Capstone 2: Biodiversity Project

Introduction

You are a biodiversity analyst working for the National Parks Service. You're going to help them analyze some data about species at various national parks.

Note: The data that you'll be working with for this project is *inspired* by real data, but is mostly fictional.

Step 1

Import the modules that you'll be using in this assignment:

- `from matplotlib import pyplot as plt`
- `import pandas as pd`

In [2]:

```
from matplotlib import pyplot as plt
import pandas as pd
```

Step 2

You have been given two CSV files. `species_info.csv` with data about different species in our National Parks, including:

- The scientific name of each species
- The common names of each species
- The species conservation status

Load the dataset and inspect it:

- Load `species_info.csv` into a DataFrame called `species`

In [3]:

```
species = pd.read_csv('species_info.csv')
```

Inspect each DataFrame using `.head()` .

```
In [4]: print species.head()
```

```
  category                scientific_name  \
0   Mammal  Clethrionomys gapperi gapperi   
1   Mammal                      Bos bison   
2   Mammal                     Bos taurus   
3   Mammal                     Ovis aries   
4   Mammal                 Cervus elaphus   

                                       common_names conservation_status  
0                          Gapper's Red-Backed Vole                 NaN  
1                             American Bison, Bison                 NaN  
2  Aurochs, Aurochs, Domestic Cattle (Feral), Dom...                 NaN  
3  Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)                 NaN  
4                                     Wapiti Or Elk                 NaN  
```

Step 3

Let's start by learning a bit more about our data. Answer each of the following questions.

How many different species are in the `species` DataFrame?

```
In [5]: species.scientific_name.nunique()
```

```
Out[5]: 5541
```

What are the different values of `category` in `species` ?

```
In [6]: species.category.unique()
```

```
Out[6]: array(['Mammal', 'Bird', 'Reptile', 'Amphibian', 'Fish', 'Vascular Plant',
               'Nonvascular Plant'], dtype=object)
```

What are the different values of `conservation_status` ?

```
In [7]: species.conservation_status.unique()
```

```
Out[7]: array([nan, 'Species of Concern', 'Endangered', 'Threatened',
               'In Recovery'], dtype=object)
```

Step 4

Let's start doing some analysis!

The column `conservation_status` has several possible values:

- `Species of Concern` : declining or appear to be in need of conservation
- `Threatened` : vulnerable to endangerment in the near future
- `Endangered` : seriously at risk of extinction
- `In Recovery` : formerly `Endangered` , but currnetly neither in danger of extinction throughout all or a significant portion of its range

We'd like to count up how many species meet each of these criteria. Use `groupby` to count how many `scientific_name` meet each of these criteria.

In [8]:
```
species.groupby('conservation_status').scientific_name.nunique().reset_index()
```

Out[8]:
```
     conservation_status  scientific_name
0             Endangered               15
1            In Recovery                4
2     Species of Concern              151
3             Threatened               10
```

As we saw before, there are far more than 200 species in the `species` table. Clearly, only a small number of them are categorized as needing some sort of protection. The rest have `conservation_status` equal to `None` . Because `groupby` does not include `None` , we will need to fill in the null values. We can do this using `.fillna` . We pass in however we want to fill in our `None` values as an argument.

Paste the following code and run it to see replace `None` with `No Intervention` :

```
species.fillna('No Intervention', inplace=True)
```

In [9]:
```
species.fillna('No Intervention', inplace=True)
```

Great! Now run the same `groupby` as before to see how many species require `No Protection` .

In [10]:
```
species.groupby('conservation_status').scientific_name.nunique().reset_index()
```

Out[10]:
```
     conservation_status  scientific_name
0             Endangered               15
1            In Recovery                4
2        No Intervention             5363
3     Species of Concern              151
4             Threatened               10
```

Let's use `plt.bar` to create a bar chart. First, let's sort the columns by how many species are in each categories. We can do this using `.sort_values` . We use the the keyword `by` to indicate which column we want to sort by.

Paste the following code and run it to create a new DataFrame called `protection_counts` , which is sorted by `scientific_name` :

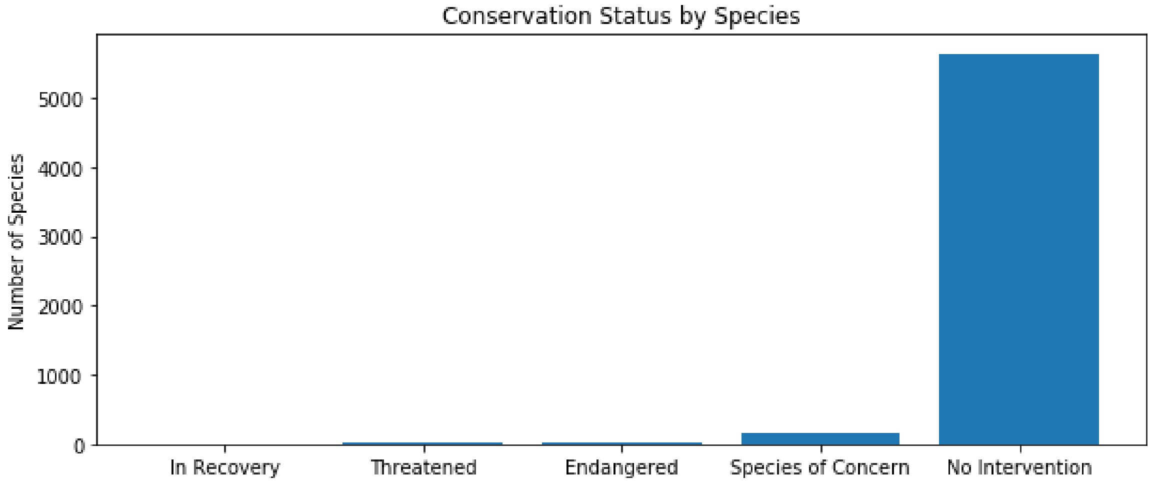
```
protection_counts = species.groupby('conservation_status')\
    .scientific_name.count().reset_index()\
    .sort_values(by='scientific_name')
```

```
In [11]: protection_counts = species.groupby('conservation_status')\
             .scientific_name.count().reset_index()\
             .sort_values(by='scientific_name')
```

Now let's create a bar chart!

1. Start by creating a wide figure with `figsize=(10, 4)`
2. Start by creating an axes object called `ax` using `plt.subplot`.
3. Create a bar chart whose heights are equal to `scientific_name` column of `protection_counts`.
4. Create an x-tick for each of the bars.
5. Label each x-tick with the label from `conservation_status` in `protection_counts`
6. Label the y-axis `Number of Species`
7. Title the graph `Conservation Status by Species`
8. Plot the grap using `plt.show()`

```
In [12]: plt.figure(figsize=(10,4))
         ax = plt.subplot()
         plt.bar(range(len(protection_counts)),protection_counts.scientific_name)
         ax.set_xticks(range(len(protection_counts)))
         ax.set_xticklabels(protection_counts.conservation_status)
         plt.ylabel('Number of Species')
         plt.title('Conservation Status by Species')
         plt.show()
```



Step 4

Are certain types of species more likely to be endangered?

Let's create a new column in `species` called `is_protected`, which is `True` if `conservation_status` is not equal to `No Intervention`, and `False` otherwise.

```
In [13]: species['is_protected'] = species.conservation_status.apply(lambda x: 'True' if x
```

Let's group by *both* `category` and `is_protected`. Save your results to `category_counts`.

In [14]:

```
category_counts = species.groupby(['category', 'is_protected']).scientific_name.c
```

Examine `category_count` using `head()` .

In [15]:

```
print category_counts.head()
```

```
    category is_protected  scientific_name
0  Amphibian        False               73
1  Amphibian         True                7
2       Bird        False              442
3       Bird         True               79
4       Fish        False              116
```

It's going to be easier to view this data if we pivot it. Using `pivot` , rearange `category_counts` so that:

- `columns` is `conservation_status`
- `index` is `category`
- `values` is `scientific_name`

Save your pivoted data to `category_pivot` . Remember to `reset_index()` at the end.

In [16]:

```
category_pivot = category_counts.pivot(index='category', columns='is_protected',
```

Examine `category_pivot` .

In [17]:

```
print category_pivot
```

```
is_protected           category  False  True
0                     Amphibian     73     7
1                          Bird    442    79
2                          Fish    116    11
3                        Mammal    176    38
4             Nonvascular Plant    328     5
5                       Reptile     74     5
6                Vascular Plant   4424    46
```

Use the `.columns` property to rename the categories `True` and `False` to something more description:

- Leave `category` as `category`
- Rename `False` to `not_protected`
- Rename `True` to `protected`

In [18]:

```
category_pivot = category_pivot.rename(columns={'False': 'not_protected', 'True':
```

Let's create a new column of `category_pivot` called `percent_protected` , which is equal to `protected` (the number of species that are protected) divided by `protected` plus `not_protected` (the total number of species).

In [19]:

```
category_pivot['percent_protected'] = category_pivot.protected / (category_pivot.
```

Examine `category_pivot`.

```
In [20]: print category_pivot
```

```
is_protected           category  not_protected  protected  percent_protected
0                     Amphibian             73          7           0.087500
1                          Bird            442         79           0.151631
2                          Fish            116         11           0.086614
3                        Mammal            176         38           0.177570
4             Nonvascular Plant            328          5           0.015015
5                       Reptile             74          5           0.063291
6                Vascular Plant           4424         46           0.010291
```

It looks like species in category `Mammal` are more likely to be endangered than species in `Bird`. We're going to do a significance test to see if this statement is true. Before you do the significance test, consider the following questions:

- Is the data numerical or categorical?
- How many pieces of data are you comparing?

Based on those answers, you should choose to do a *chi squared test*. In order to run a chi squared test, we'll need to create a contingency table. Our contingency table should look like this:

	protected	not protected
Mammal	?	?
Bird	?	?

Create a table called `contingency` and fill it in with the correct numbers

```
In [21]: contingency = [[38, 176], [79, 442]]
```

In order to perform our chi square test, we'll need to import the correct function from scipy. Past the following code and run it:

```
from scipy.stats import chi2_contingency
```

```
In [22]: from scipy.stats import chi2_contingency
```

Now run `chi2_contingency` with `contingency`.

```
In [23]: chi2, pval, dof, expected = chi2_contingency(contingency)
         print pval
```

```
0.445901703047197
```

It looks like this difference isn't significant!

Let's test another. Is the difference between `Reptile` and `Mammal` significant?

```
In [24]: contingency_2 = [[5, 74], [38, 176]]
         chi2, pval, dof, expected = chi2_contingency(contingency_2)
         print pval

0.02338465214871547
```

Yes! It looks like there is a significant difference between `Reptile` and `Mammal` !

Step 5

Conservationists have been recording sightings of different species at several national parks for the past 7 days. They've saved sent you their observations in a file called `observations.csv` . Load `observations.csv` into a variable called `observations` , then use `head` to view the data.

```
In [25]: import pandas as pd
         observations = pd.read_csv('observations.csv')
         print observations.head()

            scientific_name                            park_name  observations
0        Vicia benghalensis  Great Smoky Mountains National Park            68
1            Neovison vison  Great Smoky Mountains National Park            77
2         Prunus subcordata               Yosemite National Park           138
3      Abutilon theophrasti                  Bryce National Park            84
4  Githopsis specularioides  Great Smoky Mountains National Park            85
```

Some scientists are studying the number of sheep sightings at different national parks. There are several different scientific names for different types of sheep. We'd like to know which rows of `species` are referring to sheep. Notice that the following code will tell us whether or not a word occurs in a string:

```
In [26]: # Does "Sheep" occur in this string?
         str1 = 'This string contains Sheep'
         'Sheep' in str1

Out[26]: True
```

```
In [27]: # Does "Sheep" occur in this string?
         str2 = 'This string contains Cows'
         'Sheep' in str2

Out[27]: False
```

Use `apply` and a `lambda` function to create a new column in `species` called `is_sheep` which is `True` if the `common_names` contains `'Sheep'` , and `False` otherwise.

```
In [28]: species['is_sheep'] = species.common_names.apply(lambda x: 'Sheep' in x)
```

Select the rows of `species` where `is_sheep` is `True` and examine the results.

In [29]:

```
species[species.is_sheep]
```

Out[29]:

	category	scientific_name	common_names	conservation_status	is_protected	is_sheep
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True
1139	Vascular Plant	Rumex acetosella	Sheep Sorrel, Sheep Sorrell	No Intervention	False	True
2233	Vascular Plant	Festuca filiformis	Fineleaf Sheep Fescue	No Intervention	False	True
3014	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
3758	Vascular Plant	Rumex acetosella	Common Sheep Sorrel, Field Sorrel, Red Sorrel,...	No Intervention	False	True
3761	Vascular Plant	Rumex paucifolius	Alpine Sheep Sorrel, Fewleaved Dock, Meadow Dock	No Intervention	False	True
4091	Vascular Plant	Carex illota	Sheep Sedge, Smallhead Sedge	No Intervention	False	True
4383	Vascular Plant	Potentilla ovina var. ovina	Sheep Cinquefoil	No Intervention	False	True
4446	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True	True

Many of the results are actually plants. Select the rows of `species` where `is_sheep` is `True` and `category` is `Mammal` . Save the results to the variable `sheep_species` .

In [30]:

```
sheep_species = species[(species.is_sheep) & (species.category == 'Mammal')]
print sheep_species
```

```
     category           scientific_name  \
3      Mammal                Ovis aries
3014   Mammal           Ovis canadensis
4446   Mammal   Ovis canadensis sierrae

                                         common_names conservation_status  \
3     Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)     No Intervention
3014                         Bighorn Sheep, Bighorn Sheep  Species of Concern
4446                          Sierra Nevada Bighorn Sheep          Endangered

     is_protected  is_sheep
3           False      True
3014         True      True
4446         True      True
```

Now merge `sheep_species` with `observations` to get a DataFrame with observations of sheep. Save this DataFrame as `sheep_observations` .

In [31]:

```
sheep_observations = pd.merge(sheep_species, observations)
print sheep_observations
```

```
   category          scientific_name  \
0    Mammal               Ovis aries   
1    Mammal               Ovis aries   
2    Mammal               Ovis aries   
3    Mammal               Ovis aries   
4    Mammal          Ovis canadensis   
5    Mammal          Ovis canadensis   
6    Mammal          Ovis canadensis   
7    Mammal          Ovis canadensis   
8    Mammal  Ovis canadensis sierrae   
9    Mammal  Ovis canadensis sierrae   
10   Mammal  Ovis canadensis sierrae   
11   Mammal  Ovis canadensis sierrae   

                                         common_names conservation_status  \
0   Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)     No Intervention   
1   Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)     No Intervention   
2   Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)     No Intervention   
3   Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)     No Intervention   
4                        Bighorn Sheep, Bighorn Sheep  Species of Concern   
5                        Bighorn Sheep, Bighorn Sheep  Species of Concern   
6                        Bighorn Sheep, Bighorn Sheep  Species of Concern   
7                        Bighorn Sheep, Bighorn Sheep  Species of Concern   
8                         Sierra Nevada Bighorn Sheep          Endangered   
9                         Sierra Nevada Bighorn Sheep          Endangered   
10                        Sierra Nevada Bighorn Sheep          Endangered   
11                        Sierra Nevada Bighorn Sheep          Endangered   

   is_protected  is_sheep                            park_name  observations
0         False      True              Yosemite National Park           126
1         False      True  Great Smoky Mountains National Park            76
2         False      True                 Bryce National Park           119
3         False      True            Yellowstone National Park           221
4          True      True            Yellowstone National Park           219
5          True      True                 Bryce National Park           109
6          True      True              Yosemite National Park           117
7          True      True  Great Smoky Mountains National Park            48
8          True      True            Yellowstone National Park            67
9          True      True              Yosemite National Park            39
10         True      True                 Bryce National Park            22
11         True      True  Great Smoky Mountains National Park            25
```

How many total sheep observations (across all three species) were made at each national park? Use `groupby` to get the `sum` of `observations` for each `park_name`. Save your answer to `obs_by_park`.

This is the total number of sheep observed in each park over the past 7 days.

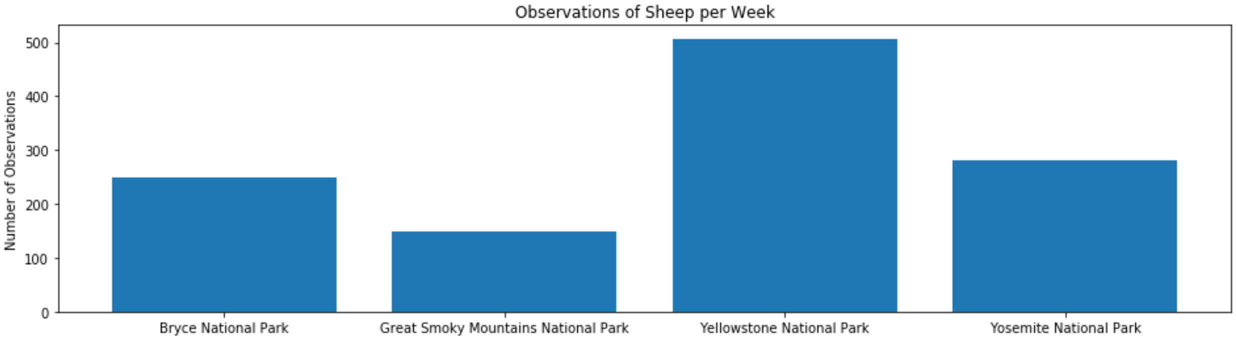
```
In [32]: obs_by_park = sheep_observations.groupby('park_name').observations.sum().reset_in
         print obs_by_park
```

```
                             park_name  observations
0                 Bryce National Park           250
1   Great Smoky Mountains National Park           149
2           Yellowstone National Park           507
3              Yosemite National Park           282
```

Create a bar chart showing the different number of observations per week at each park.

1. Start by creating a wide figure with `figsize=(16, 4)`
2. Start by creating an axes object called `ax` using `plt.subplot`.
3. Create a bar chart whose heights are equal to `observations` column of `obs_by_park`.
4. Create an x-tick for each of the bars.
5. Label each x-tick with the label from `park_name` in `obs_by_park`
6. Label the y-axis `Number of Observations`
7. Title the graph `Observations of Sheep per Week`
8. Plot the grap using `plt.show()`

```
In [33]: plt.figure(figsize=(16, 4))
         ax=plt.subplot()
         plt.bar(range(len(obs_by_park)),obs_by_park.observations)
         ax.set_xticks(range(len(obs_by_park)))
         ax.set_xticklabels(obs_by_park.park_name)
         plt.ylabel('Number of Observations')
         plt.title('Observations of Sheep per Week')
         plt.show()
```



Our scientists know that 15% of sheep at Bryce National Park have foot and mouth disease. Park rangers at Yellowstone National Park have been running a program to reduce the rate of foot and mouth disease at that park. The scientists want to test whether or not this program is working. They want to be able to detect reductions of at least 5 percentage point. For instance, if 10% of sheep in Yellowstone have foot and mouth disease, they'd like to be able to know this, with confidence.

Use the sample size calculator at Optimizely (https://www.optimizely.com/sample-size-calculator/) to calculate the number of sheep that they would need to observe from each park. Use the default level of significance (90%).

Remember that "Minimum Detectable Effect" is a percent of the baseline.

In [34]:

```
'''Baseline = 15
Minimum_detectable_effect = 100*5 percentage points/baseline = 33.33%
Statistical significance = 90%
Sample_Size_Per_Park = 510
## From sample size calculator at Optimizely.'''
```

Out[34]:

```
'Baseline = 15\nMinimum_detectable_effect = 100*5 percentage points/baseline =
33.33%\nStatistical significance = 90%\nSample_Size_Per_Park = 510\n## From sa
mple size calculator at Optimizely.'
```

How many weeks would you need to observe sheep at Bryce National Park in order to observe enough sheep? How many weeks would you need to observe at Yellowstone National Park to observe enough sheep?

In [35]:

```
'''Bryce Weeks Observing = sample_size_per_park/Bryce observations
510/250 = 2.04 weeks
We need to spend 2.04 weeks at Bryce National Park in order to observe enough she
'''Yellowstone Weeks Observing = sample_size_per_park/Yellowstone observations
510/507 = 1.00 week
We need to spend 1 week at Yellowstone National Park in order to observe enough s
```

Out[35]:

```
'Yellowstone Weeks Observing = sample_size_per_park/Yellowstone observations\n
510/507 = 1.00 week\nWe need to spend 1 week at Yellowstone National Park in o
rder to observe enough sheep.'
```

In [ ]:

```
```