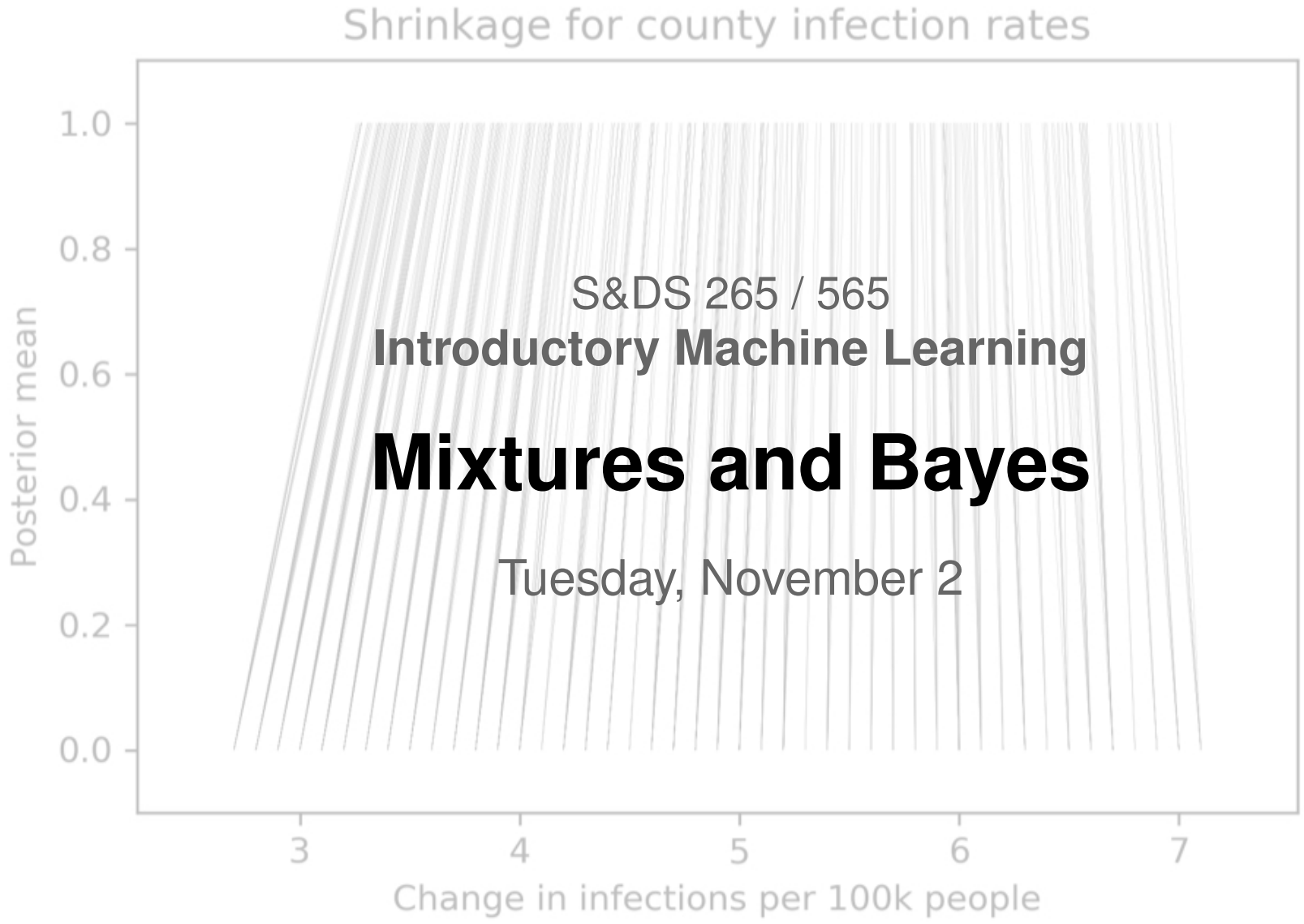


S&DS 265 / 565

Introductory Machine Learning

Mixtures and Bayes

Tuesday, November 2

Yale

Goings on

- Assignment 4 due tonight
- Assignment 5 is out
- Questions?

For Today

- Embeddings
- Mixtures
- Bayes

Core idea of embeddings

- Form a language model but replace classes by vectors, one for each word
- Use PMI-like scores to fit the vectors
- Optimize the embedding vectors using SGD
- “Two words are similar if they appear with similar words”
- Can be applied whenever have cooccurrence data (Spotify)

Method 1: Word2Vec

Language model is

$$p(w_2 \mid w_1) = \frac{\exp(\phi(w_2)^T \phi(w_1)}{\sum_w \exp(\phi(w)^T \phi(w_1))}.$$

- Trained on pairs of words that co-occur in a window
- Stochastic gradient descent over embedding vectors $\phi(w) \in \mathbb{R}^d$

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

Method 2: PCA

A different, related approach is to use PCA of PMI:

- Form $V \times V$ matrix of pointwise mutual information values

$$\log\left(\frac{p_{\text{near}}(w_1, w_2)}{p(w_1)p(w_2)}\right)$$

- Compute top k eigenvectors $\phi_1, \ldots, \phi_k$
- For each word w, define embedding as

$$\phi(w) \equiv (\phi_{1w}, \phi_{2w}, \ldots, \phi_{kw})^T$$

Method 3: GloVe

Try to minimize the squared error

$$\mathcal{O}(\phi) = \sum_{w_1, w_2} f(c_{w_1,w_2}) \left(\phi(w_1)^T \phi(w_2) - \log c_{w_1,w_2} \right)^2$$

where $c_{w,w'}$ are cooccurrence counts in a window (PMI)

- The parameters are the embedding vectors $\{\phi(w)\}$
- Unlike least-squares regression, this is a nonconvex minimization problem (wiggly, not bowl-shaped)
- Trained using stochastic gradient descent

Pennington et al., "GloVe: Global vectors for word representation," (2015)

SGD for Word2Vec

- Sample a word pair (w_1, w_2)
- Loss is $\mathcal{L} = -\log p(w_2 \mid w_1)$
- Gradient update for $\phi(w_2)$ is

$$\begin{aligned}\phi(w_2) &\leftarrow \phi(w_2) - \eta \frac{\partial \mathcal{L}}{\partial \phi(w_2)} \\ &= \phi(w_2) + \eta \left(1 - p(w_2 \mid w_1)\right) \phi(w_1)\end{aligned}$$

Vector $\phi(w_2)$ is updated to be more similar to $\phi(w_1)$

SGD for GloVe

- Sample a word pair (w_1, w_2)
- Loss is $\mathcal{L} = \left(\phi^T(w_1)phi(w_2) - \log c_{w_1,w_2}\right)^2$
- Gradient update for $\phi(w_2)$ is

$$\begin{aligned}\phi(w_2) &\leftarrow \phi(w_2) - \eta \frac{\partial \mathcal{L}}{\partial \phi(w_2)} \\ &= \phi(w_2) + 2\eta \left(\log c_{w_1,w_2} - \phi(w_1)^T \phi(w_1)\right) \phi(w_1)\end{aligned}$$

Vector $\phi(w_2)$ is updated to be more similar to $\phi(w_1)$

Where are we going?

- So far: Mostly “surface representations” & explicit features
  - Listing information for CA housing
  - Political blogs — word counts

Where are we going?

- So far: Mostly “surface representations” & explicit features
  - Listing information for CA housing
  - Political blogs — word counts
- Next: “Hidden” representations and latent variables

George Washington

1789

Among the many interesting objects which will engage your attention that of providing for the common defense will merit particular regard. To be prepared for war is one of the most effectual means of preserving peace.

Franklin Roosevelt

1941

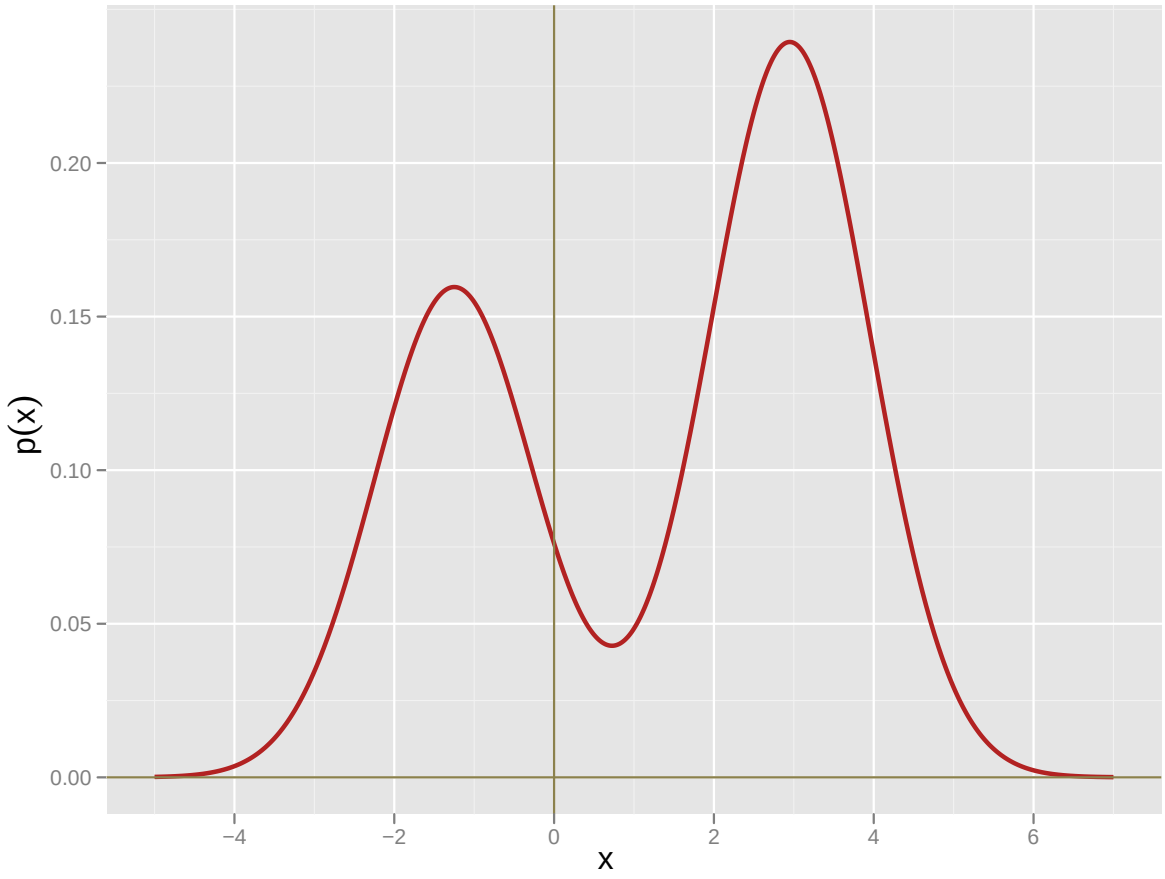
Such a peace would bring no security for us or for our neighbors. As a nation, we may take pride in the fact that we are softhearted; but we cannot afford to be soft-headed.

The elephants in the room

Mixtures

- Key technique: Mixture models
- Mixtures have latent variables
- Flexible tool
- Simple and difficult at the same time

Gaussian Mixture



$$p(x) = \frac{2}{5}\phi(x; -1.25, 1) + \frac{3}{5}\phi(x; 2.95, 1)$$

Mixtures

- *Mixture of f and g*:

$$p(x) = \theta f(x) + (1 - \theta) g(x)$$

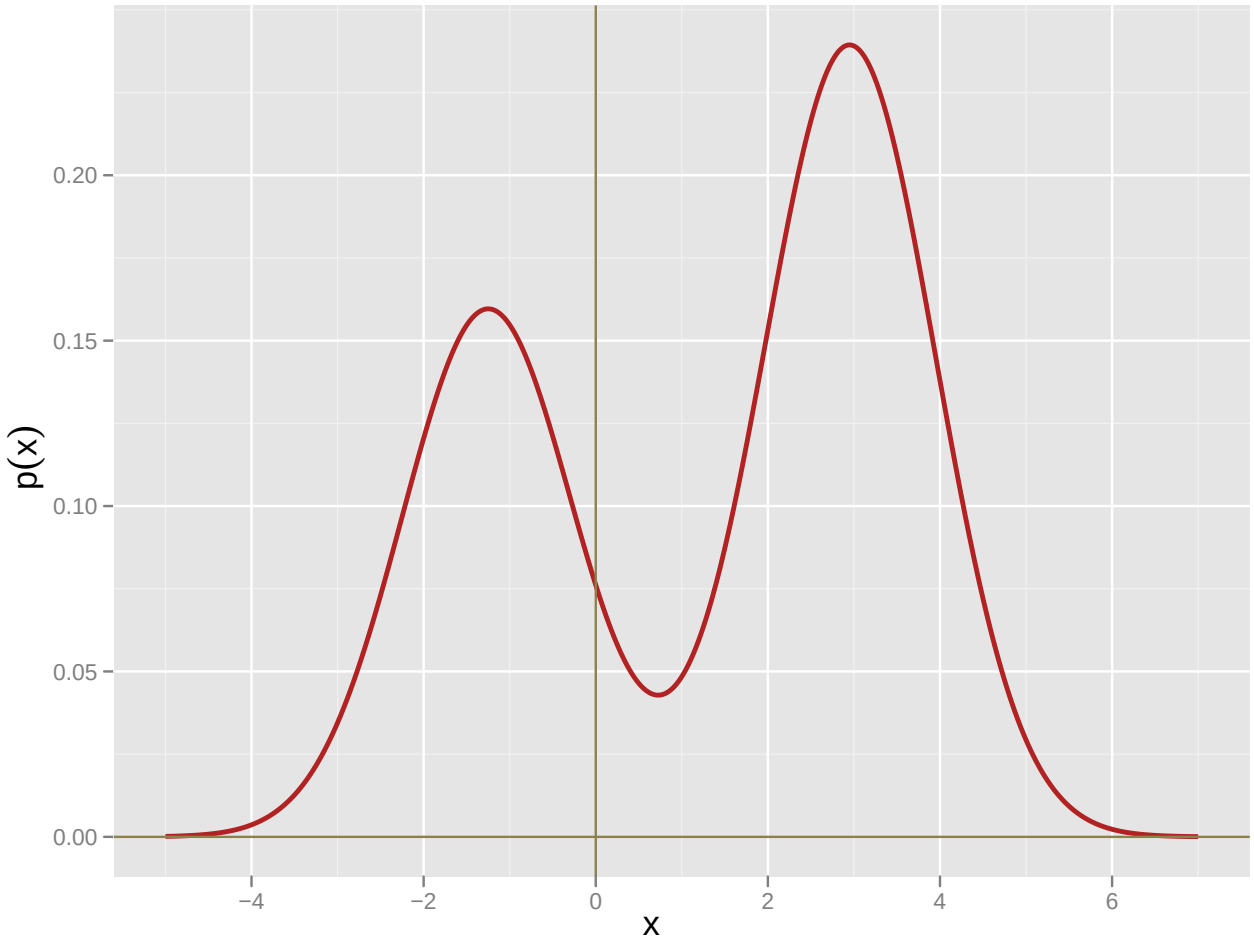
  Simplest, most common kind of latent variable model

- *Hidden variable represention*: Define $Z \sim \text{Bernoulli}(\theta)$ and

$$p(x) = \sum_{z=0,1} p(x \mid z)\, p(z)$$

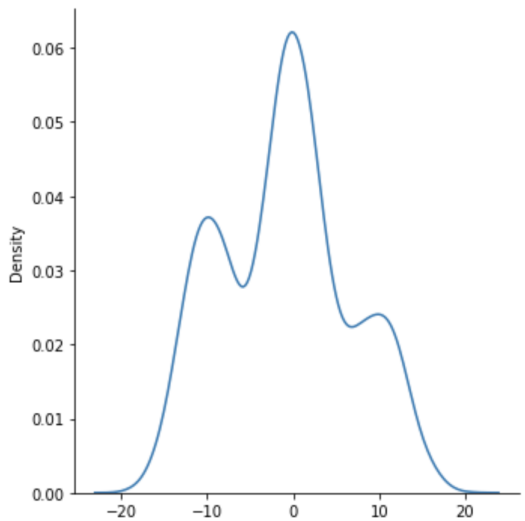
  with $p(x \mid 1) = f(x)$, $p(x \mid 0) = g(x)$, $p(z) = \theta^z (1 - \theta)^{(1-z)}$.

Gaussian Mixture: All the Key Concepts



Let’s go to the notebook

In [3]:
```
from scipy.stats import multinomial
n = 10000
eta = [.3, .5, .2]
sigma = 3
z = multinomial.rvs(p=eta, n=1, size=n)

x0 = norm.rvs(loc=-10, scale=sigma, size=n)
x1 = norm.rvs(loc=0, scale=sigma, size=n)
x2 = norm.rvs(loc=10, scale=sigma, size=n)

x = z[:,0]*x0 + z[:,1]*x1 + z[:,2]*x2

_ = sbn.displot(x, kind='kde')
```



Bayesian Inference

The parameter θ of a model is viewed as a random variable. Inference usually carried out as follows:

- Choose a *generative model* $p(x \mid \theta)$ for the data.
- Choose a *prior distribution* $\pi(\theta)$ that expresses beliefs about the parameter before seeing any data.
- After observing data $\mathcal{D}_n = \{x_1, \ldots, x_n\}$, update beliefs and calculate the *posterior distribution* $p(\theta \mid \mathcal{D}_n)$.

Bayes' Theorem

A simple consequence of conditional probability:

$$\begin{aligned}\mathbb{P}(A \mid B) &= \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} \\ &= \frac{\mathbb{P}(B \mid A)\, \mathbb{P}(A)}{\mathbb{P}(B)}\end{aligned}$$

Bayes' Theorem

The posterior distribution can be written as

$$p(\theta \mid x_1, \ldots, x_n) = \frac{p(x_1, \ldots, x_n \mid \theta)\pi(\theta)}{p(x_1, \ldots, x_n)} = \frac{\mathcal{L}_n(\theta)\pi(\theta)}{c_n} \propto \mathcal{L}_n(\theta)\pi(\theta)$$

where $\mathcal{L}_n(\theta)$ is the *likelihood function* and

$$c_n = p(x_1, \ldots, x_n) = \int p(x_1, \ldots, x_n \mid \theta)\pi(\theta)d\theta = \int \mathcal{L}_n(\theta)\pi(\theta)d\theta$$

is the normalizing constant, which is also called *evidence*.

Important Example

Take model $X \sim \text{Bernoulli}(\theta)$.

This is a "coin flip": $X = 1$ means "heads" and $X = 0$ means "tails."

Natural prior is Beta(α, β) distribution

$$\pi_{\alpha,\beta}(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

Important Example

Take model $X \sim \text{Bernoulli}(\theta)$.

Natural prior is Beta(α, β) distribution

$$\pi_{\alpha,\beta}(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

The scaling constant is scary looking:

$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$$

where $\Gamma(\cdot)$ is the "Gamma function"

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x}\, dx$$

---

https://en.wikipedia.org/wiki/Gamma_function

Important Example

$X \sim$ Bernoulli(θ) with data $\mathcal{D}_n = \{x_1, \ldots, x_n\}$. Prior Beta($\alpha, \beta$) distribution

$$\pi_{\alpha,\beta}(\theta) \;\propto\; \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

Let $s = \sum_{i=1}^{n} x_i$ be the number of "heads"

Posterior distribution $\theta \,|\, \mathcal{D}_n$ is another beta distribution!

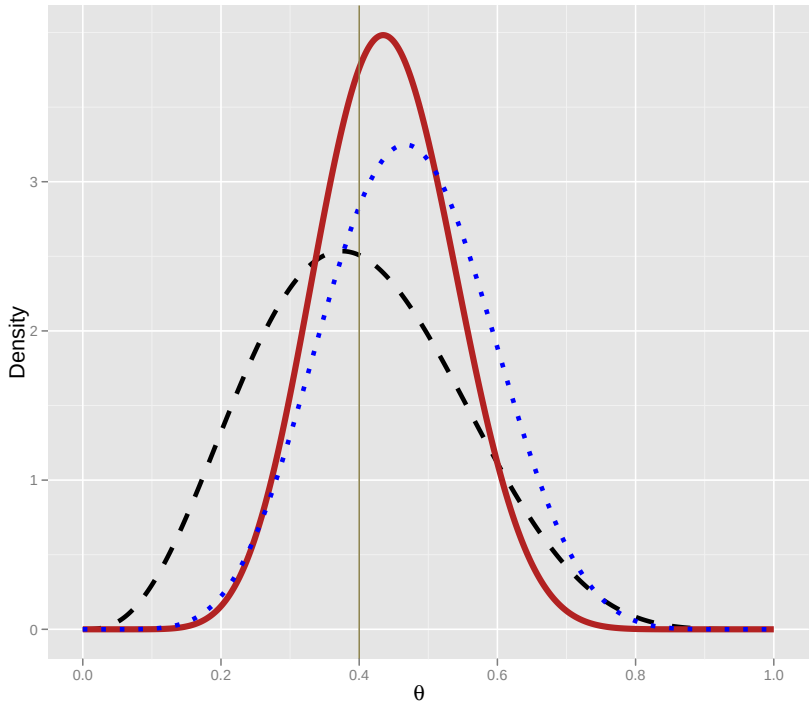
Specifically, with

$$\widetilde{\alpha} = \alpha + \text{number of heads} = \alpha + s$$
$$\widetilde{\beta} = \beta + \text{number of tails} = \beta + n - s$$

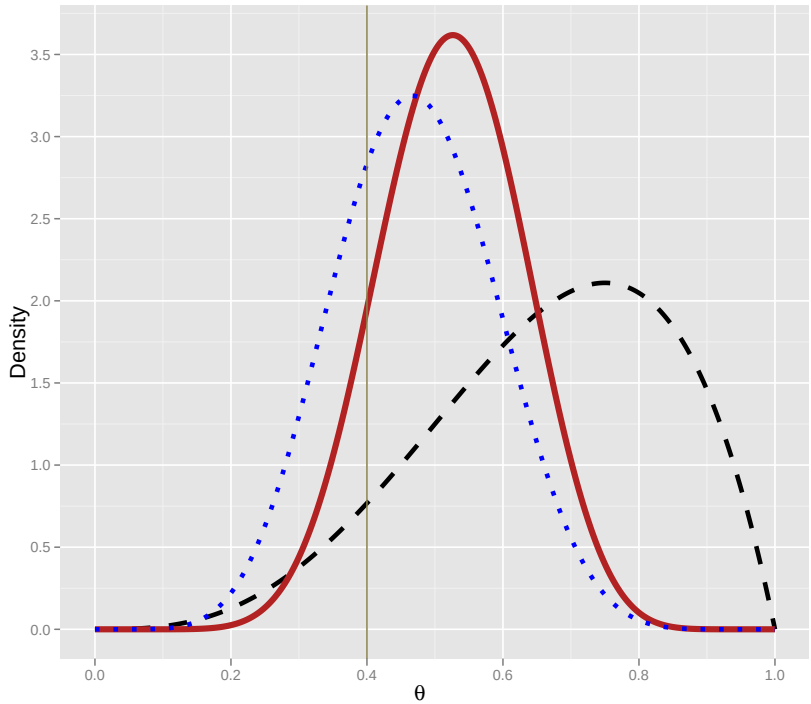
Showing this just uses the simple fact that $\theta^{\alpha-1}\theta^{x} = \theta^{x+\alpha-1}$

Example

$n = 15$ points sampled as $X \sim \text{Bernoulli}(\theta = 0.4)$, with $s = 7$ heads.



”good prior”



”bad prior”

Prior distribution (black-dashed), likelihood function (blue-dotted), posterior distribution (red-solid).

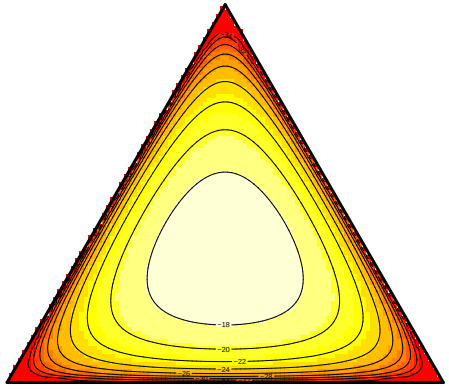
Dirichlet

Multinomial model with Dirichlet prior is generalization of the Bernoulli/Beta model.

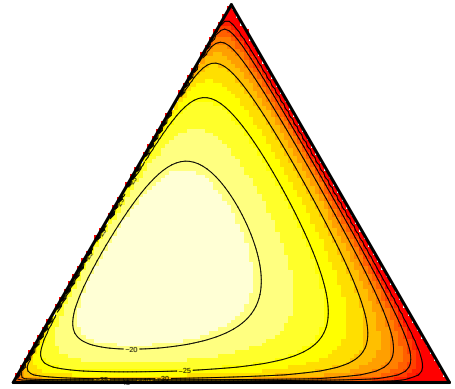
$$\text{Dirichlet}_\alpha(\theta) \;\propto\; \theta_1^{\alpha_1 - 1}\, \theta_2^{\alpha_2 - 1} \;\cdots \theta_K^{\alpha_K - 1}$$

where $\alpha = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}_+^K$ is a non-negative vector.

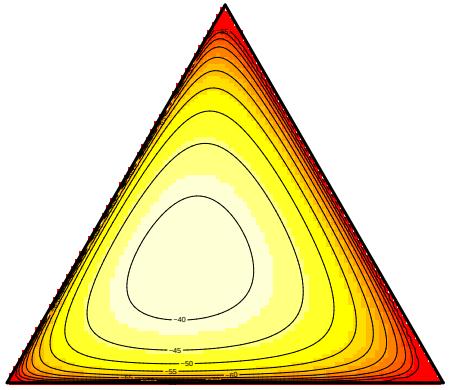
Example



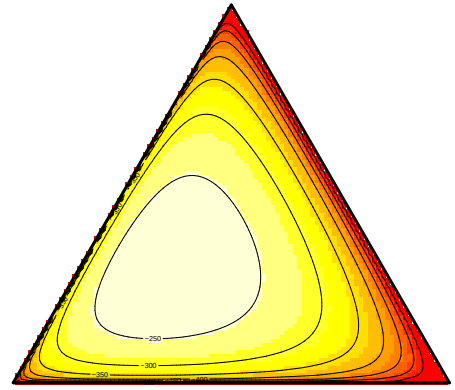
prior with Dirichlet(6,6,6)



likelihood function with $n = 20$



posterior distribution with $n = 20$



posterior distribution with $n = 200$

Let’s go to the notebook!

Demo code for Bayesian analysis

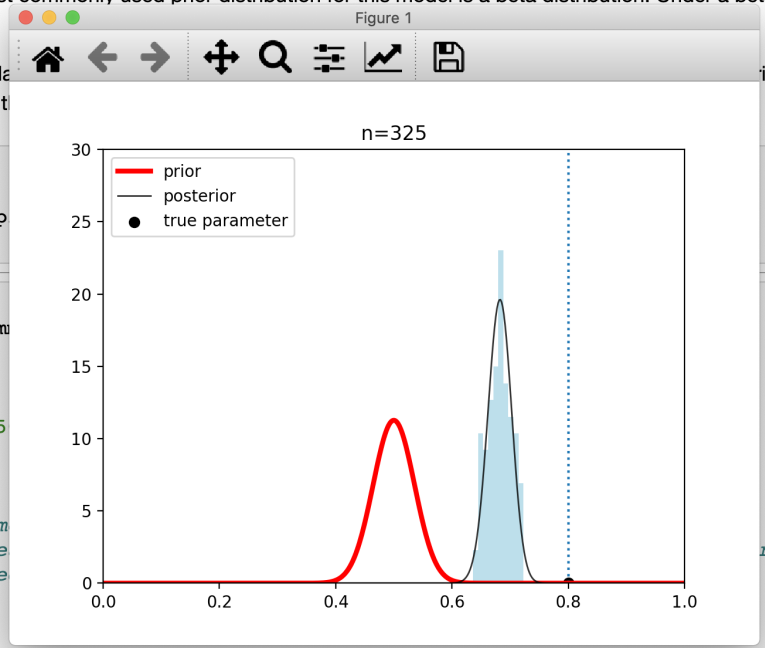
In this notebook we illustrate some of the basic models and priors for Bayesian inference. These concepts will be important for our discussions about "topic models."

First, we illustrate the situation where the parameter θ that we are modeling is a Bernoulli parameter. This can be thought of as the probability that flipping a certain coin comes up heads. The most commonly used prior distribution for this model is a beta distribution. Under a beta prior, the posterior distribution is again a beta distribution.

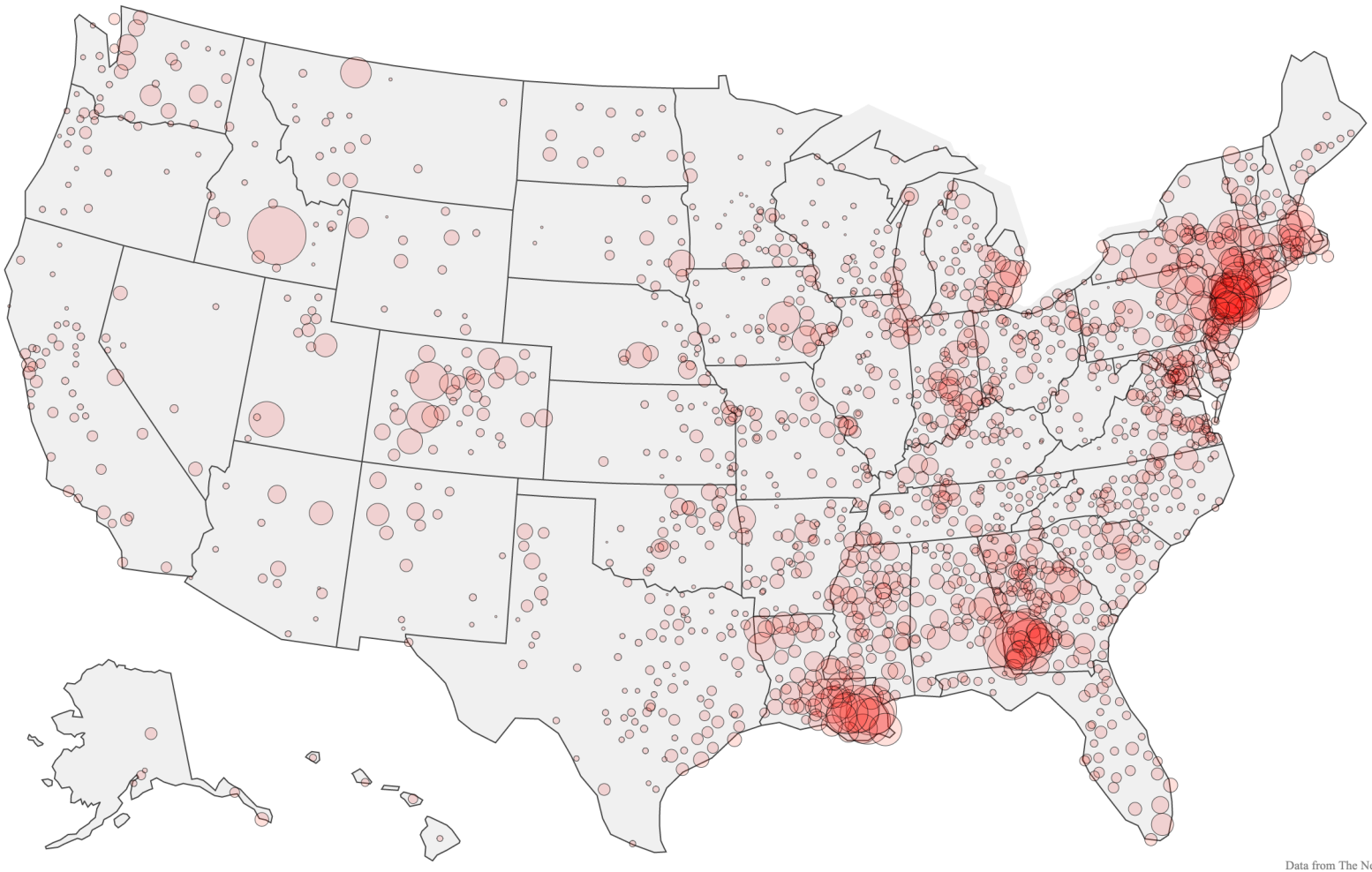
This is illustrated in the following simula ior becomes concentrated on the true parameter. But as the variance of t

In [5]:
```
import os, gzip
import numpy as np
import matplotlib.pyplot as p
```

In [*]:
```
%matplotlib qt
from scipy.special import gam
from scipy import random
from scipy.stats import beta

theta = np.linspace(0,1,num=5
fig = plt.figure(1)
plt.ion()

# The following are the param                    the
# variance of the prior decre                    required for
# the posterior to be centere

scale = 100
a0 = scale*1
b0 = scale*1

sample_size = 100
```



Covid Cases per 100,000 people (in April 2020)



Hierarchical models

We are interested in modeling the infection rates across counties

In a Bayesian hierarchical model, we tie together our inferences across regions

In each county c we test n_c people (at random) and observe how many people Y_c have Covid-19.

$Y_c \sim \text{Binomial}(n_c, \theta_c)$. This is equivalent to n_c coin flips (Bernoulli) each with probability of heads θ_c.

A simple hierarchical model

For each county c,

$$\theta_c \sim F$$
$$Y_c \,|\, \theta_c \sim \text{Binomial}(n_c, \theta_c)$$

This ties together the parameters and makes better use of the data

A simple hierarchical model

A simple transformation makes the use of Gaussians more appropriate.

Transform by $\psi_c \equiv \log\left(\frac{\theta_c}{1-\theta_c}\right)$ to use Gaussian approximation:

$$\psi_c \sim N(\mu, \tau^2)$$
$$Z_c \,|\, \psi_c \sim N(\psi_c, \sigma_c^2)$$

- Posterior distribution may not be written down explicitly
- But we can sample from it
- This approximates the posterior as a mixture of Gaussians

A run from April 2020

File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code Dashboard View:

In [1]:
```
import covid19 as cvd
import covid19_predict as cvd_predict
```

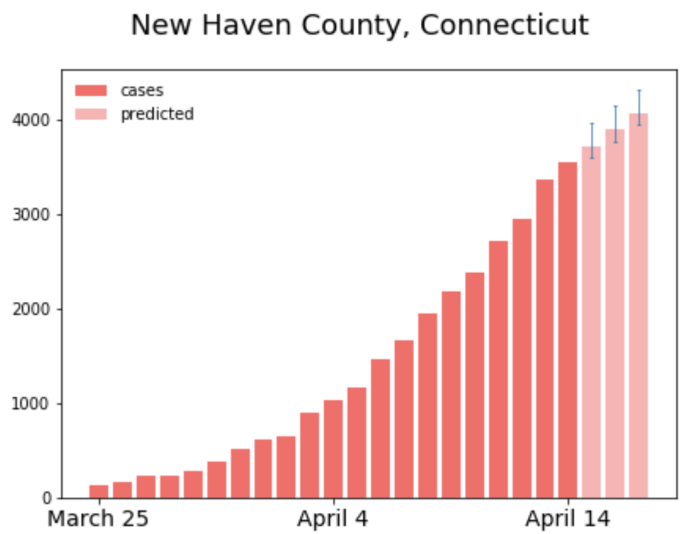
```
covid19: Most recent NY Times data: Tuesday April 14, 2020
covid19_predict: initializing for simulation
covid19_predict: running Gibbs sampler: n=3193, B=10000
covid19_predict: making predictions
```

In [2]:
```
cvd_predict.df[cvd_predict.df['county']=='New Haven']
```

Out[2]:

	date	county	state	cases	deaths	population	cases_per_100k	delta	delta_bar	delta_95	delta_05	cases_predicted	cases_95_credible
83	2020-04-14	New Haven	Connecticut	3543	151	854757	414.5	21.6	20.36	28.79	14.42	3716	3789

In [3]:
```
_ = cvd_predict.plot_predictions_for_addr('New Haven, CT', show=True)
```



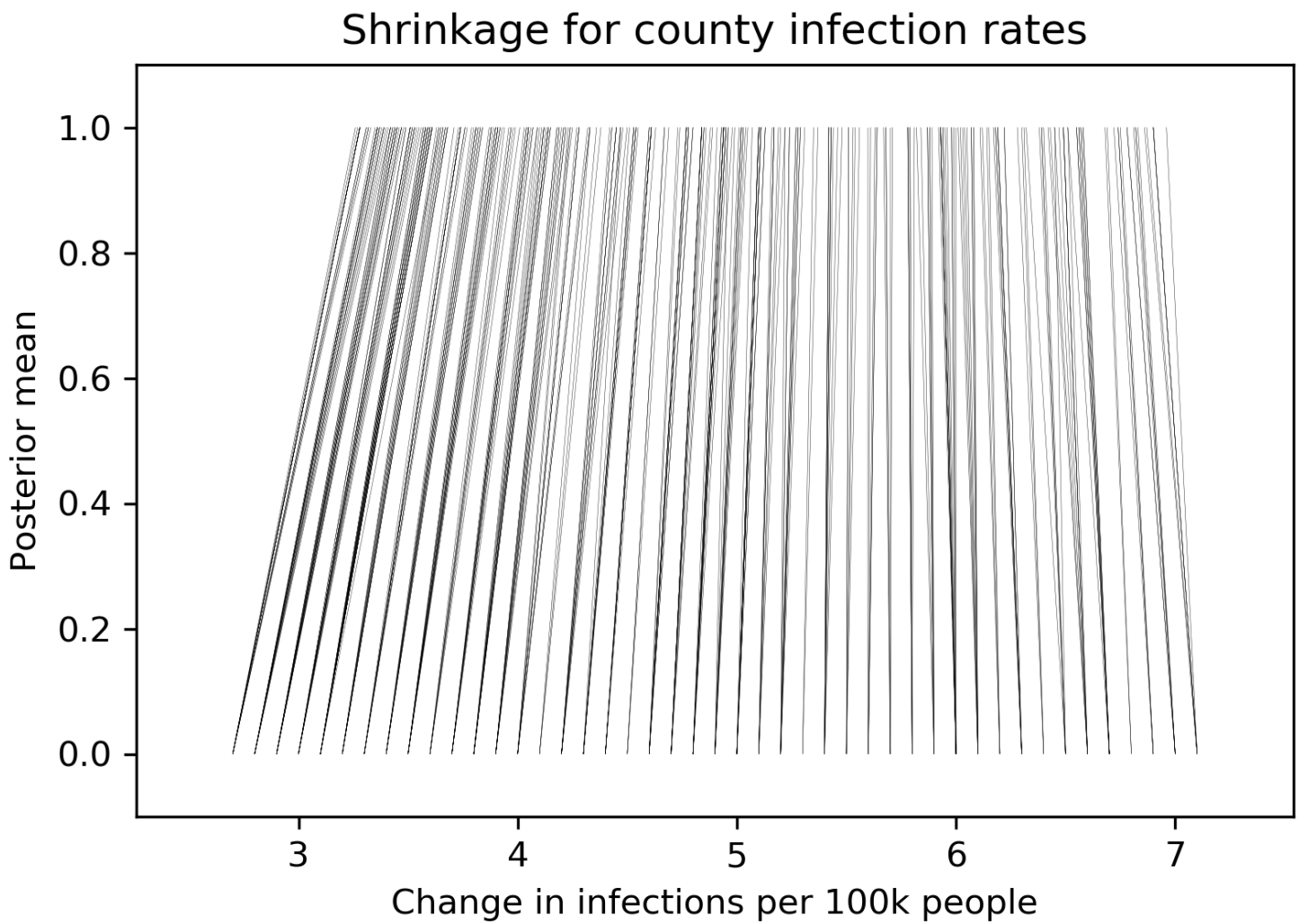
Shrinkage

Tying the parameters together results in the small probabilities being increased and the larger probabilities being decreased.

This is called “Shrinkage.” The posterior estimates are closer together than the raw frequencies.

This is different than but conceptually related to shrinkage from regularization

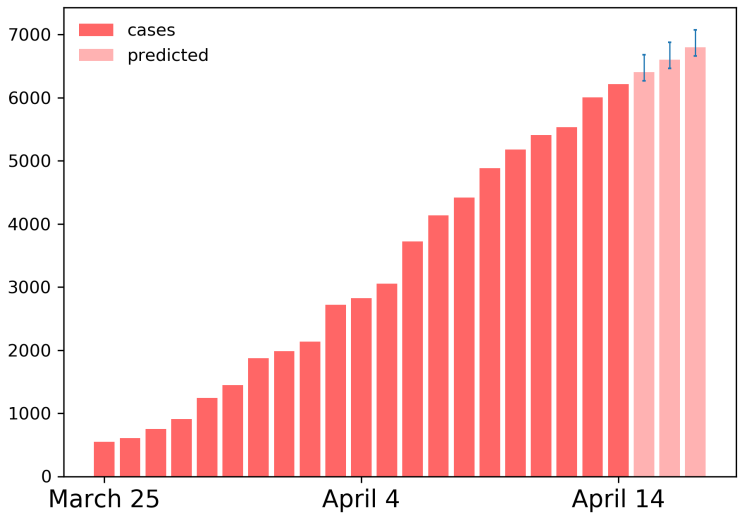
Shrinkage



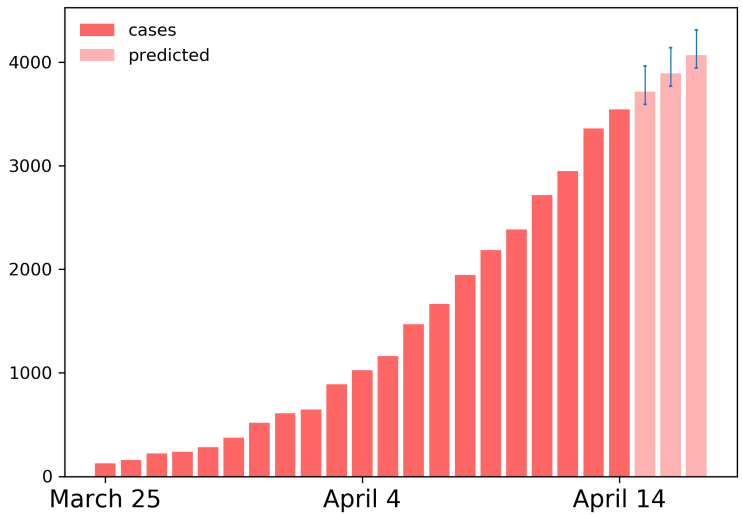
This is different than but conceptually related to shrinkage from regularization

Examples

Fairfield County, Connecticut



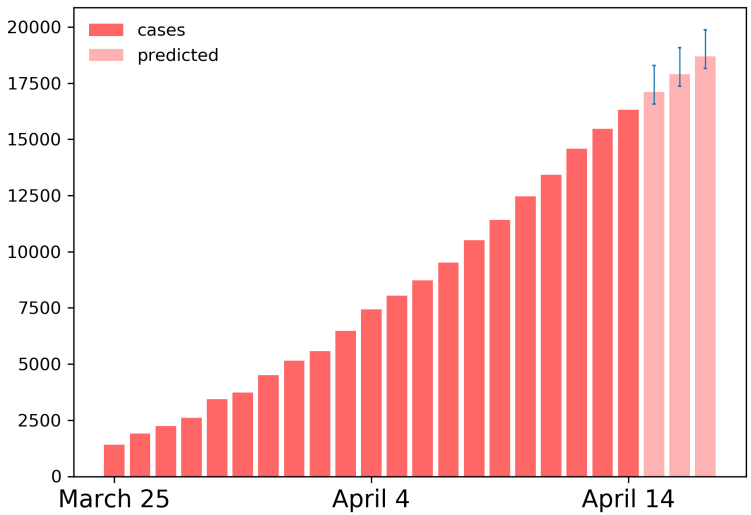
New Haven County, Connecticut



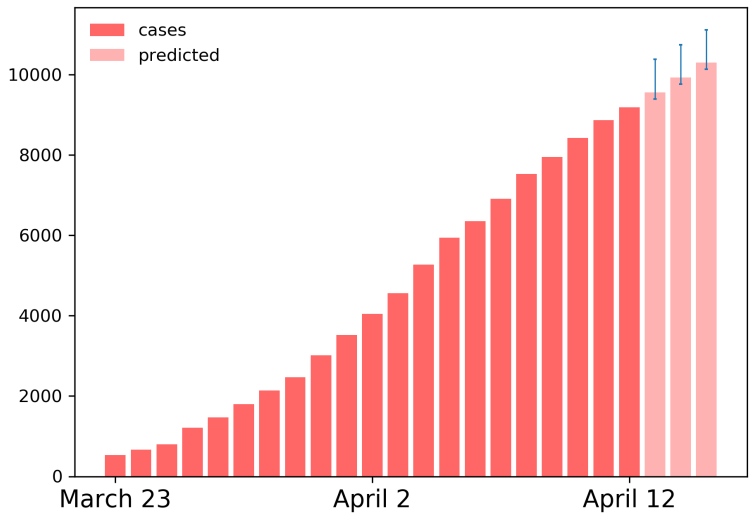
The “credible intervals” trap 95% of the sampled parameters under the simulation.

Examples

Cook County, Illinois



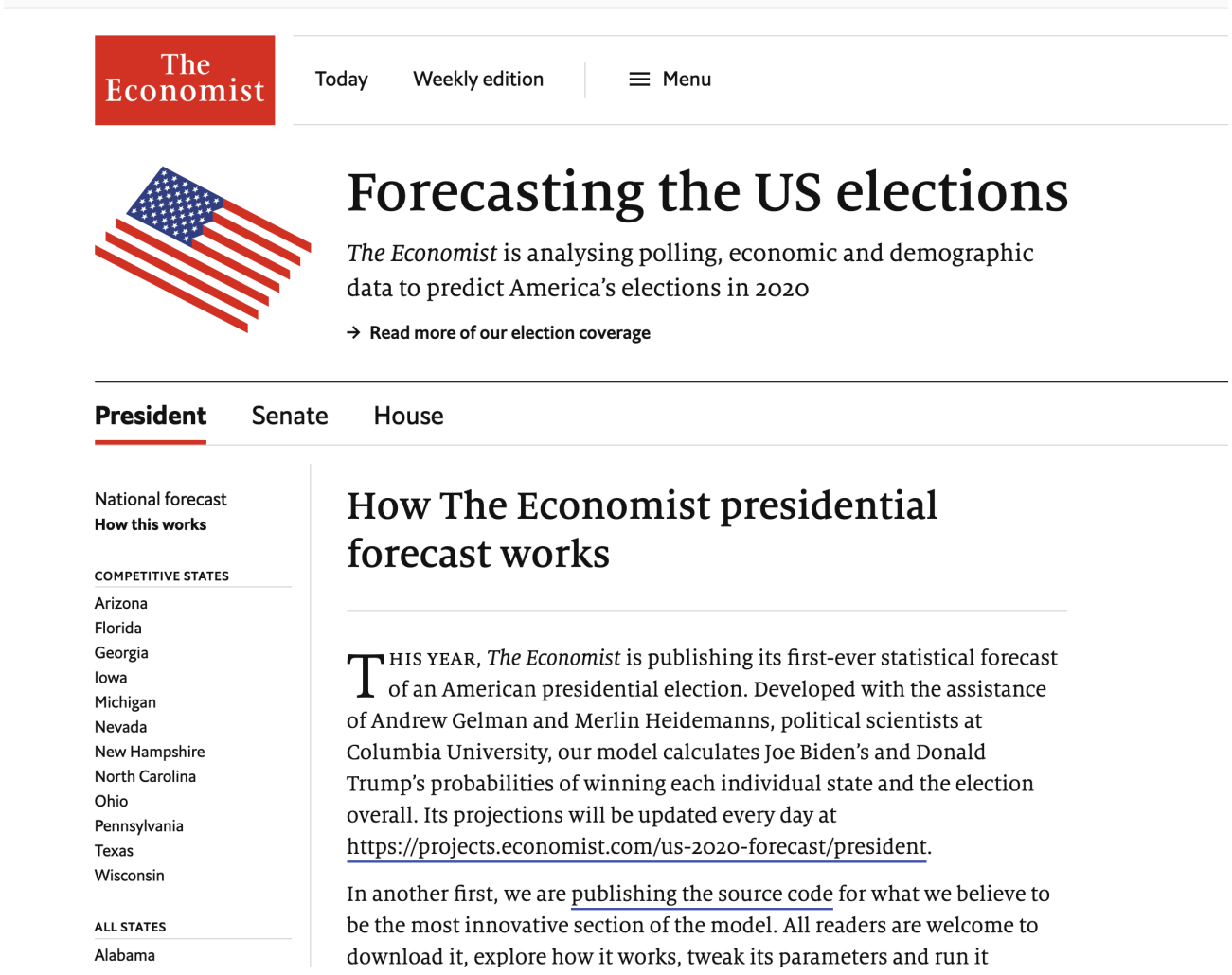
Los Angeles County, California



The “credible intervals” trap 95% of the sampled parameters under the simulation.

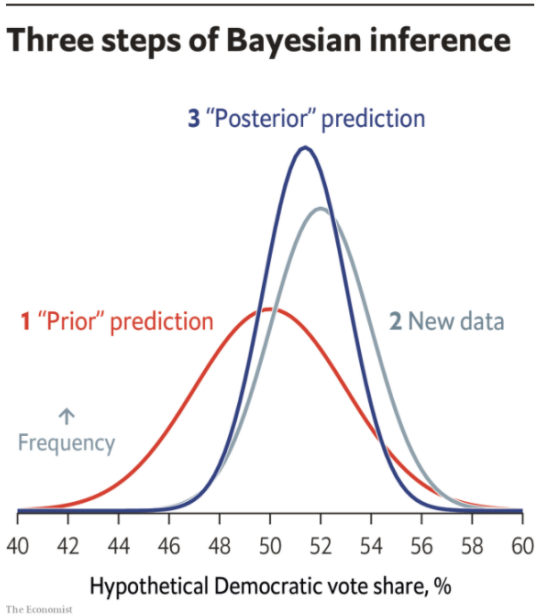
Another Example: Election Forecasting

https://projects.economist.com/us-2020-forecast/president/how-this-works



Another Example: Election Forecasting

https://projects.economist.com/us-2020-forecast/president/how-this-works



Back to Bayes-ics

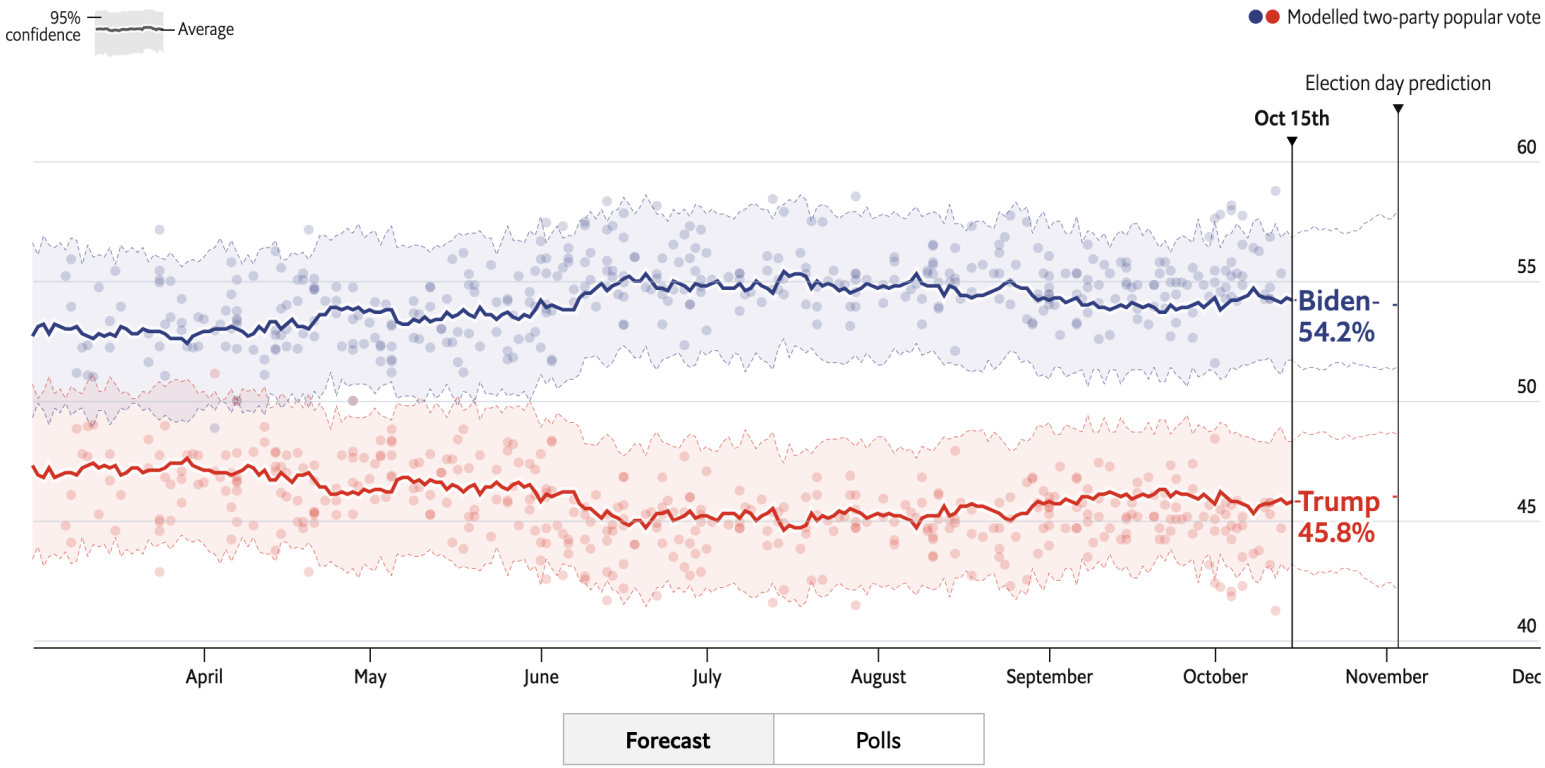
Readers acquainted with the workings of similar forecasting models may be surprised that the phrase "state polls" has not yet entered the equation. This exclusion is by design. Our model follows a logical structure first developed by Thomas Bayes, an 18th-century reverend whose ideas have shaped a large and growing family of statistical techniques. His approach works in two stages. First, before conducting a study, researchers explicitly state what they believe to be true, and how confident they are in that belief. This is called a "prior". Next, after acquiring data, they update this prior to reflect the new information—gaining more confidence if it confirms the prior, and generally becoming more uncertain if it refutes the prior (though not if the new numbers are so definitive that leave little room for doubt). In this framework, the expected distribution of potential vote shares in each state derived above is the prior, and state polls that trickle in during the course of the campaign are the new data. The result—a "posterior", in Bayesian lingo—is our forecast.

Another Example: Election Forecasting

https://projects.economist.com/us-2020-forecast/president/how-this-works

Modelled popular vote on each day

The model first averages the polls, weighting them by their sample sizes and correcting them for tendencies to overestimate support for one party. It then combines this average with our forecast based on non-polling data, pulling vote shares on each day slightly towards the final election-day projection.



Reading

- See notes on Bayesian inference on iML site (`bayes-notes.pdf`)
- These are more advanced, from S&DS 365.
- Not necessary to understand everything—but you should be able to do some of the basic "coin flipping" calculations
- We'll build on this when discussing topic models

Summary

- Mixtures are latent variable models
- The mixing weight encodes a hidden variable
- Computing with mixtures uses basic probabilistic reasoning
- Bayesian inference is popular in ML
- In a Bayesian approach, the parameters are random, and the data are fixed.
- Bayesian hierarchical models tie together data using latent variables