Published in Towards Data Science

Will Koehrsen Follow

Nov 28, 2018 · 12 min read

(Source)

Estimating Probabilities with Bayesian Modeling in Python

A simple application of Probabilistic Programming with PyMC3 in Python

It started, as the best projects always do, with a few tweets:

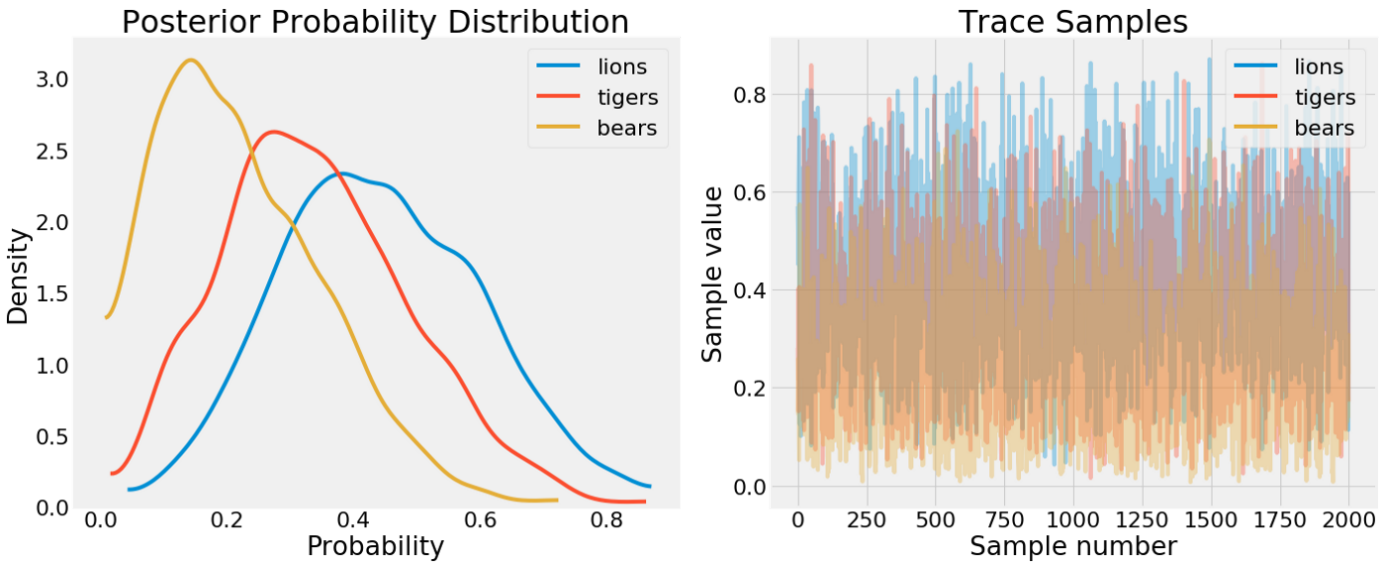
Twitter is a great resource for data science!

This may seem like a simple problem — the prevalences are simply the same as the observed data (50% lions, 33% tigers and 17% bears) right? If you believe observations we make are a perfect representation of the underlying truth, then yes, this problem could

First, how can we be sure this single trip to the preserve was indicative of all trips? What if we went during the winter when the bears were hibernating? We need to include *uncertainty* in our estimate considering the limited data. Second, how can we incorporate *prior* beliefs about the situation into this estimate? If we have heard from a friend the preserve has an equal number of each animal, then surely this should play some role in our estimate.

> *Fortunately, there is a solution that allows to express uncertainty* and *incorporate prior information into our estimate: Bayesian Inference.*

In this article, we'll explore the problem of estimating probabilities from data in a Bayesian framework, along the way learning about probability distributions, Bayesian Inference, and basic probabilistic programming with PyMC3. The complete code is available as a Jupyter Notebook on GitHub.



PDF and trace values from PyMC3

Background: Concepts

Often, especially in statistics, I find the theory behind a solution *more confusing* than

than reading endless equations. Therefore, when I approached this problem, I studied just enough of the ideas to code a solution, and only *after* did I dig back into the concepts.

This reflects my general *top-down* approach to learning new topics. Instead of starting with the fundamentals — which are usually tedious and difficult to grasp — find out how to implement an idea so you know *why it's useful* and then go back to the formalisms. So, if you feel yourself getting frustrated with the theory, move on to the solution (starting with the Inference section below), and then come back to the concepts if you're still interested.

(This top-down philosophy is exemplified in the excellent fast.ai courses on deep learning. These courses, besides effectively teaching neural networks, have been influential in my approach to learning new techniques.)

Bayesian Model

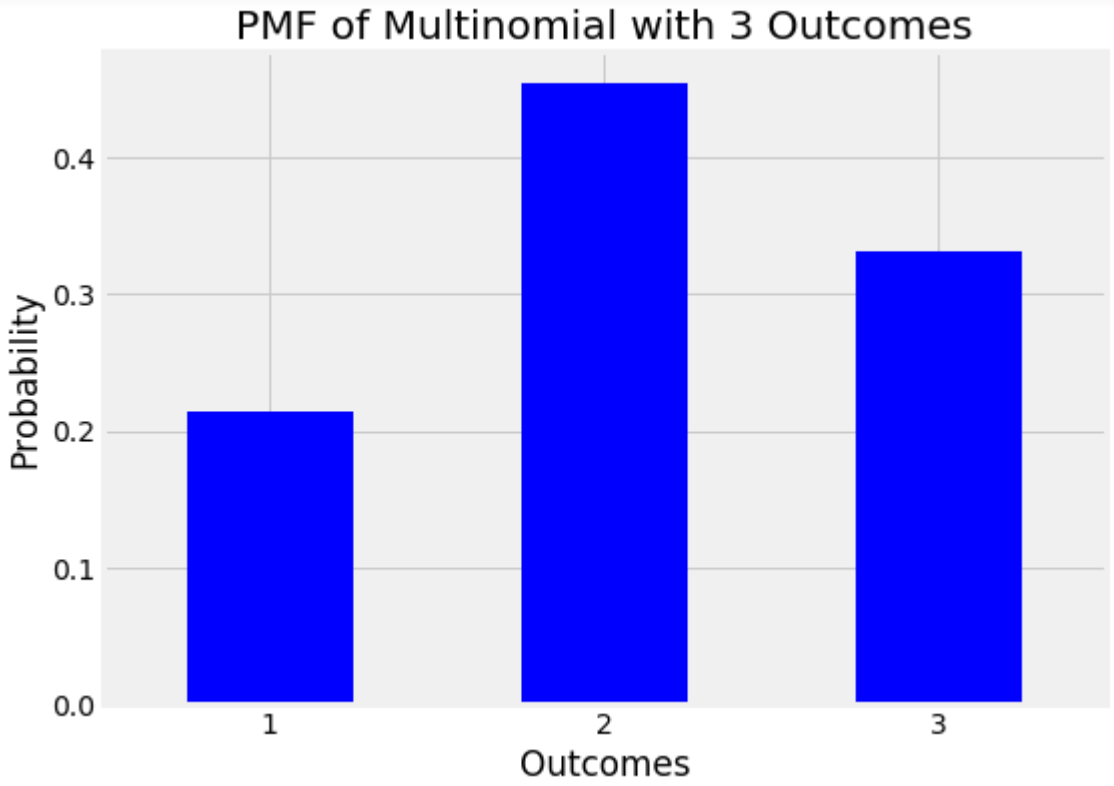
Since we want to solve this problem with Bayesian methods, we need to construct a model of the situation. The basic set-up is we have a series of observations: 3 tigers, 2 lions, and 1 bear, and from this data, we want to estimate the prevalence of each species at the wildlife preserve. That is, we are looking for the posterior probability of seeing each species given the data.

Before we begin we want to establish our assumptions:

- Treat each observation of one species as an independent trial.
- Our initial (prior) belief is each species is equally represented.

The overall system, where we have 3 **discrete** choices (species) each with an unknown probability and 6 total observations is a **multinomial distribution**. The multinomial distribution is the extension of the binomial distribution to the case where there are more than 2 outcomes. A simple application of a multinomial is 5 rolls of a dice each of which





Probability Mass Function (PMF) of a multinomial with 3 outcomes

A Multinomial distribution is characterized by k, the number of outcomes, n, the number of trials, and **p**, a vector of probabilities for each of the outcomes. For this problem, **p** is our ultimate objective: we want to figure out the probability of seeing each species from the observed data. In Bayesian statistics, the parameter vector for a multinomial is drawn from a **Dirichlet Distribution,** which forms the prior distribution for the parameter.

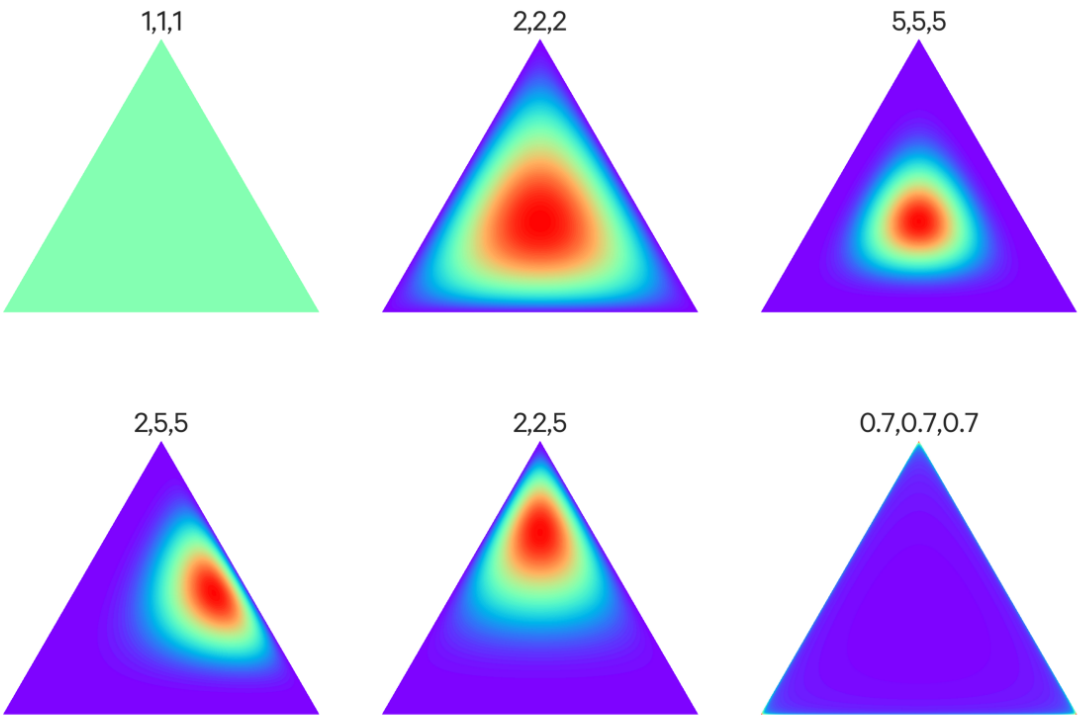
The Dirichlet Distribution, in turn, is characterized by, k, the number of outcomes, and **alpha**, a vector of positive real values called the concentration parameter. This is called a **hyperparameter** because it is a *parameter of the prior*. (This chain can keep going: if **alpha** comes from another distribution then this is a *hyperprior* which could have *its own parameters* called *hyperyhyperparameters*!). We'll stop our model at this level by explicitly setting the values of alpha, which has one entry for each outcome.

Hyperparameters and Prior Beliefs

The best way to think of the Dirichlet parameter vector is as pseudocounts, observations

Conversely, if we expected to see more bears, we could use a hyperparameter vector like [1, 1, 2] (where the ordering is [lions, tigers, bears]. The exact value of the pseudocounts reflects the level of confidence we have in our prior beliefs. Larger pseudocounts will have a greater effect on the posterior estimate while smaller values will have a smaller effect and will let the data dominate the posterior. We'll see this when we get into inference, but for now, remember that the hyperparameter vector is pseudocounts, which in turn, represent our prior belief.

A Dirichlet distribution with 3 outcomes is shown below with different values of the hyperparameter vector. Color indicates the concentration weighting.



Effect of the hyperparameter vector alpha on the Dirichlet Distribution (source).

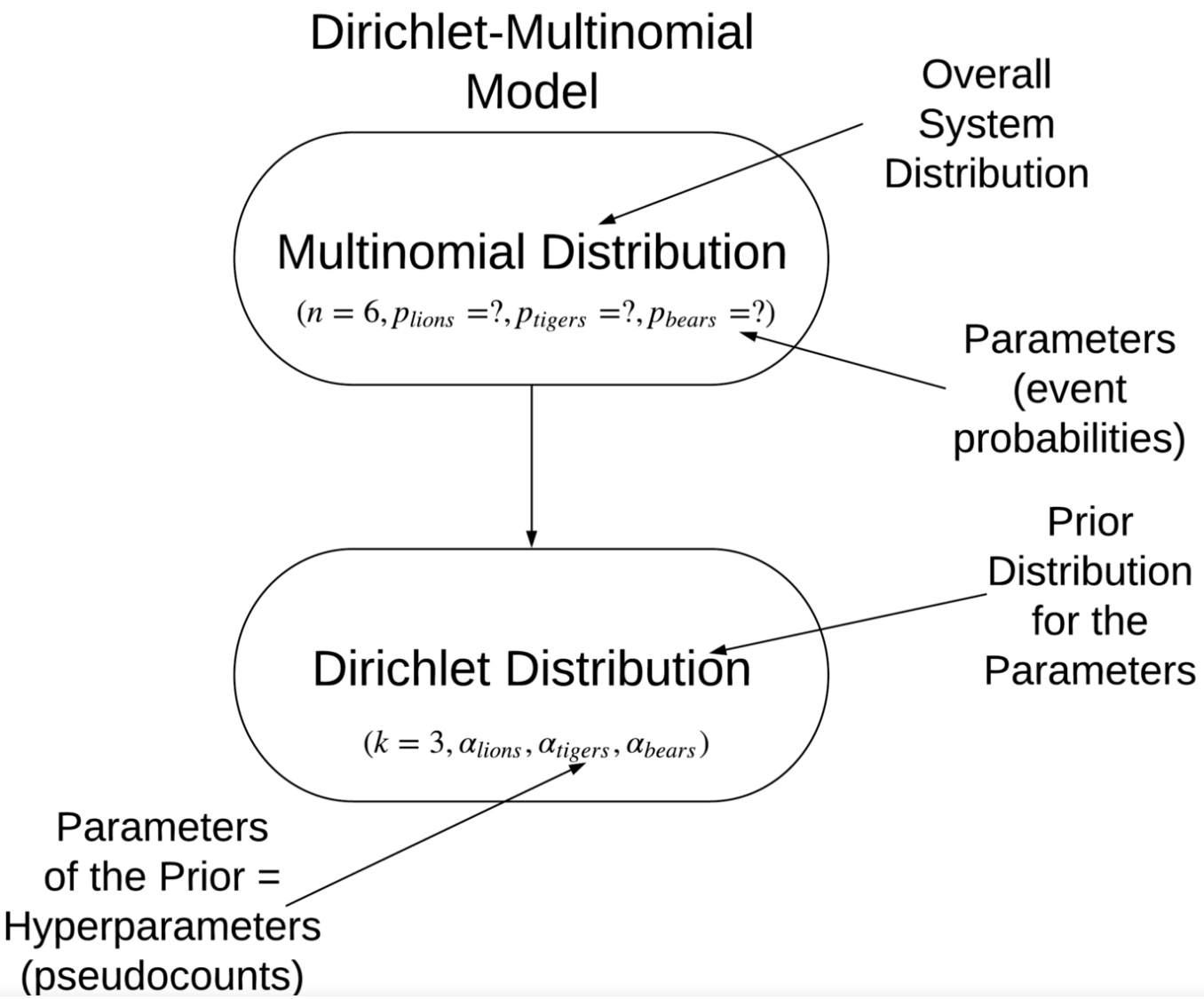
There's a lot more detail we don't need to get into here, but if you're still curious, see some

Our ultimate goal is to estimate the posterior distribution for the probability of observing each species, **p**, conditioned on the data and hyperparameters:

$$(\mathbf{p} \mid \mathbb{X}, \boldsymbol{\alpha})$$

The posterior distribution of the parameter is our objective. X is observations and alpha is hyperparameters.

Our final model, consisting of a multinomial distribution with Dirichlet priors is called a Dirichlet-Multinomial and is visualized below:



A summary of the problem specifics is below:

Model Specifics

Name	Symbol	Value	Constraints
trials (number of events)	n	6	$n > 0$
parameters of *multinomial* (event probabilities)	$\mathbf{p}$	$p_{lions} = ?$ $p_{tigers} = ?$ $p_{bears} = ?$	$\Sigma p_i = 1$
support (data)	$\mathbf{c}$	$c_{lions} = 3$ $c_{tigers} = 2$ $c_{bears} = 1$	$\Sigma c_i = n$
parameters of *Dirichlet* hyperparameters (pseudocounts)	α	$\alpha_{lions} = 1$ $\alpha_{tigers} = 1$ $\alpha_{bears} = 1$	$\alpha > 0$
number of outcomes	k	3	$k \geq 2$

Solving for Posterior of event

Model specifics

If you still want more background details, here are some of the sources I relied on (the first is probably the most valuable):

Sources:

1. Bayesian Inference for Dirichlet-Multinomials
2. Categorical Data / Multinomial Distribution
3. Dirichlet-Multinomial Wikipedia Article
4. Multinomial Distribution Wikipedia Article
5. Alpha in the Dirichlet Distribution
6. Dirichlet Distribution Wikipedia Article
7. Hyperparameter Wikipedia Article
8. Deriving the MAP estimate for Dirichlet-Multinomials

There are also other ways to approach this problem; see here for Allen Downey's solution which yields similar results.

Inference: Making Estimates from Data

Now that we have the model of the problem, we can solve for the posteriors using Bayesian methods. Inference in statistics is the process of estimating (inferring) the unknown parameters of a probability distribution from data. Our unknown parameters are the prevalence of each species while the data is our single set of observations from the wildlife preserve. Our goal is to find the posterior distribution of the probability of seeing each species.



intractable, and, as the number of samples increases, the estimated posterior converges to the true posterior.

The result of MCMC is not just one number for our answer, but rather a range of samples that lets us quantify our uncertainty especially with limited data. We'll see how to perform Bayesian inference in Python shortly, but if we do want a single estimate, we can use the *Expected Value* of the distribution.

Expected Value

The Expected Value is the mean of the posterior distribution. For a Dirichlet-Multinomial, it can be analytically expressed:

$$\mathrm{E}[p_i \mid \mathbb{X}, \boldsymbol{\alpha}] = \frac{c_i + \alpha_i}{N + \sum_k \alpha_k}$$

Expected value of a Multinomial with Dirichlet priors.

Once we start plugging in numbers, this becomes easy to solve. N is the number of trials, 6, c_i is the *observed count* for each category, and alpha_i is the *pseudocount* (hyperparameter) for each category. Setting all alphas equal to 1, the expected species probabilities can be calculated:

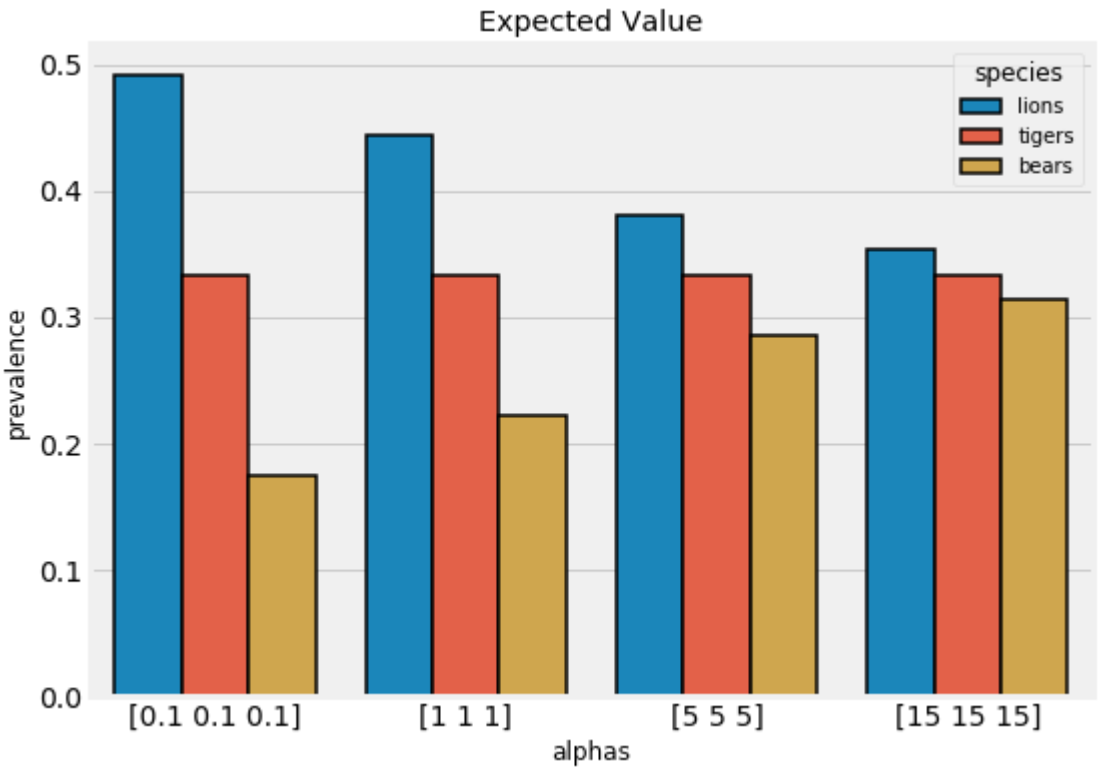
```
species = ['lions', 'tigers', 'bears']
# Observations
c = np.array([3, 2, 1])
#Pseudocounts
alphas = np.array([1, 1, 1])

expected = (alphas + c) / (c.sum() + alphas.sum())
```

```
Species: lions    Prevalence: 44.44%.
Species: tigers   Prevalence: 33.33%.
Species: bears    Prevalence: 22.22%.
```



We can adjust our level of confidence in this prior belief by increasing the magnitude of the pseudocounts. This forces the expected values closer to our initial belief that the prevalence of each species is equal. The expected values for several different hyperparameters are shown below:

lions	tigers	bears	alphas
0.492063	0.333333	0.174603	[0.1 0.1 0.1]
0.444444	0.333333	0.222222	[1 1 1]
0.380952	0.333333	0.285714	[5 5 5]
0.352941	0.333333	0.313725	[15 15 15]



Expected values for different pseudocounts.

Our choice of hyperparameters has a large effect. If we are more confident in our belief,

While this result provides a point estimate, it's misleading because it does not express any uncertainty. We only went to the wildlife preserve once, so there should be a large amount of uncertainty in these estimates. With Bayesian Inference, we can get both point estimates and the uncertainty.

Bayesian Inference in Python with PyMC3

To get a range of estimates, we use Bayesian inference by constructing a model of the situation and then *sampling from the posterior to approximate the posterior.* This is implemented through Markov Chain Monte Carlo (or a more efficient variant called the No-U-Turn Sampler) in PyMC3. Compared to the theory behind the model, setting it up in code is simple:

```
import pymc3 as pm
import numpy as np

alphas = np.array([1, 1, 1])
c = np.array([3, 2, 1])

# Create model
with pm.Model() as model:
    # Parameters of the Multinomial are from a Dirichlet
    parameters = pm.Dirichlet('parameters', a=alphas, shape=3)
    # Observed data is from a Multinomial distribution
    observed_data = pm.Multinomial(
        'observed_data', n=6, p=parameters, shape=3, observed=c)
```

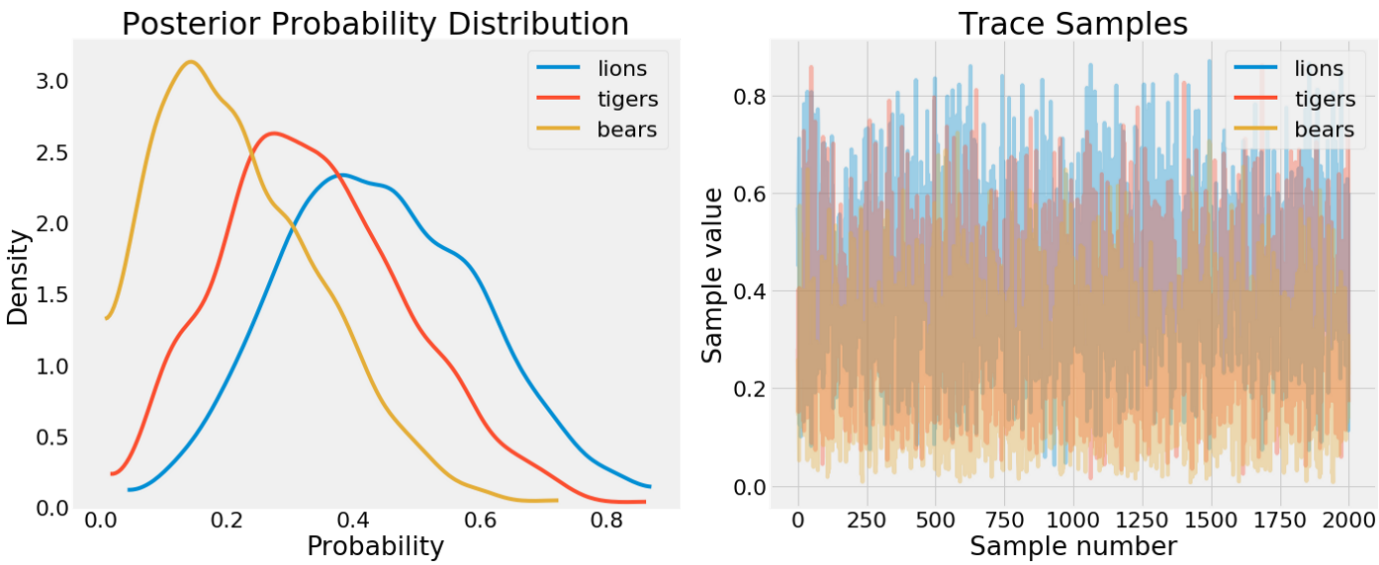
wildlife_preserve_model.py hosted with ❤ by GitHub view raw

Then, we can sample from the posterior:

```
with model:
    # Sample from the posterior
```

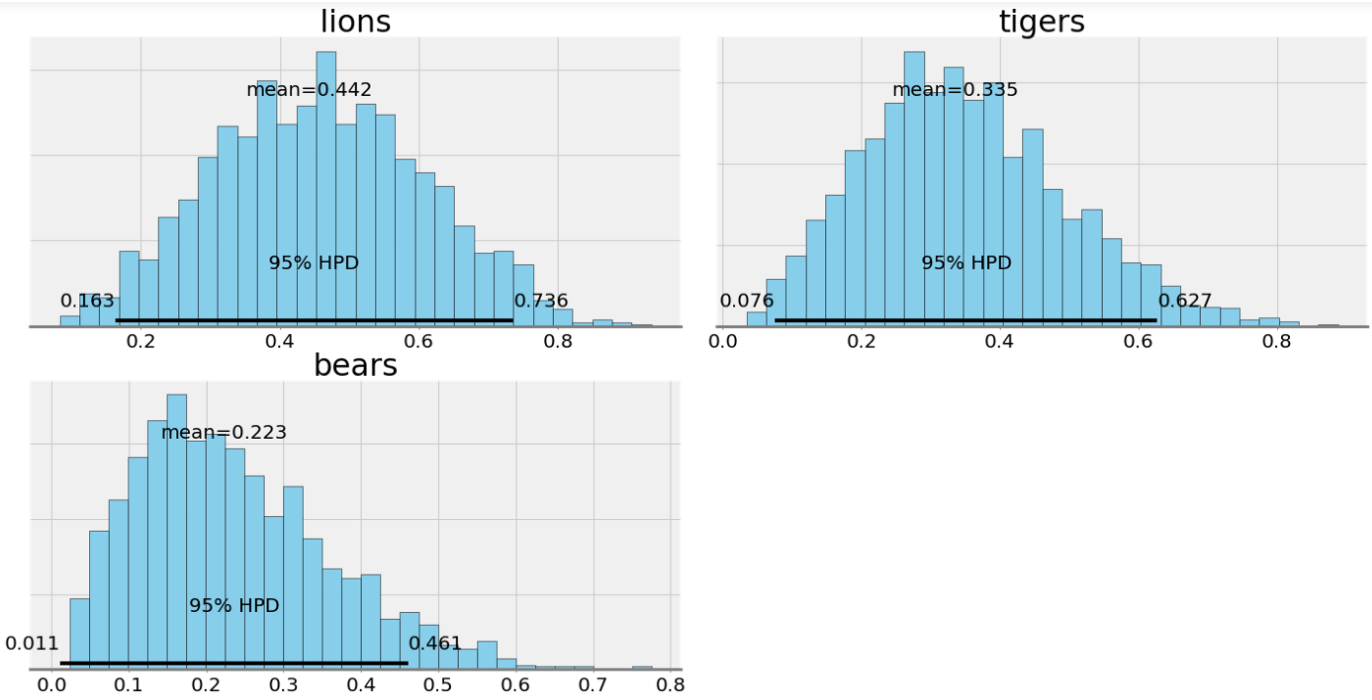
This code draws 1000 samples from the posterior in 2 different chains (with 500 samples for tuning that are discarded). We are left with a `trace` which contains all of the samples drawn during the run. We use this trace to estimate the posterior distribution.

PyMC3 has many methods for inspecting the trace such as `pm.traceplot`:



PDF and trace of samples.

On the left we have a kernel density estimate for the sampled parameters — a PDF of the event probabilities. On the right, we have the complete samples drawn for each free parameter in the model. We can see from the KDE that p_bears<p_tigers<p_lions as expected but there is some uncertainty. A better way to view this uncertainty is through `pm.posterior_plot`:



Posterior plots from PyMC3

Here are histograms indicating the number of times each probability was sampled from the posterior. We have a point estimate for the probabilities — the mean — as well as the Bayesian equivalent of the confidence interval — the 95% highest probability density (also known as a credible interval). We see an extreme level of uncertainty in these estimates, as befits the limited data.

To quantify the level of uncertainty we can get a dataframe of the results:

	mean	sd	mc_error	hpd_2.5	hpd_97.5
lions	0.442073	0.151600	0.003309	0.163009	0.736219
tigers	0.335250	0.146021	0.003246	0.075918	0.627365
bears	0.222677	0.126508	0.002890	0.010689	0.461220

This shows the best estimate (mean) for the prevalence but also that the 95% credible interval is very large. We can only nail down the prevalence of lions to between 16.3%

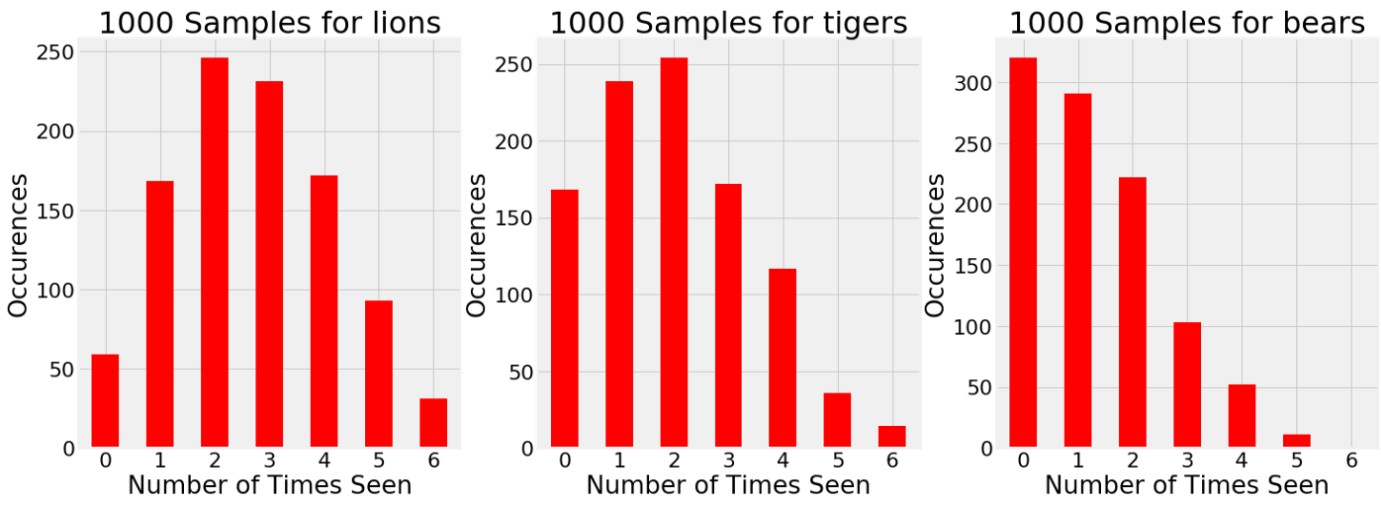
an estimate from data we have to show this uncertainty. For this problem, no one is going to be hurt if we get the percentage of bears at the wildlife preserve incorrect, but what if we were doing a similar method with medical data and inferring disease probability?

Sampling from the Posterior

Once we have the trace, we can draw samples from the posterior to simulate additional trips to the preserve. For example, let's consider going 1000 more times. How many of each species can we expect to see on each trip?

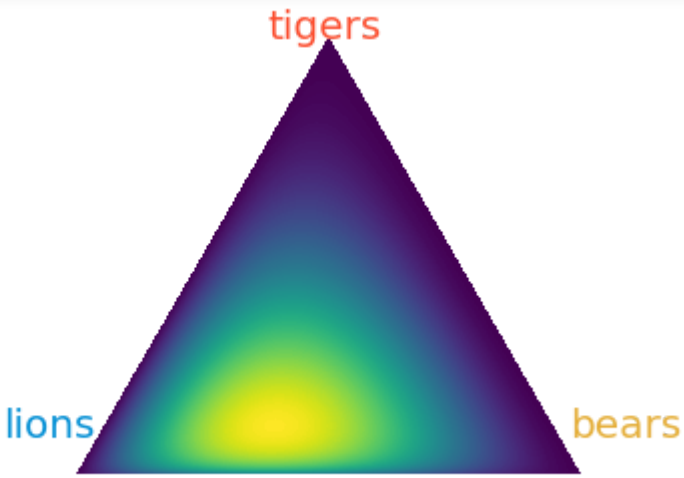
```
# Sample from the posterior
with model:
    samples = pm.sample_ppc(trace, samples = 1000)
```

sample_ppc.py hosted with ❤ by GitHub view raw



1000 samples drawn from the estimated posterior.

Based on the evidence, there are times when we go to the preserve and see 5 bears and 1 tiger! Granted, this is not very likely, graphs such as these show the entire *range of possible outcomes* instead of only one. Our single trip to the preserve was just one outcome: 1000



Dirichlet distribution after sampling.

Incorporating Additional Information

What happens when we go 4 times to the preserve and want to incorporate additional observations in our model? In PyMC3, this is simple:

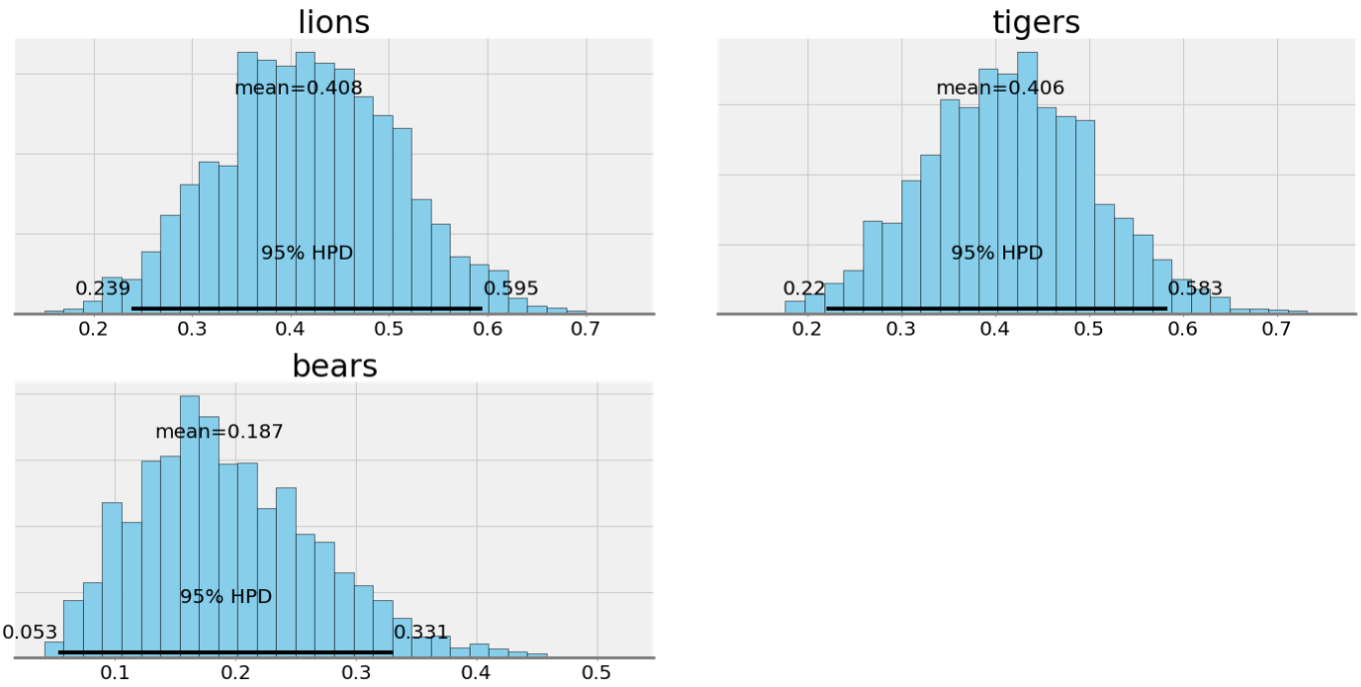
```
# Observations from multiple trips
c = np.array([[3, 2, 1],
              [2, 3, 1],
              [3, 2, 1],
              [2, 3, 1]])

with pm.Model() as model:
    # Parameters are a dirichlet distribution
    parameters = pm.Dirichlet('parameters', a=alphas, shape=3)
    # Observed data is a multinomial distribution
    observed_data = pm.Multinomial(
        'observed_data', n=6, p=parameters, shape=3, observed=c)

    trace = pm.sample(draws=1000, chains=2, tune=500, discard_tuned_samples=True)
```

more_data_sample.py hosted with ❤ by GitHub view raw

The uncertainty in the posterior should be reduced with a greater number of observations, and indeed, that is what we see both quantitatively and visually. Intuitively

	mean	sd	mc_error	hpd_2.5	hpd_97.5
lions	0.407799	0.091164	0.001813	0.238817	0.595130
tigers	0.405678	0.092595	0.002054	0.220425	0.583333
bears	0.186523	0.075623	0.001722	0.052582	0.331158

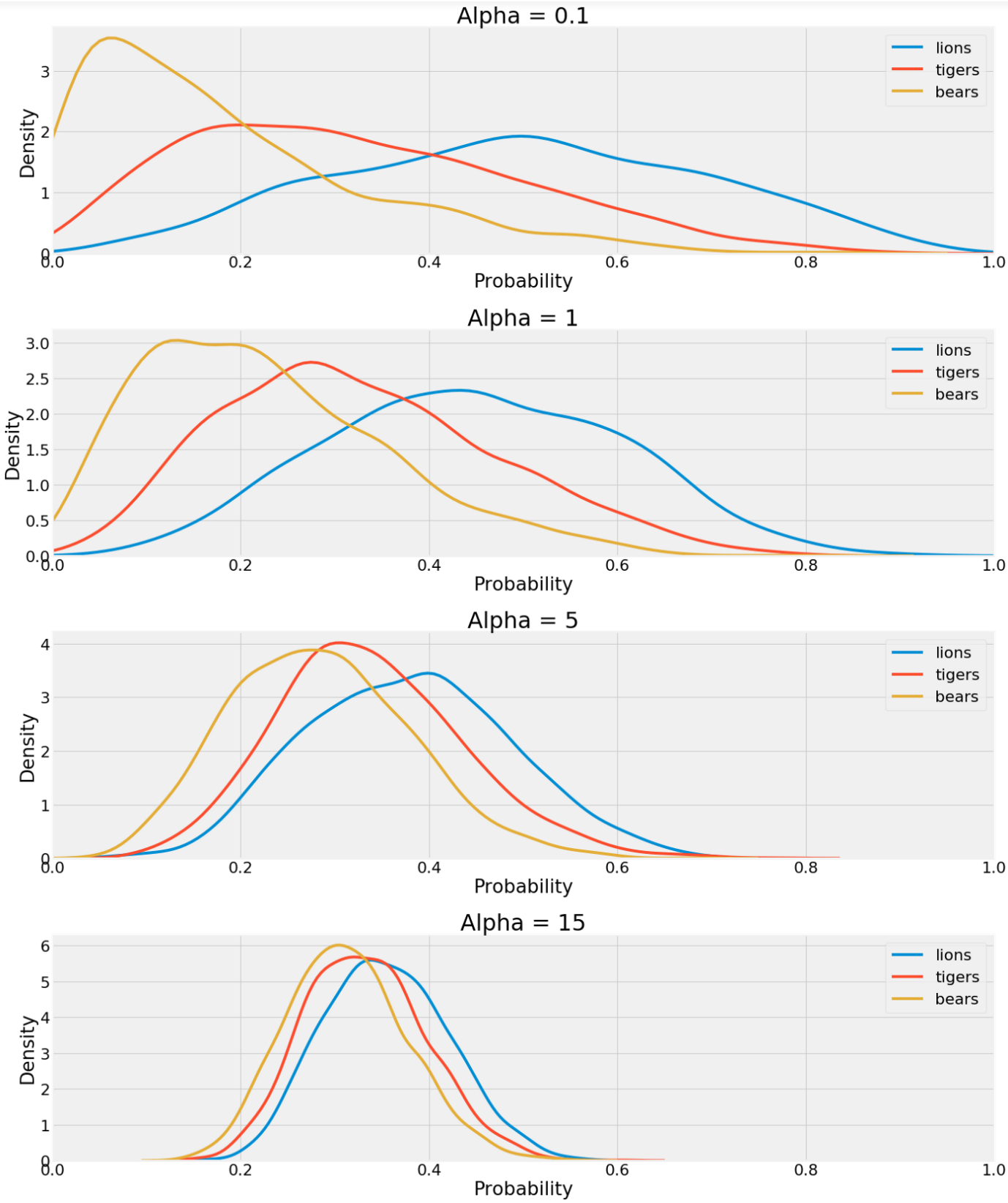
Posterior with More Observations



Posteriors with more data

Increasing and Decreasing Confidence in Prior Beliefs

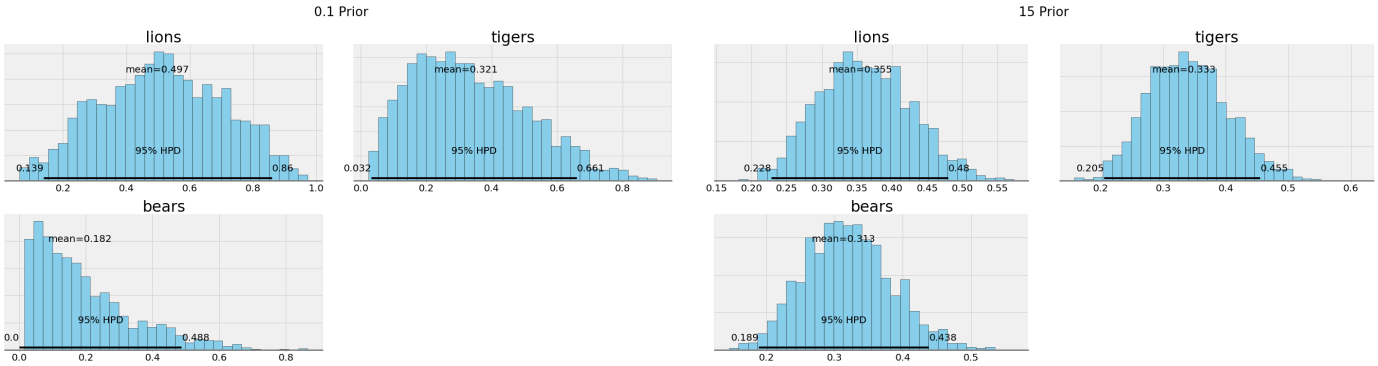
Earlier we discussed how the hyperparameters can be thought of as pseudocounts that represent our prior belief. If we set all the values of alpha equal to 1, we get the results we've seen so far. What about if we decrease or increase our confidence in our initial theory that the prevalence is equal? To do so, all we have to do is alter the alpha vector. Then, we sample from the posterior again (using the original observations) and inspect the results.



The hyperparameters have a large influence on the outcome! A lower value means the

on one another. We'd need a lot of data to overcome our strong hyperparameters in the last case.

We can compare the posterior plots with alpha = 0.1 and alpha = 15:



Ultimately, our choice of the hyperparameters depends on our confidence in our belief. If we have a good reason to think the prevalence of species is equal, then we should make the hyperparameters have a greater weight. If we want to let the data speak, then we can lower the effect of the hyperparameters.

Conclusions

Well, what should our final answer be to the question of prevalences? If we are good Bayesians, then we can present a point estimate, but only with attached uncertainty (95% credible intervals):

- **Lions: 44.5% (16.9% — 75.8%)**
- **Tigers: 32.7% (6.7% — 60.5%)**
- **Bears: 22.7% (1.7% — 50.0%)**

And our estimate that the next observation is a bear? Based on the posterior sampling, about 23%. While these results may not be satisfying to people who want a simple answer

The benefits of Bayesian Inference are we can incorporate our prior beliefs and we get uncertainty estimates with our answers. The world is uncertain, and, as responsible data scientists, Bayesian methods provide us with a framework for dealing with uncertainty.

Furthermore, as we get more data, our answers become more accurate. As with many aspects of Bayesian Inference, this is in line with our intuitions and how we naturally go about the world, becoming less wrong with additional information. Ultimately, Bayesian statistics is enjoyable and useful because it is statistics that finally makes sense.

As always, I welcome feedback and constructive criticism. I can be reached on Twitter @koehrsen_will or through my personal website willk.online.