Final_Project_With_All_Graphs

July 31, 2020

0.1 Import Necessary Modules

Import any modules that may be useful in data cleaning and graphing.

[40]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import statistics
import seaborn as sns
import scipy as sp
from matplotlib import rcParams
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import statistics as stats
```

0.2 Import Neural Data

Create dataframes for cortex and cerebellum neuron datasets.

We do the following in the code below: - Save cortex neural data to a dataframe and store it in a variable called cortex_df - Save cerebellum neural data to a dataframe and store it in a variable called cerebellum_df

[41]:
```
cortex_df = pd.read_csv('Cortex_Density.csv')
cerebellum_df = pd.read_csv('Cerebellum_Density.csv')
cortex_df.head()
```

[41]:
```
                  Species         Order      Mass, g                  N, n  \
0           Sorex fumeus   Eulipotyphla  0.084±0.009      9,730,000±352,000
1            Mus musculus       Glires  0.173±0.015   13,688,162±2,242,257
2      Blarina brevicauda  Eulipotyphla  0.197±0.012   11,876,000±1,569,000
3  Heterocephalus glaber        Glires  0.184±0.026    6,151,875±1,065,587
4      Condylura cristata  Eulipotyphla 0.420± 0.024 17,250,000± 3,105,000

                  O, n          N/mg          O/mg         O/N
0    9,290,000±1,112,000  116,727±9,387  111,754±18,566  0.958±0.135
1   12,061,838±3,668,594   78,672±7,683   68,643±15,807  0.870±0.177
2   15,820,000±1,158,000   60,214±4,935    80,729±8,731  1.357±0.250
```

```
3    8,398,125±1,197,056   33,374±2,063   45,894±11,497   1.365±0.125
4  32,010,000± 8,822,000   40,777±5,145   76,995±25,019   1.966±0.924
```

[42]:
```
cerebellum_df.head()
```

[42]:
```
                 Species         Order      Mass, g                 N, n  \
0           Sorex fumeus  Eulipotyphla  0.020±0.002   20,870,000±4,660,000
1      Blarina brevicauda  Eulipotyphla  0.037±0.005   33,430,000±5,821,000
2   Heterocephalus glaber        Glires  0.048±0.004   15,742,270±2,849,254
3            Mus musculus        Glires  0.056±0.005   42,219,708±9,277,647
4  Amblysomus hottentotus    Afrotheria        0.084   34,488,379±3,207,000

                 O, n               N/mg            O/mg          O/N
0  5,290,000±2,120,000  1,038,666±214,440  258,073±85,510        0.253
1  4,410,000±1,280,000     919,942±19,721  118,736±25,620        0.132
2  5,482,730±1,274,352     327,280±48,331  115,748±32,952  0.356±0.106
3  6,947,791±1,502,773    746,691±128,541  123,493±25,715  0.165±0.017
4    8,155,621±813,000     409,687±18,667     96,849±5,069  0.236±0.002
```

0.3 Parse the datasets to split on the ± sign

Now we need to parse our datasets and format it in a way that we can analyze. Currently the dataset has a ± symbol for Cortex and Cerebellum Density and Mass. So we split the data on this symbol and exctract the correct value.

Create our Cortex dataset to then store in a dataframe:

[43]:
```
cortex_nmg1 = []
cortex_nmg2 = []
cortex_omg1 = []
cortex_omg2 = []
cortex_on1 = []
cortex_on2 = []

for i in cortex_df['N/mg']:
    cortex_nmg1.append(i.split('±'))

for i in cortex_df['O/mg']:
    cortex_omg1.append(i.split('±'))

for i in cortex_df['O/N']:
    cortex_on1.append(i.split('±'))


cortex_nmg2 = [x[0] for x in cortex_nmg1]
cortex_omg2 = [y[0] for y in cortex_omg1]
cortex_on2 = [z[0] for z in cortex_on1]
```

Create our Cerebellum dataset to then store in a dataframe:

[44]:
```
cerebellum_nmg1 = []
cerebellum_nmg2 = []
cerebellum_omg1 = []
cerebellum_omg2 = []
cerebellum_on1 = []
cerebellum_on2 = []

for i in cerebellum_df['N/mg']:
    cerebellum_nmg1.append(i.split('±'))

for i in cerebellum_df['O/mg']:
    cerebellum_omg1.append(i.split('±'))

for i in cerebellum_df['O/N']:
    cerebellum_on1.append(i.split('±'))


cerebellum_nmg2 = [x[0] for x in cerebellum_nmg1]
cerebellum_omg2 = [y[0] for y in cerebellum_omg1]
cerebellum_on2 = [z[0] for z in cerebellum_on1]
```

0.4 Import Animal Lifespan Data

Create a dataframe for the animal lifespan data.

We do the following in the code below: - Save animal max lifespan data to a variable called animal_lifespans_df - Rename the lifespan dataframe columns to remove special symbols

[45]:
```
animal_lifespans_df = pd.read_excel('animal_max_lifespans.xlsx')
animal_lifespans_df.rename(columns={'Order/Family':'order_family',
                                    'Genus/Species':'scientific_name',
                                    'Common Name':'common_name',
                                    'Wild':'wild',
                                    'Capt.':'captive',
                                    'M/F':'m_f',
                                    'Reference':'reference'}, inplace=True)
animal_lifespans_df.head()
```

[45]:
```
      order_family         scientific_name common_name wild captive  m_f  \
0     Artiodactyla                     NaN         NaN  NaN     NaN  NaN
1  Antilocapridae  Antilocapra americana   Pronghorn   10             x
2                  Antilocapra americana   Pronghorn         11.8    x
3                  Antilocapra americana   Pronghorn   10      12    x
4          Bovidae    Addax nasomaculatus       Addax           19    x
```

```
  reference
0       NaN
1     [100]
2     [143]
3     [192]
4     [100]
```

0.5 Find mean lifespans of each unique species in our dataset

Here we would like to take the average lifespan of every unique species in our dataset for animals in the wild vs. in captivity. There is a lot of variability in the lifespans for one species, and so we decided it would be best to find the mean of the average lifespans for each species.

We do the following in the code below: - Find a list of all of the unique species in our dataframe and save it in a variable called animals - Remove the NaN value at index 0 of the list - For every unique species, find the mean of the wild column for that species - For every unique species, find the mean of the captive column for that species - Save those values in lists called all_wild_means and all_captive_means

```
[46]: animals = animal_lifespans_df.scientific_name.unique().tolist()
animals.pop(0)

all_wild_means = []
all_captive_means = []
for animal in animals:

    animal_df = animal_lifespans_df[animal_lifespans_df['scientific_name'] ==␣
↪animal]
    wild = animal_df['wild'].tolist()

    # there are empty spaces ' ' for some of the values, so we must filter␣
↪those out
    wild = [x for x in wild if x != ' ']

    # some species have no lifetime values at all, in which case we set the␣
↪mean to be 0
    wild_mean = 0
    if len(wild) > 0:
        wild_mean = statistics.mean(wild)
    all_wild_means.append(wild_mean)

    # same logic for computing captive means
    captive = animal_df['captive'].tolist()
    captive = [x for x in captive if x != ' ']
    captive_mean = 0
    if len(captive) > 0:
        captive_mean = statistics.mean(captive)
    all_captive_means.append(captive_mean)
```

0.6 Findings

We find that there are actually very many species in our dataset with no value for their respective wild or captive columns. This data is not useful for our analysis all, so we must filter it out of the dataset. We set the mean for these in the previous step to be 0, and these represent values that are empty in our dataset.

The following code deletes all the species that have no value for their lifespan and saves them as a list of tuples [(species_name, mean_lifespan)].

[47]:
```
# original length of animals with multiple lifespan values
print('Original length of animals with multiple lifespan values: ' +␣
 ↪str(len(animals)))

species_wild_tuple = [(animals[i], all_wild_means[i]) for i in␣
 ↪range(len(animals)) if all_wild_means[i] != 0]
species_captive_tuple = [(animals[i], all_captive_means[i]) for i in␣
 ↪range(len(animals)) if all_captive_means[i] != 0]

# length of animals after means are calculated
print('Length of after means are calculated: ' + str(len(species_wild_tuple)))
print('In total there were ' + str(len(animals) - len(species_wild_tuple)) + '␣
 ↪animals with empty values for their lifetime')
```

```
Original length of animals with multiple lifespan values: 928
Length of after means are calculated: 497
In total there were 431 animals with empty values for their lifetime
```

[48]:
```
wild_lifespans = pd.DataFrame(species_wild_tuple, columns=['species',␣
 ↪'avg_max_lifespan'])
wild_lifespans.head()
```

[48]:
```
                  species  avg_max_lifespan
0   Antilocapra americana              10.0
1      Aepyceros melampus              14.0
2   Alcelaphus buselaphus              20.0
3        Ammotragus lervia             10.0
4  Antidorcas marsupialis              20.0
```

0.7 Find Overlap Between Datasets

Make a list of animals that have data in all three datasets. We want to compare the lifespan data with the cortex data because cortex has fewer species than the cerebellum data.

We do the following in the code below: - Take the species column from the relevant dataframes and assign them to variables named s1 and s2 - Use pandas to find the intersection between s1 and s2 and assign the output to a variable called animals - Transform animals into a list using the .tolist() function

[49]:
```
s1 = pd.Series(wild_lifespans['species'])
s2 = pd.Series(cortex_df['Species'])

animals = s1[s1.isin(s2)]

animals = animals.tolist()
print(animals)
print(len(animals))
```

```
['Giraffa camelopardalis', 'Procavia capensis', 'Blarina brevicauda', 'Tupaia
glis', 'Parascalops breweri', 'Scalopus aquaticus', 'Oryctolagus cuniculus',
'Elephantulus myurus', 'Callithrix jacchus', 'Aotus trivirgatus', 'Cebus
apella', 'Saimiri sciureus', 'Macaca mulatta', 'Macaca radiata', 'Loxodonta
africana', 'Cavia porcellus', 'Mesocricetus auratus', 'Mus musculus', 'Sciurus
carolinensis']
19
```

0.8 Create a New Dataframe

For the sake of organization, want a dataframe that only contains relevant data. We can place the cleaned up data back into a new dataframe which can then be used for analysis.

We do the following in the code below: - Use the intersecting list, animals, in order to loop through cortex_df columns and cerebellum_df columns that contain data that we want to use - Be sure to pull values from the cleaned up lists - Save values to lists that can be used to construct a new dataframe called data_df

[50]:
```
#pull values for overlapping animals
cortex_nmg = []
cortex_on = []
cerebellum_nmg = []
cerebellum_on = []
wild_lifespans_values = []

for i in range(len(cortex_df['Species'])):
    if cortex_df['Species'][i] in animals:
        cortex_nmg.append(cortex_nmg2[i])


for i in range(len(cortex_df['Species'])):
    if cortex_df['Species'][i] in animals:
        cortex_on.append(cortex_on2[i])
```

```
for i in range(len(cerebellum_df['Species'])):
    if cerebellum_df['Species'][i] in animals:
        cerebellum_nmg.append(cerebellum_nmg2[i])

for i in range(len(cerebellum_df['Species'])):
    if cerebellum_df['Species'][i] in animals:
        cerebellum_on.append(cerebellum_on2[i])


for i in range(len(wild_lifespans['species'])):
    if wild_lifespans['species'][i] in animals:
        wild_lifespans_values.append(wild_lifespans['avg_max_lifespan'][i])


print(cortex_on)
print(len(cortex_on))
print(cerebellum_on)
print(len(cerebellum_on))
print(wild_lifespans_values)
```

```
['0.870', '1.357', '2.581', '1.383', '1.032', ' 2.507', '1.417', ' 2.492',
'2.709', '3.566', '1.615', ' 1.883', '1.574', '1.201', '2.237', '2.3', '3.082',
'15.9', '26.844']
19
['0.132', '0.165', '0.07', ' 0.121', '0.11', ' 0.261', '0.108', '0.216',
'0.137', ' 0.336', '0.14', '0.182', '0.073', '0.099', '0.222', '0.205', '0.622',
'0.154']
18
[29.099999999999998, 11.25, 2.5, 12.0, 4.5, 3.0, 12.333333333333334, 1.1, 10.0,
16.0, 40.0, 21.0, 30.0, 20.0, 65.0, 14.8, 6.5, 6.0, 19.866666666666667]
```

[51]:
```
#put all relevant data into its own dataframe
data_df = pd.DataFrame(list(zip(animals, cortex_nmg, cortex_on, cerebellum_nmg,␣
 ↪cerebellum_on, wild_lifespans_values)),
                columns=['species','cortex N/mg', 'cortex O/N', 'cerebellum N/
 ↪mg', 'cerebellum O/N', 'avg_max_lifespan'])


data_df
```

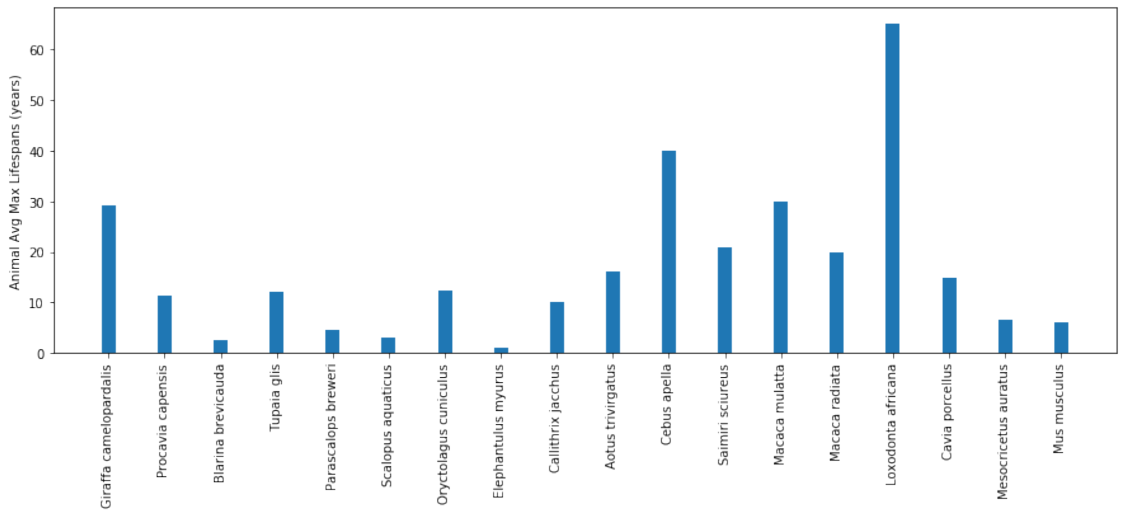
[51]:
```
                    species cortex N/mg cortex O/N cerebellum N/mg  \
0   Giraffa camelopardalis      78,672      0.870         919,942
1        Procavia capensis      60,214      1.357         746,691
2       Blarina brevicauda      36,727      2.581         997,370
3              Tupaia glis      60,461      1.383         424,002
4      Parascalops breweri      54,644      1.032       1,037,390
```

```
5       Scalopus aquaticus      39,099      2.507         531,494
6     Oryctolagus cuniculus     42,900      1.417         571,460
7      Elephantulus myurus      22,508      2.492         339,755
8       Callithrix jacchus      28,384      2.709         494,970
9        Aotus trivirgatus      16,063      3.566         392,363
10            Cebus apella      44,280      1.615         605,080
11        Saimiri sciureus      19,134      1.883         242,415
12          Macaca mulatta      41,990      1.574         424,000
13          Macaca radiata      64,930      1.201         540,310
14      Loxodonta africana      29,180      2.237         354,655
15         Cavia porcellus      34,298        2.3         590,800
16    Mesocricetus auratus      24,470      3.082         131,080
17            Mus musculus       4,339       15.9         213,983

   cerebellum O/N  avg_max_lifespan
0           0.132         29.100000
1           0.165         11.250000
2            0.07          2.500000
3           0.121         12.000000
4            0.11          4.500000
5           0.261          3.000000
6           0.108         12.333333
7           0.216          1.100000
8           0.137         10.000000
9           0.336         16.000000
10           0.14         40.000000
11          0.182         21.000000
12          0.073         30.000000
13          0.099         20.000000
14          0.222         65.000000
15          0.205         14.800000
16          0.622          6.500000
17          0.154          6.000000
```

0.9 Average life span of each species

Here we see a bar chart displaying the average maximum lifespan among all of the species in the intersection of both datasets.

[52]:
```python
fig = plt.figure(figsize=(15,5))

N = len(data_df['species'])
X = np.arange(N)
plt.bar(X, data_df['avg_max_lifespan'], 0.25)

plt.xticks(np.arange(N), data_df['species'], rotation=90)
plt.ylabel('Animal Avg Max Lifespans (years)')
plt.show()
```

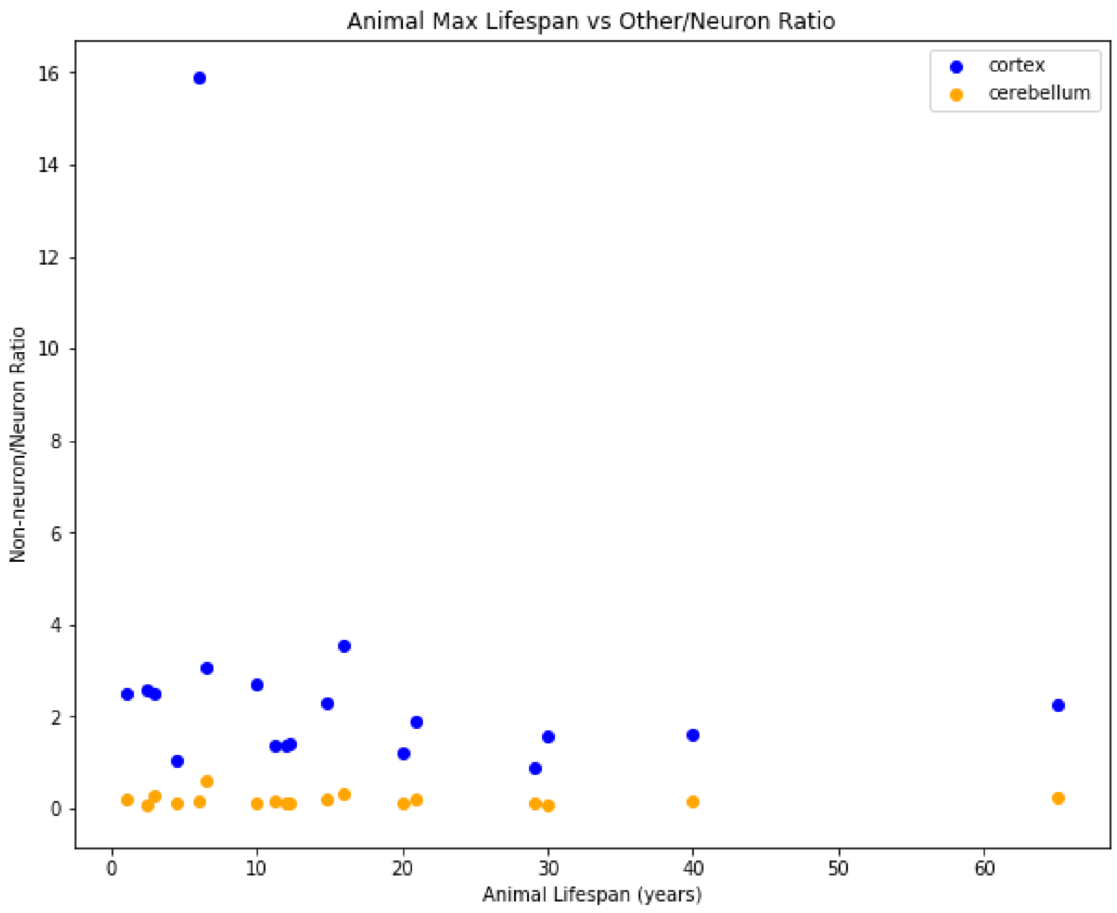


0.10 Plot Lifespan vs Non-Neuron/Neuron Ratios (O/N)

Compare the average max lifespans to the non-neuron/neuron ratio, listed in the data as "O/N", for both cortex and cerebellum. Visualize this with a scatterplot and compute p-values.

We do the following in the code below: - Create a scatterplot for lifespans vs O/N values - Be sure to include labels - Run sp.stats.pearsonr() to find p-values for any correlation in the cortex and cerebellum data and assign the outputs to r_cortex, p_cortex and r_cerebellum, p_cerebellum respectively

[53]:
```
fig = plt.figure(figsize=(10,8))

plt.scatter(pd.to_numeric(data_df['avg_max_lifespan']), pd.
 ↪to_numeric(data_df["cortex O/N"]), color='blue', label='cortex')
plt.scatter(pd.to_numeric(data_df['avg_max_lifespan']), pd.
 ↪to_numeric(data_df["cerebellum O/N"]), color='orange', label='cerebellum')

plt.title('Animal Max Lifespan vs Other/Neuron Ratio')
plt.xlabel('Animal Lifespan (years)')
plt.ylabel('Non-neuron/Neuron Ratio')
plt.legend()
plt.show()
```



```
[54]: r_cortex_on, p_cortex_on = sp.stats.pearsonr(pd.
       ↪to_numeric(data_df['avg_max_lifespan'].tolist()), pd.
       ↪to_numeric(data_df["cortex O/N"].tolist()))
      print(r_cortex, p_cortex)
```

```
-0.2099173758845573 0.40313166730908334
```

```
[55]: r_cerebellum_on, p_cerebellum_on = sp.stats.pearsonr(pd.
       ↪to_numeric(data_df['avg_max_lifespan'].tolist()), pd.
       ↪to_numeric(data_df["cerebellum O/N"].tolist()))
      print(r_cerebellum, p_cerebellum)
```

```
-0.1114654992393401 0.6596951614468209
```

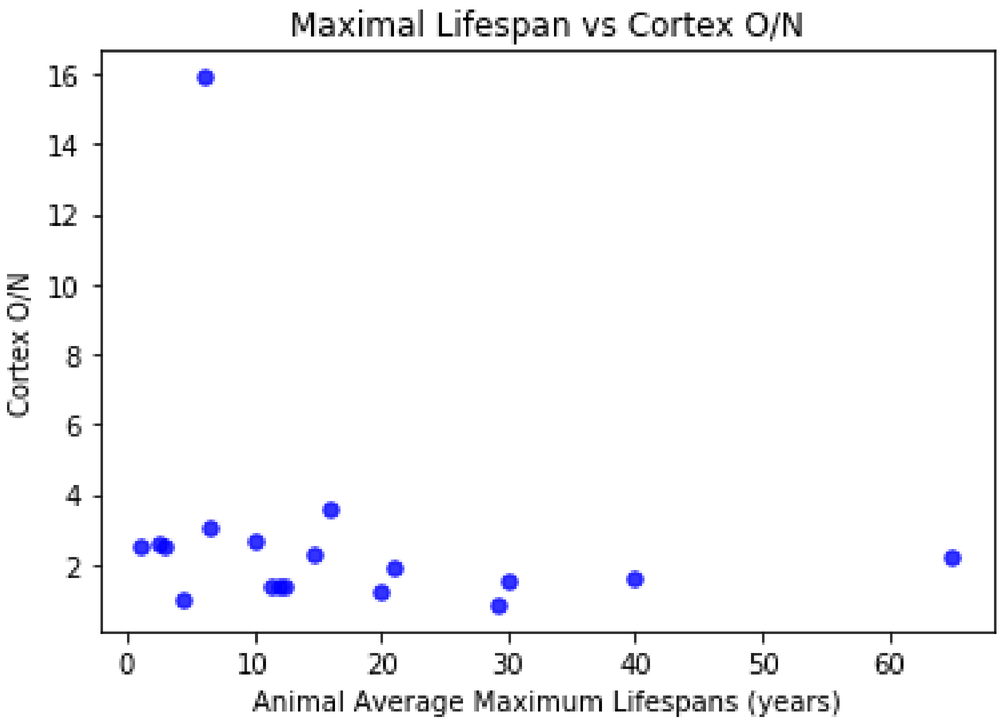
0.11 P-Values are Not Significant

The p-values of 0.40 and 0.66 for cortex O/N and cerebellum O/N respectively cannot be considered significant, indicating that there is no correlation between the max lifespan of these species and their non-neuron/neuron ratio in either of the considered brain regions.

0.12 Cortex/Cerebellum O/N vs. Lifespan

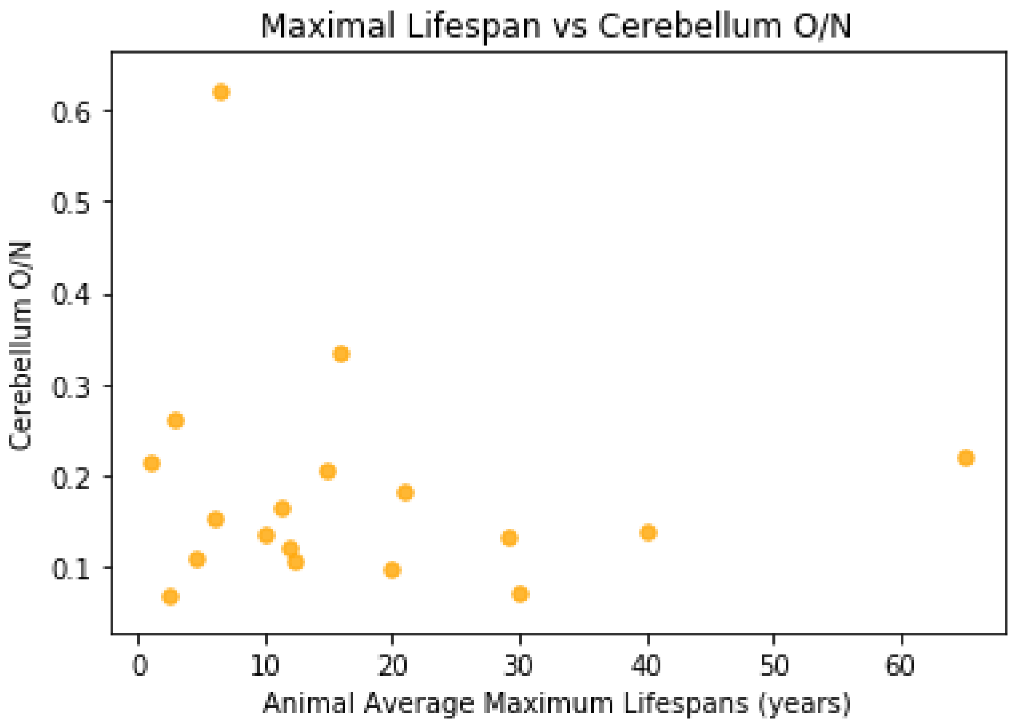
The two graphs below are scatter plots of Cortex O/N versus maximal lifespan and cerebellum O/N vs lifespan, respectively. For both graphs, "O/N" refers to other cells per neuron.

```
[56]: cortex_mass = data_df['cortex O/N'].tolist()
      cortex_mass = pd.DataFrame([float(x) for x in cortex_mass])

      fig = plt.figure()

      plt.scatter(data_df['avg_max_lifespan'], cortex_mass, alpha=0.8, color='blue',␣
       ↪s=30, linewidth=1)
      plt.xlabel('Animal Average Maximum Lifespans (years)')
      plt.ylabel('Cortex O/N')
      plt.title('Maximal Lifespan vs Cortex O/N')
      plt.show()
```



Above we see the scatterplot of Cortex O/N vs the species' maximum lifespan and below is the plot of Cerebellum O/N vs the species' maximum lifespan.

[57]:
```
cerebellum_mass = data_df['cerebellum O/N']
cerebellum_mass = pd.DataFrame([float(x) for x in cerebellum_mass])
fig = plt.figure()

plt.scatter(data_df['avg_max_lifespan'], cerebellum_mass, alpha=0.8,␣
 ↪color='orange', s=30, linewidth=1)
plt.xlabel('Animal Average Maximum Lifespans (years)')
plt.ylabel('Cerebellum O/N')
plt.title('Maximal Lifespan vs Cerebellum O/N')
plt.show()
```



1 Cortex and Cerebellum Density (N/mg) vs Lifespan

The two graphs below illustrate the comparison between neuron density in both the cortex and cerebellum with the selected species maximal lifespans.

- Start by placing the data frame columns into lists and then remove any blank spaces ”_”.
- Sort the data for plotting

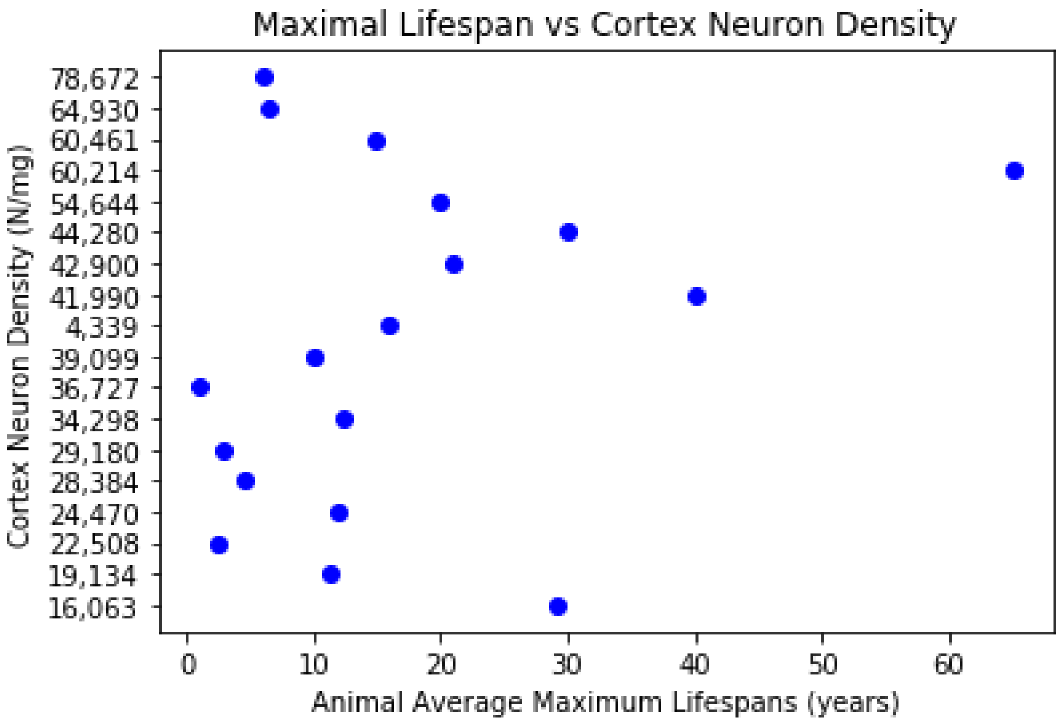
[58]:
```
data_list = []
data_sorted = []
data_list = data_df['cortex N/mg'].tolist()
```

```
counter = 0

for i in data_list:
    data_list[counter] = i.strip()
    counter += 1

data_sorted = sorted(data_list)
```

[59]:
```
data_list_2 = []
data_sorted_2 = []
data_list_2 = data_df['cerebellum N/mg'].tolist()
counter_2 = 0

for i in data_list_2:
    data_list_2[counter_2] = i.strip()
    counter_2 += 1

data_sorted_2 = sorted(data_list_2)
```

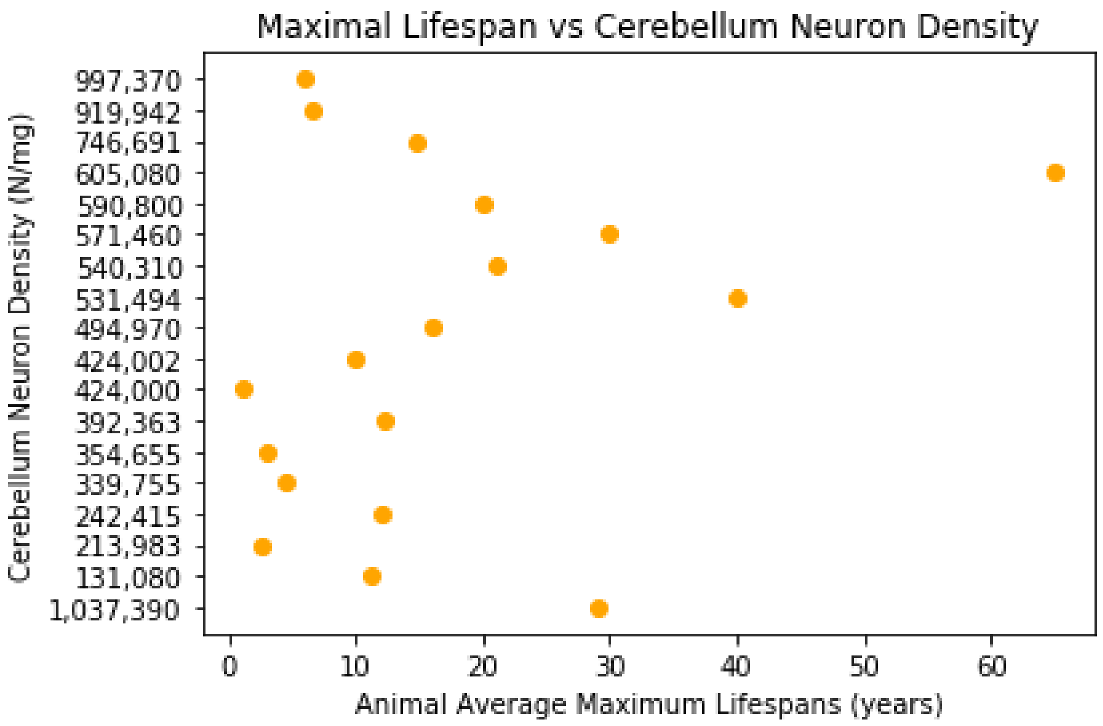
- Plot the Data as a Bar Graph to see how an increase in cortex neuron density relates to an increase in maximal lifespans or if there is no relationship.

[60]:
```
plt.scatter(data_df['avg_max_lifespan'], data_sorted, color='blue')
plt.xlabel('Animal Average Maximum Lifespans (years)')
plt.ylabel('Cortex Neuron Density (N/mg)')
plt.title('Maximal Lifespan vs Cortex Neuron Density')
plt.show()
```



- Plot the Data as a Bar Graph to see how an increase in cerebellum neuron density relates to an increase in maximal lifespans or if there is no relationship.

[61]:
```
plt.scatter(data_df['avg_max_lifespan'], data_sorted_2, color='orange')
plt.xlabel('Animal Average Maximum Lifespans (years)')
plt.ylabel('Cerebellum Neuron Density (N/mg)')
plt.title('Maximal Lifespan vs Cerebellum Neuron Density')
plt.show()
```



2 Conclusion

In this experiment, data was collected from two separate sources and then processed so that it could be visually presented in a meaningful way in order to determine if there is a correlation between the information. The focus was on mammals because mammals with larger brains relative to their body size have been suggested to have longer life spans, perhaps because of their increased capacity for learning and behavioral flexibility (Universitat Autonoma de Barcelona, 2010). The maximal lifespans of mammals from one database and the neuron vs. non-neuronal cell densities from another database were scanned in order to determine what information between the two databases overlapped. After finding species that had comparable data and graphing the corresponding statistics, it was possible to analyze them for a relationship. Unfortunately, it appears that there is no direct correlation between the cortex and cerebellum neuron/non-neuronal cell ratio and maximal lifespan, or the cortex and cerebellum neuron density and maximal lifespan. However, there does appear to be a trend of convergence around the cortex O/N ratio at 2.0 where maximal lifespan reaches its optimal point, which could imply that this ratio constitutes a form of maximum efficiency. For future studies, it is recommended to utilize alternative data, preferably containing a different measure of animal lifespan such as a median or mean and to be able to measure variance among the datasets.

```
[ ]:
```