BIOST 546: Machine Learning for Biomedical Big Data

Ali Shojaie

Lecture 10: Dimension Reduction - Part II
Spring 2017

Recap

- Dimension reduction
- PCA basics

Today's Class

- PCA, continued
- MDS

PCA and Dimension Reduction

Recall the USArrests data:

```
> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

PCA and Dimension Reduction

Recall the USArrests data:

```
> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

Q: With 4 PC's again have a 4 dimensional space, just as the original data, so why is PCA a dimension reduction technique?

PCA and Dimension Reduction

Recall the USArrests data:

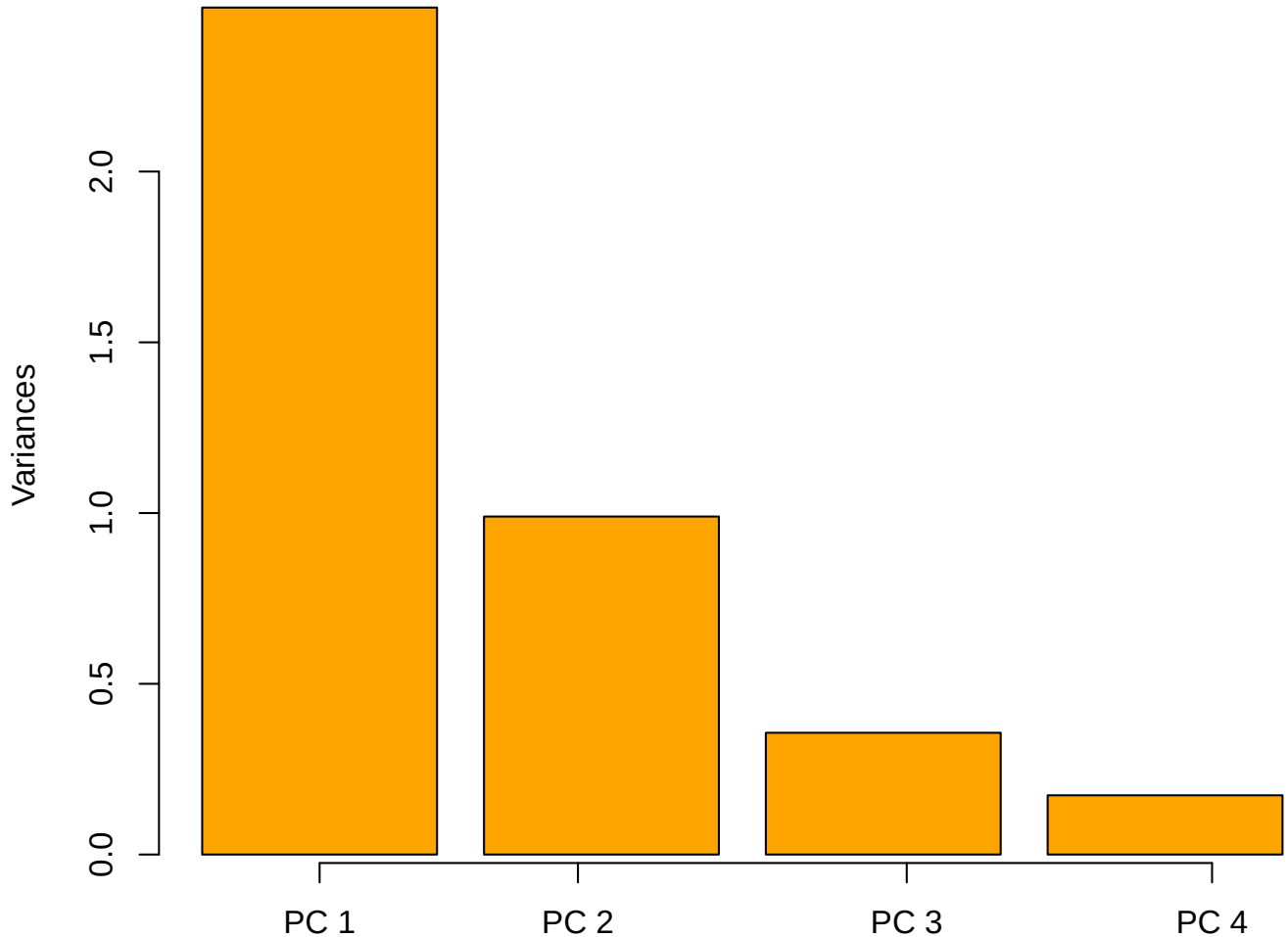
```
> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

Q: With 4 PC's again have a 4 dimensional space, just as the original data, so why is PCA a dimension reduction technique?

A: Not if we use all 4 PC's! The **dimension reduction** in PCA comes from the fact that often few PC's explain most of the variation in the data.

Variances Explained by Each PC

```
pc.out <- prcomp(USArrests, scale=TRUE, retx=TRUE)
screeplot(pc.out)
```



The Proportion of Variance Explained

- In general, we would like to know, “how much of the information in the data is lost by projecting the observations onto the first M principal components”.

The Proportion of Variance Explained

- In general, we would like to know, “how much of the information in the data is lost by projecting the observations onto the first M principal components”.
- In other words, “how much of the variance in the data is not contained in the first M principal components”?

The Proportion of Variance Explained

- In general, we would like to know, “how much of the information in the data is lost by projecting the observations onto the first M principal components”.
- In other words, “how much of the variance in the data is not contained in the first M principal components”?
- To answer these question, we need to know the proportion of variance explained (PVE) by each principal component.

The Proportion of Variance Explained (ctd.)

- The total variance in the data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{i=1}^{n}\sum_{j=1}^{p} X_{ij}^2$$

The Proportion of Variance Explained (ctd.)

- The total variance in the data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{i=1}^{n}\sum_{j=1}^{p} X_{ij}^2$$

- The variance explained by a PC loading vector w is:

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{p} X_{ij} w_j\right)^2$$

The Proportion of Variance Explained (ctd.)

- The total variance in the data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{i=1}^{n}\sum_{j=1}^{p} X_{ij}^2$$

- The variance explained by a PC loading vector w is:

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{p} X_{ij}w_j\right)^2$$

- So, the proportion of variance explained by each principal component loading vector is:

$$\frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p} X_{ij}w_j\right)^2}{\sum_{i=1}^{n}\sum_{j=1}^{p} X_{ij}^2}$$

Eigenvalues & Eigenvectors: A (Brief) Overview

Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists \lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ

Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists \lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ
- Geometrically, Σ only stretches w, but does not change its direction

Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ
- Geometrically, Σ only stretches w, but does not change its direction

Consider the transformation matrix

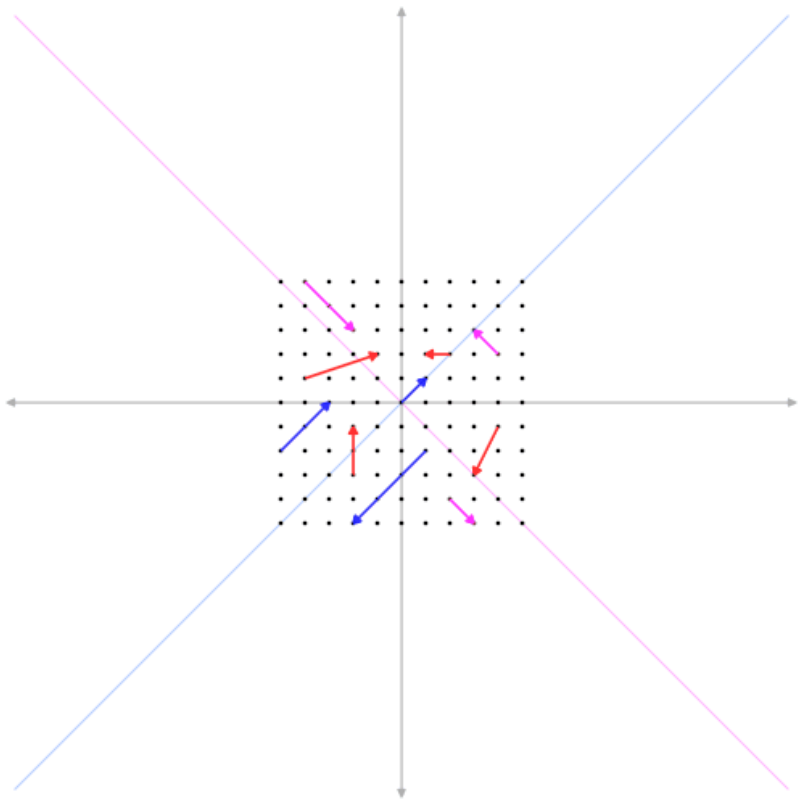
$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \text{evecs}(A) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ
- Geometrically, Σ only stretches w, but does not change its direction

Consider the transformation matrix

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \text{evecs}(A) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

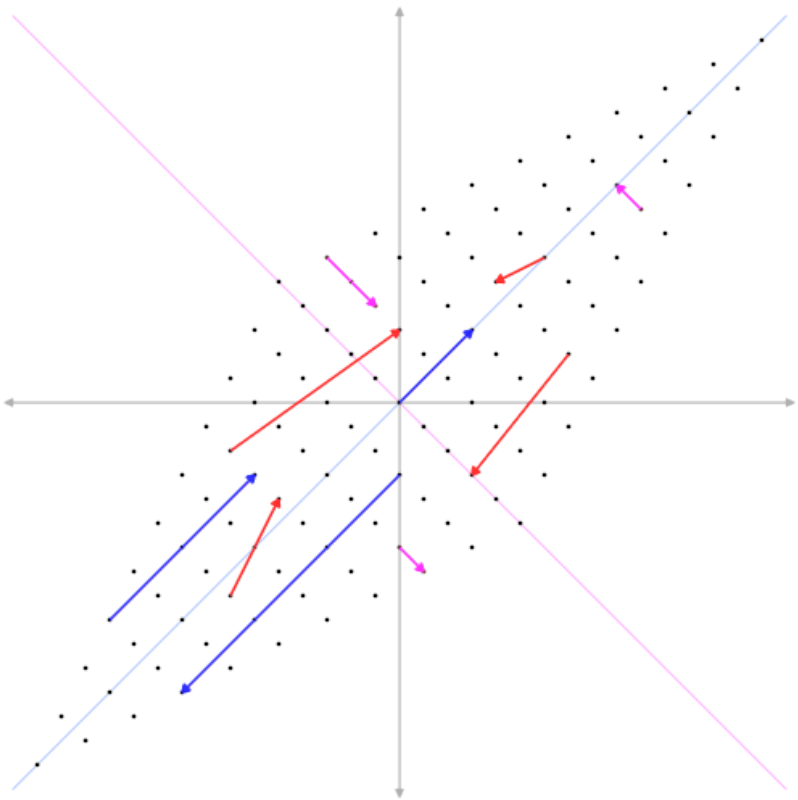


Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists \lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ
- Geometrically, Σ only stretches w, but does not change its direction

Consider the transformation matrix

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \text{evecs}(A) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



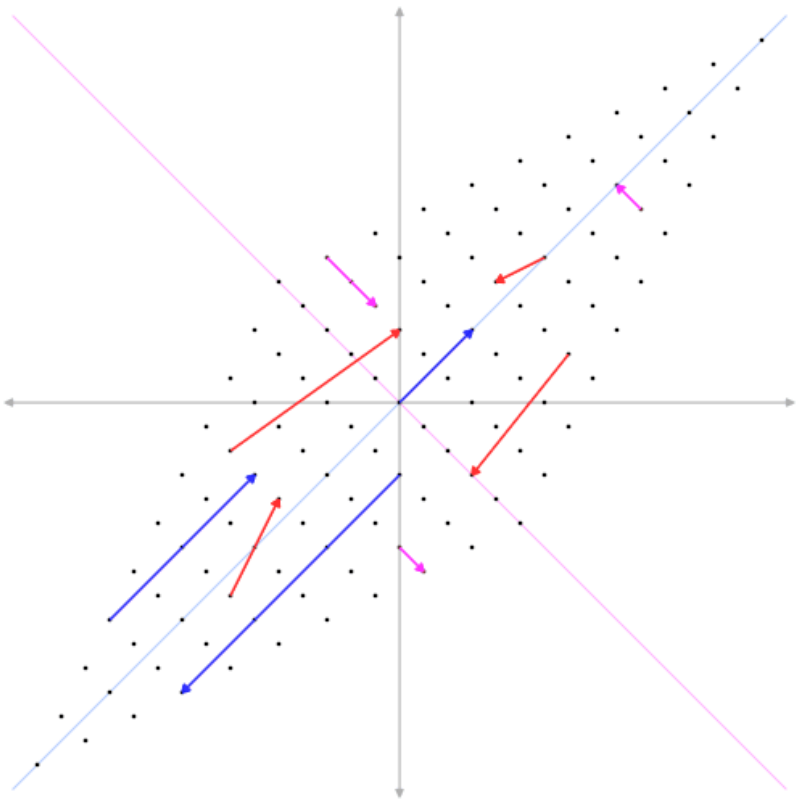
Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ
- Geometrically, Σ only stretches w, but does not change its direction

Consider the transformation matrix

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \text{evecs}(A) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

- A preserves the direction of vectors parallel to w and v.



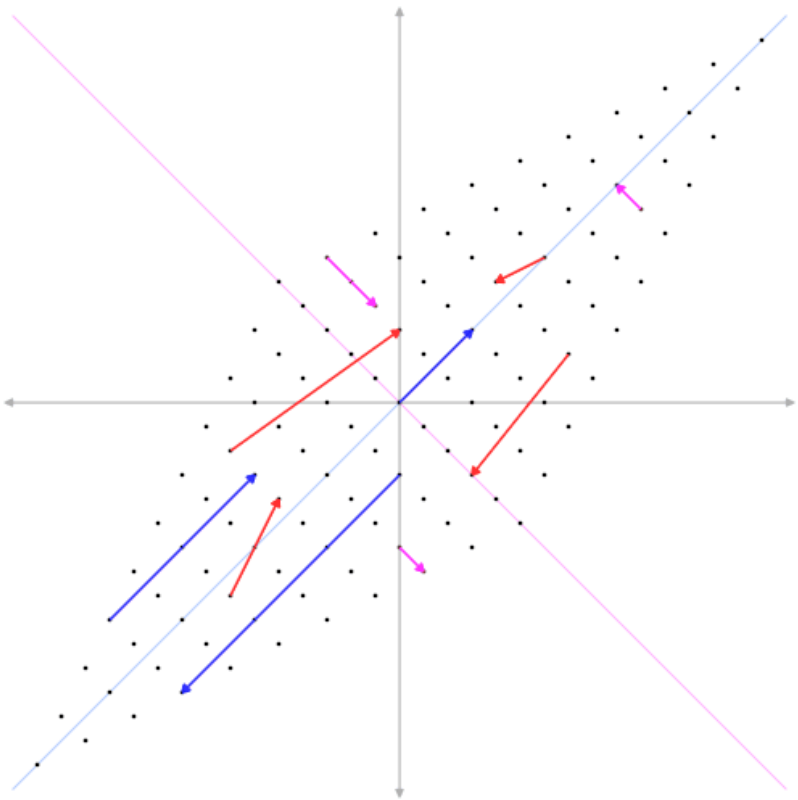
Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ
- Geometrically, Σ only stretches w, but does not change its direction

Consider the transformation matrix

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \text{evecs}(A) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

- A preserves the direction of vectors parallel to w and v.
- The red vectors are not parallel to either eigenvector, so, their directions change.



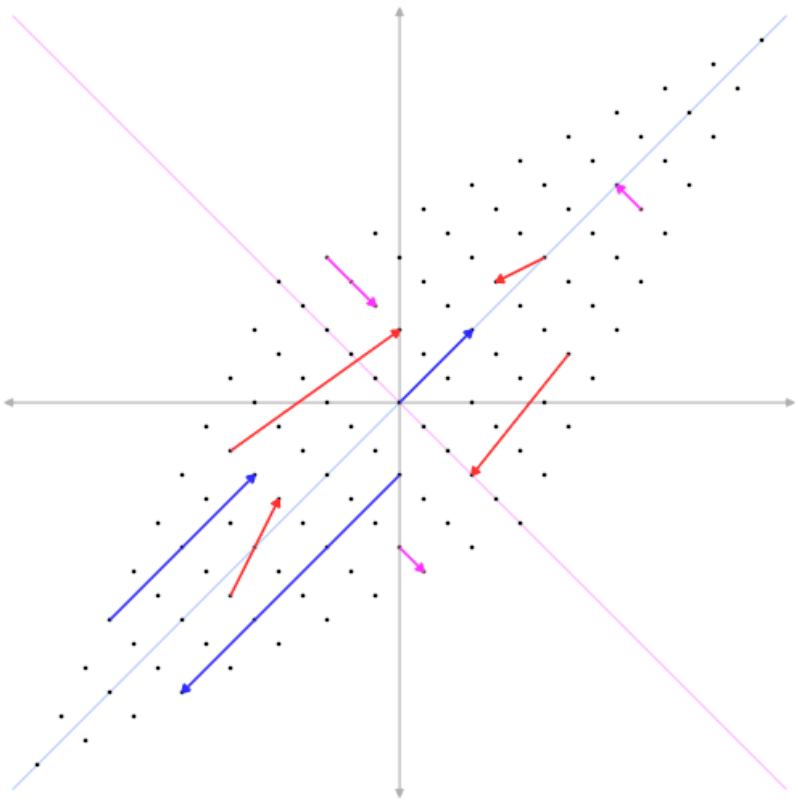
Eigenvalues & Eigenvectors: A (Brief) Overview

- w is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma w = \lambda w$
  - λ is an eigenvalue of Σ
- Geometrically, Σ only stretches w, but does not change its direction

Consider the transformation matrix

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \text{evecs}(A) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

- A preserves the direction of vectors parallel to w and v.
- The red vectors are not parallel to either eigenvector, so, their directions change.
- The length of purple vectors does not change, so $\lambda_w = 1$.



Eigenvectors/Eigenvalues of Σ

Eigenvectors/Eigenvalues of Σ

- The covariance matrix Σ defines the shape of the data cloud.

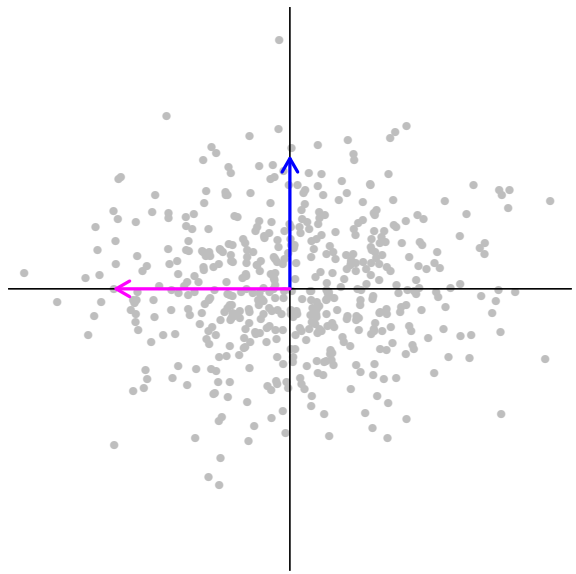
Eigenvectors/Eigenvalues of Σ

- The covariance matrix Σ defines the shape of the data cloud.
- The eigenvectors Σ capture the directions of spread, while its eigenvalues capture the amount of spread in each direction.

Eigenvectors/Eigenvalues of Σ

- The covariance matrix Σ defines the shape of the data cloud.
- The eigenvectors Σ capture the directions of spread, while its eigenvalues capture the amount of spread in each direction.

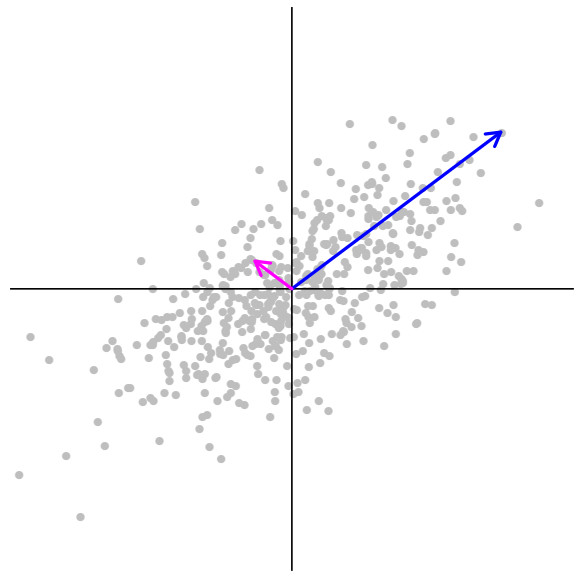
- $\Sigma = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, i.e., uncorrelated data
  - Here, eigenvectors are $\binom{0}{1}$ and $\binom{-1}{0}$ and the eigenvalues are 1 and 1.



Eigenvectors/Eigenvalues of Σ

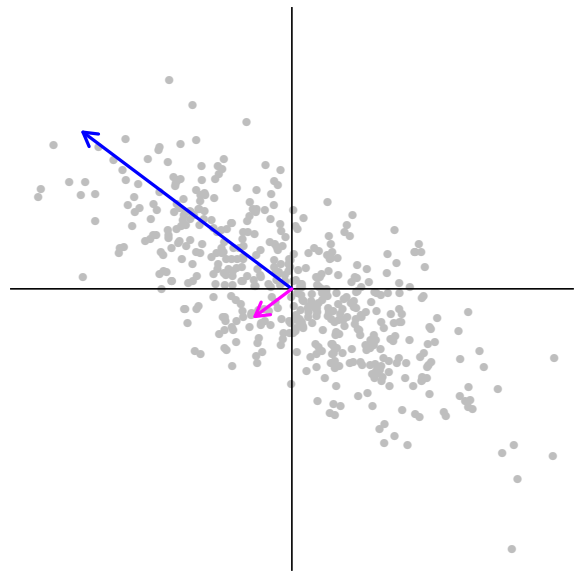
- The covariance matrix Σ defines the shape of the data cloud.
- The eigenvectors Σ capture the directions of spread, while its eigenvalues capture the amount of spread in each direction.

- $\Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$, i.e., cor $= 0.7$
  - Here, eigenvectors are $\begin{pmatrix} 0.7 \\ 0.7 \end{pmatrix}$ and $\begin{pmatrix} -0.7 \\ 0.7 \end{pmatrix}$ and the eigenvalues are 1.7 and 0.3.



Eigenvectors/Eigenvalues of Σ

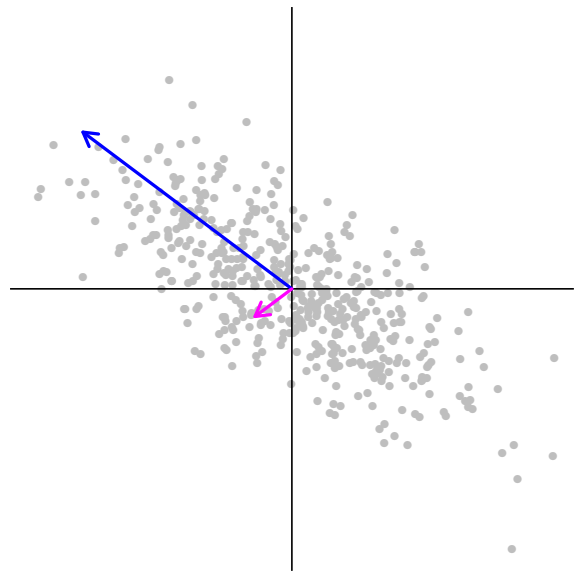
- The covariance matrix Σ defines the shape of the data cloud.
- The eigenvectors Σ capture the directions of spread, while its eigenvalues capture the amount of spread in each direction.



- $\Sigma = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 1 \end{bmatrix}$, i.e., cor $= -0.7$
  - Here, eigenvectors are $\begin{pmatrix} -0.7 \\ 0.7 \end{pmatrix}$ and $\begin{pmatrix} -0.7 \\ -0.7 \end{pmatrix}$ and the eigenvalues are 1.7 and 0.3.

Eigenvectors/Eigenvalues of Σ

- The covariance matrix Σ defines the shape of the data cloud.
- The eigenvectors Σ capture the directions of spread, while its eigenvalues capture the amount of spread in each direction.



- $\Sigma = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 1 \end{bmatrix}$, i.e., cor $= -0.7$
  - Here, eigenvectors are $\begin{pmatrix} -0.7 \\ 0.7 \end{pmatrix}$ and $\begin{pmatrix} -0.7 \\ -0.7 \end{pmatrix}$ and the eigenvalues are 1.7 and 0.3.

$\Rightarrow$ PC loadings are eigenvectors of the covariance matrix.

Variances Explained by Each PC

- Let's look at the variances for each of the PC's:

Variance	PC1	PC2	PC3	PC4
	2.4802	0.9898	0.3566	0.1734

Variances Explained by Each PC

- Let's look at the variances for each of the PC's:

Variance	PC1	PC2	PC3	PC4
	2.4802	0.9898	0.3566	0.1734

- These are the values plotted previously in the barplot

Variances Explained by Each PC

- Let's look at the variances for each of the PC's:

Variance	PC1	PC2	PC3	PC4
	2.4802	0.9898	0.3566	0.1734

- These are the values plotted previously in the barplot
- If we take all of the PCs, we can explain all the variance in the data (if $p < n$)

Variances Explained by Each PC

- Let's look at the variances for each of the PC's:

Variance	PC1	PC2	PC3	PC4
	2.4802	0.9898	0.3566	0.1734

- These are the values plotted previously in the barplot
- If we take all of the PCs, we can explain all the variance in the data (if $p < n$)
- To compute the PVE for each principal component, we simply divide the variance explained by each principal component by the total variance explained by all four principal components:

PVE	PC1	PC2	PC3	PC4
	0.620	0.247	0.089	0.043

Variances Explained by Each PC

- Let's look at the variances for each of the PC's:

Variance	PC1	PC2	PC3	PC4
	2.4802	0.9898	0.3566	0.1734

- These are the values plotted previously in the barplot
- If we take all of the PCs, we can explain all the variance in the data (if $p < n$)
- To compute the PVE for each principal component, we simply divide the variance explained by each principal component by the total variance explained by all four principal components:

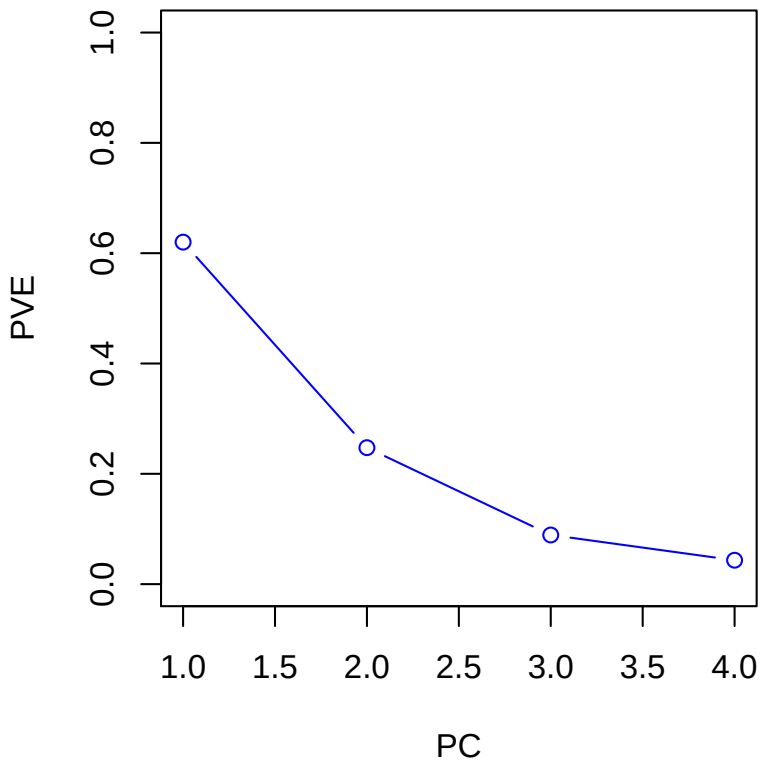
PVE	PC1	PC2	PC3	PC4
	0.620	0.247	0.089	0.043

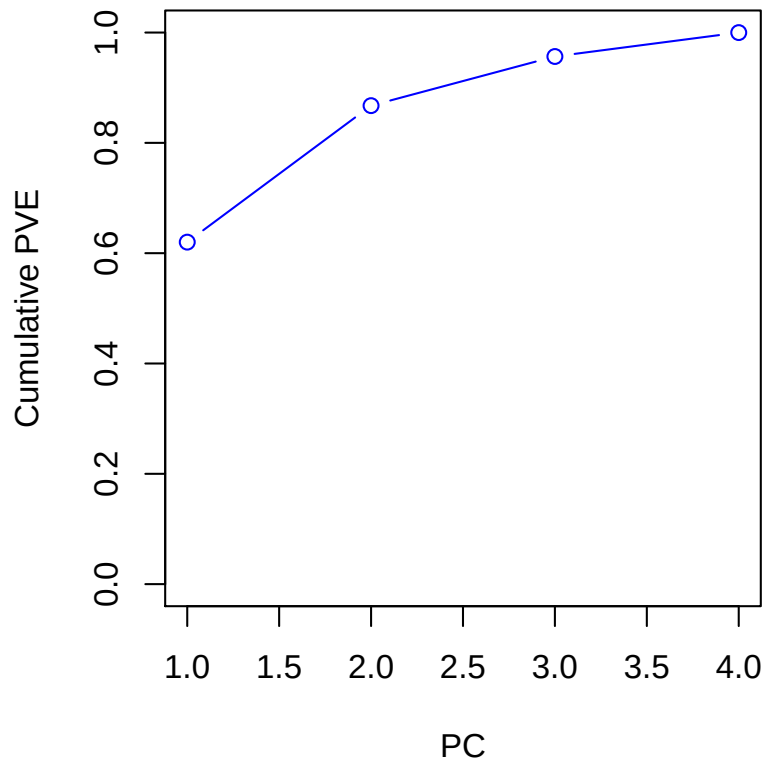
- We can see that the first 2 PC's explain 87% of the variance in the data.

Scree Plots

- We can plot the PVE as well as the cumulative PVE:

```
plot((pc.out$sdev^2)/sum(pc.out$sdev^2), xlab="PC",
     ylab="PVE", ylim=c(0, 1), type='b')
plot(cumsum(pc.out$sdev^2)/sum(pc.out$sdev^2), xlab="PC",
    ylab="Cumulative PVE", ylim=c(0, 1), type='b')
```





How Many PC's?

- A natural question that arises in applications of PCA in high dimensional settings, is how many PC's should we use?

How Many PC's?

- A natural question that arises in applications of PCA in high dimensional settings, is how many PC's should we use?
- On the one hand, by increasing the number of PC's, we include more of the variation in the data.

How Many PC's?

- A natural question that arises in applications of PCA in high dimensional settings, is how many PC's should we use?
- On the one hand, by increasing the number of PC's, we include more of the variation in the data.
- On the other hand, if we use too many PC's, we don't get the benefits of dimension reduction (e.g. in visualization and prediction).

How Many PC's?

- A natural question that arises in applications of PCA in high dimensional settings, is how many PC's should we use?
- On the one hand, by increasing the number of PC's, we include more of the variation in the data.
- On the other hand, if we use too many PC's, we don't get the benefits of dimension reduction (e.g. in visualization and prediction).
- We want to use the smallest number of PC's that would give us a "good" understanding of the data.

How Many PC's?

- A natural question that arises in applications of PCA in high dimensional settings, is how many PC's should we use?
- On the one hand, by increasing the number of PC's, we include more of the variation in the data.
- On the other hand, if we use too many PC's, we don't get the benefits of dimension reduction (e.g. in visualization and prediction).
- We want to use the smallest number of PC's that would give us a "good" understanding of the data.
- Unfortunately, like many other questions in unsupervised learning, there is no easy answer to this question.

How Many PC's?

- One idea is to use enough PC's that explain a significant proportion of variances in the data.

How Many PC's?

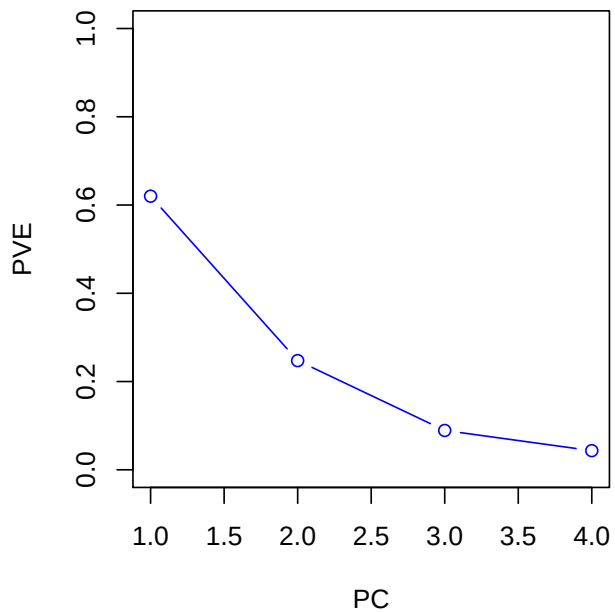
- One idea is to use enough PC's that explain a significant proportion of variances in the data.
- To answer this question, we can use the plots in the previous slide to decide the number of PCs to include.

How Many PC's?

- One idea is to use enough PC's that explain a significant proportion of variances in the data.
- To answer this question, we can use the plots in the previous slide to decide the number of PCs to include.
- This is done by eyeballing the scree plot, and looking for a point at which the PVE by each subsequent PC drops off. This is often referred to as an "elbow" in the scree plot.

How Many PC's?

- One idea is to use enough PC's that explain a significant proportion of variances in the data.
- To answer this question, we can use the plots in the previous slide to decide the number of PCs to include.
- This is done by eyeballing the scree plot, and looking for a point at which the PVE by each subsequent PC drops off. This is often referred to as an "elbow" in the scree plot.



PCA in High Dimensions: NCI60 Data

- The dataset includes gene expression data for 6830 genes from 64 cancer samples (from different cancer subtypes)
- Data can be downloaded form
  `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`
- Missing values have been imputed
- For this analysis (and to simplify the plots), I have removed 5 subtypes with only 1 samples:
  `UNKNOWN, K562B-repro, K562A-repro, MCF7A-repro, MCF7D-repro`
- Let's perform PCA on this data!

PCA in High Dimensions: NCI60 Data

- First, which one of these datasets should we use?

```
> dat.1 <- read.table('nci-mod.data')
> dim(dat.1)
[1] 6830   59
> dat.2 <- t(dat.1)      #transposing the data
> dim(dat.2)
[1]   59 6830
```

- How many PC's will we get from applying PCA to `dat.1`? How about `dat.2`?
- What's the difference? Which one should we use?

PCA in High Dimensions: NCI60 Data (ctd.)

Let's try both!

```
> pc.1 <- prcomp(dat.1, scale=TRUE, retx=TRUE)
> pc.2 <- prcomp(dat.2, scale=TRUE, retx=TRUE)

> dim(pc.1$rot)
[1] 59 59
> dim(pc.1$x)
[1] 6830   59
> dim(pc.2$rot)
[1] 6830   59
> dim(pc.2$x)
[1] 59 59
```

PCA in High Dimensions: NCI60 Data (ctd.)

Let's try both!

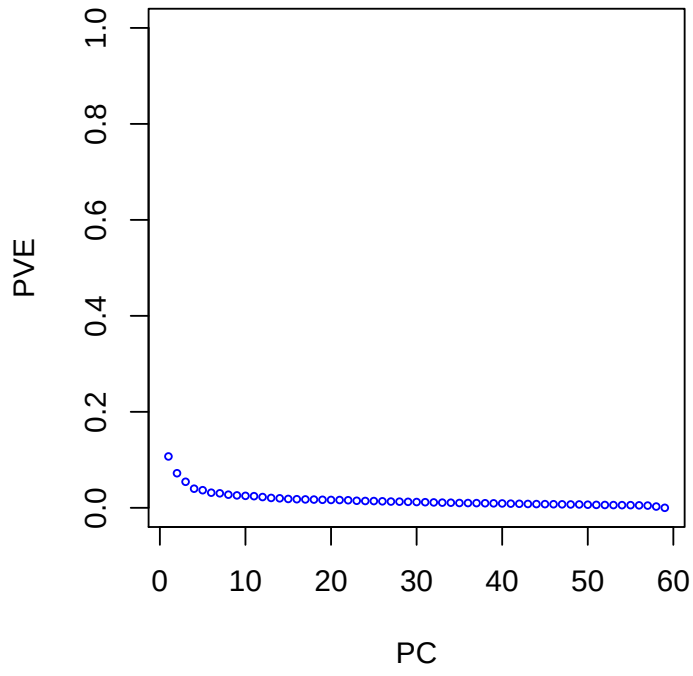
```
> pc.1 <- prcomp(dat.1, scale=TRUE, retx=TRUE)
> pc.2 <- prcomp(dat.2, scale=TRUE, retx=TRUE)

> dim(pc.1$rot)
[1] 59 59
> dim(pc.1$x)
[1] 6830   59
> dim(pc.2$rot)
[1] 6830   59
> dim(pc.2$x)
[1] 59 59
```

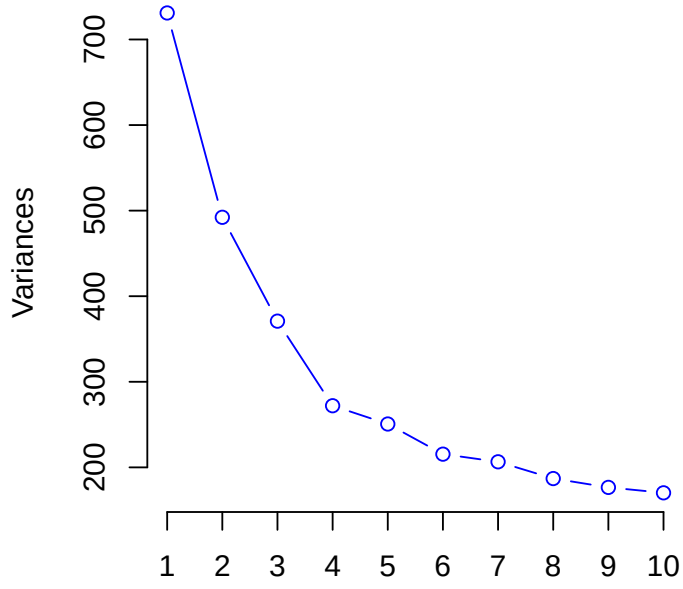
Remarks:

- Each PC loading should be of the same length as the number of variables! (each column of `$rot` is a PC loading)
- There are at most $\min(n,p)$ PC loadings; for $p \gg n$, we have at most n PC's.
- Variables should always be in columns of X!

How Many PC's?

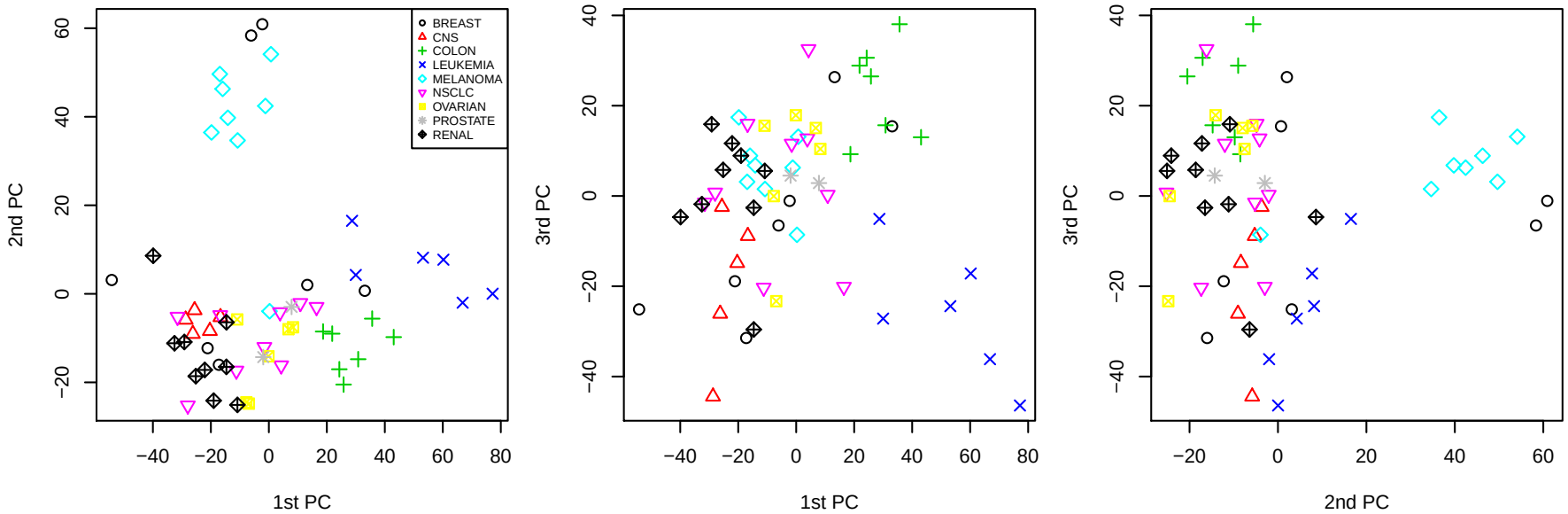
- In high dimensions, the proportion of variance explained for each PC is often small.
- However, the first few PC's usually provide a good projection, and we can usually find an "elbow" in the scree plot. Here, 4 PC's seems to be a good choice.





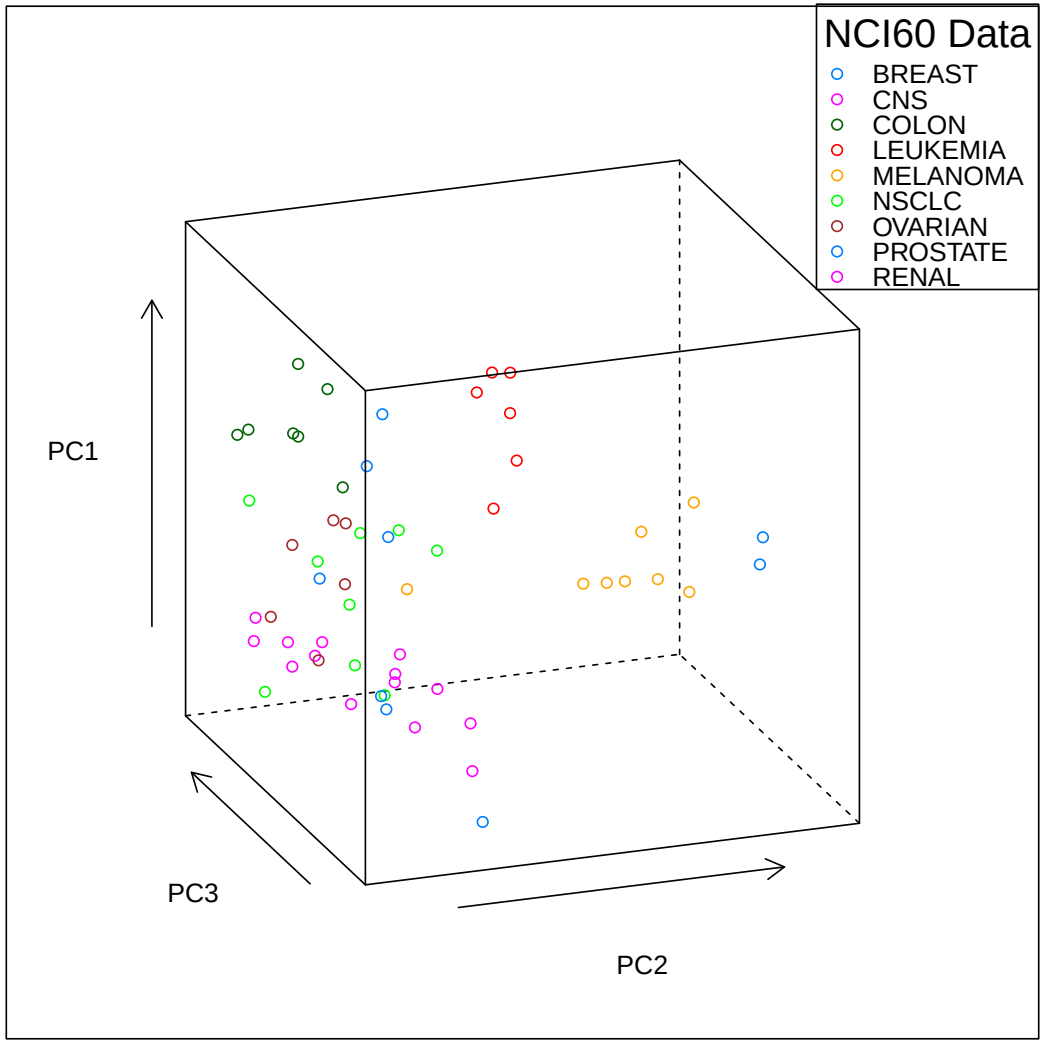
Projecting Samples into the PC Space

- Let's look at the space of the first 3 PC's



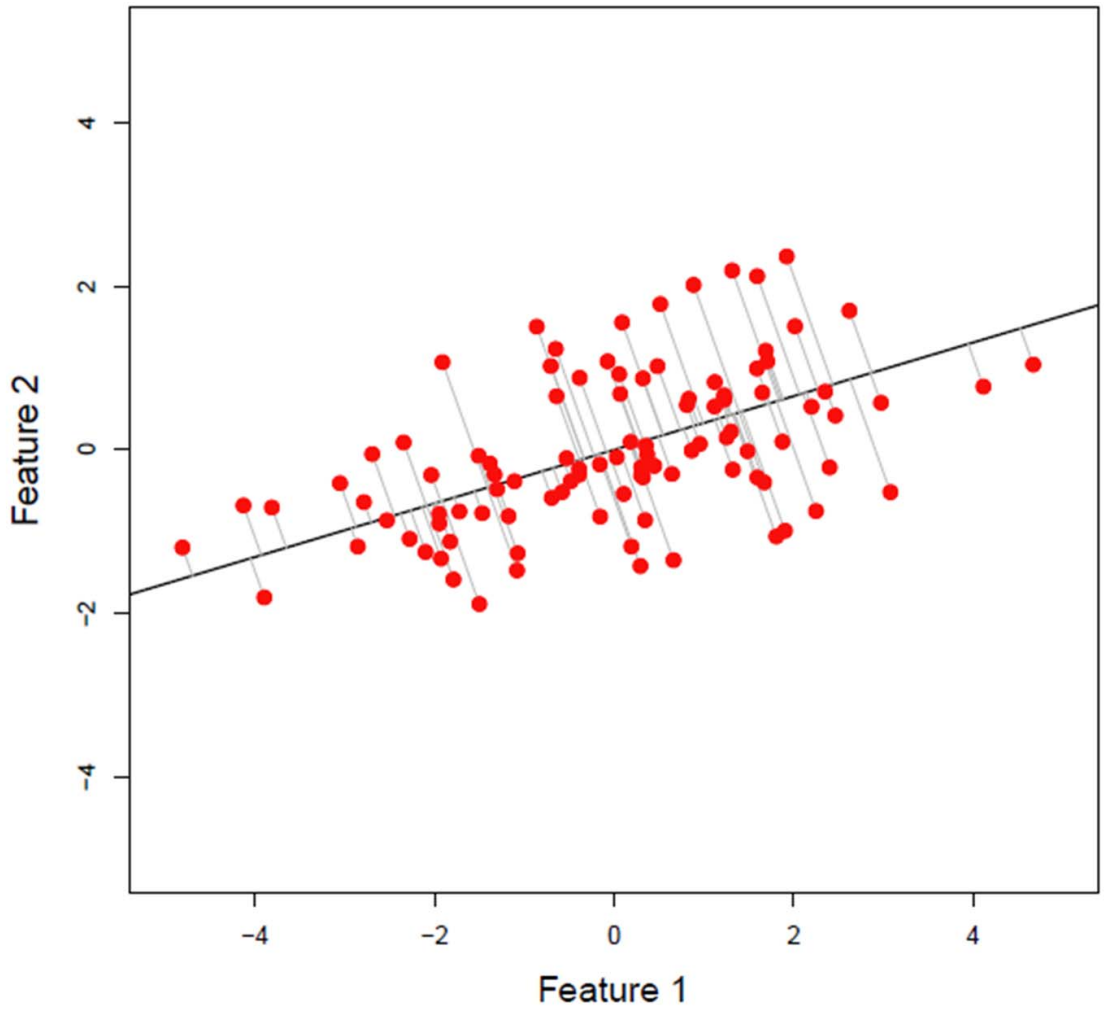
- After projecting the samples into the space of PC's, we can perform clustering or classification in the new space.

A Different View



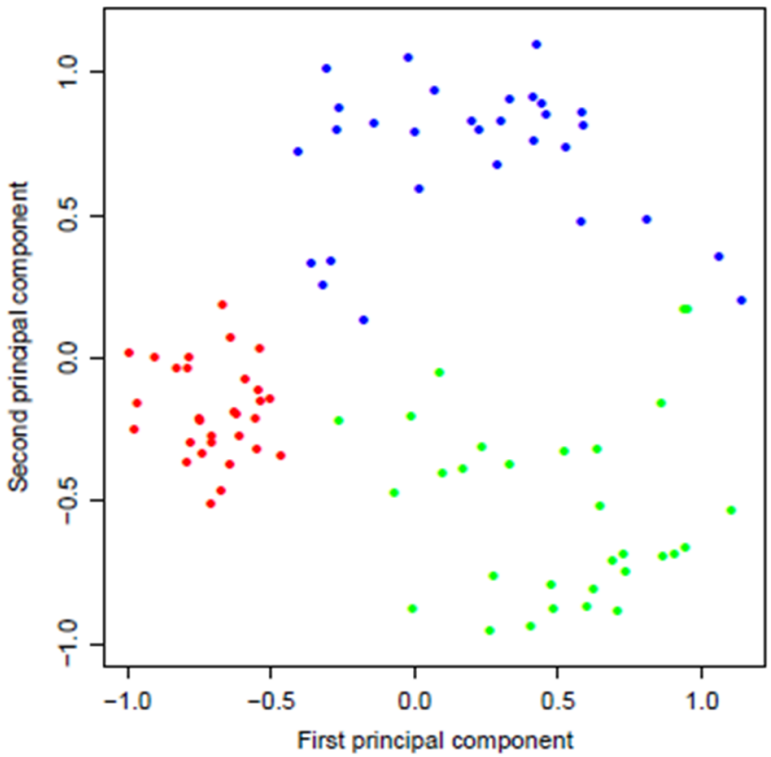
Another look at PCA

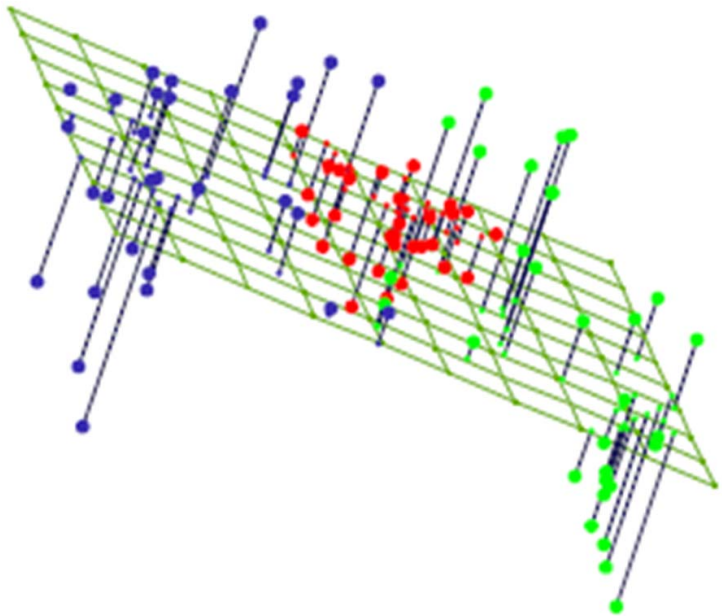
The first PC gives the direction with minimum orthogonal distance from the observations.



Another look at PCA

In general, the PCA Solution finds a *hyperplane* that is closest to the data in terms of squared orthogonal distance.





The PCA Solution

- Recall the PCA problem:

$$w_m = \max_{w:\|w\|=1} \mathrm{Var}(w^\mathsf{T} X) \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

The PCA Solution

- Recall the PCA problem:

$$w_m = \max_{w:\|w\|=1} \mathrm{Var}(w^\mathsf{T} X) \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

- Using the fact that, $\mathrm{Var}(w^\mathsf{T} X) = w^\mathsf{T} \mathrm{Var}(X) w = w^\mathsf{T} \Sigma w$ (with Σ the covariance matrix of X), we can write this problem as:

$$w_m = \max_{w:\|w\|=1} w^\mathsf{T} \Sigma w \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

The PCA Solution

- Recall the PCA problem:

$$w_m = \max_{w:\|w\|=1} \mathrm{Var}(w^\mathsf{T} X) \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

- Using the fact that, $\mathrm{Var}(w^\mathsf{T} X) = w^\mathsf{T} \mathrm{Var}(X) w = w^\mathsf{T} \Sigma w$ (with Σ the covariance matrix of X), we can write this problem as:

$$w_m = \max_{w:\|w\|=1} w^\mathsf{T} \Sigma w \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

- In compact matrix form, the problem can be written as:

$$\max_{W:W^\mathsf{T} W = I} W^\mathsf{T} \Sigma W$$

The PCA Solution

- Recall the PCA problem:

$$w_m = \max_{w:\|w\|=1} \mathrm{Var}(w^\mathsf{T} X) \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

- Using the fact that, $\mathrm{Var}(w^\mathsf{T} X) = w^\mathsf{T} \mathrm{Var}(X) w = w^\mathsf{T} \Sigma w$ (with Σ the covariance matrix of X), we can write this problem as:

$$w_m = \max_{w:\|w\|=1} w^\mathsf{T} \Sigma w \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

- In compact matrix form, the problem can be written as:

$$\max_{W:W^\mathsf{T} W = I} W^\mathsf{T} \Sigma W$$

- This is a well-known problem in mathematics, and its solution is known to be given by the eigenvectors of Σ (i.e. PC loadings are eigenvectors of the covariance matrix).

The PCA Solution (ctd.)

- v is an eigenvector of Σ iff $\exists \lambda$ such that $\Sigma v = \lambda v$; λ is an eigenvalue of Σ

The PCA Solution (ctd.)

- v is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma v = \lambda v$; λ is an eigenvalue of Σ
- This means that Σ only stretches the vector v, but does not change its direction

The PCA Solution (ctd.)

- v is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma v = \lambda v$; λ is an eigenvalue of Σ
- This means that Σ only stretches the vector v, but does not change its direction
- We can also write the eigen decomposition of Σ as $V\Lambda V^{\mathsf{T}} = \sum_1^p \lambda_j v_j v_j^{\mathsf{T}}$.

The PCA Solution (ctd.)

- v is an eigenvector of Σ iff $\exists\lambda$ such that $\Sigma v = \lambda v$; λ is an eigenvalue of Σ
- This means that Σ only stretches the vector v, but does not change its direction
- We can also write the eigen decomposition of Σ as $V\Lambda V^{\mathsf{T}} = \sum_1^p \lambda_j v_j v_j^{\mathsf{T}}$.
  - Here, Λ is a diagonal matrix of eigenvalues.
  - The matrix of eigenvectors V gives the PC loading vectors.
  - The eigenvalues give the amount of variance explained by each eigenvector
  - For symmetric matrices, the eigenvectors are orthogonal: $v_j^{\mathsf{T}} v_k = 0, j \neq k$

The PCA Solution (ctd.)

- In practice, the PCA problem is solved using singular value decomposition (SVD) on the data matrix X, which gives equivalent, but more robust results (especially for high dimensional problem).

The PCA Solution (ctd.)

- In practice, the PCA problem is solved using singular value decomposition (SVD) on the data matrix X, which gives equivalent, but more robust results (especially for high dimensional problem).
- The `R` function `prcomp` solves the PCA problem using the SVD of X.

The PCA Solution (ctd.)

- In practice, the PCA problem is solved using singular value decomposition (SVD) on the data matrix X, which gives equivalent, but more robust results (especially for high dimensional problem).
- The `R` function `prcomp` solves the PCA problem using the SVD of X.
- However, there is another function in R for PCA: `princomp`, which solves the PCA problem by finding the eigen-decomposition of X.

The PCA Solution (ctd.)

- In practice, the PCA problem is solved using singular value decomposition (SVD) on the data matrix X, which gives equivalent, but more robust results (especially for high dimensional problem).
- The `R` function `prcomp` solves the PCA problem using the SVD of X.
- However, there is another function in R for PCA: `princomp`, which solves the PCA problem by finding the eigen-decomposition of X.
- In most cases, these functions produce identical results. However, the method of estimation of PC's in the two functions is different.

More on PCA

- PCA doesn't yield feature selection – all of the original predictors are involved in the final model.

More on PCA

- PCA doesn't yield feature selection – all of the original predictors are involved in the final model.
- But when M is small, then PCA offers efficient dimension reduction, which can result in better prediction and visualization.

More on PCA

- PCA doesn't yield feature selection – all of the original predictors are involved in the final model.
- But when M is small, then PCA offers efficient dimension reduction, which can result in better prediction and visualization.
- Often PCA results are used to gain insight into high dimensional data. Trying to guess what the components might mean is a good idea, but you should not over-interpret the results of PCA.

More on PCA

- PCA doesn't yield feature selection – all of the original predictors are involved in the final model.
- But when M is small, then PCA offers efficient dimension reduction, which can result in better prediction and visualization.
- Often PCA results are used to gain insight into high dimensional data. Trying to guess what the components might mean is a good idea, but you should not over-interpret the results of PCA.
- The ideas and insight that we get from PCA is for exploration and pattern discovery, and we cannot say with certainty whether our interpretations of the PC's are accurate (e.g. the sign is arbitrary)

More on PCA

- PCA doesn't yield feature selection – all of the original predictors are involved in the final model.
- But when M is small, then PCA offers efficient dimension reduction, which can result in better prediction and visualization.
- Often PCA results are used to gain insight into high dimensional data. Trying to guess what the components might mean is a good idea, but you should not over-interpret the results of PCA.
- The ideas and insight that we get from PCA is for exploration and pattern discovery, and we cannot say with certainty whether our interpretations of the PC's are accurate (e.g. the sign is arbitrary)
- And finally, remember that this is unsupervised learning!

Be Careful!

nature genetics

Interpreting principal component analyses of spatial population genetic variation

John Novembre[1,3] & Matthew Stephens[1,2]

Nearly 30 years ago, Cavalli-Sforza *et al.* pioneered the use of principal component analysis (PCA) in population genetics and used PCA to produce maps summarizing human genetic variation across continental regions[1]. They interpreted gradient and wave patterns in these maps as signatures of specific migration events[1–3]. These interpretations have been controversial[4–7], but influential[8], and the use of PCA has become widespread in analysis of population genetics data[9–13]. However, the behavior of PCA for genetic data showing continuous spatial variation, such as might exist within human continental groups, has been less well characterized. Here, we find that gradients and waves observed in Cavalli-Sforza *et al.*'s maps resemble sinusoidal mathematical artifacts that arise generally when PCA is applied to spatial data, implying that the patterns do not necessarily reflect specific migration events. Our findings aid interpretation of PCA results and suggest how PCA can help correct for continuous population structure in association studies.

Multi Dimensional Scaling (MDS)

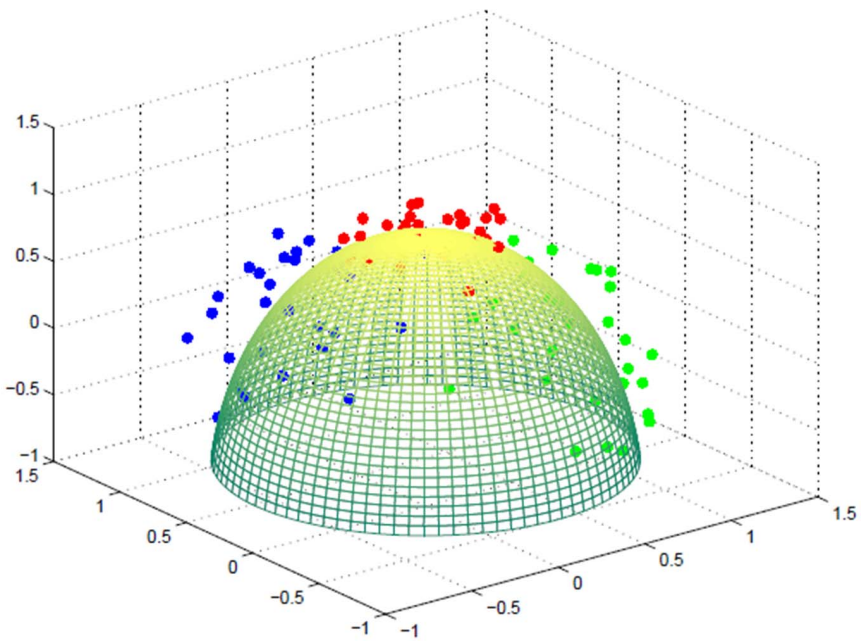
- PCA is a linear dimension reduction method: it uses linear combinations of the original variables that explain the maximum variation in the data.

Multi Dimensional Scaling (MDS)

- PCA is a linear dimension reduction method: it uses linear combinations of the original variables that explain the maximum variation in the data.
- However, in some cases, linear transformations of the variables may not work well.
- This is especially the case, if the data truly belongs on a nonlinear space (called a *manifold*).

Multi Dimensional Scaling (MDS)

- PCA is a linear dimension reduction method: it uses linear combinations of the original variables that explain the maximum variation in the data.
- However, in some cases, linear transformations of the variables may not work well.
- This is especially the case, if the data truly belongs on a nonlinear space (called a *manifold*).



Multi Dimensional Scaling (MDS)

Multi Dimensional Scaling (MDS)

- In PCA, similarities between variables are defined based on covariance.

Multi Dimensional Scaling (MDS)

- In PCA, similarities between variables are defined based on covariance.
- Covariances are really inner products, which correspond to the Euclidean distance.

Multi Dimensional Scaling (MDS)

- In PCA, similarities between variables are defined based on covariance.
- Covariances are really inner products, which correspond to the Euclidean distance.
- One way to make dimension reduction more flexible is to use a more general similarity and/or “distance”.

Multi Dimensional Scaling (MDS)

- In PCA, similarities between variables are defined based on covariance.
- Covariances are really inner products, which correspond to the Euclidean distance.
- One way to make dimension reduction more flexible is to use a more general similarity and/or “distance”.
- MDS directly works with a distance/dissimilarity matrix of observations.

Multi Dimensional Scaling (MDS)

- In PCA, similarities between variables are defined based on covariance.
- Covariances are really inner products, which correspond to the Euclidean distance.
- One way to make dimension reduction more flexible is to use a more general similarity and/or “distance”.
- MDS directly works with a distance/dissimilarity matrix of observations.
- It then finds a lower dimensional representation (often 2-3D) to preserve the pairwise distances as well as possible. For example, MDS for 2 coordinates optimizes the object locations for a 2D scatterplot.

Multi Dimensional Scaling (MDS)

- In PCA, similarities between variables are defined based on covariance.
- Covariances are really inner products, which correspond to the Euclidean distance.
- One way to make dimension reduction more flexible is to use a more general similarity and/or “distance”.
- MDS directly works with a distance/dissimilarity matrix of observations.
- It then finds a lower dimensional representation (often 2-3D) to preserve the pairwise distances as well as possible. For example, MDS for 2 coordinates optimizes the object locations for a 2D scatterplot.

MDS: Some details

MDS: Some details

- Let D be a (arbitrary) dissimilarity matrix for objects $1, \ldots, n$. So, the dissimilarity between individuals i and i' is $d_{ii'}$.

MDS: Some details

- Let D be a (arbitrary) dissimilarity matrix for objects $1, \ldots, n$. So, the dissimilarity between individuals i and i' is $d_{ii'}$.
- The goal of MDS is to find a representation of the data in low-dimensions (usually $\mathbb{R}^2$ or $\mathbb{R}^3$) so that the distances are preserved
- Formally, we want to find coordinates $z_i \in \mathbb{R}^m$ (i.e. vectors of length d) such that $\|z_i - z_{i'}\| \approx d_{ii'}$ for all i, i'

MDS: Some details

- Let D be a (arbitrary) dissimilarity matrix for objects $1, \ldots, n$. So, the dissimilarity between individuals i and i' is $d_{ii'}$.
- The goal of MDS is to find a representation of the data in low-dimensions (usually $\mathbb{R}^2$ or $\mathbb{R}^3$) so that the distances are preserved
- Formally, we want to find coordinates $z_i \in \mathbb{R}^m$ (i.e. vectors of length d) such that $\|z_i - z_{i'}\| \approx d_{ii'}$ for all i, i'
- More formally, MDS tries to find n vectors of length m, $z_i, i = 1, \ldots, n$, that minimize the following *stress function*:

$$S_M(z_1, z_2, \ldots, z_n) = \sum_{i \neq i'} (d_{ii'} - \|z_i - z_{i'}\|)^2$$

MDS: Some comments

MDS: Some comments

- Note that the only thing needed for the MDS is the dissimilarity matrix D. In general, different dissimilarity measures can give different MDS solutions

MDS: Some comments

- Note that the only thing needed for the MDS is the dissimilarity matrix D. In general, different dissimilarity measures can give different MDS solutions
- The "classical" MDS uses the Euclidean distance, which gives results somewhat similar to PCA (though the objective is different). Classical MDS is also referred to as Principal Coordinate Analysis

MDS: Some comments

- Note that the only thing needed for the MDS is the dissimilarity matrix D. In general, different dissimilarity measures can give different MDS solutions
- The "classical" MDS uses the Euclidean distance, which gives results somewhat similar to PCA (though the objective is different). Classical MDS is also referred to as Principal Coordinate Analysis
  - PCA: finds a lower-dimensional representation that maximizes the variance
  - MDS: finds a lower-dimensional representation that best preserves the distances between the points

MDS: Some comments

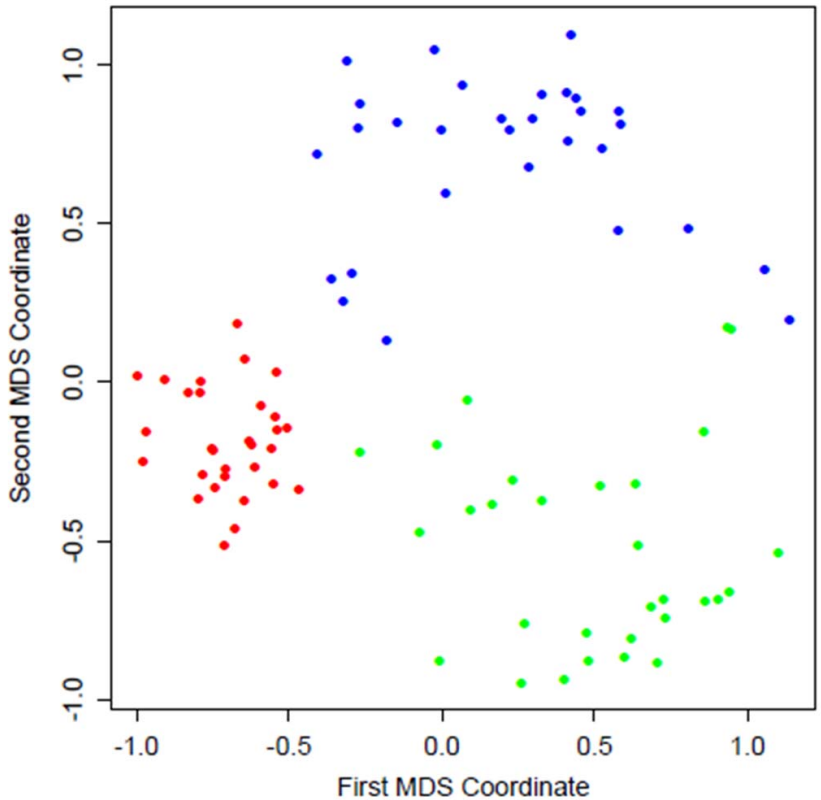
- Note that the only thing needed for the MDS is the dissimilarity matrix D. In general, different dissimilarity measures can give different MDS solutions
- The "classical" MDS uses the Euclidean distance, which gives results somewhat similar to PCA (though the objective is different). Classical MDS is also referred to as Principal Coordinate Analysis
  - PCA: finds a lower-dimensional representation that maximizes the variance
  - MDS: finds a lower-dimensional representation that best preserves the distances between the points
- We can use any other objective function or distance; this defines the general class of metric MDS

MDS: Some comments

- Note that the only thing needed for the MDS is the dissimilarity matrix D. In general, different dissimilarity measures can give different MDS solutions
- The "classical" MDS uses the Euclidean distance, which gives results somewhat similar to PCA (though the objective is different). Classical MDS is also referred to as Principal Coordinate Analysis
  - PCA: finds a lower-dimensional representation that maximizes the variance
  - MDS: finds a lower-dimensional representation that best preserves the distances between the points
- We can use any other objective function or distance; this defines the general class of metric MDS
- We can also use rankings instead of distances which gives rise to *non-metric* MDS

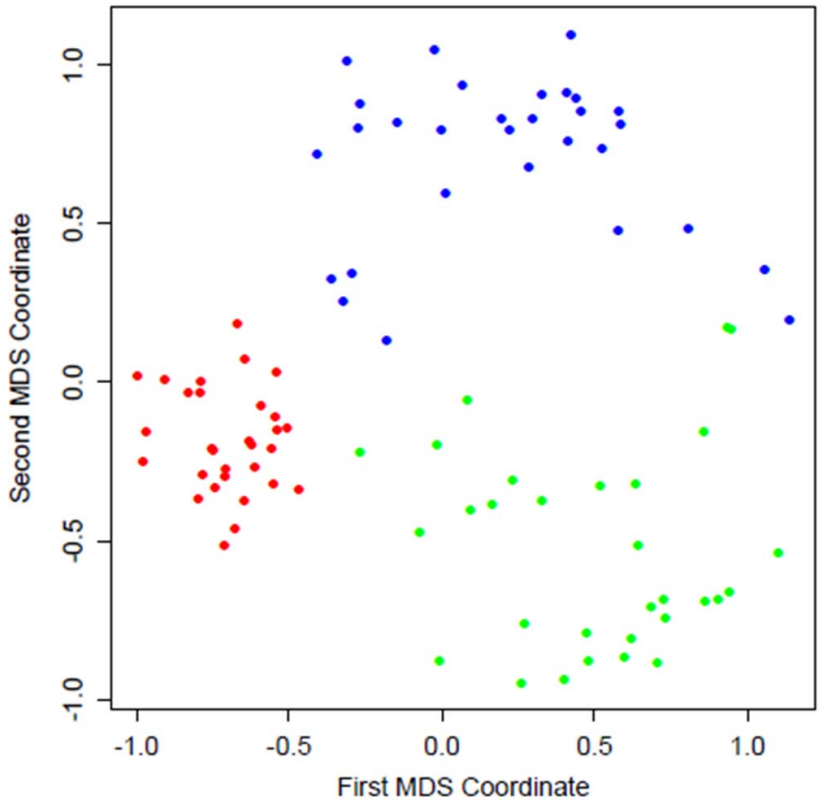
MDS Applied to the Half-Sphere Data

In the new space, the points are well separated from each other:



MDS Applied to the Half-Sphere Data

In the new space, the points are well separated from each other:



MDS can be very useful if we believe nonlinear distances represent the data better, for instance in ecology, where we can incorporate the phylogenetic tree to define more informative distances

MDS for the `USArrests` Data

MDS for the `USArrests` Data

- Recall that MDS only uses a dissimilarity matrix, so given the data, we need to create one.

MDS for the `USArrests` Data

- Recall that MDS only uses a dissimilarity matrix, so given the data, we need to create one.
- The function `daisy()` in the `R`-package `cluster` has a couple of options to get a distance/dissimilarity matrix.

MDS for the `USArrests` Data

- Recall that MDS only uses a dissimilarity matrix, so given the data, we need to create one.
- The function `daisy()` in the `R`-package `cluster` has a couple of options to get a distance/dissimilarity matrix.
- The default is Euclidean (or ℓ_2) distance: $d(i,j) = \sqrt{\sum_{k=1}^{p} (X_{ik} - X_{jk})^2}$

MDS for the `USArrests` Data

- Recall that MDS only uses a dissimilarity matrix, so given the data, we need to create one.
- The function `daisy()` in the `R`-package `cluster` has a couple of options to get a distance/dissimilarity matrix.
- The default is Euclidean (or ℓ_2) distance: $d(i,j) = \sqrt{\sum_{k=1}^{p} (X_{ik} - X_{jk})^2}$
- Can also use the "Manhattan" (or ℓ_1) distance: $d(i,j) = \sum_{k=1}^{p} |X_{ik} - X_{jk}|$.

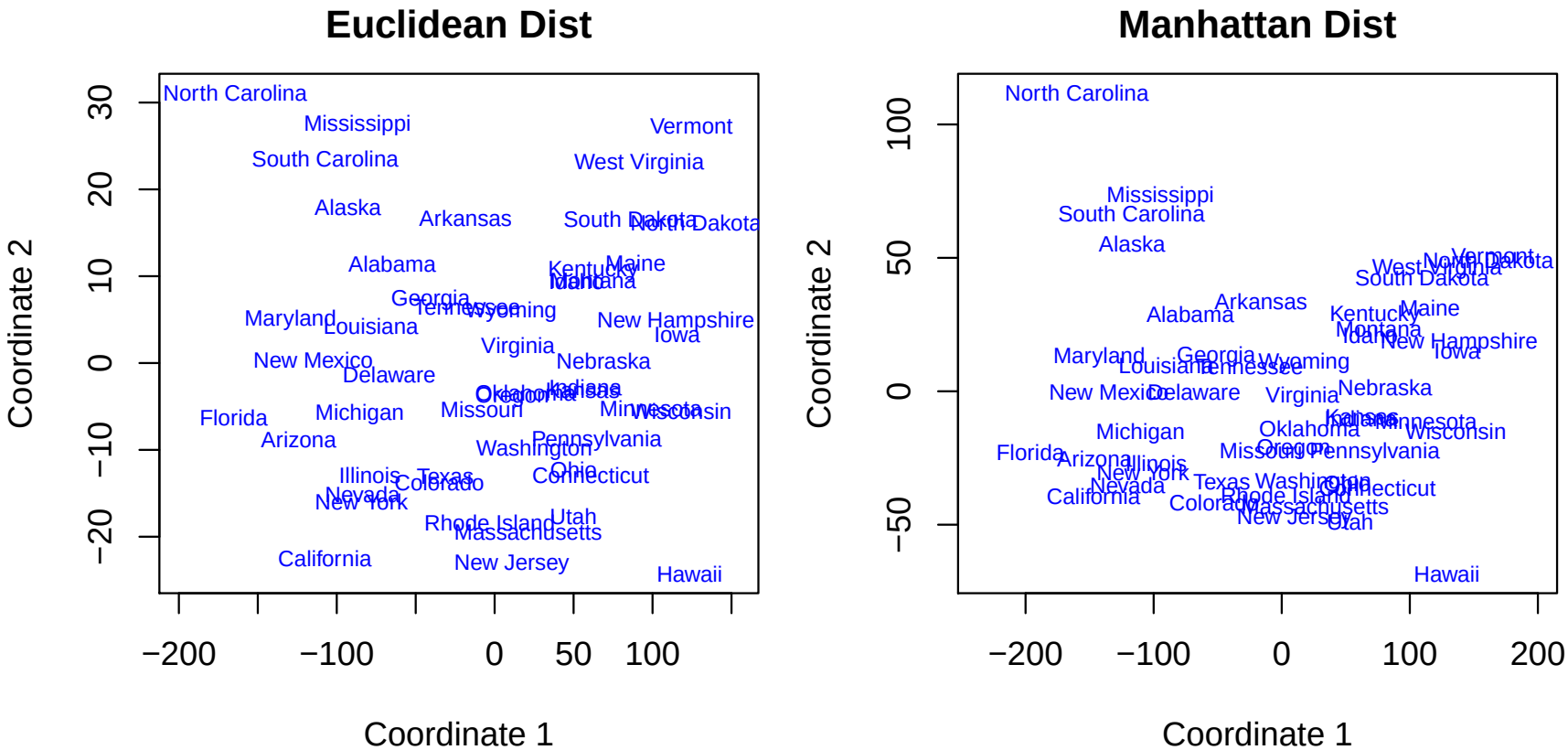
MDS for the `USArrests` Data

- Recall that MDS only uses a dissimilarity matrix, so given the data, we need to create one.
- The function `daisy()` in the `R`-package `cluster` has a couple of options to get a distance/dissimilarity matrix.
- The default is Euclidean (or ℓ_2) distance: $d(i,j) = \sqrt{\sum_{k=1}^{p} (X_{ik} - X_{jk})^2}$
- Can also use the "Manhattan" (or ℓ_1) distance: $d(i,j) = \sum_{k=1}^{p} |X_{ik} - X_{jk}|$.
- The function `cmdscale()` gives the "classic" MDS; `isoMDS()` gives a non-metric (rank-based) MDS

MDS for the `USArrests` Data

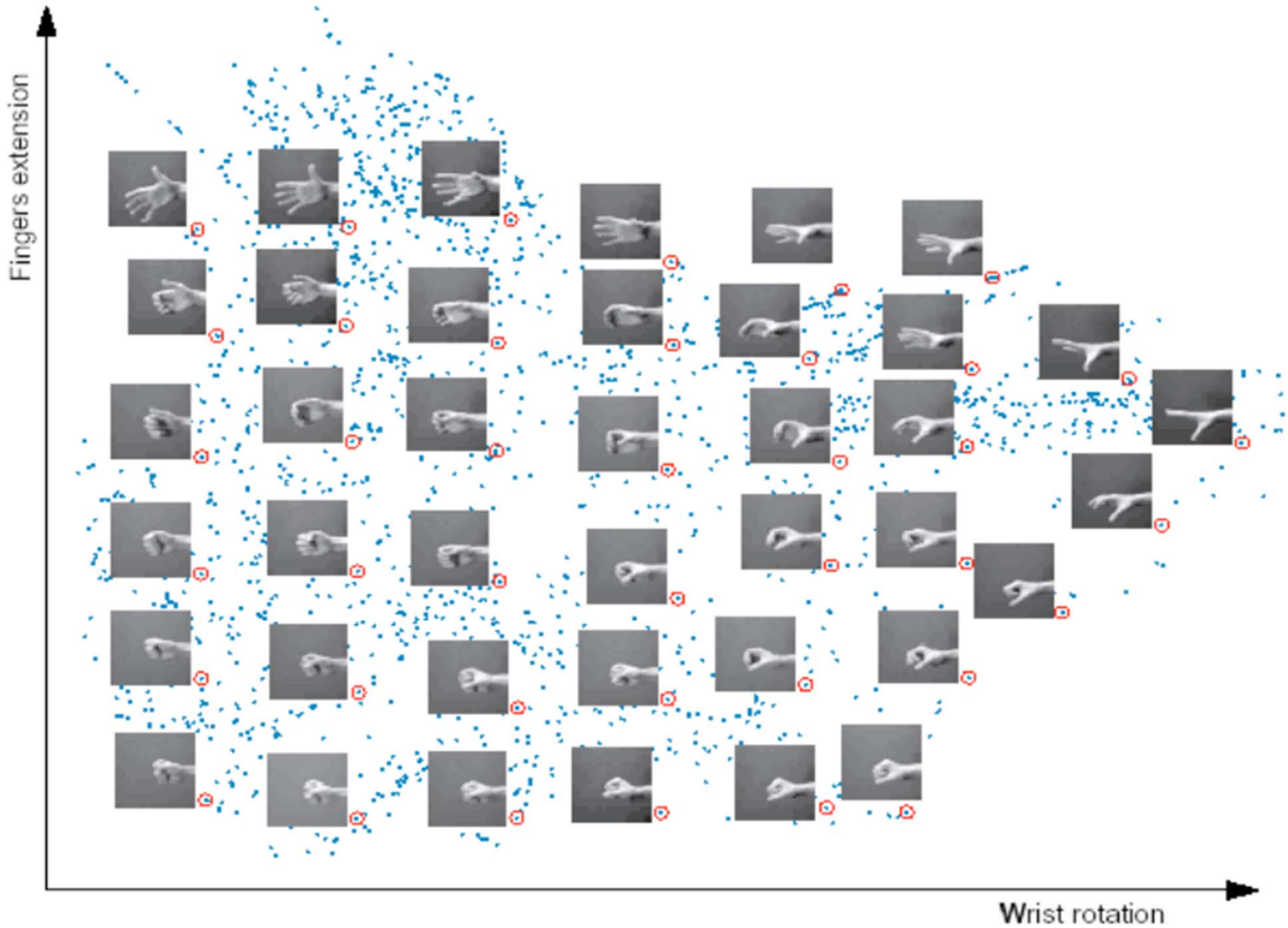
```
> dist.1=daisy(USArrests); mds.1=cmdscale(dist.1)
> dist.2=daisy(USArrests, metric="manhattan"); mds.2=cmdscale(dist.2)
>
> x=mds.1[,1]
> y=mds.1[,2]
>
> plot(x, y, xlab = "Coordinate 1", ylab = "Coordinate 2",
xlim = range(x)*1.2, type = "n", main='Euclidean Dist')
> text(x, y, labels = rownames(USArrests), col='blue', cex=0.7)
```

MDS for the `USArrests` Data



`isoMDS` gives slightly better results in this case!

An Interesting Application



From *Tenenbaum et. al.*

Remarks

Remarks

- MDS belongs to the class of nonlinear dimension reduction techniques.

Remarks

- MDS belongs to the class of nonlinear dimension reduction techniques.
- A number of other methods for linear and nonlinear dimension reduction have been proposed.

Remarks

- MDS belongs to the class of nonlinear dimension reduction techniques.
- A number of other methods for linear and nonlinear dimension reduction have been proposed.
- The main challenge in unsupervised learning is that we don't know whether a given methods has performed well in our dataset. This becomes even more challenging in high dimensional settings.

Remarks

- MDS belongs to the class of nonlinear dimension reduction techniques.
- A number of other methods for linear and nonlinear dimension reduction have been proposed.
- The main challenge in unsupervised learning is that we don't know whether a given methods has performed well in our dataset. This becomes even more challenging in high dimensional settings.
- More importantly, even if we can validate the performance of the method in our dataset, there is no guarantee that the method would work on similar datasets, or on subsequent data gathered for our study.

Remarks

- MDS belongs to the class of nonlinear dimension reduction techniques.
- A number of other methods for linear and nonlinear dimension reduction have been proposed.
- The main challenge in unsupervised learning is that we don't know whether a given methods has performed well in our dataset. This becomes even more challenging in high dimensional settings.
- More importantly, even if we can validate the performance of the method in our dataset, there is no guarantee that the method would work on similar datasets, or on subsequent data gathered for our study.
- Therefore, it is important to be cautious about interpreting the results of unsupervised learning; it is often easy to get excited with some aspects of the study!