MS1b Statistical Data Mining

Yee Whye Teh
Department of Statistics
Oxford

http://www.stats.ox.ac.uk/~teh/datamining.html

Outline

Outline

Course Structure

Lectures

- Wednesdays 1100-1200, Weeks 1-8.
- Thursdays 1100-1200, Weeks 1,3,5,7.

Problem Sheets

- 7 problem sheets: due Mondays at noon, Weeks 2-8.

Part C students

- Practical classes: Thursdays 1100-1200, Weeks 2,4,6,8.
- Problem classes: Wednesdays time to be decided, Weeks 2-8.

MSc students

- Miniproject: over Easter break.

Outline

Syllabus I

Part I: Dimensionality Reduction

- Principal Components Analysis
- Multidimensional Scaling
- Isomap

Part II: Clustering

- Hierarchical clustering
- K-means
- Vector Quantization
- Mixture Models
- Probabilistic Latent Variable Models and EM algorithm

Part III: Classification and Regression

- Empirical Risk Minimization
- Nearest Neighbours, Prototype Based Methods
- Classification and Regression Trees
- Linear Regression

Syllabus II

- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Naive Bayes
- Bayesian Methods
- Logistic Regression
- Neural Networks

Part IV: Ensemble Methods

- Bootstrap, Bagging
- Random Forests
- Boosting

R

- Learning how to use R for Data Mining

Outline

What is Data Mining?

Traditional Problems in Applied Statistics

Well formulated question that we would like to answer.
Expensive to gathering data and/or expensive to do computation.
Create specially designed experiments to collect high quality data.

Current Situation

Information Revolution

- improvements in data storage devices (both larger and cheaper).
- powerful data capturing devices (bioassays, microphones, cameras, satellites).

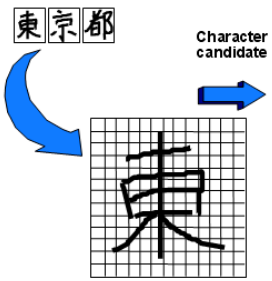
$\rightarrow$ lots of data with potentially valuable information available.
$\rightarrow$ Big Data....

What is Data Mining?

- To gain insight from data.
- Often working with huge datasets.
  - Typically many variables (up to thousands or millions).
  - Often, but not always many observations (dozens to millions).
- Secondary data sources possibly collected for other purposes.
- Uncurated data, missing data, unstructured data, multi-aspect data.
- Gain understanding without specific goals.

Applications of Data Mining

- Pattern Recognition



  - Sorting Cheques
  - Reading License Plates
  - Sorting Envelopes
  - Eye/ Face/ Fingerprint Recognition

Applications of Data Mining

- Business applications
  - Help companies intelligently find information
  - Credit scoring
  - Predict which products people are going to buy
  - Recommender systems
  - Autonomous trading
- Scientific applications
  - Predict cancer occurence/type and health of patients/personalized health
  - Make sense of complex physical, biological, ecological, sociological models

...It is just a nice name for multivariate statistics (‘minus model checking’).

NY Times: Data Mining in Walmart (URL)

November 14, 2004

What Wal-Mart Knows About Customers' Habits

By CONSTANCE L. HAYS

HURRICANE FRANCES was on its way, barreling across the Caribbean, threatening a direct hit on Florida's Atlantic coast. Residents made for higher ground, but far away, in Bentonville, Ark., executives at Wal-Mart Stores decided that the situation offered a great opportunity for one of their newest data-driven weapons, something that the company calls predictive technology.

A week ahead of the storm's landfall, Linda M. Dillman, Wal-Mart's chief information officer, pressed her staff to come up with forecasts based on what had happened when Hurricane Charley struck several weeks earlier. Backed by the trillions of bytes' worth of shopper history that is stored in Wal-Mart's computer network, she felt that the company could "start predicting what's going to happen, instead of waiting for it to happen," as she put it.

The experts mined the data and found that the stores would indeed need certain products - and not just the usual flashlights. "We didn't know in the past that strawberry Pop-Tarts increase in sales, like seven times their normal sales rate, ahead of a hurricane," Ms. Dillman said in a recent interview. "And the pre-hurricane top-selling item was beer."

Thanks to those insights, trucks filled with toaster pastries and six-packs were soon speeding down Interstate 95 toward Wal-Marts in the path of Frances. Most of the products that were stocked for the storm sold quickly, the company said.

August 6, 2009

For Today's Graduate, Just One Word: Statistics

By STEVE LOHR

MOUNTAIN VIEW, Calif. — At Harvard, Carrie Grimes majored in anthropology and archaeology and ventured to places like Honduras, where she studied Mayan settlement patterns by mapping where artifacts were found. But she was drawn to what she calls "all the computer and math stuff" that was part of the job.

"People think of field archaeology as Indiana Jones, but much of what you really do is data analysis," she said.

Now Ms. Grimes does a different kind of digging. She works at Google, where she uses statistical analysis of mounds of data to come up with ways to improve its search engine.

Ms. Grimes is an Internet-age statistician, one of many who are changing the image of the profession as a place for dronish number nerds. They are finding themselves increasingly in demand — and even cool.

"I keep saying that the sexy job in the next 10 years will be statisticians," said Hal Varian, chief economist at Google. "And I'm not kidding."

The rising stature of statisticians, who can earn $125,000 at top companies in their first year after getting a doctorate, is a byproduct of the recent explosion of digital data. In field after field, computing and the Web are creating new realms of data to explore — sensor signals, surveillance tapes, social network chatter, public records and more. And the digital data surge only promises to accelerate, rising fivefold by

January 7, 2009

Data Analysts Captivated by R's Power

By ASHLEE VANCE

To some people R is just the 18th letter of the alphabet. To others, it's the rating on racy movies, a measure of an attic's insulation or what pirates in movies say.

R is also the name of a popular programming language used by a growing number of data analysts inside corporations and academia. It is becoming their lingua franca partly because data mining has entered a golden age, whether being used to set ad prices, find new drugs more quickly or fine-tune financial models. Companies as diverse as Google, Pfizer, Merck, Bank of America, the InterContinental Hotels Group and Shell use it.

But R has also quickly found a following because statisticians, engineers and scientists without computer programming skills find it easy to use.

"R is really important to the point that it's hard to overvalue it," said Daryl Pregibon, a research scientist at Google, which uses the software widely. "It allows statisticians to do very intricate and complicated analyses without knowing the blood and guts of computing systems."

It is also free. R is an open-source program, and its popularity reflects a shift in the type of software used inside corporations. Open-source software is free for anyone to use and modify. I.B.M., Hewlett-Packard and Dell make billions of dollars a year selling servers that run the open-source Linux operating system.

Types of Data Mining

Unsupervised Learning

'Unclassified' data from which we would like to uncover hidden 'structure' or groupings

- Given detailed phone usage from many people, find interesting groups of people with similar behaviour.
- Shopping habits for people using loyalty cards: find groups of 'similar' shoppers.
- Given expression measurements of 1000s of genes for 100s of patients, find groups of functionally similar genes.

Goal: Hypothesis generation, visualization.

Types of Data Mining

Supervised Learning

A database of 'classified' examples with predefined groupings

- Given detailed phone usage of many users *along with their historic churn*, predict when/if people are going to change contracts again.
- Given expression measurements of 1000s of genes for 100s of patients *along with a binary variable indicating absence or presence of a specific cancer*, predict if the cancer is present for a new patient.
- Given expression measurements of 1000s of genes for 100s of patients *along with survival length*, predict survival time.

Goal: Prediction.

Further Readings

- Leo Breiman: Statistical Modeling: The Two Cultures (URL)
- NY Times: Big Data's Impact In the World (URL)
- Economist: Data, Data Everywhere (URL)
- McKinsey: Big data: The Next Frontier for Competition (URL)

Other recent news on Big Data, Data Mining, Machine Learning:

- New York Times: Sure, Big Data Is Great. But So Is Intuition (URL)
- New York Times: How Many Computers to Identify a Cat? 16,000 (URL)
- New York Times: Scientists See Promise in Deep-Learning Programs (URL)
- New Yorker: Is "Deep Learning" a Revolution in Artificial Intelligence? (URL)

Outline

Outline

Notation

- Data consists of p measurements (variables/attributes) on n examples (observations/cases)
- X is a $n \times p$-matrix with $X_{ij} :=$ the j-th measurement for the i-th example

$$X = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1j} & \dots & X_{1p} \\ X_{21} & X_{22} & \dots & X_{2j} & \dots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ X_{i1} & X_{i2} & \dots & X_{ij} & \dots & X_{ip} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{nj} & \dots & X_{np} \end{bmatrix}$$

Crabs Data ($n = 200$, $p = 5$)

Campbell (1974) studied rock crabs of the genus *leptograpsus*. One species, *L. variegatus*, had been split into two new species, previously grouped by colour, orange and blue. Preserved specimens lose their colour, so it was hoped that morphological differences would enable museum material to be classified.

Data are available on 50 specimens of each sex of each species, collected on sight at Fremantle, Western Australia. Each specimen has measurements on the width of the frontal lip `FL`, the rear width `RW`, and length along the midline `CL` and the maximum width `CW` of the carapace, and the body depth `BD` in mm.

Crabs Data

Looking at the crabs dataset, $n = 200$ measurements on $p = 5$ morphological features of crabs

- 'FL' frontal lip size (mm)
- 'RW' rear width (mm)
- 'CL' carapace length (mm)
- 'CW' carapace width (mm)
- 'BD' body depth (mm)

Also available, the colour ('B' or 'O') and sex ('M' or 'F').

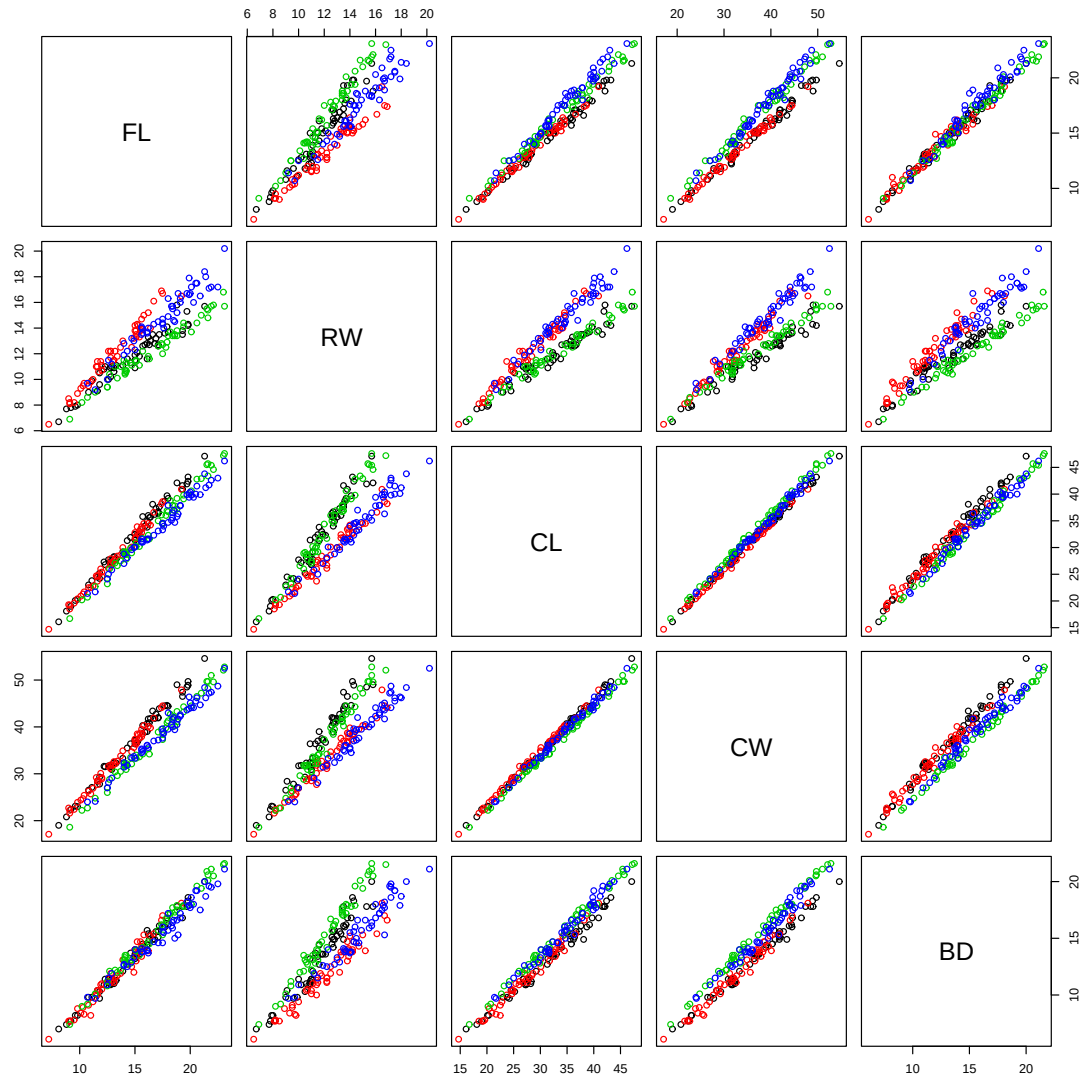
```
## load package MASS containing the data
library(MASS)
## look at data
crabs
```

```
     sp sex index   FL   RW   CL   CW   BD
1    B   M      1    8.1  6.7 16.1 19.0  7.0
2    B   M      2    8.8  7.7 18.1 20.8  7.4
...
```

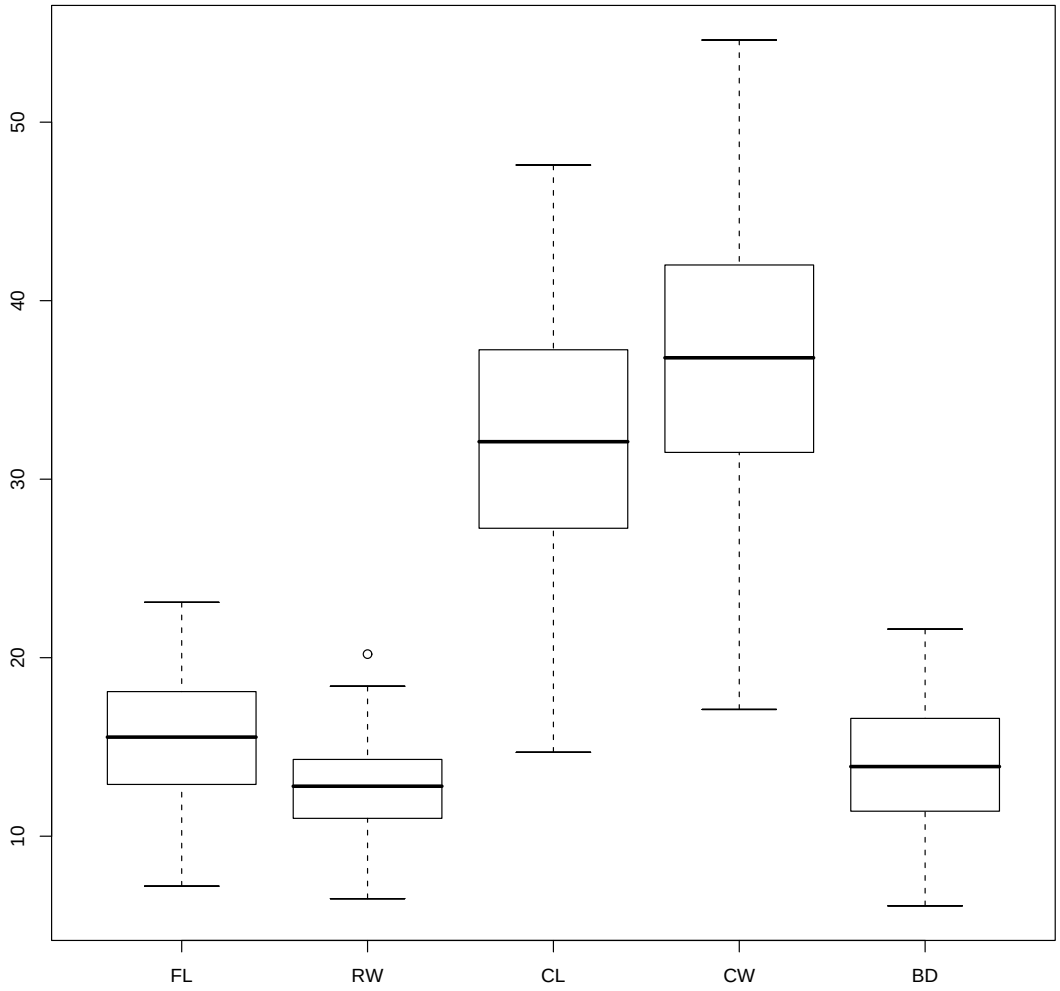
R code

```
## assign predictor and class variables
Crabs <- crabs[,4:8]
Crabs.class <- factor(paste(crabs[,1],crabs[,2],sep=""))

## plot data using pair plots
plot(Crabs,col=unclass(Crabs.class))

##boxplots
boxplot(Crabs)



## parallel coordinates
parcoord(Crabs)
```

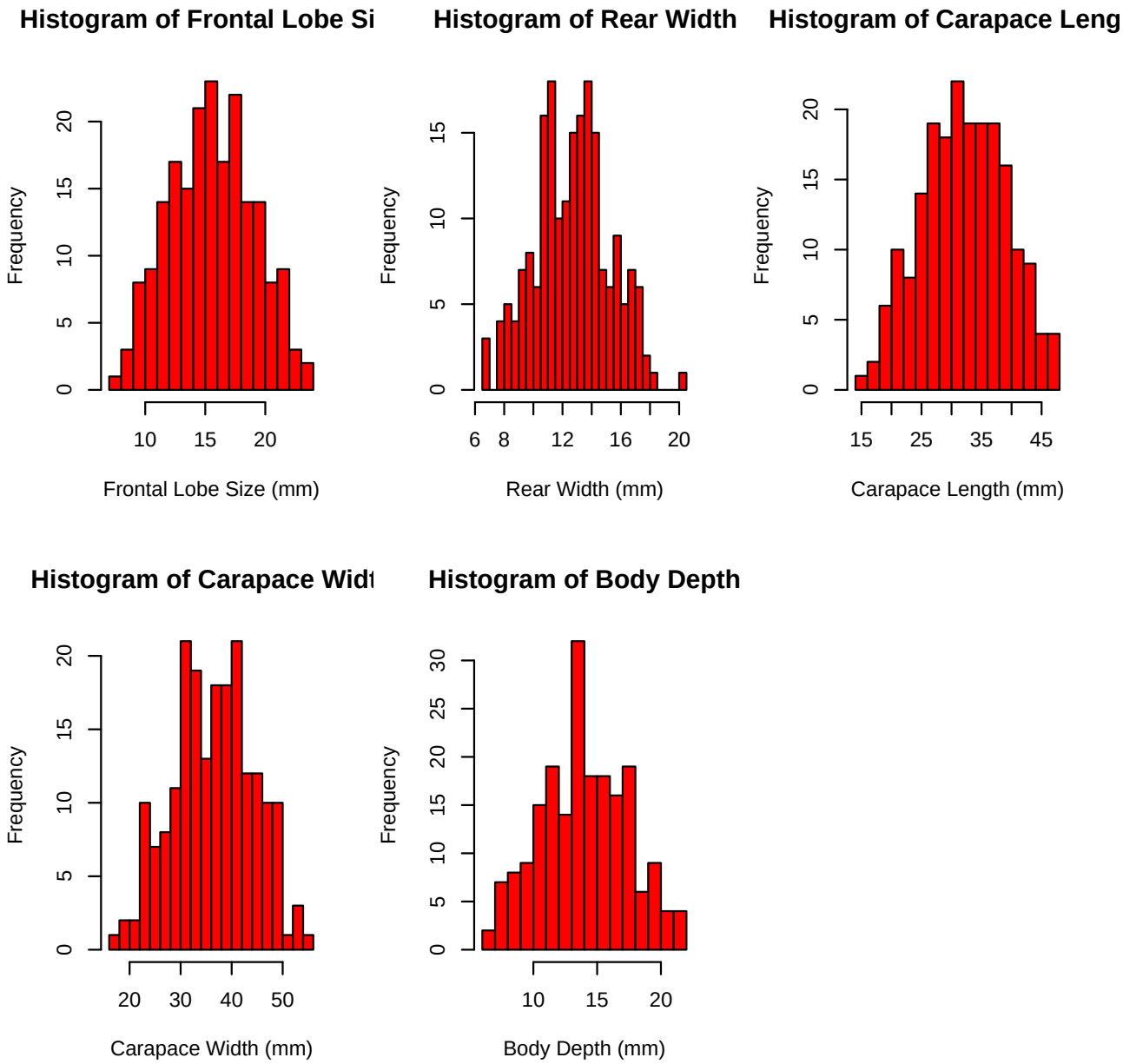
Simple Pairwise Scatterplots



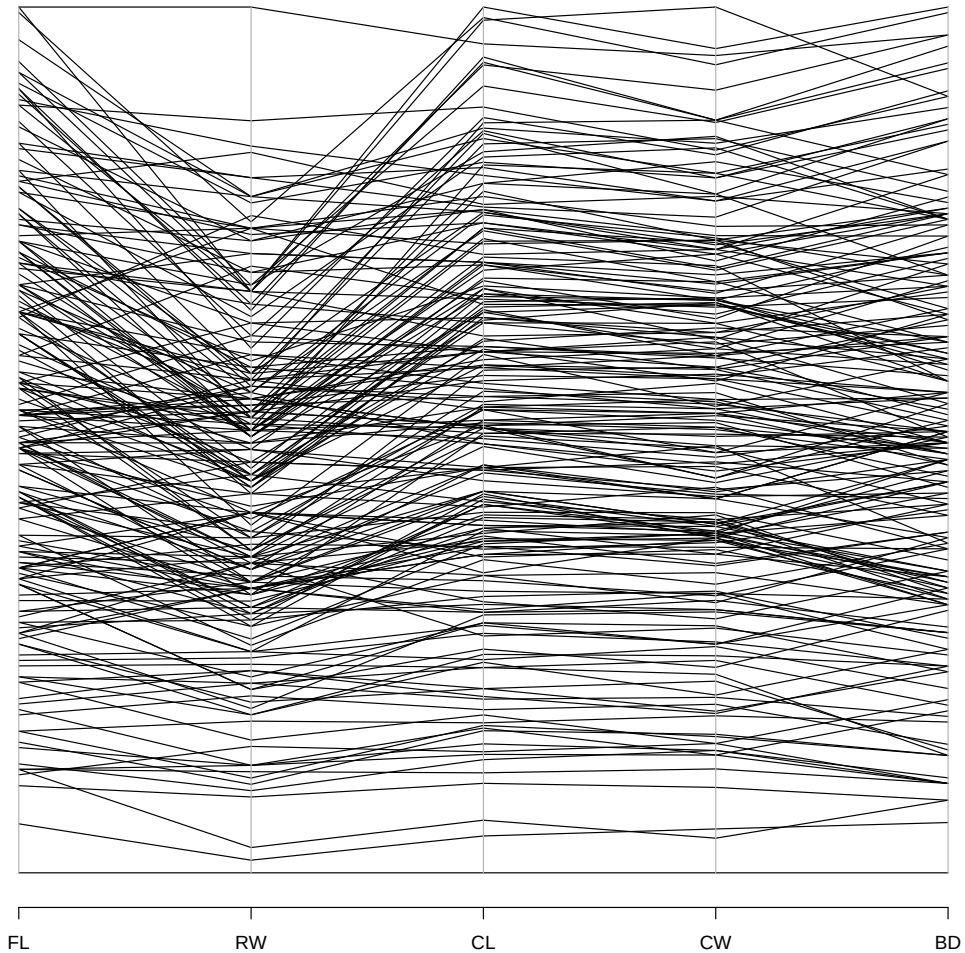
Univariate Boxplots



Univariate Histograms



Parallel Coordinate Plots



These summary plots are helpful, but do not really help very much if the dimensionality of the data is high (a few dozen or thousands).
Possible approaches for higher-dimensional problems.

- We are constrained to view data in 2 or 3 dimensions
- Look for 'interesting' projections of X into lower dimensions
- Hope that for large p, considering only $k \ll p$ dimensions is just as informative

Outline

Principal Components Analysis (PCA)

- Seek to rotate data to a new basis that represents the data in a more 'interesting' way.
- PCA considers interesting to be directions with greatest *variance*.
- Builds up an orthogonal basis where new basis vectors are chosen to explain the greatest variance in data, the first few PCs should represent most of the variance-covariance structure in the data, i.e. the subspace spanned by first k PCs represents the 'best' k-dimensional view of the data.

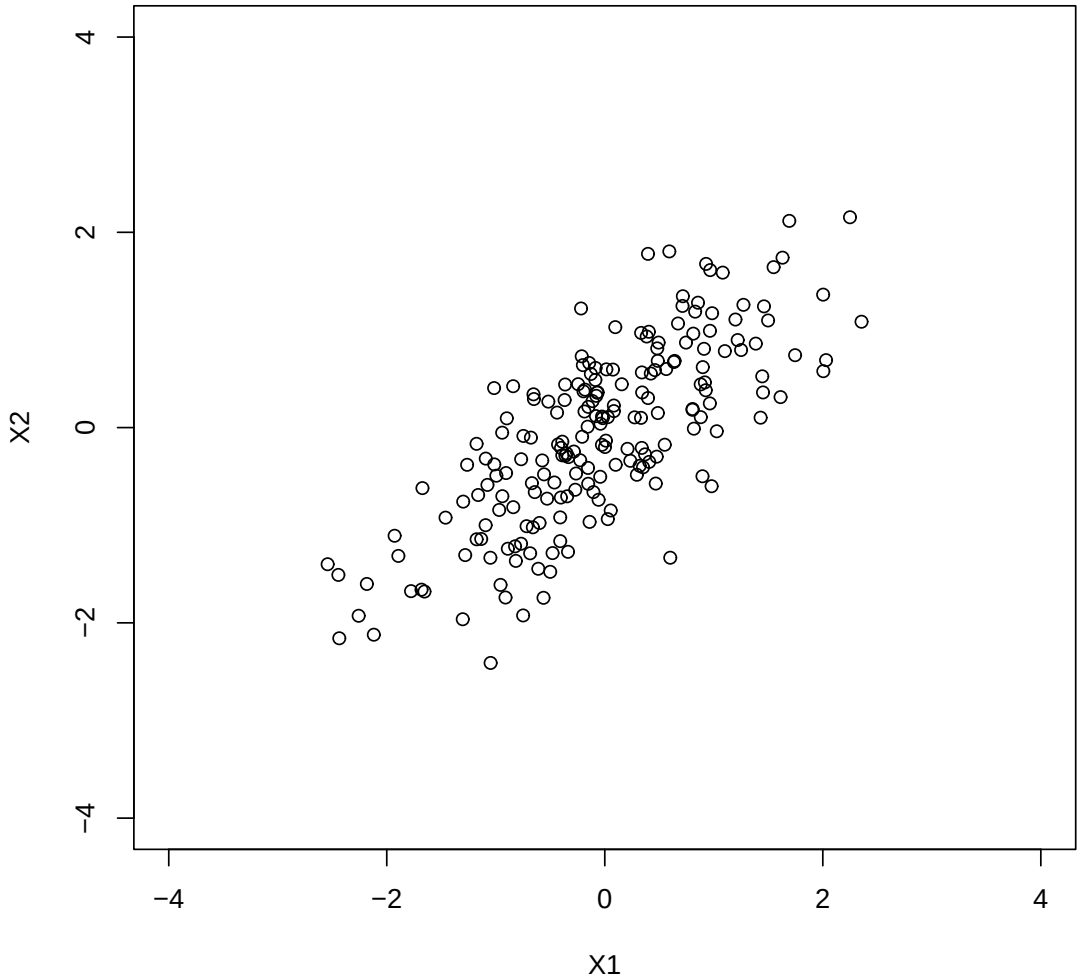
Principal Components Analysis (PCA)

- Consider a set of real-valued variables $X = (X_1 \dots X_p)^T$.
- For the 1^{st} PC, we seek a derived variable of the form

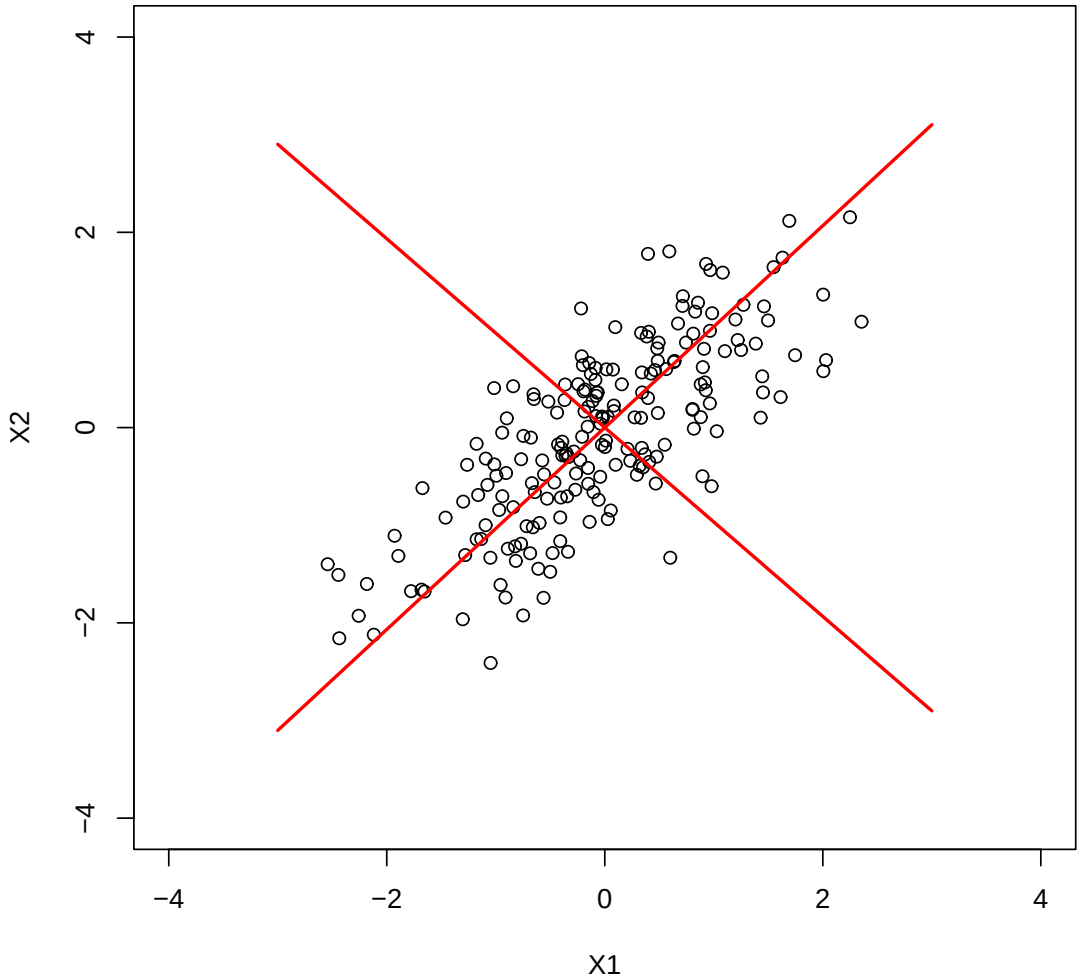
$$Z_1 = a_{11}X_1 + a_{21}X_2 + \cdots + a_{p1}X_p = X^T \mathbf{a}_1$$

  where $a_{1i} \in \mathbb{R}$ are chosen to maximise var(Z_1).
- To get a well defined problem, we fix $\mathbf{a}_1^T \mathbf{a}_1 = 1$.
- The 1^{st} PC attempts to capture the common variation in all variables using a single derived variable.
- The 2^{nd} PC Z_2 is chosen to be orthogonal with the 1^{st} and is computed in a similar way. It will have the largest variance in the remaining $p-1$ dimensions, etc.

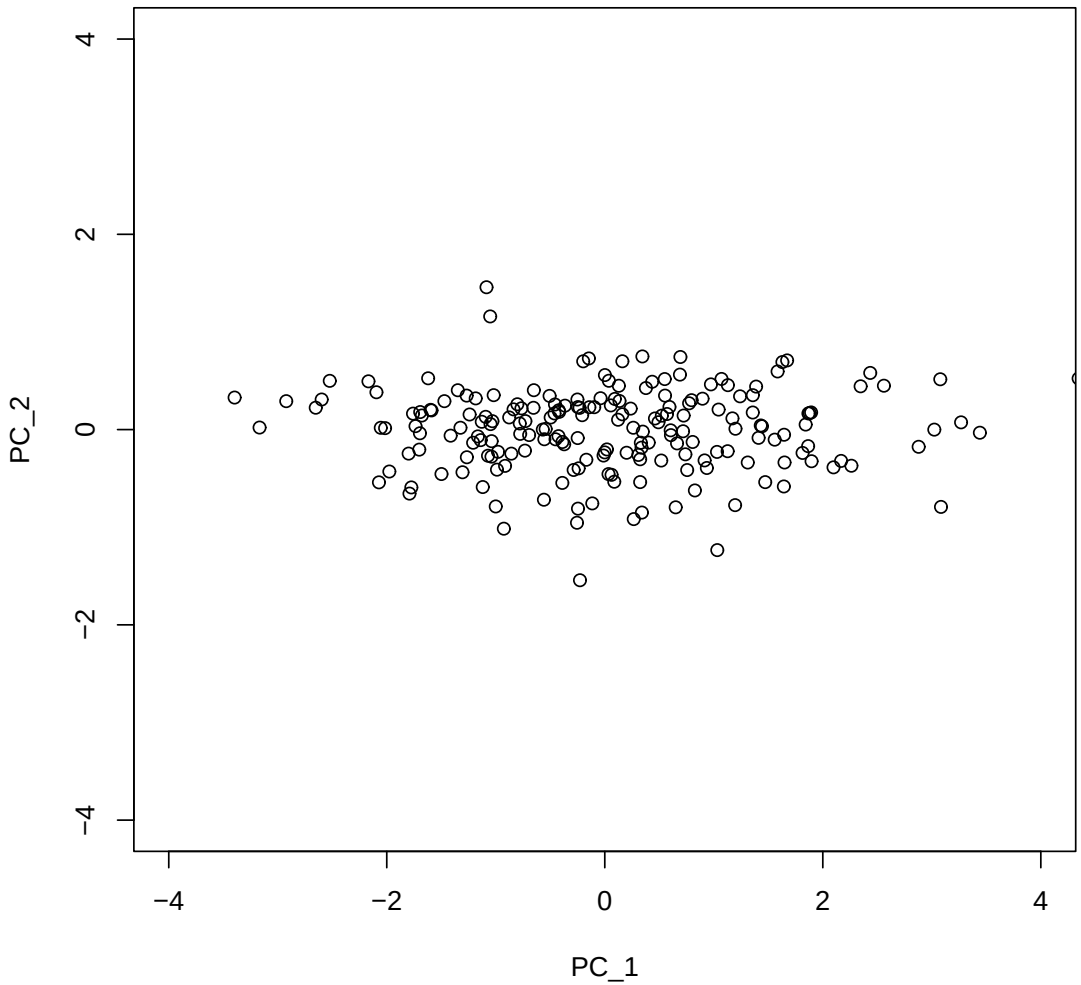
Principal Components Analysis (PCA)



Principal Components Analysis (PCA)



Principal Components Analysis (PCA)



How to Obtain the Coefficients?

To find the 1^{st} PC given by $Z_1 = X^T \mathbf{a}_1$

- Maximise $var(Z_1) = var(X\mathbf{a}_1) = \mathbf{a}_1^T cov(X)\mathbf{a}_1 \approx \mathbf{a}_1^T S\mathbf{a}_1$ subject to $\mathbf{a}_1^T\mathbf{a}_1 = 1$ where $S = n^{-1}X^TX$ is a $p \times p$ sample covariance matrix of the centred $n \times p$ data matrix X.
- Rewriting this as a constrained maximisation problem,

$$\text{Maximise } F(\mathbf{a}_1) = \mathbf{a}_1^T S\mathbf{a}_1 - \lambda_1\left(\mathbf{a}_1^T\mathbf{a}_1 - 1\right) \text{ w.r.t. } \mathbf{a}_1.$$

- The corresponding vector of partial derivatives yields

$$\frac{\partial F}{\partial \mathbf{a}_1} = 2S\mathbf{a}_1 - 2\lambda_1\mathbf{a}_1.$$

- Setting this to zero reveals the eigenvector equation, i.e. $\mathbf{a}_1$ must be an eigenvector of S and λ_1 the corresponding eigenvector.
- Since $\mathbf{a}_1^T S\mathbf{a}_1 = \lambda_1\mathbf{a}_1^T\mathbf{a}_1 = \lambda_1$, the 1^{st} PC must be the eigenvector associated with the largest eigenvalue of S.

How to Obtain the Coefficients?

How about the 2^{nd} PC?

- Proceed as before but include the additional constraint that the 2^{nd} PC must be orthogonal to the 1^{st} PC

$$\text{Maximise } F(\mathbf{a}_2) = \mathbf{a}_2^T S \mathbf{a}_2 - \lambda_2 \left(\mathbf{a}_2^T \mathbf{a}_2 - 1\right) - \mu \left(\mathbf{a}_1^T \mathbf{a}_2\right) \text{ w.r.t. } \mathbf{a}_2$$

- Solving this shows that a_2 must be the eigenvector of S associated with the 2^{nd} largest eigenvalue, and so on
- The eigenvalue decomposition of S is given by $S = A \Lambda A^T$ where Λ is a diagonal matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ and A is a $p \times p$ orthogonal matrix whose columns are the p eigenvectors of S.

Properties of the Principal Components

- PCs are *uncorrelated*

$$cov(X^T \mathbf{a}_i, X^T \mathbf{a}_j) \approx \mathbf{a}_i^T S \mathbf{a}_j = 0 \text{ for } i \neq j.$$

- The total sample variance is given by

$$\text{Total sample variance} = \sum_{i=1}^{p} s_{ii} = \lambda_1 + \ldots + \lambda_p,$$

so the proportion of total variance explained by the k^{th} PC is

$$\frac{\lambda_k}{\lambda_1 + \lambda_2 + \ldots + \lambda_p} \quad k = 1, 2, \ldots, p$$

R code

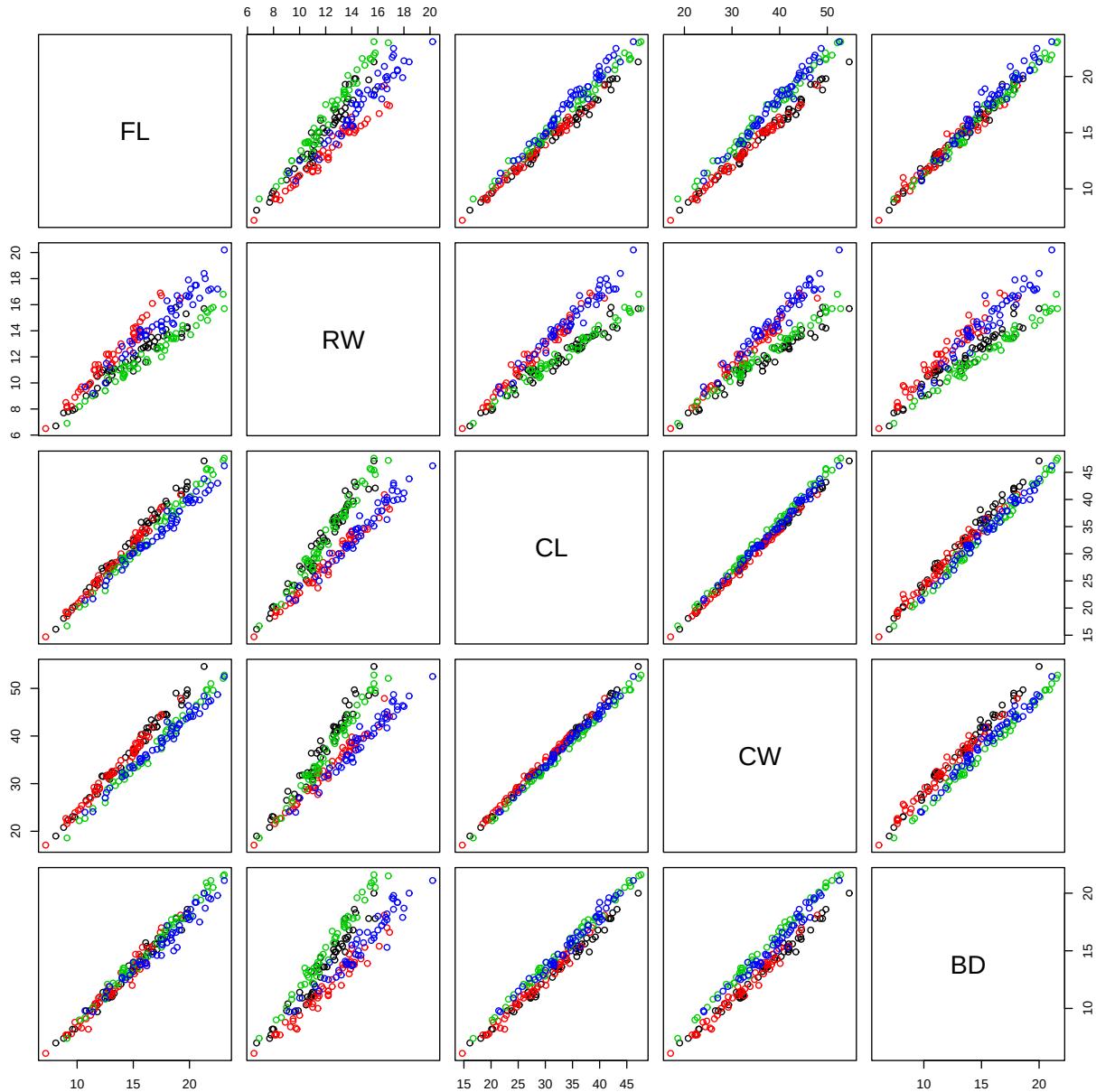
This is what we have had before:

```
library(MASS)
Crabs <- crabs[,4:8]
Crabs.class <- factor(paste(crabs[,1],crabs[,2],sep=""))
plot(Crabs,col=unclass(Crabs.class))
```

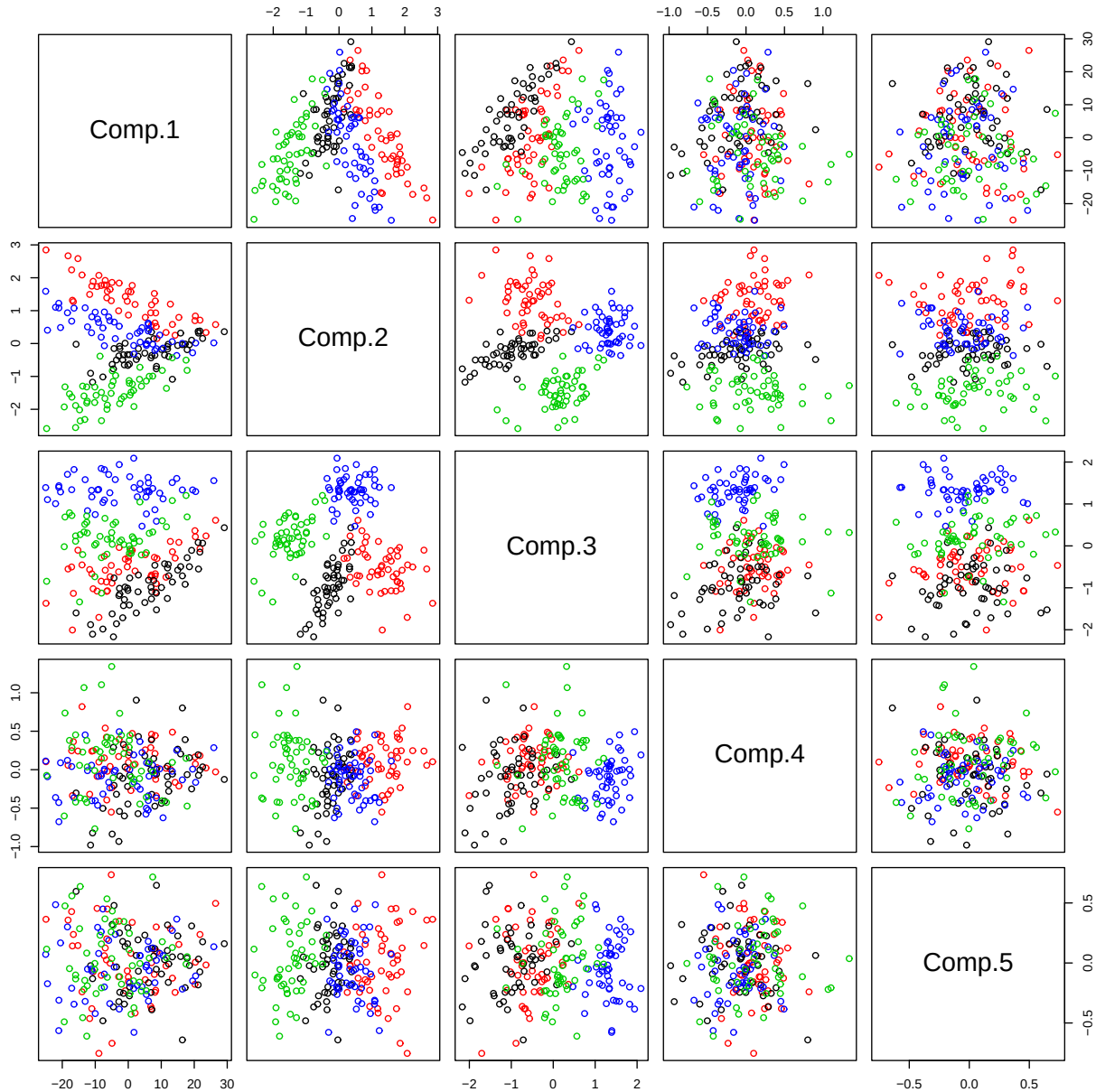
Now perform PCA analysis with function `princomp`.
Alternatively, use `eigen` or `svd` instead (later).

```
Crabs.pca <- princomp(Crabs,cor=FALSE)
plot(Crabs.pca)
pairs(predict(Crabs.pca),col=unclass(Crabs.class))
```

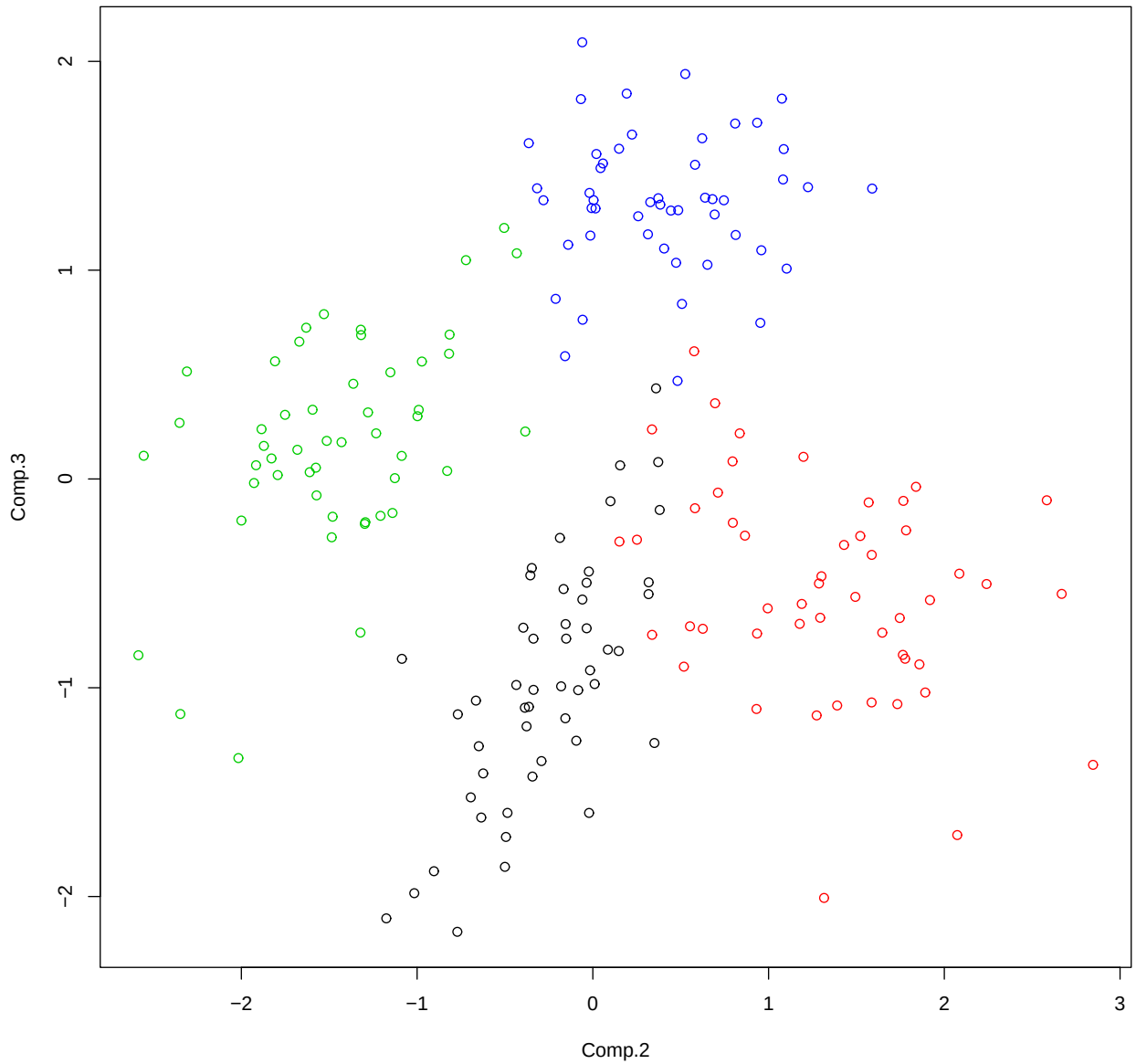
PCA Example 1: Original crabs data



PCA Example 1: Rotated crabs data



PCA Example 1: Crabs Data ($n = 200$, $p = 5$)



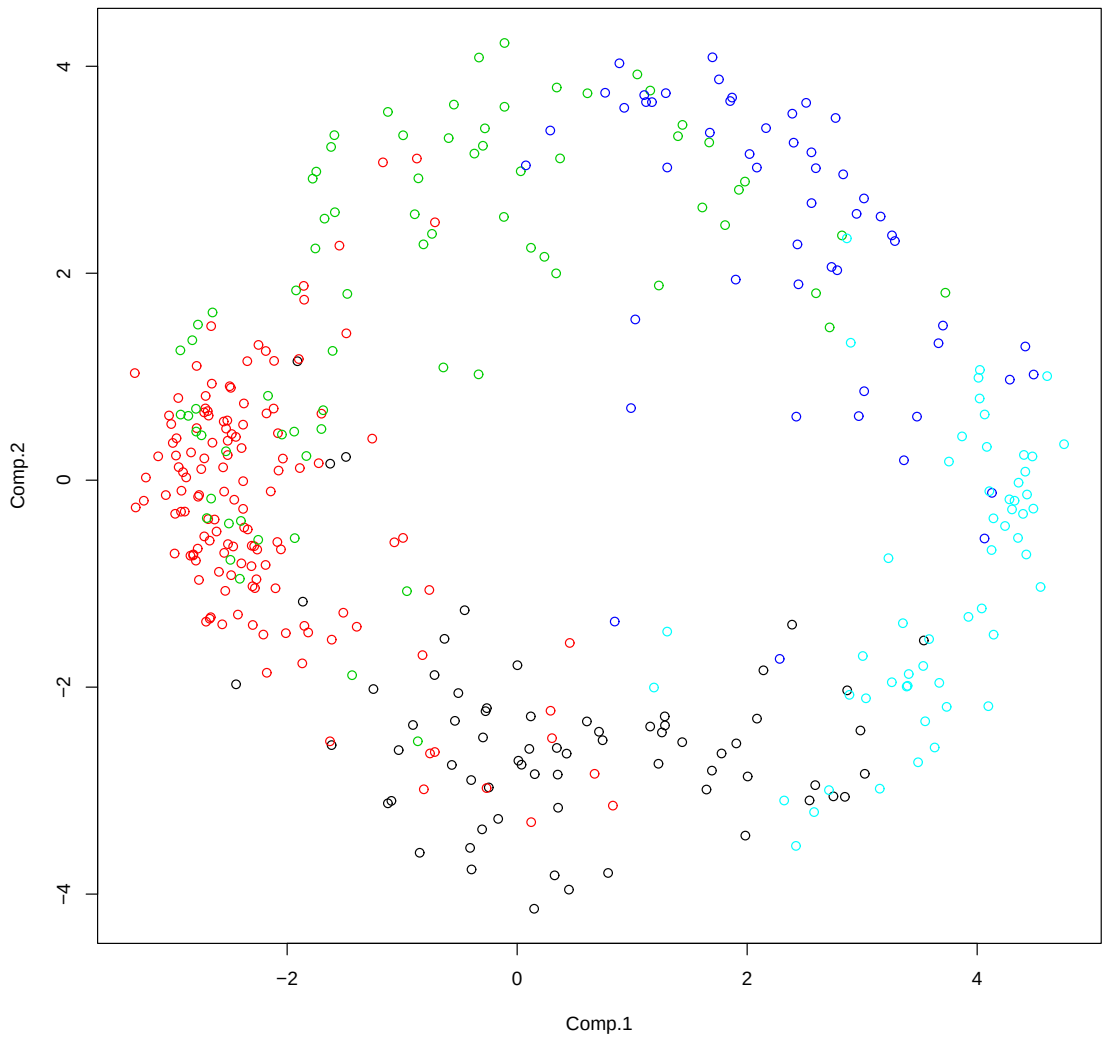
PCA Example 2: Yeast Cell Cycle Data ($n = 384$, $p = 17$)

Cho *et al* (1998) present gene expression data on the cell cycle of yeast. They identify a subset of genes that can be categorised into five different phases of the cell-cycle. Changes in expression for the genes are measured over two cell cycles (17 time points).

The data were normalised so that the expression values for each gene has mean zero and unit variance across the cell cycles.

We visualise the 384 genes in the space of the first two prinicipal components.

PCA Example 2: Yeast Cell Cycle Data ($n = 384$, $p = 17$)



Comments on the use of PCA

- PCA commonly used to project data X onto the first k PCs giving the 'best' k-dimensional view of the data.
- PCA commonly used for lossy compression of high dimensional data.
- Emphasis on variance is where the weaknesses of PCA stem from:
  - The PCs depend heavily on the units measurement. Where the data matrix contains measurements of vastly differing orders of magnitude, the PC will be greatly biased in the direction of larger measurement. It is therefore recommended to calculate PCs from $cor(X)$ instead of $cov(X)$.
  - Robustness to outliers is also an issue. Variance is affected by outliers therefore so are PCs.
- Although PCs are uncorrelated, scatterplots sometimes reveal structures in the data other than linear correlation.

Biplots

- When viewing projections of data matrix X into its PC space, it is instructive to view the contribution from the original variables to the PCs that are plot.
- Biplots overlay projection of *unit vectors* of the original variables into the PC space
- As PCs are linear combinations of the original variables, it is straightforward to invert this relationship to yield the contributions of the original variables to the PCs

Biplots

Biplots show us an *image* of the data and unit vectors of the original axes into the projected space.

- The distance of projected points away from the projected original axes tell us its original location.
- Unit vectors of the original variables give us a common denominator to compare how much weighting each PC gives to the original variables.
- It can be shown that $\cos\theta$ (where θ is the angle that subtends two projected original axes) approximates the correlation between these variables.

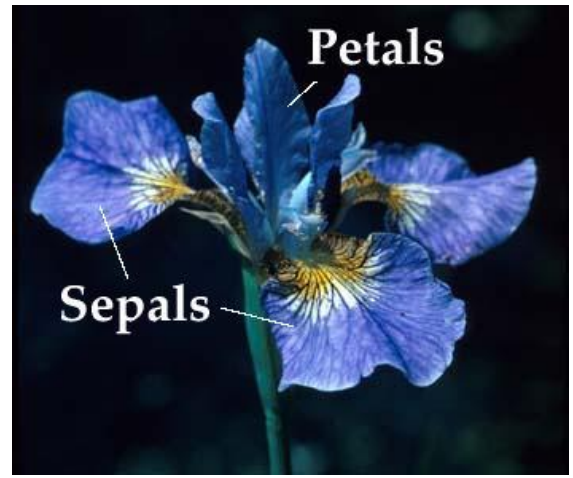
However, the quality of this image depends on the proportion of variance explained by the PCs used.

Biplot Example 1: Fisher's Iris Data

50 sample from 3 species of iris: *iris setosa*, *versicolor*, and *virginica*

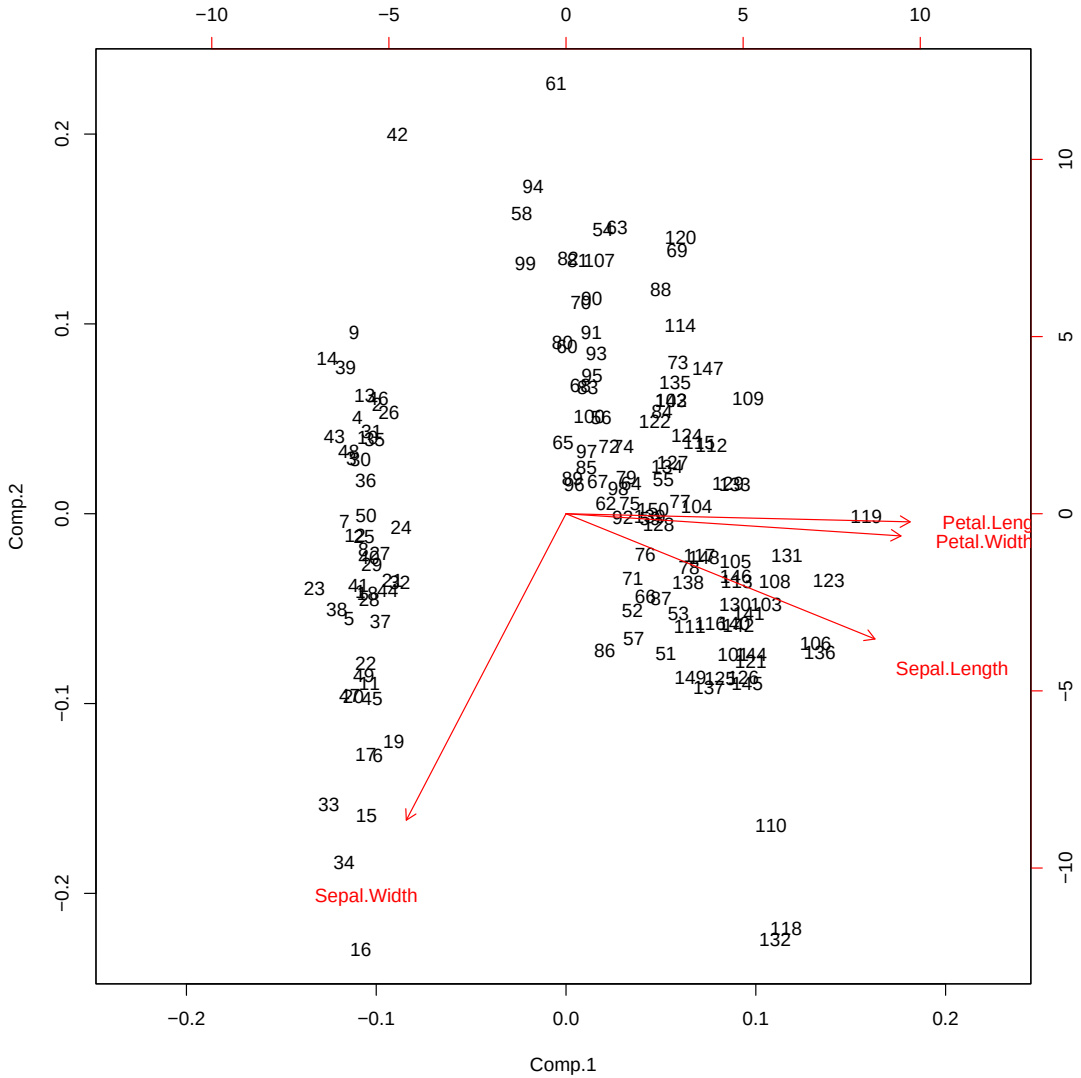
Each measuring the length and widths of both sepal and petals

Collected by E. Anderson (1935) and analysed by R.A. Fisher (1936)



Using again function `princomp` and `biplot`.

```
iris1 <- iris
iris1 <- iris1[,-5]
biplot(princomp(iris1,cor=T))
```

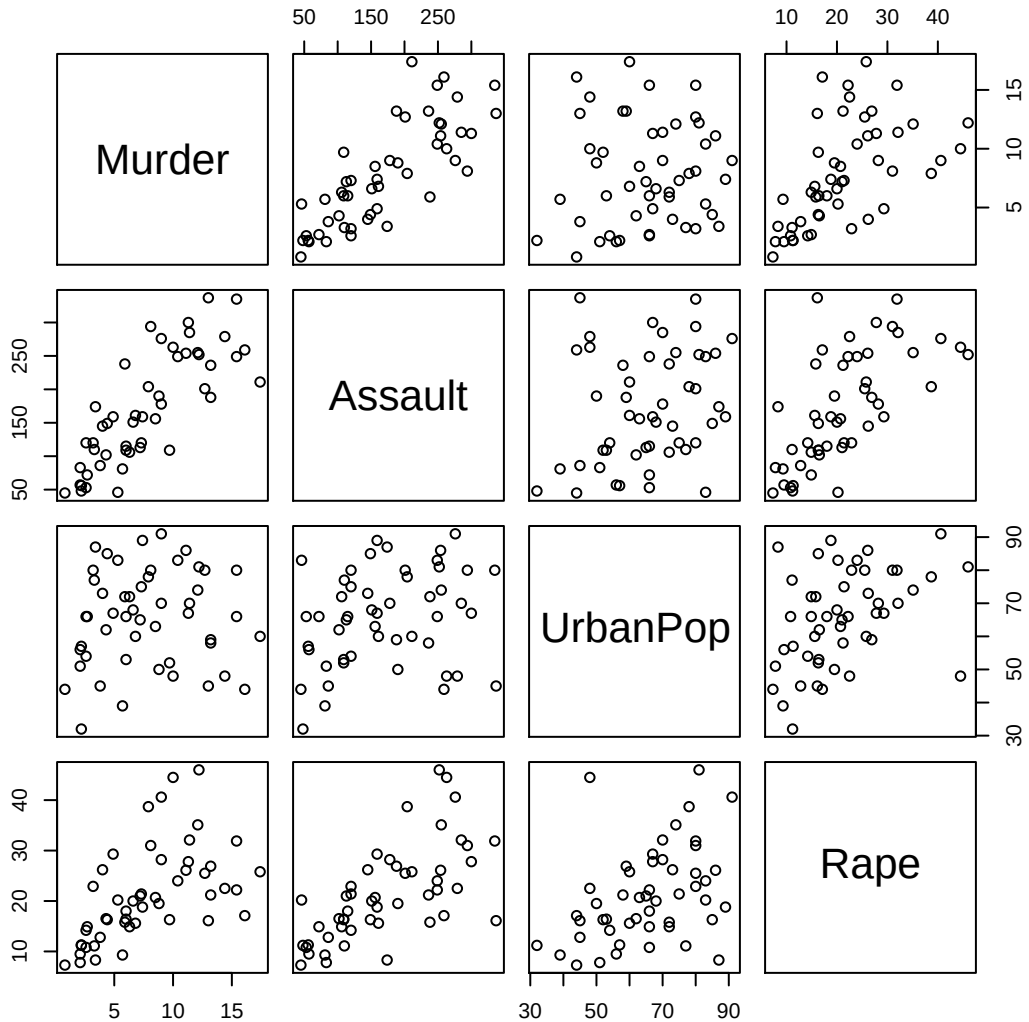

Comp.1
Comp.2
Petal.Leng
Petal.Width
Sepal.Length
Sepal.Width

Biplot Example 2: US Arrests

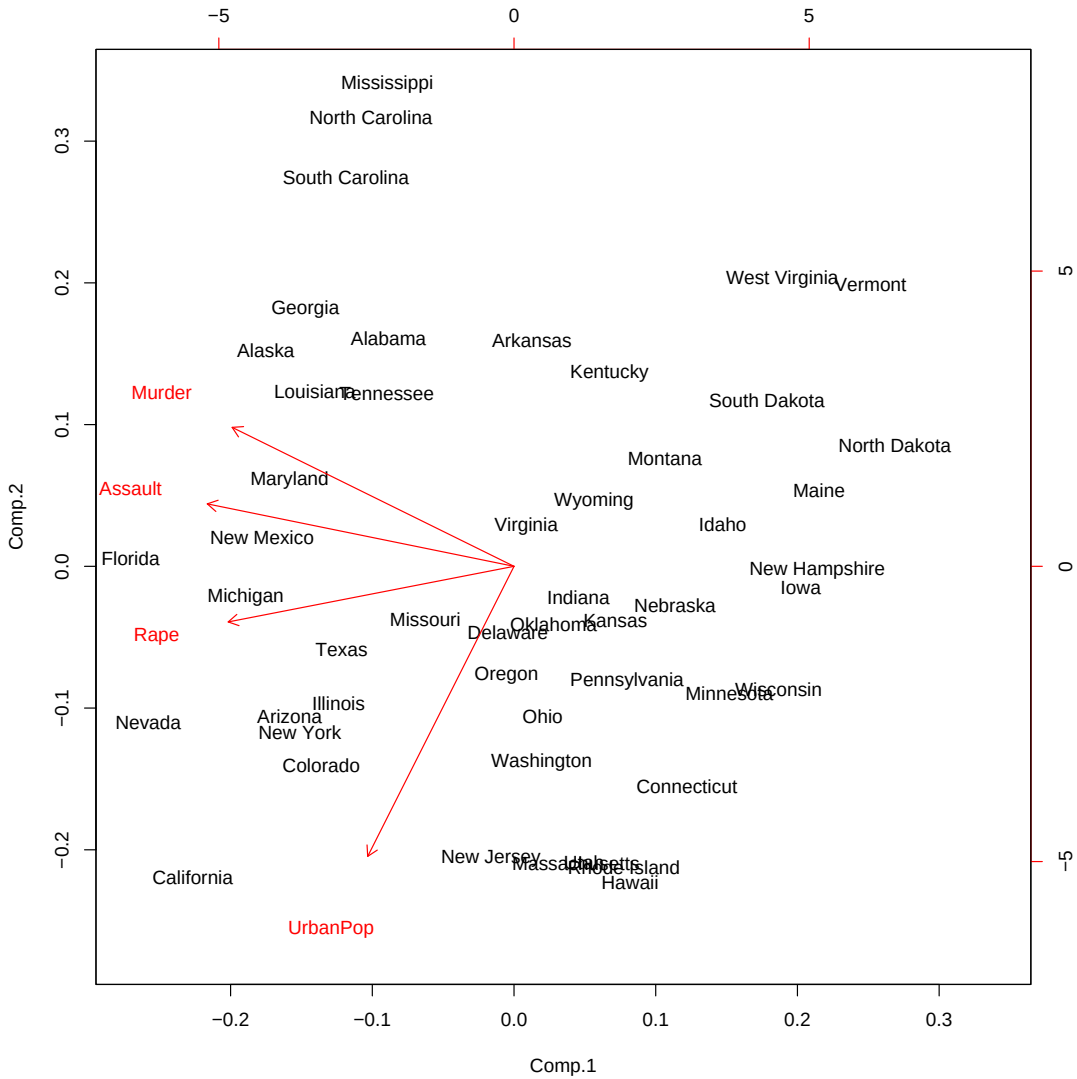
This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

```
pairs(USArrests)
usarrests.pca <- princomp(USArrests,cor=T)
plot(usarrests.pca)

pairs(predict(usarrests.pca))
biplot(usarrests.pca)
```

Pairs Plot: US Arrests



Biplot Example 2: US Arrests



Biplot Example 3: US State data

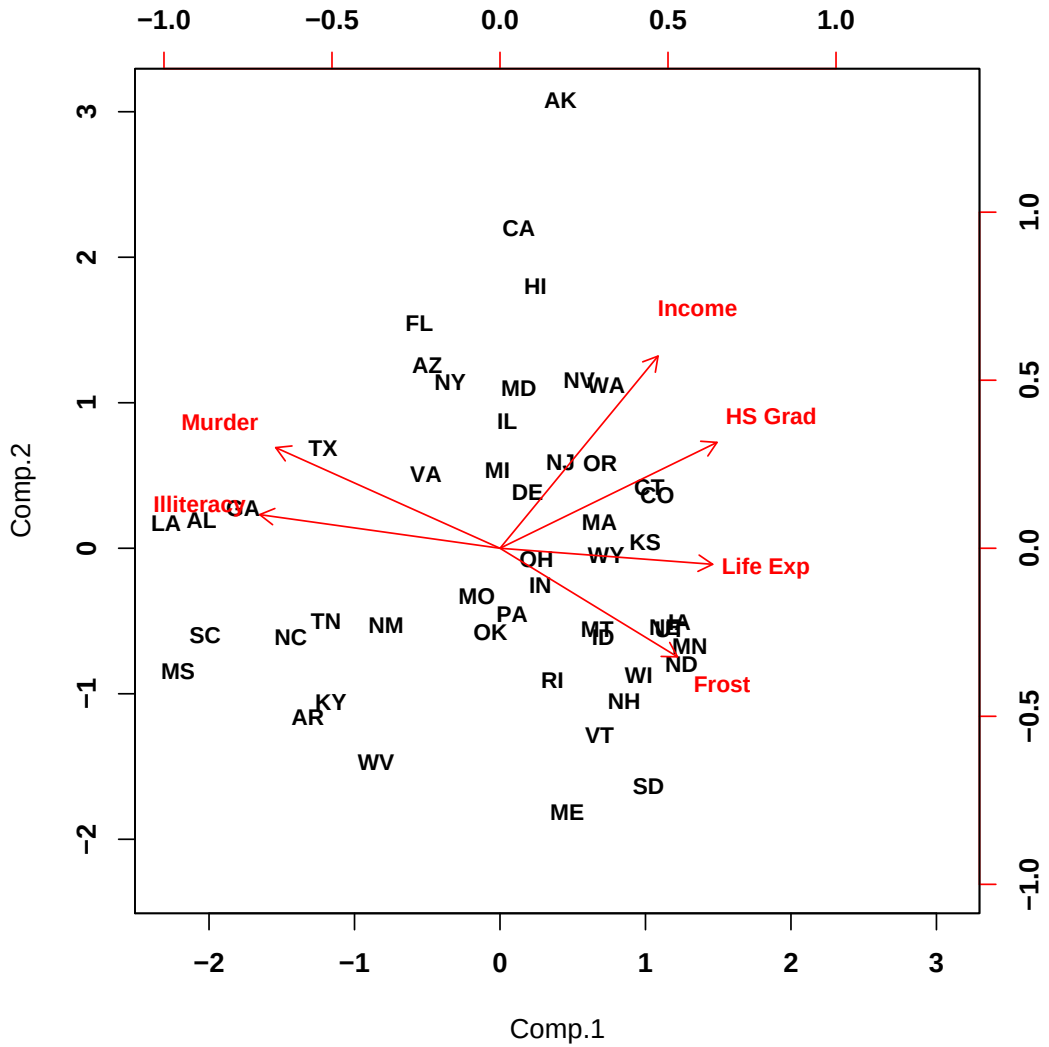
This data set contains statistics like illiteracy and life expectancy on 50 US states.

```
data(state)                          ## load state data
state <- state.x77[, 2:7]            ## extract useful info
row.names(state)<-state.abb
state[1:5,]                          ## lets have a look
```

```
   Income Illiteracy Life Exp Murder HS Grad Frost
AL   3624        2.1    69.05   15.1    41.3    20
AK   6315        1.5    69.31   11.3    66.7   152
AZ   4530        1.8    70.55    7.8    58.1    15
AR   3378        1.9    70.66   10.1    39.9    65
CA   5114        1.1    71.71   10.3    62.6    20
```

```
## calculate the pc's of the data and show biplot
state.pca <- princomp(state,cor=TRUE)
biplot(state.pca,pc.biplot=TRUE,cex=0.8,font=2,expand=0.9)
```

Biplot: US States



Outline

Eigenvalue Decomposition (EVD)

Eigenvalue decomposition places significant role in PCA. PCs are eigenvectors of $X^\top X$ and PCA properties are derived from those of eigenvectors and eigenvalues.

- For any $p \times p$ *symmetric* matrix S (think for example $X^\top X$), there exists p eigenvectors $v_1, \ldots, v_p$ that are pairwise orthogonal and p associated eigenvalues $\lambda_1, \ldots, \lambda_p$ which satisfy the eigenvalue equation $Sv_i = \lambda_i v_i \ \forall i$.
- S can be written as $S = V\Lambda V^\top$ where
  - $V = [v_1, \ldots, v_p]$ is a $p \times p$ orthogonal matrix
  - $\Lambda = diag\{\lambda_1, \ldots, \lambda_p\}$
  - and if $S_{ij} \in \mathbb{R} \ \forall i, j$, $\lambda_i \in \mathbb{R} \ \forall i$
- The relevant R-command is `eigen`. Look at `?eigen` to get help on the command.

Singular Value Decomposition (SVD)

The SVD of a matrix X is an equally useful matrix factorisation that is related to the EVD.

- Though the EVD does not exist for $\mathbb{R}^{n \times p}$ matrices if $p \neq n$, SVDs *always* exists.
- X can be written as $X = UDV^\top$ where
  - U is an $n \times n$ matrix with orthogonal columns.
  - D is a $n \times p$ matrix with decreasing non-negative elements on the diagonal (the singular values) and zero off-diagonal elements.
  - V is a $p \times p$ matrix with orthogonal columns.

  The relevant R-command is `svd`.
- SVD can be computed using very fast and numerically stable algorithms.

Some Properties of the SVD

- Let $X = UDV^\top$ be again the SVD of the $n \times p$ matrix X.
- Note that

$$X^\top X = (UDV^\top)^\top (UDV^\top) = VD^\top U^\top UDV^\top = VD^\top DV^\top,$$

  using orthogonality ($U^\top U = I_n$) of U.
- The eigenvalues of $S = X^\top X$ are thus the squares of the singular values of X and the columns of the orthogonal matrix V are the eigenvectors of S.
- We also have

$$XX^\top = (UDV^\top)(UDV^\top)^\top = UDV^\top VD^\top U^\top = UDD^\top U^\top,$$

  using orthogonality ($V^\top V = I_p$) of V.
- Consider the following optimization problem:

$$\min_{\tilde{X}} \|\tilde{X} - X\|^2 \qquad \text{s.t. } \tilde{X} \text{ has maximum rank } r < n, p.$$

  This problem can be solved by keeping only the r largest singular values of X, zeroing out the smaller singular values in the SVD.

Outline

Multidimensional Scaling (MDS)

MDS is a class of methods based on representing high-dimensional data in a lower dimensional space so that inter-point distances are preserved as "best" as possible. MDS effectively "squeezes" a high-dimensional cloud of points into a smaller number of dimensions, generally 2 or 3.
Given $x_1, \ldots, x_n \in \mathbb{R}^p$, we can obtain a matrix of pairwise distances D with entries $d_{ij} = d(x_i, x_j)$ using some measure of dissimilarity d. For example Euclidean distance $d_{ij} = \|x_i - x_j\|_2$. In most applications, only D is available. MDS finds representations $z_1, \ldots, z_n \in \mathbb{R}^k$ such that

$$d(x_i, x_j) \approx \tilde{d}(z_i, z_j),$$

where d represents dissimilarity in the original p-dimensional space and $\tilde{d}$ represents dissimilarity in the reduced k-dimensional space. The 'best' values of z_i are chosen to minimise some *stress function*.

Metric vs Non-Metric Stress Functions

Metric

Where closeness is considered geometrically, Euclidean distance $d_{ij} = \|x_i - x_j\|_2$ is commonly measured with the classical stress function

$$S_{\text{metric}}(d_{ij}, \tilde{d}_{ij}) = \sum_{i \neq j} \left(d_{ij} - \tilde{d}_{ij}\right)^2$$

Non-Metric

Sometimes it is more important to retain the ordering of d_{ij} as good as possible rather than the actual values assigned. Non-metric stress functions have been developed for ordered distances

$S_{\text{non-metric}}(d_{ij}, \tilde{d}_{ij}) = \min_{g \text{ monotone}} \frac{1}{\sum_{i \neq j} \tilde{d}_{ij}^2} \sum_{i \neq j} \left(g(d_{ij}) - \tilde{d}_{ij}\right)^2$

Solving the Metric MDS Problem

Suppose we only have an $n \times n$ matrix of Euclidean distances $D = (d_{ij})$ but not the points X themselves. The Classical MDS problem is to find a configuration of n points in p-dimensional space that yields the same Euclidean distance matrix as X.
Infinitely many solutions exist as the distance matrix is invariant to rigid motions (rotations, reflections and translations).
As distances are Euclidean, can write $d_{ij} = \|x_i - x_j\|_2$ for some points $x_1, \ldots, x_n \in \mathbb{R}^p$, where

$$\begin{aligned} d_{ij}^2 &= \|x_i - x_j\|_2^2 \\ &= (x_i - x_j)(x_i - x_j)^\top \\ &= x_i x_i^\top + x_j x_j^\top - 2 x_i x_j^\top \end{aligned} \tag{1}$$

Solving the Metric MDS Problem

We define matrix B with entries $b_{ij} = x_i x_j^\top$, we can compute D from B but also B from D. From this, it is possible to recover a configuration which solves this problem.
Writing (1) in terms of b_{ij}, we have

$$d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij} \tag{2}$$

$\Rightarrow$ If two configurations of n objects in p-dimensional space have identical matrix $B = XX^\top$, then they also share the same distance matrix D.

We can also compute b_{ij} in terms of d_{ij} assuming $\sum_i x_i = 0$ (problem sheet).

Solving the Metric MDS Problem

If two configurations of n objects in p-dimensional space have identical matrix $B = XX^\top$, then they also share the same distance matrix D.
Considering the eigendecomposition of B, we see that $B = XX^\top = ULU^\top$ for some orthogonal matrix U with columns $U = (u_1, \ldots, u_n)$ and diagonal matrix L with entries $\lambda_1, \ldots, \lambda_n$. [1]
So if $n > p$ we can write

$$\tilde{X} = [\sqrt{\lambda_1}U_1, \ldots, \sqrt{\lambda_p}U_p]$$

i.e. we have found a p-dimensional configuration of n points $\tilde{X}$ with the *same* distance matrix D as X.

[1] If $X = UDV^\top$ is again the SVD of X, then $XX^\top = UDD^\top U^\top$. The matrix U is thus the same in the EVD of $XX^\top$ and the $n \times n$-matrix $L = DD^\top$ has the same diagonal entries as the $p \times p$-matrix $\Lambda = D^\top D$ in the SVD of $X^\top X$.

MDS Example: US City Flight Distances

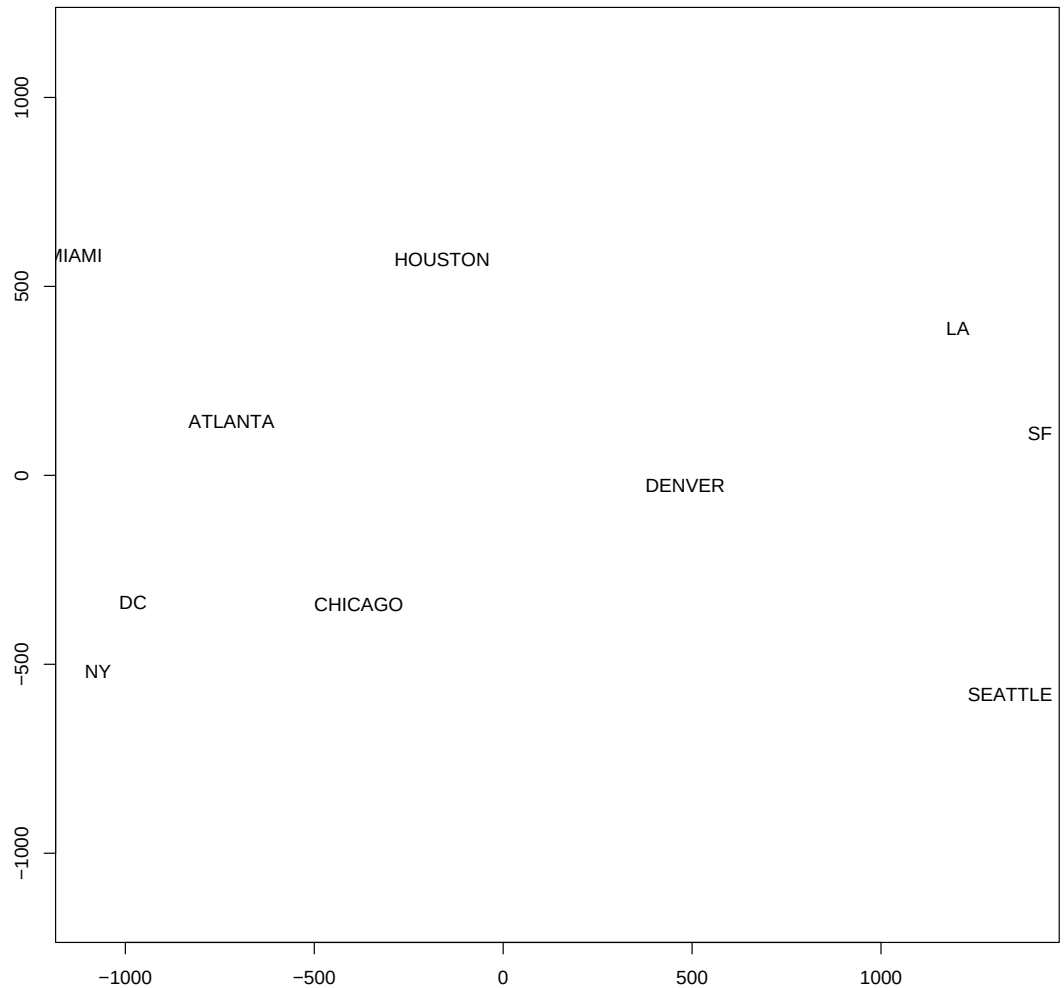
We present a table of flying mileages between 10 American cities, distances calculated from our 2-dimensional world. Using D as the starting point, metric MDS finds a configuration with the same distance matrix.

ATLA	CHIG	DENV	HOUS	LA	MIAM	NY	SF	SEAT	DC
0	587	1212	701	1936	604	748	2139	2182	543
587	0	920	940	1745	1188	713	1858	1737	597
1212	920	0	879	831	1726	1631	949	1021	1494
701	940	879	0	1374	968	1420	1645	1891	1220
1936	1745	831	1374	0	2339	2451	347	959	2300
604	1188	1726	968	2339	0	1092	2594	2734	923
748	713	1631	1420	2451	1092	0	2571	2408	205
2139	1858	949	1645	347	2594	2571	0	678	2442
2182	1737	1021	1891	959	2734	2408	678	0	2329
543	597	1494	1220	2300	923	205	2442	2329	0

MDS Example: US City Flight Distances

```
library(MASS)

us <- read.csv("http://www.stats.ox.ac.uk/
        ~teh/teaching/datamining/data/uscities.csv")

## use the classical stress function
## to find lower dimensional views of the data
## recover X in 2 dimensions

us.classical <- cmdscale(d=us,k=2)

plot(us.classical)
text(us.classical,labels=names(us))
```

MDS Example: US City Flight Distances



Lower-dimensional Reconstructions

Having managed to reconstruct a set of p-dimensional points with the same distance matrix D, we would like to find lower dimensional representations which minimise the stress function S_{metric}.
If the SVD of X is given by $X = UDV^\top$, then

$$B = XX^\top = UDD^\top U = ULU^\top$$

Generally the representation of $\tilde{X}$ (chosen so that $\tilde{X}$ and X have the same distance matrix) can be written as

$$\tilde{X} = [\sqrt{\lambda_1}U_1, \ldots, \sqrt{\lambda_r}U_r]$$

where r is the rank of B.
Setting the smallest eigenvalues to zero reveals the 'best' k-dimensional view of the data (where k is the number of non-zero eigenvalues), minimizing the stress function (proof not given).
This is analogous to PCA, where the smallest eigenvalues of $X^\top X$ are effectively suppressed. Indeed, both PCA and MDS under Euclidean distance are dual and yield effectively the same result (yet MDS can also be applied to distance matrices not generated under Euclidean distance measure).

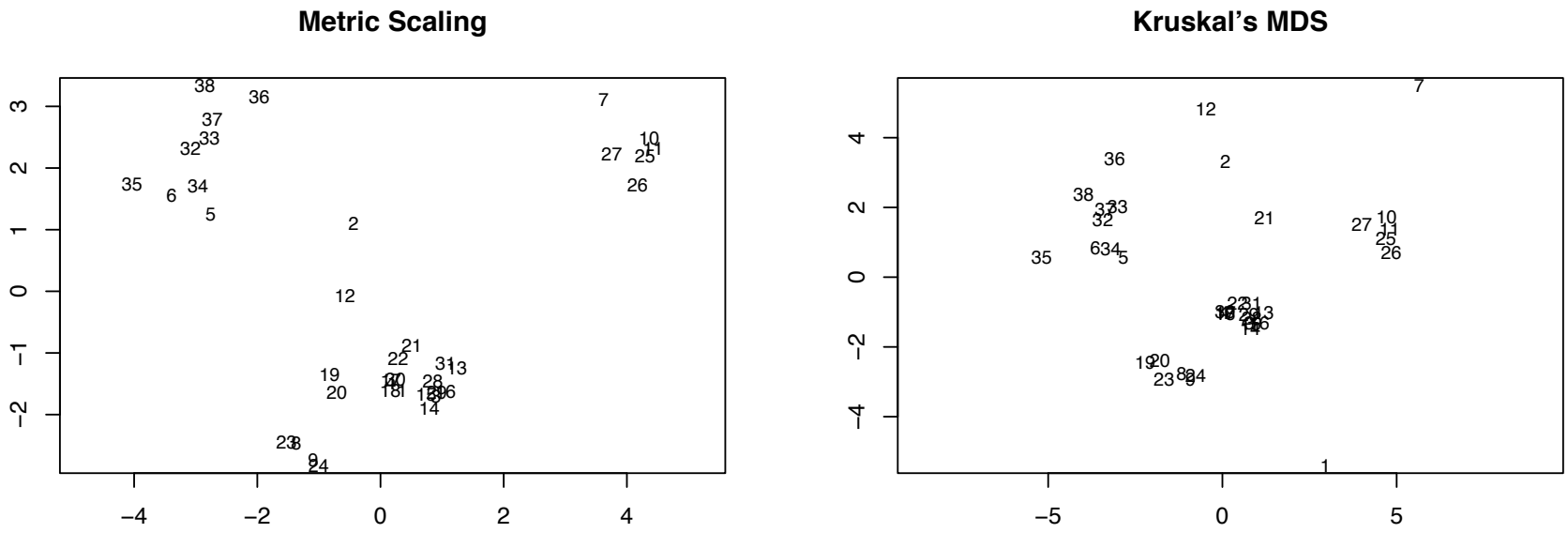
MDS Example: Virus Data

A data set on 39 viruses with rod-shaped particles affecting various crops (tobacco, tomato, cucumber and others), described by Fauquet *et al.* (1988). These are *Tobamoviruses* with monopartite genomes spread by contact.

There are 18 measurements on each virus, the number of amino acid residues per molecule of coat protein; the data come from a total of 26 sources.

We want to investigate whether there are subgroups within this group of viruses.

MDS Example: Virus Data



Distance-based representations of the *Tobamovirus* group of viruses (the variables were scaled before Euclidean distance was used).

MDS Example: Virus Data

MDS reveals some clear subgroups within the *Tobamoviruses*.

Viruses 7 (cucumber green mottle mosaic virus) and 21 (pepper mild mottle virus) have been clearly separated from the other viruses in the non-metric MDS plot, which is not the case in the metric version.

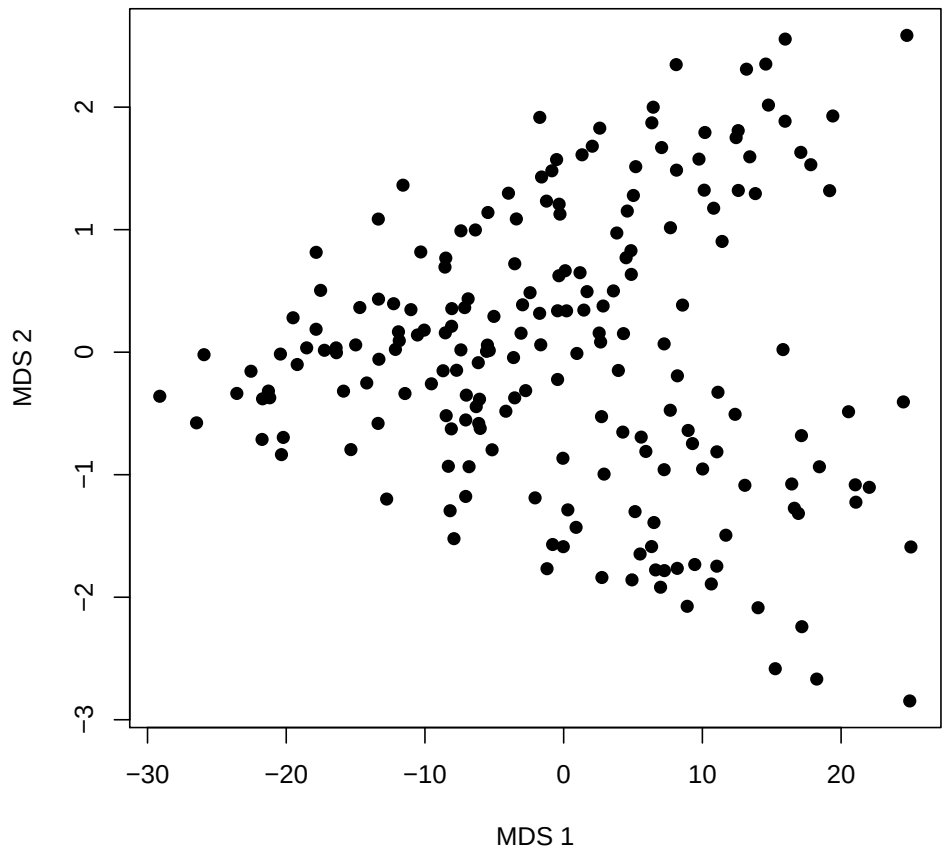
Ripley (1996) states that the non-metric MDS plot shows interpretable groupings. The upper right is the cucumber green mottle virus, the upper left is the ribgrass mosaic virus. The one group of viruses at the bottom, namely 8,9,19,20,23,24, are the tobacco mild green mosaic and odontoglossum ringspot viruses.

Example: Crabs data

```
library(MASS)
Crabs <- crabs[,4:8]
Crabs.class <- factor(paste(crabs[,1],crabs[,2],sep=""))

crabsmds <- cmdscale(d= dist(Crabs),k=2)
plot(crabsmds, pch=20, cex=2)
```

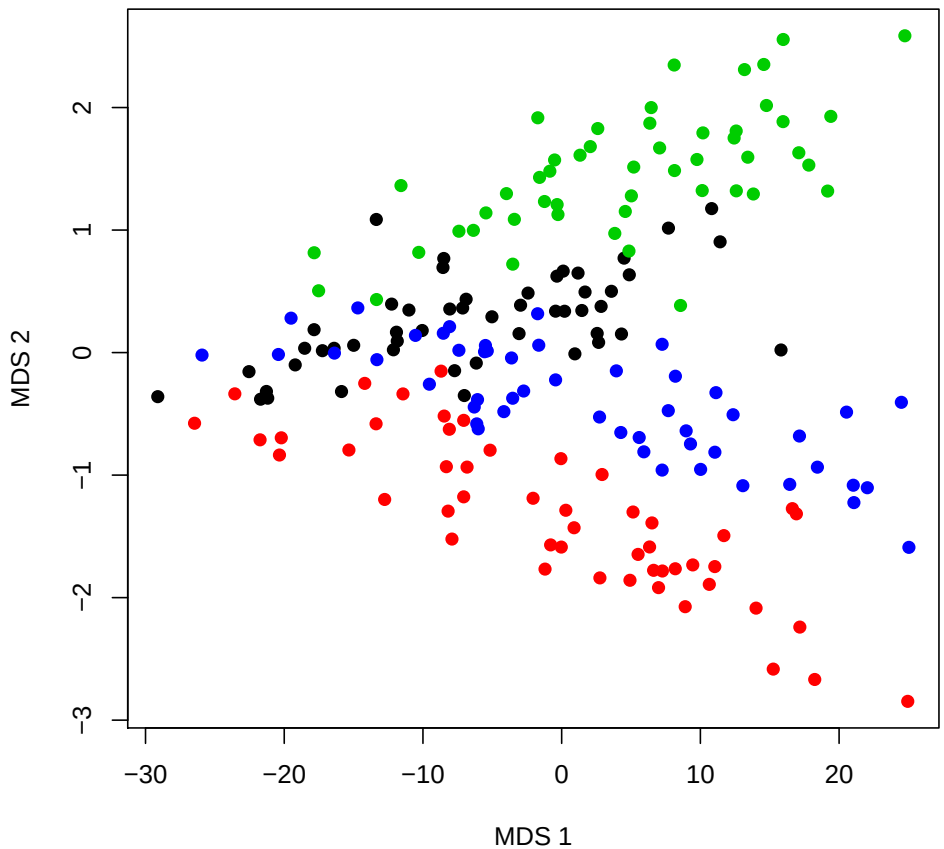
Example: Crabs data

First two MDS components.



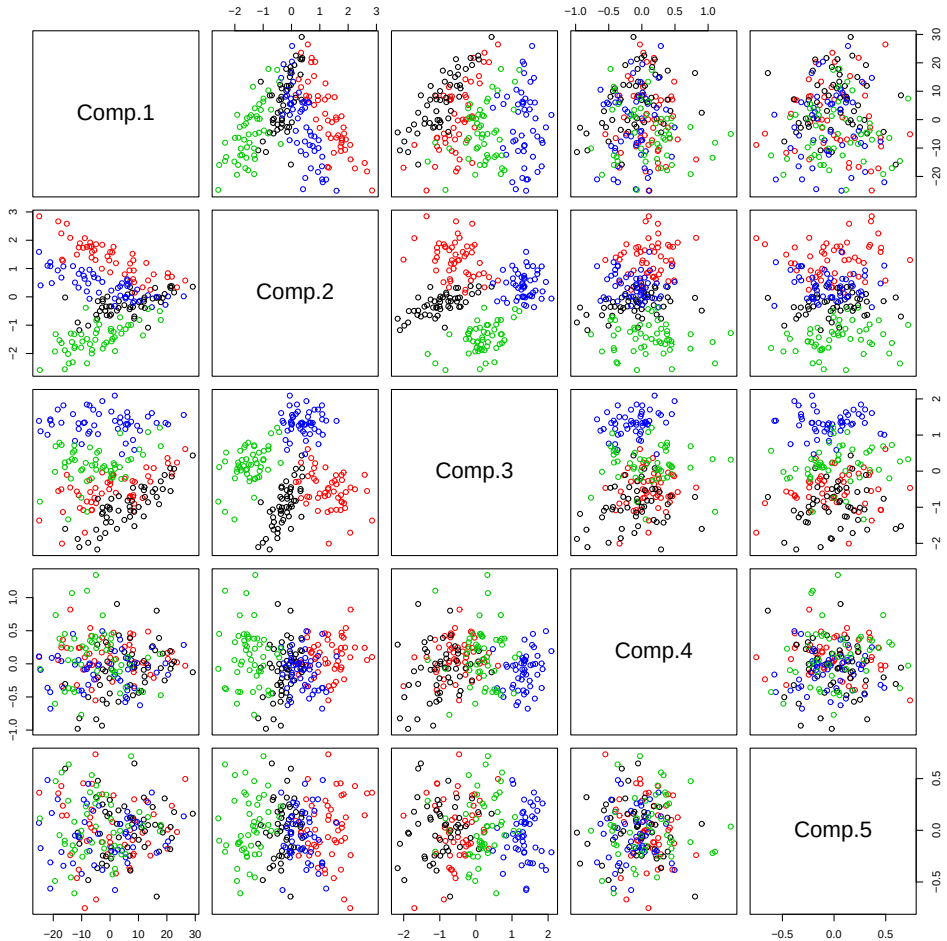
Example: Crabs data

With grouping information.



Example: Crabs data

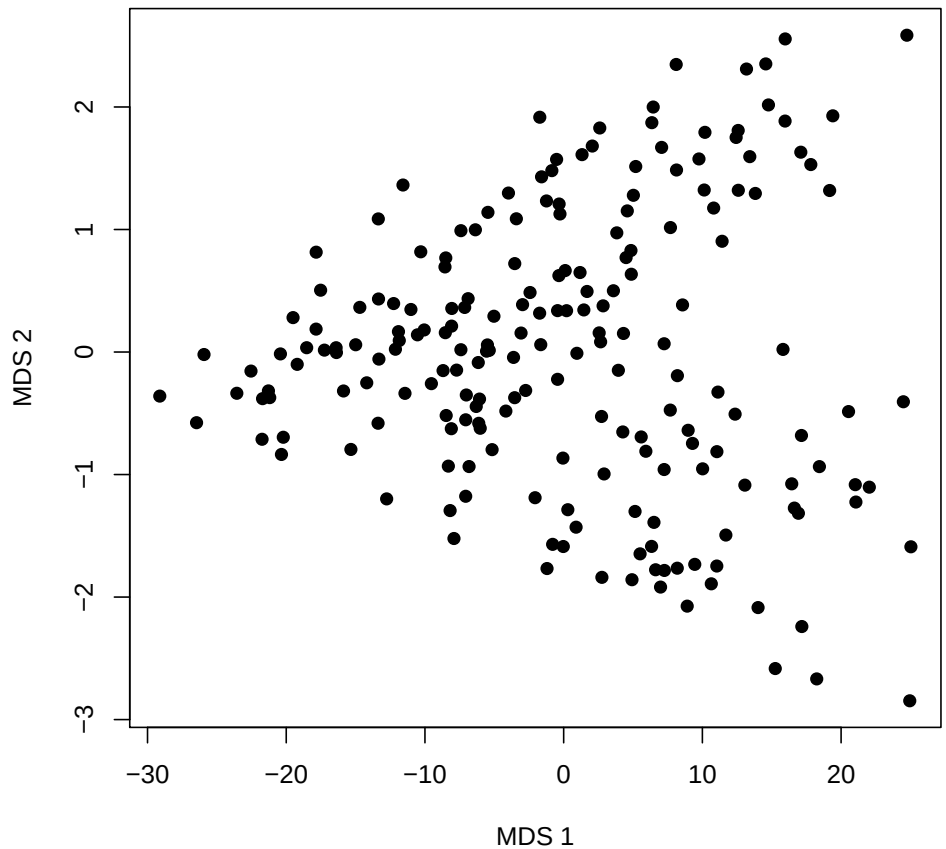
Compare with previous PCA analysis.
MDS solution corresponds to the first 2 PCs as metric scaling was used.



Example: Crabs data

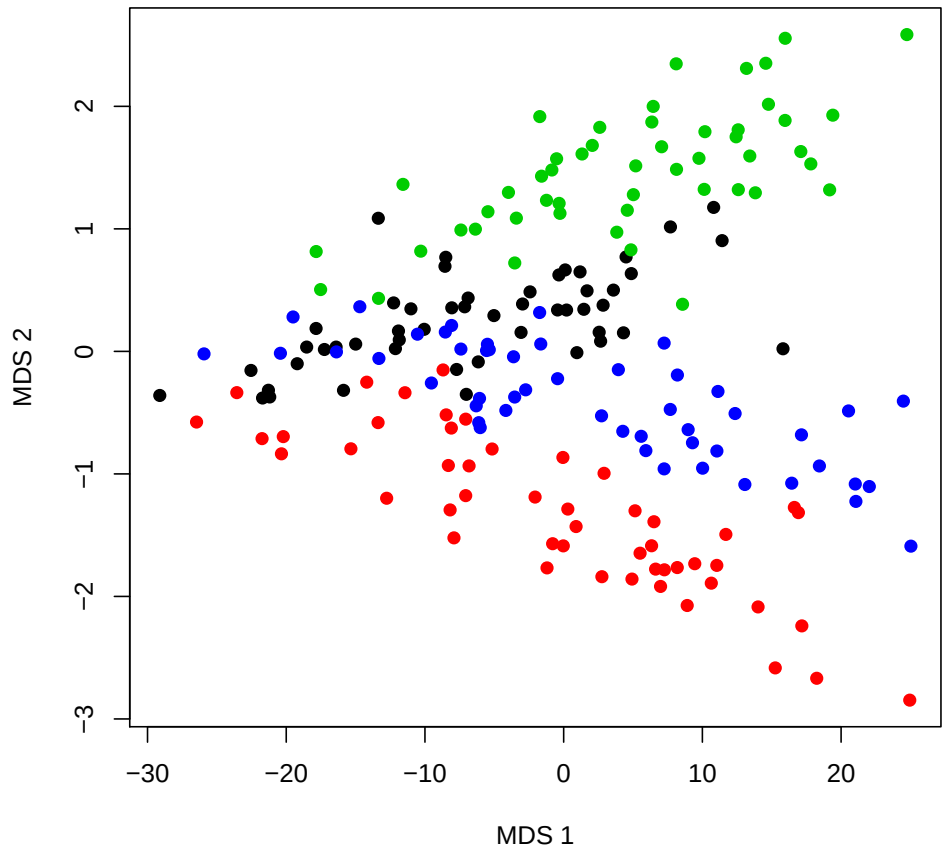
Use Kruskals non-metric multi-dimensional scaling instead.

```
crabsmds <- isoMDS(d= dist(Crabs),k=2)
plot(crabsmds$points, pch=20, cex=2)
```



Example: Crabs data

With grouping information.



Example: Language data

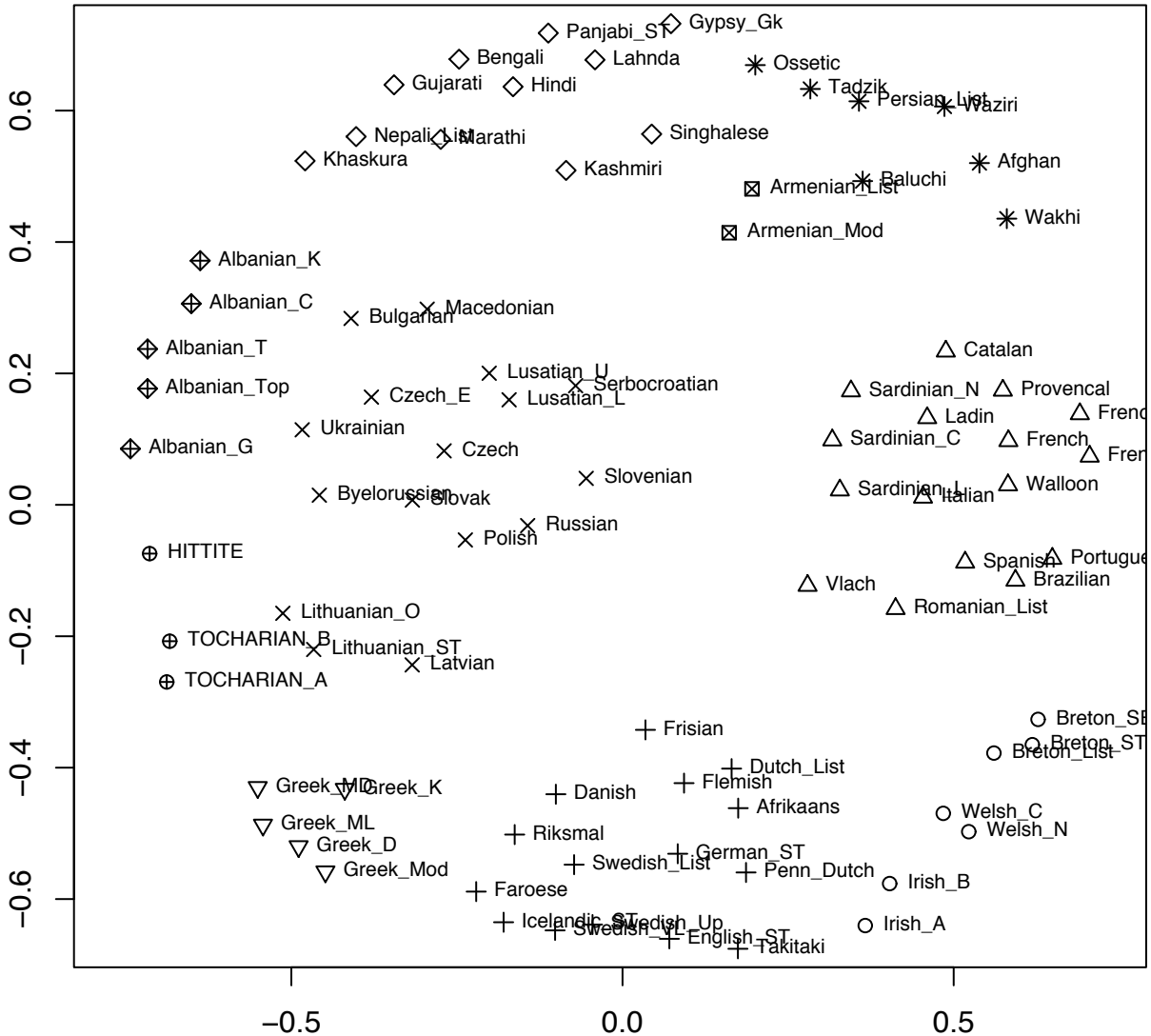
Presence or absence of 2867 homologous traits in 87 Indo-European languages.

```
> X[1:15,1:16]
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
Irish_A	0	0	0	0	1	0	0	0	0	0	0	0	0
Irish_B	0	0	0	0	1	0	0	0	0	0	0	0	0
Welsh_N	0	0	0	1	0	0	0	0	0	0	0	0	0
Welsh_C	0	0	0	1	0	0	0	0	0	0	0	0	0
Breton_List	0	0	0	0	1	0	0	0	0	0	0	0	0
Breton_SE	0	0	0	0	1	0	0	0	0	0	0	0	0
Breton_ST	0	0	0	0	1	0	0	0	0	0	0	0	0
Romanian_List	0	1	0	0	0	0	0	0	0	0	0	0	0
Vlach	0	1	0	0	0	0	0	0	0	0	0	0	0
Italian	0	1	0	0	0	0	0	0	0	0	0	0	0
Ladin	0	1	0	0	0	0	0	0	0	0	0	0	0
Provencal	0	1	0	0	0	0	0	0	0	0	0	0	0
French	0	1	0	0	0	0	0	0	0	0	0	0	0
Walloon	0	1	0	0	0	0	0	0	0	0	0	0	0
French_Creole_C	0	1	0	0	0	0	0	0	0	0	0	0	0

Example: Language data

Using MDS with non-metric scaling.

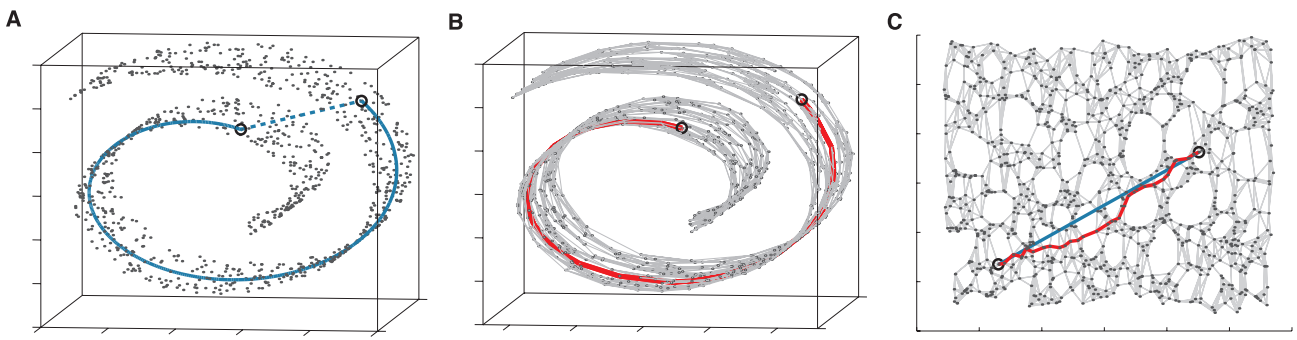


Outline

Isomap

Isomap is useful for non-linear dimension reduction

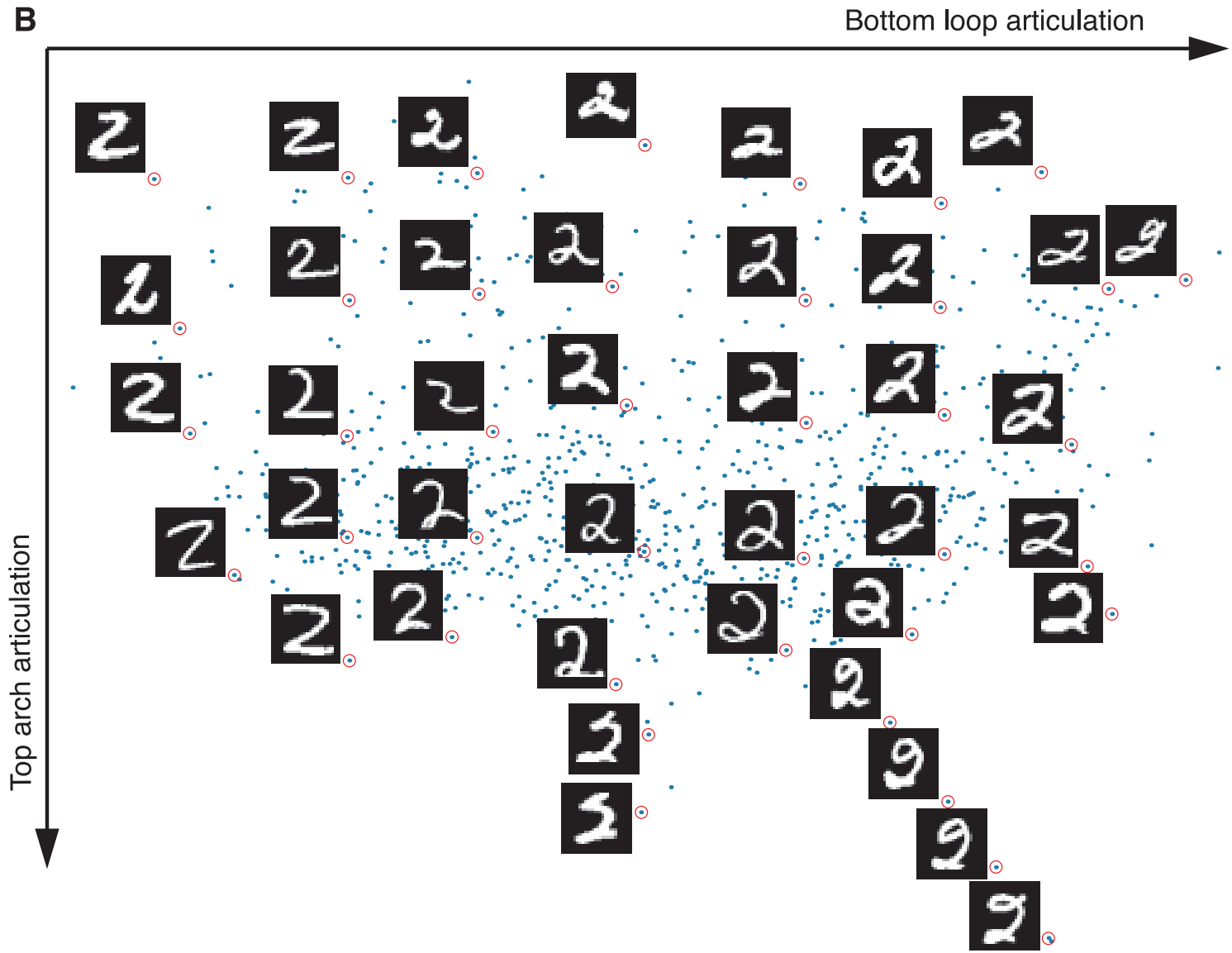
1. Calculate distances d_{ij} for $i, j = 1, \ldots, n$ between all data points, using the Euclidean distance.
2. Form a graph G with the n samples as nodes, and edges between the respective K nearest neighbors (in Euclidean metric).
3. Replace distances d_{ij} by 'shortest-path' distance d_{ij}^G [2] and perform classical MDS, using these distances.



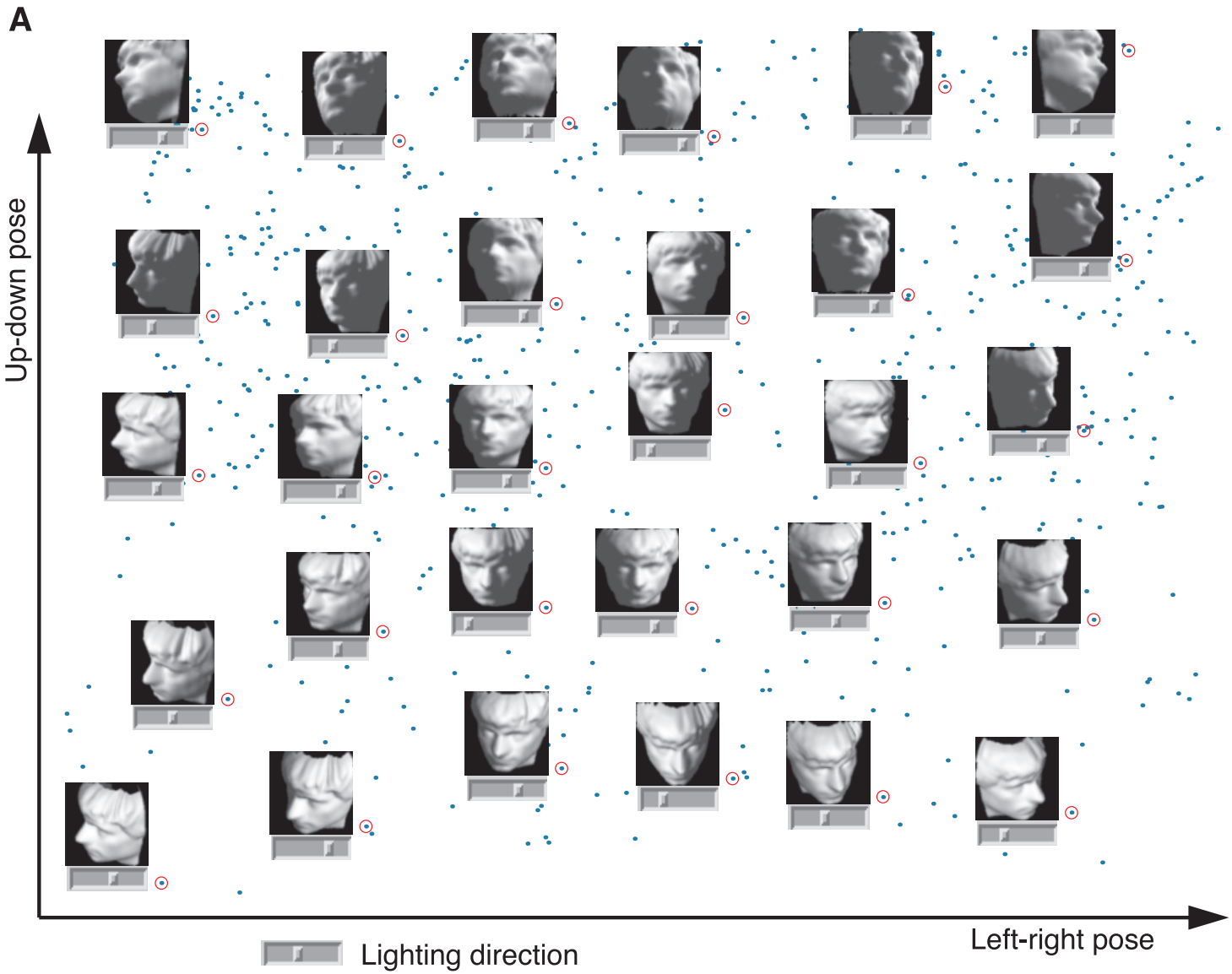
Examples from Tenenbaum et al. (2000)

[2] The path-distance in the graph is, for a given path $i_1 \to i_2 \to \ldots \to i_m$ between two nodes i_1 and i_m that follows the edges of the graph, the sum of the original distances $\sum_{k=1}^{m-1} d_{i_k i_{k+1}}$. The shortest path distance between two points i and j is the minimal path distance along all paths starting in i and ending in j.

Embedding Handwritten Characters



Embedding Faces



Outline

Outline

Clustering

- Cluster analysis is a range of methods that reveal structural information about high-dimensional data directly.
- Given a set of unclassified points X, cluster analysis seeks to arrange them into clusters based on some notion of between cluster and within cluster distance/dissimilarity.
- Partition based methods:
  - Allocate points into K clusters.
  - The number of cluster is usually fixed beforehand or investigated for various values of K as part of the analysis.
- Hierarchical clustering methods:
  - Allocate points into clusters and clusters into super-clusters forming a hierarchy.
  - Typically the hierarchy forms a binary tree (a dendrogram) where each cluster has two “children”.

Outline

Hierarchical Clustering Methods

- Hierarchically structured data can be found everywhere (measurements of different species and different individuals within species), hierarchical methods attempt to understand data by looking for clusters.
- There are two general strategies for generating hierarchical clusters. Both proceed by seeking to minimize some measure of dissimilarity.
  - Agglomerative / Bottom-Up / Merging
  - Divisive / Top-Down / Splitting

  Hierarchical clusters are generated where at each level, clusters are created by merging clusters at lower levels. This process can easily be viewed by a dendogram/tree.

Agglomerative Strategies

- The essence of agglomerative strategies is very simple: starting with each observation as a separate cluster, recursively merge the two most similar clusters (with the smallest dissimilarity) until we are left with a single cluster.
- The way in which we measure dissimilarity between clusters affects the resulting dendograms in a predictable way. If clusters exist however, it is clear by inspecting the dendograms using whatever way we measure dissimilarity between clusters.

Measuring Dissimilarity

To find hierarchical clusters, we need some way to measure the dissimilarity between clusters

- Given two points x_i and x_j, it is straightforward to measure their dissimilarity, say $d(x_i, x_j) = \|x_i - x_j\|_2$.
- It is unclear however how to extend this to measure dissimilarity between *clusters*, $D(C_i, C_j)$ for clusters C_i and C_j.

Many such proposals though no concensus as to which is best.

(a) *Single-Link Clustering*

$$D(C_i, C_j) = \min_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

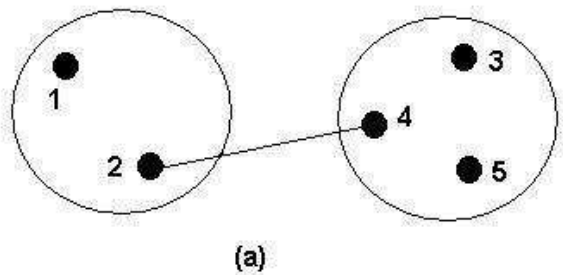
(b) *Complete-Link Clustering*

$$D(C_i, C_j) = \max_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

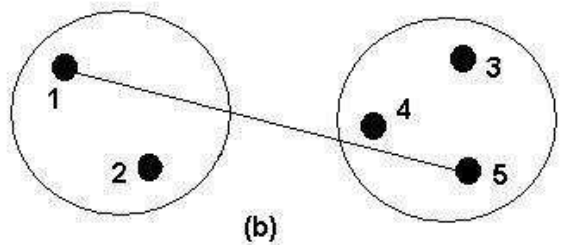
(c) *Group-Average Clustering*

$$D(C_i, C_j) = avg_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

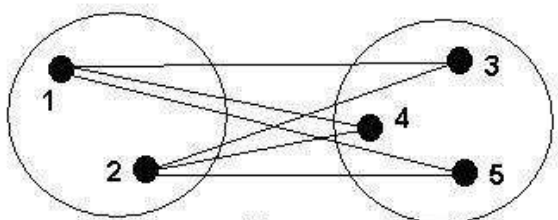
Measuring Dissimilarity



(a)



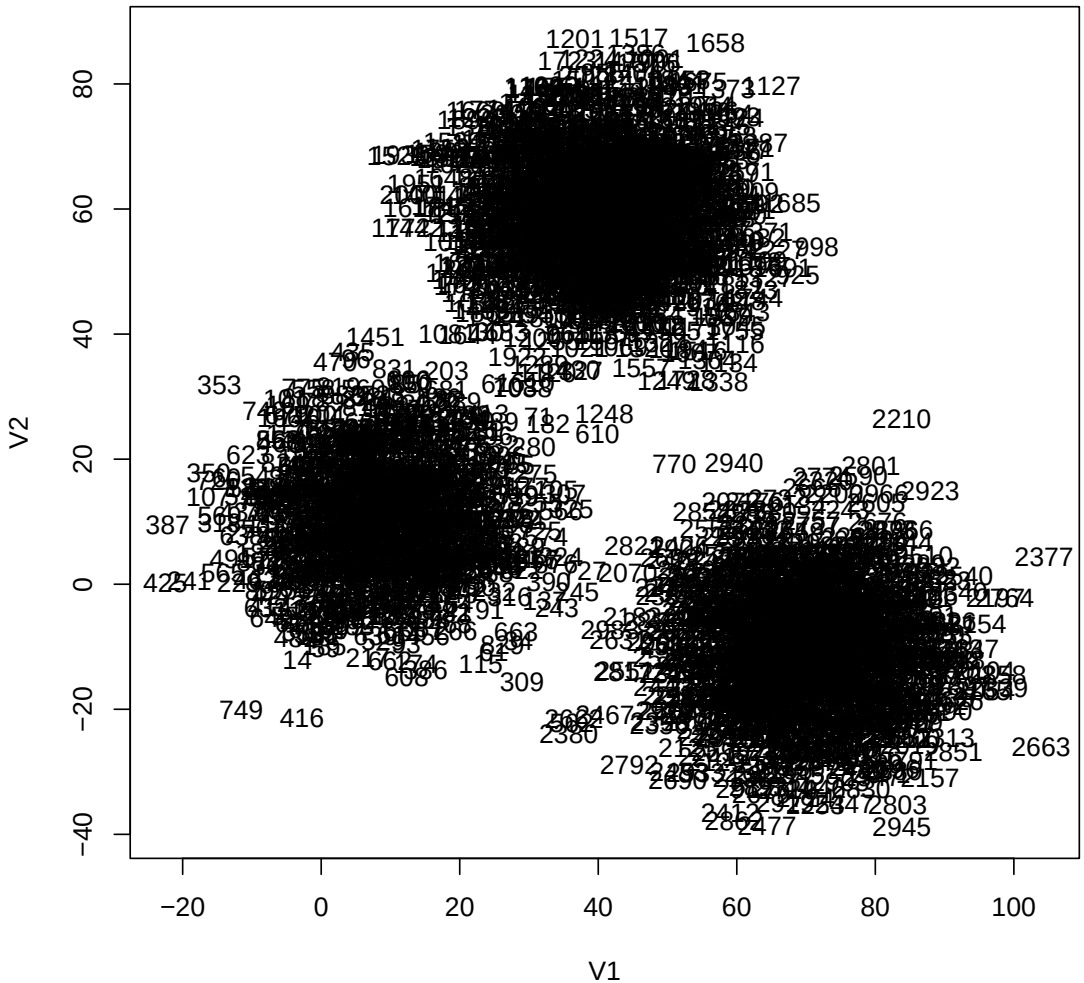
(b)



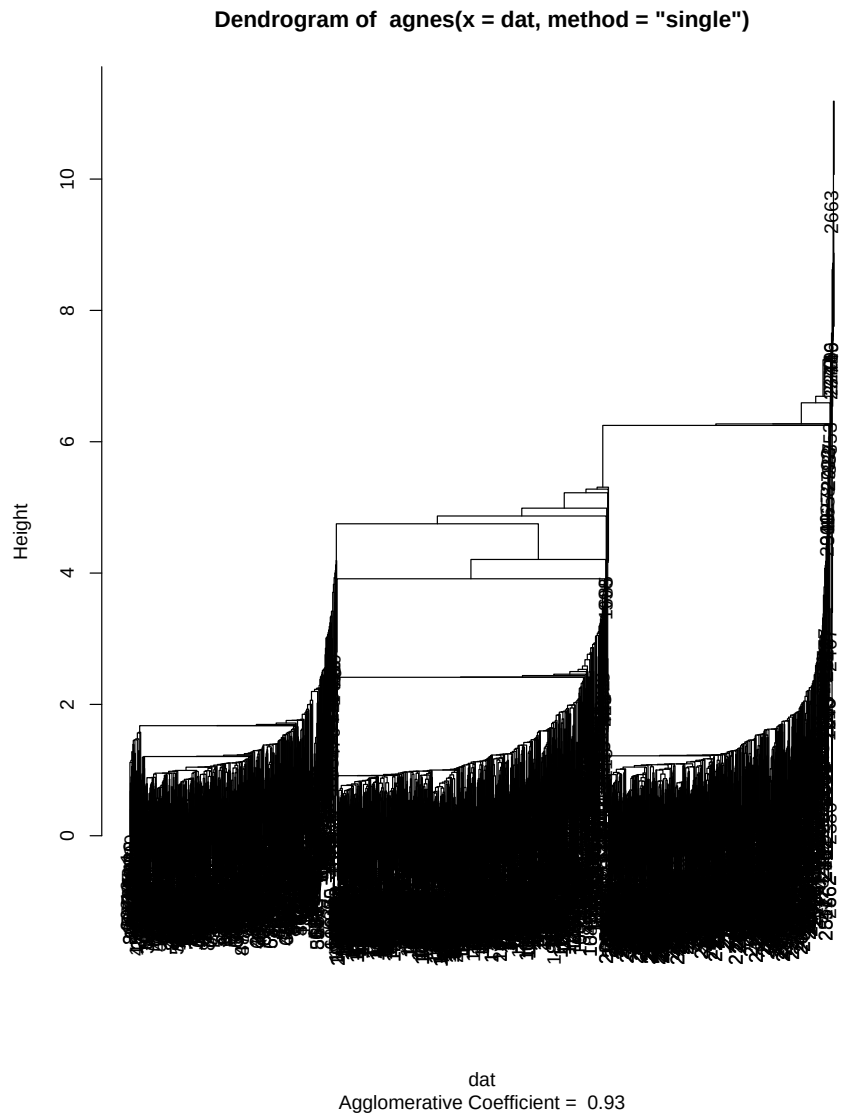
(c)

Cluster Distance
d24
d15
$\frac{d13+d14+d15+d23+d24+d25}{6}$

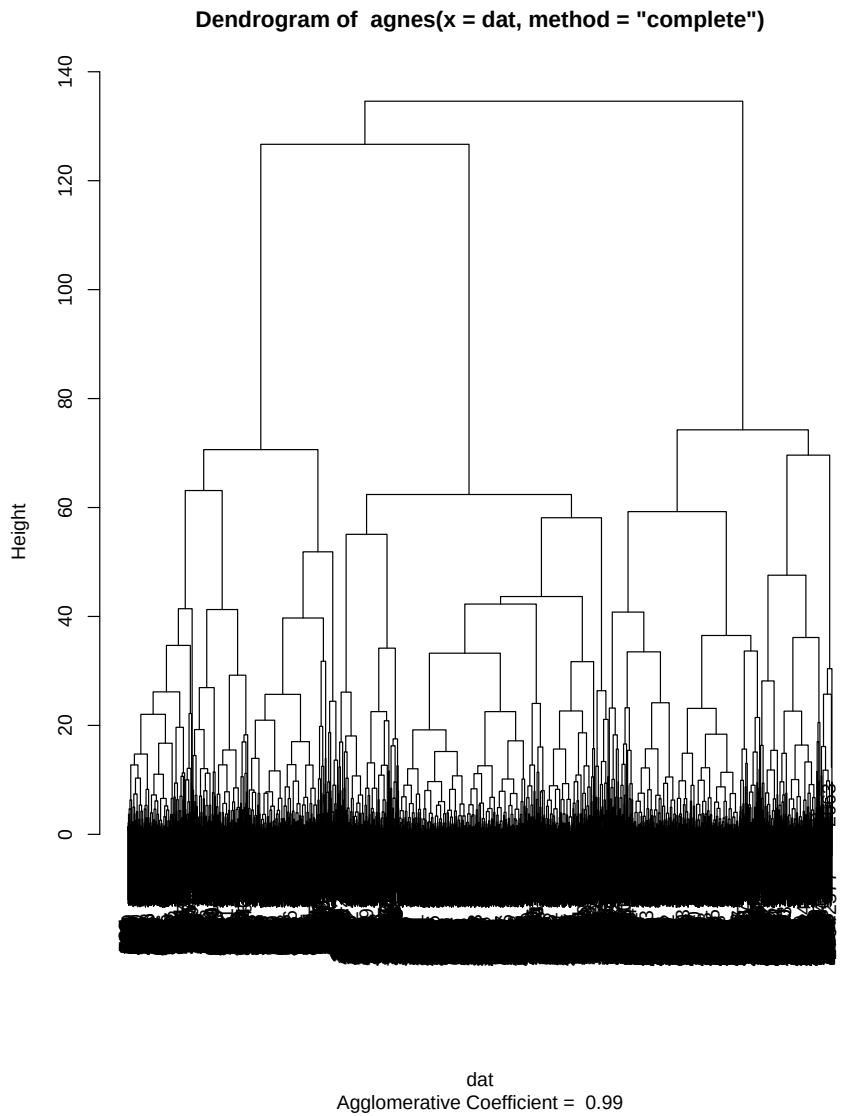
Hierarchical Clustering Example: Artificial Dataset



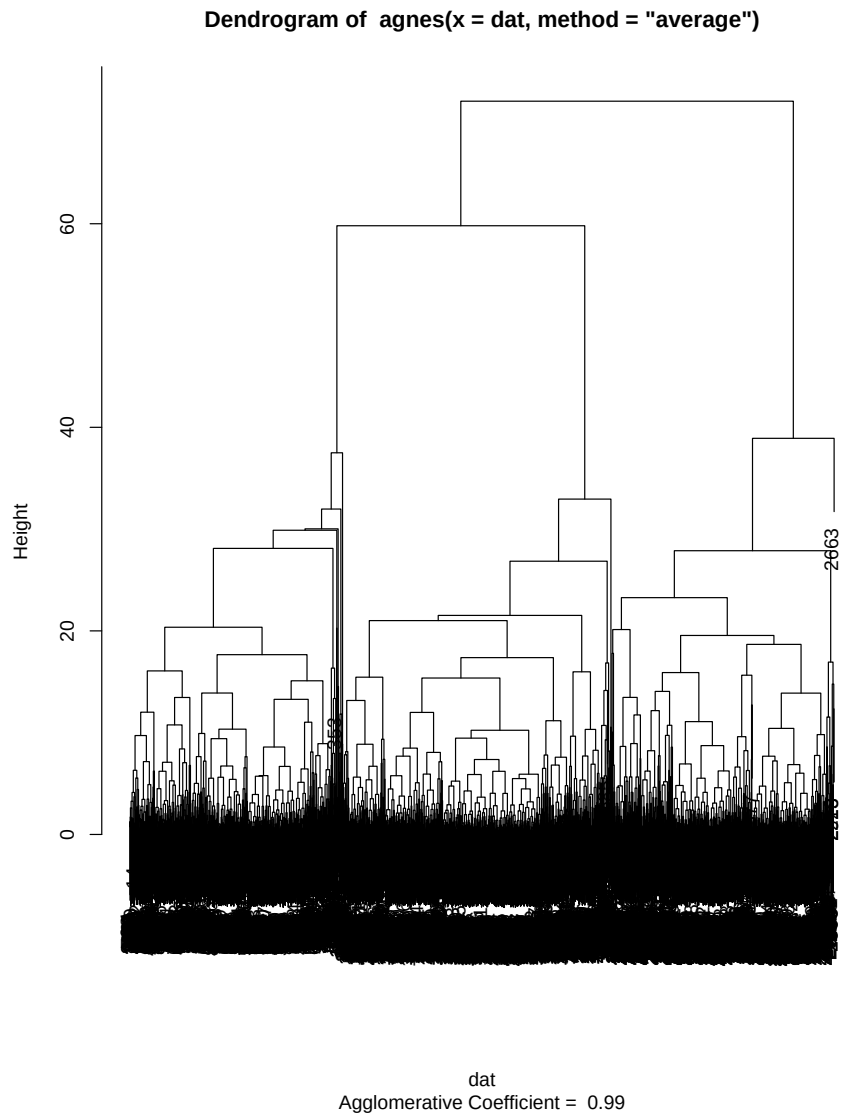
Hierarchical Clustering Example: Artificial Dataset



Hierarchical Clustering Example: Artificial Dataset



Hierarchical Clustering Example: Artificial Dataset



R Code

```
#start afresh
dat=xclara #3000 x 2
library(cluster)

#plot the data
plot(dat,type="n")
text(dat,labels=row.names(dat))

plot(agnes(dat,method="single"))
plot(agnes(dat,method="complete"))
plot(agnes(dat,method="average"))
```

Divisive Strategies

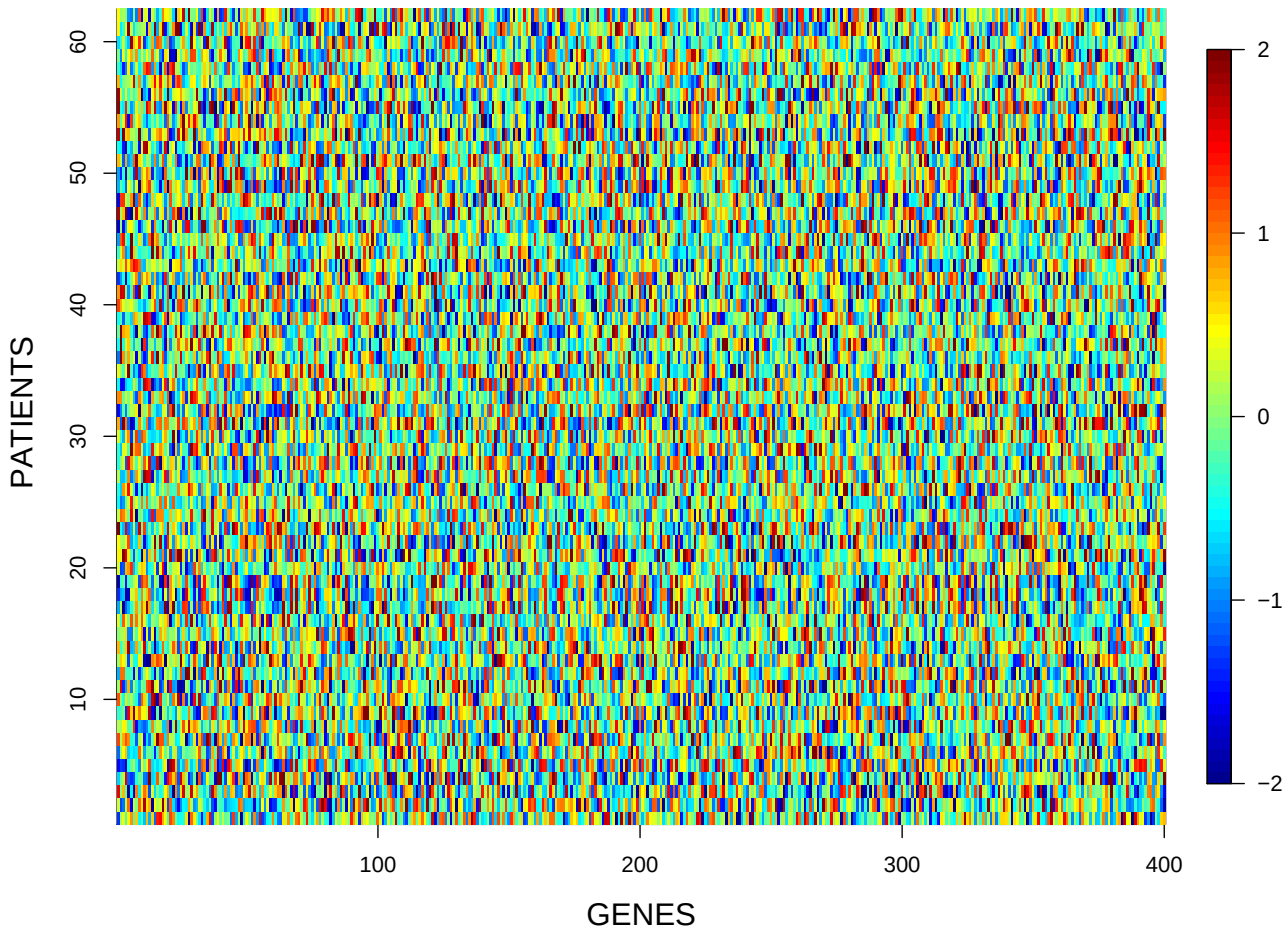
- Divisive strategies work in the opposite direction: starting with a single cluster which holds every observation, we recursively proceed as follows. For all clusters, partition the one that results in the greatest increase in dissimilarity that can arise when it is split in two. This recurses until each observation is a cluster on its own.
- If there are s observations in any cluster, there are $2^{s-1} - 1$ possible ways of partitioning it into two non-empty sets, a computationally infeasible task. Approximate methods are employed to tackle this problem which search though a subset of these possibilities.
- Divisive approaches are better than agglomerative approaches at showing structure near the top of the tree and so are preferred when interest is focused on partitioning data into a relatively small numbers of clusters.
- Divisive approaches are less known and so are much less used than agglomerative strategies.

Using Dendograms

- Different ways of measuring dissimilarity result in different trees.
- Dendograms are useful for getting a feel for the structure of high-dimensional data though they don't represent distances between observations well.
- Dendograms show hierarchical clusters with respect to increasing values of dissimilarity between clusters, cutting a dendogram horizontally at a particular height partitions the data into disjoint clusters which are represented by the vertical lines it intersects. Cutting horizontally effectively reveals the state of the clustering algorithm when the dissimilarity value between clusters is no more than the value cut at.
- Despite the simplicity of this idea and the above drawbacks, hierarchical clustering methods provide users with interpretable dendograms that allow clusters in high-dimensional data to be better understood.

Example: Lymphoma Gene Expression Data

Gene expression values taken of 4026 genes of 62 patients in a lymphoma cancer study. Color coded expression values for 500 randomly chosen genes look as follows.



Example: Lymphoma Gene Expression Data

Figure was generated by `R` code:

```
load(file="lymphoma.rda")
library(fields)

X <- lymphoma.x
X <- scale(X)
X <- X[,sample(1:ncol(X),400)]
for (k in 1:nrow(X)) X[k,] <- pmin(2,pmax(-2,X[k,]))

indn <- sample(1:nrow(X),nrow(X))

image.plot(1:ncol(X),1:nrow(X),t(X[indn,]),
        col=tim.colors(200),
        xlab="GENES", ylab="PATIENTS", cex.lab=1.4)
```

Example: Lymphoma Gene Expression Data

Now lets do hierarchical clustering with function `hclust`.

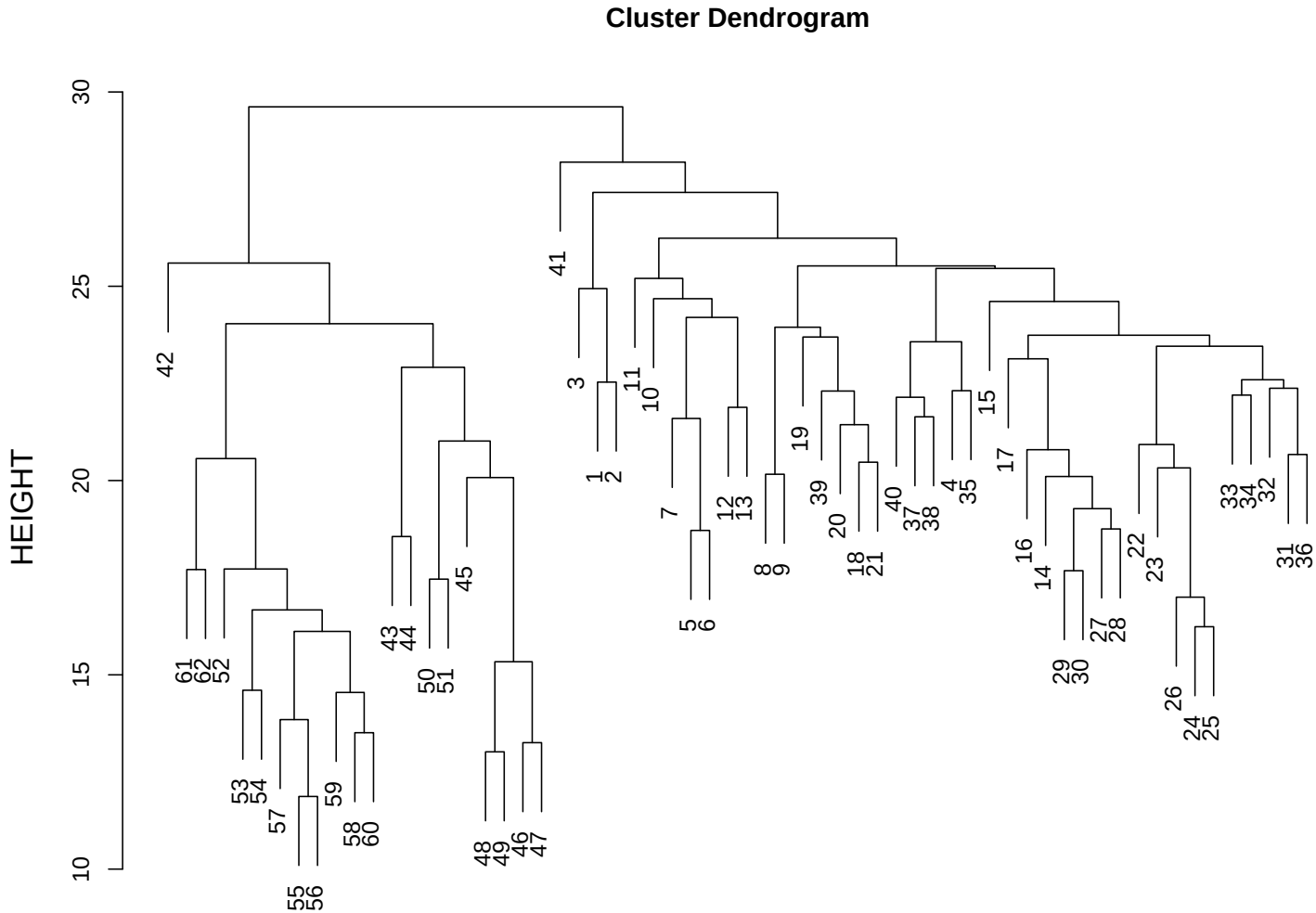
```
dd <- dist(t(X))
hh <- hclust(dd,method="average")

ddn <- dist(X)
hhn <- hclust(ddn,method="average")

plot(hh)
plot(hhn)
```

...or a bit more fancy

```
plot(hh,cex.lab=1.3,xlab="",ylab="HEIGHT",sub="")
plot(hhn,cex.lab=1.3,xlab="",ylab="HEIGHT",sub="")
```

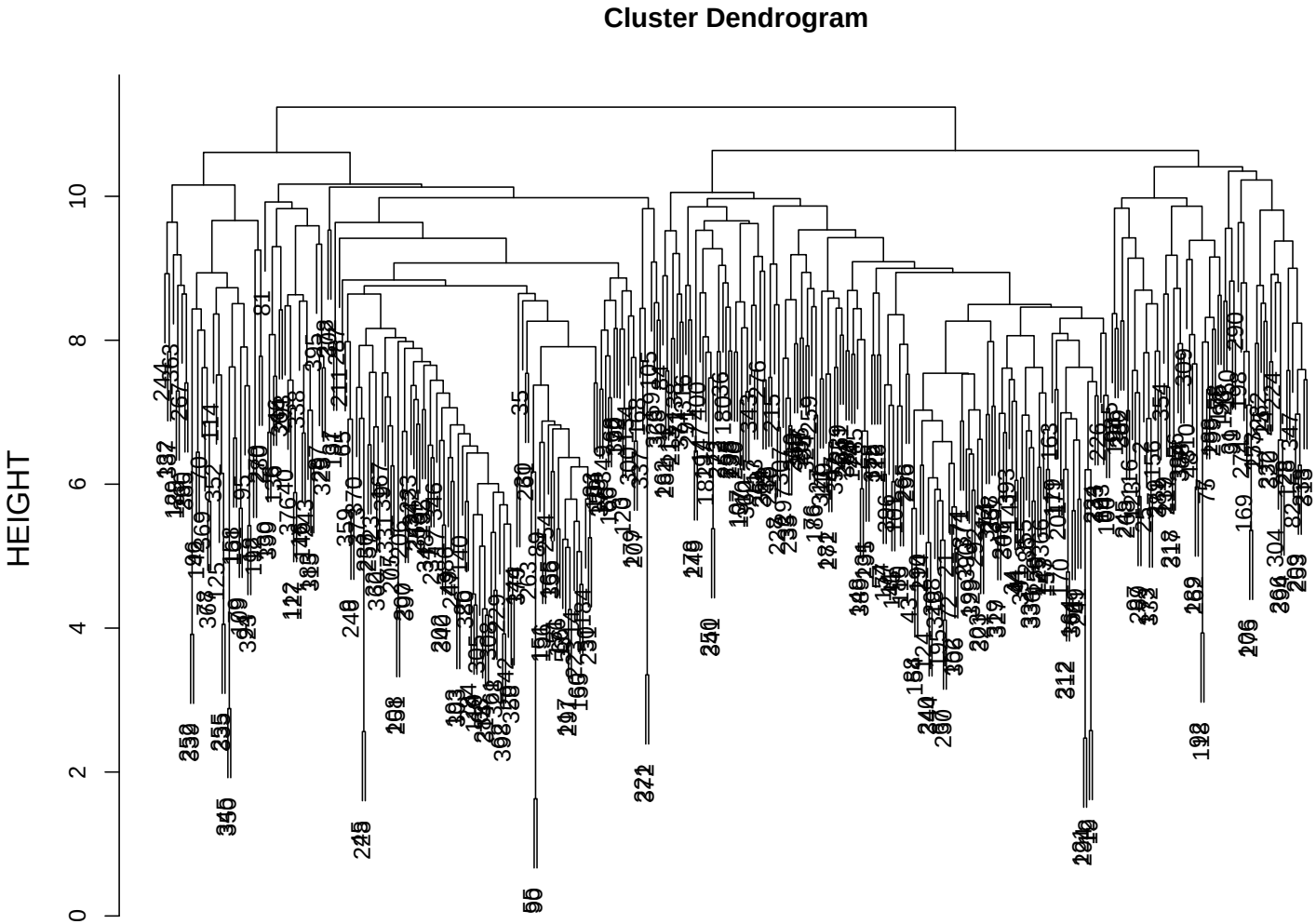
Example: Lymphoma Gene Expression Data

Using hierarchical clustering with average linkage for the 62 patients yields:



Example: Lymphoma Gene Expression Data

Using hierarchical clustering on the genes (instead of patients):



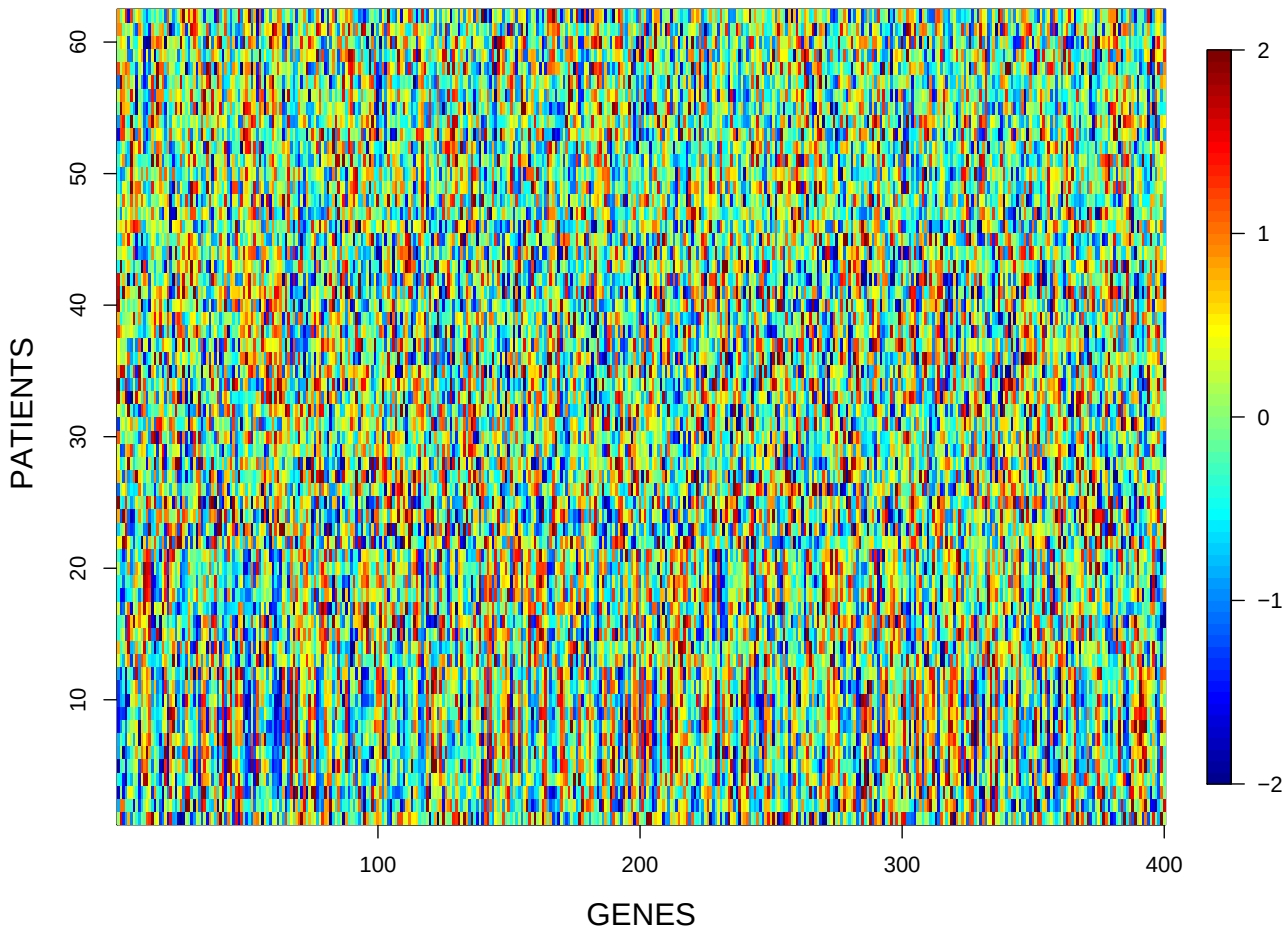
Example: Lymphoma Gene Expression Data

Can order the patients according to the ordering implied by `hclust`.

```
ord <- hh$order
ordn <- hhn$order

image.plot(1:ncol(X),1:nrow(X),t(X[ordn,]),
    col=tim.colors(200),cex.lab=1.4,
    xlab="GENES",ylab="PATIENTS")

image.plot(1:ncol(X),1:nrow(X),t(X[ordn,ord]),
    col=tim.colors(200),cex.lab=1.4,
    xlab="GENES",ylab="PATIENTS")
for (k in 1:nrow(X))
    text( ncol(X)-200, k, labels= "",cex=0)
for (k in 1:nrow(X))
    mtext((lymphoma.y[ordn])[k],side=4,at=k,cex=1.1)
```

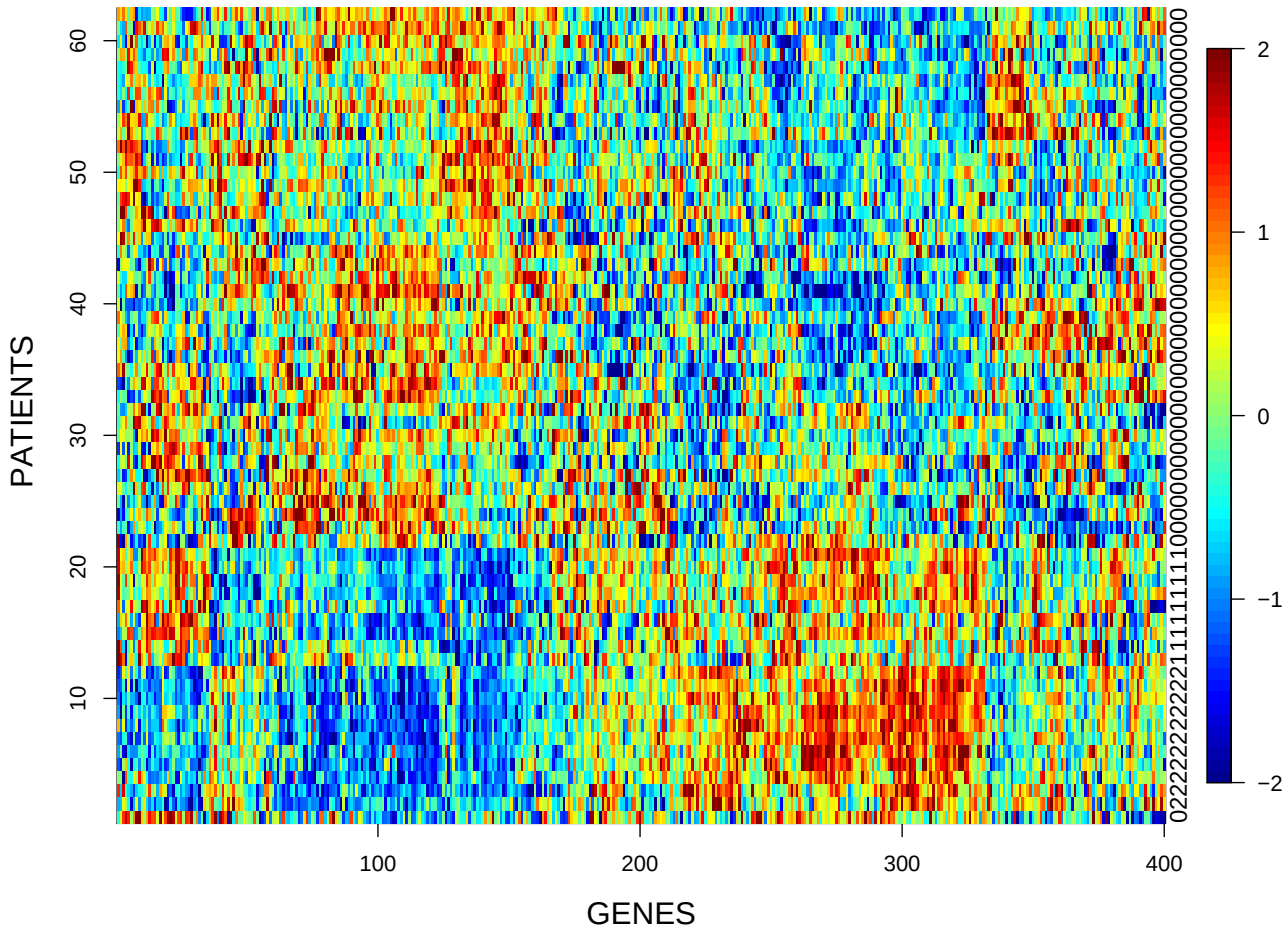
Example: Lymphoma Gene Expression Data

Using the ordering of patients implied by hierarchical clustering yields the following expression matrix.



Example: Lymphoma Gene Expression Data

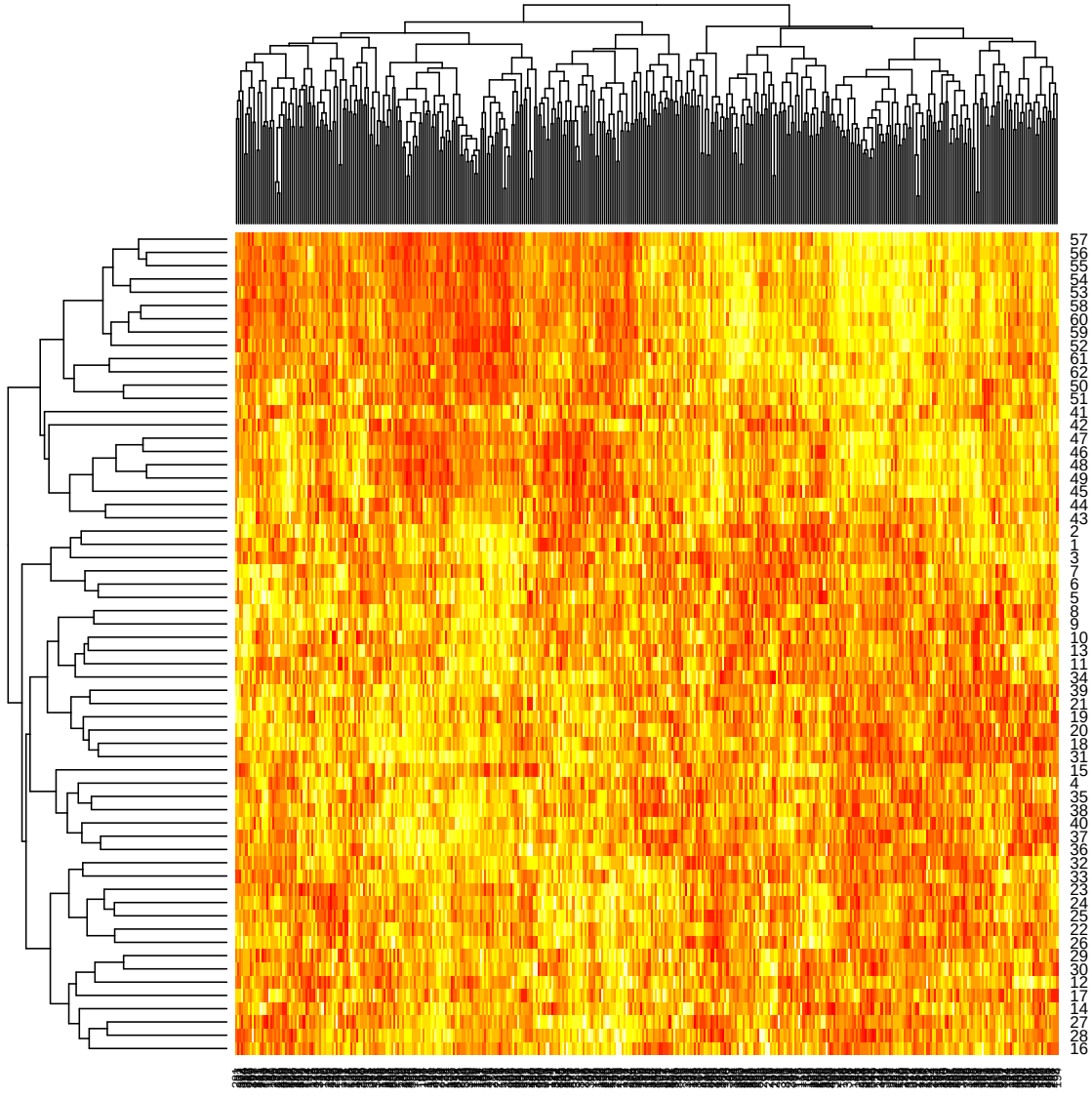
Using the ordering of patients **and** genes implied by hierarchical clustering yields the following expression matrix.



Different subtypes of lymphoma cancer (coded here as classes 0,1,2) are discovered in this way!

Example: Lymphoma Gene Expression Data

Or simply use command `heatmap(X)`.



Outline

K-means

Partition methods seek to divide examples into a pre-assigned number of clusters $C_1, \ldots, C_K$ where for all $k, k' \in \{1, \ldots, K\}$,

- $C_k \subset \{x_1, \ldots, x_n\}$
- $C_k \cap C_{k'} = \emptyset \quad \forall k \neq k'$
- $\cup_{k=1}^{K} C_k = \{x_1, \ldots, x_n\}$

For Euclidean space, we can assign a centre r_k to each cluster in order to measure within-cluster deviance

$$W_{C_k}(r_k) = \sum_{i:x_i \in C_k} \|x_i - r_k\|_2^2.$$

K-means

The overall objective is to choose both the cluster centres and allocation of points to minimize total within-cluster deviance given by

$$W = \sum_{k=1}^{K} W_{C_k}(r_k).$$

Given the contents of a cluster, simple differentiation of $W_{C_k}(r_k)$ with respect to r_k shows that within-cluster deviance is least when

$$r_k = \frac{1}{|C_k|} \sum_{i:x_i \in C_k} x_i,$$

where $|C_k| = \#\{i : x_i \in C_k\}$ is the number of members of cluster k.
The hard part is the combinatorial task of allocating points to clusters.

K-means

The K-means algorithm is a well-known method that heuristically minimizes W to partition $x_1, \ldots, x_n$ into K clusters for some K.

1. Randomly fix K cluster centres $r_1, \ldots, r_K$.
2. For each $i = 1, \ldots, n$, assign each x_i to the cluster with the nearest centre,

$$x_i \in C_k \quad \Leftrightarrow \quad \|x_i - r_k\| \leq \|x_i - r'_k\| \quad \forall k' \neq k.$$

3. Move cluster centres $r_1, \ldots, r_K$ to the average of the new clusters.
4. Repeat 2 and 3 until there is no change.
5. Return the partitions $C_1, \ldots, C_K$ at the end.

Some Properties

Some notes about the K-means algorithm.

- **The algorithm stops in a finite number of iterations.** Between steps 2 and 3, W either stays constant -in which case we stop- or it decreases, this implies that we never revisit the same partition. As there are only finitely many partitions, the number of iterations cannot exceed this.
- **The K-means algorithm need not converge to a globally optimal assignment.** K-means is a heuristic search algorithm so it can get stuck at suboptimal configurations. The result depends on the starting configuration.
- **Other partition based methods.** There are many other partition based methods that employ related ideas for example K-medoids differs from K-means in requiring cluster centres r_i to be an observation x_i.

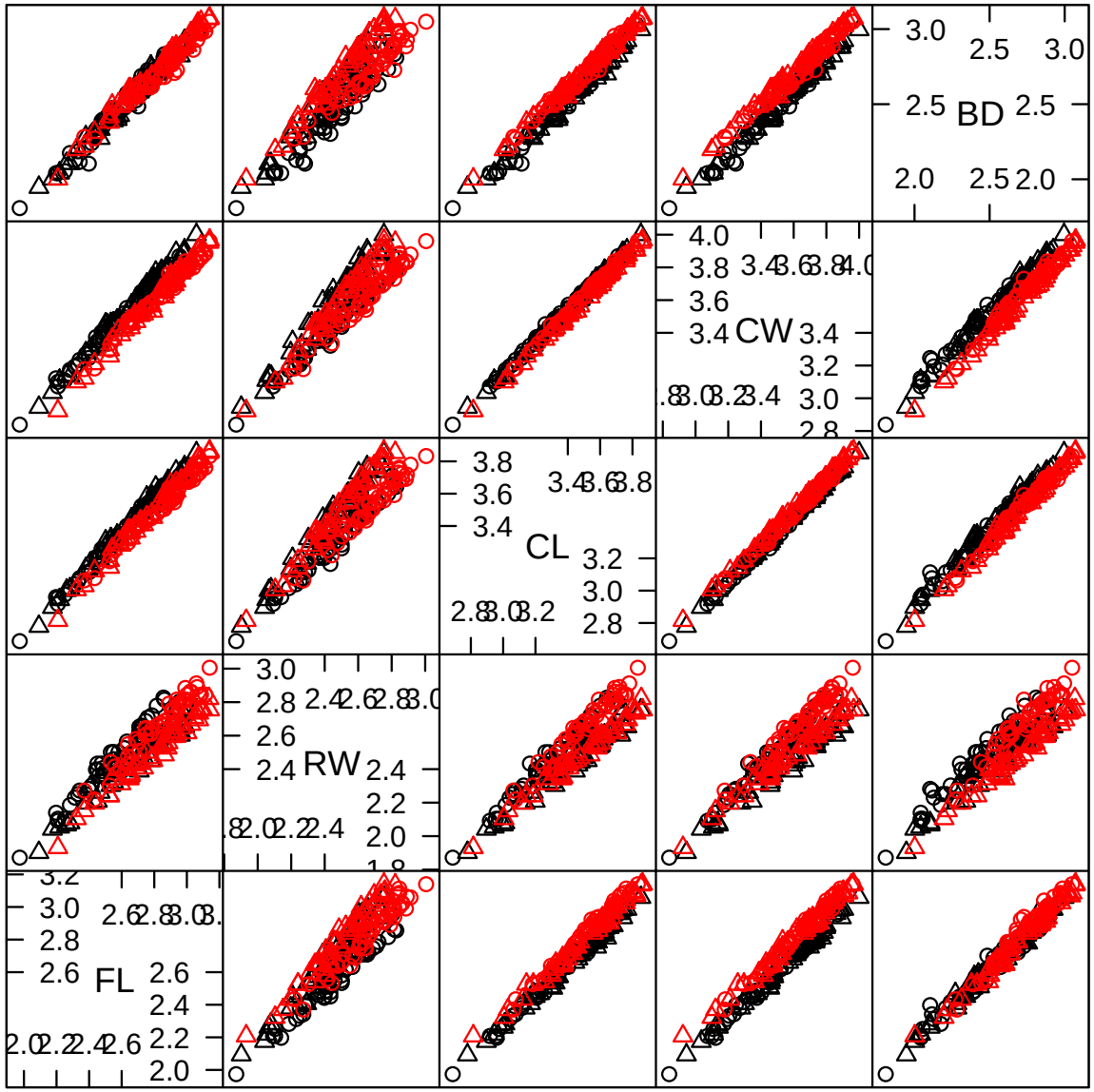
Example: Crabs

Looking at the Crabs data again.

```
library(MASS)
library(lattice)
data(crabs)

splom(~log(crabs[,4:8]),
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

Example: Crabs

circle/triangle is gender, black/red is species



Scatter Plot Matrix

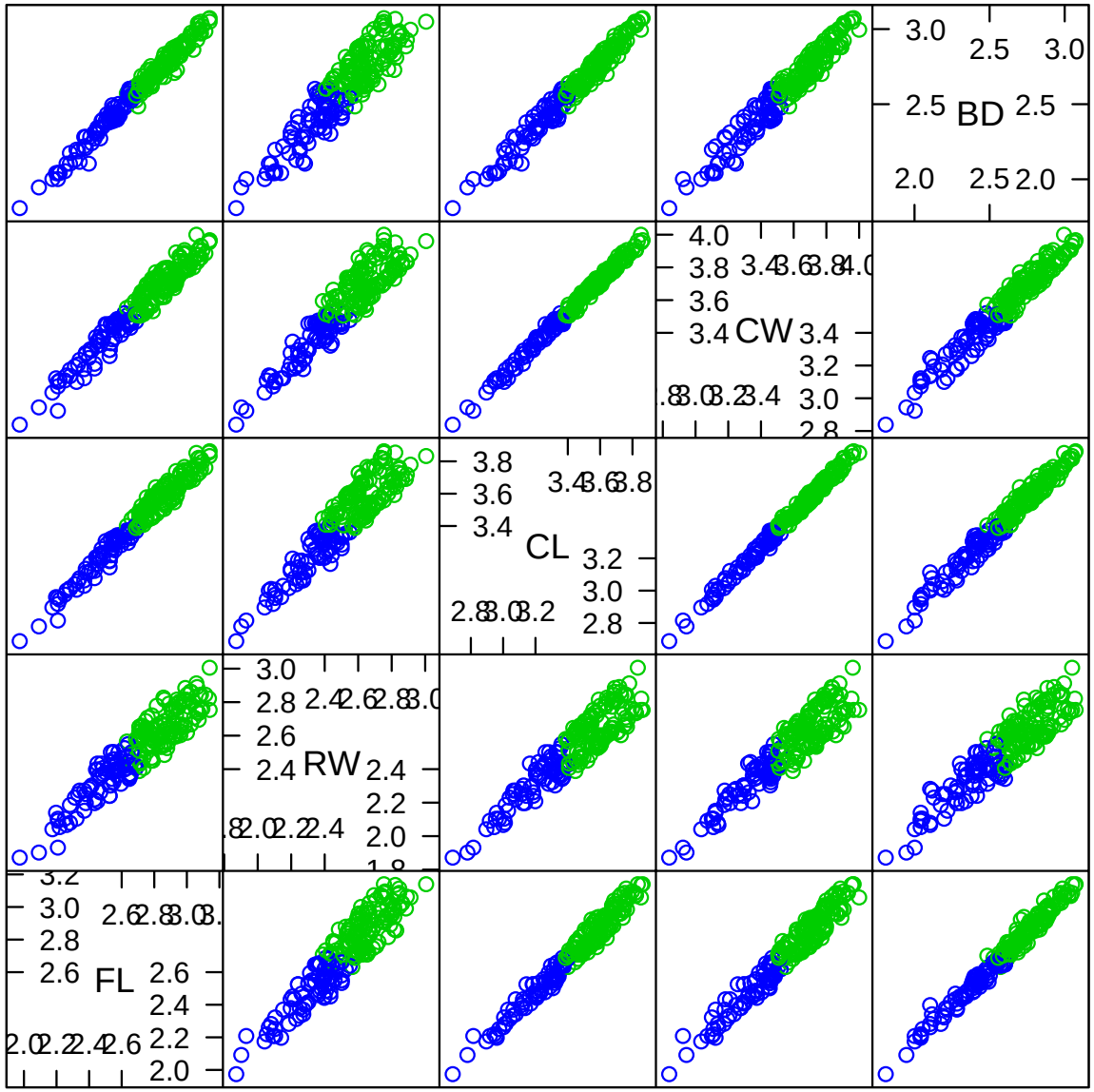
Example: Crabs

Apply kmeans with 2 clusters and plot results.

```
cl <- kmeans( log(crabs[,4:8]), 2, nstart=1, iter.max=10)

splom(~log(crabs[,4:8]),
      col=cl$cluster+2,
      main="blue/green is cluster finds big/small")
```

Example: Crabs

blue/green is cluster finds big/small



Scatter Plot Matrix

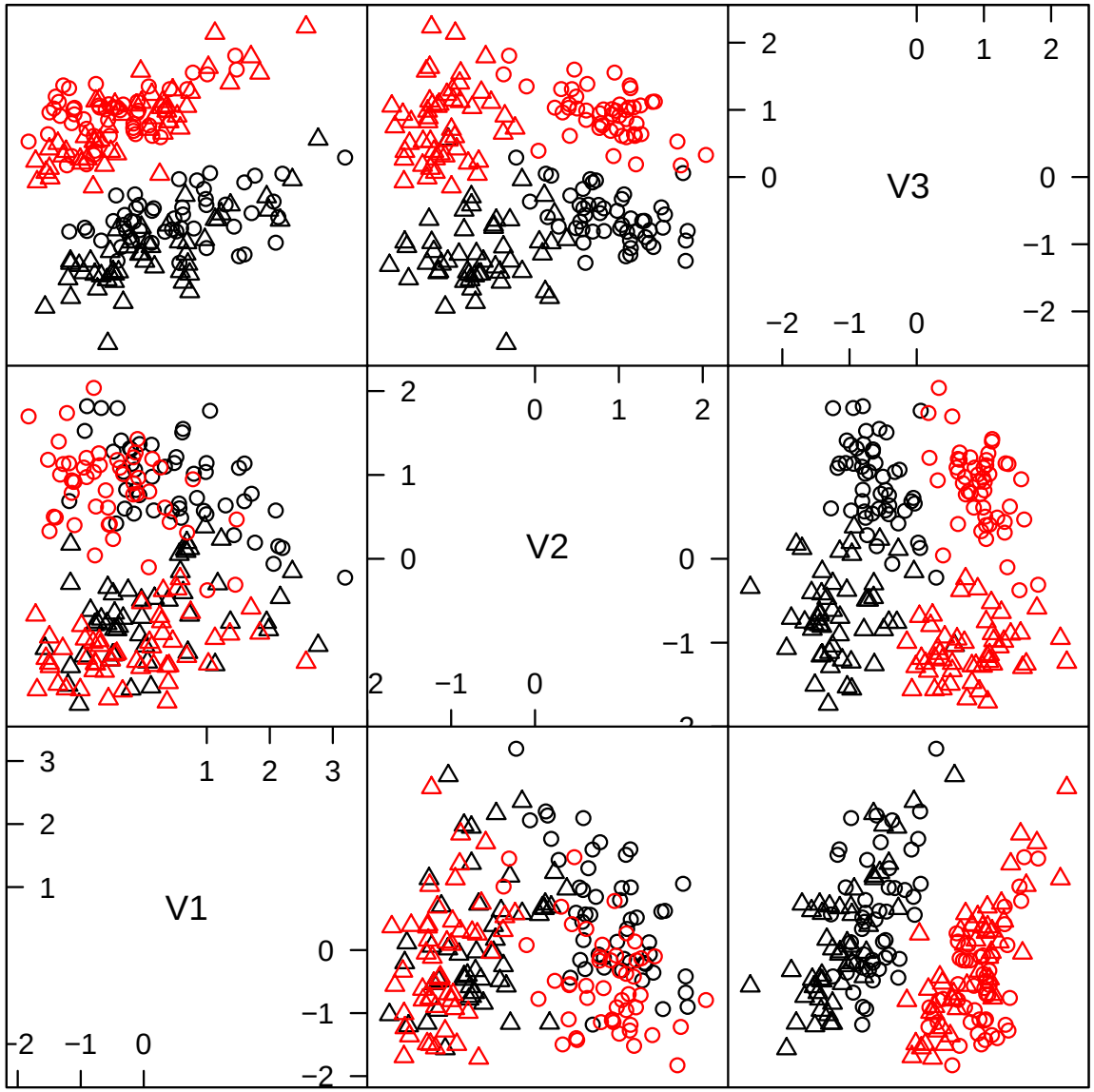
Example: Crabs

Sphere the data.

```
pcp <- princomp( log(crabs[,4:8]) )
spc <- pcp$scores %*% diag(1/pcp$sdev)
splom( ~spc[,1:3],
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

Example: Crabs

circle/triangle is gender, black/red is species



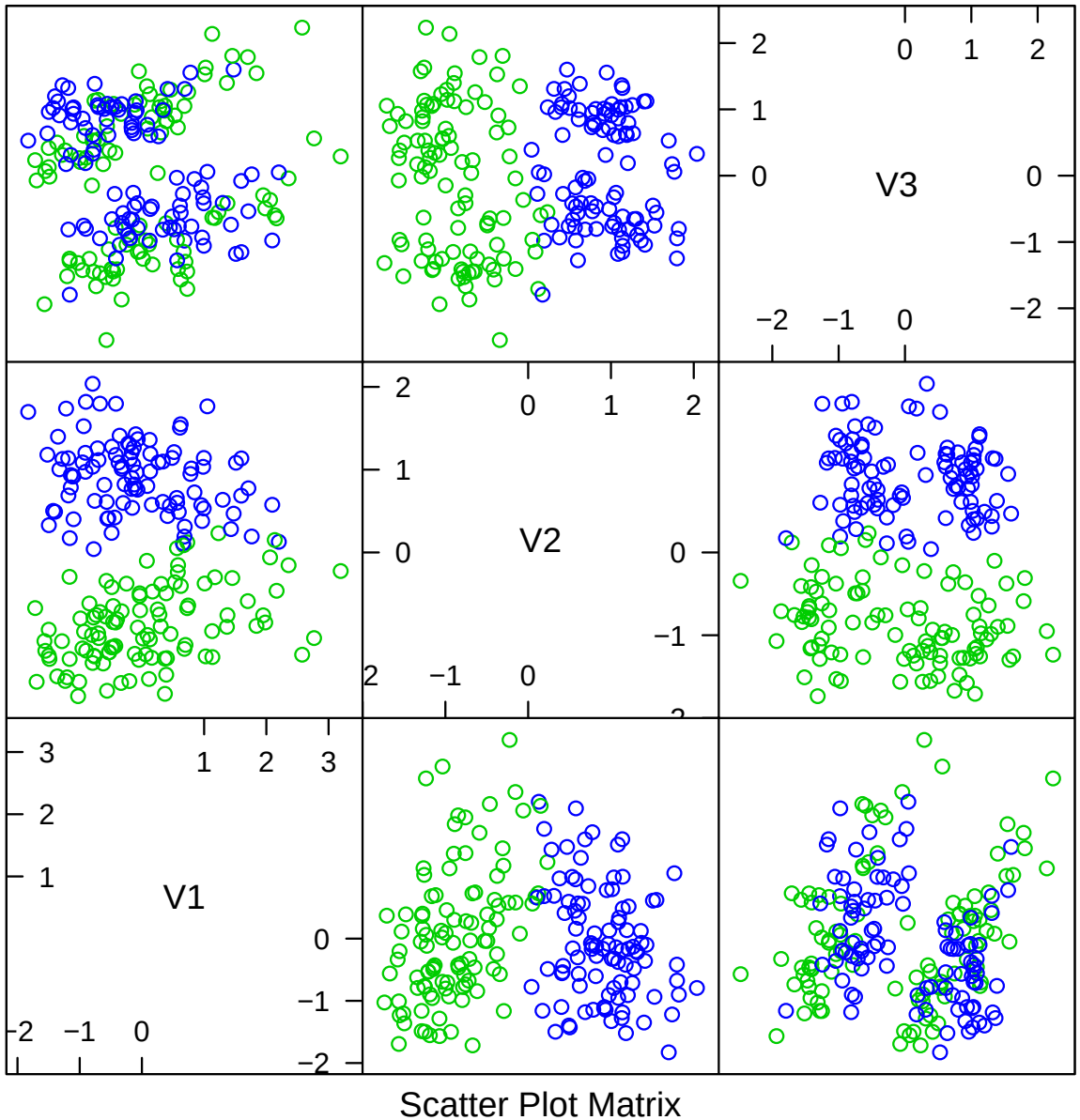
Scatter Plot Matrix

Example: Crabs

And apply K-means again.

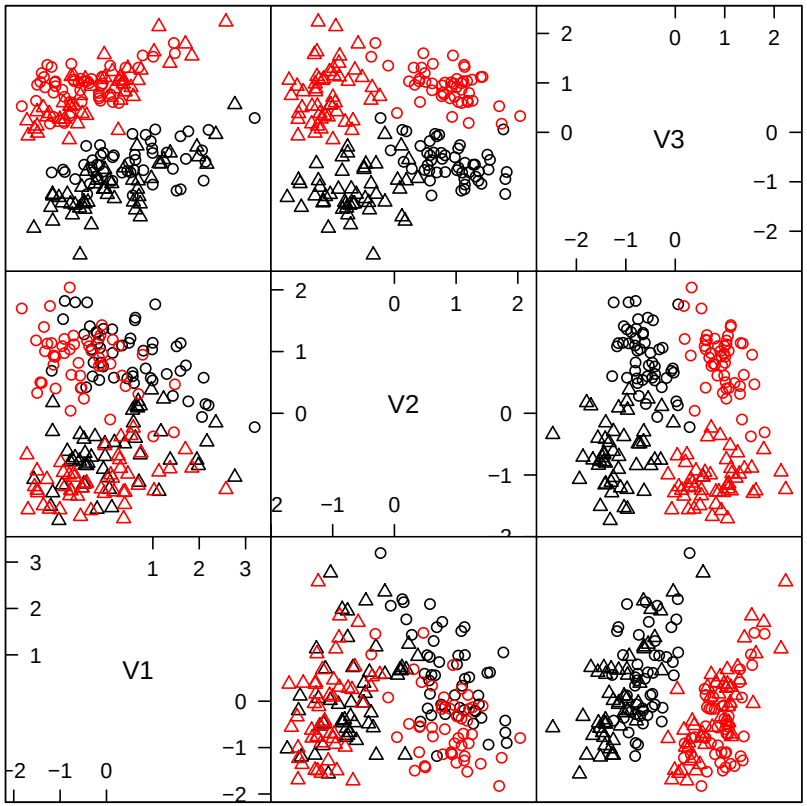
```
cl <- kmeans(spc, 2, nstart=1, iter.max=20)
splom( ~spc[,1:3],
      col=cl$cluster+2, main="blue/green is cluster")
```

Example: Crabs



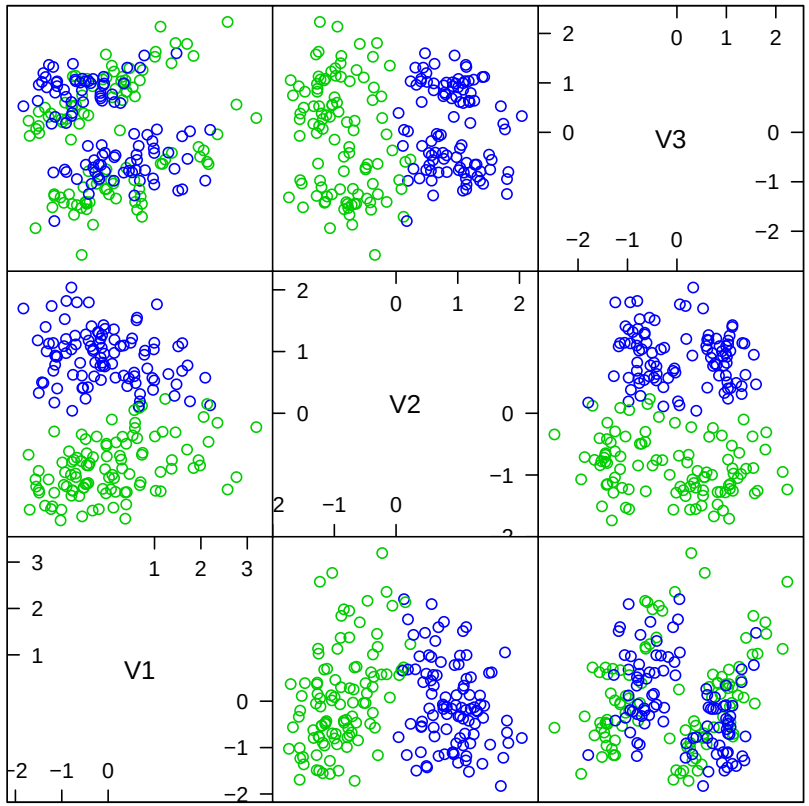
Example: Crabs

circle/triangle is gender, black/red is species



Scatter Plot Matrix

blue/green is cluster



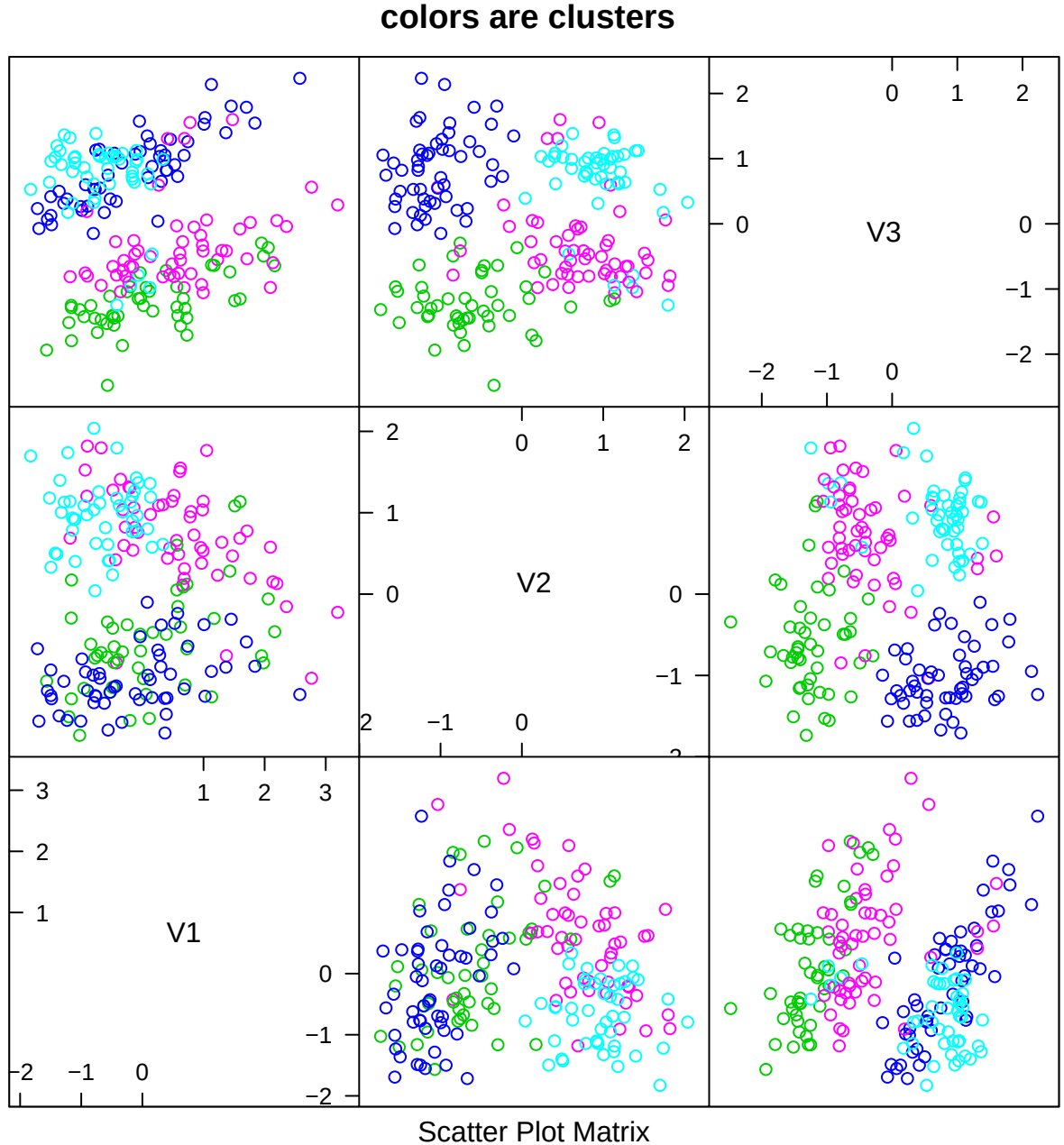
Scatter Plot Matrix

Discovers gender difference...

Results depends crucially on sphering the data first.

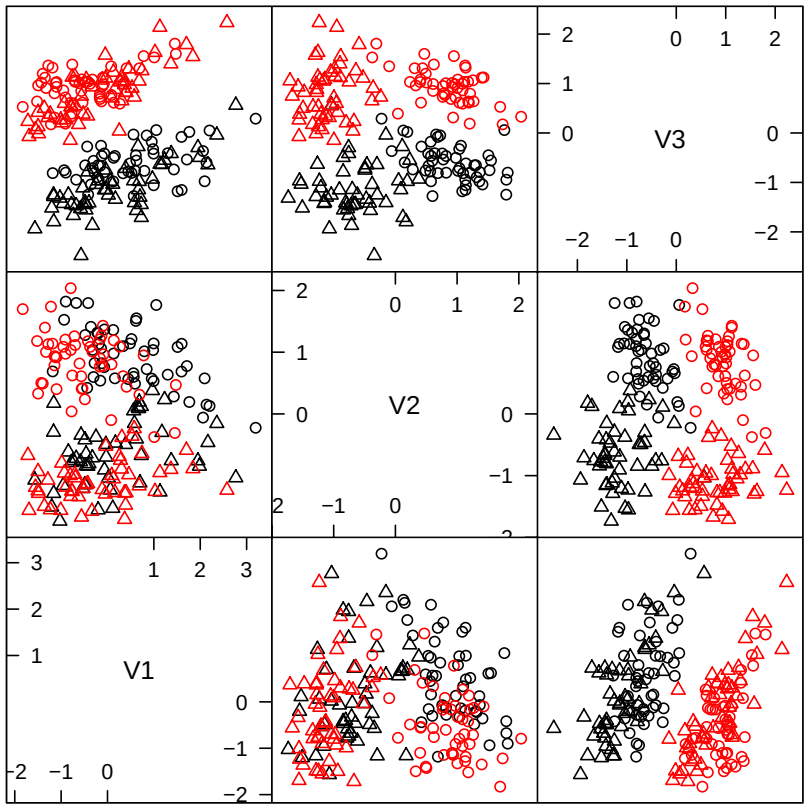
Example: Crabs

Using 4 cluster centers.



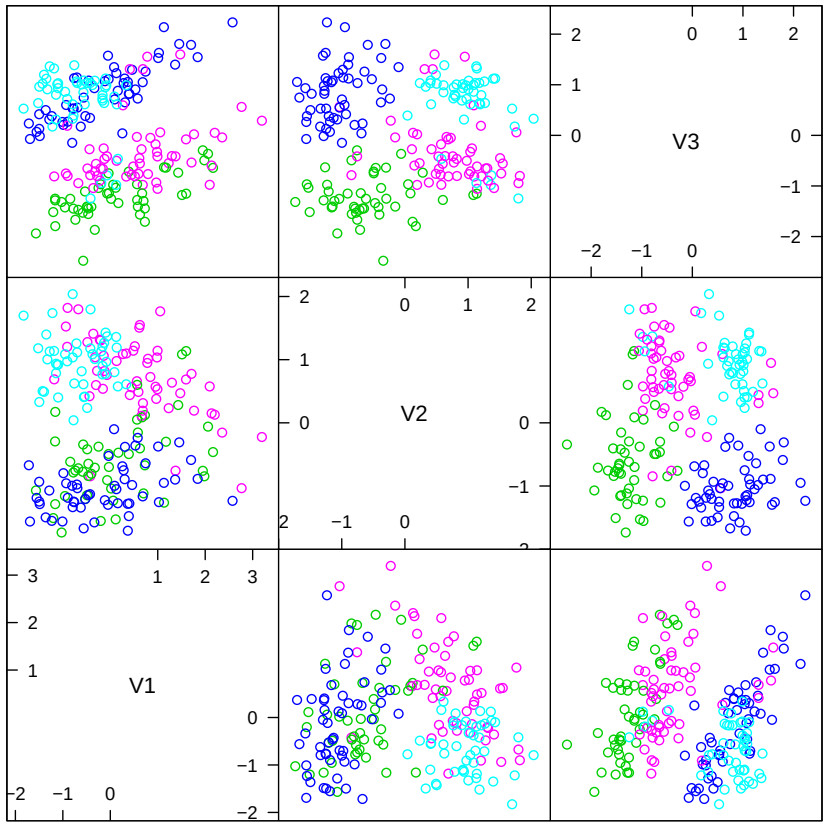
Example: Crabs

circle/triangle is gender, black/red is species



Scatter Plot Matrix

colors are clusters



Scatter Plot Matrix

Outline

Vector Quantisation

- Originally developed by the signal processing community for data compression (audio, image and video compression), the VQ idea has been picked up the statistics community and extended to tackle a variety of tasks (including clustering and classification).
- VQ is a simple idea for summarising data by use of codewords.
- The algorithm is very closely related to the K-means algorithm, yet works sequentially through the data when updating cluster centers.

Vector Quantisation

- Given p-dimensional data, a finite set of vectors $Y = \{y_1, \ldots, y_K\}$ of the same dimensionality must be found. Vectors y_k are called *codewords* and Y the *codebook*.
- All n observations are mapped to the indices of the code book using the following rule,

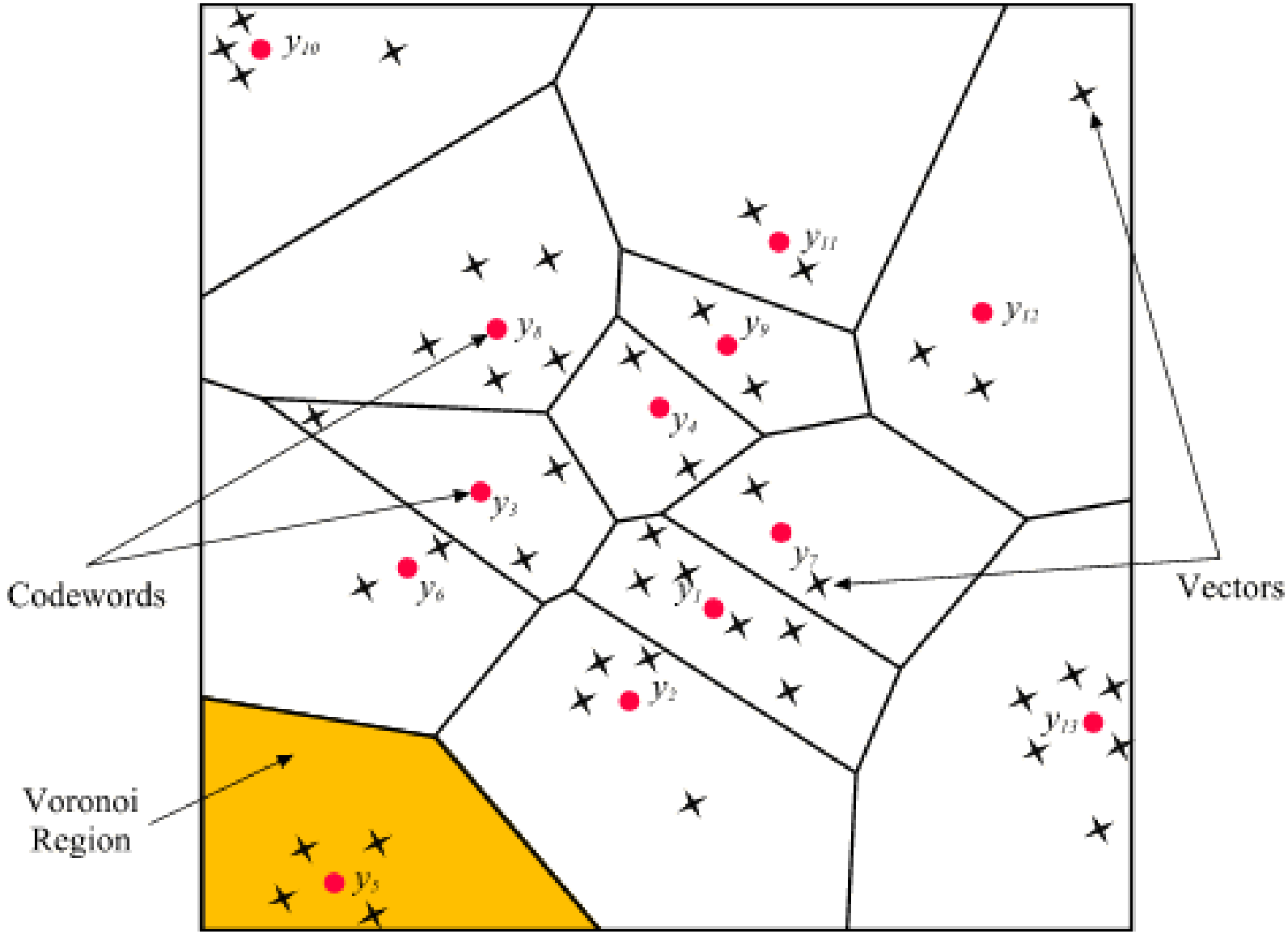
$$x_i \to y_k \quad \Leftrightarrow \quad |x_i - y_k| \leq |x_i - y_{k'}| \; \forall k'.$$

- Such a mapping induces a partition of $\mathbb{R}^p$ into Voronoi regions defined as

$$V_k = \{x \in \mathbb{R}^p : |x - y_k| \leq |x - y_{k'}| \forall k'\}$$

where $\cup_{k=1}^{K} V_k = \mathbb{R}^p$ and V_k's are disjoint except for boundaries.

Voronoi Regions



Finding a Useful Codebook

- As with K-means, a predefined number of K codewords must be found. They should be chosen to give the greatest compression in the data with minimal loss in data quality.
- Where we have more codewords than clusters, it is easy to see that we should simply place codewords at the center of areas of high density, i.e. good codebooks find cluster centers.

Vector Quantisation

The following iterative algorithm finds a good approximate solutions to this problem.

1. Randomly choose K observations to initialise the codebook.
2. Sample an observation x and let V_c be the Voronoi region where it falls.
3. Update the codebook

$$\begin{aligned} y_c &= y_c + \alpha(t)\,[x - y_c] \\ y_k &= y_k \;\; \forall k \neq c. \end{aligned}$$

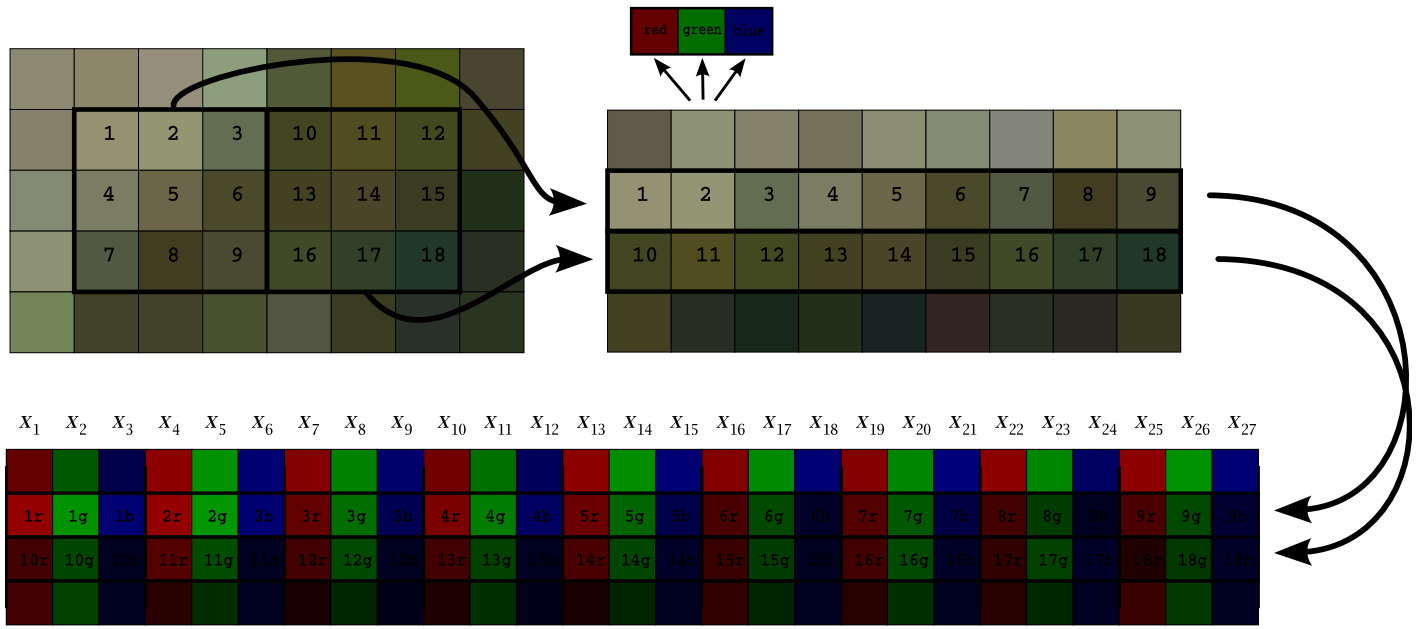
   $\alpha(t)$ quantifies the amount by which y_c moves towards of the x and decays over time to 0.
4. Repeat 2-3 until there is no change.
5. Return the codebook $Y = \{y_1, \ldots, y_K\}$

Compression

- For compression purposes, any observation $x \in \mathbb{R}^p$ is now just mapped to the set $\{1, \ldots, K\}$ of codewords, according to which Voronoi region the observation falls into.
- If a large number of observations $x_1, \ldots, x_n$ needs to be transferred, alternatively the vector of corresponding codewords in $\{1, \ldots, K\}^n$ can be transferred to achieve a compression (with a certain loss of information). Some audio and video codecs use this method.
- As with K-means, K must be specified. Increasing K 'improves the quality of the compressed image' but worsens the 'data compression rate', so there is a clear tradeoff. (For clustering, the choice of K is harder and does not have an entirely satisfactory answer).

Example: Image Compression

3×3 block VQ: View each block of 3×3 pixels as single observation



Example: Image Compression

Original image (24 bits/pixel, uncompressed size 1,402 kB)

Example: Image Compression

Codebook length 1024 (1.11 bits/pixel, total size 88kB)

Example: Image Compression

Codebook length 128 (0.78 bits/pixel, total size 50kB)

Example: Image Compression

Codebook length 16 (0.44 bits/pixel, total size 27kB)

Outline

Probabilistic Methods

- So far, we have found clusters in high-dimensional data by posing sensible partition based problems and hierarchical clustering problems which were tackled with heuristic approaches.
- Probabilistic methods attempt to find clusters in high-dimensional data using a model based approach by fitting mixture models to data.
- Though well founded in probabilistic arguments, such an approach comes at the expense of greater computation.
- Such methods can work well if good models are proposed (or if the distribution of the data is close to the proposed model in a suitable sense).
- We again need to specify/estimate the number of clusters K.

Mixture Models

- Probabilistic methods for clustering work by seeking to model the distribution of points in $\mathbb{R}^p$ using mixture models. In doing so, areas of high density (i.e. clusters) can be accurately described.
- Mixture models have densities of the form

$$f(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k f(\mathbf{x}|\phi_k)$$

  for some densities $f_k(\mathbf{x}|\phi_k)$ and priors over these densities $\pi_1, \ldots, \pi_K$ which satisfy $\pi_k \geq 0 \; \forall k$ and $\sum_{k=1}^{K} \pi_k = 1$.
- We want to estimate the unknown parameters $\theta = \{\pi_k, \phi_k\}_{k=1}^{K}$ given $\mathbf{x}_{1:n}$.

Mixture Models

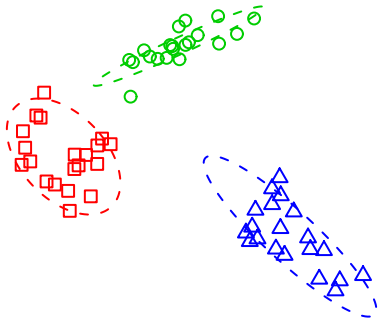
- To make things easier, let $f(\mathbf{x}|\theta_k) = f(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where

$$f(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^p \cdot |\boldsymbol{\Sigma}_k|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right\}.$$

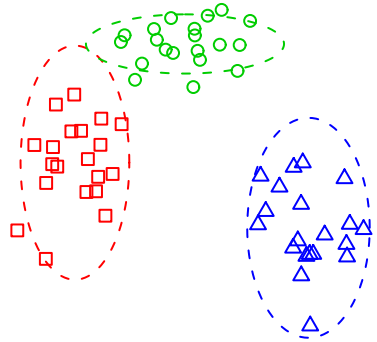
- Posing a Gaussian Mixture Model corresponds to assuming that each of the K clusters that we intend to model...
  - is Gaussian with different means $\boldsymbol{\mu}_k$ and covariance structures $\boldsymbol{\Sigma}_k$.
  - and each observation $\mathbf{x}$ comes from cluster k with probability π_k.
- Allowing each cluster to have its own mean and covariance structure allows greater flexibility in the model.

Gaussian Mixture Models: Examples

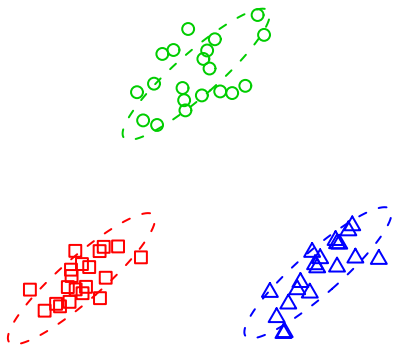
Different covariances



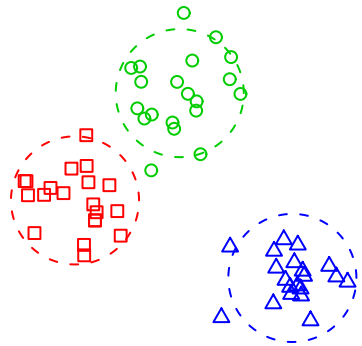
Different, but diagonal covariances



Identical covariances



Identical and spherical covariances



Fitting Gaussian Mixture Models

- To fit such a model, we need to estimate the parameters

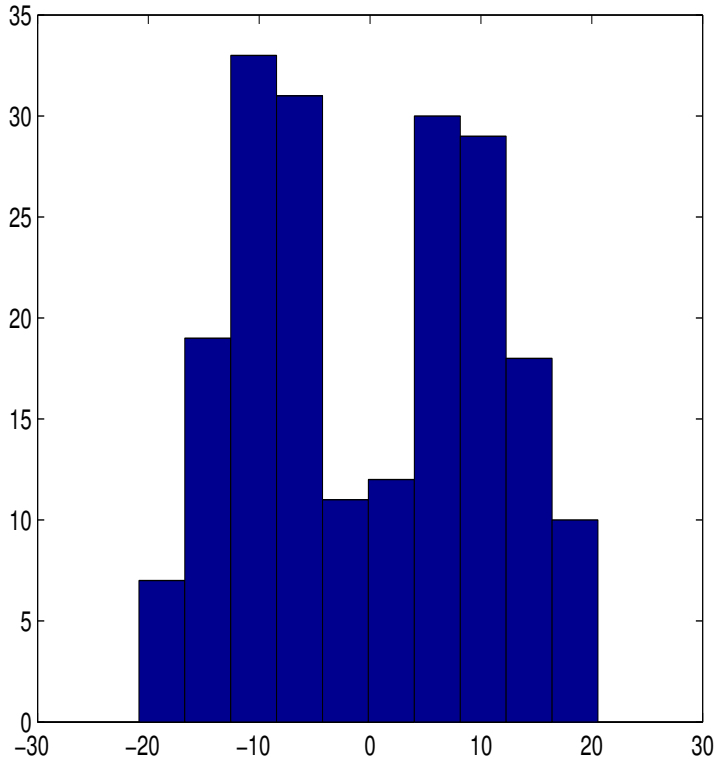
$$\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

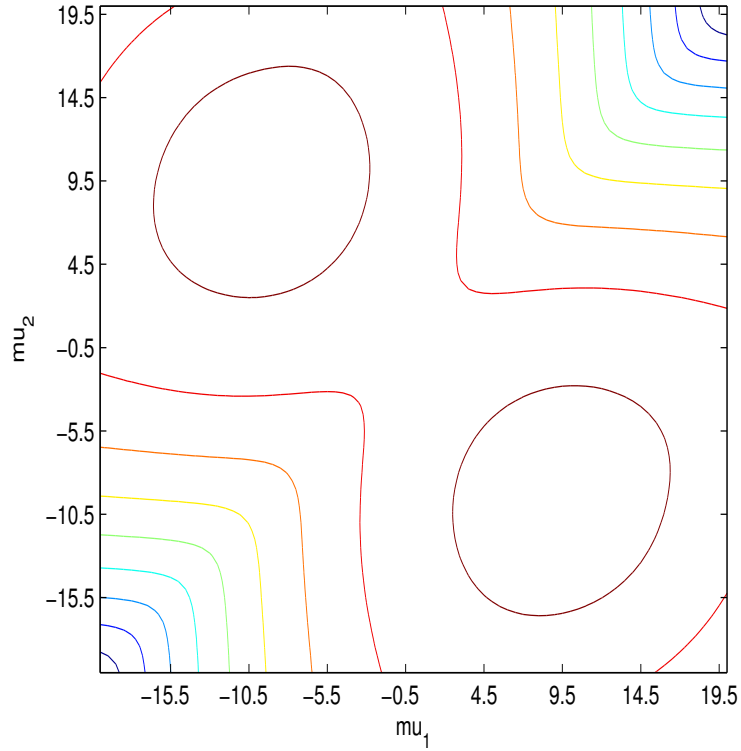
  from the data.
- We can do this by maximum likelihood choosing θ to maximise $L(\theta) = \prod_{i=1}^{n} f(\mathbf{x}_i|\theta)$ or equivalently $\ell(\theta) = \sum_{i=1}^{n} \log f(\mathbf{x}_i|\theta)$ where

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(\pi_1 f_{\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1}(\mathbf{x}_i) + \ldots + \pi_K f_{\boldsymbol{\mu}_K,\boldsymbol{\Sigma}_K}(\mathbf{x}_i)\right).$$

- Differentiating to maximise such a log-likelihood analytically or even numerically is difficult as there are too many unknowns to handle simultaneously.
- The Expectation-Maximisation (EM) Algorithm is a very popular method to help find maximum likelihood estimates in the presence of unobserved variables.

Likelihood Surface for a Simple Example





(left) $n = 200$ data points from a mixture of two 1D Gaussians with $\pi_1 = \pi_2 = 0.5$, $\sigma_1 = \sigma_2 = 5$ and $\mu_1 = -\mu_2 = 10$. (right) Log-Likelihood surface $\ell(\mu_1, \mu_2)$, all the other parameters being assumed known.

The EM Algorithm

- EM is a very popular approach to maximize $\ell(\theta)$ in this missing data context.
- The key idea is to introduce explicitly the unobserved cluster labels z_i which indicate from which cluster data $\mathbf{x}_i$ is coming from.
- If the cluster labels where known then we would estimate θ by maximizing the so-called complete likelihood

$$\begin{aligned} \ell_c(\theta) &= \sum_{i=1}^{n} \log\ p(\mathbf{x}_i, z_i | \theta) \\ &= \sum_{i=1}^{n} \log\ \pi_{z_i} f(\mathbf{x}_i | \phi_{z_i}) \end{aligned}$$

Maximization of Complete Likelihood

- We have

$$\begin{aligned}\ell_c(\theta) &= \sum_{k=1}^{K}\left(\sum_{i:z_i=k}\log\ \pi_{z_i} f(\mathbf{x}_i|\phi_{z_i})\right)\\ &= \sum_{k=1}^{K} n_k \log(\pi_k) + \sum_{i:z_i=k}\log f(\mathbf{x}_i|\phi_k)\end{aligned}$$

  where $n_k = \sum_{i:z_i=k} 1$ is the number of observations assigned to cluster k.
- We would obtain the MLE for the complete likelihood

$$\begin{aligned}\widehat{\pi}_k &= \frac{n_k}{n},\\ \widehat{\phi}_k &= \arg\max_{\phi_k} \sum_{i=1:z_i=k}^{n} \log f(\mathbf{x}_i|\phi_k)\end{aligned}$$

Finite Mixture of Scalar Gaussians

- In this case, $\phi = \left(\mu, \sigma^2\right)$

$$f\left(\mathbf{x}|\phi\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(\mathbf{x}-\mu\right)^2}{2\sigma^2}\right)$$

  and $\theta = \left\{\pi_k, \mu_k, \sigma^2\right\}_{k=1}^{K}$.
- The resulting MLE estimate of the complete likelihood is

$$\begin{aligned}
\widehat{\pi}_k &= \frac{n_k}{n}, \\
\widehat{\mu}_k &= \frac{1}{n_k} \sum_{i=1:z_i=k}^{n} \mathbf{x}_i, \\
\widehat{\sigma}_k^2 &= \frac{1}{n_k} \sum_{i=1:z_i=k}^{n} \left(\mathbf{x}_i - \widehat{\mu}_k\right)^2
\end{aligned}$$

- Problem: We don't have access to the cluster labels!

Expectation-Maximization

EM is an iterative algorithm which generates a sequence of estimates $\left\{\theta^{(t)}\right\}$ such that

$$\ell\left(\theta^{(t)}\right) \geq \ell\left(\theta^{(t-1)}\right).$$

At iteration t, we compute

$$\begin{aligned}
&\mathcal{F}\left(\theta, \theta^{(t-1)}\right) \\
=&\mathbb{E}\left[\ell_c\left(\theta\right) \middle| \mathbf{x}_{1:n}, \theta^{(t-1)}\right] \\
=&\sum_{z_{1:n} \in \{1,2,\ldots,K\}^n} p\left(z_{1:n} \middle| \mathbf{x}_{1:n}, \theta^{(t-1)}\right)\left(\sum_{i=1}^{n} \log\ p\left(\mathbf{x}_i, z_i \middle| \theta\right)\right) \\
=&\sum_{i=1}^{n}\sum_{k=1}^{K} p\left(z_i = k \middle| \mathbf{x}_i, \theta^{(t-1)}\right) \log\ p\left(\mathbf{x}_i, z_i = k \middle| \theta\right)
\end{aligned}$$

and set

$$\theta^{(t)} = \arg\max_{\theta}\ \mathcal{F}\left(\theta, \theta^{(t-1)}\right)$$

Expectation-Maximization

We have

$$\begin{aligned}
\mathcal{F}\left(\theta, \theta^{(t-1)}\right) &= \sum_{i=1}^{n} \sum_{k=1}^{K} p\left(z_i = k | \mathbf{x}_i | \theta^{(t-1)}\right) \log\ p\left(\mathbf{x}_i, z_i = k | \theta\right) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right) \{\log\ \pi_k + \log f\left(\mathbf{x}_i | \phi_k\right)\} \\
&= \sum_{k=1}^{K} \left(\sum_{i=1}^{n} p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right)\right) \{\log \pi_k + \log f\left(\mathbf{x}_i | \phi_k\right)\}
\end{aligned}$$

We obtain

$$\begin{aligned}
\widehat{\pi}_k^{(t)} &= \frac{\sum_{i=1}^{n} p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right)}{n}, \\
\phi_k^{(t)} &= \arg\max_{\phi_k} \sum_{i=1}^{n} p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right) \log f\left(\mathbf{x}_i | \phi_k\right)
\end{aligned}$$

Finite mixture of scalar Gaussians

In this case, the EM algorithm iterates

$$\widehat{\pi}_k^{(t)} = \frac{\sum_{i=1}^n p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right)}{n}$$

$$\widehat{\mu}_k^{(t)} = \frac{\sum_{i=1}^n \mathbf{x}_i p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right)}{\sum_{i=1}^n p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right)},$$

$$\widehat{\sigma}_k^{2\,(t)} = \frac{\sum_{i=1}^n p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right)\left(\mathbf{x}_i - \widehat{\mu}_k^{(t)}\right)^2}{\sum_{i=1}^n p\left(z_i = k | \mathbf{x}_i, \theta^{(t-1)}\right)}.$$

with

$$p\left(z_i = k | \mathbf{x}_i, \theta\right) = \frac{\pi_k f\left(\mathbf{x}_i | \phi_k\right)}{\sum_\ell \pi_\ell f\left(\mathbf{x}_i | \phi_\ell\right)}$$

Proof of Convergence for EM Algorithm

Proposition: $\ell\left(\theta^{(t+1)}\right) \geq \ell\left(\theta^{(t)}\right)$ for $\theta^{(t+1)} = \arg\max_{\theta} \mathcal{F}\left(\theta, \theta^{(t)}\right)$.

Proof: We have

$$p\left(z_{1:n} \middle| \theta, \mathbf{x}_{1:n}\right) = \frac{p\left(\mathbf{x}_{1:n}, z_{1:n} \middle| \theta\right)}{p\left(\mathbf{x}_{1:n} \middle| \theta\right)} \Leftrightarrow p\left(\mathbf{x}_{1:n} \middle| \theta\right) = \frac{p\left(\mathbf{x}_{1:n}, z_{1:n} \middle| \theta\right)}{p\left(z_{1:n} \middle| \theta, \mathbf{x}_{1:n}\right)}$$

thus

$$\ell(\theta) = \log p\left(\mathbf{x}_{1:n} \middle| \theta\right) = \log p\left(\mathbf{x}_{1:n}, z_{1:n} \middle| \theta\right) - \log p\left(z_{1:n} \middle| \theta, \mathbf{x}_{1:n}\right)$$

and for any value $\theta^{(t)}$

$$\begin{aligned} \ell(\theta) &= \underbrace{\sum_{z_{1:n}} p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right) \log p\left(\mathbf{x}_{1:n}, z_{1:n} \middle| \theta\right)}_{=\mathcal{F}\left(\theta, \theta^{(t)}\right)} \\ &\quad - \sum_{z_{1:n}} p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right) \log p\left(z_{1:n} \middle| \theta, \mathbf{x}_{1:n}\right). \end{aligned}$$

Proof of Convergence for EM Algorithm

We want to show that $\ell\left(\theta^{(t+1)}\right) \geq \ell\left(\theta^{(t)}\right)$ for the EM, so if we prove that

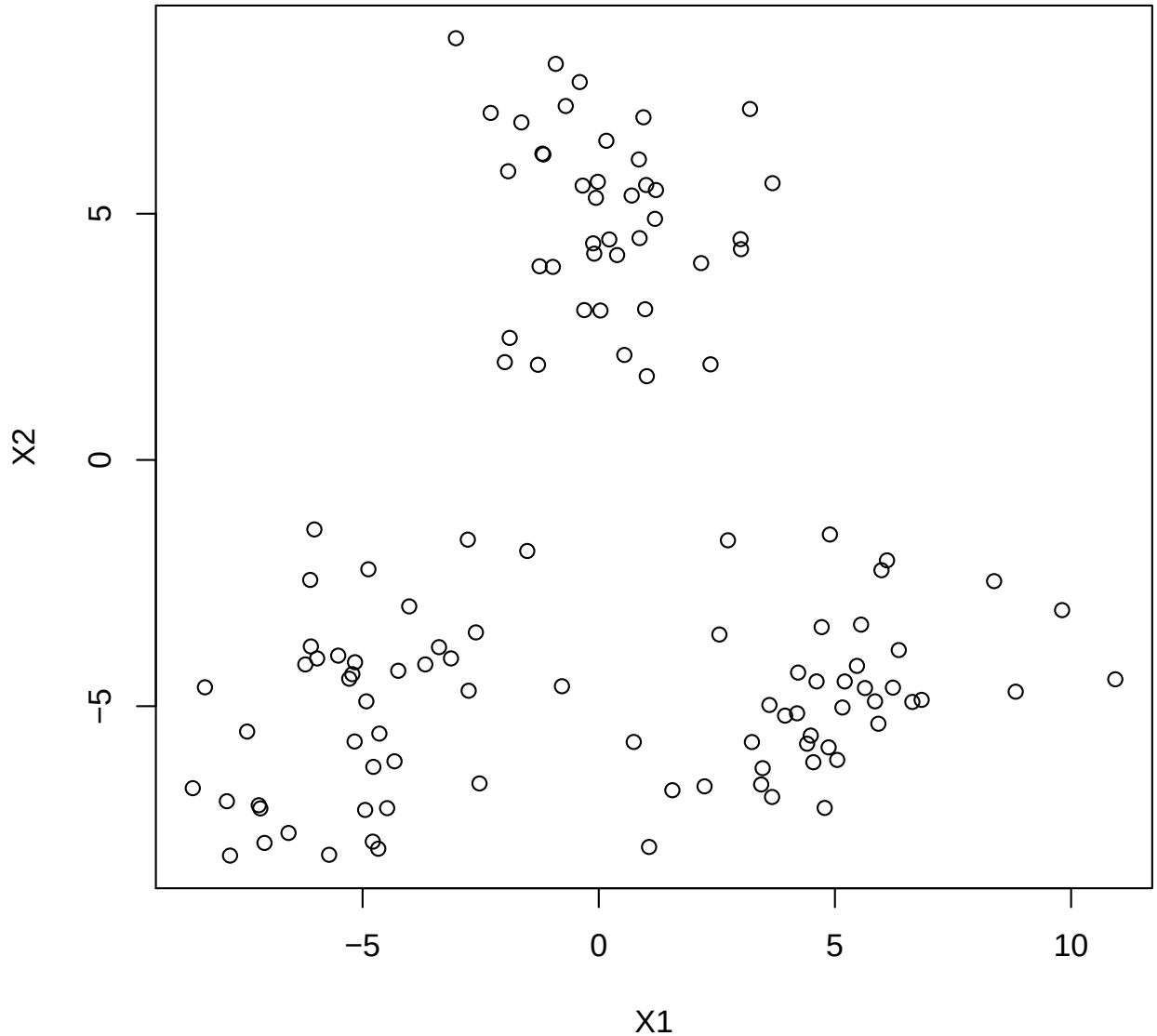
$$\begin{aligned} & \sum_{z_{1:n}} p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right) \log p\left(z_{1:n} \middle| \theta^{(t+1)}, \mathbf{x}_{1:n}\right) \\ \leq & \sum_{z_{1:n}} p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right) \log p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right) \end{aligned}$$

then we are done. We have

$$\begin{aligned} & \sum_{z_{1:n}} p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right) \log \frac{p\left(z_{1:n} \middle| \theta^{(t+1)}, \mathbf{x}_{1:n}\right)}{p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right)} \\ \leq & \log \sum_{z_{1:n}} p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right) \frac{p\left(z_{1:n} \middle| \theta^{(t+1)}, \mathbf{x}_{1:n}\right)}{p\left(z_{1:n} \middle| \theta^{(t)}, \mathbf{x}_{1:n}\right)} \quad \text{(Jensen)} \\ = & \log 1 = 0. \end{aligned}$$

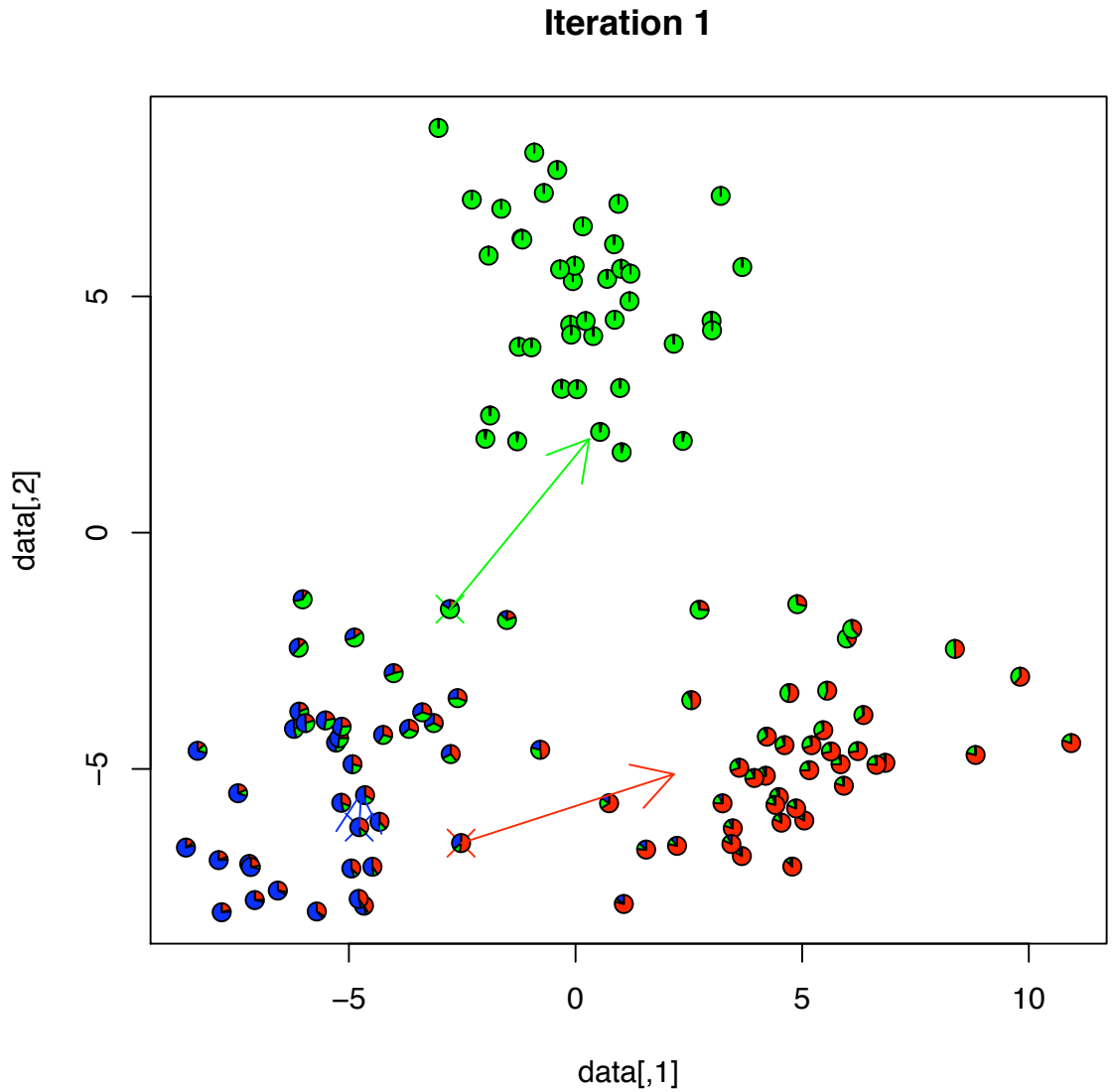
Example: Mixture of 3 Gaussians

An example with 3 clusters.



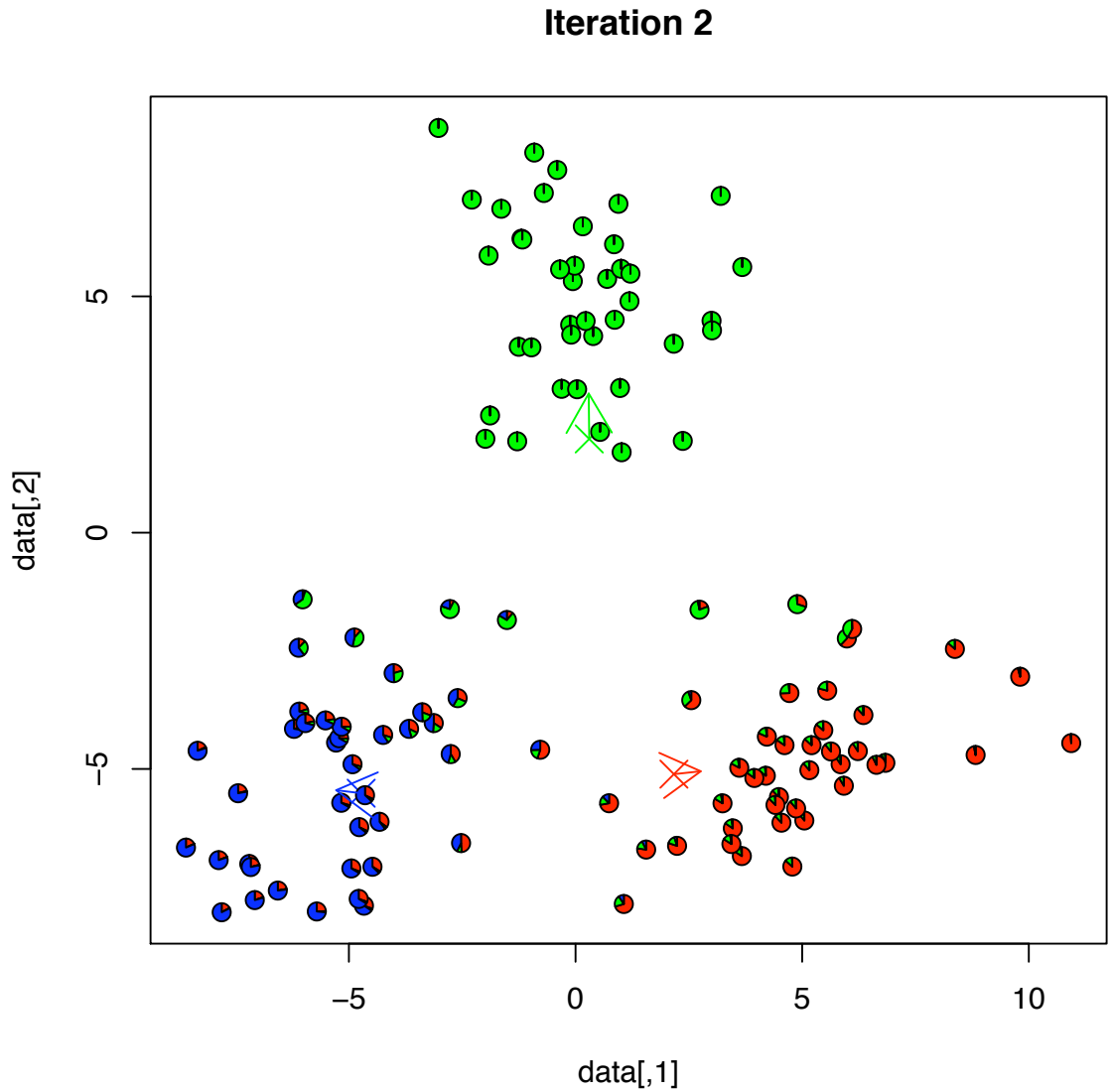
Example: Mixture of 3 Gaussians

After 1st E and M step.



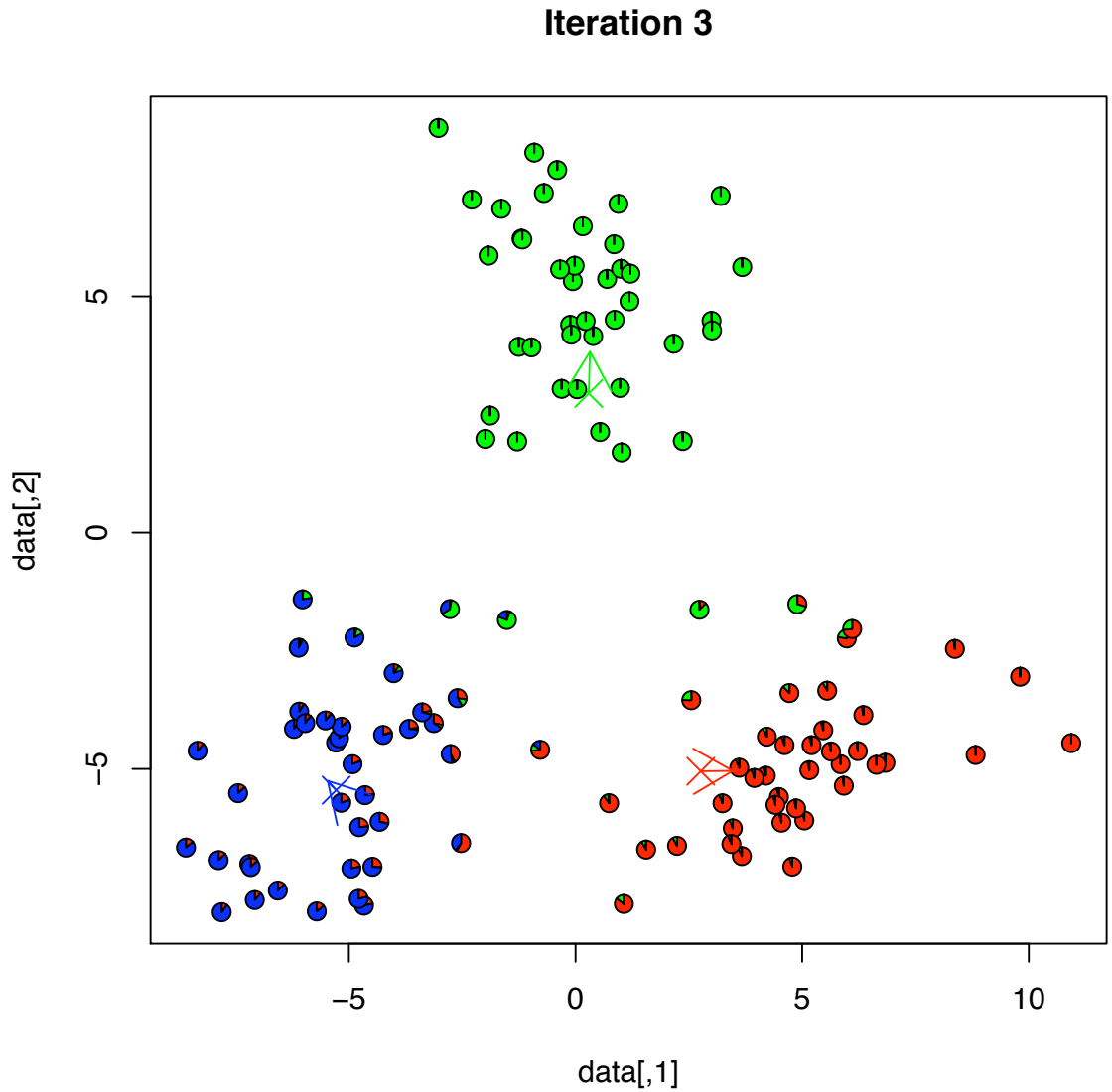
Example: Mixture of 3 Gaussians

After 2nd E and M step.



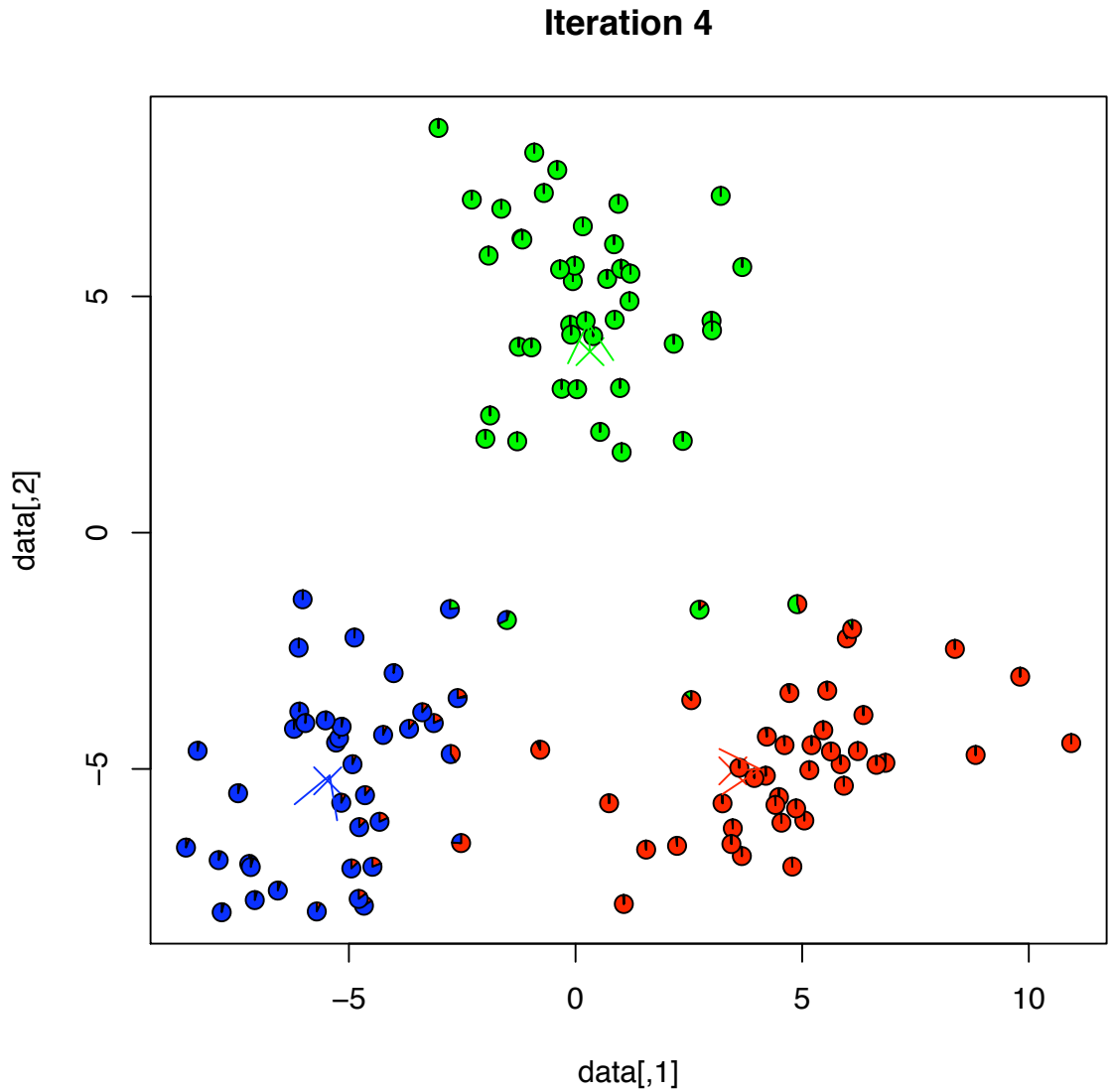
Example: Mixture of 3 Gaussians

After 3rd E and M step.



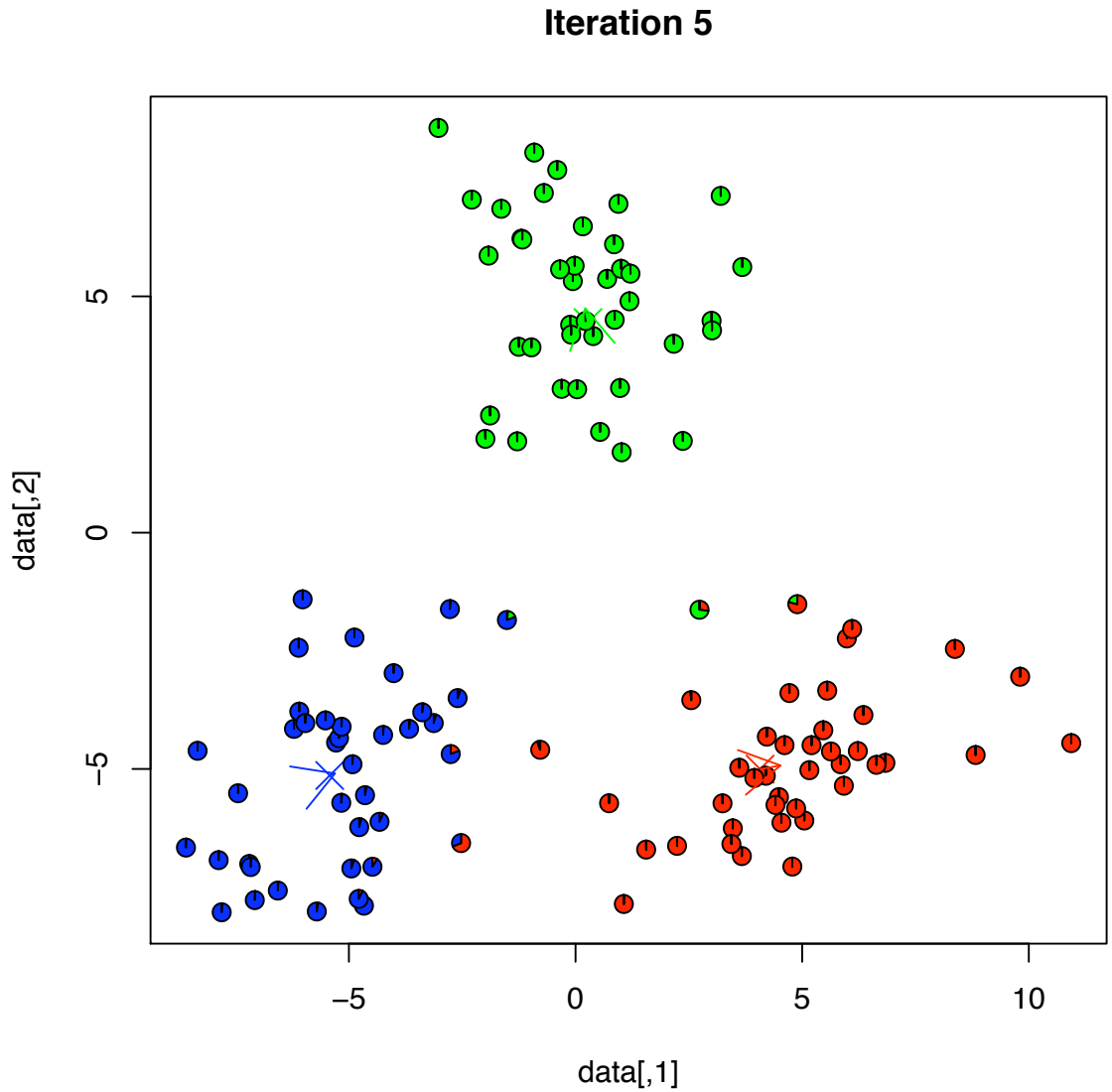
Example: Mixture of 3 Gaussians

After 4th E and M step.



Example: Mixture of 3 Gaussians

After 5th E and M step.



Pros and Cons of the EM Algorithm

Some good things about EM

- no learning rate (step-size) parameter
- automatically enforces parameter constraints
- very fast for low dimensions
- each iteration guaranteed to improve likelihood

Some bad things about EM

- can get stuck in local minima so multiple starts are recommended
- can be slower than conjugate gradient (especially near convergence)
- requires expensive inference step