Manvir Kaur 253852
Carla Gironella 254136

LAB 06 STATISTICAL MODELS

Problem 1

Are the faces of babies informative about their sex? There is no easy answer to this question, since identifying if a baby is a boy or a girl only by looking at his/her face is not trivial. To shed light on this matter, in this problem we will analyze a dataset consisting of 137 shape features extracted from 3D facial scans of 285 babies (174 boys and 111 girls). Specifically, file SM23_BabyFeatures.xlsx contains the above information organized as follows:
• Each row of the file contains the information corresponding to one baby.
• The first column indicates the sex (0 = girls; 1 = boys).
• There are no column or row headers, so that you can easily read the data with xlsread().
We immediately notice that this dataset has a large number of dimensions p = 137, and we wish to see if we can analyze it using $p' < p$ dimensions. For this problem we will use PCA to obtain a reduced-dimensionality representation of our data. We also wish that the dimensions on this reduced representation are significant, in the sense that we could reject such dimensions to have originated due to chance, at significance level $\alpha = 0.05$:

a) Use permutation tests to determine how many Principal Components are significant (in the sense indicated above). To this end you should perform at least R = 1000 rounds of permutations of the data matrix. For each of these rounds, you must permute the entries independently for each dimension, decompose the resulting matrix into eigenvalues and eigenvectors, and keep track of the eigenvalues (adequately sorted). After this, you will have R replicates $\hat{\lambda}^*$ for the magnitude of each of your eigenvalues under the null hypothesis of randomness in the data. Use these replicates to determine a p-value for each of the eigenvalues $\hat{\lambda}$ obtained by applying PCA to your input data. Clearly indicate the number of components p' that can be considered significant at $\alpha = 0.05$ according to your analysis.

First we have loaded the data and gotten the features separated from the labels to perform the PCA to both groups: boys and girls.

```
data = xlsread('SM23_BabyFeatures.xlsx');
features = data(:, 2:end);
```

Computing first the Principal Components from the original data (without doing a permutation test) we go with the following lines of code:

```
%A)
mean_features = mean(features, 1);
%centered_features = features - mean_features;
cov_matrix = cov(features);

[eigen_vec, eigen_val] = eig(cov_matrix);

[eigen_values, indices] = sort(diag(eigen_val), 'descend');
eigen_vectors = eigen_vec(:, indices);
```

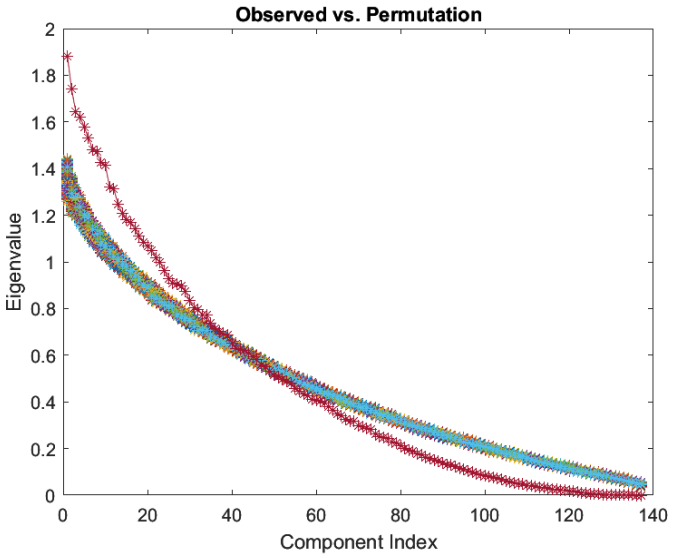
Getting the eigenvalues of the original data kept in a variable for later use when computing the p values (adequately sorted).
For the second step we perform the replicates to determine the p value for each of the eigenvalues obtained by PCA to our data (the ones we have kept in the previous screenshot)

Manvir Kaur 253852
Carla Gironella 254136

```
R = 1000;
replicates = zeros(R, size(eigen_values, 1));
permfeatures = zeros(285, 137);
for i = 1:R
    for j=1:137
        permuted_indices = randperm(size(features, 1))';
        permuted_features = features(:, j);
        permuted = permuted_features(permuted_indices)';
        permfeatures(:,j) = permuted;
    end

    permuted_cov_matrix = cov(permfeatures);

    [permuted_eigen_vectors, permuted_eigen_values] = eig(permuted_cov_matrix);
    permuted_eigen_values = sort(diag(permuted_eigen_values), 'descend');

    replicates(i, :) = permuted_eigen_values';
end
```

Here is the code in how we obtain the eigenvalues for each premuted feature. To try to make a prediction on how many dimensions we would get and a general view of the replicates matrix together with the original eigenvalues we made the following plot:

```
plot(replicates', '*')
hold on;
plot(eigen_values, '-*' )
title('Observed vs. Permutation');
xlabel('Component Index');
ylabel('Eigenvalue');
legend('Observed ', 'Permutation');
hold off
```



Here in this plot we can observe from each component (dimension) the eigenvalues of the permuted samples and the eigenvalue of the original data (the red line). In the following steps we will try to find those dimensions that are over the eigenvalues of the permutation eigenvalues because they indicate statistically significant differences compared to the permuted samples. Identifying dimensions with eigenvalues surpassing the corresponding values from permuted data suggests that these dimensions contribute significantly to the original dataset's characteristics or patterns.

Now, after looking and getting some conclusions on the graph, we can proceed to compute which of the components are seen as significant with the following code:

```
alpha = 0.05;
p_values = zeros(1, 137);
%compute the p-value for each column separately
for i=1:137
    n_as_high_as_observed = sum(replicates(:,i) > eigen_values(i)');
    p_values(1,i) = (1+n_as_high_as_observed)/(1+R);
end
p_values;
significant_components = find(p_values < alpha)
```

Where we compute the p value for each of the columns by seeing the total number of permuted eigenvalues (of a certain dimension) are larger than the eigenvalue of that dimension of the original data. We get all these sums for all the dimensions of the original data and then (with this information computed) perform the computation of the p_value for each dimension. To do the PCA, we will only get the dimensions in which p_value is lower than the chosen significance level (alpha = 0.05). We get the following resulting dimensions:

significant_components = 1×41

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 44

Everytime we run we get different results (since it consists of a random process), but they are always very similar. There could be some inconsistencies in the following steps to executing again but the results would be very similar than if we used these significant components.

b) Estimate the percentage of variance retained by the p ′ significant components that you have obtained in (a). Use bootstrap to estimate a 95% confidence interval of the percentage of variance retained by those p ′ components1 .

To perform the previous statement we have performed the following steps. First of all, we have computed the retained variance of the eigenvalues of the principal components extracted from 1.a. we got the following results:

```
eigVals=eigen_val(indices,indices);
original_eigen_vectors=eigen_vec(:,indices);

my_size = size(significant_components,2);
sum_var = trace(eigVals);
principal_component = sum(eigen_values(1:my_size));

percentage = principal_component*100/sum_var
```

percentage = 69.7749

The result consists on the sum of the eigenvalues of the principal components only (that is, 41 eigenvalues) that, in theory, retain a big part of the variance of the dataset divided by the total sum of eigenvalues (that is, the whole variance of the dataset itself).

Once this is computed, we proceed on the second part of the assignment in this part. To perform a 95% confidence interval on the percentage of variance retained by p' components with bootstrap we have sued the following code:

1. We store only the eigenvalues and eigenvectors of the principal components chosen in 1.a

```
selected_eigenvalues = eigen_values(1:my_size);
selected_eigenvectors = eigen_vectors(:, indices(1:my_size));
```

2. And then we perform the loop to create the bootstrap and perform the PCA for each sample and the retained variance

```
% Bootstrap resampling
B = 1000; % Number of bootstrap samples
bootstrap_percentages = zeros(B, 1);
samplesize = length(features)
for b = 1:B
    bootstrap_indices =  ceil (samplesize*rand(1,samplesize));
    bootstrap_sample = features(bootstrap_indices,:);

    bootstrap_cov_matrix = cov(bootstrap_sample);
    [bootstrap_eigen_vec, bootstrap_eigen_val] = eig(bootstrap_cov_matrix);

    [bootstrap_eigen_values, bootstrap_indices] = sort(diag(bootstrap_eigen_val), 'descend');
    bootstrap_eigen_vectors = bootstrap_eigen_vec(:, bootstrap_indices);

    bootstrap_selected_eigenvalues = bootstrap_eigen_values(1:my_size);
    bootstrap_percentage = sum(bootstrap_selected_eigenvalues) * 100 / sum_var;

    bootstrap_percentages(b) = bootstrap_percentage;
end
bootstrap_percentages
```

Getting as a result 1000 bootstrap retained variance (just like the one computed in 1.b)

```
bootstrap_percentages = 1000×1
    77.1446
    79.0336
    80.2079
    75.8685
    79.3655
    80.1271
    80.9398
    78.3546
    78.1345
    73.9119
       ⋮
```

3. Perform the computation of the confidence interval with alpha = 0.05

```
alpha=0.05;
sort_boot = sort(bootstrap_percentages)

t_low = sort_boot(B * alpha/2)
t_high= sort_boot(B * (1- (alpha/2)))

CI_percentiles = [2*percentage - t_high, 2*percentage - t_low]
```

Getting the following result:

```
t_low = 74.1789
t_high = 81.9058
CI_percentiles = 1×2
    57.6440   65.3709
```

We have that the retained variance computed in 1.a is outside the 95% confidence interval. If the computed retained variance from the sampling is not within the 95% confidence interval obtained through bootstrapping, it

suggests a discrepancy between the observed variance and the variability estimated through bootstrapping. Possible reasons include a small sample size, non-normality or skewness in the data, model assumptions not being met, the influence of outliers, or potential computational issues.

c) Now we can address our research question in PCA space. State the null and alternative hypotheses required for the following research question: "On average, are there shape differences between the faces of male and female babies?"

We have the following null and alternative hypothesis required for the question:

H_0: $\mu_1 = \mu_2$ (The mean shape in the PCA space for male baby faces is equal to the mean shape for female baby faces.)

H_1: $\mu_1 \neq \mu_2$ (The mean shape in the PCA space for male baby faces is not equal to the mean shape for female baby faces.)

d) Select an appropriate method to test the hypotheses in (c) (an appropriate method is one for which all necessary assumptions are fulfilled by the data). Because you may end up with p ′ relatively large, testing for multivariate normality is intensive and thus we will skip it for now; in other words, let us assume that the input data comes from a multivariate normal distribution. Additionally, be aware that a large p ′ will challenge any attempt to assume large n for these data.

Since we are comparing two means of two populations on many features we have many options of test to choose. To proceed, we have chosen to do a Hotelling's T2. This approach has some assumptions to be fulfilled to ensure that the results are reliable.

The assumptions are the following:

1. The sample $\mathbf{X}_{11}, \mathbf{X}_{12}, \ldots, \mathbf{X}_{1n_1}$, is a random sample of size n_1 from a p-variate population with mean vector $\boldsymbol{\mu}_1$ and covariance matrix $\boldsymbol{\Sigma}_1$.
2. The sample $\mathbf{X}_{21}, \mathbf{X}_{22}, \ldots, \mathbf{X}_{2n_2}$, is a random sample of size n_2 from a p-variate population with mean vector $\boldsymbol{\mu}_2$ and covariance matrix $\boldsymbol{\Sigma}_2$.
3. Also, $\mathbf{X}_{11}, \mathbf{X}_{12}, \ldots, \mathbf{X}_{1n_1}$, are independent of $\mathbf{X}_{21}, \mathbf{X}_{22}, \ldots, \mathbf{X}_{2n_2}$. (6-19)

However, since we have big dimensionality on features, we can't say that n can be considered large in any of the populations. Therefore, further assumptions need to be assessed:

1. Both populations are multivariate normal.
2. Also, $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$ (same covariance matrix).

Since we are skipping the multivariate normality testing for now, we just need to check the second assumption where we need to check the following inequality → Computing the division between the numbers on the diagonals of the two covariance matrices we check if the results are compressed between ¼ and 4.

```
%D)
alpha = 0.05
female = data(data(:, 1) == 0, :);
female_features = female(:, 2:end);
female_features = female_features(:, significant_components);

male =  data(data(:, 1) == 1, :);
male_features = male(:, 2:end);
male_features = male_features(:, significant_components);

[n1, p1] = size(female_features);
[n2, p2] = size(male_features);

female_mean = mean(female_features,1);
male_mean = mean(male_features, 1);

female_s = cov(female_features)
male_s = cov(male_features)

diag_female_s = diag(female_s);
diag_male_s = diag(male_s);

ratio_check = all(diag_female_s ./ diag_male_s >= 1/4 & diag_female_s ./ diag_male_s <= 4);

if ratio_check
    disp('The diagonals are approximately equal (the ratio is between 1/4 and 4).');
else
    disp('The diagonals are not approximately equal.');
end
```

We first separated the data into females and males to perform the test. The previous computations needed to be performed with the two populations together to get the same new spaces to make a reliable analysis. From the code above, we got:

```
The diagonals are approximately equal (the ratio is between 1/4 and 4).
```

And therefore, from the data until now, we can assume that they have the same covariance matrix.

e) Test the hypotheses in (c) with the method selected in (d) under the assumption of multivariate normality. Interpret the obtained result in terms of the research question.

Using the following formulas:

We set

$$\mathbf{S}_{\text{pooled}} = \frac{\sum_{j=1}^{n_1}(\mathbf{x}_{1j}-\bar{\mathbf{x}}_1)(\mathbf{x}_{1j}-\bar{\mathbf{x}}_1)' + \sum_{j=1}^{n_2}(\mathbf{x}_{2j}-\bar{\mathbf{x}}_2)(\mathbf{x}_{2j}-\bar{\mathbf{x}}_2)'}{n_1+n_2-2}$$
$$= \frac{n_1-1}{n_1+n_2-2}\mathbf{S}_1 + \frac{n_2-1}{n_1+n_2-2}\mathbf{S}_2 \qquad (6\text{-}21)$$

$$T^2 = [\overline{\mathbf{X}}_1 - \overline{\mathbf{X}}_2 - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)]' \left[\left(\frac{1}{n_1}+\frac{1}{n_2}\right)\mathbf{S}_{\text{pooled}}\right]^{-1} [\overline{\mathbf{X}}_1 - \overline{\mathbf{X}}_2 - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)]$$

$$c^2 = \frac{(n_1+n_2-2)p}{(n_1+n_2-p-1)} F_{p,n_1+n_2-p-1}(\alpha)$$

We provided the following code:

```
spooled = ((n1-1)/(n1+n2-2))*female_s + ((n2-1)/(n1+n2-2))*male_s;
my_inverse = inv(((1/n1)+(1/n2))*spooled);
t_obs = (female_mean - male_mean)*(my_inverse)*(female_mean-male_mean).
t_crit = (((n1+n2-2)*p1)/(n1+n2-p1-1))*finv(1-alpha, p1,n1+n2-p1-1)

if (t_obs>crit_t)
    display('Reject')
else
    display('Fail to reject')
end
```

And obtained the following results:

```
t_obs = 60.1539
t_crit = 68.7370
```

In which we can say that we don't have enough evidence to reject the null hypothesis.

f) Now we revisit the assumption of multivariate normality. Because assessing this for a large p ′ is very time-consuming, we will instead check what would happen if we cannot assume normality. In such case, we could still use the same statistic selected in (d) if we use bootstrap to assess its distribution under our null hypothesis. Proceed in this way to repeat (e) but without the assumption of multivariate normality. Compare the results obtained in (e) and (f).

We have performed the same steps as before computing for all the bootstrap samples (of both different males and females) the new observed T2 onto an array called tBOOT.

```
R = 1000;

size_male = size(male_features,1);
size_female = size(female_features,1);
tBOOT = zeros(1,R);

for b=1: R
    random_indices_male = ceil(size_male*rand(1, size_male));
    random_indices_female = ceil(size_female*rand(1, size_female));

    male_boot =male_features(random_indices_male, :);
    female_boot =female_features(random_indices_female, :);

    female_mean = mean(female_boot,1);
    male_mean = mean(male_boot, 1);
    female_cov = cov(female_boot);
    male_cov = cov(male_boot);

    spooled_boot = ((n1-1)/(n1+n2-2))*female_cov + ((n2-1)/(n1+n2-2))*male_cov;
    my_inverse_boot = inv(((1/n1)+(1/n2))*spooled_boot);

    tBOOT(b) = (female_mean - male_mean)*my_inverse_boot*(female_mean - male_mean).';

end
p_value = sum(abs(tBOOT(i))>crit_t)+1/(R+1)

if p_value < alpha
    display('Reject');
else
    display('Fail to reject')
end
```

To compute the p_value we just have applied the formula you can see above (which takes into account how many observed Ts are higher than the critical value) and applying the formula for the p_value we get the following results:

```
p_value = 0.0020


Reject
```

In which we can say, that we have enough evidence to be able to reject the null hypothesis.

PROBLEM 2

Manvir Kaur 253852
Carla Gironella 254136

We continue the analysis of Problem 1, but in this case we will apply ISOMAP to reduce the dimensionality.

a) Compute a graph to approximate the geodesic distances of your data manifold for all pairs of samples. The number of neighbors that you must use to build the graph will depend on your NIA:

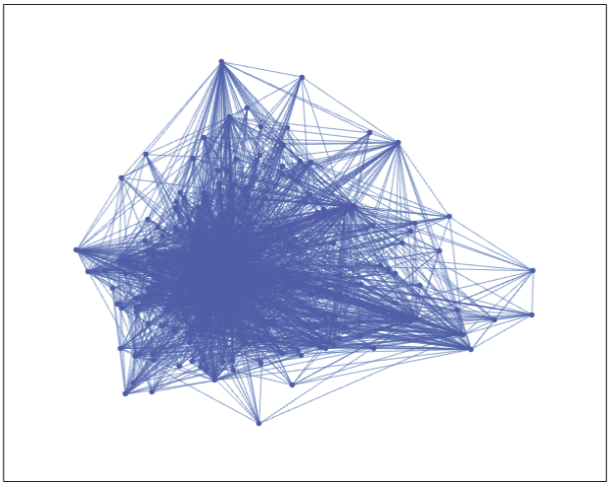
$$k = 5 + \sum_{\forall j \in Group} L_j$$

where k is the number of neighbors to use, Lj is the last digit of the NIA of student j, and j iterates over all students of your group (i.e. 1, 2 or 3 members). Display the resulting graph.2

```
N = size(features)

DM = zeros(N(1));

for i= 1 : N(1) -1
    for j =i +1 :N(1)
        DM(i,j) = sqrt(sum(power(features(i,:) - features(j,:),2)));
        DM(j,i) = DM(i,j);
    end
end
DM

manvir_nia = 2;
carla_nia = 6;

k = manvir_nia + carla_nia + 5;

A = zeros(N(1))
for i =1 : N(1)
    [s_dist, s_idx] = sort(DM(i,:), 'ascend');
    A(i, s_idx(2:k+1)) = 1;
end

% we make the neighbours matrix symetric
AS = (A+A')>0;

for i= 1 : N(1)
    for j =1 :N(1)
        GeoDist(i,j) = AS(i,j) * DM(i,j);
    end
end
```

We have computed the geodesic distances for each pair of points getting a symmetric matrix of distances. Then, we compute a matrix in which we will have the k=13 nearest neighbors of each sample (getting to the following plotted graph):

```
mygraph = graph(GeoDist);
plot(mygraph)
GD = distances(mygraph)
```



Manvir Kaur 253852
Carla Gironella 254136

b) Use the graph created in (a) to reduce the dimensionality of your data to p ′ = 3 dimension using ISOMAP. Compare the proportion of variance retained by these three ISOMAP dimensions with respect to the percentage of variance retained by the first three Principal Components that you computed in Problem 1.

```
H = eye(N(1)) - power(N(1),-1)* ones(N(1));

A_mds = (1/2)* GD^2;
B_mds = H * A_mds * H;

[eig_vec,eig_val] = eig(B_mds);
[sort_L, sort_idx] = sort(diag(eig_val),'descend');
U = eig_vec(:, sort_idx);
```

Here is the code we have used to compute which are the first 3 principal components when using ISOMAP. For the proportion of variance we used this code and got the resulting:

```
% Extract the first three ISOMAP dimensions
isomap_dimensions = U(:, 1:3);

variance_retained_isomap = sum(sort_L(1:3)) / sum(diag(eig_val))

my_size = 3;
sum_var = trace(eigVals);
principal_component = sum(eigen_values(1:my_size));

percentage = principal_component/sum_var
```

```
variance_retained_isomap = 0.5919
```

```
percentage = 0.0817
```

This big difference between the retained variance of ISOMAP principal components and PCA principal components suggests that the nonlinear structure captured by ISOMAP is important for explaining the variability in our data.

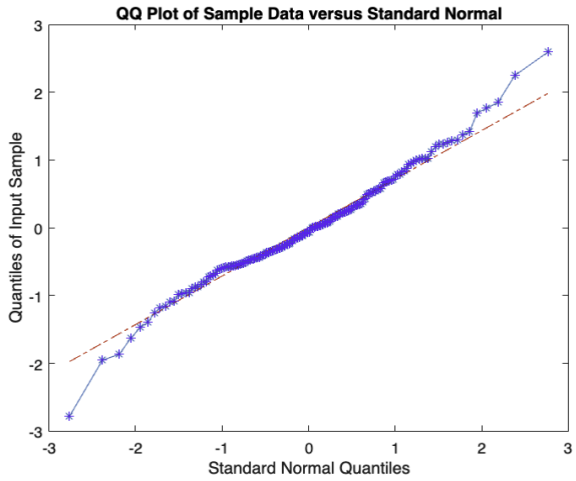
c) Select an appropriate method to test the hypotheses in (1.c) (an appropriate method is one for which all necessary assumptions are fulfilled by the data). Because now p ′ = 3, testing for multivariate normality would not be a problem, if needed. Justify your choice.

The assumptions to test in this case (because we are using again the Hotelling T^2) are the following:.

1. The sample $\mathbf{X}_{11}, \mathbf{X}_{12}, \ldots, \mathbf{X}_{1n_1}$, is a random sample of size n_1 from a p-variate population with mean vector $\boldsymbol{\mu}_1$ and covariance matrix $\boldsymbol{\Sigma}_1$.
2. The sample $\mathbf{X}_{21}, \mathbf{X}_{22}, \ldots, \mathbf{X}_{2n_2}$, is a random sample of size n_2 from a p-variate population with mean vector $\boldsymbol{\mu}_2$ and covariance matrix $\boldsymbol{\Sigma}_2$.
3. Also, $\mathbf{X}_{11}, \mathbf{X}_{12}, \ldots, \mathbf{X}_{1n_1}$, are independent of $\mathbf{X}_{21}, \mathbf{X}_{22}, \ldots, \mathbf{X}_{2n_2}$. (6-19)

We don’t need further assumptions because we assume n is large. However, we need to check if the populations follow a multivariate normality.

Manvir Kaur 253852
Carla Gironella 254136




```
corr1 = 0.9904
```

Carla Gironell

Table 4.2 Critical Points for the Q–Q Plot Correlation Coefficient Test for Normality

Sample size n	Significance levels α .01	.05	.10
5	.8299	.8788	.9032
10	.8801	.9198	.9351
15	.9126	.9389	.9503
20	.9269	.9508	.9604
25	.9410	.9591	.9665
30	.9479	.9652	.9715
35	.9538	.9682	.9740
40	.9599	.9726	.9771
45	.9632	.9749	.9792
50	.9671	.9768	.9809
55	.9695	.9787	.9822
60	.9720	.9801	.9836
75	.9771	.9838	.9866
100	.9822	.9873	.9895
150	.9879	.9913	.9928
200	.9905	.9931	.9942
300	.9935	.9953	.9960

Figure 1: Critical Point for the Q-Q Plot Correlation Coefficient Test for Normality

n of male is 174, and alpha is 0.05. The correlation coefficient computed from the plot is not enough to be able to not reject that the first dimension on male data follows a univariate normal distribution. We won't proceed because we will never be able to asses multivariate normality.

d) Test the hypotheses in (1.c) with the method selected in (2.c). Interpret the obtained result in terms of the research question.
To perform the test although not being sure that we are in front of multivariate normality distribution we will use bootstrap to asses normality and be able to perform the Hotelling-T^2. **Here's the code we have used:**

PROBLEM 3

We continue the analysis of Problem 1, but in this case we will apply Kernel PCA to reduce the dimensionality.

a) Apply Kernel-PCA to reduce the dimensionality of your input data and display the first 2 resulting components. Hint: the following kernel function is a reasonable choice for this dataset:

Manvir Kaur 253852
Carla Gironella 254136

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i^T \mathbf{x}_j\|\right)$$

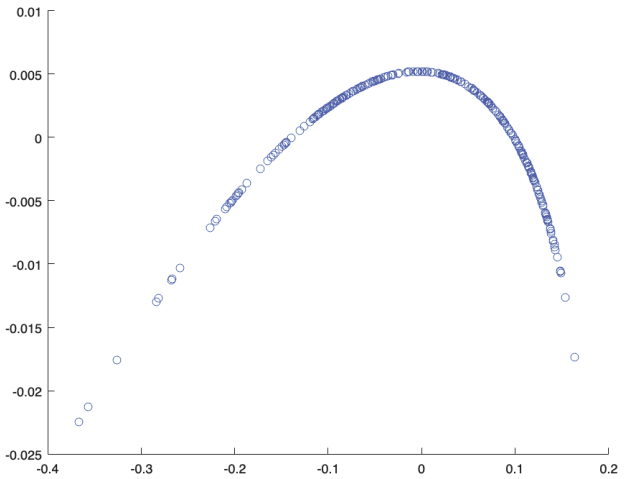
The value of γ can be tuned by visual inspection of the result (e.g. displaying the first two components and/or displaying a scree plot). It is also possible to use a different kernel of your choice.

Gamma is a crucial parameter which controls the shape of the kernel function. After some trial and error we have concluded to pick gamma = 0.01.

```
% Parameters
gamma = 0.01;

% Compute the kernel matrix
N = size(features, 1);
K = zeros(N(1), N(1));
for i = 1:N(1)
    for j = 1:N(1)
        K(i, j) = exp(-gamma * norm(features(i, :)' * features(j, :)));
    end
end


K

% We compute Kernel PCA:
H = eye(N(1)) -1/N(1) * ones(N(1));
my_matrix = H * K * H;


% We compute the eigenvectors and eigenvalues
cov_my_matrix = cov(my_matrix);
[V, D] = eig(cov_my_matrix);
[eigen_values_sorted, idx_sorted] = sort(diag(D), 'descend');

eigen_vectors_sorted = V(:, idx_sorted);

W = eigen_vectors_sorted(:,1:2);

Y = W' * my_matrix;


scatter(Y(1,:), Y(2,:))
xlabel('dim1');
ylabel('dim2');
```

We computed the kernel matrix and then we double centered the matrix (which is the my_matrix) after that we computed the covariance matrix to find it's eigenvalues and eigenvectors. W is the matrix of the sorted eigenvectors, we were asked to keep only the 2 first principal components, so this one only have 2 dimensions. Then we project this data and compute Y to then plot the scatter.
This is what we get:

Manvir Kaur 253852
Carla Gironella 254136



We obtained something looking like a parabola.

b) Choose the number of components to keep, p ′ , as the one that allows you to retain at least as much variance as the one that was retained by the significant components of PCA in Problem 1.

```
%B)

%we only keep one principal component
myeigVals=D(idx_sorted,idx_sorted)
myoriginal_eigen_vectors=V(:,idx_sorted)


mysum_var = trace(myeigVals)
myprincipal_component = sum(eigen_values_sorted(1:1))

mypercentage = myprincipal_component*100/mysum_var
```

In this section we computed the the sorted eigenvalues and eigenvectors. As we can observe here:

```
myeigVals = 285×285
    0.0134         0         0         0         0         0         0  ...
         0    0.0000         0         0         0         0         0
         0         0    0.0000         0         0         0         0
         0         0         0    0.0000         0         0         0
         0         0         0         0    0.0000         0         0
         0         0         0         0         0    0.0000         0
         0         0         0         0         0         0    0.0000
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
    ⋮
```

only the first eigenvalue is the one that looks significant, the other values are really small.
Here we can see the sorted eigenvalues:

```
myoriginal_eigen_vectors = 285×285
   -0.0276    0.0458   -0.0521   -0.0306   -0.0567    0.0598   -0.0167  ...
   -0.0383    0.0292   -0.0593    0.0007   -0.0548    0.0318    0.0050
   -0.0524   -0.0056   -0.0470    0.0435   -0.0113    0.0986   -0.0054
   -0.0019    0.0618   -0.0135   -0.0649   -0.0063   -0.0032    0.0002
   -0.0449    0.0151   -0.0576    0.0220   -0.0396   -0.0183   -0.0400
   -0.0341    0.0366   -0.0576   -0.0124   -0.0575   -0.0249   -0.0033
   -0.0098    0.0598   -0.0267   -0.0610   -0.0251   -0.0008    0.0098
   -0.0449    0.0152   -0.0577    0.0219   -0.0379   -0.0208   -0.0159
   -0.0217    0.0521   -0.0449   -0.0439   -0.0492   -0.0075   -0.0078
   -0.0551   -0.0144   -0.0401    0.0489    0.0009   -0.0203    0.0148
    ⋮
```

Manvir Kaur 253852
Carla Gironella 254136

We thought of keeping only the first principal component, and to see if this makes sense we computed the percentage explained by this principal component. We got a 99.8137% which means that it explains almost all the variance.

```
mysum_var = 0.0134
myprincipal_component = 0.0134
mypercentage = 99.8137
```

c) Select an appropriate method to test the hypotheses in (1.c) (an appropriate method is one for which all necessary assumptions are fulfilled by the data). Because you may end up with p ′ relatively large, testing for multivariate normality is intensive and thus we will skip it for now; in other words, let us assume that the input data comes from a multivariate normal distribution. Additionally, be aware that a large p ′ will challenge any attempt to assume large n for these data.

To do this exercise we will assume everything done in previous exercises is correct, so we will pick the test considering that. We are comparing the means of two populations (male and female), and we kept only one dimension because of the previous exercise, so as we only have one dimension we need to do t-student.

these are the formulas used:

$$df = n_1 + n_2 - 2 \qquad t = \frac{(\overline{x_1} - \overline{x_2})}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}$$

d) Test the hypotheses in (1.c) with the method selected in (3.c) under the assumption of multivariate normality. Interpret the obtained result in terms of the research question.

We only keept the first principal component, computing the W selected and the projection. We got the dataset of male and female of only one dimension.

```
%D)
% Keep only the first significant dimension
Wsel = eigen_vectors_sorted(:, 1);

% Project the data onto the significant dimension
Ysel = Wsel' * my_matrix;

% Scatter plot for visualization
scatter(Ysel, zeros(size(Ysel)));
xlabel('Significant Dimension');


index_male = 1;
index_female = 1;

dataset_male = zeros(174, 1);
dataset_female = zeros(111, 1);


for i=1:N
    if data(i,1) == 1
        dataset_male(index_male,:) = Ysel(:,i);
        index_male = index_male+1;
    end
    if data(i,1) == 0
        dataset_female(index_female, :) = Ysel(:,i);
        index_female = index_female+1;
    end

end

dataset_male;
dataset_female;
```

Manvir Kaur 253852
Carla Gironella 254136

```
%------t-student--> only have 1 dimension

% means
mymean_male = mean(dataset_male);
mymean_female = mean(dataset_female);

% Sample sizes
myn1 = size(dataset_male, 1);
myn2 = size(dataset_female, 1);

% Calculate degrees of freedom
dof = myn1 + myn2 - 2;

% Calculate standard deviations
mystd_male = std(dataset_male);
mystd_female = std(dataset_female);

% Calculate observed t-statistic
myobs_t = (mymean_female - mymean_male) / sqrt((mystd_male^2 / myn1) + (mystd_female^2 / myn2));

% Find the critical t-value for a two-tailed test
t_critical = tinv(1 - alpha/2, dof);

if abs(myobs_t) > t_critical
    disp('Reject');
else
    disp('Fail to reject');
end
```

```
Fail to reject
```

As a result we got that it fails to reject the null hypothesis.

e) Now we revisit the assumption of multivariate normality. Because assessing this for a large p ′ is very time-consuming, we will instead check what would happen if we cannot assume normality. In such case, we could still use the same statistic selected in (c) if we use bootstrap to assess its distribution under our null hypothesis. Proceed in this way to repeat (e) but without the assumption of multivariate normality. Compare the results obtained in (d) and (e).

We have performed the same steps as before computing for all the bootstrap samples (of both different males and females) the new observed t onto an array called tBOOT.

```
%E)

R = 1000;
tBOOT = zeros(1,R);

for b=1: R
    random_indices_male = ceil(myn1*rand(1, myn1));
    random_indices_female = ceil(myn2*rand(1, myn2));

    male_boot =dataset_male(random_indices_male, :);
    female_boot =dataset_female(random_indices_female, :);

    female_mean = mean(female_boot,1);
    male_mean = mean(male_boot, 1);

    mystd_male = std(male_boot);
    mystd_female = std(female_boot);

    tBOOT(b) = (female_mean - mymean_male) / sqrt((mystd_male/myn1) + (mystd_female/myn2));
end

mypvalue = (mean(abs(tBOOT) > abs(myobs_t))+1)/(R+1);

if mypvalue < alpha
    display('Reject');
else
    display('Fail to reject')
end
```

```
Reject
```

.In Part (d), the analysis assumes the presence of multivariate normality, where the hypothesis test relies on a statistic derived from this assumption. The test statistic is computed under the assumption of multivariate normality, and critical values are determined from the t-squared distribution.

Moving on to Part (e), the analysis loosens the assumption of multivariate normality. Instead, it adopts a bootstrap resampling approach to estimate the distribution of the test statistic under the null hypothesis. Bootstrap samples are drawn from the observed data, and the test statistic is calculated for each resampled dataset. The p-value is subsequently determined based on the distribution of these bootstrap test statistics.

Manvir Kaur 253852
Carla Gironella 254136

The divergence in results between Parts (d) and (e) is primarily attributed to the sensitivity of the test to the normality assumption. Any deviation of the data from multivariate normality could potentially violate the assumption in Part (d), leading to less accurate results. In contrast, the bootstrap approach in Part (e) offers a more robust alternative that doesn't hinge on assumptions of normality. Through resampling from the observed data, it constructs an empirical distribution for the test statistic, providing a more accurate evaluation of statistical significance.