Cpts_575: Data Science

Assignment III

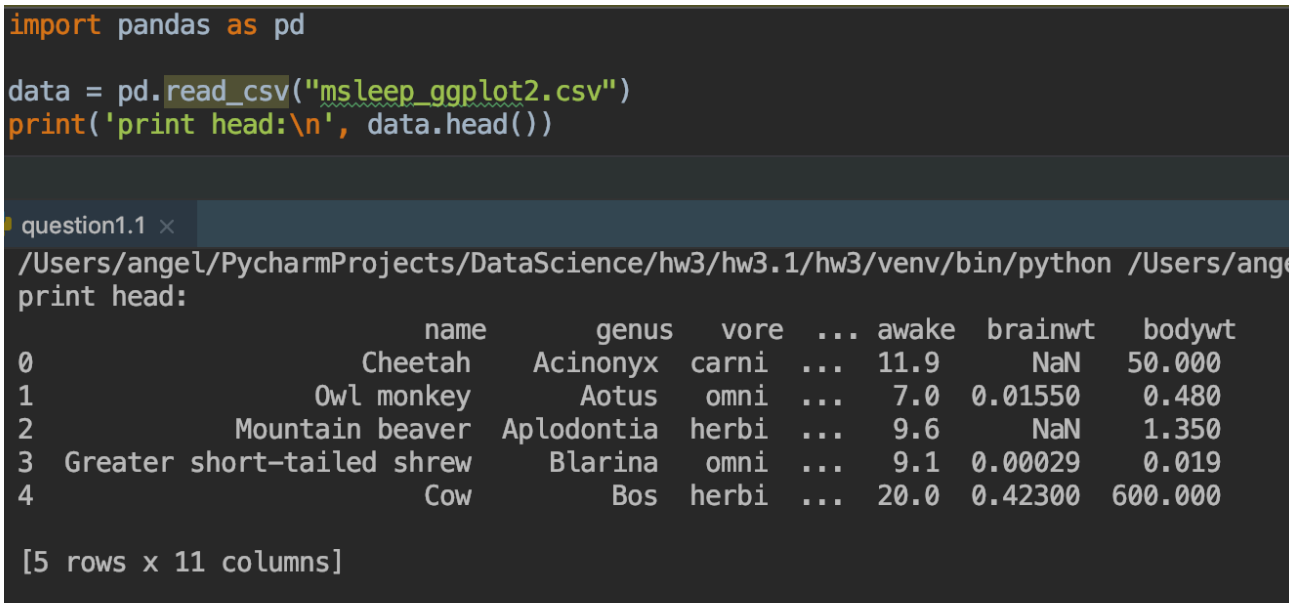
09/19/2019

Instructor: Assefaw Gebremedhin

Student: Yu-Chieh Wang 11641327

1. msleep_ggplot2

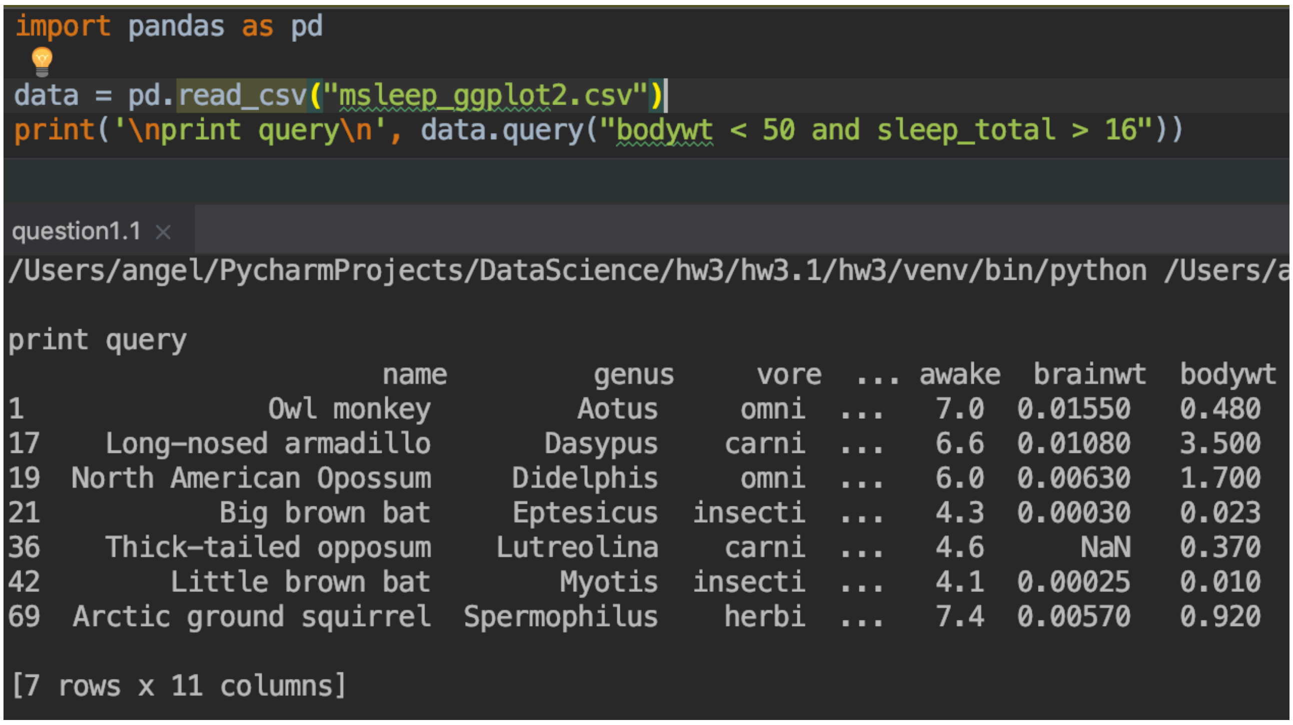
-print the first few values of the columns.



```
import pandas as pd

data = pd.read_csv("msleep_ggplot2.csv")
print('print head:\n', data.head())
```

```
question1.1
/Users/angel/PycharmProjects/DataScience/hw3/hw3.1/hw3/venv/bin/python /Users/ange
print head:
                          name       genus   vore  ... awake  brainwt   bodywt
0                     Cheetah   Acinonyx  carni  ...  11.9      NaN   50.000
1                   Owl monkey      Aotus   omni  ...   7.0  0.01550    0.480
2              Mountain beaver  Aplodontia  herbi  ...   9.6      NaN    1.350
3   Greater short-tailed shrew     Blarina   omni  ...   9.1  0.00029    0.019
4                          Cow         Bos  herbi  ...  20.0  0.42300  600.000

[5 rows x 11 columns]
```

(A) **Count the number of animals which weight under 50 kilograms and sleep more than 16 hours a day.**

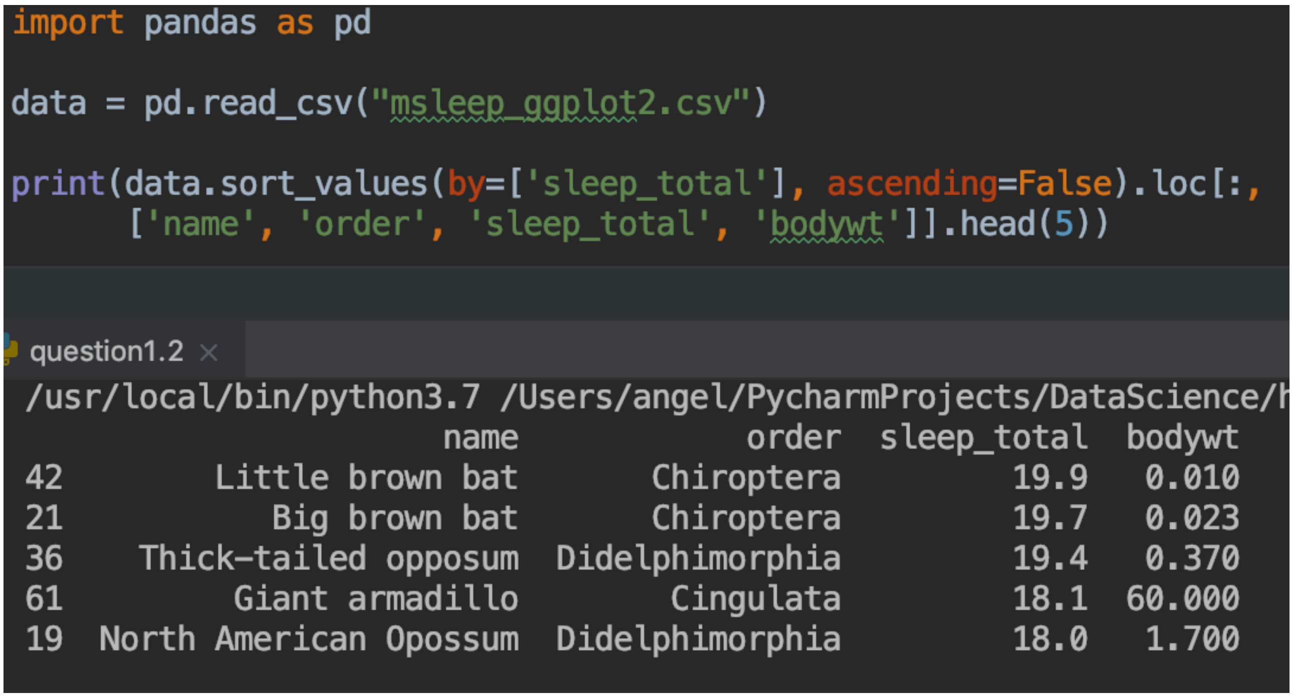


```
import pandas as pd

data = pd.read_csv("msleep_ggplot2.csv")
print('\nprint query\n', data.query("bodywt < 50 and sleep_total > 16"))
```

```
question1.1
/Users/angel/PycharmProjects/DataScience/hw3/hw3.1/hw3/venv/bin/python /Users/a

print query
                          name        genus     vore  ... awake  brainwt  bodywt
1                   Owl monkey        Aotus     omni  ...   7.0  0.01550   0.480
17       Long-nosed armadillo      Dasypus    carni  ...   6.6  0.01080   3.500
19   North American Opossum    Didelphis     omni  ...   6.0  0.00630   1.700
21              Big brown bat   Eptesicus  insecti  ...   4.3  0.00030   0.023
36       Thick-tailed opposum   Lutreolina    carni  ...   4.6      NaN   0.370
42            Little brown bat       Myotis  insecti  ...   4.1  0.00025   0.010
69   Arctic ground squirrel  Spermophilus    herbi  ...   7.4  0.00570   0.920

[7 rows x 11 columns]
```

-It has 7 animals match the conditions.

(B) **Print the name, order, sleep time and bodyweight of the animals with the 5 longest sleep times, in order of sleep time.**

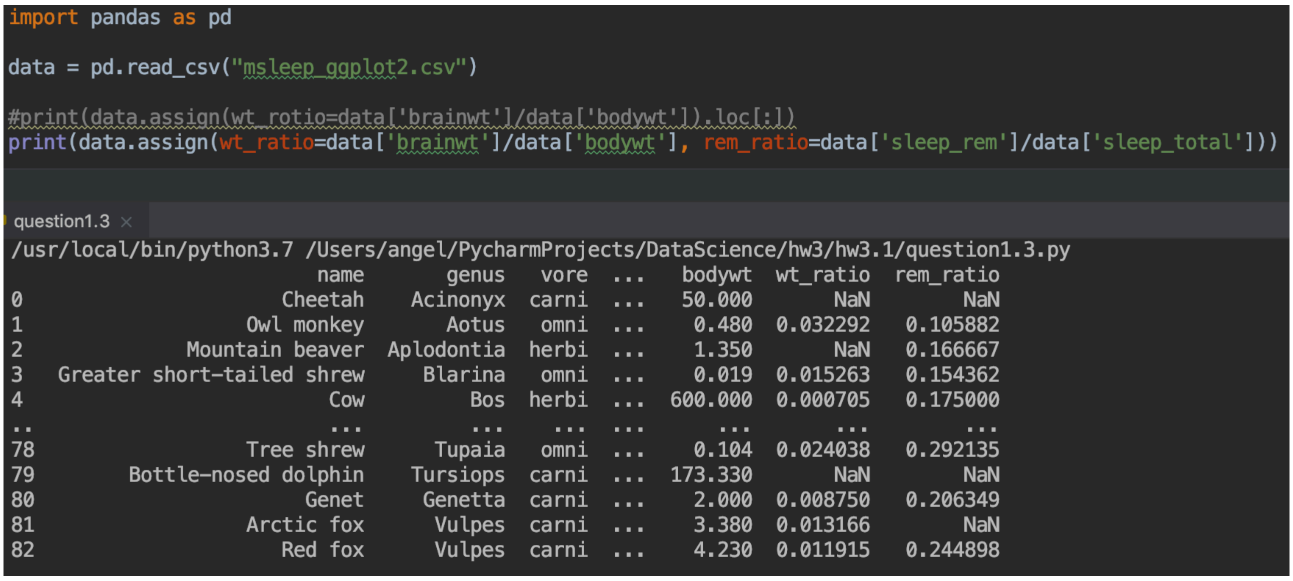


```
import pandas as pd

data = pd.read_csv("msleep_ggplot2.csv")

print(data.sort_values(by=['sleep_total'], ascending=False).loc[:,
      ['name', 'order', 'sleep_total', 'bodywt']].head(5))
```

question1.2

```
/usr/local/bin/python3.7 /Users/angel/PycharmProjects/DataScience/h
                      name            order  sleep_total  bodywt
42         Little brown bat       Chiroptera         19.9   0.010
21            Big brown bat       Chiroptera         19.7   0.023
36     Thick-tailed opposum  Didelphimorphia         19.4   0.370
61          Giant armadillo        Cingulata         18.1  60.000
19  North American Opossum  Didelphimorphia         18.0   1.700
```

(C) **Add two new columns to the dataframe; wt_ratio with the ratio of brain size to body weight, rem_ratio with the ratio of rem sleep to sleep time. If you think they might be useful, feel free to extract more features than these, and describe what they are.**

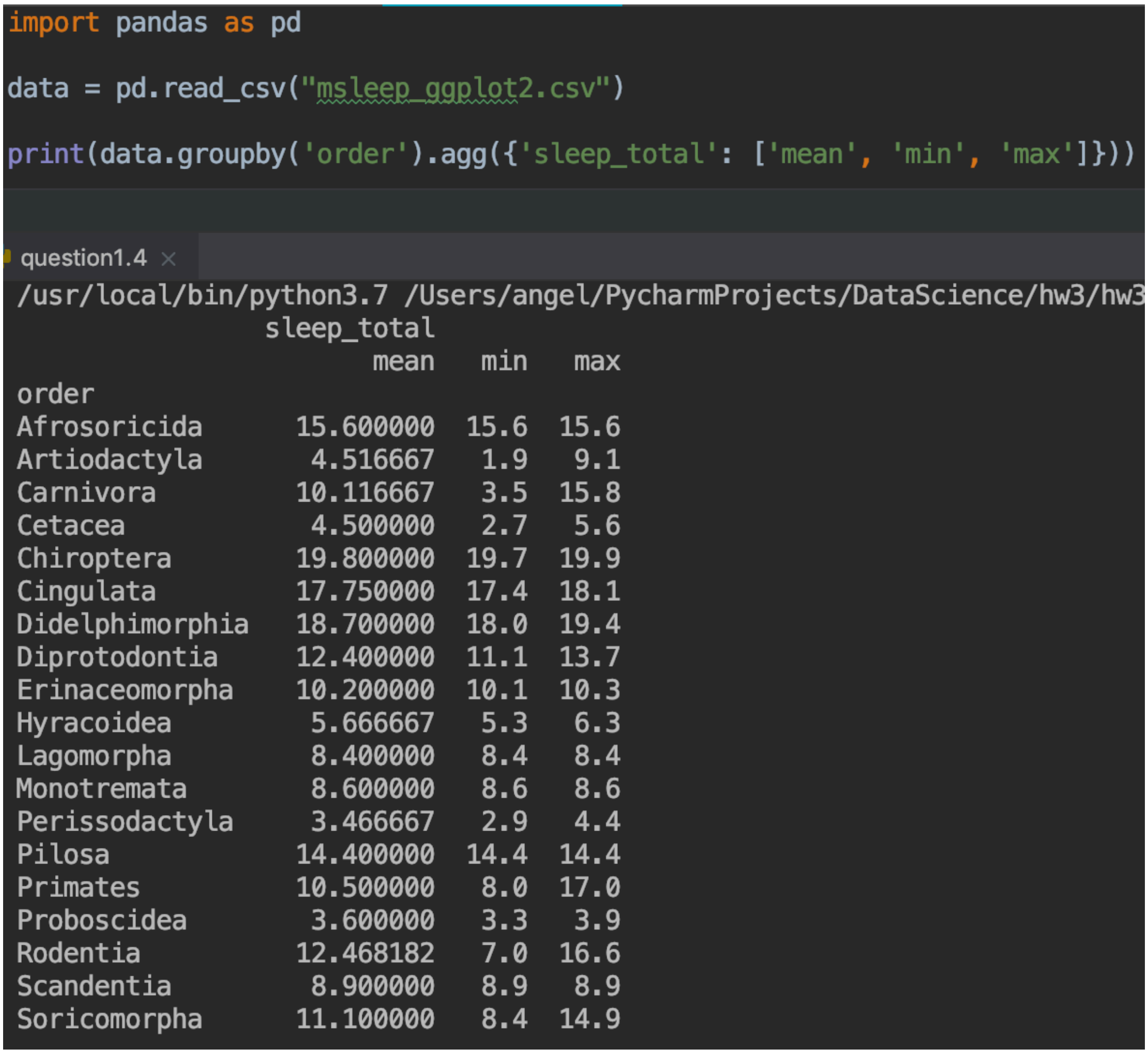


```
import pandas as pd

data = pd.read_csv("msleep_ggplot2.csv")

#print(data.assign(wt_rotio=data['brainwt']/data['bodywt']).loc[:])
print(data.assign(wt_ratio=data['brainwt']/data['bodywt'], rem_ratio=data['sleep_rem']/data['sleep_total']))
```

question1.3

```
/usr/local/bin/python3.7 /Users/angel/PycharmProjects/DataScience/hw3/hw3.1/question1.3.py
                          name       genus   vore  ...   bodywt  wt_ratio  rem_ratio
0                      Cheetah    Acinonyx  carni  ...   50.000       NaN        NaN
1                   Owl monkey       Aotus   omni  ...    0.480  0.032292   0.105882
2              Mountain beaver  Aplodontia  herbi  ...    1.350       NaN   0.166667
3   Greater short-tailed shrew     Blarina   omni  ...    0.019  0.015263   0.154362
4                          Cow         Bos  herbi  ...  600.000  0.000705   0.175000
..                         ...         ...    ...  ...      ...       ...        ...
78                  Tree shrew      Tupaia   omni  ...    0.104  0.024038   0.292135
79        Bottle-nosed dolphin    Tursiops  carni  ...  173.330       NaN        NaN
80                       Genet     Genetta  carni  ...    2.000  0.008750   0.206349
81                  Arctic fox      Vulpes  carni  ...    3.380  0.013166        NaN
82                     Red fox      Vulpes  carni  ...    4.230  0.011915   0.244898
```

(D) **Display the average, min and max sleep times for each order.**



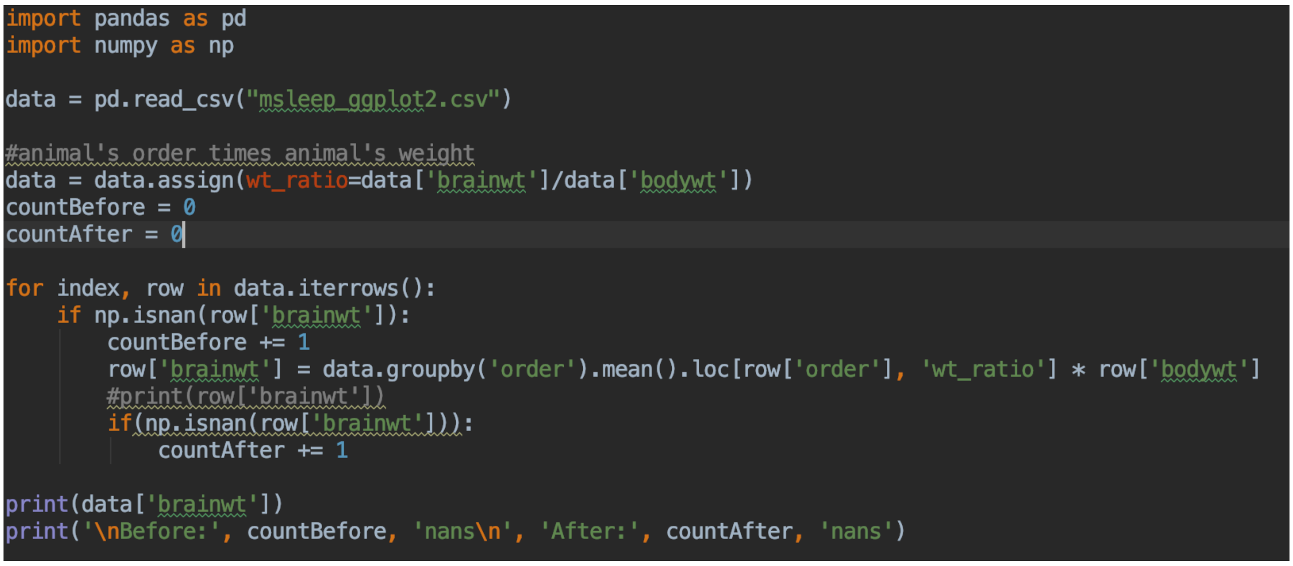
```
import pandas as pd

data = pd.read_csv("msleep_ggplot2.csv")

print(data.groupby('order').agg({'sleep_total': ['mean', 'min', 'max']}))
```

question1.4

```
/usr/local/bin/python3.7 /Users/angel/PycharmProjects/DataScience/hw3/hw3
                sleep_total
                       mean   min   max
order
Afrosoricida      15.600000  15.6  15.6
Artiodactyla       4.516667   1.9   9.1
Carnivora         10.116667   3.5  15.8
Cetacea            4.500000   2.7   5.6
Chiroptera        19.800000  19.7  19.9
Cingulata         17.750000  17.4  18.1
Didelphimorphia   18.700000  18.0  19.4
Diprotodontia     12.400000  11.1  13.7
Erinaceomorpha    10.200000  10.1  10.3
Hyracoidea         5.666667   5.3   6.3
Lagomorpha         8.400000   8.4   8.4
Monotremata        8.600000   8.6   8.6
Perissodactyla     3.466667   2.9   4.4
Pilosa            14.400000  14.4  14.4
Primates          10.500000   8.0  17.0
Proboscidea        3.600000   3.3   3.9
Rodentia          12.468182   7.0  16.6
Scandentia         8.900000   8.9   8.9
Soricomorpha      11.100000   8.4  14.9
```

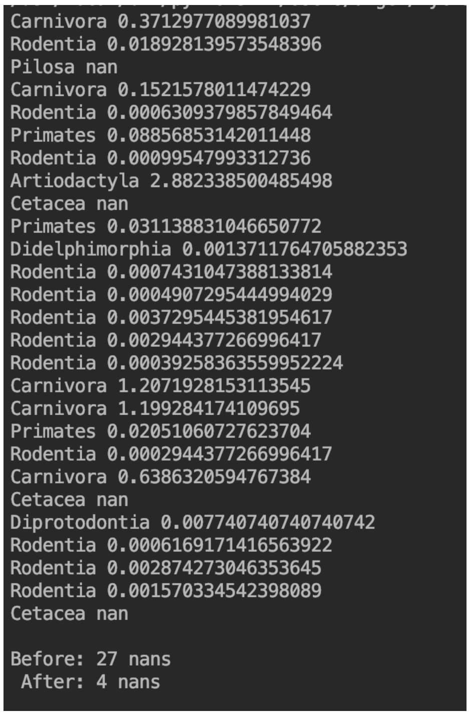
(E) **Impute the missing brain weights as the average wt_ratio for that animal's order times the animal's weight. Make a second copy of your dataframe, but this time impute missing brain weights with the average brain weight for that animal's order. What assumptions do these data filling methods make? Which is the best way to impute the data, or do you see a better way, and why? You may impute or remove other variables as you find appropriate. Briefly explain your decisions.**

-First, impute the missing brain weights as the average wt_ratio for the animal's order times the animal's weight.



```python
import pandas as pd
import numpy as np

data = pd.read_csv("msleep_ggplot2.csv")

#animal's order times animal's weight
data = data.assign(wt_ratio=data['brainwt']/data['bodywt'])
countBefore = 0
countAfter = 0

for index, row in data.iterrows():
    if np.isnan(row['brainwt']):
        countBefore += 1
        row['brainwt'] = data.groupby('order').mean().loc[row['order'], 'wt_ratio'] * row['bodywt']
        #print(row['brainwt'])
        if(np.isnan(row['brainwt'])):
            countAfter += 1

print(data['brainwt'])
print('\nBefore:', countBefore, 'nans\n', 'After:', countAfter, 'nans')
```

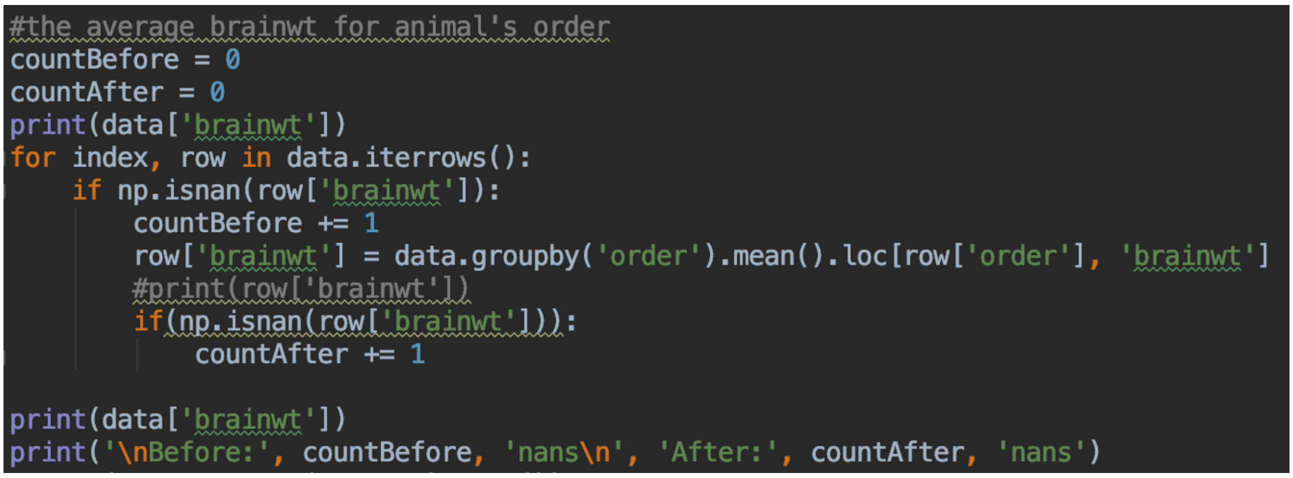
-The result is as following:



```
Carnivora 0.3712977089981037
Rodentia 0.018928139573548396
Pilosa nan
Carnivora 0.1521578011474229
Rodentia 0.0006309379857849464
Primates 0.08856853142011448
Rodentia 0.00099547993312736
Artiodactyla 2.882338500485498
Cetacea nan
Primates 0.031138831046650772
Didelphimorphia 0.0013711764705882353
Rodentia 0.0007431047388133814
Rodentia 0.0004907295444994029
Rodentia 0.0037295445381954617
Rodentia 0.002944377266996417
Rodentia 0.00039258363559952224
Carnivora 1.2071928153113545
Carnivora 1.199284174109695
Primates 0.02051060727623704
Rodentia 0.0002944377266996417
Carnivora 0.6386320594767384
Cetacea nan
Diprotodontia 0.007740740740740742
Rodentia 0.0006169171416563922
Rodentia 0.002874273046353645
Rodentia 0.001570334542398089
Cetacea nan

Before: 27 nans
 After: 4 nans
```

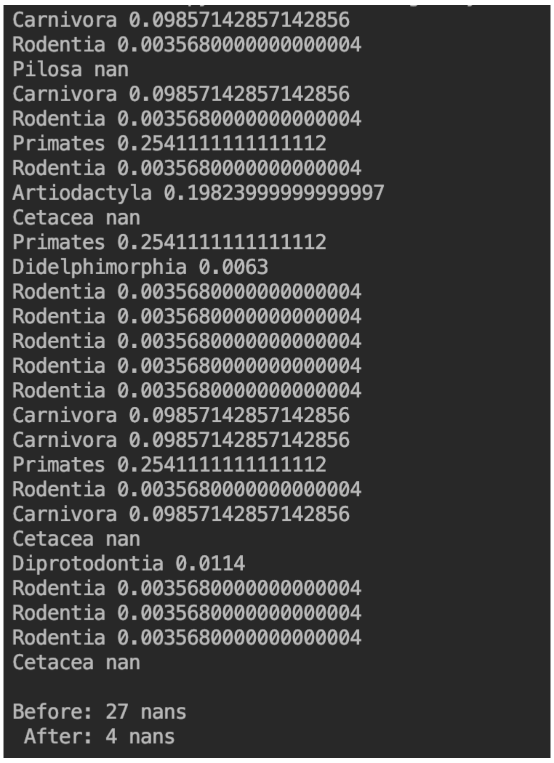
-We have 27 missing brain weights before, but 4missing brain weights after using the impute method.

- Next, impute missing brain weights with the average brain weight for that animal's order.



```
#the average brainwt for animal's order
countBefore = 0
countAfter = 0
print(data['brainwt'])
for index, row in data.iterrows():
    if np.isnan(row['brainwt']):
        countBefore += 1
        row['brainwt'] = data.groupby('order').mean().loc[row['order'], 'brainwt']
        #print(row['brainwt'])
        if(np.isnan(row['brainwt'])):
            countAfter += 1

print(data['brainwt'])
print('\nBefore:', countBefore, 'nans\n', 'After:', countAfter, 'nans')
```

-The result is as following:



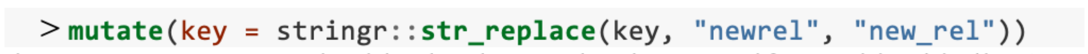
```
Carnivora 0.09857142857142856
Rodentia 0.0035680000000000004
Pilosa nan
Carnivora 0.09857142857142856
Rodentia 0.0035680000000000004
Primates 0.2541111111111112
Rodentia 0.0035680000000000004
Artiodactyla 0.19823999999999997
Cetacea nan
Primates 0.2541111111111112
Didelphimorphia 0.0063
Rodentia 0.0035680000000000004
Rodentia 0.0035680000000000004
Rodentia 0.0035680000000000004
Rodentia 0.0035680000000000004
Rodentia 0.0035680000000000004
Carnivora 0.09857142857142856
Carnivora 0.09857142857142856
Primates 0.2541111111111112
Rodentia 0.0035680000000000004
Carnivora 0.09857142857142856
Cetacea nan
Diprotodontia 0.0114
Rodentia 0.0035680000000000004
Rodentia 0.0035680000000000004
Rodentia 0.0035680000000000004
Cetacea nan

Before: 27 nans
 After: 4 nans
```

-Analysis:

First of all, Both methods can give most of the missing data a new value. However, they cannot solve the missing problem totally because their calculations are related to the "brainwt". If other animals with the same order don't have the value of "brainwt", all of their "brainwt" cannot be calculated. Next, to compare the two methods, the first one is better because it is related to "bodywt". In my opinion, I don't believe animals in the same order have the same brain weight, so I cannot accept the second method. Finally, I think that some values in column "brainwt" are "nan" can be accept. If we try to fill these empty space in order to complete the form, the authenticity of the data will be lost.

2. Tuberculosis cases (Tidy data)

(A) **Explain why this line**



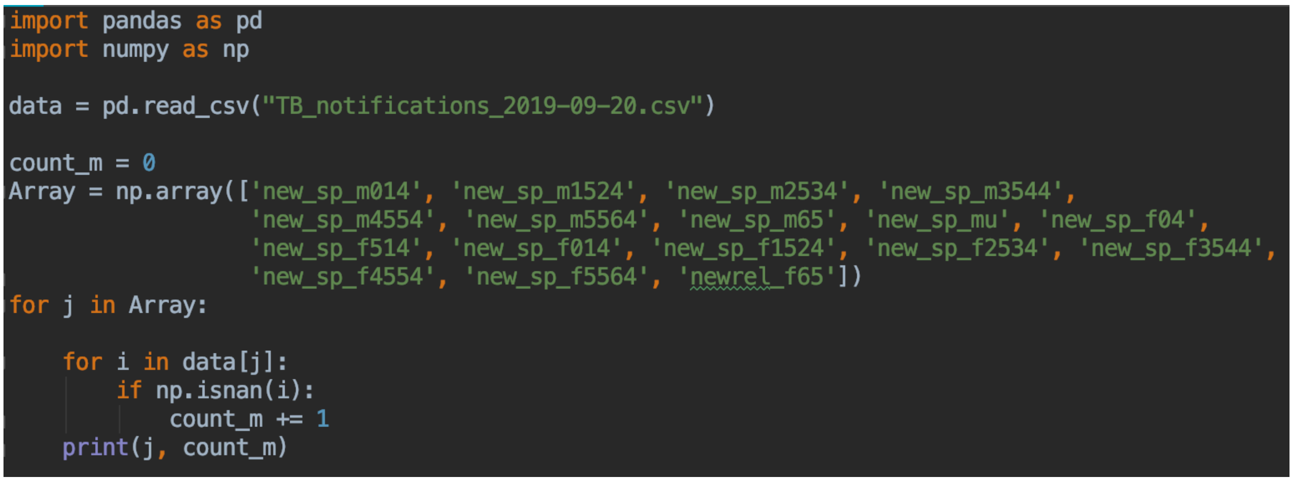
```
> mutate(key = stringr::str_replace(key, "newrel", "new_rel"))
```

is necessary to properly tidy the data. What happens if you skip this line?

-This line is necessary because there is a variable has two different name. If we skip this line, the program does not know to separate the values "newrel". Finally, it gives a warning of too few values.

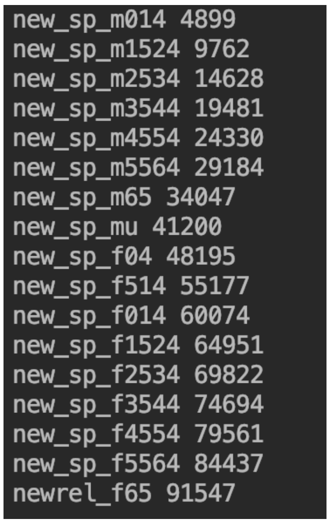
(B) **How many entries are removed from the dataset when you set na.rm to true in the gather command (in this dataset)?**

-Use python to count those empty cells:



```
import pandas as pd
import numpy as np

data = pd.read_csv("TB_notifications_2019-09-20.csv")

count_m = 0
Array = np.array(['new_sp_m014', 'new_sp_m1524', 'new_sp_m2534', 'new_sp_m3544',
                  'new_sp_m4554', 'new_sp_m5564', 'new_sp_m65', 'new_sp_mu', 'new_sp_f04',
                  'new_sp_f514', 'new_sp_f014', 'new_sp_f1524', 'new_sp_f2534', 'new_sp_f3544',
                  'new_sp_f4554', 'new_sp_f5564', 'newrel_f65'])
for j in Array:

    for i in data[j]:
        if np.isnan(i):
            count_m += 1
    print(j, count_m)
```

-The result is as following:



```
new_sp_m014 4899
new_sp_m1524 9762
new_sp_m2534 14628
new_sp_m3544 19481
new_sp_m4554 24330
new_sp_m5564 29184
new_sp_m65 34047
new_sp_mu 41200
new_sp_f04 48195
new_sp_f514 55177
new_sp_f014 60074
new_sp_f1524 64951
new_sp_f2534 69822
new_sp_f3544 74694
new_sp_f4554 79561
new_sp_f5564 84437
newrel_f65 91547
```

-There are totally 91547 entries will be removed.

(C) **Explain the difference between an explicit and implicit missing value, in general. Can you find any implicit missing values in this dataset, if so where?**

-Explicit uses “NA” to flag missing values. Implicit choose not to present the data of missing values.

(D) **Looking at the features (country, year, var, sex, age, cases) in the tidied data, are they all appropriately typed? Are there any features you think would be better suited as a different type? Why or why not?**

-I believe the features in the tidied data are appropriately typed because it removes duplicate and non-indicative things(iso2, iso3, and “new” in key), and separates the key points(type, sex, and age). Therefore, I don’t think it needs more features because it looks simple and clear.

(E) **Explain in your own words what a gather operation is, and give an example of a situation when it might be useful. Do the same for spread.**

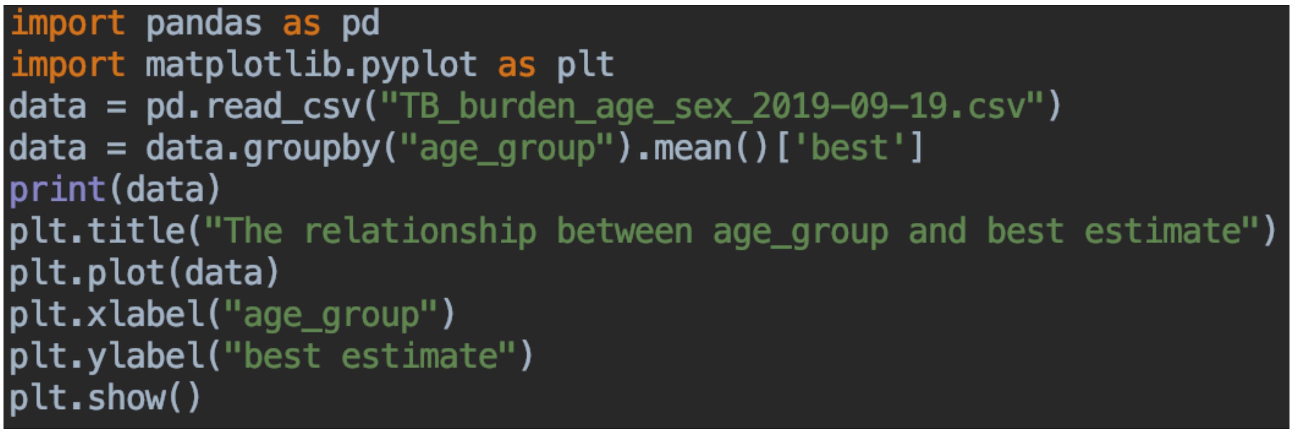
-Gather is used on making some attributes become values in a column, and give a column to store the original values of those attributes.

It is useful when we want to sort the values of different columns. We can gather the values from different columns into one columns, and give a new attribute. Then, we can sort the values of the column from large to small.

-Spread is the opposite of gather(). It makes the values in a column become new attributes, so a table will has more columns but less rows. It is useful when selecting values of a specific condition.

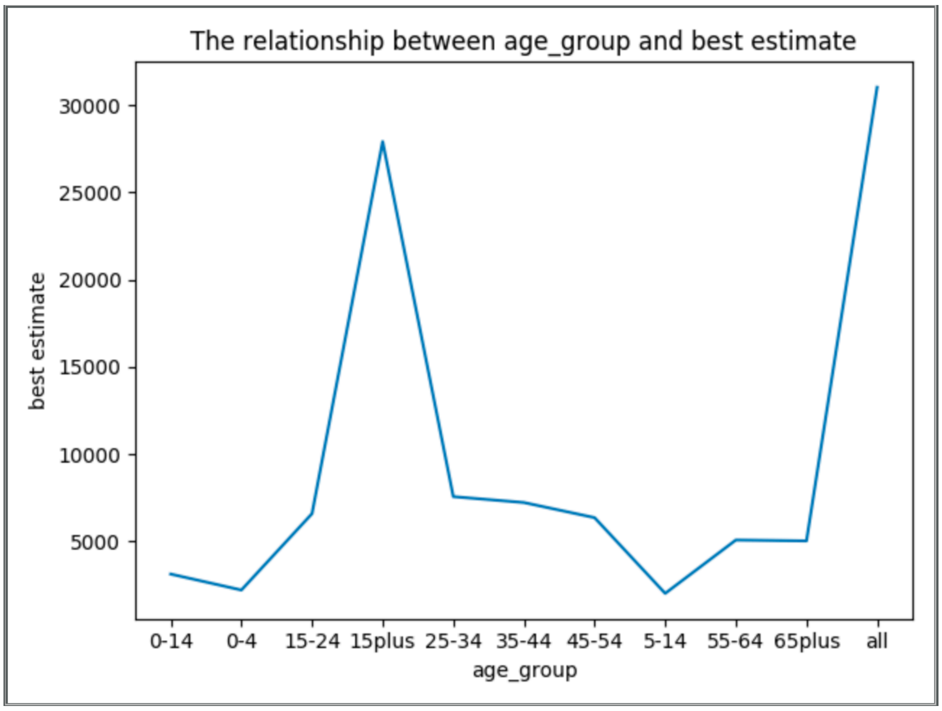
(F) **Generate an informative visualization, which shows something about the data. Give a brief description of what it shows, and why you thought it would be interesting to investigate.**

-Use TB_burden_age_sex database to show the relationship between age_group and best estimate.



```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("TB_burden_age_sex_2019-09-19.csv")
data = data.groupby("age_group").mean()['best']
print(data)
plt.title("The relationship between age_group and best estimate")
plt.plot(data)
plt.xlabel("age_group")
plt.ylabel("best estimate")
plt.show()
```

-The result is as following:

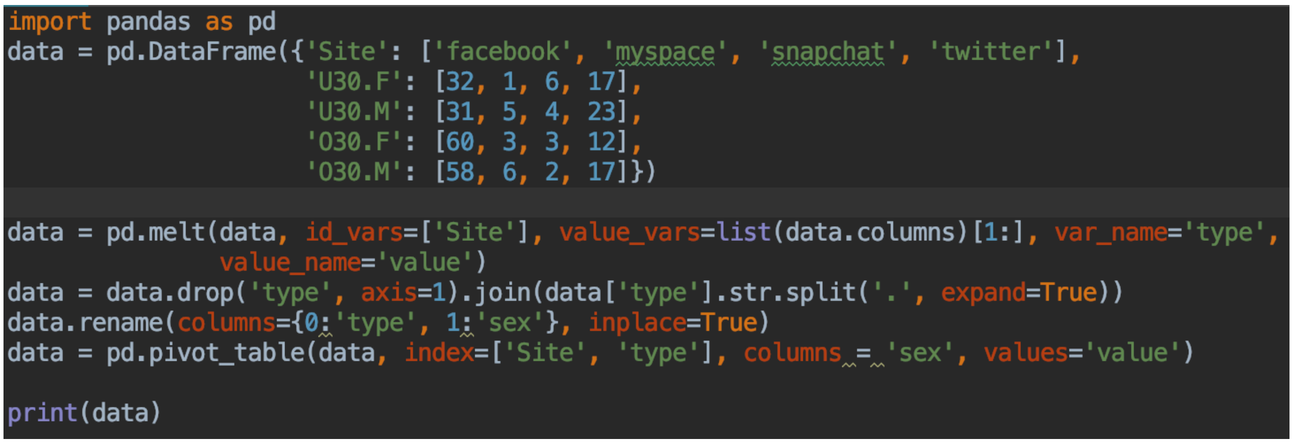


-According to the plot, we can find that most people who have TB are older than 15 years old. TB is not common among teenagers.

(G) **Suppose you have the following dataset called siteDemo:**

Site	**U30.F**	**U30.M**	**O30.F**	**O30.M**
facebook	32	31	60	58
myspace	1	5	3	6
snapchat	6	4	3	2
twitter	17	23	12	17

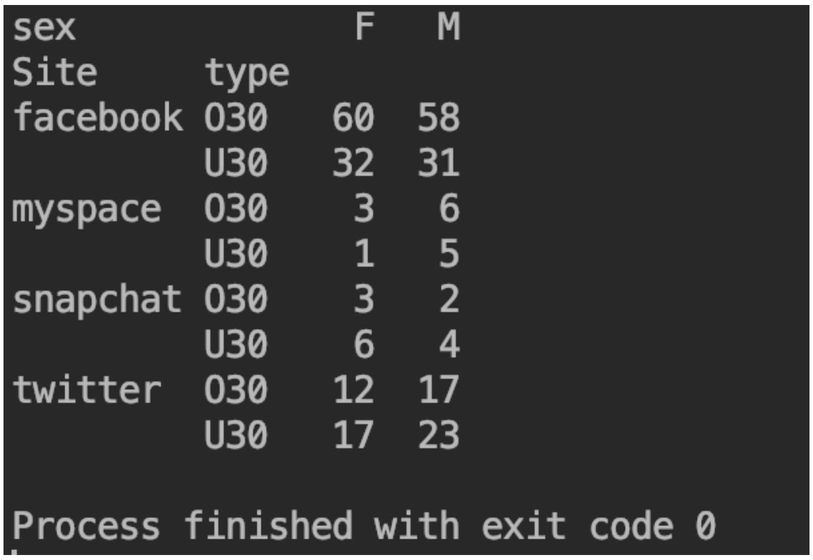
You know that the U30.F column is the number of female users under 30 on the site, O30.M denotes the number of male users 30 or older on the site, etc. Construct this table, and show the code you would use to tidy this dataset (using gather(), separate() and mutate() or melt(), pivot(), and assign()) such that the columns are organized as: Site, AgeGroup, Gender and Count.



```python
import pandas as pd
data = pd.DataFrame({'Site': ['facebook', 'myspace', 'snapchat', 'twitter'],
                     'U30.F': [32, 1, 6, 17],
                     'U30.M': [31, 5, 4, 23],
                     'O30.F': [60, 3, 3, 12],
                     'O30.M': [58, 6, 2, 17]})

data = pd.melt(data, id_vars=['Site'], value_vars=list(data.columns)[1:], var_name='type',
               value_name='value')
data = data.drop('type', axis=1).join(data['type'].str.split('.', expand=True))
data.rename(columns={0:'type', 1:'sex'}, inplace=True)
data = pd.pivot_table(data, index=['Site', 'type'], columns = 'sex', values='value')

print(data)
```

-In this question, I use both of melt and pivot functions. First of all, I try to make four column names split '.' , so I use melt function to let these name become values in the table, and then I can separate age types and sexes. Next, I use pivot-table function to make a new table which makes the values 'F' and 'M' in column 'sex' become new two columns.

-The result is as following:



```
sex            F   M
Site     type
facebook O30   60  58
         U30   32  31
myspace  O30    3   6
         U30    1   5
snapchat O30    3   2
         U30    6   4
twitter  O30   12  17
         U30   17  23

Process finished with exit code 0
```

-I didn't plan to use pivot-table function at the beginning ; however, there was an error message said that the pivot function cannot has two index values which are essential elements for inducting a table, so I had to find a solution for it or change my idea. Luckily, there is a function called pivot-table which is really similar to the pivot function. As a result, I don't have to change my idea, and I can get a pretty table also.