# Estimating Probabilities with Bayesian Inference

Maimoona Khilji

Institute of Management Science

Course Code: Pattern Recognition

Mian Ibad Ali Shah

20th October, 2021

# Estimating Probabilities with Bayesian Inference

**Problem:**

Suppose we visit a wild animal preserve where we know that the only animals are lions and tigers and bears, but we don't know how many of each there are. During the tour, we see 3 lions, 2 tigers, and 1 bear. Assuming that every animal had an equal chance to appear in our sample, estimate the prevalence of each species. What is the probability that the next animal we see is a bear?

**Goal:**

Our goal is to:

- Estimate the prevalence of each species.
- Determine the probability that the next animal is a bear.

## Important Terms

### Multinomial Distribution

- It is a generalization of Binomial Distribution.
- It is probability of the outcomes from multiple (multinomial) experiments.
- The basic difference between binomial and multinomial is that Binomial experiment can have two outcomes (success or failure) while multinomial can have multiple outcomes.
  For example:
  **Binomial:** Rolling a die ten times to see **how** many times you roll a specific number (like six)?
  **Multinomial:** Rolling a die ten times to see **what** number you roll? (So here the resulting outcome can be 1, 2, 3, 4, 5 or 6)

- It describes a situation with **n** independent trials, each with **k** possible outcomes.

  Here in this wild preserve problem,
  k = 3 (lion + tiger + bear)
  n = 6 (No. of lions + No. of tigers + No. of Bears)

  As
  Lions : Tigers : Bears :: 3 : 2 : 1

### Dirichlet Distribution

- The prior for a multinomial distribution in Bayesian statistics is a **Dirichlet distribution**.

**Hyperparameters and Prior Beliefs**

- The vector parameter of Dirichlet distribution is known as **hyperparameter** vector and is denoted as $\boldsymbol{\alpha}$ vector.
- This hyperparameter vector is like pseudo-count that is used to show the prior belief for the prevalence of each species.
    - Pseudo-count: an amount added to the number of observed cases in order to alter the expected probability obtained by prior knowledge.
- We can alter (increase our decrease) the effect of the priors by increasing or decreasing the values of the hyper parameter vector.
- This is useful when we have less confidence in our prior beliefs.

In this problem, we initially work with $\boldsymbol{\alpha}$ = [1, 1, 1], considering 1 for each of the specie reflecting that each of them has same occurrence.

**Solution in Python**

- Import the important libraries:

| Libraries | Usage |
|---|---|
| Pandas | provides fast, expressive, and flexible data structures |
| Numpy | advanced math function |
| Matplotlib.pyplot | Data analyzing, Numerical plotting library |
| Seaborn | a data visualization library built on top of matplotlib |
| Warnings | useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program |
| Pymc3 | Package for Bayesian statistical modeling and probabilistic machine learning |
| Util libraries | collection of small Python functions and classes which make common patterns shorter and easier |

**Import Libraries**

```
import pandas as pd
import numpy as np

# Visualizations
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
plt.rcParams['font.size'] = 22
%matplotlib inline

from matplotlib import MatplotlibDeprecationWarning

import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore', category=MatplotlibDeprecationWarning)

import pymc3 as pm

# Helper functions
from utils import draw_pdf_contours, Dirichlet, plot_points, annotate_plot, add_legend, display_probs
```

- After importing the important libraries, we will store the data (species' names, observed cases and hyper parameters) in respective lists.
  - Species = "Names of species"
  - C = "No. of observed cases for each specie"
  - Alphas = "pseudo-counts or hyperparameter for each specie"

**Problem Specifics**

We'll mainly use one version of the hyperparameters, $\alpha = [1, 1, 1]$. We'll also try some other values to see how that changes the problem. Remember, altering the hyperparameters is like chaning our confidence in our initial beliefs.

```
species = ['Lions', 'Tigers', 'Bears']

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


# pseudo-counts or hyperparameters (initially all equal)
alphas = np.array([1, 1, 1])

```

- The expected species probability can be calculated as:

$$\mathrm{E}[\mathrm{p}_i \mid \mathbb{X}, \alpha] = \frac{ci + \alpha i}{N + \sum_k \alpha_k}$$

**Expected Value**

```
display_probs(dict(zip(species, (alphas + c) / (c.sum() + alphas.sum()))))
```

```
Species: Lions    Prevalence: 44.44%.
Species: Tigers   Prevalence: 33.33%.
Species: Bears    Prevalence: 22.22%.
```

This is as expected.

```
display_probs(dict(zip(species, (4/9, 3/9, 2/9))))
```

```
Species: Lions    Prevalence: 44.44%.
Species: Tigers   Prevalence: 33.33%.
Species: Bears    Prevalence: 22.22%.
```

- This is the expected value based on our initial belief about the case where pseudo-counts are same for all. i.e. $\boldsymbol{\alpha}$ = [1,1,1]
- Now we will change the value of hyperparameter vector and will analyze the difference or alteration in the expected value.

```
values = []
for alpha_new in alpha_list:
    values.append((alpha_new + c) / (c.sum() + alpha_new.sum()))

value_df = pd.DataFrame(values, columns = species)
value_df['alphas'] = [str(x) for x in alpha_list]
value_df
```

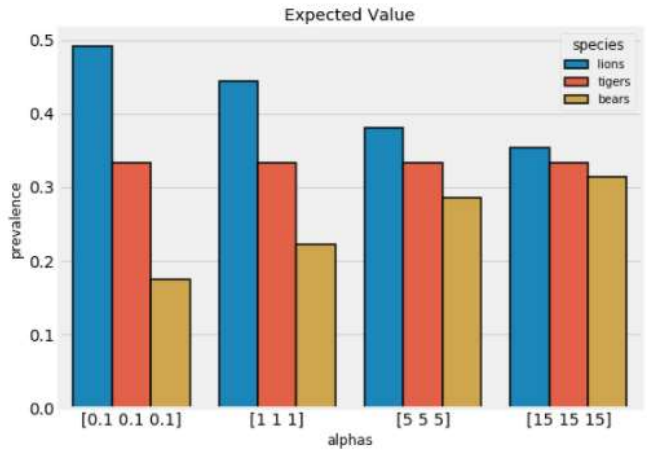
| | Lions | Tigers | Bears | alphas |
|---|---|---|---|---|
| 0 | 0.492063 | 0.333333 | 0.174603 | [0.1 0.1 0.1] |
| 1 | 0.444444 | 0.333333 | 0.222222 | [1 1 1] |
| 2 | 0.380952 | 0.333333 | 0.285714 | [5 5 5] |
| 3 | 0.352941 | 0.333333 | 0.313725 | [15 15 15] |

- For different hyperparameter vector, the expected value is different.
- As in above data, we can see that as the value of hyperparameter vector increases, the expected value of species varies:

| Species | Values of Hyperparameter vector | Expected probability of specie |
|---|---|---|
| Lion | increases ↑ | decreases ↓ |
| Tiger | increases ↑ | remains same – |
| Bear | increases ↑ | increases ↑ |

- There are two things: our belief and data. If we are confident on our belief then we will increase the pseudo-counts (hyperparameter vector values) but if we are less confident on our belief then we will rely on data and will reduce the hyperparameter weight.

```
melted = pd.melt(value_df, id_vars = 'alphas', value_name='prevalence',
        var_name = 'species')

plt.figure(figsize = (8, 6))
sns.barplot(x = 'alphas', y = 'prevalence', hue = 'species', data = melted,
            edgecolor = 'k', linewidth = 1.5);
plt.xticks(size = 14); plt.yticks(size = 14)
plt.title('Expected Value');
```



- Now we will start building our Bayesian model in pymc3 using markov chain monte carlo variant to draw samples from posterior.

**Bayesian Inference in Python with PyMC3**

```
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)
```

```
model
```

parameters_stickbreaking__ ~ TransformedDistribution
parameters ~ Dirichlet
observed_data ~ Multinomial

**Sampling from the Model**

The cell below samples 1000 draws from the posterior in 2 chains. We use 500 samples for tuning which are discarded. This means that for each random variable in the model - the `parameters` - we will have 2000 values drawn from the posterior distribution.

```
with model:
    # Sample from the posterior
    trace = pm.sample(draws=1000, chains=2, tune=500,
                      discard_tuned_samples=True)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (2 chains in 2 jobs)
NUTS: [parameters]
```

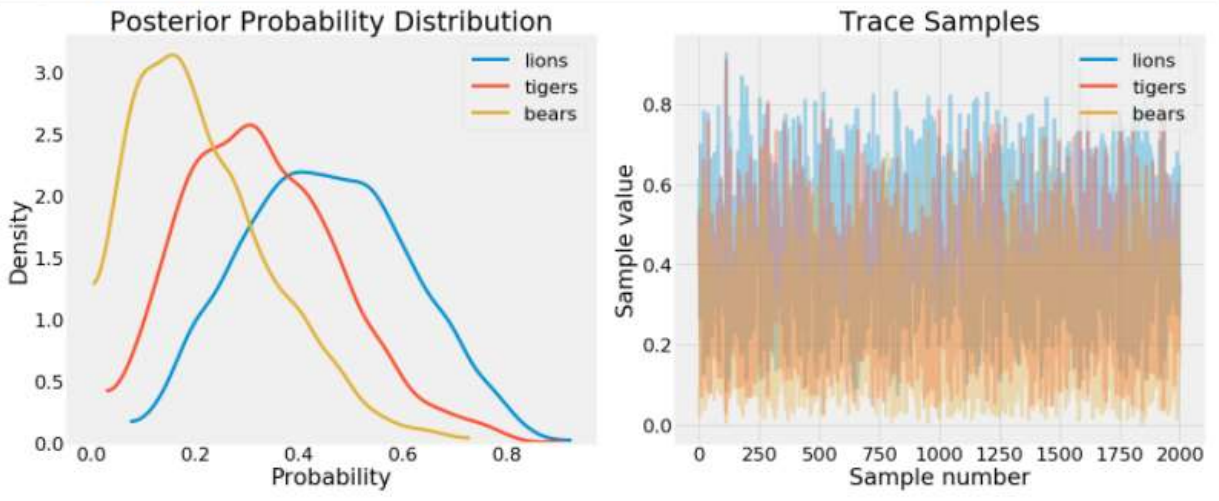
100.00% [3000/3000 00:56<00:00 Sampling 2 chains, 0 divergences]

```
Sampling 2 chains for 500 tune and 1_000 draw iterations (1_000 + 2_000 draws total) took 91 seconds.
```

- Trace plot shows the density estimate in first plot while in second plot, it shows all the samples that were drawn.

**Traceplot**

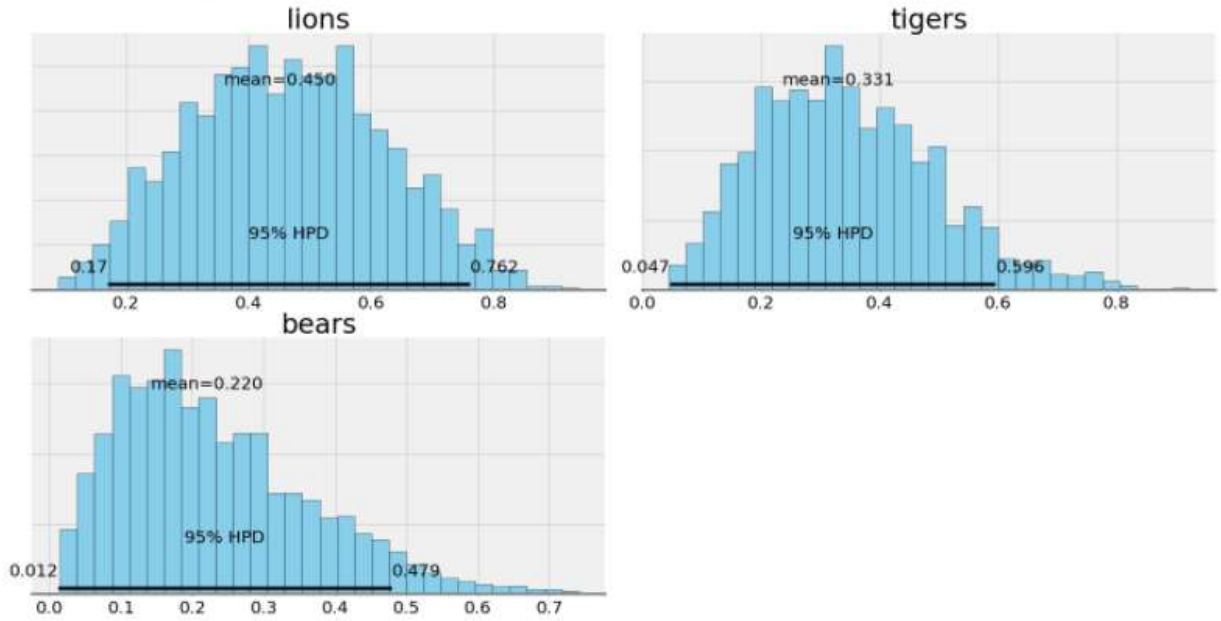
```
prop_cycle = plt.rcParams['axes.prop_cycle']
cs = [x['color'] for x in list(prop_cycle)]

ax = pm.traceplot(trace, varnames = ['parameters'], figsize = (20, 8), combined = True);
ax[0][0].set_title('Posterior Probability Distribution'); ax[0][1].set_title('Trace Samples');
ax[0][0].set_xlabel('Probability'); ax[0][0].set_ylabel('Density');
ax[0][1].set_xlabel('Sample number');
add_legend(ax[0][0])
add_legend(ax[0][1])
```



- From above plot, we can easily see that probability of bear is less than tigers and probability of tigers is less than the lions.

**Posterior Plot**

```
ax = pm.plot_posterior(trace, varnames = ['parameters'],
                       figsize = (20, 10), edgecolor = 'k');

plt.rcParams['font.size'] = 22
for i, a in enumerate(animals):
    ax[i].set_title(a);
```




- HPD stands for highest posterior density.
- In above posterior plot, the HDP is same for all the species that is 95%. It indicates extreme level of uncertainty.

```
summary = pm.summary(trace)
summary.index = animals
summary
```

| | mean | sd | mc_error | hpd_2.5 | hpd_97.5 | n_eff | Rhat |
|---|---|---|---|---|---|---|---|
| lions | 0.449862 | 0.158911 | 0.003371 | 0.170494 | 0.761727 | 1790.050932 | 0.999643 |
| tigers | 0.330618 | 0.148583 | 0.003256 | 0.047436 | 0.596449 | 1747.828016 | 1.000120 |
| bears | 0.219520 | 0.134556 | 0.003160 | 0.011707 | 0.479332 | 1763.772811 | 0.999678 |

- We can see that the mean of the samples is very close to the expected value. However, instead of just getting one number, we get a range of uncertainty as indicated by the large standard deviation and 95% highest probability interval.

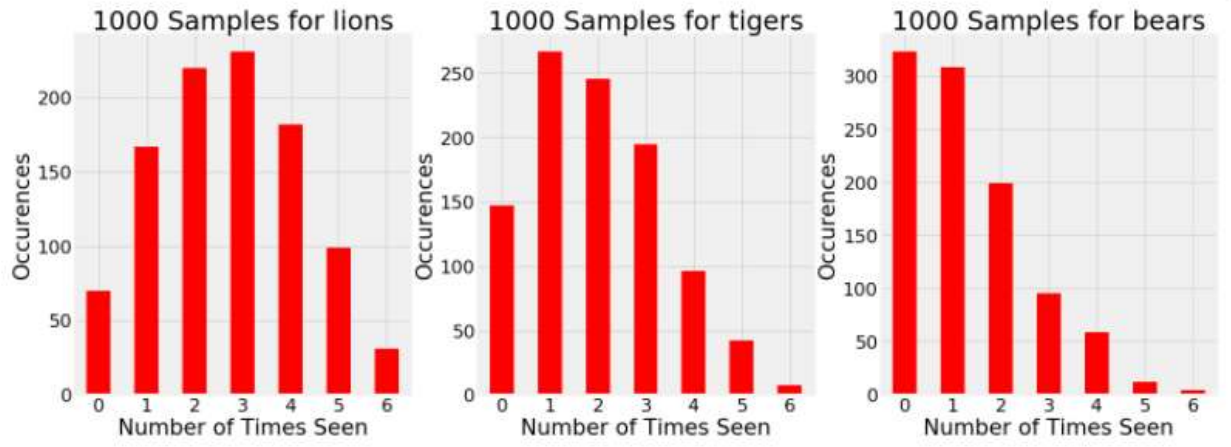
- We can now use the posterior (contained in the `trace`) to draw samples of data. For example, we can simulate 1000 trips to the wildlife preserve as follows.

**Sample From Posterior**

```
with model:
    samples = pm.sample_ppc(trace, samples = 1000)

dict(zip(animals, samples['observed_data'].mean(axis = 0)))
```

```
100%|██████████| 1000/1000 [00:01<00:00, 764.67it/s]
{'lions': 2.709, 'tigers': 1.98, 'bears': 1.311}
```

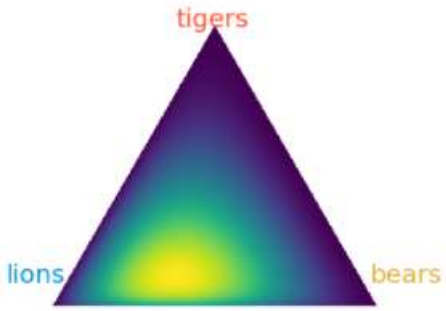
```
sample_df = pd.DataFrame(samples['observed_data'], columns = animals)

plt.figure(figsize = (22, 8))
for i, animal in enumerate(sample_df):
    plt.subplot(1, 3, i+1)
    sample_df[animal].value_counts().sort_index().plot.bar(color = 'r');
    plt.xticks(range(7), range(7), rotation = 0);
    plt.xlabel('Number of Times Seen'); plt.ylabel('Occurences');
    plt.title(f'1000 Samples for {animal}');
```




- These plots show the entire range of possible outcomes instead of only one and are so likely.
- After sampling, Dirichlet distribution looks like:

**Dirichlet Distribution**

```
draw_pdf_contours(Dirichlet(6 * pvals))
annotate_plot();
```



- From above triangle, we can see the balance shift towards the lions.
- Because of sampling, the uncertainty in posterior reduced. It is a fact that more data can make a person more sure about the event.
- With the increase in data, our estimates will totally depends on the true values and the priors will be then useless.

```
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)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (2 chains in 4 jobs)
NUTS: [parameters]
Sampling 2 chains: 100%|██████████| 3000/3000 [00:14<00:00, 201.98draws/s]
```

```
summary = pm.summary(trace)
summary.index = animals
summary
```

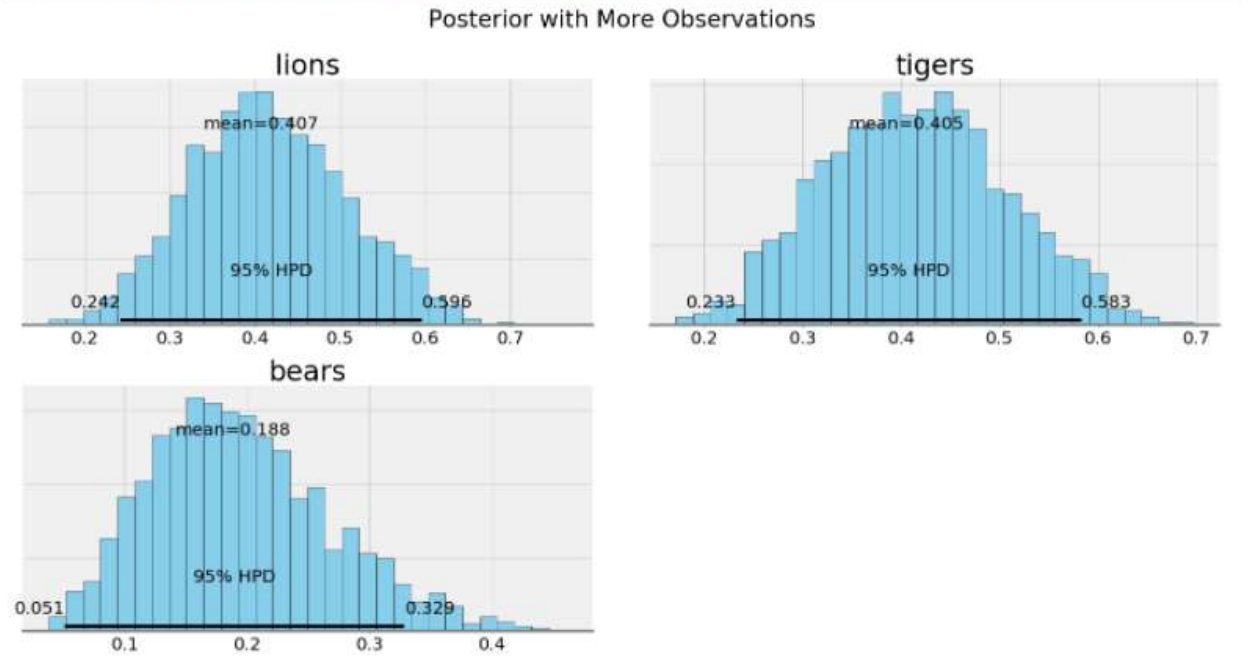
| | mean | sd | mc_error | hpd_2.5 | hpd_97.5 | n_eff | Rhat |
|---|---|---|---|---|---|---|---|
| lions | 0.406787 | 0.092886 | 0.001974 | 0.242218 | 0.596186 | 1827.764874 | 0.999828 |
| tigers | 0.405396 | 0.092945 | 0.002071 | 0.232870 | 0.582661 | 1935.575356 | 1.000273 |
| bears | 0.187817 | 0.073799 | 0.001752 | 0.050565 | 0.328844 | 2017.615361 | 0.999932 |

- From the above summary, it is stated that uncertainty of the prevalence of bear is decreasing while the uncertainty of tiger and lion is almost same.
- With the increase in data, uncertainty decreases that trains the model to give more accurate estimates.

$$\text{Amount of data } \alpha \frac{1}{\text{uncertainty}}$$

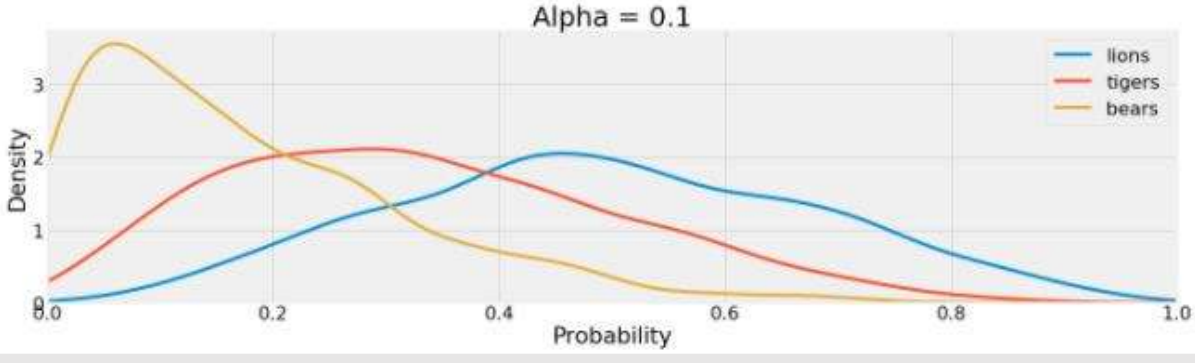
- **Posterior with more observations**

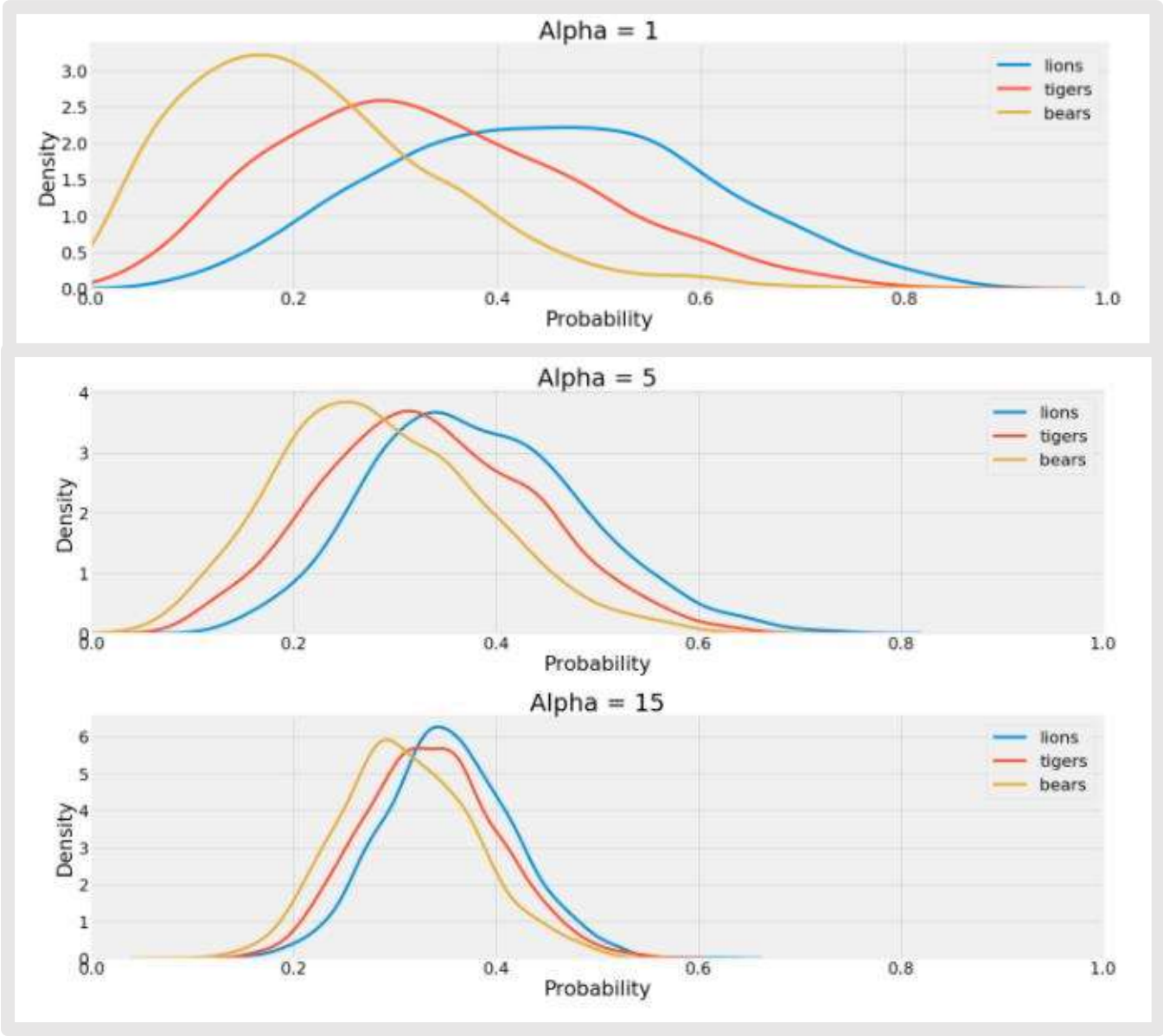
```
ax = pm.plot_posterior(trace, varnames = ['parameters'],
                       figsize = (20, 10), edgecolor = 'k');

plt.rcParams['font.size'] = 22
for i, a in enumerate(animals):
    ax[i].set_title(a);

plt.suptitle('Posterior with More Observations', y = 1.05);
```




- We can increase and decrease the confidence in our initial beliefs by altering the hyperparameter vector.
- Initially, we use $\boldsymbol{\alpha}$ = [1, 1, 1] but now we will alter the values to see the impact.

```
plt.figure(figsize = (20, 24))

for ii, (alpha, trace) in enumerate(trace_dict.items()):
    plt.subplot(4, 1, ii + 1)
    array = trace['parameters']
    for jj, animal in enumerate(animals):
        sns.kdeplot(array[:, jj], label = f'{animal}')
    plt.legend();
    plt.xlabel('Probability'); plt.ylabel('Density')
    plt.title(f'Alpha = {alpha}');
    plt.xlim((0, 1));

plt.tight_layout();
plt.show();
```

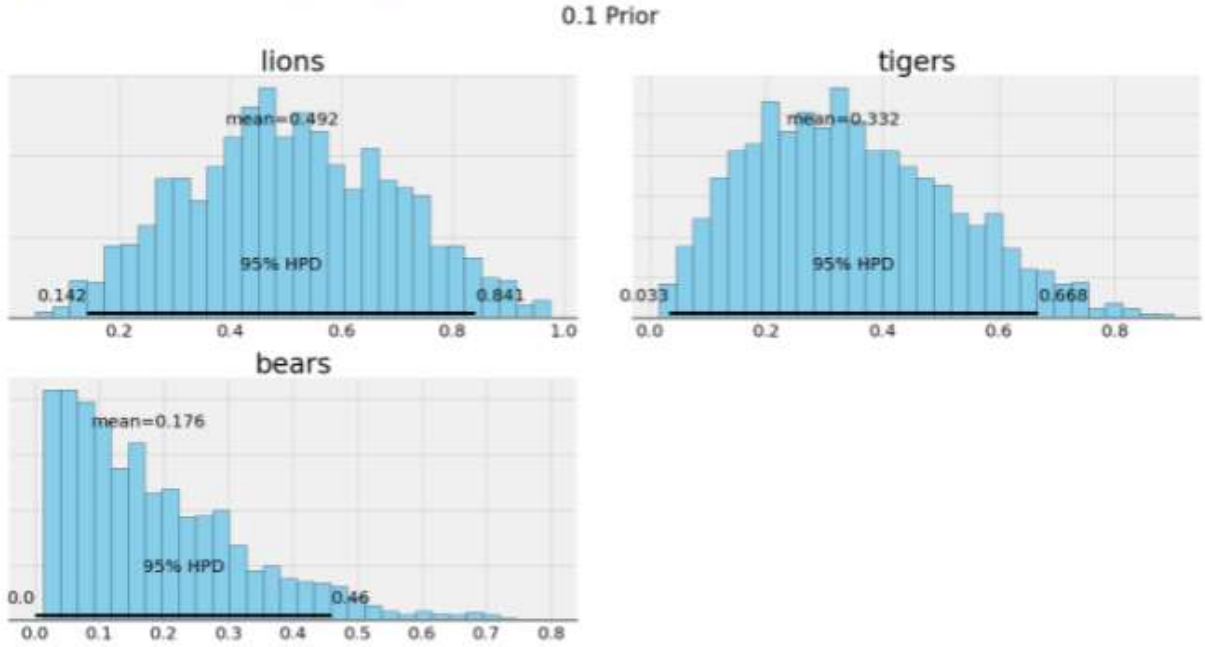




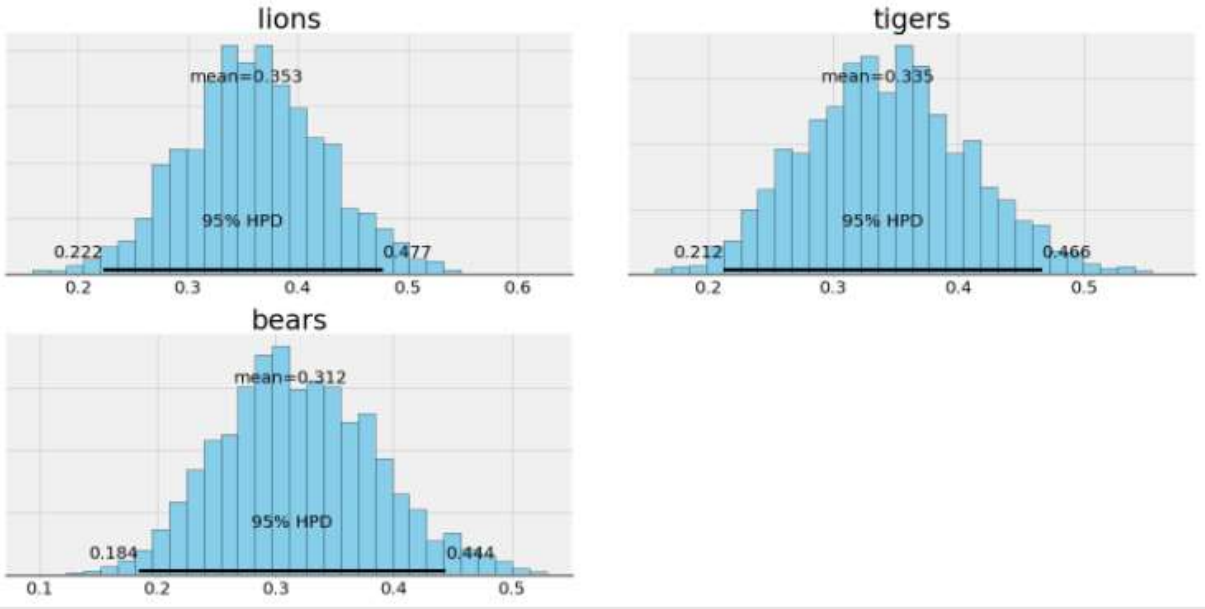

- The above plot explains the fact that hyperparameter have a large impact on the outcome.
- As the value increase, the distribution converge on one another.
- A lower value means the data itself has a greater weighting in the posterior, while a higher value results in greater weight placed on the pseudo-counts.

- Now we will compare the posterior plots with the least and high alpha value ( hyperparameter vector) to see the impact in more better way:

```
prior = '0.1'
trace = trace_dict[prior]

ax = pm.plot_posterior(trace, varnames = ['parameters'],
                       figsize = (20, 10), edgecolor = 'k');

plt.rcParams['font.size'] = 22
for i, a in enumerate(animals):
    ax[i].set_title(a);

plt.suptitle(f'{prior} Prior', y = 1.05);
```




```
prior = '15'
trace = trace_dict[prior]

ax = pm.plot_posterior(trace, varnames = ['parameters'],
                       figsize = (20, 10), edgecolor = 'k');

plt.rcParams['font.size'] = 22
for i, a in enumerate(animals):
    ax[i].set_title(a);

plt.suptitle(f'{prior} Prior', y = 1.05);
```



# Conclusion:

### Estimated Prevalence

Below are the means and 95% HPD for the estimates

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

### Probability of Next Observation is a Bear

- Based on the sampling, the probability of the prevalence of next animal is Bear = 22.9%.

# Summary:

1. Import libraries
2. Store the data.
3. Calculate the expected probability of species with $\boldsymbol{\alpha} = [1, 1, 1]$.
4. Calculate the expected probability of species with different $\boldsymbol{\alpha}$ values.
5. Build the Bayesian model
6. Draw samples from posterior in 2 chains.
7. Draw the trace plot.
8. Draw the posterior plot.
9. Analyze the summary statistics.
10. Visualize Dirichlet distribution triangle.
11. Increase the data by adding more observations and check the outcomes.
12. Variate the values of $\boldsymbol{\alpha}$ (Hyperparameter vector) and check the outcomes.
13. Compare the least and high $\boldsymbol{\alpha}$ value using plots.
14. Draw conclusion.