Lecture 12

Bayesian Stats

Last time: Metropolis, MH

- stationarity and ergodicity
- metropolis
- dont rejection sample
- MH uses asymmetric proposals
- tuning width up decreases acceptance, down increases acceptance

some rules of thumb

- want acceptance at about 30-40%
- want autocorrelation low, traceplots to look like white noise
- perform burnin (get to stationarity) and perhps thinning (reduce autocorrelation if you need to save space, generally its not needed; with thinning you will need a longer chain)

Last time: 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: $posterior = \frac{likelihood \times prior}{evidence}$

Bayesian, contd.

- evidence: $p(y) = E_{p(\theta)}[\mathcal{L}] = \int d\theta p(y|\theta)p(\theta)$ a normalization, irrelevant for sampling

Today

- discrete sampling
- Bayes revisited and the normal normal model through sampling
- the posterior predictive and decision theory
- Bayesian workflow in the macro
- conjugate priors
- sufficient statistics, exchangeability and the poisson-gamma

Discrete distribution MCMC

- proposal distribution becomes proposal matrix
- index the discrete outcomes
- can use symmetric or asymmetric proposal as long as rows sum to 1
- **make sure proposal matrix is irreducible**: ie you can get from any index to any other one.

Rain-Sun transition matrix

```
transition_matrix = np.array([[0.3, 0.7],[0.5, 0.5]])
print(np.linalg.matrix_power(transition_matrix,10))
```

```
The transition matrix
[[ 0.3  0.7]
 [ 0.5  0.5]]
Stationary distribution
[[ 0.41666673  0.58333327]
 [ 0.41666662  0.58333338]]
```

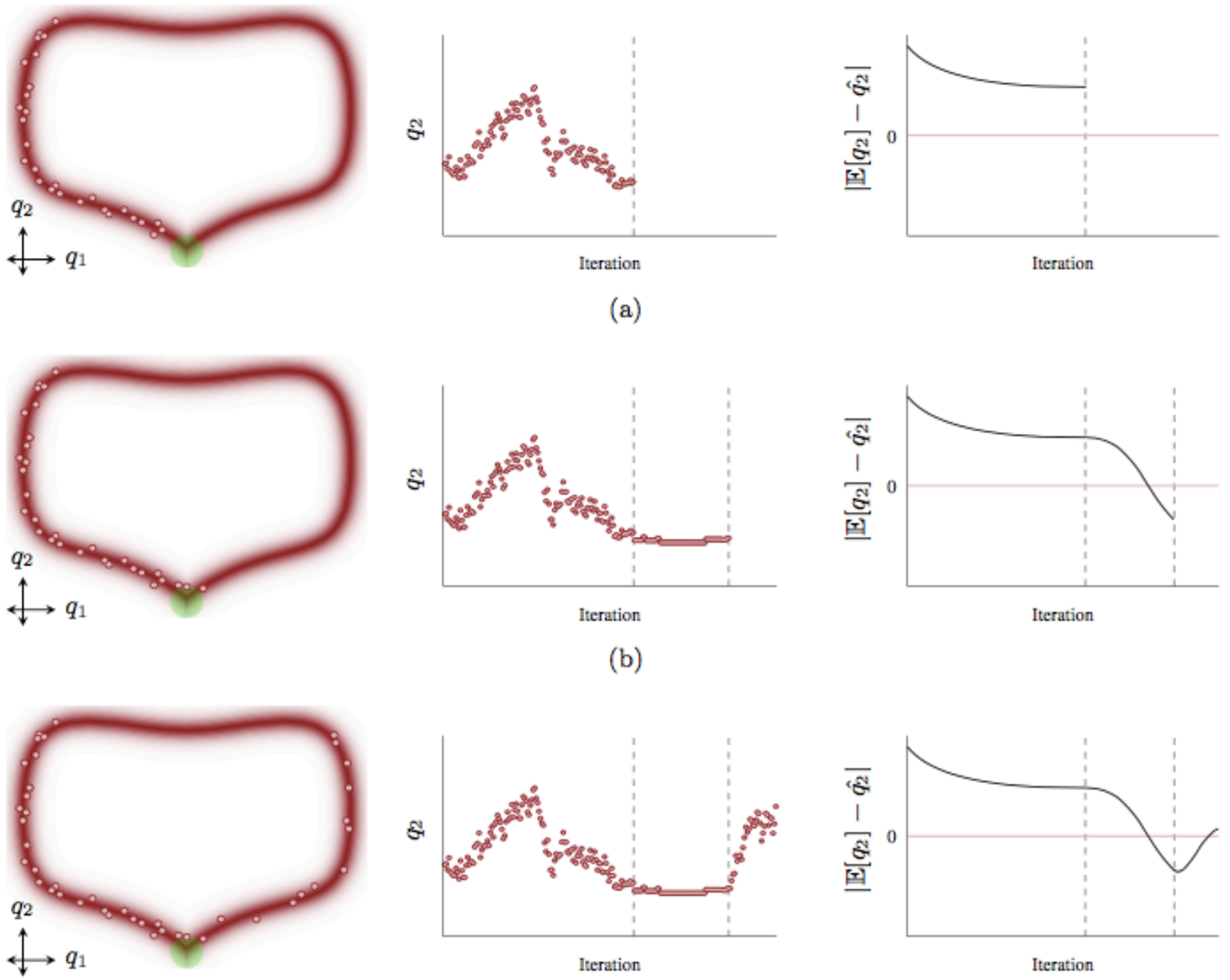
```
In [14]: f = lambda a,b: np.array([[a, 1-a],[b,1-b]])

In [15]: fp = lambda a,b: np.linalg.matrix_power(f(a,b), 100)

In [16]: fp(0.5, 0.7)
Out[16]:
array([[ 0.58333333,  0.41666667],
       [ 0.58333333,  0.41666667]])

In [17]: fp(0.1, 0.2)
Out[17]:
array([[ 0.18181818,  0.81818182],
       [ 0.18181818,  0.81818182]])
```

Problems with samplers, even with stationarity



Ergodicity to the rescue..

..if the universe does not die a heat death first...

conceptually:

If there exists a stationary $s(x)$, you can construct a T such that

$$\lim_{t \to \infty} T^n \text{ is stationary and converges to } s \text{, and}$$

- an ergodic law of large numbers exists
- an ergodic central limit theorem exists

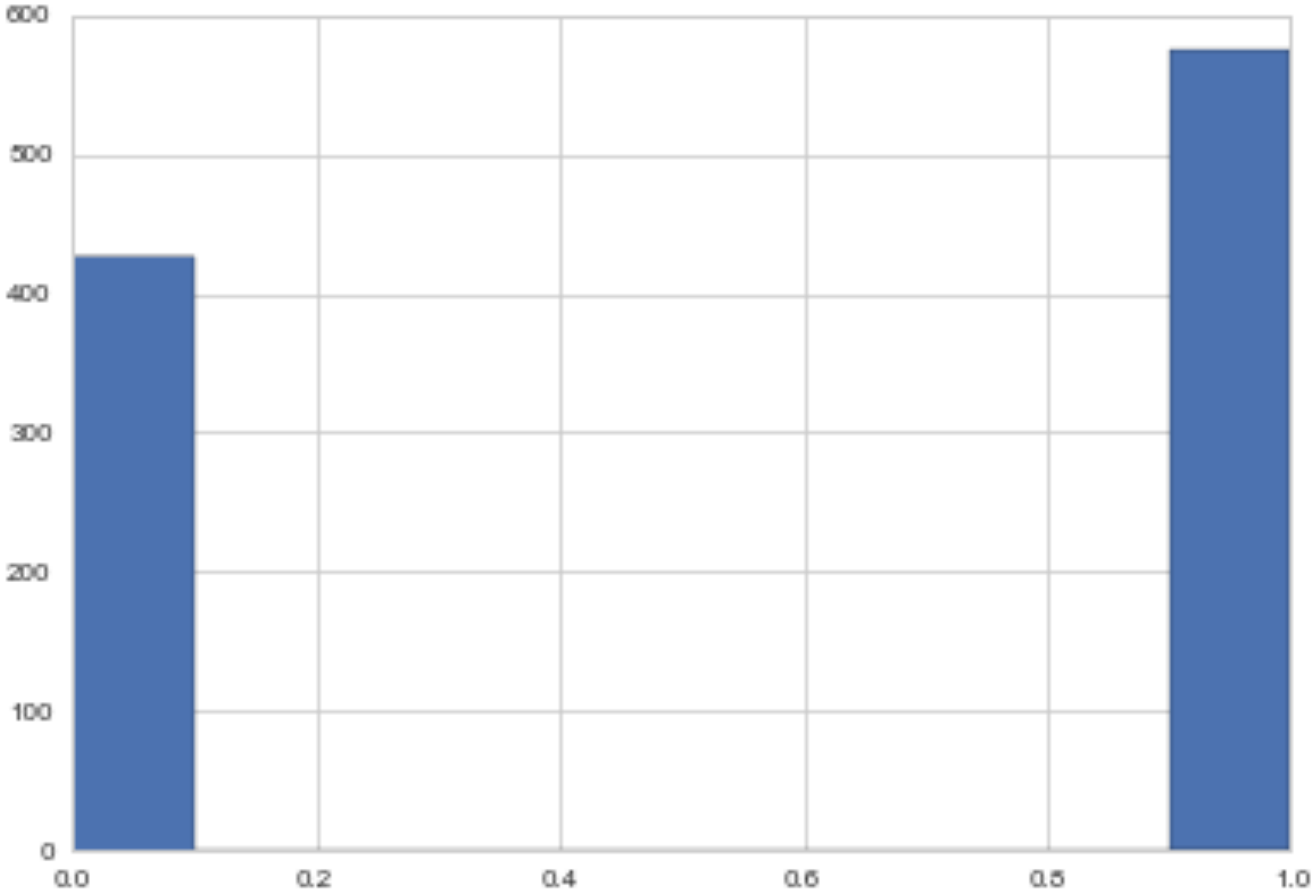
Start from stationary distribution

```python
def rainsunpmf(state_int):
    p = 0.416667
    if state_int==0:
        return p
    else:#anything else is treated as a 1
        return 1 - p
```

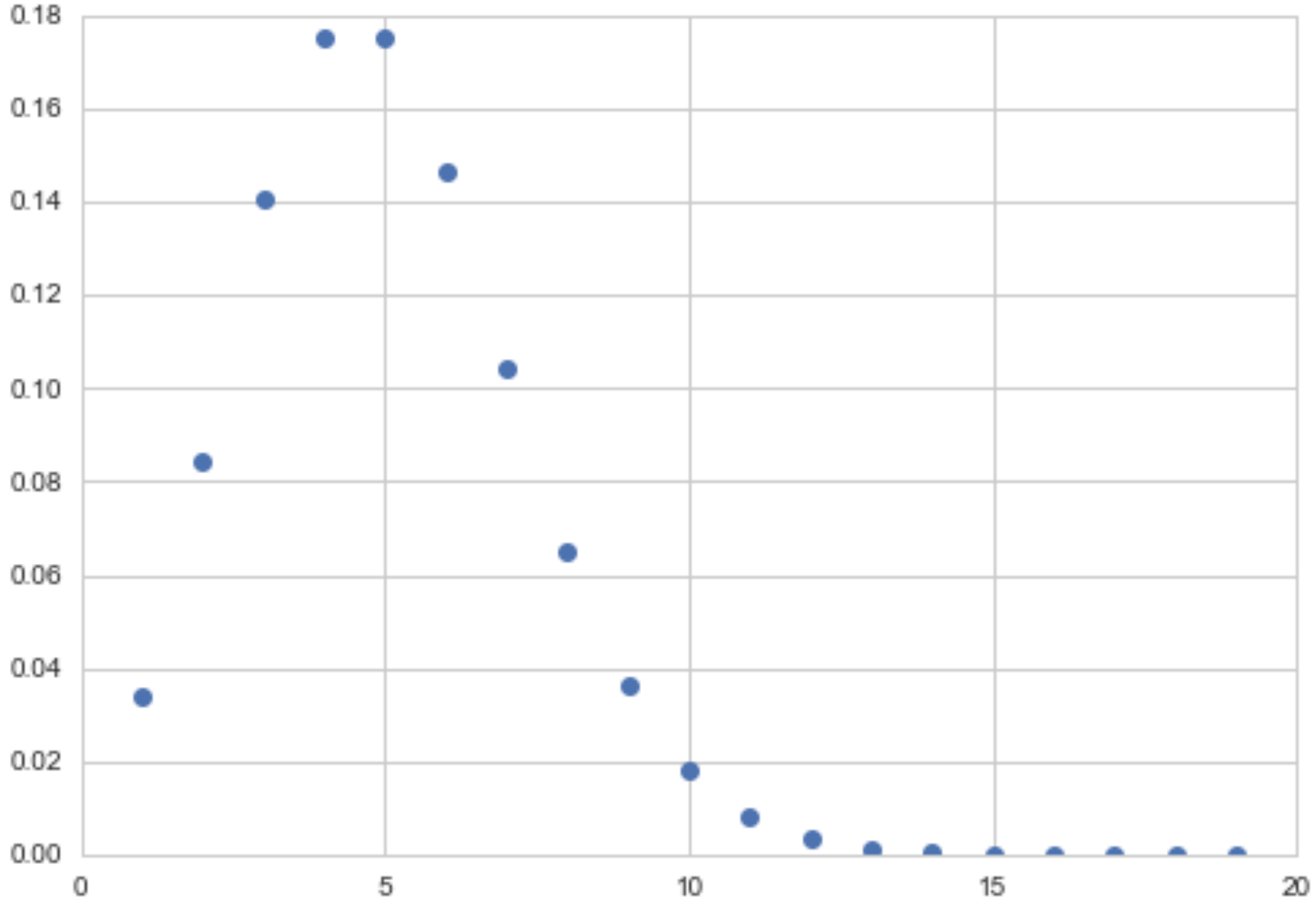
```python
p_sym = np.array([[0.1, 0.9],[0.9, 0.1]])
p_asym = np.array([[0.1, 0.9],[0.3, 0.7]])

def rainsunprop(sint_old):
    return np.random.choice(2,p=p_sym[sint_old])
def rainsunprop_asym(sint_old):
    return np.random.choice(2,p=p_asym[sint_old])
def rainsunpropfunc_asym(sint_new, sint_old):
    return p_asym[sint_old][sint_new]
samps_dis, acc_dis = metropolis(rainsunpmf, rainsunprop, 1000, 1)
samps_dis2, acc_dis2 = metropolis_hastings(rainsunpmf,
     rainsunpropfunc_asym, rainsunprop_asym, 1000, 1)
```

Both give same stationary distribution

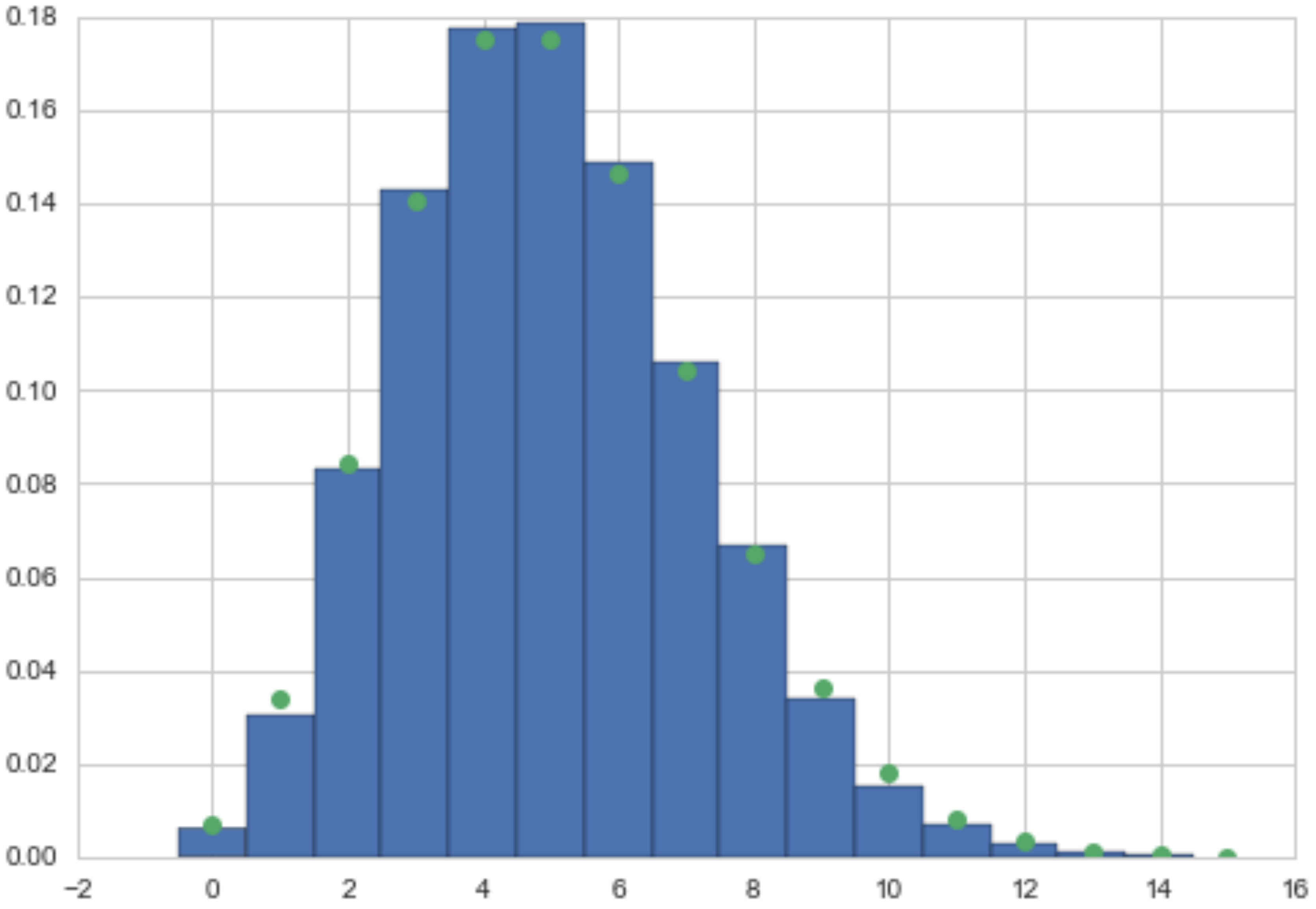


Example: generate poisson



$$Q = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & & \cdots \\ 1/2 & 0 & 1/2 & 0 & 0 & \cdots \\ 0 & 1/2 & 0 & 1/2 & 0 & \cdots \\ 0 & 0 & 1/2 & 0 & 1/2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

$$p(k) = e^{-\mu}\frac{\mu^k}{k!}.$$




```
def prop_draw(ifrom):
    u = np.random.uniform()
    if ifrom !=0:
        if u < 1/2:
            ito = ifrom -1
        else:
            ito = ifrom + 1
    else:
        if u < 1/2:
            ito=0
        else:
            ito=1
    return ito

def prop_pdf(ito, ifrom):
    if ito == ifrom - 1:
        return 0.5
    elif ito == ifrom + 1:
        return 0.5
    elif ito == ifrom and ito == 0:#needed to make first row sum to 1
        return 0.5
    else:
        return 0
```

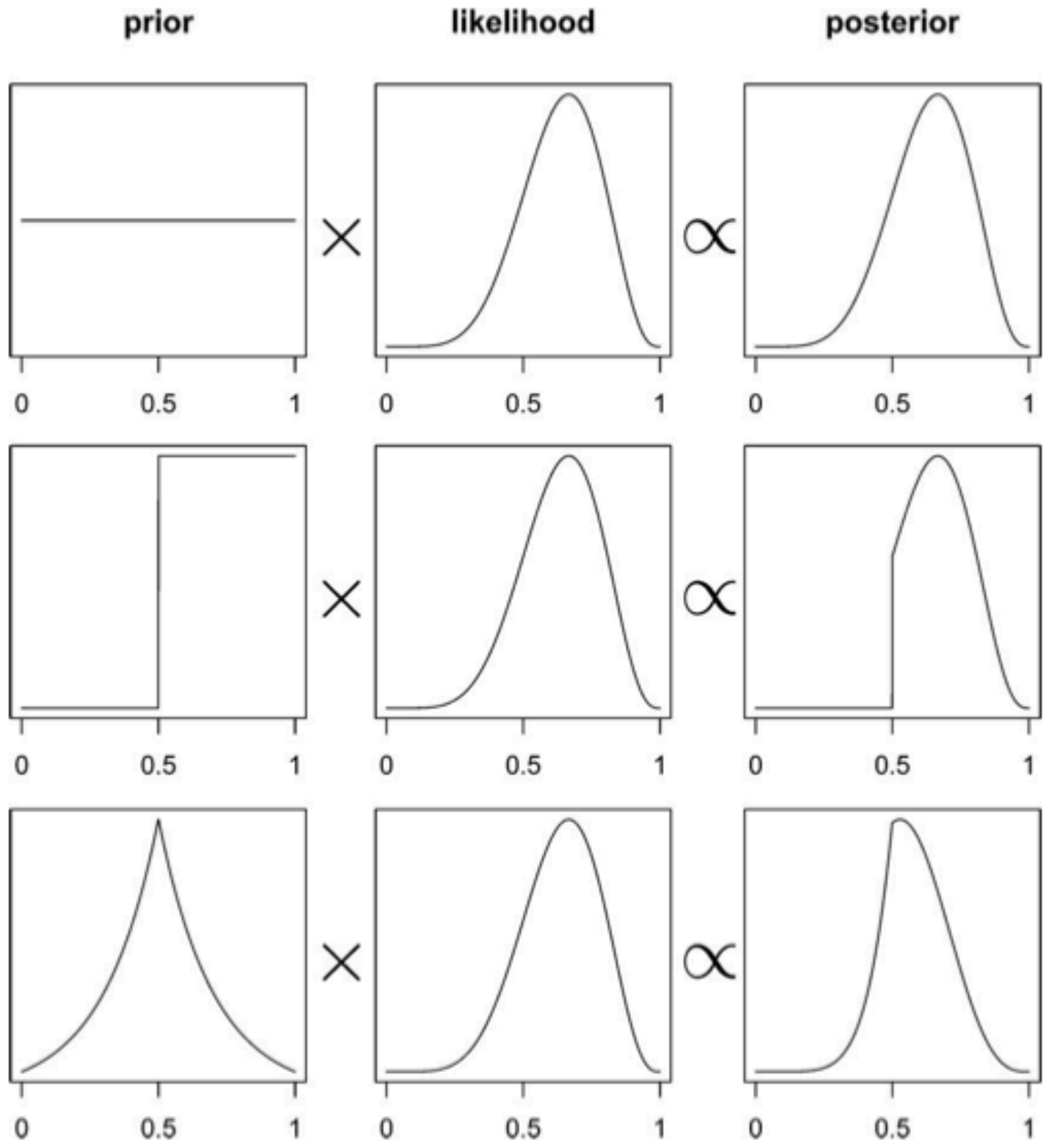
Bayesian Stats: posterior distribution

$$p(\theta|y) = \frac{p(y|\theta)\, p(\theta)}{p(y)}$$

with the evidence $p(D)$ or $p(y)$ the expected likelihood (on existing data points) over the prior $E_{p(\theta)}[\mathcal{L}]$:

$$p(y) = \int d\theta p(y|\theta)p(\theta).$$

- $posterior = \frac{likelihood \times prior}{evidence}$
- evidence is just the normalization
- usually dont care about normalization (until model comparison), just samples



2 key slides: Marginalization

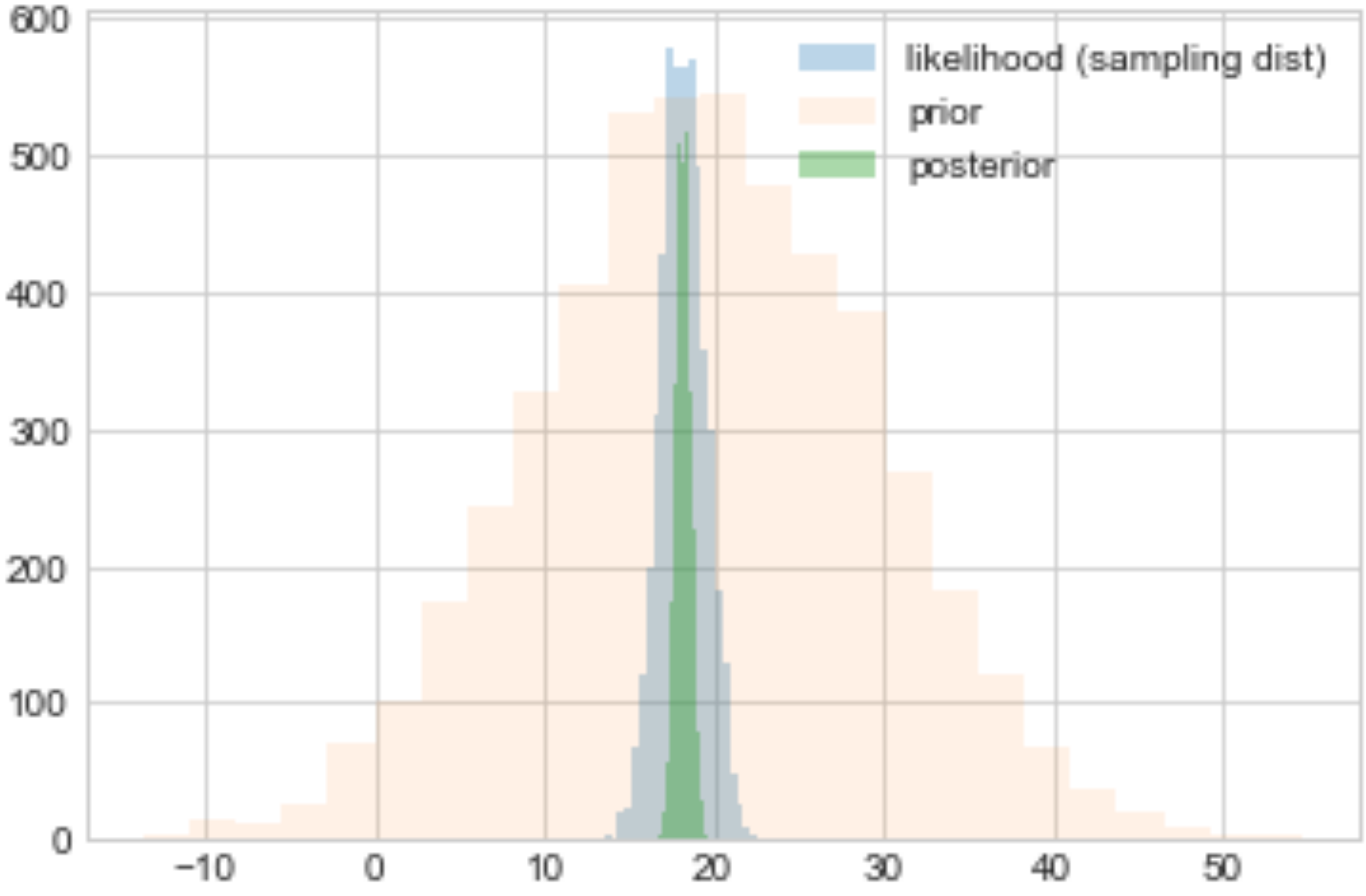
What if θ is multidimensional?

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

Normal-Normal Model

$$p(\mu, \sigma^2) = p(\mu|\sigma^2)p(\sigma^2)$$

- **fixed σ prior**: $p(\sigma^2) = \delta(\sigma^2 - \sigma_0^2)$
- **non-fixed σ prior**: Choose a functional form that is mildly informative, e.g., normal, half cauchy, half normal
- **μ prior**: Mildly informative normal with prior mean and wide standard deviation



- fixed σ

```
logprior = lambda mu:
    norm.logpdf(mu, loc=mu_prior, scale=std_prior)
loglike = lambda mu:
    np.sum(norm.logpdf(Y, loc=mu, scale=np.std(Y)))
logpost = lambda mu:
    loglike(mu) + logprior(mu)
```

- non-fixed σ:

```
logprior = lambda mu, sigma:
    norm.logpdf(mu, loc=mu_prior, scale=std_prior) +
    norm.logpdf(sigma, loc=sig_data, scale=2)
loglike = lambda mu, sigma:
    np.sum(norm.logpdf(Y, loc=mu, scale=sigma))
logpost = lambda mu, sigma:
    loglike(mu, sigma) + logprior(mu, sigma)
```

Marginalization

Marginal posterior:

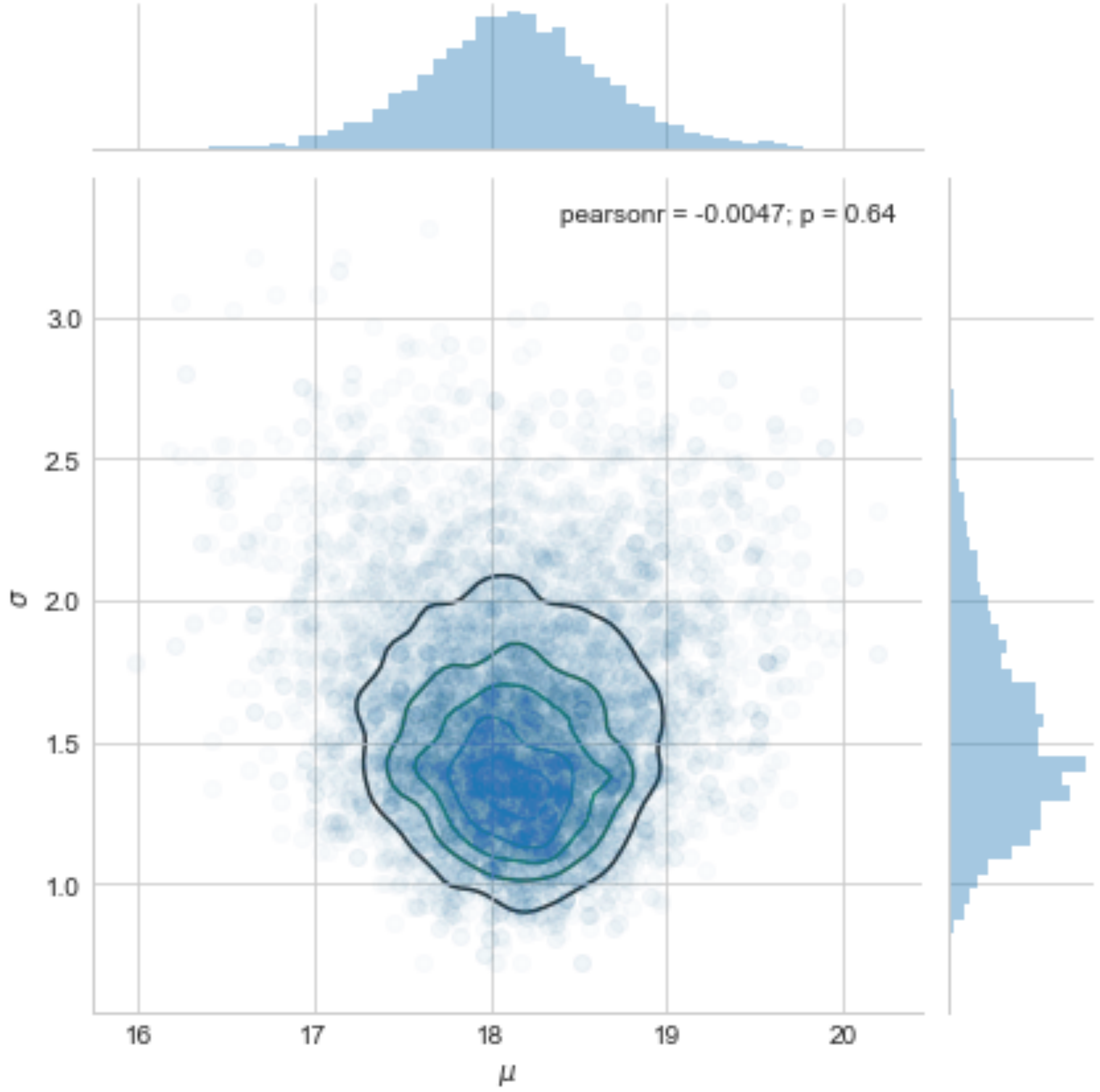
$$p(\theta_1|D) = \int d\theta_{-1} p(\theta|D).$$

```
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])
```



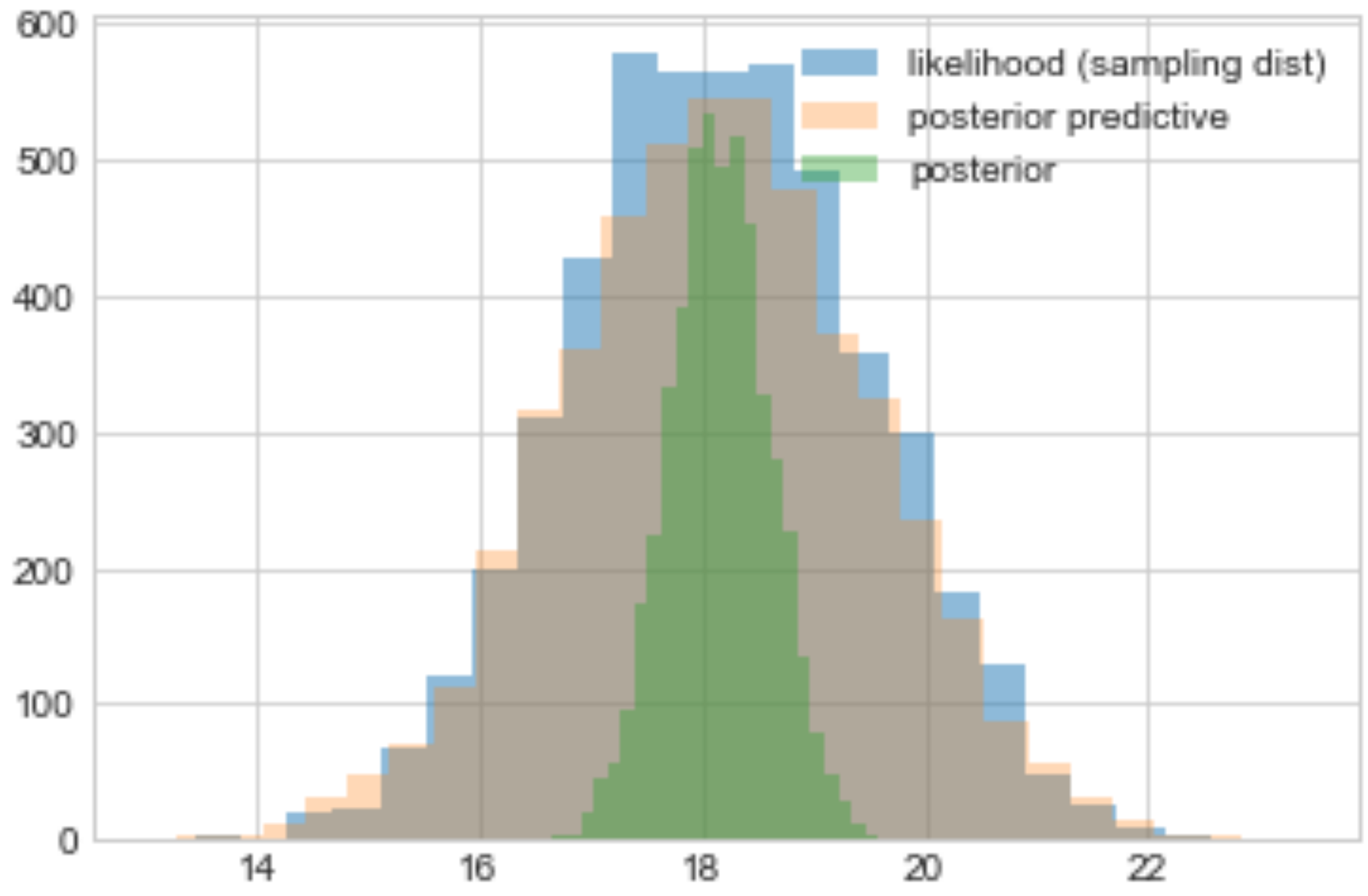
Posterior Predictive

The distribution of a future data point y^*:

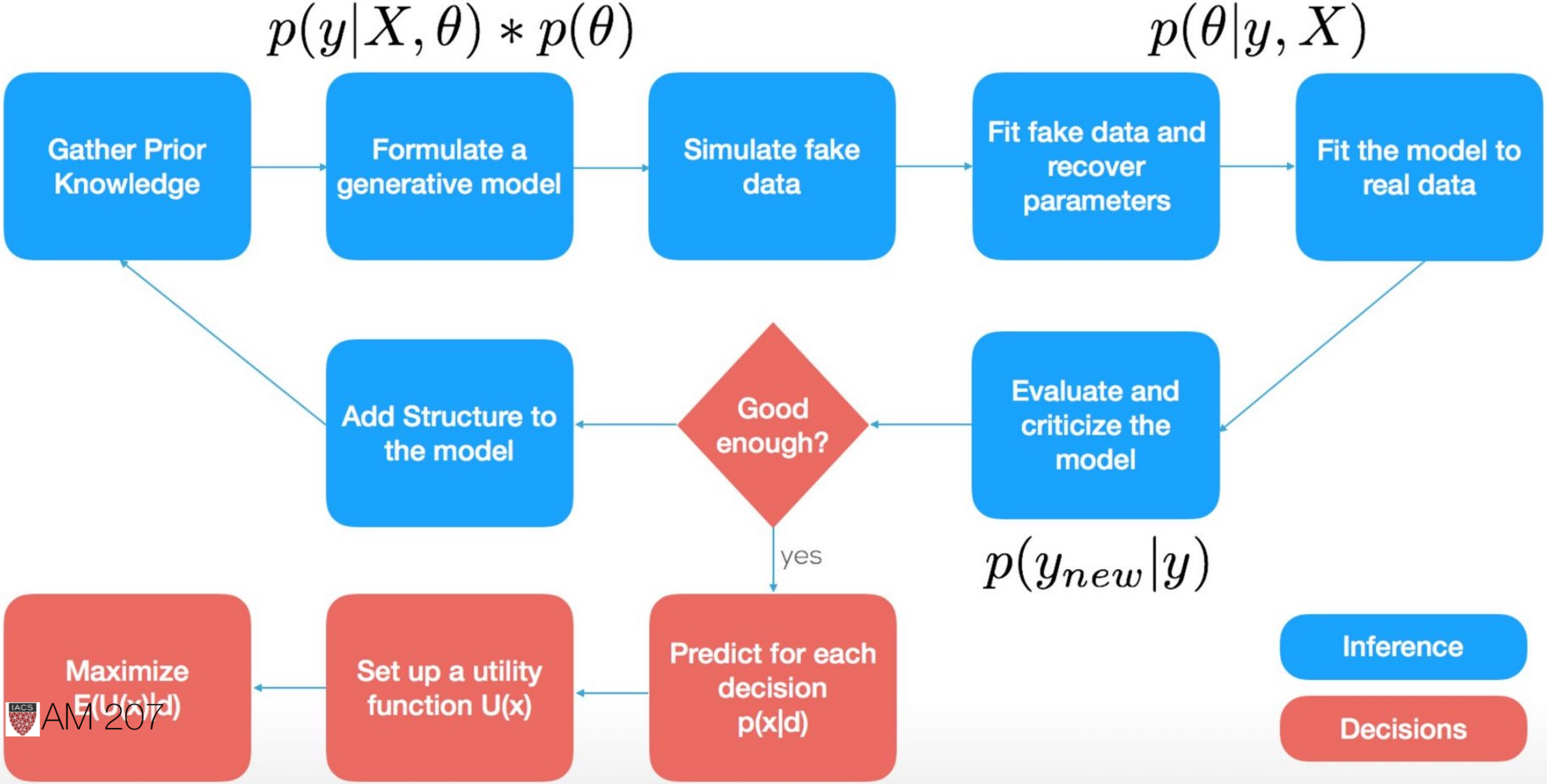
$$p(y^*|D=\{y\}) = E_{p(\theta|D)}[p(y|\theta)]$$
$$= \int d\theta p(y^*|\theta)p(\theta|\{y\}).$$

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

Coin Toss Model

- Coin tosses are modeled using the Binomial Distribution, which is the distribution of a set of Bernoulli random variables.
- The Beta distribution is conjugate to the Binomial distribution

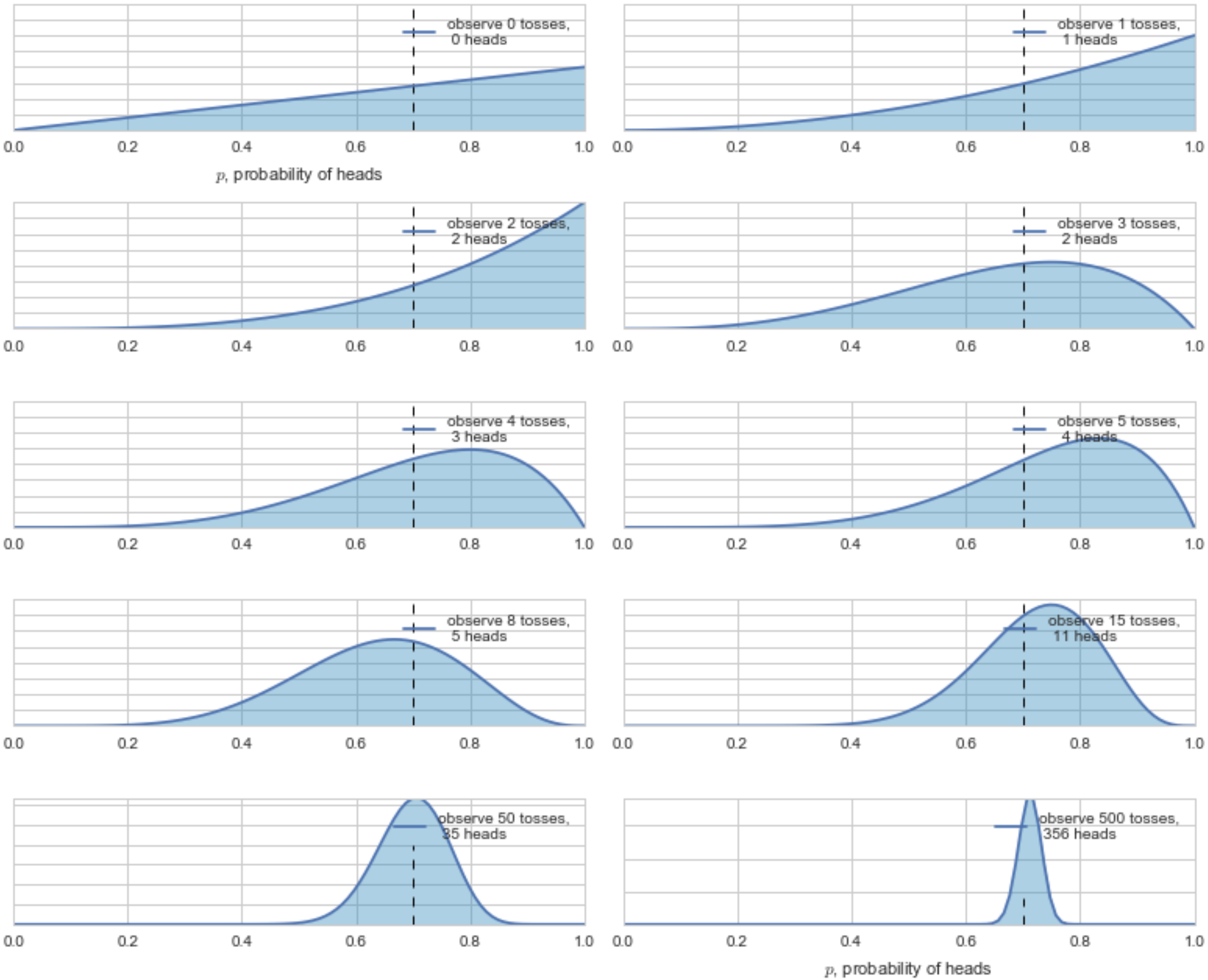
$$p(p|y) \propto p(y|p)P(p) = Binom(n, y, p) \times Beta(\alpha, \beta)$$

Because of the conjugacy, this turns out to be:

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

- think of a prior as a regularizer.
- a $Beta(1, 1)$ prior is equivalent to a uniform distribution.
- This is an **uninformative prior**. Here the prior adds one heads and one tails to the actual data, providing some "towards-center" regularization
- especially useful where in a few tosses you got all heads, clearly at odds with your beliefs.
- a $Beta(2, 1)$ prior would bias you to more heads (water in globe toss).

Bayesian updating of posterior probabilities



Bayesian Updating "on-line"

- as each piece of data comes in, you update the prior by multiplying by the one-point likelihood.

- the posterior you get becomes the prior for our next step

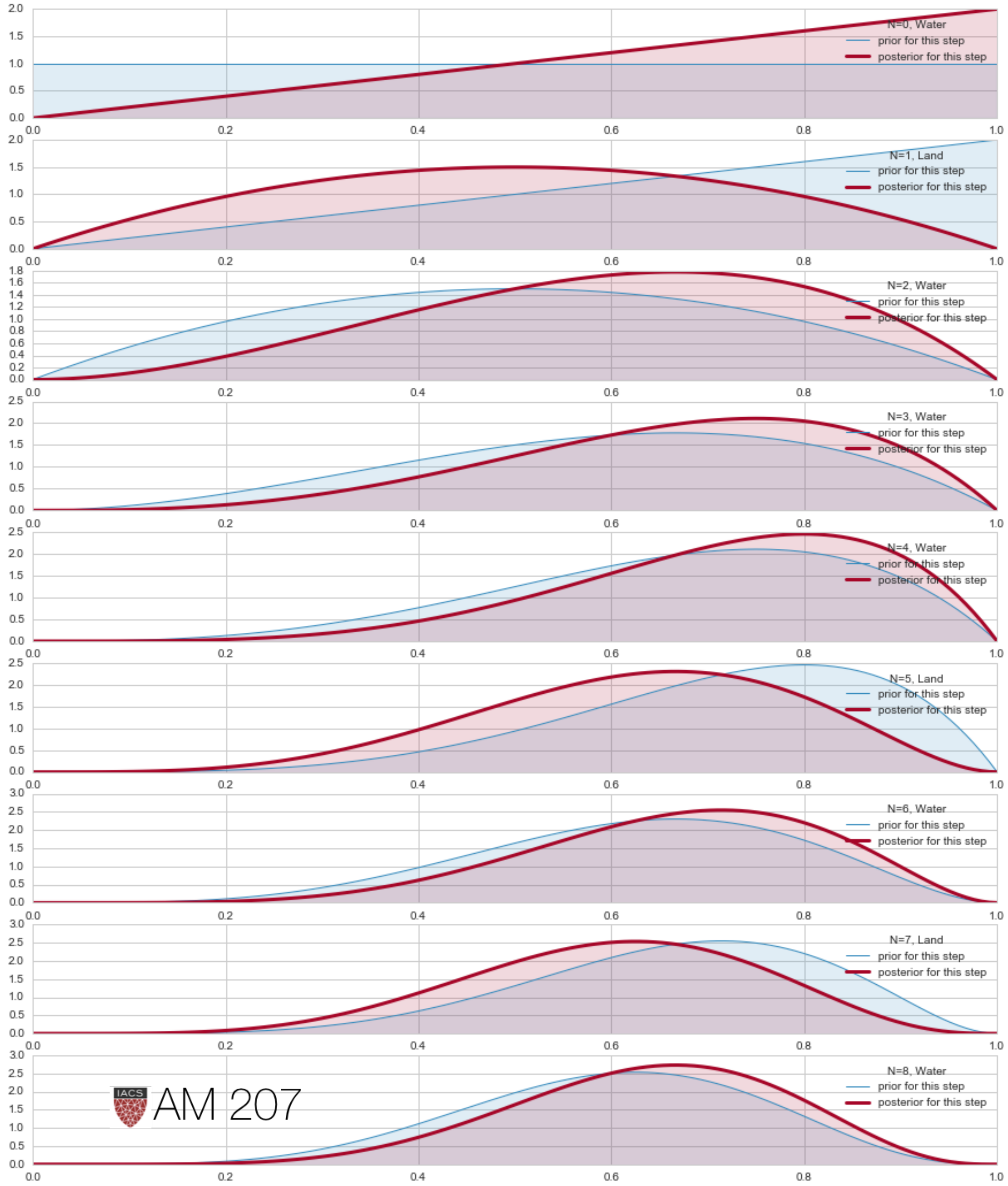
$$p(\theta \mid \{y_1, \ldots, y_{n+1}\}) \propto p(\{y_1, \ldots, y_n\} \mid \theta) \times p(\theta \mid \{y_1, \ldots, y_n\})$$

- the posterior predictive is the distribution of the next data point!

$$p(y_{n+1} | \{y_1, \ldots y_n\}) = E_{p(\theta|\{y_1, \ldots y_n\})}[p(y_{n+1}|\theta)] = \int d\theta p(y_{n+1}|\theta)p(\theta|\{y_1, \ldots y_n\})$$

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.



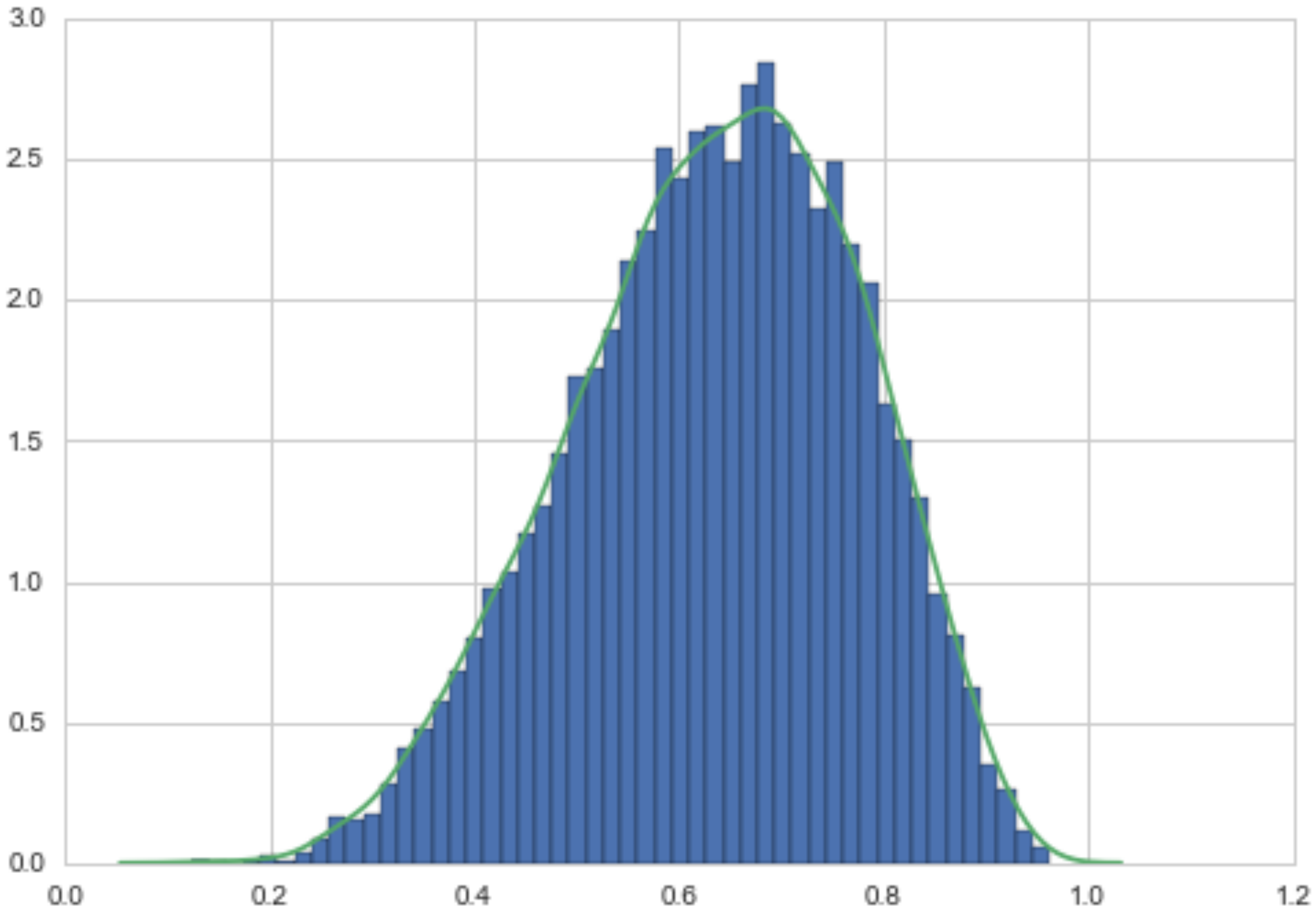
Bayesian Updating of globe

- 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, 0.647314305230314 3)`

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
```

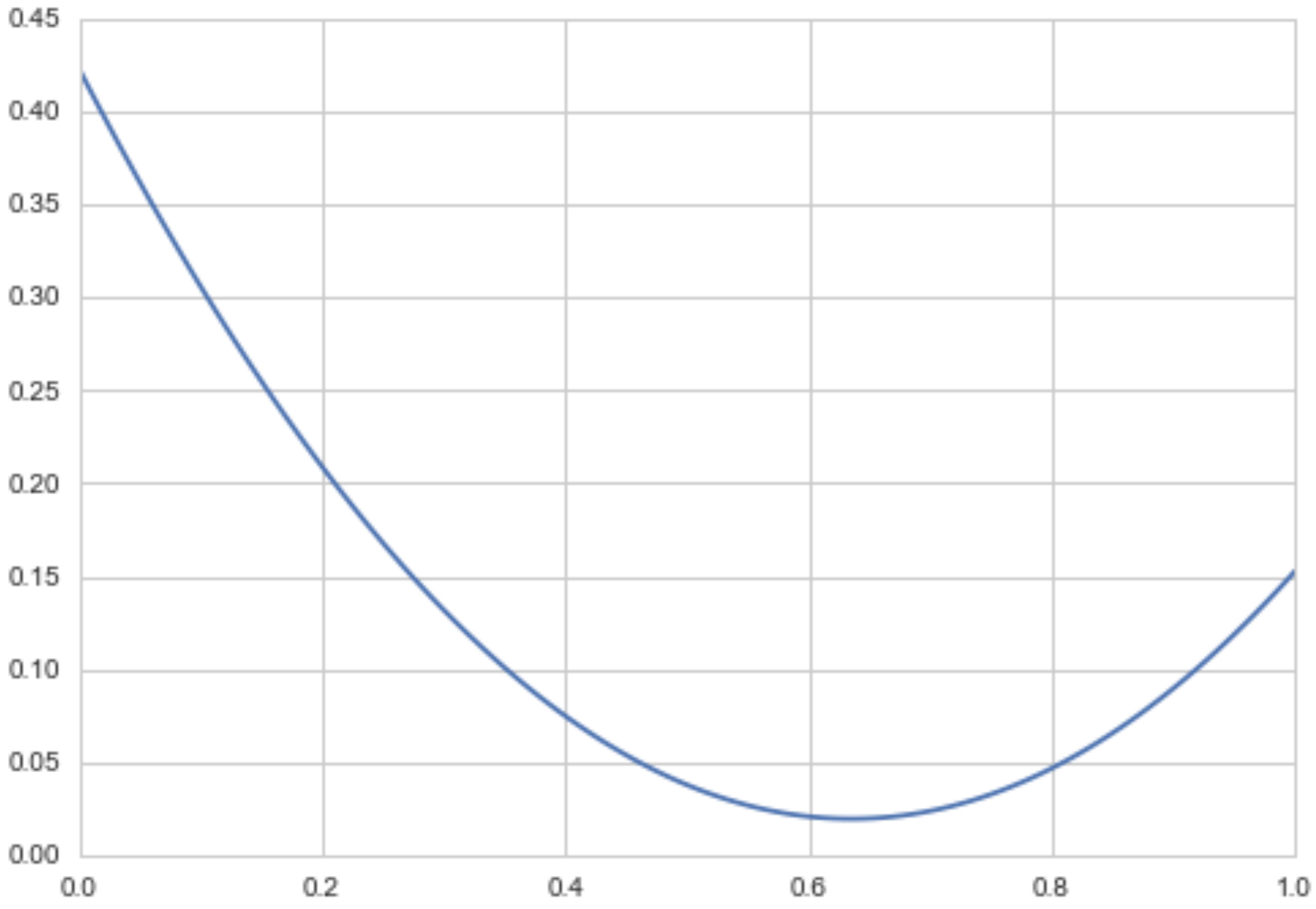
Posterior Mean minimizes squared loss

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

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

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

This is **Decision Theory**.



Posterior predictive

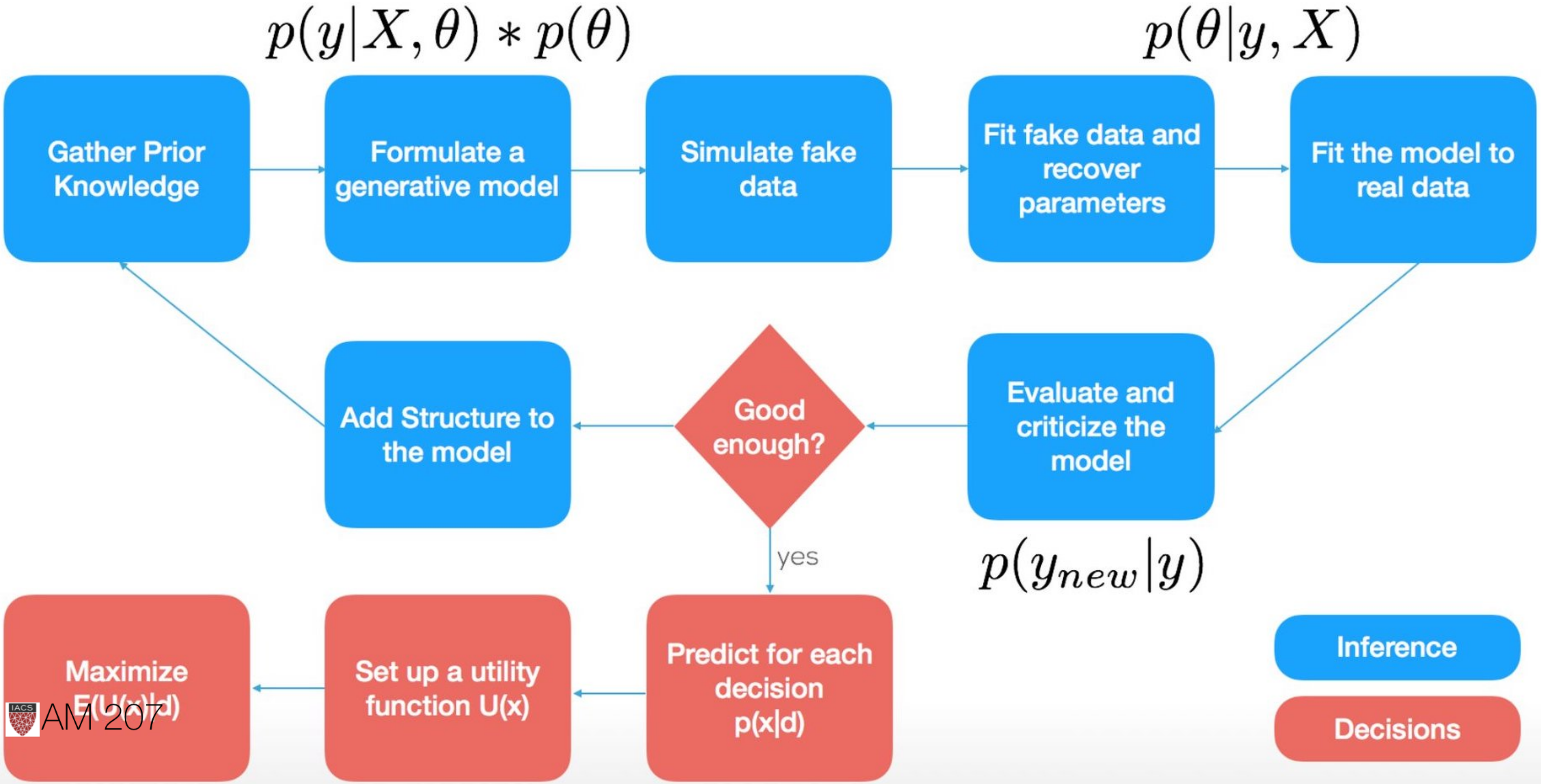
$$p(y^*|D) = \int d\theta p(y^*|\theta)p(\theta|D)$$

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.

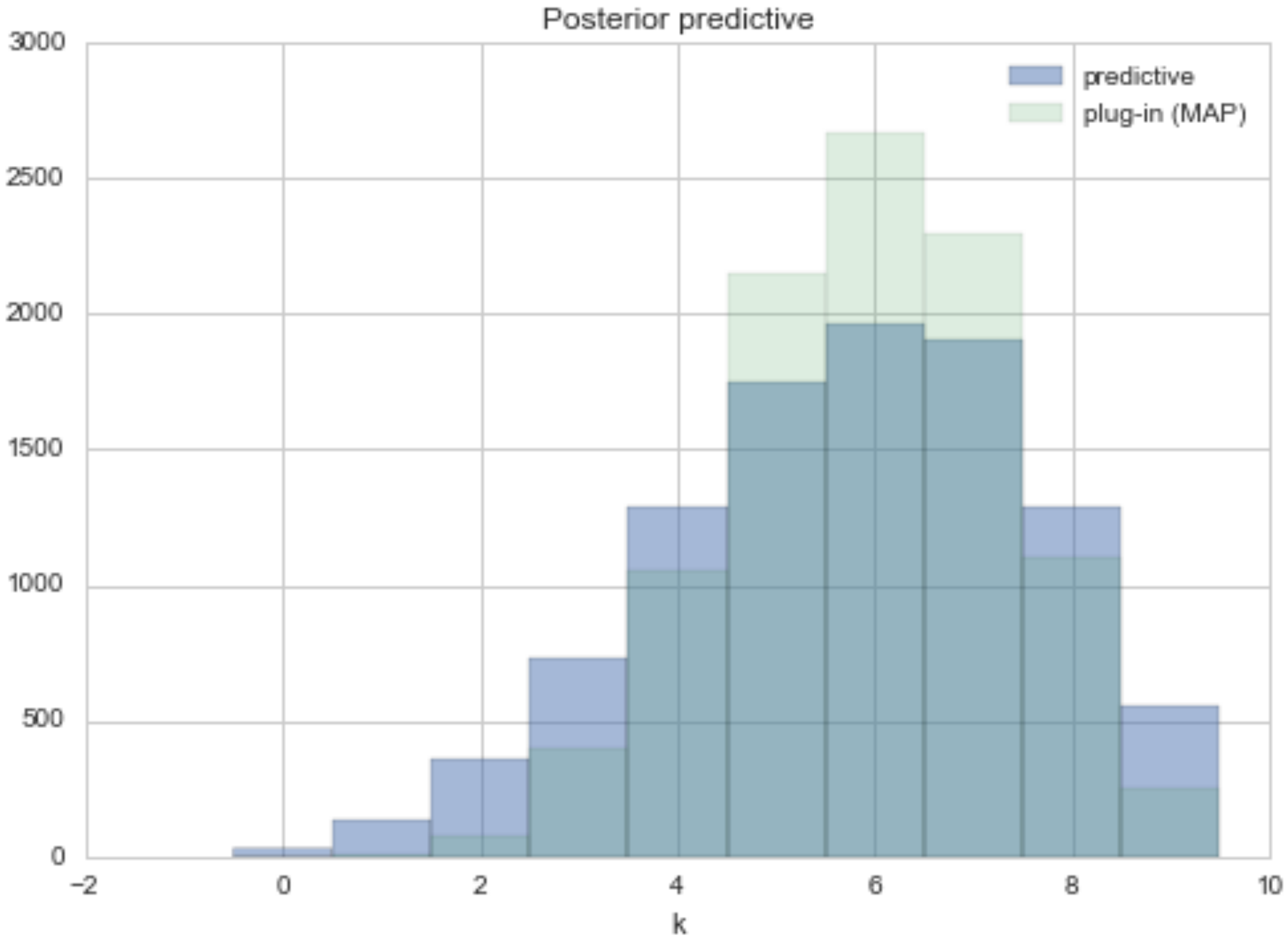
Bayesian Workflow
(from @ericnovik)



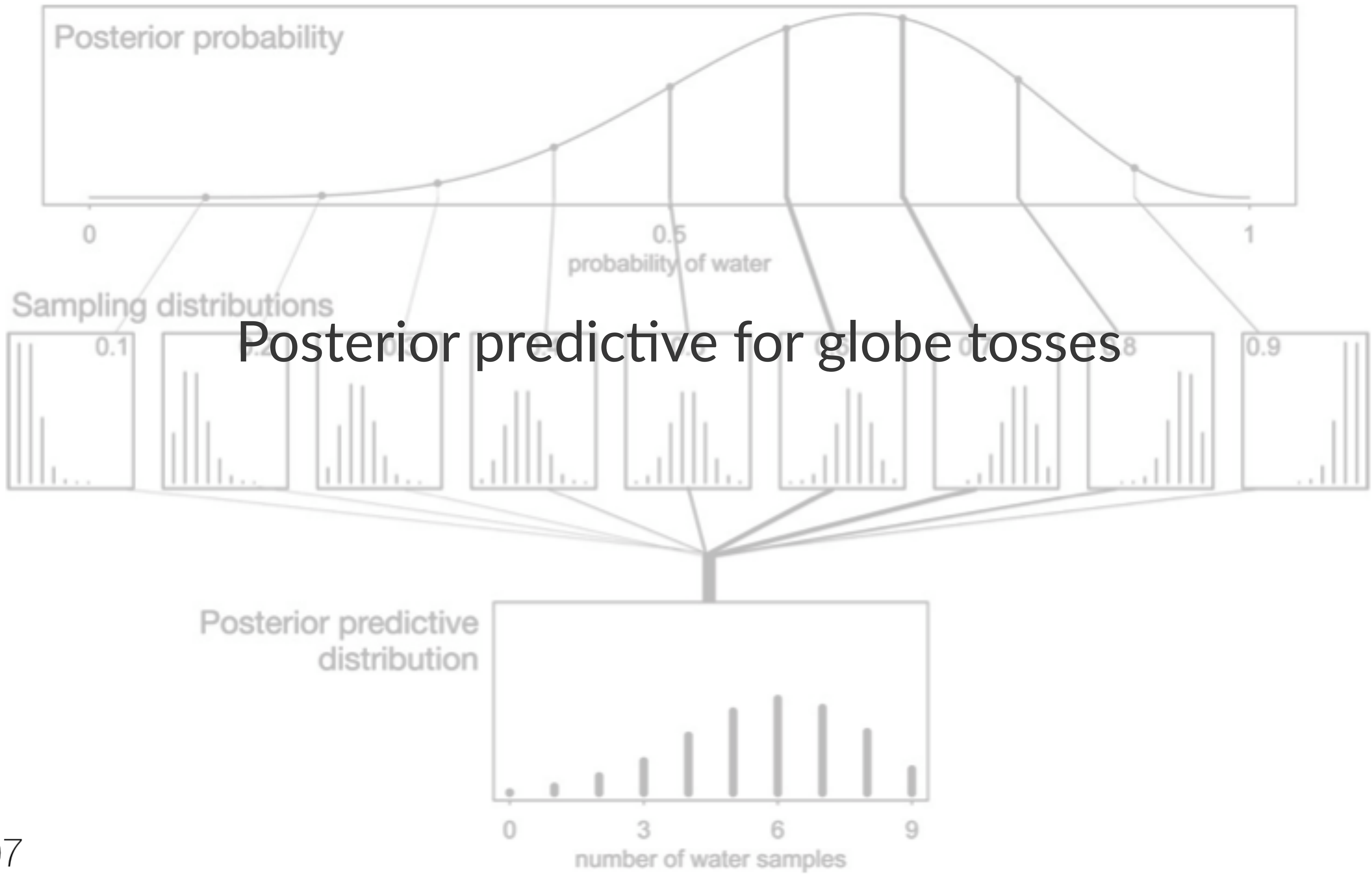
Posterior predictive from sampling

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

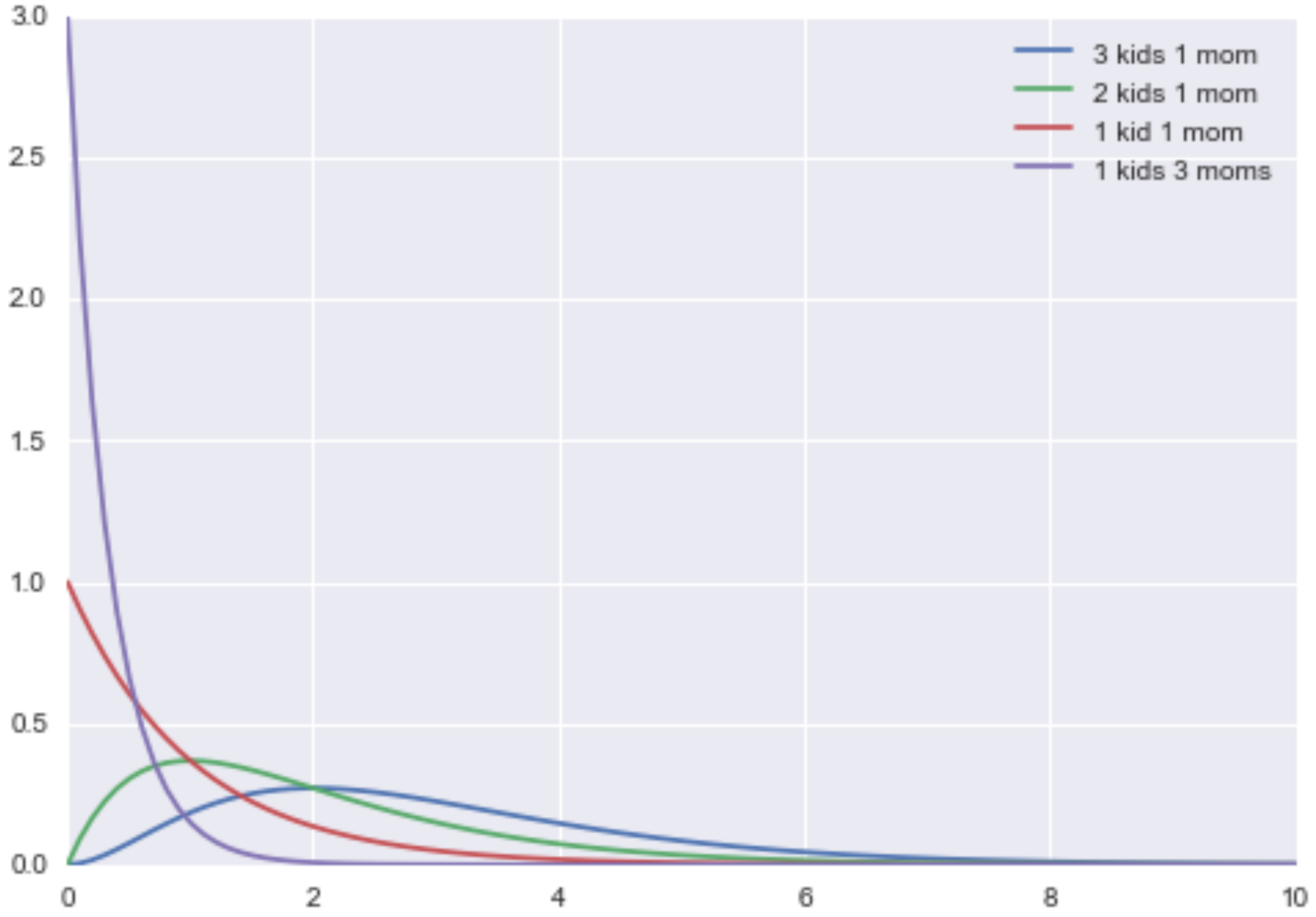
Form is of $Gamma$. In shape-rate parametrization (wikipedia)

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

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.

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

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

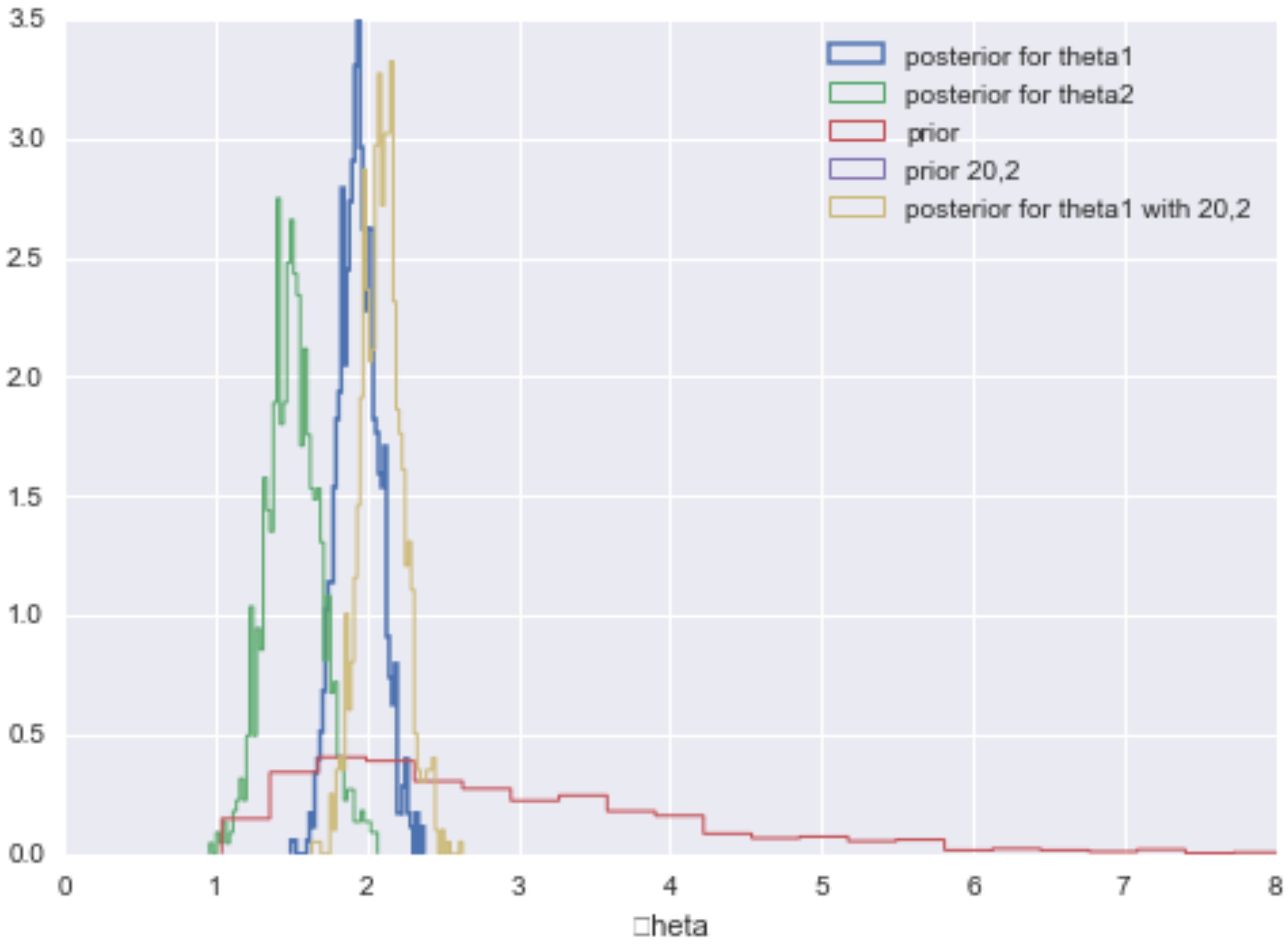
Prior mean, variance:
$E[\theta] = a/b, var[\theta] = a/b^2.$

Posteriors

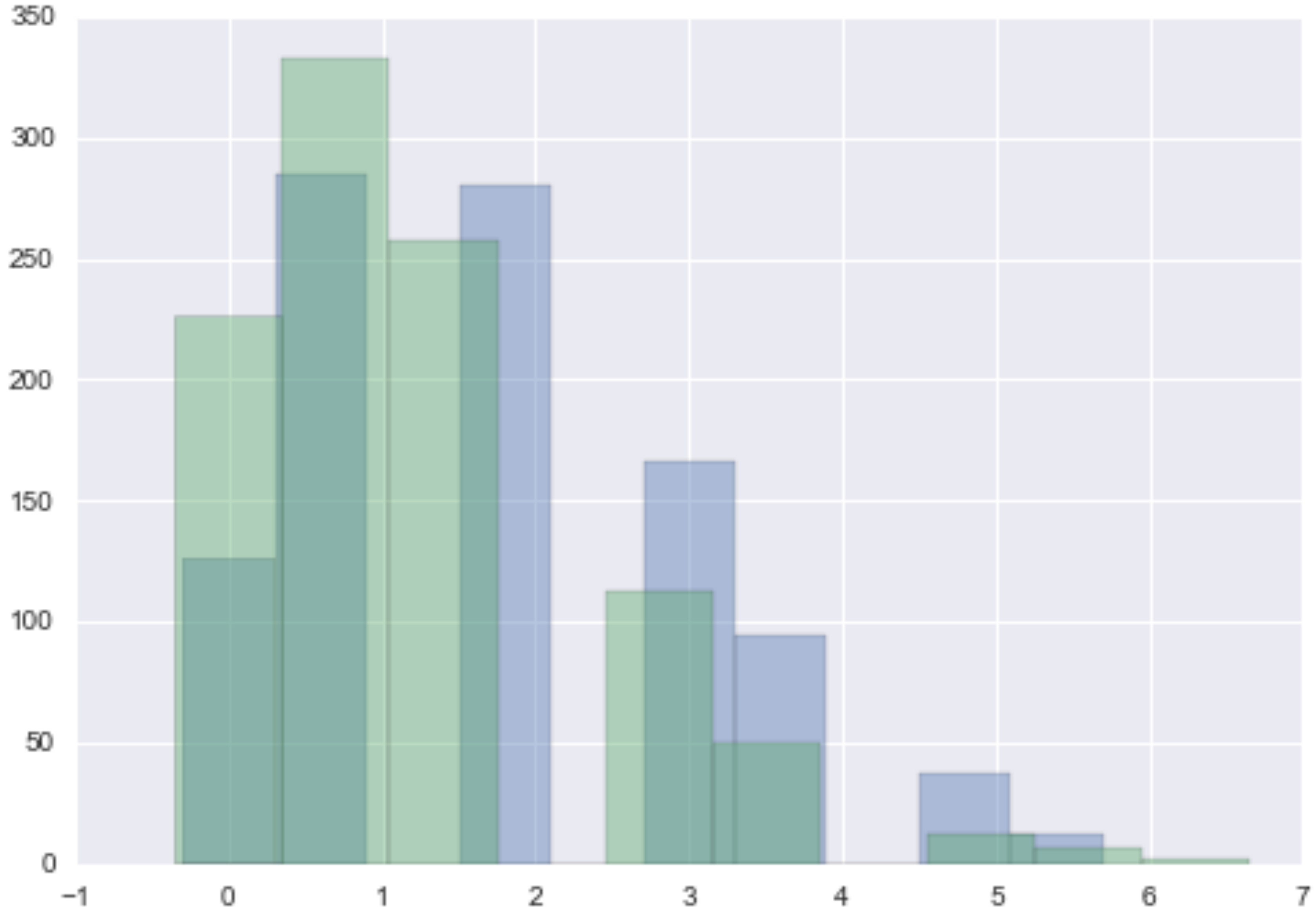
$$E[\theta] = (a + \sum y_i)/(b + N)$$
$$var[\theta] = (a + \sum y_i)/(b + N)^2.$$

```
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

Normal-Normal Model

Posterior for a gaussian likelihood:

$$p(\mu, \sigma^2 | y_1, \ldots, y_n, \sigma^2) \propto \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2} \sum (y_i - \mu)^2} \; p(\mu, \sigma^2)$$

What is the posterior of μ assuming we know σ^2?

Prior for σ^2 is $p(\sigma^2) = \delta(\sigma^2 - \sigma_0^2)$

$$p(\mu|y_1,\ldots,y_n,\sigma^2=\sigma_0^2)\propto p(\mu|\sigma^2=\sigma_0^2)\,e^{-\frac{1}{2\sigma_0^2}\sum(y_i-\mu)^2}$$

The conjugate of the normal is the normal itself.

Say we have the prior

$$p(\mu|\sigma^2)=\exp\left\{-\frac{1}{2\tau^2}(\hat{\mu}-\mu)^2\right\}$$

posterior: $p(\mu|y_1,\ldots,y_n,\sigma^2)\propto\exp\left\{-\frac{a}{2}(\mu-b/a)^2\right\}$

Here

$$a = \frac{1}{\tau^2} + \frac{n}{\sigma_0^2}, \qquad b = \frac{\hat{\mu}}{\tau^2} + \frac{\sum y_i}{\sigma_0^2}$$

Define $\kappa = \sigma^2/\tau^2$

$$\mu_p = \frac{b}{a} = \frac{\kappa}{\kappa + n}\hat{\mu} + \frac{n}{\kappa + n}\bar{y}$$

which is a weighted average of prior mean and sampling mean.

The variance is

$$\tau_p^2 = \frac{1}{1/\tau^2 + n/\sigma^2}$$

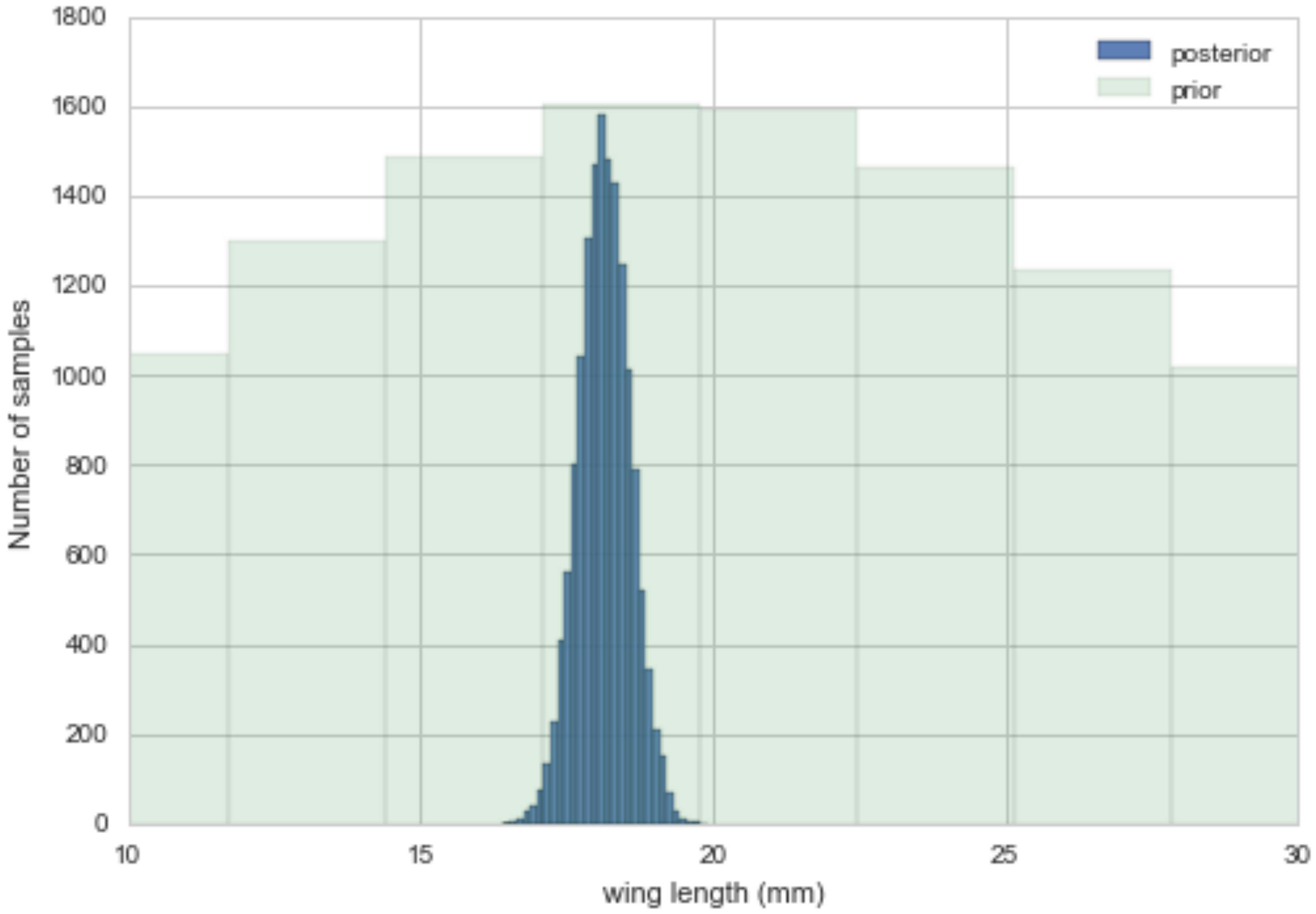
or better

$$\frac{1}{\tau_p^2} = \frac{1}{\tau^2} + \frac{n}{\sigma^2}.$$

as n increases, the data dominates the prior and the posterior mean approaches the data mean, with the posterior distribution narrowing...

Posterior vs prior

```
Y = [16.4, 17.0, 17.2, 17.4, 18.2, 18.2, 18.2, 19.9, 20.8]
#Data Quantities
sig = np.std(Y) # assume that is the value of KNOWN sigma (in the likelihood)
mu_data = np.mean(Y)
n = len(Y)
# Prior mean
mu_prior = 19.5
# prior std
tau = 10
# plug in formulas
kappa = sig**2 / tau**2
sig_post =np.sqrt(1./( 1./tau**2 + n/sig**2));
# posterior mean
mu_post = kappa / (kappa + n) *mu_prior + n/(kappa+n)* mu_data
#samples
N = 15000
theta_prior = np.random.normal(loc=mu_prior, scale=tau, size=N);
theta_post = np.random.normal(loc=mu_post, scale=sig_post, size=N);
```



Bioassay

Dose x_i log(g/ml)	Number of animals n_i	Number of deaths y_i
-0.86	5	0
-0.30	5	1
-0.05	5	3
+0.73	5	5

Bioassays are typically conducted to measure the effects of a substance on a living organism

The death rate is usually modeled as logit^{-1} with two parameters (see below). The goal is to estimate those parameters and be able to infer death rates as a function of dose.

This is a success-failure experiment with failure=death (morbid, I know).

The likelihood since is a success/fail experiment is expressed as a Binomial:

Likelihood:
$P(D_i|\theta_i) = p(y_i, n_i|\theta_i) = \mathrm{Bionomial}(y_i, n_i|\theta_i) \quad \text{for i} = 1, \ldots, 4$

where θ_i is the rate of deaths in the ith experiment.

$$\theta_i = \mathrm{logit}^{-1}(\alpha + \beta x_i) \quad \text{for i} = 1, \ldots, 4$$

We use flat priors for α, β: $p(\alpha, \beta) \propto 1$

Posterior:

$$p(\alpha, \beta | y) \propto p(\alpha, \beta) \prod_{i=1}^{k} p(y_i | \alpha, \beta, n_i, x_i)$$

$$= 1 \prod_{i=1}^{k} [\text{logit}^{-1}(\alpha + \beta x_i)]^{y_i} [1 - \text{logit}^{-1}(\alpha + \beta x_i)]^{n_i - y_i}$$

2 ways to sample:

- **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.

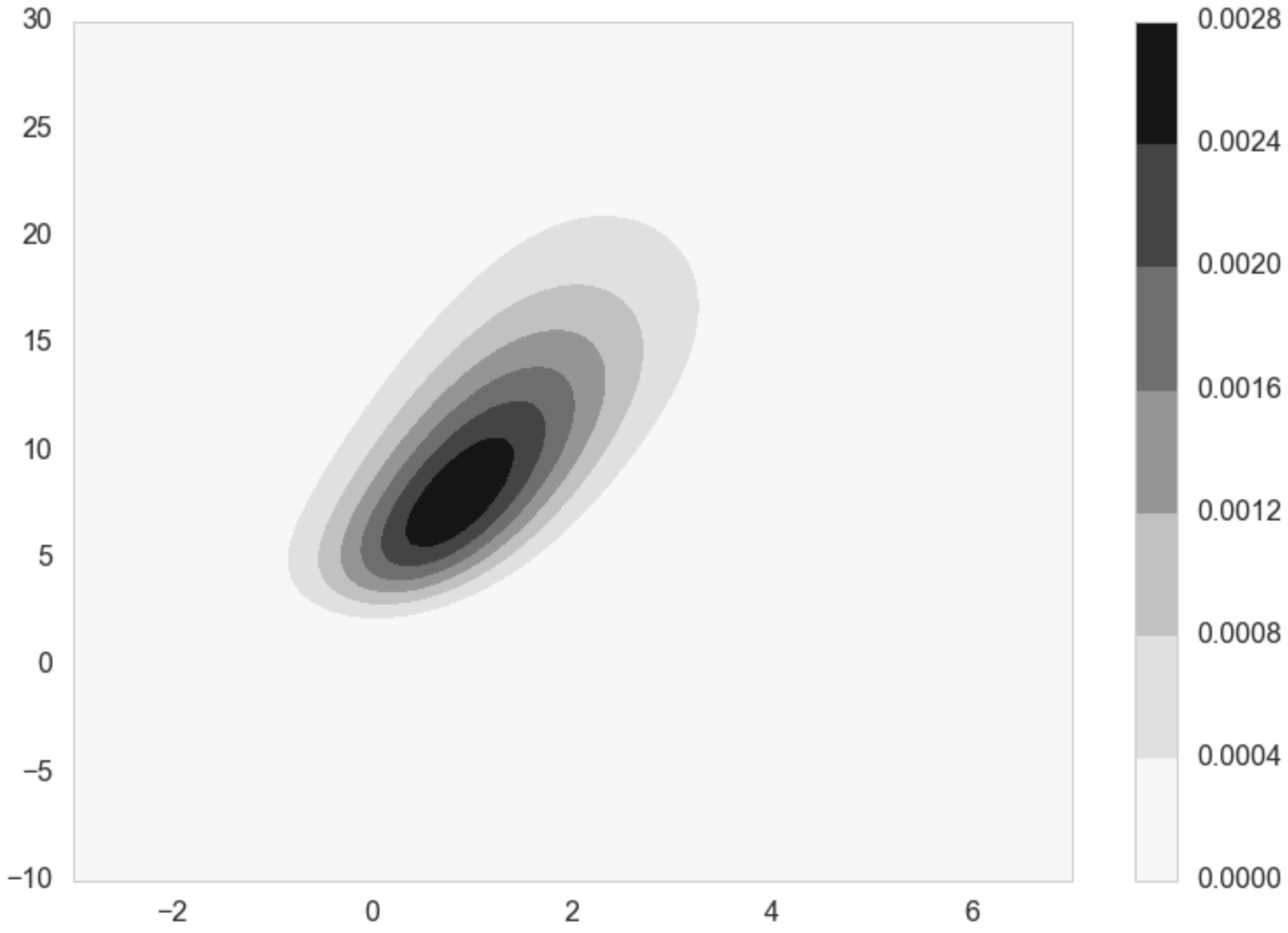
Grid Approximation

```python
## invLogit this is the same as a sigmoid
logitInv= lambda x: np.exp(x)/(1.0+np.exp(x))

## posterior
def calc_posterior(a, b, y=Y, x=X):
    # Calculate joint posterior, given values for a, b
    # x: dosage
    # y: number of deaths
    # a + b: parameters of the model
    p = np.product((logitInv(a+b*x)**y)*(1.0-logitInv( a+b*x))**(n-y))
    return p

# basically calculate the pdf on a grid
X1 = np.linspace(-3,7,101) # alpha
X2 = np.linspace(-10, 30,100) # beta
k=0;j=0
pp=np.zeros((101,100))
for x1 in X1:
    j=0
    for x2 in X2:
        pp[k,j]=calc_posterior(x1,x2)
        j +=1
    k +=1

# look at the posterior distribution
plt.contourf(X1,X2,pp.T)
plt.colorbar()
```



Posterior from componentwise sampling