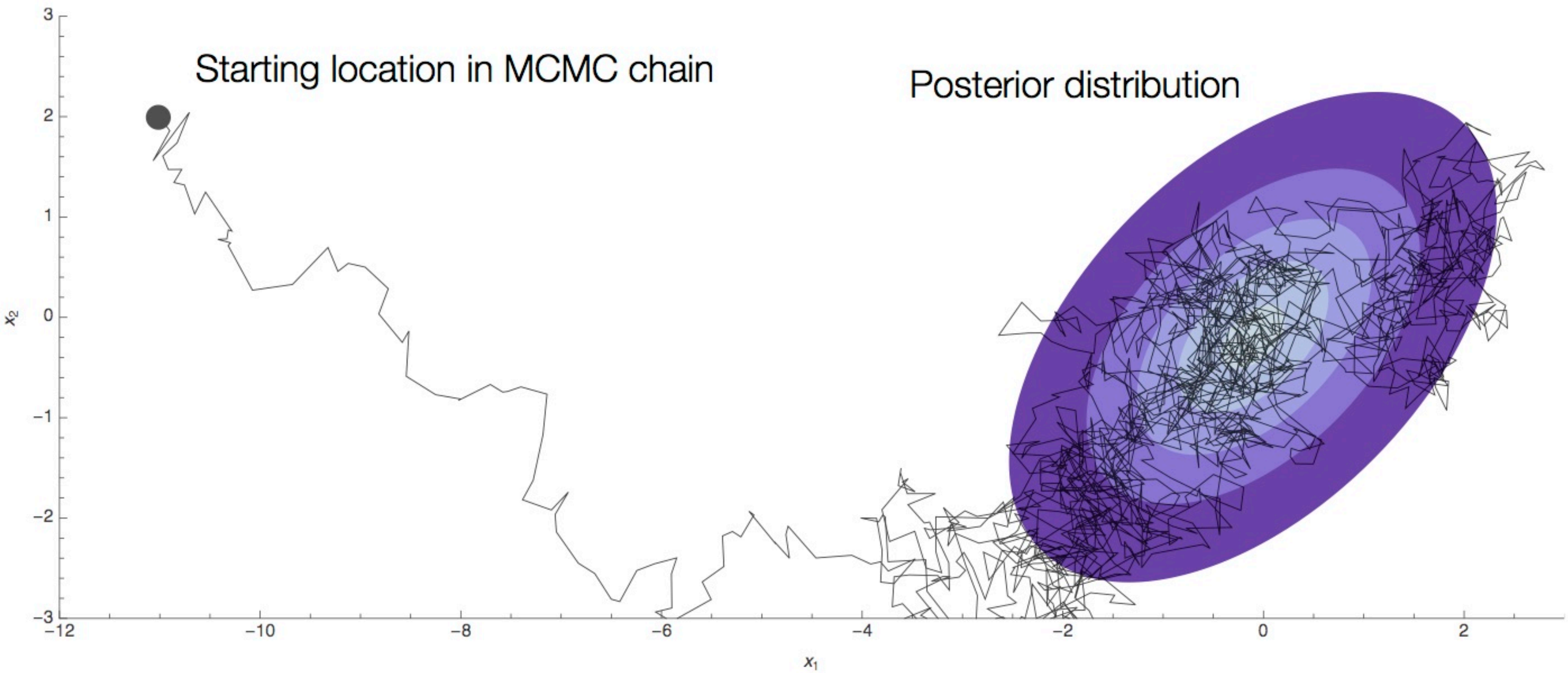
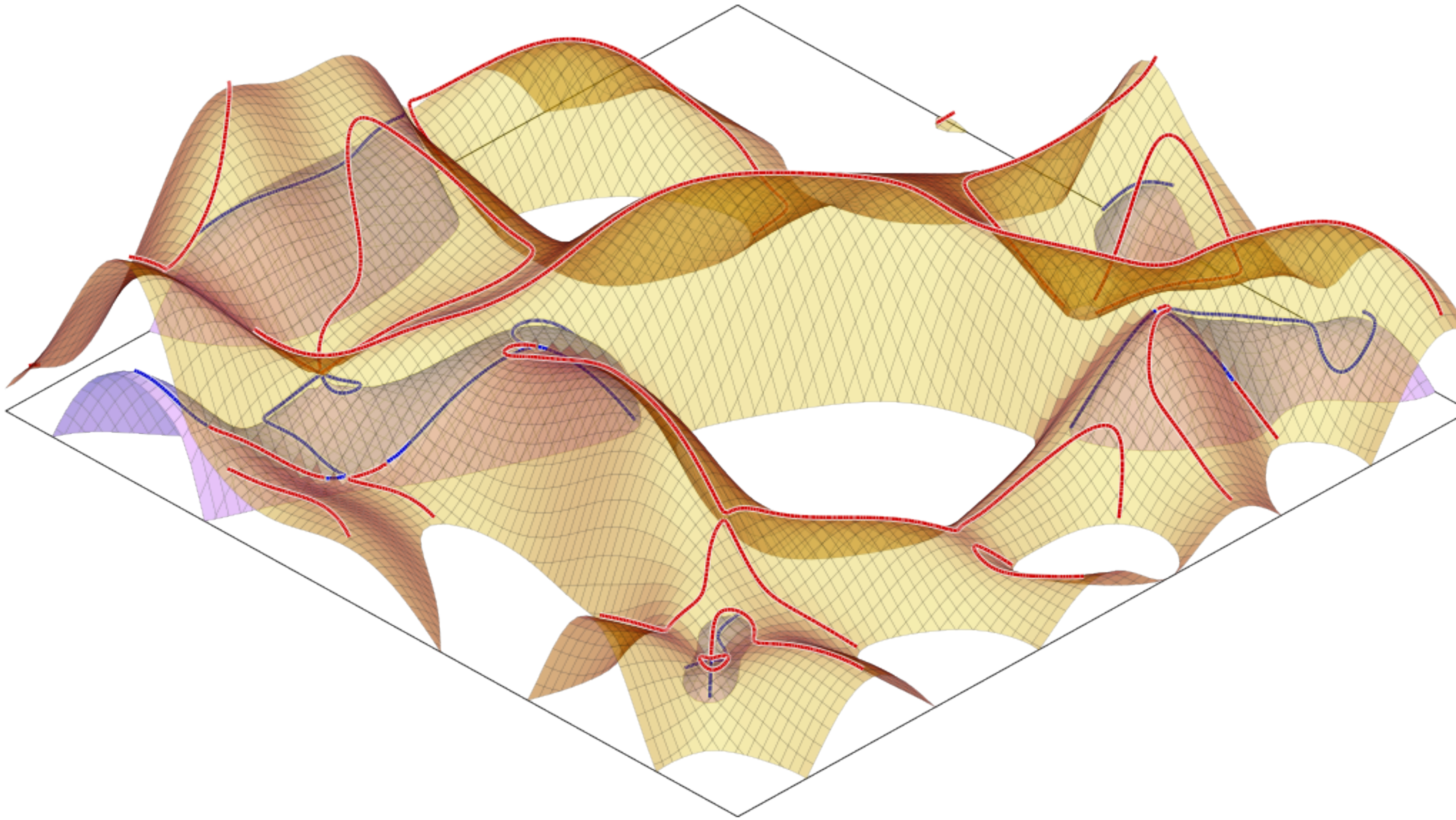
Markov Chain Monte Carlo

Methods, Algorithms, and Applications



BEAST: Bayesian Evolutionary Analysis by Sampling Trees

Black magic (noun): code based on techniques that appear to work but which lack a theoretical explanation



$$F(T) = \left(\frac{m}{2\pi i\hbar\varepsilon}\right)^{N/2} \int_{-\infty}^{\infty} \left(\prod_{l=1}^{N-1} d\chi_l\right) \exp\left\{\frac{im}{2\hbar\varepsilon}\chi_j M_{jk}\chi_k\right\}$$

Applications of MCMC

- High-dimensional integrals/ probability distributions
- Computing large hierarchical models
- Rare event sampling
- Marginalization/ Parameter estimation
- Model comparison

Top 10 Most Influential Algorithm of the 20th Century (CiSE)

Markov Chain

$$p(\theta_{i+1} \mid \{\theta_i\}) = p(\theta_{i+1} \mid \theta_i)$$

"memoryless"

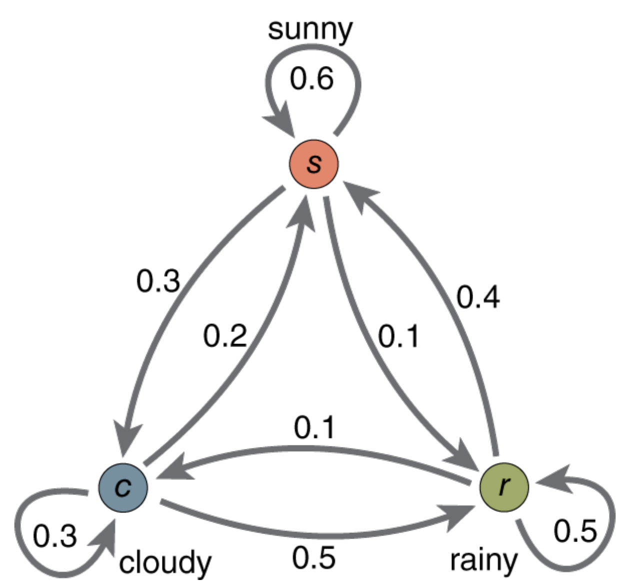
$$p(\theta_{i+1} \mid \theta_i) = p(\theta_i \mid \theta_{i+1})$$

reversible

$$p(\theta_{i+1}) = \int T(\theta_{i+1} \mid \theta_i) p(\theta_i) \, d\theta_i$$

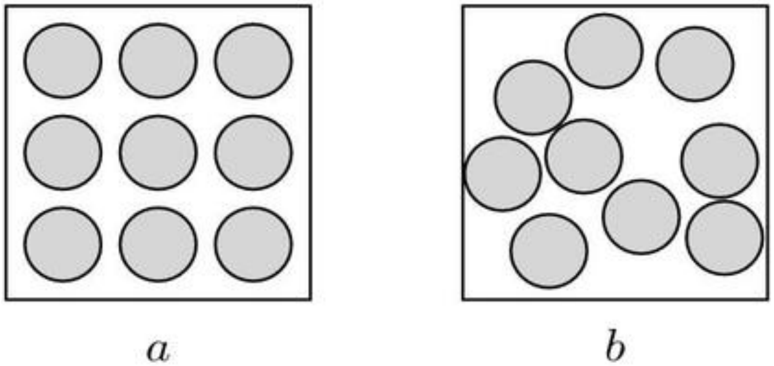
$$T(\theta_{i+1} \mid \theta_i) p(\theta_i) = T(\theta_i \mid \theta_{i+1}) p(\theta_{i+1})$$

$$p_{acc}(\theta_i, \theta_{i+1}) = \frac{K(\theta_i \mid \theta_{i+1}) p(\theta_{i+1})}{K(\theta_{i+1} \mid \theta_i) p(\theta_i)}$$



Metropolis-Hastings Algorithm

1. Choose an appropriate initial model, or "state"
2. Compute steps
3. keep or reject for each "item"
4. Calculate quantity of interest for new state
5. Iterate as needed



$$a = \frac{f(x_{i+1})}{f(x_i)}$$

$$f(x) = \frac{1}{N} \sum f(x_i)$$

Metropolis-Hastings Algorithm

ADVANTAGES

- No curse of dimensionality
  - P(rejection) α D
- High-dimensionality
  - Only option

DISADVANTAGES

- Auto-correlation
  - Jumping width
  - Thinning
- Burn-in period
  - Local minima

Gibbs Sampling

- Multi-dimensional sampling → multiple low-dimensional samples

$$X^{(i)} \rightarrow X^{(i+1)}$$

- Conditional distribution

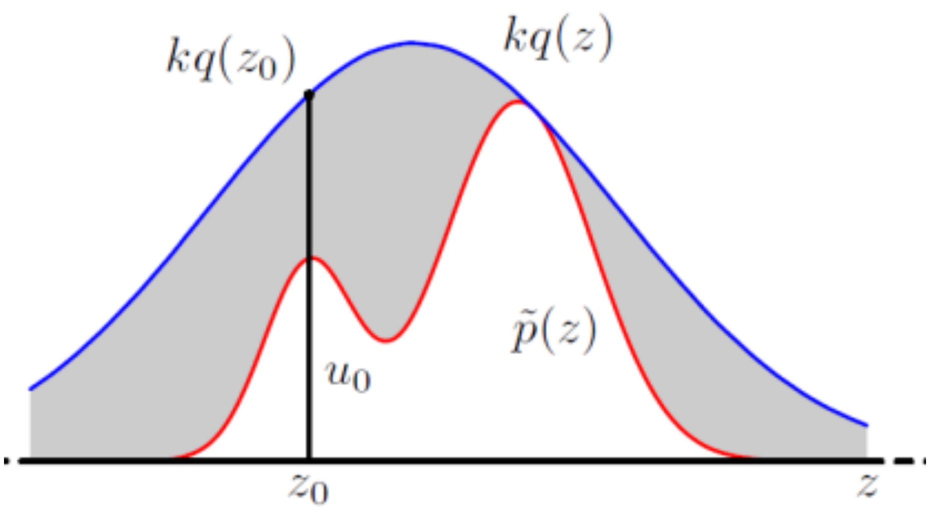
$$p(x_1, x_2, \ldots, x_n)$$

- Advantages over M-H
  - Bayesian networks
  - Reduce autocorrelation
    - Collapsed Gibbs Sampling
    - Blocked Gibbs Sampling
    - Simulated annealing
    - Ordered overrelaxation

$$p(x_j^{(i+1)} \mid x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$$

Adaptive Rejection Sampling

- Log-concave density functions
- Piece-wise linear density functions
- Avoid evaluating p(z0)



MCMC with Python

- Emcee
- PyMC
  - Bayesian statistical models and fitting algorithms
- PyStan
  - Gradient-based MCMC algorithms for Bayesian inference
  - No-U-Turn Sampling (NUTS), Hamiltonian MC

	Complexity	Execution time (100,000 samples; includes burn-in)	Ease of Installation	Learning Curve/Ease of Use	Number of Features
emcee	Very lightweight	~6 sec	Pure python; installs easily with pip	Straightforward & Pythonic	not much built-in beyond basic MCMC sampling
pymc2	Lots of functionality & options	~17 sec	Requires fortran compiler; binaries available via conda	Pythonic, but lots of pymc-specific boilerplate	Lots of built-in functionality in Python
pystan	Large package; requires coding in Stan language	~20 sec compilation + ~6 sec computation	Requires C compiler + Stan installation; no binaries available	Not pure Python; must learn Stan model specification language	Lots of built-in functionality in Stan-specific language

Jake VanderPlas - Frequentism and Bayesianism: How to be Bayesian in Python

Emcee (The MCMC Hammer)

- Affine Invariant MCMC Ensemble Sampler
- Bayesian parameter estimation

```
import numpy as np
import emcee

def lnprob(x, ivar):
    return -0.5 * np.sum(ivar * x ** 2)

ndim, nwalkers = 10, 100
ivar = 1. / np.random.rand(ndim)
p0 = [np.random.rand(ndim) for i in range(nwalkers)]

sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args=[ivar])
sampler.run_mcmc(p0, 1000)
```

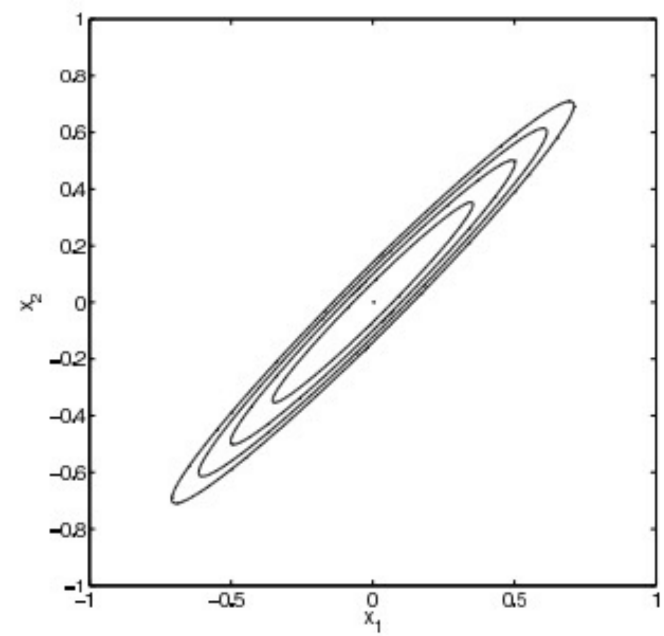
Affine Invariance

$$P(x) = \lambda^{-n} P(\lambda x)$$

$$x \longrightarrow y = ax + b$$

$$P(x) \longrightarrow P(ax+b) \propto P(x)$$

$$P(x) \propto \exp\left(\frac{-(x_1 - x_2)^2}{2\varepsilon} - \frac{(x_1 + x_2)^2}{2}\right)$$



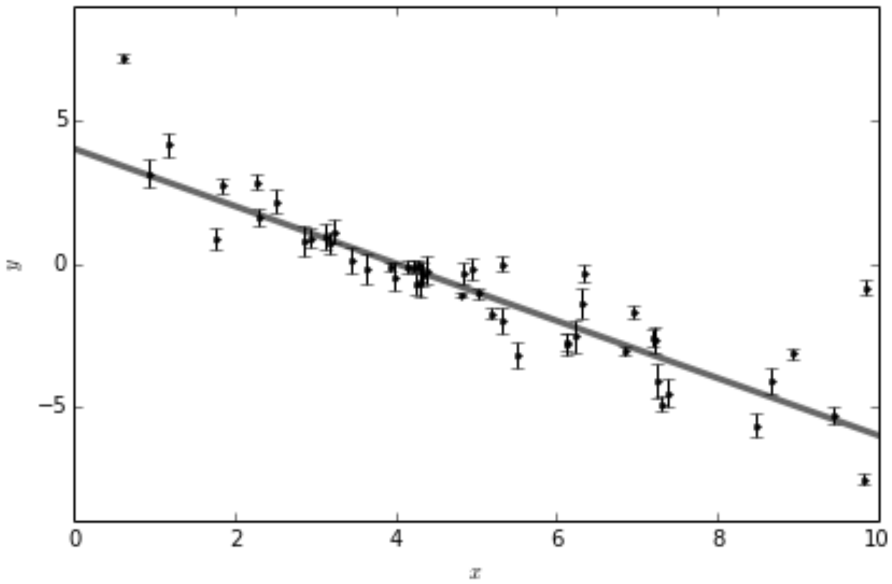
$$P(x) \propto \exp\left(\frac{-(y_1^2 + y_2^2)}{2}\right)$$

$$y_1 = \frac{x_1 - x_2}{\sqrt{\varepsilon}}, y_2 = x_1 + x_2$$

Example - Fitting a Line

```
import emcee
import corner
import numpy as np
import scipy.optimize as op
import matplotlib.pyplot as pl
from matplotlib.ticker import MaxNLocator

# Set "true" parameters
m_true = -1.0
b_true = 4.0
f_true = 0.50
```

Example – Fitting a Line




```
# Generate sample data from our "model"
np.random.seed(123)
N = 50
x = np.sort(10*np.random.rand(N))
yerr = 0.1+0.5*np.random.rand(N)
y = m_true*x+b_true
y += np.abs(f_true*y) * np.random.randn(N)
y += yerr * np.random.randn(N)
```

Example – Fitting a Line

```python
# Define the probability function as likelihood * prior.
def lnprior(theta):
    m, b, lnf = theta
    if -5.0 < m < 0.5 and 0.0 < b < 10.0 and -10.0 < lnf < 1.0:
        return 0.0
    return -np.inf

def lnlike(theta, x, y, yerr):
    m, b, lnf = theta
    model = m * x + b
    inv_sigma2 = 1.0/(yerr**2 + model**2*np.exp(2*lnf))
    return -0.5*(np.sum((y-model)**2*inv_sigma2 - np.log(inv_sigma2)))

def lnprob(theta, x, y, yerr):
    lp = lnprior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return lp + lnlike(theta, x, y, yerr)
```

LSR and MLE Results

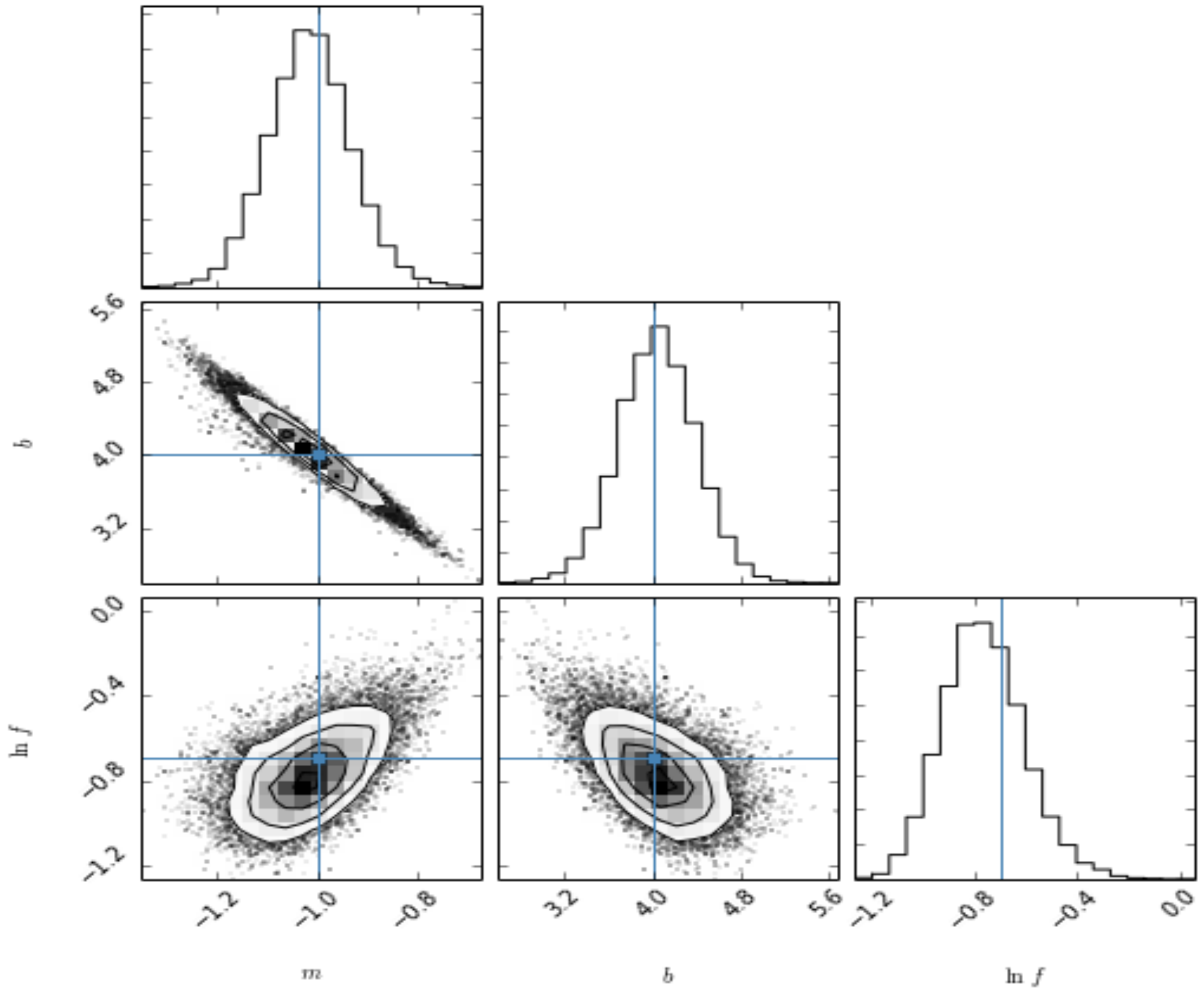
	Least Squares Regression	Maximum Likelihood Estimation	True Values
Slope	-1.0907 ± 0.0162	-1.0115	-1.0
Intercept	4.8155 ± 0.0909	4.0283	4.0

MCMC Results

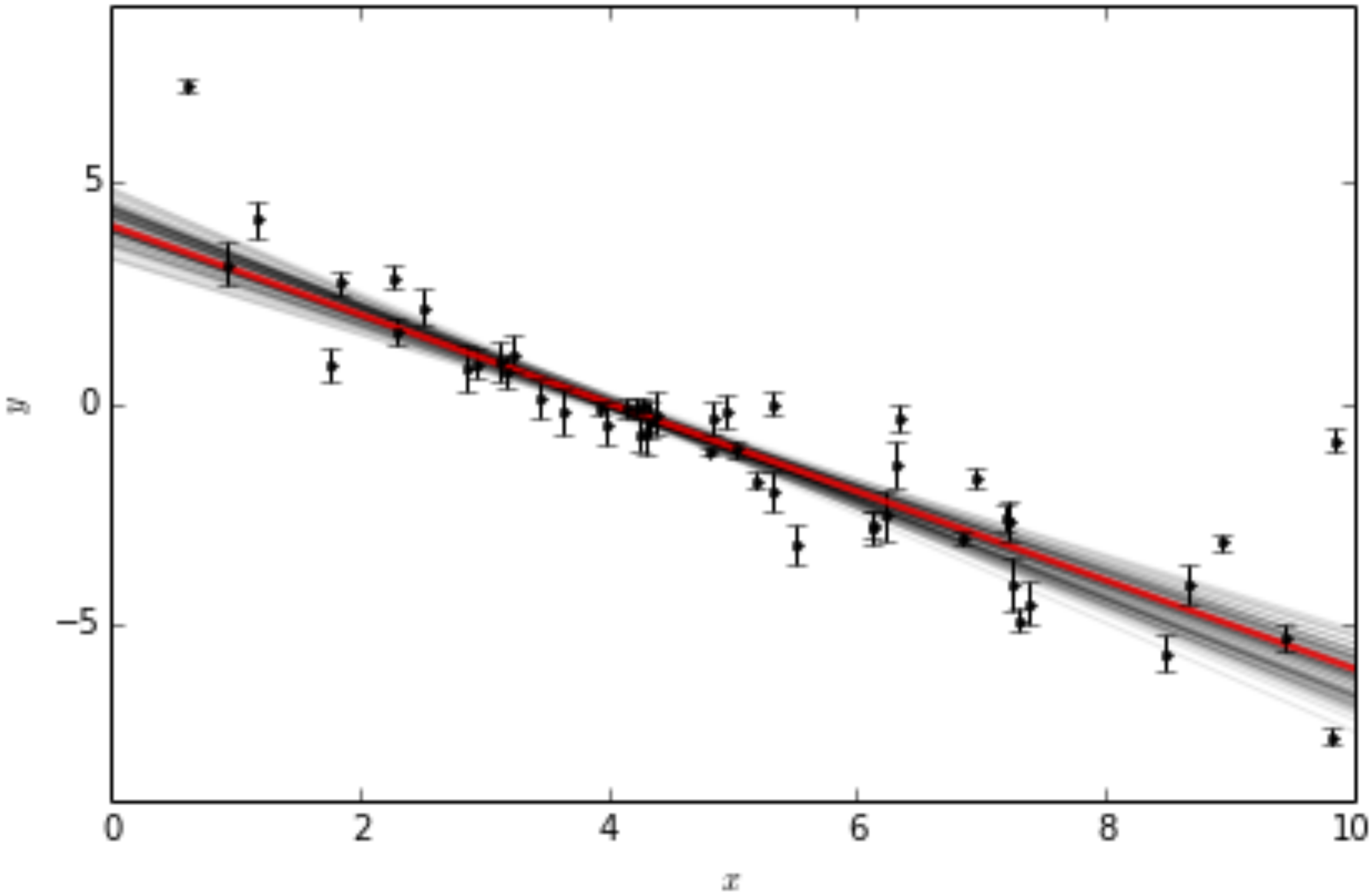
```
# Set up ensemble sampler and run MCMC
n_dim, n_walkers = 3, 100
p0 = result["x"] + np.random.normal(size=3)
pos = [p0 + 1e-4*np.random.randn(n_dim) for i in range(n_walkers)]

sampler = emcee.EnsembleSampler(n_walkers, n_dim, lnprob, args=(x, y, yerr))
sampler.run_mcmc(pos, 500, rstate0=np.random.get_state())
```

Slope	Intercept
$-1.0182^{+0.0819}_{-0.0814}$	$4.0539^{+0.0828}_{-0.0647}$

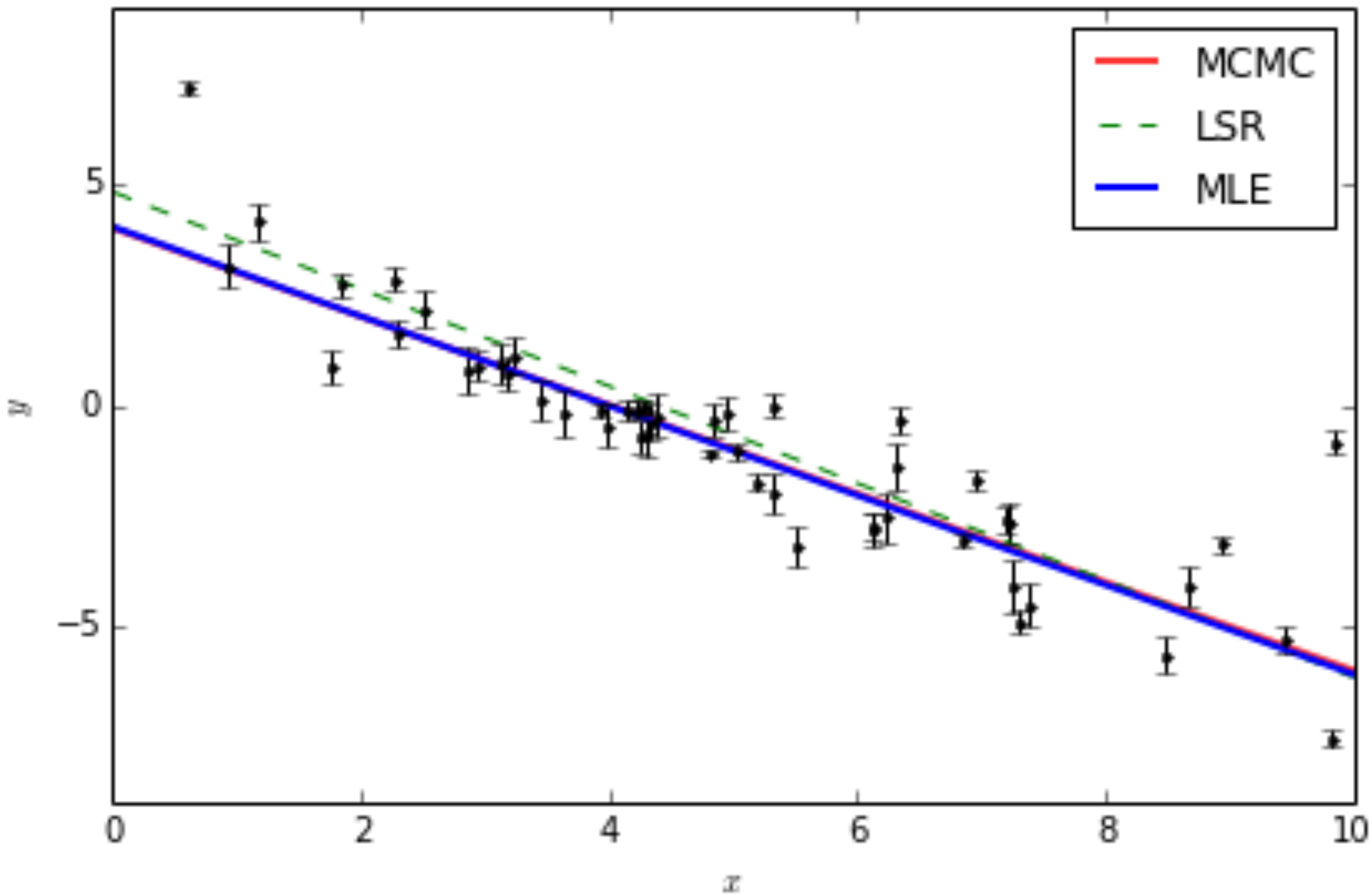
MCMC Results



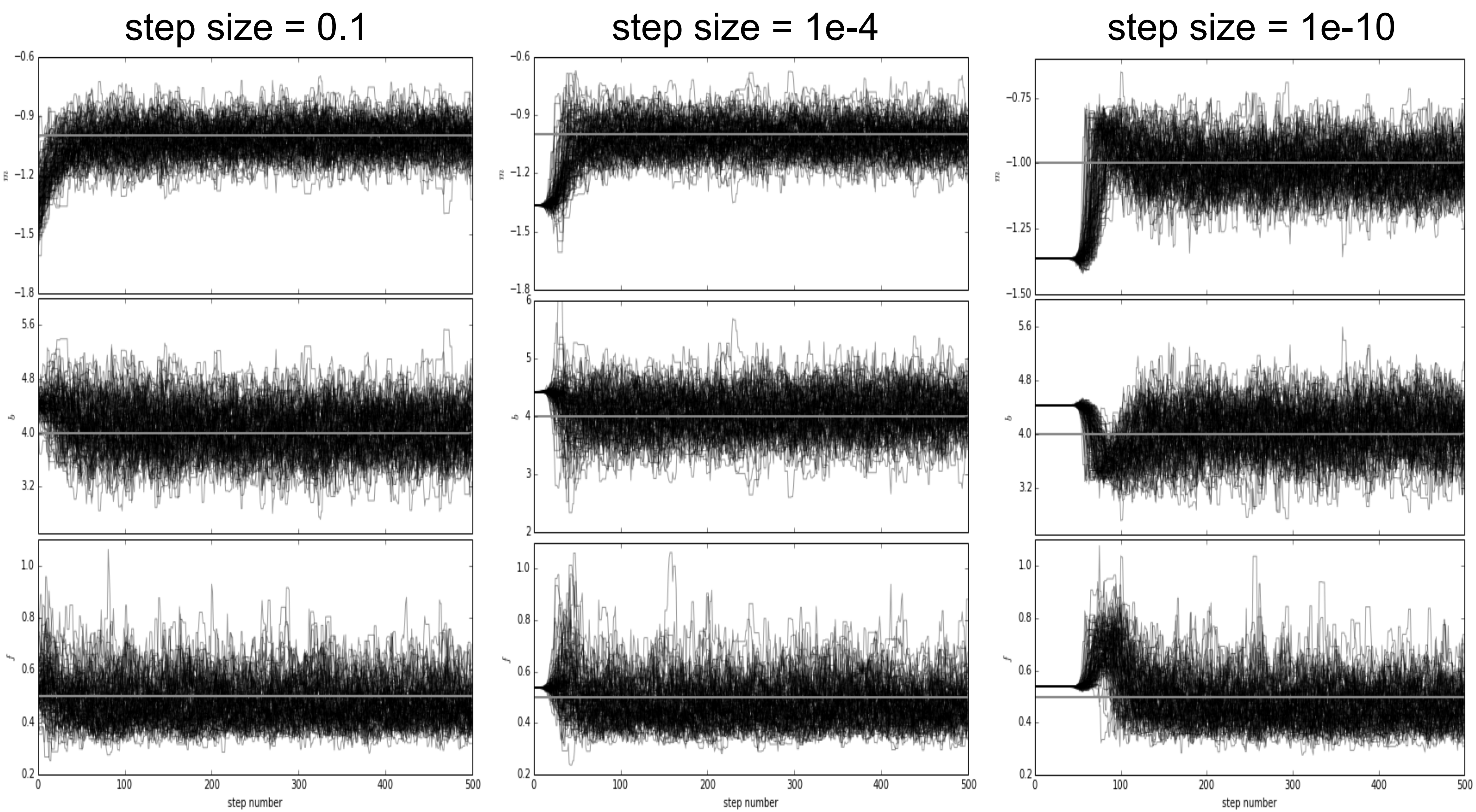
MCMC Results



MCMC Results



MCMC Results



References

1. Ivezic Section 5.8
2. https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo
3. Goodman and Weare, "Ensemble Samplers with Affine Invariance", http://msp.org/camcos/2010/5-1/camcos-v5-n1-p04-s.pdf
4. Emcee, http://dan.iel.fm/emcee/current/