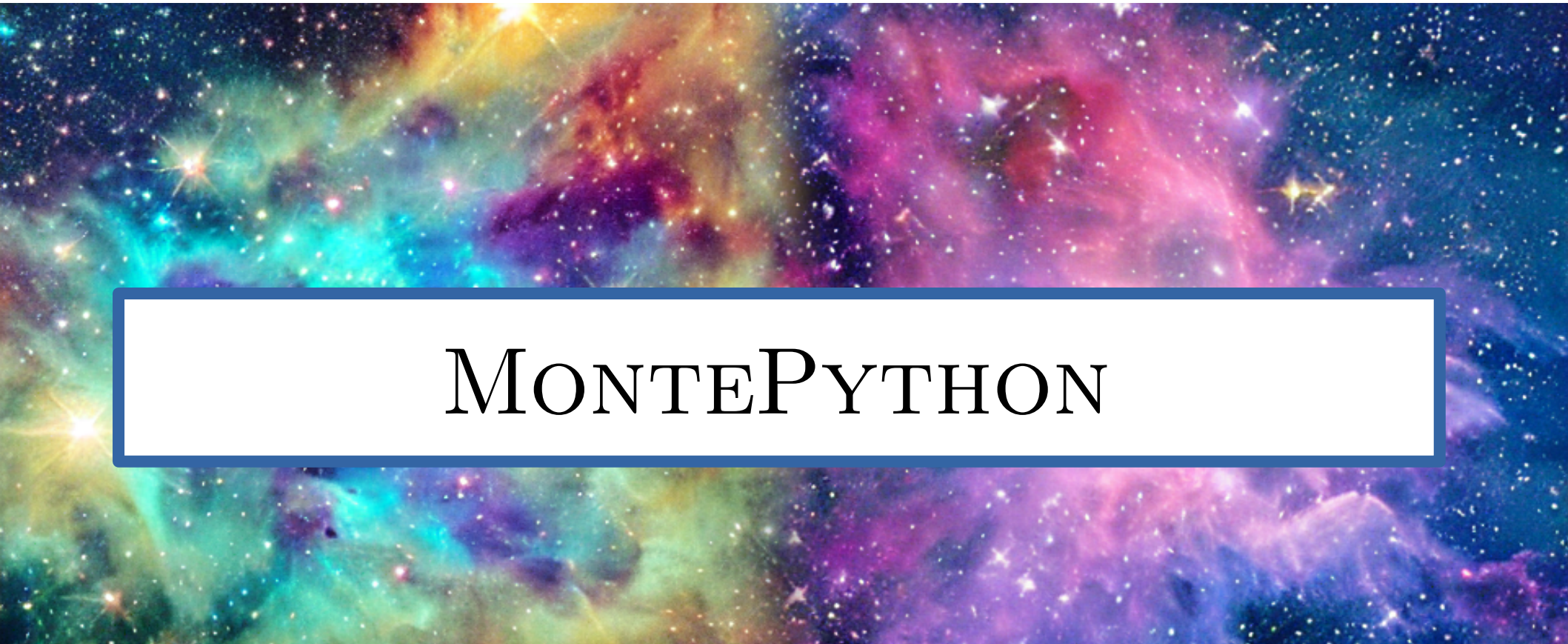
European Research Council

Established by the European Commission

Institut de Ciències del Cosmos

UNIVERSITAT DE BARCELONA

EXCELENCIA MARÍA DE MAEZTU 2020–2023



MontePython

Layout of the Tutorial

- Lecture in three parts:
  - Statistical inference & model comparison
  - MCMC chains and samplers
  - How to use MontePython
- What we will cover if there's additional time:
  - How to use Cobaya

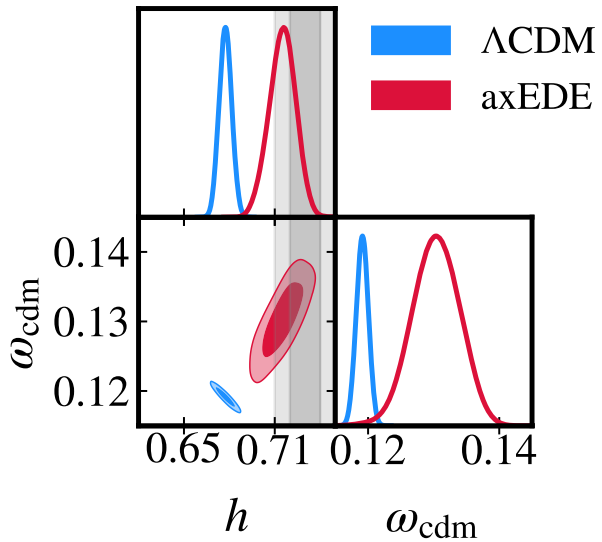
How to do theoretical science

- Science is very simple:

1) Build model of universe
2) Test with data
3) Repeat

Creativity, clever composition of existing ingredients/mechanisms

How 'good' is your model?



Poulin, Smith, Karwal 2302.09032

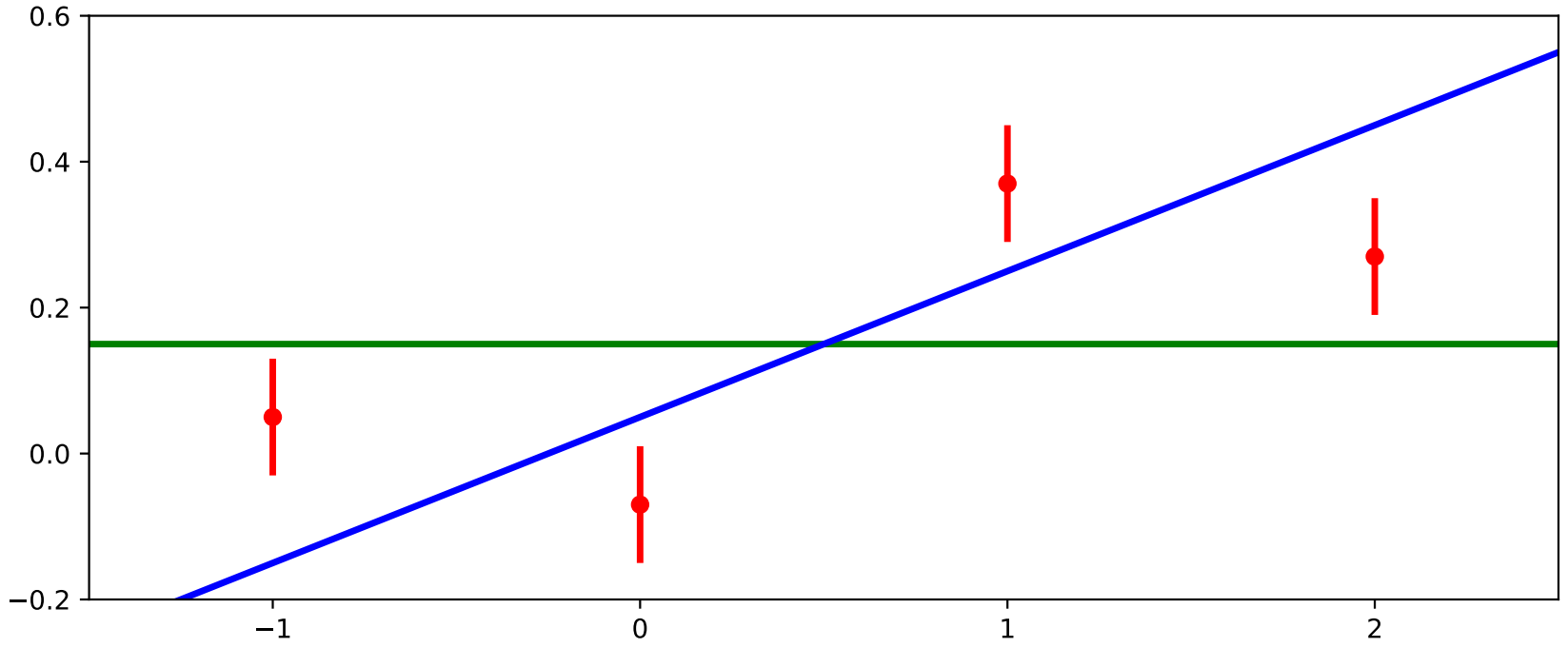
How to constrain a model

- We want to know which model (or parameters thereof) agree with the data!
- In theoretical science we usually do

  parameter inference and *model comparison*
- How can we do that?

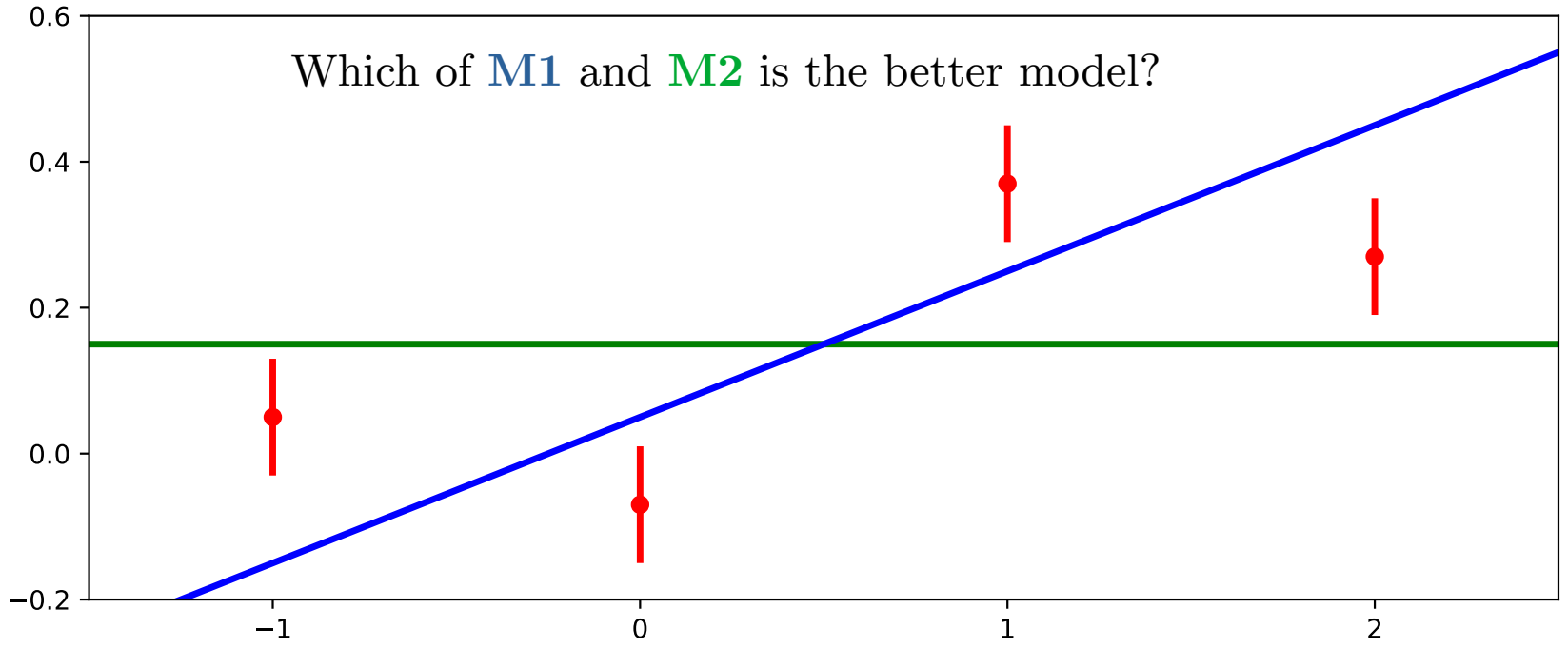
The likelihood

- Given data, how good is the model?



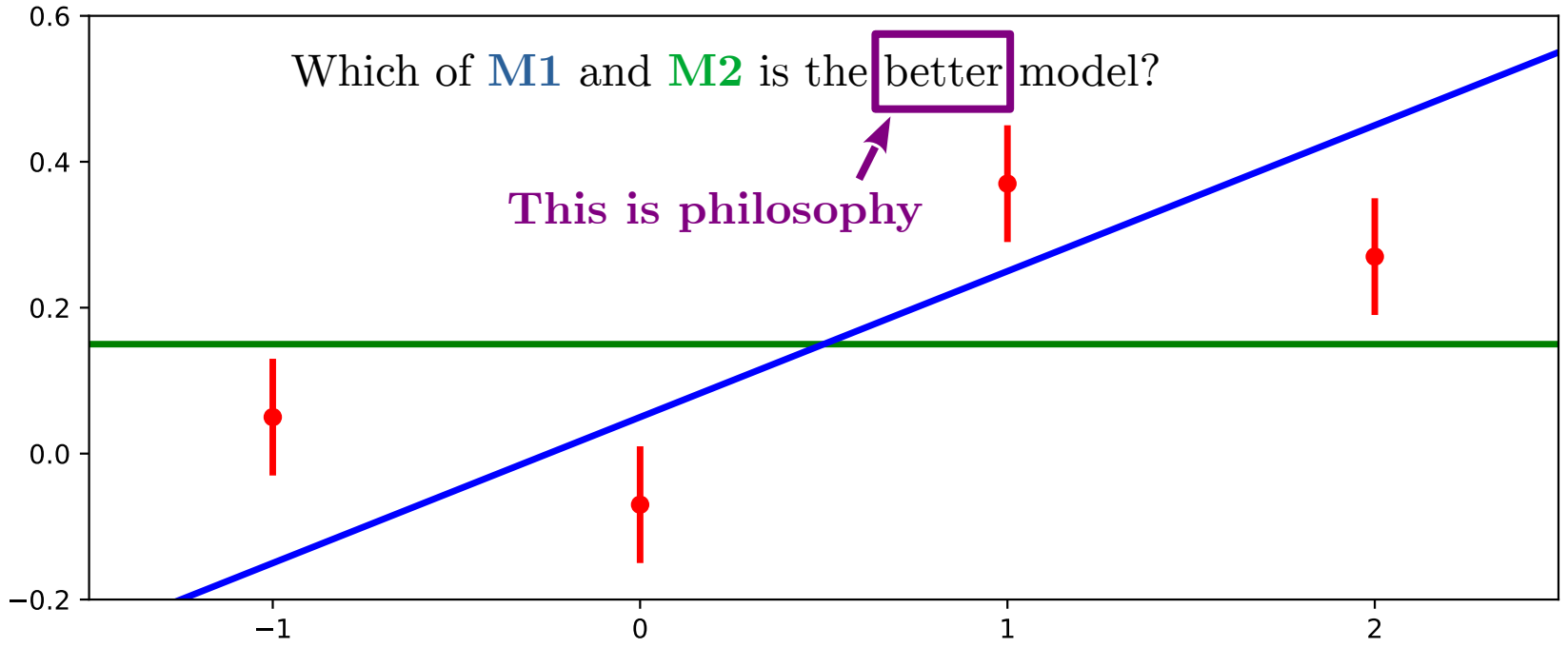
The likelihood

- Given data, how good is the model?



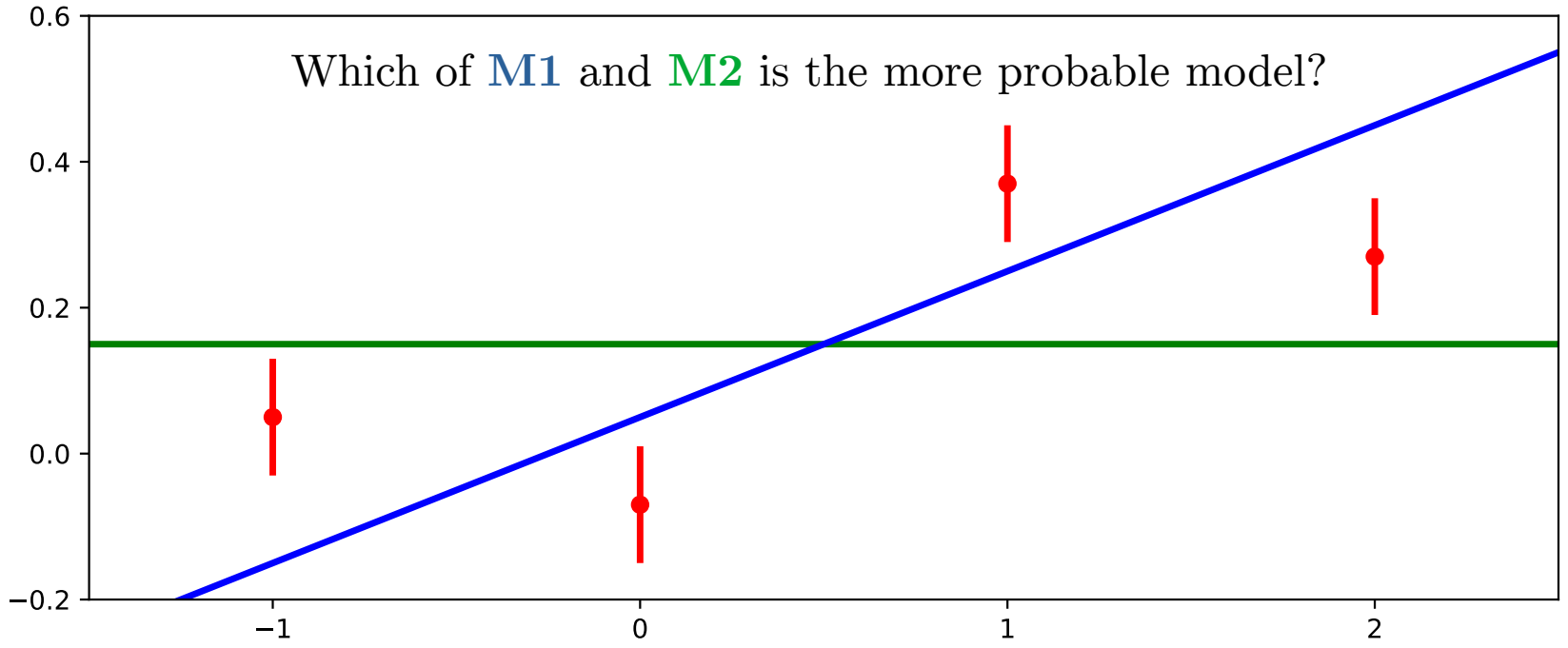
The likelihood

- Given data, how good is the model?



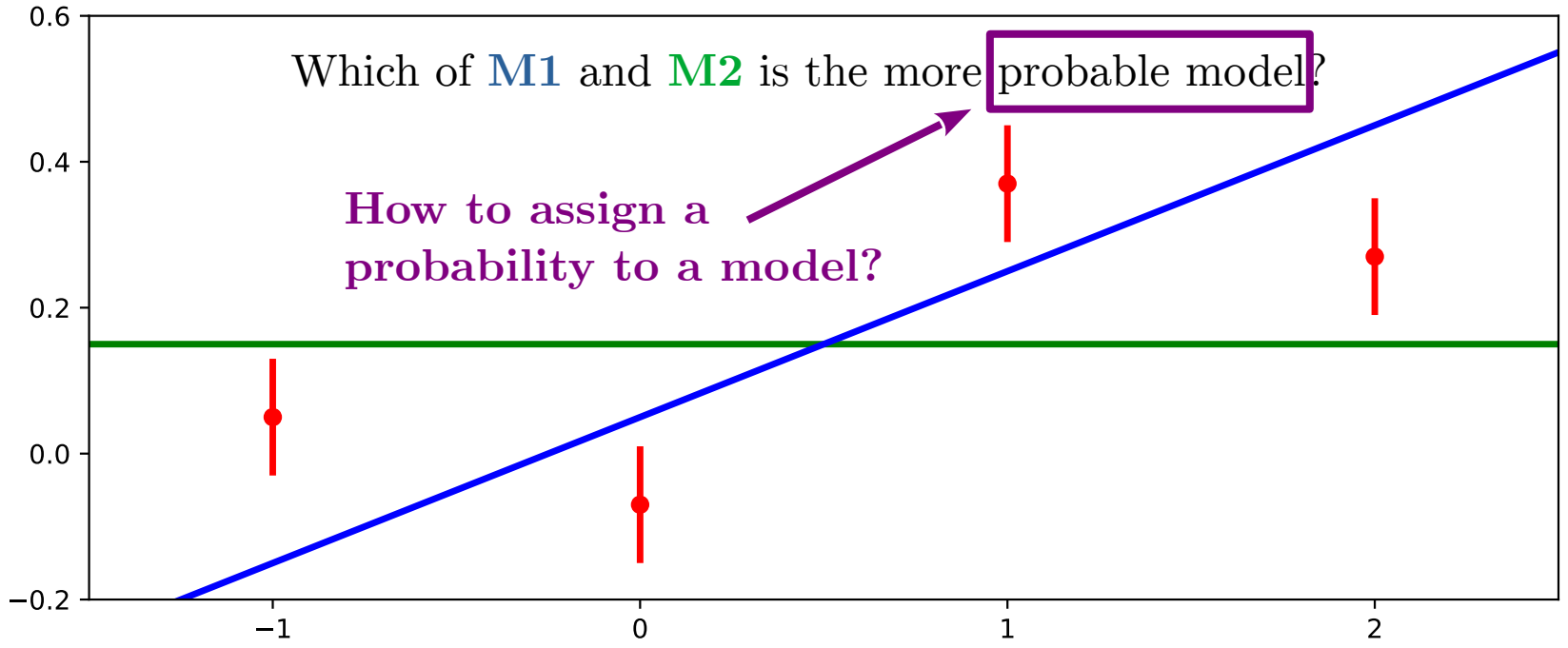
The likelihood

- Given data, how probable is the model?



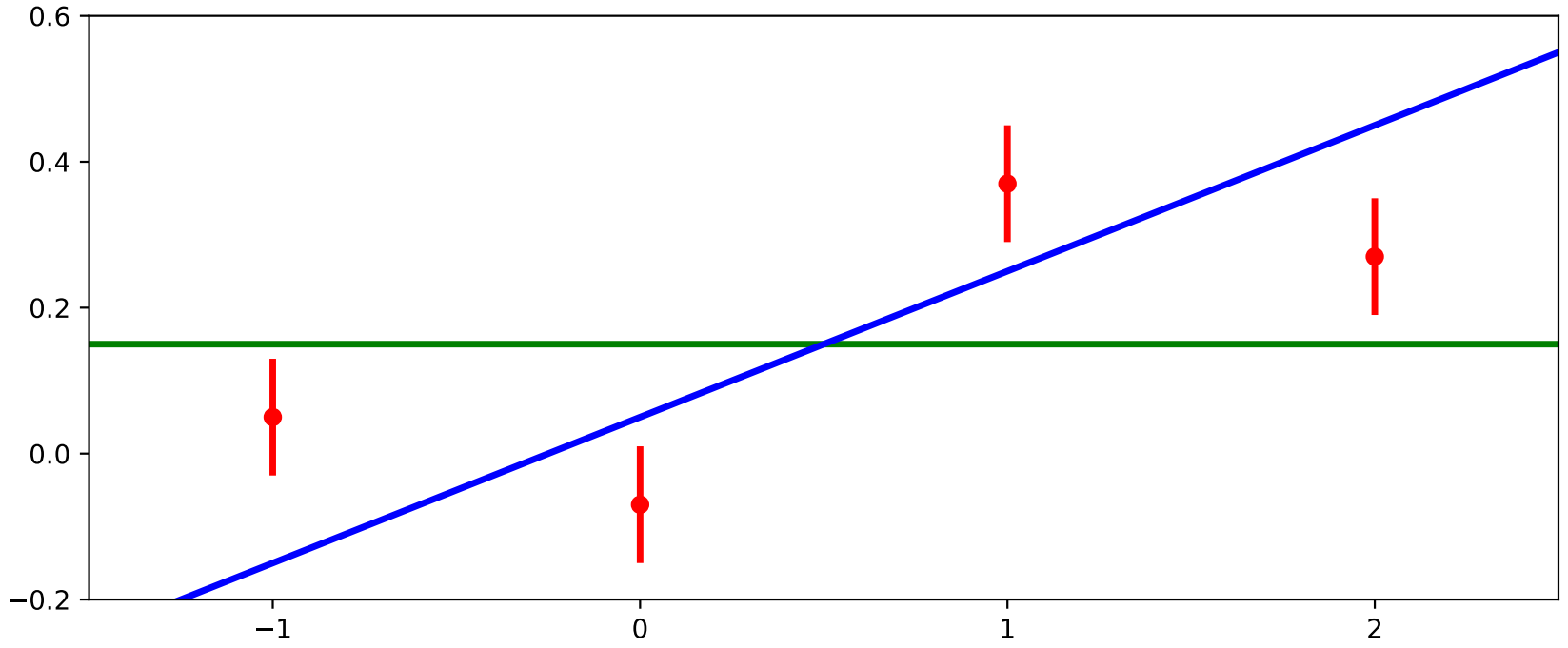
The likelihood

- Given data, how probable is the model?



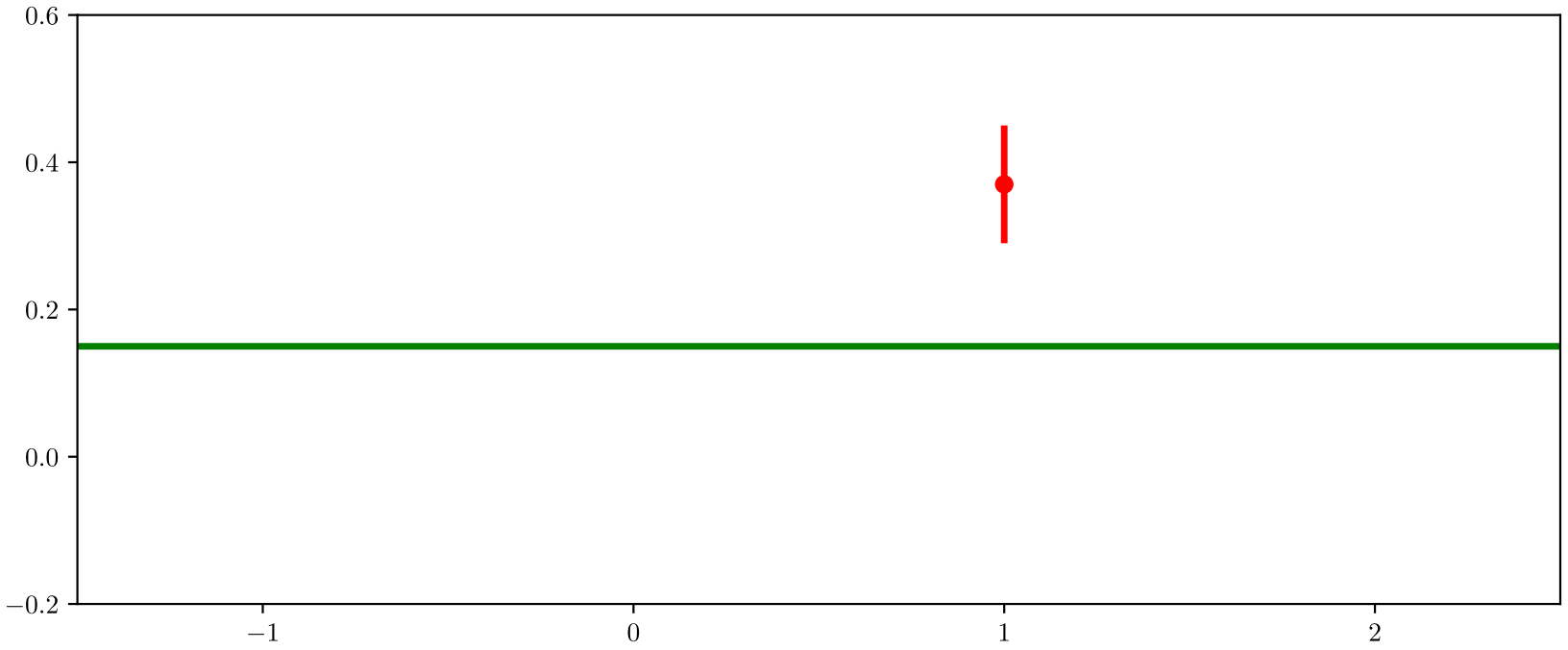
The likelihood

- Much simpler: Given model, how probable are the data to occur?



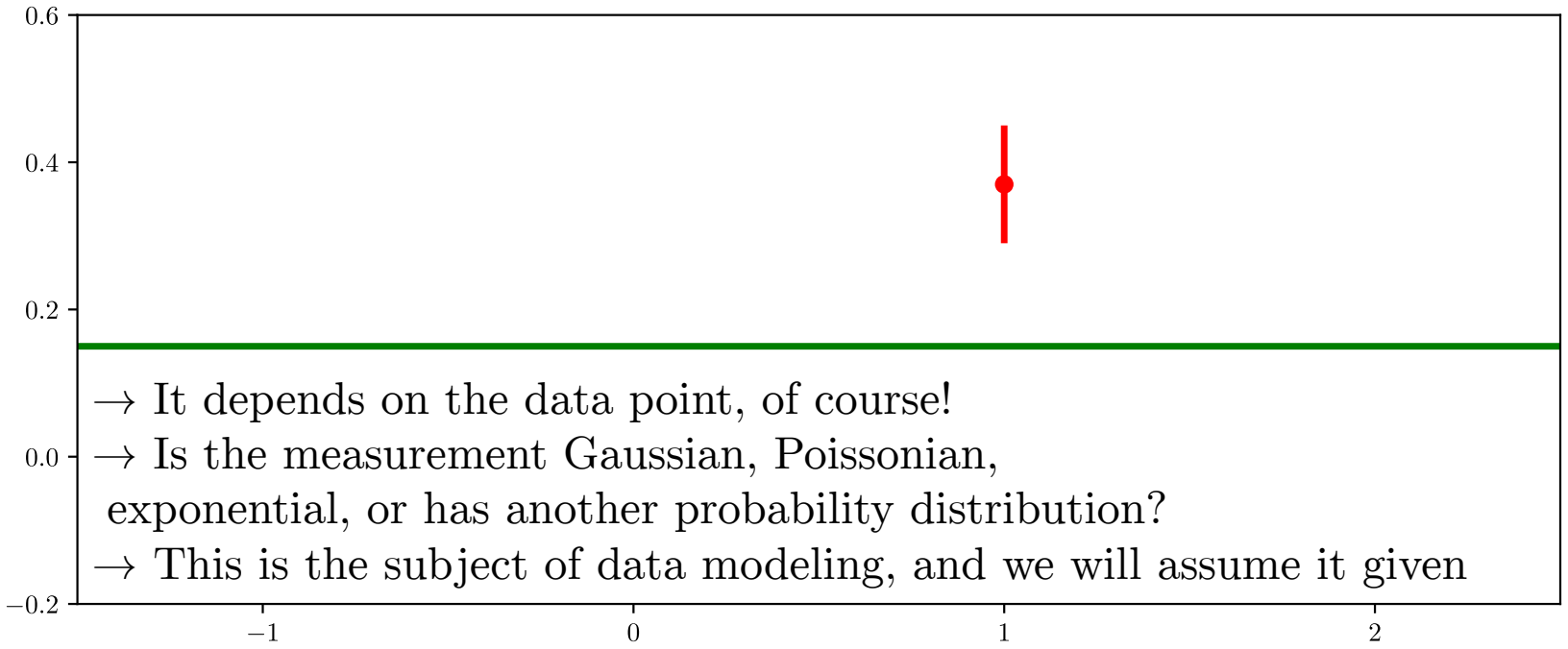
The likelihood

- Given **M2**, what is the probability of this data point to be measured?



The likelihood

- Given **M2**, what is the probability of this data point to be measured?



→ *Typically* , data are multivariate Gaussian distributed (→ central limit theorem)

The likelihood

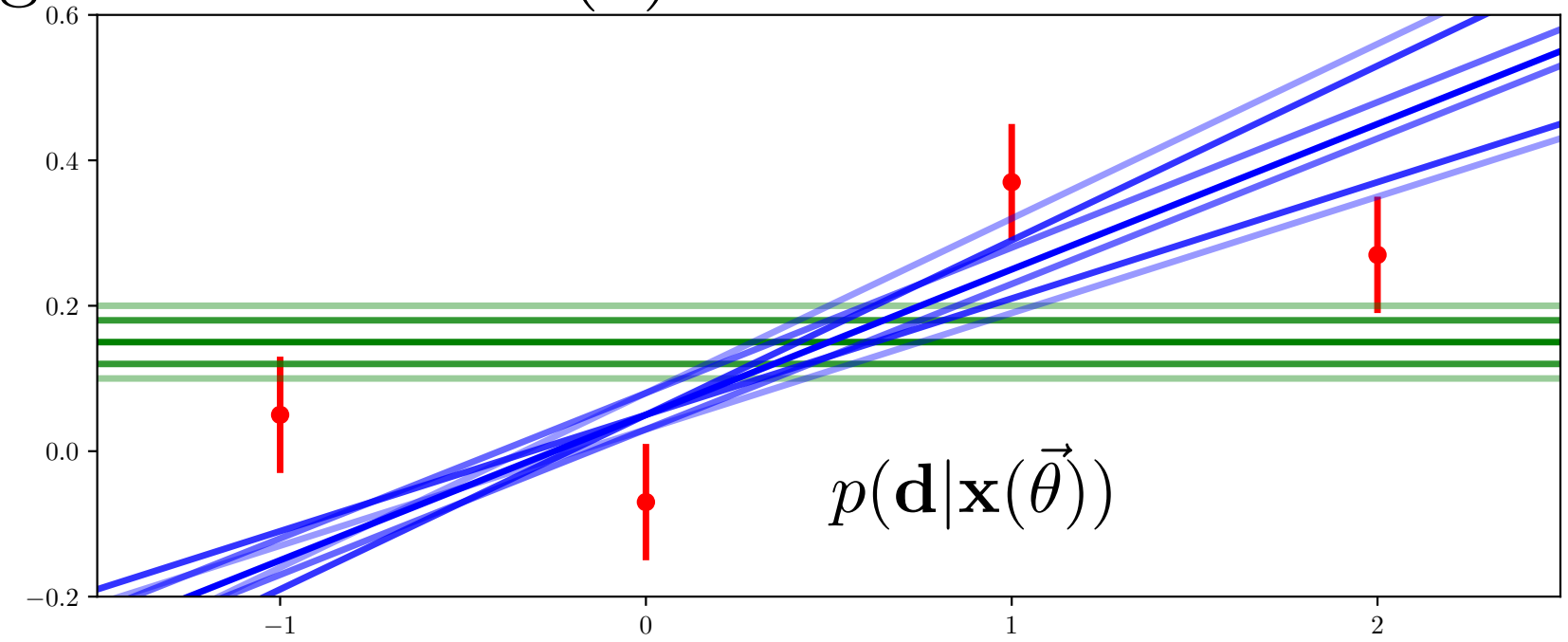
- Assuming (for the moment) multivariate Gaussian distribution:

$$p(\mathbf{d}|\mathbf{x}) = \frac{\exp(-\frac{1}{2}\left((\mathbf{x}-\mathbf{d})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{d})\right)}{\sqrt{(2\pi)^d \det[\mathbf{C}]}}$$

- Here $\mathbf{x}$ is the vector of *predicted* outcomes for the experiment, $\mathbf{d}$ the data, d its dimension, and $\mathbf{C}$ the covariance matrix

The likelihood

- The probability of the data being observed given a model $\mathbf{x}(\vec{\theta})$



The likelihood

- The *likelihood* is simply the probability of observing the data as such within a given predictive model

$$\mathcal{L}(\mathbf{d}|\vec{\theta}) = p(\mathbf{d}|\mathbf{x}(\vec{\theta}))$$

- Often we also talk about the log-likelihood or χ^2

$$\chi^2 = -2\ln\mathcal{L}(\mathbf{d}|\vec{\theta}) = -2\ln p(\mathbf{d}|\mathbf{x}(\vec{\theta}))$$

Example: Multivariate Gaussian

$$\chi^2 = (\mathbf{x}(\vec{\theta}) - \mathbf{d})^T \mathbf{C}^{-1} (\mathbf{x}(\vec{\theta}) - \mathbf{d}) + 2\ln\left(\sqrt{(2\pi)^d \det[\mathbf{C}]}\right)$$

Nils Schöneberg nils.science@gmail.com

A note on the Chi-squared

- The covariance can either be estimated from the data (*typical*), or predicted by the theory
- If predicted by the theory,

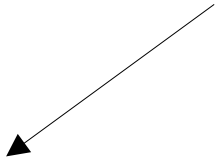
$$\chi^2 = (\mathbf{x}(\vec{\theta}) - \mathbf{d})^T \mathbf{C}(\vec{\theta})^{-1} (\mathbf{x}(\vec{\theta}) - \mathbf{d}) + 2 \ln \left(\sqrt{(2\pi)^d \det[\mathbf{C}(\vec{\theta})]} \right)$$

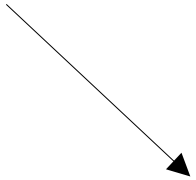
- Otherwise

$$\chi^2 = (\mathbf{x}(\vec{\theta}) - \mathbf{d})^T \mathbf{C}^{-1} (\mathbf{x}(\vec{\theta}) - \mathbf{d}) + \text{const}$$

From likelihood to inference

- We now know $p(\mathbf{d}|\vec{\theta})$
- But we originally wanted to know how '*good*' a given model is (replace '*good*' by '*probable*')





Frequentist: Nature is as nature is, the model is **already chosen**. There is no 'probability of a model'.

Bayesian: We don't know which model nature has chosen, so let us parameterize our ***belief probability*** that a given model is the true model chosen by nature.

The Frequentist Land

From likelihood to inference

- **Frequentist:**

  Reject model parameters at confidence limit α, if probability of generating data given is less than α.

- More precisely: Construct confidence intervals

$$p(u(\mathbf{d}) < \theta < v(\mathbf{d})) \geq \alpha$$

  with the functions u,v to be determined for given α

Nils Schöneberg nils.science@gmail.com

From likelihood to inference

This is some fixed true value

$$p(u(\mathbf{d}) < \theta < v(\mathbf{d})) \geq \alpha$$

These change for each new experiment (are random variables)

From likelihood to inference

- **Frequentist:**

$$p(u(\mathbf{d}) < \theta < v(\mathbf{d})) = \alpha$$

- Example model: N data are all generated with (constant) mean and variance

$$d_i \sim \mathcal{N}(\mu, \sigma^2)$$

- Then we have $p(\bar{d} - cS/\sqrt{N} \leq \mu \leq \bar{d} + cS/\sqrt{N}) = \alpha$

with $\bar{d} = \frac{1}{N}\sum d_i$ $\quad S^2 = \frac{1}{N-1}\sum (d_i - \bar{d})^2$

$$c = T^{-1}(1 - \alpha/2, N - 1)$$

Nils Schöneberg nils.science@gmail.com

Hypothesis test

- Decide between two scientific hypotheses/models:

$$H_{\text{null}} \qquad H_{\text{alt}}$$

- Get the MLE for each model $\hat{\theta} = \text{argmin}\mathcal{L}(\mathbf{d}|\mathbf{x}(\vec{\theta}))$

- Construct the likelihood ratio, and perform rejection

$$R = -2\log\left(\frac{\mathcal{L}(\mathbf{d}|\mathbf{x}(\hat{\theta}_{\text{null}}))}{\mathcal{L}(\mathbf{d}|\mathbf{x}(\hat{\theta}_{\text{alt}}))}\right) = \chi^2(\hat{\theta}_{\text{null}}) - \chi^2(\hat{\theta}_{\text{alt}})$$

$$F^{\chi^2}_{N_{\text{alt}}-N_{\text{null}}}(R) \geq \alpha$$

Frequentist Takeaways

- **Frequentist:**
- No knowledge of how 'likely' a given parameter/model is.

  → *Nature has chosen a model, and that is fixed.*
- Concepts:
  - Confidence interval, MLE, Likelihood ratio
  - Profile likelihood: Maximize likelihood of nuisance parameters

Nils Schöneberg nils.science@gmail.com

The Bayesian Land

From likelihood to inference

- **Bayesian:**
- We want to know what the **most probable model** is, not how likely the data is for a given model!
- Turn around the conditional of the probability:

$$p(\mathbf{d}|\mathbf{x}(\vec{\theta})) \rightarrow p(\mathbf{x}(\vec{\theta})|\mathbf{d})$$

Nils Schöneberg nils.science@gmail.com

From likelihood to inference

- **Bayesian:**
- Bayes law to the rescue:

$$P(s|t)P(t) = P(s \text{ and } t) = P(t|s) \cdot P(s)$$

100 patients	Medical test positive	Medical test negative
20 sick	10 positive	10 negative
80 healthy	5 positive	75 negative

$P(s) = 0.2$ $P(t) = 0.15$

$$P(s|t) = 10/(10+5) = {}^{2}/_{3}$$
$$P(t|s) = 10/(10+10) = 0.5$$

$$0.15 \cdot {}^{2}/_{3} = 0.1 = 0.5 \cdot 0.2$$

From likelihood to inference

- **Bayesian:**
- Bayes law to the rescue:

$$P(s|t)P(t) = P(s \text{ and } t) = P(t|s) \cdot P(s)$$

$$P(\mathbf{d}|\mathbf{x})P(\mathbf{x}) = P(\mathbf{d} \text{ and } \mathbf{x}) = P(\mathbf{x}|\mathbf{d})P(\mathbf{d})$$

$$p(\vec{\theta}|\mathbf{d}) = \frac{\mathcal{L}(\vec{\theta}|\mathbf{d})\pi(\vec{\theta})}{E}$$

From likelihood to inference

- **Bayesian:**
- The parameter prior $\pi(\vec{\theta})$ is our **prior belief** how likely a given value of $\vec{\theta}$ is, given previous experience.

$$p(\vec{\theta}|\mathbf{d}) = \frac{\mathcal{L}(\mathbf{d}|\vec{\theta}) \cdot \pi(\vec{\theta})}{E}$$

- The 'model evidence' E is the probability to observe the data **after considering all possible parameter values** of the given model (marginalizing over them, see later)

From likelihood to inference

- **Bayesian:**
- We manually **impose** the parameter prior

  (*Easy:* We have past experiments/experience, and we simply use that. *Hard:* We try to be 'objective' and use uninformative priors (uniform in value/logarithm, Jeffreys prior)
- How do we get 'model evidence' E ?

From likelihood to inference

- **Bayesian:**
- How do we get 'model evidence' E ?

  Trick: It does not depend on the model, so we can often ignore it!
- Example: Compute only $p(\vec{\theta}|\mathbf{d}) = \text{const} \cdot f(\vec{\theta}|\mathbf{d})$ and use $\int p(\vec{\theta}|\mathbf{d}) d\vec{\theta} \stackrel{!}{=} 1$ to compute const

  (This is done e.g. in MCMC sampling)

Nils Schöneberg nils.science@gmail.com

From likelihood to inference

- We can ***marginalize*** over irrelevant parameters

$$p(\vec{\lambda}|\mathbf{d}) = \int \mathcal{L}(\mathbf{d}|\vec{\lambda},\vec{n})\pi(\vec{\lambda},\vec{n})d\vec{n}$$

- If we marginalize over all parameters, we get

$$E = \int \mathcal{L}(\mathbf{d}|\vec{\theta})\pi(\vec{\theta})d\vec{\theta}$$

- This is estimated by sophisticated samplers like `Multinest` or `Polychord`, but requires knowledge of $\mathcal{L}(\mathbf{d}|\vec{\theta})$ even in 'uninteresting' regions. Hence often expensive

Credible Intervals

- In Bayesian context, we can now get posterior probability $p(\vec{\theta}|\mathrm{d})$
- Convert to credible interval via:

$$\int_{\mathrm{low}}^{\mathrm{high}} p(\vec{\theta}|\mathbf{d}) = \alpha \qquad \int_{\mathrm{boundary}}^{\mathrm{high}} p(\vec{\theta}|\mathbf{d}) = \alpha$$

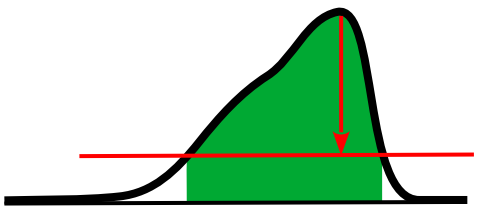
- Highest probability, symmetric, equal-tailed

Nils Schöneberg nils.science@gmail.com

Credible Intervals

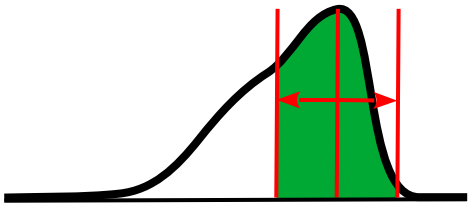
$$\int_{\text{low}}^{\text{high}} p(\vec{\theta}|\mathbf{d}) = \alpha$$

Highest probability



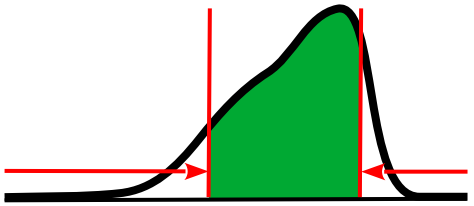
$$p(\text{low}|\mathbf{d}) = p(\text{high}|\mathbf{d})$$

symmetric



$$\text{low} = \mathfrak{m} - \delta$$
$$\text{high} = \mathfrak{m} + \delta$$

equal-tailed



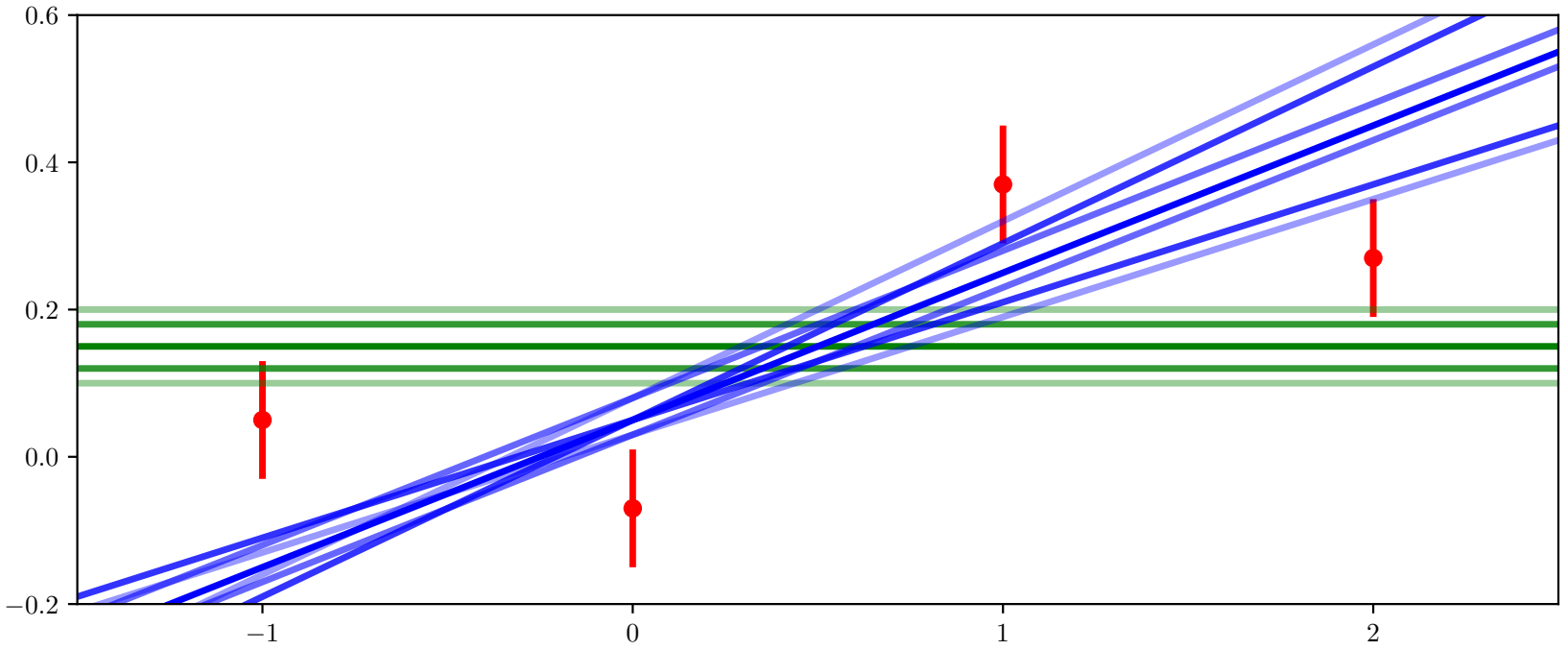
$$\int_{-\infty}^{\text{low}} p(\vec{\theta}|\text{d}) = \frac{1-\alpha}{2}$$

$$\int_{\text{high}}^{+\infty} p(\vec{\theta}|\text{d}) = \frac{1-\alpha}{2}$$

Nils Schöneberg nils.science@gmail.com

Credible Intervals (Example)

Example: Line **M1** or Constant **M2**?

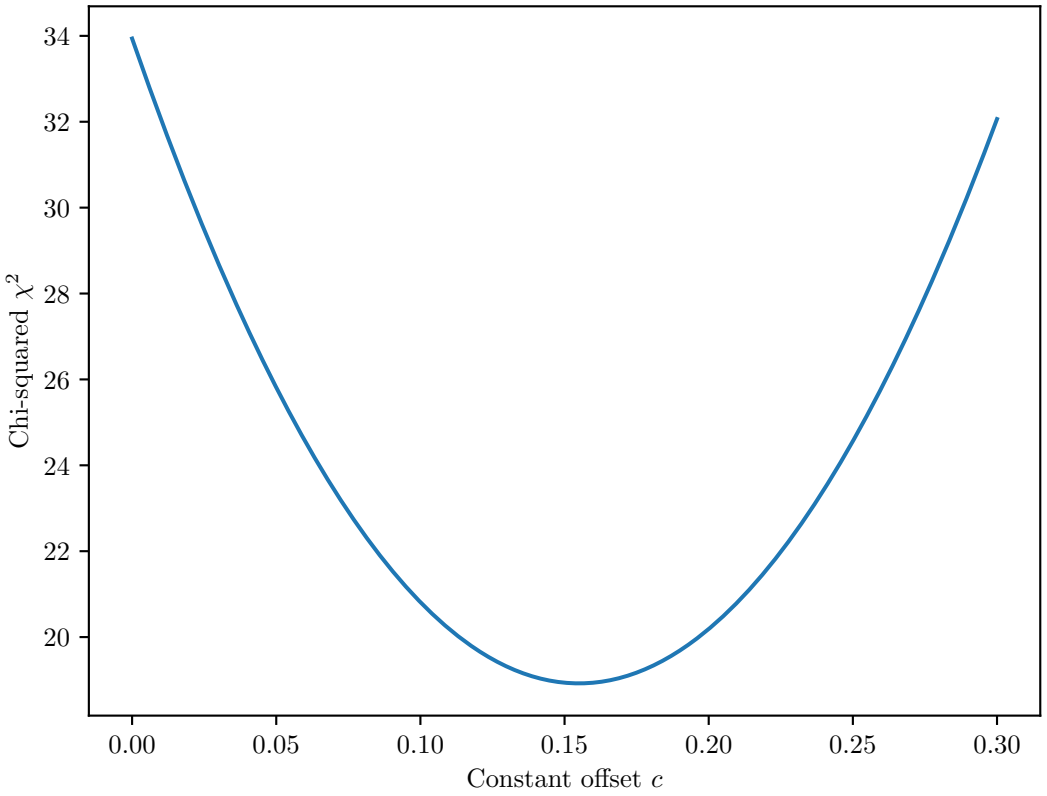


Nils Schöneberg nils.science@gmail.com

Credible Intervals (Example)

Constant **M2**

$$\chi^2_{\text{eff}} = -2 \ln p(\vec{\theta}|\mathbf{d})$$

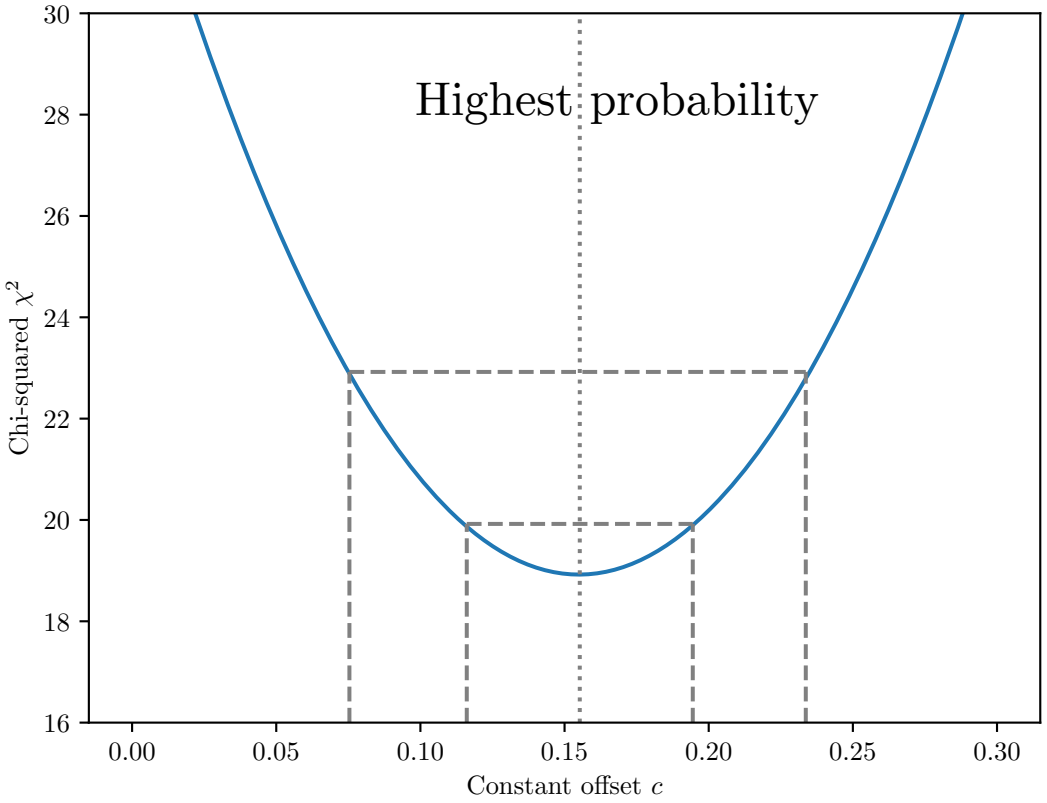


Nils Schöneberg nils.science@gmail.com

Credible Intervals (Example)

Constant **M2**

$$\chi^2_{\text{eff}} = -2 \ln p(\vec{\theta}|\mathbf{d})$$

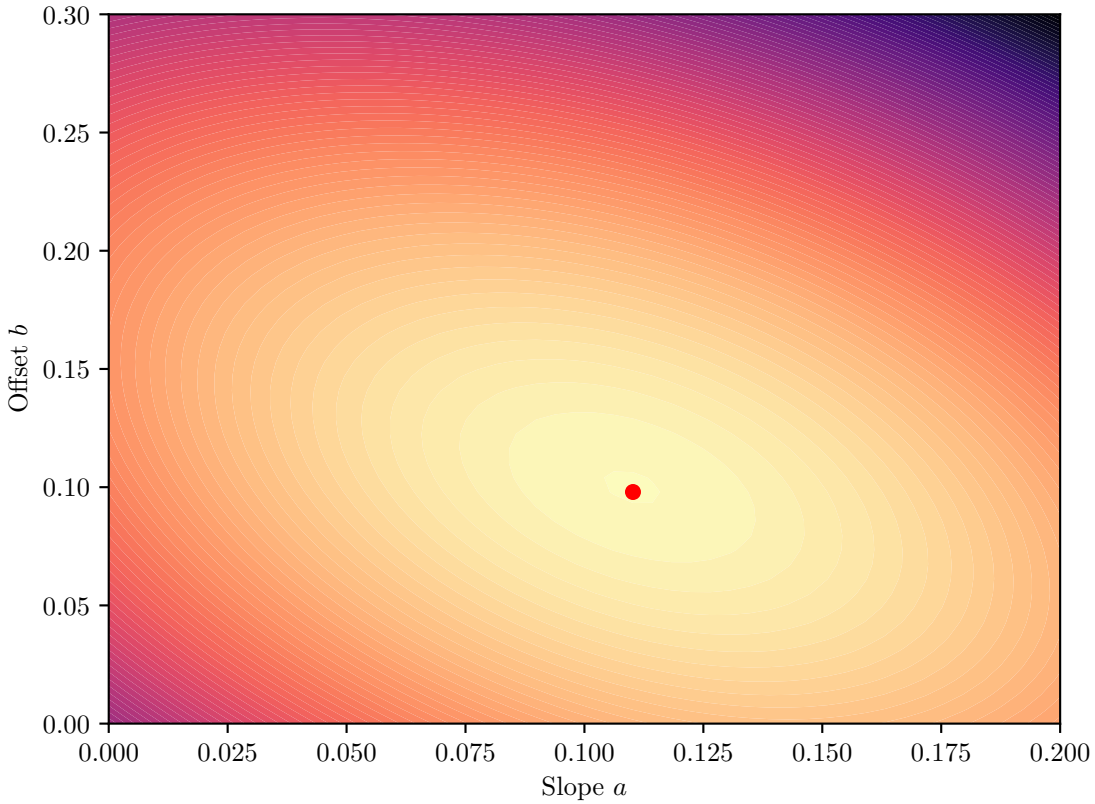


Nils Schöneberg nils.science@gmail.com

Credible Intervals (Example)

Line M1

$$\chi^2_{\text{eff}} = -2 \ln p(\vec{\theta}|\mathbf{d})$$

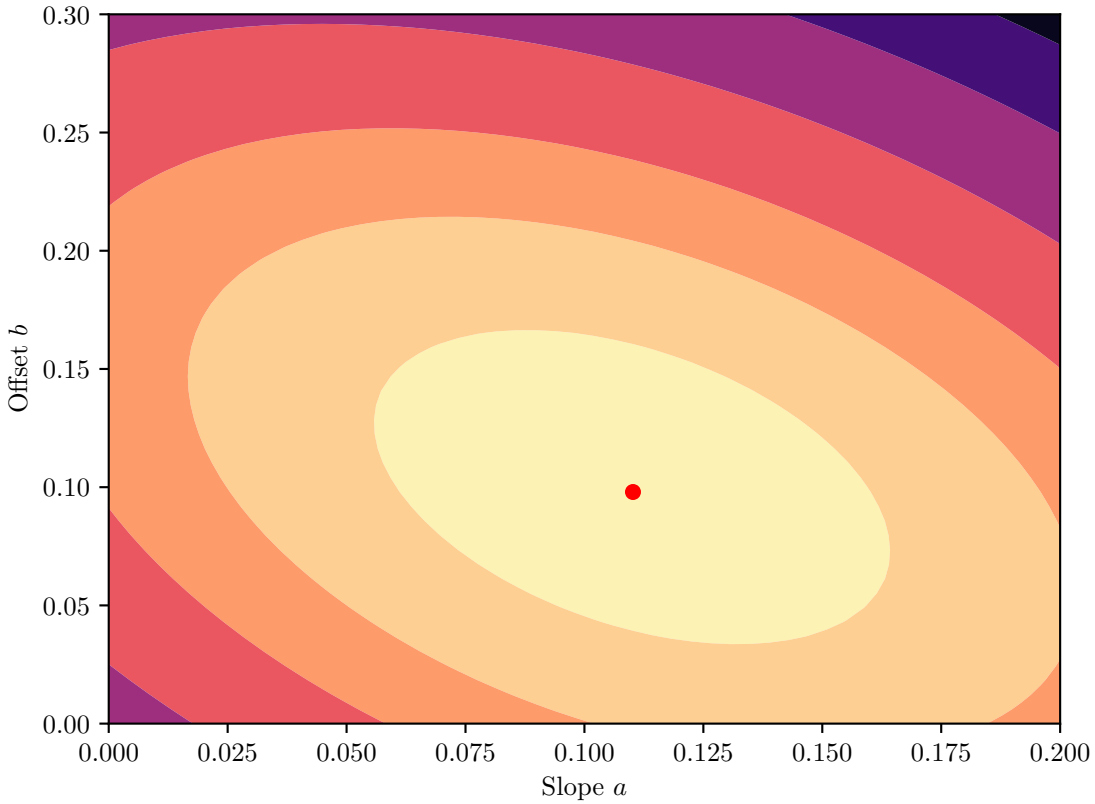


Nils Schöneberg nils.science@gmail.com

Credible Intervals (Example)

Line M1

$$\chi^2_{\text{eff}} = -2 \ln p(\vec{\theta}|\mathbf{d})$$



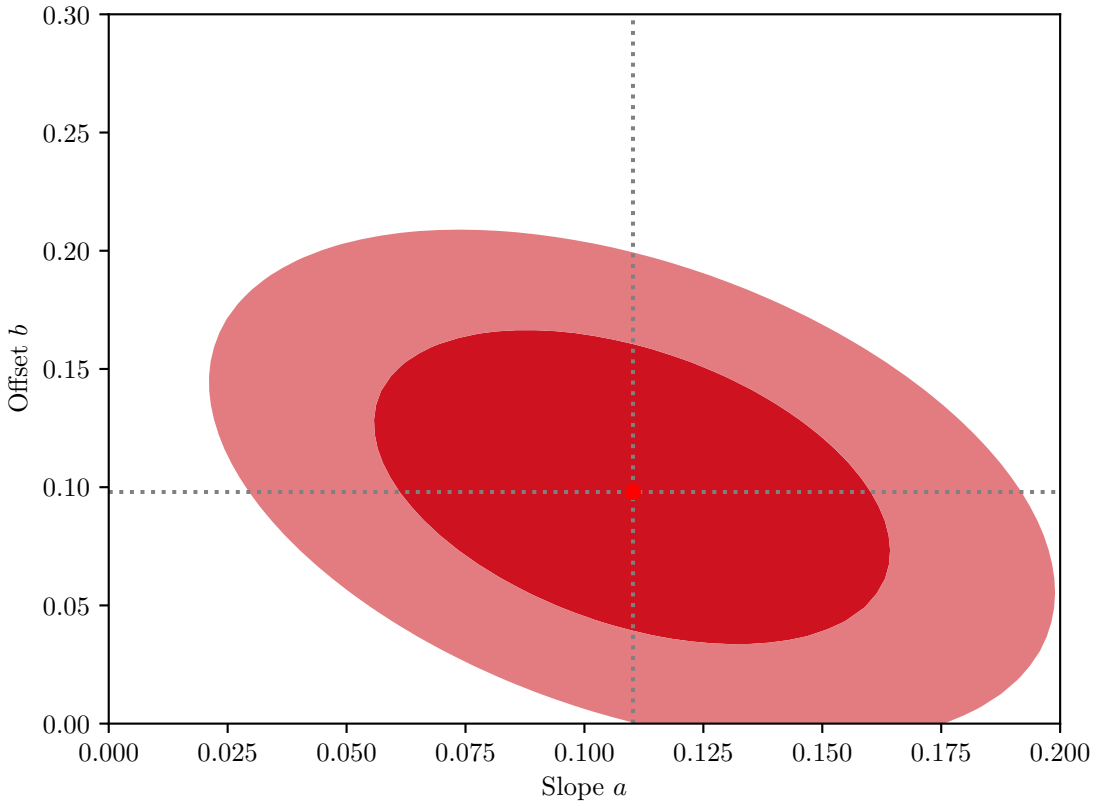
Nils Schöneberg nils.science@gmail.com

Credible Intervals (Example)

Line M1

$$\chi^2_{\text{eff}} = -2 \ln p(\vec{\theta}|\mathbf{d})$$

Highest probability



Bayesian model comparison

- We have reached the goal of specifying which parameters of a given model we believe are more probable or less probable to be compatible with the data given our prior beliefs
- How do we **compare models**?

Bayesian model comparison

- Compute *Bayes factor*

$$\frac{p(M_1|\mathbf{d})}{p(M_2|\mathbf{d})} = \frac{\pi(M_1)}{\pi(M_2)} \frac{E(\mathbf{d}|M_1)}{E(\mathbf{d}|M_2)}$$

- Advantage: Purely Bayesian
- Disadvantage:
  - Expensive to compute (evidence)
  - Prior-dependence (!)
  - Interpretability? → Jeffrey's scale

Bayesian model comparison

- **Simpler approaches (more frequentist):**
- If model is nested, then one can compute Savage-Dickey ratio

$$\frac{p(\mathbf{d}|M_1)}{p(\mathbf{d}|M_2)} = \frac{p(\vec{\theta}|\mathbf{d})}{p(\vec{\theta_*}|\mathbf{d})}$$

  (we get ratio of evidences without computing any)
- Information Criterion (AIC, BIC, ...) $\rightarrow$ Compute differences in χ^2 and penalize large parameter spaces (overfitting)
- (Comparing tensions in different models)

Nils Schöneberg nils.science@gmail.com

Information Criteria

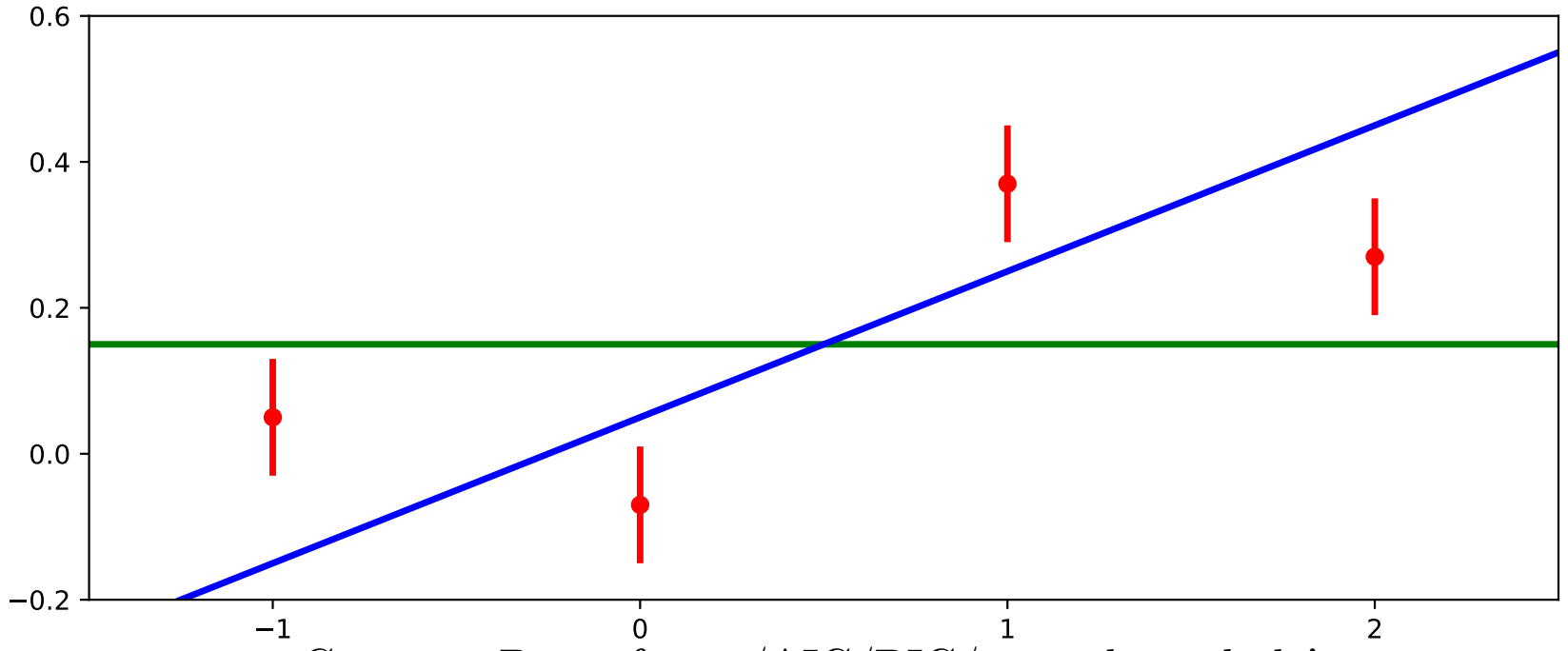
- AIC: $(\chi^2_{M1} - \chi^2_{M2}) + 2(N_{M1} - N_{M2})$
- BIC: $(\chi^2_{M1} - \chi^2_{M2}) + \ln n_{\text{data}} \cdot (N_{M1} - N_{M2})$
- DIC: $(\chi^2_{M1} - \chi^2_{M2}) + 2(g_{M1} - g_{M2})$
- ...

$$g_M = \overline{[\chi^2_M(\vec{\theta})]} - \chi^2_M(\overline{[\vec{\theta}]})$$

Nils Schöneberg nils.science@gmail.com

The likelihood

- Which of **M1** and **M2** is the better model?

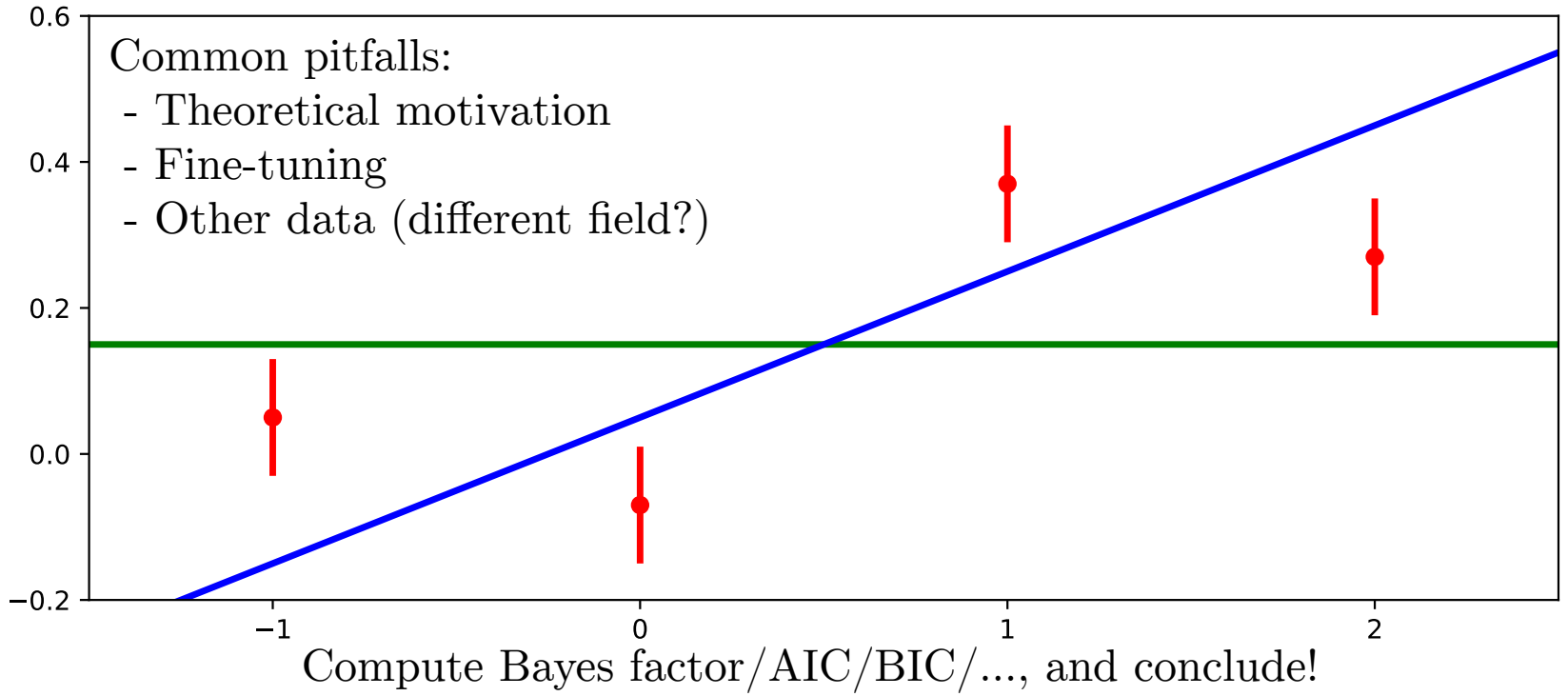


Compute Bayes factor/AIC/BIC/..., and conclude!

Nils Schöneberg nils.science@gmail.com

The likelihood

- Which of **M1** and **M2** is the better model?



Nils Schöneberg nils.science@gmail.com

Chains and Samplers

The MCMC

- Mostly based on concept of a 'chain', e.g. in Markov-chain Monte Carlo (MCMC)
- Collection of sampling locations (parameter values) with **weights** (and corresponding likelihood)
- The weighted density of the samples represents the posterior density!

The MCMC

- The weighted density of the samples represents the posterior density!

The MCMC

We use the idea of a Markov chain process:

$$P(X_{t+1} | X_t, X_{t-1}, \ldots X_1) = P(X_{t+1} | X_t)$$

- The weighted density of the samples represents the posterior density!

The MCMC

We use the idea of a Markov chain process:

$$P(X_{t+1} | X_t, X_{t-1}, \ldots X_1) = P(X_{t+1} | X_t)$$

$\rightarrow$ Can be modeled with states and transitions

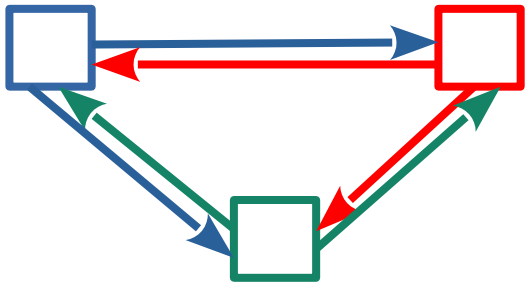
Nils Schöneberg nils.science@gmail.com

The MCMC

We use the idea of a Markov chain process:

$$P(X_{t+1} | X_t, X_{t-1}, \dots X_1) = P(X_{t+1} | X_t)$$

$\rightarrow$ Can be modeled with states and transitions



$$\begin{pmatrix} P(a \to a) & P(a \to b) & P(a \to c) \\ P(b \to a) & P(b \to b) & P(b \to c) \\ P(c \to a) & P(c \to b) & P(c \to c) \end{pmatrix} \cdot \begin{pmatrix} a_t \\ b_t \\ c_t \end{pmatrix} = \begin{pmatrix} a_{t+1} \\ b_{t+1} \\ c_{t+1} \end{pmatrix}$$

$$M\mathbf{s}_t = \mathbf{s}_{t+1}$$

$$M\mathbf{s}_\infty = \mathbf{s}_\infty$$

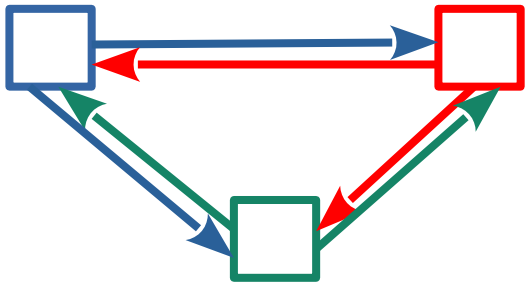
Nils Schöneberg nils.science@gmail.com

The MCMC

We use the idea of a Markov chain process:

$$P(X_{t+1} | X_t, X_{t-1}, \dots X_1) = P(X_{t+1} | X_t)$$

$\rightarrow$ Can be modeled with states and transitions



$$\begin{pmatrix} P(a \to a) & P(a \to b) & P(a \to c) \\ P(b \to a) & P(b \to b) & P(b \to c) \\ P(c \to a) & P(c \to b) & P(c \to c) \end{pmatrix} \cdot \begin{pmatrix} a_t \\ b_t \\ c_t \end{pmatrix} = \begin{pmatrix} a_{t+1} \\ b_{t+1} \\ c_{t+1} \end{pmatrix}$$

$$M\mathbf{s}_t = \mathbf{s}_{t+1}$$

$$M\mathbf{s}_\infty = \mathbf{s}_\infty$$

Stationary distribution!

The MCMC

We use the idea of a Markov chain process:

$$P(X_{t+1} | X_t, X_{t-1}, \ldots X_1) = P(X_{t+1} | X_t)$$

$\rightarrow$ Has stationary distribution $\mathbf{s_\infty}$ $\qquad M\mathbf{s}_\infty = \mathbf{s}_\infty$

Crucial insight: Make Markov chain process with $\mathbf{s_\infty}$ = **posterior density**

The MCMC

We use the idea of a Markov chain process:

$$P(X_{t+1} | X_t, X_{t-1}, \ldots X_1) = P(X_{t+1} | X_t)$$

$\rightarrow$ Has stationary distribution $\mathbf{s_\infty}$ $\qquad M\mathbf{s}_\infty = \mathbf{s}_\infty$

Crucial insight: Make Markov chain process with $\mathbf{s_\infty}$ = **posterior density**

Only need to adjust transition probabilities (M) !

The MCMC

We use the idea of a Markov chain process:

$$P(X_{t+1} | X_t, X_{t-1}, \ldots X_1) = P(X_{t+1} | X_t)$$

$\rightarrow$ Has stationary distribution $\mathbf{s_\infty}$ $\qquad M\mathbf{s}_\infty = \mathbf{s}_\infty$

Crucial insight: Make Markov chain process with $\mathbf{s_\infty}$ = **posterior density**

Only need to adjust transition probabilities (M) !

MCMC name: We generate a Markov chain by Monte Carlo simulation of jumping points

Nils Schöneberg nils.science@gmail.com

The MCMC

Detailed balance:

$$n(a)P(a \to b) = n(b)P(b \to a)$$

$$\frac{n(a)}{n(b)} = \frac{P(b \to a)}{P(a \to b)} \stackrel{!}{=} \frac{p(a|\mathbf{d})}{p(b|\mathbf{d})} = \frac{\mathcal{L}(\mathbf{d}|a)\pi(a)}{\mathcal{L}(\mathbf{d}|b)\pi(b)}$$

Naive: $P(b \to a) = p(a|\mathbf{d})$

Evidence :/

Better: $P(b \to a) = \mathcal{L}(\mathbf{d}|a)\pi(a)$

Metropolis-Hastings: $P(b \to a) = \min\left(1, \frac{\mathcal{L}(\mathbf{d}|a)\pi(a)}{\mathcal{L}(\mathbf{d}|b)\pi(b)}\right)$

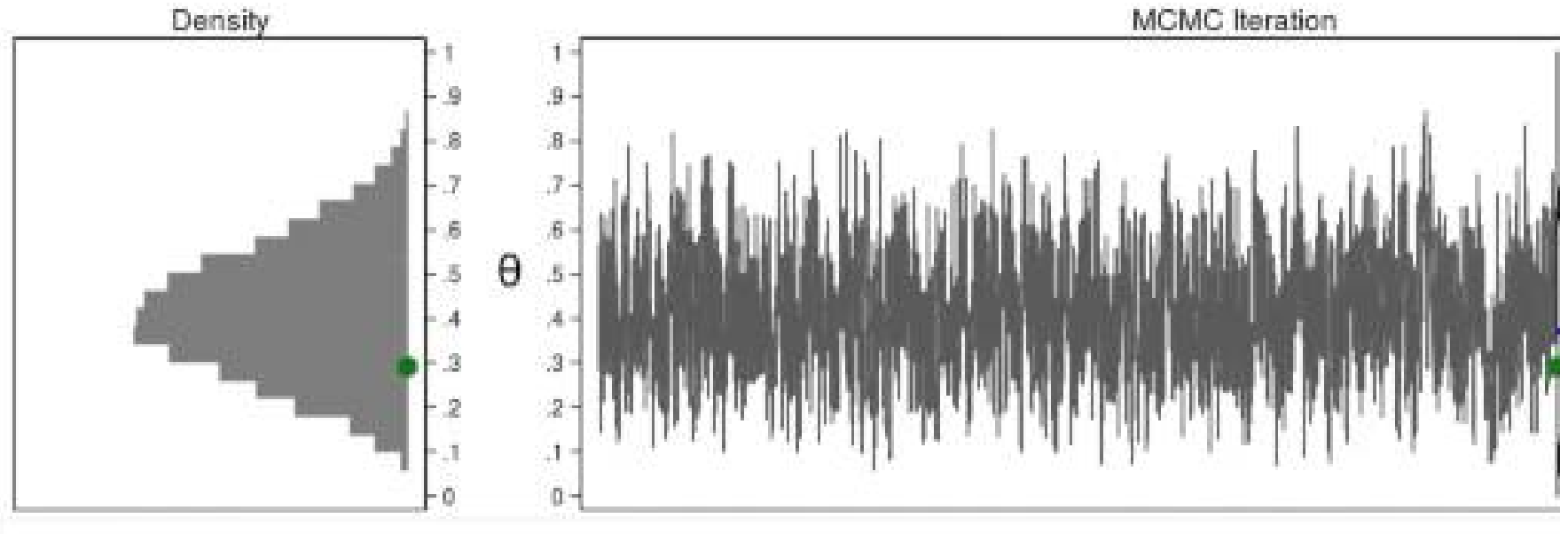
Reminder: $p(a|\mathbf{d}) = \frac{\mathcal{L}(\mathbf{d}|a)\pi(a)}{E}$

The MCMC

One of the main goals of MontePython :
Generate a MCMC sample using
Metropolis Hastings algorithm

- The weighted density of the samples represents the posterior density!

The MCMC



- The weighted density of the samples represents the posterior density!

The MCMC

- Use a ***sampler*** to generate the locations and weights.
- Metropolis-Hastings : Step **randomly**, ***BUT*** only ***accept*** step with probability

$$p = \min\left(\frac{\mathcal{L}(\vec{\theta})\pi(\vec{\theta})}{\mathcal{L}(\vec{\theta}_{\text{prev}})\pi(\vec{\theta}_{\text{prev}})}, 1\right)$$

- Otherwise, remain on same point $\rightarrow$ ***reject*** step and increase weight of current point by $+1$

Hamilton MonteCarlo

- Use a ***sampler*** to generate the locations and weights.
- Hamilton Monte-Carlo: Define steps by using Hamiltonian dynamics, most efficient when derivatives of likelihood are known. Otherwise similar

  Keep acceptance rate high (less waste) by

$$\frac{\mathcal{L}(\vec{\theta})\pi(\vec{\theta})}{\mathcal{L}(\vec{\theta}_{\text{prev}})\pi(\vec{\theta}_{\text{prev}})} \to 1$$

Using momenta to stay in typical set

Nested Sampling

- Problems of MCMC:
  - No evidence (samples only exist where posterior is large)
  - Multi-modality (cannot jump to regions separated by large gap)
- The fundamental issue is the Markov Chain
- Try instead $\rightarrow$ nested sampling

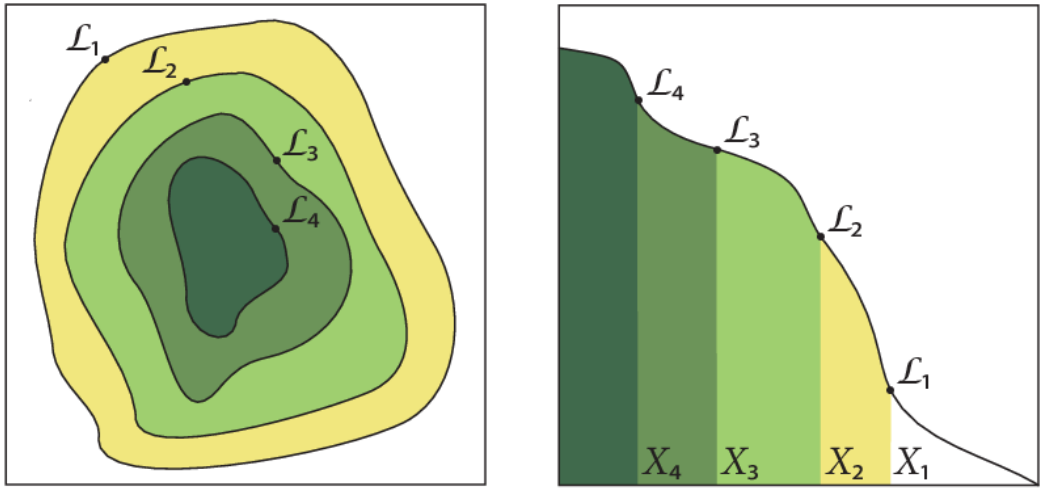
Nested Sampling

- Idea: Sample all prior, keep track of 'good' regions
$$\xi(\lambda) = \int_{\mathcal{L}(\mathbf{d}|\vec{\theta})>\lambda} \pi(\vec{\theta}) d\vec{\theta}$$
- *Slowly* increase λ until maximum found
- We get the evidence for free:
$$E = \int \mathcal{L}(\mathbf{d}|\vec{\theta}) \cdot \pi(\vec{\theta}) d\vec{\theta} = \int \mathcal{L}(\xi) d\xi$$

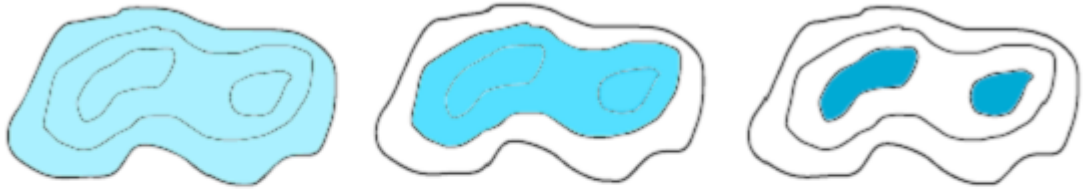
Nils Schöneberg nils.science@gmail.com

Nested Sampling

- Regions of increasing $\xi(\lambda)$ are by definition nested



Multi-modal = no problem



Nested Sampling

- In practice:
  - Keep around *live points* that obey $\mathcal{L}(\mathbf{d}|\vec{\theta}) > \lambda$

    and thus define a region $\xi(\lambda)$ from their convex hull
  - Progressively kill worst point ($\rightarrow$ *dead points*), and replace by interior point (with higher likelihood)
  - There might be multiple convex hulls necessary for multi-modal distributions $\rightarrow$ *clusters*

Note on Nested Sampling

- Upsides:

  Multi-modal and elongated posteriors can be captured ‘easily’

  You get the actual evidence (!)

- Downsides:

  Sometimes a bit fickle to install/computationally expensive

  A bit more black-box-y

Nils Schöneberg nils.science@gmail.com

Convergence

- Test convergence using |R-1|
- $$R - 1 = \frac{B/W - 1}{L}$$

$$B = \frac{L}{N_{\text{chain}} - 1} \sum_{\text{chain}} (\bar{x}_{\text{chain}} - \bar{\bar{x}})^2$$

Variance of parameter between chains

$$W = \frac{1}{N_{\text{chain}}} \sum_{\text{chain}} \frac{1}{L_j - 1} \sum (x^i_{\text{chain}} - \bar{x}_{\text{chain}})^2$$

Inference Codes

- Lots of codes on the market:
  - MontePython
  - Cobaya
  - Cosmosis
  - CosmoMC (fortran!)
  - emcee
  - zeus
  - many more

Nils Schöneberg nils.science@gmail.com

How to use MontePython

MontePython

- Pros:
  - Large likelihood library
  - Can produce triangle plots & derive constraints
  - Variety of available samplers
- Cons:
  - Currently compatible mostly only with CLASS
  - Not everything do-able in python
  - No `pip install`

MontePython

- To install:
  - Git clone:

    https://github.com/brinckmann/montepython_public
  - Change directory:

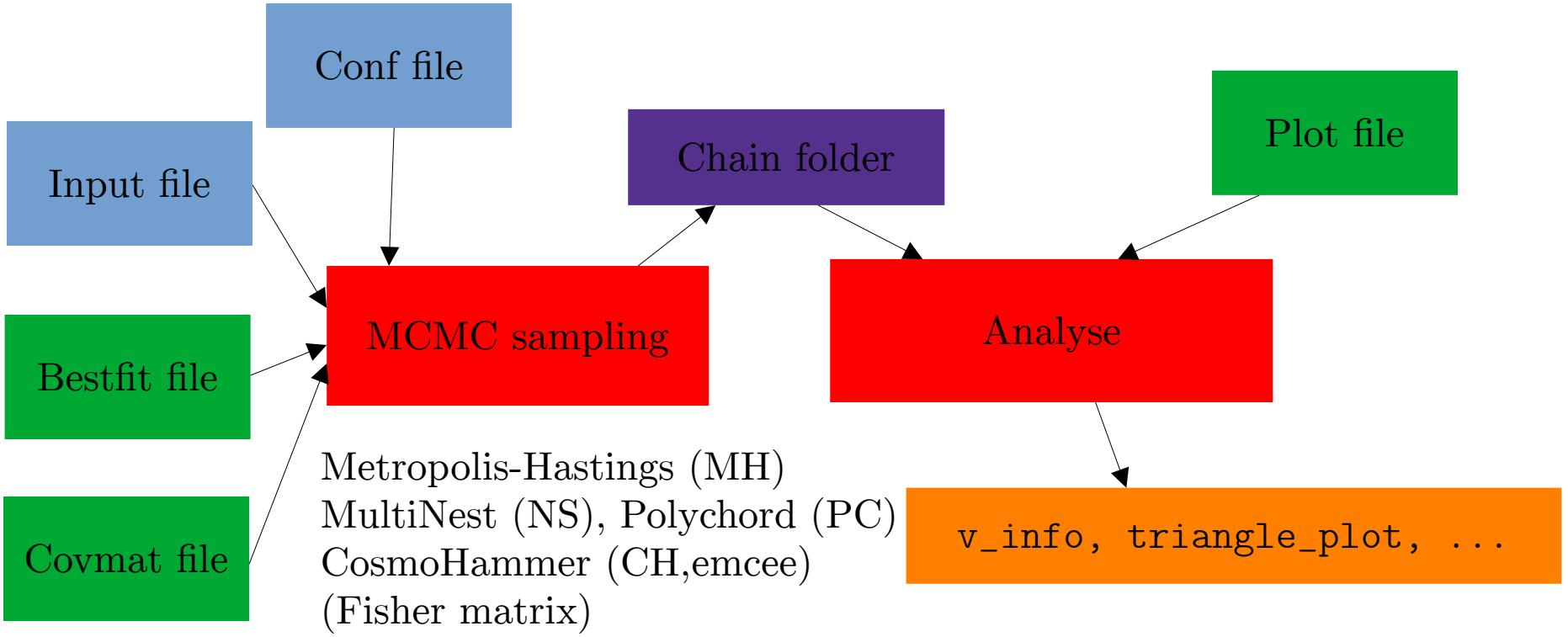
    `cd montepython_public`
  - Run test:

    `python montepython/MontePython.py run --help`

Notebook!

MontePython

- Layout of code:



MontePython

- Conf file:

```
root = 'path/to/your/home'

path['cosmo'] = root+'/path/to/class'

path['clik'] = root+'/path/to/planck/code/plc_3.0/plc-3.01'
```

Nils Schöneberg nils.science@gmail.com

MontePython

- Input file:

which likelihoods to use?

```
data.experiments = ['one', 'two', 'three']

data.oversampling = [1, 10, 5]
```

are some nuisance parameters faster?
cosmo, lkl1, lkl2, ...

```
data.parameters['name'] = [ mean, lower, upper, sigma, scale=1, 'type']
```

estimated

uniform prior

estimated, important if
no covmat file

'cosmo', 'nuisance', 'derived'

MontePython

- Input file:

```
data.cosmo_arguments[‘name’] = value/’value’

likelihood.setting_name = value/’value’
```

MontePython

Generate folder, check that everything works:

```
python montepython/MontePython.py run
   -p input/name.param -o chains/name -N 1 -f 0 --conf xxx.conf
```

Perform the actual run *(no -f 0)*:

```
mpirun -np NNN python montepython/MontePython.py run
   -p input/name.param -o chains/name -N 10000000 --conf xxx.conf

   --bestfit XXX.bestift --covmat XXX.covmat
   -f 1.6
```

MontePython

- Dangers and pitfalls:
  - The first run generates a `log.param` file, containing all info from the run. **Do not modify!**
  - If a `log.param` is detected in a folder, MontePython will ***ignore*** the input file!
  - Make sure you have a covmat or you provide reasonable `sigma`, otherwise your sampler might be stuck trying to get new points
  - Avoid multi-modality and sharp degeneracies!

Parallelization

- We use
  - MPI (multi-process)
  - OpenMPI (multi-thread)
- Class benefits from openMP up until 8-16 cores
- Not a lot of MPI communication necessary $\rightarrow$ good MPI parallelization

Which lkls are present?

- Bicep/Keck `BK15` (BB constraint)
- Kids-1000 `K1K_BandPowers`, `K1K_COSEBIs`, `K1K_CorrelationFunctions` (Weak lensing)
- Mike/HIRES `Lya_abgd` (Lyman-α 1D forest)
- Pantheon+ `Pantheon_Plus`, `Pantheon_Plus_SH0ES` (Supernovae Ia)

Which lkls are present?

- Planck 2018 `Planck_highl_TTTEEE`, `Planck_lowl_TT`, `Planck_lowl_EE` (CMB lensed anisotropies)
- Planck 2018 `Planck_lensing` (CMB lensing reconstruction)

Which lkls are present?

- BOSS+eBOSS `bao_eBOSS_DR16_Lya_auto`, `bao_eBOSS_DR16_Lya_cross_QSO`, `bao_eBOSS_DR16_gal_QSO` (Baryonic acoustic oscillations in galaxies, Quasi-stellar objects, Lyman-α forest)
- BBN `bbn_omega_b` (Primordial light element abundance, Cooke, Aver)
- CC `cosmic_clocks_2016` (Cosmic Chronometers)

Which lkls are present?

- **External likelihoods downloaded elsewhere:**
- ACT+SPT `ACTPol_lite_DR4, spt3g_y1` (CMB angular power spectra: https://github.com/ACTCollaboration/pyactlike https://github.com/SouthPoleTelescope/spt3g_y1_dist)
- Priors: H0 , S8 , ... → Program yourself (see later)

Which lkls are present?

- **Forecasting likelihoods:**
- Future CMB proposals `cmb_s4`, `core_m5`, `litebird`
- Future Large scale structure proposals `euclid_lensing`, `euclid_pk`, `ska1_pk`, `ska2_pk`
- Future Spectral distortion proposals `pico`, `pixie`

Nils Schöneberg nils.science@gmail.com

How to program your own likelihood

NAME/NAME.data:

```
NAME.property = ‘‘Hello world!’’
NAME.redshift = 3.4
NAME.h = 0.73
NAME.errH = 0.02
```

NAME/__init__.py:

```
class NAME(Likelihood):
    def __init__(self, path, data, command_line):
        print(self.property)
    def loglkl(self, cosmo, data):
        H = cosmo.Hubble(self.redshift)*2997.92458
        chi2 = (H - self.h)**2/self.errH**2
        return -0.5 * chi2
```

Inner Workings

- Covmat-Aligned:

  $\rightarrow$ Convert current C into $C = L^T L$, then use

  $\Delta\theta = L\Delta\tilde{\theta}$

- Sample orthogonal axis-aligned jumps in $\Delta\tilde{\theta}$

  $\rightarrow$ Jumping in eigenvector directions

Inner Workings

- Proposal distribution:

$$P(\theta \to \theta') = \mathcal{N} \exp\left(-\frac{1}{2c}\Delta\theta^T C^{-1} \Delta\theta\right)$$

- With C the current covariance matrix estimate

$\mathrm{c} = f^2/N_{\mathrm{par}}$

Inner Workings

- Fast-slow: $C = L^T L$

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 \\ 1 & 3 & 4 & 0 \\ 2 & 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$$

$$(\alpha, \beta) = (0, 0) \rightarrow (a, b) = (0, 0)$$

- Can sample fast nuisance parameters more frequently than slow cosmo parameters! (oversampling)

Inner Workings

- Update: Update covmat every N_{update} cycles

  $\rightarrow$ Only if $0.4 < \max(|R\text{-}1|) < 3$
- This makes new points *technically* depend on old points $\rightarrow$ **non-markovian**
- When analyzing, we have to do:

  `--keep-non-markovian`

Inner Workings

- SuperUpdate: Update jumping factor every $N_{\text{super-update}}$ cycles

  $\rightarrow$ Get acceptance rate close to 0.26
  (astro-ph/0405462)

- $f = 2.4 \rightarrow c_k = c_{k-1} + \dfrac{a - 0.26}{k - k_{\text{last-update}}}$

- $c_k = f_k^2 / N_{\text{par}}$ $\qquad f_k = \sqrt{N_{\text{par}} c_k}$

Big impact right after update, then quickly diminishing

Inner Workings

- Fisher mode:
- `--method Fisher`
- `--fisher-delta 0.1`
- `--fisher-tol 0.05`

$$F_{ij} = \frac{1}{2} \frac{\partial^2 \chi^2}{\partial \theta_i \partial \theta_j}$$

$$\Delta \mathcal{L}(\mathbf{d}|\vec{\theta}) \stackrel{!}{=} 0.1 \pm 0.05$$

$$C_{ij} \approx (F^{-1})_{ij}$$

Approximate likelihood at peak by second-order polynomial

MontePython

- Analysis step:

```
python montepython/MontePython.py info
    chains/name --extra plot_files/xxx.plot
```

Plotting options, next slide

```
--keep-non-markovian --want-covmat --noplot --noplot-2d --noplot-2d-diag
```

Keep around points even when covariance matrix has been updated

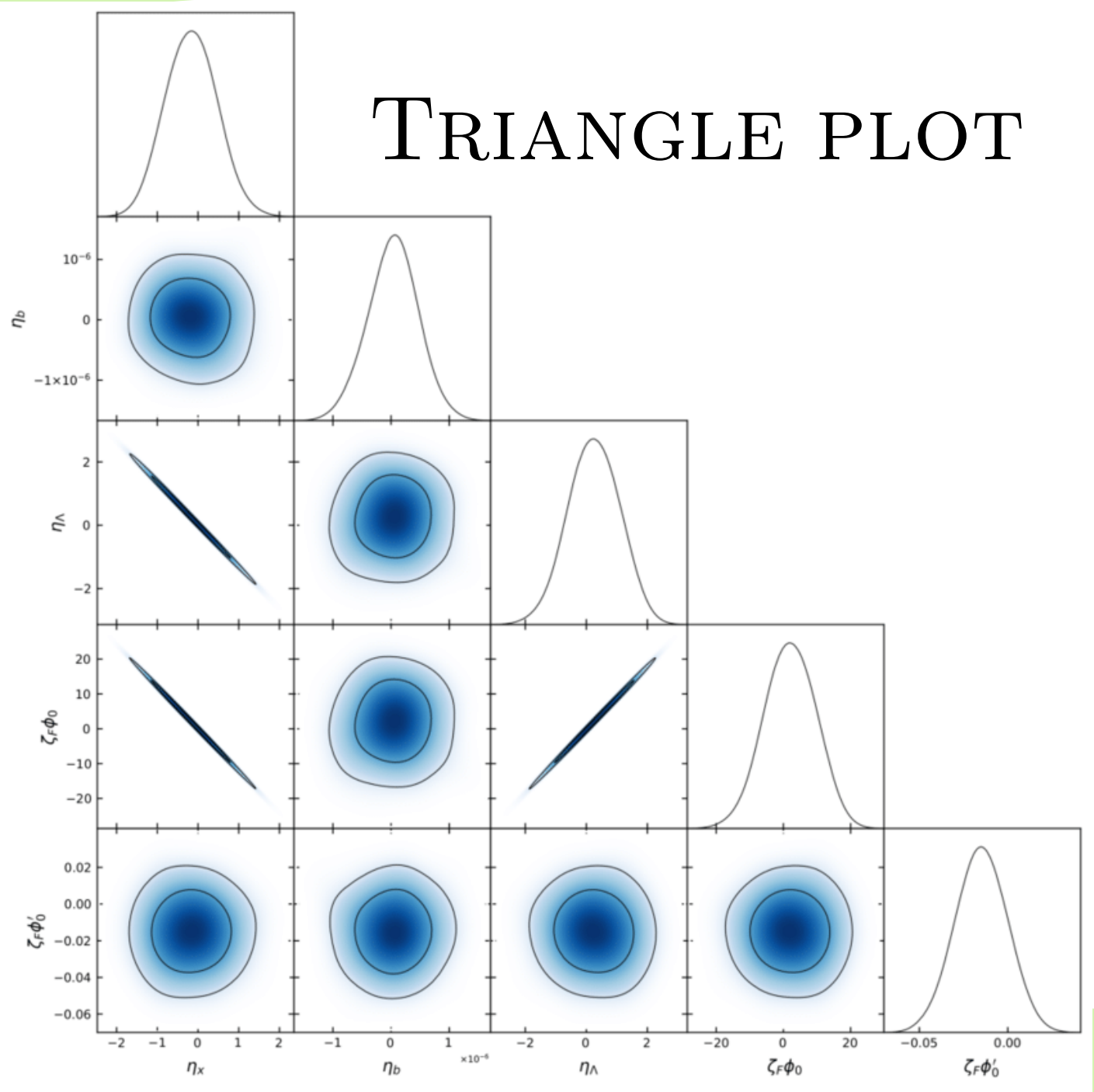
Compute the covmat (!) danger: if not enough points, this will destroy future runs in same folder

Only 1d posteriors

2d posteriors without 1d posteriors on diagonal

Nils Schöneberg nils.science@gmail.com

Triangle plot



Plot file options

```
info.to_derive = {'der1':'H0*H0*Omega_cdm'}

info.to_reorder = ['der2','H0']

info.to_change = {'rs_d':'r^s_d'}

info.to_plot = ['omega_b','H0','r^s_d']

info.force_limits = {'H0':[60,70]}

info.ticksize = 10

info.legendnames = ['run 1','run 2']

info.custom2d = ['add_cool_thing.py']
```

MontePython

- Run generates `log.param`
- Analyze generates `name.bestfit` and `name.log`
- If `--want-covmat` is set, also `name.covmat` file
- Run can auto-generate `name.bestfit` and `name.covmat` if sufficiently converged
- The file `name.v_info` or `name.h_info` or `name.tex` contains all info about |R-1|/bestfit/mean/confidence limits

Nils Schöneberg nils.science@gmail.com

How to use Cobaya

Cobaya

- Pros:
  - Compatible with CAMB & CLASS
  - Available in python, `pip install`
  - Easier to use (e.g. cobaya-install xxx)
- Cons:
  - Smaller likelihood library
  - No plotting

Cobaya

- Very simple MPI-parallelized Metropolis-Hastings MCMC:

```
mpirun -n NNN cobaya-run name.yaml
```

Cobaya

- input yaml

```
likelihood:
    lkl1:
        setting1: True
        setting2: 42
    lkl2: None
```

- Parameter without prior is derived

```
params:
    param1:
        prior:
            min: 0
            max: 1
        ref:
            dist: norm
            loc: 0
            scale: 0.25
        latex: \mathcal{cool\_latex}
    param2:
        prior:
            ...
    param3:
        latex:
```

Cobaya

- input yaml

```
params:
    param4:
        drop: True
    param5:
        value: ‘‘lambda param4:
np.exp(param4)’’
    param6:
        derived: ‘‘lambda x:x**2
```

```
Theory:
    classy:
        extra_args:
            P_k_max_1/Mpc: 10
            modes: s
output: folder
```

```
sampler:
    polychord: option_dict
```

```
prior:
    additional_prior: python_function
```

GetDist

```
from getdist import plots, loadMCSamples

samples = loadMCSamples(‘‘folder’’)
samples2 = loadMCSamples(‘‘other_folder’’)

gdp = plots.get_subplot_plotter()

gdp.triangle_plot([samples, samples2], params=[‘a’,’b’,’c’],
filled=True,legend_labels=[‘‘run1’’,’’run2’’])
```

Typically nicer plots, but comes with its own set of difficulties

Cobaya + Getdist directly

```
from cobaya.run import run
updated_info, sampler = run(model_info)
```

- `from getdist.mcsamples import MCSamplesFromCobaya`

  ```
  gd_sample = MCSamplesFromCobaya(updated_info,
  sampler.products()["sample"])
  ```

- `import getdist.plots as gdplt`

```
gdplot = gdplt.get_subplot_plotter()
gdplot.triangle_plot(gd_sample, ["a", "b"], filled=True)
```

Cobaya in python

```
from cobaya.model import get_model

model = get_model(info)

model.prior.sample()
point = {‘parameter’:value}
model.logpost(point)
```

CMB Likelihoods

- Newest version of `clik`:
  - https://github.com/benabed/clik
- Now to compile:
- `./waf configure`
- `./waf install`
- `source bin/clik_profile.sh`
- Check: `python -c "import clik"`

Nils Schöneberg nils.science@gmail.com

CMB Likelihoods

- Install actual likelihood data:
- http://pla.esac.esa.int/pla/#home
- Cosmology → Likelihood →
  - COM_Likelihood_Data-baseline_R3.00.tar.gz
  - [COM_Likelihood_Code-v3.0_R3.10.tar.gz]
- .conf:
- path['clik'] = root+'xxx/code/plc_3.0/plc-3.1'
- path = path['clik']+'../../../baseline/plc_3.0/hi_l/plik/xx.clik'
- Planck_highl_TTTEEE.path_clik = os.path.join(data.path['clik'], '../../../baseline/plc_3.0/hi_l/plik/plik_rd12_HM_v22b_TTTEEE.clik')

CMB Likelihoods

- SPT likelihood:
- https://github.com/SouthPoleTelescope/spt3g_y1_dist
- `spt3g_montepython_install /path/to/montepython /path/to/clik`
- Test: `python /path/to/montepython/montepython/MontePython.py run -o base_spt -p base_spt3g_y1.param`
- By default: `spt3g_y1.path_clik = '/path/to/montepython/data/spt_data/spt3g_Y1_v1_TTTEEE.clik'`

Any Questions?

nils.science@gmail.com