Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales

Máster en Automática y Robótica

Applied Artificial Intelligence

Assignment 2.2: Principal Component Analysis

Josep María Barberá Civera (17048)

February 26, 2024

ASSIGNMENT 2.2

Principal Component Analysis

1.1 Methodology

Here we describe the steps followed for the PCA analysis.

1. First load the synthetically created "data_D2_C2" data set and then proceed to normalize the "p.value" and "t.value" datasets.
2. Next, the 1D and 2D coordinates of the normalized data projected onto the principal component subspace (PCA) are calculated.
3. After denormalizing the projected data, they should be plotted next to the original data set (raw data).
4. Finally, the accuracy of the reconstruction should be evaluated by calculating the expected mean squared error (MSE) of the normalized data and the actual MSE for both the normalized and original data, which should provide information on the effectiveness of the PCA analysis.

1.2 Results

First, the raw data were plotted against the normalized data. It should be noted at this point that normalization involves dividing by the standard deviation and subtracting the mean or simply subtracting the mean without the standard deviation division. As it us usually recommended to use the standard deviation, we have proceeded in this way. The following figure 1 shows the original data and the normalized data.

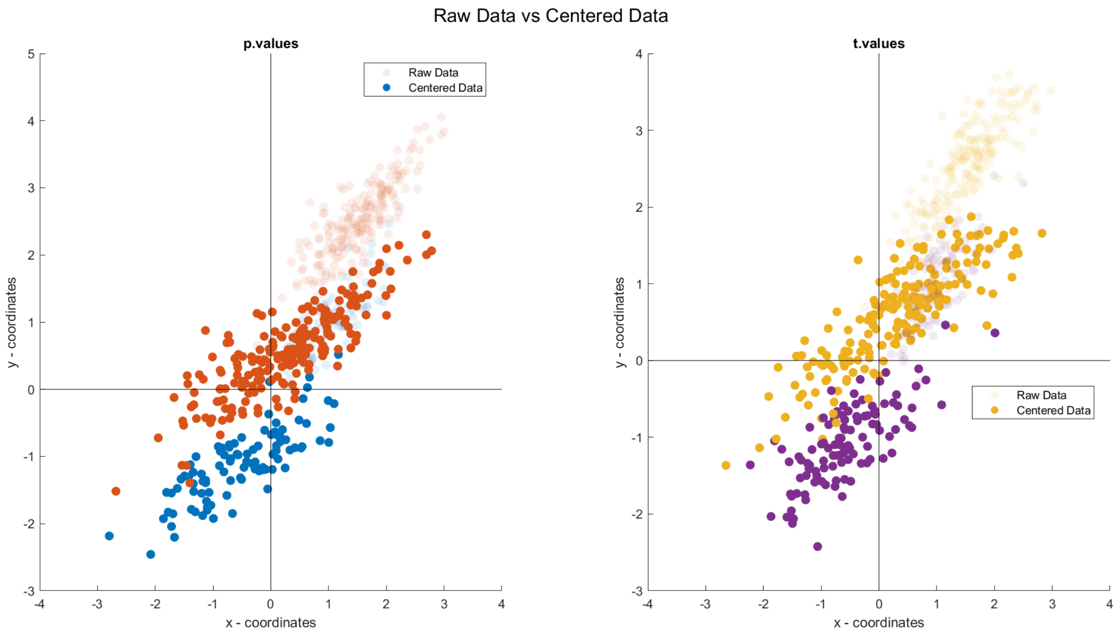

Figure 1: *Normalized Data: row data in translucent color vs. normalized data. P-values and T-values datasets are plotted the colors refer to the associated labels.*

After calculating the principal components, the normalized data are projected onto the component with the most information (the most significant) and, after deprojecting these points, they are plotted together with the original data (see Figure 2).

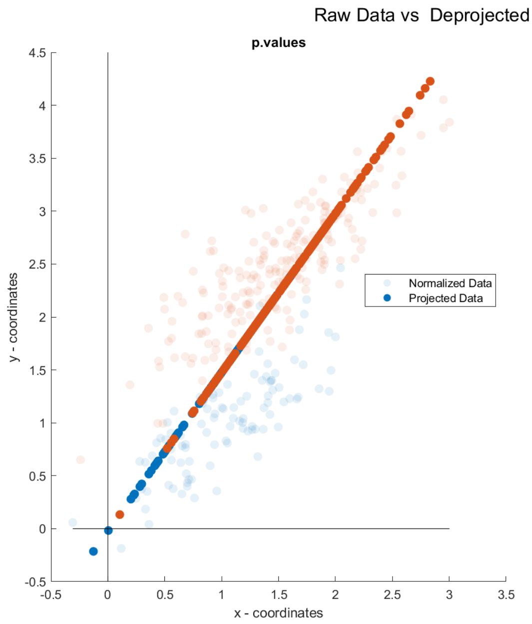

Figure 2: *Principal Component: the raw data is plotted again and now the most significant component has been used. Normalizing the data, projecting onto the eigenvector with the highest eigenvalue, deprojecting and denormalizing.*

A similar procedure was followed for the second component. The figure 3 shows the result.

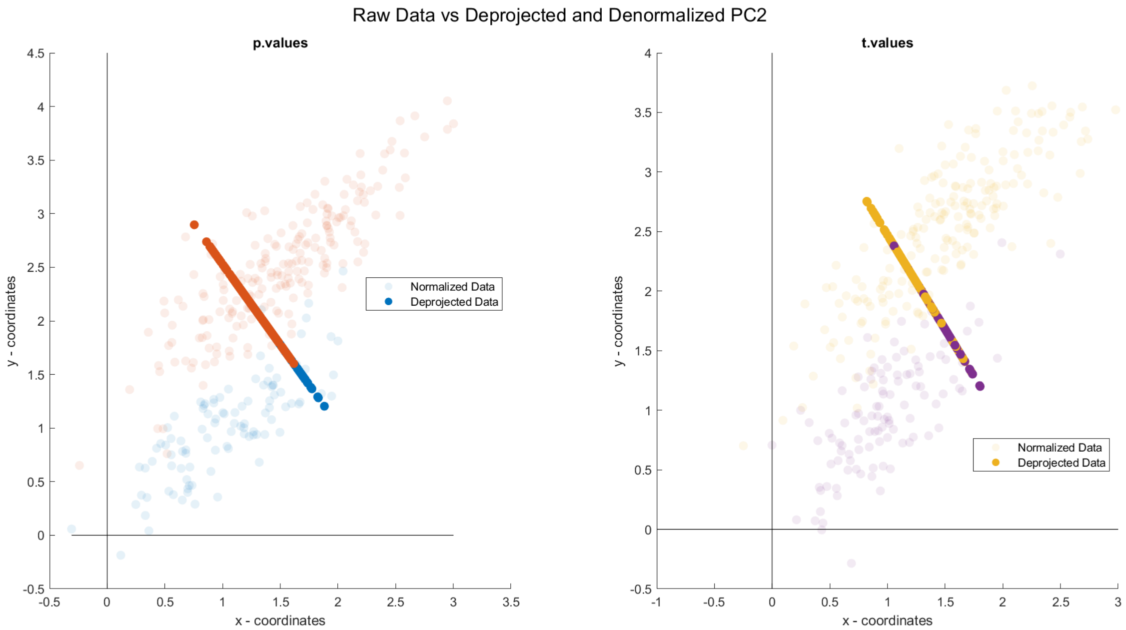

Figure 3: *The Second Component: as previous figure 2 the projected and deprojected data is plotted. Now the second eigenvector is used.*

Finally, it is projected on both components. The graph shows the result of this projection against the original data before deprojecting. In the figure 4, the deprojected data can be observed, which coincide with the original data (what we call `Raw Data`), since there are only two main components and there is no loss of information.

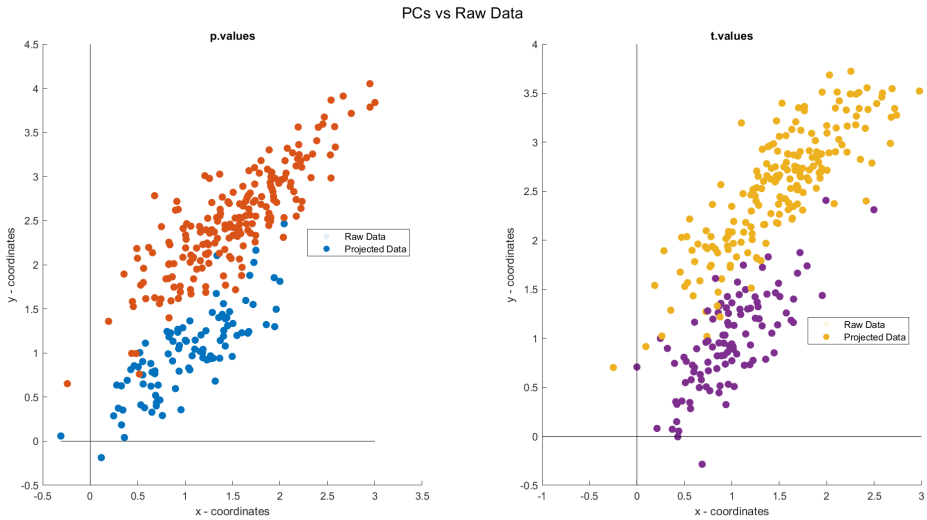

Figure 4: *PC 1 and PC 2: both components are analyzed, for this data it is meaningless because we only have two dimensions and therefore the reconstruction give us the same points than the original data (raw data).*

In addition, it has been plotted in a bar chart (Figure 5 with the value of the relative eigenvalues and we can see how the principal component is the most significant and allows us to represent ∼90% of the information with a single axis.

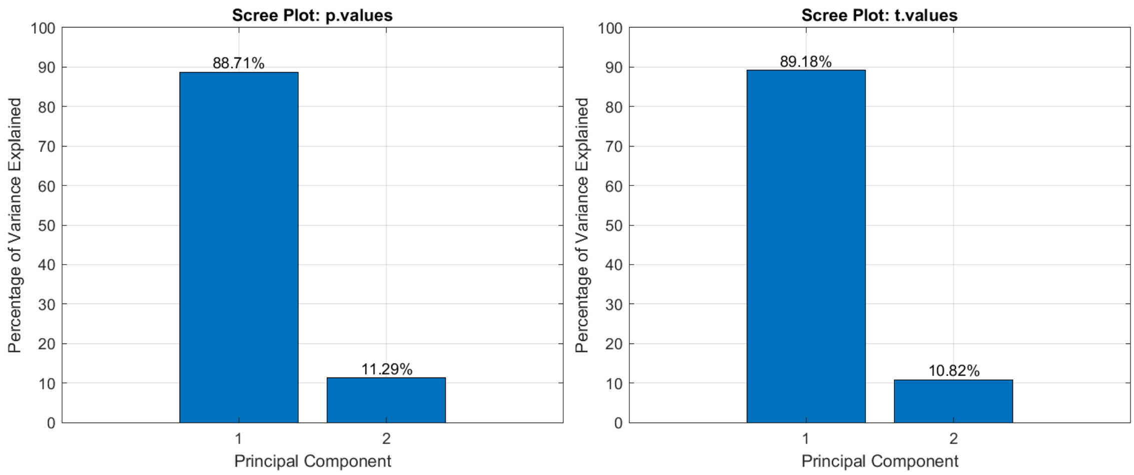

Figure 5: *Scree Plots for P-values and T-values data set. It is rapidly conclude how only one component is enough in order to meaningfully represent the information.*

1.3 Discussions and Results

We have been able to learn from several reference readings what PCA analysis consists of and its relevance in the case of image compression [1, 2].

In addition, it has been proven with synthetic data how the principal components of a point cloud can be found and how to project the information on these.

The calculation of errors in this analysis has been done in three way. First we have computed the **expected MSE for PC1** as

$$E[MSE] = \sum_{j=m+1}^{D} \lambda_j = \lambda_2$$

where D is the dimension of our data (in this case is 2), and m is the number of components we use (in this case 1). So for our case, the expected error corresponds directly with the second eigenvalue. In the case of `p-values` the $E[MSE] = 0.22587$ and in the case of `t-values` it is $E[MSE] = 0.21643$.

The other two errors computed are the **actual MSE of the normalized data** which for the `p-values` corresponds to $MSE_{norm} = 0.22512$ and for the `t-values` results $MSE_{norm} = 0.21571$. On the other hand, the **actual MSE of the not normalized data** is $MSE = 0.12886$ for the `p-values` and $MSE = 0.13140$ for the `t-values`.

References

[1] J. Shlens. *A Tutorial on Principal Component Analysis*. 2014. arXiv: 1404.1100 [cs.LG].

[2] L. I. Smith. *A tutorial on principal components analysis*. Tech. rep. Cornell University, USA, Feb. 2002. URL: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.

1.4 Relevant Code

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Master in Robotics
%              Applied Artificial Intelligence
%
% Assinment 2.2: Principal Component Analysis
% Student: Josep Barbera Civera
% ID: 17048
% Date: 20/02/2024
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 0. Load data_D2_C2 and rnrmalize the data (p.value and t.value)
% 1. Compute the 1D coordinates of the normalized test data
% projected on the PCA sub-space
% 2. Compute the 2D coordinates of the normalized test data
% projected on the PCA sub-space
% 3. De-normalize the projected data and plot it in the same
% graphic as original data
% 4. Compute the reconstruction, the expected MSE of normalized
% data, the actual MSE of the normalized data and the actual
% MSE of the not normalized data.

load data_D2_C2.mat

%% Accesing Data
pvalues = p.value;
plabels = p.class;
tvalues = t.value;
tlabels = t.class;

%% Normalizing data
disp("-------- Normalizing Data ---------------------------");
disp("Normalizing P-values");
[Mp, Np] = size(pvalues);
mn_p = mean(pvalues')';
std_p = std(pvalues')';
for i = 1:Np
    pn(:,i) = (pvalues(:,i) - mn_p)./std_p;
end
disp("Normalizing T-values");
```

```
[Mt, Nt] = size(tvalues);
mn_t = mean(tvalues')';
std_t = std(tvalues')';
for i = 1:Nt
    tn(:,i) = (tvalues(:,i) - mn_t)./std_t;
end

%% Plotting Normalized Data vs Raw Data
two_plots('Raw Data vs Centered Data', 'p.values', 't.values', ...
            'x - coordinates', 'y - coordinates', 'x - coordinates', ...
            'y - coordinates', 'Raw Data', 'Centered Data', ...
            'Raw Data', 'Centered Data', pvalues, tvalues, pn, tn, ...
            plabels, tlabels, 'centered_data_vs_raw_data.png');

%% Computing the Covariance Matrix
disp("-------- Computing the Covariance Matrix ----------------");
cov_pn = 1 / (Np-1) * pn * pn';
cov_tn = 1 / (Nt-1) * tn * tn';

%% Computing the Eigenvalues and Eigenvectors
disp("-------- Computing the Eigenvalues and Eigenvectors -----");
[PC_p, Vp] = eig(cov_pn);
[PC_t, Vt] = eig(cov_tn);

%% Sort the variances in decreasing order
disp("-------- Sorting variances in decreasing order ---------");
% Extract diagonal of matrix as vector
Vp = diag(Vp);
Vt = diag(Vt);
% Sort PC_p and convert Vp to a column vector with the eigenvalues
[~, p_rindices] = sort(-1*Vp);
Vp = Vp(p_rindices);
PC_p= PC_p(:, p_rindices);
% Sort PC_t and convert Vt to a column vector with the eigenvalues
[~, t_rindices] = sort(-1*Vt);
Vt = Vt(t_rindices);
PC_t= PC_t(:,t_rindices);
```

```
%% Projection 1D-PCA1
disp("-------- Computing PC1 ---------------------------------");
% We project on the most significant PC
PC = PC_p(:,1);
signals_p = PC' * pn;
original_pvalues_1D_1 = (PC * signals_p);
for i = 1:Np
    original_pvalues_1D_1_desnorm(:,i) = original_pvalues_1D_1(:,i).*std_p + mn_p;
end

PC = PC_t(:,1);
signals_t = PC' * tn;
original_tvalues_1D_1 = (PC * signals_t);
for i = 1:Nt
    original_tvalues_1D_1_desnorm(:,i) = original_tvalues_1D_1(:,i).*std_t + mn_t;
end

%% Plotting Normalized Data vs PC1
% two_plots('Raw Data vs Deprojected and denormalized Data on PC1', ...
%             'p.values', 't.values', 'x - coordinates', ...
%             'y - coordinates', 'x - coordinates', 'y - coordinates', ...
%           'Normalized Data', 'Projected Data', 'Normalized Data', ...
%           'Projected Data', pvalues, tvalues, original_pvalues_1D_1_desnorm, ...
%           original_tvalues_1D_1_desnorm, plabels, tlabels, ...
%           'Deprojected_and_denormalized_PC1_vs_Raw_Data.png');

%% Projection 1D-PCA2
disp("-------- Computing PC2 ------------------------------");
% We project on the **second** most significant PC
PC = PC_p(:,2);
signals_p = PC' * pn;
original_pvalues_1D_2 = (PC * signals_p);
for i = 1:Np
    original_pvalues_1D_2_desnorm(:,i) = original_pvalues_1D_2(:,i).*std_p + mn_p;
end

```

```
PC = PC_t(:,2);
signals_t = PC' * tn;
original_tvalues_1D_2 = (PC * signals_t);
for i = 1:Nt
    original_tvalues_1D_2_desnorm(:,i) = original_tvalues_1D_2(:,i).*std_t + mn_t;
end

%% Plotting Normalized Data vs PC2
two_plots('Raw Data vs Deprojected and Denormalized PC2', 'p.values', ...
            't.values', 'x - coordinates', 'y - coordinates', ...
            'x - coordinates', 'y - coordinates', ' Normalized Data', ...
            'Deprojected Data', 'Normalized Data', ' Deprojected Data', ...
            pvalues, tvalues, original_pvalues_1D_2_desnorm, ...
            original_tvalues_1D_2_desnorm, plabels, tlabels, ...
            'Deprojected_and_Denormalized_PC2_vs_Raw_Data.png');

%% Plotting Scree Plot
disp("-------- Plotting Scree Plots ----------------------------");
% s = scree_plot(Vp, ' p.values');
% saveas(s, "scree_plot_PCA_p.png");
% s = scree_plot(Vt, ' t.values');
% saveas(s, "scree_plot_PCA_t.png");

%% Computing Errors
disp("-------- Computing Error --------------------------------");
% Expected Error
mse_expected = [Vp(2) Vt(2)];
sprintf('Expected MSE for PC1: p= %0.5f, t= %0.5f', mse_expected(1), mse_expected(2))

% Actual Error with the normalized data
mse_actual_normalized_p = (pn - original_pvalues_1D_1).^2;
for i=1:Np
    mse_sum_p_norm(i) = mse_actual_normalized_p(1,i) + mse_actual_normalized_p(2,i);
end
mse_p_norm = mean(mse_sum_p_norm);
```

```
mse_actual_normalized_t = (tn - original_tvalues_1D_1).^2;
for i=1:Nt
    mse_sum_t_norm(i) = mse_actual_normalized_t(1,i) + mse_actual_normalized_t(2,i);
end
mse_t_norm = mean(mse_sum_t_norm);
sprintf('Actual MSE for Normalized PC1: p= %0.5f, t= %0.5f', mse_p_norm, mse_t_norm)

% Actual Error with the de-normalized data
mse_actual_de_normalized_p = (pvalues - original_pvalues_1D_1_desnorm).^2;
for i=1:Np
    mse_sum_p_de_norm(i) = mse_actual_de_normalized_p(1,i) + mse_actual_de_normalized_p(2,i);
end
mse_p_de_norm = mean(mse_sum_p_de_norm);

mse_actual_de_normalized_t = (tvalues - original_tvalues_1D_1_desnorm).^2;
for i=1:Np
    mse_sum_t_de_norm(i) = mse_actual_de_normalized_t(1,i) + mse_actual_de_normalized_t(2,i);
end
mse_t_de_norm = mean(mse_sum_t_de_norm);
sprintf('Actual MSE for Normalized PC1: p= %0.5f, t= %0.5f', mse_p_de_norm, mse_t_de_norm)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting Functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function two_plots(generic_title, title_subplot_1, title_subplot_2, ...
                    x_1_label, y_1_label, x_2_label, y_2_label, ...
                    legend_1_light, legend_1, legend_2_light, ...
                    legend_2, light_data_1, light_data_2, data_1, ...
                    data_2, labels_1, labels_2, saved_name)
    disp("-------- Plotting -----------------------------------");
    disp(generic_title);

```

```
    figure;
    % Generic Title
    sgtitle(generic_title);
    % First Subplot
    subplot(1, 2, 1);
    for i=1:length(labels_1)
        if labels_1(i) == 1
            scatter(light_data_1(1,i), light_data_1(2,i), 50, 'o', ...
                    'MarkerEdgeColor', 'none', 'MarkerFaceColor', [0 0.4470 0.7410], ...
                    'MarkerFaceAlpha', 0.1); hold on;
            scatter(data_1(1,i), data_1(2,i), 50, 'o', ...
                    'MarkerEdgeColor', 'none', 'MarkerFaceColor', [0 0.4470 0.7410], ...
                    'MarkerFaceAlpha', 1); hold on;
        else
            scatter(light_data_1(1,i), light_data_1(2,i), 50, 'o', ...
                    'MarkerEdgeColor', 'none', 'MarkerFaceColor', [0.8500 0.3250 0.0980], ...
                    'MarkerFaceAlpha', 0.1); hold on;
            scatter(data_1(1,i), data_1(2,i), 50, 'o', ...
                    'MarkerEdgeColor', 'none', 'MarkerFaceColor', [0.8500 0.3250 0.0980], ...
                    'MarkerFaceAlpha', 1); hold on;
        end
    end
    % Plot vertical lines for x=0 and y=0
    plot([0 0], ylim, 'k-');
    plot(xlim, [0 0], 'k-');
    % Subplot title
    title(title_subplot_1);
    % Axis labels
    xlabel(x_1_label);
    ylabel(y_1_label);
    legend({legend_1_light, legend_1}, 'Location', 'best');

    % Second Subplot
    subplot(1, 2, 2);
    for i=1:length(labels_2)
        if labels_2(i) == 1
```

```
            scatter(light_data_2(1,i), light_data_2(2,i), 50, 'o', ...
                'MarkerEdgeColor', 'none', 'MarkerFaceColor', [0.4940 0.1840 0.5560], ...
                'MarkerFaceAlpha', 0.1); hold on;
            scatter(data_2(1,i), data_2(2,i), 50, 'o', 'MarkerEdgeColor', 'none', ...
                'MarkerFaceColor', [0.4940 0.1840 0.5560], 'MarkerFaceAlpha', 1);
            hold on;
        else
            scatter(light_data_2(1,i), light_data_2(2,i), 50, 'o', ...
                'MarkerEdgeColor', 'none', 'MarkerFaceColor', [0.9290 0.6940 0.1250], ...
                'MarkerFaceAlpha', 0.1); hold on;
            scatter(data_2(1,i), data_2(2,i), 50, 'o', 'MarkerEdgeColor', 'none', ...
                'MarkerFaceColor', [0.9290 0.6940 0.1250], 'MarkerFaceAlpha', 1);
            hold on;
        end
    end
    % Plot vertical lines for x=0 and y=0
    plot([0 0], ylim, 'k-');
    plot(xlim, [0 0], 'k-');
    % Subplot title
    title(title_subplot_2);
    % Axis labels
    xlabel(x_2_label);
    ylabel(y_2_label);
    legend({legend_2_light, legend_2}, 'Location', 'best');

    pos = get(gcf, 'Position');
    set(gcf, 'Position',pos+[-900 -300 900 300])
    saveas(gca, saved_name);
end

function s = scree_plot(eigvalues, mytitle)
    s = figure;
    explained_variance = eigvalues / sum(eigvalues) * 100;
    bar(explained_variance);
    ylim([0, 100]);
    title(strcat('Scree Plot: ', mytitle));
```

```
    xlabel('Principal Component');
    ylabel('Percentage of Variance Explained');
    grid on;
    % Add value on top of each bar
    for i = 1:length(explained_variance)
        text(i, explained_variance(i), sprintf('%.2f%%', explained_variance(i)), ...
            'HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom');
    end
    tightfig;
end
```