HW 9

SDS348 Spring 2021

Rose Hedderman EID: rrh2298

This homework is due on April 26, 2021 at 8am. Submit a pdf file on Gradescope.

For all questions, include the Python commands/functions that you used to find your answer. Answers without supporting code will not receive credit. Write full sentences to describe your findings.

Question 1: (14 pts)

1.1 (2 pts) The dataset `faithful` contains information about eruptions of the Old Faithful geyser in Yellowstone National Park. Run the code below to download the dataset from GitHub. What type of object is it? Take a peek at the first few rows using `.head()` , looks familiar?

In [1]:

```
# Import package pandas
import pandas as pd
```

In [2]:

```
# Import dataset
faithful = pd.read_csv("https://raw.githubusercontent.com/laylaguyot/datasets/main//fait
hful.csv")
```

In [3]:

```
# peek at the first few rows of faithful
print(type(faithful))
faithful.head()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[3]:

	Unnamed: 0	eruptions	waiting
0	1	3.600	79
1	2	1.800	54
2	3	3.333	74
3	4	2.283	62
4	5	4.533	85

The faithful dataset is a dataframe, same as the ones we worked with in R.

1.2 (2 pts) What are the minimum and maximum values of the variables `eruptions` and `waiting` (both measured in minutes)? You can access individual variables in a dataframe using the `.` operator (e.g., `faithful.eruptions`).

In [4]:

```
# minimum eruptions
min_e = min(faithful.eruptions)
# maximum eruptions
max_e = max(faithful.eruptions)
# minimum waiting
min_w = min(faithful.waiting)
# maximum waiting
max_w = max(faithful.waiting)
print("The minimum eruptions were",min_e,". The maximum eruptions were",max_e,". The minimum waiting was",min_w,". The maximum waiting was",max_w,".")
```

```
The minimum eruptions were 1.6 . The maximum eruptions were 5.1 . The minimum waiting was 43 . The maximum waiting was 96 .
```

The minimum eruptions were 1.6 while the maximum eruptions were 5.1. The minimum waiting was 43 while the maximum waiting was 96.

1.3 (2 pts) Using the package `numpy` , what is the standard deviation of the variable `eruptions` ?

In [5]:

```
# Import package numpy
import numpy as np
```

In [6]:

```
# stdev of eruptions
np.std(faithful.eruptions)
```

Out[6]:

```
1.1392712102257678
```

The standard deviation of the variable eruptions is 1.139 eruptions.

1.4 (2 pts) Recall how logical indexing of a dataframe works in Python. In the example code below, I ask Python for the number of rows and columns in the dataset where the variable `waiting` takes on values greater than 60. Then I ask for the average of the variable `eruptions` when the variable `waiting` is above 60. What is the mean of the variable `eruptions` when waiting is *less than* 1 hour?

In [7]:

```
# rows and columns
faithful[faithful.waiting>60].shape
```

Out[7]:

```
(189, 3)
```

In [8]:

```
# mean of eruptions for waiting values greater than 60
np.mean(faithful[faithful.waiting > 60].eruptions)
```

Out[8]:

```
4.138587301587303
```

In [9]:

```
# mean of eruptions when waiting is less than an hour
np.mean(faithful[faithful.waiting < 60].eruptions)
```

Out[9]:

```
1.9982727272727274
```

The mean of eruptions when waiting is less than an hour is 1.998.

1.5 (2 pts) What is the standard deviation of the variable `eruptions` when `waiting` is *greater than* the median?

In [10]:
```
#  standard deviation of the variable eruptions when waiting is greater than the median
np.std(faithful[faithful.waiting > np.median(faithful.waiting)].eruptions)
```

Out[10]:
```
0.37160308361435546
```

The standard deviation of the variable eruptions when waiting is greater than the median is 0.3716.

1.6 (4 pts) Both variables are measured in minutes. Create two new variables named `eruptions_h` and `waiting_h` that give each variable in hours rather than minutes and add them to the dataset `faithful` . To help get you started, I have given you code that creates a new variable called `eruptions_minus_one` . Instead, computes the requested transformation. Take a peek at the first few rows using `.head()` . What is the mean `waiting` time in minutes? in hours?

In [11]:
```
# Create a variable eruptions_minus_one
faithful['eruptions_minus_one'] = (faithful['eruptions'] - 1)

# Delete that variable from the dataset
del faithful['eruptions_minus_one']
```

In [12]:
```
# create eruptions_h
faithful['eruptions_h'] = (faithful['eruptions']/60)

# create waiting_h
faithful['waiting_h'] = (faithful['waiting']/60)

# preview the dataset
faithful.head()

# mean of waiting time in hours
mean_w_h = np.mean(faithful.waiting_h)

# mean of waiting time in minutes
mean_w_m = np.mean(faithful.waiting)

print("The mean waiting time in hours is", mean_w_h,"while the mean waiting time in minu
tes is", mean_w_m)
```

```
The mean waiting time in hours is 1.1816176470588227 while the mean waiting time in minu
tes is 70.8970588235294
```

The mean waiting time in hours is 1.182 while the mean waiting time in minutes is 70.897.

Question 2: (11 pts)

2.1 (3 pts) Create a list `food` containing the names of your favorite foods. Your list should contain at least 5 different kinds of food. Sort the list so that the names appear in alphabetical order. How many items are in the list?

In [13]:

```
# create a list of foods
rose_food = ["chocolate", "ice cream", "steak", "pasta", "quinoa", "mozzerella sticks"]

# sort the list of foods
rose_food = np.sort(rose_food)

# print sorted list
print(rose_food)

# find how many items are in the list
print("There are", len(rose_food),"items in the list")
```

```
['chocolate' 'ice cream' 'mozzerella sticks' 'pasta' 'quinoa' 'steak']
There are 6 items in the list
```

There are six items in the list.

2.2 (2 pts) Using the function `.sort()` , sort the list in alphabetical order. What is your first favorite food in alphabetical order?

In [14]:

```
# sort the list in alphabetical order
rose_food_sorted = np.sort(rose_food)
print(rose_food_sorted)
```

```
['chocolate' 'ice cream' 'mozzerella sticks' 'pasta' 'quinoa' 'steak']
```

My favorite food in alphabetical order is chocolate.

2.3 (3 pts) Imagine that you have spent a week eating only your favorite foods. Create a dictionary `food_dict` that contains the names of your favorite foods as `keys` & counts for each time you ate that food as `values` . How many times did you eat that week?

In [15]:

```
# create a dictionary of favorite foods and number of times eaten
rose_food_dict = {'chocolate': 20, 'ice cream': 2, 'mozzerlla sticks': 1, 'pasta': 3, 'q
uinoa': 7, 'steak': 1}
# sum up values to determine how many times I ate that week
sum(rose_food_dict.values())
```

Out[15]:

```
34
```

I ate 34 times that week.

2.4 (1 pt) Which of your favorite food did you eat the most often?

In [16]:

```
# favorite favorite food
sorted_fave = {}
sorted_fave_key = sorted(rose_food_dict, key= rose_food_dict.get)

for u in sorted_fave_key:
    sorted_fave[u] = rose_food_dict[u]

keys = list(sorted_fave.keys())
keys[-1]
```

Out[16]:

```
'chocolate'
```

I ate chocolate the most often.

2.5 (2 pts) Using a `for` loop, multiply the values in your dictionary by 4 so that you can estimate how many times you would eat your favorite food per month. Has this ever happened in real life?!

In [17]:

```
# use a for loop to estimate how many times I would eat my favorite food in real life
month = rose_food_dict
for key in rose_food_dict:
    month[key] *= 4
print(month)
    # value + value*i
# can't we mulitply the values in dictionary by four once and get those numbers
# what is accomplished by implementing a for loop
# am i understanding the questio correctly?
```

```
{'chocolate': 80, 'ice cream': 8, 'mozzerlla sticks': 4, 'pasta': 12, 'quinoa': 28, 'ste
ak': 4}
```

This has never happened in real life. I have never only eaten my favorite foods for a straight month.