Lecture 15

Bayesian Regression

and

pymc3

Last time: Bayes

- Gibbs Sampling samples from conditionals
- Hierarchical models have a graph structure
- Makes conditional sampling easy
- Best to use log posteriors

Today

- recap
- bayesian regression and updating
- regularization and the ridge
- from the normal model to regression using pymc
- posterior vs predictive in regression problems

Law of Large numbers

Let $x_1, x_2, \ldots, x_n$ be a sequence of IID values from random variable X, which has finite mean μ. Let:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

Then:

$$S_n \to \mu \, as \, n \to \infty.$$

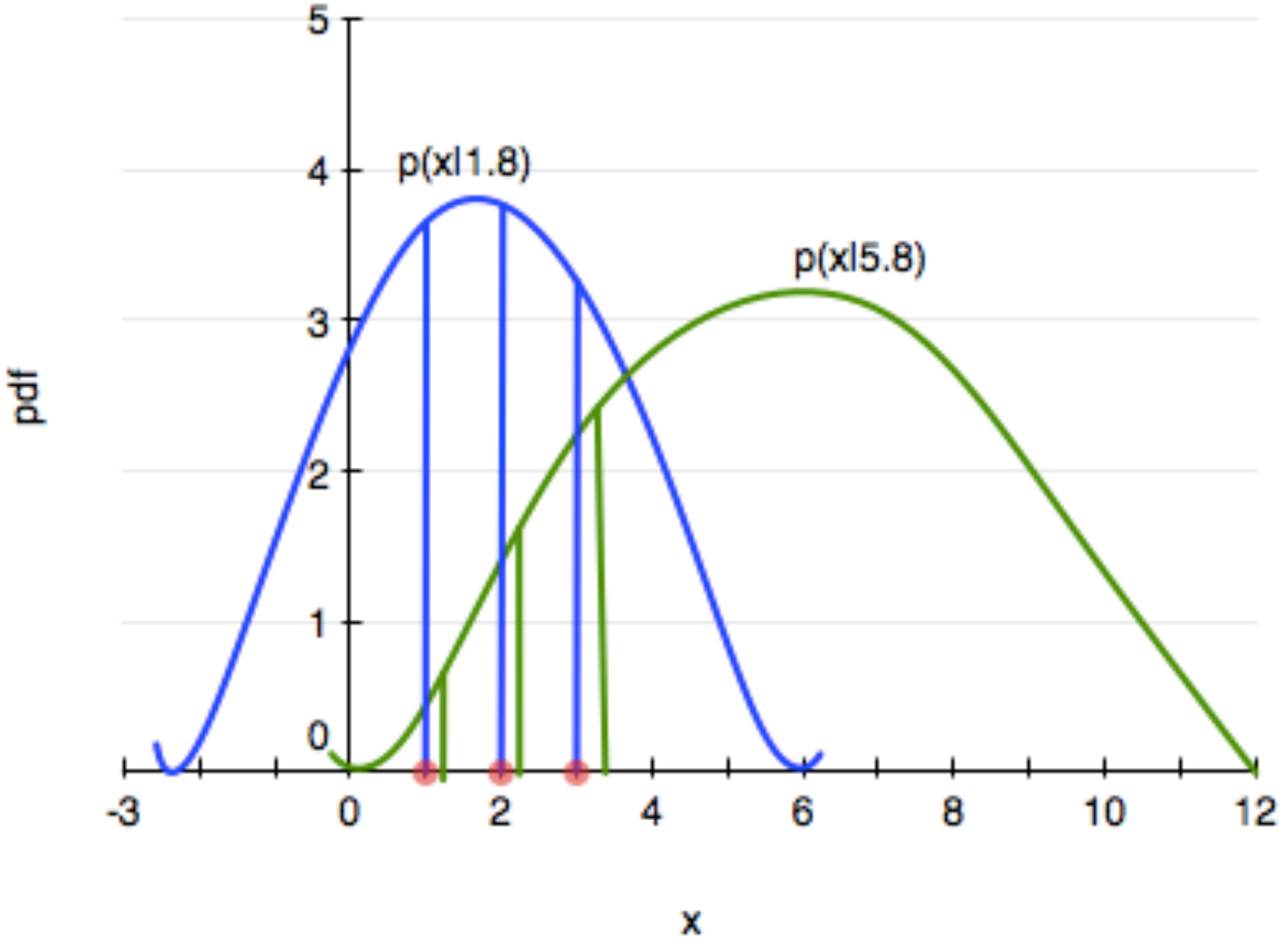
Law of Large numbers (LLN)

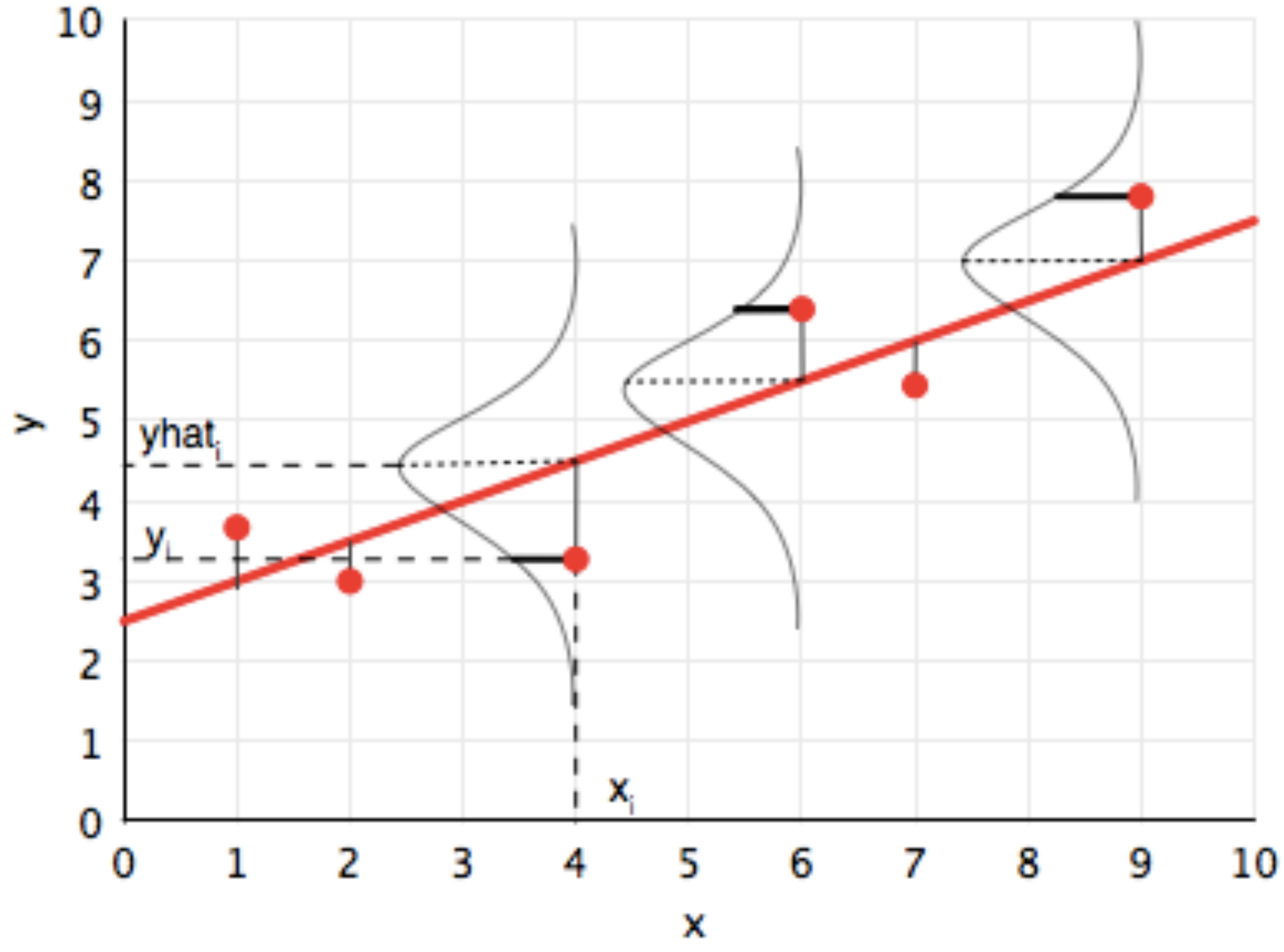
- Expectations become sample averages. Convergence for large N.

$$E_f[g] = \int g(x)dF = \int g(x)f(x)dx$$
$$= \lim_{n \to \infty} \frac{1}{N} \sum_{x_i \sim f} g(x_i)$$

- allows for monte-carlo

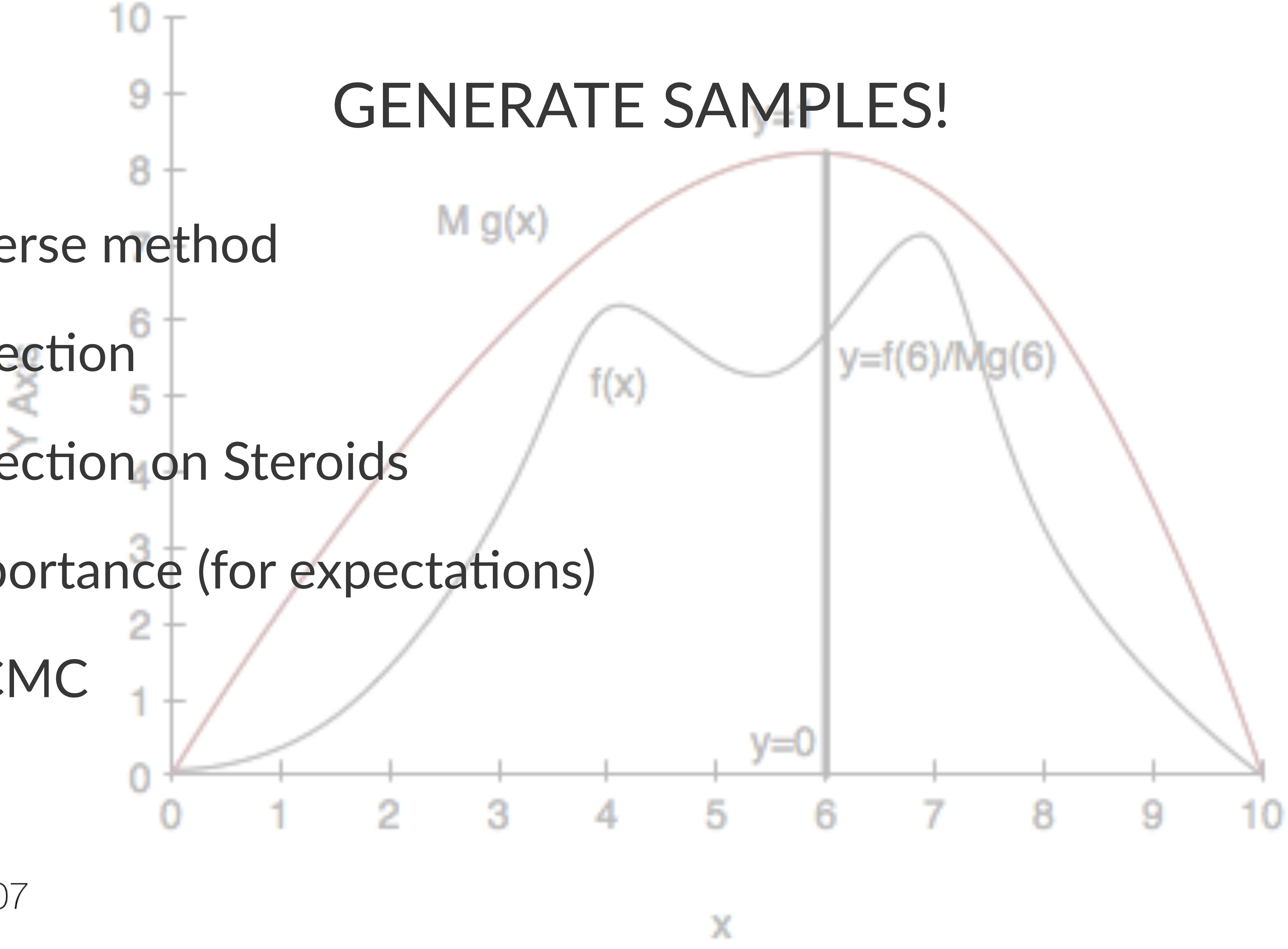
MLE



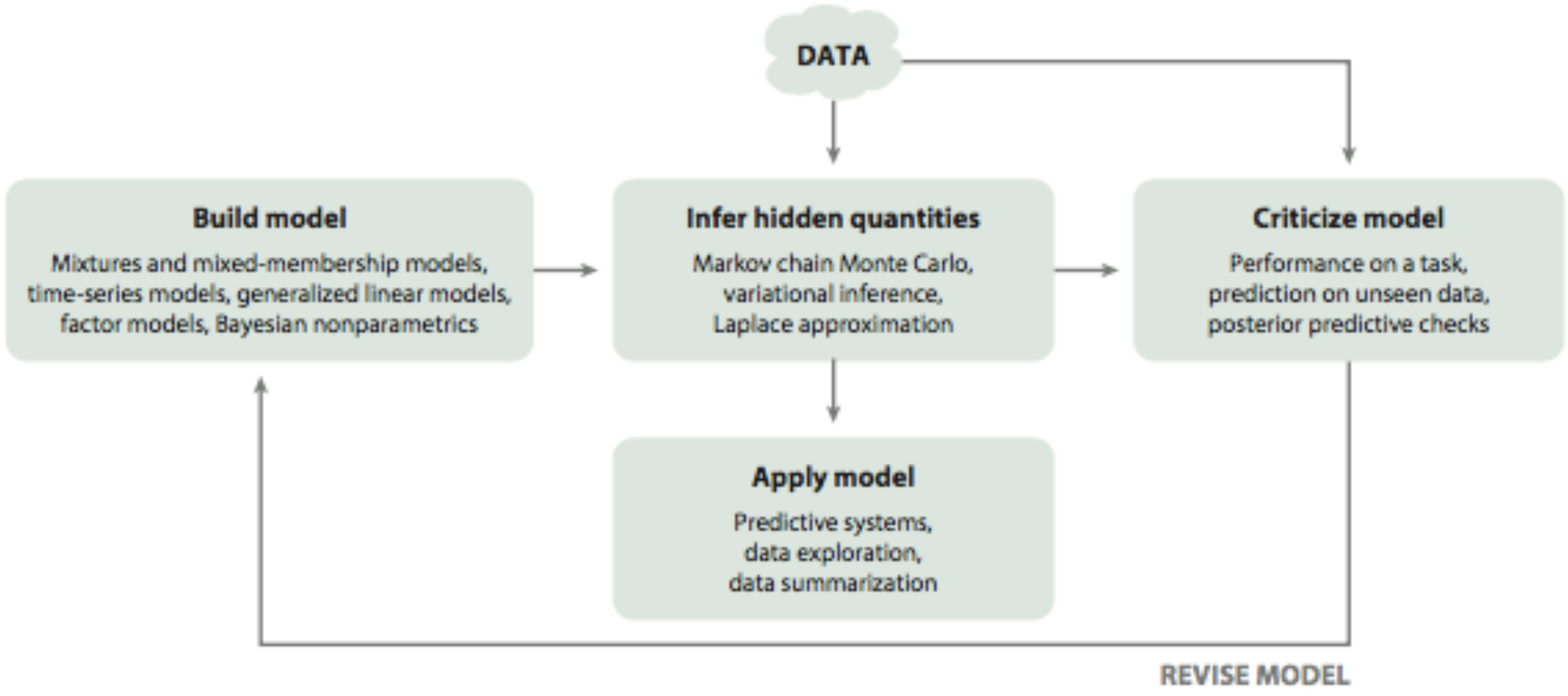


GENERATE SAMPLES!

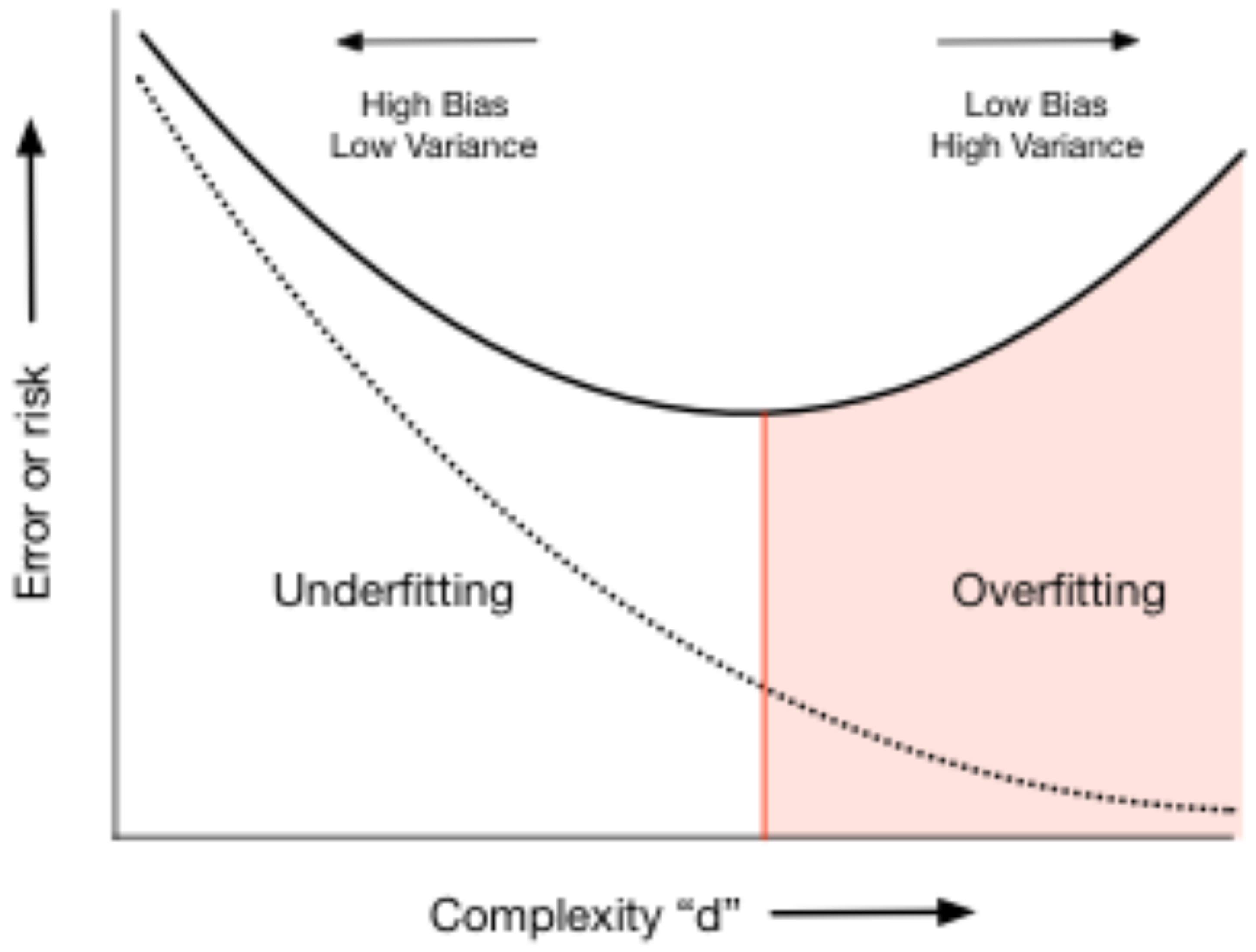
- Inverse method
- Rejection
- Rejection on Steroids
- Importance (for expectations)
- MCMC



Box's loop

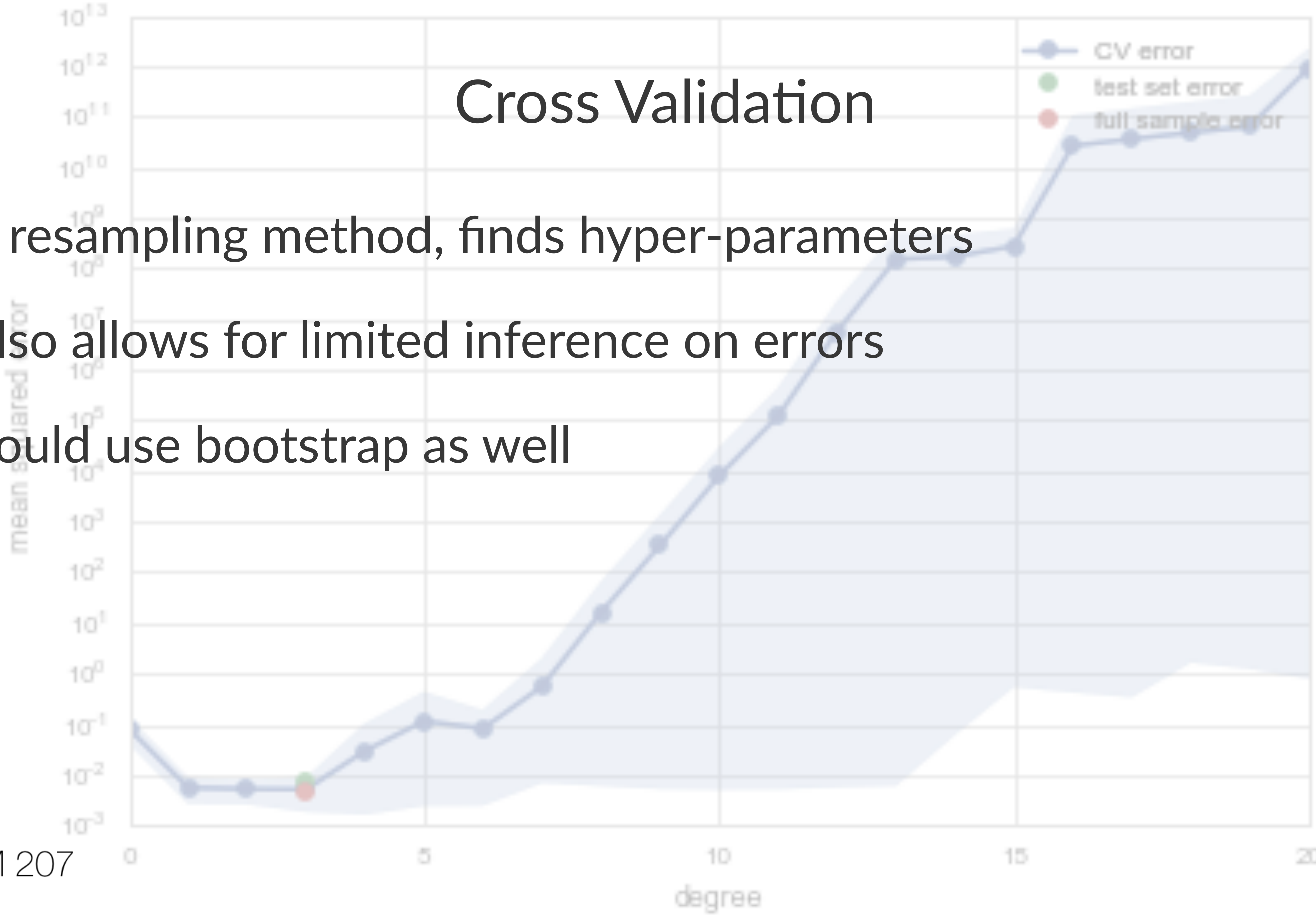

DATA
Build model
Mixtures and mixed-membership models,
time-series models, generalized linear models,
factor models, Bayesian nonparametrics
Infer hidden quantities
Markov chain Monte Carlo,
variational inference,
Laplace approximation
Criticize model
Performance on a task,
prediction on unseen data,
posterior predictive checks
Apply model
Predictive systems,
data exploration,
data summarization
REVISE MODEL

Complexity, Risk, Bias, and Variance

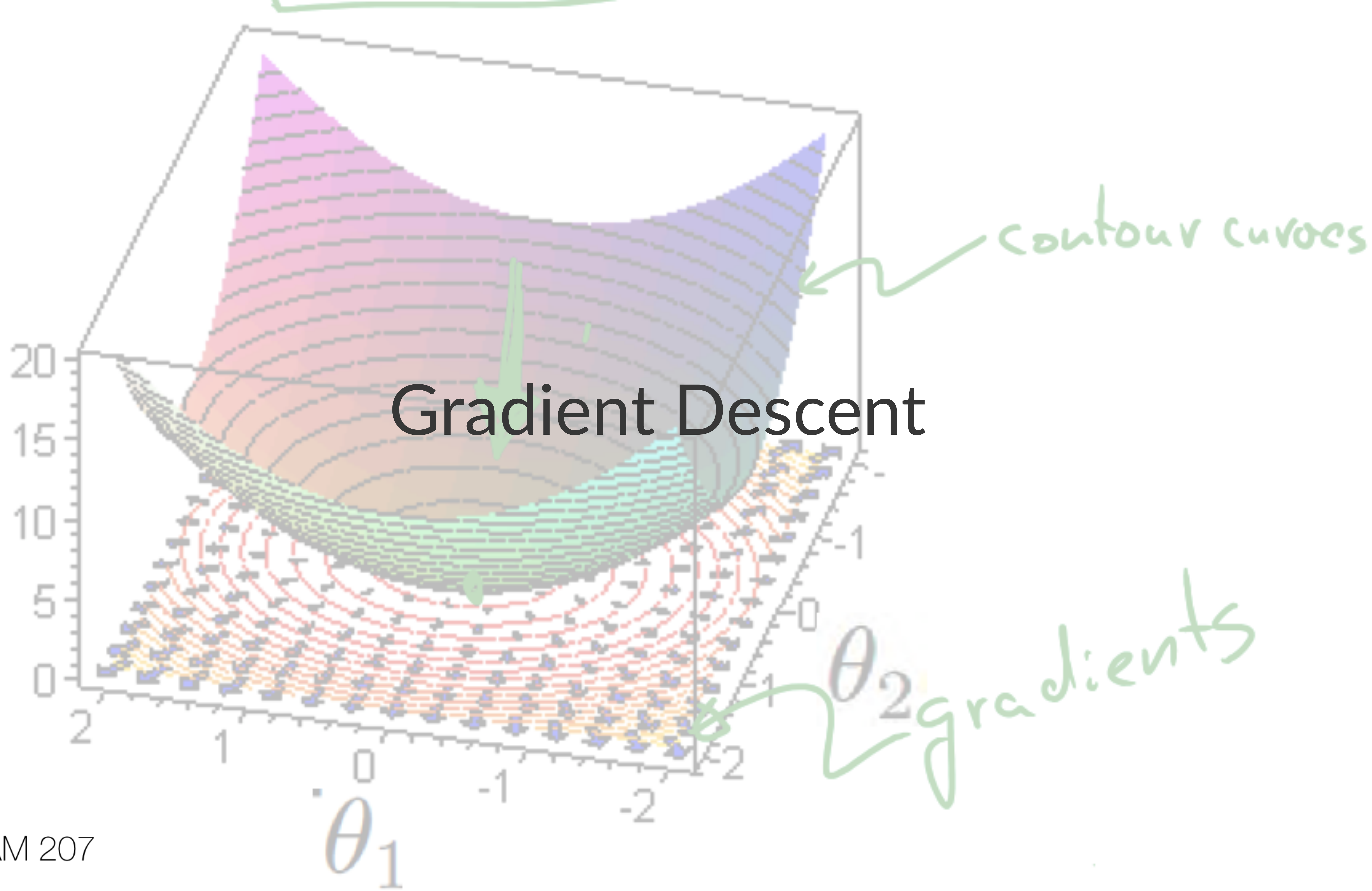


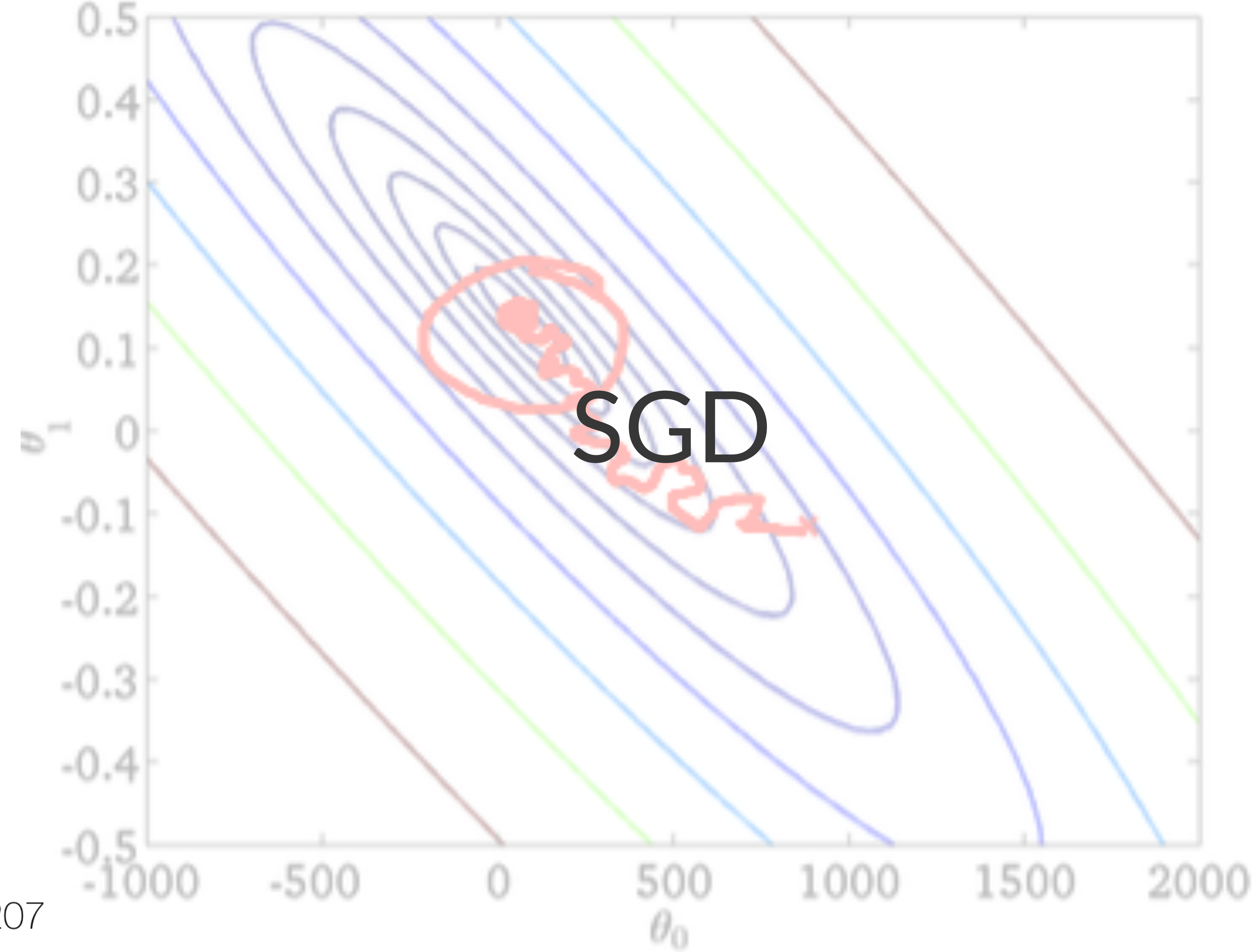
Cross Validation

- a resampling method, finds hyper-parameters
- also allows for limited inference on errors
- could use bootstrap as well



Gradient Descent




0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
θ_1
-1000
-500
0
500
1000
1500
2000
θ_0

SGD

Simulated Annealing

Minimize f by identifying with the energy of an imaginary physical system undergoing an annealing process.

Move from x_i to x_j via a **proposal**.

If the new state has lower energy, accept x_j.

If the new state has higher energy, accept with $A = \exp\left(-\Delta f / kT\right)$

Lowering temperature slowly, the system reaches "thermal equilibrium" at each temperature. Boltzmann's distribution:

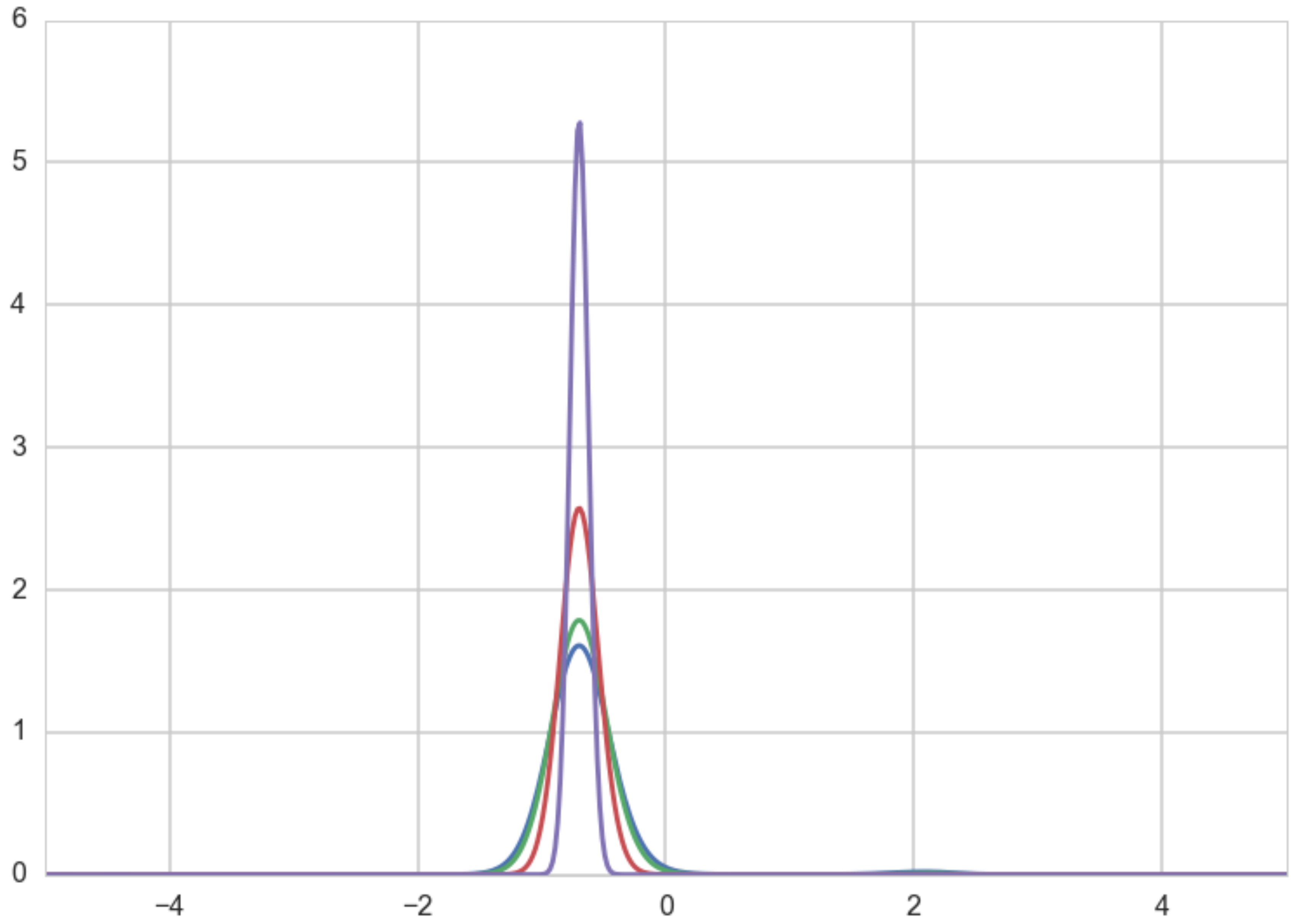
$$p(X = i) = \frac{1}{Z(T)} \exp\left(\frac{-E_i}{kT}\right)$$

If you identify

$$p_T(x) = e^{-f(x)/T} \text{ and } p(x) = e^{-f(x)}$$

Then:

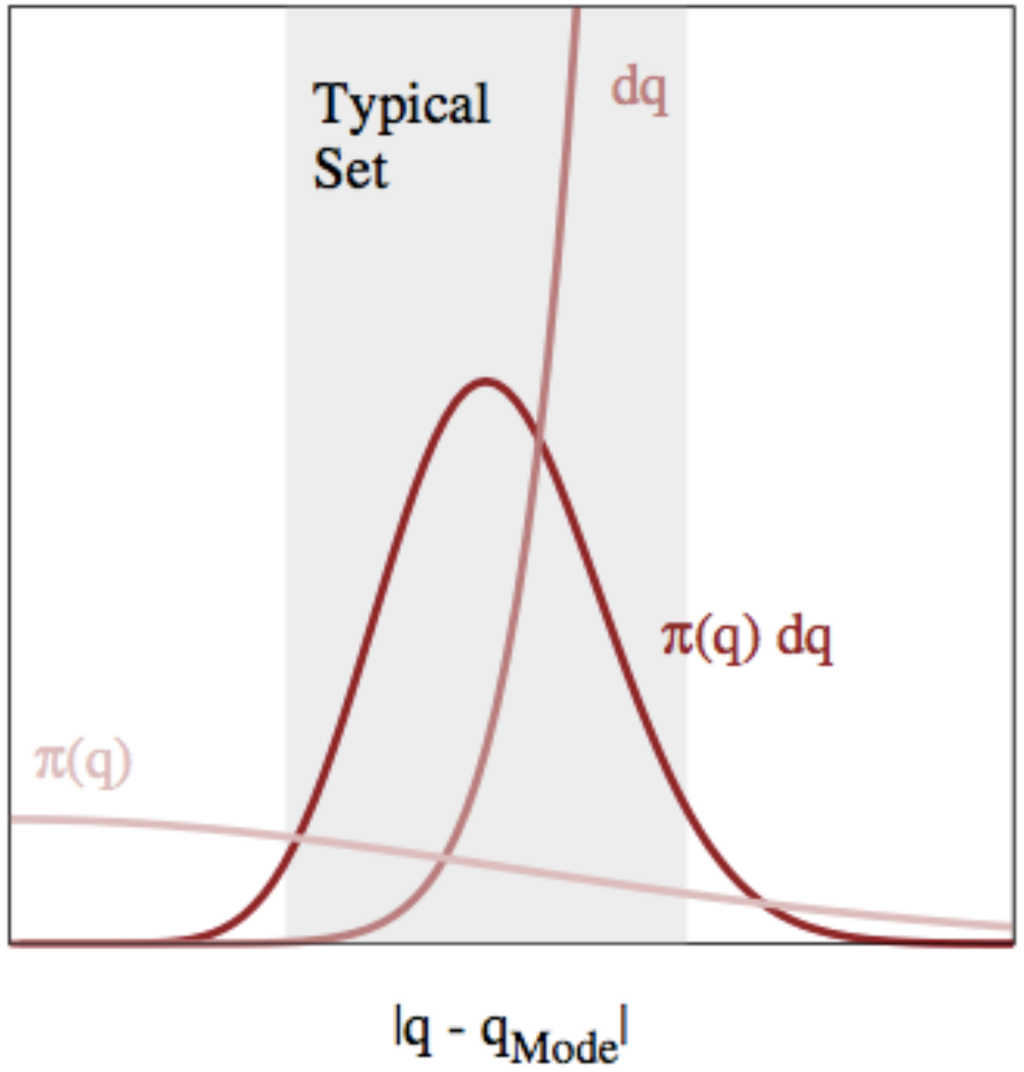
$$P_T(x) = P(x)^{1/T}$$



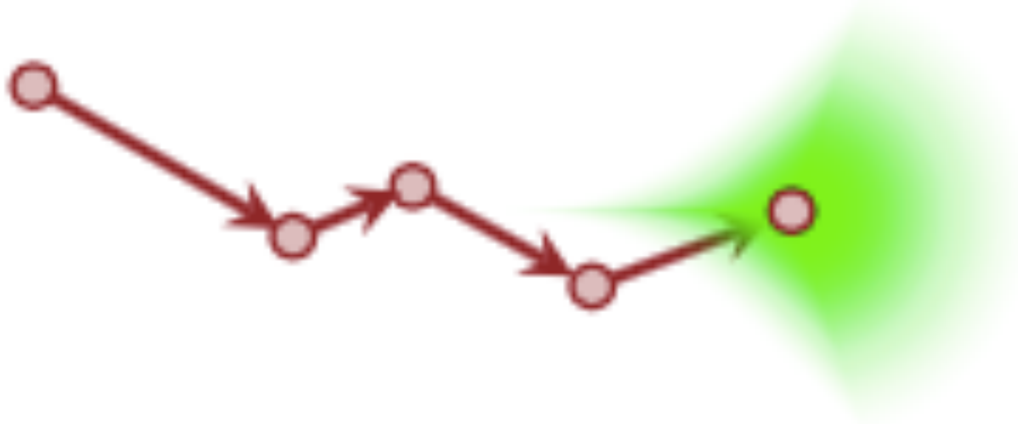
Sampling a Distribution

- Turn the question on its head.
- Suppose we wanted to sample from a distribution $p(x)$ (corresponding to a minimization of energy $-log(p(x))$).
- keep our symmetric proposal (reversibility!). Need irreducibility to sample from full distribution
- set T=1, and use our simulated annealing method: Metropolis

Critical: explore the typical set



Metropolis and MH



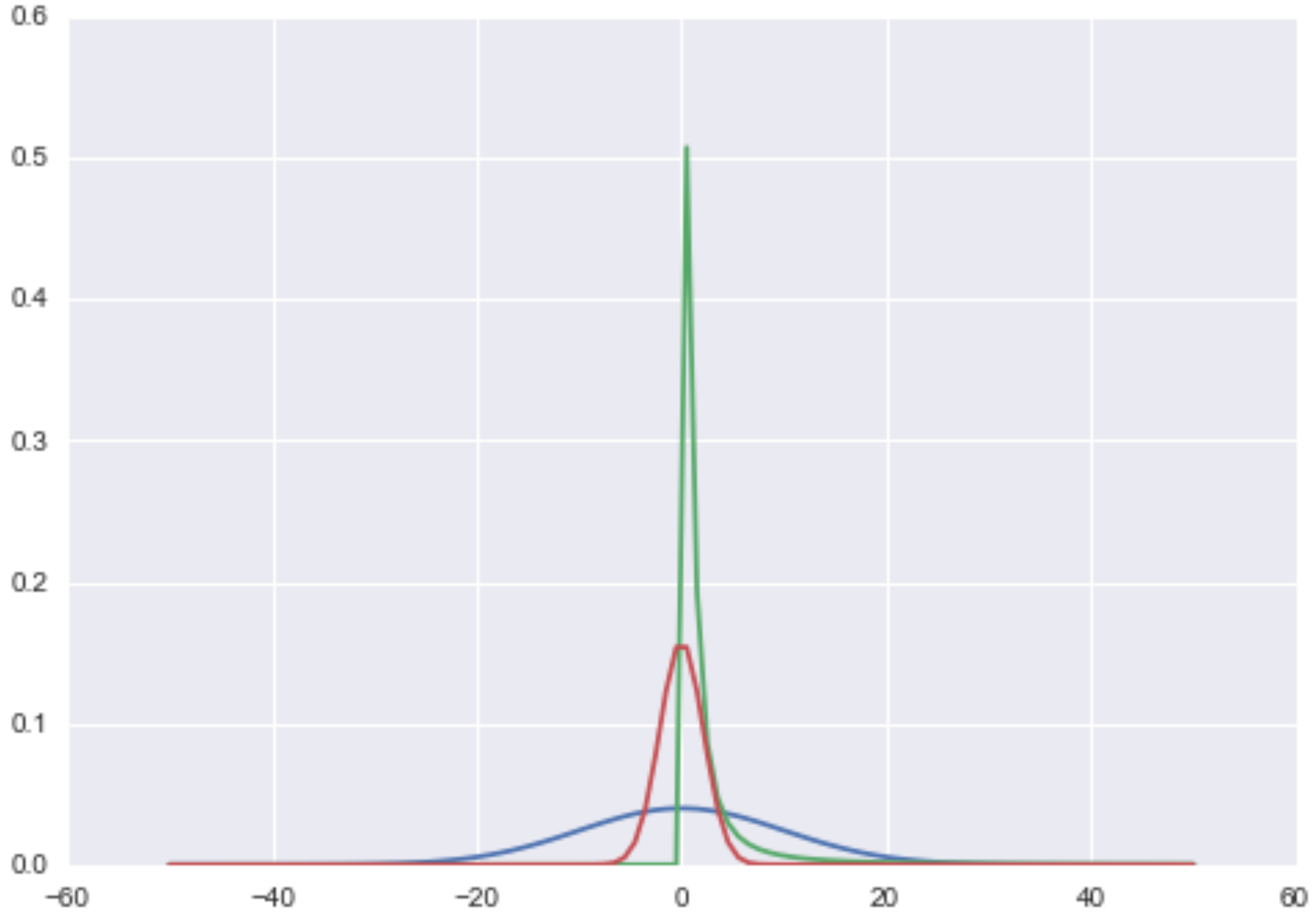
Bayesian

- sample is the data fixed
- parameter is stochastic, has prior and posterior distribution
- posterior: $p(\theta|y) = \frac{p(y|\theta)\,p(\theta)}{p(y)}$, can summarize via MAP
- just bayes rule: $posterior = \frac{likelihood \times prior}{evidence}$

- prior-predictive = evidence: $p(y) = E_{p(\theta)}[\mathcal{L}] = \int d\theta p(y|\theta)p(\theta)$ a normalization, irrelevant for sampling, useful for EB
- What if θ is multidimensional? Marginal posterior: $p(\theta_1|D) = \int d\theta_{-1} p(\theta|D)$.
- posterior predictive: the distribution of a future data point y^*:

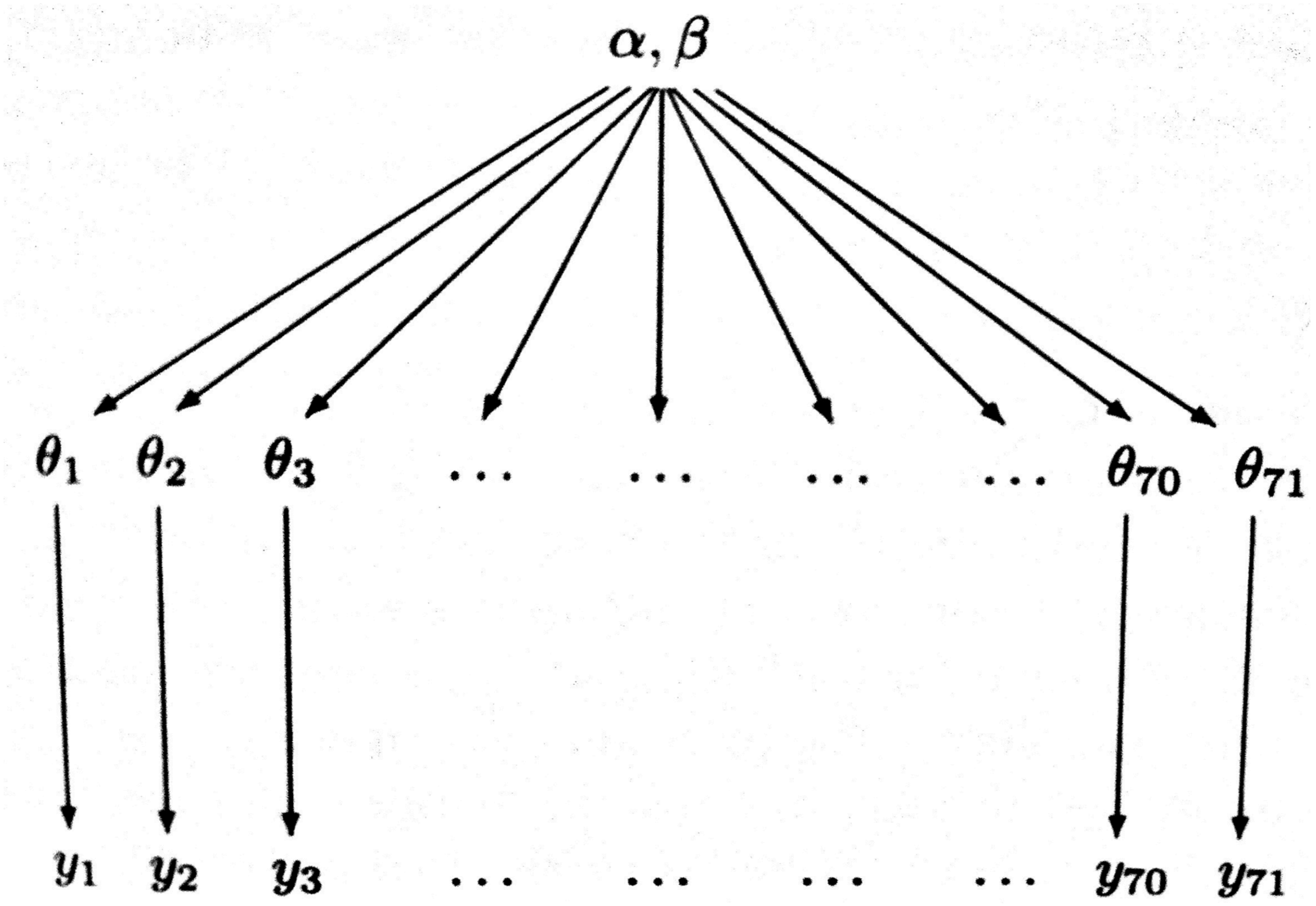
$$p(y^*|D = \{y\}) = E_{p(\theta|D)}[p(y|\theta)] = \int d\theta p(y^*|\theta)p(\theta|\{y\}).$$

priors



- choose likelihoods with MAXENT
- choose priors as non-informative, e.g. uniform or Jeffreys
- better still: choose priors as weakly informative/regularizing
- helps with sampler performance

Hierarchical models



Key Idea: Share statistical strength

- Some **units** (experiments) statistically more robust
- Non-robust experiments have smaller samples or outlier like behavior
- Borrow strength from all the data as a whole through the estimation of the hyperparameters
- **regularized partial pooling model** in which the "lower" parameters (θs) tied together by "upper level" hyperparameters.

First idea: estimate directly from data

Posterior-predictive distribution, as a function of upper level parameters $\eta = (\alpha, \beta)$.

$$p(y^* \vert D, \eta) = \int d\theta \, p(y^* \vert \theta) \, p(\theta \vert D, \eta)$$

A likelihood with parameters η and simply use maximum-likelihood with respect to η to estimate these η using our "data" y^*

Or match moments...

Full Sampling

- Fix α and β, we have a Gibbs step for all of the θ_is
- For α and β, everything else fixed, use stationary metropolis step, as conditionals not simply sampled distributions
- when we sample for α, we will propose a new value using a normal proposal, while holding all the θs and β constant at the old value. ditto for β.
- OR just specify in pymc and go!

Howto Hierarchical models

- a DAG, with observations at the bottom of a tree, next layer intermediate parameters, upper layers hyper-parameters
- sample conditionals from parents up the tree.
- the y_j were exchangeable since we had no additional information about experimental conditions.
- if specific groups of experiments came from specific laboratories, assume experiments interchangeable if from the same lab.

Bayesian Formulation of Regression

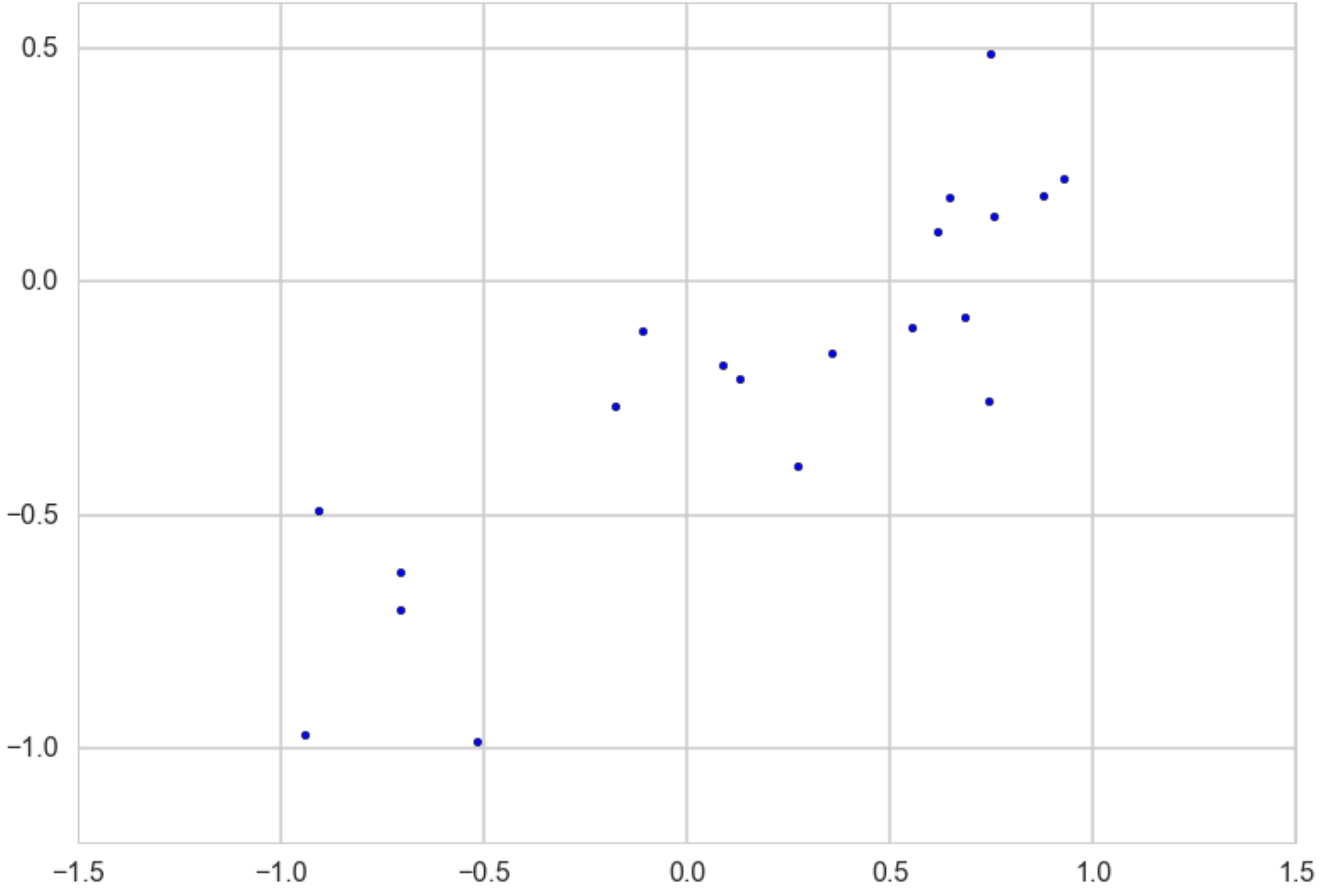
Data
$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

All data points are combined into a $D \times n$ matrix X.

Model:

$$y = \mathbf{x}^{\mathbf{T}}\mathbf{w} + \epsilon$$

$$\epsilon \sim N(0, \sigma_n^2)$$

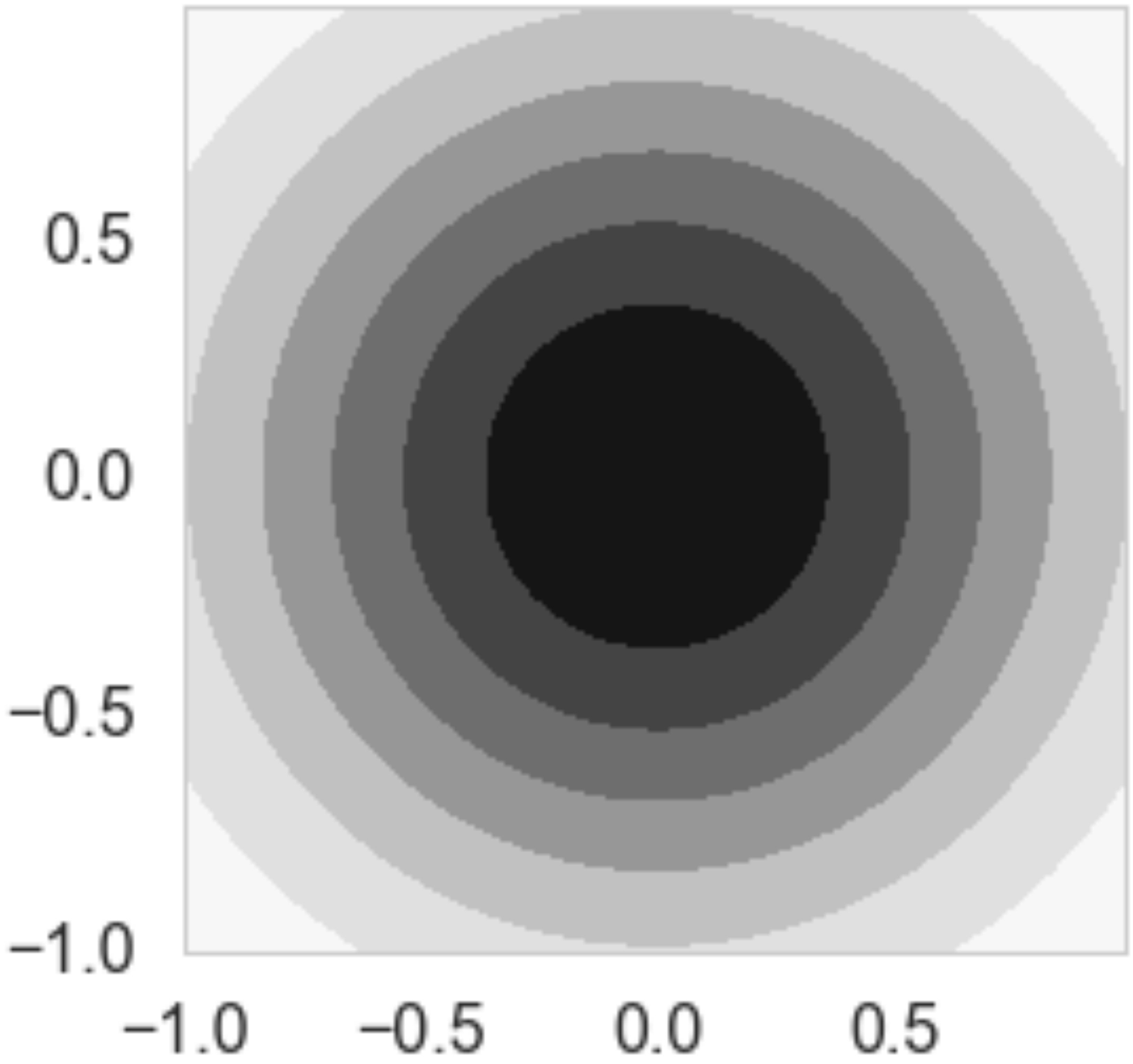


Likelihood

The likelihood is, because we assume independency, the product

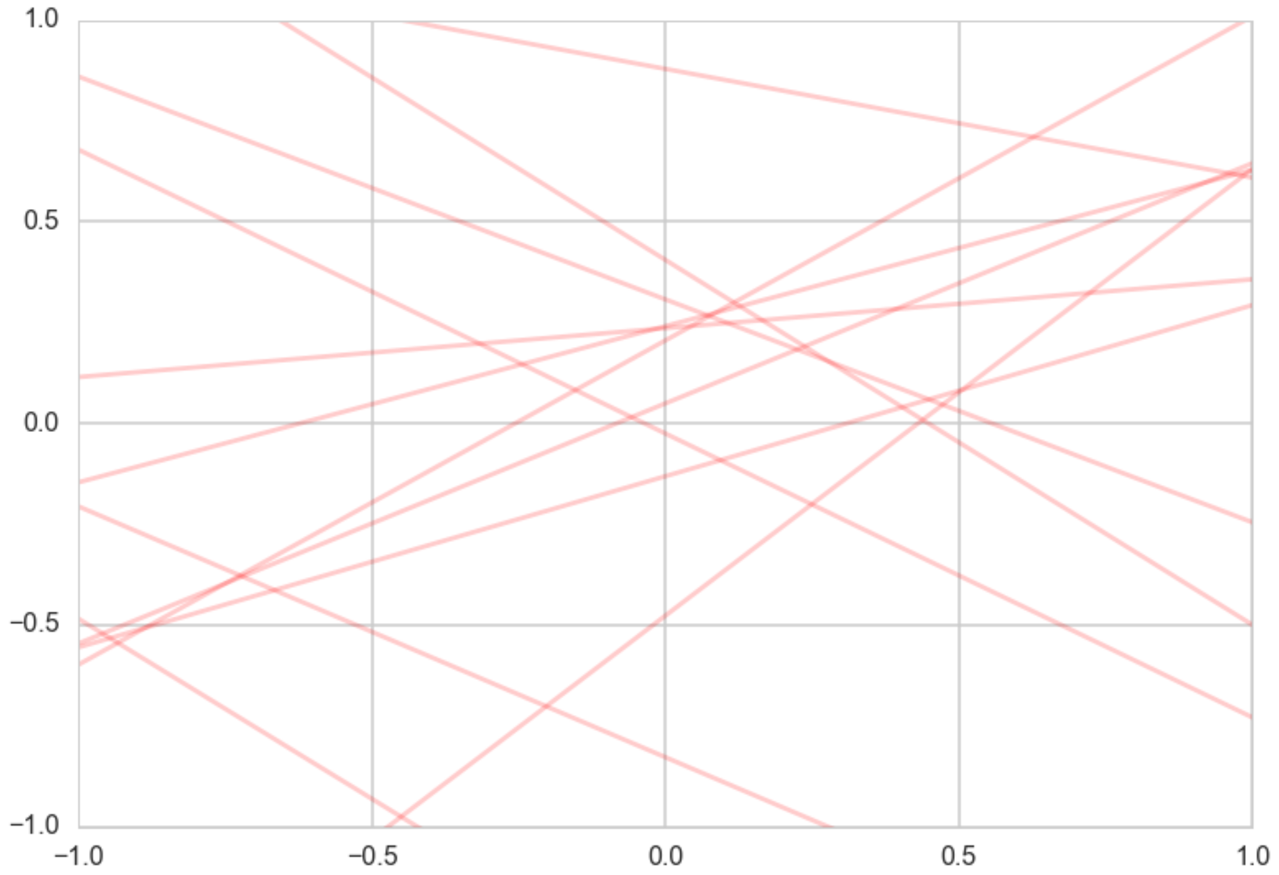
$$\mathcal{L} = p(\mathbf{y}|\mathbf{X},\mathbf{w}) = \prod_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{p}(\mathbf{y_i}|\mathbf{X_i},\mathbf{w}) = \prod_{\mathbf{i=1}}^{\mathbf{n}} \frac{\mathbf{1}}{\sqrt{\mathbf{2}\pi}\sigma_{\mathbf{n}}} \exp\left(-\frac{(\mathbf{y_i}-\mathbf{X_i^T w})^{\mathbf{2}}}{\mathbf{2}\sigma_{\mathbf{n}}^{\mathbf{2}}}\right)$$

$$\propto \exp\left(-\frac{|\mathbf{y}-\mathbf{X^T w}|^{\mathbf{2}}}{2\sigma_n^2}\right) \propto N(X^T\mathbf{w}, \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I})$$



Prior $\mathbf{w} \sim \mathbf{N}(\mathbf{w_0}, \mathbf{\Sigma})$

$$\mathbf{w} \sim \mathbf{N}(\mathbf{w_0}, \tau^2 \mathbf{I})$$



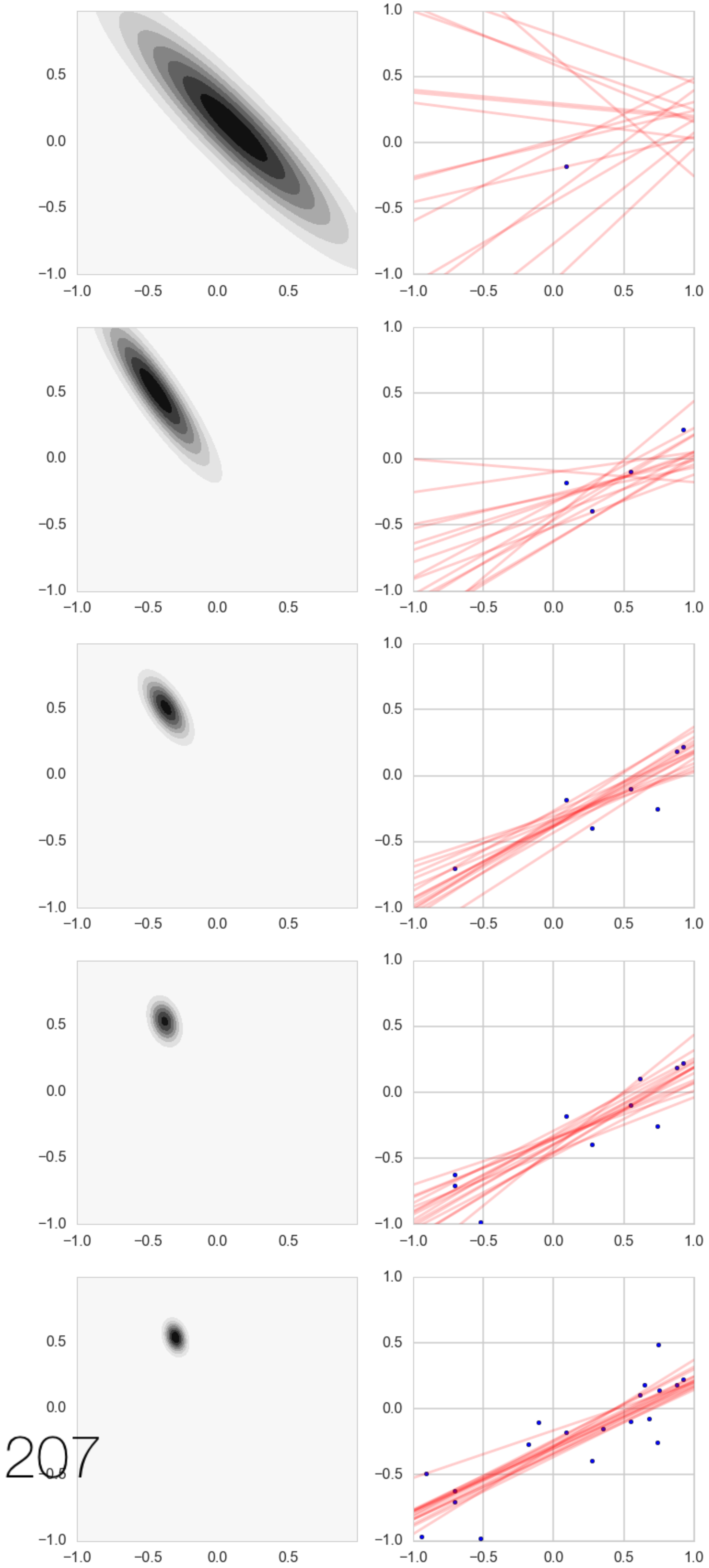
Posterior

$$p(\mathbf{w}|\mathbf{y},\mathbf{X}) \propto p(\mathbf{y}|\mathbf{X},\mathbf{w})\,\mathbf{p(w)}$$
$$\propto \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y}-\mathbf{X^T w})^\mathbf{T}(\mathbf{y}-\mathbf{X^T w})\right)\exp\left(-\frac{1}{2}\mathbf{w^T\Sigma^{-1}w}\right)$$

$$p(\mathbf{w}|\mathbf{y},\mathbf{X}) \propto \exp\left(-\frac{\mathbf{1}}{\mathbf{2}}(\mathbf{w}-\bar{\mathbf{w}})^\mathbf{T}(\frac{\mathbf{1}}{\sigma_\mathbf{n}^\mathbf{2}}\mathbf{XX^T}+\mathbf{\Sigma^{-1}})(\mathbf{w}-\bar{\mathbf{w}})\right)$$

Inverse covariance $A = \sigma_n^{-2}XX^T + \Sigma^{-1}$

where the new mean is $\bar{\mathbf{w}} = A^{-1}\Sigma^{-1}\mathbf{w_0} + \sigma_n^{-2}(A^{-1}X^T\mathbf{y})$



Bayesian updating

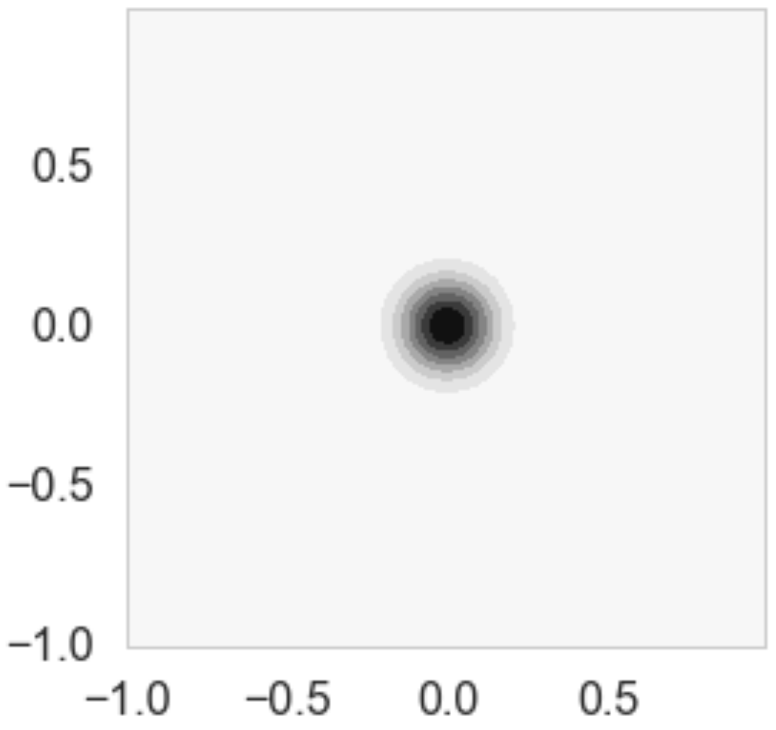
```
def update(x,y,likelihoodPrecision,priorMu,priorCovariance):
    postCovInv  = np.linalg.inv(priorCovariance) + likelihoodPrecision*np.outer(x.T,x)
    #The outer product looks wrong but when updating we need a 2x1 matrix while x is 1x2
    postCovariance = np.linalg.inv(postCovInv)
    postMu =
        np.dot(np.dot(postCovariance,np.linalg.inv(priorCovariance)),
            priorMu) +likelihoodPrecision*
            np.dot(postCovariance,np.outer(x.T,y)).flatten()
    postW = lambda w:multivariate_normal.pdf(w,postMu,postCovariance)
    return postW, postMu, postCovariance
```

Posterior predictive

$$p(y^*|x^*,\mathbf{x},\mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{w})p(\mathbf{w}|\mathbf{X},\mathbf{y})\mathbf{dw}$$

$$= \mathcal{N}\left(y|\bar{\mathbf{w}}^T x^*, \sigma_n^2 + x^{*^T} A^{-1} x^*\right),$$

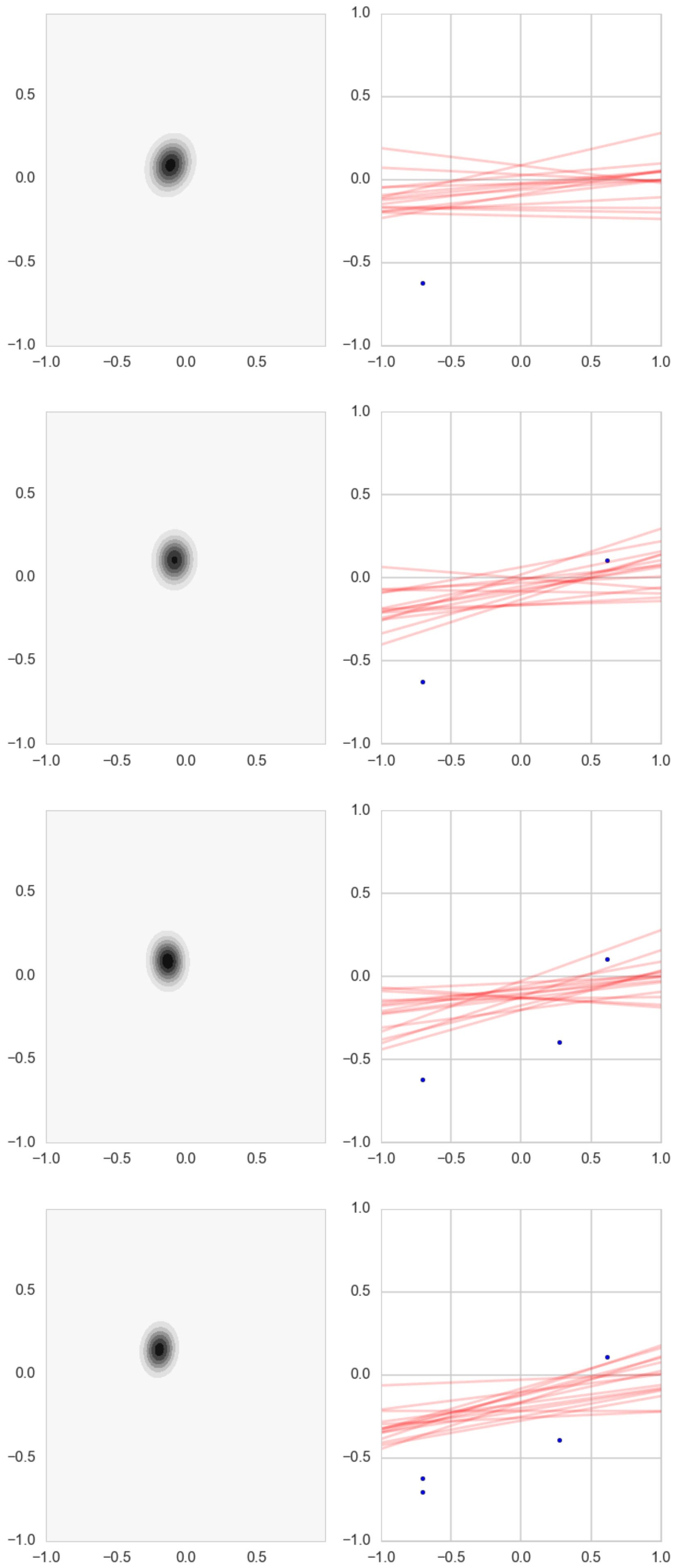
Regularization



`priorPrecision/likelihoodPrecision`

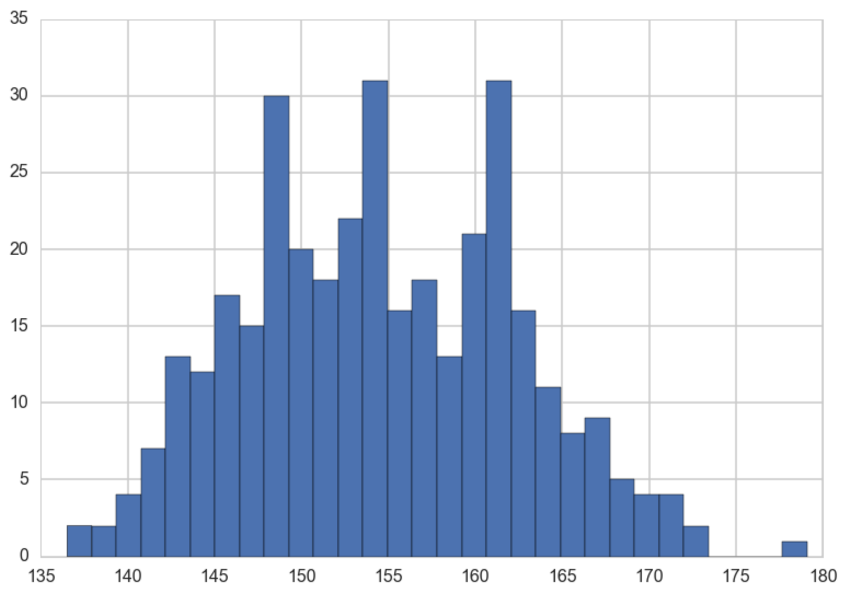
`4.0`

This ratio is the ridge α.



Howell's data

- These are census data for the Dobe area !Kung San people
- Nancy Howell conducted detailed quantitative studies of this Kalahari foraging population in the 1960s.

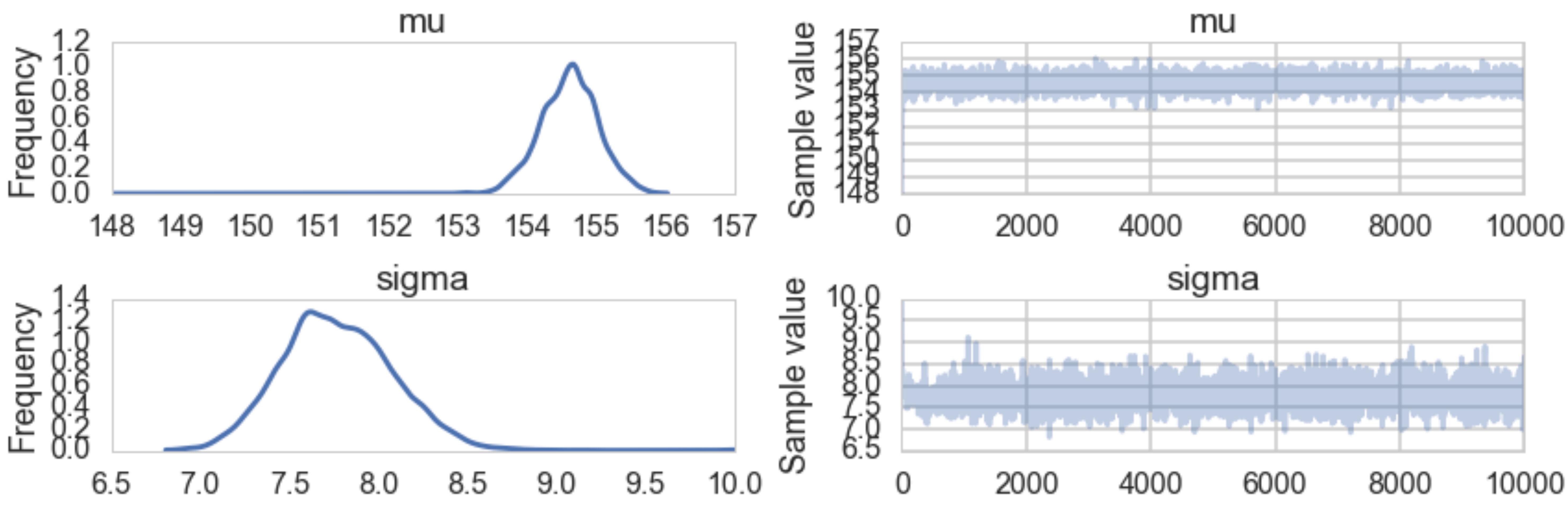


	height	weight	age	male
0	151.765	47.825606	63.0	1
1	139.700	36.485807	63.0	0
2	136.525	31.864838	65.0	0
3	156.845	53.041915	41.0	1
4	145.415	41.276872	51.0	0

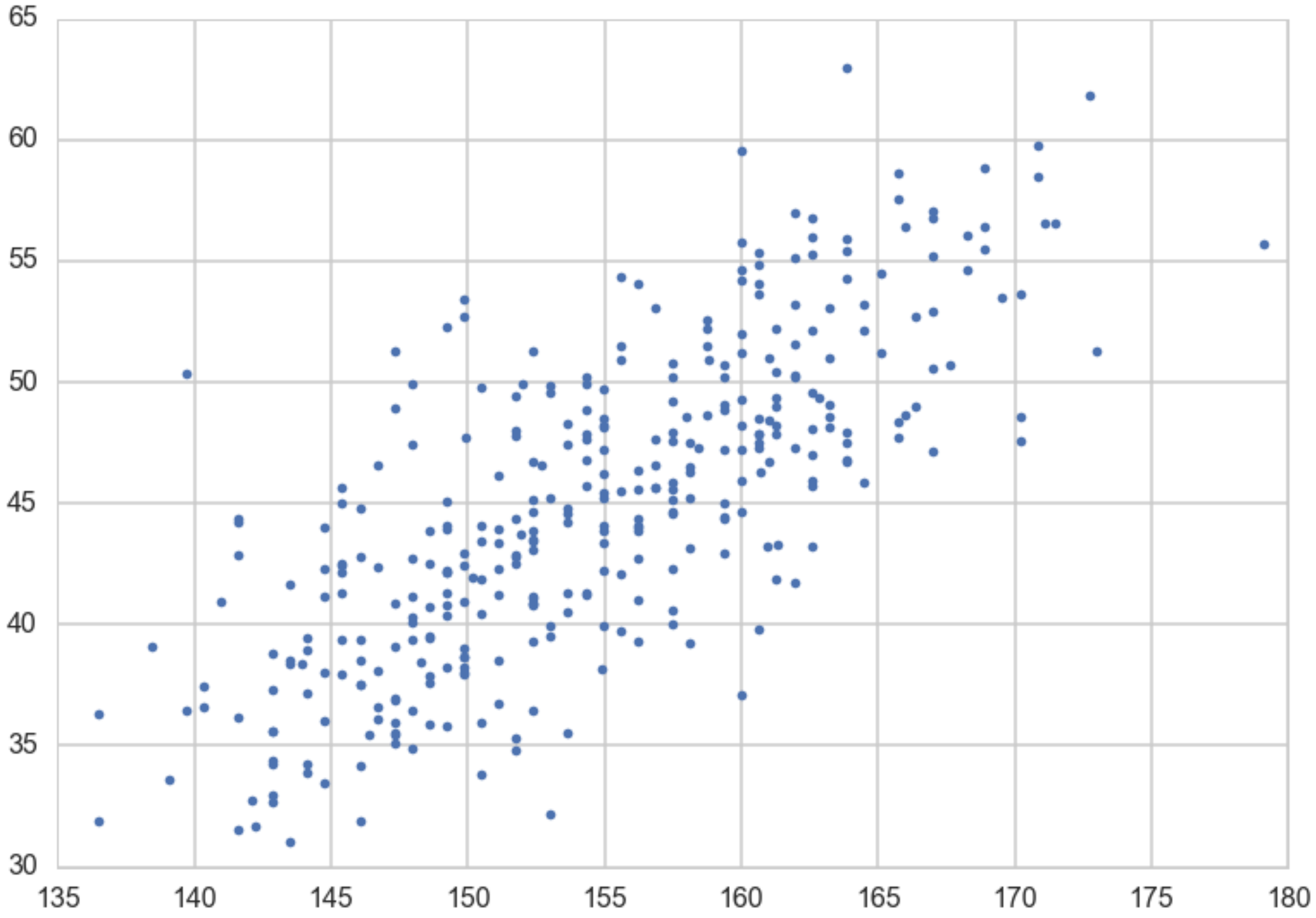
Model

$$h \sim N(\mu, \sigma)$$
$$\mu \sim Normal(148, 20)$$
$$\sigma \sim Unif(0, 20)$$

```
with pm.Model() as hm1:
    mu = pm.Normal('mu', mu=148, sd=20)#parameter
    sigma = pm.Uniform('sigma', lower=0, upper=20)#testval=df2.height.mean()
    height = pm.Normal('height', mu=mu, sd=sigma, observed=df2.height)

with hm1:
    stepper=pm.Metropolis()
    tracehm1=pm.sample(10000, step=stepper)# a start argument could be used here
    #as well

100%|██████████| 10000/10000 [00:02<00:00, 4180.50it/s] | 1/10000 [00:00<16:55,  9.84it/s]
```




```
def acceptance(trace, paramname):
    accept = np.sum(trace[paramname][1:] != trace[paramname][:-1])
    return accept/trace[paramname].shape[0]

acceptance(tracehm1, 'mu'), acceptance(tracehm1, 'sigma')
(0.3896, 0.30009999999999998)
```

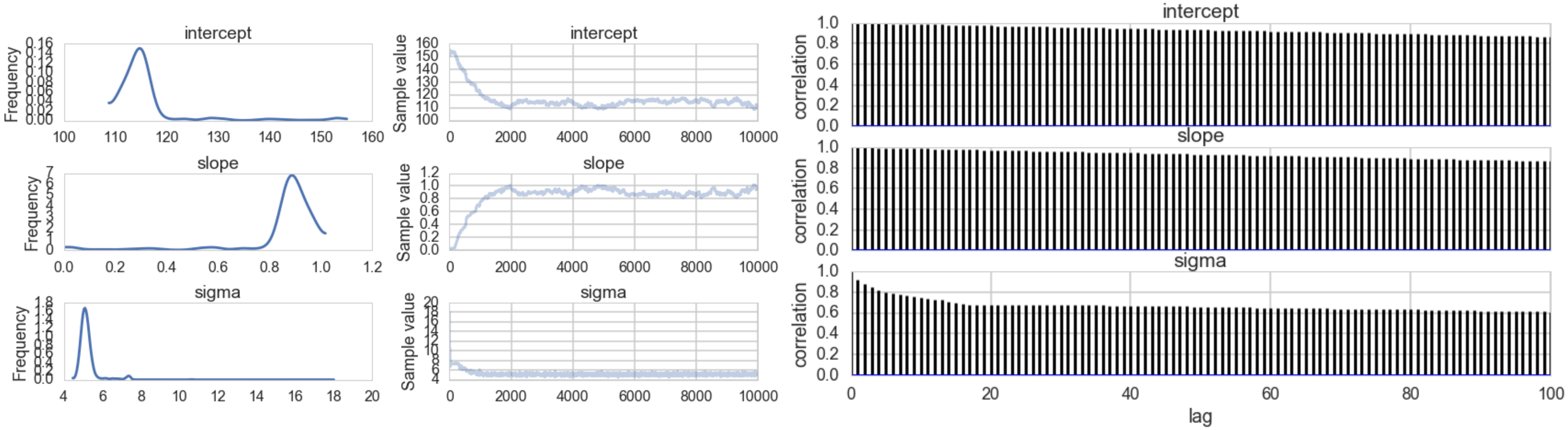


Regression, adding a predictor, weight

$$h \sim N(\mu, \sigma)$$
$$\mu = intercept + slope \times weight$$
$$intercept \sim N(150, 100)$$
$$slope \sim N(0, 10)$$
$$\sigma \sim Unif(0, 50)$$

```
with pm.Model() as hm2:
    intercept = pm.Normal('intercept', mu=150, sd=100)
    slope = pm.Normal('slope', mu=0, sd=10)
    sigma = pm.Uniform('sigma', lower=0, upper=50)
    # below is a deterministic
    mu = intercept + slope * df2.weight
    height = pm.Normal('height', mu=mu, sd=sigma, observed=df2.height)
    stepper=pm.Metropolis()
    tracehm2 = pm.sample(10000, step=stepper)
```

Traces are awful



The slope and intercept are very highly correlated: -0.99!

Non-Identifiability in sum model

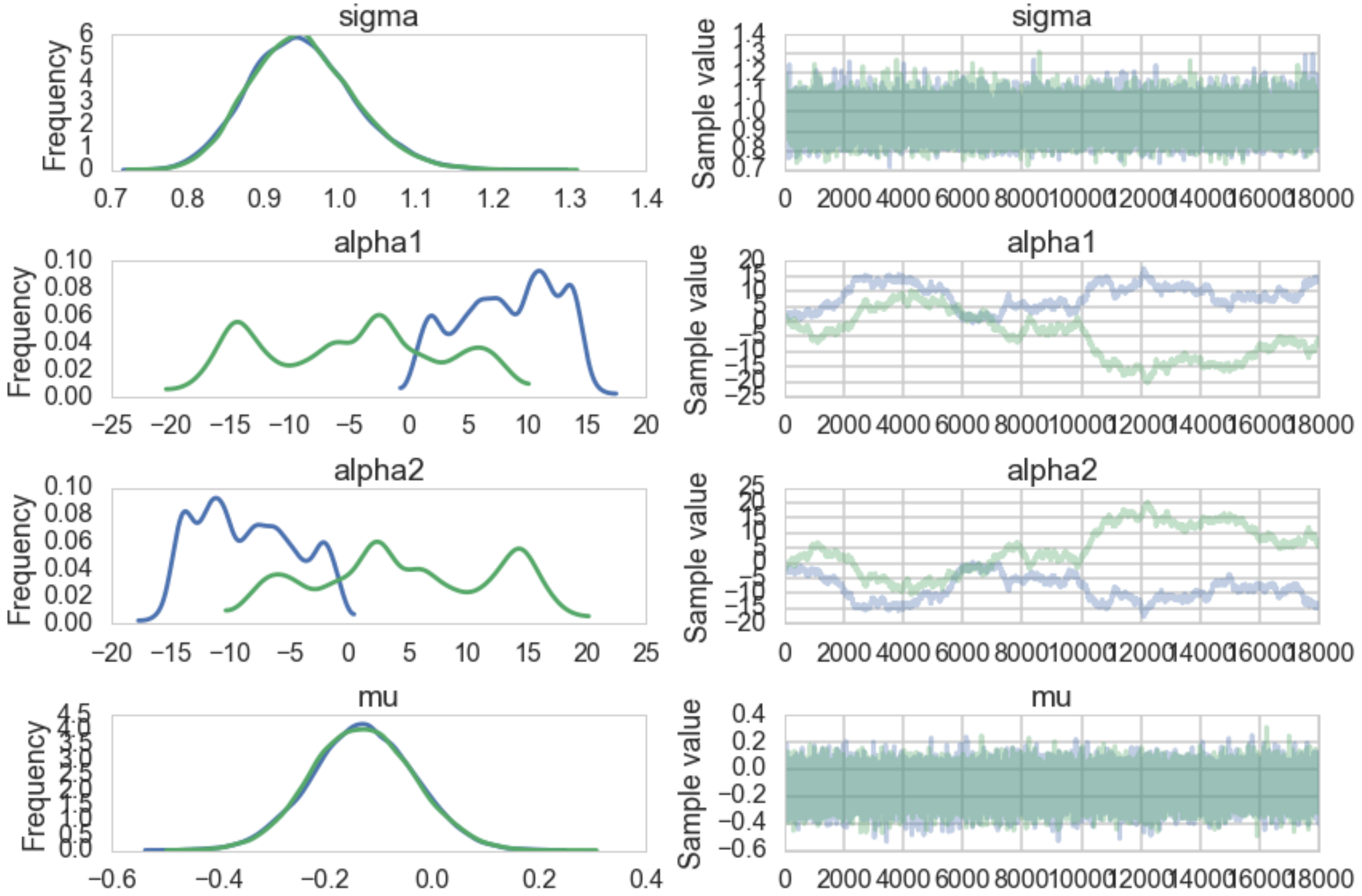
Generate data from N(0,1). Then fit:

$$y \sim N(\mu, \sigma)$$

$$\mu = \alpha_1 + \alpha_2$$

$$\alpha_1 \sim Unif(-\infty, \infty)$$
$$\alpha_2 \sim Unif(-\infty, \infty)$$
$$\sigma \sim HalfCauchy(0, 1)$$

Non-Identifiability




```
with pm.Model() as ni:
    sigma = pm.HalfCauchy("sigma", beta=1)
    alpha1=pm.Uniform('alpha1', lower=-10**6, upper=10**6)
    alpha2=pm.Uniform('alpha2', lower=-10**6, upper=10**6)
    mu = pm.Deterministic("mu", alpha1 + alpha2)
    y = pm.Normal("data", mu=mu, sd=sigma, observed=data)
    stepper=pm.Metropolis()
    traceni = pm.sample(100000, step=stepper, njobs=2)

100%|██████████| 100000/100000 [01:17<00:00, 1286.59it/s]| 1/100000 [00:00<3:20:31,  8.31it/s]
```

```
df=pm.trace_to_dataframe(traceni)
df.corr()
```

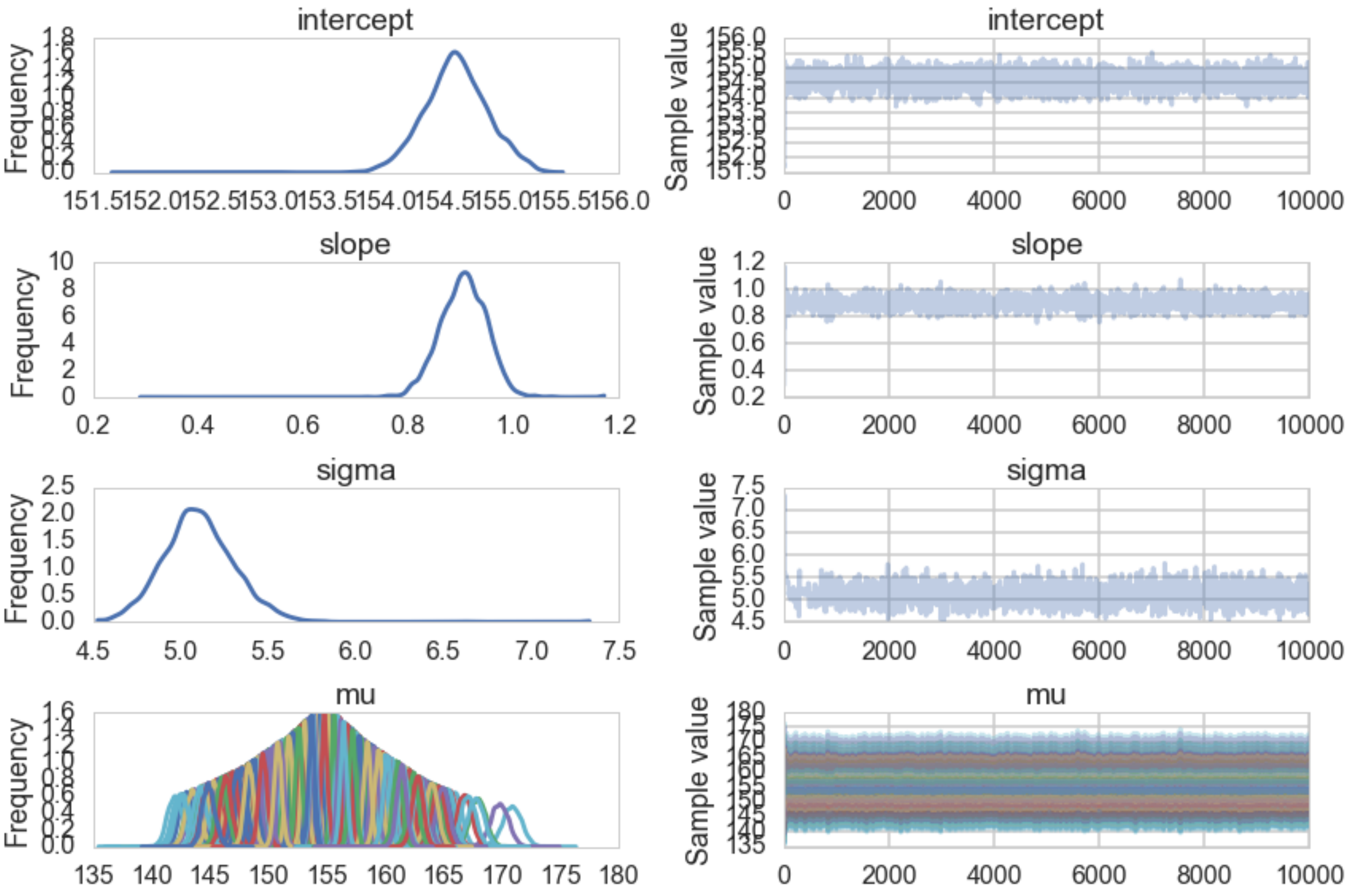
	sigma	mu	alpha1	alpha2
sigma	1.000000	-0.000115	-0.003153	0.003152
mu	-0.000115	1.000000	0.002844	0.008293
alpha1	-0.003153	0.002844	1.000000	-0.999938
alpha2	0.003152	0.008293	-0.999938	1.000000

```
>>>pm.effective_n(traceni)
{'alpha1': 1.0,
 'alpha1_interval_': 1.0,
 'alpha2': 1.0,
 'alpha2_interval_': 1.0,
 'mu': 26411.0,
 'sigma': 39215.0,
 'sigma_log_': 39301.0}
 >>>pm.gelman_rubin(traceni)
 {'alpha1': 1.7439881580327452,
  'alpha1_interval_': 1.7439881580160093,
  'alpha2': 1.7438626593529831,
  'alpha2_interval_': 1.7438626593368223,
  'mu': 0.99999710182062695,
  'sigma': 1.0000248056117549,
  'sigma_log_': 1.0000261752214563}
```

Regression traces

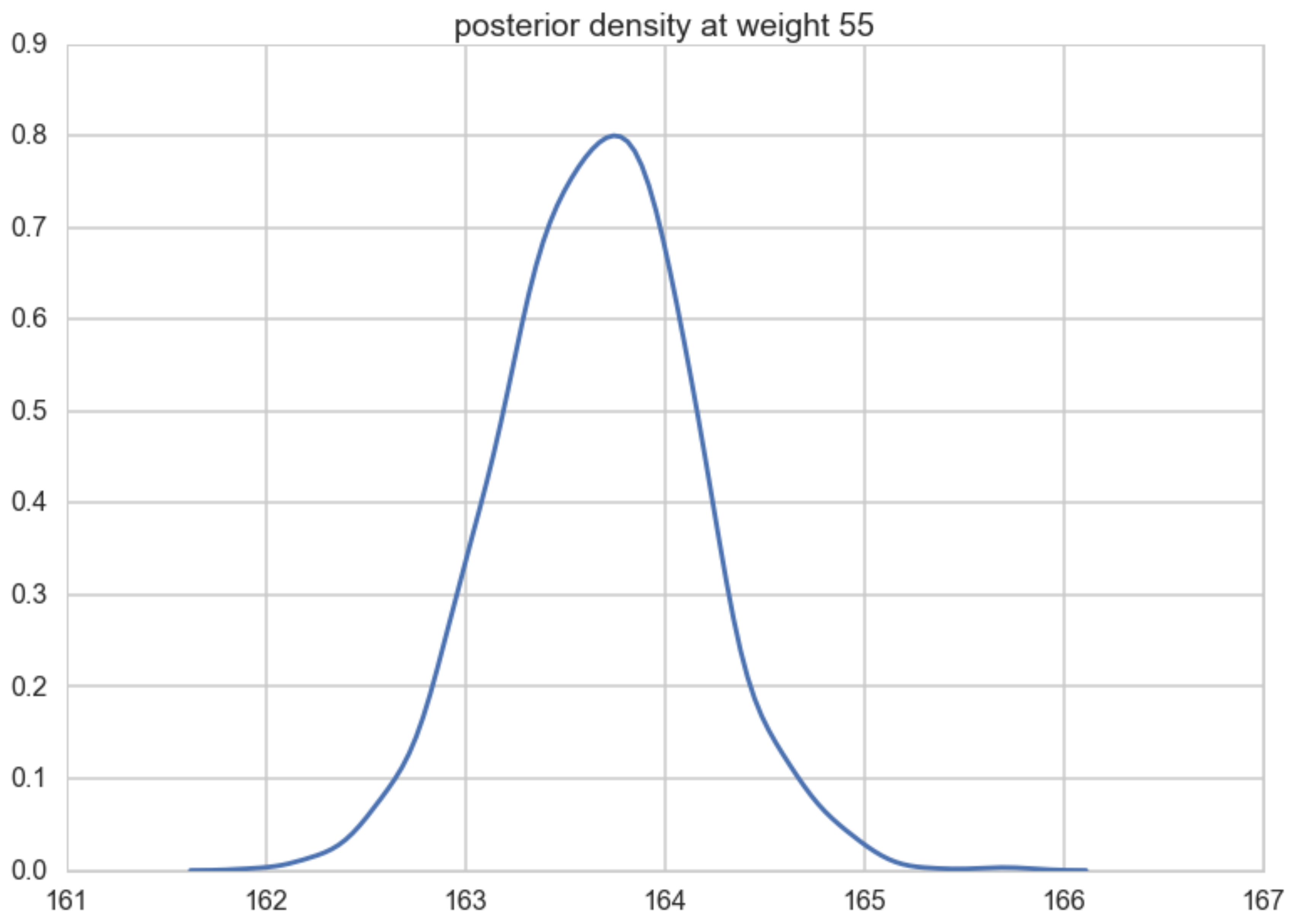
- symptom of shared information and identifiability
- fix by centering. intercept then gives response when predictor=mean.

```
with pm.Model() as hm2c:
    intercept = pm.Normal('intercept', mu=150, sd=100)
    slope = pm.Normal('slope', mu=0, sd=10)
    sigma = pm.Uniform('sigma', lower=0, upper=50)
    mu = pm.Deterministic('mu', intercept + slope * (df2.weight -df2.weight.mean()))
    height = pm.Normal('height', mu=mu, sd=sigma, observed=df2.height)
    stepper=pm.Metropolis()
    tracehm2c = pm.sample(10000, step=stepper)
```

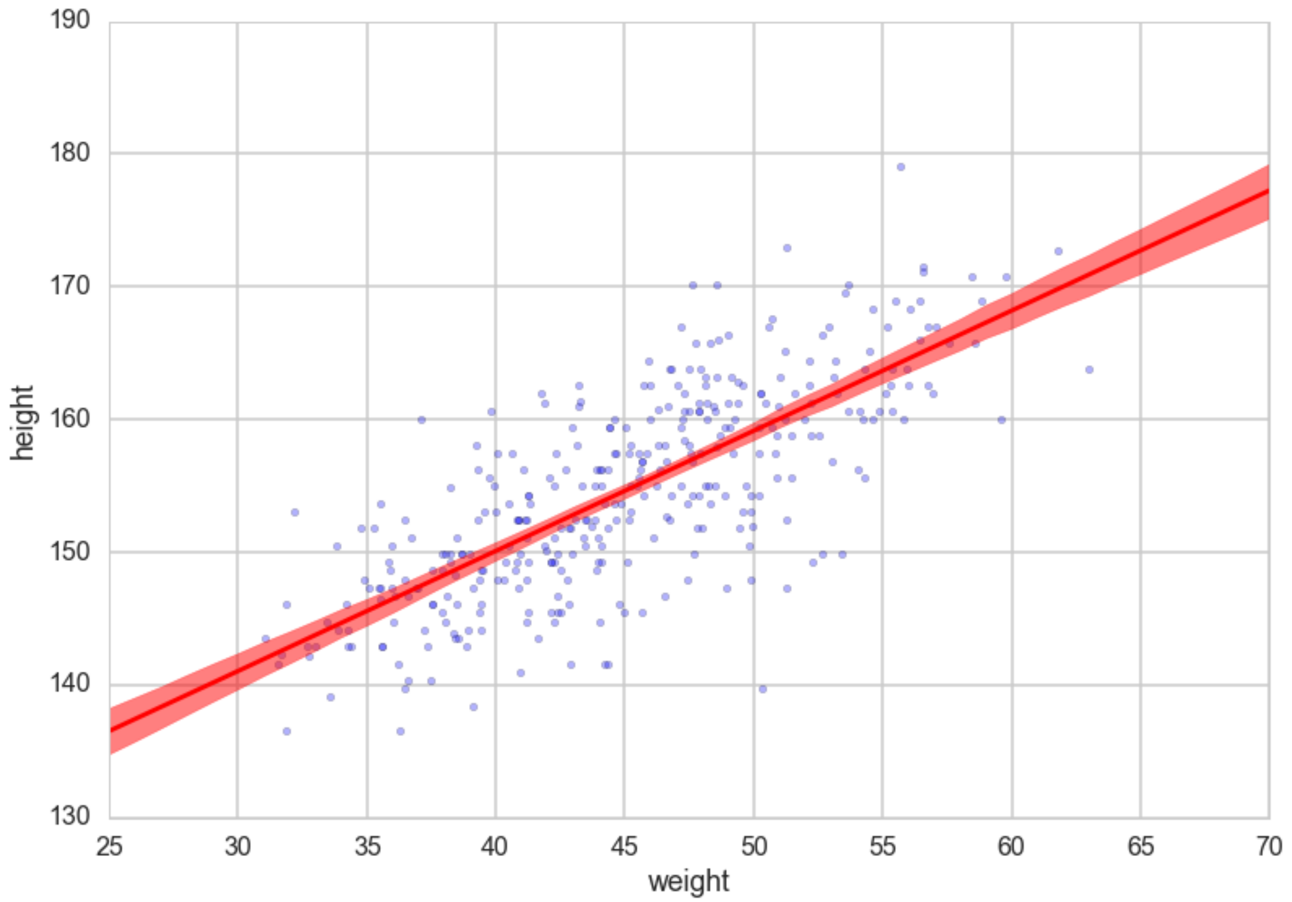

intercept
Frequency
151.5152.0152.5153.0153.5154.0154.5155.0155.5156.0
intercept
Sample value
0
2000
4000
6000
8000
10000
slope
Frequency
10
8
6
4
2
0
0.2
0.4
0.6
0.8
1.0
1.2
slope
Sample value
1.2
1.0
0.8
0.6
0.4
0.2
sigma
Frequency
2.5
2.0
1.5
1.0
0.5
0.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
sigma
Sample value
7.5
7.0
6.5
6.0
5.5
5.0
4.5
mu
Frequency
135
140
145
150
155
160
165
170
175
180
mu
Sample value

Posteriors

```
meanweight = df2.weight.mean()
weightgrid = np.arange(25, 71)
mu_pred = np.zeros((len(weightgrid), len(tr2c)))
for i, w in enumerate(weightgrid):
    mu_pred[i] = tr2c['intercept'] + tr2c['slope'] * (w - meanweight)

mu_mean = mu_pred.mean(axis=1)
mu_hpd = pm.hpd(mu_pred.T)
```



Posteriors on a grid



Why so tight?

Posterior predictive

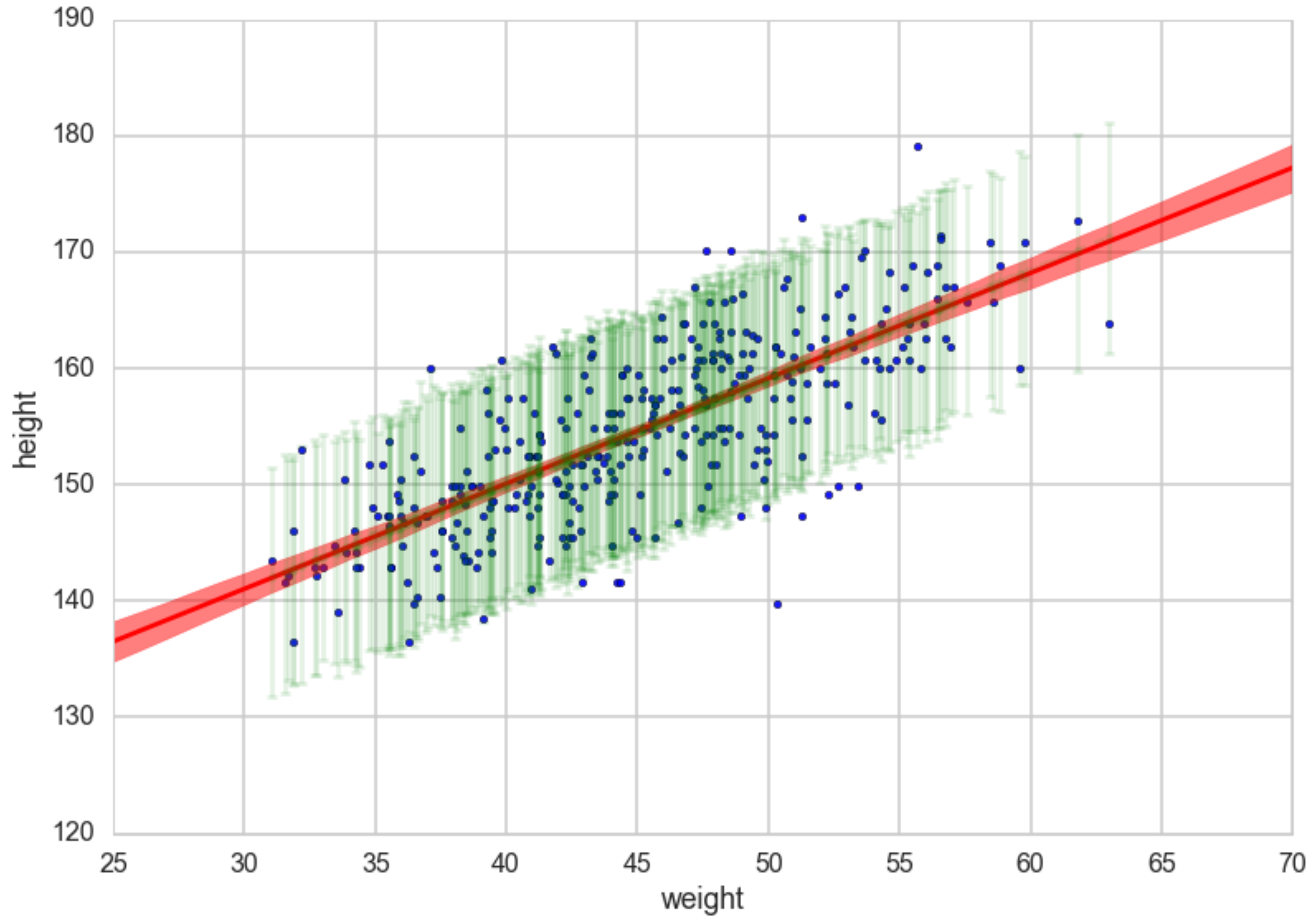
At data:

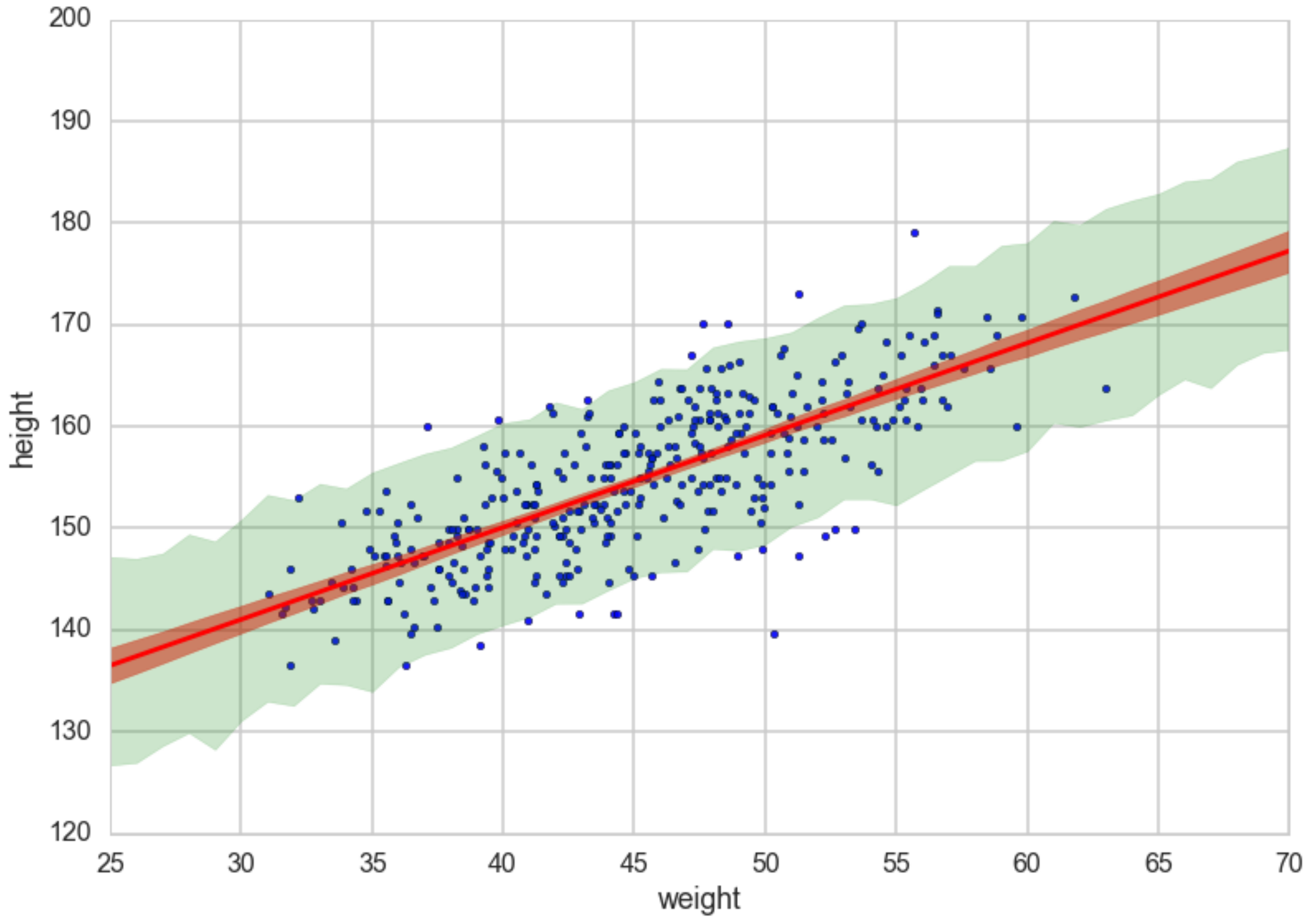
```
postpred = pm.sample_ppc(tr2c, 1000, hm2c)
100%|██████████| 1000/1000 [00:19<00:00, 57.56it/s]    | 1/1000 [00:00<08:17,  2.01it/s]
```

On a full grid:

```
n_ppredsamps=1000
weightgrid = np.arange(25, 71)
meanweight = df2.weight.mean()
ppc_samples=np.zeros((len(weightgrid), n_ppredsamps))

for j in range(n_ppredsamps):
    k=np.random.randint(len(tr2c))#samples with replacement
    musamps = tr2c['intercept'][k] + tr2c['slope'][k] * (weightgrid - meanweight)
    sigmasamp = tr2c['sigma'][k]
    ppc_samples[:,j] = np.random.normal(musamps, sigmasamp)
```

Predictives at data and on grid



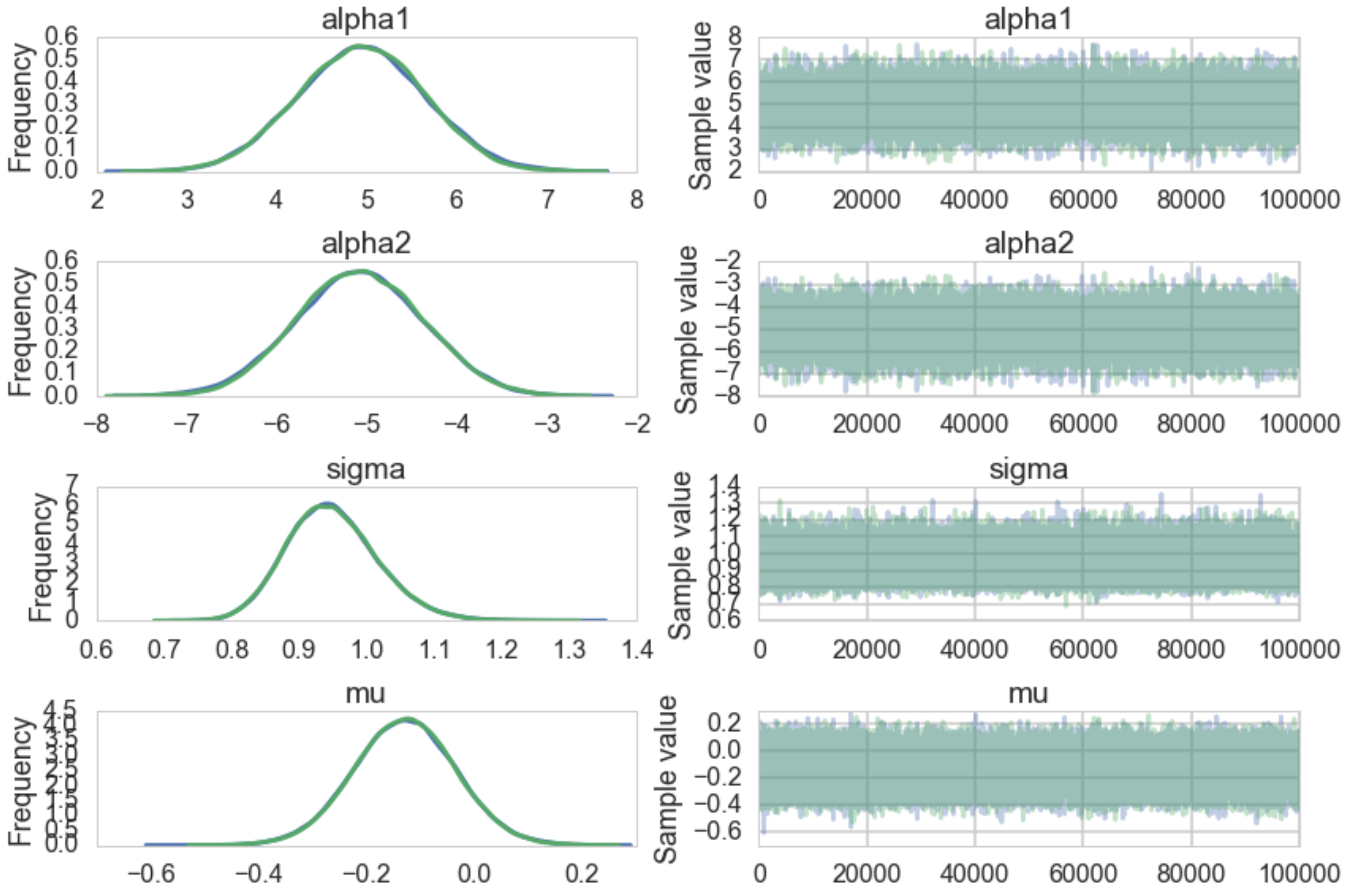


Attempt to fix non-identifiability of sum

```
with pm.Model() as ni2:
    sigma = pm.HalfCauchy("sigma", beta=1)
    alpha1=pm.Normal('alpha1', mu=5, sd=1)
    alpha2=pm.Normal('alpha2', mu=-5, sd=1)
    mu = pm.Deterministic("mu", alpha1 + alpha2)
    y = pm.Normal("data", mu=mu, sd=sigma, observed=data)
    #stepper=pm.Metropolis()
    #traceni2 = pm.sample(100000, step=stepper, njobs=2)
    traceni2 = pm.sample(100000, njobs=2)

Average ELBO = -143.13: 100%|██████████| 200000/200000 [00:18<00:00, 10759.64it/s], 9912.87it/s]
100%|██████████| 100000/100000 [06:30<00:00, 255.83it/s]
```

NUTS sampler slower but covers better for this

eff_n = 20000 odd, GR 1.07, but correlation still super high



Ridge, Lasso, and Identifiability

Construct a model: $y = 10x_1 + 10x_2 + 0.1x_3$

where $x_1 \sim N(0,1)$, $x_2 = -x_1 + N(0, 10^{-3})$ and $x_3 \sim N(0,1)$

Thus our real model is

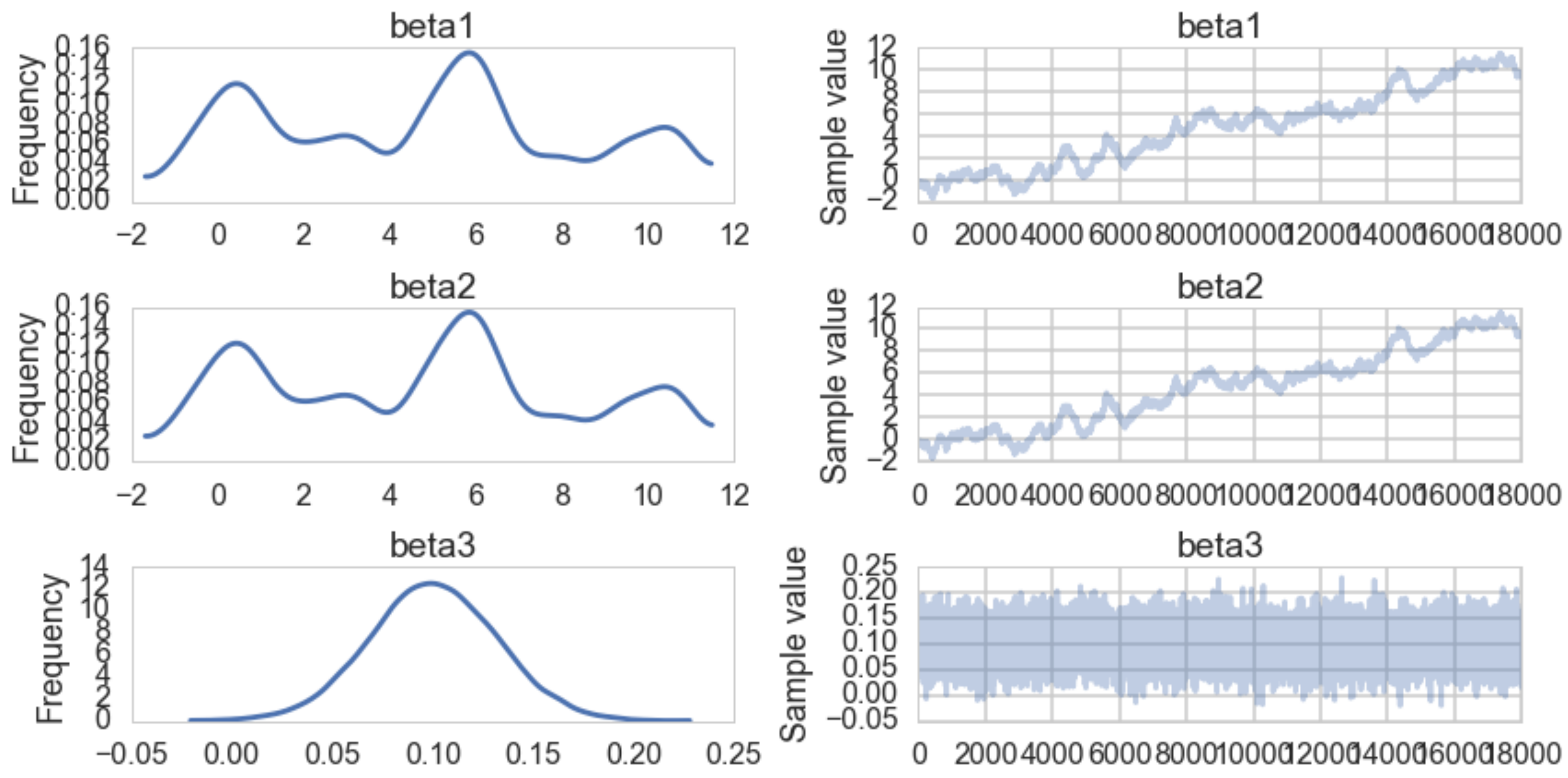
$$y = 10N(0, 10^{-3}) + 0.1N(0,1)$$

```
>>>np.dot(np.dot(np.linalg.inv(np.dot(X.T, X)), X.T), y)
array([ 10. ,  10. ,   0.1])
```

Model 1: uniform priors

```
beta_min = -10**6
beta_max = 10**6
with pm.Model() as uni:
    beta1 = pm.Uniform('beta1', lower=beta_min, upper=beta_max)
    beta2 = pm.Uniform('beta2', lower=beta_min, upper=beta_max)
    beta3 = pm.Uniform('beta3', lower=beta_min, upper=beta_max)
    mu = beta1*x1 + beta2*x2 + beta3*x3
    ys = pm.Normal('ys', mu=mu, tau=1.0, observed=y)
    stepper=pm.Metropolis()
    traceuni = pm.sample(100000, step=stepper)
```

```
100%|██████████| 100000/100000 [00:35<00:00, 2856.75it/s]| 1/100000 [00:00<4:16:19,  6.50it/s]
beta3:

  Mean             SD               MC Error         95% HPD interval
  -------------------------------------------------------------------

  0.100            0.032            0.000            [0.040, 0.165]

  Posterior quantiles:
  2.5            25             50             75             97.5
  |--------------|==============|==============|--------------|

  0.038          0.079          0.100          0.122          0.163
```

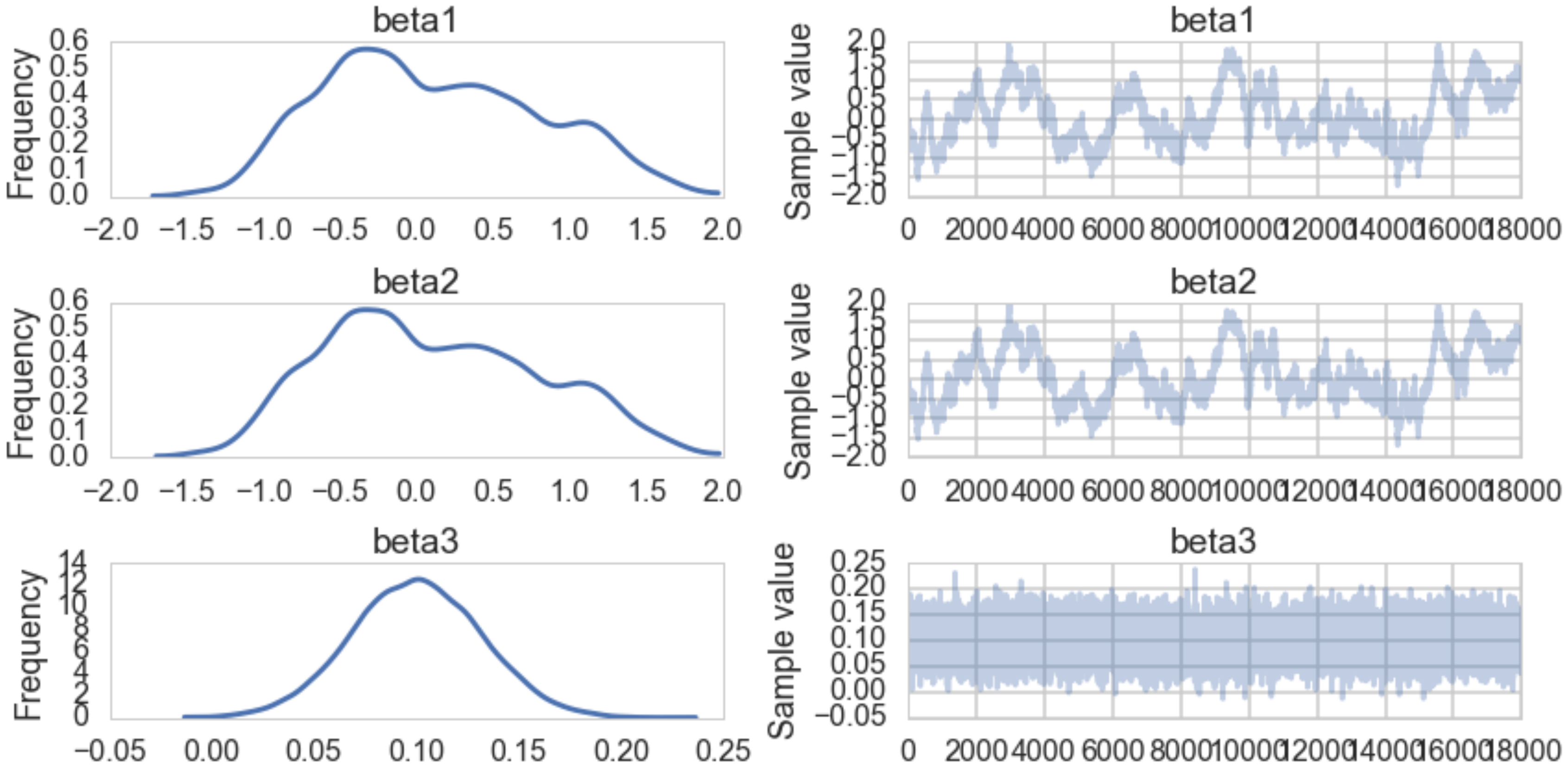


Model2: Ridge

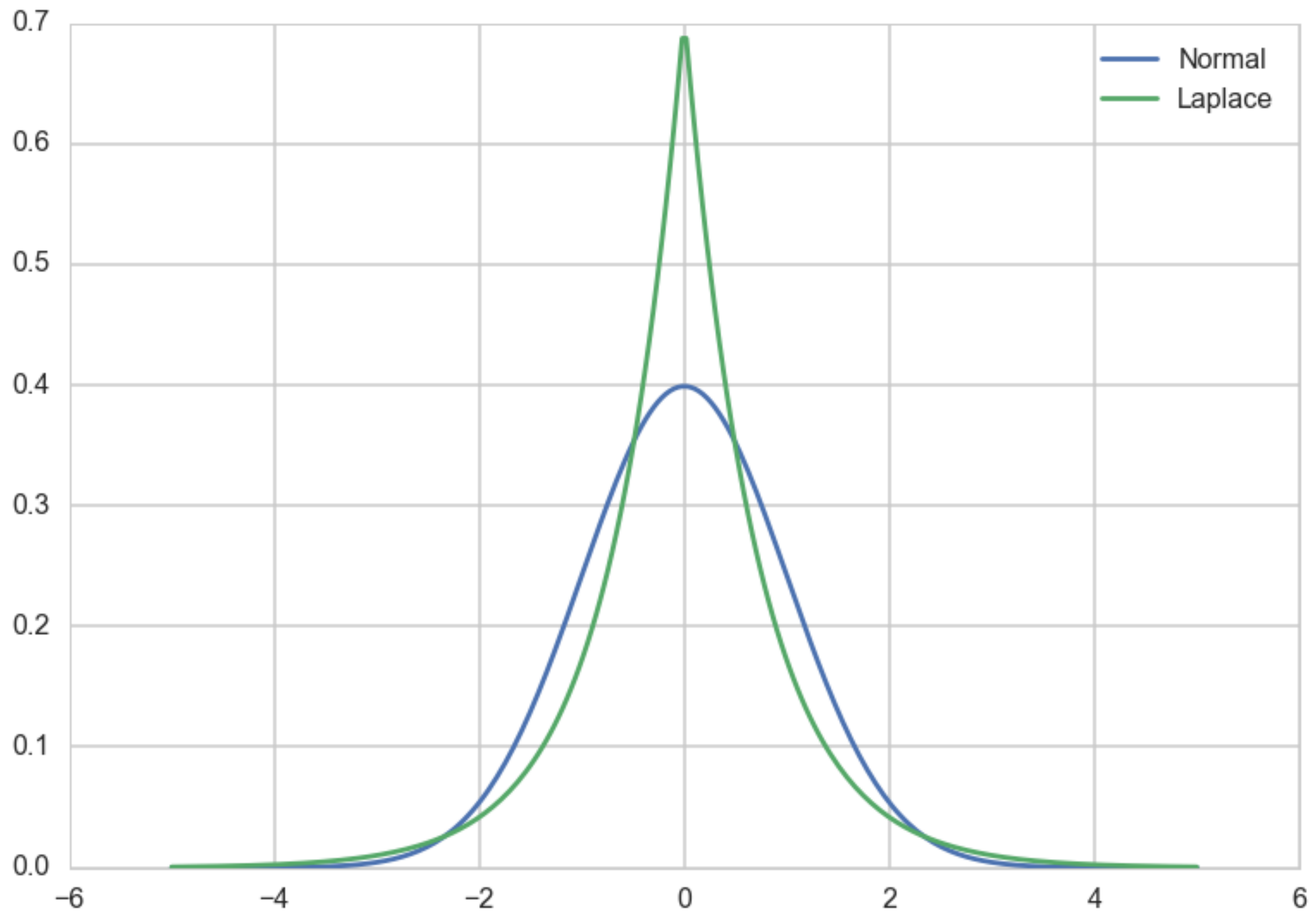
```python
with pm.Model() as ridge:
    beta1 = pm.Normal('beta1', mu=0, tau=1.0)
    beta2 = pm.Normal('beta2', mu=0, tau=1.0)
    beta3 = pm.Normal('beta3', mu=0, tau=1.0)
    mu = beta1*x1 + beta2*x2 + beta3*x3
    ys = pm.Normal('ys', mu=mu, tau=1.0, observed=y)
    stepper=pm.Metropolis()
    traceridge = pm.sample(100000, step=stepper)
```

```
100%|██████████| 100000/100000 [00:28<00:00, 3487.86it/s]| 68/100000 [00:00<02:27, 679.28it/s]
beta3:

  Mean             SD               MC Error         95% HPD interval
  -------------------------------------------------------------------

  0.100            0.032            0.000            [0.035, 0.159]

  Posterior quantiles:
  2.5            25             50             75             97.5
  |--------------|==============|==============|--------------|

  0.038          0.079          0.100          0.122          0.162
```

```python
with ridge:
    mapridge = pm.find_MAP()
{'beta1': array(0.004526796692482796),
 'beta2': array(0.005064112237104185),
 'beta3': array(0.10005872308519308)}
```



Laplace vs Gaussian Prior



Model 3: Lasso

```
b = 1.0 / np.sqrt(2.0 * sigma2)
with pm.Model() as lasso:
    beta1 = pm.Laplace('beta1', mu=0, b=b)
    beta2 = pm.Laplace('beta2', mu=0, b=b)
    beta3 = pm.Laplace('beta3', mu=0, b=b)
    mu = beta1*x1 + beta2*x2 + beta3*x3
    ys = pm.Normal('ys', mu=mu, tau=1.0, observed=y)
    stepper=pm.Metropolis()
    tracelasso = pm.sample(100000, step=stepper)
```

```
beta3:

  Mean             SD               MC Error         95% HPD interval
  -------------------------------------------------------------------

  0.099            0.032            0.000            [0.040, 0.164]

  Posterior quantiles:
  2.5            25             50             75             97.5
  |--------------|==============|==============|--------------|

  0.037          0.078          0.099          0.120          0.162
```

```
with lasso:
    maplasso = pm.find_MAP()
{'beta1': array(-7.255541060919206e-05),
 'beta2': array(8.485263161675386e-05),
 'beta3': array(0.10015818579834601)}
```