DAT405 Assignment 3 – Group 53

Venkata Sai Dinesh Uddagiri - (24 hrs)
Madumitha Venkatesan - (24 hrs)

November 22, 2022

Problem 1

Show the distribution of phi and psi combinations using

a. A scatter plot

```
#Load the data to data frame
protein_chain_df_raw=pd.read_csv("assignment3-data-1.csv")
protein_angle_df=protein_chain_df_raw[[’phi’,’psi’]]
plt.figure(figsize=(8, 8))
#Plot a scatter plot (phi vs psi)
plt.scatter(protein_angle_df[’phi’],protein_angle_df[’psi’],s=0.5)
# Defining labels and title of plot
plt.xlabel(’phi’, fontsize=18)
plt.ylabel(’psi’, fontsize=18)
plt.title(’Scatter plot of bond rotation by phi and psi angle’, fontsize=18)
plt.show()
```

Listing 1: Distribution of phi and psi combinations using scatter plot code

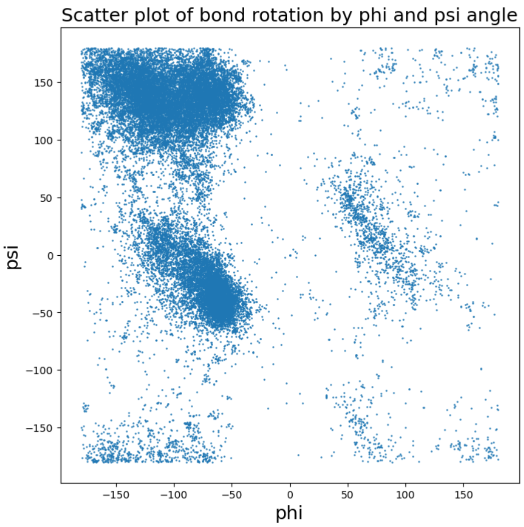

Figure 1: Distribution of phi and psi combinations using scatter plot

b. A heatmap

```
plt.figure(figsize=(8, 8))
#Plot a Heat map (phi vs psi)
plt.hist2d(protein_angle_df['phi'],protein_angle_df['psi'], bins=50)
# Plot a colorbar
plt.colorbar().set_label('Count of samples', fontsize=18)
# Defining labels and title of plot
plt.xlabel('phi', fontsize=18);
plt.ylabel('psi', fontsize=18)
plt.title('Heatmap of bond rotation by phi and psi angle', fontsize=18)
plt.show()
```

Listing 2: Distribution of phi and psi combinations using Heat map code

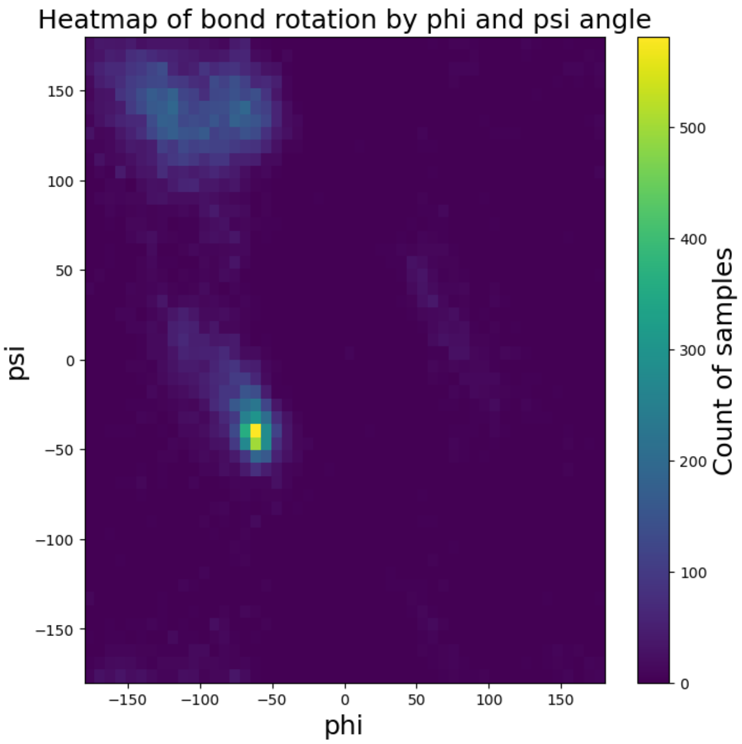

Figure 2: Distribution of phi and psi combinations using Heat map

Problem 2

Use the K-means clustering method to cluster the phi and psi angle combinations in the data file.

a. Experiment with different values of K. Suggest an appropriate value of K for this task and motivate this choice.

```
# Defining list of k values to find appropriate value of K
k_values=list(range(1, 7))
rows=2
coloumns=3
fig, ax = plt.subplots(rows,coloumns, figsize=(20, 8))
count=0
mean_values=[]
silhouette_scores=[]
for i in range(rows):
    for j in range(coloumns):
        k=k_values[count]
        count=count+1
        # Instance of Kmeans and fit the model
        km = KMeans(n_clusters=k, random_state=0).fit(protein_angle_df)
        pred = km.predict(protein_angle_df)
        # Append the kmean inertia to mean_values list, to plot elbow plot later
        mean_values.append(km.inertia_)
        # Scatter plot with kmeans clustring
        ax[i][j].scatter(x=protein_angle_df[’phi’],y=protein_angle_df[’psi’],c=pred,s
    =0.5)
        ax[i][j].scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], c=’red’
    , marker=’x’,s=5)
        # Defining labels and title for each subplot
        ax[i][j].set_xlabel(’phi\n’)
        ax[i][j].set_ylabel(’psi’)
        ax[i][j].set_title(’k = ’+str(k))
        # Append the silhouette_score to silhouette_scores list, to validate cluster
    later
        if(k!=1):
            score = silhouette_score(protein_angle_df, km.labels_, metric=’euclidean’)
            silhouette_scores.append(score)
# Defining space and title for plot
plt.subplots_adjust(wspace=0.3, hspace=0.3)
fig.suptitle(’KMeans cluster, scatterplots with k ranging 2 to 9’, fontsize=18)
plt.show()
```

Listing 3: KMeans cluster - Experiment with different values of K code

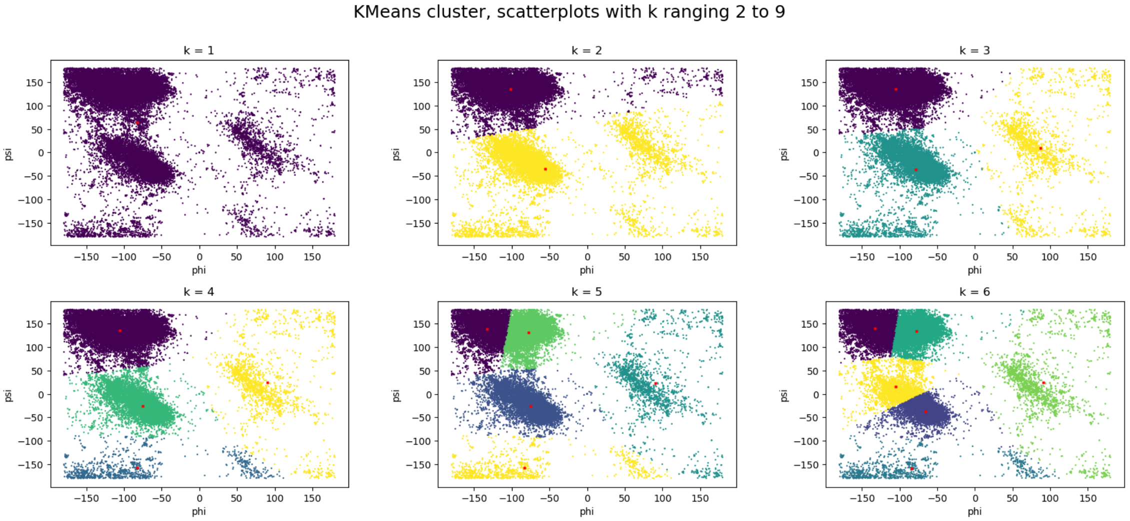

Figure 3: KMeans cluster, Experiment with different values of K

```
#Elbow plot to find Optimal k value
plt.plot(k_values, mean_values, marker='o')
# Defining labels and title of plot
plt.xlabel('k Value')
plt.ylabel('Distance')
plt.title('Elbow plot to find optimal kvalue')
plt.show()
```

Listing 4: Elbow plot - Experiment with different values of K code

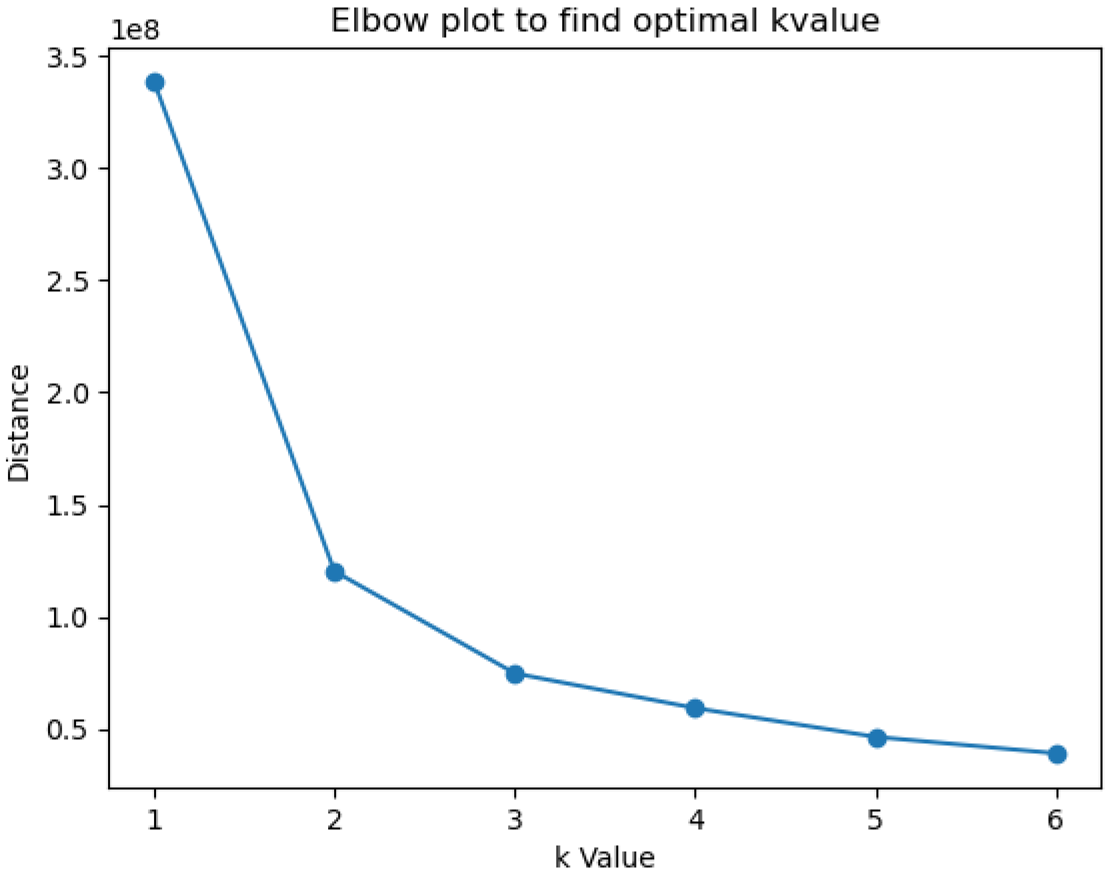

Figure 4: Elbow plot, Experiment with different values of K

We have plotted the kmean cluster scatter plot for k between 1 and 6 to identify the suitable value for k (k=1 is merely included to calculate kmean inertia, which is useful for plotting elbow plot). When the k=2 cluster groups are too far dispersed, the k=3 groupings appear excellent because the centroid is in a dense area, for k=4 groupings looks good similarly to k=3, but the k=5 and 6 groupings look strange since the dense areas are fragmented into distinct clusters. We can infer from the observation above that either 3 or 4 is a suitable choice for k. But we wouldn’t be able to conclude on k value as this only our observation. So, We have ploted an elbow graph in order to analytically find the proper k value. The elbow graph has diameter, which was determined using inertia, is on the y axis while the k values are on the x axis. Inertia of k means cluster is ’Sum of squared distances of samples to their closest cluster center’. By observing the Elbow graph, we observed the elbow at k=3. Therefore, we propose that appropriate value for k is 3. Which also aligns with the prior knowledge we gained from studying scatter plots for various k values and heat maps.

b. Validate the clusters that are found with the chosen value of K. Hint: See the slides for Lecture 6, Section Validating clustering.

```
import math
fig, ax = plt.subplots(2,2, figsize=(12, 8))
for i in range(2):
    for j in range(2):
        #25 percent random indexes array of protein_angle_df
        random_data_drop = np.random.choice(protein_angle_df.index, math.floor(0.25 *
    protein_angle_df.shape[0]), replace=False)
        #Drop 25 percent indexes in random_data_drop array, from data frame
        protein_angle_df_subset = protein_angle_df.drop(random_data_drop)
         # Instance of Kmeans and fit the model with subset after droping 25%
        km = KMeans(n_clusters=3, random_state=0).fit(protein_angle_df_subset)
        pred = km.predict(protein_angle_df_subset)
         # Scatter plot with kmeans clustring
        ax[i][j].scatter(x=protein_angle_df_subset[’phi’],y=protein_angle_df_subset[’
    psi’],c=pred,s=0.5)
        ax[i][j].scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], c=’red’
    , marker=’x’,s=5)
        # Defining labels and title for each subplot
        ax[i][j].set_xlabel(’phi\n’)
        ax[i][j].set_ylabel(’psi’)
        ax[i][j].set_title(’k =’+str(3))
# Defining space and title for plot
plt.subplots_adjust(wspace=0.2, hspace=0.3)
plt.suptitle(’Validating Kmean cluster with k =3, By removing random 25 percent of
    data each time’, fontsize=16)
plt.show()
protein_angle_df_subset.shape
```

Listing 5: Validate KMean cluster - with k equal to 3 using Stability on subsets method

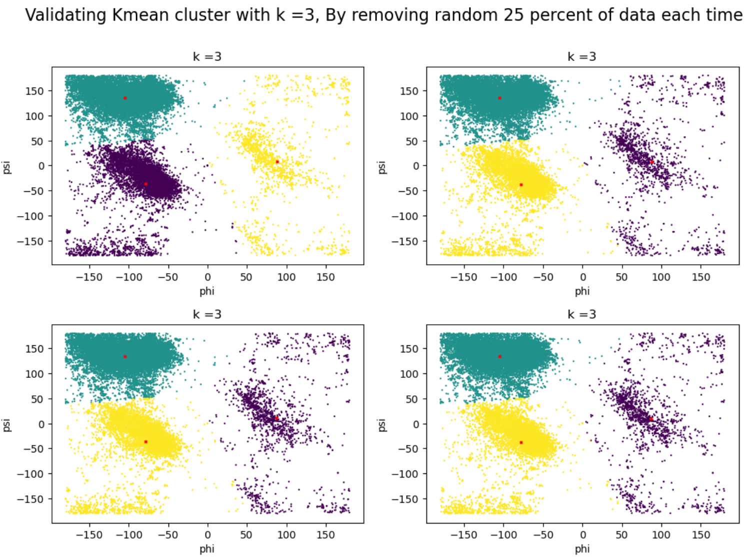

Figure 5: Validate KMean cluster - with k equal to 3 using Stability on subsets method

```
k_values=list(range(2, 7))
#silhouette_scores plot to Validate clusters
plt.plot(k_values, silhouette_scores, marker=’o’)
# Defining labels and title of plot
plt.xlabel(’k Value’)
plt.ylabel(’silhouette_scores’)
plt.xlim(1,7)
plt.title(’KMean cluster validation using silhouette_scores’)
plt.show()
```

Listing 6: Validate KMean cluster - with k equal to 3 using Silhouette coefficient code

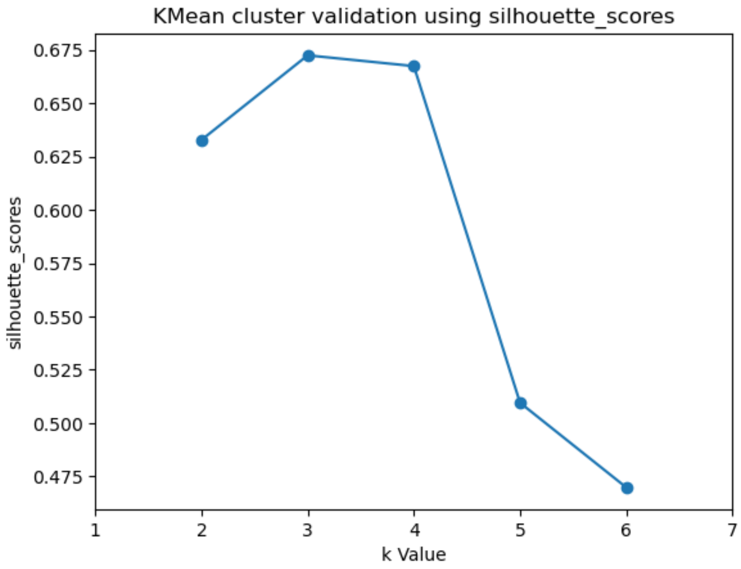

Figure 6: Validate KMean cluster - with k equal to 3 using Silhouette coefficient

To further check whether k=3 is good value for clustering the data or not. We have used the Stability on subsets method, which is, if removing a proportion of random points that does not change the clustering fundamentally, then clustering can be considered as stable. So to validate clustering using stability on subsets method, we have removed the 25 percent of data randomly and plotted KMean scatter plots for 4 different data subsets. By observing the scatter plots we didn't find any fundamental changes in clustering. Thus k=3 is appropriate.

We also used another technique to validate appropriate k value for cluster. Which is using Silhouette coefficient. We have obtained the Silhouette coefficient for for k values 2 to 6 and plotted the graph between Silhouette coefficient and k value. By observing the graph we can see that k=3 has highest Silhouette coefficient value. Thus by above 2 validations we can say that k=3 is appropriate.

c. Do the clusters found in part (a) seem reasonable?

By refrence http://bioinformatics.org/molvis/phipsi/ the physically allowed angle combinations (that avoid clashes) correspond largely to the secondary structures observed in proteins are alpha helices, beta sheets, and turns. In the same link mentioned, above we can observe the Ramachandran Plot for different phi and psi angle, the conformations are called beta sheets, which is represent in upper left cluster, alpha helics, represented below beta sheets and turns are represented in right of the plot at center. By making comparision of above refrence with our kmean scatter plot with k=3, the cluster obtained covered all the data points which are away from dense areas also. But in refrence plot given, cluster is only the area with dense plots. By this observation we can say that k means cluster also includes noise/outlier data in to there clusters. One more comparision is by observing the refrence plot, we noticed cut in beta sheets cluster. As the phi and psi angles cover 360 degrees and each axis warap around. The cut portion of beta sheets can be observed below alpha helics. Due to this when we performed KMean cluster on data we have one cluster data is merged in to another cluster. By above observations we can say that clusters found are not reasonable completely.

d. Can you change the data to get better results (or the same results in a simpler way)? (Hint: since both phi and psi are periodic attributes, you can think of shifting/translating

them by some value and then use the modulo operation.)

```
phi_shift=230
psi_shift=130
#Creating copy of original data frame, for shifting phi&psi angles without modifying
    original data frame
protein_angle_shift=protein_chain_df_raw.copy()
#shifting phi angle by 230 and performing modulo operation by 360
protein_angle_shift['phi']=(protein_angle_shift['phi']+phi_shift)%360
#shifting phi angle by 110 and performing modulo operation by 360
protein_angle_shift['psi']=(protein_angle_shift['psi']+psi_shift)%360
 # Instance of Kmeans and fit the model
km = KMeans(n_clusters=3, random_state=0).fit(protein_angle_shift[['phi','psi']])
pred = km.predict(protein_angle_shift[['phi','psi']])
plt.figure(figsize=(8, 8))
#Plot a Kmean scatter plot (phi vs psi) after shifting angles
plt.scatter(x=protein_angle_shift['phi'],y=protein_angle_shift['psi'],c=pred,s=0.5)
plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], c='red', marker='x',
    s=5)
# Defining labels and title of plot
plt.xlabel('phi_shifted')
plt.ylabel('psi_shifted')
plt.title('KMean Scatter plot by shifting phi and psi angle\n k = 3')
plt.show()
```

Listing 7: KMean Scatter plot by shifting phi and psi angle code

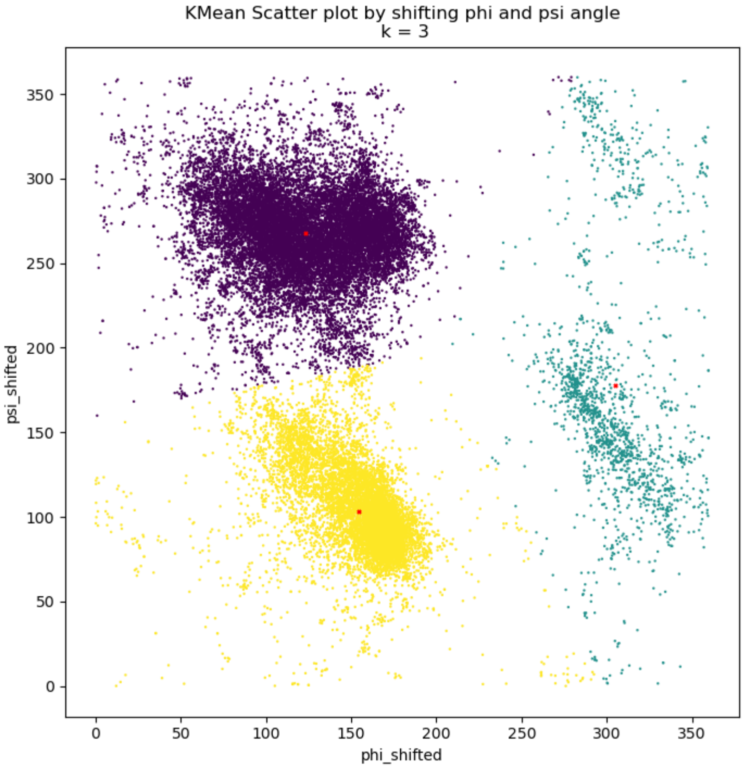

Figure 7: KMean Scatter plot by shifting phi and psi angle

If we shift the angle of phi and psi by 180 degrees, the range (-180 ,180) changes to (0 ,360). By this we can obtain same plot with change, only in range of x axis and y axis. But obtaining the same plot doesn't make any sense and doesn't make results better. By cleanly observing the scatter plot in 1(a) we notice that plot is like a world map, each axis wraps around. By observing the scatter plot 1(a) clearly, we notice there is cut in beta sheets cluster at top and top left. By our observation we noticed that their is separations between the alpha helices and beta sheets at bottom left around psi -130 degrees. Similarly we observed the separation of beta sheets at top right around the phi 150 degrees. So we have shifted our phi angle by 210 degrees(360-150) and psi angle by 130 degrees to obtain the x axis and y axis in the range of 0 to 360 degrees with out any cut in clusters. By this we have obtained better results.

Problem 3

Use the DBSCAN method to cluster the phi and psi angle combinations in the data file.

```
#Defining eps and min samples
eps = 11
min_samples = 100
# Instance of DBSCAN and fit the model
dbscan = DBSCAN(eps = eps, min_samples = min_samples)
pred = dbscan.fit_predict(protein_angle_shift[[’phi’, ’psi’]])
plt.figure(figsize=(8, 8))
# Scatter plot with DBSCAN clustring
plt.scatter(protein_angle_shift[’phi’], protein_angle_shift[’psi’], c=pred, s=0.5)
# Defining labels and title of plot
plt.title(’DBSCAN clustring with eps =’+str(eps)+’, min smaples =’+ str(min_samples))
plt.xlabel(’phi_shifted’)
plt.ylabel(’psi_shifted’)
plt.show()
```

Listing 8: DBSCAN method to cluster the phi and psi angle combinations code

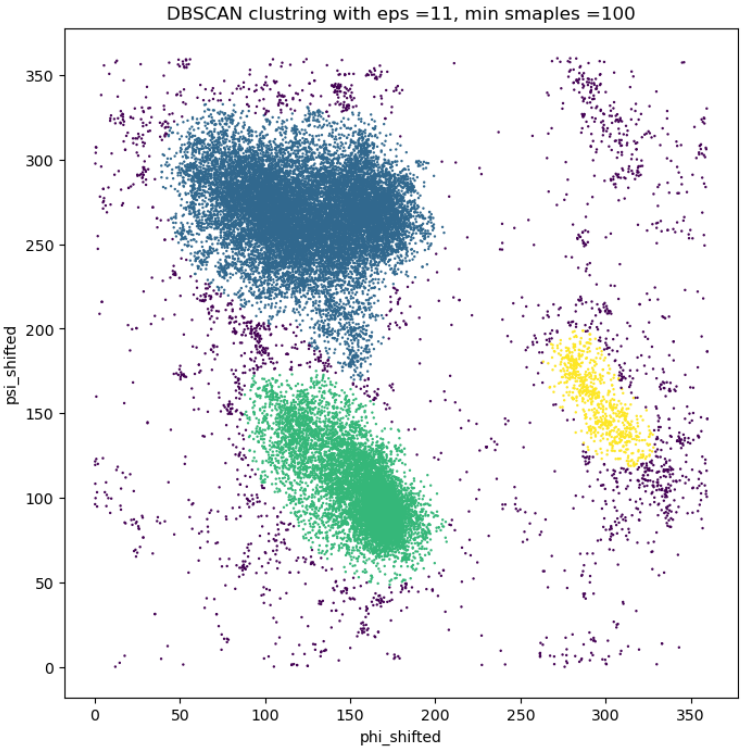

Figure 8: DBSCAN cluster scatter plot for the phi and psi angle combinations

a. Motivate:
i. the choice of the minimum number of samples in the neighbourhood for a point to be considered as a core point, and
ii. the choice of the maximum distance between two samples belonging to the same neighbourhood (“eps” or “epsilon”).

In general we need to have domain knowledge to set minimum number of samples value and maximum distance between two samples value. As we don’t have good understanding with data, we have followed trail and error approach. The data set consists of 30000 samples approximately, we should have minimum number of samples value little larger. If the minimum number of samples value is to Small then more clusters will be obtained. If the maximum distance between two samples value is large then clusters obtained will get merged. By having all the considerations in mind we have changed the values of maximum distance between two samples and minimum number of samples randomly and observed the behaviour of clusters formed. By the observations made in above questions and by references from http://bioinformatics.org/molvis/phipsi/ we considered obtaining 3 clusters is suitable for this data set. So we have tested by changing minimum number of samples and maximum distance between two samples value till we found 3 clusters.

At last we have made we have made choice of the minimum number of samples to be 100 and maximum distance between two samples belonging to the same neighbourhood is to be 11.

b. Highlight the clusters found using DBSCAN and any outliers in a scatter plot. How many outliers are found? Plot a bar chart to show which amino acid residue types are most frequently outliers.

```
protein_angle_shift[’label’]=dbscan.labels_
#creating new outlier data frame
outlier_protein_df=protein_angle_shift[protein_angle_shift.label == -1].copy()
#count of outliers for each residual type
outlier=outlier_protein_df[’residue name’].value_counts(sort=True)
# Bar plot to show which amino acid residue types are most frequently outliers
plt.bar(outlier.index,outlier.values)
print(’Total number of outlier are: ’,sum(outlier.values))
# Defining labels and title of plot
plt.title(’Number of outliers per residue type’)
plt.xlabel(’\nresidue names’)
plt.ylabel(’Number of outliers’)
plt.xticks(rotation =’vertical’)
plt.show()
```

Listing 9: Bar plot - Number of outliers per residue type code

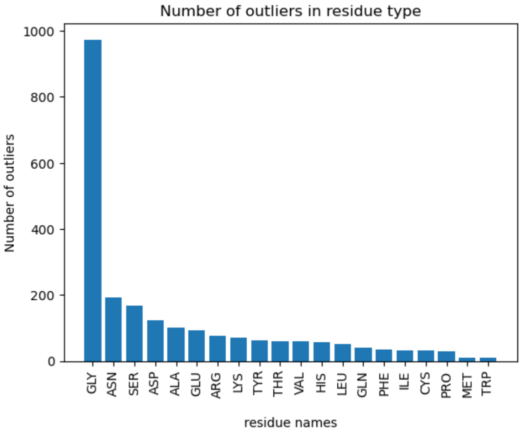

Figure 9: Bar plot - Number of outliers per residue type

The clusters found using the DBSCAN method, can be observed in figure 8, clusters are highlighted with blue, green and yellow colours. The outliers are highlighted in Violet colour. The total number of outliers in the scatter plot are 2275. From the bar plot we can observe that GLY amino acid residue type are most frequent outliers nearly 1000 which are approximately equal to 45 percent of total amino acids residue types outlier.

c. Compare the clusters found by DBSCAN with those found using K-means.

By comparing plots figure 7 and figure 8, we can see that both techniques were successful in grouping the data points into three clusters. K-means clustering method, include all the data points in to the clusters, even though points are widely scattered, which is disadvantage for this data. Where as DBSCAN exclude points which are present in low density area such as outliers/noise points. K-means cluster works best when cluster shapes are spherical. DBSCAN is capable of locating clusters of any shape. Conclusion: Since there are numerous data points that are dispersed and would count as noise, DBSCAN was a preferable strategy for clustering this data set.

d. Discuss whether the clusters found using DBSCAN are robust to small changes in the minimum number of samples in the neighbourhood for a point to be considered as a core point, and/or the choice of the maximum distance between two samples belonging to the same neighbourhood (“eps” or “epsilon”).

```
rows=3
coloumns=5
fig, ax = plt.subplots(rows,coloumns, figsize=(20, 8))
#Defining eps and min samples
eps_values=[9,11,13]
min_samples_values=[30,70,100,130,180]
for i in range(rows):
    eps=eps_values[i]
    for j in range(coloumns):
        min_samples=min_samples_values[j]
        # Instance of DBSCAN and fit the model
        db = DBSCAN(eps = eps, min_samples = min_samples)
        pred_dbscan = db.fit_predict(protein_angle_shift[[’phi’, ’psi’]])
        # Scatter plot with DBSCAN clustring
        ax[i][j].scatter(x=protein_angle_shift[’phi’],y=protein_angle_shift[’psi’],c=
    pred_dbscan,s=0.5)
        # Defining labels and title for each subplot
        ax[i][j].set_xlabel(’phi\n’)
        ax[i][j].set_ylabel(’psi’)
        ax[i][j].set_title(’eps =’+str(eps_values[i])+’ & min_samples =’+str(
    min_samples_values[j]))
# Defining labels and title of plot
plt.subplots_adjust(wspace=0.3, hspace=0.7)
fig.suptitle(’Scatter plots with DBSCAN cluster with small change in eps/min_samples’,
     fontsize=16)
plt.show()
```

Listing 10: Validate clusters found using DBSCAN are robust to small changes code

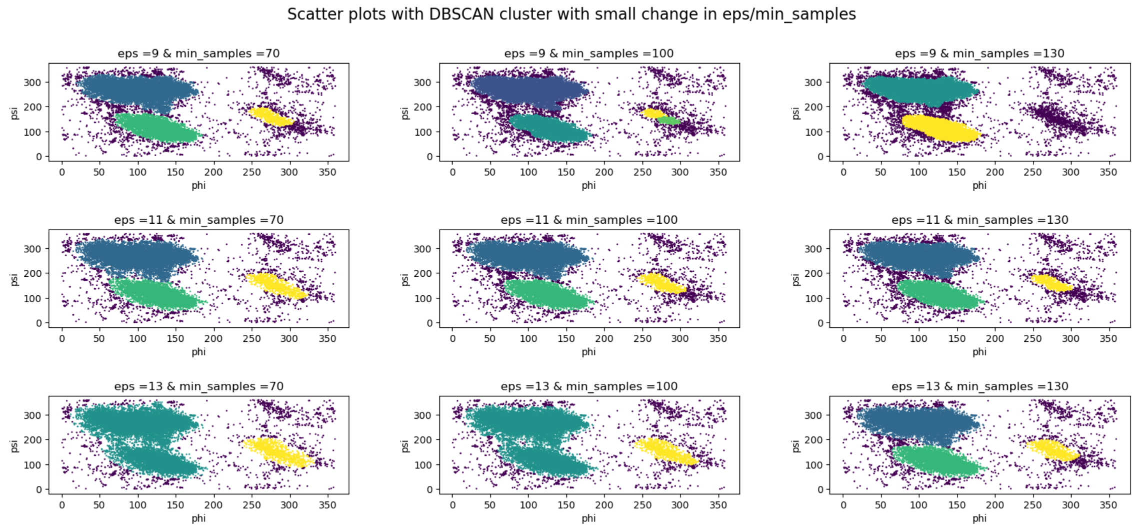

Figure 10: Scatter plots with DBSCAN cluster with small change in eps or min samples

We are considering increasing/decreasing 30(approximately one percent of data) samples as small change for min_samples and increasing/decreasing eps value by two as small change for eps. So according to our consideration we have plotted the DBSCAN scatter plots for values eps = [9, 11, 13] and min_samples = [70, 100, 130]. By looking at the plots, we can conclude that in our instance, with a little modification in min samples, the clusters produced remain unchanged. The clusters have merged when the eps value has increased, and a new cluster has been generated when the eps value has decreased. In contrast, if we changed both the eps and min samples, sometimes new clusters would develop and other times they would merge. But by over all observation we can say that DBSCAN is not robust to small changes.

Problem 4

The data file can be stratified by amino acid residue type.

a. Use DBSCAN to cluster the data that have residue type PRO. Investigate how the clusters found for amino acid residues of type PRO differ from the general clusters (i.e., the clusters that you get from DBSCAN with mixed residue types in question 3).

```
#loading PRO amino acid data
PRO_protein_data=protein_angle_shift[protein_angle_shift[’residue name’]==’PRO’]
fig, ((ax1, ax2)) = plt.subplots(1, 2,figsize=(12,8))
#Scatter plot for PRO amino acid
ax1.scatter(PRO_protein_data[’phi’], PRO_protein_data[’psi’], s=0.5)
# Defining labels and title of ax1 subplot
ax1.set_title(’Scatter plot for PRO amino acid\n’)
ax1.set_xlabel(’phi_shifted’)
ax1.set_ylabel(’psi_shifted’)
ax1.set_xlim(0,360)
#Defining eps and min samples
eps = 11
min_samples = 30
# Instance of DBSCAN and fit the model for PRO amino acid
db = DBSCAN(eps = eps, min_samples = min_samples)
pred = db.fit_predict(PRO_protein_data[[’phi’, ’psi’]])
plt.figure(figsize=(8, 8))
labels = db.labels_
n_noise_ = list(labels).count(-1)
print(’Number of noise points of PRO amino acid: %d’ % n_noise_)
# Scatter plot with DBSCAN cluster for PRO amino acid
ax2.scatter(PRO_protein_data[’phi’], PRO_protein_data[’psi’], c=pred, s=0.5)
# Defining labels and title of ax2 subplot
ax2.set_title(’DBSCAN cluster scatter plot for PRO amino acid\n eps = ’+str(eps)+’ &
    min_samples = ’+str(min_samples))
ax2.set_xlabel(’phi_shifted’)
ax2.set_ylabel(’psi_shifted’)
ax2.set_xlim(0,360)
plt.show()
```

Listing 11: DBSCAN cluster for PRO amino acid code

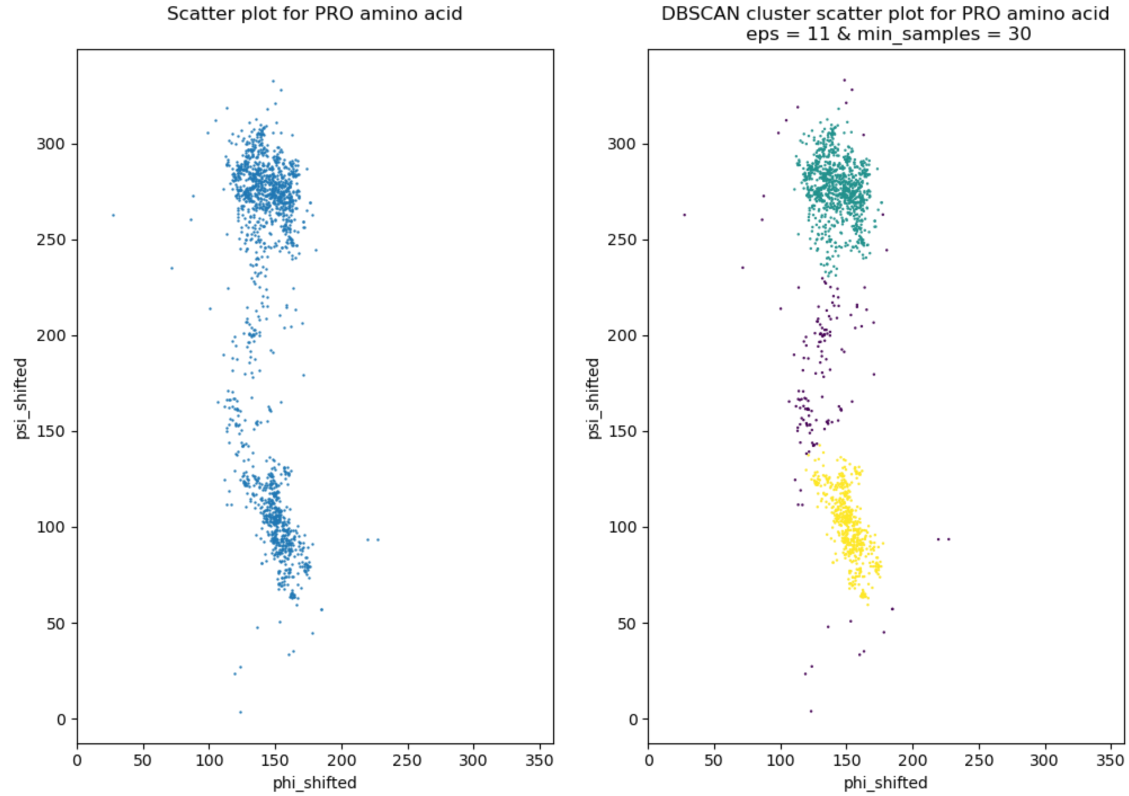

Figure 11: Scatter plot with DBSCAN cluster for PRO amino acid

The PRO residue points can be seen in the scatter plot above, and because they are densely packed and not widely distributed, there will be fewer noise points and outliers. The bar plot in figure 9 also supports this finding, showing that pro residues are correctly classified and rarely comprise outliers. Thus, we think about using the trial and error method to discover optimal values for eps and min samples in order to find two clusters(by observing scatter plot we see data points mostly fit in to two clusters as they are dense and located in left half) for the PRO residue type that have as less outliers. Let’s begin the PRO residue DBSCAN clustering trial and error process with the eps and min samples values we discovered earlier for the whole data, which are eps = 11 and min samples = 100. Although there are two clusters, the noise levels are higher than those observed with the entire data set. Because the data are a subset of the original data, a reduction in min samples might reduce noise points and increase cluster size. By trail and error method we found that eps = 11 and min_samples = 30 is best choice for DBSCAN cluster for PRO amino acid type with less outliers.

When comparing the DBSCAN cluster scatter plot for PRO amino acid with the Original DBSCAN figure with all amino acid data in figure 8, The clusters appear to be subsets of the clusters observed in figure 8. We can see that the clusters are, respectively, subsets of the beta sheets and the alpha helices clusters. We might infer from this picture that the PRO class of amino acid residues only has those two conformations and no turns.

b. Now use DBSCAN to cluster the data that have residue type GLY. Investigate how the clusters found for amino acid residues of type GLY differ from the general clusters.

```
#loading GLY amino acid data
GLY_protein_data=protein_angle_shift[protein_angle_shift[’residue name’]==’GLY’]
fig, ((ax1, ax2)) = plt.subplots(1, 2,figsize=(12,8))
#Scatter plot for GLY amino acid
ax1.scatter(GLY_protein_data[’phi’], GLY_protein_data[’psi’], s=0.5)
# Defining labels and title of ax1 subplot
ax1.set_title(’Scatter plot for GLY amino acid\n’)
ax1.set_xlabel(’phi_shifted’)
ax1.set_ylabel(’psi_shifted’)
ax1.set_xlim(0,360)
#Defining eps and min samples
eps = 17
min_samples = 40
# Instance of DBSCAN and fit the model for PRO amino acid
db = DBSCAN(eps = eps, min_samples = min_samples)
pred_dbscan = db.fit_predict(GLY_protein_data[[’phi’, ’psi’]])
labels = db.labels_
n_noise_ = list(labels).count(-1)
print(’Number of noise points: %d’ % n_noise_)
plt.figure(figsize=(8, 8))
# DBSCAN cluster scatter plot for GLY amino acid
ax2.scatter(GLY_protein_data[’phi’], GLY_protein_data[’psi’], c=pred_dbscan, s=0.5,
    cmap=’gnuplot’)
# Defining labels and title of plot
ax2.set_title(’DBSCAN cluster scatter plot for GLY amino acid\n eps = ’+str(eps)+’ &
    min_samples = ’+str(min_samples))
ax2.set_xlabel(’phi_shifted’)
ax2.set_ylabel(’psi_shifted’)
ax2.set_xlim(0,360)
plt.show()
```

Listing 12: DBSCAN cluster for GLY amino acid code

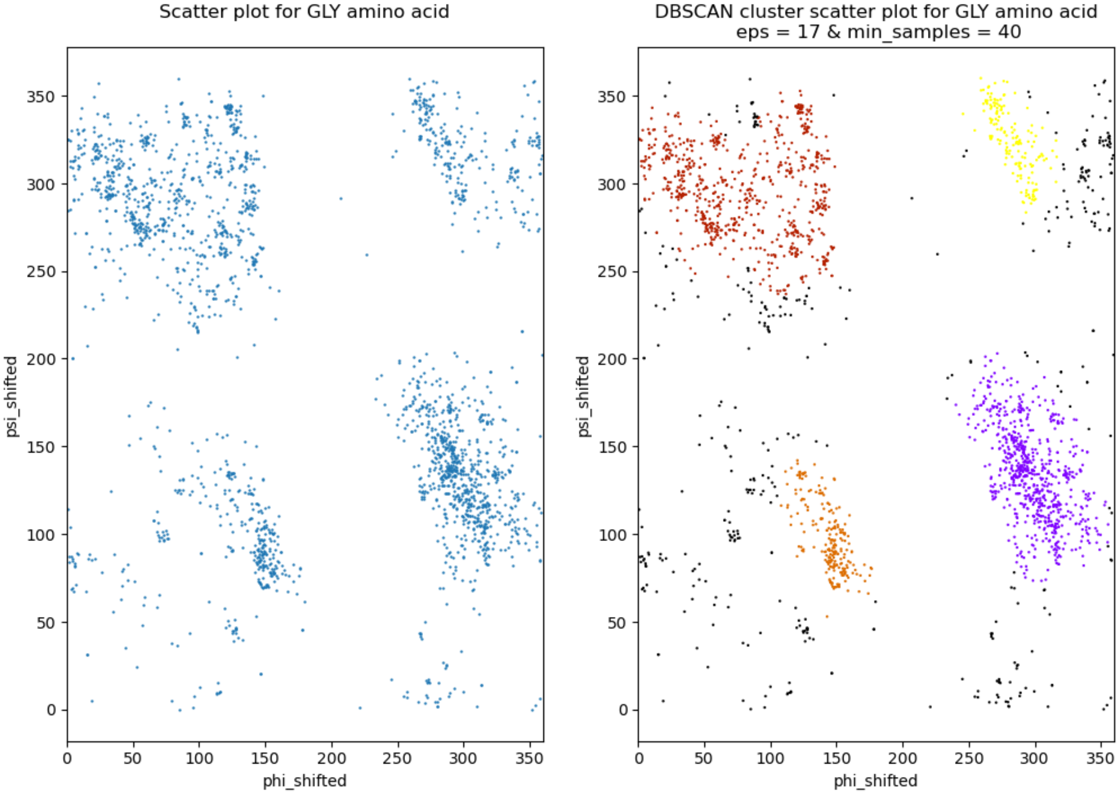

Figure 12: Scatter plot with DBSCAN cluster for GLY amino acid

The GLY residue points can be seen in the scatter plot above, because they are widely distributed, there will be more noise points/outliers. The bar plot in figure 9 also supports this finding, showing that GLY residues comprise of more outliers. Thus, we think about using the trial and error method to discover optimal values for eps and min samples in order to find three clusters(by observing scatter plot we see data points moslty fit in to four clusters) for the GLY residue type that have as less outliers. Let's begin the GLY residue DBSCAN clustering trial and error process with the eps and min samples values we discovered earlier for the whole data, which are eps = 11 and min samples = 100. Which has given only one cluster, to increase the number of clusters we need to decrease the min_samples. By decreasing min samples we got 4 clusters but outliers are high. To decrease the number of outliers we have increased the eps value. By trail and error method we found that eps = 17 and min_samples = 40 is best choice for DBSCAN cluster for GLY amino acid type with less outliers.

When comparing the DBSCAN cluster scatter plot for GLY amino acid with the Original DBSCAN figure with all amino acid data in figure 8, The clusters appear to be subsets of the clusters observed in figure 8 but we have one new cluster obtained. We can see that the three out of 4 clusters are, respectively, subsets of the beta sheets, alpha helices and turns clusters. The new cluster obtained is subset of noise points. But one can not say what the new cluster would imply, with out domain knowledge. By observation from plots above we can say that amino acids of GLY type can be formed with more angle combinations.

References

[1] The Ramachandran Principle Phi and Psi Angles in Proteins: http://bioinformatics.org/molvis/phipsi/