Project 2

GENERAL INSTRUCTIONS:

this is NOT a group project

- **CLEARLY** mark where you are answering each question (all written questions must be answered in Markdown cells, NOT as comments in code cells)
- Show all code necessary for the analysis, but remove superfluous code

---

DONUTS

Using the dataset *krispykreme.csv*,

- **a)** make 3 scatterplots using ggplot to show:
  - `Sodium_100g vs Total_Fat_100g`
  - `Sodium_100g vs. Sugar_100g`
  - `Sugar_100g vs Total_Fat_100g`

    You will be graded on the effectiveness of the graphs as well as their content.
- **b)** Using the scatterplots from part **a** as well as the donuts dataset, **thouroughly discuss which clustering method** (KMeans, Gaussian Mixture Models (EM), Hierarchical Clustering, or DBSCAN) **you think would be best for this data and WHY**. Be sure to include discussions of assumptions each algorithm does/does not make, and what types of data they are good/bad for (**mention each of the 4 algorithms at least once**) and how they apply to this specific dataset. (*IN A MARKDOWN CELL*)
  - you should be making statements that both 1) discuss characteristics of the algorithm and 2) specifically discuss how that characteristic applies (or doesn't) to this dataset.

Please note that for this assignment, "It's easier to code" does not count as a valid reason. The reasons should be based on the algorithms/data.

(You must use "**" to make any mention of one of the algorithms bold in your discussion. For

example "I think **DBSCAN** is the best algorithm ever!" will make the word "**DBSCAN**" bold in a Markdown cell).

- **c) Implement the TWO algorithms** you think will work BEST (1 algo) and WORST (1 algo) here using all 3 variables `Sodium_100g`, `Total_Fat_100g` and `Sugar_100g`, and describe **how you chose any hyperparameters** (such as distance, # of clusters, min_samples, eps, linkage...etc). Make sure to z-score your variables. (*IN A MARKDOWN CELL*)
- **d) Thouroughly discuss the performance** of your clustering models. For each algorithm (best and worst):
    - which metric did you use to asses your model? (*IN A MARKDOWN CELL*)
    - how did your model perform? (*IN A MARKDOWN CELL*)
    - remake the 3 graphs from part a, but color by cluster assignment. Describe what characterizes each cluster, and give an example of a label for that cluster (e.g. "these donuts are low fat, and low sugar so I would call these healthy donuts") (*IN A MARKDOWN CELL*)
- **e)** Choose ONE other of the `_100g` variables from the data set to **add to your clustering model** to improve it.
    - explain why you chose this variable. Either based on improvement in metrics, or outside knowledge you have about food/donuts (*IN A MARKDOWN CELL*)
    - make a new model, identical to the model you thought would be best in part c, but also including your new variable.
    - did this variable improve the fit of your clustering model? How can you tell? (*IN A MARKDOWN CELL*)

Note: The columns with `_100g` at the end represent the amount of that nutrient per 100 grams of the food. For example, `Total_Fat` tells you the total amount of fat in that food, whereas `Total_Fat_100g` tells you how much fat there is per 100 grams of that food.

```
# import necessary packages
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import silhouette_score
from pandas import Series
import scipy.cluster.hierarchy as sch
from matplotlib import pyplot as plt

from plotnine import *
%matplotlib inline
```

```
df = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392Parle
df.head()
```

	Restaurant_Item_Name	restaurant	Restaurant_ID	Item_Name	Item_Description
0	Krispy Kreme Apple Fritter	Krispy Kreme	49	Apple Fritter	Apple Fritter Doughnuts
1	Krispy Kreme Chocolate Iced Cake Doughnut	Krispy Kreme	49	Chocolate Iced Cake Doughnut	Chocolate Iced Cake Doughnut Doughnuts
2	Krispy Kreme Chocolate Iced Custard Filled Dou...	Krispy Kreme	49	Chocolate Iced Custard Filled Doughnut	Chocolate Iced Custard Filled Doughnut Doughnuts
3	Krispy Kreme Chocolate Iced Glazed Doughnut	Krispy Kreme	49	Chocolate Iced Glazed Doughnut	Chocolate Iced Glazed Doughnut Doughnuts
4	Krispy Kreme Chocolate Iced Glazed Cruller Dou...	Krispy Kreme	49	Chocolate Iced Glazed Cruller Doughnut	Chocolate Iced Glazed Crulle Doughnut Doughnuts

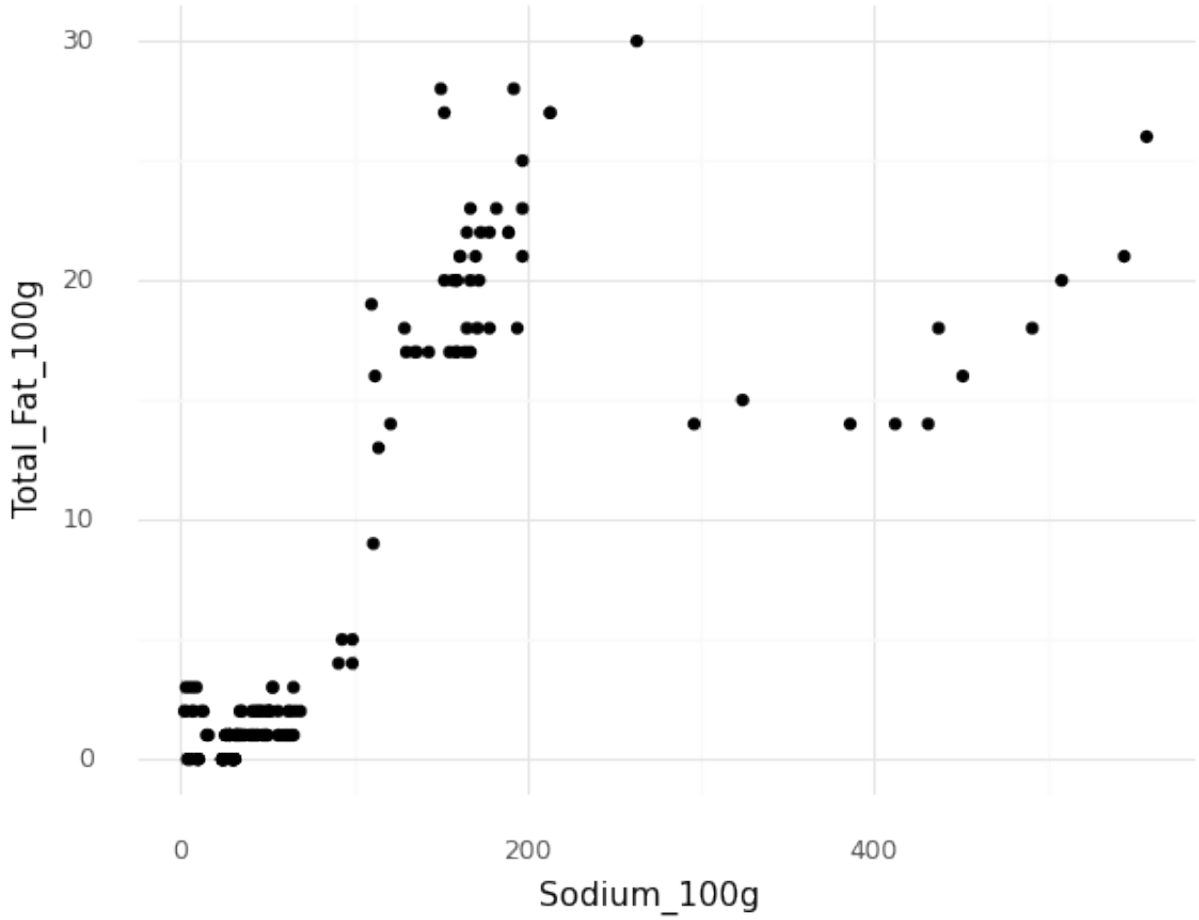
5 rows × 32 columns

```
df.isnull().sum()
```

```
Restaurant_Item_Name        0
restaurant                  0
Restaurant_ID               0
Item_Name                   0
Item_Description            0
Food_Category               0
Serving_Size                0
Serving_Size_text         205
Serving_Size_Unit           0
Serving_Size_household    198
Calories                    0
Total_Fat                   0
Saturated_Fat               0
Trans_Fat                   0
Cholesterol                 0
Sodium                      0
Potassium                 166
Carbohydrates               0
Protein                     0
Sugar                       0
Dietary_Fiber              31
Calories_100g               0
Total_Fat_100g              0
Saturated_Fat_100g          0
Trans_Fat_100g              0
Cholesterol_100g            0
Sodium_100g                 0
Potassium_100g            166
Carbohydrates_100g          0
Protein_100g                0
Sugar_100g                  0
Dietary_Fiber_100g         31
dtype: int64
```

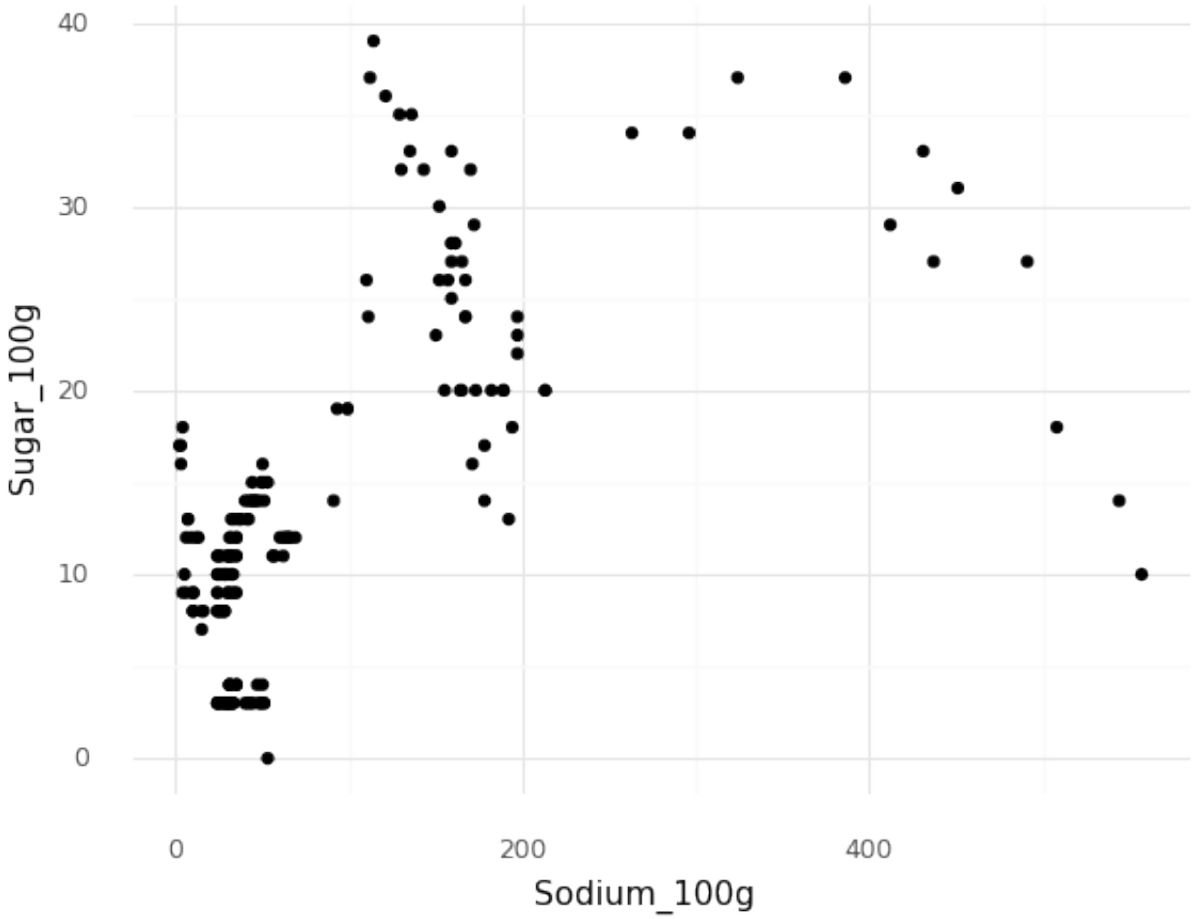
A)

```
#plot of Sodium_100g vs. Total_Fat_100g
ggplot(df, aes(x = 'Sodium_100g', y = 'Total_Fat_100g')) + geom_point() + theme_mir
```



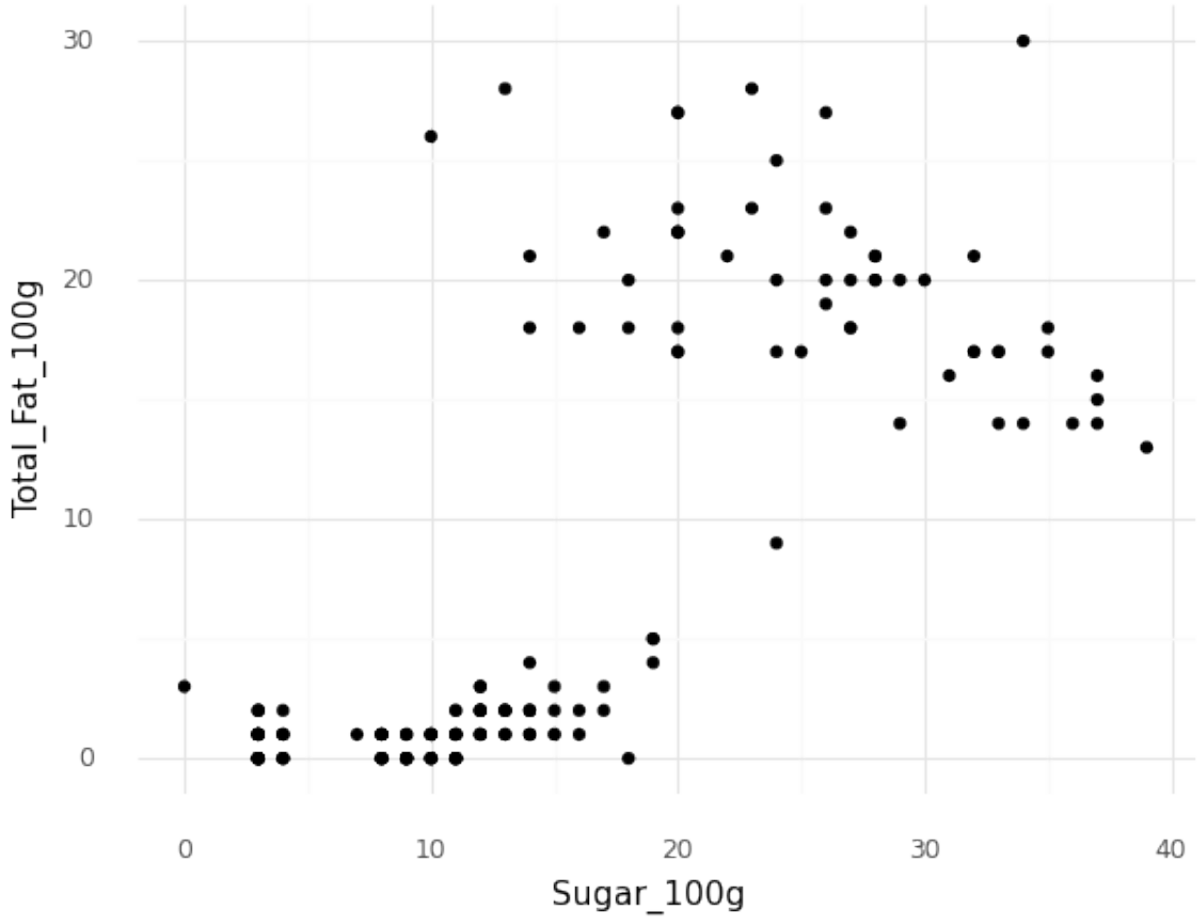

```
<ggplot: (8742190794509)>
```

```
#plot of Sodium_100g vs. Sugar_100g
ggplot(df, aes(x = 'Sodium_100g', y = 'Sugar_100g')) + geom_point() + theme_minimal
```




```
<ggplot: (8792502880101)>
```

```
#plot of Sugar_100g vs. Total_Fat_100g
ggplot(df, aes(x = 'Sugar_100g', y = 'Total_Fat_100g')) + geom_point() + theme_mini
```




```
<ggplot: (8792502833709)>
```

B)

The method I am choosing to cluster this algorithim is **Heirarchal Agglomerative Clustering.**

The **K Means** algorithm carries certain assumptions when clustering data points, which include:

- Clusters will form circluar shapes
- Clusters will contain relatively the same amount of data points
- All variables roughly have the same variance

None of the clusters pictured in the scatterplots show clear signs of having circular shapes. In addition, the data is scattered around in each of these observed "clusters" and none of them

really show the same amount of data in each one. For these reasons, I decided to dismiss the K-Means method.

On the other hand, **Guassian Mixture Models (EM)** carry the assumptions that:

- Each cluster (or mixture of clusters) will show a Gaussian distribtuion
- Clusters will form elitpical shapes

When looking at the scatterplots, I felt that the clusters formed in all 3 graphs did not show a Gaussian Distribution, and led me to believe it would affect the accuracy of the final model. Because of this reason, I dediced to toss out the Gaussian Mixture model.

Next, the **DBSCAN** method is a similar clustering method that is based on density of points in a cluster. It does not carry the assumptions of fitting the clusters to a certain shape (like K Means and EM algorithim) and has the ability to pick up on "noise" data (outliers), which there seems to be a good amount of in the scatterplots above. But, DBSCAN performs poorly with high dimensional data (data with lots of features/predictors), overlapping clusters and clusters with different densities. This dataset contains a good amount of features, but the main reason I dismissed this method of clustering was due to the scatterplots above. The third plot does not have distinct clusters (overlapping), and each scatterplot contains clusters with different densities of points. I believe the **DBSCAN model will perform the worst** in comparison to the other clustering models. There are a lot of overlapping clusters and, similar to the K-Means model, none of the observed clusters show signs of having similar densities. For these reasons, I believe the DBSCAN model will perform the worst.

Lastly, I believe the **Heirarchal Alglomerative Clustering method will be the best method** to cluster this data. HAC models look to cluster data represented by a heirarchal structure (like a dendrogram). HAC is different from the previous clustering methods because it has the ability to portray the relationships between these clusters, and also show how similar each cluster is. In addition, it is very flexible with number of clusters observed and the type of linkage chosen to build the model and clusters are not forced to fit a certain shape. The HAC algorithim does have some disadvanatges, which are:

1. This method of clustering performs very slowly (has Big O Runtime of O(n^3)).
2. Once a cluster has been formed, you cannot go back and unmerge these clusters.

Though there are disadvantages, these should not be applicable to this dataset. With the advantages of flixibility and not carrying some of the disadvantages of the other clustering

methods, I think HAC will perform best.

C)

Algorithim That Will Work Best: HAC Model

For the Heirarchal Agglomerative Clustering method, there are a few paramters that have to be specified before the model is run. Firstly, I chose number of clusters to be 3. One advantage of HAC models is that it is very flexible with the number of clusters that can be chosen, meaning there is a not a threshold that it has to be over or under. I chose 3 clusters because from visually examining the scatterplots, it's apparent that in a majority of the graphs there are 3 distinct clusters. Next, a distance metric has to be specified, for which I chose euclidian, which is the standard distance metric. Lastly, I chose Ward's method as the linkage criteria. Ward's method asks, that two clusters were both put on the same scale, how much am I increasing average deviation from the mean to each data point in the cluster. I decided to dismiss Single Linkage and Complete Linkage because in some of the clusters, there are straggling outlier points, and I thought this would not be the best way to model these clusters. In addition, I decided to dimiss Average Linkage because a majority of the clusters pictured above do not have the same number of data points, and I also thought this would misrepresent the data. Using Ward's method, the model is putting all clusters on the same scale and comparing the increase of deviation from the mean.

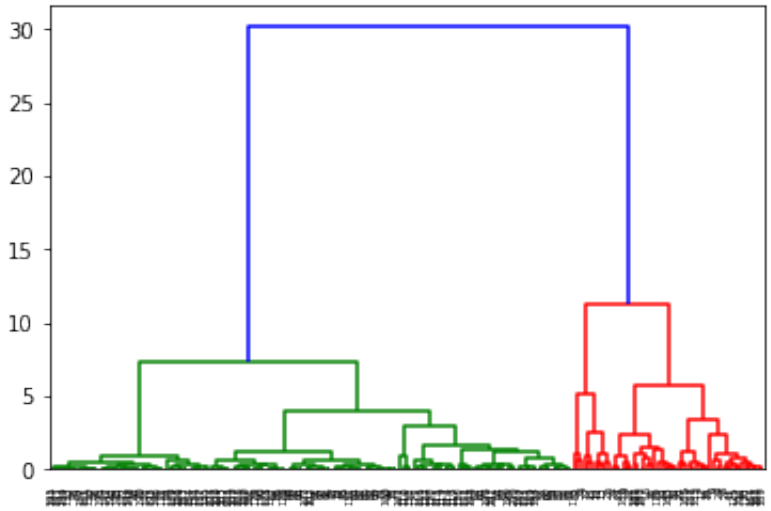
```
features = df[['Sodium_100g', 'Sugar_100g', 'Total_Fat_100g']]
z = StandardScaler()
z.fit(features)

X = z.transform(features)

hac = AgglomerativeClustering(n_clusters = 3,
                              affinity = "euclidean",
                              linkage = "ward")
```

```
hac.fit(X)

dendro = sch.dendrogram(sch.linkage(X, method = 'ward'))
```



Algorithim That Will Work Worst: DBSCAN

For the DBSCAN model, there are a couple parameters that need to be specified before running as well. The epsilon value is a distance metric chosen to determine how far from itself a data point will look for neighbors. To find the optimal epsilon value, I created a K-Dist graph plotting the values of K distance from each data point. Once the K-Dist graph was made, I look for the inflection point, or the "elbow", and in this case, it is around .525. Next, the min_points needs to be specified, which is the minimum number of neighbors needed for something to be considered a cluster. I tried different values and tested which would lead to the best clusters, and decided on 10 as minimum_points value.

```
#Set Up Array For Neighbor Distances
mins = 3
nn = NearestNeighbors(n_neighbors= mins+1)

nn.fit(X)

distances, neighbors = nn.kneighbors(X)
distances.shape
```

```
(205, 4)
```

```
#Sorting the Distances
distances = np.sort(distances[:, mins], axis = 0)
distances
```

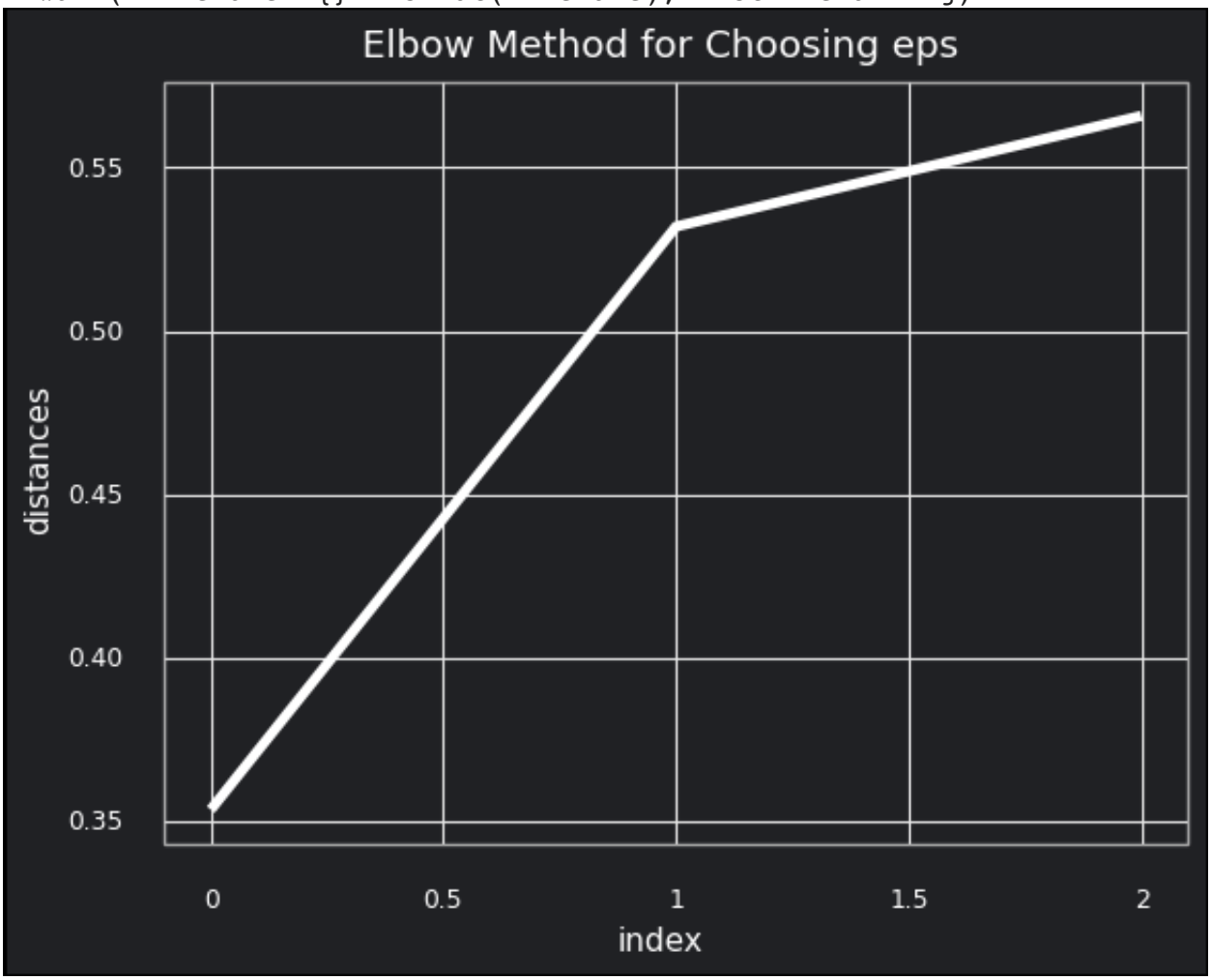
```
array([0.35348823, 0.53200703, 0.5658845 ])
```

```
#Elbow Method: Plotting the Distances on K-Dist Graph
#plot the distances
distances_df = pd.DataFrame({"distances": distances,
                             "index": list(range(0,distances.shape[0]))})

plt = (ggplot(distances_df, aes(x = "index", y = "distances")) +
 geom_line(color = "white", size = 2) + theme_minimal() +
 labs(title = "Elbow Method for Choosing eps") +
 theme(panel_grid_minor = element_blank(),
      rect = element_rect(fill = "#202124ff"),
      axis_text = element_text(color = "white"),
      axis_title = element_text(color = "white"),
      plot_title = element_text(color = "white"),
      panel_border = element_line(color = "darkgray"),
      plot_background = element_rect(fill = "#202124ff")
      ))
ggsave(plot=plt, filename='elbow.png', dpi=300)

plt
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/ggplot.py:729: PlotnineWarning
  from_inches(height, units), units), PlotnineWarning)
/usr/local/lib/python3.7/dist-packages/plotnine/ggplot.py:730: PlotnineWarning
  warn('Filename: {}'.format(filename), PlotnineWarning)
```




```
<ggplot: (8755369507661)>
```

```
#DBSCAN with 0.5 eps
db = DBSCAN(eps = 0.525, min_samples = 10).fit(X)

labsList = ["Noise"]
labsList = labsList  + ["Cluster " + str(i) for i in range(1,len(set(db.labels_)))]
df["assignments"] = db.labels_
```

D)

HAC Model

For the HAC model, I used the silhouette score coefficient to evaluate the accuracy of this model. The silhouette score is calculated based on the cohesion of the data points and the separation of the clusters. It will output a value with range [-1, 1], with values being closer to 1 are considered having good cohesion and separation. The HAC model outputted a silhouette score .7386, which indicates the model has dense clusters and good separation between the clusters. It did well clustering the data in the first two graphs, but had trouble on the third graph as it was not able to clearly cluster the two models on the right.

```
membership = hac.labels_
```

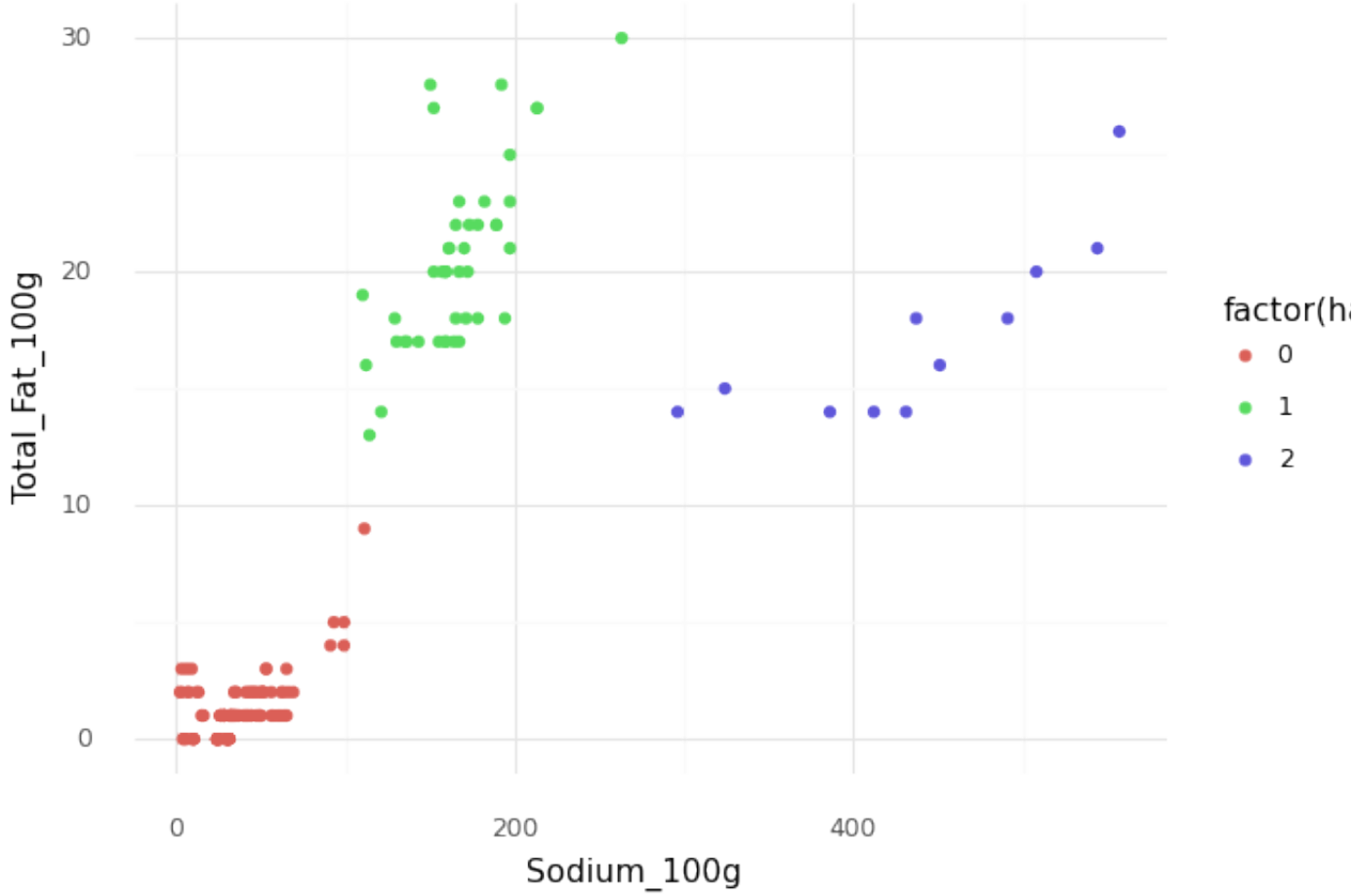
```
silhouette_score(X, membership)
```

```
    0.7386187851618576
```

```
df["hac_clust"] = membership
```

```
#plot of Sodium_100g vs. Total_Fat_100g
ggplot(df, aes(x = 'Sodium_100g', y = 'Total_Fat_100g', color = 'factor(hac_clust)'
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```

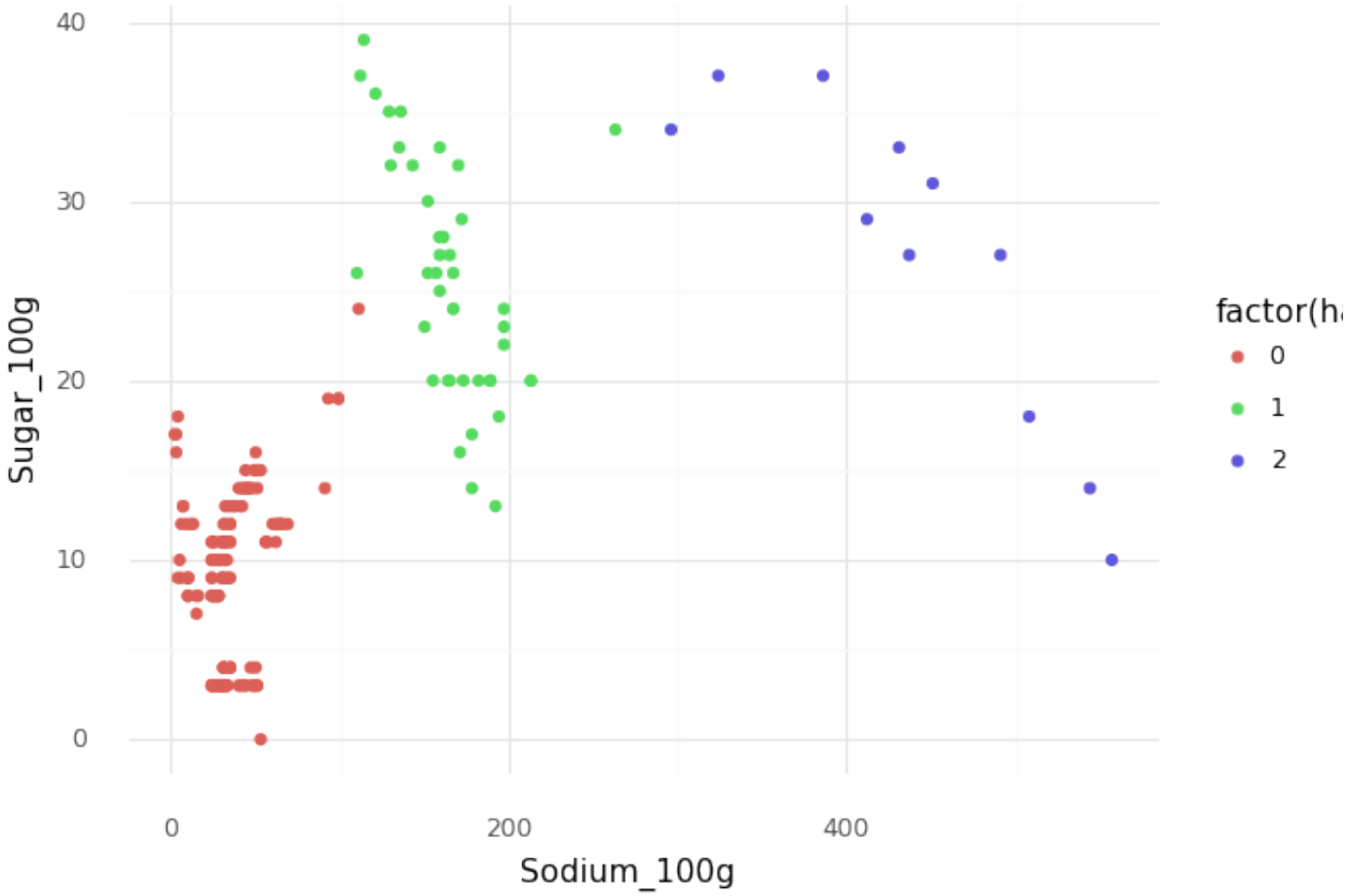



```
<ggplot: (8755366356433)>
```

For this graph, the red cluster (0) represents low sodium and low total fat, so I characterized this cluster as a "healtheir donut". The green cluster (1) has high fat but low to mid-tier sodium, so I can characterize this cluster as a "normal donut" (healthier than the blue cluster but not as healthy as red cluster). The last blue cluster is high in sodium and fat, so this cluster can be characterized as an "unhealthy donut".

```
#plot of Sodium_100g vs. Sugar_100g
ggplot(df, aes(x = 'Sodium_100g', y = 'Sugar_100g', color = 'factor(hac_clust)')) +
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```

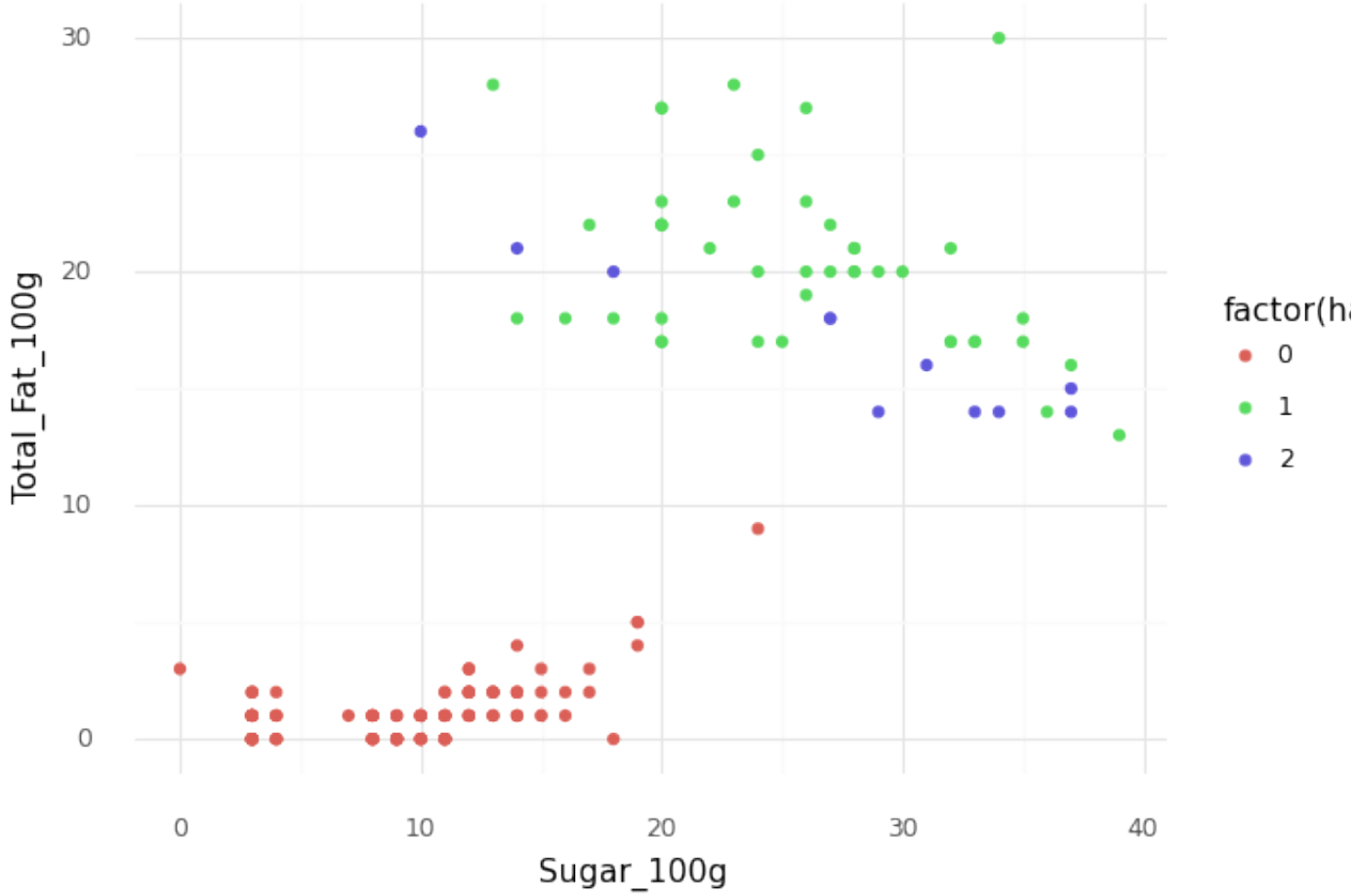



```
<ggplot: (8755364364109)>
```

For this graph, the red cluster (0) represents low sugar and low sodium, so I characterized this cluster as a "healthy (vegan) donut". The green cluster (1) has high sugar but low to mid-tier sodium, so I can characterize this cluster as a "normal donut" (contains high sugar but not a lot of sodium, like a normal donut). The last blue cluster is high in sodium and sugar, so this cluster can be characterized as an "unhealthy donut".

```
#plot of Sugar_100g vs. Total_Fat_100g
ggplot(df, aes(x = 'Sugar_100g', y = 'Total_Fat_100g', color = 'factor(hac_clust)')
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```




```
<ggplot: (8755364299457)>
```

For this graph, the red cluster (0) represents low sugar and low total fat, so I characterized this cluster as a "very healthy donut". The green cluster (1) has high fat and high sugar, so I can characterize this cluster as a "unhealthy donut". Similarly, the last blue cluster is high in sugar and fat, so this cluster can also be characterized as an "unhealthy donut".

DBSCAN Model

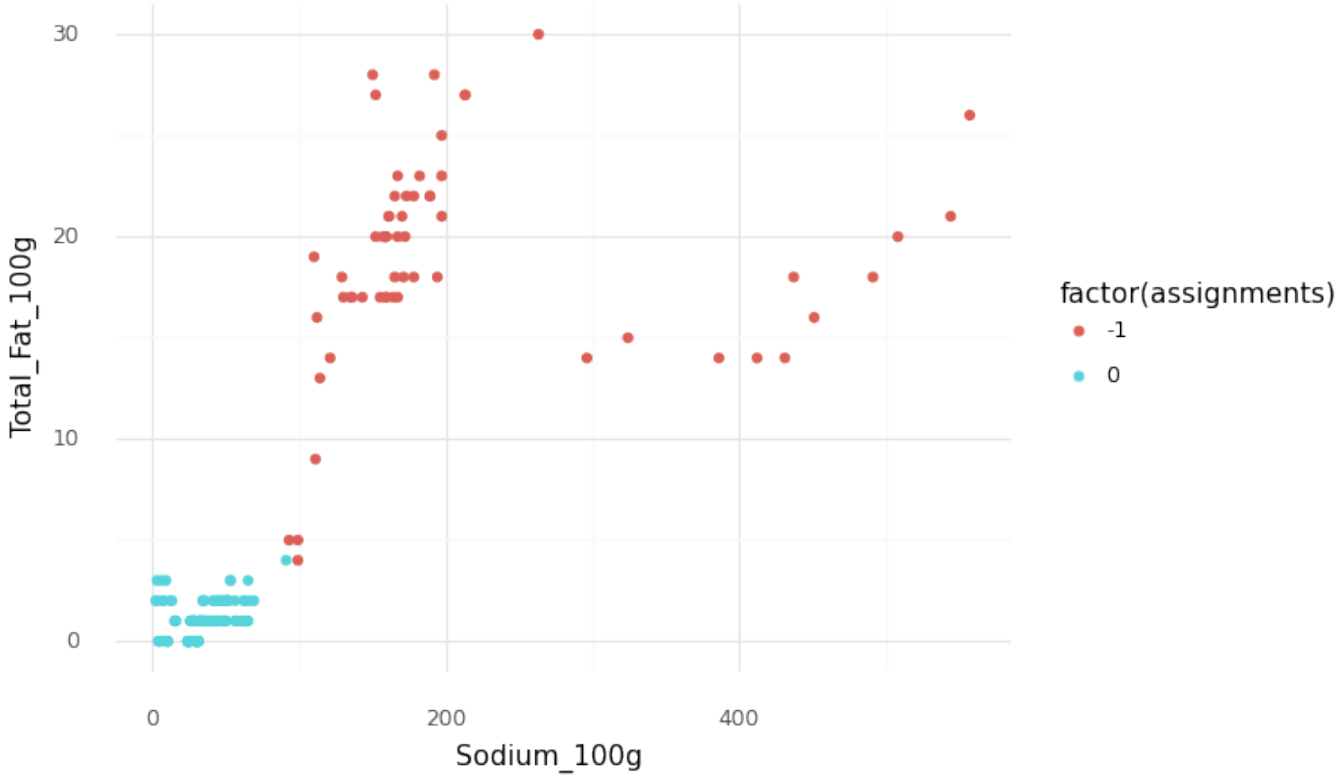
As mentioned above, the silhouette score is metric used to determine how well a model clusters data points, using cohesion and separation as part of the equation. The DBSCAN model returned a silhouette score of .61, which was worse than the K-Means model. The DBSCAN model was not able to effectively cluster 3 different algorithms, and included random points in some of the clusters. For the bottom two graphs, there are multiple overlapping clusters, meaning it did not do a good job clustering the data.

```
silhouette_score(X, df['assignments'])
```

```
0.6074279419252253
```

```
#plot of Sodium_100g vs. Total_Fat_100g
ggplot(df, aes(x = 'Sodium_100g', y = 'Total_Fat_100g', color = 'factor(assignments
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```

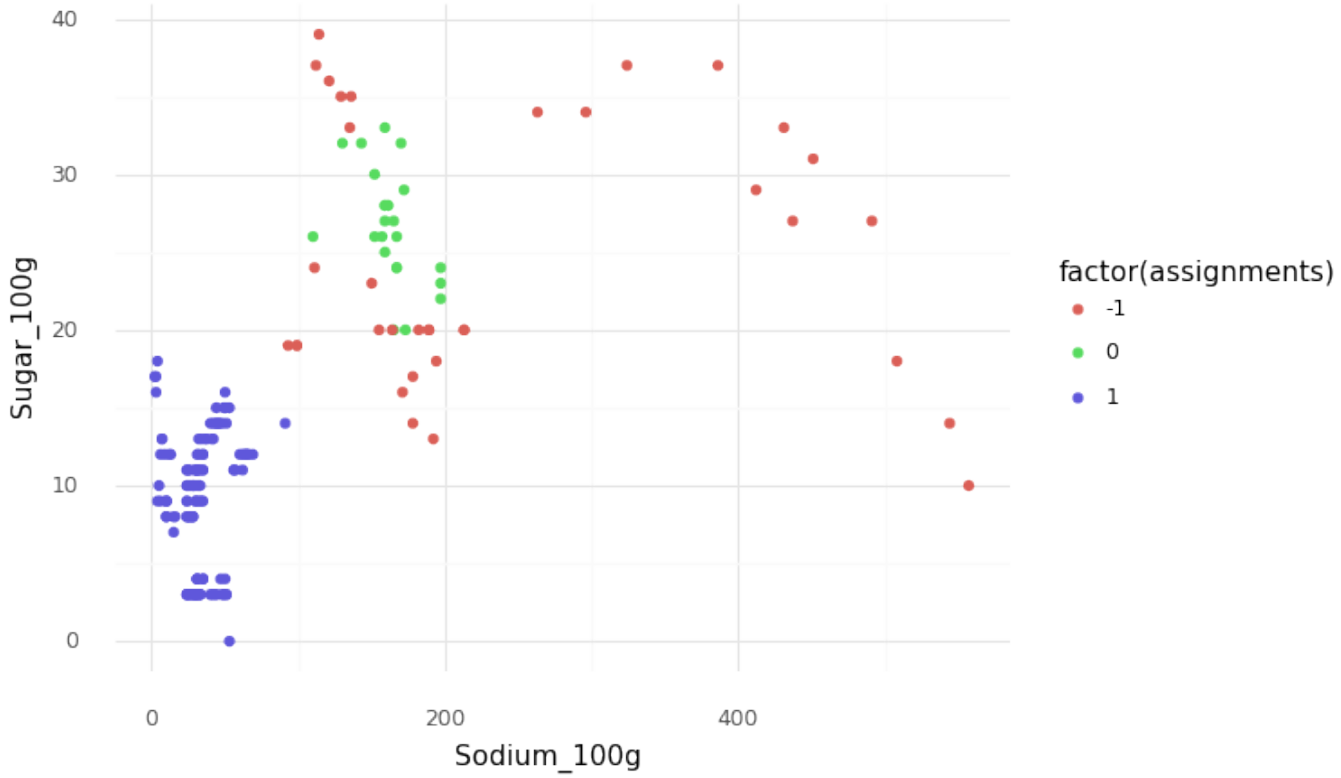



```
<ggplot: (8754486296581)>
```

For this scatter plot, the blue cluster (0) represents a donut that is low in Sodium and low in Total Fat. For these reasons, I classified this donut as "very healthy". For the red cluster (-1), it has high variance across the entire graph, showing average sodium and above average total fat. For these reasons, I classified this cluster as a normal donut, since total fat it usually high in donts, with an average amount of sodium.

```
#plot of Sodium_100g vs. Sugar_100g
ggplot(df, aes(x = 'Sodium_100g', y = 'Sugar_100g', color = 'factor(assignments)'))
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```

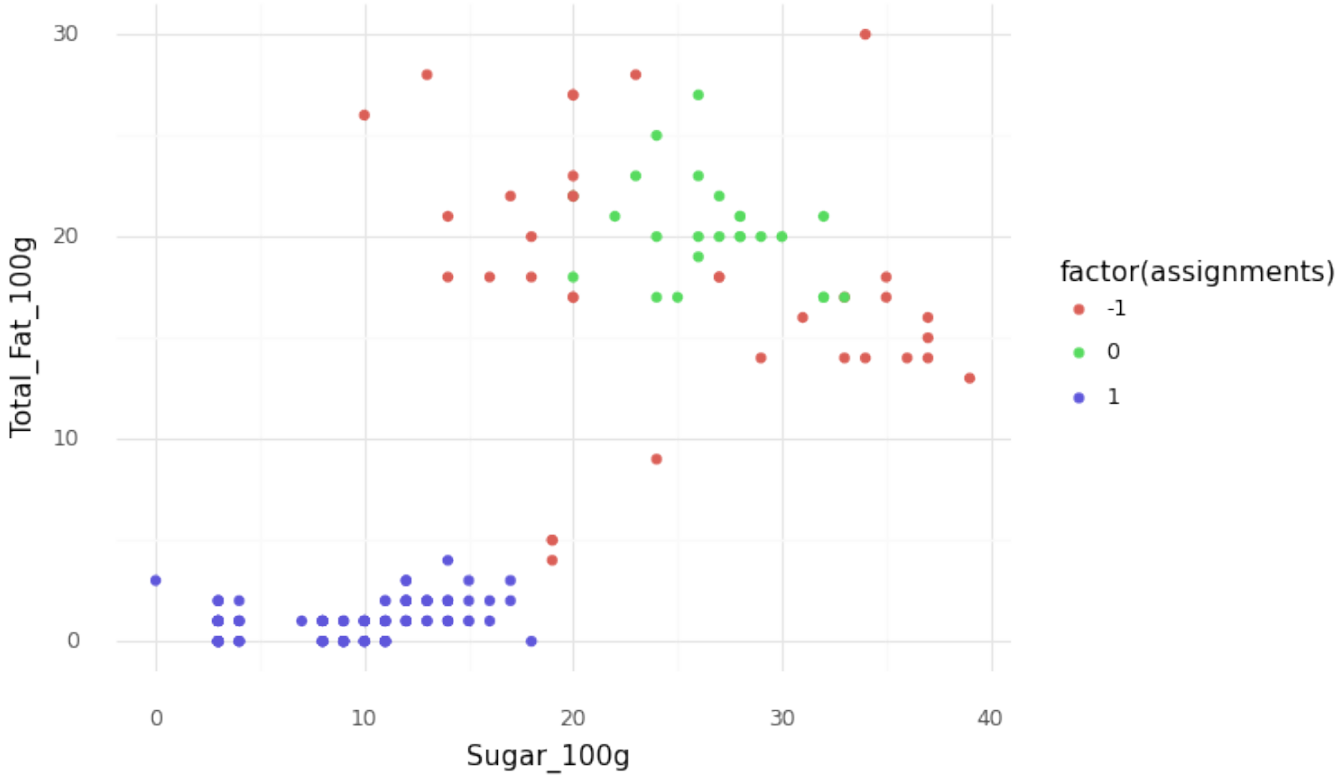



```
<ggplot: (8754486246625)>
```

For this scatterplot, the blue cluster (1) represents a donut that is low in sugar and low in sodium, which led me to classify this cluster as a "vegan donut". The green cluster (0) shows high sugare and below average sodium, which led me to classify this cluster as a "sugar-coated donut". The red cluster (-1) shows high variance across the entire graph, also showing signs of high sugar and high sodium. For these reasons, I classified this donut as a "psycho donut".

```
#plot of Sugar_100g vs. Total_Fat_100g
ggplot(df, aes(x = 'Sugar_100g', y = 'Total_Fat_100g', color = 'factor(assignments)
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```




```
<ggplot: (8754486233949)>
```

In this scatterplot, the blue cluster (1) represents a donut with low Total Fat and low Sugar, which led me to classify this cluster as a "Whole Foods donut". The red cluster (-1) represents a donut with high total fat and an average amount of sugar (highly varied), which led me to classify this cluster as a "normal donut". Lastly, the green cluster (0) represents a donut with high fat and high sugar, which led me to classify this cluster as an "unhealthy donut"

E)

By using the scatterplots and the silhouette score, I can determine if adding in this new factor variable helped cluster the data. I decided to choose Saturated_Fat_100g to add to the HAC model for a couple reasons. I tested a couple different _100g variables to see which would have the most improvement to the model, and Saturated_Fat_100g was one of the better variables. In addition, when thinking about different factors that make up calories in foods, saturated fat seemed appropriate when compared to variables Total_Fat_100g, Sugars_100g, and Sodium_100g, and looked like it would have the biggest correlation (which subsequently leads to better clustering).

```
features = df[['Sodium_100g', 'Sugar_100g', 'Total_Fat_100g', 'Saturated_Fat_100g']]
z = StandardScaler()
z.fit(features)

X = z.transform(features)

hac2 = AgglomerativeClustering(n_clusters = 3,
                               affinity = "euclidean",
                               linkage = "ward")
hac2.fit(X)
```

```
AgglomerativeClustering(n_clusters=3)
```

```
membership = hac2.labels_
```

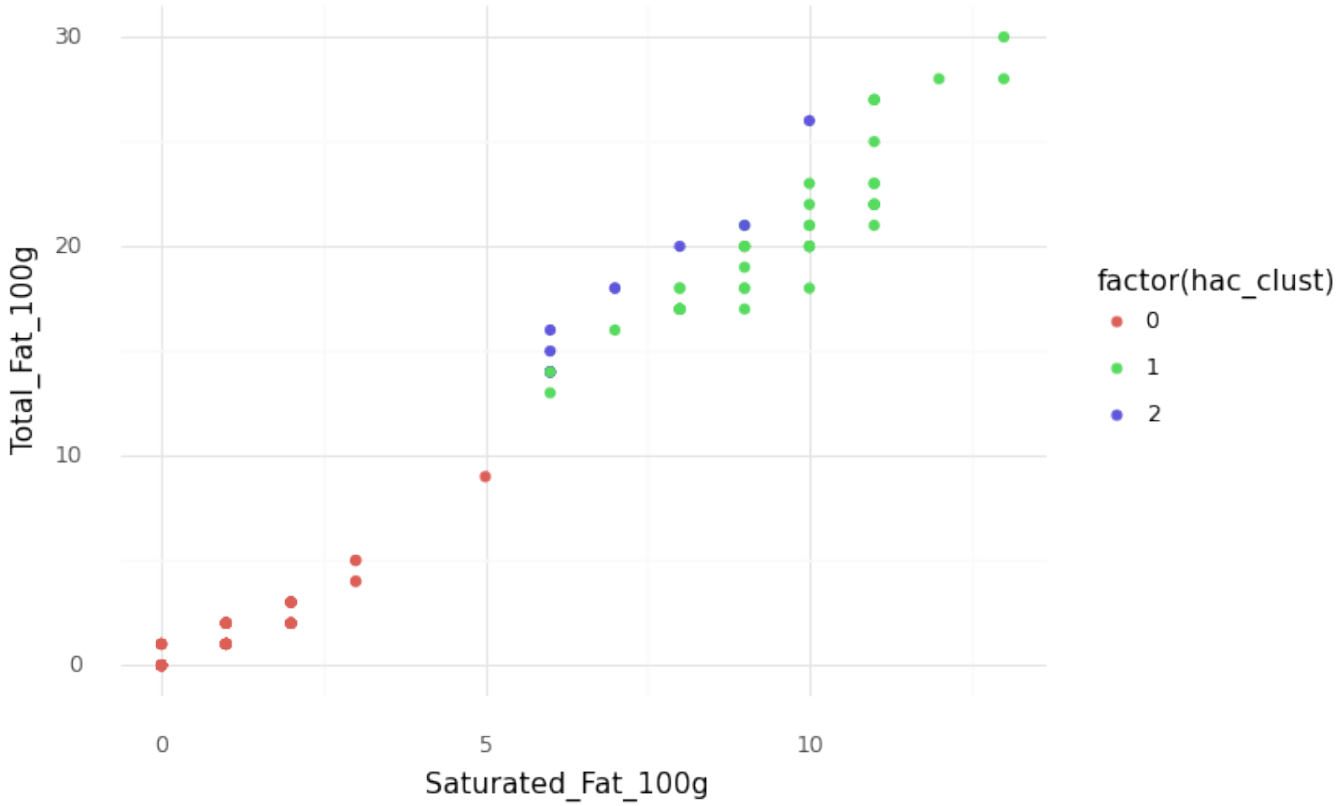
```
silhouette_score(X, membership)
```

```
0.7532544502237266
```

```
df["hac_clust"] = membership
```

```
ggplot(df, aes(x = 'Saturated_Fat_100g', y = 'Total_Fat_100g', color = 'factor(hac_
```

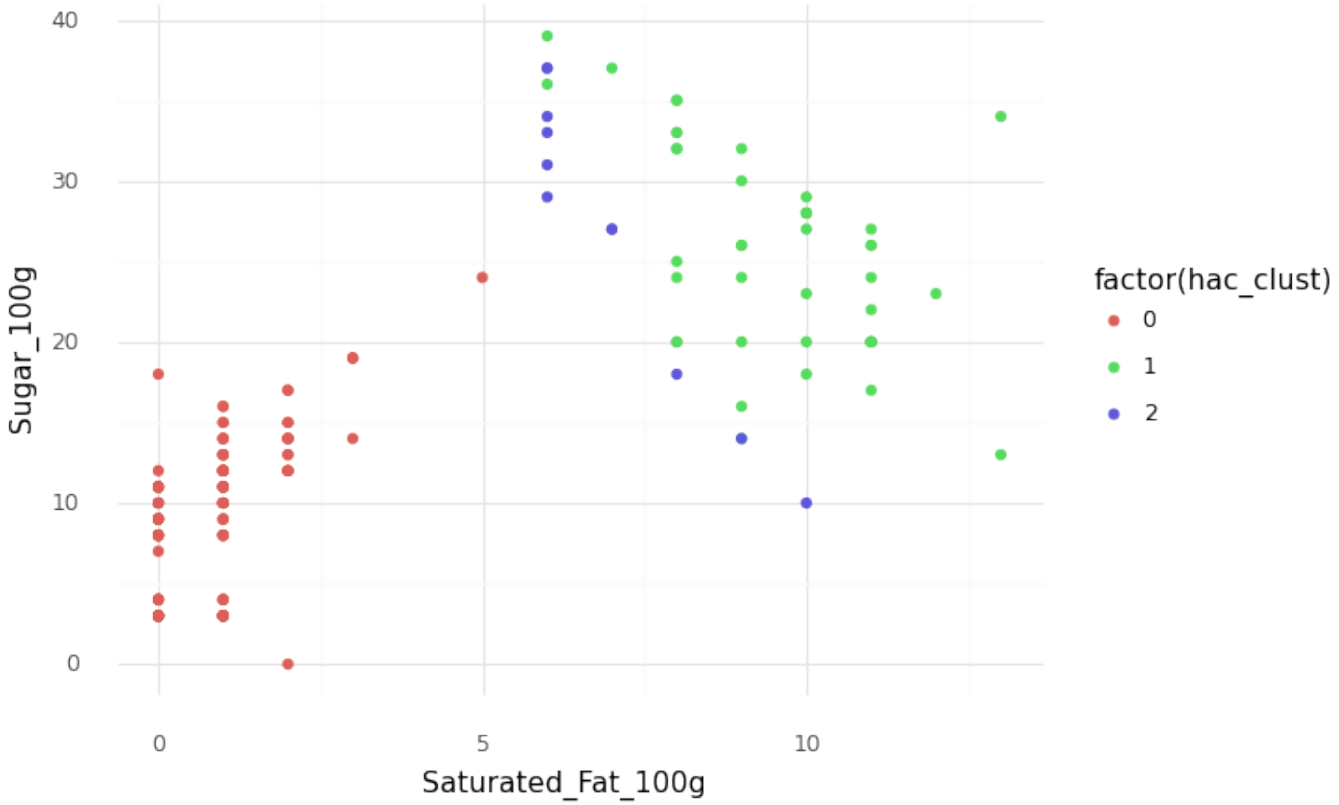
```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```




```
<ggplot: (8755364294465)>
```

```
#plot of Saturated_Fat_100g vs. Sugar_100g
ggplot(df, aes(x = 'Saturated_Fat_100g', y = 'Sugar_100g', color = 'factor(hac_clus
```

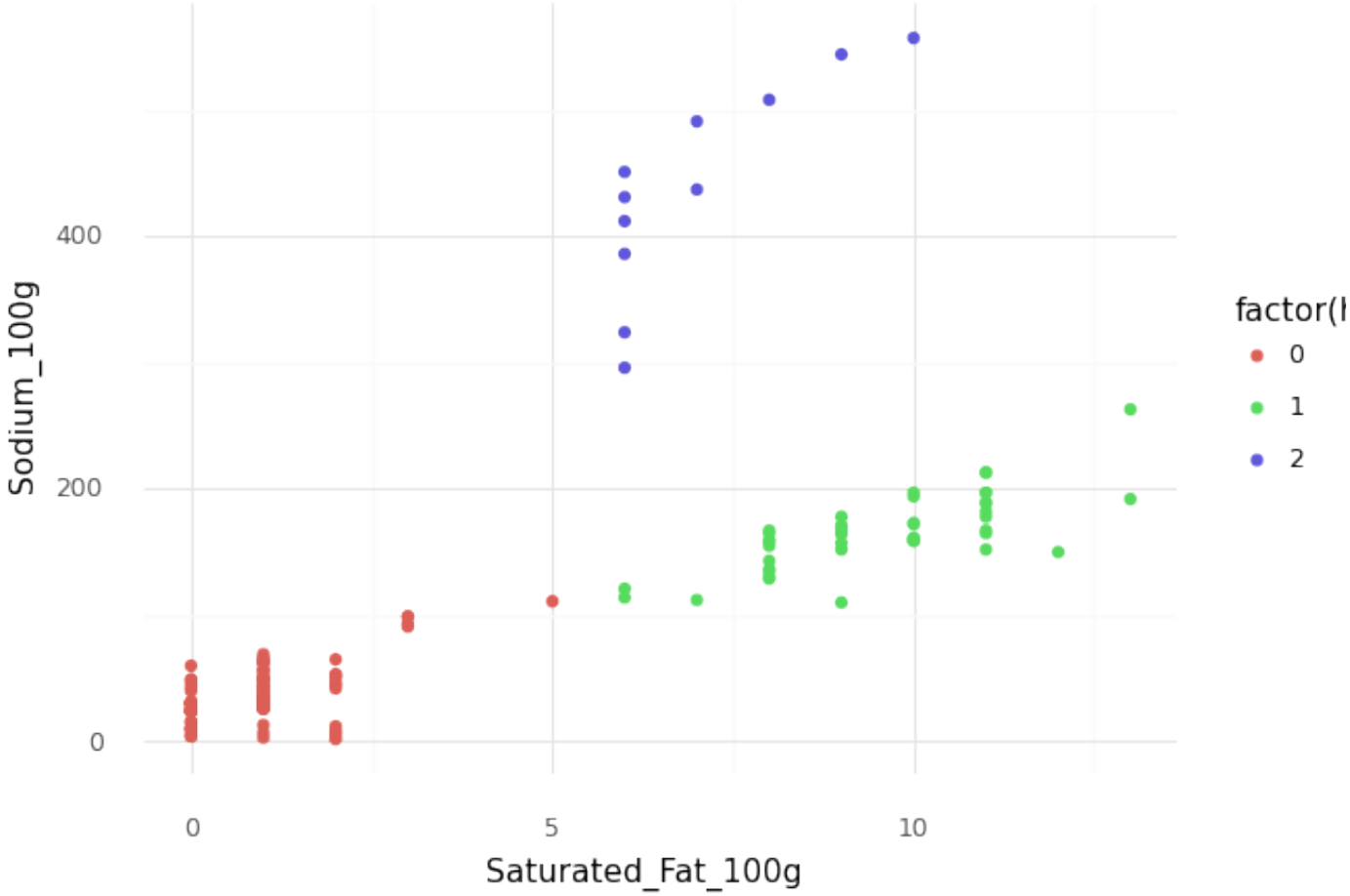
```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```




```
<ggplot: (8755369438937)>
```

```
#plot of Sugar_100g vs. Saturated_Fat_100g
ggplot(df, aes(x = 'Saturated_Fat_100g', y = 'Sodium_100g', color = 'factor(hac_clu
```

```
/usr/local/lib/python3.7/dist-packages/plotnine/utils.py:1246: FutureWarning:
  if pdtypes.is_categorical(arr):
```




```
<ggplot: (8755369427757)>
```

By adding Saturated_Fat_100g, the new model actually performed better than the previous HAC model. When looking at silhoutte scores, the new HAC model (model with Saturated_Fat_100g), this model poroduced a score of .7533. This is a very good score (closer to 1 means model performed well) and shows us that this K-Means model was able to effectively cluster the data.

When examining the ggplots created using the new variables, the HAC model was able to create distinct clusters for a majority of the data (exluding the second scatterplot). The clusters created show good cohesion within the clusters, and good separation for the most part.

This new model performed better than my previous HAC model, and this can be clearly shown by the resulting silhouette scores. The previous HAC model produced a score of .7386, while the new model produced a score of .7533.