Wildlife Prevalence (Data Science Project 1)

August 22, 2020

Data Science Project 1: Wildlife Prevalence

Nicole Chantzi

The Problem On a trip to a high mountain forest we observe six animals. If we observed 1 bear, 2 foxes, and 3 rabbits and if we furthermore assume that these three are the only observable species, what is the probability that on our next encounter we will see a bear? What about a fox? Assume that each subpopulation is sufficiently large. We will tackle the forementioned problem by using Bayesian methods and comparing the obtained results with the respective ML estimators.

[1]:
```
import numpy as np
import seaborn as sns
from scipy.special import gamma, factorial
import matplotlib.pyplot as plt
%matplotlib inline
```

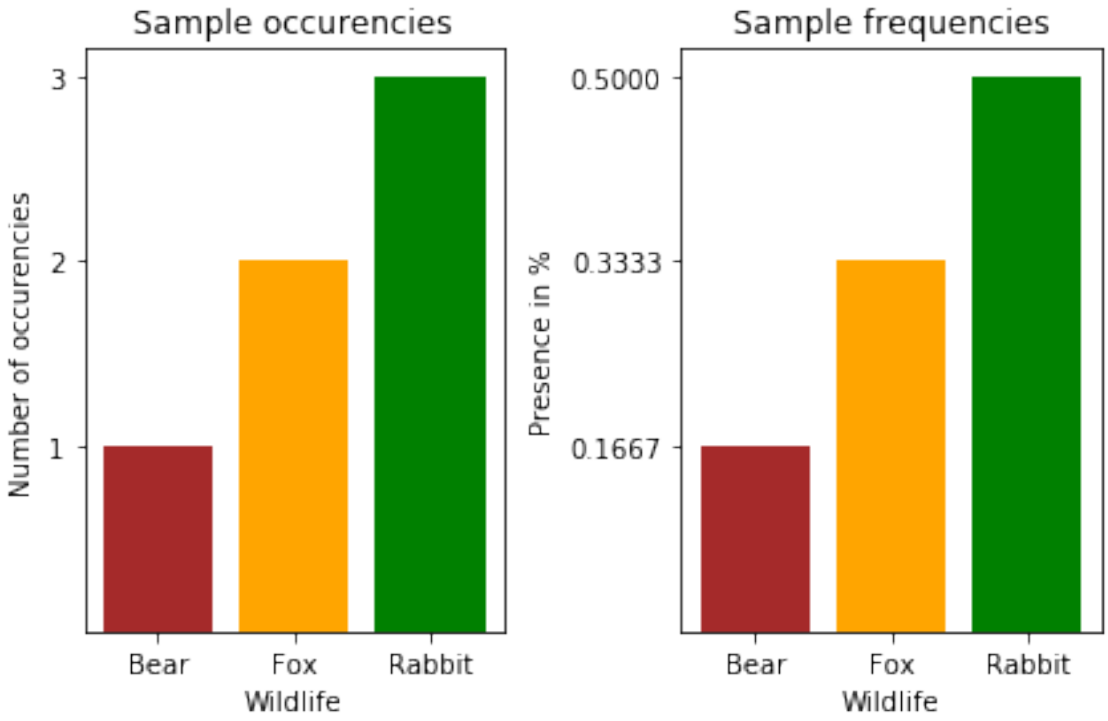
Data/Observations/Sample

[2]:
```
#3 rabbits, 2 foxes, 1 bear
Data = np.arange(1,4)
N = Data.sum()
```

Data Plotting

[3]:
```
#occurences
Species = np.array(['Bear','Fox','Rabbit'])
plt.subplot(1,2,1)
plt.bar(Species , Data , color=['brown','orange','green'])
plt.xlabel('Wildlife')
plt.ylabel('Number of occurencies')
plt.yticks(Data)
plt.title('Sample occurencies')
plt.tight_layout()

#Initial Frequencies
plt.subplot(1,2,2)
plt.bar(Species, np.true_divide(Data , Data.sum()) ,␣
 ↪color=['brown','orange','green'])
plt.xlabel('Wildlife')
```

```
plt.ylabel('Presence in %')
plt.yticks(np.true_divide(Data , Data.sum()))
plt.title('Sample frequencies')
plt.tight_layout()
plt.show()
```

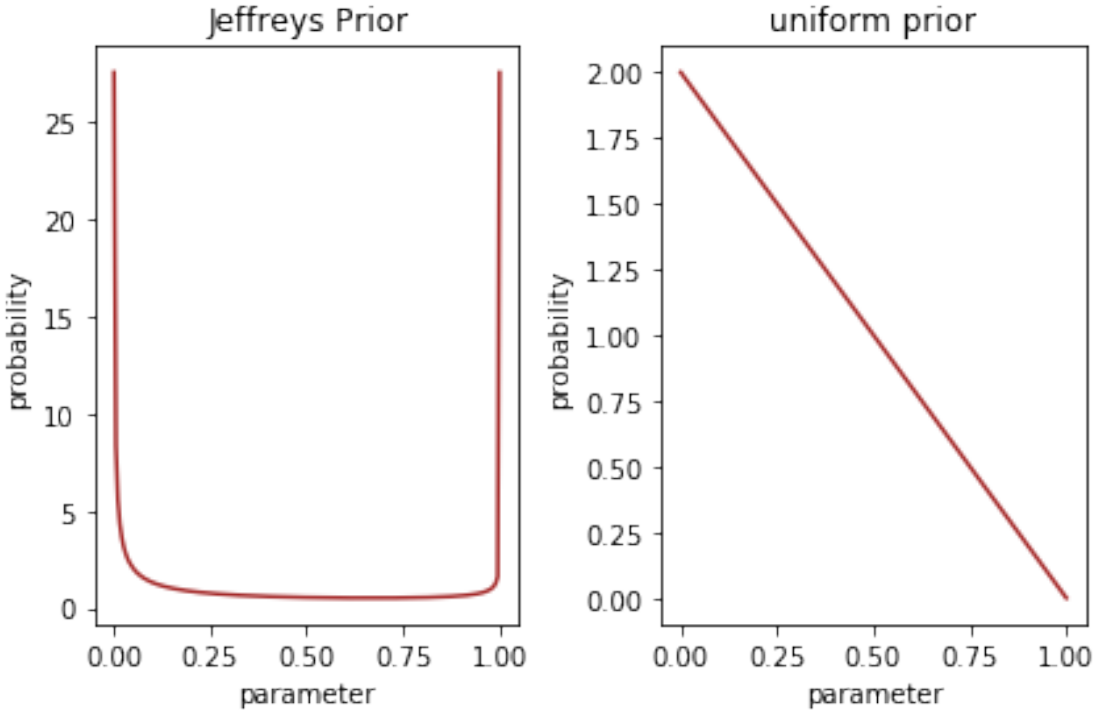


Prior Distribution

[4]:
```
def beta(x,y):
    return gamma(x)*gamma(y)/gamma(x+y)

def betapdf(x,a,b):
    return ((x**(a-1))*((1-x)**(b-1)))/beta(a,b)
```

[5]:
```
#Hyperparameter (pseudocounts) for Jeffey's prior
a = np.array([1/3,1/3,1/3])
a0 = a.sum()

#Hyperparameter (pseudocounts) Uniform Prior
b = np.array([1,1,1])
b0 = b.sum()

#Jeffrey's prior
plt.subplot(1,2,1)
```

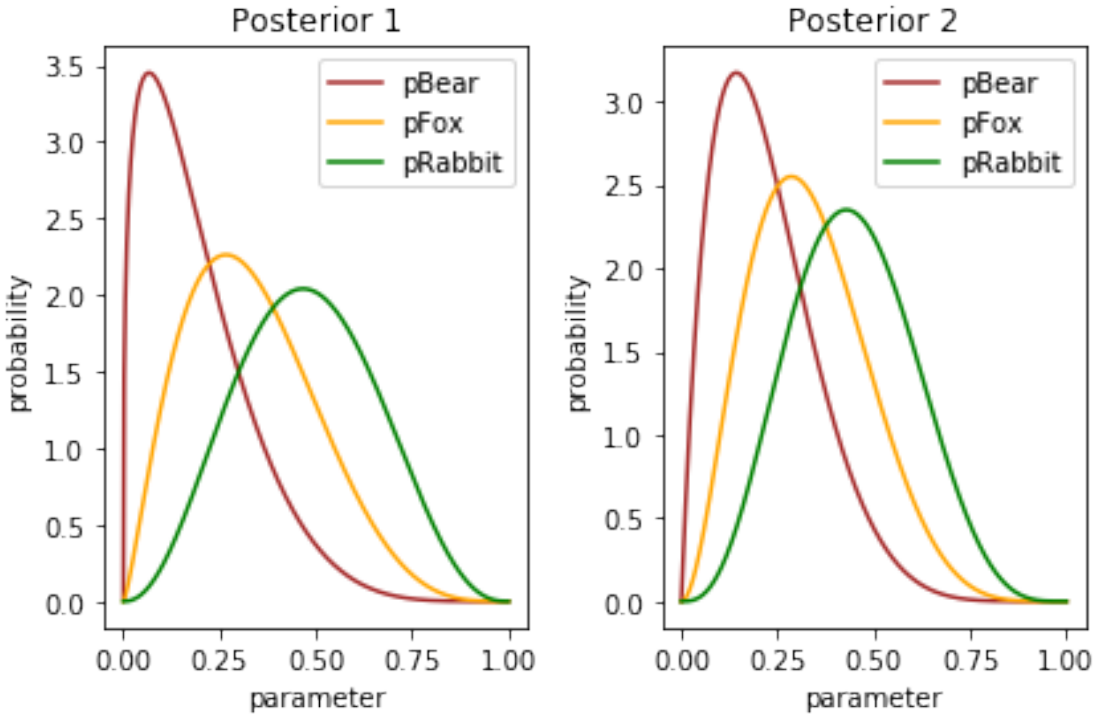
```
x = np.linspace(10e-4,0.999999,200)
y = betapdf(x,a[0],a0-a[0])
plt.plot(x,y,color='brown')
plt.xlabel('parameter')
plt.ylabel('probability')
plt.title('Jeffreys Prior')
plt.tight_layout()

#Uniform/Laplace prior
plt.subplot(1,2,2)
x = np.linspace(10e-4,0.9999,200)
y = betapdf(x,b[0],b0 - b[0])
plt.plot(x,y,color='brown')
plt.xlabel('parameter')
plt.ylabel('probability')
plt.title('uniform prior')
plt.tight_layout()
plt.show()
```



Posterior Distribution

[6]:
```
#Posterior from Jeffrey's Prior
x = np.linspace(10e-9,0.999999,1000)
A = a0 + N
```

```
colors = ['brown','orange','green']
plt.subplot(1,2,1)
for i in range(0,3):
        y = betapdf(x,a[i] + Data[i], A - (a[i] + Data[i]))
        plt.plot(x , y , color = colors[i])
plt.xlabel('parameter')
plt.ylabel('probability')
plt.legend(['pBear','pFox','pRabbit'],loc=0)
plt.title('Posterior 1')
plt.tight_layout()

#Posterior from uniform prior
x = np.linspace(10e-6,0.9999,1000)
B = b0 + N
plt.subplot(1,2,2)
for i in range(0,3):
        y = betapdf(x,b[i] + Data[i], B - (b[i] + Data[i]))
        plt.plot(x , y , color = colors[i])
plt.xlabel('parameter')
plt.ylabel('probability')
plt.legend(['pBear','pFox','pRabbit'],loc=0)
plt.title('Posterior 2')
plt.tight_layout()
plt.show()
```



Bayes Estimator

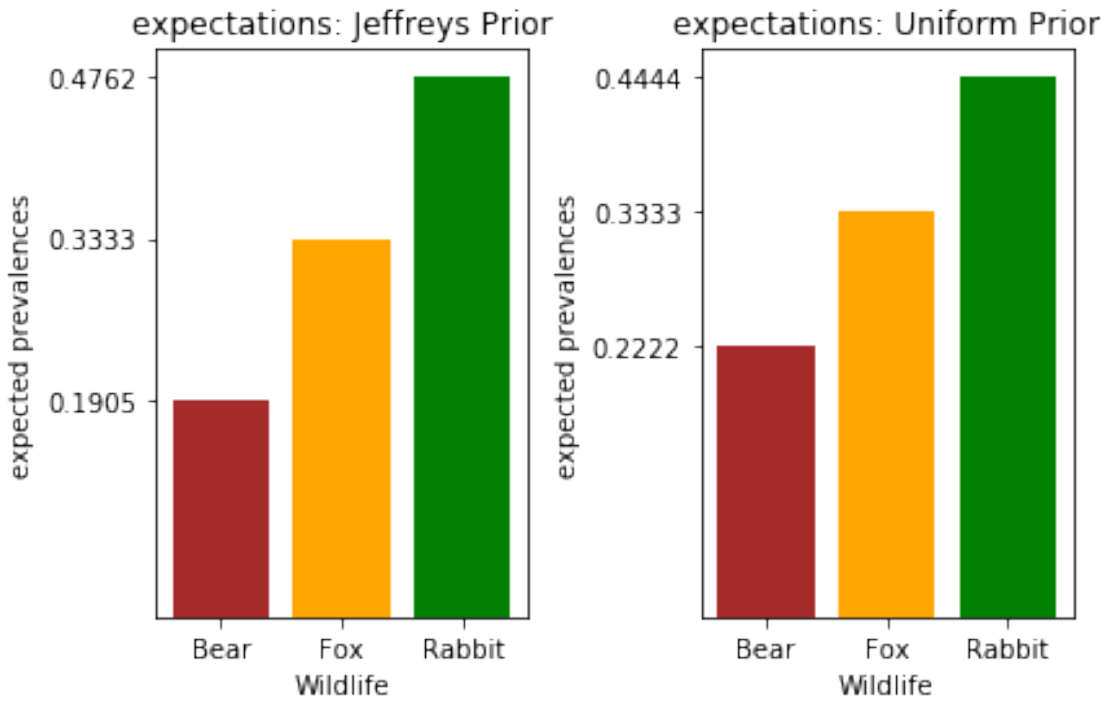
$E[p|Data] = \frac{a_i + Data_i}{\sum_i Data_i + \sum_i a_i}$

[7]:
```
def BE(a,x,N,a0):
    return (a+x)/(N+a0)

counts1 = []
counts2 = []
for i in range(0,3):
    counts1.append(BE(a[i],Data[i],N,a0))
    counts2.append(BE(b[i],Data[i],N,b0))
expect1 = np.array(counts1)
expect2 = np.array(counts2)

print('Bayes estimator (from Jeffreys prior)\n')
for i in range(0,3):
    print(Species[i], ": " , "{:1.3f}".format(BE(a[i],Data[i],N,a0)))
print('\n')

print('Bayes estimator (from uniform prior)\n')
for i in range(0,3):
    print(Species[i], ": " , "{:1.3f}".format(BE(b[i],Data[i],N,b0)))
```

```
Bayes estimator (from Jeffreys prior)

Bear :  0.190
Fox :  0.333
Rabbit :  0.476


Bayes estimator (from uniform prior)

Bear :  0.222
Fox :  0.333
Rabbit :  0.444
```

[8]:
```
#Expectations for Jeffrey's prior
plt.subplot(1,2,1)
plt.bar(Species , expect1, label = 'posterior wildlife prevalence',␣
 ↪color=['brown','orange','green'])
plt.xlabel('Wildlife')
plt.ylabel('expected prevalences')
plt.yticks(expect1)
plt.title('expectations: Jeffreys Prior')
plt.tight_layout()
```

```
#Expectations for Uniform Prior
plt.subplot(1,2,2)
plt.bar(Species , expect2, label = 'posterior wildlife prevalence',␣
 ↪color=['brown','orange','green'])
plt.xlabel('Wildlife')
plt.ylabel('expected prevalences')
plt.yticks(expect2)
plt.title('expectations: Uniform Prior')
plt.tight_layout()
plt.show()
```



MLE estimator $\hat{p}_i = \frac{Data[i]}{\sum_i Data[i]} = \frac{Data[i]}{N}$

[9]:
```
t = []
for i in range(0,3):
    q = Data[i]/N
    t.append(q)
p = np.array(t)
print("Maximum Likelihood Estimator \n")
for i in range(0,3):
    print(Species[i], ": " , "{:1.3f}".format(p[i]))
```

```
Maximum Likelihood Estimator
```

```
Bear :  0.167
Fox :  0.333
Rabbit :  0.500
```

[10]:
```
#Different Data Set
Data2 = np.array([0,2,4])
t = []
for i in range(0,3):
    q = Data2[i]/N
    t.append(q)
p = np.array(t)
print("Maximum Likelihood Estimator \n")
for i in range(0,3):
    print(Species[i], ": " , "{:1.3f}".format(p[i]))
```

```
Maximum Likelihood Estimator

Bear :  0.000
Fox :  0.333
Rabbit :  0.667
```

The problem arising from ML estimators is that they match the dataset in the best possible way or, equivalently, the MLE method adopts the pdf that most likely produced the dataset. For example, on the first trip to the wilflife habitat we could have encountered 6 animals of which 4 rabbits and 2 foxes. In this case, the ML estimator would suggest that $p_{bear} = 0$, i.e. there are no bears! Of course, bears share a smaller proportion of wildlife population and consequently it would be rather likely in a small isolated sample not to encounter a bear. This doesn't mean that bears do not participate in the wildlife population. Of course, that's exactly how ML estimator operates. In our previous theoretical encounter with 6 animals two of which were foxes and the rest rabbits, it would adapt the (multinomial) model into one which would most likely produce the observed sample. And that, of course, is now intuitively obvious, that the most likely model to produce our sample, is a population without bears. This results in ignoring sampling variations, especially large deviations, leading into possible (potentially huge) misconceptions about the data structure of the population. MLEs are much "safer" when dealing with a sufficiently large sample. Bayesian methods incorporate uncertainty into probabilistic model, exactly foreseeing these sampling variations.

Random Sampling from Posteriors with Laplace Prior

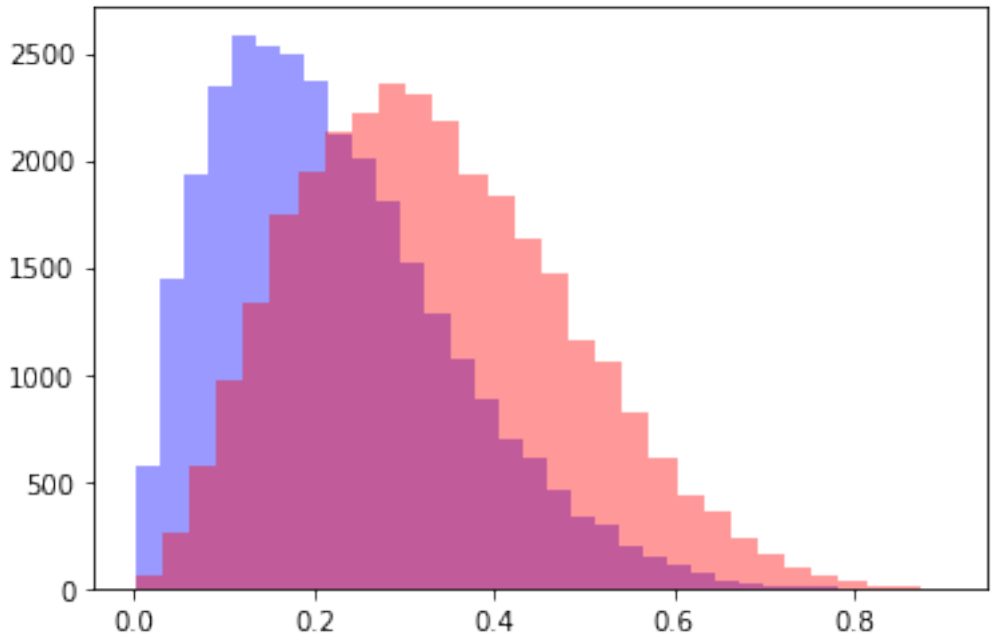
[11]:
```
#uniform prior
K = 30000
V = K
countsbear = []
countsfox = []
countsrabbit = []
freqbear =[]
freqfox = []
freqrabbit = []
prev = [0 for i in range(0,len(Data))] ####wildlife sample prevalences
```

```
while K>0:
    x = np.random.dirichlet(b + Data , size=None)
    y = np.random.multinomial(6 , x , size=None)
    countsbear.append(y[0])
    countsfox.append(y[1])
    countsrabbit.append(y[2])
    freqbear.append(x[0])
    freqfox.append(x[1])
    freqrabbit.append(x[2])
    K-=1

K = 6*V
print(sum(countsbear)/K , sum(countsfox)/K , sum(countsrabbit)/K)
```

```
0.2214611111111111 0.33331666666666665 0.44522222222222224
```

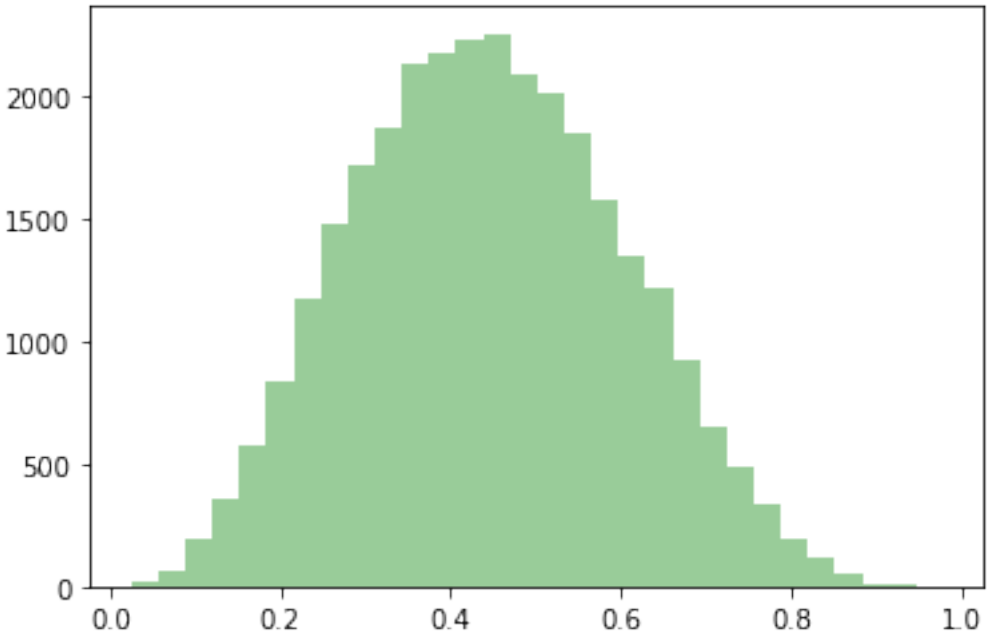
[12]:
```
#histograms of produced pseudo-samples
sns.distplot(freqbear, bins = 30, kde = False , color = 'blue')
sns.distplot(freqfox, bins = 30, kde = False , color = 'red')
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26940188588>
```



[13]:
```
sns.distplot(freqrabbit, bins = 30, kde = False , color='green')
```

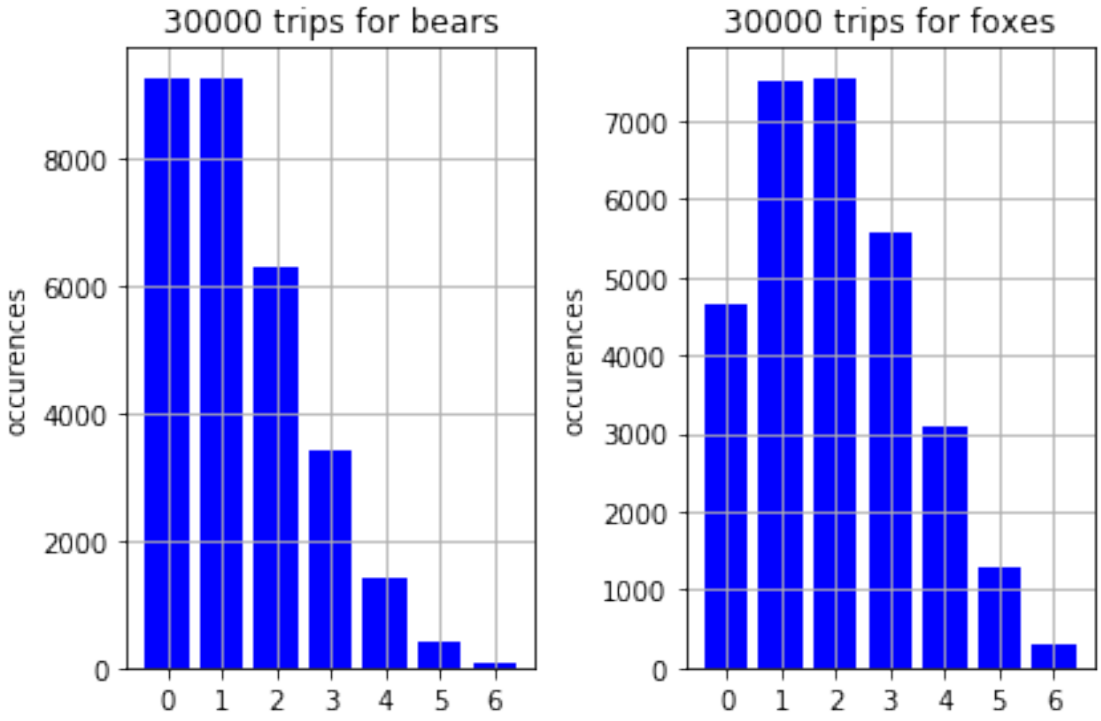
[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26940805588>
```



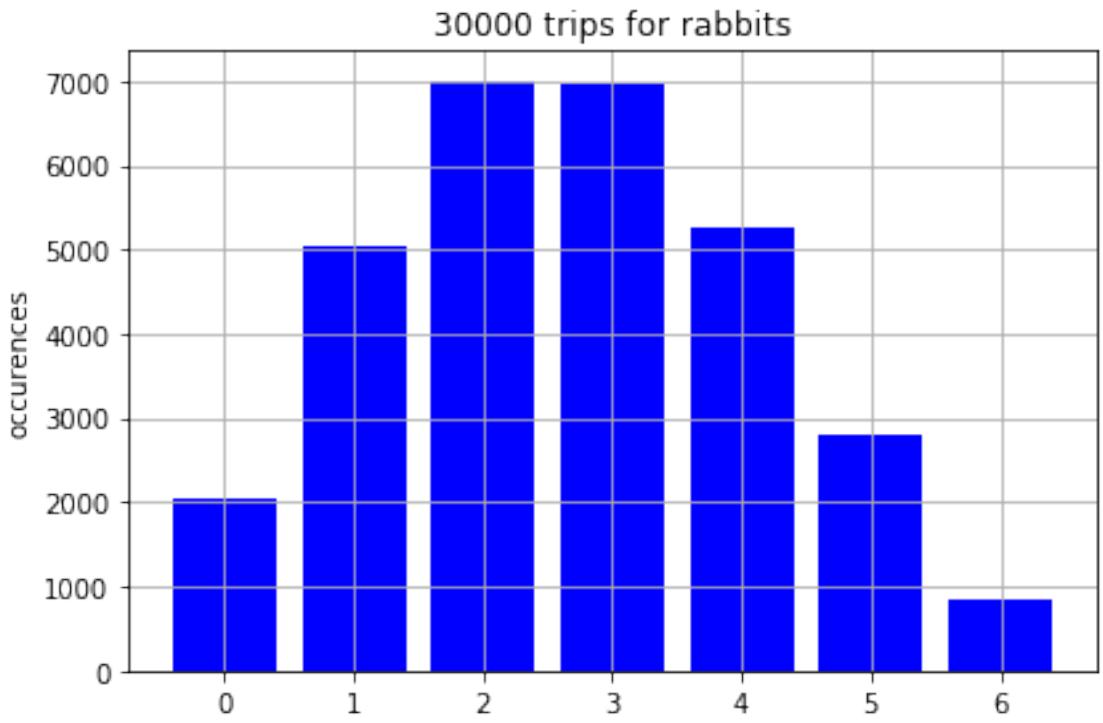
[14]:
```
def count(mylist, N):
    occur = []
    for i in range(0,N):
        occur.append(mylist.count(i))
    return occur
```

[15]:
```
occur1 = count(countsbear,N+1)
occur2 = count(countsfox, N+1)
occur3 = count(countsrabbit, N+1)

ax = [i for i in range(0,N+1)]
plt.subplot(1,2,1)    #bears
plt.bar(ax,occur1,color='blue')
plt.ylabel('occurences')
plt.xticks(ax)
plt.grid()
plt.tight_layout()
plt.title('{:d} trips for bears'.format(V))

plt.subplot(1,2,2) #foxes
plt.bar(ax,occur2,color='blue')
plt.ylabel('occurences')
plt.xticks(ax)
plt.grid()
plt.tight_layout()
plt.title('{:d} trips for foxes'.format(V))
```

```
plt.show()
```



[16]:
```
#rabbits
plt.bar(ax,occur3,color='blue')
plt.title('{:d} trips for rabbits'.format(V))
plt.ylabel('occurences')
plt.xticks(ax)
plt.grid()
plt.tight_layout()
plt.show()
```



Final Answer

Bayes estimator (from Jeffreys prior)

Bear : 0.190 Fox : 0.333 Rabbit : 0.476

Bayes estimator (from uniform prior)

Bear : 0.222 Fox : 0.333 Rabbit : 0.444

Hence, we would expect to observe a bear on our next encounter with 19% chance. The MLE is closer to the Bayes estimator that is produced by a non-informative prior, namely Jeffreys prior, than the informative one, i.e. Laplace's prior. Jeffrey's prior, introduces a small deviation of 0.02 additional probability of encountering a bear, than the sample frequency. That being said, it somehow predicts that the sampling frequency may not be a representive sample of the whole population.

Furthermore, we see that the posterior probability of encountering a fox is the same as the MLE and the sampling mean. This is due to the fact that the obtained data did not add any new knowledge about the fox population, since it verified - by initially observing 2 foxes out of 6 encounters - what was merely anticipated.