Lecture 13

Bayesian Stats(contd)

Previously

- sampling
- sampling with pymc3
- bayesian setup
- posteriors
- posterior predictives
- conjugate priors

pymc3 from lab

Data: $wingspan \sim N(\mu, \sigma)$

$\mu \sim Normal(19, 1.34), \sigma \sim Unif(0, 10)$

```
import pymc3 as pm
with pm.Model() as model12:
    mu = pm.Normal('mu', mu=19, sd=1.34)
    sigma = pm.Uniform('sigma', lower=0, upper=10)
    wingspan = pm.Normal('wingspan', mu=mu, sd=sigma,
    observed=Y)
    stepper=pm.Metropolis()
    tracemodel2=pm.sample(100000, step=stepper)
```

Marginalization

Marginal posterior:

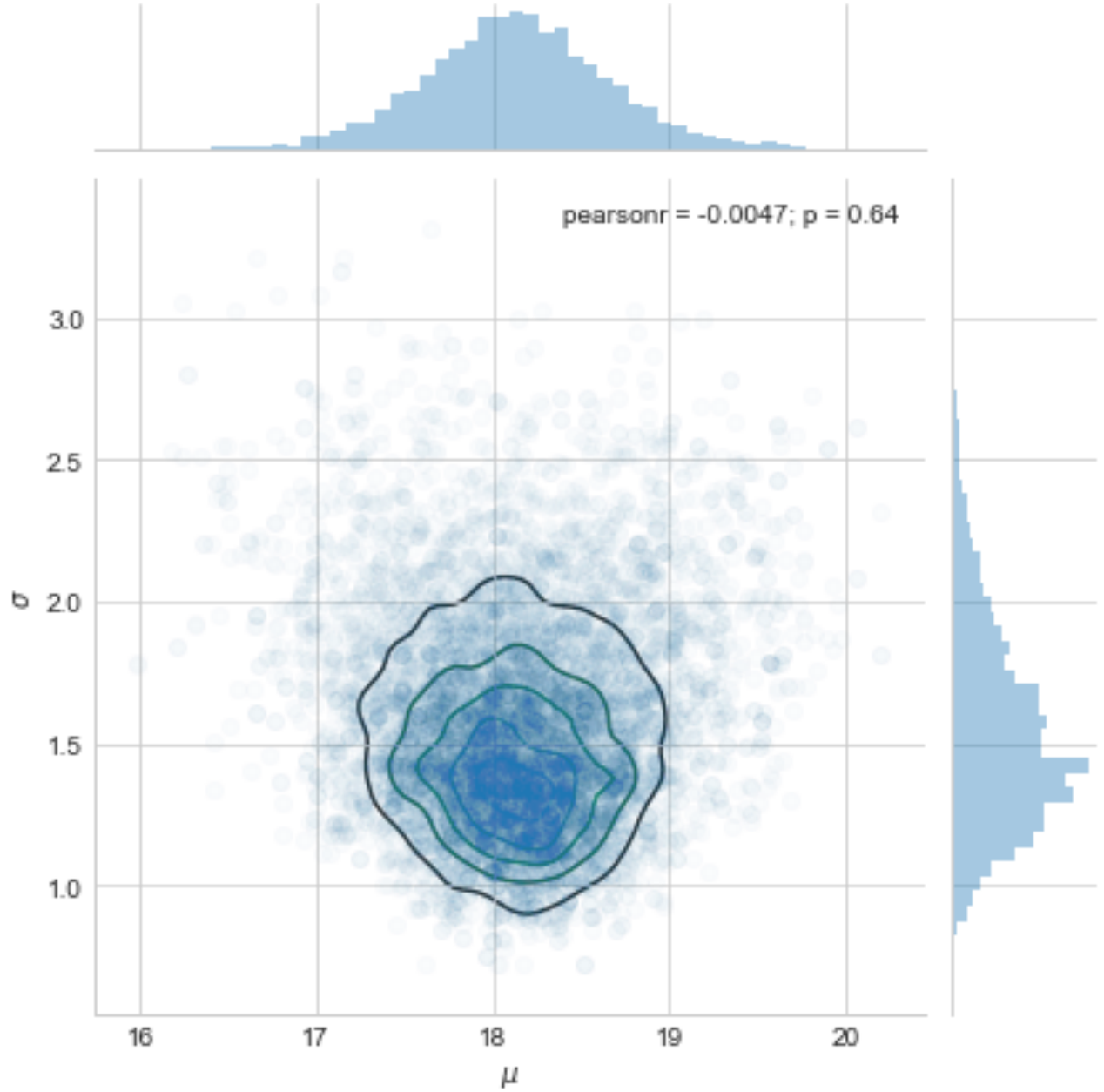
$$p(\theta_1|D) = \int d\theta_{-1} p(\theta|D).$$

```
samps[20000::,:].shape #(10001, 2)
```

```
sns.jointplot(
    pd.Series(samps[20000::,0], name="$\mu$"),
    pd.Series(samps[20000::,1], name="$\sigma$"),
    alpha=0.02)
    .plot_joint(
        sns.kdeplot,
    zorder=0, n_levels=6, alpha=1)
```

Marginals are just 1D histograms

```
plt.hist(samps[20000::,0])
```



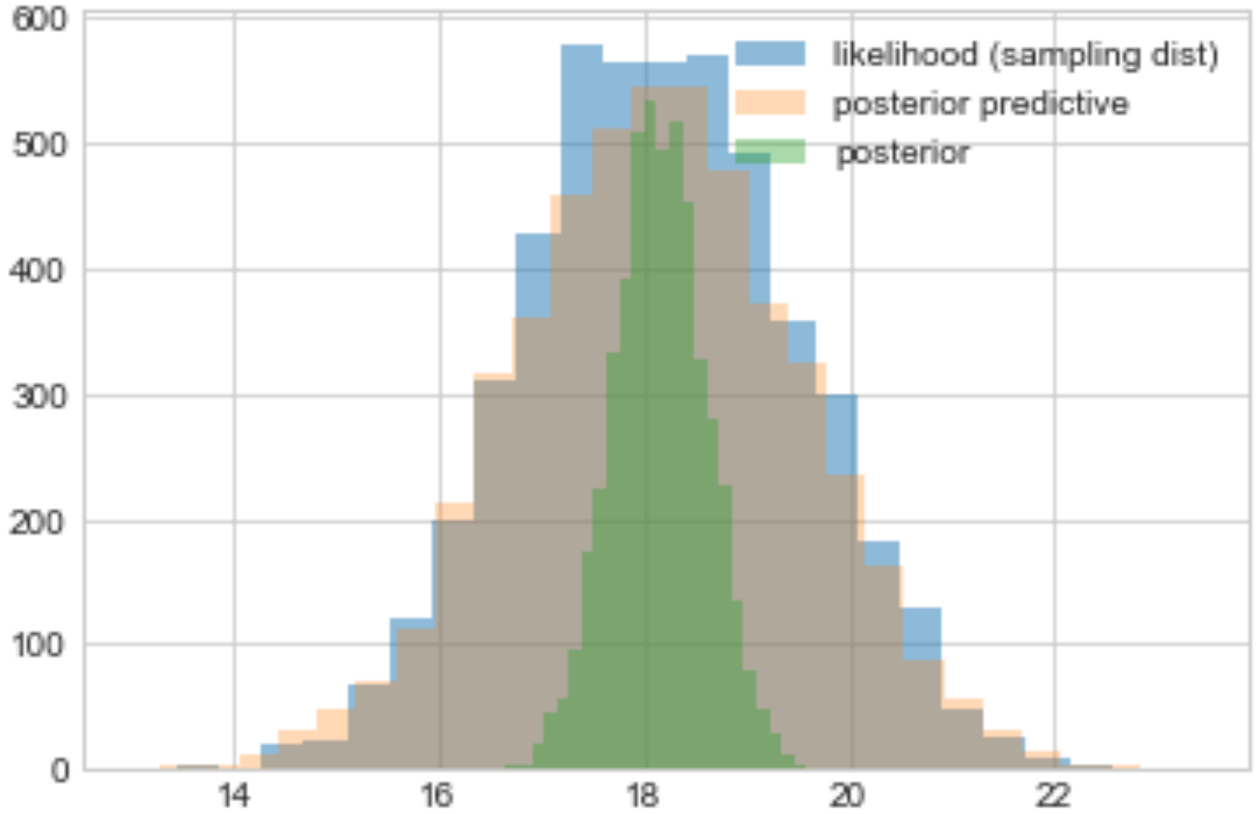
Posterior Predictive

The distribution of a future data point y^*:

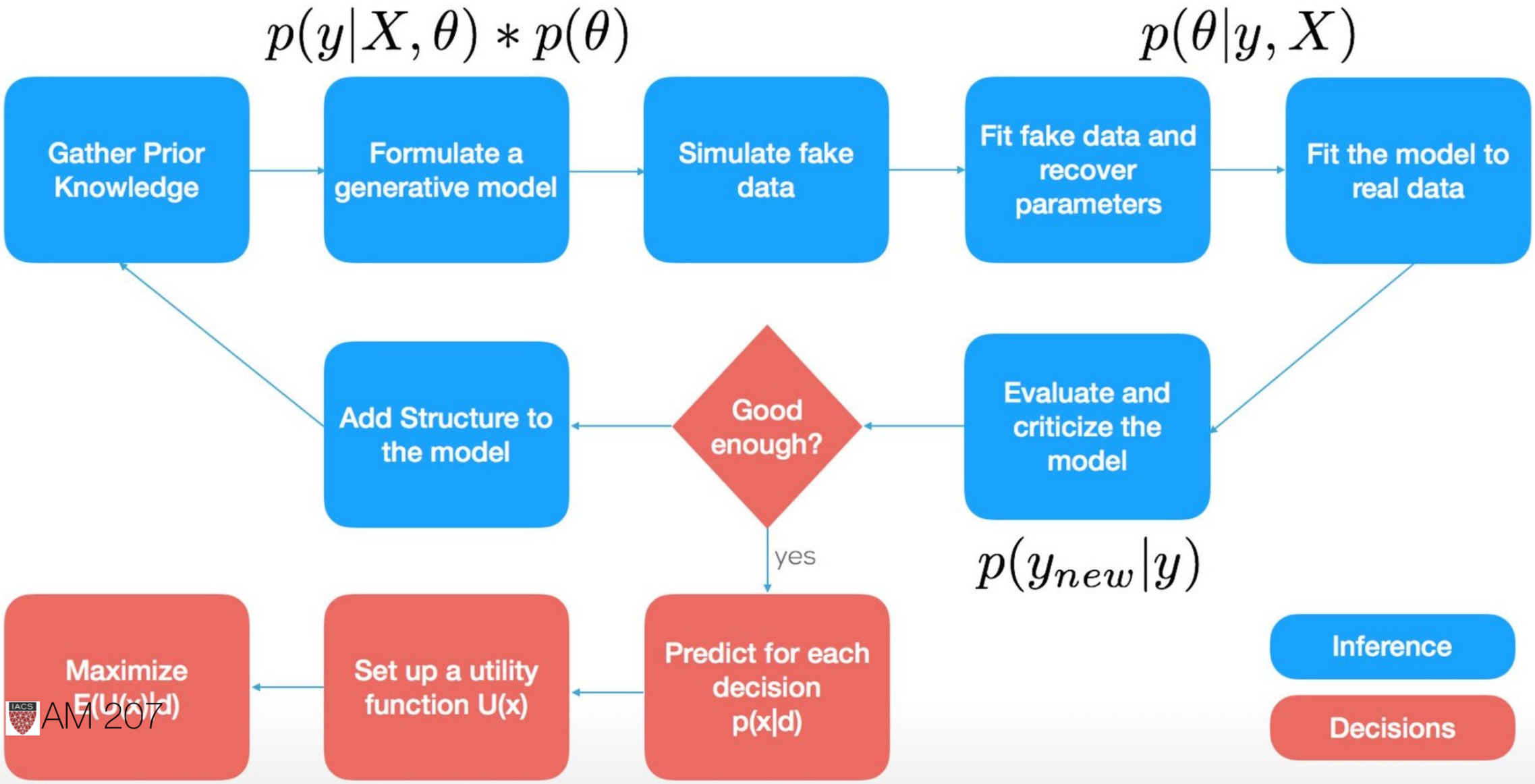
$$p(y^*|D=\{y\}) = E_{p(\theta|D)}[p(y|\theta)]$$
$$= \int d\theta p(y^*|\theta)p(\theta|\{y\}).$$

First draw the thetas from the posterior, then draw y's from the likelihood (these are draws from joint y, θ)

```
post_pred_func = lambda post: norm.rvs(loc = post, scale = sig)
post_pred_samples = post_pred_func(post_samples)
```



Bayesian Workflow
(from @ericnovik)



Conjugate Prior

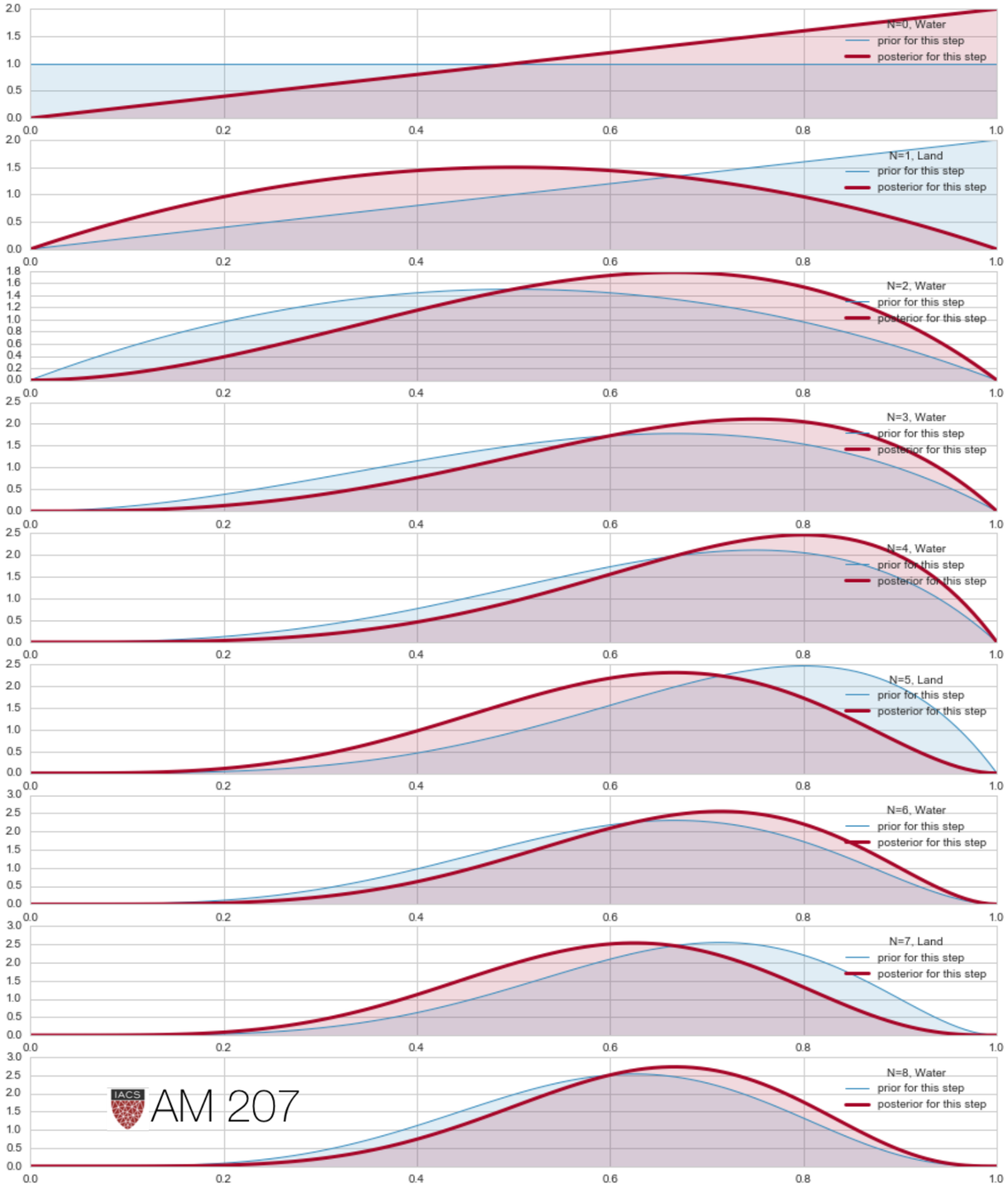
- A **conjugate prior** is one which, when multiplied with an appropriate likelihood, gives a posterior with the same functional form as the prior.
- Likelihoods in the exponential family have conjugate priors in the same family
- analytical tractability AND interpretability

Today

- globe toss
- poisson: sufficient stats and exchangeability
- priors
- a switchpoint model
- data imputation using posterior predictives
- Formal tests: Gewecke, Gelman-Rubin, and ESS

Globe Toss Model

- Seal tosses globe, θ is true water fraction
- The Beta distribution is conjugate to the Binomial distribution $p(\theta|y) \propto p(y|\theta)P(\theta) = Binom(n, y, \theta) \times Beta(\alpha, \beta)$
- Because of the conjugacy, this turns out to be: $Beta(y + \alpha, n - y + \beta)$
- a $Beta(1, 1)$ prior is equivalent to a uniform distribution.



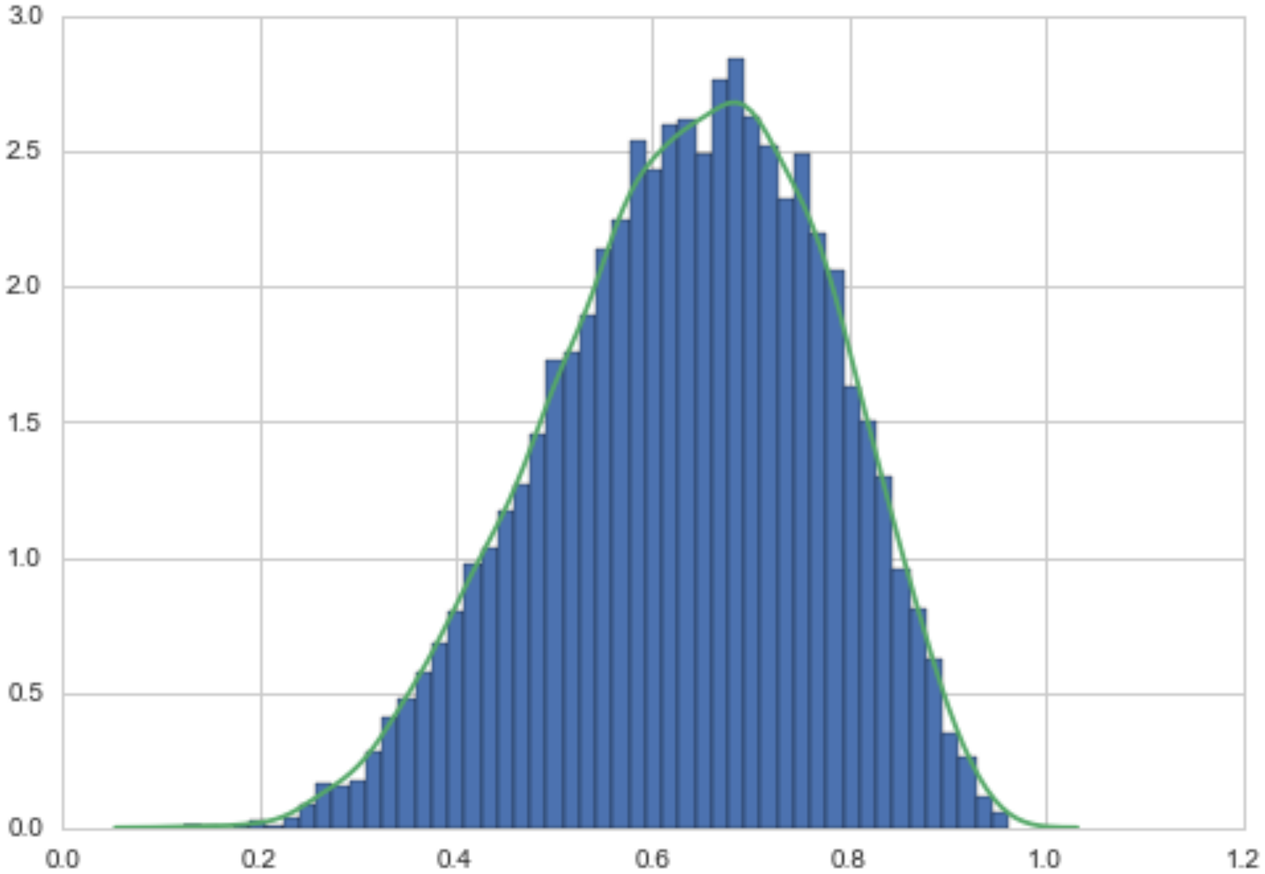
Bayesian Updating of globe

- data `WLWWWLWLW`
- notice how the posterior shifts left and right depending on new data

At each step:

$$Beta(y + \alpha, n - y + \beta)$$

Posterior



- The probability that the amount of water is less than 50%: `np.mean(samples < 0.5) = 0.173`
- Credible Interval: amount of probability mass. `np.percentile(samples, [10, 90]) = [ 0.44604094, 0.81516349]`
- `np.mean(samples), np.median(samples) = (0.6378734340335842, 0.6473143052303143)`

MAP, a point estimate

$$\begin{aligned}\theta_{\text{MAP}} &= \arg\max_{\theta} p(\theta|D) \\ &= \arg\max_{\theta} \frac{\mathcal{L}\, p(\theta)}{p(D)} \\ &= \arg\max_{\theta} \mathcal{L}\, p(\theta)\end{aligned}$$

```
sampleshisto = np.histogram(samples, bins=50)
maxcountindex = np.argmax(sampleshisto[0])
mapvalue = sampleshisto[1][maxcountindex]
print(maxcountindex, mapvalue)
```

```
31 0.662578641304
```

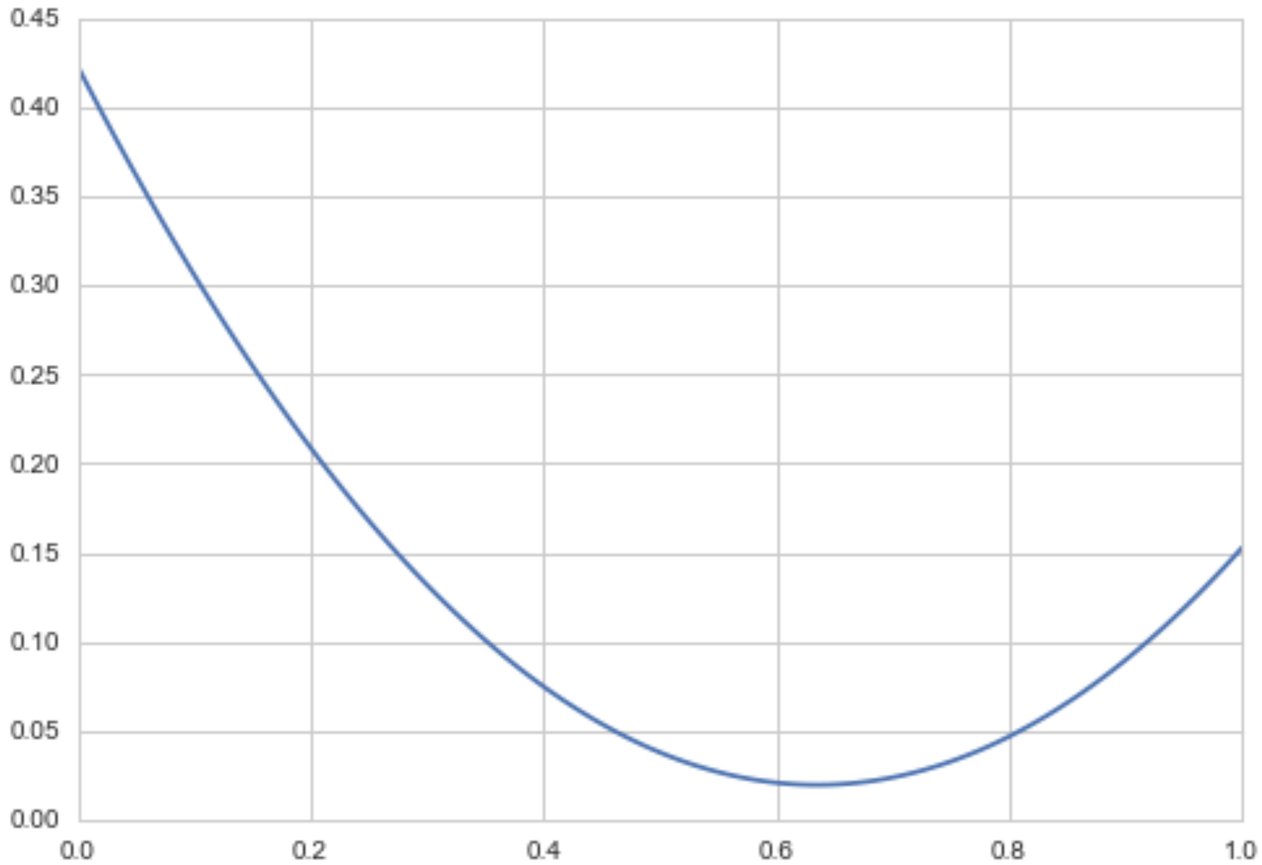
Posterior Mean minimizes squared loss

$$R(t) = E_{p(\theta|D)}[(\theta - t)^2] = \int d\theta (\theta - t)^2 p(\theta|D)$$

$$\frac{dR(t)}{dt} = 0 \implies t = \int d\theta \theta \, p(\theta|D)$$

```
mse = [np.mean((xi-samples)**2) for xi in x]
plt.plot(x, mse);
```

This is **Decision Theory**.



Posterior predictive

$$p(y^*|D) = \int d\theta p(y^*|\theta)p(\theta|D)$$

Risk Minimization holds here too: $y_{minmse} = \int dy\, y\, p(y|D)$

Plug-in Approximation: $p(\theta|D) = \delta(\theta - \theta_{MAP})$ and then draw

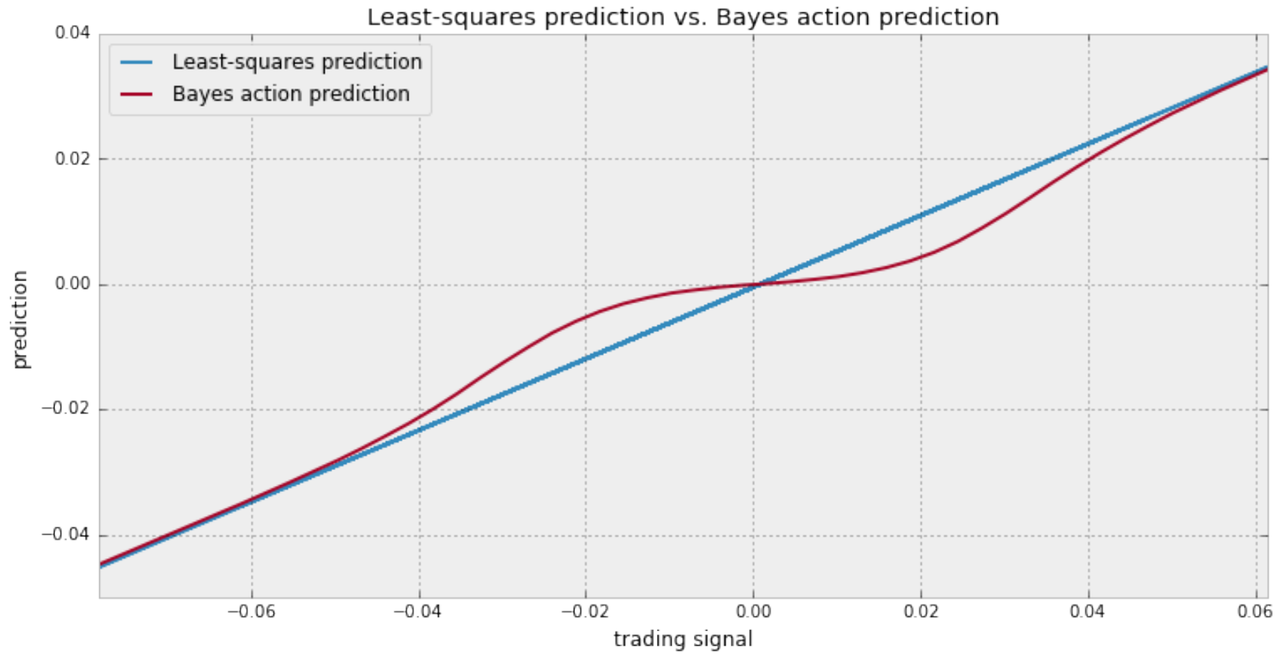
$$p(y^*|D) = p(y^*|\theta_{MAP})$$ a sampling distribution.

An unsual regression loss

```
def stock_loss(price, pred, coef = 500):
    """vectorized for numpy"""
    sol = np.zeros_like(price)
    ix = price*pred < 0
    sol[ix] = coef*pred**2 - np.sign(price[ix])*pred + abs(price[ix])
    sol[~ix] = abs(price[~ix] - pred)
    return sol
noise = std_samples*np.random.randn(N)


possible_outcomes = lambda signal: alpha_samples + \
 beta_samples*signal + noise


opt_predictions = np.zeros(50)
trading_signals =  np.linspace(X.min(), X.max(), 50)
for i, _signal in enumerate(trading_signals):
        _possible_outcomes = possible_outcomes(_signal)
        tomin = lambda pred:
            stock_loss(_possible_outcomes, pred).mean()
        opt_predictions[i] = fmin(tomin, 0, disp = False)
```

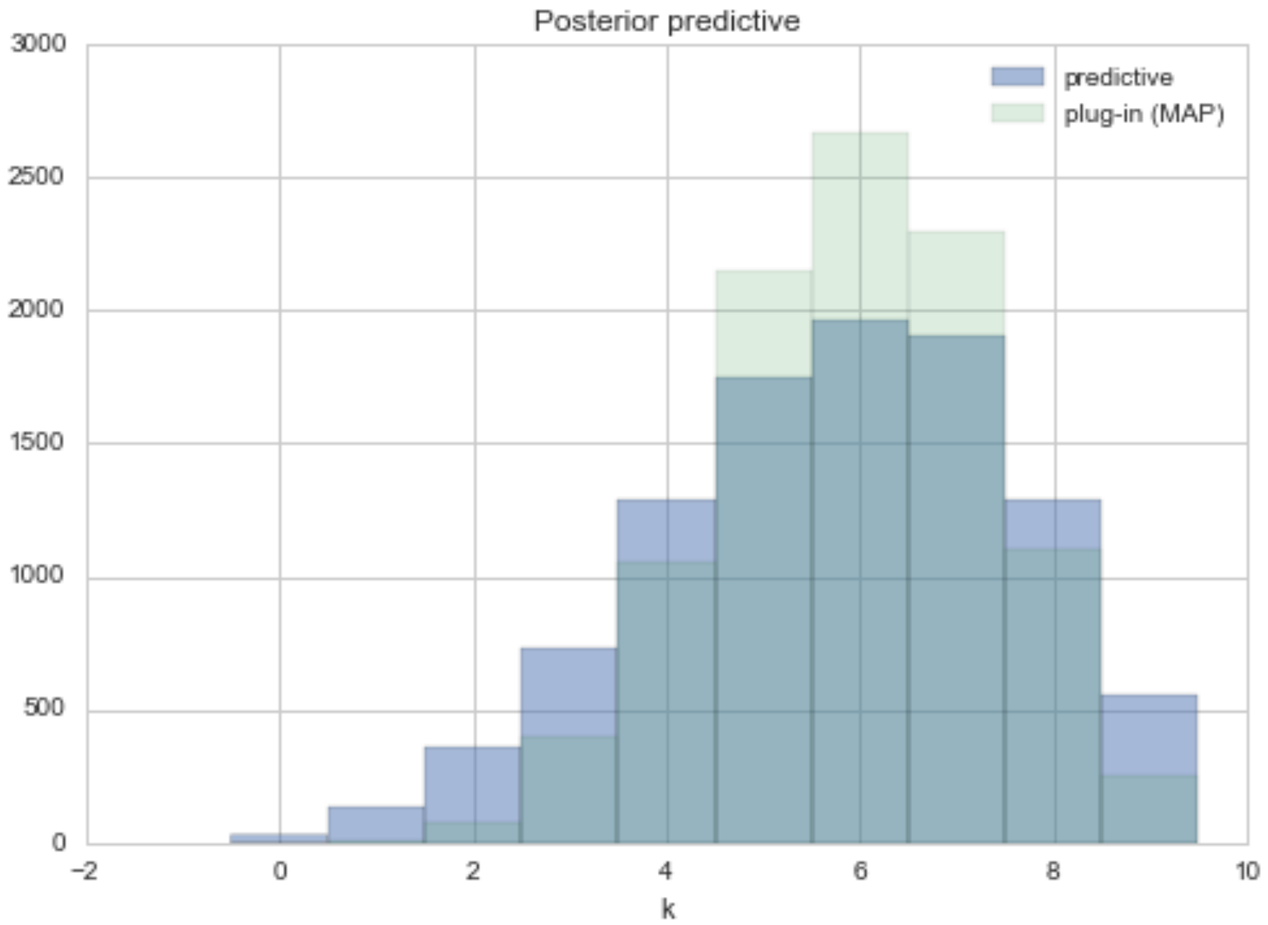


Also see textbook ch 4

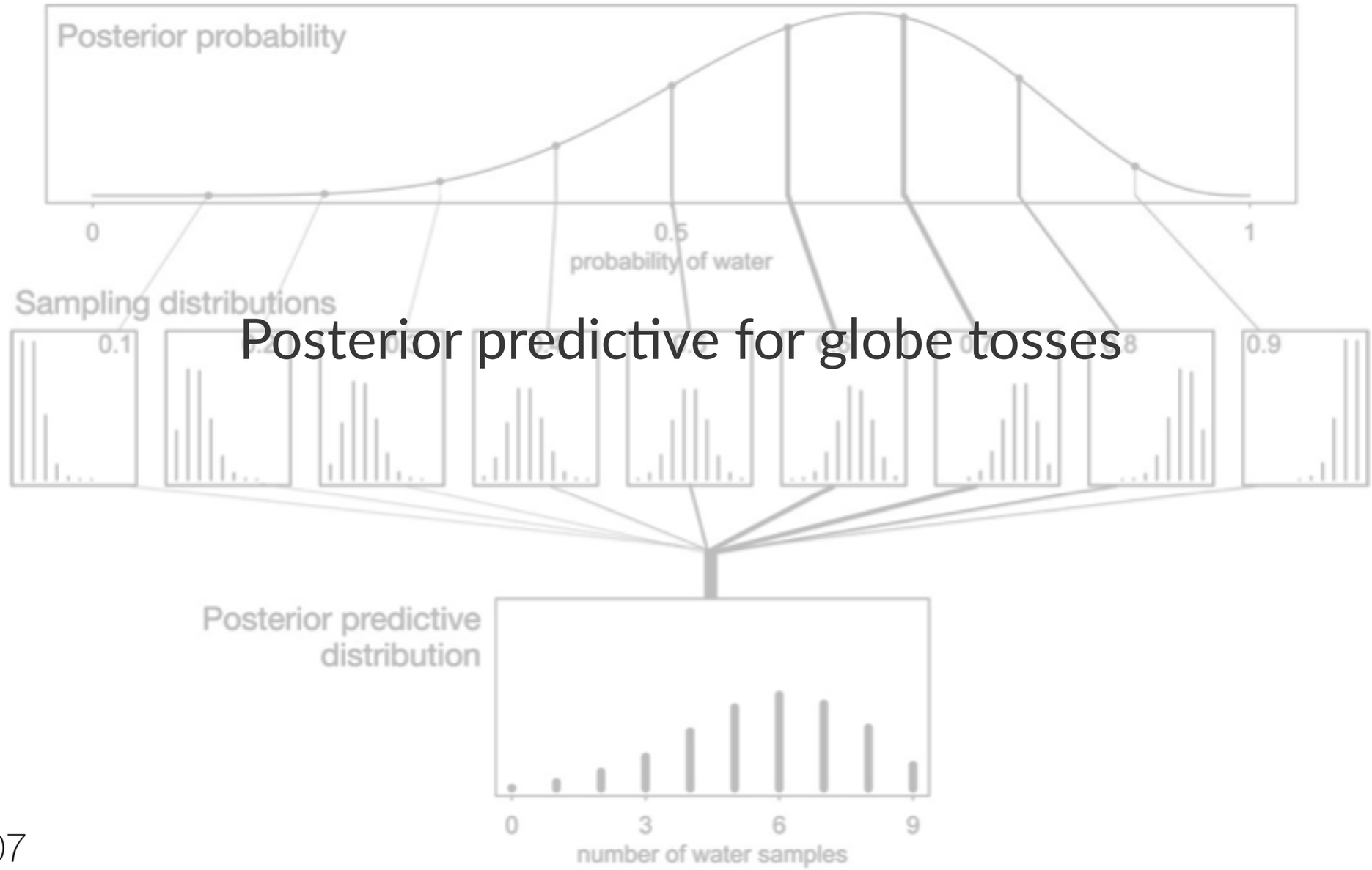
Posterior predictive from sampling

- first draw the thetas from the posterior
- then draw y's from the likelihood
- and histogram the likelihood
- these are draws from joint y, θ

```
postpred = np.random.binomial( len(data), samples);
```



Posterior predictive for globe tosses



Sufficient Statistics and the exponential family

$$p(y_i|\theta) = f(y_i)g(\theta)e^{\phi(\theta)^T u(y_i)}.$$

Likelihood: $p(y|\theta) = \left(\prod_{i=1}^{n} f(y_i)\right) g(\theta)^n \ \exp\left(\phi(\theta)\sum_{i=1}^{n} u(y_i)\right)$

$\sum_{i=1}^{n} u(y_i)$ is said to be a **sufficient statistic** for θ

Poisson Gamma Example

The data consists of 155 women who were 40 years old. We are interested in the birth rate of women with a college degree and women without. We are told that 111 women without college degrees have 217 children, while 44 women with college degrees have 66 children.

Let $Y_{1,1}, \ldots, Y_{n_1,1}$ children for the n_1 women without college degrees, and $Y_{1,2}, \ldots, Y_{n_2,2}$ for n_2 women with college degrees.

Exchangeability

Lets assume that the number of children of a women in any one of these classes can me modelled as coming from ONE birth rate.

The in-class likelihood for these women is invariant to a permutation of variables.

This is really a statement about what is IID and what is not.

It depends on how much knowledge you have...

Poisson likelihood

$$Y_{i,1} \sim Poisson(\theta_1), Y_{i,2} \sim Poisson(\theta_2)$$

$$p(Y_{1,1}, \ldots, Y_{n_1,1}|\theta_1) = \prod_{i=1}^{n_1} p(Y_{i,1}|\theta_1) = \prod_{i=1}^{n_1} \frac{1}{Y_{i,1}!} \theta_1^{Y_{i,1}} e^{-\theta_1}$$

$$= c(Y_{1,1}, \ldots, Y_{n_1,1}) \ (n_1\theta_1)^{\sum Y_{i,1}} e^{-n_1\theta_1} \sim Poisson(n_1\theta_1)$$

$$Y_{1,2}, \ldots, Y_{n_1,2}|\theta_2 \sim Poisson(n_2\theta_2)$$

Posterior

$$c_1(n_1, y_1, \ldots, y_{n_1})\ (n_1\theta_1)^{\sum Y_{i,1}} e^{-n_1\theta_1}\ p(\theta_1) \times c_2(n_2, y_1, \ldots, y_{n_2})\ (n_2\theta_2)^{\sum Y_{i,2}} e^{-n_2\theta_2}\ p(\theta_2)$$

$\sum Y_i$, total number of children in each class of mom, is **sufficient statistics**

Conjugate prior

Sampling distribution for θ: $p(Y_1, \ldots, y_n|\theta) \sim \theta^{\sum Y_i} e^{-n\theta}$

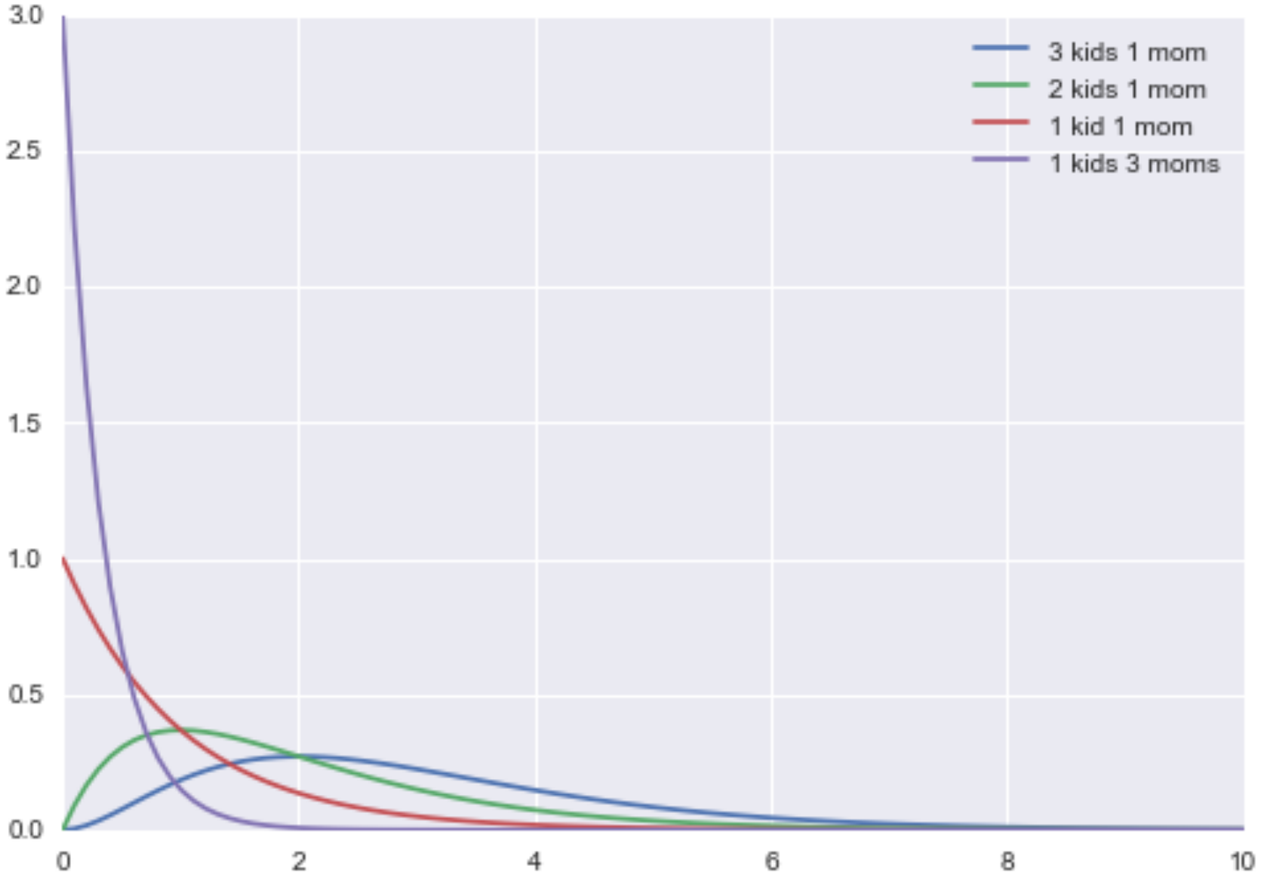
Form is of $Gamma$. In shape-rate parametrization (wikipedia)

$$p(\theta) = \mathrm{Gamma}(\theta, \mathrm{a}, \mathrm{b}) = \frac{\mathrm{b}^{\mathrm{a}}}{\Gamma(\mathrm{a})}\theta^{\mathrm{a}-1}\mathrm{e}^{-\mathrm{b}\theta}$$

Posterior:

$$p(\theta|Y_1, \ldots, Y_n) \propto p(Y_1, \ldots, y_n|\theta)p(\theta) \sim \mathrm{Gamma}(\theta, \mathrm{a} + \sum \mathrm{Y_i}, \mathrm{b} + \mathrm{n})$$

Priors and Posteriors



We choose 2,1 as our prior.

$$p(\theta_1|n_1, \sum_i^{n_1} Y_{i,1}) \sim \text{Gamma}(\theta_1, 219, 112)$$

$$p(\theta_2|n_2, \sum_i^{n_2} Y_{i,2}) \sim \text{Gamma}(\theta_2, 68, 45)$$

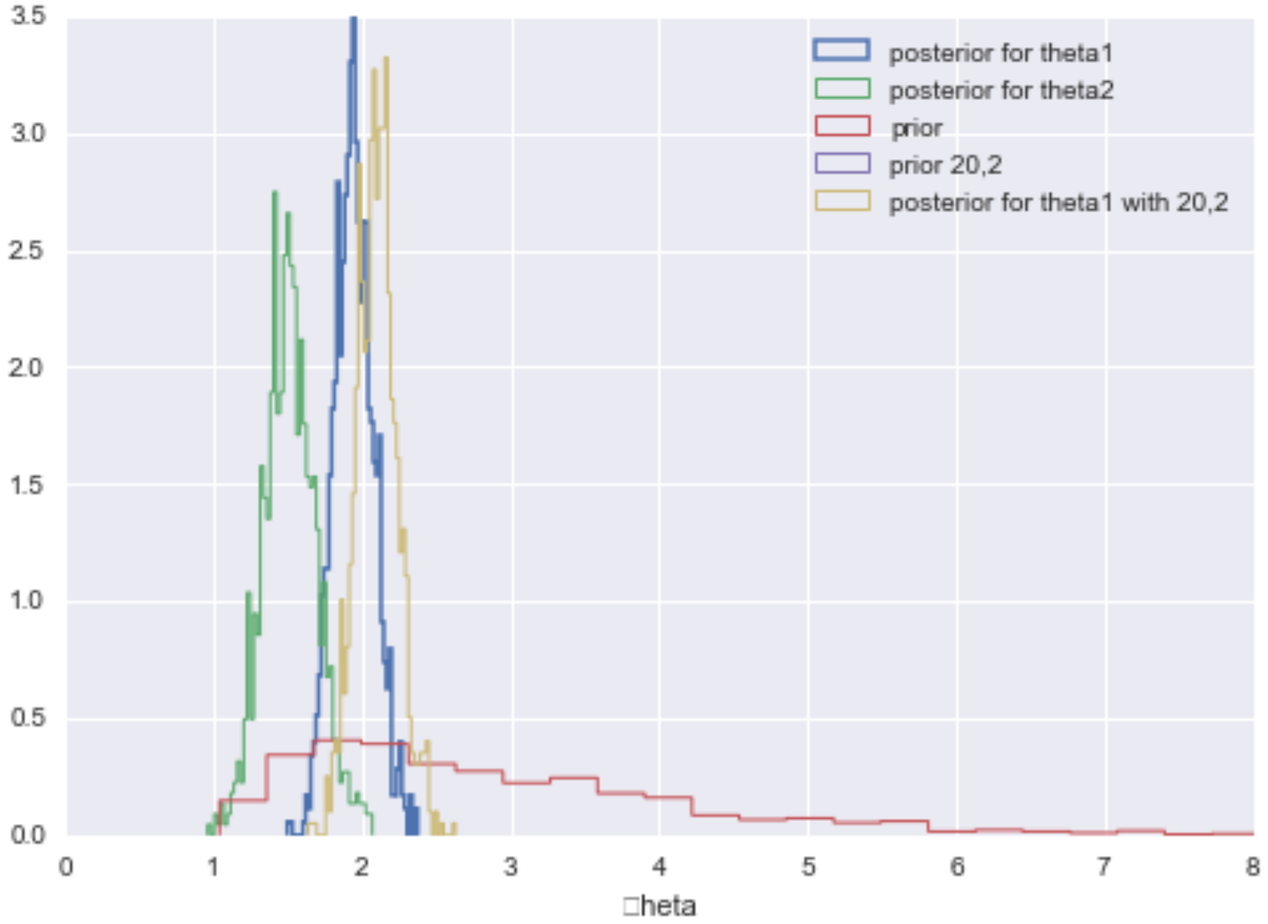
Prior mean, variance:
$E[\theta] = a/b, var[\theta] = a/b^2.$

Posteriors

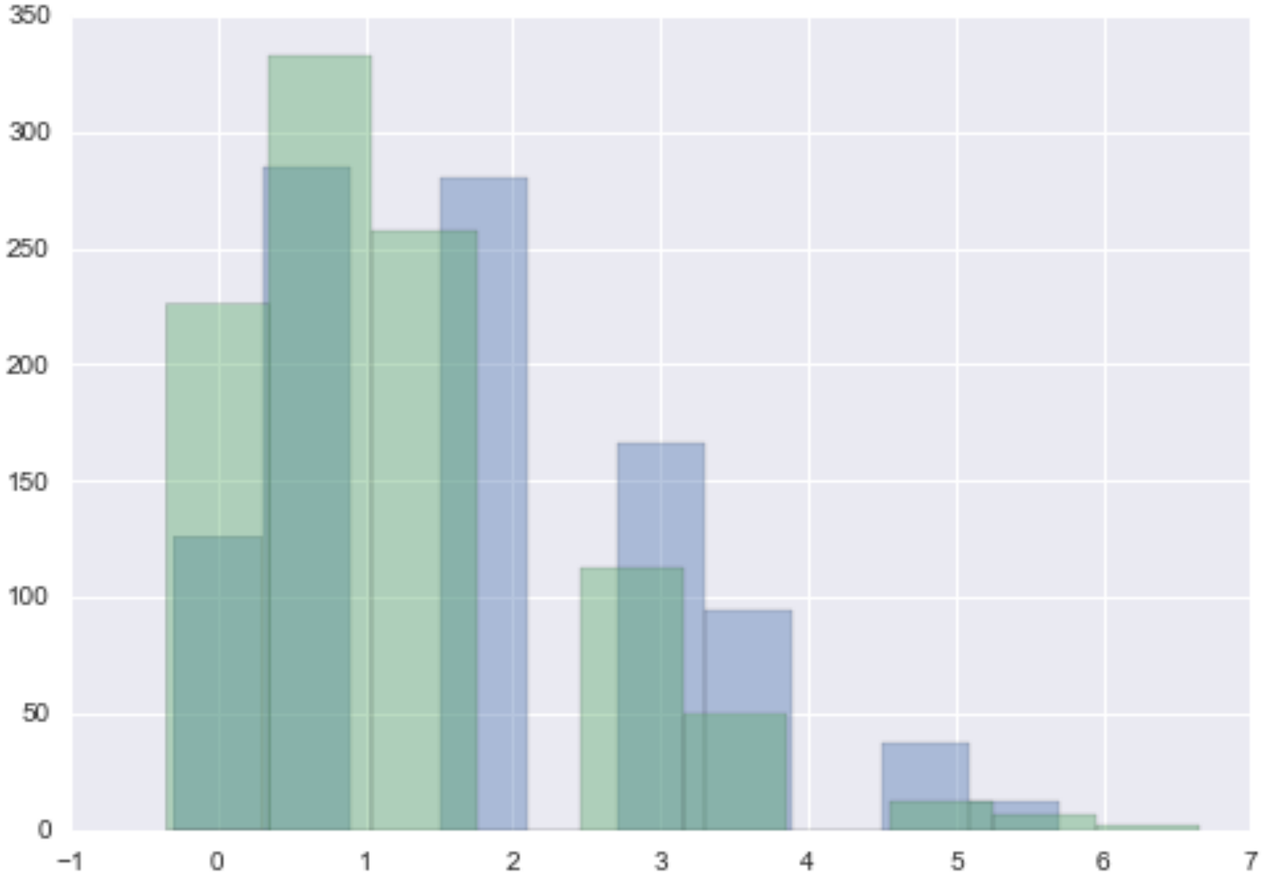
$$E[\theta] = (a + \sum y_i)/(b + N)$$
$$var[\theta] = (a + \sum y_i)/(b + N)^2.$$

```
np.mean(theta1), np.var(theta1)
= (1.9516881521791478,
0.018527204185785785)

np.mean(theta2), np.var(theta2)
= (1.5037252100213609,
0.034220717257786061)
```



Posterior Predictives



$$p(y^*|D) = \int d\theta p(y^*|\theta)p(\theta|D)$$

Sampling makes it easy:

```
postpred1 = poisson.rvs(theta1)
postpred2 = poisson.rvs(theta2)
```

Negative Binomial:

$$E[y^*] = \frac{(a + \sum y_i)}{(b+N)}$$

$$var[y^*] = \frac{(a + \sum y_i)}{(b+N)^2}(N + b + 1).$$

But see width:

```
np.mean(postpred1), np.var(postpred1)=(1.976,
1.8554239999999997)
```

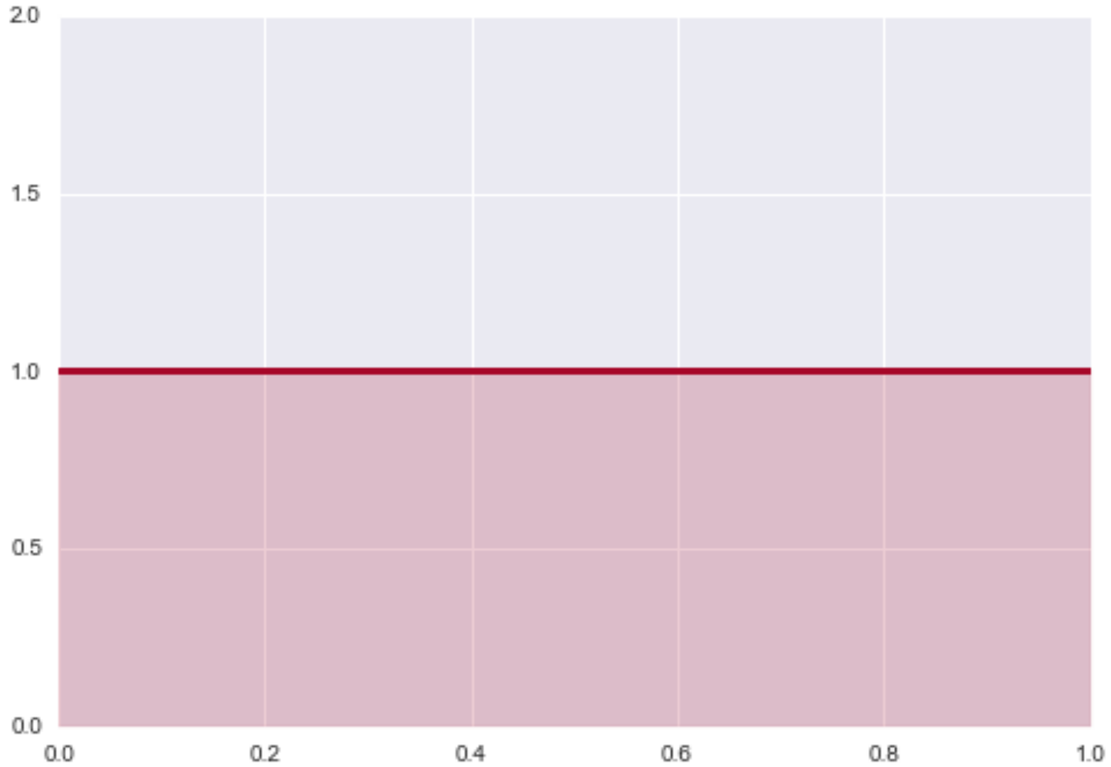
Posterior predictive smears out posterior error with sampling distribution

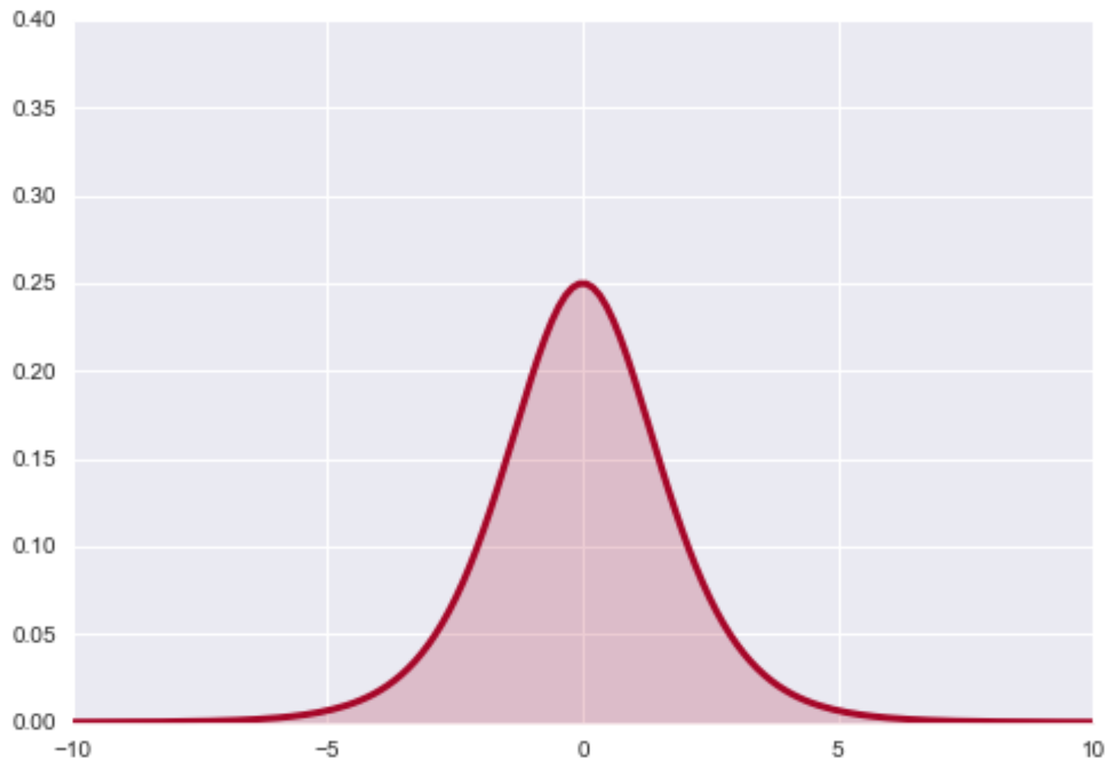
- use for making predictions
- use for model checking using cross-validation; also for data visualization

What Priors?

- we'll ask this question throughout the course
- also see https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations
- choose something reasonable, and then spread it out some

Uninformative priors on location





- used transform $psi = log(\frac{\theta}{1-\theta})$ and then $dP_\psi = dP_\theta$. Shape comes in through jacobian.
- despite transformation change, flat priors still used for location priors
- may even be improper, ie integrate to ∞ as long as posterior integral is finite
- e.g. flat prior on mean in normal-normal model with strong likelihood.

Jeffreys prior

noninformative prior on scale variables $p_J(\theta) \propto \mathbf{I}(\theta)^{1/2}$

where

$$\mathbf{I}(\theta) = det(-E\left[\frac{d^2 \log p(X|\theta)}{d\theta_i \theta_j}\right])$$

is the Fisher Information, and expectation is with respect to the likelihood.

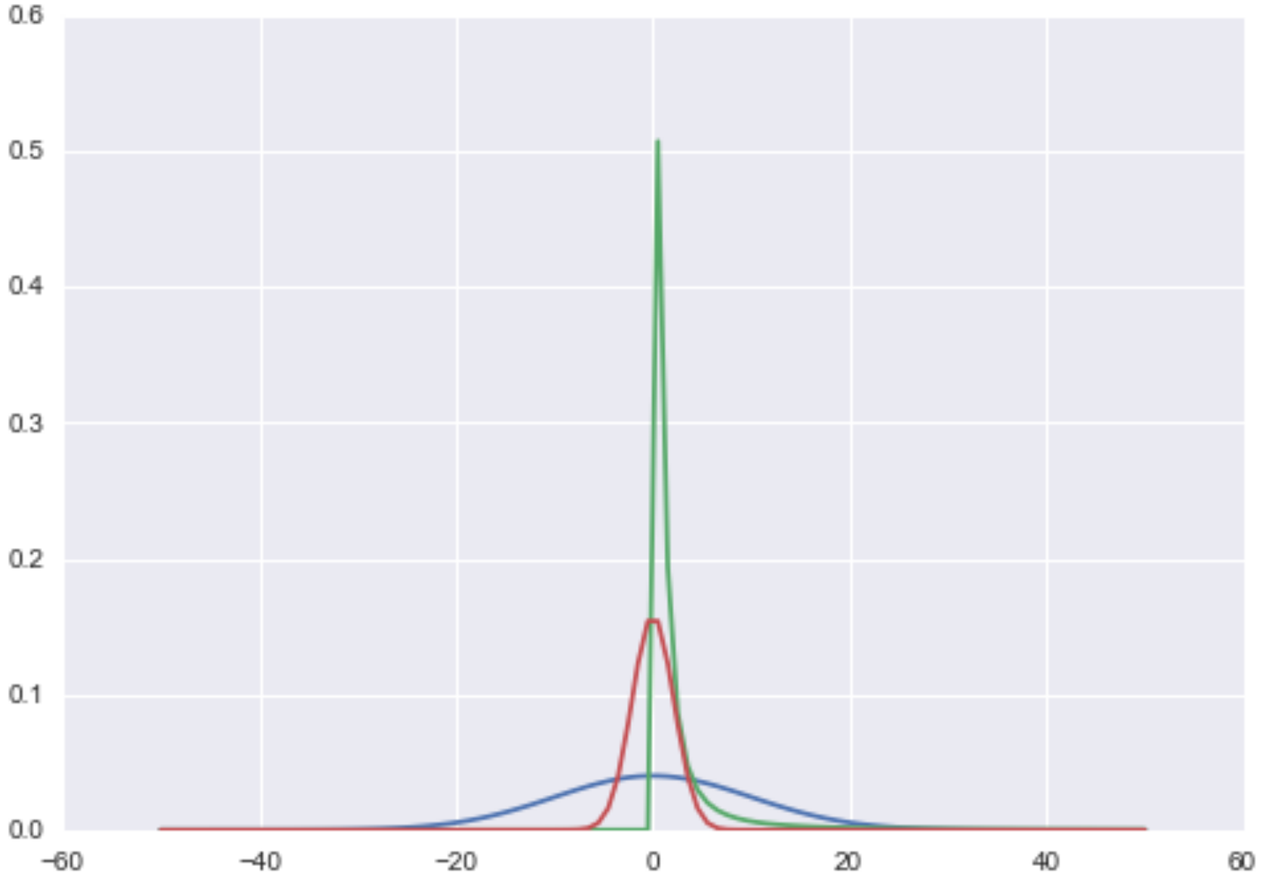
Weakly informative or regularizing priors

- these are the priors we will concern ourselves most with
- restrict parameter ranges
- help samplers
- regularizing priors may use the data "twice" as we shall see

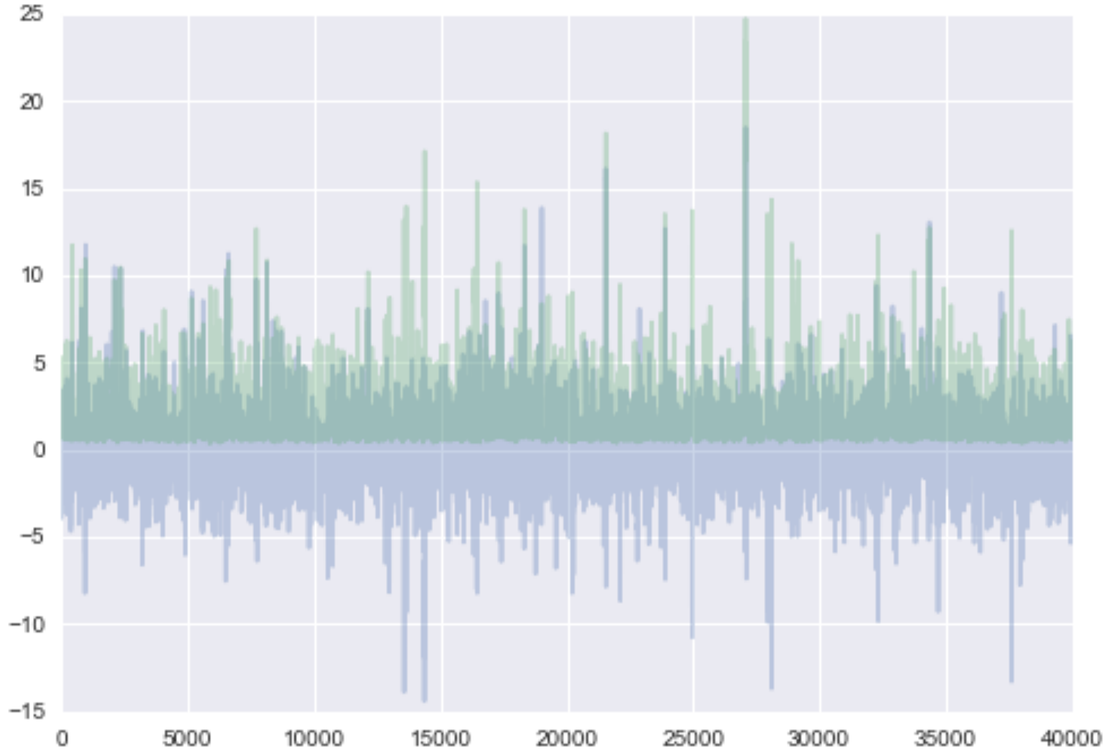
Normal model Example

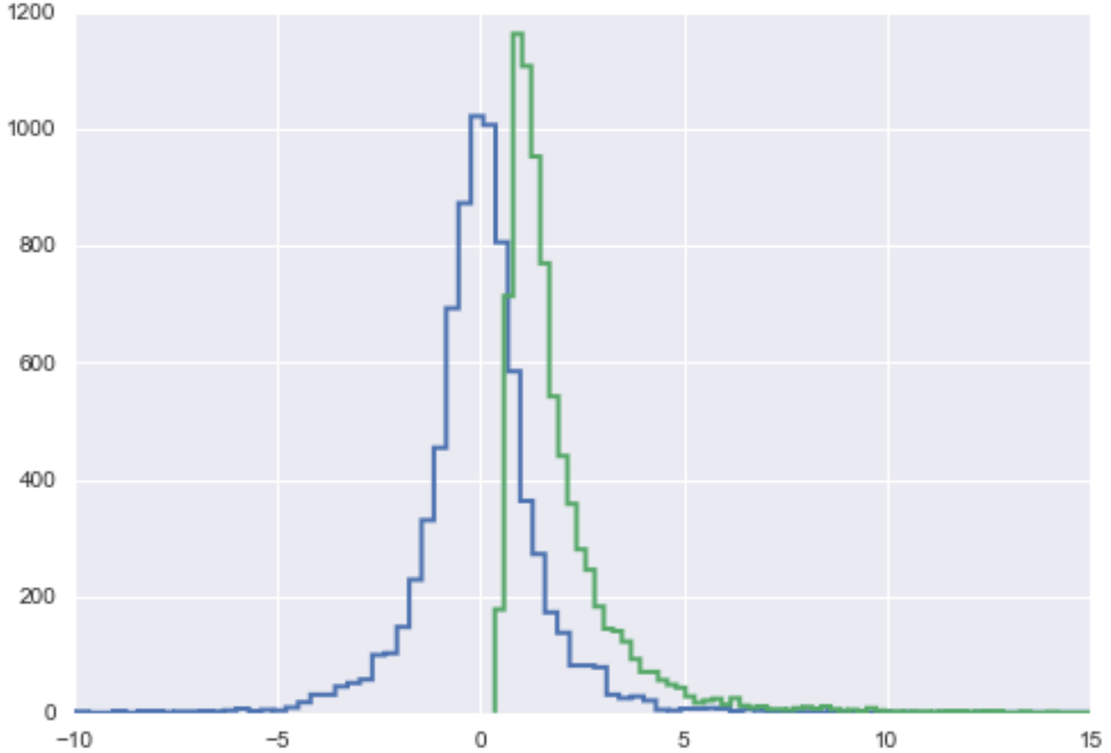
- two data points 1 and -1
- flat improper priors on $\mu, \sigma > 0$
- model drifts wildly as less data
- flat priors say extreme implausible values quite likely
- extreme drifts overwhelm chain

weakly regularizing priors



- choose $\mu \sum N(0, 10)$
- choose $\sigma \sim HalfCauchy(0, 1)$
- lets mean vary widely but not crazily
- HalfCauchy lets variance be positive and occasionally can have high value samples

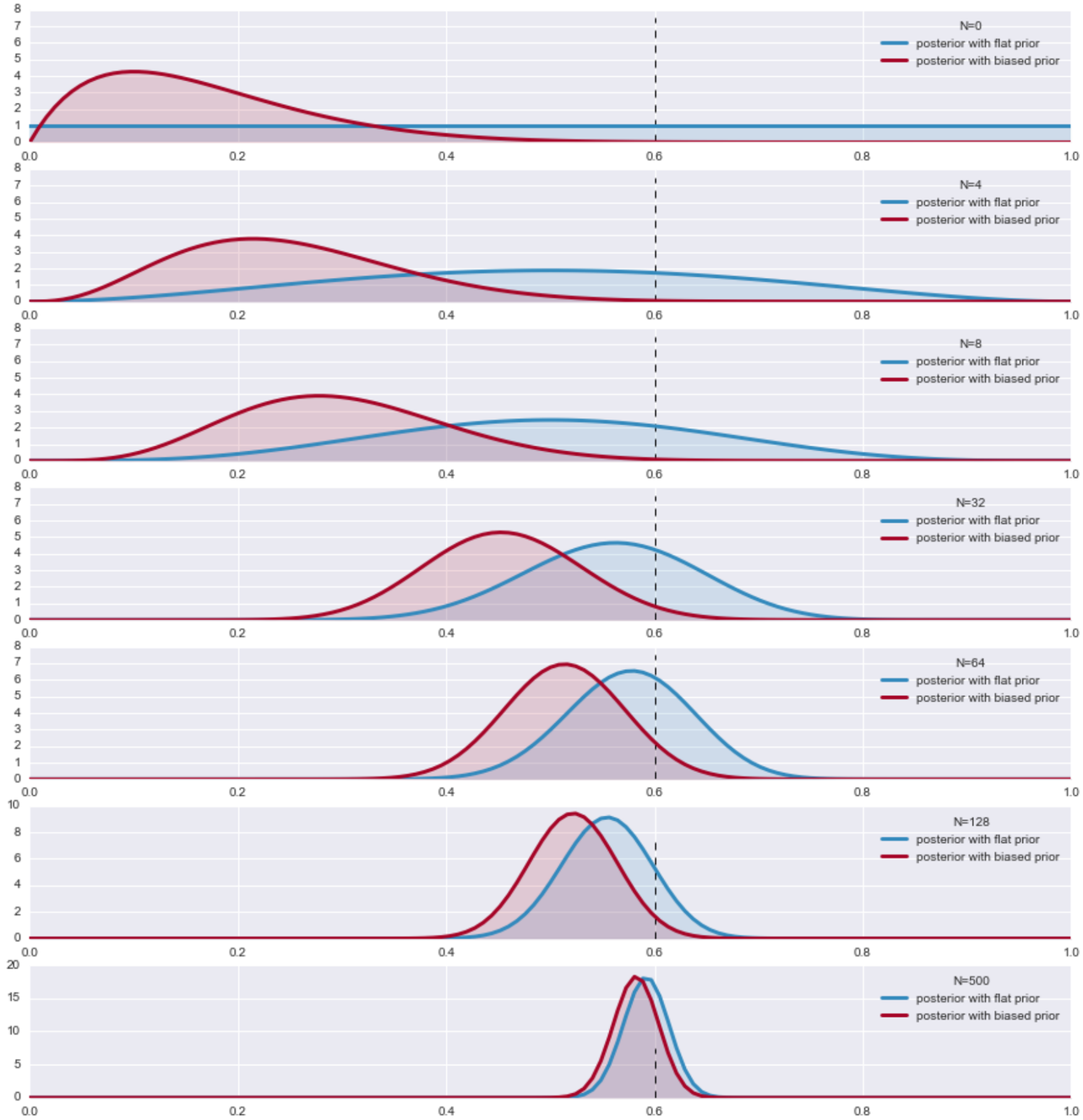

25
20
15
10
5
0
−5
−10
−15
0
5000
10000
15000
20000
25000
30000
35000
40000

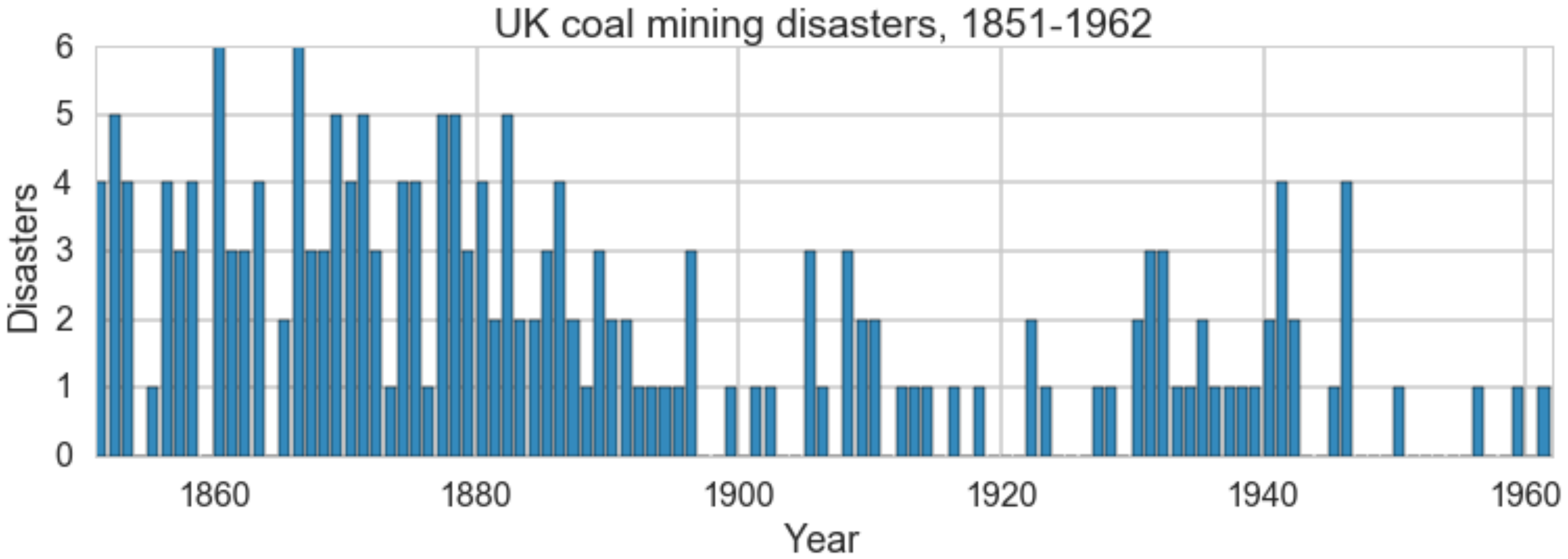

1200
1000
800
600
400
200
0
−10
−5
0
5
10
15

Other priors

- KL Maximization non-informative prior by Bernardo
- Maximum Entropy prior when some assumptions but no more..
- Empirical bayes prior: usee data! in hierarchical models

Data overwhelms prior




UK coal mining disasters, 1851-1962
Disasters
0
1
2
3
4
5
6
1860
1880
1900
1920
1940
1960
Year

Model

$$y|\tau, \lambda_1, \lambda_2 \sim Poisson(r_t)$$

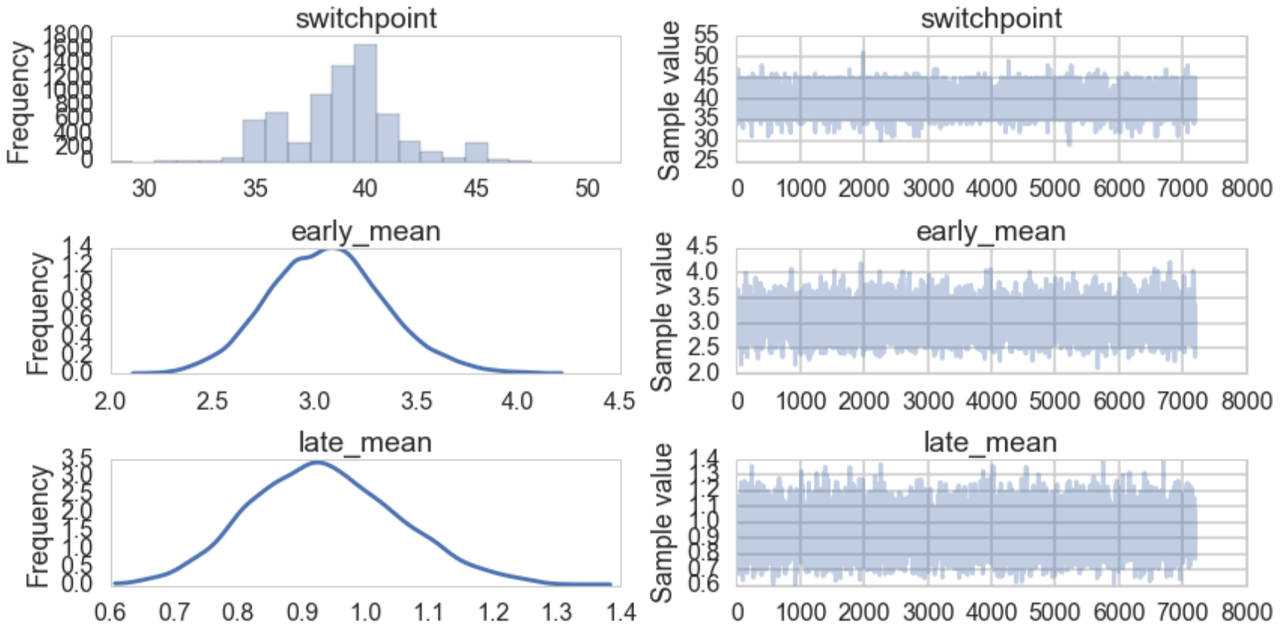
$$r_t = \lambda_1 \text{ if } t < \tau \text{ else } \lambda_2 \text{ for } t \in [t_l, t_h]$$
$$\tau \sim DiscreteUniform(t_l, t_h)$$

$$\lambda_1 \sim Exp(a)$$
$$\lambda_2 \sim Exp(b)$$

```
from pymc3.math import switch
with pm.Model() as coaldis1:
    early_mean = pm.Exponential('early_mean', 1)
    late_mean = pm.Exponential('late_mean', 1)
    switchpoint = pm.DiscreteUniform('switchpoint', lower=0, upper=n_years)
    rate = switch(switchpoint >= np.arange(n_years), early_mean, late_mean)
    disasters = pm.Poisson('disasters', mu=rate, observed=disasters_data)
```

```
with coaldis1:
    stepper=pm.Metropolis()
    trace = pm.sample(40000, step=stepper)
```

```
100%|██████████| 40000/40000 [00:12<00:00, 3326.53it/s] | 229/40000 [00:00<00:17, 2289.39it/s]
```



```
>>>coaldis1.vars #stochastics
[early_mean_log_, late_mean_log_, switchpoint]
>>>coaldis1.deterministics #deterministics
[early_mean, late_mean]
>>>coaldis1.observed_RVs
[disasters]
>>>ed=pm.Exponential.dist(1)
<class 'pymc3.distributions.continuous.Exponential'>
>>>ed.random(size=10)
array([ 1.18512233,  2.45533355,  0.04187961,  3.32967837,  0.0268889 ,
        0.29723148,  1.30670324,  0.23335826,  0.56203427,  0.15627659])
>>>type(switchpoint), type(early_mean)
(pymc3.model.FreeRV, pymc3.model.TransformedRV)
>>>switchpoint.logp({'switchpoint':55,
    'early_mean_log_':1, 'late_mean_log_':1})
array(-4.71849887129509 4)
```

- **Blockwise Updating** in which we simply use a 2D-proposal function like a 2-D gaussian. Simple and you can make diagonal moves, but the disadvantage to this is that it can take a very long time to cover the space.
- **Componentwise Updating**. Steps only in one dimension at a time. You then accept or not, and repeat. The advantage is that you can make big strides. The disadvantage is that you may sample only in one axis for a bit, but this evens out in the long run.

Imputation

```
>>>disasters_missing = np.array([ 4, 5, 4, 0, 1, 4, 3, 4, 0, 6, 3, 3, 4, 0, 2, 6,
3, 3, 5, 4, 5, 3, 1, 4, 4, 1, 5, 5, 3, 4, 2, 5,
2, 2, 3, 4, 2, 1, 3, -999, 2, 1, 1, 1, 1, 3, 0, 0,
1, 0, 1, 1, 0, 0, 3, 1, 0, 3, 2, 2, 0, 1, 1, 1,
0, 1, 0, 1, 0, 0, 0, 2, 1, 0, 0, 0, 1, 1, 0, 2,
3, 3, 1, -999, 2, 1, 1, 1, 1, 2, 4, 2, 0, 0, 1, 4,
0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1])
>>>disasters_masked = np.ma.masked_values(disasters_missing, value=-999)
```

An array with mask set to True where data is missing.

```python
with pm.Model() as missing_data_model:
    switchpoint = pm.DiscreteUniform('switchpoint', lower=0, upper=len(disasters_masked))
    early_mean = pm.Exponential('early_mean', lam=1.)
    late_mean = pm.Exponential('late_mean', lam=1.)
    idx = np.arange(len(disasters_masked))
    rate = pm.Deterministic('rate', switch(switchpoint >= idx, early_mean, late_mean))
    disasters = pm.Poisson('disasters', rate, observed=disasters_masked)

with missing_data_model:
    stepper=pm.Metropolis()
    trace_missing = pm.sample(10000, step=stepper)

pm.summary(trace_missing, varnames=['disasters_missing'])
```

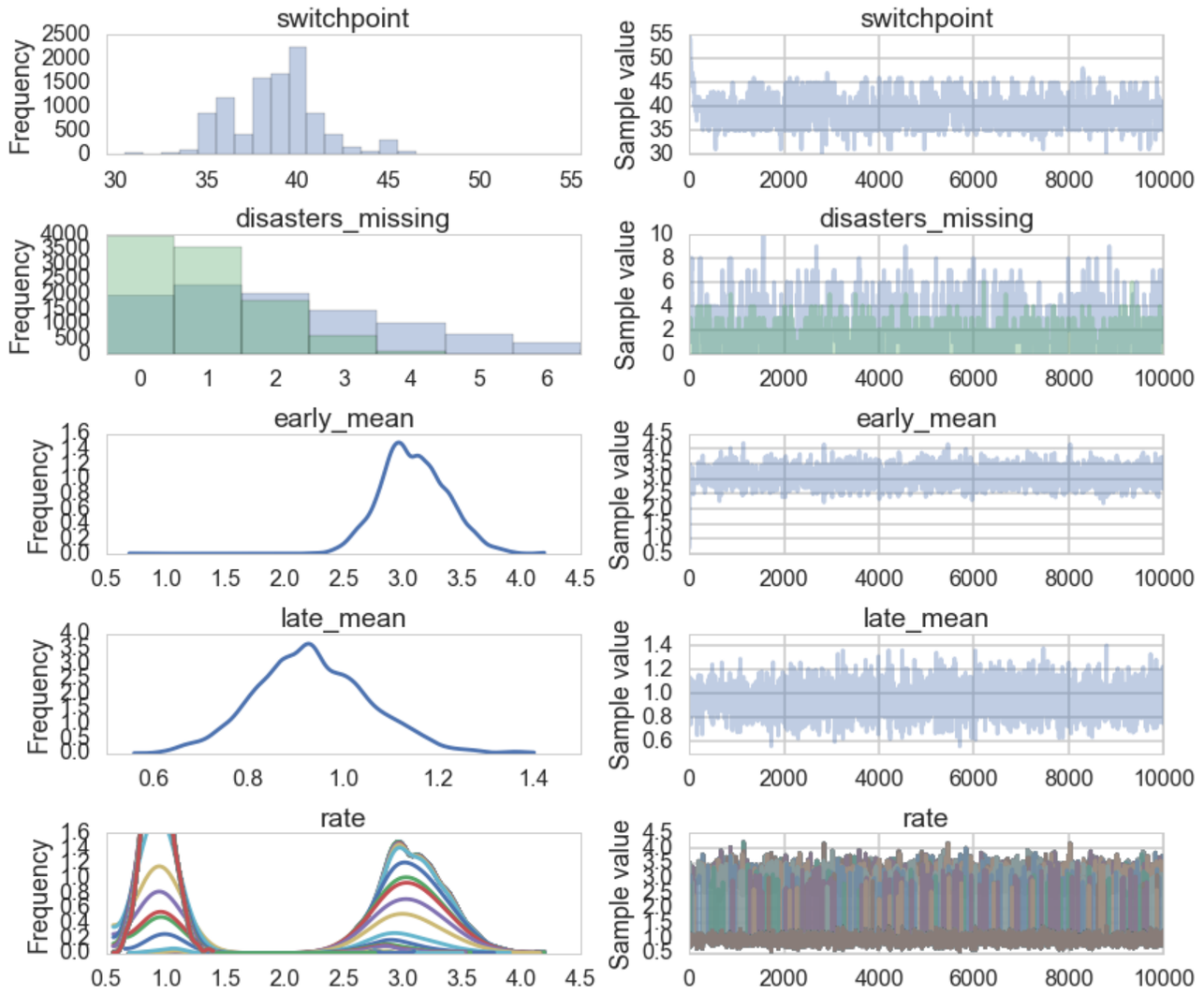
```
disasters_missing:

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


  2.189            1.825            0.078            [0.000, 6.000]
  0.950            0.980            0.028            [0.000, 3.000]

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

  0.000            1.000            2.000            3.000            6.000
  0.000            0.000            1.000            2.000            3.000
```

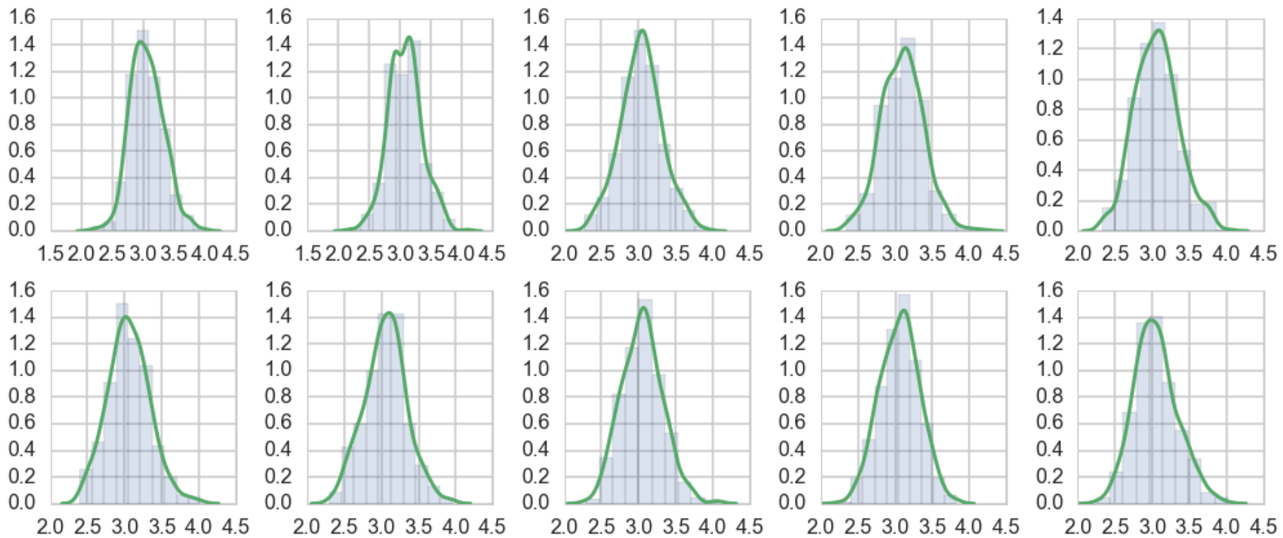


Model convergence

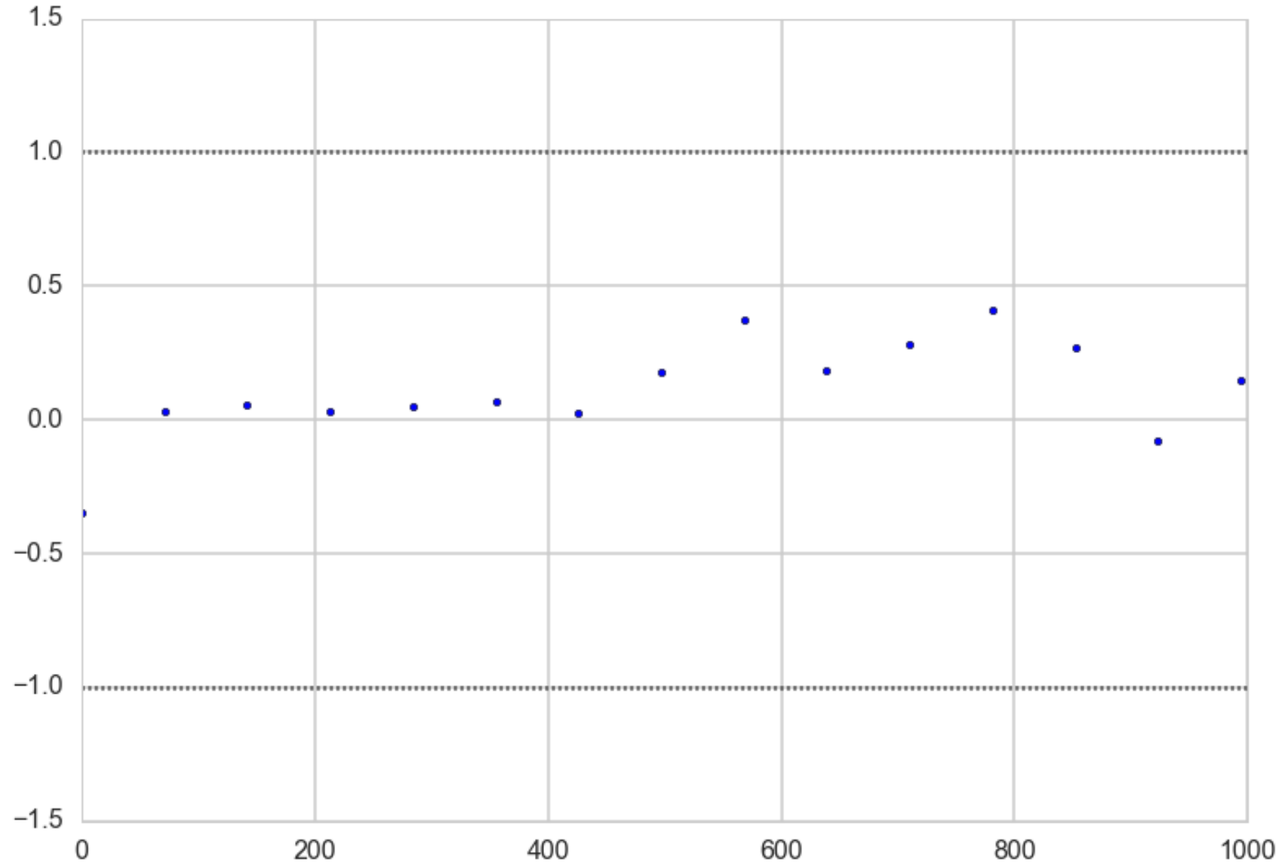
- traces white noisy
- diagnose autocorrelation, check parameter correlations

```
pm.trace_to_dataframe(trace).corr()
```

- visually inspect histogram every m samples
- traceplots from different starting points, different chains
- formal tests: Gewecke, Gelman-Rubin, Effective Sample Size



Gewecke: difference of means



$$H_0 : \mu_{\theta_1} - \mu_{\theta_2} = 0 \implies \mu_{\theta_1 - \theta_2} = 0$$

$$\sigma_{\theta_1 - \theta_2} = \sqrt{\frac{var(\theta_1)}{n_1} + \frac{var(\theta_2)}{n_2}}$$

$$|\mu_{\theta_1} - \mu_{\theta_2}| < 2\sigma_{\theta_1 - \theta_2}$$

```
with coaldis1:
    stepper=pm.Metropolis()
    tr = pm.sample(2000, step=stepper)

z = geweke(tr, intervals=15)

plt.scatter(*z['early_mean'].T)
plt.hlines([-1,1], 0, 1000, linestyles='dotted')
plt.xlim(0, 1000)
```

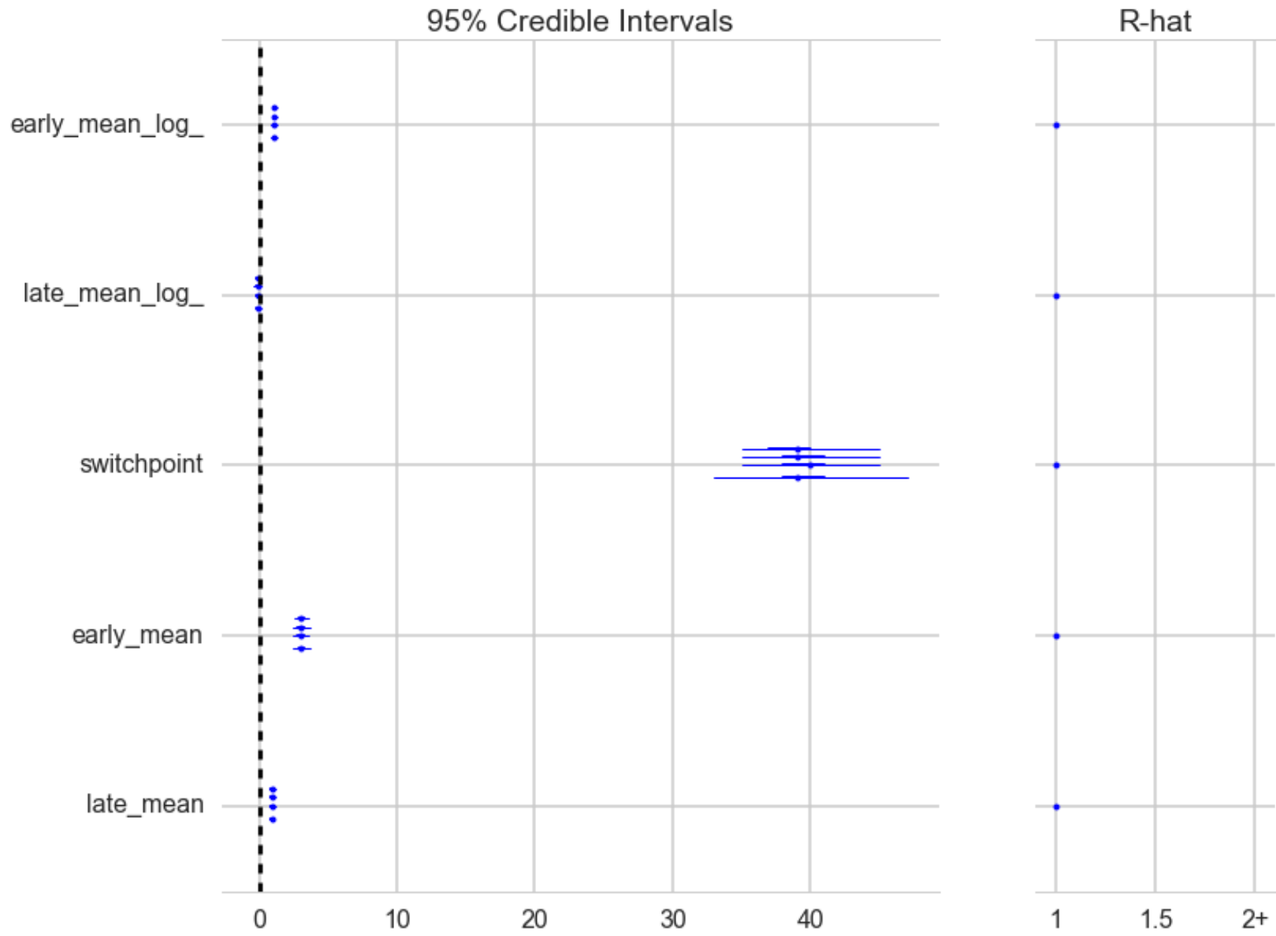
Gelman-Rubin

Multiple chains..compute within chain variance and compare to between chain variance

$$s_j^2 = \frac{1}{n-1}\sum_i (\theta_{ij} - \mu_{\theta_j})^2$$

$$w = \frac{1}{m}\sum_j s_j^2; \;\; \mu = \frac{1}{m}\sum_j \mu_{\theta_j}$$

$$B = \frac{n}{m-1}\sum_j (\mu_{\theta_j} - \mu)^2$$



Use weighted average of w and B to estimate variance of the stationary distribution `pm.gelman_rubin(trace)`:

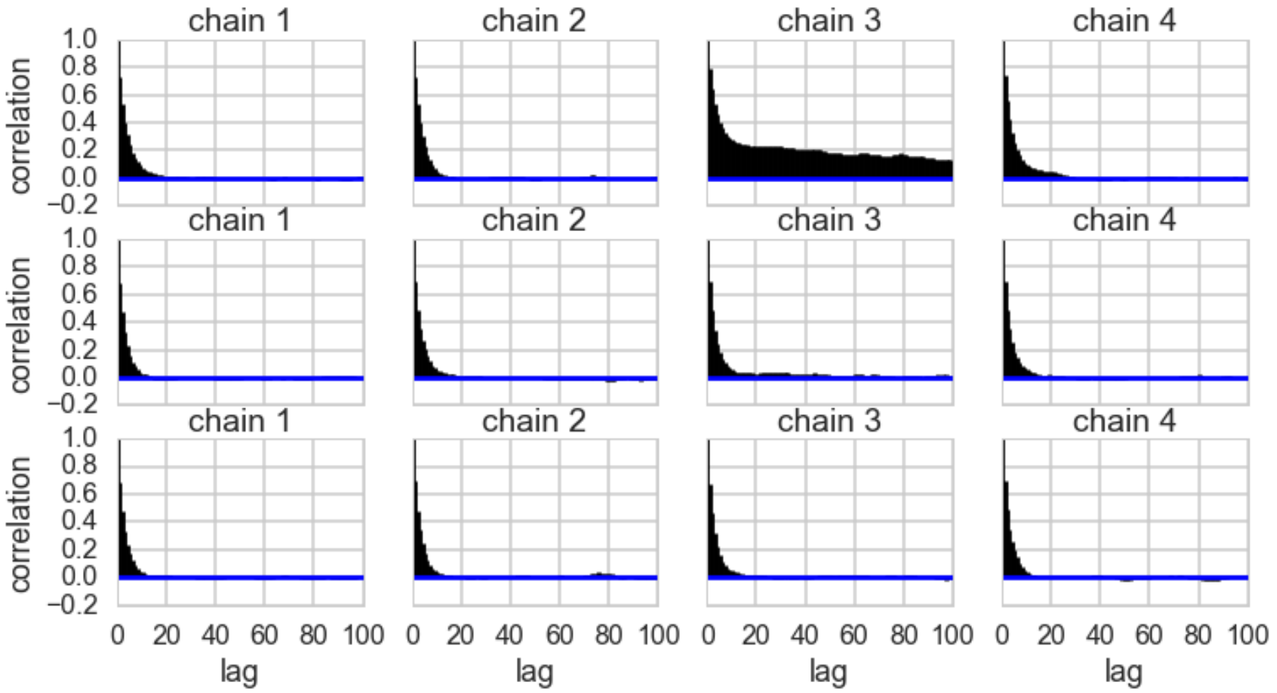
$$\hat{Var}(\theta) = (1 - \frac{1}{n})w + \frac{1}{n}B$$

Overestimates our variance, but unbiased under stationarity.

Ratio of the estimated distribution variance to asymptotic one:

$$\hat{R} = \sqrt{\frac{\hat{Var}(\theta)}{w}}$$

ESS: Effective Sample Size



IIDness of draws decreases

```
pm.effective_n(trace)
```

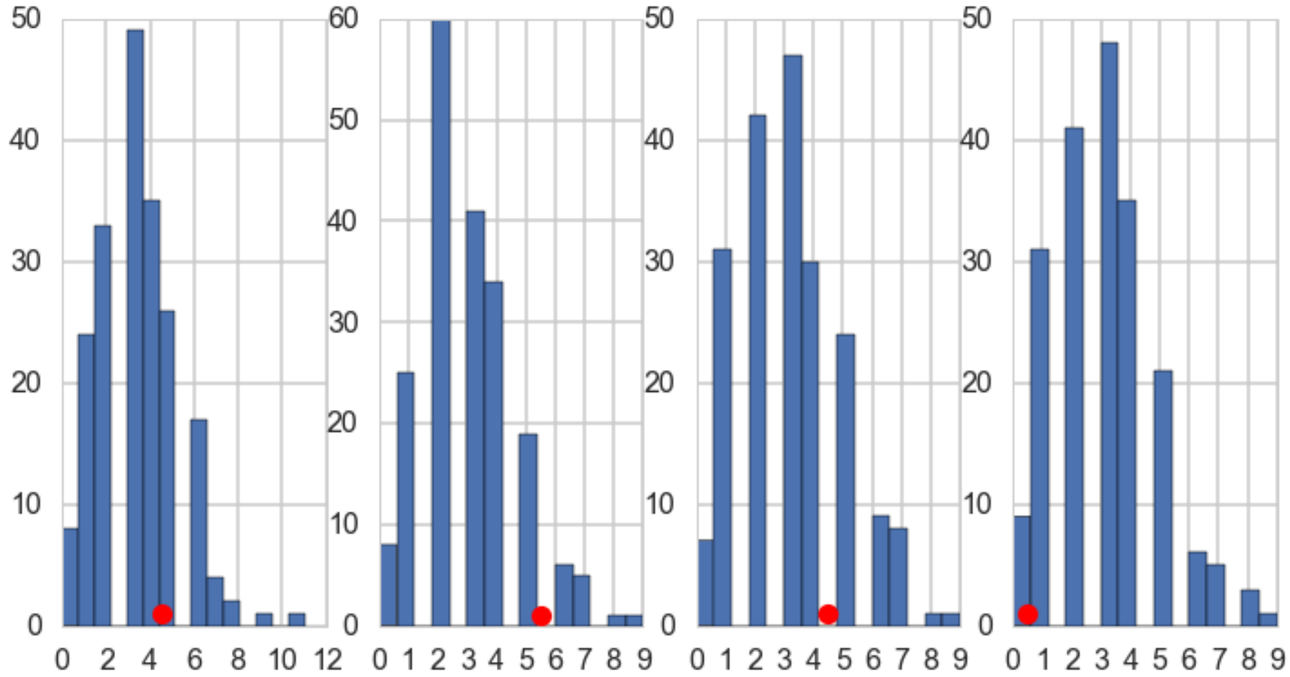
```
{'early_mean': 16857.0,
 'early_mean_log_': 12004.0,
 'late_mean': 27344.0,
 'late_mean_log_': 27195.0,
 'switchpoint': 195.0}
```

(40000 samples)

$$n_{eff} = \frac{mn}{1 + 2\sum_{\Delta t} \rho_{\Delta t}}$$

Posterior Predictive Checks

```
with coaldis1:
    sim = pm.sample_ppc(t2, samples=200)
```



Non-Identifiability

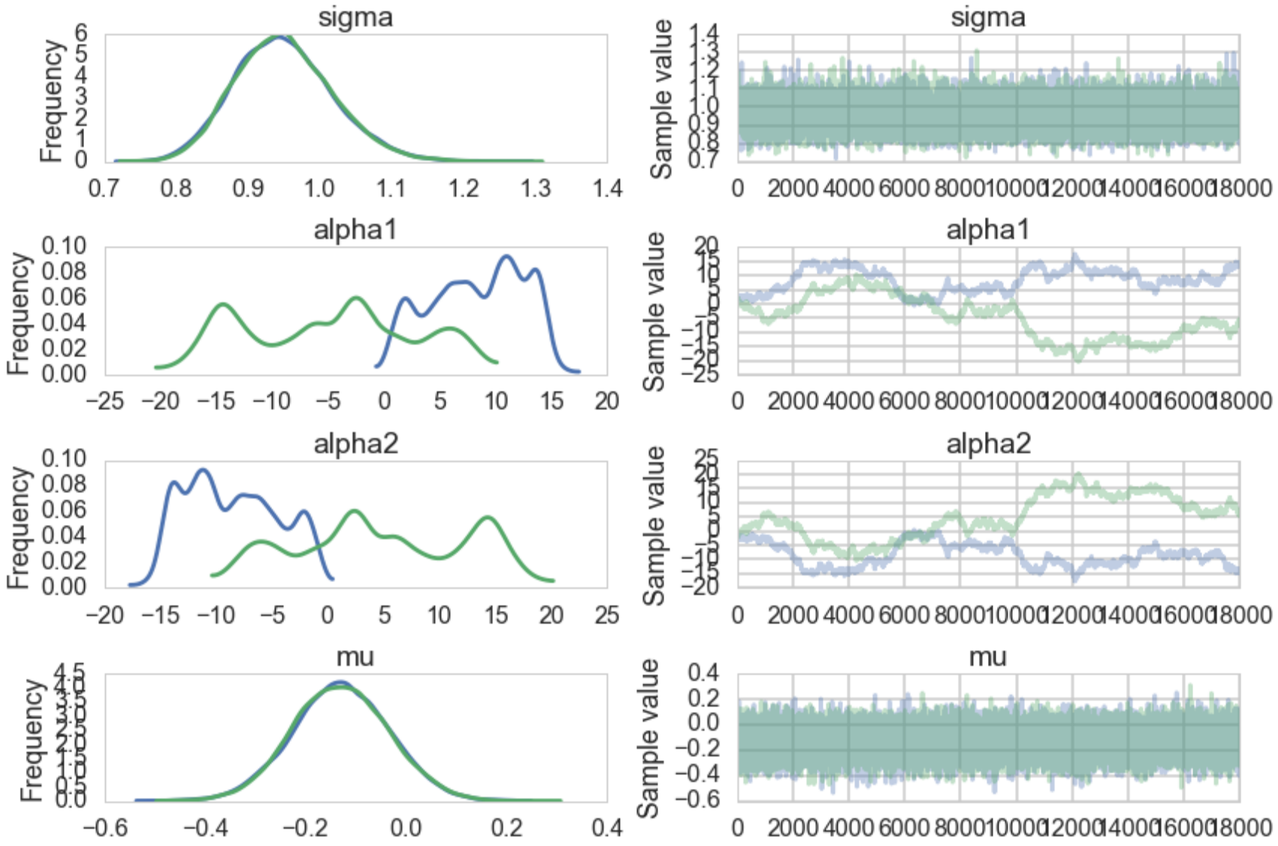
Generate data from N(0,1). Then fit:

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

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

$$\begin{aligned} \alpha_1 &\sim Unif(-\infty, \infty) \\ \alpha_2 &\sim Unif(-\infty, \infty) \\ \sigma &\sim HalfCauchy(0, 1) \end{aligned}$$

Correlation diagnostic




```
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)
```

```
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': 0.9999971018206269 5,
  'sigma': 1.0000248056117549,
  'sigma_log_': 1.0000261752214563}
```

Attempt to fix

```
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