CSCI2291 Homework 3

Jack Moffatt

February 3, 2022

Problem 1

(a) We want to represent μ_r in terms of μ, σ^2, N, and r. We can begin by defining μ as

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

It follows then that we can define μ_r after replacing the r missing values with μ as

$$\mu_r = \frac{\sum_{i=1}^{N} x_i + r\mu}{N + r}$$

Now, using our definition for μ, we express μ_r and simplify

$$\begin{aligned}
\frac{\sum_{i=1}^{N} x_i + r\mu}{N + r} &= \frac{\sum_{i=1}^{N} x_i + r(\frac{\sum_{i=1}^{N} x_i}{N})}{N + r} \\
&= \frac{(\frac{N(\sum_{i=1}^{N} x_i)}{N}) + (\frac{r\sum_{i=1}^{N} x_i}{N})}{N + r} \\
&= \frac{(N + r)\sum_{i=1}^{N} x_i}{N(N + r)} \\
&= \frac{\sum_{i=1}^{N} x_i}{N}
\end{aligned}$$

So after some algebra we see that $\mu = \mu_r$.

(b) Now, we want to follow similar steps to represent σ_r^2 in terms of μ, σ^2, N, and r. We can again begin by defining σ^2 as

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N-1}$$

It follows that we can then define σ_r^2 after replacing the r missing values with μ as

$$\sigma_r^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2 + \sum_{i=N+1}^{N+r}(\mu - \mu)^2}{N+r-1}$$

Now, since clearly evaluates to 0, as $\mu - \mu = 0$, we can use some algebra to see

$$\begin{aligned}
\sigma_r^2 &= \frac{\sum_{i=1}^{N}(x_i - \mu)^2 + \sum_{i=N+1}^{N+r}(\mu - \mu)^2}{N+r-1} \\
\sigma_r^2 \cdot \frac{N+r-1}{N-1} &= \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N+r-1} \cdot \frac{N+r-1}{N-1} \\
\sigma_r^2 \cdot \frac{N+r-1}{N-1} &= \sigma^2 \\
\sigma_r^2 &= \sigma^2 \cdot \frac{N-1}{N+r-1}
\end{aligned}$$

So we determine, since $\frac{N-1}{N+r-1} < 1$, that $\sigma_r^2 < \sigma^2$.

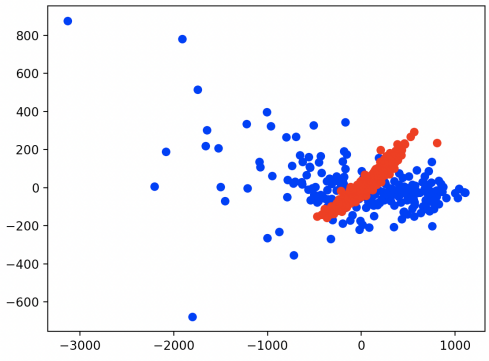
Problem 2

(a) We will use the following code as a solution

```
1   from sklearn.manifold import MDS
2   import matplotlib.pyplot as plt
3   from sklearn import datasets
4
5   breast_cancer = datasets.load_breast_cancer(as_frame = True)
6   benign = breast_cancer.data.loc[breast_cancer.target == 0]
7   malignant = breast_cancer.data.loc[breast_cancer.target == 1]
8
9   projection = MDS(n_components = 2 )
10  projected_benign = projection.fit_transform(benign)
11  projected_malignant = projection.fit_transform(malignant)
12
13  plt.scatter(projected_benign[:,0], projected_benign[:,1], c=['blue'])
14  plt.scatter(projected_malignant[:,0], projected_malignant[:,1], c=['red'])
15  plt.show()
```

We begin by loading the breast cancer data set from the datasets package of sklearn. By setting the as frame parameter to True, we will receive our data formatted as a pandas DataFrame. Then, we sort our data into two seperate DataFrame objects by sorting based on the target value, which is benign and malignant.

Then, we can intialize our 2D projection, using the MDS module. And then we can transform each of our datasets. Finally, using matplotlib we can make a scatter plot, with blue points for the benign dataset and red points for the malignant dataset. The resulting scatter plot:



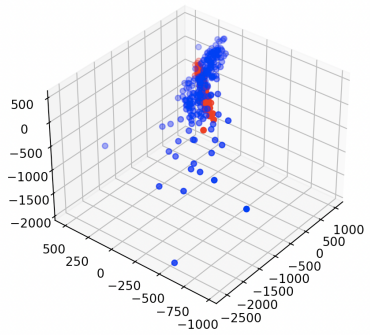
(b) We will use the following code as a solution

```
1   from mpl_toolkits.mplot3d import Axes3D
2   import matplotlib.pyplot as plt
3   from sklearn.manifold import MDS
4   from sklearn import datasets
5
6   breast_cancer = datasets.load_breast_cancer(as_frame = True)
7   benign = breast_cancer.data.loc[breast_cancer.target == 0]
8   malignant = breast_cancer.data.loc[breast_cancer.target == 1]
9
10  fig = plt.figure()
11  ax = fig.add_subplot(111, projection = '3d')
12
13  projection = MDS(n_components = 3)
14  projected_benign = projection.fit_transform(benign)
15  projected_malignant = projection.fit_transform(malignant)
16
17  benign_scatter = ax.scatter(projected_benign[:,0], projected_benign[:,1], projected_benign[:,2],
        c=['blue'] )
18  malignant_scatter = ax.scatter(projected_malignant[:,0], projected_malignant[:,1],
      projected_malignant[:,2], c=['red'] )
19  plt.show()
```

Lines 1-8 are identical to the solution in part **a**. In lines 10 and 11 we intialize our 3D figure, following the guidelines provided in the documentation.

Again, we initialize our projection, except this time, we set the number of components to 3, indicating we want a 3 dimensional projection. We then fit our projection onto each of our two data sets.

Finally, using the figure we initialized in lines 10-11, we add scatter plots using the 3 columns of our data sets as the X, Y, and Z axes. Again, our benign data points are colored blue, while malignant data points are colored red. The resulting 3D scatter plot:



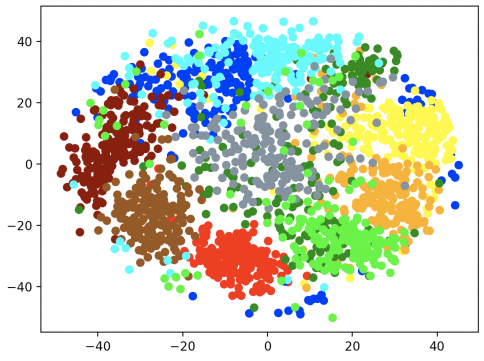
(c) We will use the following code as a solution

```
from sklearn import datasets
from sklearn.manifold import MDS
import matplotlib.pyplot as plt

digits = datasets.load_digits(as_frame = True)
projection = MDS(n_components = 2, )
projected_digits = projection.fit_transform(digits.data)

color_map = {
    '0' : 'red', '1' : 'blue',
    '2' : 'yellow', '3' : 'orange',
    '4' : 'maroon', '5' : 'green',
    '6' : 'saddlebrown', '7' : 'cyan',
    '8' : 'slategray', '9' : 'lime'
}

for target in digits.target_names:
    sorted_data = projected_digits[digits.target == target]
    plt.scatter(sorted_data[:,0], sorted_data[:,1], c= [color_map[str(target)]] )
plt.show()
```

As in the first section of the problem, we load the digits data set and set as frame to True. Additionally, we initialize an MDS projection to 2D space. Then we transform the digits data to 2 dimensions.

As there are 10 target attributes, we set up a map to map each target name to a unique color to use for our visualization.

Now we use a for loop to iterate over the target names, which for this particular dataset is a list of integers between 0 and 9 inclusive. For each target value, we filter the projected data for only the the points corresponding to the particular target value. Then, we add a scatter plot colored according to the color map. After plotting the data for each target, we have the scatter plot:

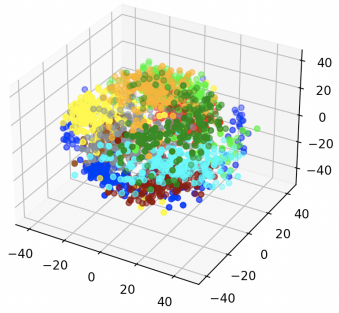


(d) We will use the following code for a solution:

```
1   from mpl_toolkits.mplot3d import Axes3D
2   import matplotlib.pyplot as plt
3   from sklearn.manifold import MDS
4   from sklearn import datasets
5
6   digits = datasets.load_digits(as_frame = True)
7   projection = MDS(n_components = 3, )
8   projected_digits = projection.fit_transform(digits.data)
9
10  color_map = {
11      '0' : 'red', '1' : 'blue',
12      '2' : 'yellow', '3' : 'orange',
13      '4' : 'maroon', '5' : 'green',
14      '6' : 'saddlebrown', '7' : 'cyan',
15      '8' : 'slategray', '9' : 'lime'
16  }
17
18  fig = plt.figure()
19  ax = fig.add_subplot(111, projection = '3d')
20
21  for target in digits.target_names:
22      sorted_data = projected_digits[digits.target == target]
23      ax.scatter(sorted_data[:,0], sorted_data[:,1], sorted_data[:,2], c = [color_map[str(target)
        ]] )
24  plt.show()
```

[language = python] Again, we load the data set, configure a 3D MDS projection and transform our data. Additionally, we can configure a color map to give unique colors for the data on each of our target datasets.

Now, we can configure our 3D figure with the same lines that we did in part **b**. Using the same for loop as in part **c** we create scatter plots for each target value. The only difference is now we invoke a 3D scatter plot and pass the three columns of our projected data as the X, Y, and Z axes. After plotting the data for each target, we have the scatter plot:



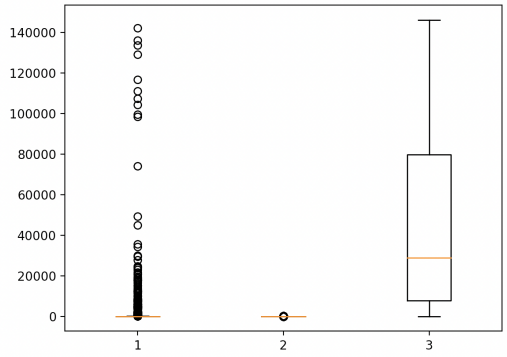
Problem 3

(a) We will use the following solution as our code

```
1  import pandas as pd
2  from matplotlib import pyplot as plt
3
4  covid_variants = pd.read_csv("HW3/python/datasets/covid-variants.csv")
5  us_variants = covid_variants.loc[covid_variants.location == "United States"]
6
7  plt.boxplot([us_variants.num_sequences, us_variants.perc_sequences, us_variants.
     num_sequences_total])
8  plt.show()
```

We begin by importing pandas the pyplot package of matplotlib. Then, using the read csv function of pandas, we read the covid variants data. Next we filter our data to get only the entries from the United States.

Then we create a boxplot, using each of the descritive attributes in a list as the argument to plot each box on a different index. We get the resulting scatter plot



(b) We will use the following code as our solution:

```
1   import pandas as pd
2
3   covid_variants = pd.read_csv("HW3/python/datasets/covid-variants.csv")
4   us_variants = covid_variants.loc[covid_variants.location == "United States"]
5   weeks_list = us_variants.date.unique()
6
7   count = 0
8
9   for week in range(0, len(weeks_list)):
10      if weeks_list[week][0:4] == '2021':
11          num_sequences_total = us_variants.num_sequences_total.loc[us_variants.date == weeks_list
      [week]].mean()
12          count += num_sequences_total
13
14  print(f"Total number of US records in 2021: {int(count)}")
```

Lines 1-5 are identical to part **a**. Then we will additionally get a list of the weeks that data samples were taken at by taking a unique indexed list of the dates column of the data frame.

Then we will initialize a count and iterate over the list of dates and for each date that occurs in 2021 we will take a count of the total number of sequences in this week. Then we will increment this count by the numebr of sequences. Upon running this code we get the output of

```
1   >>> Total number of US records in 2021: 1941782
```

(c) We will use the following code as our solution:

```
import pandas as pd

covid_variants = pd.read_csv("HW3/python/datasets/covid-variants.csv")
us_variants = covid_variants.loc[covid_variants.location == "United States"]
canada_variants = covid_variants.loc[covid_variants.location == "Canada"]

us_perc_sequences = us_variants.perc_sequences
canada_perc_sequences = canada_variants.perc_sequences

us_mean = us_perc_sequences.mean()
canada_mean = canada_perc_sequences.mean()

print(f"{'US Mean' :<15}{us_mean}")
print(f"{'Canada Mean' :<15}{canada_mean}")
```

Again we load our data and sort out the US data and we additionally sort out the canadian data samples as well. Next, we seperate out the percentage of sequences data attribute and then take the mean of each of these attribute values. Printing our results we see

```
>>> US Mean        6.239027777777777
>>> Canada Mean    6.402651515151515
```

(d) We will use the following code as our solution:

```
1   import pandas as pd
2   import numpy as np
3   import scipy.stats as stat
4
5   covid_variants = pd.read_csv("HW3/python/datasets/covid-variants.csv")
6   us_variants = covid_variants.loc[covid_variants.location == "United States"]
7   canada_variants = covid_variants.loc[covid_variants.location == "Canada"]
8
9   us_perc_sequences = us_variants.perc_sequences.to_numpy()
10  canada_perc_sequences = canada_variants.perc_sequences.to_numpy()
11
12  normalized_us =  (us_perc_sequences - np.mean(us_perc_sequences)) / np.std(us_perc_sequences)
13  normalized_canada = (canada_perc_sequences - np.mean(canada_perc_sequences)) / np.std(
      canada_perc_sequences)
14
15  t_test = stat.ttest_ind(normalized_us, normalized_canada)
16
17  print(t_test)
```

Lines 5-10 are identical to part **c**, however this time we convert the percentage of sequences into NumPy arrays. Next, we normalize our data so that we can apply a t test to the data.

We select the independent t test because our samples come from separate populations, so there is no relationship between the two datasets. Passing our two normalized attribute value vectors to the t test we receive both a t value (statistic) and a p value which is a measure of our interval of confidence. Running the print statement, we get the values

```
1   >>> Ttest_indResult(statistic=-7.839338635406006e-16, pvalue=0.9999999999999993)
```

We can see that our t value is 0 and since our p value is essentially 1, we have a very high interval of connfidence. This tells us that the difference in the sample means is incredibly likely to occur in the respective populations as well.

Problem 4

Our group currently consists of myself, Alexander Benati, Bryan Kim, and Ayush Patel. We are currently planning on working on a music recommendation system based on Spotify music data.