Quantifying Uncertainty

Bayesian Data Analysis in Python

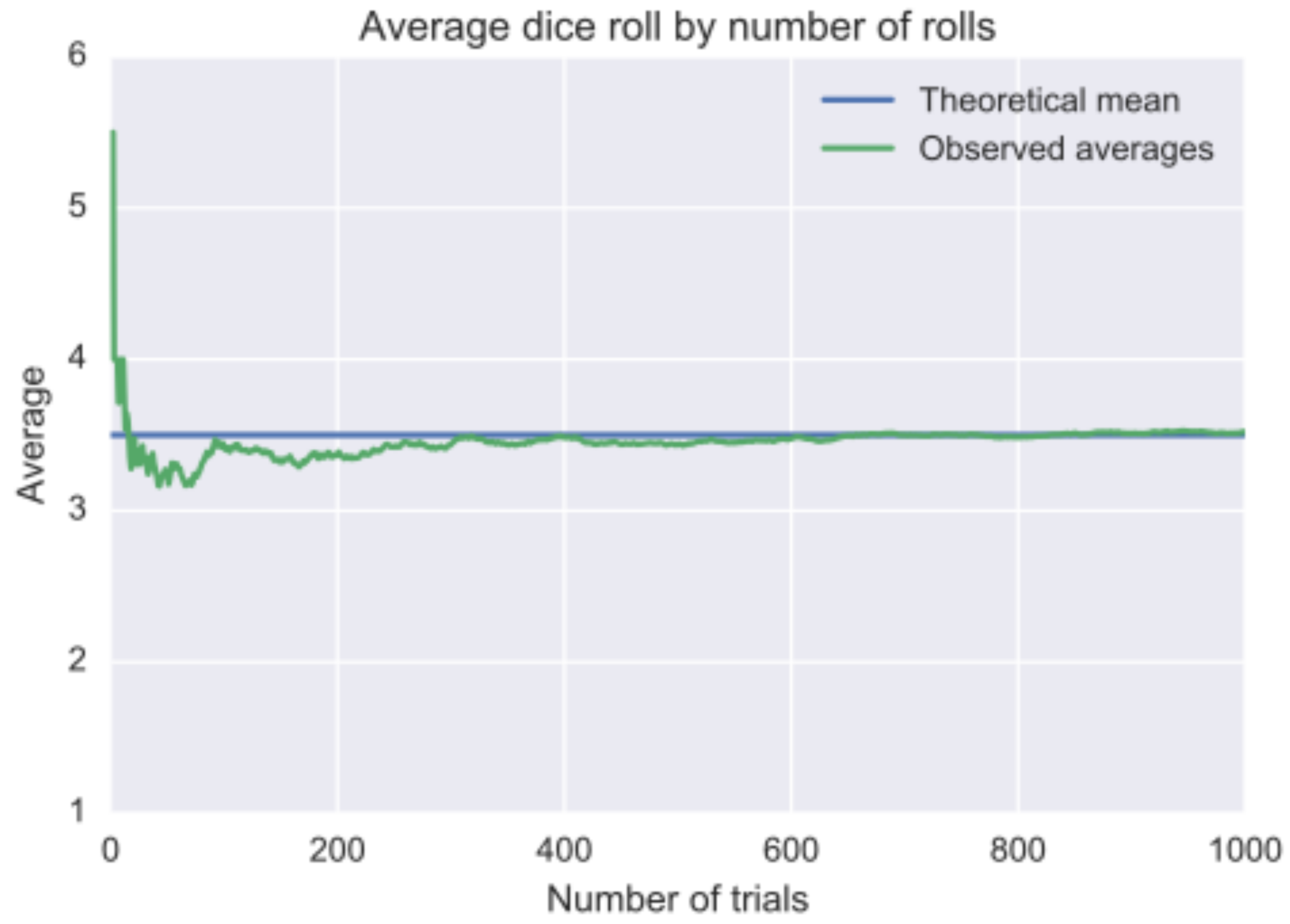
Mat Leonard

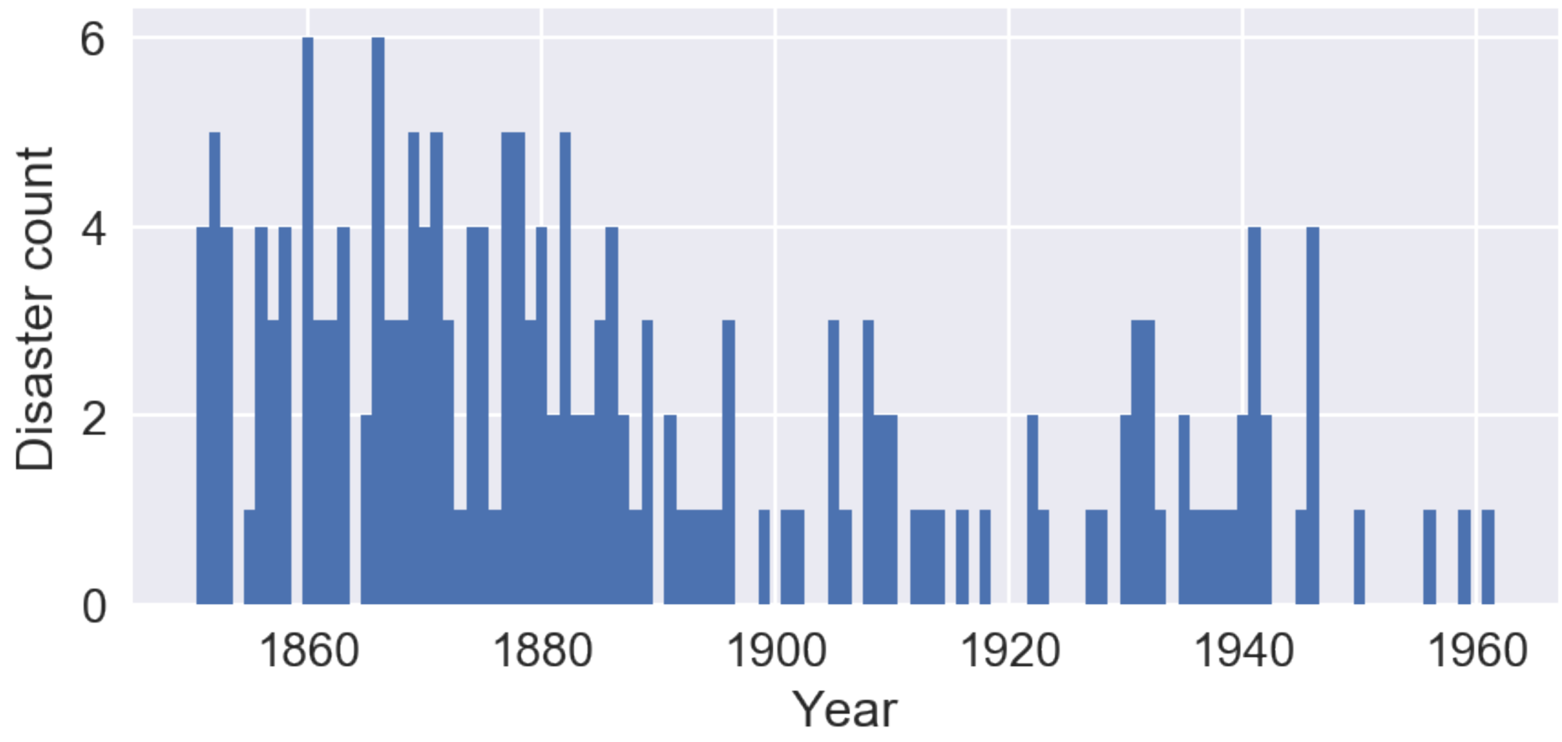
Head of Learning, AI

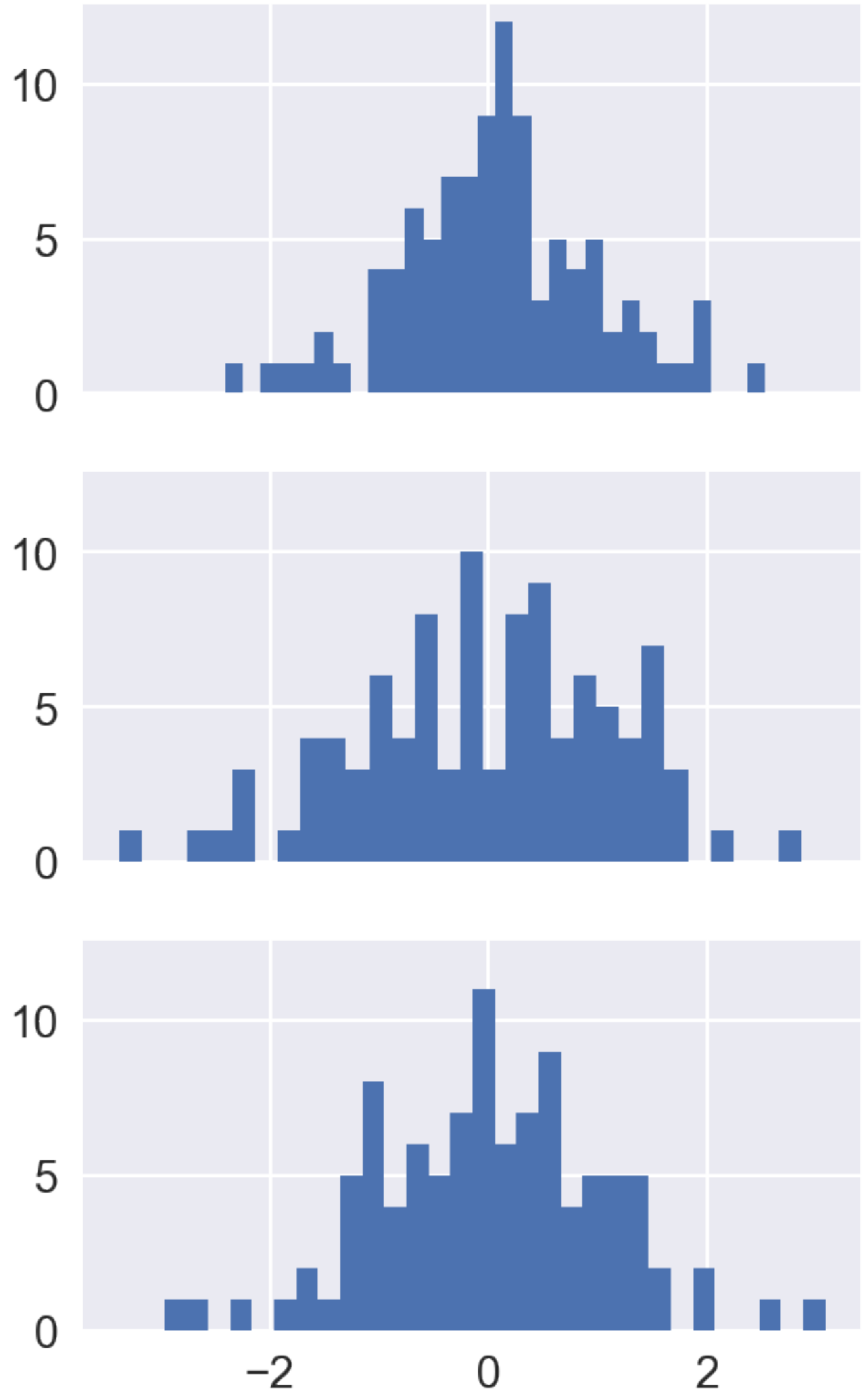
Udacity

@MatDrinksTea

Data is Life, Life is Uncertain





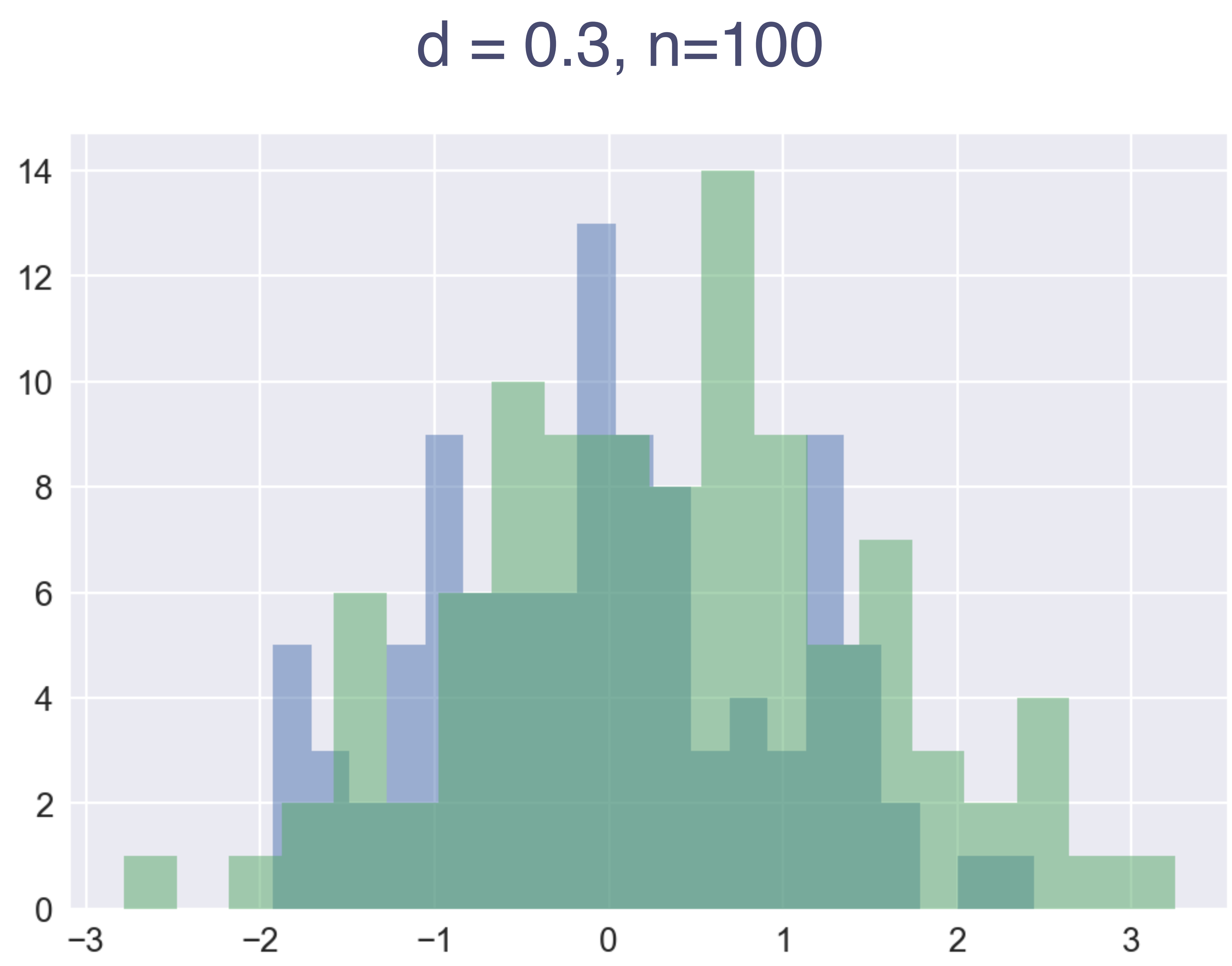


Quantify Uncertainty

with

STATISTICS

Hypothesis Testing Refresher

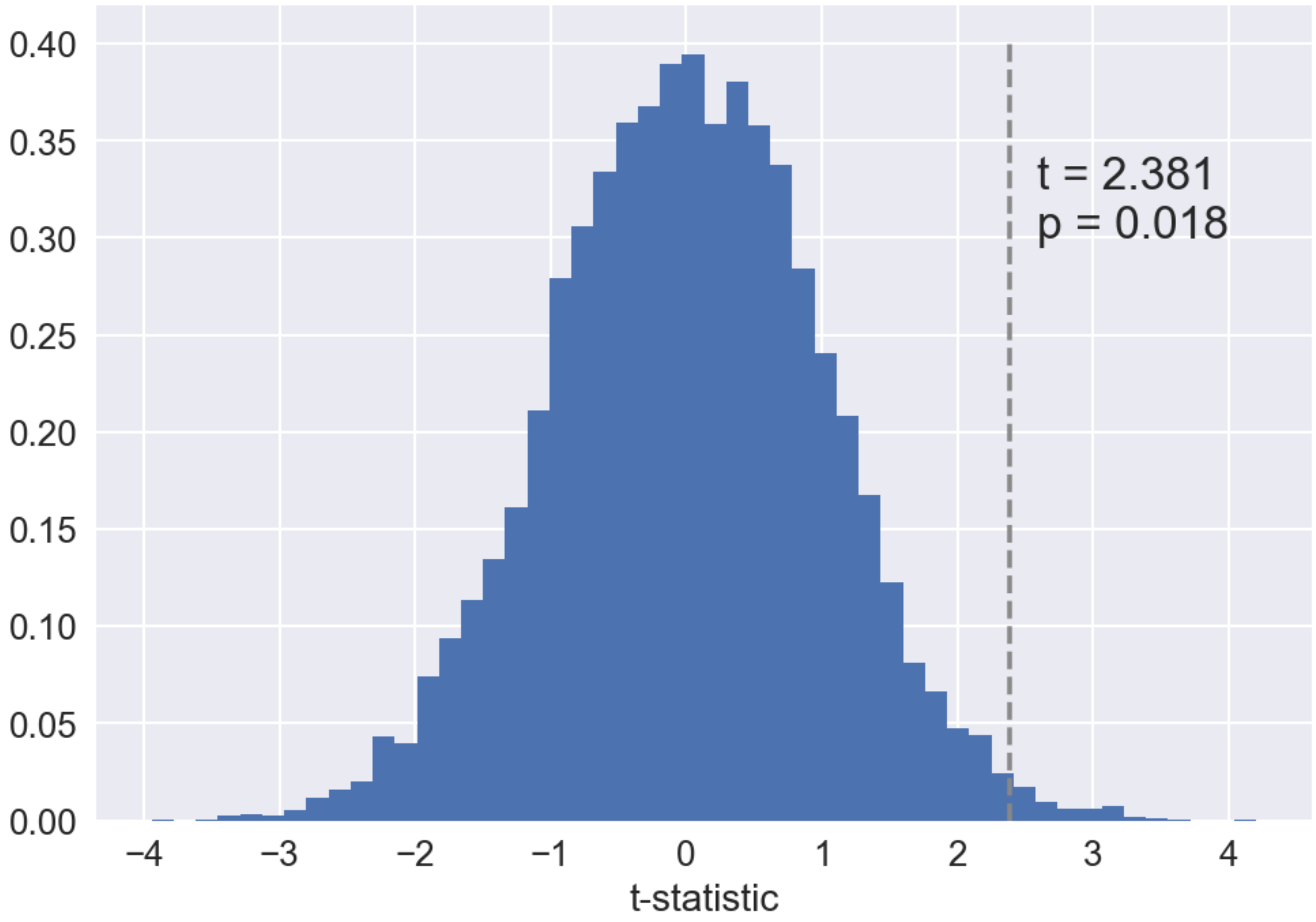


$$t = \sqrt{n}\,\frac{\bar{X}_1 - \bar{X}_2}{\sqrt{s_{X_1}^2 + s_{X_2}^2}}$$

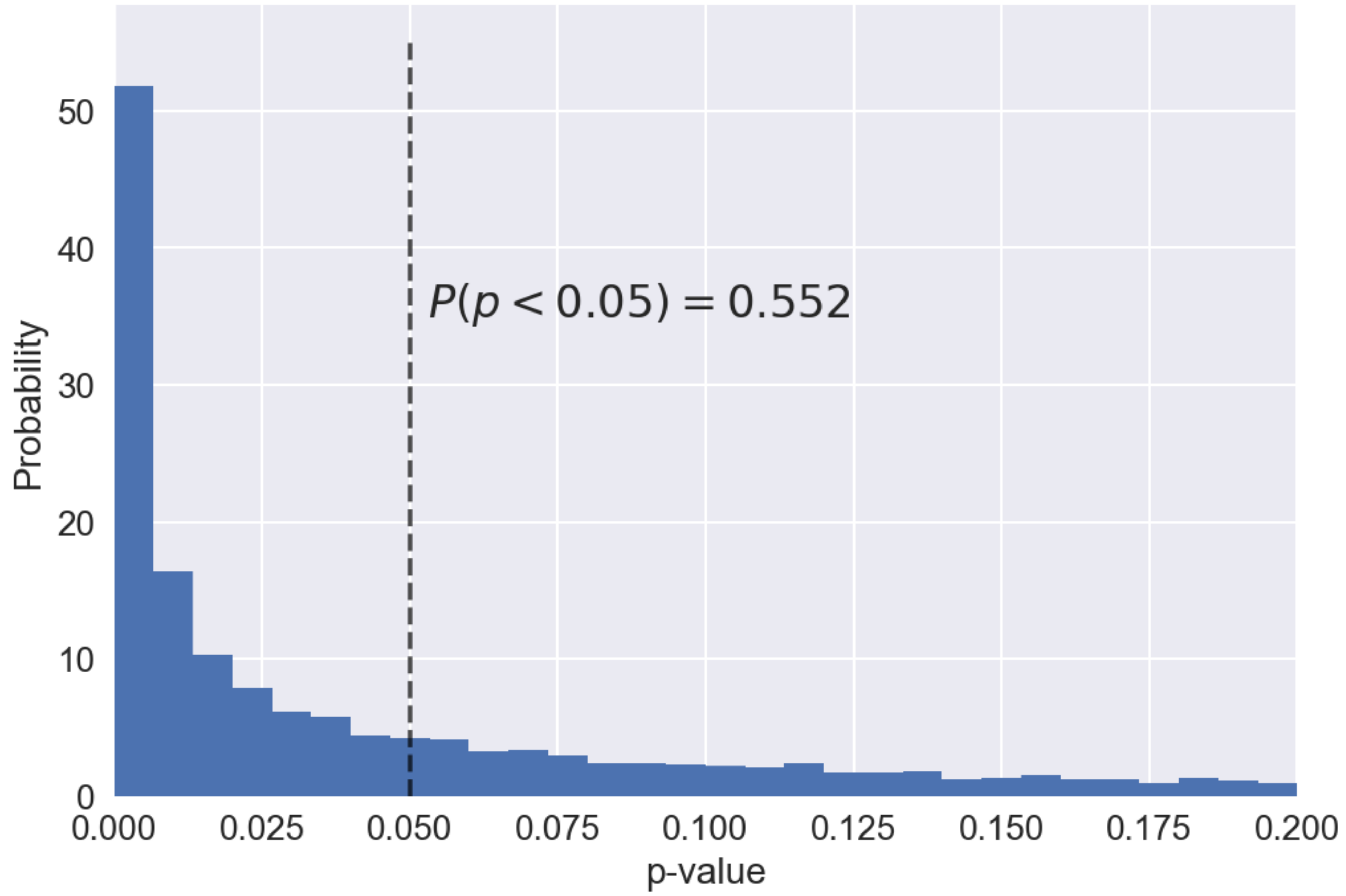
$$H_0 : t = 0$$

$$H_1 : t \neq 0$$

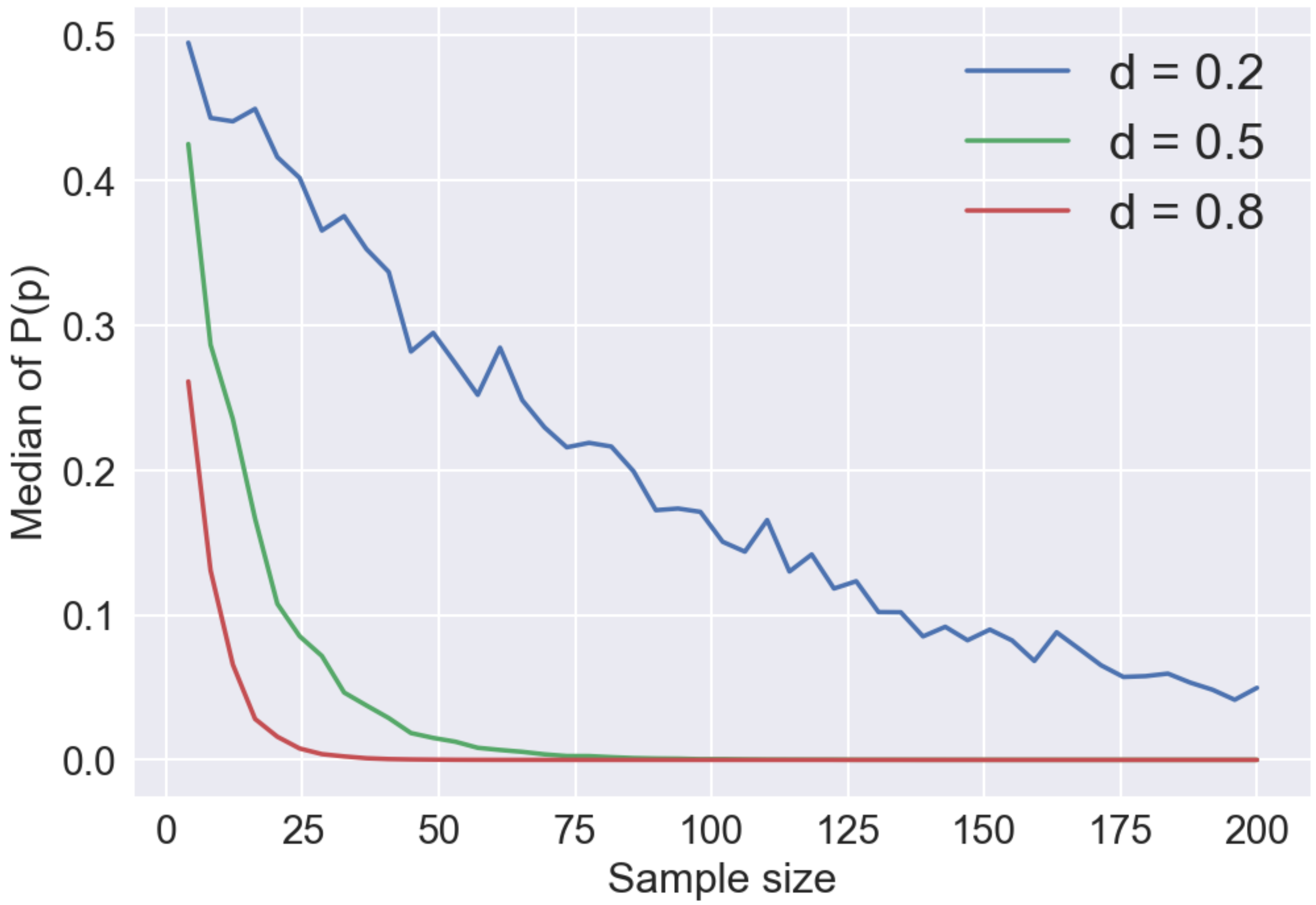
$$H_0 : t = 0$$



d = 0.3, n=100



Median of the p sampling distribution: function of the effect size (d) and the sample size



p-values are *not* a measure of uncertainty

Parameter estimation with confidence intervals?

What people think (and want)

95% probability the parameter is in the 95% confidence interval

What it actually is

If you perform your experiment an infinite number of times, calculating the 95% confidence interval for each experiment, then for 95% of those experiments the true parameter will fall within the 95% confidence interval

Frequentist CI theory says nothing at all about the probability that a particular, observed confidence interval contains the true value; it is either 0 (if the interval does not contain the parameter) or 1 (if the interval does contain the true value).

The fallacy of placing confidence in confidence intervals
Morey, R.D., Hoekstra, R., Rouder, J.N. et al. Psychon Bull Rev
(2016) 23: 103

Bayesian Inference

Bayes' Theorem

$$P(\theta \mid y) \propto \prod_{i}^{N} P(y_i \mid \theta) P(\theta)$$

Bayes' Theorem

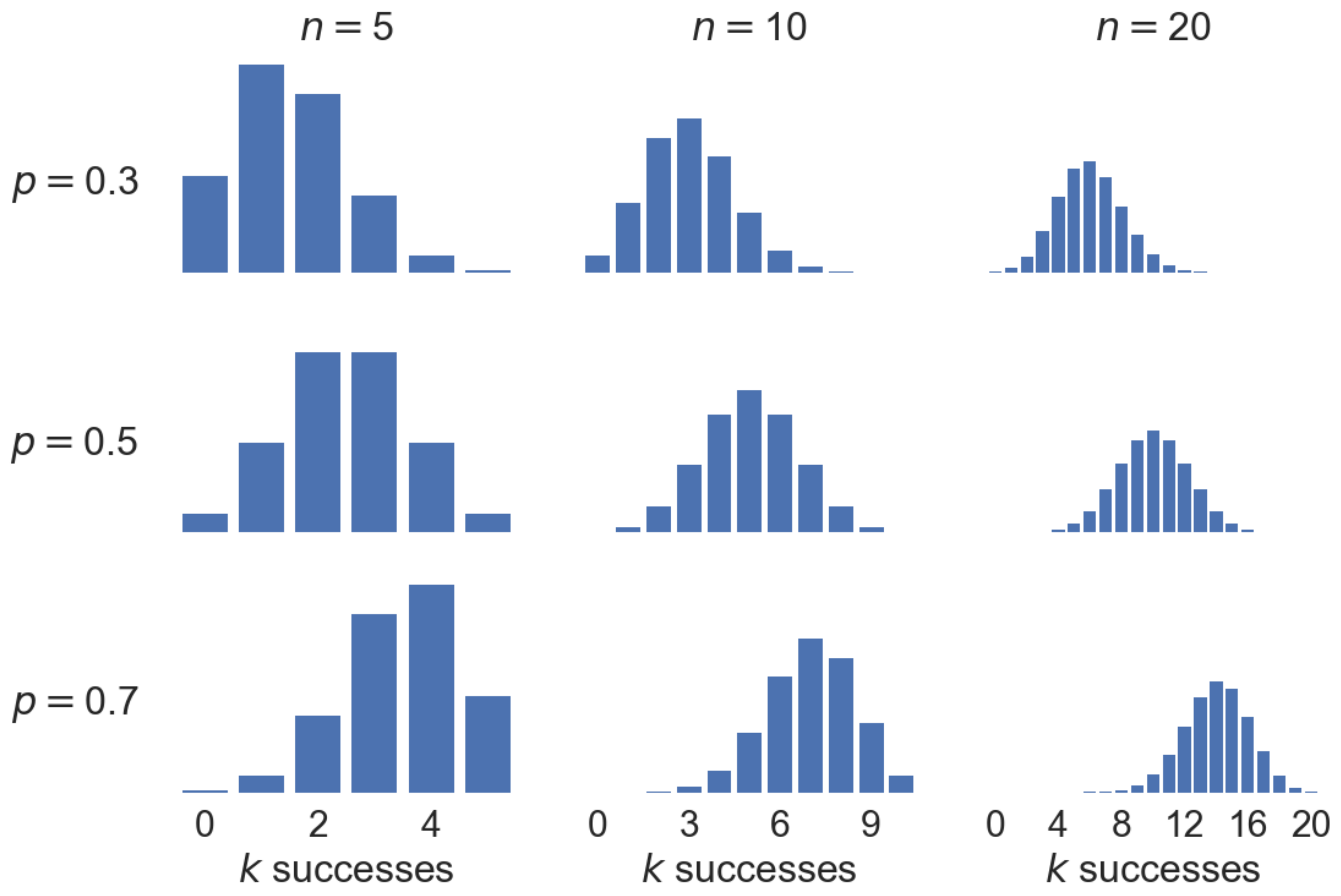
$$\overbrace{P(\theta \mid y)}^{\text{Posterior}} \propto \prod_{i}^{N} \underbrace{P(y_i \mid \theta)}_{\text{Data Likelihood}} \overbrace{P(\theta)}^{\text{Prior}}$$

Flipping Coins, a Classic

Binomial distribution

Attempts — Probability of success — Successes

$$\Pr(n, k, p) = \binom{n}{k} p^k (1-p)^{n-k}$$



Is our coin fair???

Bayesian Model

Likelihood

$$\mathrm{Binomial}\left(k,n \mid p\right)$$

Prior

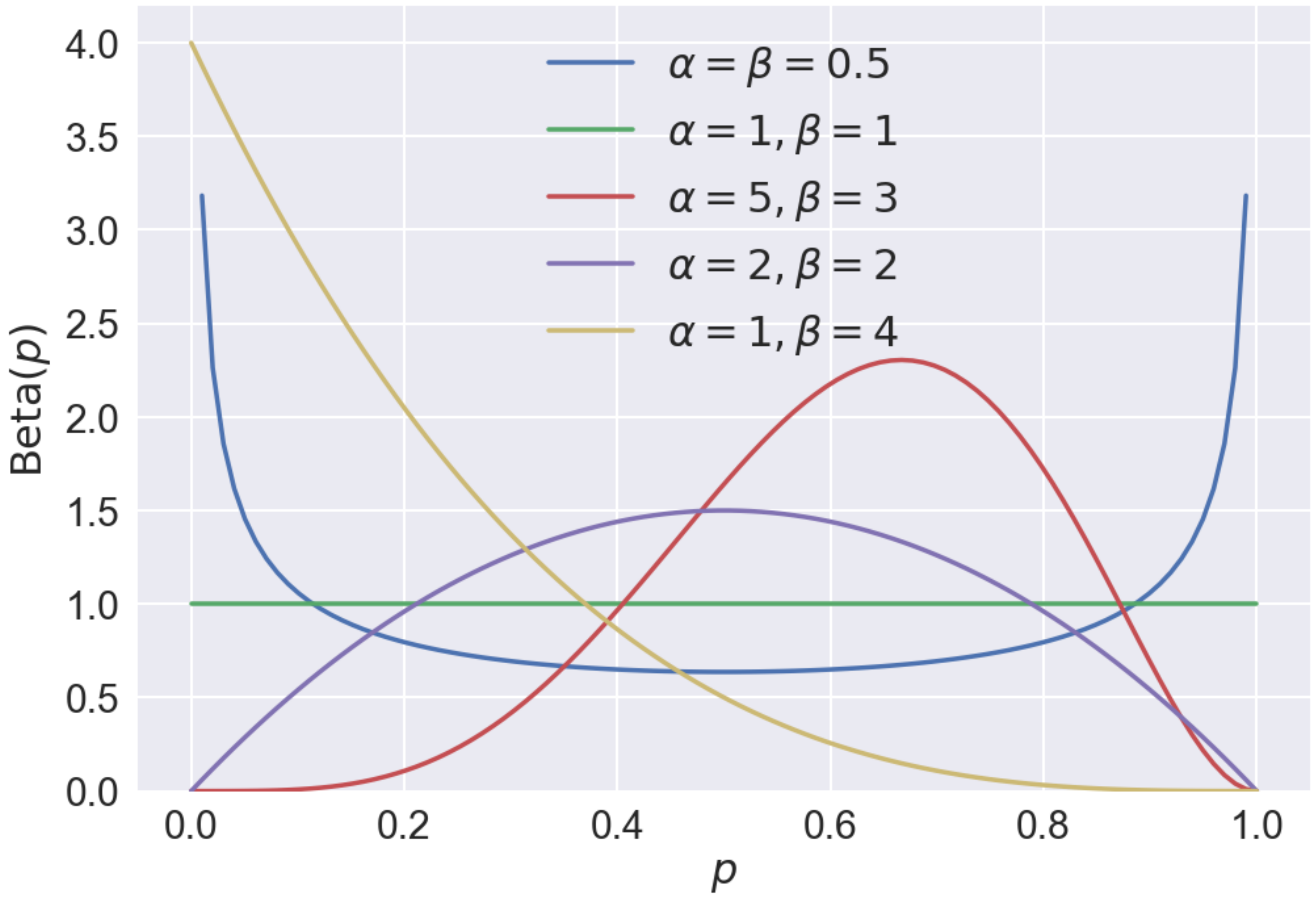
$$p \sim \mathrm{Beta}\left(\alpha,\beta\right)$$

Posterior

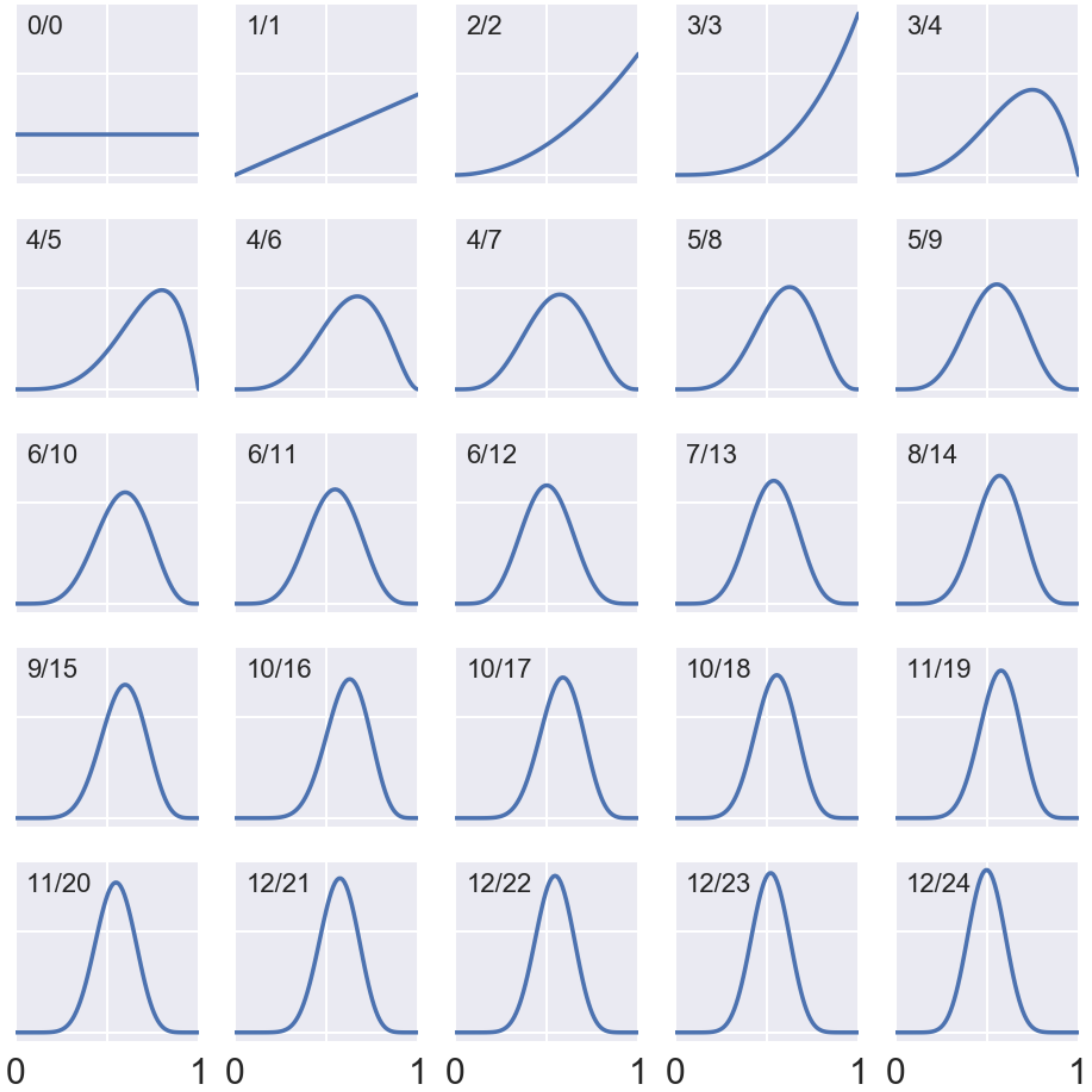
$$P\left(p \mid k,n\right) \propto P\left(k,n \mid p\right) P\left(p\right)$$

$$P\left(p \mid k,n\right) = \mathrm{Beta}\left(\alpha+k,\beta+n-k\right)$$

$p \sim \mathrm{Beta}\left(\alpha,\beta\right)$



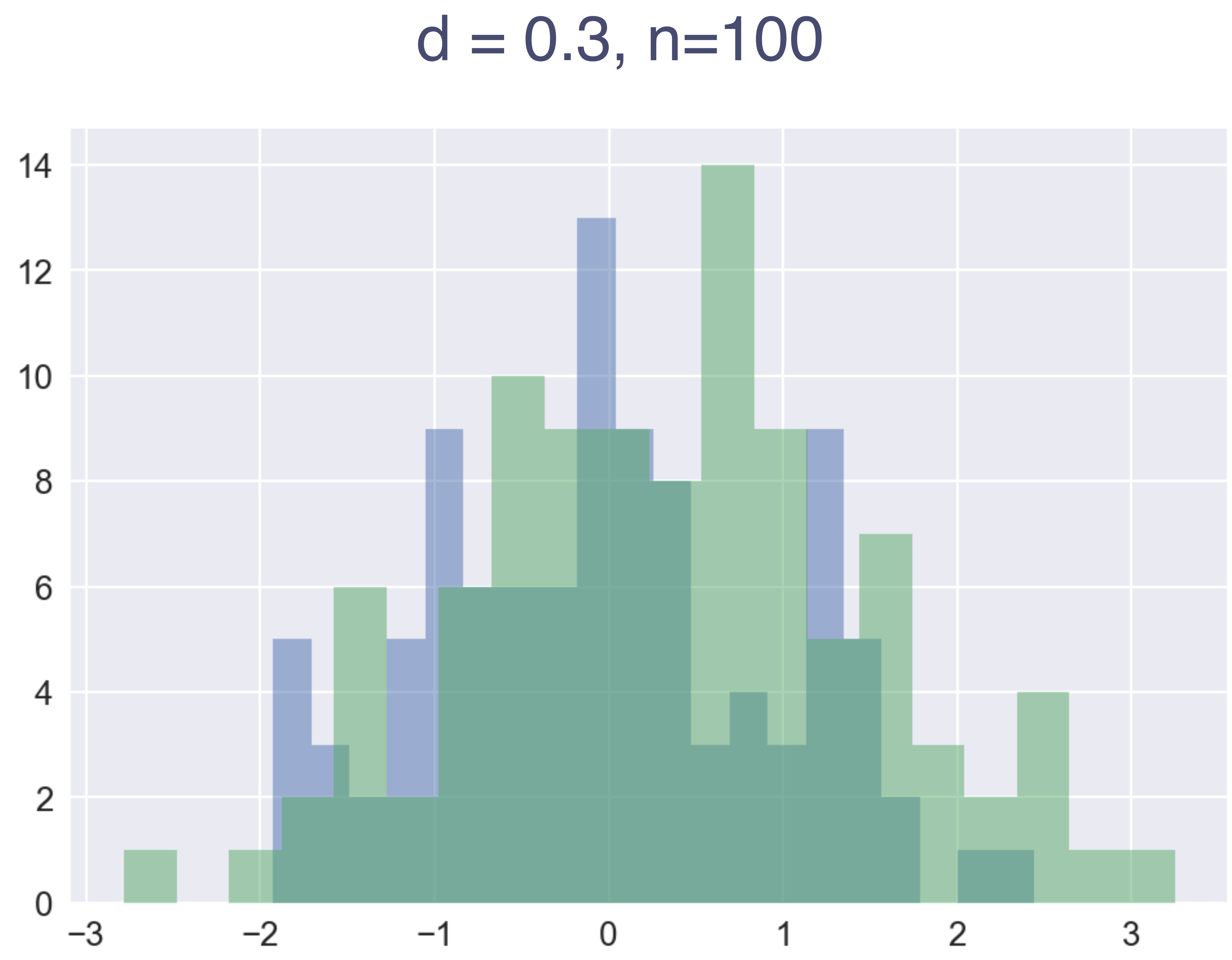
Posterior distributions, p = 0.5



Now with Python

```python
from scipy import stats

# Our data, events, array of 0s and 1s
successes = events.sum()
failures = len(events) - successes

# Prior parameters
α, β = 1, 1

# Calculate posterior distribution
posterior = stats.beta(α + successes, β + failures)

# Mean and 95% credible interval
mean = posterior.mean()
CR = posterior.interval(0.95)

# Posterior distribution for plotting
xs = np.linspace(0, 1, num=100)
pdf = posterior.pdf(xs)
```

Back to the T-test



What do we want?

Means: μ_1, μ_2

Standard deviations: σ_1, σ_2

Estimates: $\mu_2 - \mu_1$ and $\sigma_2 - \sigma_1$

Assess the ***uncertainty*** in our estimates

Forget the T-test

Likelihood

$y_1 \sim \text{Normal}(\mu_1, \sigma_1)$

$y_2 \sim \text{Normal}(\mu_2, \sigma_2)$

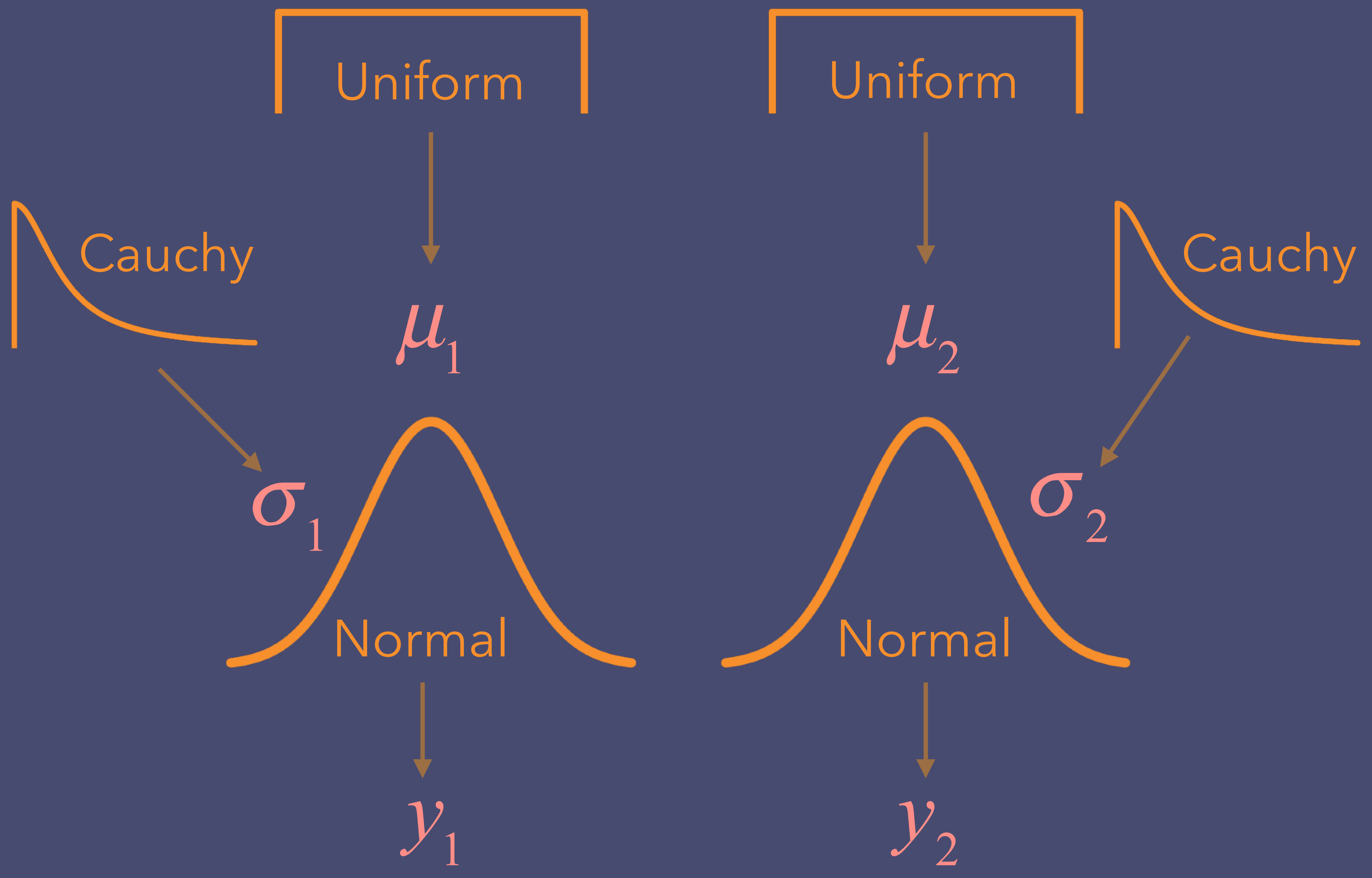
Priors

$\mu_1 \sim \text{Uniform}(-5, 5)$

$\sigma_1 \sim \text{Half Cauchy}(\gamma = 5)$

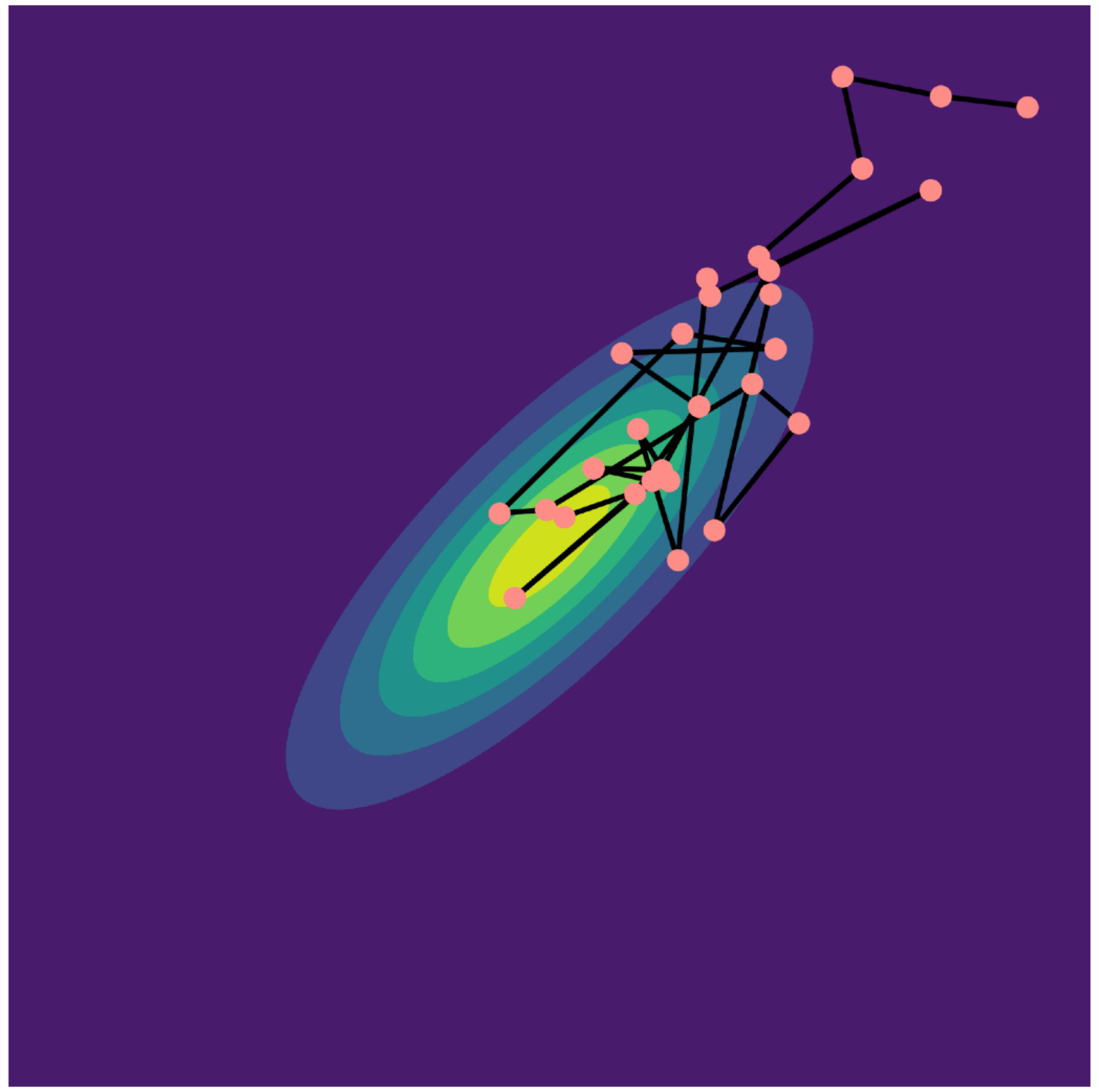
$\mu_2 \sim \text{Uniform}(-5, 5)$

$\sigma_2 \sim \text{Half Cauchy}(\gamma = 5)$

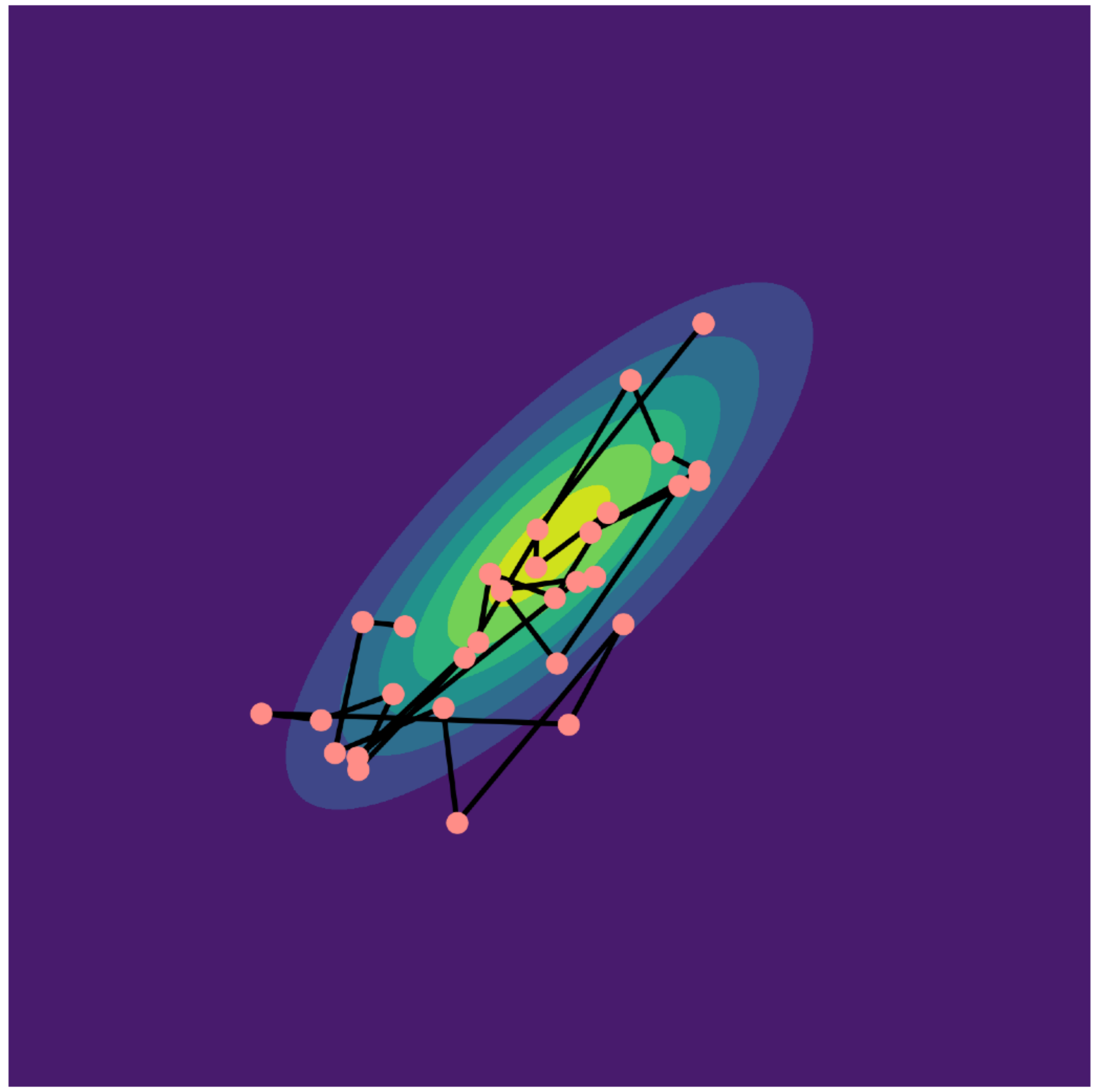


Markov Chain Monte Carlo

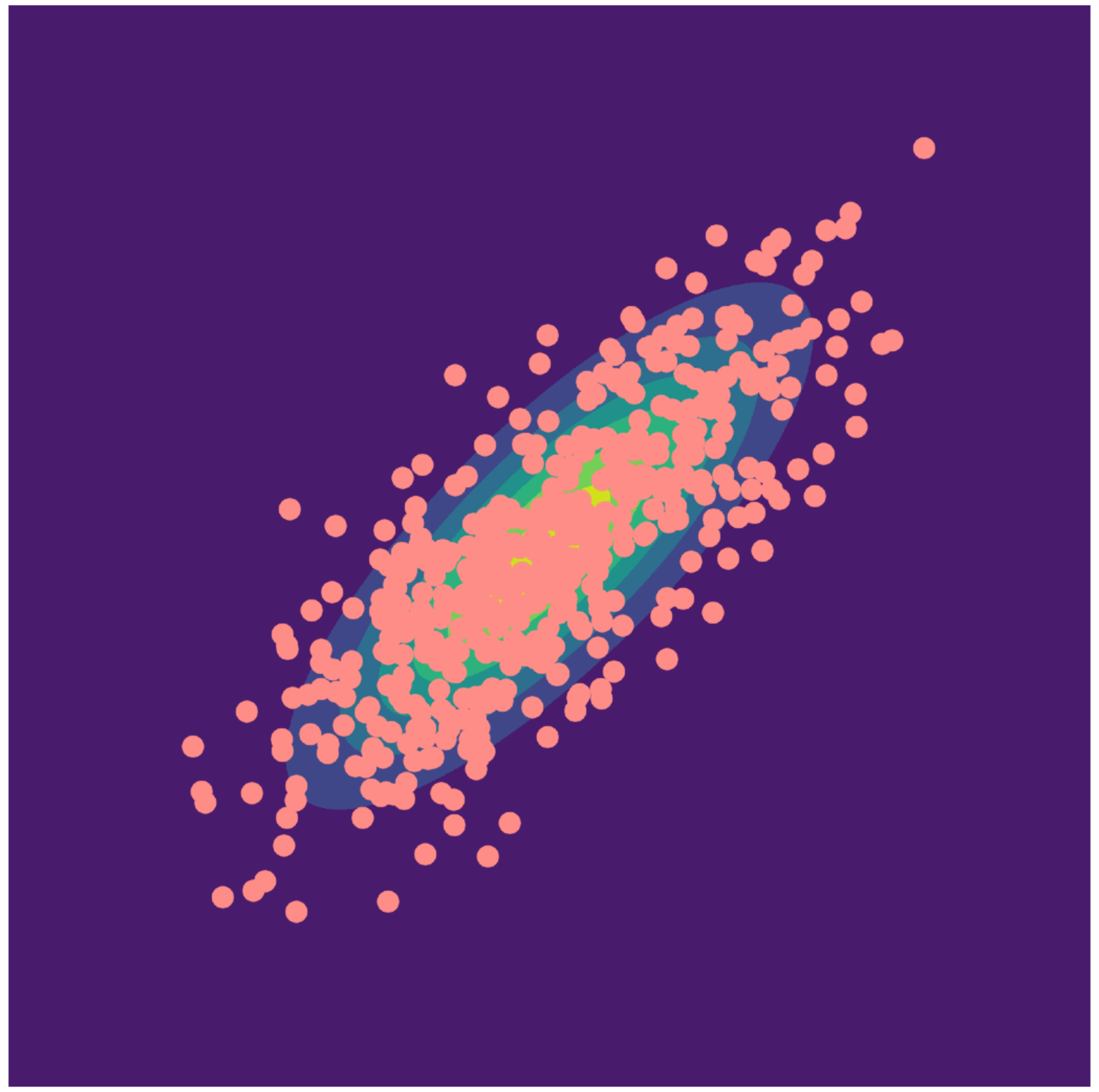
Starting out



Converged



Sampled



Bayesian Data Analysis in Python

- PyMC
- PyStan
- Emcee
- Edward
- **Sampyl**

Independent Samples Model

Define the model in Sampyl

```python
import sampyl as smp
from sampyl import np
```

Independent Samples Model

Define the model in Sampyl

```
def logp(μ_1, σ_1, μ_2, σ_2):

        model = smp.Model()

        # Priors for means
        model.add(smp.uniform(μ_1, -5, 5),
                  smp.uniform(μ_2, -5, 5))

        # Priors for standard devs
        model.add(smp.half_cauchy(σ_1, beta=5),
                  smp.half_cauchy(σ_2, beta=5))

        # Data Likelihood
        model.add(smp.normal(group1, mu=μ_1, sig=σ_1),
                  smp.normal(group2, mu=μ_2, sig=σ_2))

        return model()
```

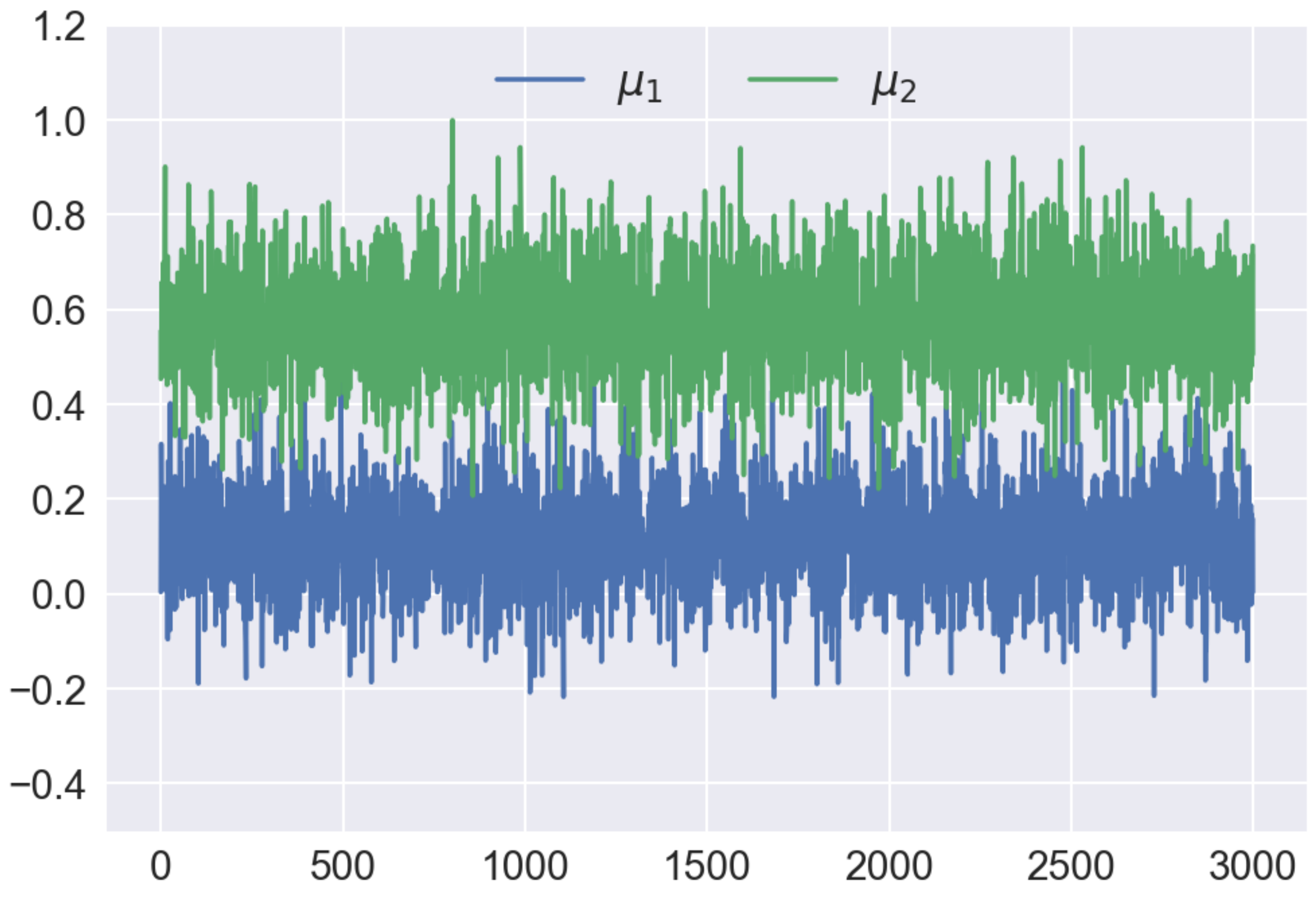
Independent Samples Model

Sample from the posterior

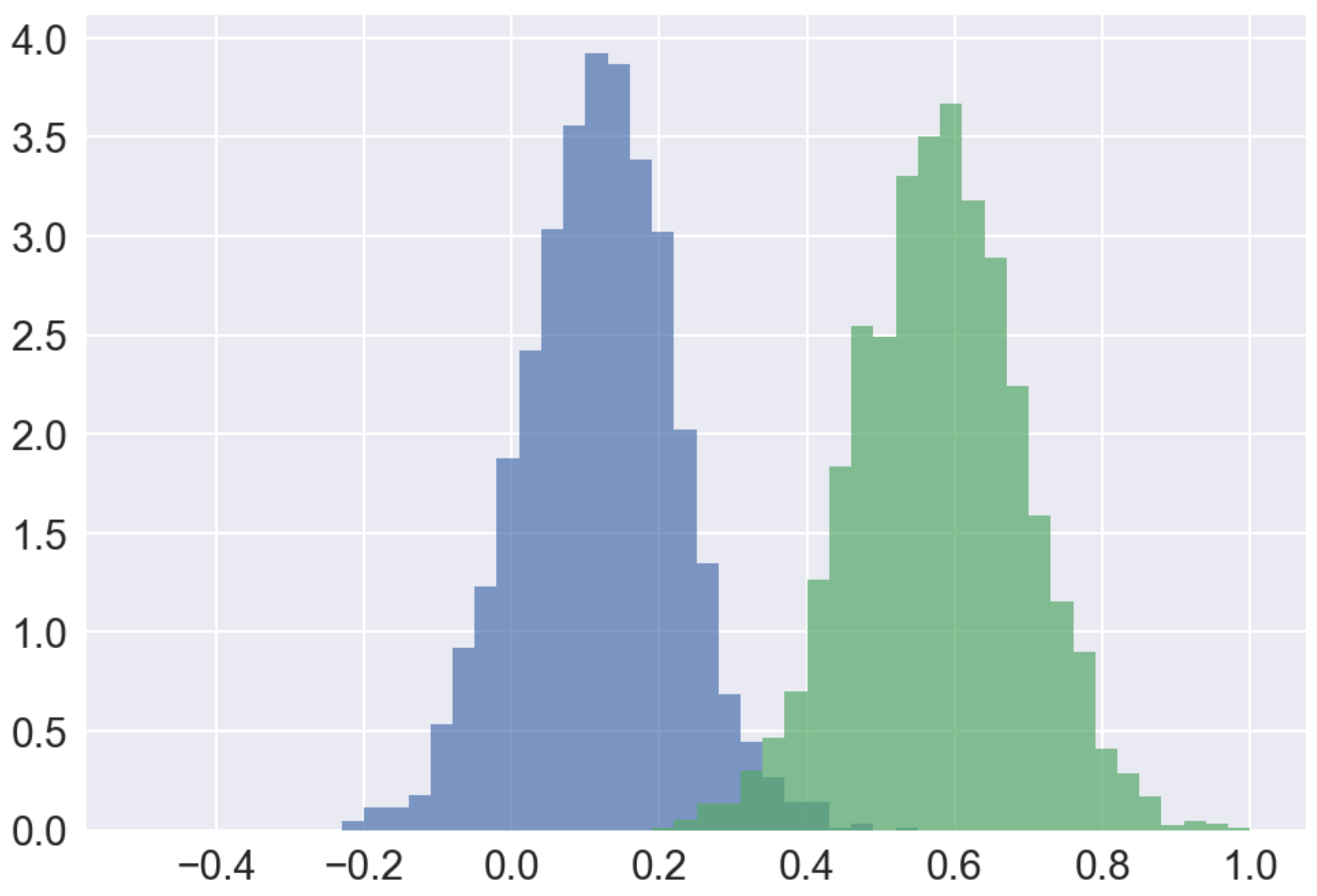
```
start = {'μ_1': 0., 'σ_1': 1., 'μ_2': 0., 'σ_2': 1.}
sampler = smp.NUTS(logp, start)
chain = sampler(6100, burn=100, thin=2)
```

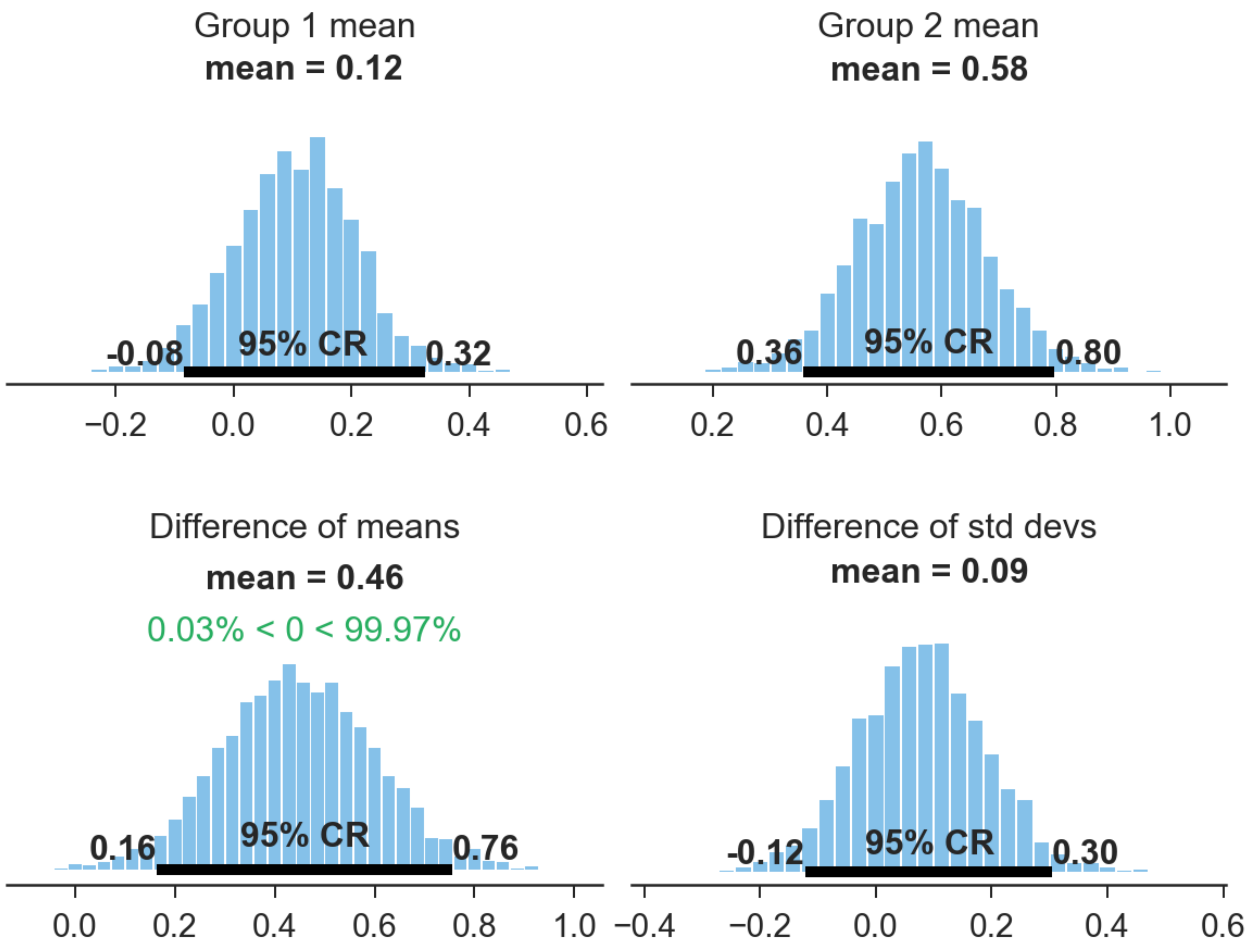
```
Progress: [##############################] 6100 of 6100 samples
```

Posterior chains



Posterior distributions





Group 1 mean
mean = 0.12
-0.08
95% CR
0.32
−0.2
0.0
0.2
0.4
0.6
Group 2 mean
mean = 0.58
0.36
95% CR
0.80
0.2
0.4
0.6
0.8
1.0
Difference of means
mean = 0.46
0.03% < 0 < 99.97%
0.16
95% CR
0.76
0.0
0.2
0.4
0.6
0.8
1.0
Difference of std devs
mean = 0.09
-0.12
95% CR
0.30
−0.4
−0.2
0.0
0.2
0.4
0.6

Boston Housing Prices

Linear regression, Bayesian style

From attributes predict median home value

ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
11	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
12	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
13	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
14	0.62976	0.0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
15	0.63796	0.0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2

Linear Model

Define the model

Likelihood

$$\hat{y} = \sum_{i}^{m} \beta_i x_i + \alpha$$

$$y \sim \mathrm{Normal}\left(\hat{y}, \epsilon\right)$$

Priors

$$\beta \sim \mathrm{Uniform}\left(-10, 10\right)$$

$$\alpha \sim \mathrm{Uniform}\left(0, 100\right)$$

$$\epsilon \sim \mathrm{Half\,Cauchy}\left(\gamma = 5\right)$$

Linear Model

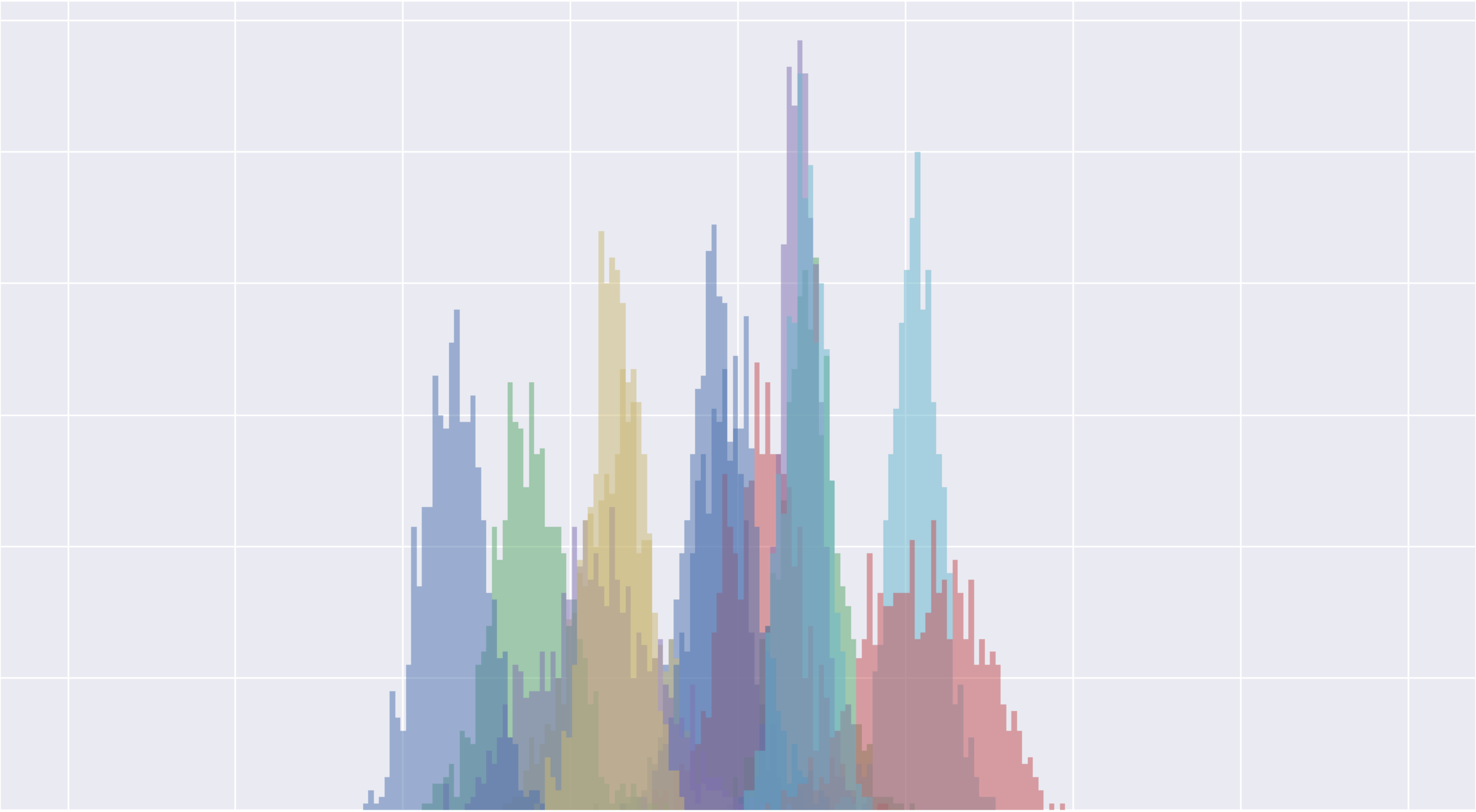
Define the model in Sampyl

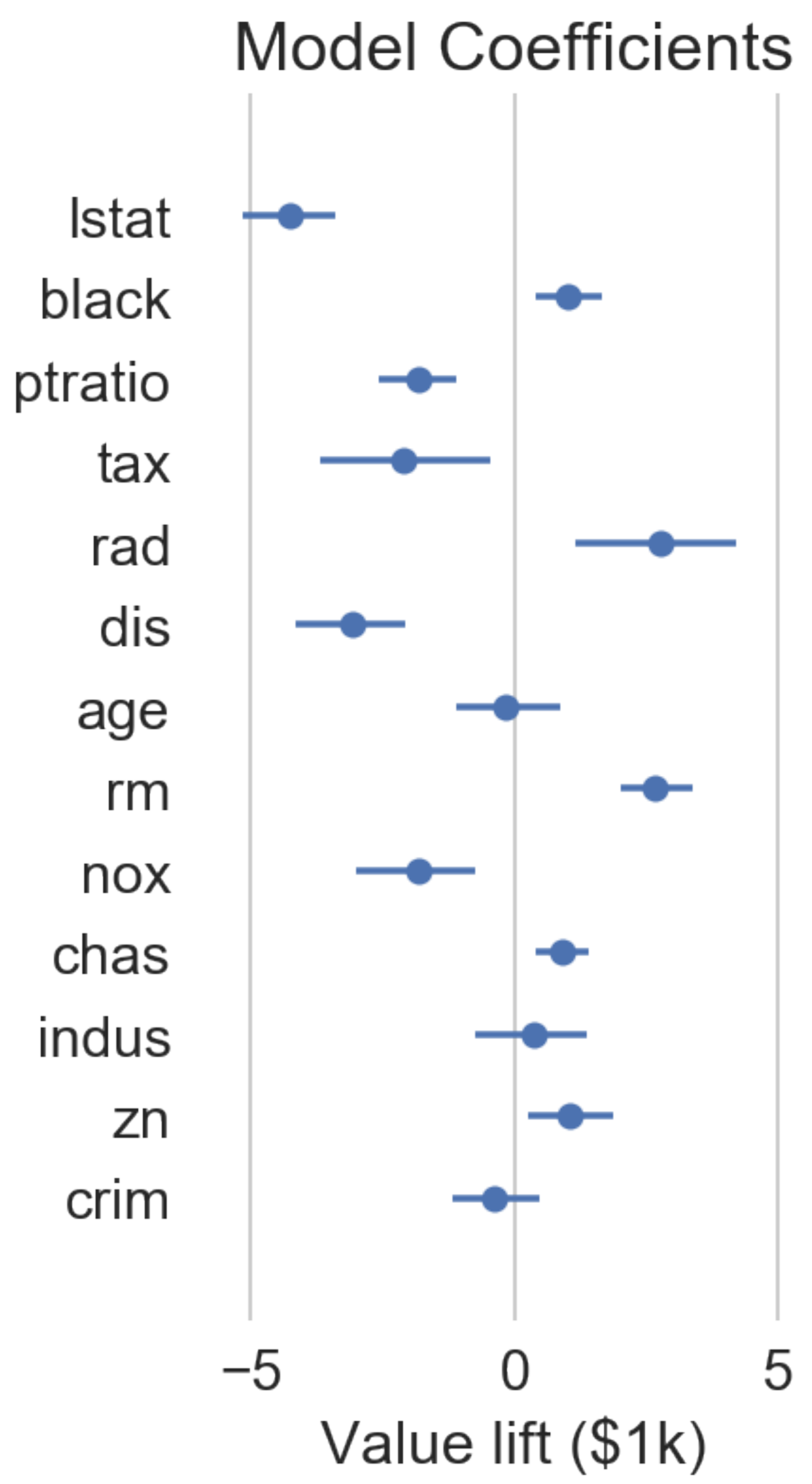
```
# Here, β is an array of coefficients
def logp(β, α, ϵ):

    model = smp.Model()

    model.add(smp.half_cauchy(ϵ, 5))

    model.add(smp.uniform(β, lower=-10, upper=10),
              smp.uniform(α, lower=0, upper=100))

    estimate = np.dot(X, β) + α
    model.add(smp.normal(Y, mu=estimate, sig=ϵ))

    return model()
```

Linear Model

Sample from the posterior

```
start = {'β': np.zeros(X.shape[1]), 'α': 0., 'ϵ': 1.}
sampler = smp.Slice(logp, start)
chain = sampler(10000, burn=1000, thin=3)
```

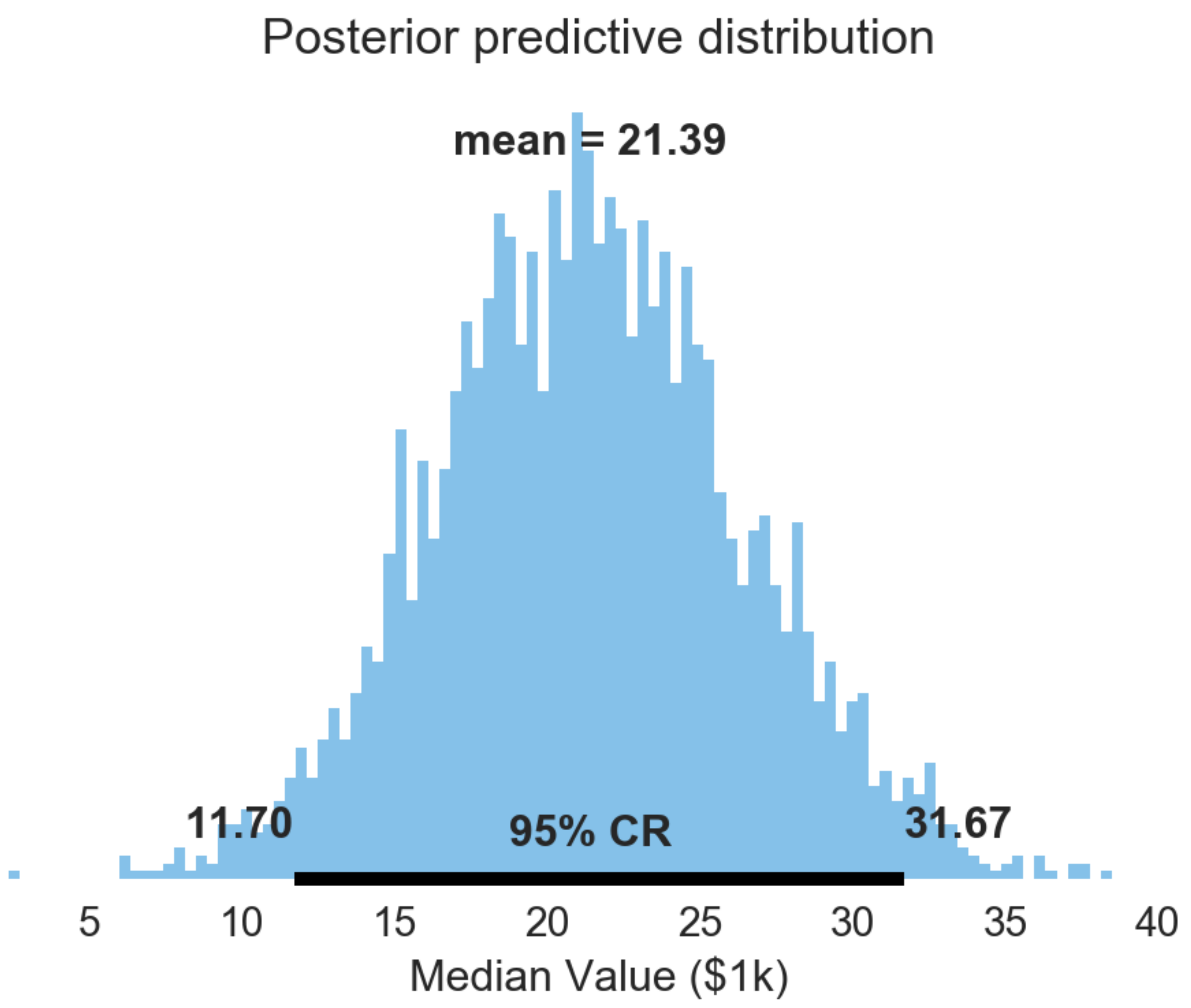



Model Coefficients
lstat
black
ptratio
tax
rad
dis
age
rm
nox
chas
indus
zn
crim
−5
0
5
Value lift ($1k)

Linear Model

Making Predictions

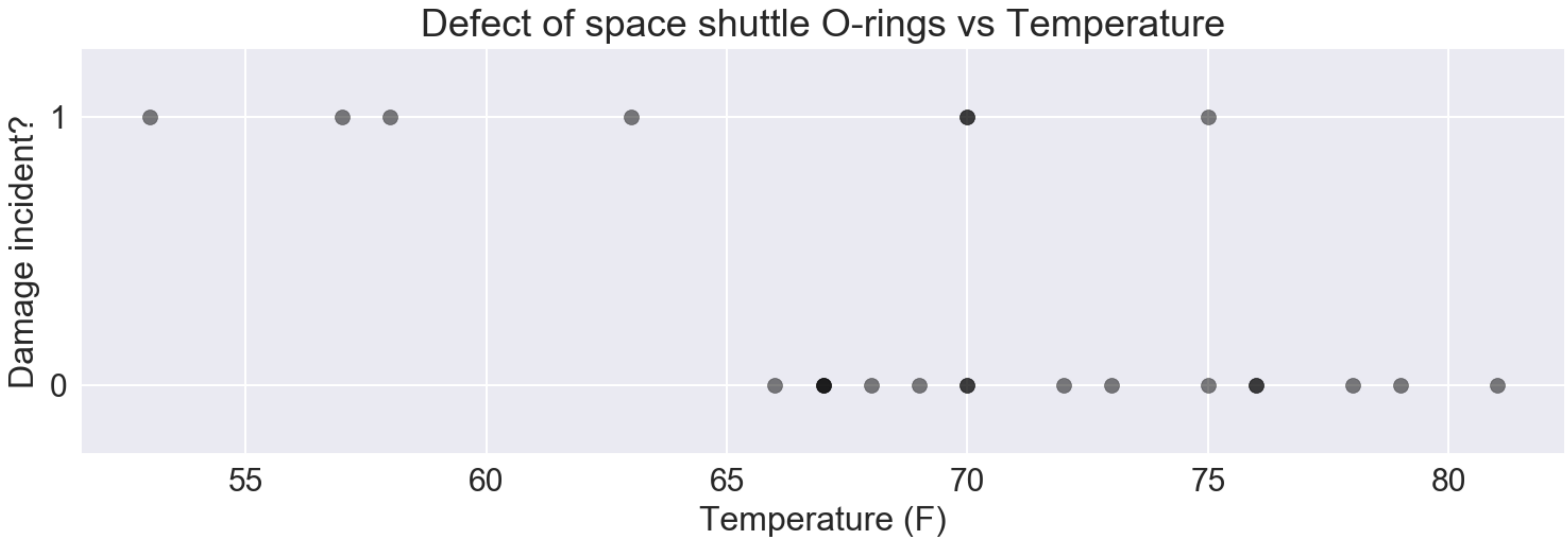
```
# Get one random input
predict_data = np.random.randn(X.shape[1])

idx = np.random.choice(np.arange(chain.size))
post_sample = chain[idx]

estimate = np.dot(predict_data, post_sample.β) + post_sample.α
single_predict = stats.norm(loc=estimate, scale=post_sample.ϵ).rvs()

# Now make posterior predictive distribution to get uncertainty
estimate = np.dot(predict_data, chain.β.T) + chain.α
post_predict = stats.norm(loc=estimate, scale=chain.ϵ).rvs()
```



Posterior predictive distribution
mean = 21.39
11.70
95% CR
31.67
5
10
15
20
25
30
35
40
Median Value ($1k)

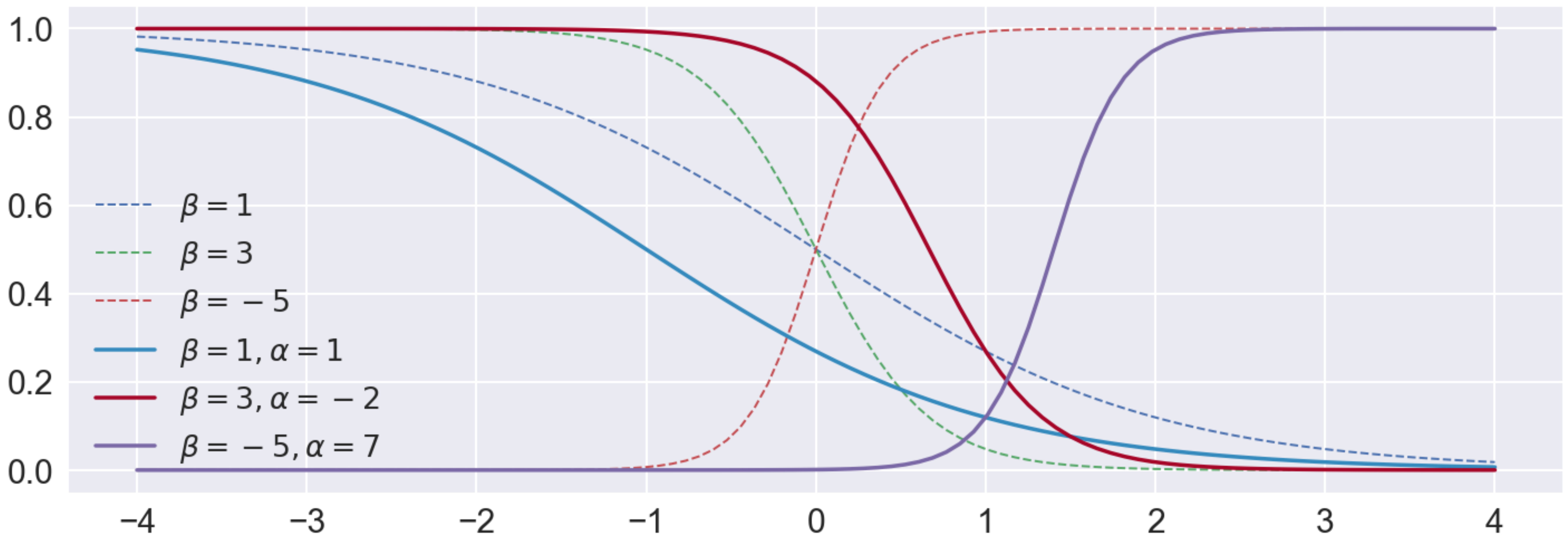
Challenger Disaster

Logistic regression, Bayesian style



Defect of space shuttle O-rings vs Temperature
Damage incident?
1
0
55
60
65
70
75
80
Temperature (F)

The logistic function $\sigma = \dfrac{1}{1+e^{\beta t+\alpha}}$



Logistic Model

Define the model

Likelihood

$$\sigma = \frac{1}{1+e^{x}}$$

$$p = \sigma\left(\beta t + \alpha\right)$$

$$y \sim \text{Bernoulli}\left(p\right)$$

Priors

$$\alpha \sim \text{Normal}\left(0, 1000\right)$$

$$\beta \sim \text{Normal}\left(0, 1000\right)$$

Logistic Model

Define the model in Sampyl

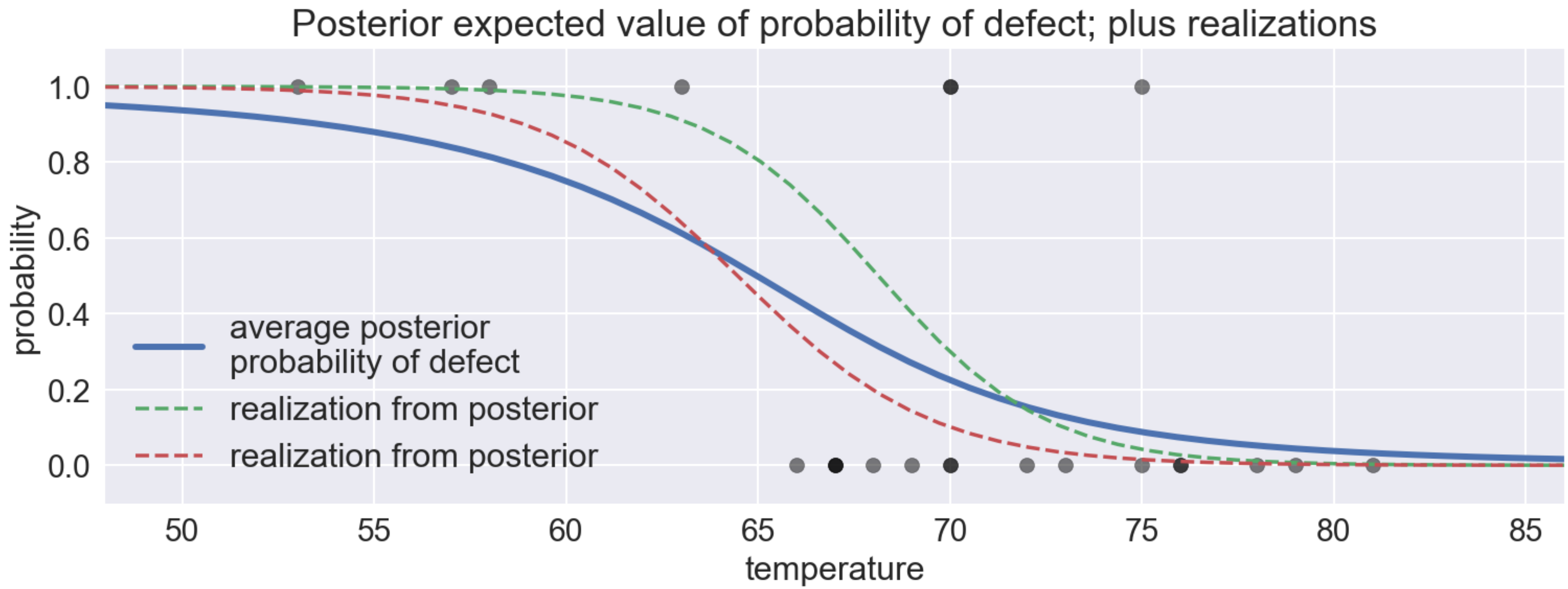
```
def logistic(x):
    return 1/(1 + np.exp(x))

def logp(β, α):

    model = smp.Model()

    # Wide normal priors on coefficients
    model.add(smp.normal(α, sig=1000),
              smp.normal(β, sig=1000))

    # Add log-likelihood
    model.add(smp.bernoulli(damage, logistic(temperature*β + α)))

    return model()
```

Logistic Model

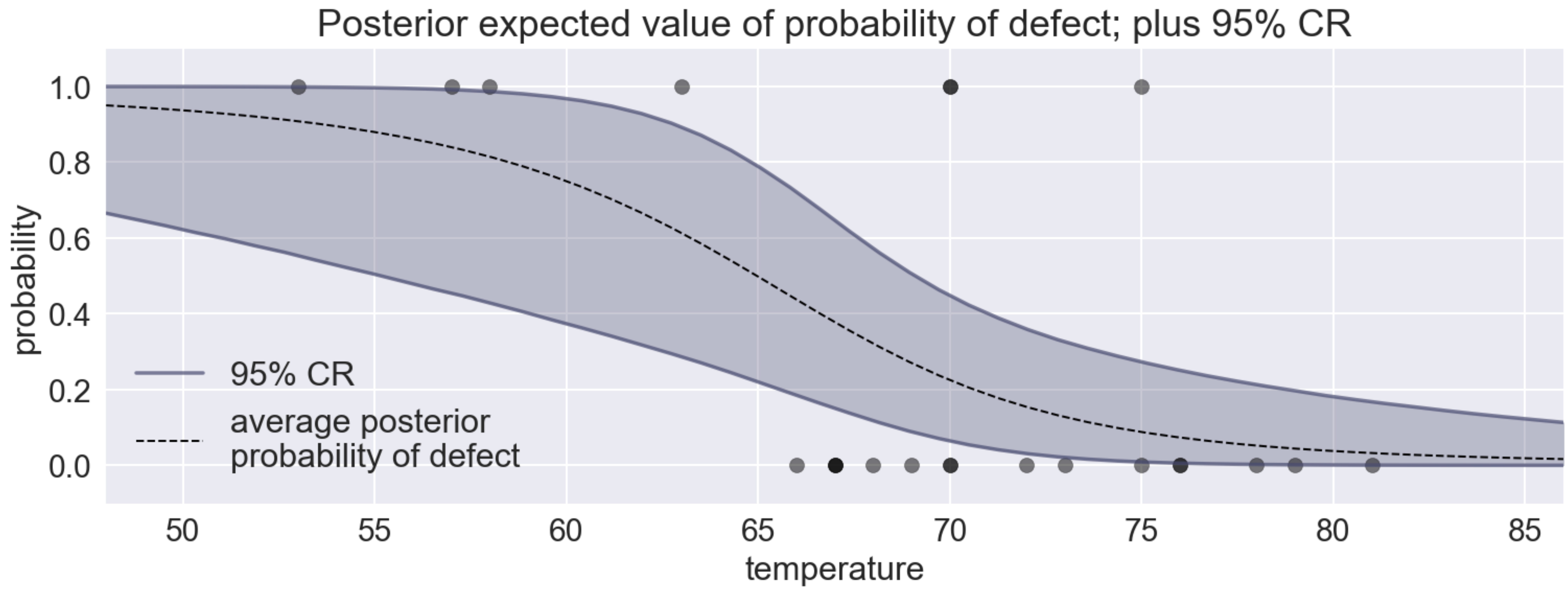
Sample from the posterior

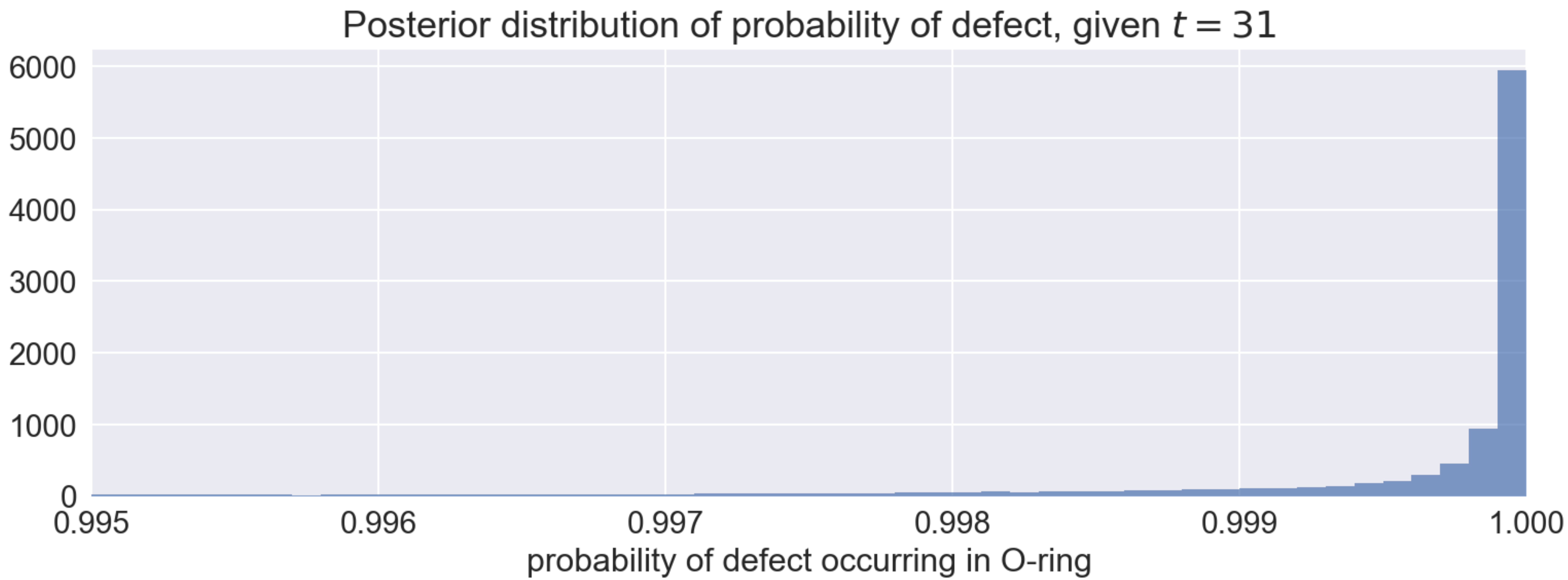
```
start = {'β': 0., 'α': 0.}
sampler = smp.Slice(logp, start)
chain = sampler(220000, burn=20000, thin=3)
```

3000
2000
1000
0
−40
−30
−20
−10
0
Posterior for α
3000
2000
1000
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Posterior for β



Posterior expected value of probability of defect; plus realizations
probability
temperature
1.0
0.8
0.6
0.4
0.2
0.0
50
55
60
65
70
75
80
85
average posterior
probability of defect
realization from posterior
realization from posterior


Posterior expected value of probability of defect; plus 95% CR
probability
1.0
0.8
0.6
0.4
0.2
0.0
95% CR
average posterior
probability of defect
50
55
60
65
70
75
80
85
temperature


Posterior distribution of probability of defect, given $t = 31$
6000
5000
4000
3000
2000
1000
0
0.995
0.996
0.997
0.998
0.999
1.000
probability of defect occurring in O-ring

Thank you!

Questions?