

# Lecture 13

# 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
- Gibbs can have strong correlations

# Today

- the normal-normal model with MCMC
- then with pymc3
- bayesian regression and updating
- regularization and the ridge
- from the normal model to regression using pymc
- posterior vs predictive in regression problems

# The levels of Bayesian analysis (from last time)

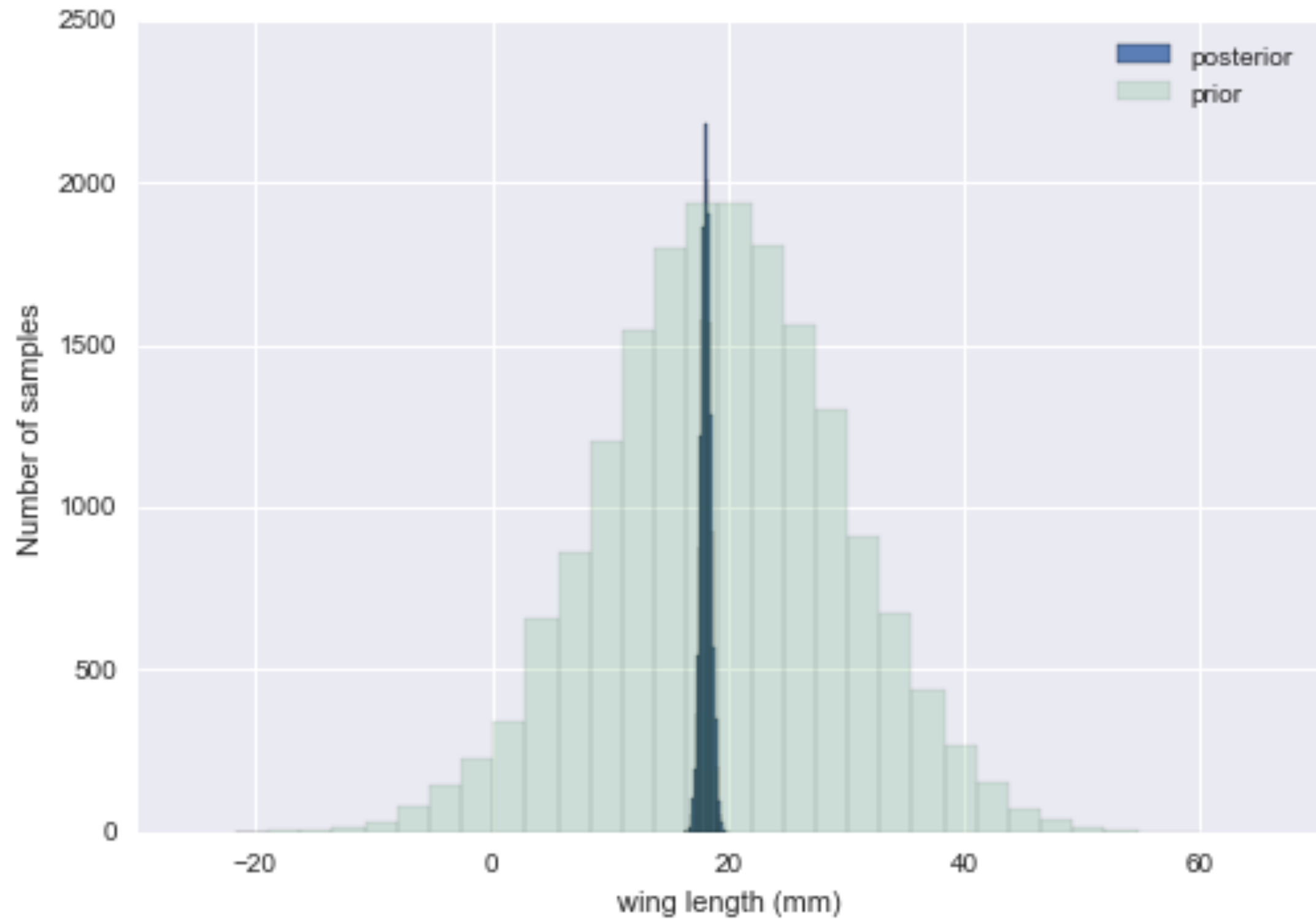
Method	Definition
Maximum Likelihood	$\hat{\theta} = \operatorname{argmax}_{\theta} p(D \theta)$
MAP estimation	$\hat{\theta} = \operatorname{argmax}_{\theta} p(D \theta)p(\theta \eta)$
ML-2 (Empirical Bayes)	$\hat{\eta} = \operatorname{argmax}_{\eta} \int d\theta p(D \theta)p(\theta \eta) = \operatorname{argmax}_{\eta} p(D \eta)$
MAP-2	$\hat{\eta} = \operatorname{argmax}_{\eta} \int d\theta p(D \theta)p(\theta \eta)p(\eta) = \operatorname{argmax}_{\eta} p(D \eta)p(\eta)$
Full Bayes	$p(\theta, \eta D) \propto p(D \theta)p(\theta \eta)p(\eta)$

# Normal-normal model

We have data on the wing length in millimeters of a nine members of a particular species of moth. We wish to make inferences from those measurements on the population mean  $\mu$ .

Other studies show the wing length to be around 19 mm. We also know that the length must be positive. We can choose a prior that is normal and most of the density is above zero ( $\mu=19.5, \tau=10$ ).

$Y = [16.4, 17.0, 17.2, 17.4, 18.2, 18.2, 18.2, 19.9, 20.8]$



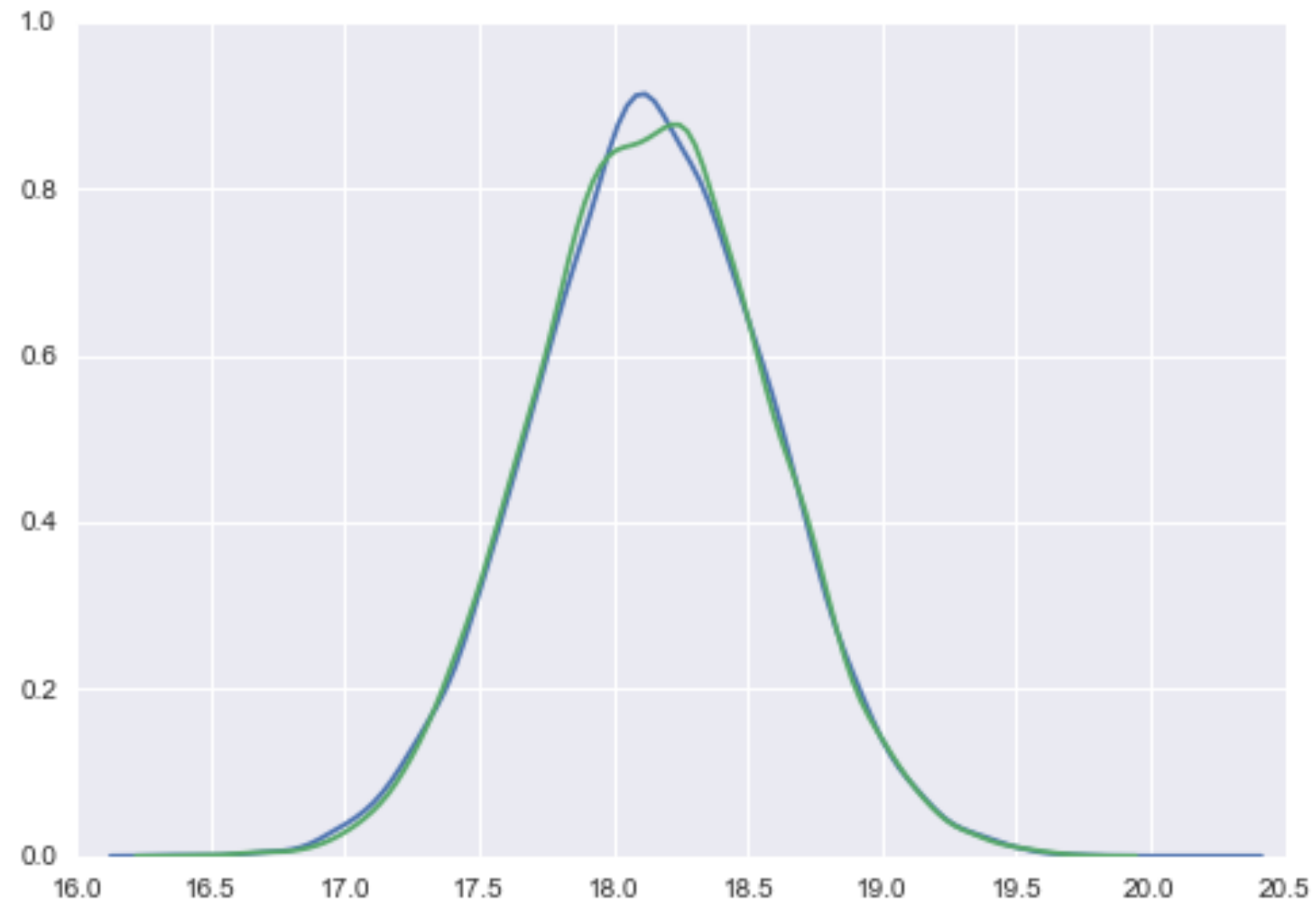
```

def metropolis(logp, qdraw, stepsize, nsamp, xinit):
    samples=np.empty(nsamp)
    x_prev = xinit
    accepted = 0
    for i in range(nsamp):
        x_star = qdraw(x_prev, stepsize)
        logp_star = logp(x_star)
        logp_prev = logp(x_prev)
        logpdfratio = logp_star -logp_prev
        u = np.random.uniform()
        if np.log(u) <= logpdfratio:
            samples[i] = x_star
            x_prev = x_star
            accepted += 1
        else:#we always get a sample
            samples[i]= x_prev

    return samples, accepted

logprior = lambda mu: norm.logpdf(mu, loc=19.5, scale=10)
loglike = lambda mu: np.sum(norm.logpdf(Y, loc=mu, scale=np.std(Y)))
logpost = lambda mu: loglike(mu) + logprior(mu)

```



# Sampling with pymc

```
conda install pymc3 patsy
```

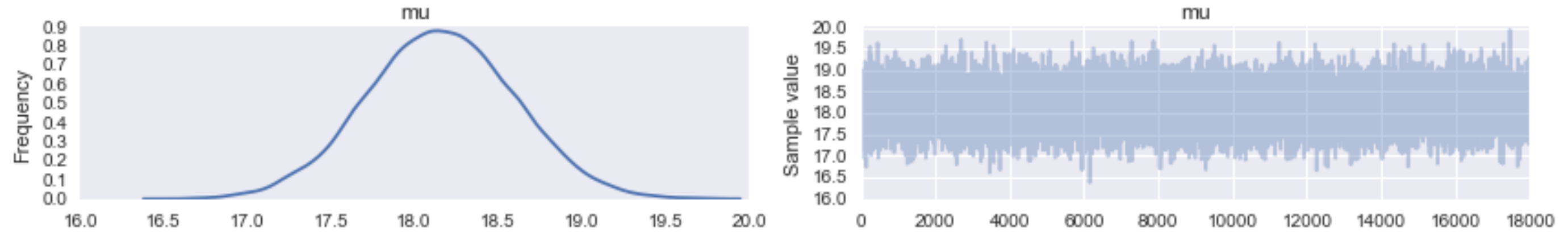
Installed 3.0rc4 for me.

```
import pymc3 as pm
with pm.Model() as model1:
    mu = pm.Normal('mu', mu=19.5, sd=10)#parameter's prior
    wingspan = pm.Normal('wingspan', mu=mu, sd=np.std(Y), observed=Y)#likelihood
    stepper=pm.Metropolis()
    tracemodel1=pm.sample(100000, step=stepper)
```

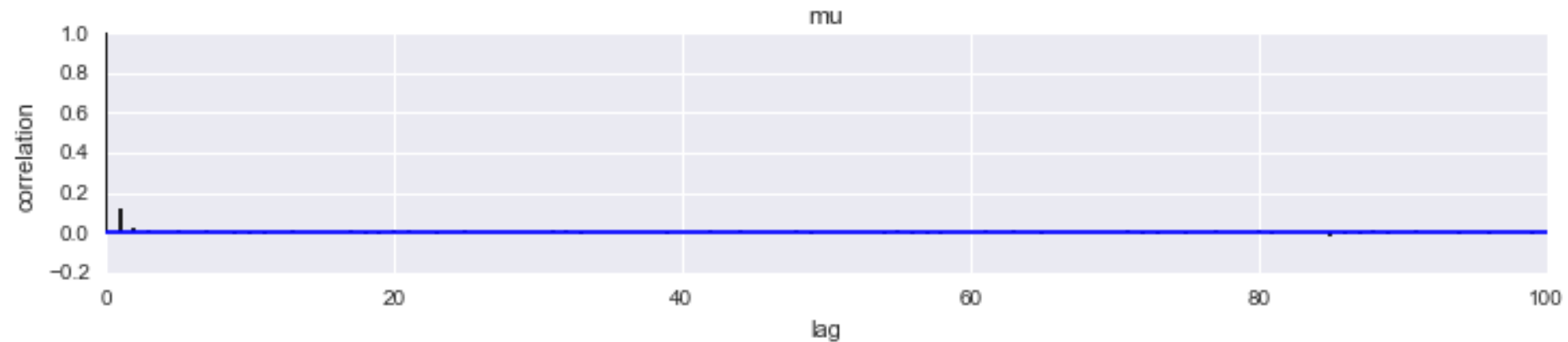
```
100%|██████████| 100000/100000 [00:10<00:00, 9878.33it/s]| 528/100000 [00:00<00:18, 5279.00it/s]
```



```
pm.traceplot(tracemodel1[10000::5]);
```



```
pm.autocorrplot(tracemodel1[10000::5]);
```

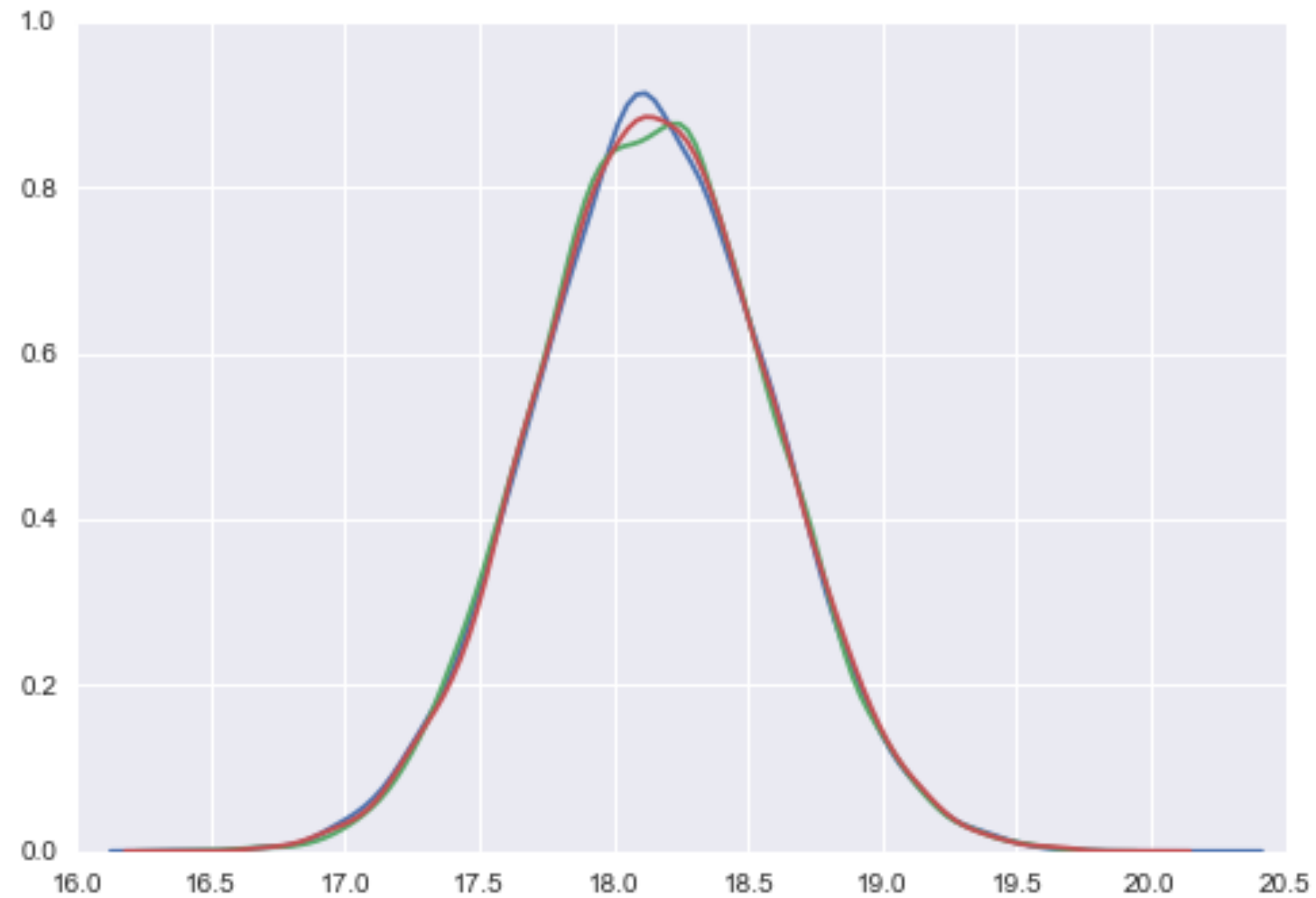


# HPD: The highest-posterior-density smallest width Bayesian Credible Interval.

```
pm.summary(tracemodel1)
mu:
```

Mean	SD	MC Error	95% HPD interval	
-----				
18.148	0.443	0.003	[17.285, 19.019]	
Posterior quantiles:				
2.5	25	50	75	97.5
-----	=====	=====	-----	
17.277	17.849	18.146	18.446	19.012

# Posteriors all match up!



# Posterior predictives

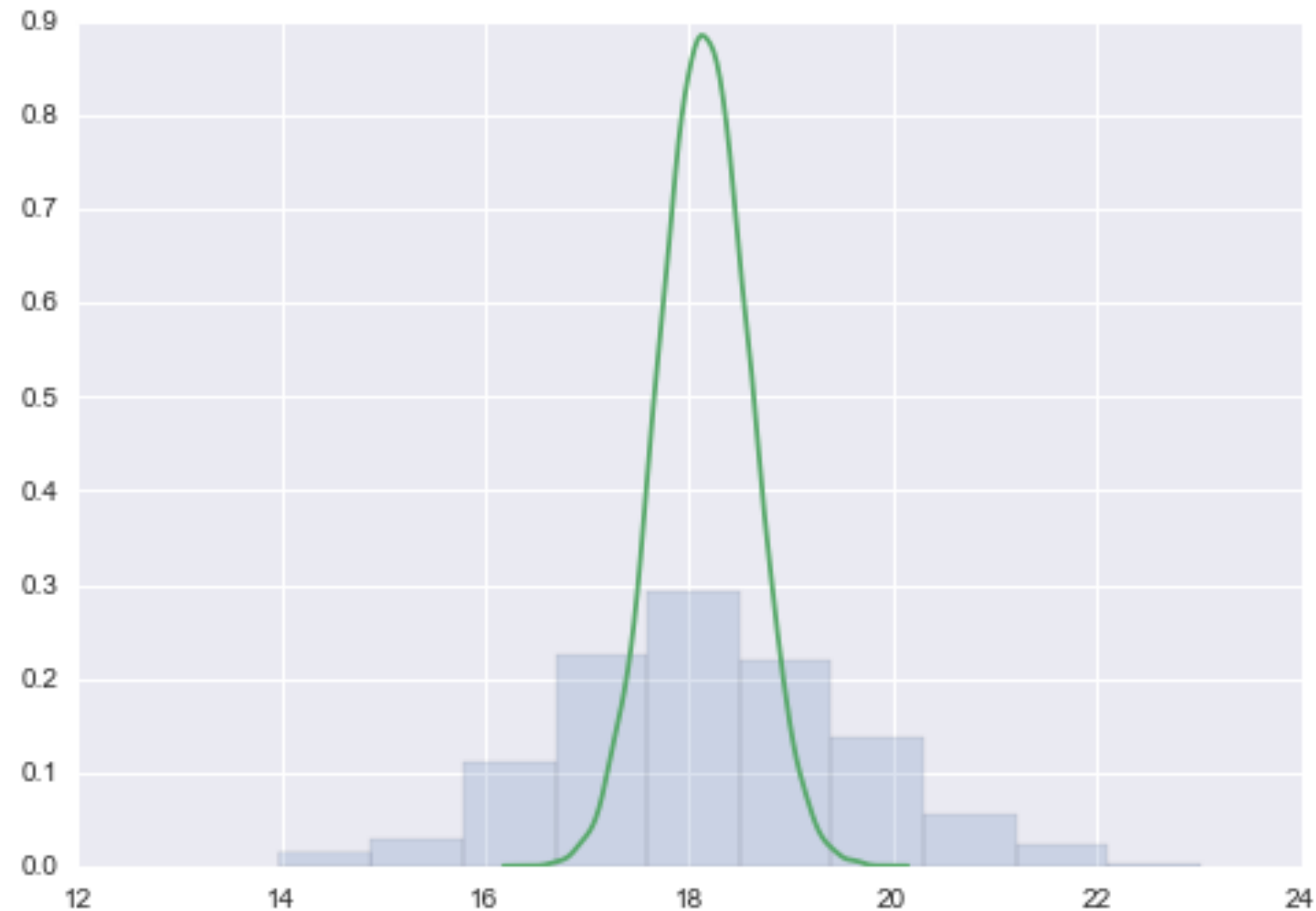
The posterior predictive is accessed via the `sample_ppc` function:

```
model1.observed_RVs
```

```
[wingspan]
```

```
tr1 = tracemodel1[10000::5]  
postpred = pm.sample_ppc(tr1, 1000, model1)
```

100%|██████████| 1000/1000 [00:01<00:00, 510.20it/s] | 25/1000 [00:00<00:03, 244.20it/s]



# Bayesian Formulation of Regression

Data

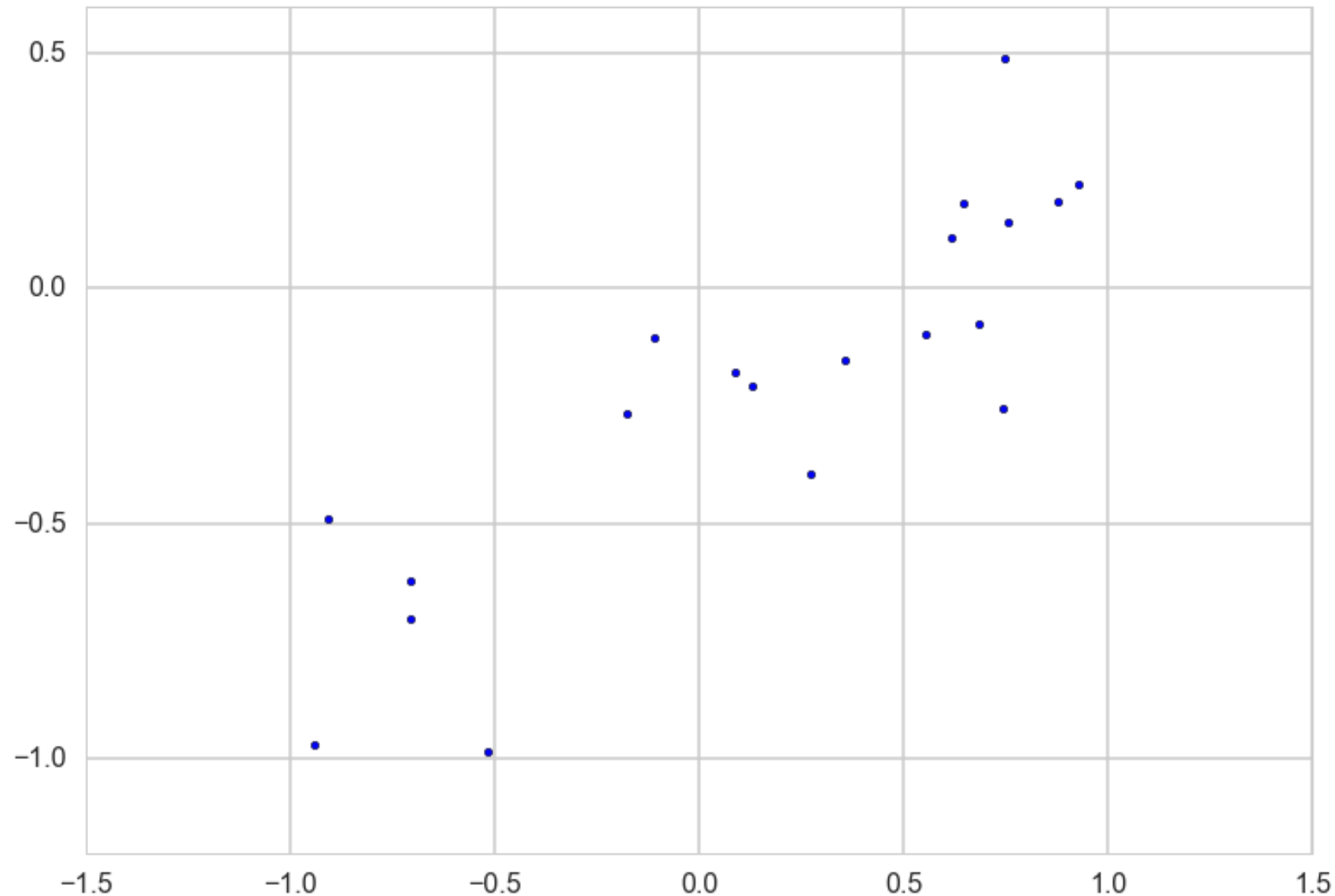
$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$$

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

Model:

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

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



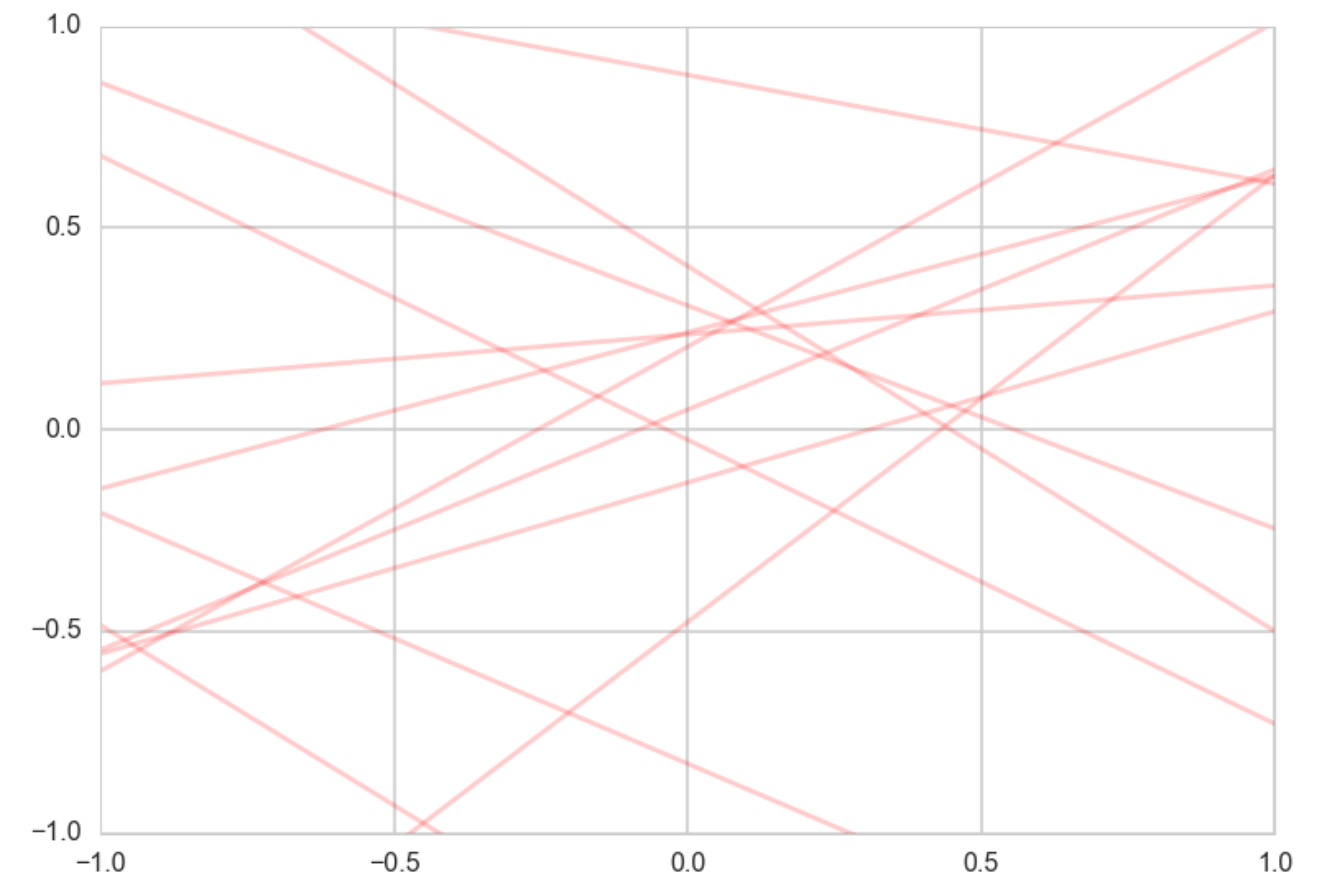
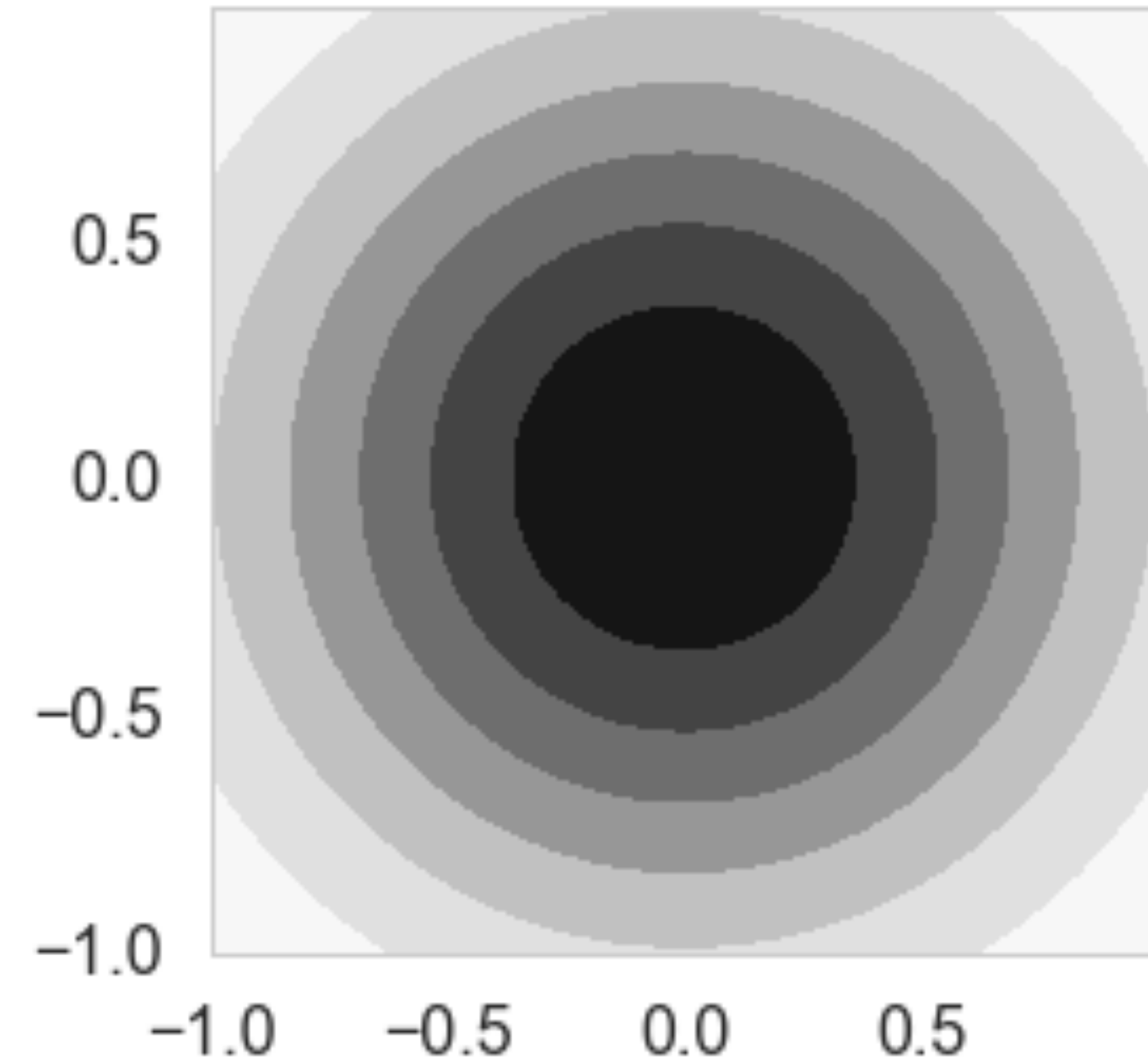
# Likelihood

The likelihood is, because we assume independency, the product

$$\begin{aligned}\mathcal{L} = p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^n p(y_i|\mathbf{X}_i, \mathbf{w}) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \mathbf{X}_i^T \mathbf{w})^2}{2\sigma_n^2}\right) \\ &\propto \exp\left(-\frac{|\mathbf{y} - \mathbf{X}^T \mathbf{w}|^2}{2\sigma_n^2}\right) \propto N(\mathbf{X}^T \mathbf{w}, \sigma_n^2 \mathbf{I})\end{aligned}$$

Prior  $\mathbf{w} \sim \mathbf{N}(\mathbf{w}_0, \Sigma)$

$$\mathbf{w} \sim \mathbf{N}(\mathbf{w}_0, \tau^2 \mathbf{I})$$



## Posterior

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

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

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

where the new mean is  $\bar{\mathbf{w}} = A^{-1} \Sigma^{-1} \mathbf{w}_0 + \sigma_n^{-2} (A^{-1} \mathbf{X}^T \mathbf{y})$



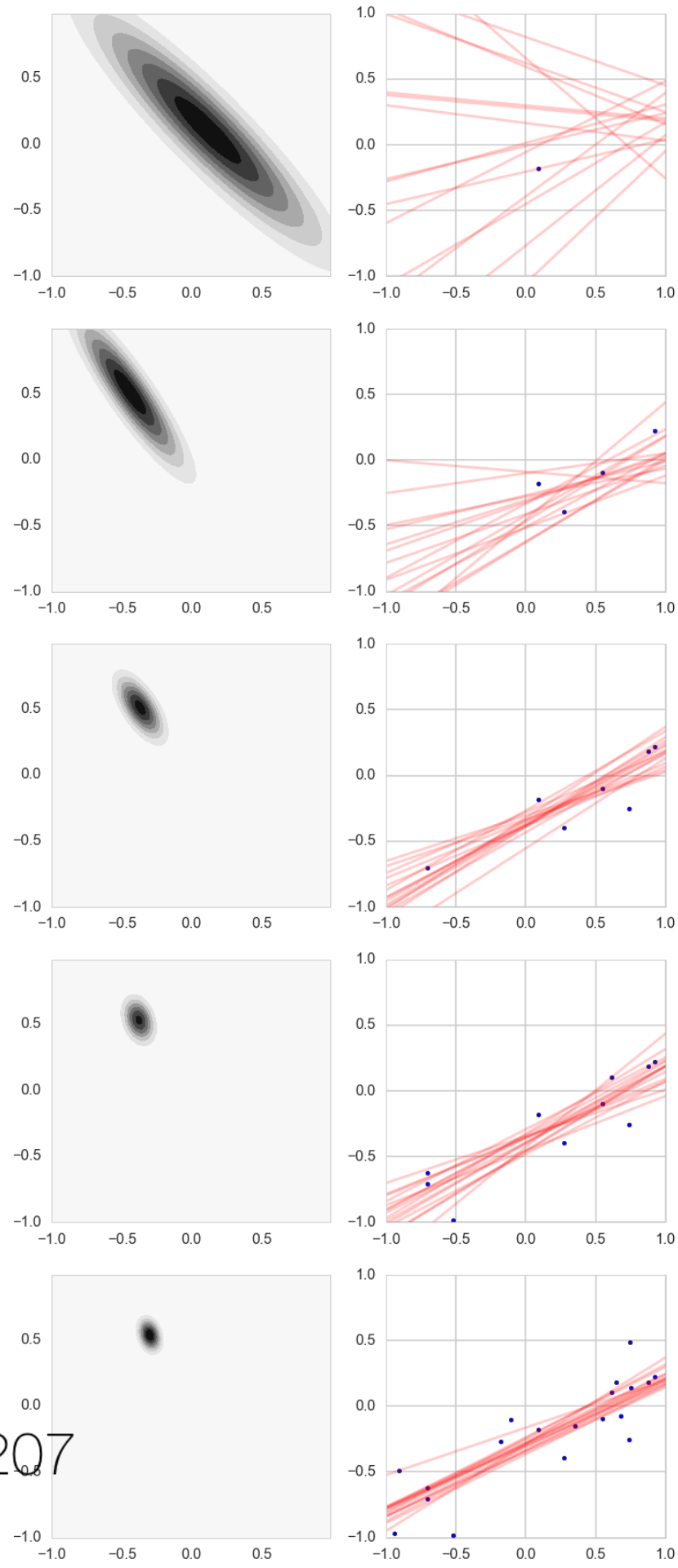
# Bayesian updating

```
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
```

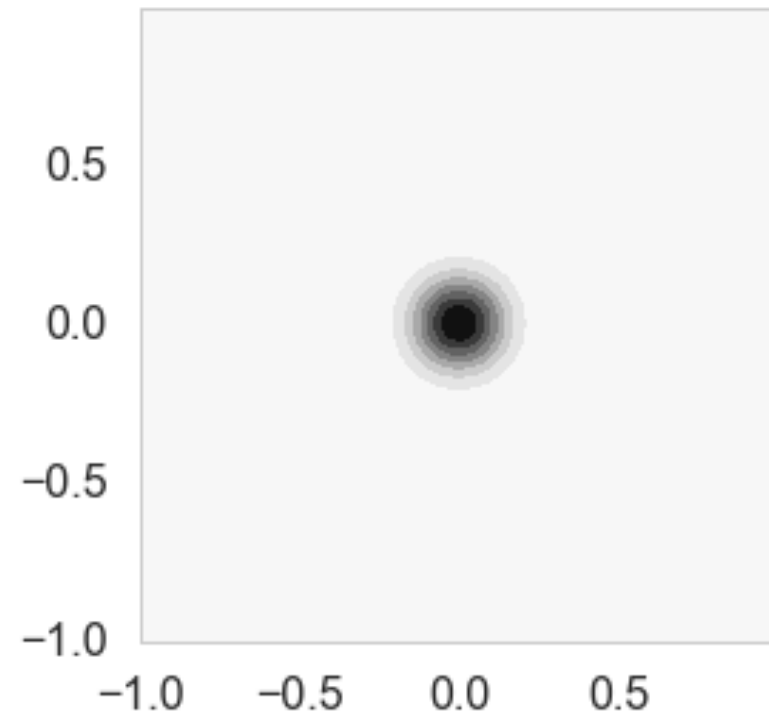
## Posterior predictive

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

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



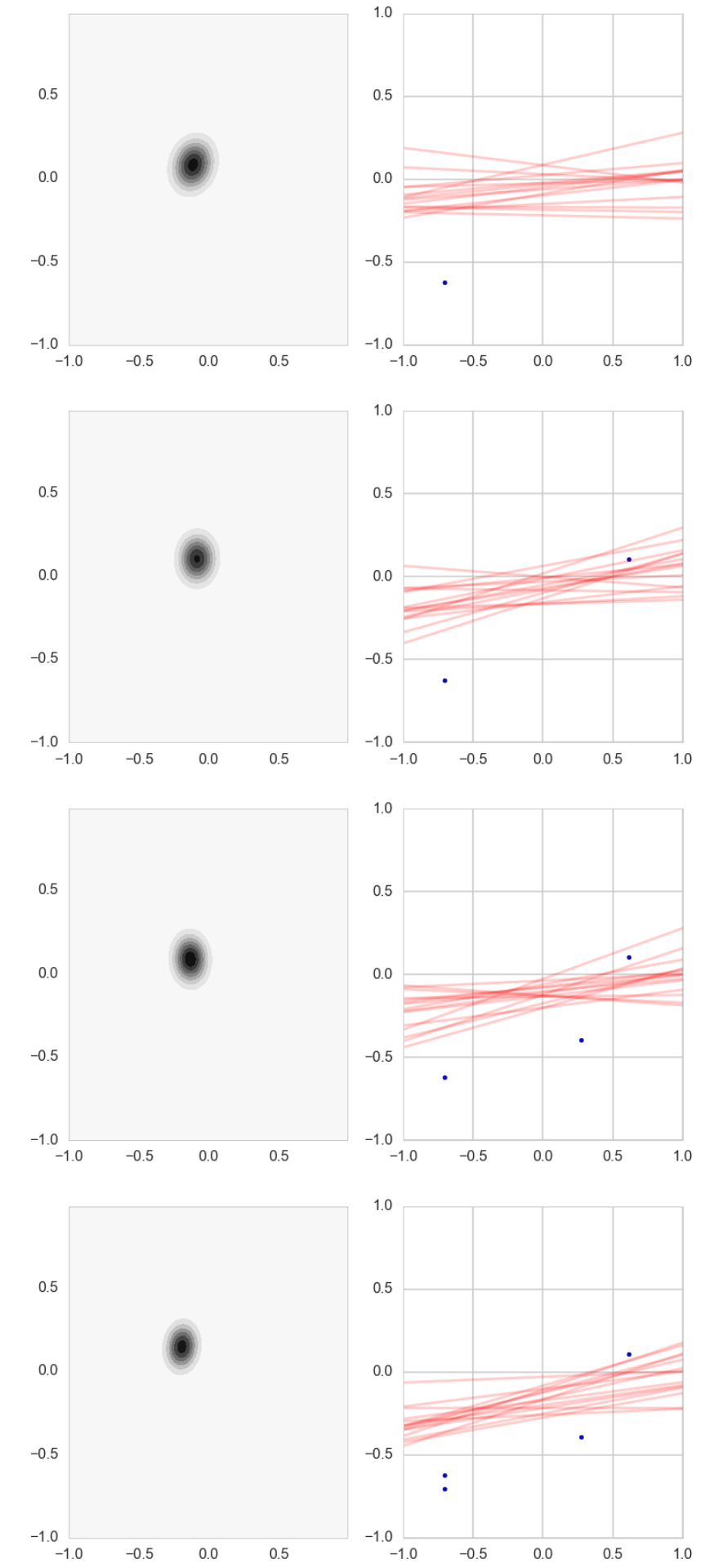
# Regularization



`priorPrecision/likelihoodPrecision`

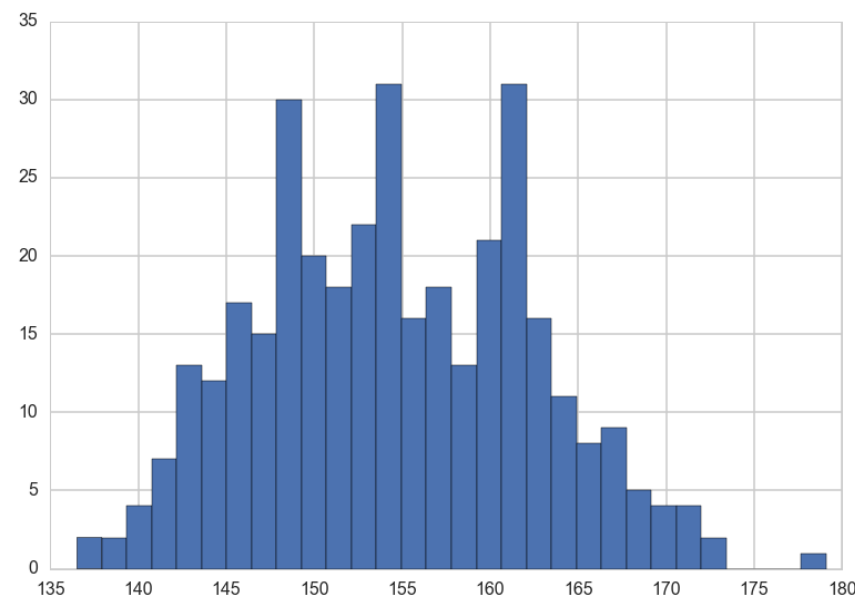
4.0

This ratio is the ridge  $\alpha$ .



# 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

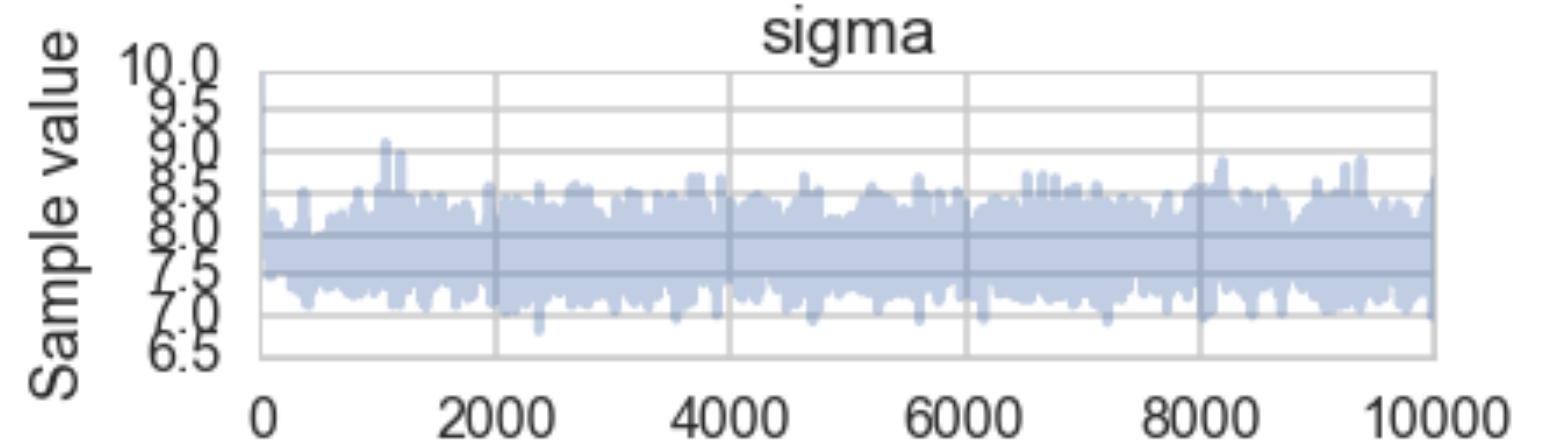
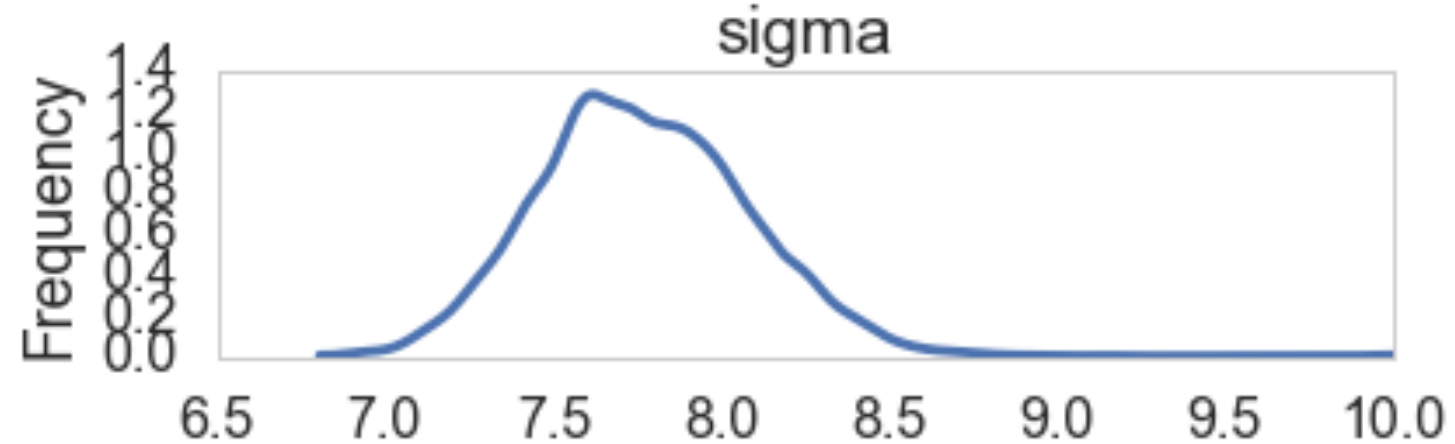
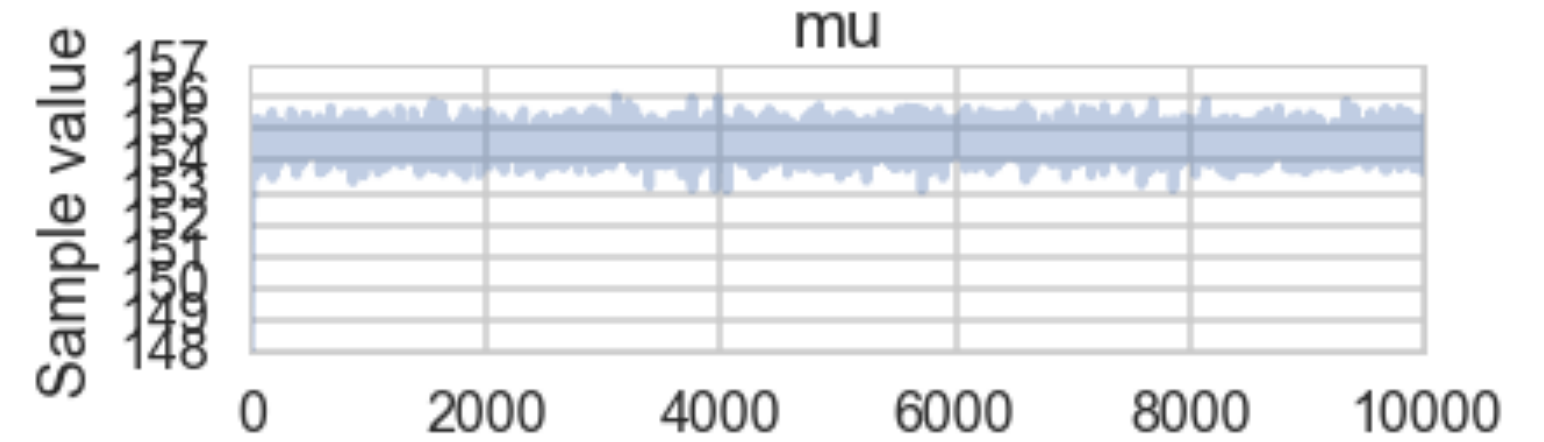
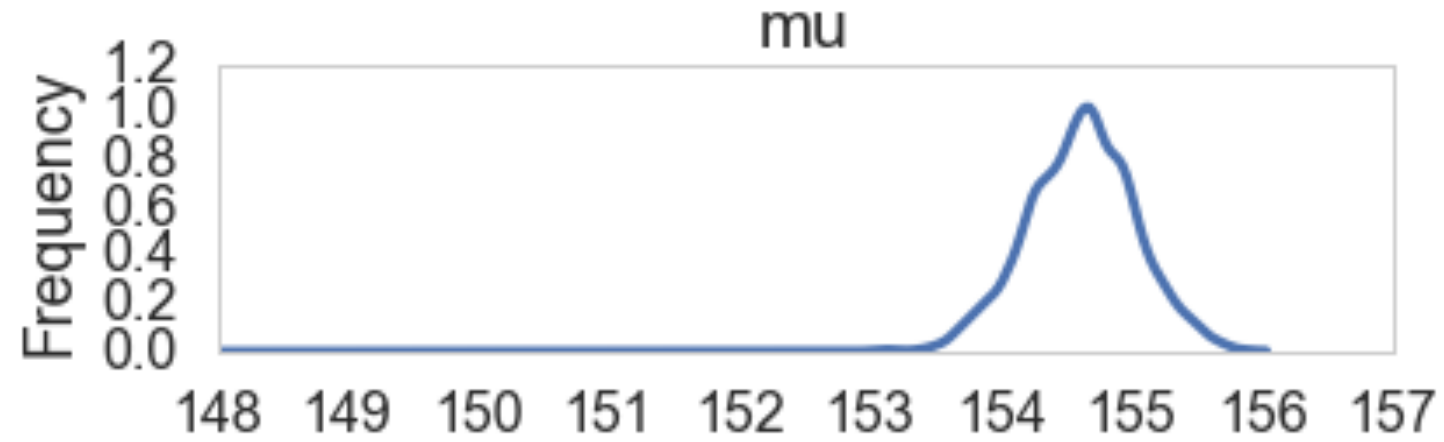
# Model

$$\begin{aligned}h &\sim N(\mu, \sigma) \\ \mu &\sim Normal(148, 20) \\ \sigma &\sim Unif(0, 50)\end{aligned}$$

```
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]
```



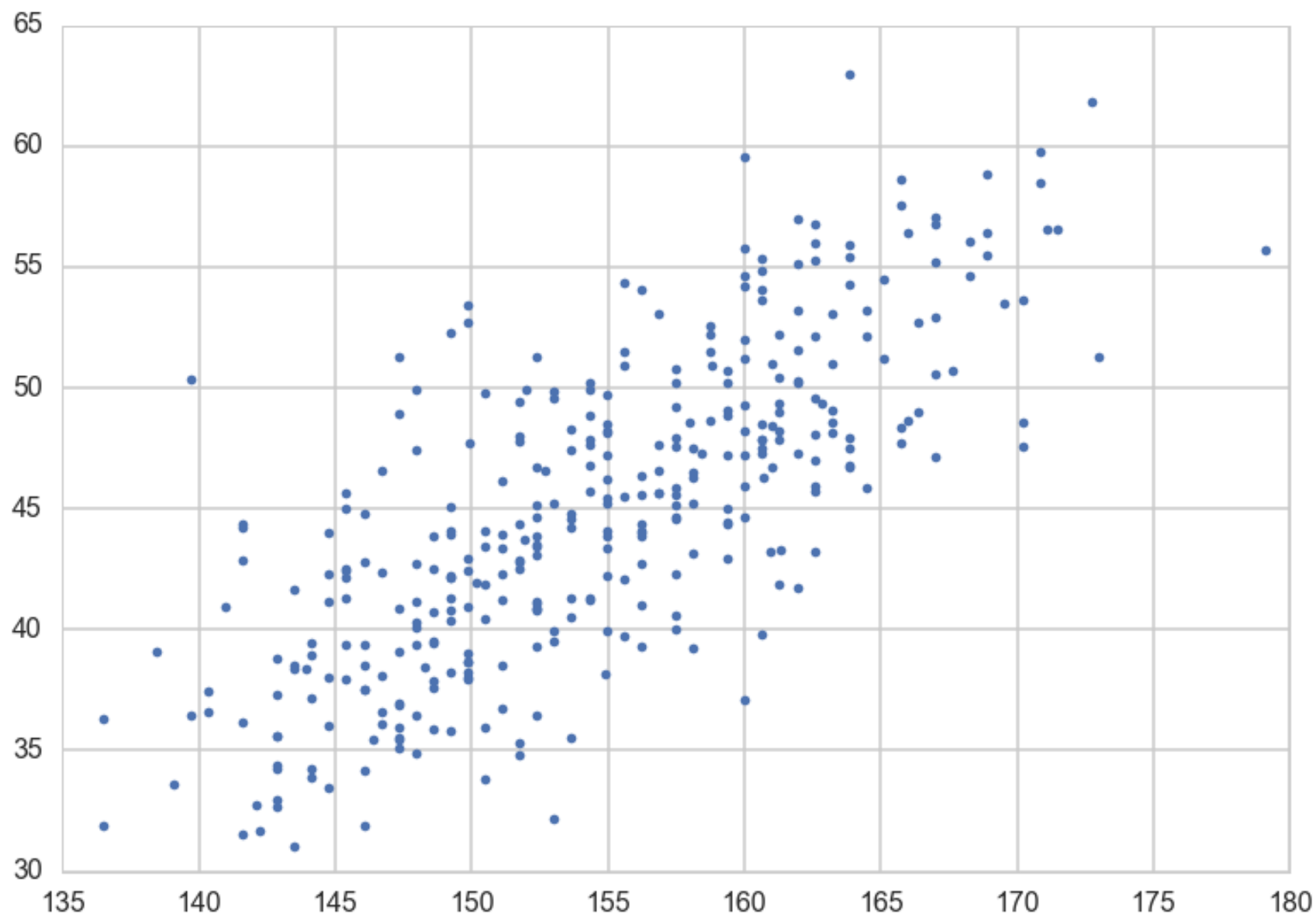
```
def acceptance(trace, paramname):
    accept = np.sum(trace[paramname][1:] != trace[paramname][:-1])
    return accept/trace[paramname].shape[0]
```

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

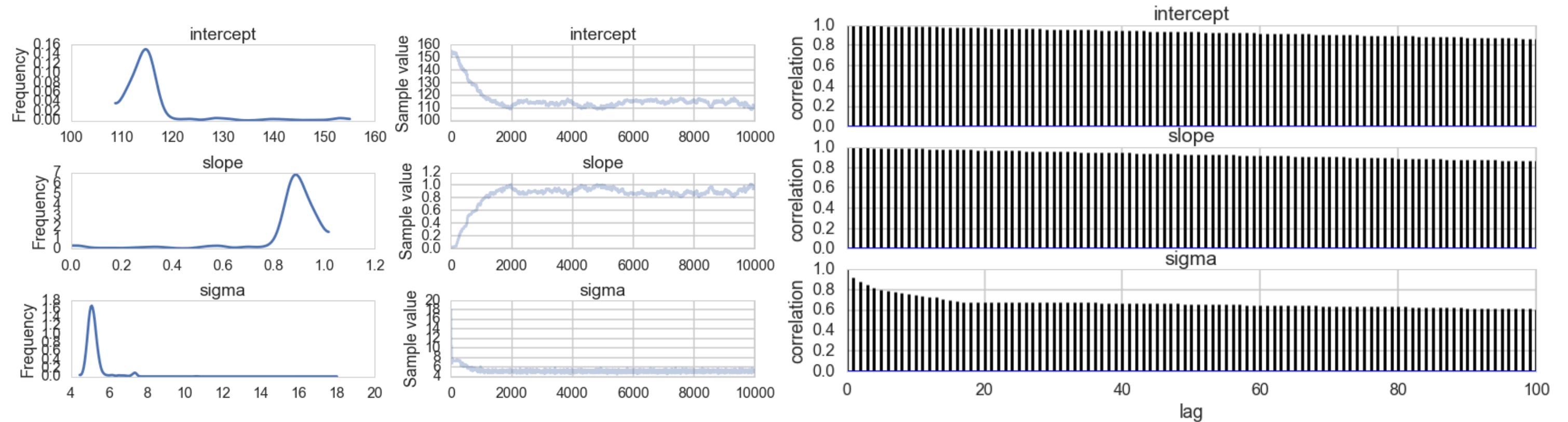
# Regression, adding a predictor, weight

$$\begin{aligned}h &\sim N(\mu, \sigma) \\ \mu &= \text{intercept} + \text{slope} \times \text{weight} \\ \text{intercept} &\sim N(150, 100) \\ \text{slope} &\sim N(0, 10) \\ \sigma &\sim \text{Unif}(0, 50)\end{aligned}$$

```
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



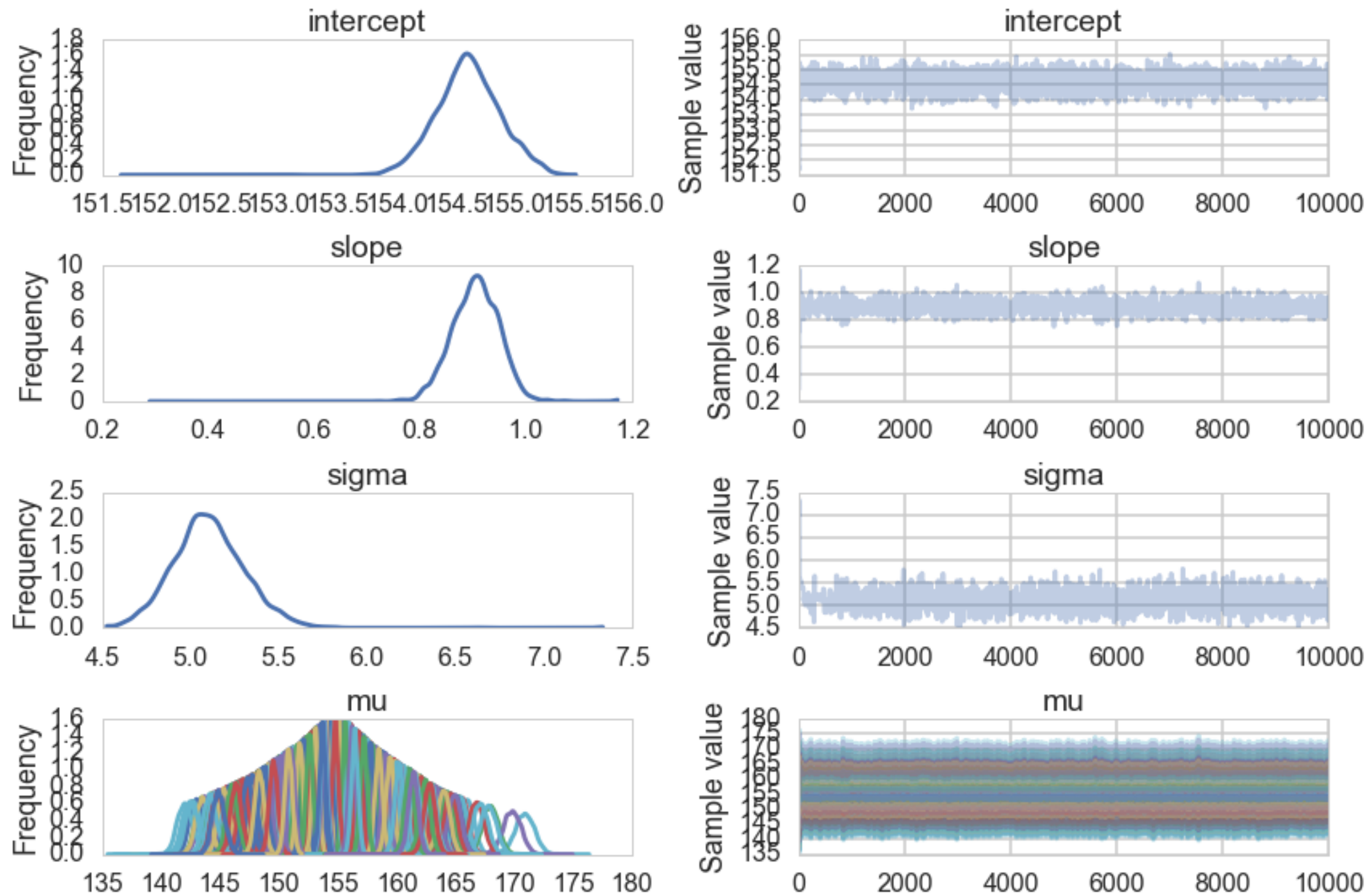
The slope and intercept are very highly correlated: -0.99!

# Sympton 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)
    # below is a deterministic
    #mu = intercept + slope * (df2.weight - df2.weight.mean())
    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)
```

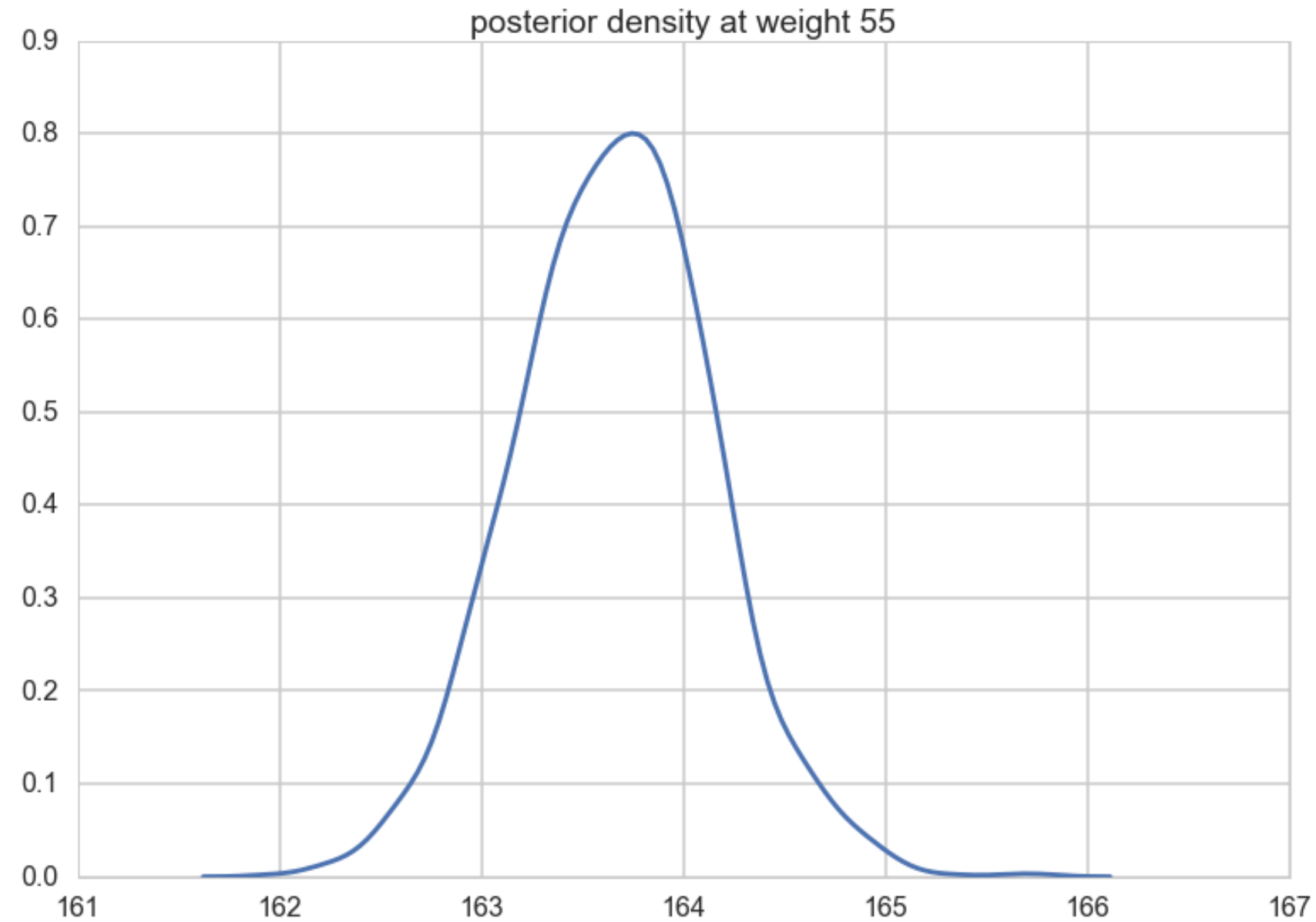




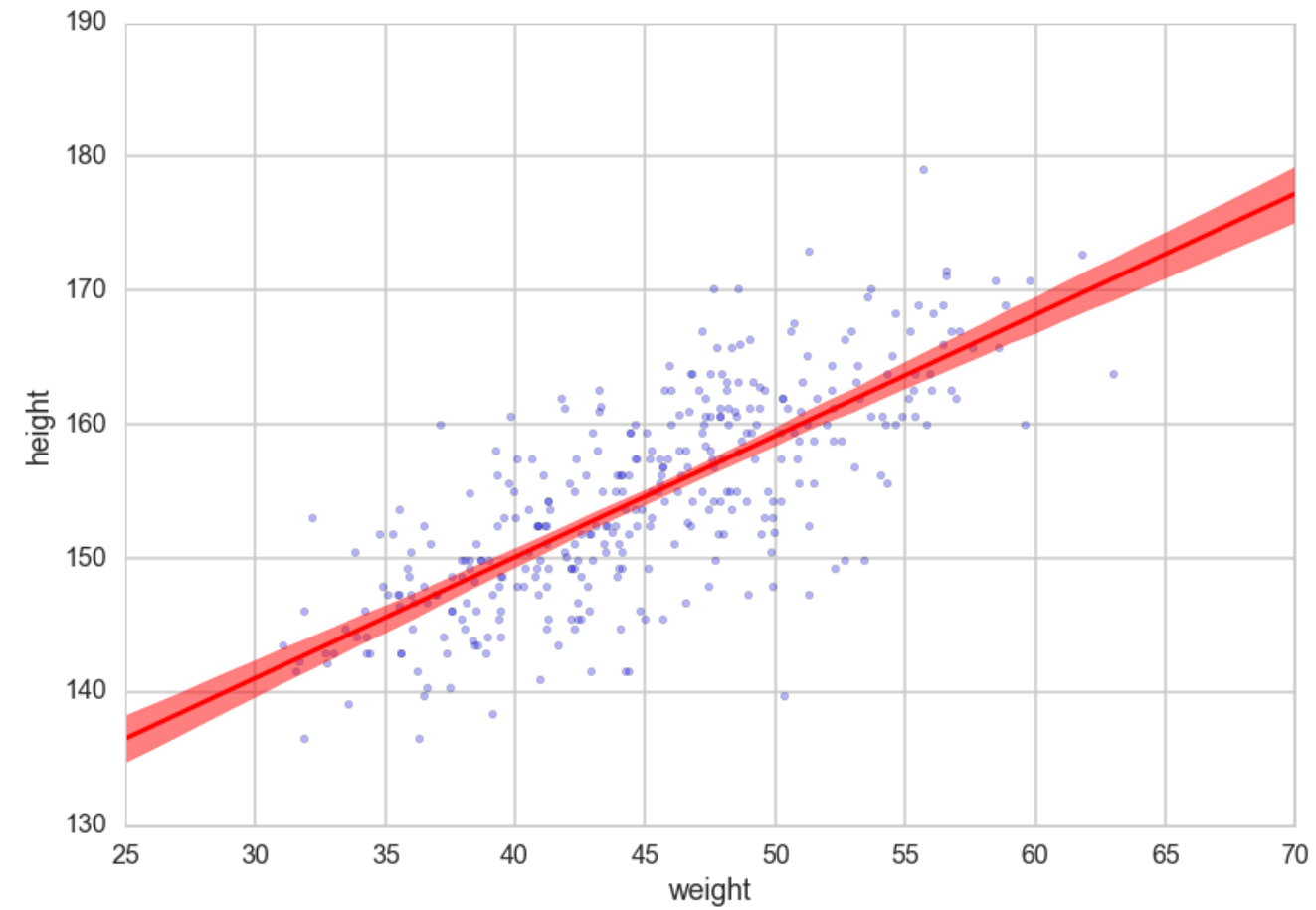
# Posteriors

```
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

