This is a spike to research the understanding of principal components analysis. As a data analyst, I need to understand principal components analysis which will allow me to create and understand preventive technology more.

Acceptance Criteria:

- Show an understanding of what principal components analysis means
- Teach team members on what principal components analysis means

---

Context:

1. What is PCA and what it does
2. Why we are using PCA
3. A pseudocode example of PCA

1. What is PCA and what it does

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables. PCA is mostly used as a tool in exploratory data analysis and for making predictive models. It's often used to visualize genetic distance and relatedness between populations. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after

mean centering (and normalizing or using Z-scores) the data matrix for each attribute. The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score). PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualised as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection of this object when viewed from its most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

---

2. Why we are using PCA

There are two primary reasons for using PCA:

1. Data Reduction- PCA is most commonly used to condense the information contained in a large number of original variables into a smaller set of new composite dimensions, with a minimum loss of information.

2. Interpretation- PCA can be used to discover important features of a large data set. It often reveals relationships that were previously unsuspected, thereby allowing interpretations that would not ordinarily result

PCA is a very practical way to discover important features of a large data set. It often reveals relationships that were previously unsuspected, thereby allowing interpretations that would not ordinarily result.

3. A pseudocode example of PCA

```
coeff = pca(X)
```

```
coeff = pca(X,Name,Value)
```

```
[coeff,score,latent] = pca(__)
```

```
[coeff,score,latent,tsquared] = pca(__)
```

```
[coeff,score,latent,tsquared,explained,mu] = pca(__)
```

Description

example

`coeff = pca(X)` returns the principal component coefficients, also known as loadings, for the n-by-p data matrix `X`. Rows of `X` correspond to observations and columns correspond to variables. The coefficient matrix is p-by-p. Each column of `coeff` contains coefficients for one principal component, and the columns are in descending order of component variance. By default, `pca` centers the data and uses the singular value decomposition (SVD) algorithm.

example

`coeff = pca(X,Name,Value)` returns any of the output arguments in the previous syntaxes using additional options for computation and handling of special data types, specified by one or more `Name,Value` pair arguments.

For example, you can specify the number of principal components `pca` returns or an algorithm other than SVD to use.

example

`[coeff,score,latent] = pca(__)` also returns the principal component scores in `score` and the principal component variances in `latent`. You can use any of the input arguments in the previous syntaxes.

Principal component scores are the representations of `X` in the principal component space. Rows of `score` correspond to observations, and columns correspond to components.

The principal component variances are the eigenvalues of the covariance matrix of `X`.

example

`[coeff,score,latent,tsquared] = pca(__)` also returns the Hotelling's T-squared statistic for each observation in `X`.

example

`[coeff,score,latent,tsquared,explained,mu] = pca(__)` also returns `explained`, the percentage of the total variance explained by each principal component and `mu`, the estimated mean of each variable in `X`.

Examples

collapse all

Principal Components of a Data Set

Load the sample data set.

```
load hald
```

The ingredients data has 13 observations for 4 variables.

Find the principal components for the ingredients data.

```
coeff = pca(ingredients)
```

```
coeff =

   -0.0678   -0.6460    0.5673    0.5062
   -0.6785   -0.0200   -0.5440    0.4933
    0.0290    0.7553    0.4036    0.5156
    0.7309   -0.1085   -0.4684    0.4844
```

The rows of `coeff` contain the coefficients for the four ingredient variables, and its columns correspond to four principal components.

PCA in the Presence of Missing Data

Find the principal component coefficients when there are missing values in a data set.

Load the sample data set.

```
load imports-85
```

Data matrix `X` has 13 continuous variables in columns 3 to 15: wheel-base, length, width, height, curb-weight, engine-size, bore, stroke, compression-ratio, horsepower, peak-rpm, city-mpg, and highway-mpg. The variables bore and stroke are missing four values in rows 56 to 59, and the variables horsepower and peak-rpm are missing two values in rows 131 and 132.

Perform principal component analysis.

```
coeff = pca(X(:,3:15));
```

By default, `pca` performs the action specified by the `'Rows','complete'` name-value pair argument. This option removes the observations with `NaN` values before calculation. Rows of `NaN`s are reinserted into `score` and `tsquared` at the corresponding locations, namely rows 56 to 59, 131, and 132.

Use `'pairwise'` to perform the principal component analysis.

```
coeff = pca(X(:,3:15),'Rows','pairwise');
```

In this case, `pca` computes the (i,j) element of the covariance matrix using the rows with no `NaN` values in the columns i or j of `X`. Note that the resulting covariance matrix might not be positive definite. This option applies when the algorithm `pca` uses is eigenvalue decomposition. When you don't specify the algorithm, as in this example, `pca` sets it to `'eig'`. If you require `'svd'` as the algorithm, with the `'pairwise'` option, then `pca` returns a warning message, sets the algorithm to `'eig'` and continues.

If you use the `'Rows','all'` name-value pair argument, `pca` terminates because this option assumes there are no missing values in the data set.

```
coeff = pca(X(:,3:15),'Rows','all');
```

```
Error using pca (line 180)
Raw data contains NaN missing value while 'Rows' option is set to 'all'. Consider
using 'complete' or pairwise' option instead.
```

Weighted PCA

Use the inverse variable variances as weights while performing the principal components analysis.

Load the sample data set.

```
load hald
```

Perform the principal component analysis using the inverse of variances of the ingredients as variable weights.

```
[wcoeff,~,latent,~,explained] = pca(ingredients,...
'VariableWeights','variance')
```

```
wcoeff =

   -2.7998    2.9940   -3.9736    1.4180
   -8.7743   -6.4411    4.8927    9.9863
    2.5240   -3.8749   -4.0845    1.7196
    9.1714    7.5529    3.2710   11.3273

latent =

    2.2357
    1.5761
    0.1866
    0.0016

explained =

   55.8926
   39.4017
    4.6652
    0.0406
```

Note that the coefficient matrix, `wcoeff`, is not orthonormal.

Calculate the orthonormal coefficient matrix.

```
coefforth = inv(diag(std(ingredients)))* wcoeff
```

```
coefforth =

   -0.4760    0.5090   -0.6755    0.2411
   -0.5639   -0.4139    0.3144    0.6418
    0.3941   -0.6050   -0.6377    0.2685
```

```
    0.5479    0.4512    0.1954    0.6767
```

Check orthonormality of the new coefficient matrix, `coefforth`.

```
coefforth*coefforth'
```

```
ans =

    1.0000         0   -0.0000    0.0000
         0    1.0000    0.0000   -0.0000
   -0.0000    0.0000    1.0000    0.0000
    0.0000   -0.0000    0.0000    1.0000
```

PCA Using ALS for Missing Data

Find the principal components using the alternating least squares (ALS) algorithm when there are missing values in the data.

Load the sample data.

```
load hald
```

The ingredients data has 13 observations for 4 variables.

Perform principal component analysis using the ALS algorithm and display the component coefficients.

```
[coeff,score,latent,tsquared,explained] = pca(ingredients);
coeff
```

```
coeff =

   -0.0678   -0.6460    0.5673    0.5062
   -0.6785   -0.0200   -0.5440    0.4933
    0.0290    0.7553    0.4036    0.5156
```

```
    0.7309   -0.1085   -0.4684    0.4844
```

Introduce missing values randomly.

```
y = ingredients;
rng('default'); % for reproducibility
ix = random('unif',0,1,size(y))<0.30;
y(ix) = NaN
```

```
y =

     7    26     6   NaN
     1    29    15    52
   NaN   NaN     8    20
    11    31   NaN    47
     7    52     6    33
   NaN    55   NaN   NaN
   NaN    71   NaN     6
     1    31   NaN    44
     2   NaN   NaN    22
    21    47     4    26
   NaN    40    23    34
    11    66     9   NaN
    10    68     8    12
```

Approximately 30% of the data has missing values now, indicated by `NaN`.

Perform principal component analysis using the ALS algorithm and display the component coefficients.

```
[coeff1,score1,latent,tsquared,explained,mu1] = pca(y,...
'algorithm','als');
```

```
coeff1
```

```
coeff1 =

   -0.0362    0.8215   -0.5252    0.2190
   -0.6831   -0.0998    0.1828    0.6999
    0.0169    0.5575    0.8215   -0.1185
    0.7292   -0.0657    0.1261    0.6694
```

Display the estimated mean.

```
mu1
```

```
mu1 =

    8.9956   47.9088    9.0451   28.5515
```

Reconstruct the observed data.

```
t = score1*coeff1' + repmat(mu1,13,1)
```

```
t =

    7.0000   26.0000    6.0000   51.5250
    1.0000   29.0000   15.0000   52.0000
   10.7819   53.0230    8.0000   20.0000
   11.0000   31.0000   13.5500   47.0000
    7.0000   52.0000    6.0000   33.0000
   10.4818   55.0000    7.8328   17.9362
    3.0982   71.0000   11.9491    6.0000
    1.0000   31.0000   -0.5161   44.0000
    2.0000   53.7914    5.7710   22.0000
   21.0000   47.0000    4.0000   26.0000
   21.5809   40.0000   23.0000   34.0000
   11.0000   66.0000    9.0000    5.7078
```

```
   10.0000   68.0000    8.0000   12.0000
```

The ALS algorithm estimates the missing values in the data.

Another way to compare the results is to find the angle between the two spaces spanned by the coefficient vectors. Find the angle between the coefficients found for complete data and data with missing values using ALS.

```
subspace(coeff,coeff1)
```

```
ans =

   6.6886e-16
```

This is a small value. It indicates that the results if you use `pca` with `'Rows','complete'` name-value pair argument when there is no missing data and if you use `pca` with `'algorithm','als'` name-value pair argument when there is missing data are close to each other.

Perform the principal component analysis using `'Rows','complete'` name-value pair argument and display the component coefficients.

```
[coeff2,score2,latent,tsquared,explained,mu2] = pca(y,...
'Rows','complete');
coeff2
```

```
coeff2 =

   -0.2054    0.8587    0.0492
   -0.6694   -0.3720    0.5510
    0.1474   -0.3513   -0.5187
    0.6986   -0.0298    0.6518
```

In this case, `pca` removes the rows with missing values, and `y` has only four rows with no missing values. `pca` returns only three principal components. You cannot use the `'Rows','pairwise'` option because the covariance matrix is not positive semidefinite and `pca` returns an error message.

Find the angle between the coefficients found for complete data and data with missing values using listwise deletion (when `'Rows','complete'`).

```
subspace(coeff(:,1:3),coeff2)
```

```
ans =

    0.3576
```

The angle between the two spaces is substantially larger. This indicates that these two results are different.

Display the estimated mean.

```
mu2
```

```
mu2 =

    7.8889   46.9091    9.8750   29.6000
```

In this case, the mean is just the sample mean of `y`.

Reconstruct the observed data.

```
score2*coeff2'
```

```
ans =

       NaN       NaN       NaN       NaN
   -7.5162  -18.3545    4.0968   22.0056
       NaN       NaN       NaN       NaN
```

```
     NaN       NaN       NaN       NaN
 -0.5644    5.3213   -3.3432    3.6040
     NaN       NaN       NaN       NaN
     NaN       NaN       NaN       NaN
     NaN       NaN       NaN       NaN
     NaN       NaN       NaN       NaN
 12.8315   -0.1076   -6.3333   -3.7758
     NaN       NaN       NaN       NaN
     NaN       NaN       NaN       NaN
  1.4680   20.6342   -2.9292  -18.0043
```

This shows that deleting rows containing `NaN` values does not work as well as the ALS algorithm. Using ALS is better when the data has too many missing values.

Principal Component Coefficients, Scores, and Variances

Try it in MATLAB

Find the coefficients, scores, and variances of the principal components.

Load the sample data set.

```
load hald
```

The ingredients data has 13 observations for 4 variables.

Find the principal component coefficients, scores, and variances of the components for the ingredients data.

```
[coeff,score,latent] = pca(ingredients)
```

```
coeff =

   -0.0678   -0.6460    0.5673    0.5062
   -0.6785   -0.0200   -0.5440    0.4933
    0.0290    0.7553    0.4036    0.5156
    0.7309   -0.1085   -0.4684    0.4844
```

```
score =

   36.8218   -6.8709   -4.5909    0.3967
   29.6073    4.6109   -2.2476   -0.3958
  -12.9818   -4.2049    0.9022   -1.1261
   23.7147   -6.6341    1.8547   -0.3786
   -0.5532   -4.4617   -6.0874    0.1424
  -10.8125   -3.6466    0.9130   -0.1350
  -32.5882    8.9798   -1.6063    0.0818
   22.6064   10.7259    3.2365    0.3243
   -9.2626    8.9854   -0.0169   -0.5437
   -3.2840  -14.1573    7.0465    0.3405
    9.2200   12.3861    3.4283    0.4352
  -25.5849   -2.7817   -0.3867    0.4468
  -26.9032   -2.9310   -2.4455    0.4116


latent =

  517.7969
   67.4964
   12.4054
    0.2372
```

Each column of `score` corresponds to one principal component. The vector, `latent`, stores the variances of the four principal components.

Reconstruct the centered ingredients data.

```
Xcentered = score*coeff'
```

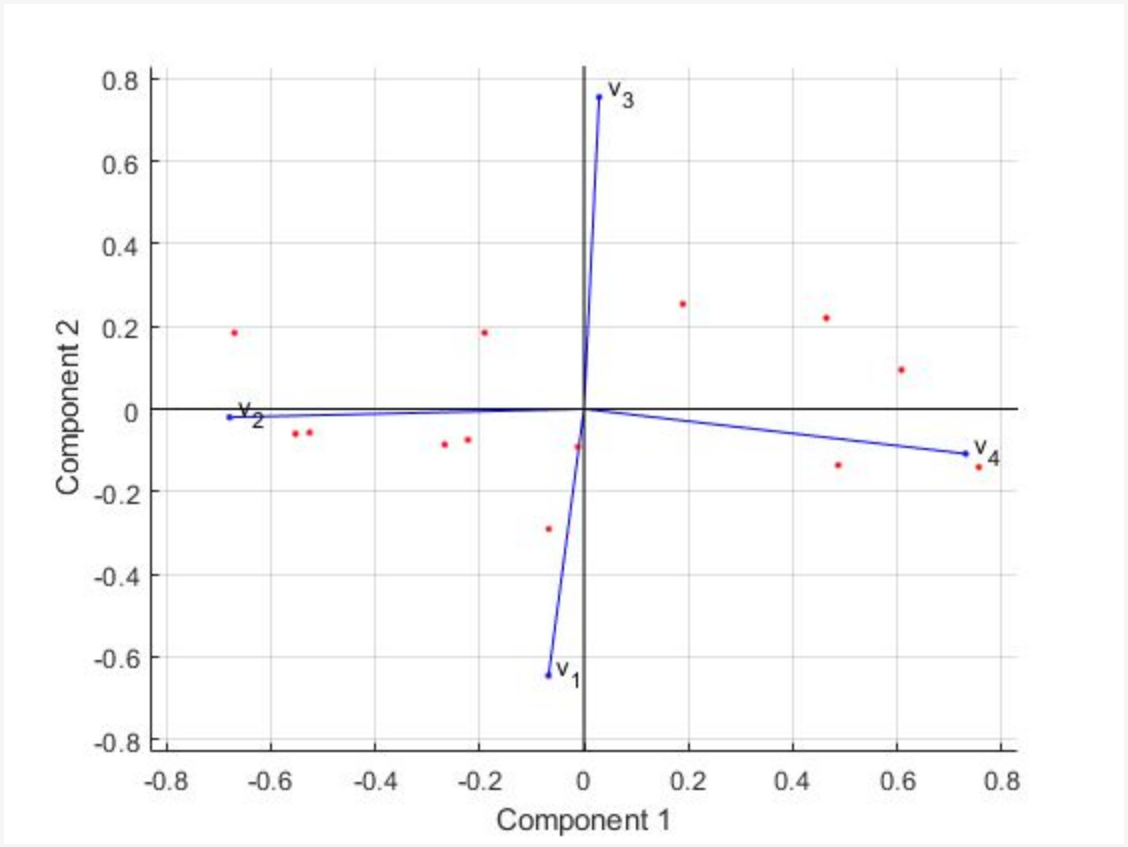
```
Xcentered =

   -0.4615  -22.1538   -5.7692   30.0000
```

```
   -6.4615  -19.1538    3.2308   22.0000
    3.5385    7.8462   -3.7692  -10.0000
    3.5385  -17.1538   -3.7692   17.0000
   -0.4615    3.8462   -5.7692    3.0000
    3.5385    6.8462   -2.7692   -8.0000
   -4.4615   22.8462    5.2308  -24.0000
   -6.4615  -17.1538   10.2308   14.0000
   -5.4615    5.8462    6.2308   -8.0000
   13.5385   -1.1538   -7.7692   -4.0000
   -6.4615   -8.1538   11.2308    4.0000
    3.5385   17.8462   -2.7692  -18.0000
    2.5385   19.8462   -3.7692  -18.0000
```

The new data in `Xcentered` is the original ingredients data centered by subtracting the column means from corresponding columns.

Visualize both the orthonormal principal component coefficients for each variable and the principal component scores for each observation in a single plot.

```
biplot(coeff(:,1:2),'scores',score(:,1:2),'varlabels',{'v_1','v_2','v_3','v_4'});
```



All four variables are represented in this biplot by a vector, and the direction and length of the vector indicate how each variable contributes to the two principal components in the plot. For example, the first principal component, which is on the horizontal axis, has positive coefficients for the third and fourth variables. Therefore, vectors v_3 and v_4 are directed into the right half of the plot. The largest coefficient in the first principal component is the fourth, corresponding to the variable v_4.

The second principal component, which is on the vertical axis, has negative coefficients for the variables v_1, v_2, and v_4, and a positive coefficient for the variable v_3.

This 2-D biplot also includes a point for each of the 13 observations, with coordinates indicating the score of each observation for the two principal components in the plot. For example, points near the left edge of the plot have the lowest scores for the first principal component. The points are scaled with respect to the maximum score value and maximum coefficient length, so only their relative locations can be determined from the plot.