Dimension Reduction Techniques

Principal Component Analysis and Factor Analysis

by

Dr. Tanujit Chakraborty

Assistant Professor in Statistics
Sorbonne University Abu Dhabi
`https://www.ctanujit.org`

Mail: tanujitisi@gmail.com

High Dimensional Paradigm

- Most of the modern day applications produce data that are high dimensional.

High Dimensional Paradigm

- Most of the modern day applications produce data that are high dimensional.
- A 1024×1024 gray scale image contain 1048576 pixels and each pixel having the gray scale value.

- Can be considered as a 1048576 component vector or a data point in $\mathbb{R}^{1048576}$.
- For medical images and massive astrophysics images, the number p of pixels can be larger than the number n of images.

Curse of dimensionality

- Being able to sense too many variables often causes a problem which we called **curse of dimensionality**.
  - cannot apply ordinary least square estimates
  - problem in local regression techniques, like K-NN
  - accumulation of errors

Intrinsic low dimensional structure

- A fact of hope is that observations are not randomly scattered in $\mathbb{R}^d$, rather they are concentrated around some low dimensional subspaces.

Intrinsic low dimensional structure

- A fact of hope is that observations are not randomly scattered in $\mathbb{R}^d$, rather they are concentrated around some low dimensional subspaces.
- These low dimensional structures in the data are the result of the inherent simplicity in the system producing the data.

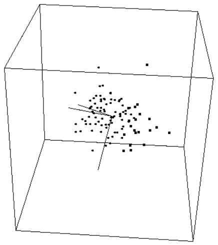
Intrinsic low dimensional structure

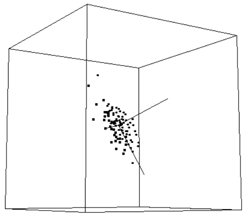
- A fact of hope is that observations are not randomly scattered in $\mathbb{R}^d$, rather they are concentrated around some low dimensional subspaces.
- These low dimensional structures in the data are the result of the inherent simplicity in the system producing the data.
- But the problem is that these low dimensional structures are unknown to us.

Intrinsic low dimensional structure

- A fact of hope is that observations are not randomly scattered in $\mathbb{R}^d$, rather they are concentrated around some low dimensional subspaces.
- These low dimensional structures in the data are the result of the inherent simplicity in the system producing the data.
- But the problem is that these low dimensional structures are unknown to us.
- Our objective today is to identify the low dimensional structures in the data.

Consider the 3-d scatterplot





- Rotating it we see that the points are all actually in a plane.
- So the intrinsic dimension is 2, and not 3.

How do we reduce the dimension?

- In the example above, we could identify the low dimensional structure by rotating the scatterplot.

How do we reduce the dimension?

- In the example above, we could identify the low dimensional structure by rotating the scatterplot.
- In case of further high dimension, we cannot make such plot.

How do we reduce the dimension?

- In the example above, we could identify the low dimensional structure by rotating the scatterplot.
- In case of further high dimension, we cannot make such plot.
- How can we identify the low dimensional structures?

How do we reduce the dimension?

- In the example above, we could identify the low dimensional structure by rotating the scatterplot.
- In case of further high dimension, we cannot make such plot.
- How can we identify the low dimensional structures?
- This is the main point of concern of all **dimension reduction techniques**.

Both PCA and FA reduce dimensions

- Both Principal component analysis and Factor Analysis reveal the low dimensional structure in the data.

Both PCA and FA reduce dimensions

- Both Principal component analysis and Factor Analysis reveal the low dimensional structure in the data.
- But they are not same.

Both PCA and FA reduce dimensions

- Both Principal component analysis and Factor Analysis reveal the low dimensional structure in the data.
- But they are not same.
- PCA reveals low dimensional structures derived from the original variables.

Both PCA and FA reduce dimensions

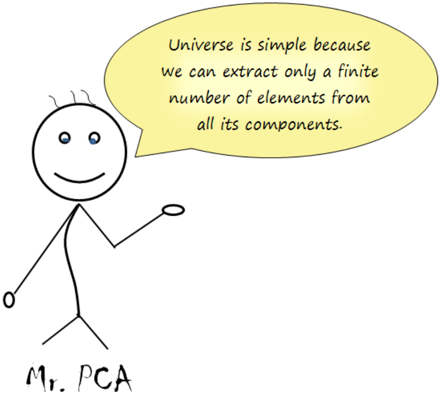
- Both Principal component analysis and Factor Analysis reveal the low dimensional structure in the data.
- But they are not same.
- PCA reveals low dimensional structures derived from the original variables.
- Factor Analysis looks back at history: tells us the story how the data was generated.

PCA vs Factor Analysis

- Question: How complex is the Universe?

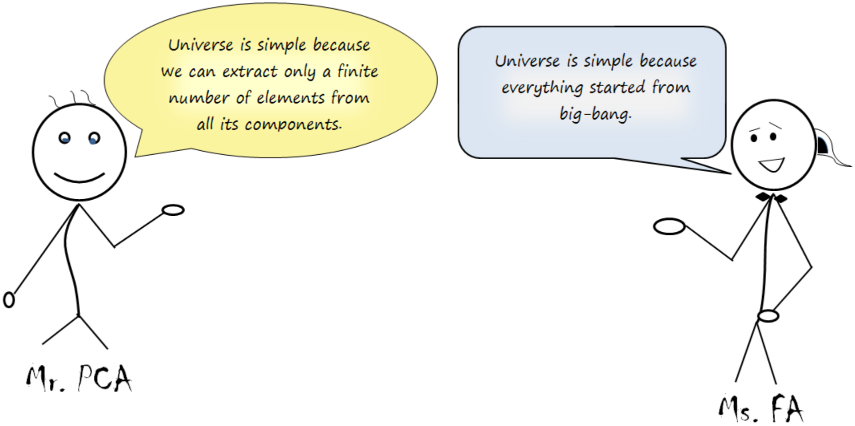
PCA vs Factor Analysis

- Question: How complex is the Universe?


Universe is simple because we can extract only a finite number of elements from all its components.
Mr. PCA

PCA vs Factor Analysis

- Question: How complex is the Universe?


Universe is simple because we can extract only a finite number of elements from all its components.
Universe is simple because everything started from big-bang.
Mr. PCA
Ms. FA

Taking Linear Projections

- We shall focus only on **linear dimensionality reduction**, in which, given high-dimensional data points $x_t \in \mathbb{R}^d$, we aim to derive lower dimensional "useful" representations $y_t \in \mathbb{R}^k$ as $y_t = Lx_t$ where $L = \begin{pmatrix} \ell_1^T \\ \vdots \\ \ell_k^T \end{pmatrix}$ is a $k \times d$ matrix.

Taking Linear Projections

- We shall focus only on **linear dimensionality reduction**, in which, given high-dimensional data points $x_t \in \mathbb{R}^d$, we aim to derive lower dimensional "useful" representations $y_t \in \mathbb{R}^k$ as

$$y_t = Lx_t \text{ where } L = \begin{pmatrix} \ell_1^T \\ \vdots \\ \ell_k^T \end{pmatrix} \text{ is a } k \times d \text{ matrix.}$$

- But how do we construct such a matrix L?

Taking Linear Projections

- We shall focus only on **linear dimensionality reduction**, in which, given high-dimensional data points $x_t \in \mathbb{R}^d$, we aim to derive lower dimensional “useful” representations $y_t \in \mathbb{R}^k$ as

$$y_t = Lx_t \text{ where } L = \begin{pmatrix} \ell_1^T \\ \vdots \\ \ell_k^T \end{pmatrix} \text{ is a } k \times d \text{ matrix.}$$

- But how do we construct such a matrix L?
- As we shall see, PCA requires that we should choose the rows ℓ_i^T of L as mutually orthonormal rows.

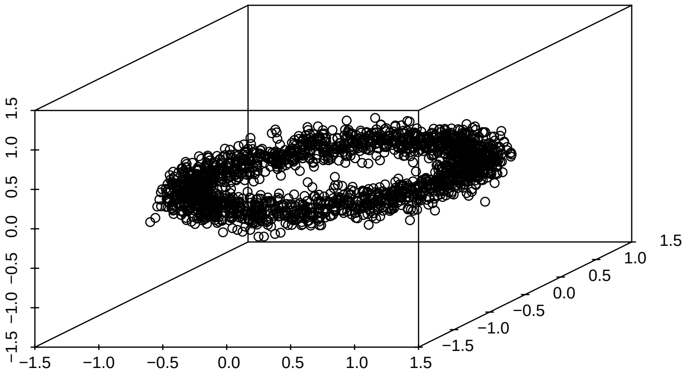
Taking Linear Projections

- We shall focus only on **linear dimensionality reduction**, in which, given high-dimensional data points $x_t \in \mathbb{R}^d$, we aim to derive lower dimensional "useful" representations $y_t \in \mathbb{R}^k$ as

$$y_t = Lx_t \text{ where } L = \begin{pmatrix} \ell_1^T \\ \vdots \\ \ell_k^T \end{pmatrix} \text{ is a } k \times d \text{ matrix.}$$

- But how do we construct such a matrix L?
- As we shall see, PCA requires that we should choose the rows ℓ_i^T of L as mutually orthonormal rows.
- But will any L with mutually orthonormal rows work?

Example: Donut Dataset



Example (Continued)

- This is an artificial data set where we have 200 observations on 3 variables x, y and z.

Example (Continued)

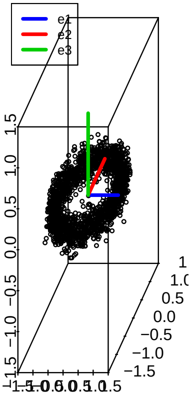
- This is an artificial data set where we have 200 observations on 3 variables x, y and z.
- The variable z is sampled from $N(0, 0.01)$ distribution and the variables x and y are generated as follows:

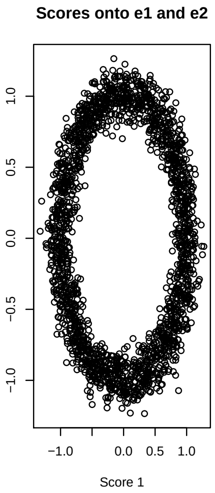
$$x = \cos(\theta) + \epsilon_1$$

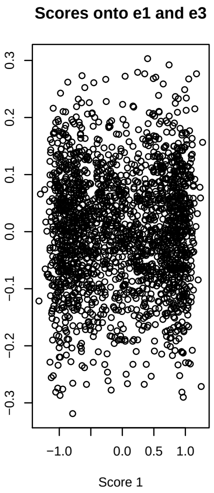
$$y = \sin(\theta) + \epsilon_2$$

where ϵ_1 and ϵ_2 are iid $N(0, 0.01)$ variables and θ is a constant which takes $n = 200$ equidistant values θ_i's between 0 and 2π for the generation of the data (x_i, y_i).

Projection onto orthonormal vectors







What do we get?

- The figure shows that not all linear projections are equal!

What do we get?

- The figure shows that not all linear projections are equal!
- So a natural question is: What makes a good one?

What do we get?

- The figure shows that not all linear projections are equal!
- So a natural question is: What makes a good one?
- The answer to this question is principal component analysis.

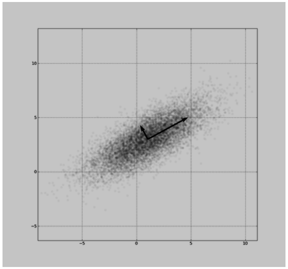
What do we get?

- The figure shows that not all linear projections are equal!
- So a natural question is: What makes a good one?
- The answer to this question is principal component analysis.
- In PCA , we just not only take linear projection onto orthonormal directions but also the variance along those directions is maximum.

- PCA finds an **orthonormal** basis (just a set of mutually perpendicular Cartesian axes, with origin at the centre of the data cloud).

- PCA finds an **orthonormal** basis (just a set of mutually perpendicular Cartesian axes, with origin at the centre of the data cloud).
- We may think of PCA as choosing a new coordinate system for the data.

- PCA finds an **orthonormal** basis (just a set of mutually perpendicular Cartesian axes, with origin at the centre of the data cloud).
- We may think of PCA as choosing a new coordinate system for the data.
- The coordinate system has the additional feature that the spread of the data cloud along the first direction is the maximum, that along the second direction is the second maximum and so on.



Data Setup

We are given a data matrix X , having n observations (row vectors) and d features (column vectors).
The n observations are $x_1, x_2, ..., x_n$ all belonging to $\mathbb{R}^d$
That is

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix}$$

Data Setup

Let $\bar{x}$ be the mean vector and S be the dispersion matrix based on the data matrix X

Then $\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k = \frac{1}{n} X^T \mathbf{1}$ and

$$S = \frac{1}{n}\sum_{k=1}^{n}(x_k - \bar{x})(x_k - \bar{x})^T = \frac{1}{n}(X - \mathbf{1}\bar{x}^T)^T(X - \mathbf{1}\bar{x}^T)$$

Data Setup

Then we define the centered data matrix as $Y = X - \mathbf{1}\bar{x}^T$
This centered matrix has mean $\bar{y} = 0$ and dispersion matrix
$S_Y = \frac{1}{n} Y^T Y = \frac{1}{n}(X - \mathbf{1}\bar{x}^T)^T(X - \mathbf{1}\bar{x}^T) = S$

We shall work with this centered matrix $Y = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{pmatrix}$

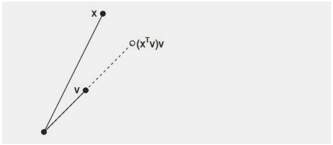
(Because it simplifies calculations)

Review: projections onto unit vectors

- The projection of $\mathbf{x} \in \mathbb{R}^d$ onto (the direction of) $\mathbf{v}$ is $(\mathbf{x}^T\mathbf{v})\mathbf{v}$.

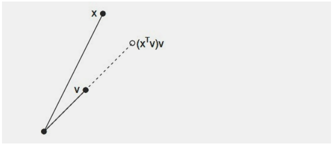
Review: projections onto unit vectors

- The projection of $\mathbf{x} \in \mathbb{R}^d$ onto (the direction of) $\mathbf{v}$ is $(\mathbf{x}^T\mathbf{v})\mathbf{v}$.
- This can be viewed as $c \times \mathbf{v}$, where c is a coefficient (or score along the direction of $\mathbf{v}$) $= \mathbf{x}^T\mathbf{v}$.



Review: projections onto unit vectors

- The projection of $\mathbf{x} \in \mathbb{R}^d$ onto (the direction of) $\mathbf{v}$ is $(\mathbf{x}^T\mathbf{v})\mathbf{v}$.
- This can be viewed as $c \times \mathbf{v}$, where c is a coefficient (or score along the direction of $\mathbf{v}$) $= \mathbf{x}^T\mathbf{v}$.



- Consider a matrix $X_{n\times d}$ and consider projecting each row $\mathbf{x}_i^T \in \mathbb{R}^d$ onto $\mathbf{v}$.
- The entries of $X\mathbf{v} = \begin{bmatrix} \mathbf{x}_1^T\mathbf{v} \\ \mathbf{x}_2^T\mathbf{v} \\ \vdots \\ \mathbf{x}_n^T\mathbf{v} \end{bmatrix} \in \mathbb{R}^n$ are the scores, and the rows of $X\mathbf{v}\mathbf{v}^T$ are the projected vectors.

Principal Components

- The first principal component direction of Y is the **unit vector** $\mathbf{v}_1 \in \mathbb{R}^d$ that maximizes the variance of $Y\mathbf{v}_1$ when compared to all other unit vectors.
- Now variance of $Y\mathbf{v}$ is

$$\frac{1}{n}\sum_{k=1}^{n}\left(y_k^T\mathbf{v}\right)^2$$

Principal Components

- The first principal component direction of Y is the **unit vector** $\mathbf{v}_1 \in \mathbb{R}^d$ that maximizes the variance of $Y\mathbf{v}_1$ when compared to all other unit vectors.
- Now variance of $Y\mathbf{v}$ is

$$\frac{1}{n}\sum_{k=1}^{n}\left(y_k^T\mathbf{v}\right)^2$$

$$= \frac{1}{n}\sum_{k=1}^{n}\mathbf{v}^T(y_k y_k^T)\mathbf{v}$$

Principal Components

- The first principal component direction of Y is the **unit vector** $\mathbf{v}_1 \in \mathbb{R}^d$ that maximizes the variance of $Y\mathbf{v}_1$ when compared to all other unit vectors.
- Now variance of $Y\mathbf{v}$ is

$$\frac{1}{n}\sum_{k=1}^{n}\left(y_k^T\mathbf{v}\right)^2$$

$$= \frac{1}{n}\sum_{k=1}^{n}\mathbf{v}^T(y_k y_k^T)\mathbf{v}$$

$$= \mathbf{v}^T\frac{1}{n}Y^TY\mathbf{v}$$

Principal Components

- The first principal component direction of Y is the **unit vector** $\mathbf{v}_1 \in \mathbb{R}^d$ that maximizes the variance of $Y\mathbf{v}_1$ when compared to all other unit vectors.
- Now variance of $Y\mathbf{v}$ is

$$\frac{1}{n}\sum_{k=1}^{n}\left(y_k^T\mathbf{v}\right)^2$$

$$=\frac{1}{n}\sum_{k=1}^{n}\mathbf{v}^T(y_k y_k^T)\mathbf{v}$$

$$=\mathbf{v}^T\frac{1}{n}Y^TY\mathbf{v}$$

$$=\mathbf{v}^TS\mathbf{v}$$

Principal Components

- Thus

$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||_{l_2}=1}{argmax}\, \mathbf{v}^T S \mathbf{v}$$

Principal Components

- Thus
$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||_{l_2}=1}{argmax}\, \mathbf{v}^T S \mathbf{v}$$
- It can be shown that $\mathbf{v}_1$ is the eigenvector corresponding to the **largest** eigenvalue λ_1 of S.

Principal Components

- Thus
$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||_{l_2}=1}{argmax}\, \mathbf{v}^T S \mathbf{v}$$
- It can be shown that $\mathbf{v}_1$ is the eigenvector corresponding to the **largest** eigenvalue λ_1 of S.
- Given the first principal component direction $\mathbf{v}_1$, we define the second principal component direction $\mathbf{v}_2$ to be the unit vector, **orthogonal** to $\mathbf{v}_1$ (i.e. $\mathbf{v}_2^T \mathbf{v}_1 = 0$), such that $X\mathbf{v}_2$ have maximal sample variance over all unit vectors orthogonal to $\mathbf{v}_1$.

Principal Components (Contd.)

- Thus

$$\mathbf{v}_2 = \underset{\mathbf{v}:||\mathbf{v}||_{l_2}=1, \mathbf{v}^T\mathbf{v}_1=0}{argmax} \mathbf{v}^T S \mathbf{v}$$

Principal Components (Contd.)

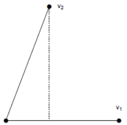
- Thus
$$\mathbf{v}_2 = \underset{\mathbf{v}:||\mathbf{v}||_{l_2}=1, \mathbf{v}^T\mathbf{v}_1=0}{argmax} \mathbf{v}^T S \mathbf{v}$$
- It can be shown (will show) that $\mathbf{v}_2$ is the eigenvector corresponding to the **second largest** eigenvalue λ_2 of S

Principal Components (Contd.)

- Thus
$$\mathbf{v}_2 = \underset{\mathbf{v}:||\mathbf{v}||_{l_2}=1, \mathbf{v}^T\mathbf{v}_1=0}{argmax} \mathbf{v}^T S \mathbf{v}$$
- It can be shown (will show) that $\mathbf{v}_2$ is the eigenvector corresponding to the **second largest** eigenvalue λ_2 of S
- Proceeding in this way, the k^{th} principal component $\mathbf{v}_k$ is the eigenvector corresponding to the k^{th} **largest** eigenvalue λ_k of S

Why Orthogonal ?

- Because we've already explained the variance in Y along $\mathbf{v}_1$,
- Now we want to look at variance in a different direction.
- Any direction not orthogonal to $\mathbf{v}_1$ would necessarily have some overlap with $\mathbf{v}_1$, i.e., it would create some redundancy in explaining the variance in Y



Also, it makes the math easier !!!

Scores

- The vector $Y\mathbf{v}_1 \in \mathbb{R}^n$ is called the first principal component score of Y.

Scores

- The vector $Y\mathbf{v}_1 \in \mathbb{R}^n$ is called the first principal component score of Y.
- The vector $Y\mathbf{v}_2 \in \mathbb{R}^n$ is called the second principal component score of Y and so on.

Scores

- The vector $Y\mathbf{v}_1 \in \mathbb{R}^n$ is called the first principal component score of Y.
- The vector $Y\mathbf{v}_2 \in \mathbb{R}^n$ is called the second principal component score of Y and so on.
- These are the projections of the data points along the principal components.

Scores

- The vector $Y\mathbf{v}_1 \in \mathbb{R}^n$ is called the first principal component score of Y.
- The vector $Y\mathbf{v}_2 \in \mathbb{R}^n$ is called the second principal component score of Y and so on.
- These are the projections of the data points along the principal components.
- We have already mentioned that the principal components may be viewed as a new reference frame.

Scores

- The vector $Y\mathbf{v}_1 \in \mathbb{R}^n$ is called the first principal component score of Y.
- The vector $Y\mathbf{v}_2 \in \mathbb{R}^n$ is called the second principal component score of Y and so on.
- These are the projections of the data points along the principal components.
- We have already mentioned that the principal components may be viewed as a new reference frame.
- Then the scores are the coordinates of the points w.r.t. this frame.

Variance Explained

- The amount of variance explained by the first principal component is $\frac{1}{n}(Y\mathbf{v}_1)^T(Y\mathbf{v}_1)$

Variance Explained

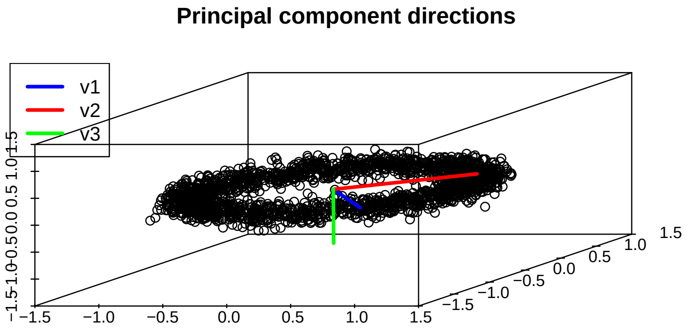
- The amount of variance explained by the first principal component is $\frac{1}{n}(Y\mathbf{v}_1)^T(Y\mathbf{v}_1)$
- We can show that this variance is equal to the **largest** eigenvalue λ_1 of S.

Variance Explained

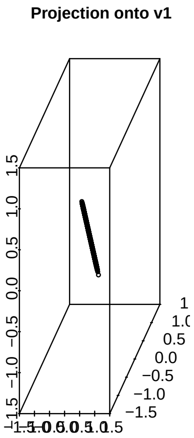
- The amount of variance explained by the first principal component is $\frac{1}{n}(Y\mathbf{v}_1)^T(Y\mathbf{v}_1)$
- We can show that this variance is equal to the **largest** eigenvalue λ_1 of S.
- Proceeding in this way we get that amount of variance explained by k^{th} principal component is $\frac{1}{n}(Y\mathbf{v}_k)^T(Y\mathbf{v}_k)$, which is the k^{th} largest eigenvalue λ_k of S.

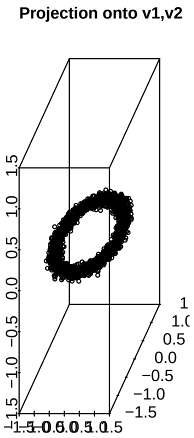
Donut Example revisited

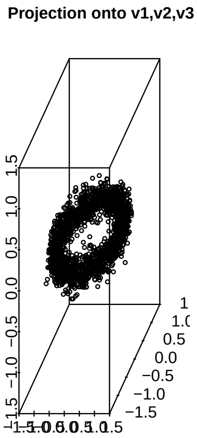
- Let us plot the principal component directions in the data cloud. Here v_1, v_2, v_3 are the unit vectors or in other words the principal component directions.



Projections of the data







What do we get?

- The first plot is a very crude approximation of the original data and cannot capture the original shape in the cloud.

What do we get?

- The first plot is a very crude approximation of the original data and cannot capture the original shape in the cloud.
- Further the second plot approximates the original data cloud to a very good extent.

What do we get?

- The first plot is a very crude approximation of the original data and cannot capture the original shape in the cloud.
- Further the second plot approximates the original data cloud to a very good extent.
- Hence we can easily approximate the original data by the projection onto v_1, v_2.

What do we get?

- The first plot is a very crude approximation of the original data and cannot capture the original shape in the cloud.
- Further the second plot approximates the original data cloud to a very good extent.
- Hence we can easily approximate the original data by the projection onto v_1, v_2.
- The third plot looks exactly the same as the original data. This is not a coincidence.

What do we get?

- The first plot is a very crude approximation of the original data and cannot capture the original shape in the cloud.
- Further the second plot approximates the original data cloud to a very good extent.
- Hence we can easily approximate the original data by the projection onto v_1, v_2.
- The third plot looks exactly the same as the original data. This is not a coincidence.
- If we choose all the principal component directions, then there is no loss of information and no reduction in dimension.

What do we get?

- The first plot is a very crude approximation of the original data and cannot capture the original shape in the cloud.
- Further the second plot approximates the original data cloud to a very good extent.
- Hence we can easily approximate the original data by the projection onto v_1, v_2.
- The third plot looks exactly the same as the original data. This is not a coincidence.
- If we choose all the principal component directions, then there is no loss of information and no reduction in dimension.
- The resultant projections are just a rotation of the original dataframe.

Dimension Reduction again

- Suppose that the data really are k-dimensional.

Dimension Reduction again

- Suppose that the data really are k-dimensional.
- Then S will have only k positive eigenvalues, and $d - k$ zero eigenvalues.

Dimension Reduction again

- Suppose that the data really are k-dimensional.
- Then S will have only k positive eigenvalues, and $d-k$ zero eigenvalues.
- If the data fall near a k-dimensional subspace, then $d-k$ of the eigenvalues will be nearly zero.

Dimension Reduction again

- Suppose that the data really are k-dimensional.
- Then S will have only k positive eigenvalues, and $d - k$ zero eigenvalues.
- If the data fall near a k-dimensional subspace, then $d - k$ of the eigenvalues will be nearly zero.
- In general, it may happen that the data really falls in a $m(< d)$ dimensional subspace and possibly near a $k(< m)$ dimensional subspace.

Dimension Reduction again

- Suppose that the data really are k-dimensional.
- Then S will have only k positive eigenvalues, and $d - k$ zero eigenvalues.
- If the data fall near a k-dimensional subspace, then $d - k$ of the eigenvalues will be nearly zero.
- In general, it may happen that the data really falls in a $m(< d)$ dimensional subspace and possibly near a $k(< m)$ dimensional subspace.
- In that case, we immediately discard the $d - m$ zero eigenvalues and search for further possible dimension reduction by checking whether additional $m - k$ eigenvalues can be found close to zero.

Case where $d > n$

- When $n < d$, PCA will surely be effective in reducing the dimension of the data.

Case where $d > n$

- When $n < d$, PCA will surely be effective in reducing the dimension of the data.
- Any two points define a line, and three points define a plane, etc.,

Case where $d > n$

- When $n < d$, PCA will surely be effective in reducing the dimension of the data.
- Any two points define a line, and three points define a plane, etc.,
- So if there are fewer data points than variables, it is necessarily true that they fall on a low-dimensional subspace.

Case where $d > n$

- When $n < d$, PCA will surely be effective in reducing the dimension of the data.
- Any two points define a line, and three points define a plane, etc.,
- So if there are fewer data points than variables, it is necessarily true that they fall on a low-dimensional subspace.
- This means that some of the eigenvalues of S will be exactly zero so that we can have a immediate reduction of dimension.

Case where $d > n$

- When $n < d$, PCA will surely be effective in reducing the dimension of the data.
- Any two points define a line, and three points define a plane, etc.,
- So if there are fewer data points than variables, it is necessarily true that they fall on a low-dimensional subspace.
- This means that some of the eigenvalues of S will be exactly zero so that we can have a immediate reduction of dimension.
- The point is we still need to search for additional small eigenvalues to check if we can further reduce the dimension.

How do we choose k?

- In any case, we can define the R^2 of the projection as the fraction of the original variance kept by the image vectors,

$$R^2 = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

How do we choose k?

- In any case, we can define the R^2 of the projection as the fraction of the original variance kept by the image vectors,

$$R^2 = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

- This is just as the R^2 of a linear regression is the fraction of the original variance of the dependent variable kept by the fitted values.

How do we choose k?

- In any case, we can define the R^2 of the projection as the fraction of the original variance kept by the image vectors,

$$R^2 = \frac{\displaystyle\sum_{i=1}^{k} \lambda_i}{\displaystyle\sum_{i=1}^{d} \lambda_i}$$

- This is just as the R^2 of a linear regression is the fraction of the original variance of the dependent variable kept by the fitted values.

- Usually, to get an R^2 of 1, we need to use all d principal components (unless there are some immediate variable dependency)

How do we choose k?

- In any case, we can define the R^2 of the projection as the fraction of the original variance kept by the image vectors,

$$R^2 = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

- This is just as the R^2 of a linear regression is the fraction of the original variance of the dependent variable kept by the fitted values.

- Usually, to get an R^2 of 1, we need to use all d principal components (unless there are some immediate variable dependency)

- How many principal components we should use depends on our data, and how big an R^2 we need.

How do we choose k?

- In any case, we can define the R^2 of the projection as the fraction of the original variance kept by the image vectors,

$$R^2 = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

- This is just as the R^2 of a linear regression is the fraction of the original variance of the dependent variable kept by the fitted values.
- Usually, to get an R^2 of 1, we need to use all d principal components (unless there are some immediate variable dependency)
- How many principal components we should use depends on our data, and how big an R^2 we need.
- Generally k is chosen such that R^2 is close to 1 (say more than 0.9).

Screeplots

- People sometimes like to make plots of the eigenvalues, in decreasing order.

Screeplots

- People sometimes like to make plots of the eigenvalues, in decreasing order.
- Ideally, we start with a few big eigenvalues, and then there is a clear drop-off to a remainder of small, comparatively negligible eigenvalues.

Screeplots

- People sometimes like to make plots of the eigenvalues, in decreasing order.
- Ideally, we start with a few big eigenvalues, and then there is a clear drop-off to a remainder of small, comparatively negligible eigenvalues.
- These diagrams are called scree plots.

Screeplots

- People sometimes like to make plots of the eigenvalues, in decreasing order.
- Ideally, we start with a few big eigenvalues, and then there is a clear drop-off to a remainder of small, comparatively negligible eigenvalues.
- These diagrams are called scree plots.
- Some people make similar plots, but show $1 - R^2$ or R^2 versus the number of components, rather than the individual eigenvalues.

Screeplots

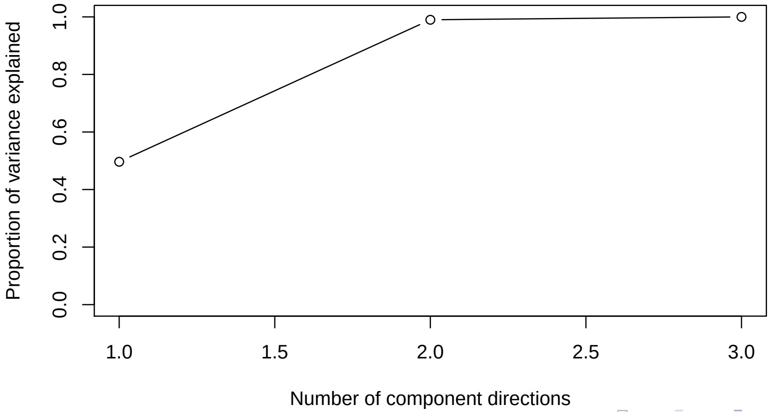
- People sometimes like to make plots of the eigenvalues, in decreasing order.
- Ideally, we start with a few big eigenvalues, and then there is a clear drop-off to a remainder of small, comparatively negligible eigenvalues.
- These diagrams are called scree plots.
- Some people make similar plots, but show $1 - R^2$ or R^2 versus the number of components, rather than the individual eigenvalues.
- Folklore recommends find the "base of the cliff" or "elbow" in the plot, the place where the number eigenvalues decrease dramatically and then level off to the right, and then retaining that number of components.

Screeplots

- People sometimes like to make plots of the eigenvalues, in decreasing order.
- Ideally, we start with a few big eigenvalues, and then there is a clear drop-off to a remainder of small, comparatively negligible eigenvalues.
- These diagrams are called scree plots.
- Some people make similar plots, but show $1 - R^2$ or R^2 versus the number of components, rather than the individual eigenvalues.
- Folklore recommends find the "base of the cliff" or "elbow" in the plot, the place where the number eigenvalues decrease dramatically and then level off to the right, and then retaining that number of components.
- But this suggestion is nothing more than intuition, and offers no recommendation for what to do when there is no clear cliff or elbow in the scree plot.

Donut Example once again

- We can plot the cumulative variance explained by the principal components.



From PCA to Factor Analysis

- Suppose our data matrix is $X_{n\times d}$.

From PCA to Factor Analysis

- Suppose our data matrix is $X_{n \times d}$.
- Since the original observations are d dimensional, we have d principal components each of which are a vector of length d.

From PCA to Factor Analysis

- Suppose our data matrix is $X_{n \times d}$.
- Since the original observations are d dimensional, we have d principal components each of which are a vector of length d.
- Let us stack these components in a matrix $L_{d \times d}$.

From PCA to Factor Analysis

- Suppose our data matrix is $X_{n \times d}$.
- Since the original observations are d dimensional, we have d principal components each of which are a vector of length d.
- Let us stack these components in a matrix $L_{d \times d}$.
- Finally, each data point has a projection on to each principal component, which we collect into an $n \times d$ matrix F.

From PCA to Factor Analysis

- Suppose our data matrix is $X_{n \times d}$.
- Since the original observations are d dimensional, we have d principal components each of which are a vector of length d.
- Let us stack these components in a matrix $L_{d \times d}$.
- Finally, each data point has a projection on to each principal component, which we collect into an $n \times d$ matrix F.
- Then we get an exact equation

$$X = FL$$

From PCA to Factor Analysis

- Suppose our data matrix is $X_{n \times d}$.
- Since the original observations are d dimensional, we have d principal components each of which are a vector of length d.
- Let us stack these components in a matrix $L_{d \times d}$.
- Finally, each data point has a projection on to each principal component, which we collect into an $n \times d$ matrix F.
- Then we get an exact equation

$$X = FL$$

- This exact equation is basically useless since this is nothing but rotation of axes in the d dimensional space.

PCA to Factor Analysis (Contd.)

- If we keep only to $k < d$ largest principal components, that corresponds to dropping columns from F and rows from W.

PCA to Factor Analysis (Contd.)

- If we keep only to $k < d$ largest principal components, that corresponds to dropping columns from F and rows from W.
- Denoting the truncated matrices as F_k and L_k we get

$$X \approx F_k L_k.$$

PCA to Factor Analysis (Contd.)

- If we keep only to $k < d$ largest principal components, that corresponds to dropping columns from F and rows from W.
- Denoting the truncated matrices as F_k and L_k we get

$$X \approx F_k L_k.$$

- This is an approximate equation and it involves errors of approximation.

PCA to Factor Analysis (Contd.)

- If we keep only to $k < d$ largest principal components, that corresponds to dropping columns from F and rows from W.
- Denoting the truncated matrices as F_k and L_k we get

$$X \approx F_k L_k.$$

- This is an approximate equation and it involves errors of approximation.
- The error of approximation will get smaller as we increase k upto d.

PCA to Factor Analysis (Contd.)

- If we keep only to $k < d$ largest principal components, that corresponds to dropping columns from F and rows from W.
- Denoting the truncated matrices as F_k and L_k we get
$$X \approx F_k L_k.$$
- This is an approximate equation and it involves errors of approximation.
- The error of approximation will get smaller as we increase k upto d.
- Recall that in PCA we were satisfied with this approximation if it is upto a reasonable extent.

PCA to Factor Analysis (Contd.)

- We can also make the two sides match exactly by adding an error or residual term on the right

$$X = FL + \epsilon$$

where ϵ is a $n \times d$ matrix and F and L now stands for F_k and L_k.

PCA to Factor Analysis (Contd.)

- We can also make the two sides match exactly by adding an error or residual term on the right

$$X = FL + \epsilon$$

  where ϵ is a $n \times d$ matrix and F and L now stands for F_k and L_k.

- Let us start treating the noise ϵ as random which has some distribution.

PCA to Factor Analysis (Contd.)

- We can also make the two sides match exactly by adding an error or residual term on the right

  $$X = FL + \epsilon$$

  where ϵ is a $n \times d$ matrix and F and L now stands for F_k and L_k.

- Let us start treating the noise ϵ as random which has some distribution.

- This is where we move from mere empirical data reduction to making assertions about the process that generated the data.

PCA to Factor Analysis (Contd.)

- We say that we have k **factors**.

PCA to Factor Analysis (Contd.)

- We say that we have k **factors**.
- F is the matrix of factor scores (rather than projections onto principal components).

PCA to Factor Analysis (Contd.)

- We say that we have k **factors**.
- F is the matrix of factor scores (rather than projections onto principal components).
- L is the matrix of factor loadings.

PCA to Factor Analysis (Contd.)

- We say that we have k **factors**.
- F is the matrix of factor scores (rather than projections onto principal components).
- L is the matrix of factor loadings.
- The variables in X are called observable or manifest variables, those in F are **hidden** or **latent**. (Technically ϵ is also latent.)

Example

- Consider a market study regarding customers of a brand of car.
- Data on **14 variables** are obtained.
- The customers were asked to rate their preferences in a scale (from 1 to 7) for each of these variables.

Variables

1. Low cost of repairs
2. Comes in a variety of colors
3. Roomy Interior
4. Good Gas Mileage
5. Good Handling
6. Modern Looking
7. High Resale Value
8. Comfortable
9. Large Engine
10. Sleek Appearance
11. Easy to drive
12. Eye Catching
13. Large Trunk Space
14. Easy to park

A close look at the variables

1. Low cost of repairs
2. Comes in a variety of colors
3. Roomy Interior
4. Good Gas Mileage
5. **Good Handling**
6. Modern Looking
7. High Resale Value
8. Comfortable
9. Large Engine
10. Sleek Appearance
11. **Easy to drive**
12. Eye Catching
13. Large Trunk Space
14. **Easy to park**

Variables 5, 11, 14 represents the customer choice about **driving**.

A close look at the variables

1. Low cost of repairs
2. **Comes in a variety of colors**
3. Roomy Interior
4. Good Gas Mileage
5. Good Handling
6. **Modern Looking**
7. High Resale Value
8. Comfortable
9. Large Engine
10. **Sleek Appearance**
11. Easy to drive
12. **Eye Catching**
13. Large Trunk Space
14. Easy to park

Variables 2,6,10,12 represents the customer choice about **looks**.

A close look at the variables

1. **Low cost of repairs**
2. Comes in a variety of colors
3. Roomy Interior
4. **Good Gas Mileage**
5. Good Handling
6. Modern Looking
7. **High Resale Value**
8. Comfortable
9. **Large Engine**
10. Sleek Appearance
11. Easy to drive
12. Eye Catching
13. Large Trunk Space
14. Easy to park

Variables 1,4,7,9 represents the customer choice about **economy**.

A close look at the variables

1. Low cost of repairs
2. Comes in a variety of colors
3. **Roomy Interior**
4. Good Gas Mileage
5. Good Handling
6. Modern Looking
7. High Resale Value
8. **Comfortable**
9. Large Engine
10. Sleek Appearance
11. Easy to drive
12. Eye Catching
13. **Large Trunk Space**
14. Easy to park

Variables 3,8,13 represents the customer choice about **comfortability**.

What do we get?

- Although we have 14 variables, many of the **variables are similar**.

What do we get?

- Although we have 14 variables, many of the **variables are similar**.
- In a nutshell the data can be described only by the variables **Ease of Driving**, **Looks**, **Comfort**, **Economy**.

What do we get?

- Although we have 14 variables, many of the **variables are similar**.
- In a nutshell the data can be described only by the variables **Ease of Driving**, **Looks**, **Comfort**, **Economy**.
- Thus instead of 14 observed variables we can only work with these **4 underlying variables**.

What do we get?

- Although we have 14 variables, many of the **variables are similar**.
- In a nutshell the data can be described only by the variables **Ease of Driving**, **Looks**, **Comfort**, **Economy**.
- Thus instead of 14 observed variables we can only work with these **4 underlying variables**.
- These underlying variables are called **factors**.

What do we get?

- Although we have 14 variables, many of the **variables are similar**.
- In a nutshell the data can be described only by the variables **Ease of Driving**, **Looks**, **Comfort**, **Economy**.
- Thus instead of 14 observed variables we can only work with these **4 underlying variables**.
- These underlying variables are called **factors**.
- Representing variables in terms of these unknown factors is called **factor analysis**.

Orthogonal Factor model

- It is to be noted that this model can also be stated in terms of the data matrix as

$$X_{n\times d} = F_{n\times k} L_{k\times d} + \epsilon_{n\times d}$$

- Before we can actually do much with this model, we need to say more about the distributions of these random variables.

Orthogonal Factor model

- It is to be noted that this model can also be stated in terms of the data matrix as

$$X_{n\times d} = F_{n\times k} L_{k\times d} + \epsilon_{n\times d}$$

- Before we can actually do much with this model, we need to say more about the distributions of these random variables.
- All of the latent factors have mean zero and variance 1
- The factors are uncorrelated.
  - That is, $E(F) = 0$ and $E(FF^T) = I_k$.

- The noise terms all have mean zero
- The noise terms are uncorrelated.
    - That is, $E(\epsilon) = 0$ and $E(\epsilon\epsilon^T) = \psi = diag(\psi_1, \psi_2, ..., \psi_d)$.

- The noise terms all have mean zero
- The noise terms are uncorrelated.
  - That is, $E(\epsilon) = 0$ and $E(\epsilon\epsilon^T) = \psi = diag(\psi_1, \psi_2, ..., \psi_d)$.
- The noise terms are uncorrelated with the factor variables, that is, $E(\epsilon F^T) = 0$.
- The factors are uncorrelated across individuals (rows of F) (in addition to across variables (columns of F)).
- The noise terms are uncorrelated across individuals (in addition to across observable variables).

What do these assumptions imply?

- These assumptions in the factor model has a strong implication regarding the covariance structure of X as

$$\Sigma = L^T L + \psi$$

What do these assumptions imply?

- These assumptions in the factor model has a strong implication regarding the covariance structure of X as

$$\Sigma = L^T L + \psi$$

- Thus the variance of X_i can be split as

$$Var(X_i) = \underbrace{l_{i1}^2 + l_{i2}^2 + ... + l_{ik}^2}_{\text{communality}} + \underbrace{\psi_i}_{\text{specific variance}}$$

What do these assumptions imply?

- These assumptions in the factor model has a strong implication regarding the covariance structure of X as

$$\Sigma = L^T L + \psi$$

- Thus the variance of X_i can be split as

$$Var(X_i) = \underbrace{l_{i1}^2 + l_{i2}^2 + ... + l_{ik}^2}_{\text{communality}} + \underbrace{\psi_i}_{\text{specific variance}}$$

- Thus the i^{th} communality is the sum of squares of loadings of i^{th} variable on all the k common factors.

PCA in R

- There are two R functions for doing PCA, princomp and prcomp, which differ in how they do the actual calculation.

PCA in R

- There are two R functions for doing PCA, princomp and prcomp, which differ in how they do the actual calculation.
- The latter is generally more robust.

PCA in R

- There are two R functions for doing PCA, princomp and prcomp, which differ in how they do the actual calculation.
- The latter is generally more robust.
- Alternatively we can also perform eigen analysis of the dispersion matrix.

Example: state.x77

- R contains a built-in data file, **state.x77**, with facts and figures for the various states of the USA as of about 1977.

Example: state.x77

- R contains a built-in data file, **state.x77**, with facts and figures for the various states of the USA as of about 1977.
- The data set contains population, per-capita income, the adult illiteracy rate, life expectancy, the homicide rate, the proportion of adults with at least a high-school education, the number of days of frost a year, and the state's area.

Example: state.x77

- R contains a built-in data file, **state.x77**, with facts and figures for the various states of the USA as of about 1977.
- The data set contains population, per-capita income, the adult illiteracy rate, life expectancy, the homicide rate, the proportion of adults with at least a high-school education, the number of days of frost a year, and the state's area.
- Since the variables all have different, incomparable scales, it's not a bad idea to scale them to unit variance before finding the components.

R code

```
state.pca <- prcomp(state.x77, scale. = TRUE)
```

- We can now extract the loadings or weight matrix from the state.pca object.
- For comprehensibility we'll just show the first two components.

Loading matrix

```
signif(state.pca$rotation[, 1:2], 2)

              PC1    PC2
Population  0.130  0.410
Income     -0.300  0.520
Illiteracy  0.470  0.053
Life Exp   -0.410 -0.082
Murder      0.440  0.310
HS Grad    -0.420  0.300
Frost      -0.360 -0.150
Area       -0.033  0.590
```

What do we get?

- The first component aligns with illiteracy, murder, and (more weakly) population; it's negatively aligned with high school graduation, life expectancy, cold weather, income, and (very weakly) the area of the state.

What do we get?

- The first component aligns with illiteracy, murder, and (more weakly) population; it's negatively aligned with high school graduation, life expectancy, cold weather, income, and (very weakly) the area of the state.
- The second component is positively aligned with area, income, population, high school graduation and murder, and negatively aligned, weakly, with cold weather and life expectancy.

What do we get?

- The first component aligns with illiteracy, murder, and (more weakly) population; it's negatively aligned with high school graduation, life expectancy, cold weather, income, and (very weakly) the area of the state.
- The second component is positively aligned with area, income, population, high school graduation and murder, and negatively aligned, weakly, with cold weather and life expectancy.
- The first component thus separates short-lived, violent, ill-educated, poor warm states from those with the opposite qualities.

What do we get?

- The first component aligns with illiteracy, murder, and (more weakly) population; it's negatively aligned with high school graduation, life expectancy, cold weather, income, and (very weakly) the area of the state.
- The second component is positively aligned with area, income, population, high school graduation and murder, and negatively aligned, weakly, with cold weather and life expectancy.
- The first component thus separates short-lived, violent, ill-educated, poor warm states from those with the opposite qualities.
- The second component separates big, rich, educated, violent states from those which are small (in land or people), poor, less educated, and less violent.

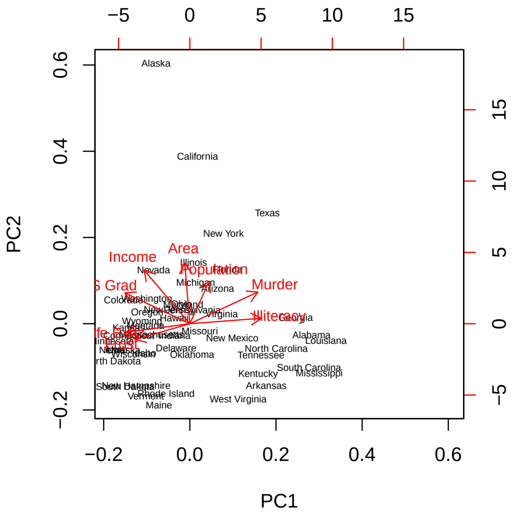
Biplot I

- To check this interpretation, we can use a useful tool called a biplot, which plots the data, along with the projections of the original variables, on to the first two components.

Biplot II

```
biplot(state.pca, cex = c(0.5, 0.75))
```

Biplot III


PC1
PC2
Alaska
California
Texas
New York
Income
Area
Population
Murder
Illiteracy
S Grad
Nevada
Illinois
Florida
Michigan
Arizona
Colorado
Washington
Maryland
Oregon
New Jersey
Pennsylvania
Virginia
Hawaii
Georgia
Wyoming
Kansas
Montana
Missouri
Connecticut
Massachusetts
Indiana
Minnesota
New Mexico
Alabama
Louisiana
Nebraska
Idaho
Wisconsin
Utah
Iowa
Delaware
Oklahoma
North Carolina
Tennessee
North Dakota
South Carolina
Mississippi
Kentucky
Arkansas
New Hampshire
South Dakota
Vermont
Rhode Island
Maine
West Virginia

Interpretation

- Since each data point has a geographic location, we can make a map, where the sizes of the symbols for each state vary with their projection on to the first principal component.

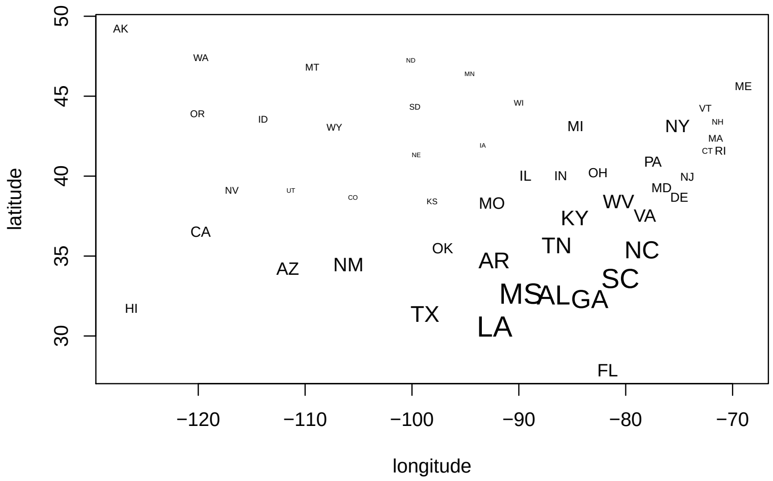
Interpretation

- Since each data point has a geographic location, we can make a map, where the sizes of the symbols for each state vary with their projection on to the first principal component.
- This suggests that the component is something we might call "southernness" — more precisely, the contrast between the South and the rest of the nation.

US states plotted in their geographic locations I

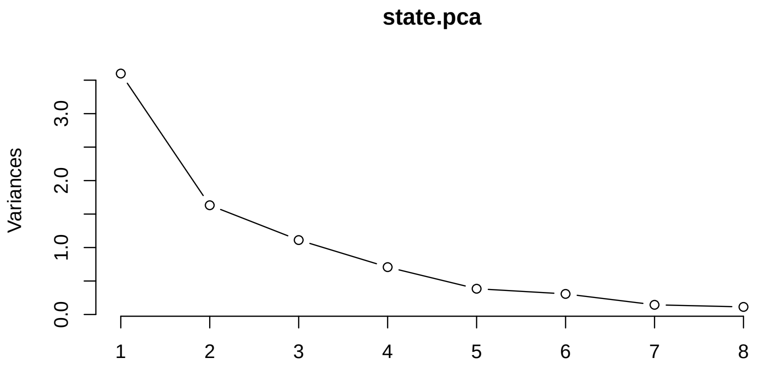
```
plot.states<-function(sizes,min.size=0.4,max.size=2,...)
{
        plot(state.center,type="n",...)
        out.range=max.size-min.size
        in.range=max(sizes)-min(sizes)
        scaled.sizes=out.range*((sizes-min(sizes))/in.range)
        text(state.center,state.abb, cex = scaled.sizes + min.size)
        invisible(scaled.sizes)
}
plot.states(state.pca$x[, 1], min.size = 0.3,
max.size=1.5,xlab="longitude",ylab="latitude")
```

US states plotted in their geographic locations II



Screeplot

```
plot(state.pca, type = "l")
```



Factor Analysis in R

- R has an inbuilt function **factanal()** which performs factor analysis by maximum likelihood method.
- We look back at the data set containing the properties of the US states around 1977.
- Let us subject this dataset to factor analysis and see what latent structures we can discover.

Let's begin in R

We begin with one factor, using the base R function **factanal**.

```
state.fa1 <- factanal(state.x77,factors=1,scores="regression")
state.fa1



Call:
factanal(x = state.x77, factors = 1, scores = "regression")

Uniquenesses:
Population     Income Illiteracy   Life Exp     Murder    HS Grad
     0.957      0.791      0.235      0.437      0.308      0.496
     Frost       Area
     0.600      0.998

Loadings:
           Factor1
Population -0.208
Income      0.458
Illiteracy -0.875
Life Exp    0.750
Murder     -0.832
HS Grad     0.710
Frost       0.632
Area

               Factor1
SS loadings      3.178
Proportion Var   0.397

Test of the hypothesis that 1 factor is sufficient.
```

What do we get?

- The uniqueness in the output here tells us what fraction of the variance in each observable comes from its own noise which is nothing but the diagonal entries in $\hat{\psi}$.

What do we get?

- The uniqueness in the output here tells us what fraction of the variance in each observable comes from its own noise which is nothing but the diagonal entries in $\hat{\psi}$.
- It also gives us the factor loadings, i.e., the rows of $\hat{L}$.

What do we get?

- The uniqueness in the output here tells us what fraction of the variance in each observable comes from its own noise which is nothing but the diagonal entries in $\hat{\psi}$.
- It also gives us the factor loadings, i.e., the rows of $\hat{L}$.
- Note that currently there's only one loading vector, since we set the numebr of factors (k) to be 1 by the argument factors= 1.

What do we get?

- The uniqueness in the output here tells us what fraction of the variance in each observable comes from its own noise which is nothing but the diagonal entries in $\hat{\psi}$.
- It also gives us the factor loadings, i.e., the rows of $\hat{L}$.
- Note that currently there's only one loading vector, since we set the numebr of factors (k) to be 1 by the argument factors= 1.
- As a courtesy, the default printing method for the loadings leaves blanks where the loadings would be very small (here, for Area).

What do we get?

- The uniqueness in the output here tells us what fraction of the variance in each observable comes from its own noise which is nothing but the diagonal entries in $\hat{\psi}$.
- It also gives us the factor loadings, i.e., the rows of $\hat{L}$.
- Note that currently there's only one loading vector, since we set the numebr of factors (k) to be 1 by the argument factors= 1.
- As a courtesy, the default printing method for the loadings leaves blanks where the loadings would be very small (here, for Area).
- Finally the last option picks between different methods of estimating the factor scores.

More than one factor

- Of course, why use just one factor?

More than one factor

- Of course, why use just one factor?
- Now given the number of observables, we can fit up to four factors before the problem becomes totally unidentified and factanal refuses to work.

More than one factor

- Of course, why use just one factor?
- Now given the number of observables, we can fit up to four factors before the problem becomes totally unidentified and factanal refuses to work.
- That function factanal automatically runs the likelihood ratio test every time it fits a model, assuming Gaussian distributions for the observables.

More than one factor

- Of course, why use just one factor?
- Now given the number of observables, we can fit up to four factors before the problem becomes totally unidentified and factanal refuses to work.
- That function factanal automatically runs the likelihood ratio test every time it fits a model, assuming Gaussian distributions for the observables.
- This can work reasonably well for non-Gaussian distributions if they're not too non- Gaussian, especially if n is much larger than the number of parameters

More than one factor

- Of course, why use just one factor?
- Now given the number of observables, we can fit up to four factors before the problem becomes totally unidentified and factanal refuses to work.
- That function factanal automatically runs the likelihood ratio test every time it fits a model, assuming Gaussian distributions for the observables.
- This can work reasonably well for non-Gaussian distributions if they're not too non- Gaussian, especially if n is much larger than the number of parameters
- Of course $n = 50$ is pretty modest. Still, we shall try it.

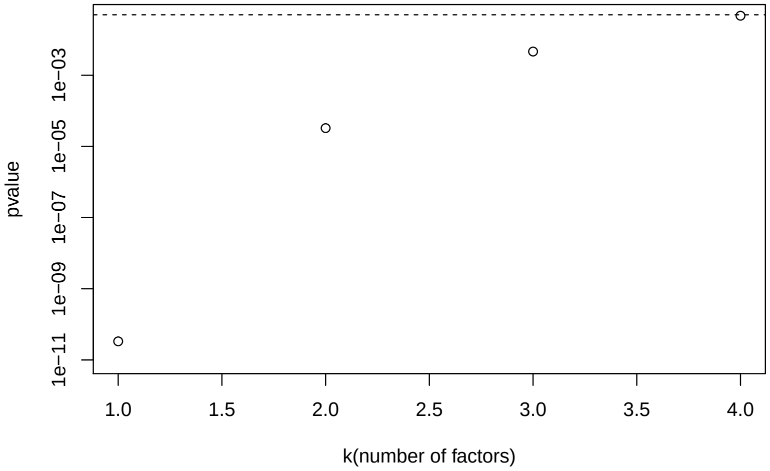
Which model is best? I

```
pvalues<-sapply(1:4,function(q){factanal(state.x77,factors=q)$PVAL})
signif(pvalues,2)

objective objective objective objective
  3.3e-11   3.3e-05   4.6e-03   4.7e-02

plot(1:4,pvalues,xlab="k(number of factors)", ylab="pvalue",
log="y",ylim=c(1e-11,0.04))
abline(h=0.05,lty="dashed")
```

Which model is best? II



What do we get?

- None of the models has a p-value crossing the conventional 0.05 level.

What do we get?

- None of the models has a p-value crossing the conventional 0.05 level.
- This means all of them show systematic, detectable departures from what the data should look like if the factor model were true.

What do we get?

- None of the models has a p-value crossing the conventional 0.05 level.
- This means all of them show systematic, detectable departures from what the data should look like if the factor model were true.
- Still, the four factor model comes close.

What do we get?

- None of the models has a p-value crossing the conventional 0.05 level.
- This means all of them show systematic, detectable departures from what the data should look like if the factor model were true.
- Still, the four factor model comes close.
- In other words, given the data, if we need to explain the variables in terms of few underlying factors, the four factor model does the best.

Loading matrix

- Notice that the first factor's loadings do not stay the same when we add more factors, unlike the first principal component.

```
print(factanal(state.x77, factors=4)$loadings)



Loadings:
           Factor1 Factor2 Factor3 Factor4
Population                          0.636
Income      0.313   0.281   0.561   0.189
Illiteracy -0.466  -0.878
Life Exp    0.891   0.191
Murder     -0.792  -0.384   0.109   0.405
HS Grad     0.517   0.418   0.581
Frost       0.128   0.679   0.105  -0.460
Area       -0.174           0.796

               Factor1 Factor2 Factor3 Factor4
SS loadings      2.054   1.680   1.321   0.821
Proportion Var   0.257   0.210   0.165   0.103
Cumulative Var   0.257   0.467   0.632   0.734
```

Principal Component Analysis

Mathematical Proof

Principal Component Analysis

The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few *retain most of the variation present in* all *of the original variables.*
[Jolliffe, Pricipal Component Analysis, 2*nd* *edition]*

Data distribution (inputs in regression analysis)

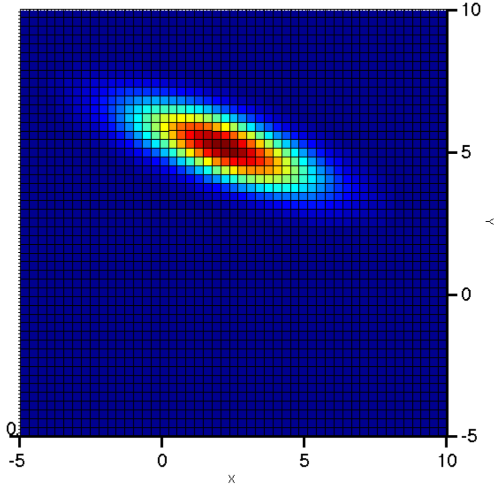


Figure: Gaussian PDF

Uncorrelated projections of principal variation

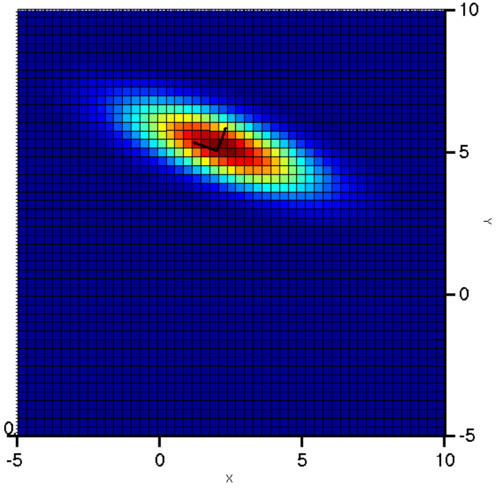


Figure: Gaussian PDF with PC eigenvectors

PCA rotation

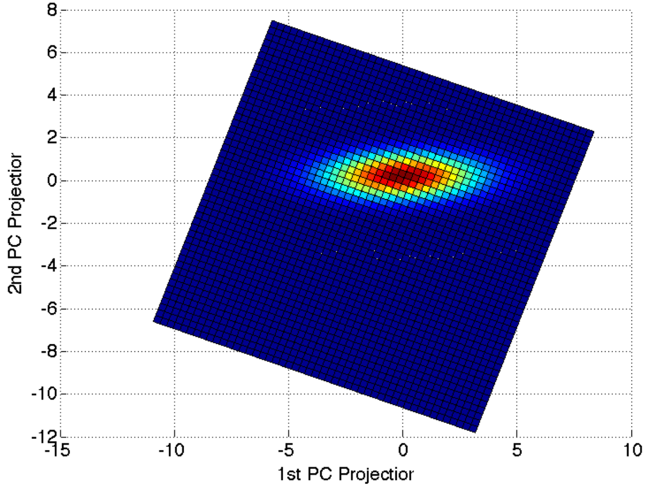


Figure: PCA Projected Gaussian PDF

PCA in a nutshell

Notation

- $\mathbf{x}$ is a vector of p random variables
- $\boldsymbol{\alpha}_k$ is a vector of p constants
- $\boldsymbol{\alpha}_k'\mathbf{x} = \sum_{j=1}^{p} \alpha_{kj} x_j$

Procedural description

- Find linear function of $\mathbf{x}$, $\boldsymbol{\alpha}_1'\mathbf{x}$ with maximum variance.
- Next find another linear function of $\mathbf{x}$, $\boldsymbol{\alpha}_2'\mathbf{x}$, uncorrelated with $\boldsymbol{\alpha}_1'\mathbf{x}$ maximum variance.
- Iterate.

Goal

It is hoped, in general, that most of the variation in $\mathbf{x}$ will be accounted for by m PC's where $m << p$.

Derivation of PCA

Assumption and More Notation

- $\mathbf{\Sigma}$ is the *known* covariance matrix for the random variable $\mathbf{x}$
- Foreshadowing : $\mathbf{\Sigma}$ will be replaced with $\mathbf{S}$, the sample covariance matrix, when $\mathbf{\Sigma}$ is unknown.

Shortcut to solution

- For $k = 1, 2, \ldots, p$ the k^{th} PC is given by $z_k = \boldsymbol{\alpha}'_k \mathbf{x}$ where $\boldsymbol{\alpha}_k$ is an eigenvector of $\mathbf{\Sigma}$ corresponding to its k^{th} largest eigenvalue λ_k.
- If $\boldsymbol{\alpha}_k$ is chosen to have unit length (i.e. $\boldsymbol{\alpha}'_k \boldsymbol{\alpha}_k = 1$) then $\text{Var}(z_k) = \lambda_k$

Derivation of PCA

First Step

- Find $\boldsymbol{\alpha}_k'\mathbf{x}$ that maximizes $\text{Var}(\boldsymbol{\alpha}_k'\mathbf{x}) = \boldsymbol{\alpha}_k'\boldsymbol{\Sigma}\boldsymbol{\alpha}_k$
- Without constraint we could pick a very big $\boldsymbol{\alpha}_k$.
- Choose normalization constraint, namely $\boldsymbol{\alpha}_k'\boldsymbol{\alpha}_k = 1$ (unit length vector).

Constrained maximization - method of Lagrange multipliers

- To maximize $\boldsymbol{\alpha}_k'\boldsymbol{\Sigma}\boldsymbol{\alpha}_k$ subject to $\boldsymbol{\alpha}_k'\boldsymbol{\alpha}_k = 1$ we use the technique of Lagrange multipliers. We maximize the function

$$\boldsymbol{\alpha}_k'\boldsymbol{\Sigma}\boldsymbol{\alpha}_k - \lambda(\boldsymbol{\alpha}_k'\boldsymbol{\alpha}_k - 1)$$

w.r.t. to $\boldsymbol{\alpha}_k$ by differentiating w.r.t. to $\boldsymbol{\alpha}_k$.

Derivation of PCA

Constrained maximization - method of Lagrange multipliers

- This results in

$$\begin{aligned}
\frac{d}{d\boldsymbol{\alpha}_k}\left(\boldsymbol{\alpha}_k'\boldsymbol{\Sigma}\boldsymbol{\alpha}_k - \lambda_k(\boldsymbol{\alpha}_k'\boldsymbol{\alpha}_k - 1)\right) &= 0 \\
\boldsymbol{\Sigma}\boldsymbol{\alpha}_k - \lambda_k\boldsymbol{\alpha}_k &= 0 \\
\boldsymbol{\Sigma}\boldsymbol{\alpha}_k &= \lambda_k\boldsymbol{\alpha}_k
\end{aligned}$$

- This should be recognizable as an eigenvector equation where $\boldsymbol{\alpha}_k$ is an eigenvector of $\Sigma_b f$ and λ_k is the associated eigenvalue.
- Which eigenvector should we choose?

Derivation of PCA

Constrained maximization - method of Lagrange multipliers

- If we recognize that the quantity to be maximized

$$\boldsymbol{\alpha}_k'\boldsymbol{\Sigma}\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_k'\lambda_k\boldsymbol{\alpha}_k = \lambda_k\boldsymbol{\alpha}_k'\boldsymbol{\alpha}_k = \lambda_k$$

  then we should choose λ_k to be as big as possible. So, calling λ_1 the largest eigenvector of Σ and $\boldsymbol{\alpha}_1$ the corresponding eigenvector then the solution to

$$\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 = \lambda_1\boldsymbol{\alpha}_1$$

  is the 1^{st} principal component of $\mathbf{x}$.
- In general $\boldsymbol{\alpha}_k$ will be the k^{th} PC of $\mathbf{x}$ and $\text{Var}(\boldsymbol{\alpha}'\mathbf{x}) = \lambda_k$
- We will demonstrate this for $k = 2$, $k > 2$ is more involved but similar.

Derivation of PCA

Constrained maximization - more constraints

- The second PC, $\boldsymbol{\alpha}_2\mathbf{x}$ maximizes $\boldsymbol{\alpha}_2\Sigma\boldsymbol{\alpha}_2$ subject to being uncorrelated with $\boldsymbol{\alpha}_1\mathbf{x}$.
- The uncorrelation constraint can be expressed using any of these equations

$$\begin{aligned} \operatorname{cov}(\boldsymbol{\alpha}_1'\mathbf{x}, \boldsymbol{\alpha}_2'\mathbf{x}) &= \boldsymbol{\alpha}_1'\boldsymbol{\Sigma}\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}_2'\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_2'\lambda_1\boldsymbol{\alpha}_1' \\ &= \lambda_1\boldsymbol{\alpha}_2'\boldsymbol{\alpha} = \lambda_1\boldsymbol{\alpha}_1'\boldsymbol{\alpha}_2 = 0 \end{aligned}$$

- Of these, if we choose the last we can write an Langrangian to maximize $\boldsymbol{\alpha}_2$

$$\boldsymbol{\alpha}_2'\boldsymbol{\Sigma}\boldsymbol{\alpha}_2 - \lambda_2(\boldsymbol{\alpha}_2'\boldsymbol{\alpha}_2 - 1) - \phi\boldsymbol{\alpha}_2'\boldsymbol{\alpha}_1$$

Derivation of PCA

Constrained maximization - more constraints

- Differentiation of this quantity w.r.t. $\boldsymbol{\alpha}_2$ (and setting the result equal to zero) yields

$$\begin{aligned} \frac{d}{d\boldsymbol{\alpha}_2}\left(\boldsymbol{\alpha}_2'\boldsymbol{\Sigma}\boldsymbol{\alpha}_2 - \lambda_2(\boldsymbol{\alpha}_2'\boldsymbol{\alpha}_2 - 1) - \phi\boldsymbol{\alpha}_2'\boldsymbol{\alpha}_1\right) &= 0 \\ \boldsymbol{\Sigma}\boldsymbol{\alpha}_2 - \lambda_2\boldsymbol{\alpha}_2 - \phi\boldsymbol{\alpha}_1 &= 0 \end{aligned}$$

- If we left multiply $\boldsymbol{\alpha}_1$ into this expression

$$\begin{aligned} \boldsymbol{\alpha}_1'\boldsymbol{\Sigma}\boldsymbol{\alpha}_2 - \lambda_2\boldsymbol{\alpha}_1'\boldsymbol{\alpha}_2 - \phi\boldsymbol{\alpha}_1'\boldsymbol{\alpha}_1 &= 0 \\ 0 - 0 - \phi 1 &= 0 \end{aligned}$$

  then we can see that ϕ must be zero and that when this is true that we are left with

$$\boldsymbol{\Sigma}\boldsymbol{\alpha}_2 - \lambda_2\boldsymbol{\alpha}_2 = 0$$

Derivation of PCA

Clearly

$$\boldsymbol{\Sigma}\boldsymbol{\alpha}_2 - \lambda_2\boldsymbol{\alpha}_2 = 0$$

is another eigenvalue equation and the same strategy of choosing α_2 to be the eigenvector associated with the second largest eigenvalue yields the second PC of $\mathbf{x}$, namely $\boldsymbol{\alpha}_2'\mathbf{x}$.

This process can be repeated for $k = 1 \dots p$ yielding up to p different eigenvectors of $\boldsymbol{\Sigma}$ along with the corresponding eigenvalues $\lambda_1, \dots \lambda_p$.

Furthermore, the variance of each of the PC's are given by

$$\text{Var}[\boldsymbol{\alpha}_k'\mathbf{x}] = \lambda_k, \qquad k = 1, 2, \dots, p$$

Properties of PCA

For any integer $q, 1 \leq q \leq p$, consider the orthonormal linear transformation

$$\mathbf{y} = \mathbf{B}'\mathbf{x}$$

where $\mathbf{y}$ is a q-element vector and $\mathbf{B}'$ is a $q \times p$ matrix, and let $\mathbf{\Sigma}_y = \mathbf{B}'\mathbf{\Sigma}\mathbf{B}$ be the variance-covariance matrix for $\mathbf{y}$. Then the trace of $\mathbf{\Sigma}_y$, denoted $\mathrm{tr}(\mathbf{\Sigma}_y)$, is maximized by taking $\mathbf{B} = \mathbf{A}_q$, where $\mathbf{A}_q$ consists of the first q columns of $\mathbf{A}$.

What this means is that if you want to choose a lower dimensional projection of $\mathbf{x}$, the choice of $\mathbf{B}$ described here is probably a good one. It maximizes the (retained) variance of the resulting variables.

In fact, since the projections are uncorrelated, the percentage of variance accounted for by retaining the first q PC's is given by

$$\frac{\sum_{k=1}^{q} \lambda_k}{\sum_{k=1}^{p} \lambda_k} \times 100$$

PCA using the sample covariance matrix

If we recall that the sample covariance matrix (an unbiased estimator for the covariance matrix of $\mathbf{x}$) is given by

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}'\mathbf{X}$$

where $\mathbf{X}$ is a $(n \times p)$ matrix with $(i,j)th$ element $(x_{ij} - \bar{x}_j)$ (in other words, $\mathbf{X}$ is a zero mean design matrix).

We construct the matrix $\mathbf{A}$ by combining the p eigenvectors of $\mathbf{S}$ (or eigenvectors of $\mathbf{X}'\mathbf{X}$ – they're the same) then we can define a matrix of PC scores

$$\mathbf{Z} = \mathbf{X}\mathbf{A}$$

Of course, if we instead form $\mathbf{Z}$ by selecting the q eigenvectors corresponding to the q largest eigenvalues of $\mathbf{S}$ when forming $\mathbf{A}$ then we can achieve an "optimal" (in some senses) q-dimensional projection of $\mathbf{x}$.

Computing the PCA loading matrix

Given the sample covariance matrix

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}'\mathbf{X}$$

the most straightforward way of computing the PCA loading matrix is to utilize the singular value decomposition of $\mathbf{S} = \mathbf{A}'\boldsymbol{\Lambda}\mathbf{A}$ where $\mathbf{A}$ is a matrix consisting of the eigenvectors of $\mathbf{S}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal elements are the eigenvalues corresponding to each eigenvector.

Creating a reduced dimensionality projection of $\mathbf{X}$ is accomplished by selecting the q largest eigenvalues in $\boldsymbol{\Lambda}$ and retaining the q corresponding eigenvectors from $\mathbf{A}$

Sample Covariance Matrix PCA

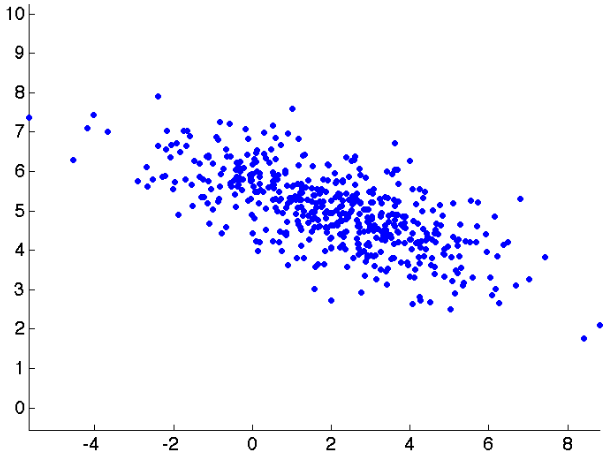


Figure: Gaussian Samples

Sample Covariance Matrix PCA

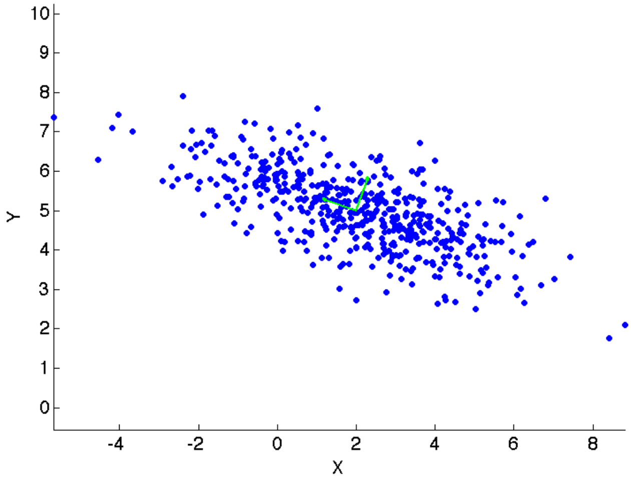


Figure: Gaussian Samples with eigenvectors of sample covariance matrix

Sample Covariance Matrix PCA

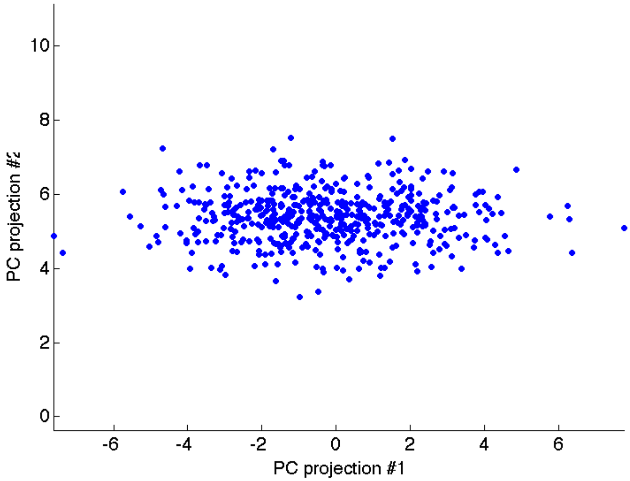


Figure: PC projected samples

Sample Covariance Matrix PCA

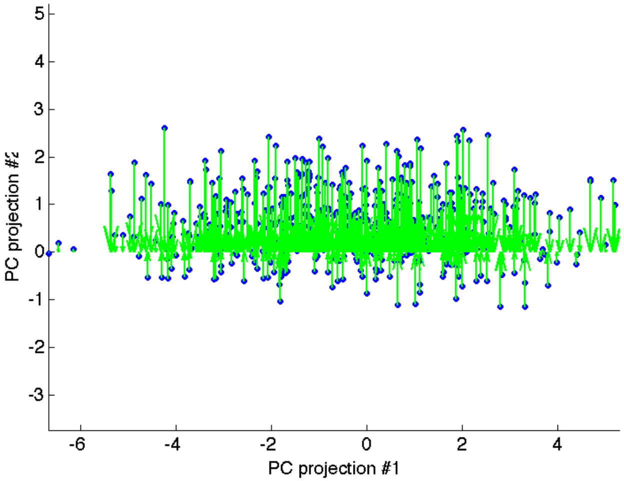


Figure: PC dimensionality reduction step

Sample Covariance Matrix PCA

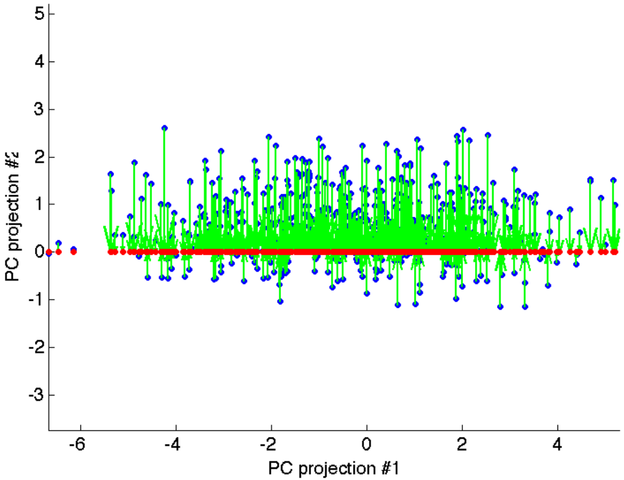


Figure: PC dimensionality reduction step

PCA in linear regression

PCA is useful in linear regression in several ways

- Identification and elimination of multicolinearities in the data.
- Reduction in the dimension of the input space leading to fewer parameters and "easier" regression.
- Related to the last point, the variance of the regression coefficient estimator is minimized by the PCA choice of basis.

We will consider the following example.

- $\mathbf{x} \sim N\left([2\ 5], \begin{bmatrix} 4.5 & -1.5 \\ -1.5 & 1.0 \end{bmatrix}\right)$
- $\mathbf{y} = \mathbf{X}[-1\ 2]'$ when no colinearities are present (no noise)
- $x_{i3} = .8x_{i1} + .5x_{i2}$ *imposed* colinearity

Noiseless Linear Relationship with No Colinearity

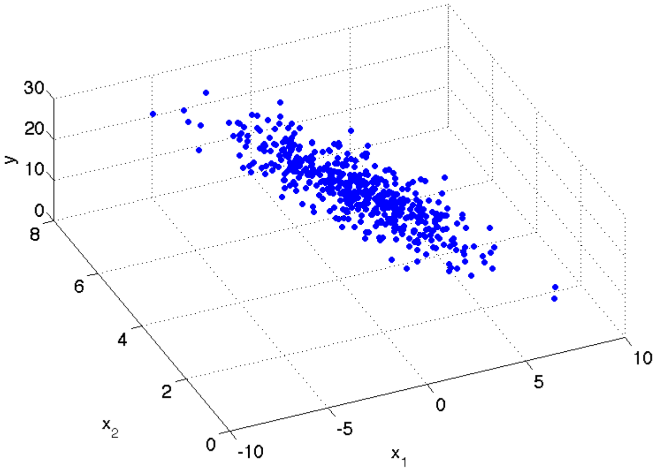


Figure: $\mathbf{y} = \mathbf{x}[-1\ 2]' + 5, \mathbf{x} \sim N([2\ 5], \begin{bmatrix} 4.5 & -1.5 \\ -1.5 & 1.0 \end{bmatrix})$

Noiseless Planar Relationship

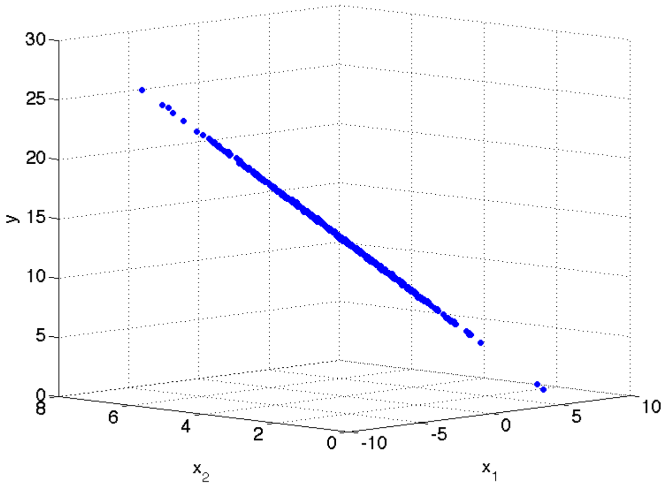


Figure: $\mathbf{y} = \mathbf{x}[-1\ 2]' + 5, \mathbf{x} \sim N([2\ 5], \begin{bmatrix} 4.5 & -1.5 \\ -1.5 & 1.0 \end{bmatrix})$

Projection of colinear data

The figures before showed the data without the third colinear design matrix column. Plotting such data is not possible, but it's colinearity is obvious by design.

When PCA is applied to the design matrix of rank q less than p the number of positive eigenvalues discovered is equal to q the true rank of the design matrix.

If the number of PC's retained is larger than q (and the data is perfectly colinear, etc.) *all* of the variance of the data is retained in the low dimensional projection.

In this example, when PCA is run on the design matrix of rank 2, the resulting projection back into two dimensions has exactly the same distribution as before.

Projection of colinear data

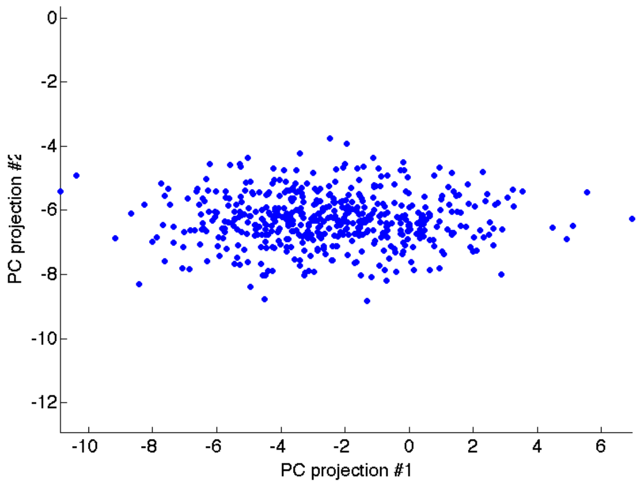


Figure: Projection of multi-colinear data onto first two PC's

Reduction in regression coefficient estimator variance

If we take the standard regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

And consider instead the PCA rotation of $\mathbf{X}$ given by

$$\mathbf{Z} = \mathbf{Z}\mathbf{A}$$

then we can rewrite the regression model in terms of the PC's

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}.$$

We can also consider the reduced model

$$\mathbf{y} = \mathbf{Z}_q\boldsymbol{\gamma}_q + \boldsymbol{\epsilon}_q$$

where only the first q PC's are retained.

Reduction in regression coefficient estimator variance

If we rewrite the regression relation as

$$\mathbf{y} = \mathbf{Z}\gamma + \epsilon.$$

Then we can, because $\mathbf{A}$ is orthogonal, rewrite

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{X}\mathbf{A}\mathbf{A}'\boldsymbol{\beta} = \mathbf{Z}\gamma$$

where $\boldsymbol{\gamma} = \mathbf{A}'\beta$.

Clearly using least squares (or ML) to learn $\hat{\boldsymbol{\beta}} = \mathbf{A}\hat{\gamma}$ is equivalent to learning $\hat{\boldsymbol{\beta}}$ directly.

And, like usual,

$$\hat{\gamma} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$$

so $\hat{\boldsymbol{\beta}} = \mathbf{A}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$

Reduction in regression coefficient estimator variance

Without derivation we note that the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$ is given by

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2 \sum_{k=1}^{p} l_k^{-1} \mathbf{a}_k \mathbf{a}_k'$$

where l_k is the k^{th} largest eigenvalue of $\mathbf{X}'\mathbf{X}$, $\mathbf{a}_k$ is the k^{th} column of $\mathbf{A}$, and σ^2 is the observation noise variance, i.e. $\boldsymbol{\epsilon} \sim \text{N}(\mathbf{0}, \sigma^2 \mathbf{I})$

This sheds light on how multicolinearities produce large variances for the elements of $\hat{\boldsymbol{\beta}}$. If an eigenvector l_k is small then the resulting variance of the estimator will be large.

Reduction in regression coefficient estimator variance

One way to avoid this is to ignore those PC's that are associated with small eigenvalues, namely, use biased estimator

$$\tilde{\boldsymbol{\beta}} = \sum_{k=1}^{m} l_k^{-1} \mathbf{a}_k \mathbf{a}_k' \mathbf{X}' \mathbf{y}$$

where $l_{1:m}$ are the large eigenvalues of $\mathbf{X}'\mathbf{X}$ and $l_{m+1:p}$ are the small.

$$\mathrm{Var}(\tilde{\boldsymbol{\beta}}) = \sigma^2 \sum_{k=1}^{m} l_k^{-1} \mathbf{a}_k \mathbf{a}_k'$$

This is a biased estimator, but, since the variance of this estimator is smaller it is possible that this could be an advantage.

Homework: find the bias of this estimator. Hint: use the spectral decomposition of $\mathbf{X}'\mathbf{X}$.

Problems with PCA

PCA is not without its problems and limitations

- PCA assumes approximate normality of the input space distribution
  - PCA may still be able to produce a “good” low dimensional projection of the data even if the data isn’t normally distributed
- PCA may “fail” if the data lies on a “complicated” manifold
- PCA assumes that the input data is real and continuous.
- Extensions to consider
  - Collins et al, A generalization of principal components analysis to the exponential family.
  - Hyv
    ”arinen, A. and Oja, E., Independent component analysis: algorithms and applications
  - ISOMAP, LLE, Maximum variance unfolding, etc.

Non-normal data

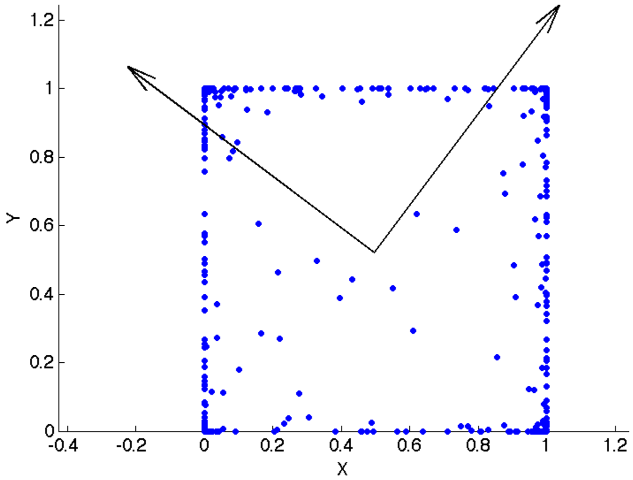


Figure: 2d Beta(.1, .1) Samples with PC's

Non-normal data

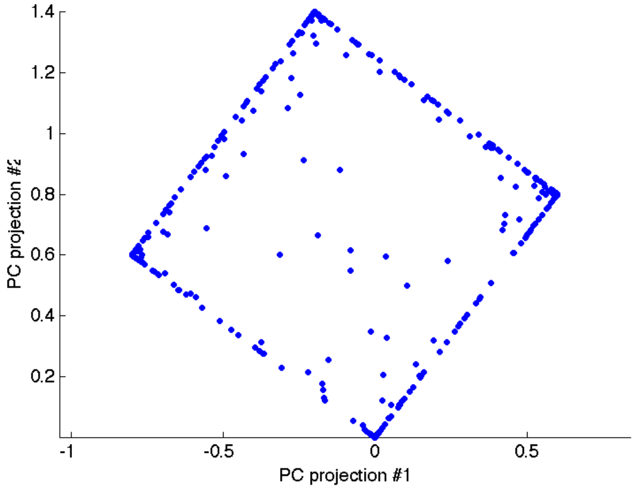


Figure: PCA Projected