Capstone Project

Biodiversity for the National Parks

using.Jupyter_Notebooks() by Michael Kalish

codecademy

Step 1

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

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

In [1]:
```
from matplotlib import pyplot as plt
import pandas as pd
```

Commentary

The module, matplotlib, is imported to visualize data that we will be analyzing.
The module, pandas, is imported initially to import the data from a CSV file.

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 [2]:
```
species = pd.read_csv('species_info.csv')
```

Commentary

Imported the CSV file from my desktop folder, **Biodiversity**: Desktop/biodiversity/biodiversity.ipynb and named the table **species.**

Inspect each DataFrame using `.head()`.

```
In [27]: species.head()
```

Out[27]:

	category	scientific_name	common_names	conservation_status
0	Mammal	Clethrionomys gapperi gapperi	Gapper's Red-Backed Vole	NaN
1	Mammal	Bos bison	American Bison, Bison	NaN
2	Mammal	Bos taurus	Aurochs, Aurochs, Domestic Cattle (Feral), Dom...	NaN
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	NaN
4	Mammal	Cervus elaphus	Wapiti Or Elk	NaN

Commentary

By using head(), I was able to see the top, unsorted (5) rows to get a general introduction to the nature of the dataset.
By reviewing the column names, I can see that the "meat" (pun not intended) of the dataframe is the conservation status for a variety of animals, which are divided into categories (e.g. mammal).
#Each animal has a scientific name and a common name. At first glance, there are holes (i.e. null values) in the dataset in the column, 'conservation_status'.

Section:

Describing the Data

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 [26]:
```
species.scientific_name.nunique()
```

Out[26]:
```
5541
```

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

In [25]:
```
species.category.unique()
```

Out[25]:
```
array(['Mammal', 'Bird', 'Reptile', 'Amphibian', 'Fish', 'Vascular Plan
t',
       'Nonvascular Plant'], dtype=object)
```

What are the different values of `conservation_status`?

In [24]:
```
species.conservation_status.unique()
```

Out[24]:
```
array([nan, 'Species of Concern', 'Endangered', 'Threatened',
       'In Recovery'], dtype=object)
```

Commentary

The initial commands in step three help me quantify the # of rows in the dataframe.
I have learned that there are 5541 unique scientific names for a limited number of categories (e.g. mammal, bird, etc) and conservation status.

Section:

Significant calculations

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 [23]:
```
species.groupby('conservation_status').scientific_name.nunique().reset_index()
```

Out[23]:

	conservation_status	scientific_name
0	Endangered	15
1	In Recovery	4
2	Species of Concern	151
3	Threatened	10

Commentary

Step 4 has helped me identify how many different animals (by scientific name) fit into each conservation status.
A disproportionate amount of the dataset is unavailable in the virtual table that I have created, as the groupby has excluded the null values.

In [59]:
```
species.fillna('No Intervention', inplace=True)
```

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

In [60]:
```
species.groupby('conservation_status').scientific_name.nunique().reset_index()
```

Out[60]:

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

Commentary

I needed to rename the null values so that groupby would include them, and I did this by using .fillna. I have pasted the following code into my notebook: species.fillna('No Intervention', inplace=True)

Ok, that's better. No I can see that the great majority of data requires "No intervention", thank goodness.

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`:

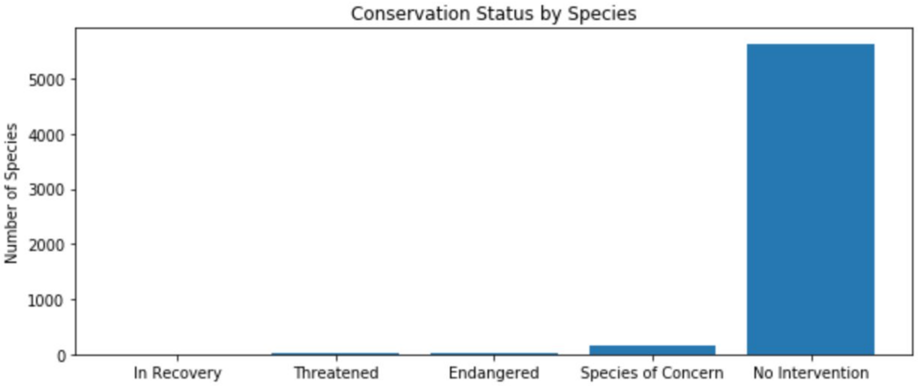
```
protection_counts = species.groupby('conservation_status')\
    .scientific_name.count().reset_index()\
    .sort_values(by='scientific_name')
```

In [30]:
```
protection_counts = species.groupby('conservation_status')\
    .scientific_name.count().reset_index()\
    .sort_values(by='scientific_name')
```

Commentary

Before I begin visualizing my data, I want to create a new dataframe in which the data is represented by the count of each unique scientific name. This will allow me to more easily visualize the relationship between the number of animals, generally, in each conservation status. I accomplish this task by grouby, count and sort_values

In [49]:
```
plt.figure(figsize=(10, 4))
ax = plt.subplot()
plt.bar(range(len(protection_counts)), protection_counts.scientific_name.values)
ax.set_xticks(range(len(protection_counts)))
ax.set_xticklabels(protection_counts.conservation_status.values)
plt.ylabel('Number of Species')
plt.title('Conservation Status by Species')
plt.show()
```



Commentary

The conservation status ‘No intervention’ has made the scale unfriendly for my to compare the other statuses. But it is clear, there are many more animals that are safe from extinction than not.

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 [64]:
```
species['is_protected'] = species.conservation_status !='No Intervention'
```

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

In [68]:
```
category_counts = species.groupby(['category', 'is_protected'])\
                        .scientific_name.nunique().reset_index()
```

Examine `category_counts` using `head()`.

In [69]:
```
print(category_counts.head())
```

```
    category  is_protected  scientific_name
0  Amphibian         False               72
1  Amphibian          True                7
2       Bird         False              413
3       Bird          True               75
4       Fish         False              115
```

Commentary

The data indicates that no fish are threatened.
In contrast, the data indicates that there are few amphibians and many birds that are threatened.
But the data is a bit hard to read and incomplete (I have only selected the top 5 rows). A pivot table would make the data more presentable.

In [76]:
```
category_pivot = category_counts.pivot(columns='is_protected', index='category', values='scientific_name').reset_index(
```

Examine `category_pivot`.

In [77]:
```
category_pivot
```

Out[77]:

is_protected	category	False	True
0	Amphibian	72	7
1	Bird	413	75
2	Fish	115	11
3	Mammal	146	30
4	Nonvascular Plant	328	5
5	Reptile	73	5
6	Vascular Plant	4216	46

Commentary

I have used pivot to consolidate all of the data into a rows. By providing a column for boolean values, the data is presented in a more readable format.

Section:

Recommendations for conservationists

In [78]:
```
category_pivot.columns = ('category','not_protected','protected')
```

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 [84]:
```
ategory_pivot['percent_protected'] = category_pivot.protected/(category_pivot.protected + category_pivot.not_protected)
```

Out[84]:
```
<pandas.core.groupby.groupby.DataFrameGroupBy object at 0x11da5d438>
```

```
Examine `category_pivot`.
```

In [89]:
```
category_pivot.sort_values(by='percent_protected')
```

Out[89]:

	category	not_protected	protected	percent_protected
6	Vascular Plant	4216	46	0.010793
4	Nonvascular Plant	328	5	0.015015
5	Reptile	73	5	0.064103
2	Fish	115	11	0.087302
0	Amphibian	72	7	0.088608
1	Bird	413	75	0.153689
3	Mammal	146	30	0.170455

Commentary

To improve the clarity of the pivot table, I have re-named the column names and introduced a column that calculates the percent protected so I can see which category of animals is at the highest risk. I have chosen to sort the data in the column 'percent_protected" using sort_values to more create additional clarity. It appears that mammals are the most protected and vascular plants are the least, so more energy and resources should be put towards protecting mammals (and birds).

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

Commentary

#**Is the data numerical or categorical?** The data involving percentage of animals (by category) is numerical.
How many pieces of data are you comparing? I am comparing 4 pieces of data: protected/not protected and bird/animal.

In [92]:
```
contingency = [[30,146], [75,413]]
```

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 [93]:
```
from scipy.stats import chi2_contingency
```

```
Now run `chi2_contingency` with `contingency`.
```

In [96]:
```
chi2_contingency(contingency)
```

Out[96]:
```
(0.1617014831654557, 0.6875948096661336, 1, array([[ 27.8313253, 148.1686747],
        [ 77.1686747, 410.8313253]]))
```

It looks like this difference isn't significant!

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

In [97]:
```
contingency = [[5,73],[30,146]]
chi2_contingency(contingency)
```

Out[97]:
```
(4.289183096203645, 0.03835559022969898, 1, array([[ 10.7480315,  67.2519685],
        [ 24.2519685, 151.7480315]]))
```

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

Commentary

The Chi Square Test for mammals and birds resulted in a p-value of 0.68, which is above the 0.05 threshold for significance. This means that the null hypothesis is not rejected.
But when the Chi Square Test was performed for mammals and reptiles, the pvalue was 0.038, which is below 0.05, and so we can reject the null hypothesis, which means we can assume that the percent of protected animals for reptiles and humans are likely dependent upon one another.

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 [98]:
```
observations = pd.read_csv('observations.csv')
observations.head()
```

Out[98]:

	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

Commentary

I have used pandas to load the csv file, "observations.csv" from my desktop folder, **biodiversity**.

In [104]:
```
species['is_sheep'] = species.common_names.apply(lambda x: 'Sheep' in x)
species.head()
```

Out[104]:

	category	scientific_name	common_names	conservation_status	is_protected	is_sheep
0	Mammal	Clethrionomys gapperi gapperi	Gapper's Red-Backed Vole	No Intervention	False	False
1	Mammal	Bos bison	American Bison, Bison	No Intervention	False	False
2	Mammal	Bos taurus	Aurochs, Aurochs, Domestic Cattle (Feral), Dom...	No Intervention	False	False
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True
4	Mammal	Cervus elaphus	Wapiti Or Elk	No Intervention	False	False

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

In [105]:
```
species[species.is_sheep]
```

Out[105]:

	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

Commentary

In effort to filter for all animals that are sheep (by common name), I added an additional column to the table that uses a boolean value (false and true are easy to filter with). In this case, I teased out the term ‘sheep’ from the common_names column.

A concern I have is that there are some things that may have sheep in their name that are not sheet (for example, a dogfish is not a dog). When I reviewed the data, it was evident that there were plants that contained the word, “sheep” in their name.

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 [112]:
```
sheep_species = species[(species.is_sheep) & (species.category == 'Mammal')]
sheep_species
```

Out[112]:

	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
3014	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
4446	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True	True

Commentary

To refine my filter to exclude plants, I was able to add a second criterion for my filter using & to include the category column. I could have created an additional column with a boolean value for mammal, but it was much easier to filter using == ‘mammal’. At first I did not get my any results, but then I noticed that my string needs to be cap sensitive (‘Mammal’).

OBSERVATIONS

	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

SHEEP_SPECIES

	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
3014	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
4446	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True	True

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

In [113]:
```
sheep_observations = observations.merge(sheep_species)
sheep_observations
```

Out[113]:

	scientific_name	park_name	observations	category	common_names	conservation_status	is_protected	is_sheep
0	Ovis canadensis	Yellowstone National Park	219	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
1	Ovis canadensis	Bryce National Park	109	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
2	Ovis canadensis	Yosemite National Park	117	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
3	Ovis canadensis	Great Smoky Mountains National Park	48	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
4	Ovis canadensis sierrae	Yellowstone National Park	67	Mammal	Sierra Nevada Bighorn Sheep	Endangered	True	True
5	Ovis canadensis sierrae	Yosemite National Park	39	Mammal	Sierra Nevada Bighorn Sheep	Endangered	True	True
6	Ovis canadensis sierrae	Bryce National Park	22	Mammal	Sierra Nevada Bighorn Sheep	Endangered	True	True
7	Ovis canadensis sierrae	Great Smoky Mountains National Park	25	Mammal	Sierra Nevada Bighorn Sheep	Endangered	True	True
8	Ovis aries	Yosemite National Park	126	Mammal	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True
9	Ovis aries	Great Smoky Mountains National Park	76	Mammal	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True
10	Ovis aries	Bryce National Park	119	Mammal	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True
11	Ovis aries	Yellowstone National Park	221	Mammal	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True

SHEEP_OBSERVATIONS

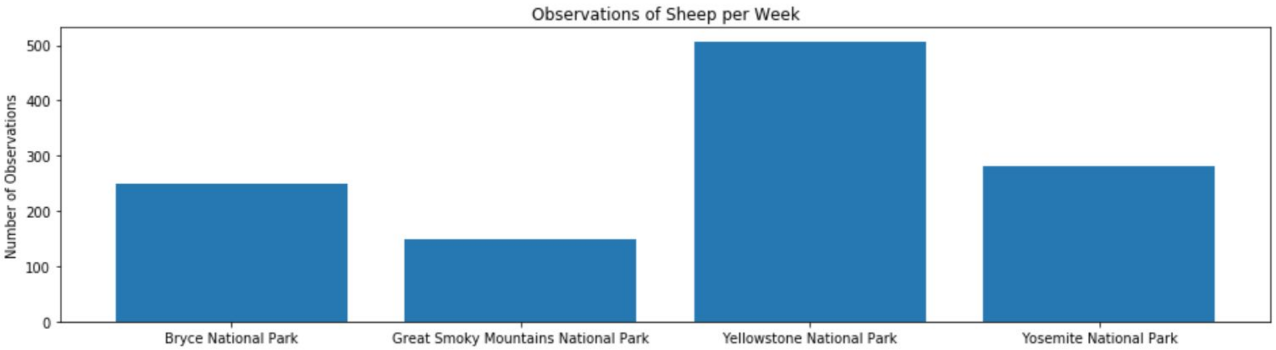
Commentary

To merge my two tables into a single (new) dataframe so that I can see the park_name for where sheep are located, I use the merge function. The table is so scattered it is difficult to see what the data says about sheep for each park.
To better organize the data so that it is more read-able, I have grouped it using groupby to get the sum of observations by park_name.

```
In [116]: obs_by_park = sheep_observations.groupby('park_name').observations.sum().reset_index()
          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

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



Commentary

To better organize the data so that it is more read-able, I have grouped it using groupby to get the sum of observations by park_name.
To visualize the data, I created a bar chart. It is clear that the majority of sheep are located in Yellowstone National Park, followed by Yosemite, then Bryce National Park and lastly Great Smokey Mountains National Park.

Section:

Sample size determination

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 points. 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 Codecademy's 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 [124]:
```
m_d_e = 100*0.05/0.15
m_d_e
baseline = 15
sample_size = 870
```

Out[124]: 33.333333333333336

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 [127]:
```
bryce = 870 / 250
bryce
```

Out[127]: 3.48

In [128]:
```
yellowstone = 810 / 507
yellowstone
```

Out[128]: 1.5976331360946745

In [ ]:
```
#3.5 weeks for Bryce NP and 1.6 weeks for Yellowstone NP
```

Baseline conversion rate: 15 %
Statistical significance: 85% 90% 95%
Minimum detectable effect: 33.3 %
Sample size: 870

Commentary

To calculate the M.D.E., I found the ratio of 5% (the minimum reduction they would like to identify) over the baseline of 15% (the known number at Bryce NP). I multiplied this by 100 to calculate the percentage, 33.33. I then plugged the MDE into the sample size calculator to determine how many sheep I will need to use (using the standard of 90% for statistical significance). I arrived at a need for 870 sheep!

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 [127]: bryce = 870 / 250
          bryce

Out[127]: 3.48

In [128]: yellowstone = 810 / 507
          yellowstone

Out[128]: 1.5976331360946745

In [ ]:   #3.5 weeks for Bryce NP and 1.6 weeks for Yellowstone NP
```

	park_name	observations
0	Bryce National Park	$\boxed{250}$
1	Great Smoky Mountains National Park	149
2	Yellowstone National Park	$\boxed{507}$
3	Yosemite National Park	282

Commentary

To calculate the # of weeks I would need in order to observe this number of sheep, I divided the number of sheep needed by the number of observations made in each park (according to obs_by_park), so I'll need to divide 870 sheep by 250 sheep observed/week for Bryce NP, and by 507 sheep observed/week for Yellowstone, to calculate the number of weeks needed to observe the total number of sheep.

Now go save those animals

Biodiversity for the National Parks

using.Jupyter_Notebooks() by Michael Kalish

codecademy