SC4/SM4 Data Mining and Machine Learning
Dimensionality Reduction

Dino Sejdinovic
Department of Statistics
Oxford

Slides and other materials available at:
http://www.stats.ox.ac.uk/~sejdinov/dmml

Course Structure

- MMath Part C & MSc in Applied Statistics

Lectures:

- Tuesdays 14:00-15:00, LG.01.
- Thursdays 12:00-13:00, LG.01.

MSc:

- 4 problem sheets, discussed at the classes: Mondays 11:00-12:00 (weeks 3,5,7,8), LG.01.
- Practicals: Fridays 14:00-16:00 (weeks 5 and **8 - group assessed**), LG.02.

Part C:

- 4 problem sheets, **solutions due Mondays 10:00 in weeks 3,5,7,8**.
- Class Tutors: Jovana Mitrovic and Leonard Hasenclever.
- Teaching Assistants: Leon Law and Fadhel Ayed.
- Classes (Group I): Wednesdays 14:30-16:00 (weeks 3,5,7,8), LG.04.
- Classes (Group II): Wednesdays 17:00-18:30 (weeks 3,5,7,8), LG.04.
- Classes (Group III): Fridays 16:30-18:00 (weeks 3,5,7,8), LG.04.
- Please **sign up for the classes today** on the sign up sheet!

Course Aims

1. Have ability to identify and use appropriate methods and models for given data and task.
2. Have ability to use the relevant software packages to analyse data, interpret results, and evaluate methods.
3. Understand the statistical theory framing machine learning and data mining.
4. Able to construct appropriate models and derive learning algorithms for given data and task.

What is Data Mining?

Oxford Dictionary

The practice of examining large pre-existing databases in order to **generate new information**.

Encyclopaedia Britannica

Also called **knowledge discovery** in databases, in computer science, the process of discovering **interesting and useful patterns and relationships** in large volumes of data.

What is Machine Learning?

Arthur Samuel, 1959

Field of study that gives computers the ability to **learn** without being explicitly programmed.

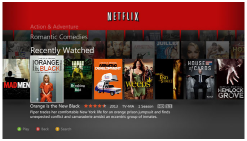
Tom Mitchell, 1997

Any computer program that **improves its performance** at some task **through experience**.

Kevin Murphy, 2012

To develop methods that can **automatically** detect **patterns in data**, and then to use the uncovered patterns to **predict** future data or other outcomes of interest.

What is Machine Learning?



recommender systems

machine translation

self-driving cars

image recognition

DQN Atari games

AlphaGo

Types of Machine Learning

Supervised learning

- Data contains “labels”: every example is an input-output pair
- classification, regression
- Goal: **prediction on new examples**

Unsupervised learning

- Extract key features of the “unlabelled” data
- clustering, signal separation, density estimation
- Goal: **representation, hypothesis generation, visualization**

Types of Machine Learning

Semi-supervised Learning

A database of examples, only a small subset of which are labelled.

Multi-task Learning

A database of examples, each of which has multiple labels corresponding to different prediction tasks.

Reinforcement Learning

An agent acting in an environment, given rewards for performing appropriate actions, learns to maximize their reward.

Software

- R
- Python: scikit-learn, mlpy, Theano
- Weka, mlpack, Torch, Shogun, TensorFlow...
- Matlab/Octave

OxWaSP

Oxford-Warwick CDT

- Doctoral Training in Next Generational Statistical Science: theory, methods and applications of Statistical Science for 21st Century data-intensive environments and large-scale models.
- Full DPhil/PhD studentships available for UK students
- Website for prospective students.
- **Deadline: January 20, 2017**

Unsupervised Learning: Dimensionality Reduction

Unsupervised Learning

Goals:

- Find the variables that summarise the data / capture relevant information.
- Discover informative ways to visualise the data.
- Discover the subgroups among the observations.

It is often much easier to obtain unlabeled data than labeled data!

Exploratory Data Analysis

Notation

- Data consists of p variables (features/attributes/dimensions) on n examples (items/observations).
- $\mathbf{X} = (x_{ij})$ is a $n \times p$-matrix with $x_{ij} :=$ the j-th variable for the i-th example

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

- Denote the i-th data item by $x_i \in \mathbb{R}^p$ (we will treat it as a column vector: it is the transpose of the i-th row of $\mathbf{X}$).
- Assume $x_1, \dots, x_n$ are **independently and identically distributed** samples of a **random vector** X over $\mathbb{R}^p$. The j-th dimension of X will be denoted $X^{(j)}$.

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 according to their 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. Each specimen has measurements on:

- the width of the frontal lobe `FL`,
- the rear width `RW`,
- the length along the carapace midline `CL`,
- the maximum width `CW` of the carapace, and
- the body depth `BD` in mm.

photo from: inaturalist.org

in addition to colour/species and sex (we will later view these as labels, but will ignore for now).

Crabs Data

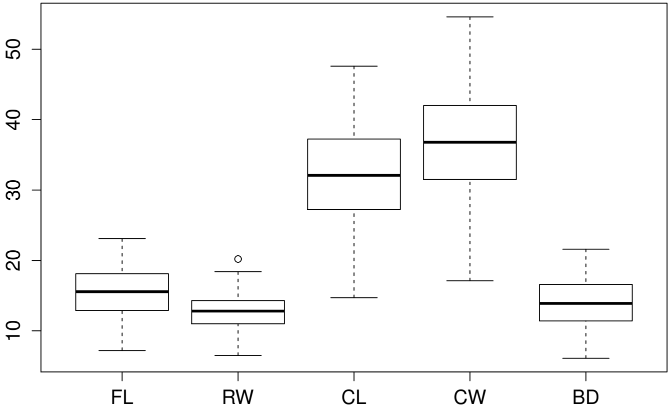
```
## load package MASS containing the data
library(MASS)

## extract variables we will look at
varnames<-c('FL','RW','CL','CW','BD')
Crabs <- crabs[,varnames]

## look at raw data
Crabs
```

Crabs Data

```
## look at raw data
Crabs

        FL   RW   CL   CW   BD
   1    8.1  6.7 16.1 19.0  7.0
   2    8.8  7.7 18.1 20.8  7.4
   3    9.2  7.8 19.0 22.4  7.7
   4    9.6  7.9 20.1 23.1  8.2
   5    9.8  8.0 20.3 23.0  8.2
   6   10.8  9.0 23.0 26.5  9.8
   7   11.1  9.9 23.8 27.1  9.8
   8   11.6  9.1 24.5 28.4 10.4
   9   11.8  9.6 24.2 27.8  9.7
   10  11.8 10.5 25.2 29.3 10.3
   11  12.2 10.8 27.3 31.6 10.9
   12  12.3 11.0 26.8 31.5 11.4
   13  12.6 10.0 27.7 31.7 11.4
   14  12.8 10.2 27.2 31.8 10.9
   15  12.8 10.9 27.4 31.5 11.0
   16  12.9 11.0 26.8 30.9 11.4
   17  13.1 10.6 28.2 32.3 11.0
   18  13.1 10.9 28.3 32.4 11.2
   19  13.3 11.1 27.8 32.3 11.3
   20  13.9 11.1 29.2 33.3 12.1
```

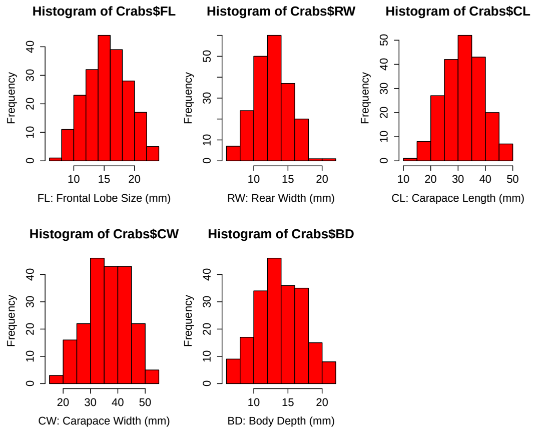
Univariate Boxplots

```
boxplot(Crabs)
```



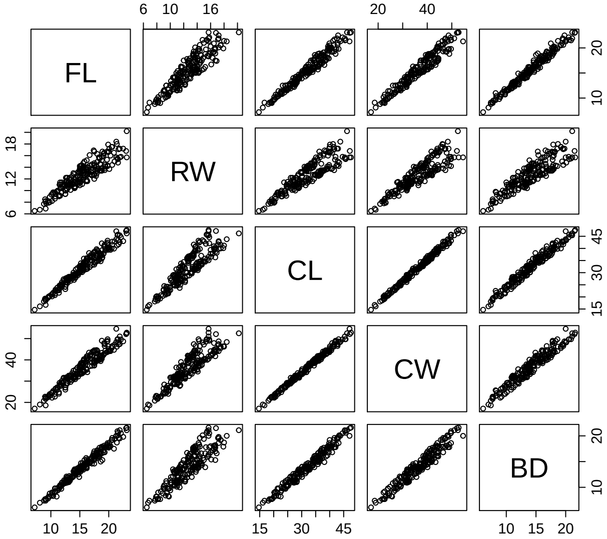
Univariate Histograms

```
par(mfrow=c(2,3))
hist(Crabs$FL,col='red',xlab='FL: Frontal Lobe Size (mm)')
hist(Crabs$RW,col='red',xlab='RW: Rear Width (mm)')
hist(Crabs$CL,col='red',xlab='CL: Carapace Length (mm)')
hist(Crabs$CW,col='red',xlab='CW: Carapace Width (mm)')
hist(Crabs$BD,col='red',xlab='BD: Body Depth (mm)')
```



Simple Pairwise Scatterplots

```
pairs(Crabs)
```



Visualisation and Dimensionality Reduction

The summary plots are useful, but limited use if the dimensionality p is high (a few dozens or even thousands).

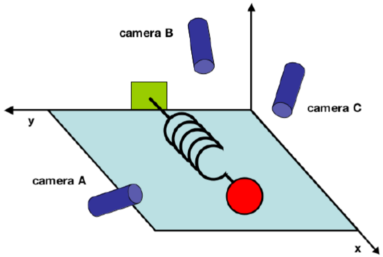
- Constrained to view data in 2 or 3 dimensions
- Approach: look for ‘interesting’ projections of $\mathbf{X}$ into lower dimensions
- Hope that even though p is large, considering only carefully selected $k \ll p$ dimensions is just as informative.

Dimensionality reduction

- For each data item $x_i \in \mathbb{R}^p$, find its lower dimensional representation $z_i \in \mathbb{R}^k$ with $k \ll p$.
- Map $x \mapsto z$ should preserve the **interesting statistical properties** in data.

Dimensionality reduction

- deceptively many variables to measure, many of them redundant / correlated to each other (large p)
- often, there is a simple but unknown underlying relationship hiding
- example: ball on a frictionless spring recorded by three different cameras
  - our imperfect measurements obfuscate the true underlying dynamics
  - are our coordinates meaningful or do they simply reflect the method of data gathering?



J. Shlens, A Tutorial on Principal Component Analysis, 2005

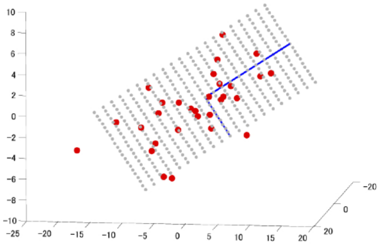
Principal Components Analysis (PCA)

- PCA considers interesting directions to be those with greatest **variance**.
- A **linear** dimensionality reduction technique: looks for a **new basis** to represent a noisy dataset.
- Workhorse for many different types of data analysis (often used for data preprocessing before supervised techniques are applied).
- Often the first thing to run on high-dimensional data.

Principal Components Analysis (PCA)

- Assume that the dataset is centred, i.e., $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = 0$.
- Sample covariance:

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\top$$
$$= \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}.$$



PCA

PCA recovers an orthonormal basis $v_1, v_2, \ldots, v_p$ in $\mathbb{R}^p$ – vectors v_i are **called principal components (PC) or loading vectors** – such that:

- The first principal component (PC) v_1 is the **direction of greatest variance** of data.
- The j-th PC v_j is the **direction orthogonal to** $v_1, v_2, \ldots, v_{j-1}$ **of greatest variance**, for $j = 2, \ldots, p$.

Principal Components Analysis (PCA)

- The k-dimensional representation of data item x_i is the vector of projections of x_i onto first k PCs:

$$z_i = V_{1:k}^\top x_i = \left[v_1^\top x_i, \dots, v_k^\top x_i\right]^\top \in \mathbb{R}^k,$$

where $V_{1:k} = [v_1, \dots, v_k]$.

- Transformed data matrix, also called the **scores matrix**

$$\mathbf{Z} = \mathbf{X} V_{1:k} \in \mathbb{R}^{n \times k}.$$

- Reconstruction of x_i:

$$\hat{x}_i = V_{1:k} V_{1:k}^\top x_i.$$

- PCA gives the **optimal linear reconstruction** of the original data based on a k-dimensional compression (problem sheets).

Deriving the First Principal Component

- Our data set is an i.i.d. sample $\{x_i\}_{i=1}^n$ of a random vector $X = \left[X^{(1)} \dots X^{(p)}\right]^\top$.
- For the 1^{st} PC, we seek a derived scalar variable of the form

$$Z^{(1)} = v_1^\top X = v_{11}X^{(1)} + v_{12}X^{(2)} + \cdots + v_{1p}X^{(p)}$$

  where $v_1 = [v_{11}, \dots, v_{1p}]^\top \in \mathbb{R}^p$ are chosen to maximise the sample variance

$$\widehat{\text{Var}}(Z^{(1)}) = v_1^\top \widehat{\text{Cov}}(X) v_1 = v_1^\top S v_1.$$

- Optimisation problem

$$\max_{v_1} \; v_1^\top S v_1$$

$$\text{subject to: } v_1^\top v_1 = 1.$$

Deriving the First Principal Component

- Lagrangian of the problem is given by:

$$\mathcal{L}\left(v_1, \lambda_1\right) = v_1^\top S v_1 - \lambda_1 \left(v_1^\top v_1 - 1\right).$$

- The corresponding vector of partial derivatives is

$$\frac{\partial \mathcal{L}(v_1, \lambda_1)}{\partial v_1} = 2Sv_1 - 2\lambda_1 v_1.$$

- Setting this to zero reveals the eigenvector equation $Sv_1 = \lambda_1 v_1$, i.e. v_1 must be an eigenvector of S and the dual variable λ_1 is the corresponding eigenvalue.
- Since $v_1^\top S v_1 = \lambda_1 v_1^\top v_1 = \lambda_1$, the first PC must be the eigenvector associated with the largest eigenvalue of S.

PCA as eigendecomposition of the covariance matrix

- The 2^{nd} PC is chosen to be orthogonal with the 1^{st} and is computed in a similar way (see notes). It will have the largest variance in the remaining $p-1$ dimensions, etc.
- The eigenvalue decomposition of S is given by

$$S = V\Lambda V^\top$$

where Λ is a diagonal matrix with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$$

and V is a $p \times p$ orthogonal matrix whose columns are the p eigenvectors of S, i.e. the principal components $v_1, \ldots, v_p$.

Properties of the Principal Components

- Derived scalar variable (projection to the j-th principal component) $Z^{(j)} = v_j^\top X$ has sample variance λ_j, for $j = 1, \ldots, p$
- S is a real symmetric matrix, so eigenvectors (principal components) are orthogonal.
- Projections to principal components are **uncorrelated**: $\widehat{\text{Cov}}(Z^{(i)}, Z^{(j)}) \approx v_i^\top S v_j = \lambda_j v_i^\top v_j = 0$, for $i \neq j$.
- The **total sample variance** is given by $\text{Tr}(S) = \sum_{i=1}^{p} S_{ii} = \lambda_1 + \ldots + \lambda_p$, so the **proportion of total variance explained** by the j^{th} PC is $\frac{\lambda_j}{\lambda_1+\lambda_2+\ldots+\lambda_p}$

R code

This is what we have had before:

```
> library(MASS)
> varnames<-c('FL','RW','CL','CW','BD')
> Crabs <- crabs[,varnames]
```

Now perform PCA with function `princomp`.
(Alternatively, solve for the PCs yourself using `eigen` or `svd`)

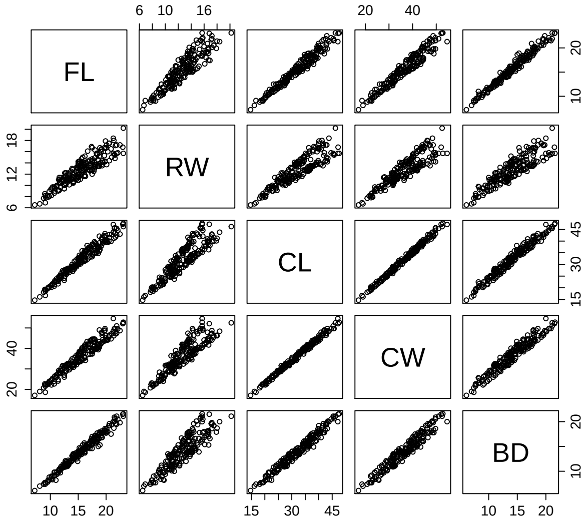
```
> Crabs.pca <- princomp(Crabs)
```

Exploring PCA output

```
> Crabs.pca <- princomp(Crabs)
> summary(Crabs.pca)

Importance of components:
                          Comp.1      Comp.2      Comp.3       Comp.4       Comp.5
Standard deviation     11.8322521 1.135936870 0.997631086 0.3669098284 0.2784325016
Proportion of Variance  0.9824718 0.009055108 0.006984337 0.0009447218 0.0005440328
Cumulative Proportion   0.9824718 0.991526908 0.998511245 0.9994559672 1.0000000000

> loadings(Crabs.pca)

Loadings:
   Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
FL -0.289 -0.323  0.507  0.734  0.125
RW -0.197 -0.865 -0.414 -0.148 -0.141
CL -0.599  0.198  0.175 -0.144 -0.742
CW -0.662  0.288 -0.491  0.126  0.471
BD -0.284 -0.160  0.547 -0.634  0.439
```

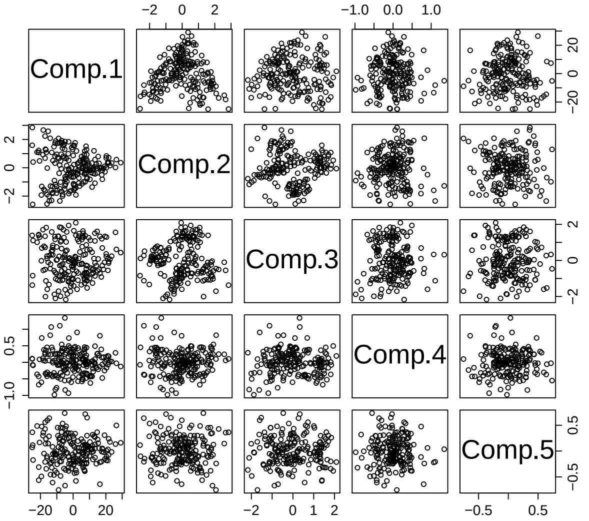
Raw Crabs Data

```
> pairs(Crabs)
```



PCA of Crabs Data

```
> Crabs.pca <- princomp(Crabs)
> pairs(predict(Crabs.pca))
```



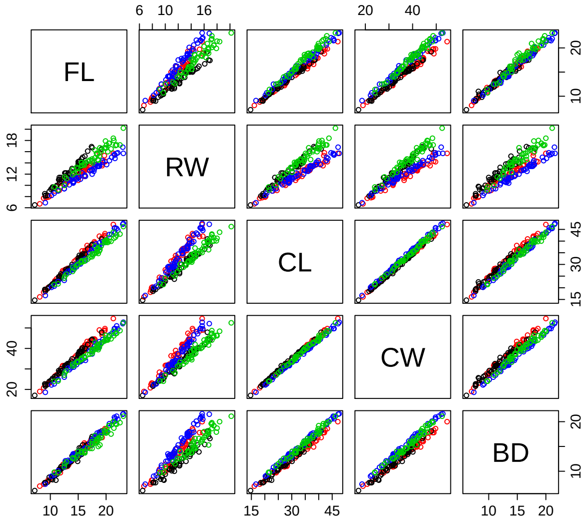
What did we discover?

Now let us use our label information (species+sex).

```
> Crabs.class <- factor(paste(crabs$sp,crabs$sex,sep=""))
> Crabs.class
  [1] BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM
 [27] BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BM BF BF
 [53] BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF
 [79] BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF BF OM OM OM OM
[105] OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM
[131] OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OM OF OF OF OF OF OF
[157] OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF
[183] OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF OF
Levels: BF BM OF OM
```

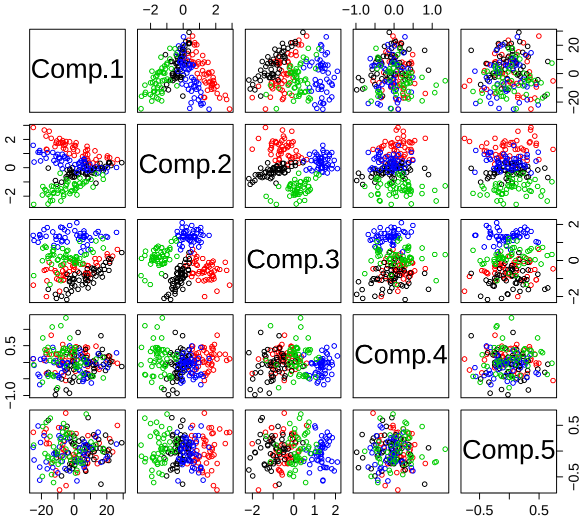
Raw Crabs Data - with labels

```
> pairs(Crabs,col=unclass(Crabs.class))
```



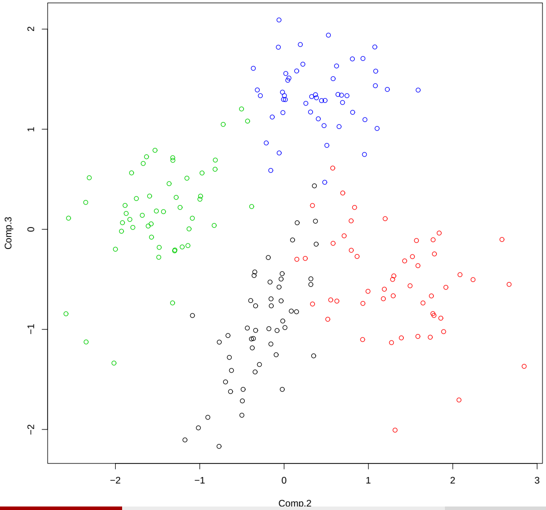
PCA of Crabs Data - with labels

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

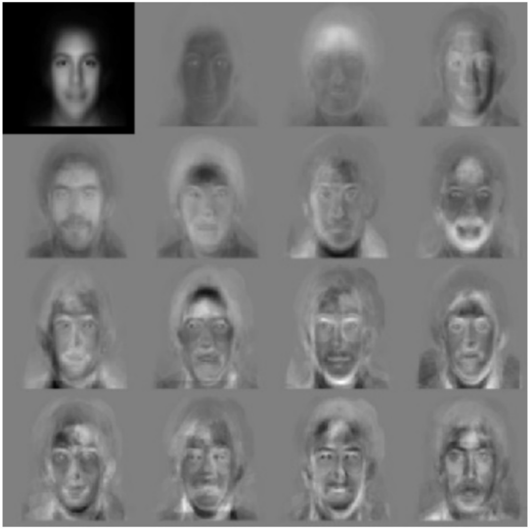


PC 2 vs PC 3

```
> Z<-predict(Crabs.pca)
> plot(Comp.3~Comp.2,data=Z,col=unclass(Crabs.class))
```

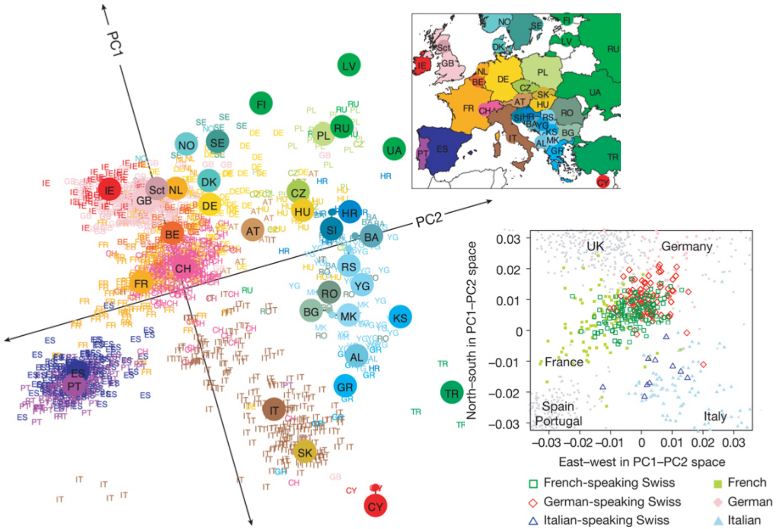


PCA on Face Images: Eigenfaces



Turk and Pentland, CVPR 1995

PCA on European Genetic Variation



Genes mirror geography within Europe, Nature 2008

Comments on the use of PCA

- PCA commonly used to project data X onto the first k PCs giving the k-dimensional view of the data that best preserves **the first two moments**.
- Although PCs are uncorrelated, scatterplots sometimes reveal structures in the data other than linear correlation.
- Emphasis on variance is where the weaknesses of PCA stem from:
  - Assuming large variances are meaningful (high signal-to-noise ratio)
  - 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. In these cases, it is recommended to calculate PCs from $\mathrm{Corr}(X)$ instead of $\mathrm{Cov}(X)$ (`cor=True` in the call of `princomp`).
  - Lack of robustness to outliers: variance is affected by outliers and so are PCs.

PCA: summary

PCA

Find an orthogonal basis $\{v_1, v_2, \dots, v_p\}$ for the data space such that:

- The first principal component (PC) v_1 is the **direction of greatest variance** of data.
- The j-th PC v_j is the **direction orthogonal to** $v_1, v_2, \dots, v_{j-1}$ **of greatest variance**, for $j = 2, \dots, p$.

- Eigendecomposition of the sample covariance matrix $S = \frac{1}{n-1} \sum_{i=1}^n x_i x_i^\top$.

$$S = V \Lambda V^\top.$$

  - Λ is a diagonal matrix with eigenvalues (variances along each principal component) $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$
  - V is a $p \times p$ orthogonal matrix whose columns are the p eigenvectors of S, i.e. the principal components $v_1, \dots, v_p$
- Dimensionality reduction by projecting $x_i \in \mathbb{R}^p$ onto first k principal components:

$$z_i = \left[v_1^\top x_i, \dots, v_k^\top x_i \right]^\top \in \mathbb{R}^k.$$

Eigendecomposition and PCA

$$S = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}.$$

- S is a **real and symmetric** matrix, so there exist 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 $Sv_i = \lambda_i v_i$. In particular, V is an orthogonal matrix:
$$VV^\top = V^\top V = I_p.$$
- S is a **positive-semidefinite** matrix, so the eigenvalues are non-negative:
$$\lambda_i \geq 0, \ \forall i.$$

Why is S symmetric? Why is S positive-semidefinite?
Reminder: A symmetric $p \times p$ matrix R is said to be positive-semidefinite if

$$\forall a \in \mathbb{R}^p, a^\top R a \geq 0.$$

Singular Value Decomposition (SVD)

SVD

Any real-valued $n \times p$ matrix $\mathbf{X}$ can be written as $\mathbf{X} = UDV^\top$ where

- U is an $n \times n$ orthogonal matrix: $UU^\top = U^\top U = I_n$
- 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 $p \times p$ orthogonal matrix: $VV^\top = V^\top V = I_p$

- SVD **always** exists, even for non-square matrices.
- Fast and numerically stable algorithms for SVD are available in most packages.The relevant R command is `svd`.

SVD and PCA

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

$$(n-1)S = \mathbf{X}^\top \mathbf{X} = (UDV^\top)^\top (UDV^\top) = VD^\top U^\top UDV^\top = VD^\top DV^\top,$$

  using orthogonality ($U^\top U = I_n$) of U.
- The eigenvalues of S are thus the diagonal entries of $\Lambda = \frac{1}{n-1} D^\top D$.
- We also have

$$\mathbf{X}\mathbf{X}^\top = (UDV^\top)(UDV^\top)^\top = UDV^\top VD^\top U^\top = UDD^\top U^\top,$$

  using orthogonality ($V^\top V = I_p$) of V.

Gram matrix

$\mathbf{B} = \mathbf{X}\mathbf{X}^\top$, $\mathbf{B}_{ij} = x_i^\top x_j$ is called the Gram matrix of dataset $\mathbf{X}$.
$\mathbf{B}$ and $(n-1)S = \mathbf{X}^\top \mathbf{X}$ have the same nonzero eigenvalues, equal to the non-zero squared singular values of $\mathbf{X}$.

PCA projections from Gram matrix

If we consider projections to **all principal components**, the transformed data matrix is

$$\mathbf{Z} = \mathbf{X}V = UDV^\top V = UD, \tag{1}$$

If $p \leq n$ this means

$$z_i = \left[U_{i1}D_{11}, \ldots, U_{ip}D_{pp}\right]^\top, \tag{2}$$

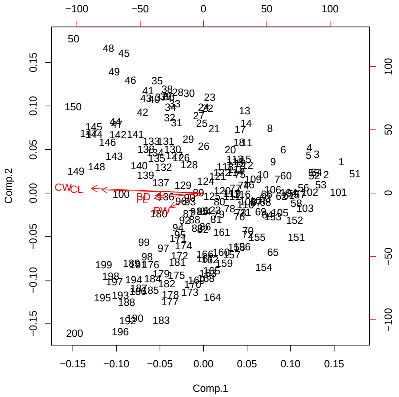
and if $p > n$ only the first n projections are defined (sample covariance will have rank at most n):

$$z_i = \left[U_{i1}D_{11}, \ldots, U_{in}D_{nn}, 0, \ldots, 0\right]^\top. \tag{3}$$

Thus, $\mathbf{Z}$ can be obtained from the eigendecomposition of Gram matrix $\mathbf{B}$. When $p \gg n$, eigendecomposition of $\mathbf{B}$ requires much less computation, $O(n^3)$, than the eigendecomposition of the covariance matrix, $O(p^3)$, so is the preferred method for PCA in that case.

Biplots

```
> biplot(Crabs.pca,scale=1)
```



- PCA plots show the data items (rows of $\mathbf{X}$) in the space spanned by PCs.
- **Biplots** allow us to visualize the **original variables** $X^{(1)}, \ldots, X^{(p)}$ (corresponding to columns of $\mathbf{X}$) in the same plot.

Biplots

Recall that $X = [X^{(1)}, \ldots, X^{(p)}]^\top$ and $\mathbf{X} = UDV^\top$ is the SVD of the data matrix.

- The "full" PC projection of x_i is the i-th row of UD (assuming $p \leq n$):

$$z_i = V^\top x_i = [U_{i1}D_{11}, \ldots, U_{ip}D_{pp}]^\top, \text{equivalently: } \mathbf{X}V = UD.$$

- The j-th unit vector $\mathbf{e}_j \in \mathbb{R}^p$ points in the direction of the original variable $X^{(j)}$. Its PC projection ν_j is:

$$\nu_j = V^\top \mathbf{e}_j = [V_{j1}, \ldots, V_{jp}]^\top \qquad \text{(the } j\text{-th row of } V\text{)}$$

- The projection of $\mathbf{e}_j$ indicates the weighting each PC gives to the original variable $X^{(j)}$.
- Dot products between these projections give entries of the data matrix:

$$x_{ij} = \sum_{\ell=1}^{\min\{n,p\}} U_{i\ell}D_{\ell\ell}V_{j\ell} = z_i^\top \nu_j.$$

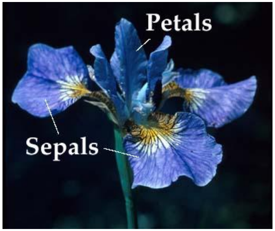
- Biplots focus on the first two PCs and the quality depends on the **proportion of variance explained** by the first two PCs.

Iris Data

50 samples from each of the 3 species of 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)

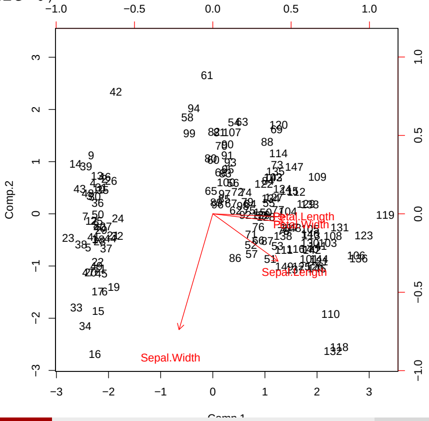


Iris Data

```
> data(iris)
> iris[sample(150,20),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
54           5.5         2.3          4.0         1.3 versicolor
33           5.2         4.1          1.5         0.1     setosa
30           4.7         3.2          1.6         0.2     setosa
73           6.3         2.5          4.9         1.5 versicolor
107          4.9         2.5          4.5         1.7  virginica
4            4.6         3.1          1.5         0.2     setosa
90           5.5         2.5          4.0         1.3 versicolor
83           5.8         2.7          3.9         1.2 versicolor
50           5.0         3.3          1.4         0.2     setosa
92           6.1         3.0          4.6         1.4 versicolor
128          6.1         3.0          4.9         1.8  virginica
57           6.3         3.3          4.7         1.6 versicolor
9            4.4         2.9          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
86           6.0         3.4          4.5         1.6 versicolor
66           6.7         3.1          4.4         1.4 versicolor
85           5.4         3.0          4.5         1.5 versicolor
147          6.3         2.5          5.0         1.9  virginica
8            5.0         3.4          1.5         0.2     setosa
41           5.0         3.5          1.3         0.3     setosa
```

Iris data biplot

```
 > iris.pca<-princomp(iris[,-5],cor=TRUE)
 > loadings(iris.pca)
               Comp.1 Comp.2 Comp.3 Comp.4
Sepal.Length  0.521 -0.377  0.720  0.261
Sepal.Width  -0.269 -0.923 -0.244 -0.124
Petal.Length  0.580        -0.142 -0.801
Petal.Width   0.565        -0.634  0.524
 > biplot(iris.pca,scale=0)
```



Biplots

- There are other projections we can consider for biplots (assuming $p \le n$):

$$x_{ij} = \sum_{\ell=1}^{p} U_{i\ell} D_{\ell\ell} V_{j\ell} = \tilde{z}_i^\top \tilde{\nu}_j$$

  where for some $\alpha \in [0,1]$:

$$\tilde{z}_i = \left[U_{i1} D_{11}^{1-\alpha}, \dots, U_{ip} D_{pp}^{1-\alpha}\right]^\top \qquad \tilde{\nu}_j = \left[D_{11}^{\alpha} V_{j1}, \dots, D_{pp}^{\alpha} V_{jp}\right]^\top$$

- In the case $\alpha = 1$, i.e. $\tilde{\mathbf{Z}} = U_{1:n,1:p}$
  - Sample covariance of the projected points is:

$$\widehat{\text{Cov}}\left(\tilde{Z}\right) = \frac{1}{n-1} U_{1:n,1:p}^\top U_{1:n,1:p} = \frac{1}{n-1} I_p.$$

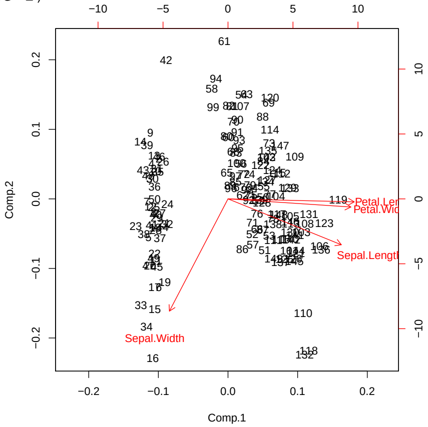
    Derived variables $\tilde{Z}^{(1)}, \dots, \tilde{Z}^{(k)}$ are uncorrelated and have equal variance.
  - Sample covariance between $X^{(i)}$ and $X^{(j)}$ is:

$$\widehat{\text{Cov}}(X^{(i)}, X^{(j)}) = \frac{1}{n-1} \left(VD^\top DV^\top\right)_{i,j} = \frac{1}{n-1} \tilde{\nu}_i^\top \tilde{\nu}_j$$

    The angle between the projected variables can be interpreted as the correlation between the original variables.

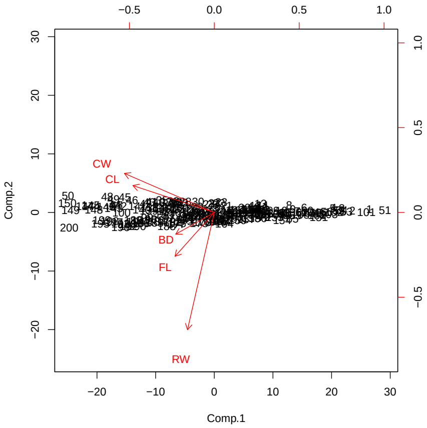
Iris Data biplot - scaled

```
 > ?biplot
...
scale: The variables are scaled by lambda ^ scale and the observations
are scaled by lambda ^ (1-scale) where lambda are the singular values
as computed by princomp. (default=1)
...
> biplot(iris.pca,scale=1)
```

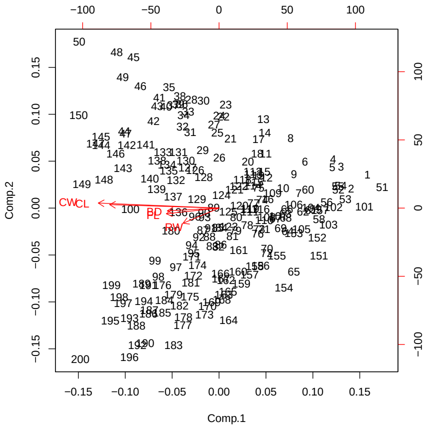


Crabs Data biplots

```
> biplot(Crabs.pca,scale=0)
```



```
> biplot(Crabs.pca,scale=1)
```



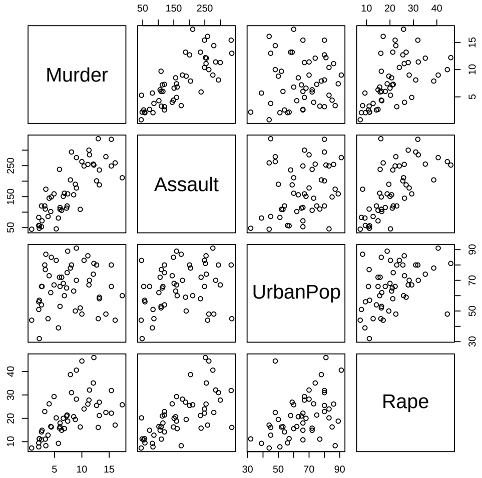
US Arrests Data

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)
```

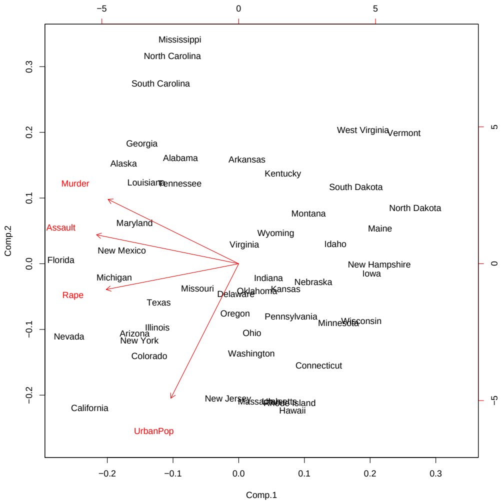
US Arrests Data Pairs Plot

```
> pairs(USArrests)
```



US Arrests Data Biplot

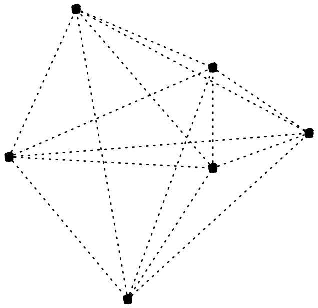
```
> biplot(usarrests.pca)
```



Multidimensional Scaling

Suppose there are n points $\mathbf{X}$ in $\mathbb{R}^p$, but we are only given the $n \times n$ matrix $\mathbf{D}$ of inter-point dissimilarities.

Can we reconstruct $\mathbf{X}$?



Multidimensional Scaling

Rigid transformations (translations, rotations and reflections) do not change inter-point distances so cannot recover $\mathbf{X}$ exactly. However $\mathbf{X}$ can be recovered up to these transformations!

- Let $\mathbf{D}_{ij} = \|x_i - x_j\|_2^2$ be the squared Euclidean distance between points x_i and x_j.

$$\begin{aligned} \mathbf{D}_{ij} &= \|x_i - x_j\|_2^2 \\ &= (x_i - x_j)^\top (x_i - x_j) \\ &= x_i^\top x_i + x_j^\top x_j - 2x_i^\top x_j \end{aligned}$$

- Let $\mathbf{B} = \mathbf{X}\mathbf{X}^\top$ be the $n \times n$ matrix of dot-products, $\mathbf{B}_{ij} = x_i^\top x_j$. The above shows that $\mathbf{D}$ can be computed from $\mathbf{B}$.
- Conversely, $\mathbf{B}$ can be recovered from $\mathbf{D}$ if we assume $\sum_{i=1}^n x_i = 0$ (exercise).

Multidimensional Scaling

- If we knew $\mathbf{X}$, then SVD gives $\mathbf{X} = UDV^\top$. As $\mathbf{X}$ has rank at most $r = \min(n, p)$, we have at most r non-zero singular values in D and we can assume $U \in \mathbb{R}^{n\times r}$, $D \in \mathbb{R}^{r\times r}$ and $V^\top \in \mathbb{R}^{r\times p}$.
- The eigendecomposition of $\mathbf{B}$ is then:

$$\mathbf{B} = \mathbf{X}\mathbf{X}^\top = UD^2U^\top = U\Lambda U^\top.$$

- This eigendecomposition can be obtained from $\mathbf{B}$ without knowledge of $\mathbf{X}$!
- Let $\tilde{x}_i^\top = U_i\Lambda^{\frac{1}{2}} \in \mathbb{R}^r$. If $r < p$, pad $\tilde{x}_i$ with 0s so that it has length p. Then,

$$\tilde{x}_i^\top \tilde{x}_j = U_i \Lambda U_j^\top = \mathbf{B}_{ij} = x_i^\top x_j$$

  and we have found a set of vectors with dot-products given by $\mathbf{B}$ (and hence distances given by $\mathbf{D}$), as desired.
- The vectors $\tilde{x}_i$ differs from x_i only via the orthogonal matrix $V^\top$ (recall that $x_i^\top = U_i D V^\top = \tilde{x}_i^\top V^\top$) so are equivalent up to rotation and reflections.

US City Flight Distances

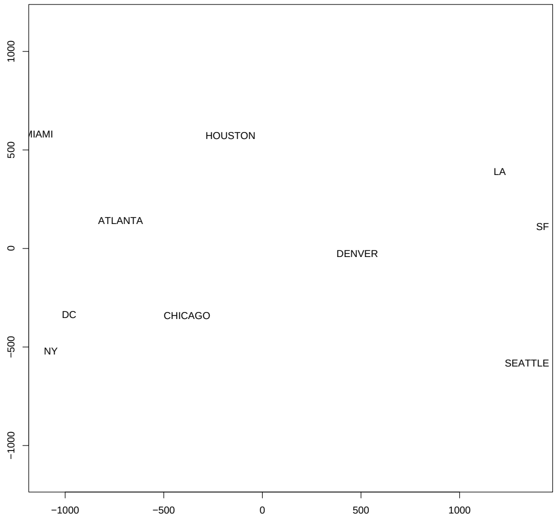
We present a table of flying mileages between 10 American cities, distances calculated from our 2-dimensional world. Using $\mathbf{D}$ as the starting point, metric MDS finds a configuration with the same distance matrix.

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

US City Flight Distances

```
library(MASS)

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

## use classical MDS 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))
```

US City Flight Distances



Lower-dimensional Reconstructions

In classical MDS derivation, we used all eigenvalues in the eigendecomposition of $\mathbf{B}$ to reconstruct

$$\tilde{x}_i = U_i \Lambda^{\frac{1}{2}}.$$

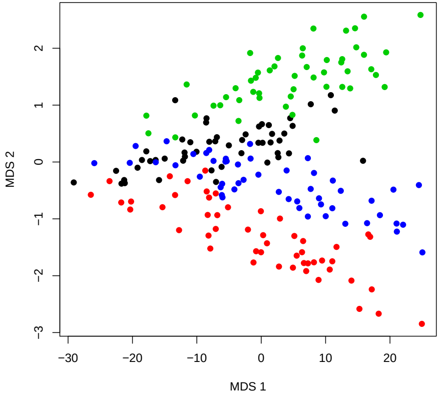
We can use only the largest $k < \min(n, p)$ eigenvalues and eigenvectors in the reconstruction, giving the ‘best’ k-dimensional view of the data.

This is analogous to PCA, where only the largest eigenvalues of $\mathbf{X}^\top \mathbf{X}$ are used, and the smallest ones effectively suppressed.

Indeed, PCA and classical MDS are duals and yield effectively the same result.

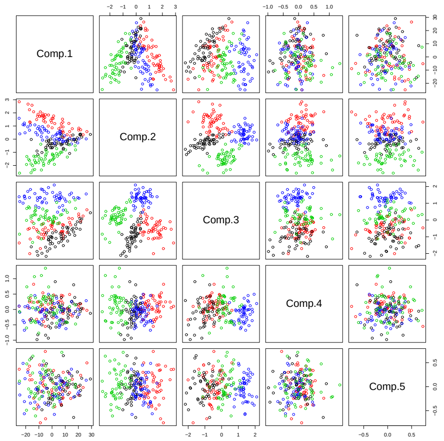
Crabs Data

```
library(MASS)
crabs$spsex=paste(crabs$sp,crabs$sex,sep="")
varnames<-c('FL','RW','CL','CW','BD')
Crabs <- crabs[,varnames]
Crabs.class <- factor(crabs$spsex)
crabsmds <- cmdscale(d= dist(Crabs),k=2)
plot(crabsmds, pch=20, cex=2,col=unclass(Crabs.class))
```



Crabs Data

Compare with previous PCA analysis.
Classical MDS solution corresponds to the first 2 PCs.



Varieties of MDS

Generally, MDS is a class of dimensionality reduction techniques which represents data points $x_1, \dots, x_n \in \mathbb{R}^p$ in a lower-dimensional space $z_1, \dots, z_n \in \mathbb{R}^k$ which tries to preserve inter-point (dis)similarities.

- It requires only the matrix $\mathbf{D}$ of pairwise dissimilarities $\mathbf{D}_{ij} = \rho(x_i, x_j)$ (not necessarily Euclidean distances).
- MDS finds representations $z_1, \dots, z_n \in \mathbb{R}^k$ such that

$$\|z_i - z_j\|_2 \approx \rho(x_i, x_j) = \mathbf{D}_{ij},$$

  and differences in dissimilarities are measured by the appropriate loss $\Delta(\mathbf{D}_{ij}, \|z_i - z_j\|_2)$.
- Goal: Find $\mathbf{Z}$ which minimizes the **stress function**

$$S(\mathbf{Z}) = \sum_{i \neq j} \Delta(\mathbf{D}_{ij}, \|z_i - z_j\|_2).$$

Varieties of MDS

- Choices of (dis)similarities and (**stress**) functions lead to different algorithms.
  - **Classical/Torgerson**: preserves inner products instead - **strain function** (`cmdscale`)

$$S(\mathbf{Z}) = \sum_{i \neq j} (\mathbf{B}_{ij} - \langle z_i - \bar{z}, z_j - \bar{z} \rangle)^2$$

  - **Metric Shephard-Kruskal**: preserves distances w.r.t. squared stress

$$S(\mathbf{Z}) = \sum_{i \neq j} (\mathbf{D}_{ij} - \|z_i - z_j\|_2)^2$$

  - **Sammon**: preserves shorter distances more (`sammon`)

$$S(\mathbf{Z}) = \sum_{i \neq j} \frac{(\mathbf{D}_{ij} - \|z_i - z_j\|_2)^2}{\mathbf{D}_{ij}}$$

  - **Non-Metric Shephard-Kruskal**: ignores actual distance values, only preserves ranks (`isoMDS`)

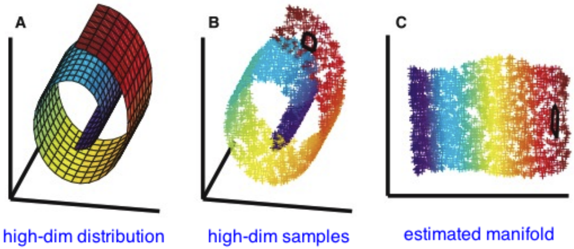
$$S(\mathbf{Z}) = \min_{g \text{ increasing}} \frac{\sum_{i \neq j} (g(\mathbf{D}_{ij}) - \|z_i - z_j\|_2)^2}{\sum_{i \neq j} \|z_i - z_j\|_2^2}$$

Nonlinear Dimensionality Reduction

Two aims of different varieties of MDS:

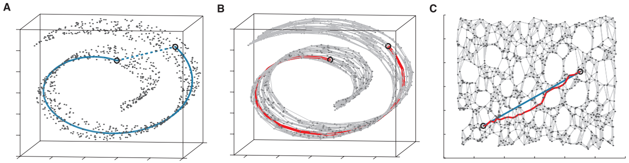
- To visualize the (dis)similarities among items in a dataset, where these (dis)disimilarities may not have Euclidean geometric interpretations.
- To perform **nonlinear** dimensionality reduction.

Many high-dimensional datasets exhibit low-dimensional structure ("live on a low-dimensional menifold").



Isomap

Isomap is a non-linear dimensional reduction technique based on classical MDS. Differs from other MDSs as it uses estimates of **geodesic distances** between the data points.

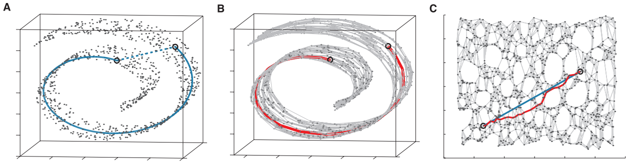


Tenenbaum et al. (2000)

Isomap

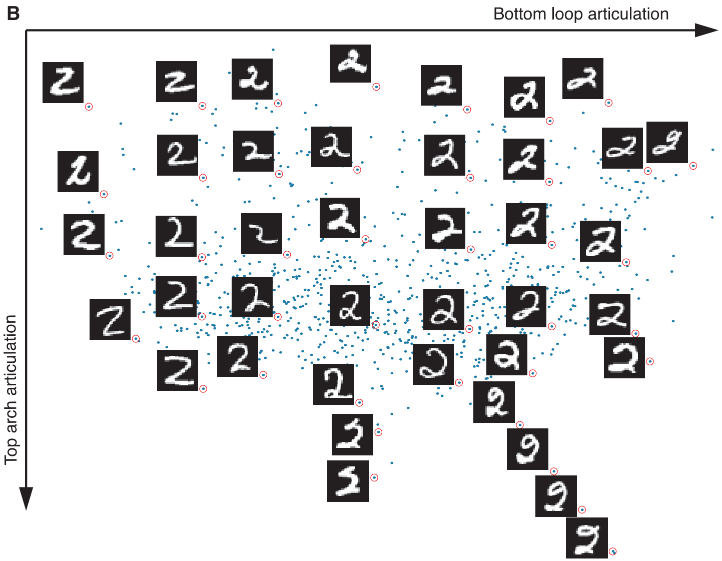
Isomap

- Calculate Euclidean distances $\mathbf{D}_{ij}$ for $i, j = 1, \ldots, n$ between all data points.
- Form a graph G with n samples as nodes, and edges between the respective K nearest neighbours (K-Isomap) or between i and j if $\mathbf{D}_{ij} < \epsilon$ (ϵ-Isomap).
- For i, j linked by an edge, set $\mathbf{D}^G_{ij} = \mathbf{D}_{ij}$. Otherwise, set $\mathbf{D}^G_{ij}$ to be the shortest-path distance between i and j in G.
- Run classical MDS using distances $\mathbf{D}^G_{ij}$.

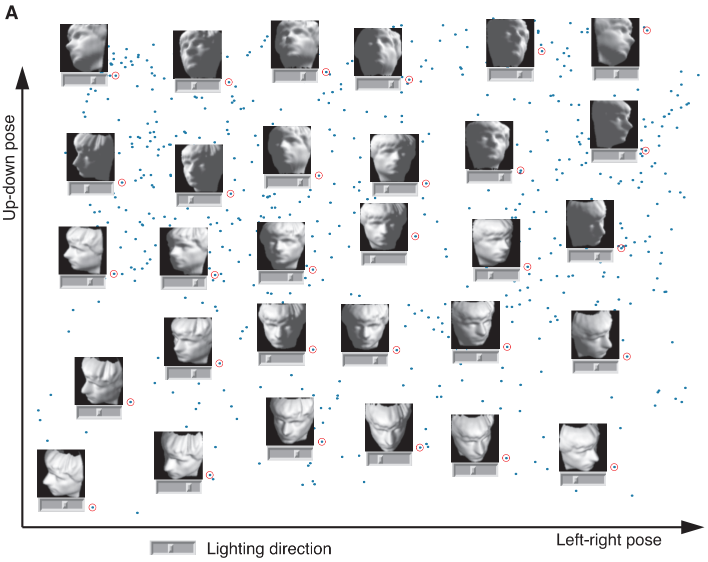


R function: `isomap{vegan}`.

Handwritten Characters



Faces



Summary

- Other dimensionality reduction techniques:
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
  - Kernel PCA
  - Locally Linear Embedding
  - Laplacian Eigenmaps
  - Maximum Variance Unfolding
- Further reading:
  - Bishop 12.1
  - Hastie et al 14.8-14.9
  - Manifold learning on scikit-learn