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 [162…
```python
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 [163…
```python
species = pd.read_csv('species_info.csv')
```

Inspect each DataFrame using `.head()`.

In [164…
```python
species.head()
```

Out[164]:

	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

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 [165…
```python
# Need to use find unique count as some scientific names are repeated.
species_unique = species.scientific_name.nunique()
print(species_unique)

# Repeated scientific names shown in the dataframe below:
species_count = species.groupby('scientific_name').category.count().reset_index()
repeated_species = species_count[species_count.category > 1]
repeated_species
```

```
5541
```

Out[165]:

	scientific_name	category
104	Agrostis capillaris	2
107	Agrostis gigantea	2
111	Agrostis mertensii	2
116	Agrostis scabra	2
118	Agrostis stolonifera	2
...	...	...
5468	Vireo solitarius	2
5481	Vulpia bromoides	2
5484	Vulpia myuros	2
5485	Vulpia octoflora	2
5530	Zizia aptera	2

274 rows × 2 columns

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

In [166…
```python
category_values = species.groupby('category').scientific_name.nunique().reset_index()
category_values
```

Out[166]:

	category	scientific_name
0	Amphibian	79
1	Bird	488
2	Fish	125
3	Mammal	176
4	Nonvascular Plant	333
5	Reptile	78
6	Vascular Plant	4262

What are the different values of `conservation_status`?

In [167…
```python
conservation_values = species.groupby('conservation_status').scientific_name.nunique().reset_index()
conservation_values
```

Out[167]:

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

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 currently 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 [168…
```python
# Done in Step 3.
conservation_values
```

Out[168]:

	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`:

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

In [169…
```python
species.fillna('No Intervention', inplace=True)
```

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

In [170…
```python
conservation_values = species.groupby('conservation_status').scientific_name.nunique().reset_index()
conservation_values
```

Out[170]:

	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`:

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

In [171…
```python
protection_counts = species.groupby('conservation_status')\
    .scientific_name.nunique().reset_index()\
    .sort_values(by='scientific_name')
protection_counts
```

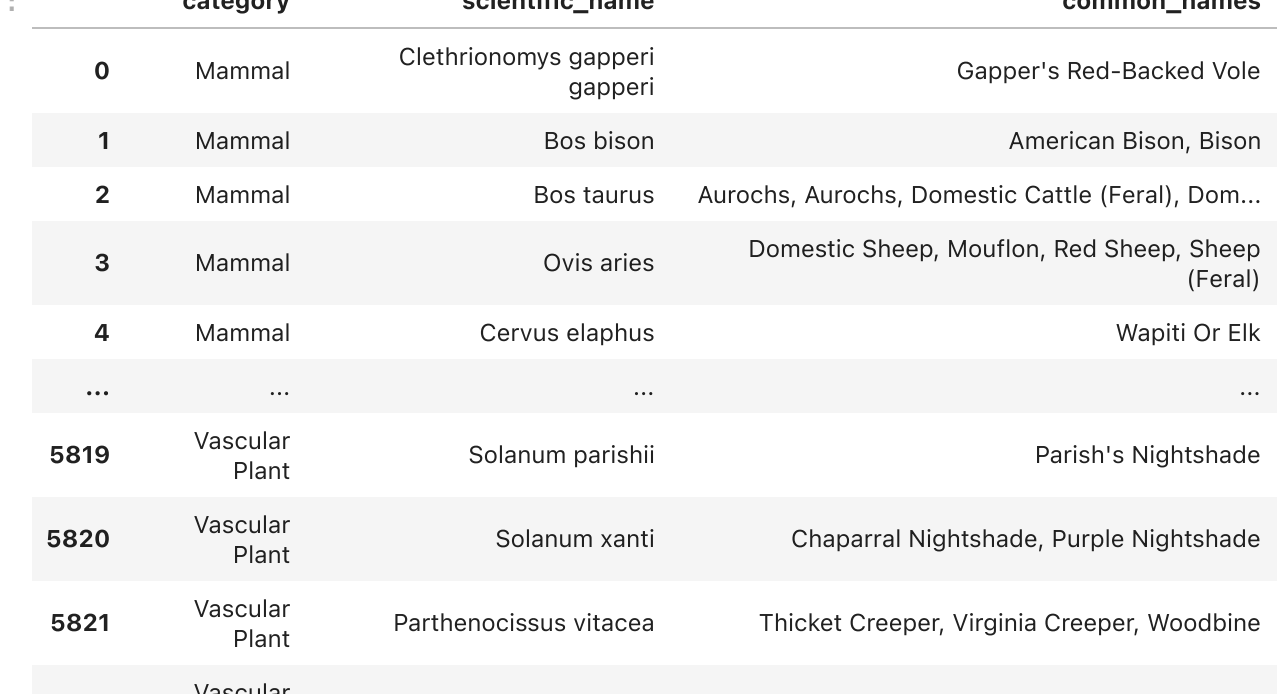
Out[171]:

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

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 [172…
```python
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.values)
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 [173…
```python
species['is_protected'] = species.conservation_status.\
                apply(lambda x: False if x == 'No Intervention' else True)
species
```

Out[173]:

	category	scientific_name	common_names	conservation_status	is_protected
0	Mammal	Clethrionomys gapperi gapperi	Gapper's Red-Backed Vole	No Intervention	False
1	Mammal	Bos bison	American Bison, Bison	No Intervention	False
2	Mammal	Bos taurus	Aurochs, Aurochs, Domestic Cattle (Feral), Dom...	No Intervention	False
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
4	Mammal	Cervus elaphus	Wapiti Or Elk	No Intervention	False
...	...	...	...	...	...
5819	Vascular Plant	Solanum parishii	Parish's Nightshade	No Intervention	False
5820	Vascular Plant	Solanum xanti	Chaparral Nightshade, Purple Nightshade	No Intervention	False
5821	Vascular Plant	Parthenocissus vitacea	Thicket Creeper, Virginia Creeper, Woodbine	No Intervention	False
5822	Vascular Plant	Vitis californica	California Grape, California Wild Grape	No Intervention	False
5823	Vascular Plant	Tribulus terrestris	Bullhead, Caltrop, Goathead, Mexican Sandbur, ...	No Intervention	False

5824 rows × 5 columns

Let's group the `species` data frame by the `category` and `is_protected` columns and count the unique `scientific_name`s in each grouping.

Save your results to `category_counts`.

In [174…
```python
category_counts = species.groupby(['category', 'is_protected']).scientific_name.\
                nunique().reset_index()
```

Examine `category_counts` using `head()`.

In [175…
```python
category_counts.head()
```

Out[175]:

	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

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

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

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

In [176…
```python
category_pivot = category_counts.pivot(
                columns='is_protected',
                index='category',
                values='scientific_name').reset_index()
```

Examine `category_pivot`.

In [177…
```python
category_pivot
```

Out[177]:

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

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 [178…
```python
category_pivot.rename(columns={False:'not_protected', True:'protected'},inplace=True)
category_pivot
```

Out[178]:

is_protected	category	not_protected	protected
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

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 [179…
```python
category_pivot['percent_protected'] = \
    category_pivot.protected /(category_pivot.protected + category_pivot.not_protected)
```

Examine `category_pivot`.

In [180…
```python
category_pivot
```

Out[180]:

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

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 [181…
```python
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:

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

In [182…
```python
from scipy.stats import chi2_contingency
```

Now run `chi2_contingency` with `contingency`.

In [183…
```python
_, pvalue, _, _ = chi2_contingency(contingency)
pvalue

'''
Null Hypothesis: There is no significant difference between mammals and birds
in terms of endangerment.
Alternate Hypothesis: Mammals are more likely to be endangered than birds.
'''

#Since p-value > 0.05, we do not reject the null hypothesis and conclude that mammals are
#not more likely to be endangered than birds.
```

Out[183]:
```
'\nNull Hypothesis: There is no significant difference between mammals and birds \nin terms of endangerment.
\nAlternate Hypothesis: Mammals are more likely to be endangered than birds. \n'
```

It looks like this difference isn't significant!

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

In [184…
```python
contingency2 = [[5, 73], [30, 146]]
_, pvalue2, _, _ = chi2_contingency(contingency2)
pvalue2

#Since p-value < 0.05, we reject the null hypothesis and conclude that there is a
#significant difference between Reptile and Mammal.
```

Out[184]:
```
0.03835559022969898
```

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 [185…
```python
observations = pd.read_csv('observations.csv')
observations.head()
```

Out[185]:

	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 [186…
```python
# Does "Sheep" occur in this string?
str1 = 'This string contains Sheep'
'Sheep' in str1
```

Out[186]:
```
True
```

In [187…
```python
# Does "Sheep" occur in this string?
str2 = 'This string contains Cows'
'Sheep' in str2
```

Out[187]:
```
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 [188…
```python
species['is_sheep'] = species.common_names.apply(lambda x: True if 'Sheep' in x else False)
```

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

In [189…
```python
species[species.is_sheep == True]
```

Out[189]:

	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 [190…
```python
sheep_species = species[(species.category == 'Mammal') & (species.is_sheep == True)]
sheep_species
```

Out[190]:

	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 [191…
```python
sheep_observations = pd.merge(sheep_species, observations, how='left')
sheep_observations
```

Out[191]:

	category	scientific_name	common_names	conservation_status	is_protected	is_sheep	park_name	observations
0	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True	Yosemite National Park	126
1	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True	Great Smoky Mountains National Park	76
2	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True	Bryce National Park	119
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True	Yellowstone National Park	221
4	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True	Yellowstone National Park	219
5	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True	Bryce National Park	109
6	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True	Yosemite National Park	117
7	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True	Great Smoky Mountains National Park	48
8	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True	True	Yellowstone National Park	67
9	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True	True	Yosemite National Park	39
10	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True	True	Bryce National Park	22
11	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	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 [192…
```python
obs_by_park = sheep_observations.groupby('park_name').observations.sum().reset_index()
obs_by_park
```

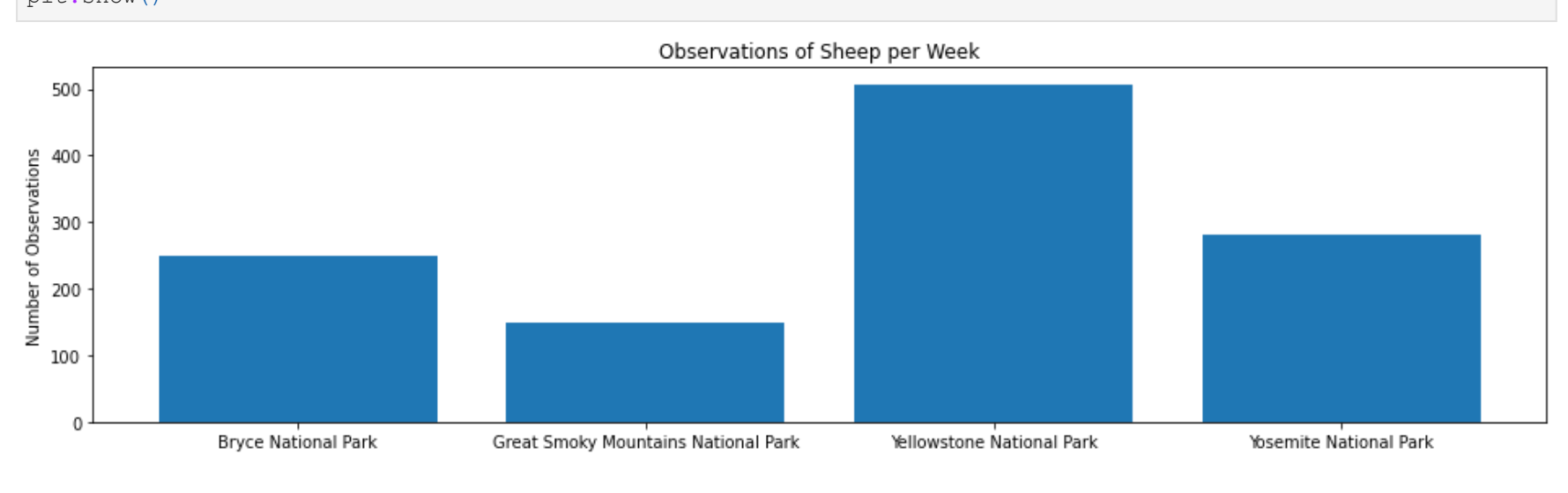
Out[192]:

	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 [193…
```python
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()
```



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 [194…
```python
baseline = 0.15
stats_signif = 0.90
# 5 percentage points: new_baseline = 20%
min_lift = (20-15)/15
#min_lift
sample_size = 870
```

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 [197…
```python
bryce_weeks = sample_size/250
str(bryce_weeks) + ' weeks'
```

Out[197]:
```
'3.48 weeks'
```

In [201…
```python
yellow_weeks = sample_size/507
str(round(yellow_weeks,2)) + ' weeks'
```

Out[201]:
```
'1.72 weeks'
```