Lecture 9

Bayesian Stats and Sampling

Last Time

- Bayesian Stats in detail
- Posterior, Posterior predictive
- Bayesian Stats

Today:

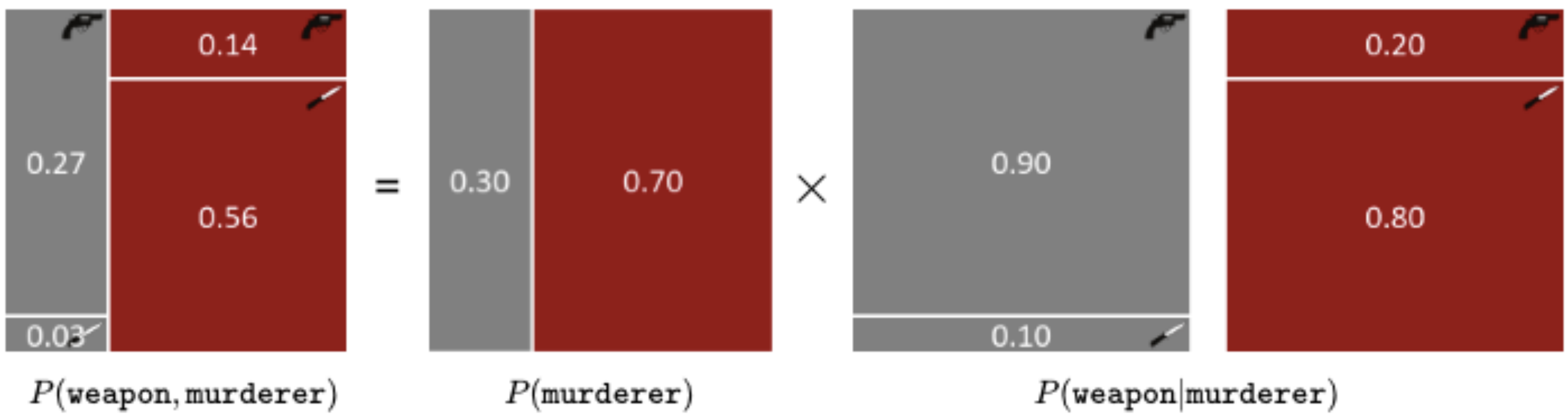
- Exchangeability and the exponential model
- Prior predictive
- Bayesian Regression
- Logistic Regression
- Inverse Transform Sampling
- Rejection Sampling

Bayesian Stats

- assume sample IS the data, no stochasticity
- parameters θ are stochastic random variables
- associate the parameter θ with a prior distribution $p(\theta)$
- The prior distribution generally represents our belief on the parameter values when we have not observed any data yet (to be qualified later)
- obtain posterior distributions
- predictive distribution from the posterior

Basic Idea

Get the joint Probability distribution



Now we condition on some random variables and learn the values of others.

Rules

1. $P(A, B) = P(A \mid B)P(B)$

2. $P(A) = \sum_B P(A, B) = \sum_B P(A \mid B)P(B)$

$P(A)$ is called the **marginal** distribution of A, obtained by summing or marginalizing over B.

Posterior

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

Posterior: $p(\theta|D) \propto p(\{y\}|\theta, \{x\})\, p(\theta)$

Evidence:

$$p(\{y\} \mid \{x\}) = \int d\theta\, p(\theta, D) = \int d\theta\, p(\{y\}|\theta, \{x\})p(\theta).$$

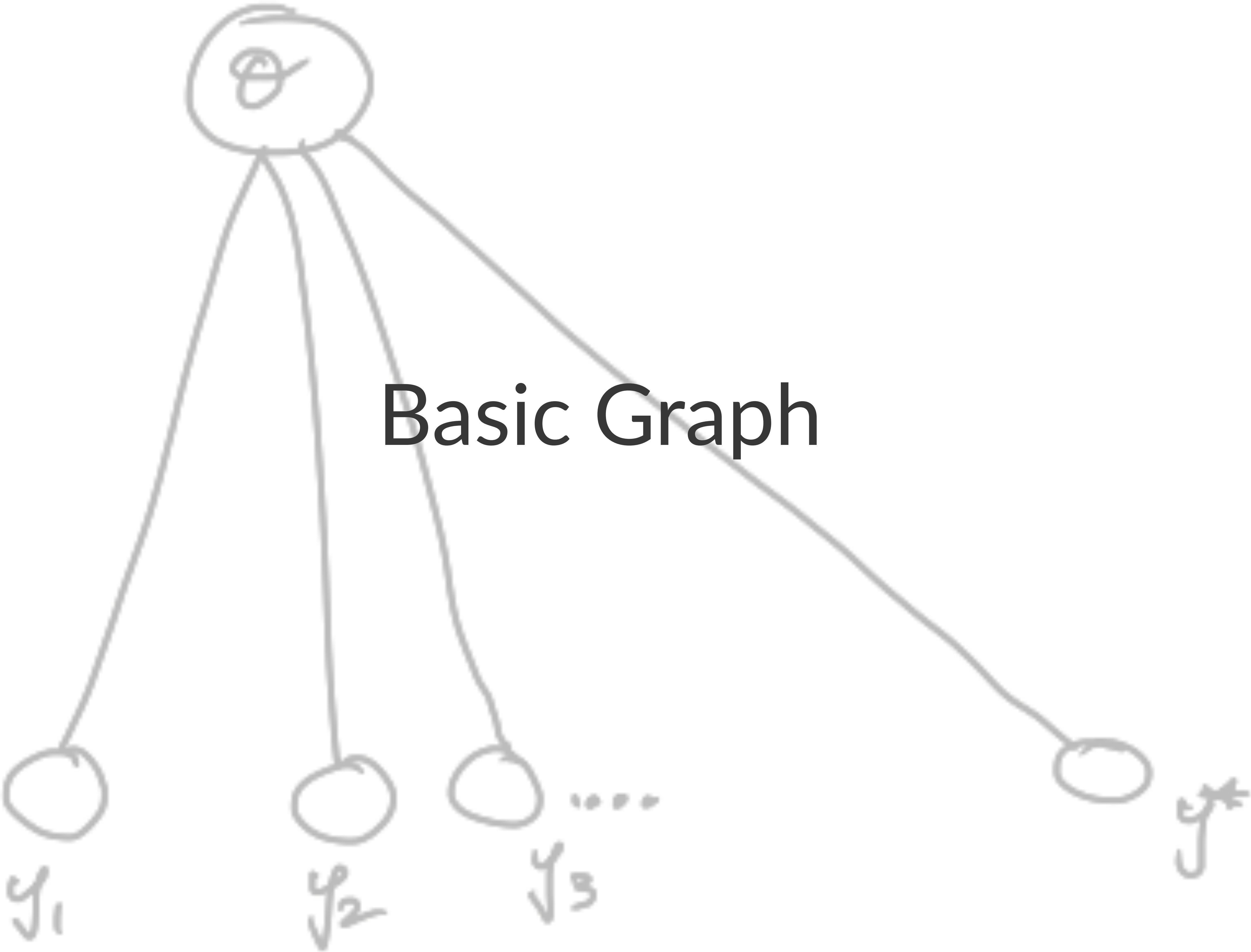
Marginalization

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

Posterior Predictive:

$$p(y^*|D = \{x, y\}) = \int d\theta p(y^*, \theta|\{x, y\}).$$

Basic Graph



Replicative Posterior Predictive

$$p(\{y^*\} \mid \{x^*\}) = \int p(\{y^*\}|\theta, \{x^*\})p(\theta|\mathcal{D})d\theta,$$ observed

data: $\mathcal{D} = \{x, y\}$

Replicated Data: $\{y_r\}$: data seen tomorrow if experiment replicated with same model and value of θ producing todays data $\{y\}$.

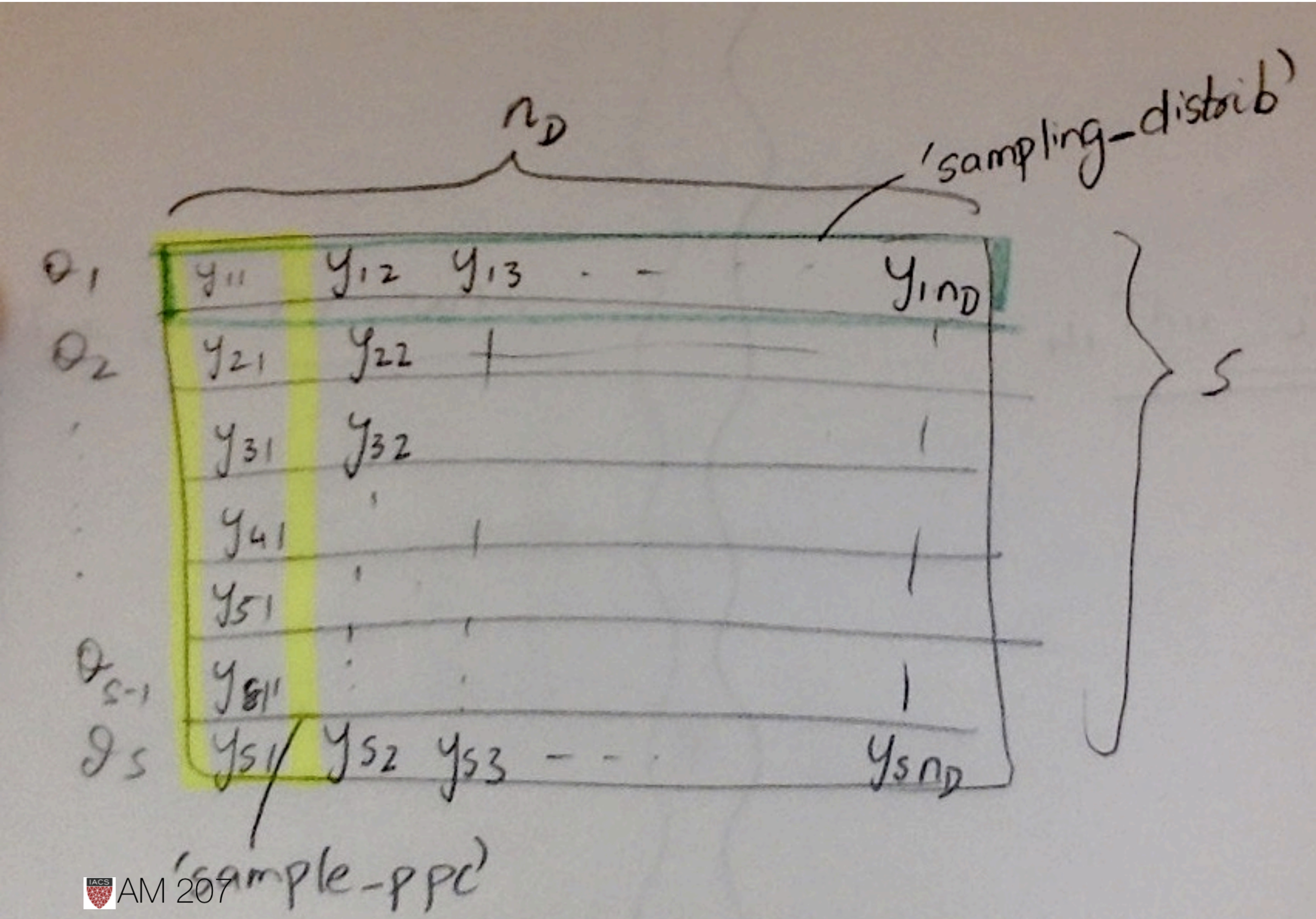
$\{y_r\}$ comes from posterior predictive. The idea is to make as many replications as the size of your dataset.

Another way to sample

```
ppc_rep=np.empty((dataset_size, num_samples))
for i in range(dataset_size):
    ppc_rep[i,:] = distrib.rvs(param=posterior_samples)
```

For each data point, sample using the likelihood(sampling distribution) from S samples of the posterior. Gives an S sized posterior predictive at each "data point".

You can then slice the other way to get a dataset sized posterior-predictive


n_D
'sampling-distrib'
θ_1 y_11 y_12 y_13 - - y_1n_D
θ_2 y_21 y_22
y_31 y_32
y_41
y_51
θ_s-1
θ_s y_s1 y_s2 y_s3 - - - y_sn_D
S
'sample-ppc'

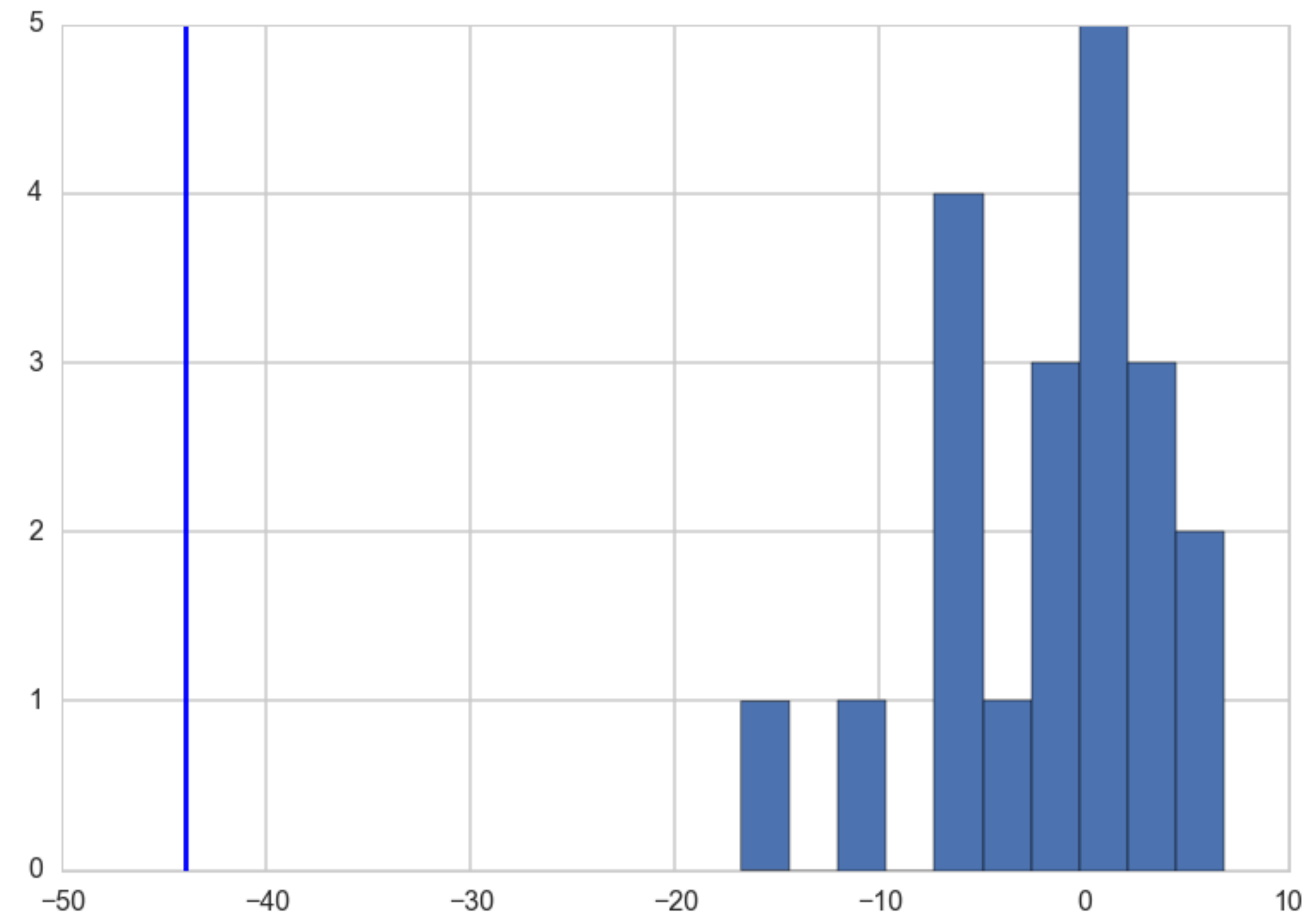
Departure from usual predictive sampling

Sample an entire $\{y_r\}$ at each θ from trace.

This allows to compute distributions from the posterior predictive replications for informal test statistics.

These processes are called **Posterior Predictive Checks**.

Replicative prior predictives are also useful for callibration.



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)

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

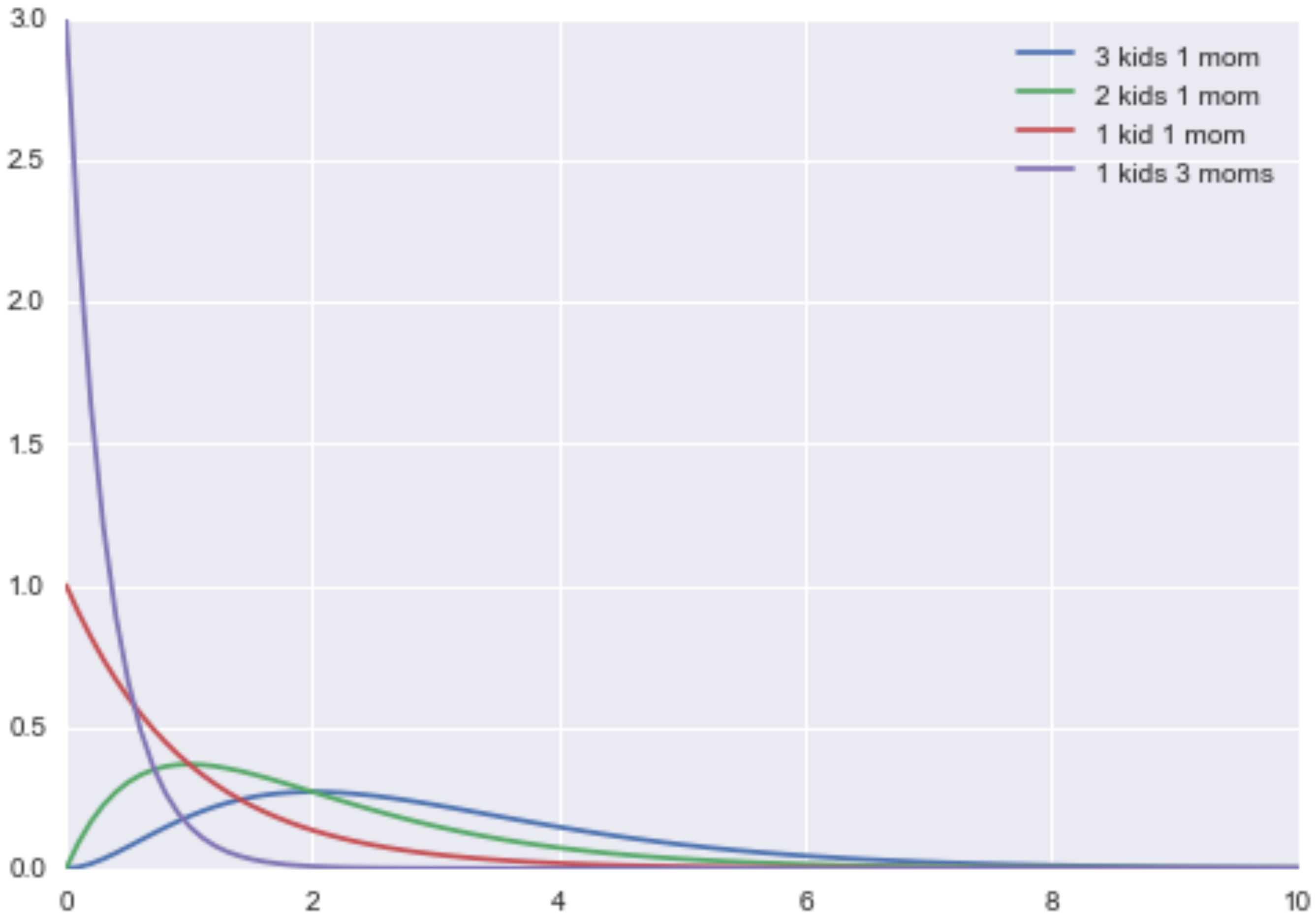
Posterior:

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

Complete Posterior

Multiplies the 2 posteriors

Priors and Posteriors



We choose 2,1 as our prior.

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

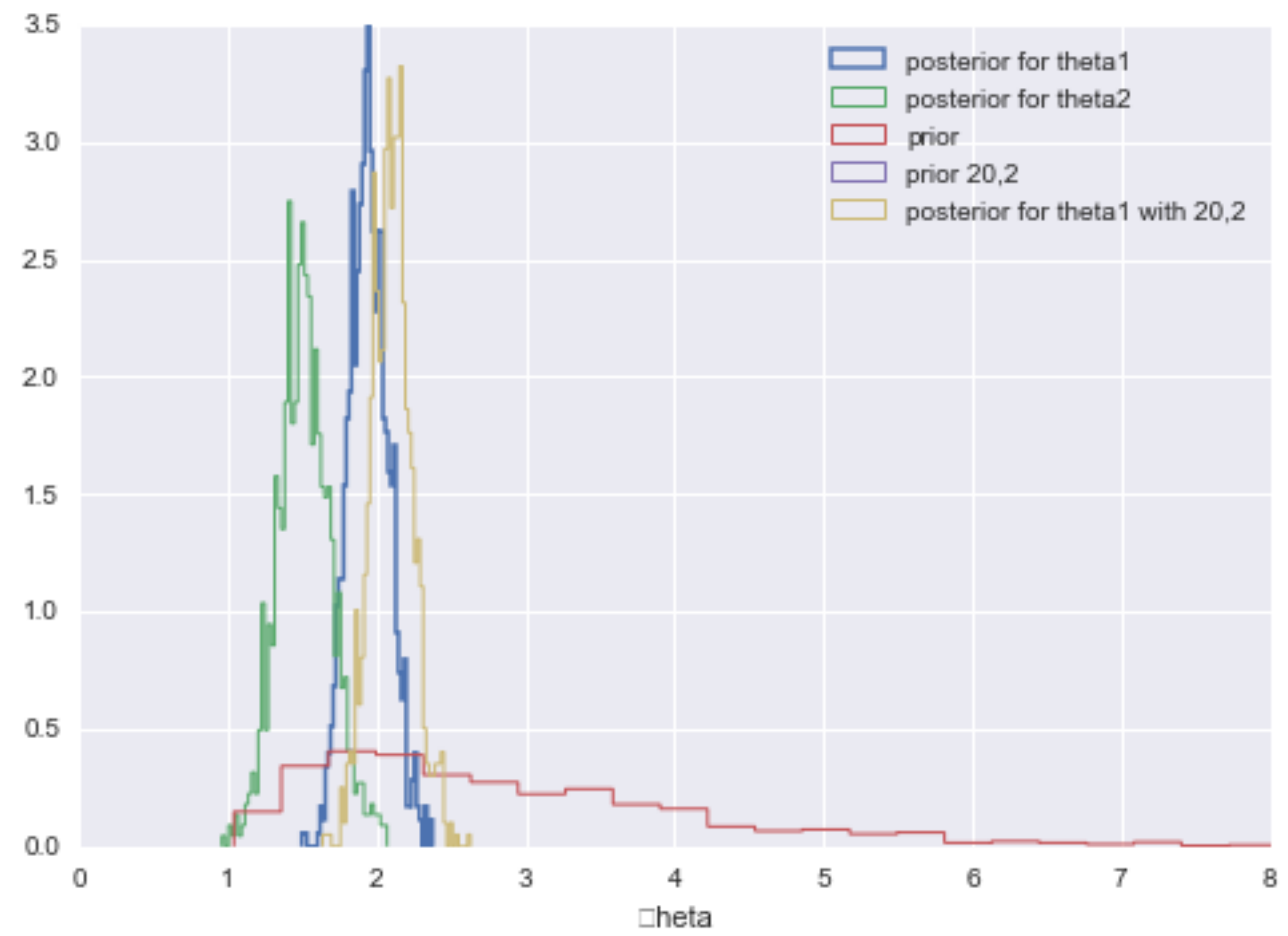
$$p(\theta_2 | n_2, \sum_i^{n_2} Y_{i,2}) \sim \text{Gamma}(\theta_2, 68, 45)$$

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:

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

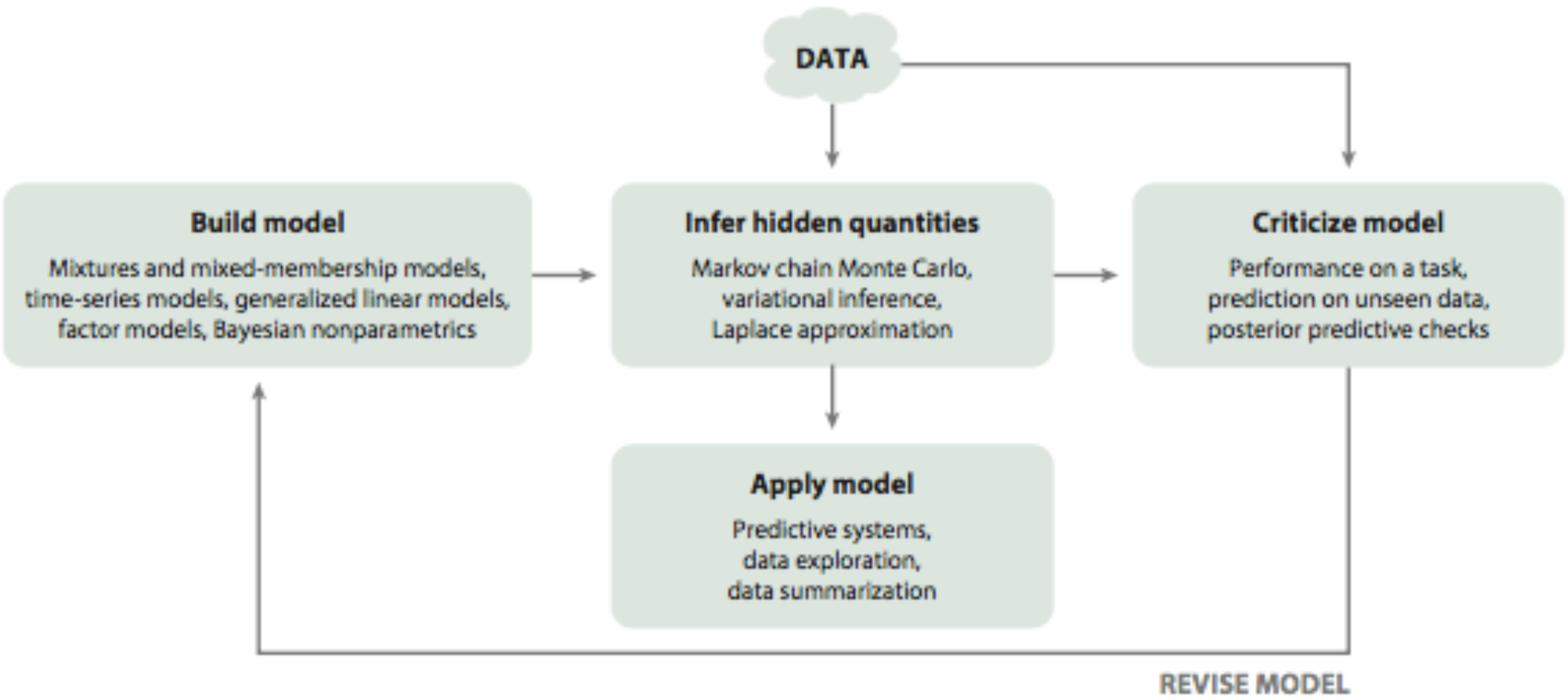
$$var[y^*] = \frac{(a + \sum y_i)}{(b + N)^2}(N + b + 1).$$

But see width:

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

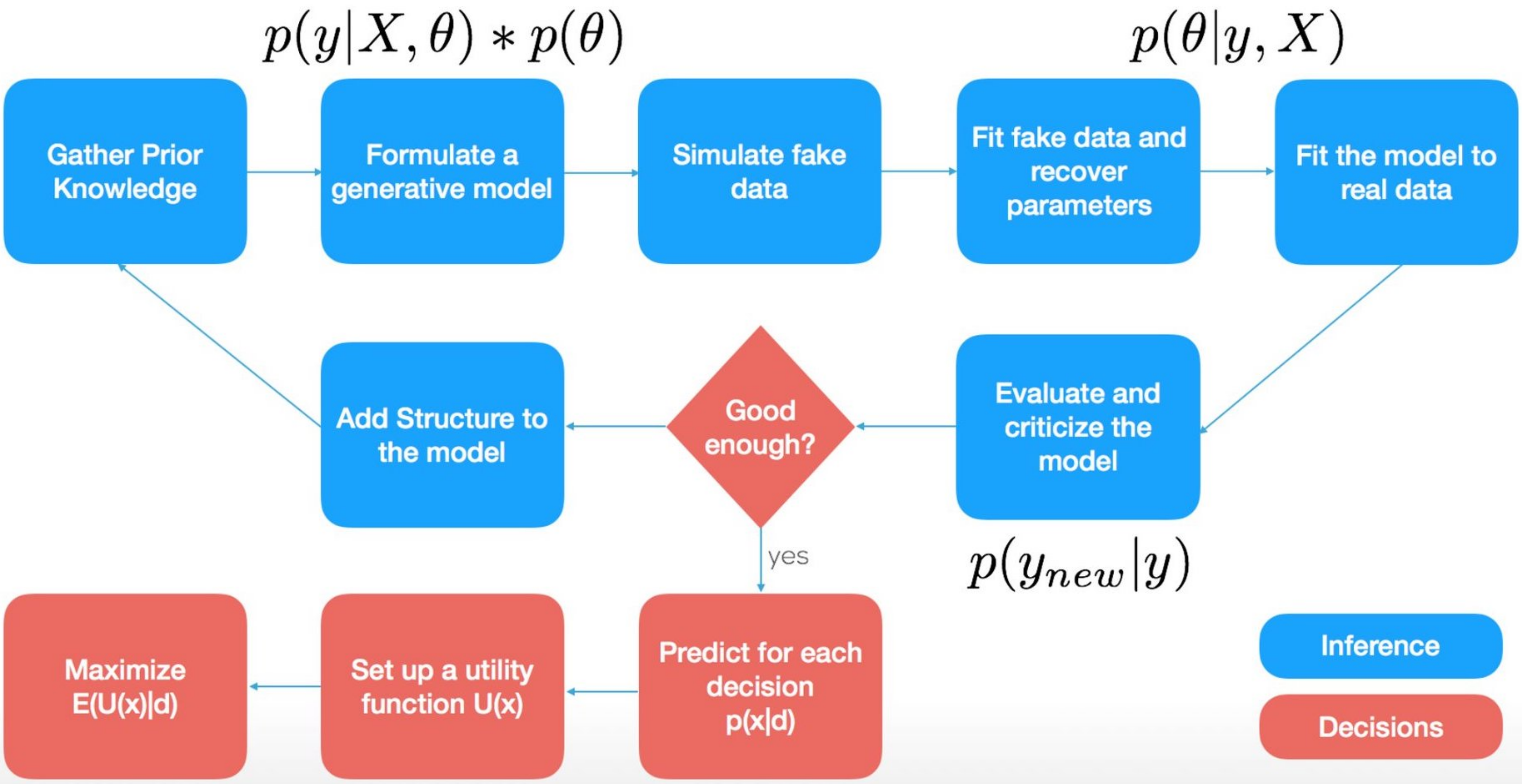
Posterior predictive smears out posterior error with sampling distribution

Box's loop

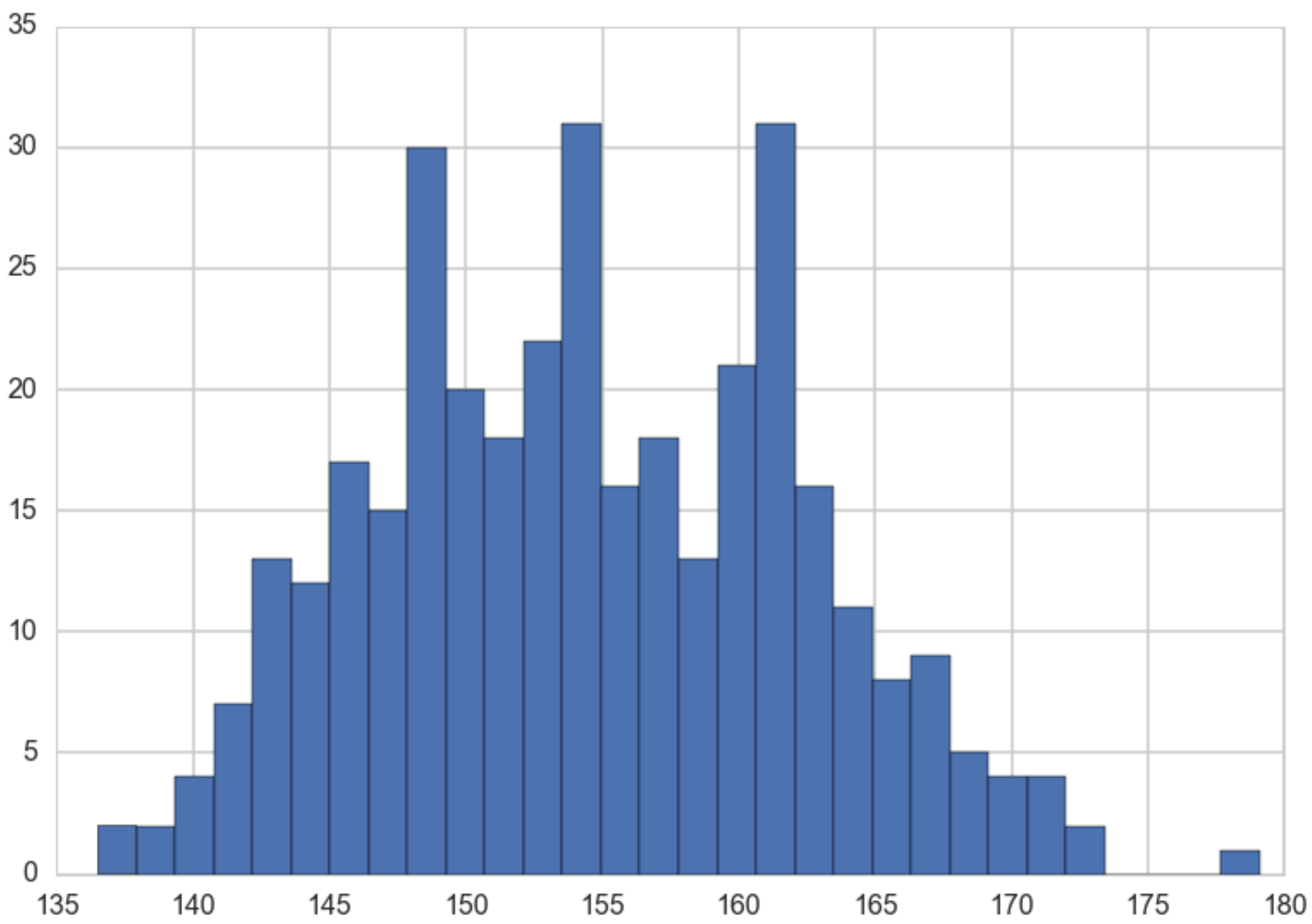

DATA
Build model
Mixtures and mixed-membership models, time-series models, generalized linear models, factor models, Bayesian nonparametrics
Infer hidden quantities
Markov chain Monte Carlo, variational inference, Laplace approximation
Criticize model
Performance on a task, prediction on unseen data,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

(from @ericnovik)

Bayesian Workflow



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

$$h \sim N(\mu, \sigma)$$
$$\mu \sim Normal(148, 20)$$
$$\sigma = samplestd$$

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...

Normal-Normal Posterior Predictive

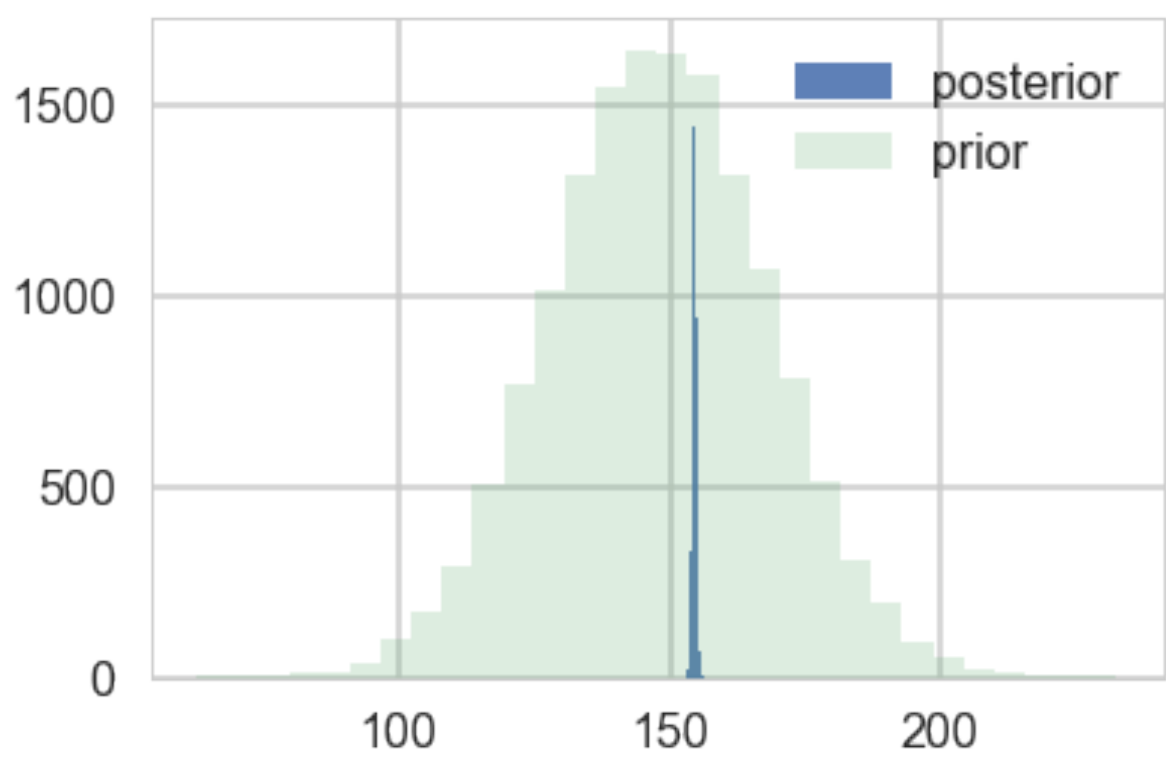
So the posterior is

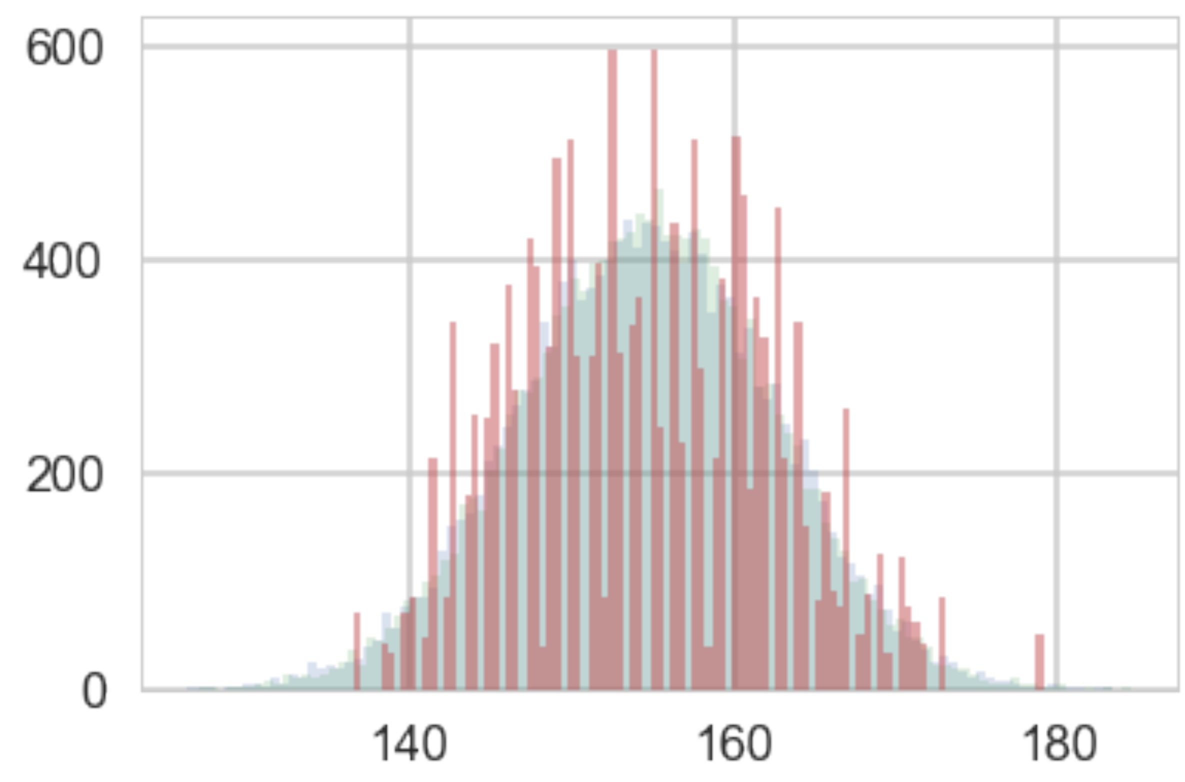
$$p(\mu \mid \{y\}, \sigma^2) = N(\mu_p, \tau_p^2)$$

The corresponding posterior predictive is:

$$p(y^* \mid \{y\}) = N(\mu_p, \tau_p^2 + \sigma^2)$$

Predictive variance is uncertainty due to the obsv. noise plus uncertainty due to the parameters.


posterior
prior
1500
1000
500
0
100
150
200


600
400
200
0
140
160
180

Bayesian Formulation of Regression

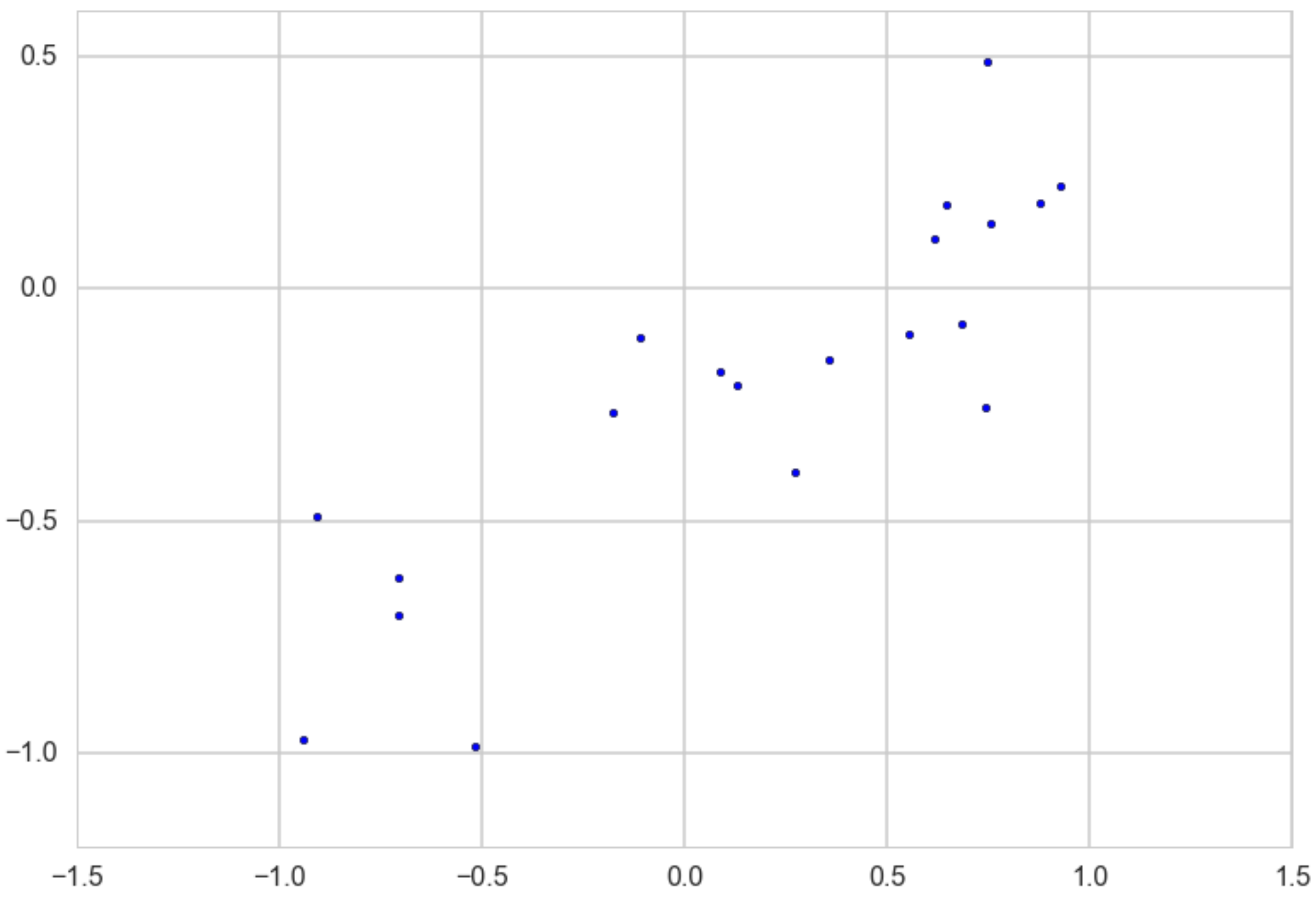
Data
$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$

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

Model:

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

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



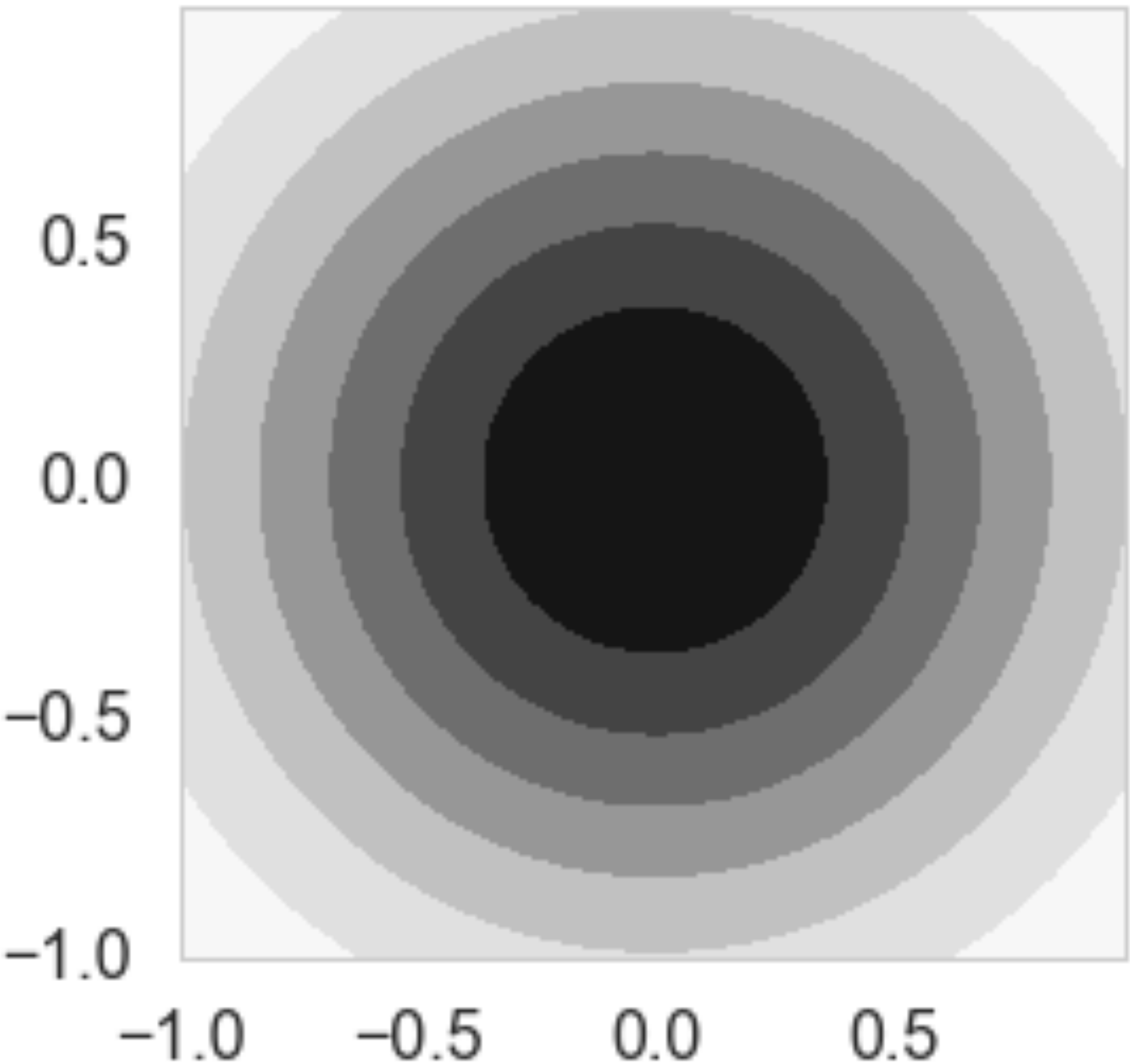
Likelihood

The likelihood is, because we assume independency, the product

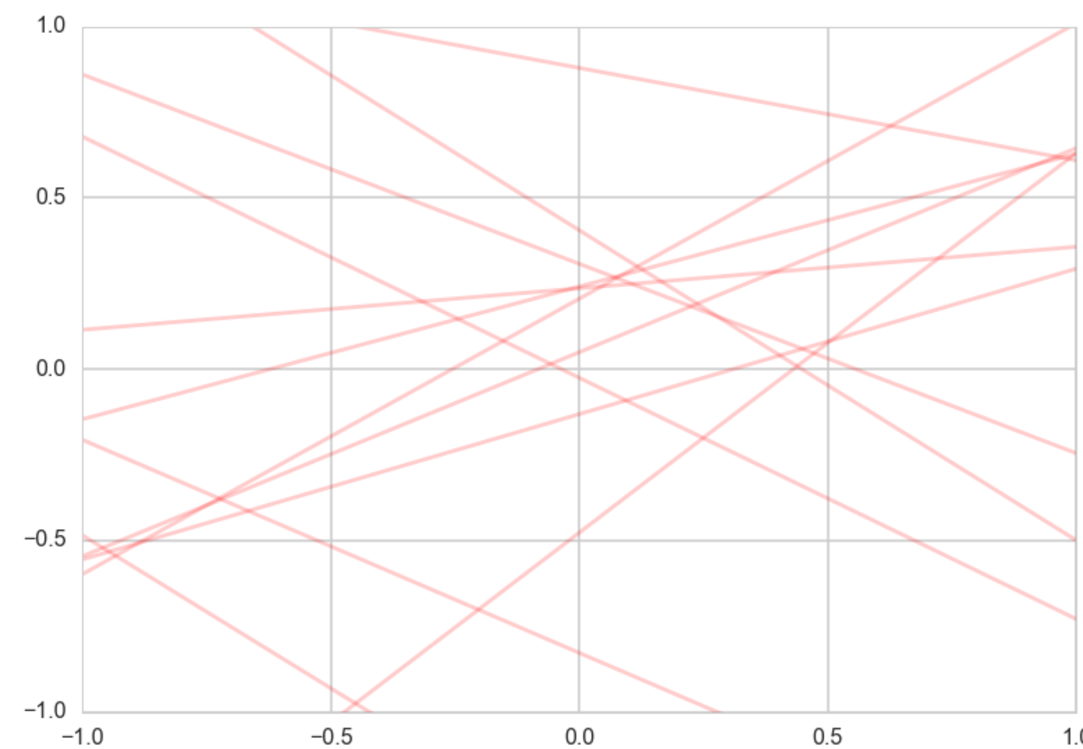
$$\mathcal{L} = p(\mathbf{y}|\mathbf{X},\mathbf{w}) = \prod_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{p(y_i|X_i,w)} = \prod_{\mathbf{i=1}}^{\mathbf{n}} \frac{\mathbf{1}}{\sqrt{\mathbf{2}\pi}\sigma_{\mathbf{n}}} \exp\left(-\frac{\mathbf{(y_i - X_i^T w)^2}}{\mathbf{2}\sigma_{\mathbf{n}}^{\mathbf{2}}}\right)$$

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

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



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



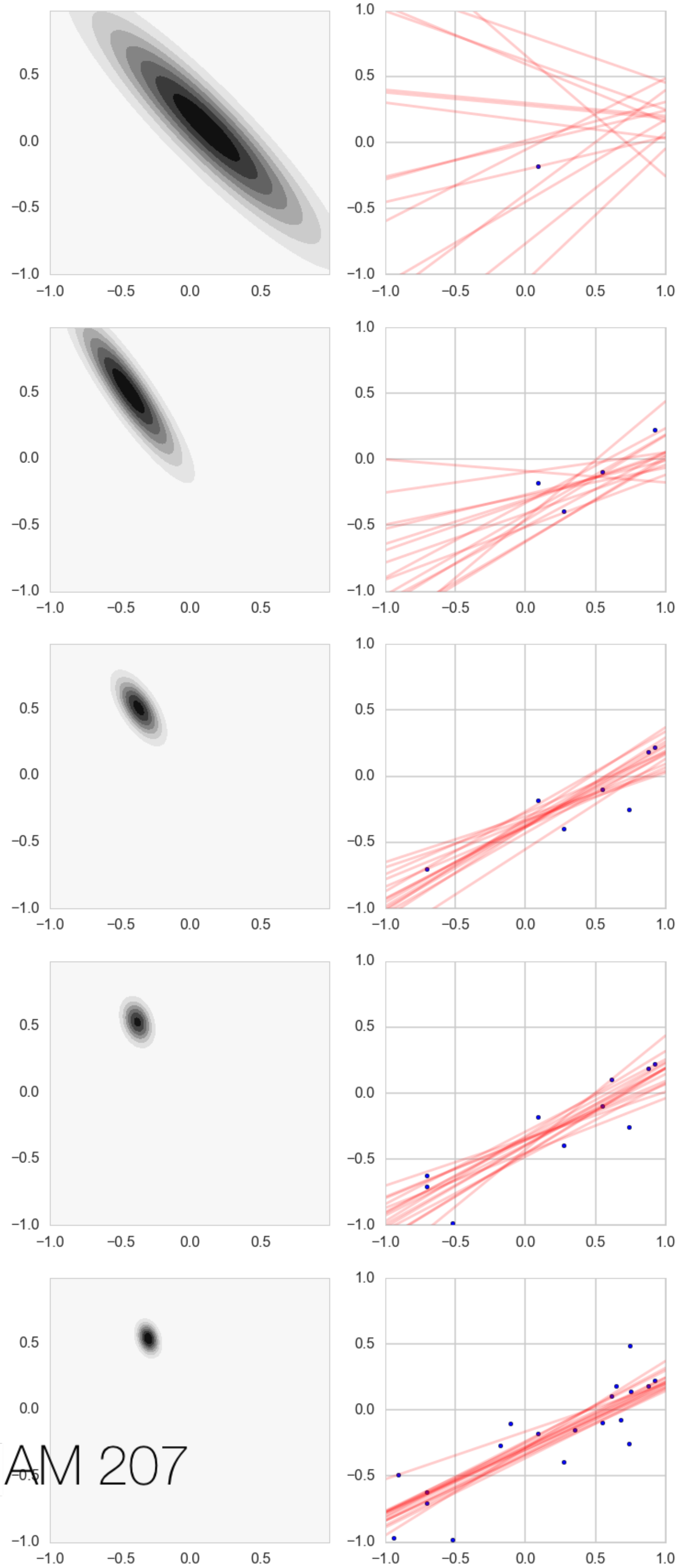
Posterior

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

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

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

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)
    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
```

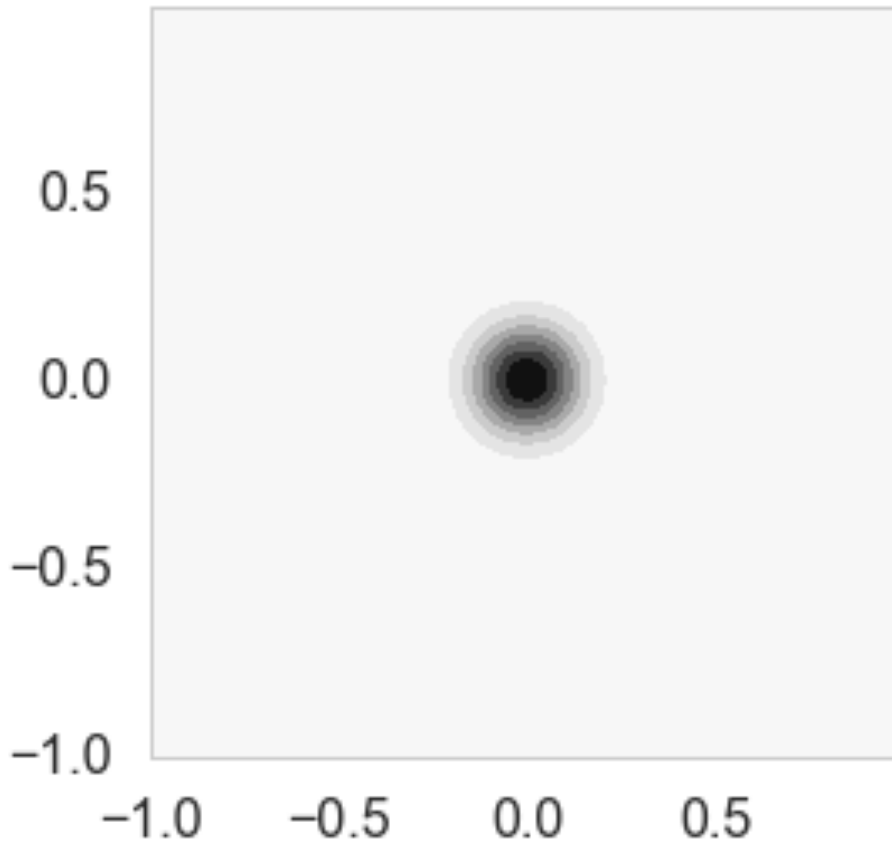
Posterior Predictive

$$p(y^* | x^*, \mathbf{x}, \mathbf{y}) =$$

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

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

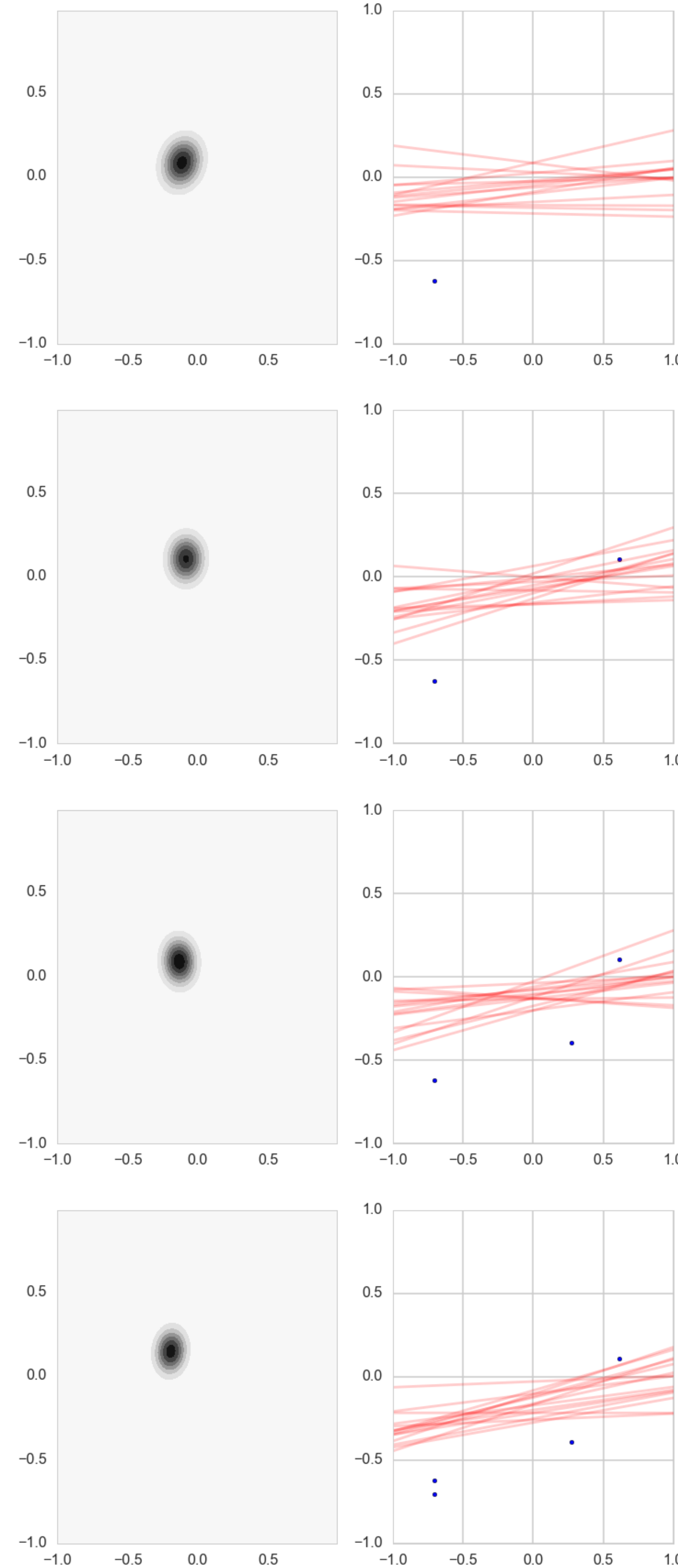
Regularization



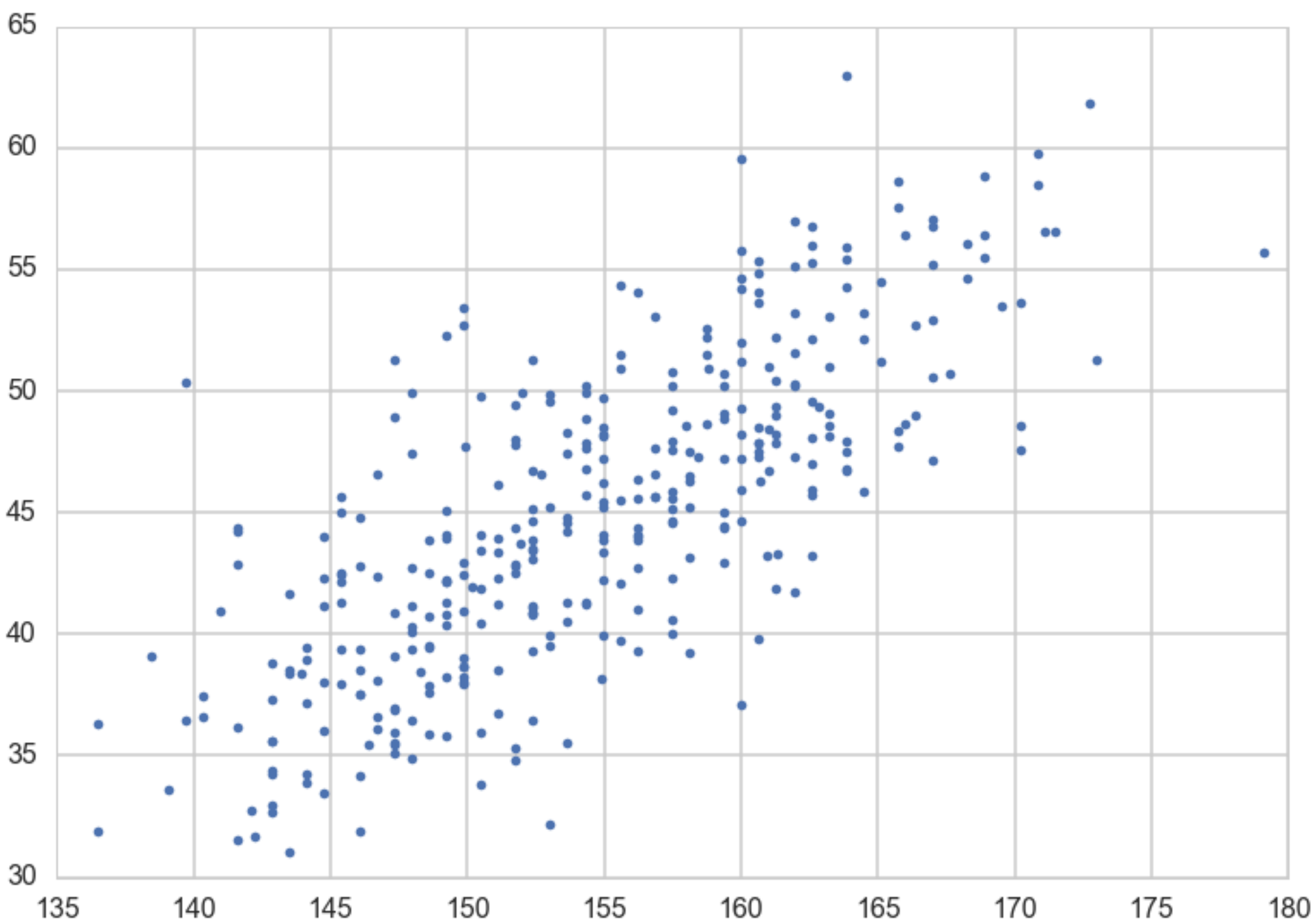

```
priorPrecision/likelihoodPrecision
```

```
4.0
```

This ratio is the ridge α.

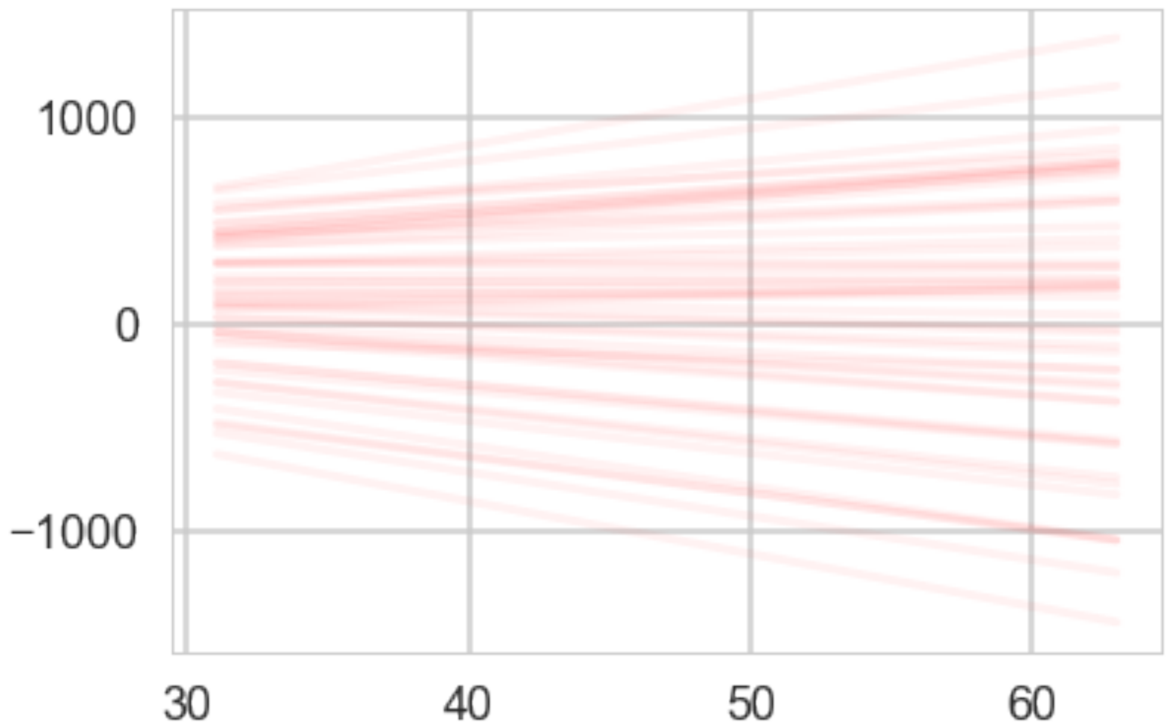


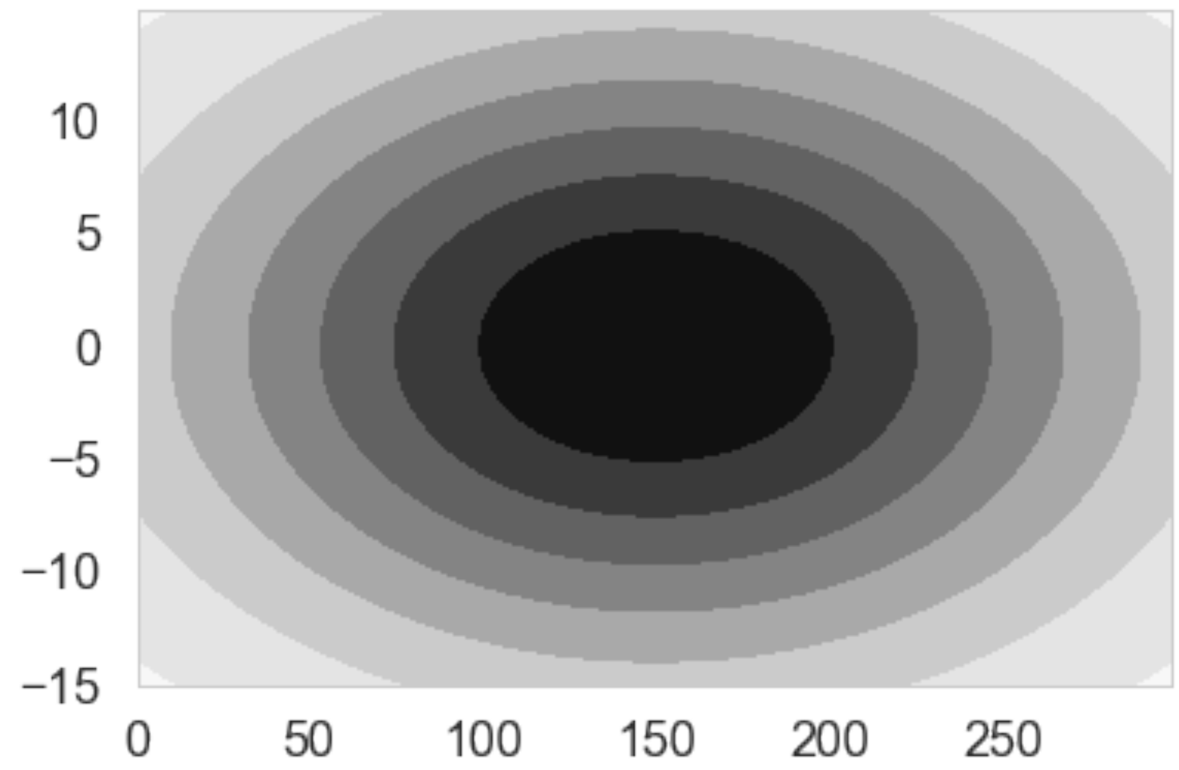
Regression, adding a predictor, weight



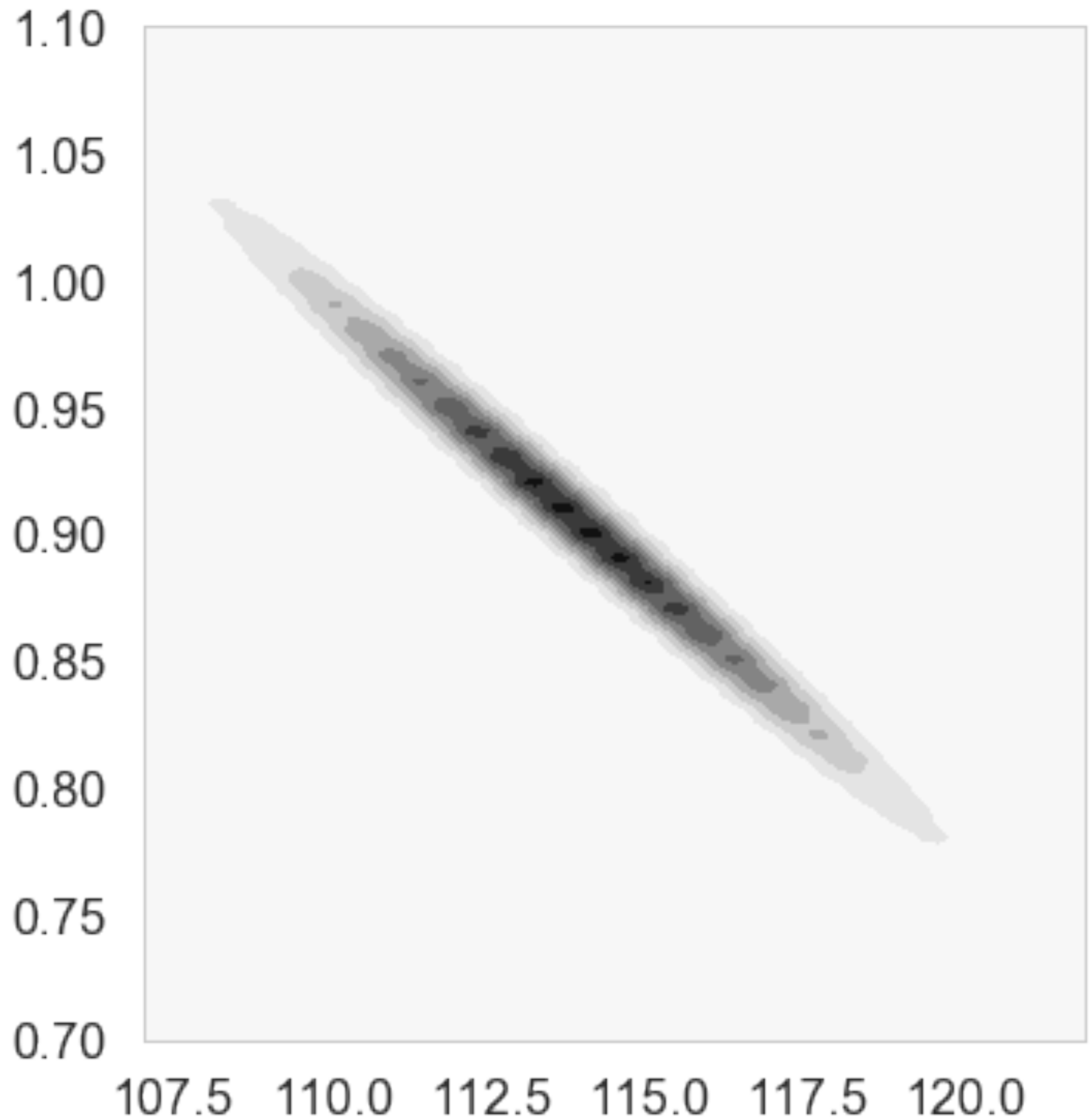
$$h \sim N(\mu, \sigma)$$
$$\mu = intercept + slope \times weight$$
$$intercept \sim N(150, 100)$$
$$slope \sim N(0, 10)$$
$$\sigma = std.dev$$

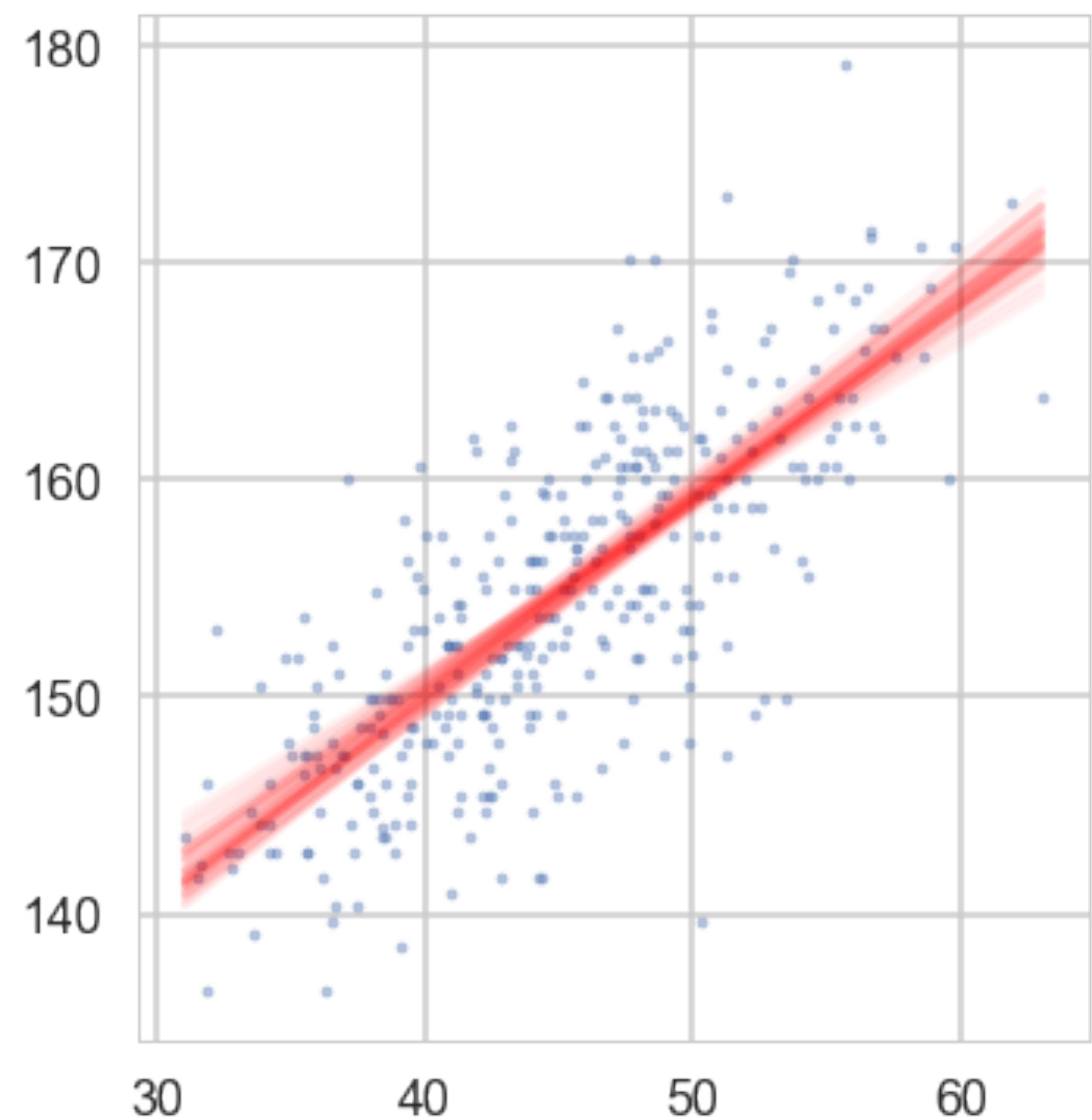
Priors





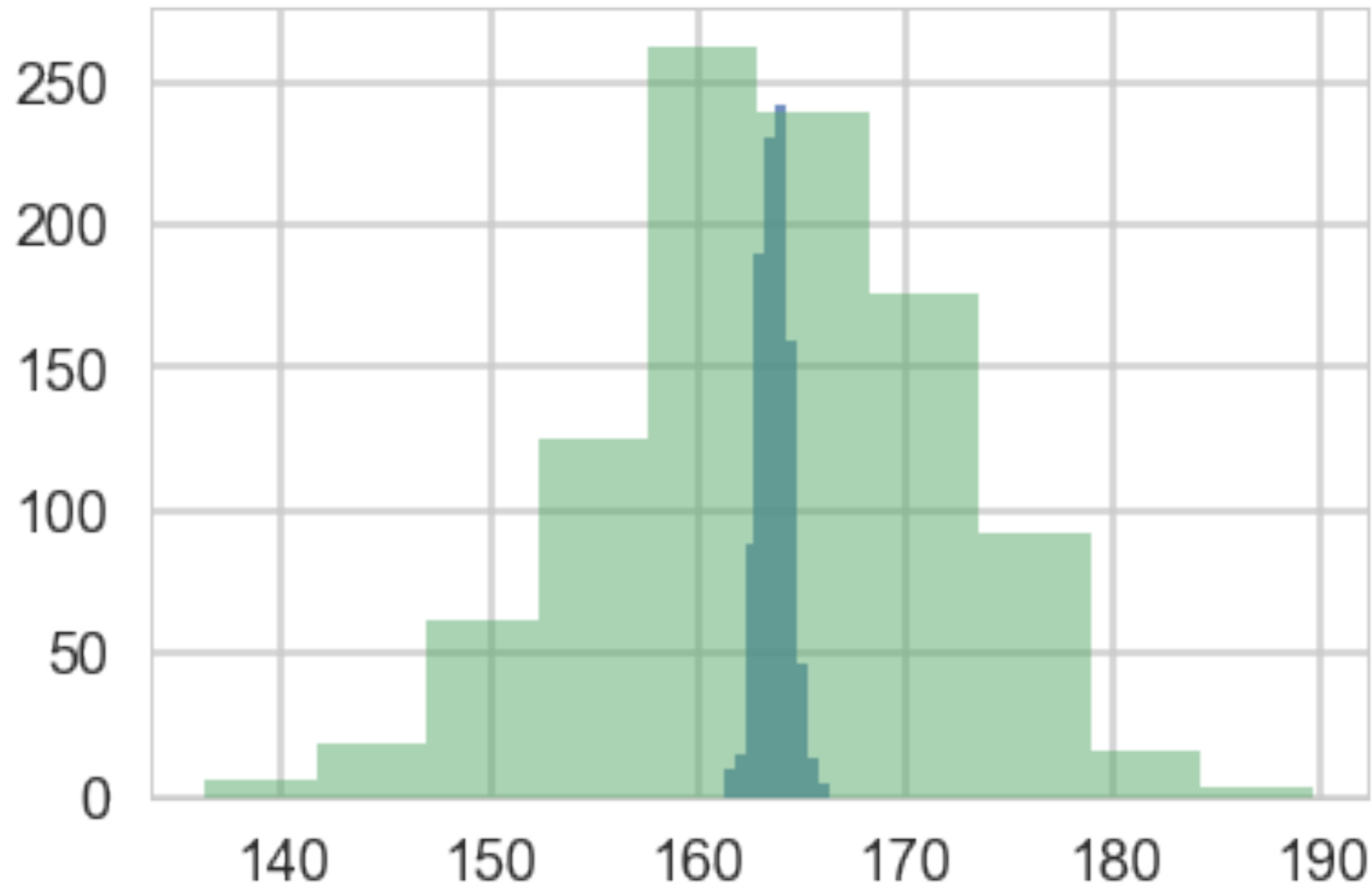
Posteriors


1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
107.5
110.0
112.5
115.0
117.5
120.0

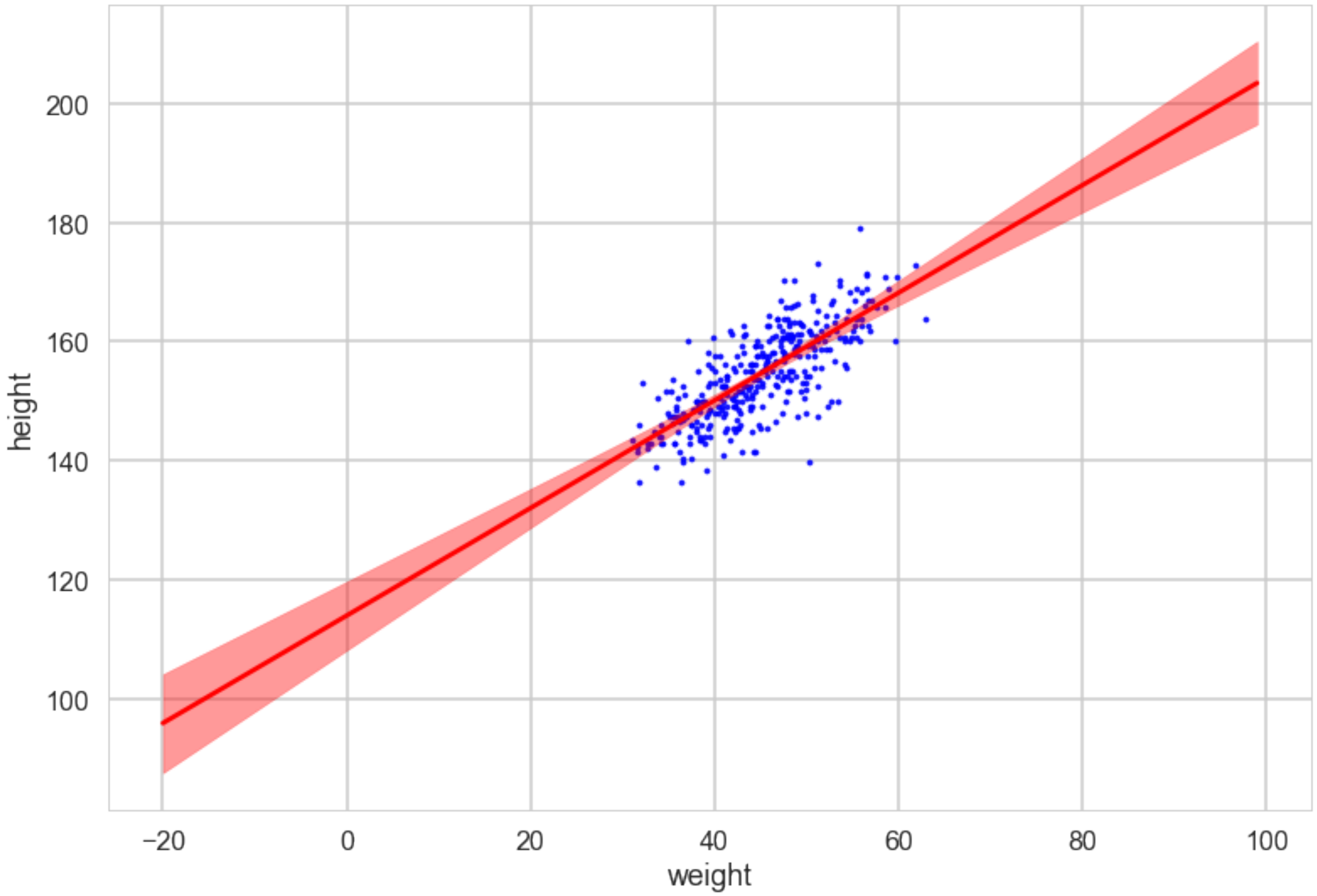

180
170
160
150
140
30
40
50
60

Posterior at weight 55

DO INTERCEPT SLOPE, AND WEIGHT 55

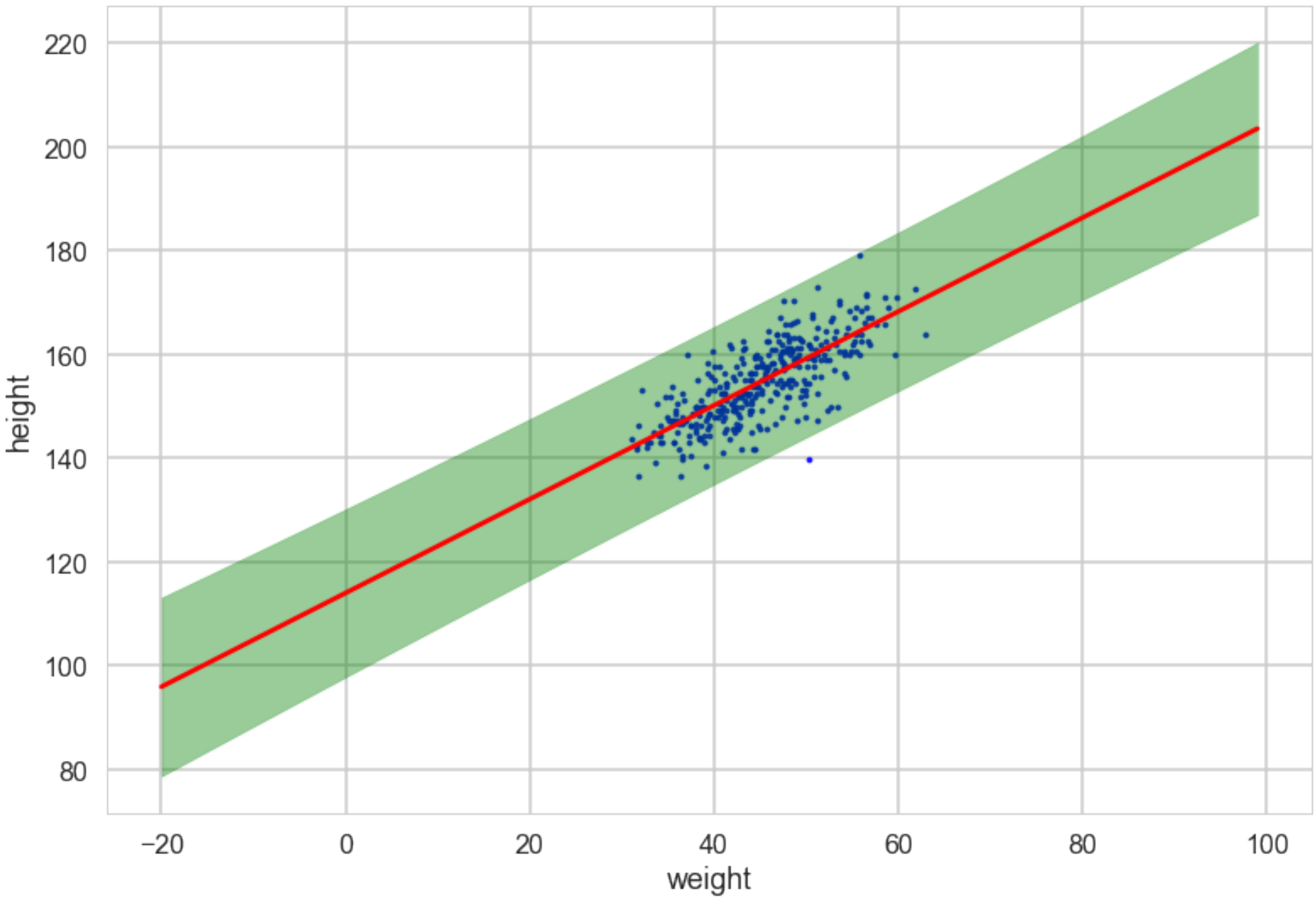


Posteriors on a grid



Why so tight?

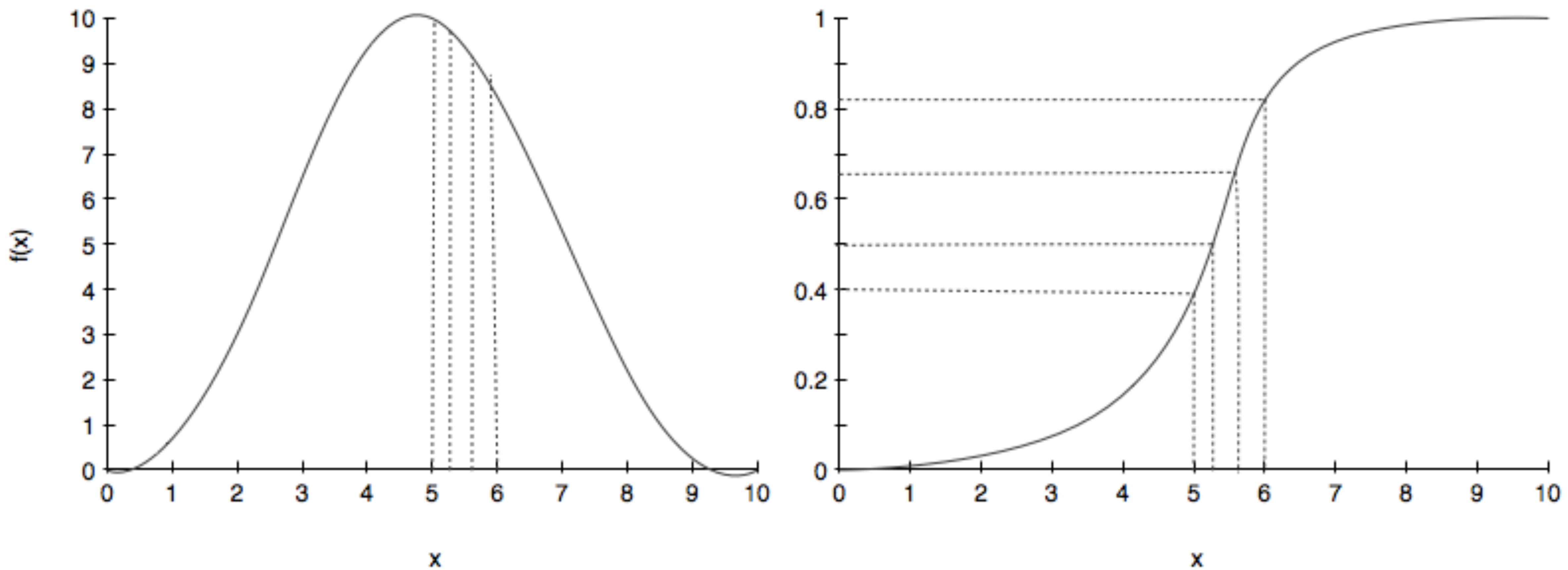
Predictives on grid



Ok. We need Samples

- to compute expectations, integrals and do statistics, we need samples
- we start that journey today
- inverse transform
- rejection sampling
- importance sampling: a direct, low-variance way to do integrals and expectations

Inverse transform



algorithm

The CDF F must be invertible!

1. get a uniform sample u from $Unif(0,1)$
2. solve for x yielding a new equation $x = F^{-1}(u)$ where F is the CDF of the distribution we desire.
3. repeat.

Why does it work?

$$F^{-1}(u) = \text{smallest x such that } F(x) >= u$$

What distribution does random variable $y = F^{-1}(u)$ follow?

The CDF of y is $p(y <= x)$. Since F is monotonic:

$$p(y <= x) = p(F(y) <= F(x)) = p(u <= F(x)) = F(x)$$

F is the CDF of y, thus f is the pdf.

Example: exponential

pdf: $f(x) = \frac{1}{\lambda} e^{-x/\lambda}$ for $x \geq 0$ and $f(x) = 0$ otherwise.

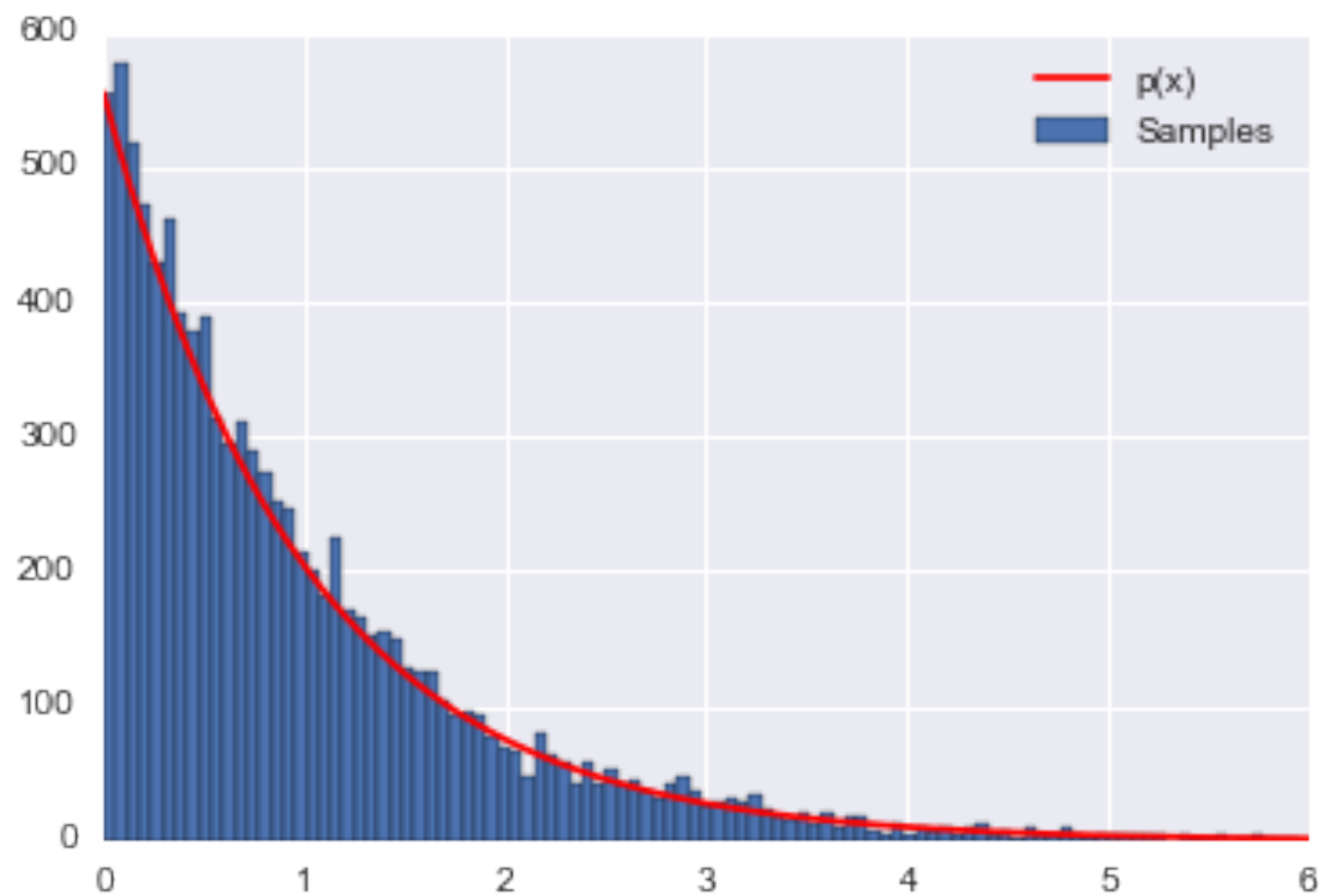
$$u = \int_0^x \frac{1}{\lambda} e^{-x'/\lambda} dx' = 1 - e^{-x/\lambda}$$

Solving for x

$$x = -\lambda \ln(1 - u)$$

code

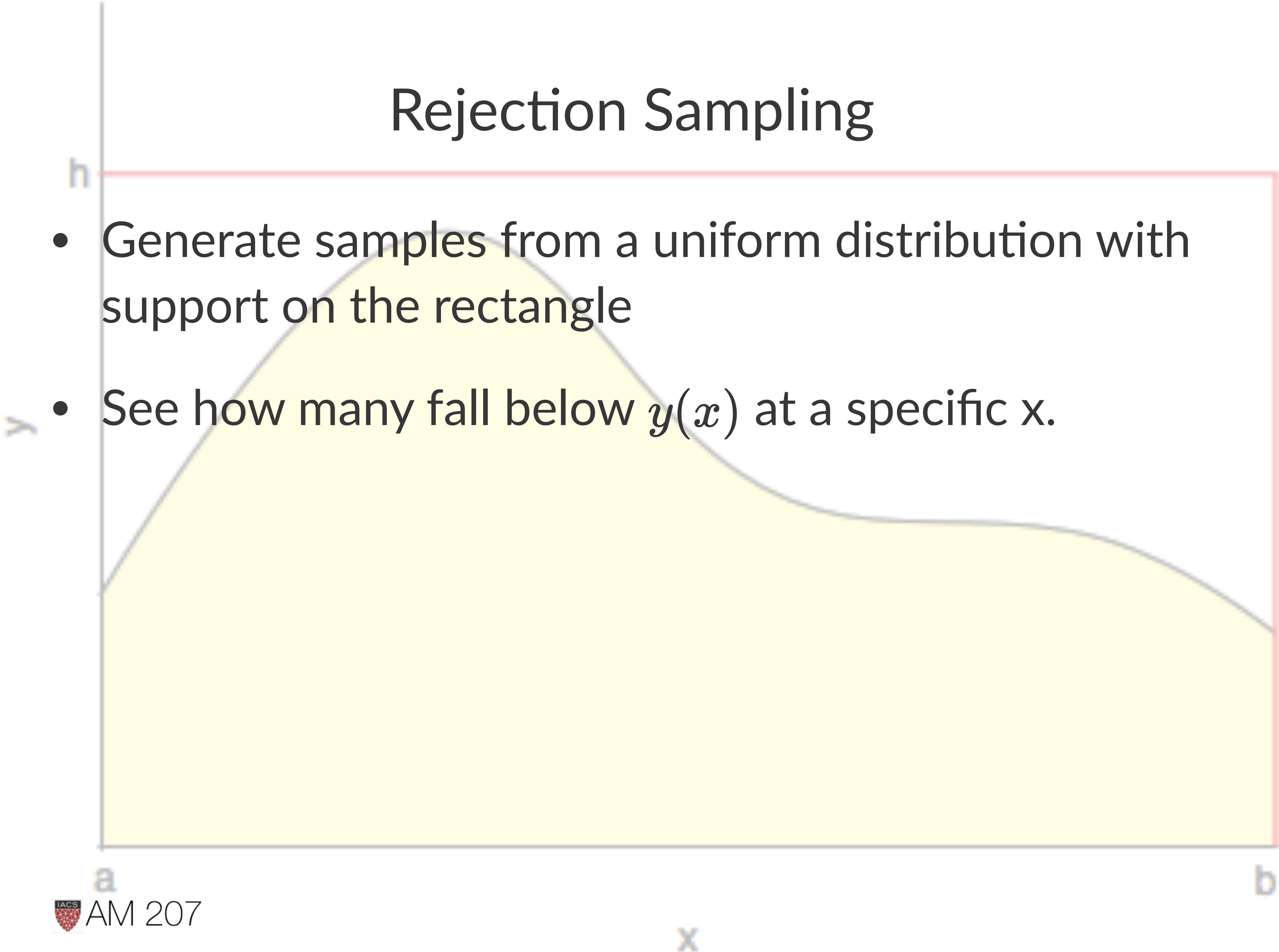
```
p = lambda x: np.exp(-x)
CDF = lambda x: 1-np.exp(-x)
invCDF = lambda r: -np.log(1-r) # invert the CDF
xmin = 0 # the lower limit of our domain
xmax = 6 # the upper limit of our domain
rmin = CDF(xmin)
rmax = CDF(xmax)
N = 10000
# generate uniform samples in our range then invert the CDF
# to get samples of our target distribution
R = np.random.uniform(rmin, rmax, N)
X = invCDF(R)
hinfo = np.histogram(X,100)
plt.hist(X,bins=100, label=u'Samples');
# plot our (normalized) function
xvals=np.linspace(xmin, xmax, 1000)
plt.plot(xvals, hinfo[0][0]*p(xvals), 'r', label=u'p(x)')
plt.legend()
```



Rejection Sampling

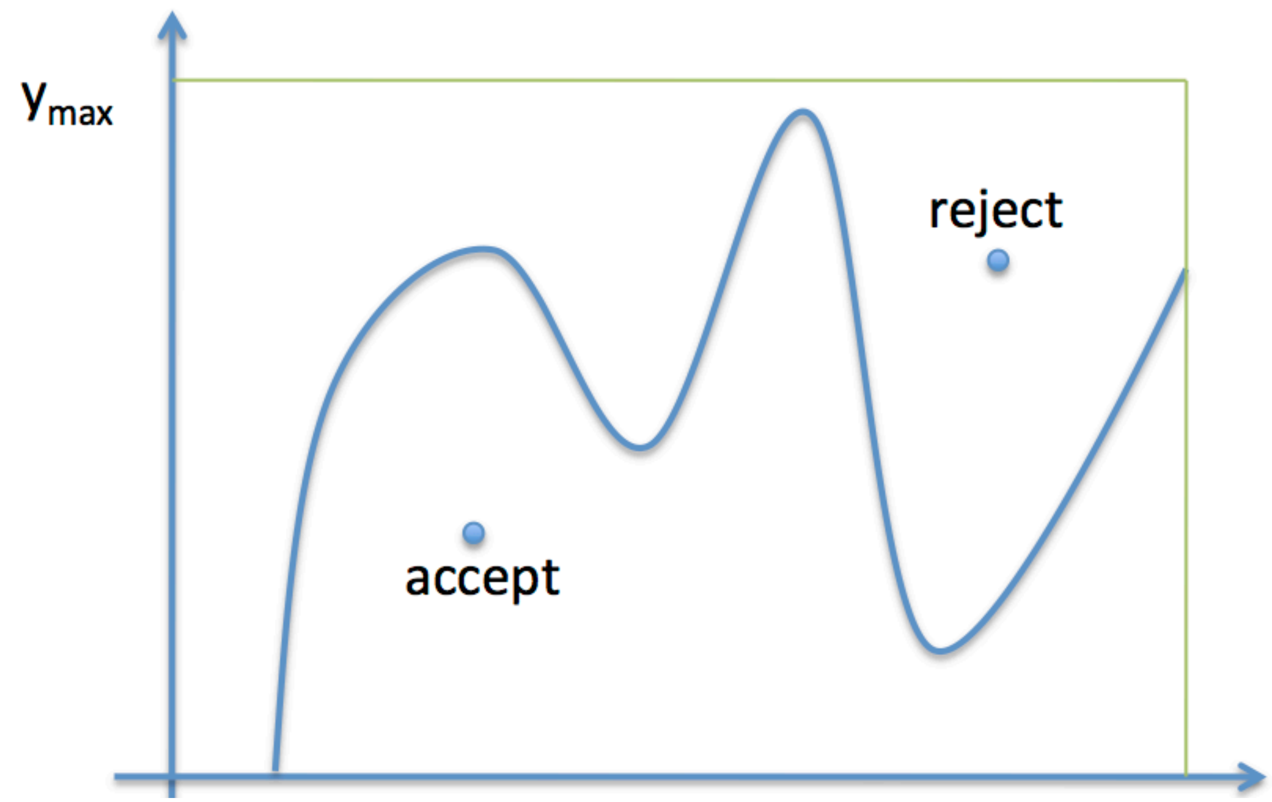
- Generate samples from a uniform distribution with support on the rectangle

- See how many fall below $y(x)$ at a specific x.



Algorithm

1. Draw x uniformly from $[x_{min},\, x_{max}]$
2. Draw y uniformly from $[0,\, y_{max}]$
3. if $y < f(x)$, accept the sample
4. otherwise reject it
5. repeat



example

```
P = lambda x: np.exp(-x)
xmin = 0 # the lower limit of our domain
xmax = 10 # the upper limit of our domain
ymax = 1
#you might have to do an optimization to find this.
N = 10000 # the total of samples we wish to generate
accepted = 0 # the number of accepted samples
samples = np.zeros(N)
count = 0 # the total count of proposals

while (accepted < N):
    # pick a uniform number on [xmin, xmax) (e.g. 0...10)
    x = np.random.uniform(xmin, xmax)
    # pick a uniform number on [0, ymax)
    y = np.random.uniform(0,ymax)
    # Do the accept/reject comparison
    if y < P(x):
        samples[accepted] = x
        accepted += 1

    count +=1

print("Count",count, "Accepted", accepted)
hinfo = np.histogram(samples,30)
plt.hist(samples,bins=30, label=u'Samples');
xvals=np.linspace(xmin, xmax, 1000)
plt.plot(xvals, hinfo[0][0]*P(xvals), 'r', label=u'P(x)')
plt.legend()
```


```
Count 100294 Accepted 10000
```

problems

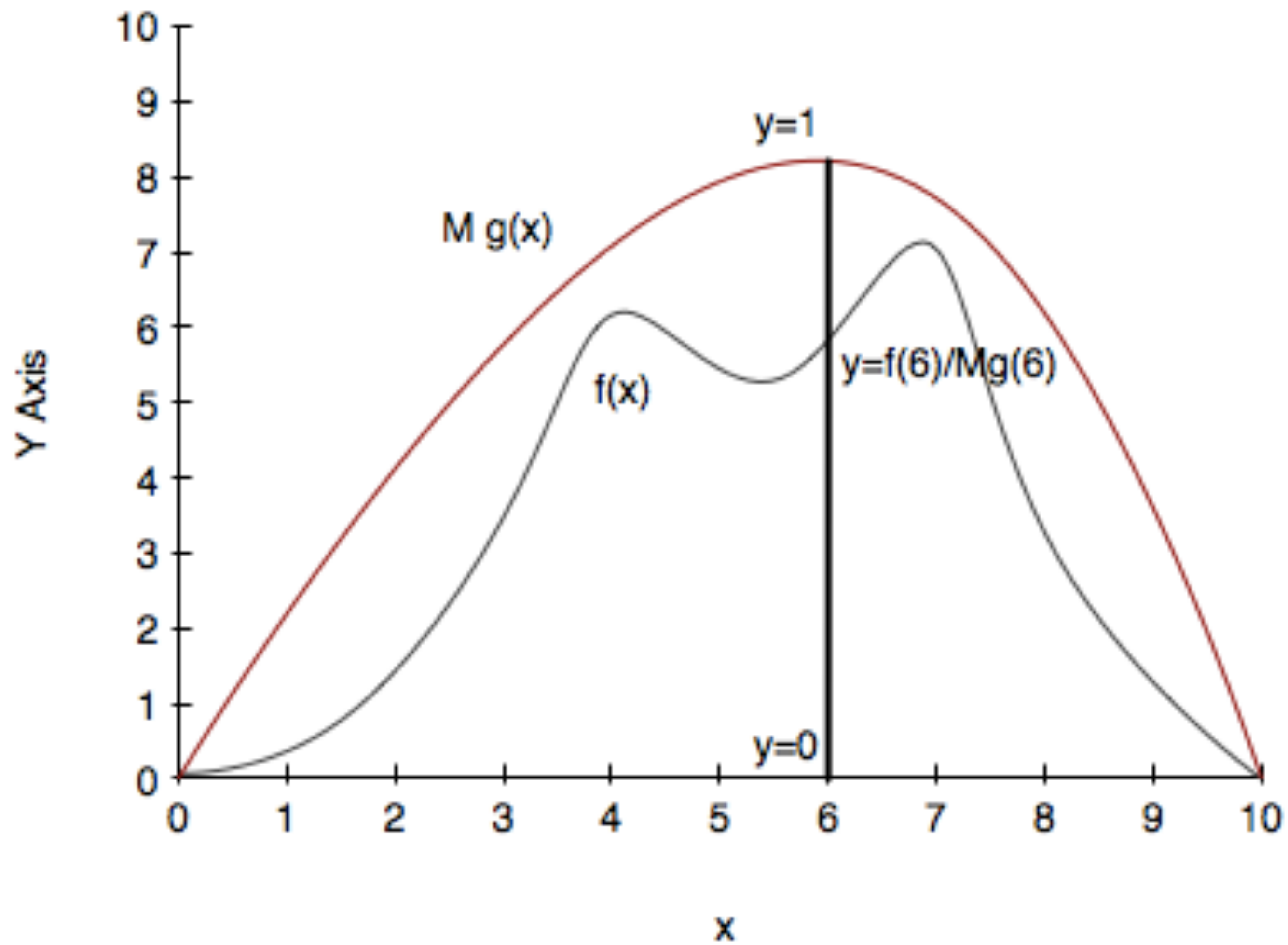
- determining the supremum may be costly
- the functional form may be complex for comparison
- even if you find a tight bound for the supremum, basic rejection sampling is very inefficient: **low acceptance probability**
- infinite support

Variance Reduction

Rejection on steroids

Introduce a **proposal density** $g(x)$.

- $g(x)$ is easy to sample from and (calculate the pdf)
- Some M exists so that $M\,g(x) > f(x)$ in your entire domain of interest
- ideally $g(x)$ will be somewhat close to f
- optimal value for M is the supremum over your domain



Algorithm

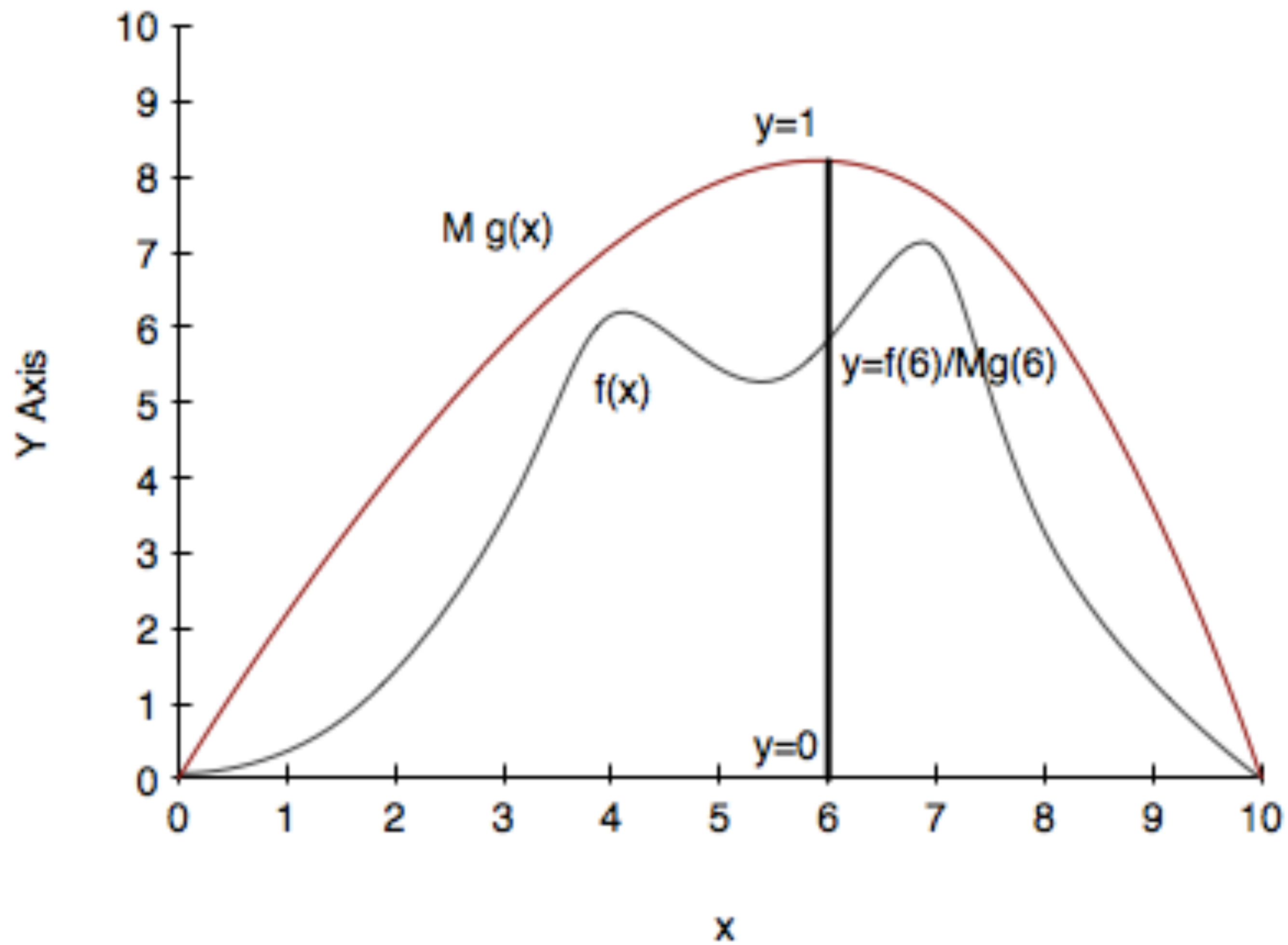
1. Draw x from your proposal distribution $g(x)$

2. Draw y uniformly from [0,1]

3. if $y < f(x)/M\, g(x)$, accept the sample

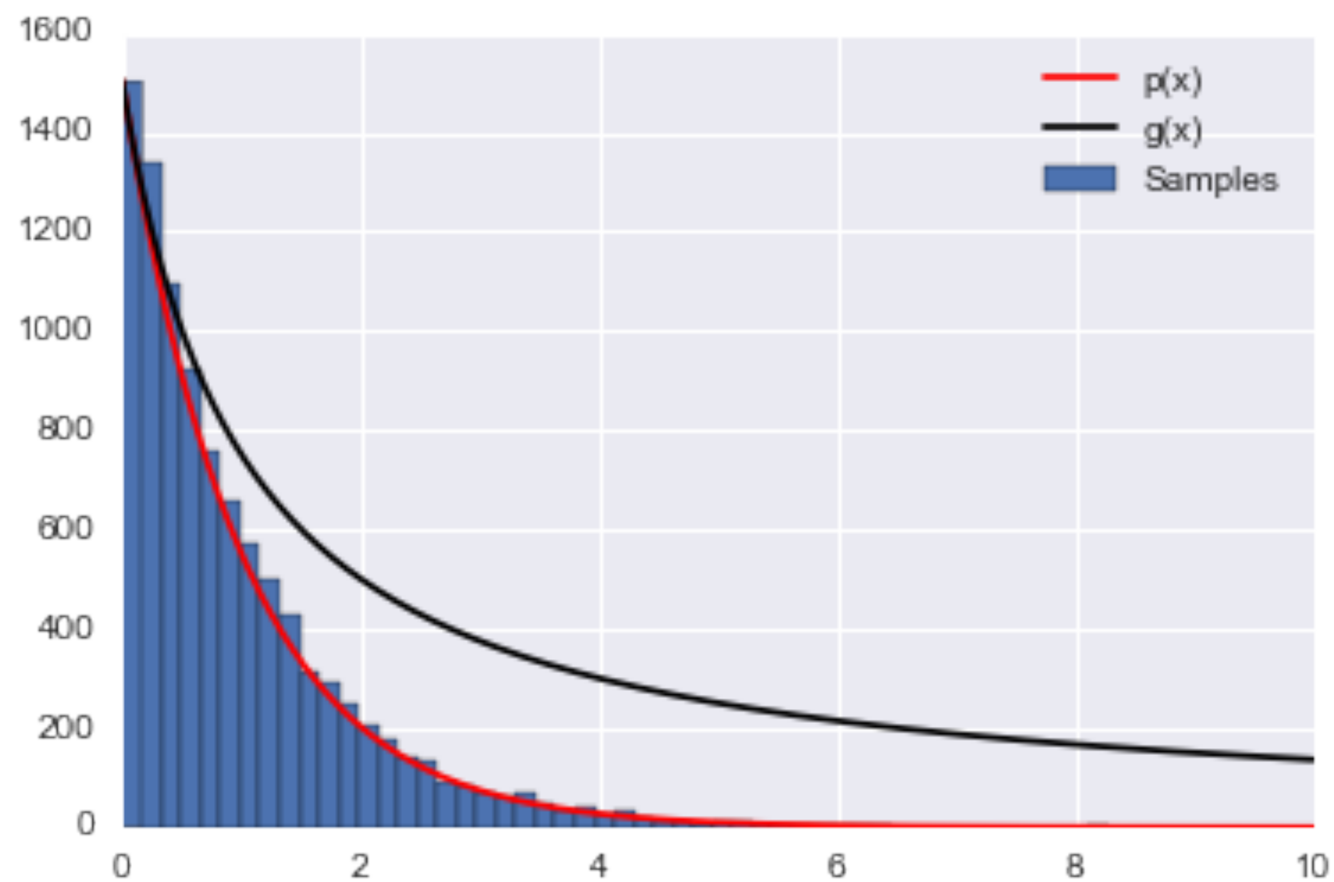
4. otherwise reject it

5. repeat



Example

```python
p = lambda x: np.exp(-x)  # our distribution
g = lambda x: 1/(x+1)  # our proposal pdf (we're thus choosing M to be 1)
invCDFg = lambda x: np.log(x +1) # generates our proposal using inverse sampling
xmin = 0 # the lower limit of our domain
xmax = 10 # the upper limit of our domain
# range limits for inverse sampling
umin = invCDFg(xmin)
umax = invCDFg(xmax)
N = 10000 # the total of samples we wish to generate
accepted = 0 # the number of accepted samples
samples = np.zeros(N)
count = 0 # the total count of proposals

while (accepted < N):

    # Sample from g using inverse sampling
    u = np.random.uniform(umin, umax)
    xproposal = np.exp(u) - 1

    # pick a uniform number on [0, 1)
    y = np.random.uniform(0,1)

    # Do the accept/reject comparison
    if y < p(xproposal)/g(xproposal):
        samples[accepted] = xproposal
        accepted += 1

    count +=1

print("Count", count, "Accepted", accepted)
# get the histogram info
hinfo = np.histogram(samples,50)
plt.hist(samples,bins=50, label=u'Samples');
xvals=np.linspace(xmin, xmax, 1000)
plt.plot(xvals, hinfo[0][0]*p(xvals), 'r', label=u'p(x)')
plt.plot(xvals, hinfo[0][0]*g(xvals), 'k', label=u'g(x)')
plt.legend()
```




```
Count 23809 Accepted 10000
```

MLE for Logistic Regression

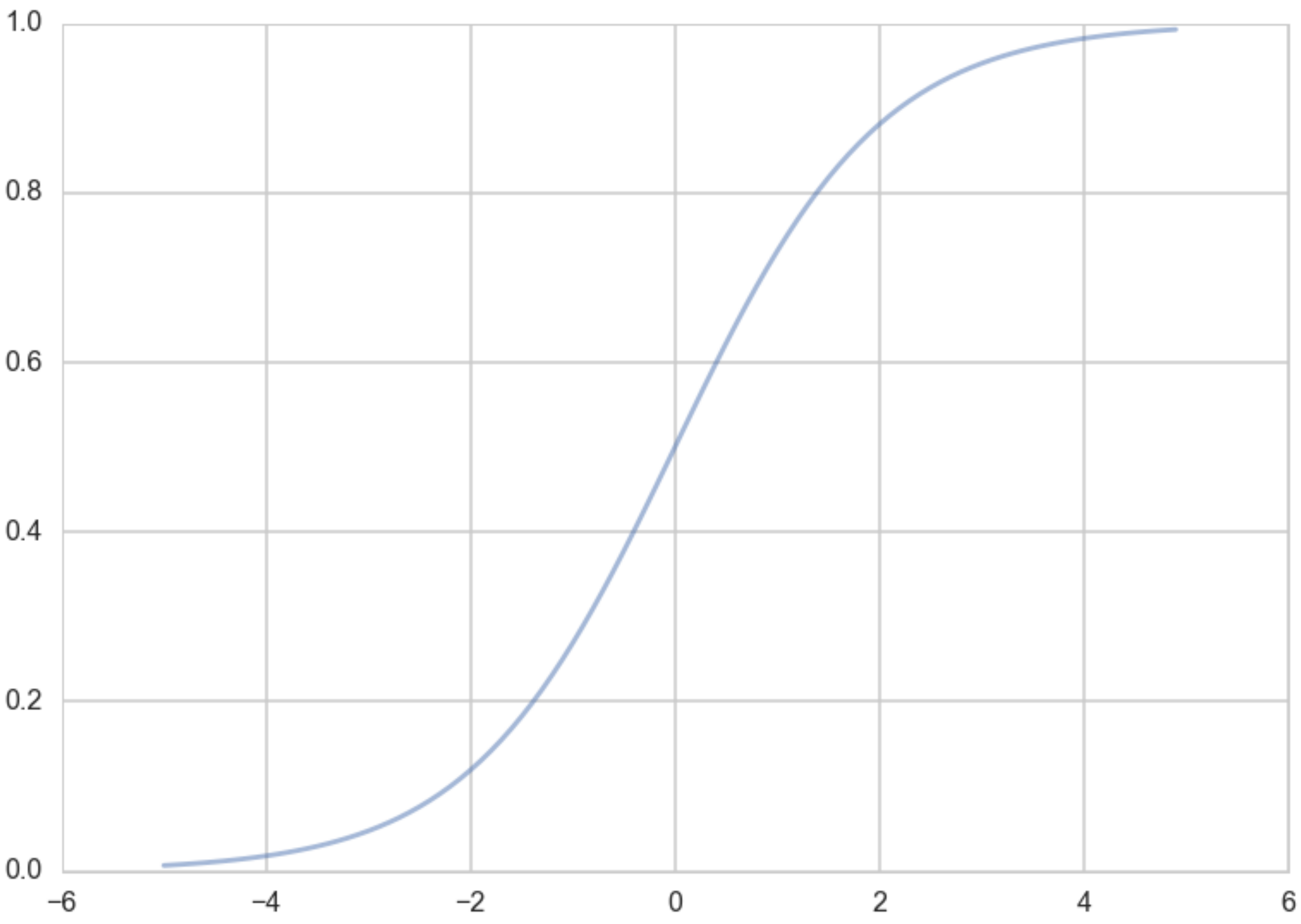
- example of a Generalized Linear Model (GLM)
- "Squeeze" linear regression through a **Sigmoid** function
- this bounds the output to be a probability
- What is the sampling Distribution?

Sigmoid function

This function is plotted below:

```
h = lambda z: 1./(1+np.exp(-z))
zs=np.arange(-5,5,0.1)
plt.plot(zs, h(zs), alpha=0.5);
```

Identify: $z = \mathbf{w} \cdot \mathbf{x}$ and $h(\mathbf{w} \cdot \mathbf{x})$ with the probability that the sample is a '1' ($y = 1$).



Then, the conditional probabilities of $y = 1$ or $y = 0$ given a particular sample's features $\mathbf{x}$ are:

$$P(y=1|\mathbf{x}) = h(\mathbf{w}\cdot\mathbf{x})$$
$$P(y=0|\mathbf{x}) = 1 - h(\mathbf{w}\cdot\mathbf{x}).$$

These two can be written together as

$$P(y|\mathbf{x},\mathbf{w}) = h(\mathbf{w}\cdot\mathbf{x})^y (1 - h(\mathbf{w}\cdot\mathbf{x}))^{(1-y)}$$

BERNOULLI!!

Multiplying over the samples we get:

$$P(y|\mathbf{x},\mathbf{w}) = P(\{y_i\}|\{\mathbf{x}_i\},\mathbf{w}) = \prod_{y_i\in\mathcal{D}} P(y_i|\mathbf{x}_i,\mathbf{w}) = \prod_{y_i\in\mathcal{D}} h(\mathbf{w}\cdot\mathbf{x}_i)^{y_i}\left(1-h(\mathbf{w}\cdot\mathbf{x}_i)\right)^{(1-y_i)}$$

A noisy y is to imagine that our data $\mathcal{D}$ was generated from a joint probability distribution $P(x,y)$. Thus we need to model y at a given x, written as $P(y \mid x)$, and since $P(x)$ is also a probability distribution, we have:

$$P(x,y) = P(y \mid x)P(x),$$

Indeed its important to realize that a particular sample can be thought of as a draw from some "true" probability distribution.

maximum likelihood estimation maximises the **likelihood of the sample y**,

$$\mathcal{L} = P(y \mid \mathbf{x}, \mathbf{w}).$$

Again, we can equivalently maximize

$$\ell = log(P(y \mid \mathbf{x}, \mathbf{w}))$$

Thus

$$
\begin{aligned}
\ell &= log\left(\prod_{y_i \in \mathcal{D}} h(\mathbf{w}\cdot\mathbf{x}_i)^{y_i}(1-h(\mathbf{w}\cdot\mathbf{x}_i))^{(1-y_i)}\right) \\
&= \sum_{y_i \in \mathcal{D}} log\left(h(\mathbf{w}\cdot\mathbf{x}_i)^{y_i}(1-h(\mathbf{w}\cdot\mathbf{x}_i))^{(1-y_i)}\right) \\
&= \sum_{y_i \in \mathcal{D}} log\, h(\mathbf{w}\cdot\mathbf{x}_i)^{y_i} + log\,(1-h(\mathbf{w}\cdot\mathbf{x}_i))^{(1-y_i)} \\
&= \sum_{y_i \in \mathcal{D}} (y_i log(h(\mathbf{w}\cdot\mathbf{x})) + (1-y_i) log(1-h(\mathbf{w}\cdot\mathbf{x})))
\end{aligned}
$$