Table of Contents

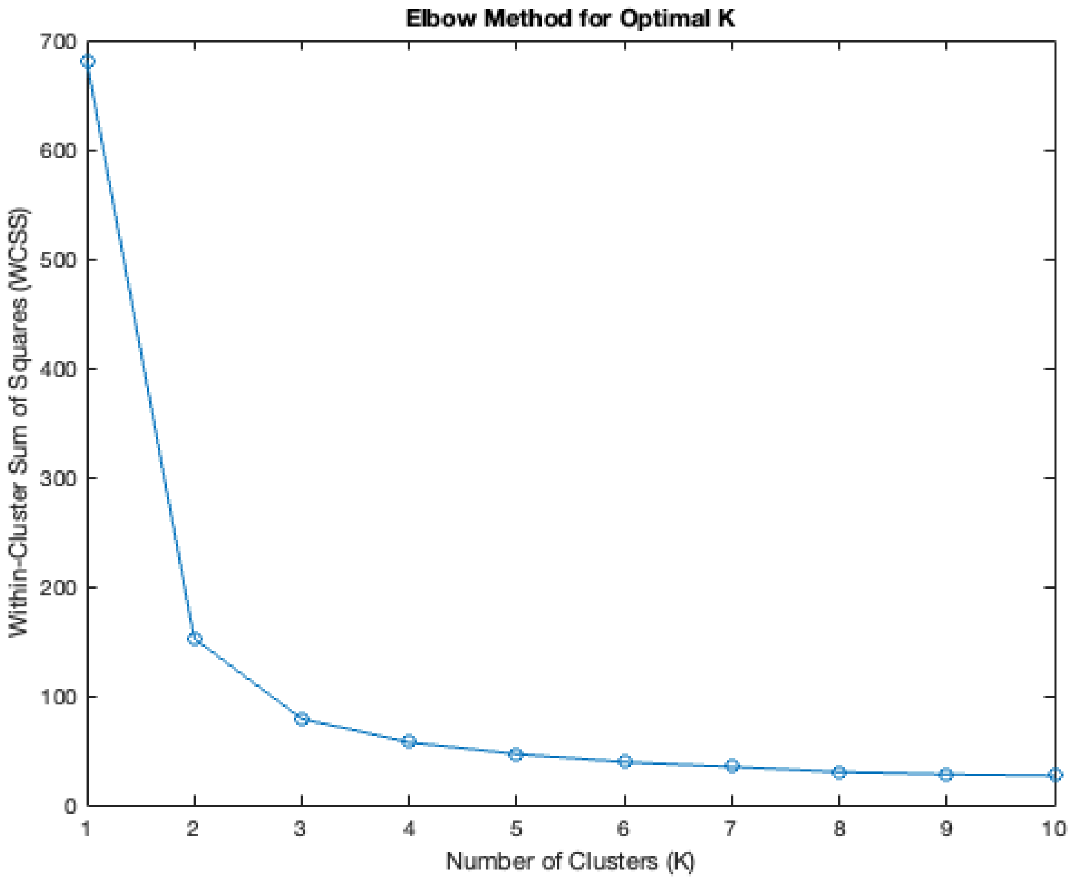

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem Set 5 - Advanced Econometrics
%
% Hude Hude
% hh3024@columbia.edu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; clc
```

Part 1a: K-means Clustering on the Iris Dataset

```
close all;
clear all;
clc;
rng(894);

% Load the Iris dataset
load fisheriris;

% Extract the features (sepal length, sepal width, petal length, petal width)
X = meas;

% True labels (Iris setosa = 1, Iris versicolor = 2, Iris virginica = 3)
true_labels = [1*ones(50,1); 2*ones(50,1); 3*ones(50,1)];

% Elbow Method to Determine Optimal K
wcss = zeros(1,10);
for k = 1:10
    [~, ~, sumd] = kmeans(X, k, 'Replicates', 5);
    wcss(k) = sum(sumd);  % Sum of distances to centroids (WCSS)
end

% Plot WCSS vs. number of clusters
figure;
plot(1:10, wcss, '-o');
xlabel('Number of Clusters (K)');
ylabel('Within-Cluster Sum of Squares (WCSS)');
title('Elbow Method for Optimal K');
```

```
% From the plot, determine the elbow (let's assume it's at K = 3 for now)
optimal_k = 3;

% K-means Clustering with Optimal K
[idx, C] = kmeans(X, optimal_k, 'Replicates', 5);

% Confusion Matrix
conf_matrix = confusionmat(true_labels, idx);

% Display confusion matrix
disp('Without Normalized');
disp('Confusion Matrix:');
disp(conf_matrix);

% Performance Metrics: Precision, Recall, F1-Score, and Accuracy
% Precision, Recall, F1, and Accuracy can be calculated based on the confusion
 matrix.
precision = diag(conf_matrix) ./ sum(conf_matrix, 2);
recall = diag(conf_matrix) ./ sum(conf_matrix, 1)';
f1_score = 2 * (precision .* recall) ./ (precision + recall);

% Accuracy
accuracy = sum(diag(conf_matrix)) / sum(conf_matrix(:));

% Display results
fprintf('Precision: %.2f\n', mean(precision));
fprintf('Recall: %.2f\n', mean(recall));
fprintf('F1 Score: %.2f\n', mean(f1_score));
fprintf('Accuracy: %.2f\n', accuracy);
```

```
Without Normalized
Confusion Matrix:
    50     0     0
     0    48     2
     0    14    36

Precision: 0.89
Recall: 0.91
F1 Score: 0.89
Accuracy: 0.89
```



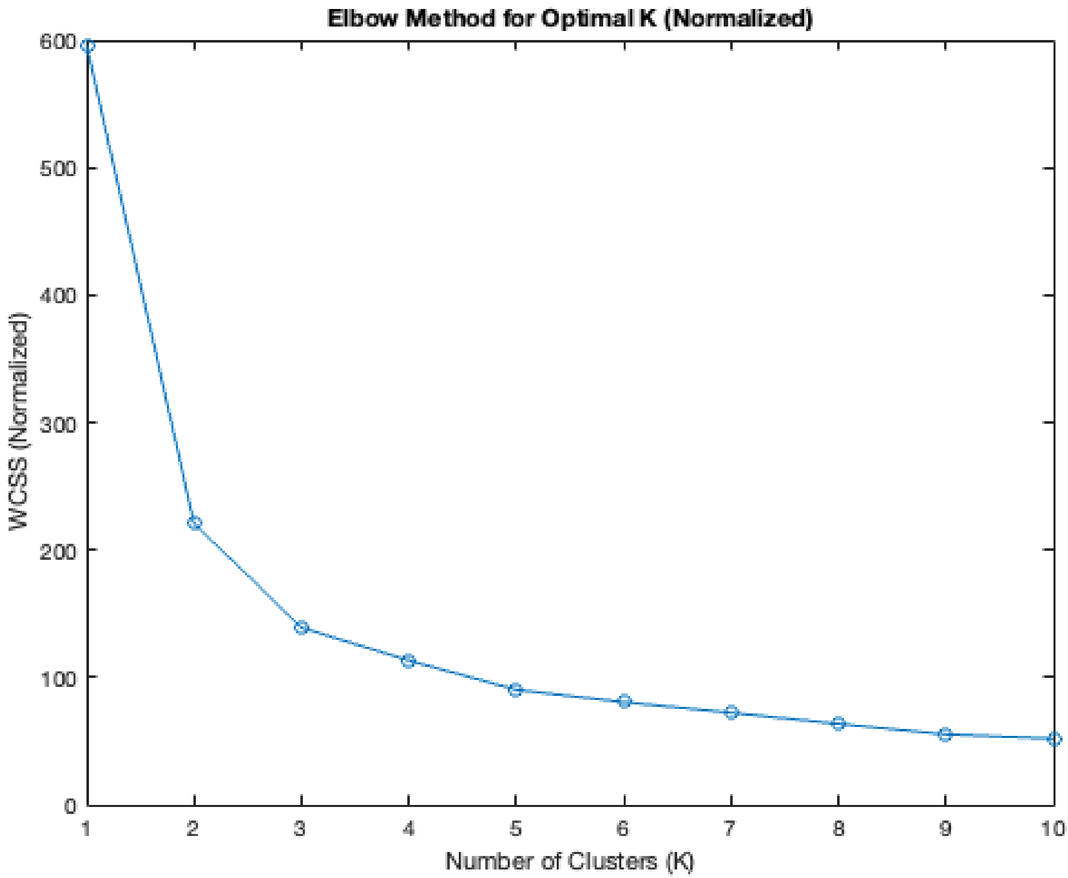
Part 1b: K-means Clustering with Normalized Data

Normalize the data (zero mean, unit standard deviation)

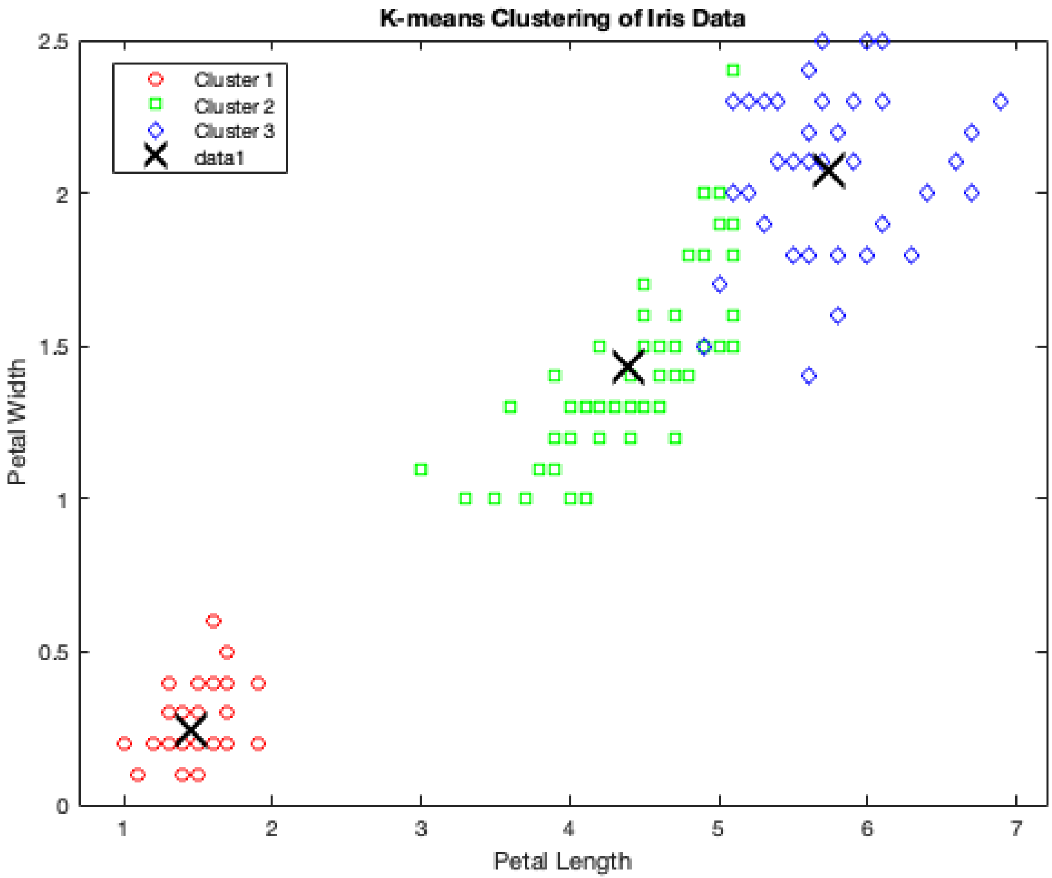
```
X_norm = (X - mean(X)) ./ std(X);

% Elbow Method for normalized data
wcss_norm = zeros(1,10);
for k = 1:10
    [~, ~, sumd] = kmeans(X_norm, k, 'Replicates', 5);
    wcss_norm(k) = sum(sumd);
end

% Plot WCSS vs. number of clusters for normalized data
figure;
plot(1:10, wcss_norm, '-o');
xlabel('Number of Clusters (K)');
ylabel('WCSS (Normalized)');
title('Elbow Method for Optimal K (Normalized)');

% K-means Clustering with optimal K (assuming K = 3)
```

```
[idx_norm, C_norm] = kmeans(X_norm, optimal_k, 'Replicates', 5);

% True labels (normalized)
true_labels_2 = [2*ones(50,1); 1*ones(50,1); 3*ones(50,1)];

% Confusion Matrix for normalized data
conf_matrix_norm = confusionmat(true_labels_2, idx_norm);

% Display confusion matrix
disp('Confusion Matrix (Normalized):');
disp(conf_matrix_norm);

% Performance Metrics for Normalized Data
precision_norm = diag(conf_matrix_norm) ./ sum(conf_matrix_norm, 2);
recall_norm = diag(conf_matrix_norm) ./ sum(conf_matrix_norm, 1)';
f1_score_norm = 2 * (precision_norm .* recall_norm) ./ (precision_norm +
 recall_norm);
accuracy_norm = sum(diag(conf_matrix_norm)) / sum(conf_matrix_norm(:));

% Display results for normalized data
fprintf('Precision (Normalized): %.2f\n', mean(precision_norm));
fprintf('Recall (Normalized): %.2f\n', mean(recall_norm));
fprintf('F1 Score (Normalized): %.2f\n', mean(f1_score_norm));
fprintf('Accuracy (Normalized): %.2f\n', accuracy_norm);
disp("So, the precision are similar under two cases");

% Choose two features to plot (e.g., petal length = column 3, petal width =
 column 4)
figure;
gscatter(meas(:,3), meas(:,4), idx, 'rgb', 'osd');

% Add labels and title
xlabel('Petal Length');
ylabel('Petal Width');
title('K-means Clustering of Iris Data');
legend('Cluster 1', 'Cluster 2', 'Cluster 3');
hold on;

% Plot the cluster centroids
plot(C(:,3), C(:,4), 'kx', 'MarkerSize', 15, 'LineWidth', 3);
hold off;
```

```
Confusion Matrix (Normalized):
    39     0    11
     0    50     0
    17     0    33

Precision (Normalized): 0.81
Recall (Normalized): 0.82
F1 Score (Normalized): 0.81
Accuracy (Normalized): 0.81
So, the precision are similar under two cases
```



Elbow Method for Optimal K (Normalized)
WCSS (Normalized)
Number of Clusters (K)
0
100
200
300
400
500
600
1
2
3
4
5
6
7
8
9
10

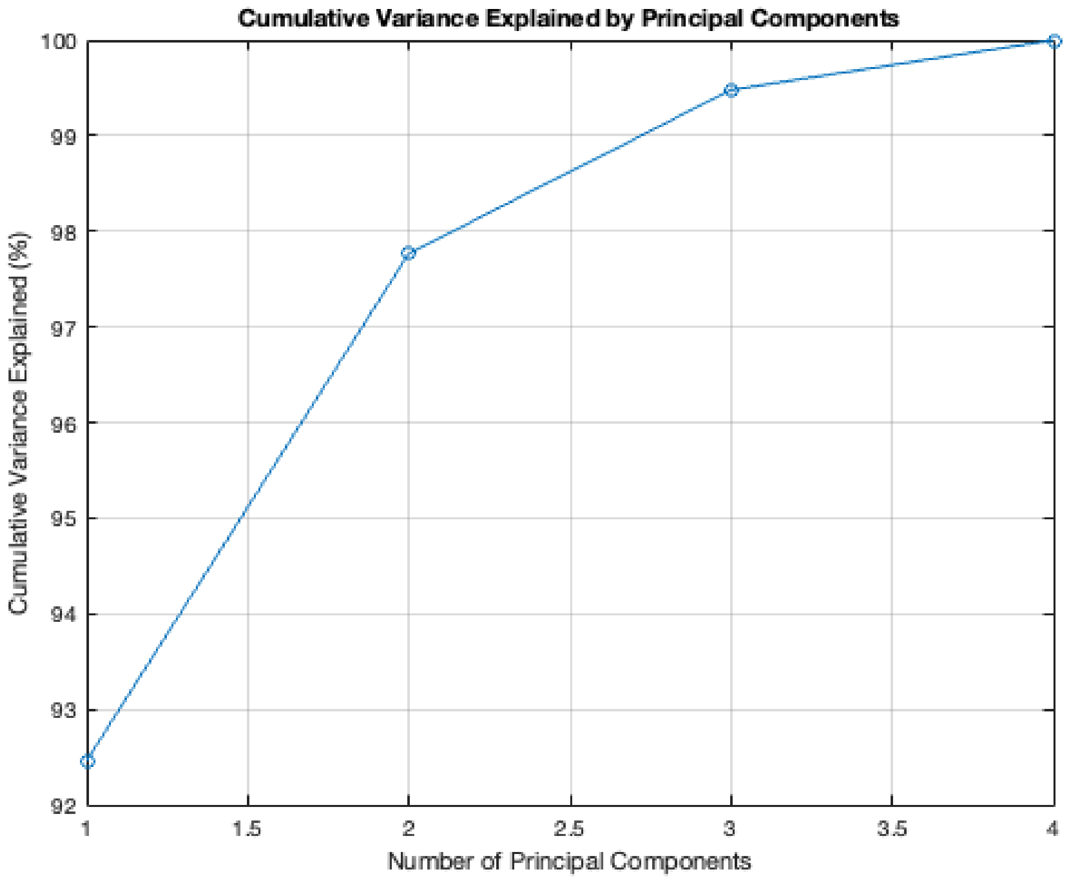


Part 2a: Principal Component Analysis (PCA) on the Iris Dataset

```
% Perform PCA
[coeff, score, latent, ~, explained] = pca(X);

% latent contains the eigenvalues (variance explained by each principal
 component)
% explained contains the percentage of variance explained by each principal
 component

% Display the explained variance for each principal component
disp('Variance explained by each principal component (%):');
disp(explained);

% Calculate the cumulative variance explained
cumulative_variance = cumsum(explained);

% Display how much variance is retained by the first 1, 2, and 3 principal
 components
```

```
disp('Cumulative variance explained by the first 1, 2, and 3 principal
 components (%):');
disp(cumulative_variance(1:3));

% Plot cumulative variance explained by the principal components
figure;
plot(1:length(explained), cumulative_variance, '-o');
xlabel('Number of Principal Components');
ylabel('Cumulative Variance Explained (%)');
title('Cumulative Variance Explained by Principal Components');
grid on;
```

```
Variance explained by each principal component (%):
   92.4619
    5.3066
    1.7103
    0.5212

Cumulative variance explained by the first 1, 2, and 3 principal components
 (%):
   92.4619
   97.7685
   99.4788
```



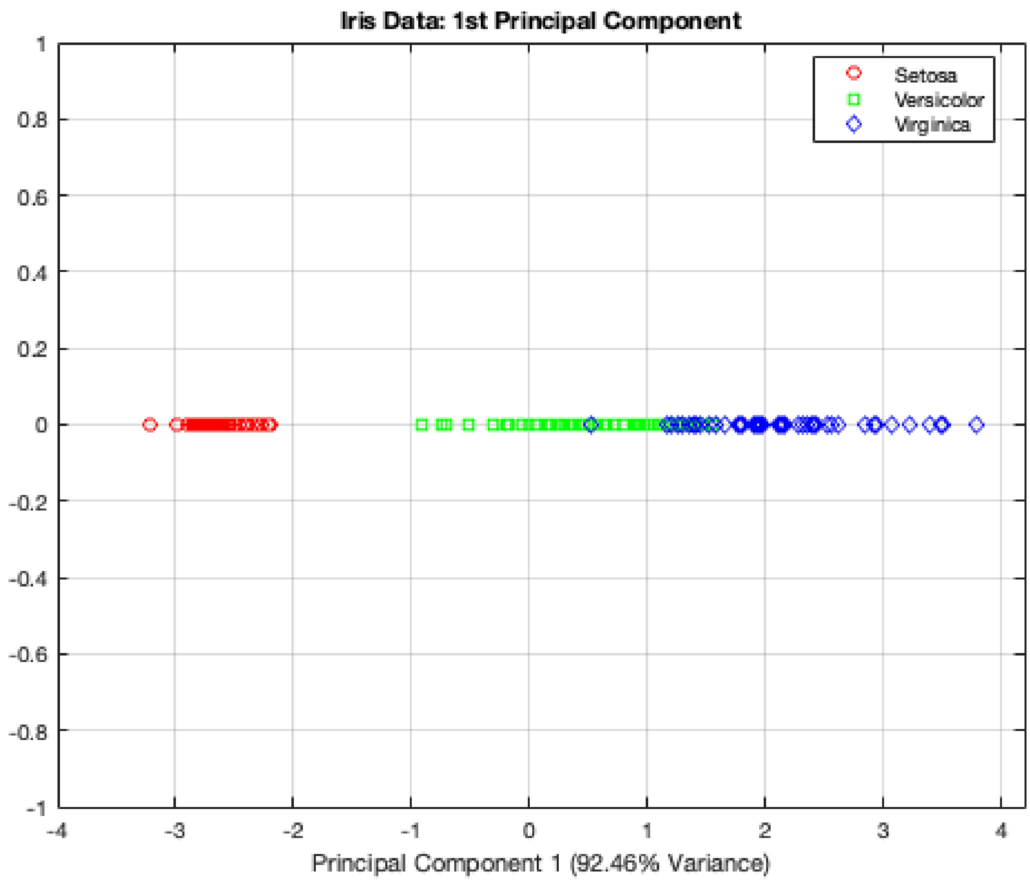
Part 2b: Plot Iris Data in the Space Spanned by the First Two Principal Components

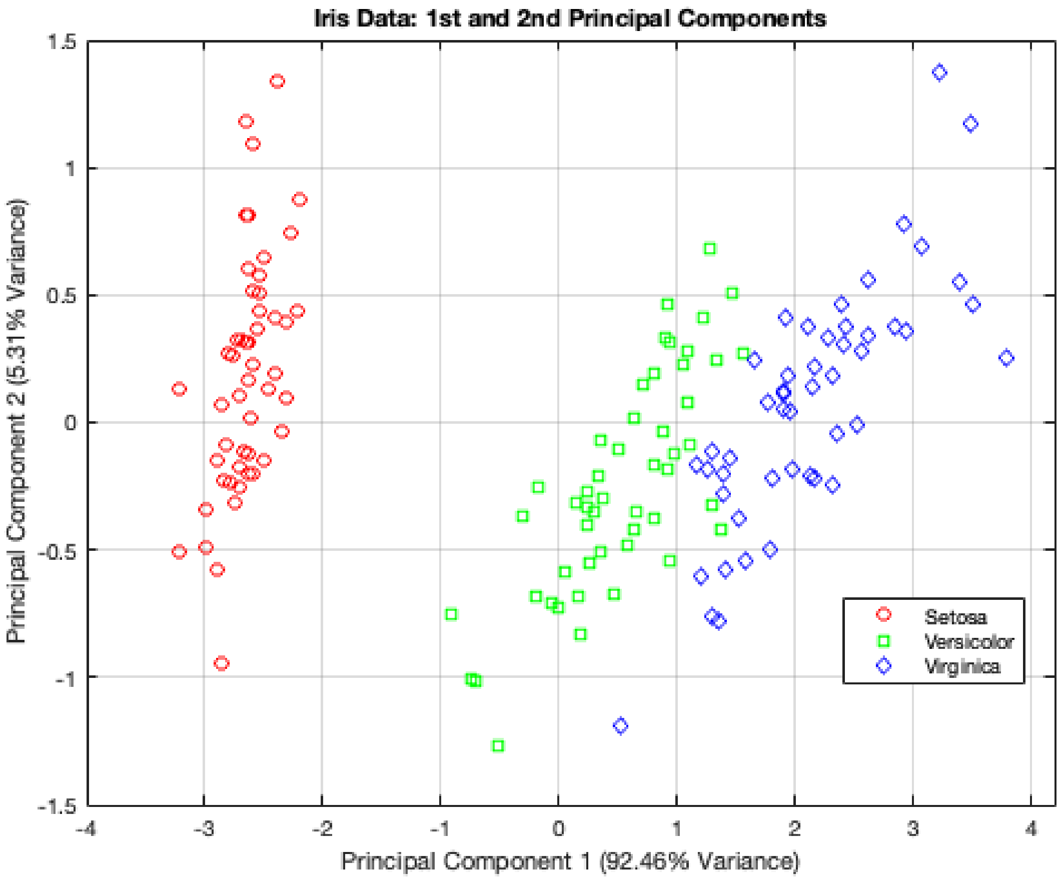
Visualization of Principal Components

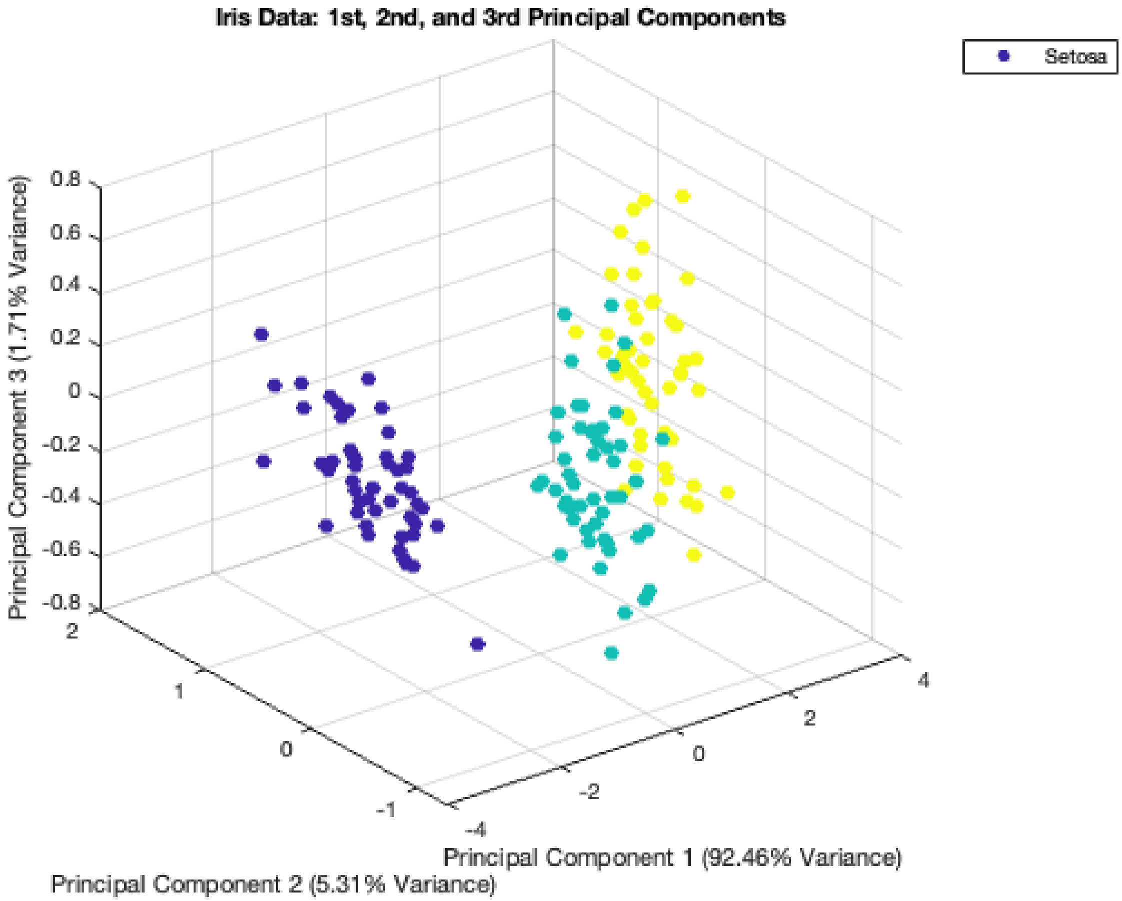
```
% Perform PCA
[coeff, score, ~, ~, explained] = pca(meas);

% True labels (1 = Setosa, 2 = Versicolor, 3 = Virginica)
true_labels_plot = [ones(50,1); 2*ones(50,1); 3*ones(50,1)];

% i) Plot 1st Principal Component on a real line
figure;
gscatter(score(:,1), zeros(size(score,1), 1), true_labels_plot, 'rgb', 'osd');
xlabel(sprintf('Principal Component 1 (%.2f%% Variance)', explained(1)));
title('Iris Data: 1st Principal Component');
legend('Setosa', 'Versicolor', 'Virginica');
grid on;

% ii) Plot 1st and 2nd Principal Components on a 2D plane
figure;
gscatter(score(:,1), score(:,2), true_labels, 'rgb', 'osd');
```

```
xlabel(sprintf('Principal Component 1 (%.2f%% Variance)', explained(1)));
ylabel(sprintf('Principal Component 2 (%.2f%% Variance)', explained(2)));
title('Iris Data: 1st and 2nd Principal Components');
legend('Setosa', 'Versicolor', 'Virginica');
grid on;

% iii) Plot 1st, 2nd, and 3rd Principal Components in 3D space
figure;
scatter3(score(:,1), score(:,2), score(:,3), 50, true_labels, 'filled');
xlabel(sprintf('Principal Component 1 (%.2f%% Variance)', explained(1)));
ylabel(sprintf('Principal Component 2 (%.2f%% Variance)', explained(2)));
zlabel(sprintf('Principal Component 3 (%.2f%% Variance)', explained(3)));
title('Iris Data: 1st, 2nd, and 3rd Principal Components');
legend('Setosa', 'Versicolor', 'Virginica');
grid on;

disp("So, the first component mainly explains the variance and determines the
 classification");
```

Warning: Ignoring extra legend entries.
So, the first component mainly explains the variance and determines the classification




Iris Data: 1st and 2nd Principal Components
Principal Component 2 (5.31% Variance)
Principal Component 1 (92.46% Variance)
1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4
Setosa
Versicolor
Virginica



Part 3a: K-means Clustering on Principal Components

```
% Extract the first two principal components
pc1 = score(:,1);  % First principal component
pc2 = score(:,2);  % Second principal component
PC = [pc1, pc2];   % Combine the first two principal components

% True labels
true_labels_3 = [3*ones(50,1); 1*ones(50,1); 2*ones(50,1)];

% K-means Clustering on Principal Components
[idx_p, C] = kmeans(PC, 3, 'Replicates', 5);

% Compute the confusion matrix
conf_matrix = confusionmat(true_labels_3, idx_p);

% Calculate precision, recall, F1-score, and accuracy
precision = diag(conf_matrix) ./ sum(conf_matrix, 2);
recall = diag(conf_matrix) ./ sum(conf_matrix, 1)';
f1_score = 2 * (precision .* recall) ./ (precision + recall);
```

```
accuracy = sum(diag(conf_matrix)) / sum(conf_matrix(:));

% Display results
disp('Confusion Matrix (Principal Components):');
disp(conf_matrix);
fprintf('Precision: %.2f\n', mean(precision));
fprintf('Recall: %.2f\n', mean(recall));
fprintf('F1 Score: %.2f\n', mean(f1_score));
fprintf('Accuracy: %.2f\n', accuracy);

% Normalizing the Principal Components
PC_norm = (PC - mean(PC)) ./ std(PC);  % Zero mean, unit standard deviation

% Perform K-means clustering on normalized principal components
[idx_p_norm, C_norm] = kmeans(PC_norm, 3, 'Replicates', 5);

% Table
true_labels_4 = [1*ones(50,1); 2*ones(50,1); 3*ones(50,1)];
% Compute the confusion matrix for normalized components
conf_matrix_norm = confusionmat(true_labels_4, idx_p_norm);

% Calculate precision, recall, F1-score, and accuracy for normalized
 components
precision_norm = diag(conf_matrix_norm) ./ sum(conf_matrix_norm, 2);
recall_norm = diag(conf_matrix_norm) ./ sum(conf_matrix_norm, 1)';
f1_score_norm = 2 * (precision_norm .* recall_norm) ./ (precision_norm +
 recall_norm);
accuracy_norm = sum(diag(conf_matrix_norm)) / sum(conf_matrix_norm(:));

% Display results for normalized components
disp('Confusion Matrix (Normalized Principal Components):');
disp(conf_matrix_norm);
fprintf('Precision (Normalized): %.2f\n', mean(precision_norm));
fprintf('Recall (Normalized): %.2f\n', mean(recall_norm));
fprintf('F1 Score (Normalized): %.2f\n', mean(f1_score_norm));
fprintf('Accuracy (Normalized): %.2f\n', accuracy_norm);

disp("So, the precision is actually better without normalization. This is
 because the first component mainly determines the type. Normalization makes
 this contribute less for classification prediction.");
```

```
Confusion Matrix (Principal Components):
    47     3     0
    14    36     0
     0     0    50

Precision: 0.89
Recall: 0.90
F1 Score: 0.89
Accuracy: 0.89
Confusion Matrix (Normalized Principal Components):
    49     1     0
     0    37    13
     0    20    30
```

```
Precision (Normalized): 0.77
Recall (Normalized): 0.78
F1 Score (Normalized): 0.77
Accuracy (Normalized): 0.77
So, the precision is actually better without normalization. This is because
 the first component mainly determines the type. Normalization makes this
 contribute less for classification prediction.
```

Part 3b. K-means Clustering on 3 Principal Components

```
PC3 = score(:,1:3);  % First three principal components

% Perform K-means clustering on 3 principal components
[idx3, C3] = kmeans(PC3, 3, 'Replicates', 5);

% Table
true_labels_3pc = [1*ones(50,1); 3*ones(50,1); 2*ones(50,1)];

% Confusion Matrix for 3 PCs
conf_matrix_3PC = confusionmat(true_labels_3pc, idx3);

% Calculate precision, recall, F1-score, and accuracy for 3 PCs
precision_3PC = diag(conf_matrix_3PC) ./ sum(conf_matrix_3PC, 2);
recall_3PC = diag(conf_matrix_3PC) ./ sum(conf_matrix_3PC, 1)';
f1_score_3PC = 2 * (precision_3PC .* recall_3PC) ./ (precision_3PC +
 recall_3PC);
accuracy_3PC = sum(diag(conf_matrix_3PC)) / sum(conf_matrix_3PC(:));

% Display results for 3 PCs
disp('Confusion Matrix (3 Principal Components):');
disp(conf_matrix_3PC);
fprintf('Precision (3 PCs): %.2f\n', mean(precision_3PC));
fprintf('Recall (3 PCs): %.2f\n', mean(recall_3PC));
fprintf('F1 Score (3 PCs): %.2f\n', mean(f1_score_3PC));
fprintf('Accuracy (3 PCs): %.2f\n', accuracy_3PC);

disp("Now, the precision increases when normalization applies.");
```

```
Confusion Matrix (3 Principal Components):
    50     0     0
     0    36    14
     0     2    48

Precision (3 PCs): 0.89
Recall (3 PCs): 0.91
F1 Score (3 PCs): 0.89
Accuracy (3 PCs): 0.89
Now, the precision increases when normalization applies.
```

Published with MATLAB® R2022a