

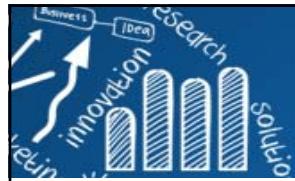
# 維度縮減

## Dimension Reduction

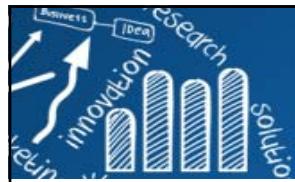
吳漢銘

國立臺北大學 統計學系





- Why Reduce Dimensionality?
- Feature Selection vs Extraction
- Dimension Reduction Techniques
  - 1) Spectral Decomposition (Eigen) 譜(特徵值)分解
  - 2) Singular Value Decomposition (SVD) 奇異值分解
  - 3) Principal Component Analysis (PCA) 主成份分析
  - 4) Biplot 雙標圖
  - 5) Factor Analysis (FA) 因子分析
  - 6) Multidimensional Scaling (MDS) 多元尺度法
  - 7) Isometric Feature Mapping (ISOMAP) 等軸距特徵映射
  - 8) Linear Discriminant Analysis (LDA) 線性區別分析
  - 9) Sliced Inverse Regression (SIR) 切片逆迴歸法
  - 10) Canonical Correlation Analysis (CCA) 典型相關分析
- The high-dimension low sample size data (HDLSS)
- DR Quality Assessment



# Dimension Reduction

keep information as much as possible without loss of information.

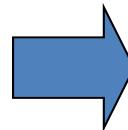
input data matrix:

X

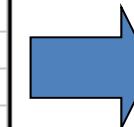
Group	Data	X1	X2	X3	...	Xp
1	subject01	0.81	-1.29	-0.50		1.13
1	subject02	0.64	2.16	-1.51		0.00
2	subject03	0.13	0.60	1.10		0.11
2	subject04	-0.17	0.31	-0.37		-0.50
3	subject05	-1.01	0.99	0.70		-0.08
3	subject06	0.95	0.75	-0.83		0.60
3	subject07	0.72	1.12	-1.35		-1.22
1	subject08	0.77	1.24	-0.04		1.03
1	subject09	-0.49	0.02	-1.73		1.61
1	subject10	1.93	0.45	-0.01		0.03
3	subject11	-0.15	-1.36	1.05		0.50
3	subject12	-1.16	0.11	-0.57		-0.80
3	subject13	-0.02	2.05	-1.18		0.45
3	subject14	-0.05	0.79	1.33		0.81
2	subject15	-0.21	-0.38	0.72		-0.61
1	subject16	-0.28	0.57	1.02		-0.01
⋮	⋮					
2	subjectN	0.33	0.01	1.19		-0.33

transformed data  
matrix: Z

PCA  
FA  
MDS



Data	Z1	Z2	...	Zk
subject01	-1.55	-0.66		0.60
subject02	0.57	-0.51		-1.03
subject03	1.99	1.44		-0.60
subject04	0.10	0.20		-1.21
subject05	-0.20	-0.64		0.24
subject06	2.85	1.32		-0.61
subject07	-0.34	-0.35		0.15
subject08	0.66	0.44		0.28
subject09	8.44	1.66		2.12
subject10	0.37	-0.17		-1.73
subject11	-1.14	0.01		0.61
subject12	-1.73	-1.13		0.81
subject13	1.53	0.67		0.48
subject14	-0.21	-0.14		-0.29
subject15	-0.03	0.66		0.17
subject16	2.56	-2.25		0.26
⋮	⋮			
subjectN	2.04	0.71		0.76

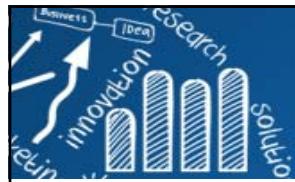


Visualization  
Clustering  
Classification  
....

Y

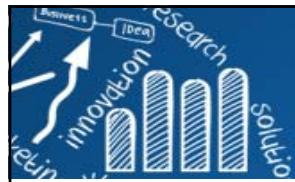


Methods using additional information y: LDA, Sufficient Dimension Reduction (SIR, SAVE, pHd, IRE,...)



# Why Reduce Dimensionality?

- **Reduces time complexity:**  
less computation
- **Reduces space complexity:**  
Less parameters
- **Saves the cost of observing the features:**  
input is unnecessary.
- **Simpler models are more robust on small datasets:**  
simpler models vary less depending on the particulars of a sample.
- **More interpretable; simpler explanation**
- **Data visualization** (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions. (Dimension reduction visualization is often adopted for presenting grouping structure for methods such as K-means.)



# DR: Feature Selection vs Extraction

## Feature Selection

- Choosing  $k < d$  important features
- Ignoring the remaining  $d - k$
- Subset selection algorithms

$$z = w^T x \quad \begin{matrix} & \downarrow & \downarrow & \downarrow & \downarrow \\ a1 & a2 & a3 & \dots & ap \end{matrix}$$

## Feature Extraction

- **Project** the original  $x_i, i = 1, \dots, d$  dimensions to new  $k < d$  dimensions,  
 $z_j, j = 1, \dots, k$ .
- $z = w^T x$
- PCA, FA, MDS, LDA, SIR

Data	X1	X2	X3	...	Xp
subject01	0.81	-1.29	-0.50		1.13
subject02	0.64	2.16	-1.51		0.00
subject03	0.13	0.60	1.10		0.11
subject04	-0.17	0.31	-0.37		-0.50
subject05	-1.01	0.99	0.70		-0.08
subject06	0.95	0.75	-0.83		0.60
subject07	0.72	1.12	-1.35		-1.22
subject08	0.77	1.24	-0.04		1.03
subject09	-0.49	0.02	-1.73		1.61
subject10	1.93	0.45	-0.01		0.03
subject11	-0.15	-1.36	1.05		0.50
subject12	-1.16	0.11	-0.57		-0.80
subject13	-0.02	2.05	-1.18		0.45
subject14	-0.05	0.79	1.33		0.81
subject15	-0.21	-0.38	0.72		-0.61
subject16	-0.28	0.57	1.02		-0.01
⋮					
subjectN	0.33	0.01	1.19		-0.33



# Subset Selection

- Finding the **best subset** of the set of features.
- **Best**: contains the **least** number of dimensions that **most** contribute to accuracy.

## Forward Search: Add the best feature at each step

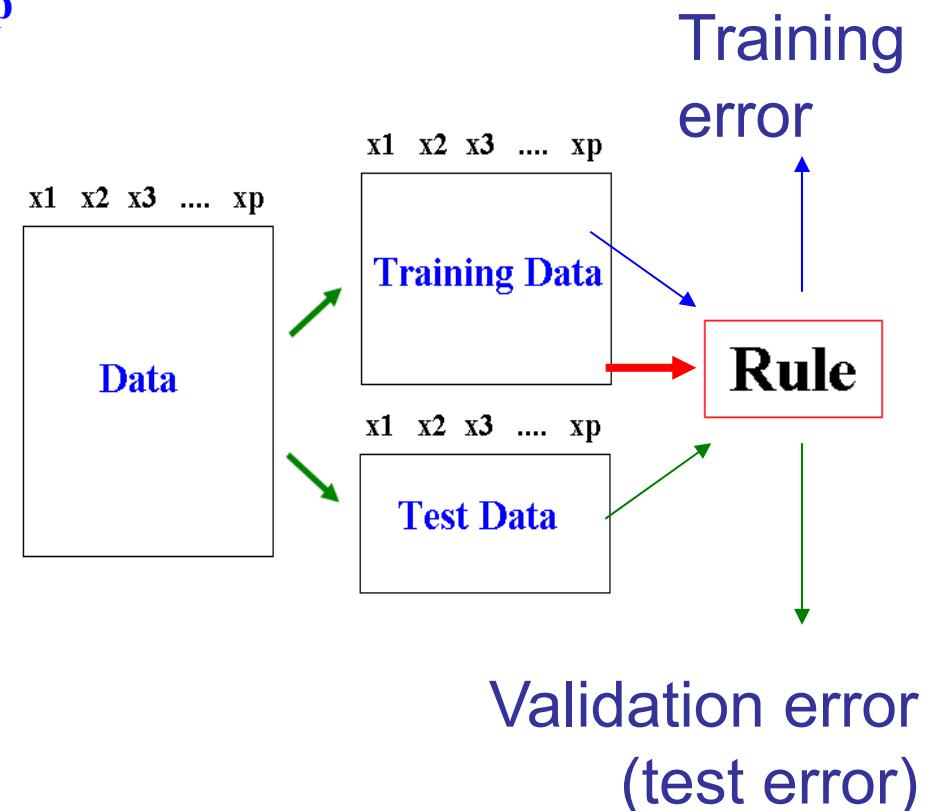
1. Set of features  $F$  initially  $\emptyset$ .
2. At each iteration, find the best new feature
3.  $j = \operatorname{argmin}_i E(F \cup x_i)$
4. Add  $x_i$  to  $F$  if  $E(F \cup x_i) < E(F)$
5. Stop:  $|E^t - E^{t+1}| < \text{epsilon}$

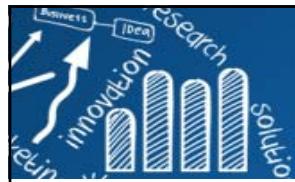
## Backward Search:

### Start with all features and remove one at a time

1. Set of features  $F$  initially **ALL**.
2. At each iteration, find the best new feature
3.  $j = \operatorname{argmin}_i E(F - x_i)$
4. remove  $x_i$  from  $F$  if  $E(F - x_i) < E(F)$
5. Stop:  $|E^t - E^{t+1}| < \text{epsilon}$

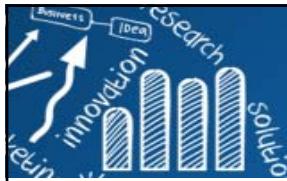
Others: Floating search (Add  $k$ , Remove  $l$ ), . . .





## (1) Spectral Decomposition (1/2)

- **Definition:** Let  $A$  be a  $m \times m$  real symmetric matrix. Then there exists an orthogonal matrix  $\mathbf{C}$  such that  $\mathbf{C}^T A \mathbf{C} = \mathbf{D}$  or  $A = \mathbf{C} \mathbf{D} \mathbf{C}^T$ , where  $\mathbf{D}$  is a diagonal matrix of eigenvalues, and the  $\mathbf{C}$  matrix has the eigenvectors.  
(A. L. Cauchy established the Spectral Decomposition in 1829.)
- **Eigen-decomposition :**  $A c_i = \lambda_i c_i, i=1,\dots,m.$
- Let  $\mathbf{Z} = \mathbf{W}^T \mathbf{X}$ , or  $\mathbf{Z} = \mathbf{W}^T (\mathbf{X} - \mathbf{m})$  center the data on the origin. To find a matrix  $\mathbf{W}$  such that when we have  $\mathbf{Z} = \mathbf{W}^T \mathbf{X}$ , we will get  $\text{Cov}(\mathbf{Z}) = \mathbf{D}$  is any diagonal matrix.



## Spectral Decomposition (2/2)

- We would like to get uncorrelated  $\mathbf{z}_i$
- Let  $\mathbf{C} = [\mathbf{c}_i]$  be the normalized eigenvector of  $\mathbf{S}$ , then

$$(1) \quad \mathbf{C}^T \mathbf{C} = \mathbf{I} \text{ and}$$

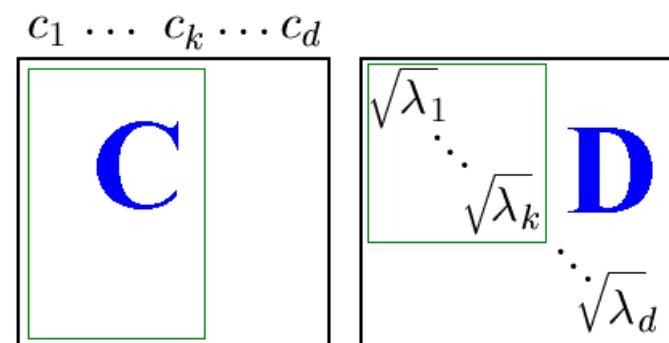
$$\begin{aligned} (2) \quad \mathbf{S} &= \mathbf{S} \mathbf{C} \mathbf{C}^T \\ &= \mathbf{S}(c_1, c_2, \dots, c_d) \mathbf{C}^T \\ &= (Sc_1, Sc_2, \dots, Sc_d) \mathbf{C}^T \\ &= (\lambda_1 c_1, \lambda_2 c_2, \dots, \lambda_d c_d) \mathbf{C}^T \\ &= \lambda_1 c_1 c_1^T + \lambda_2 c_2 c_2^T + \dots + \lambda_d c_d c_d^T \\ &= \mathbf{C} \mathbf{D} \mathbf{C}^T \end{aligned}$$

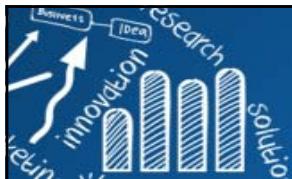
$$\begin{array}{ccc} \mathbf{S} & = & (\mathbf{C} \mathbf{D}^{1/2})(\mathbf{C} \mathbf{D}^{1/2})^T \\ & \downarrow & \downarrow \quad \downarrow \\ [d \times d] & [d \times d] & [d \times d] \end{array}$$

$$\begin{array}{ccc} \widehat{\mathbf{S}} & = & (\mathbf{C} \mathbf{D}^{1/2})(\mathbf{C} \mathbf{D}^{1/2})^T \\ & \downarrow & \downarrow \quad \downarrow \\ [d \times d] & [d \times k] & [k \times k] \end{array}$$

### Application:

- Data reduction.
- Image processing and compression.
- K-selection for K-means clustering
- Multivariate outliers detection
- Noise filtering
- Trend detection in the observations.





# Spectral Decomposition in R

```

> # Spectral Decomposition
> X <- iris[,1:4]
> (S <- cov(X))
      Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length   0.6856935 -0.0424340   1.2743154   0.5162707
Sepal.Width    -0.0424340   0.1899794  -0.3296564  -0.1216394
Petal.Length   1.2743154 -0.3296564   3.1162779   1.2956094
Petal.Width    0.5162707 -0.1216394   1.2956094   0.5810063

> (e <- eigen(S))
eigen() decomposition
$values
[1] 4.22824171 0.24267075 0.07820950 0.02383509

$vectors
      [,1]          [,2]          [,3]          [,4]
[1,]  0.36138659  0.65658877 -0.58202985  0.3154872
[2,] -0.08452251  0.73016143  0.59791083 -0.3197231
[3,]  0.85667061 -0.17337266  0.07623608 -0.4798390
[4,]  0.35828920 -0.07548102  0.54583143  0.7536574

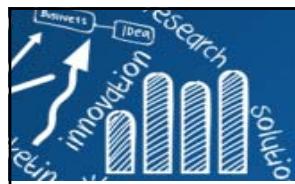
> D <- diag(e$values)
> C <- e$vectors
> C%*%D%*%t(C)
      [,1]          [,2]          [,3]          [,4]
[1,]  0.6856935 -0.0424340   1.2743154   0.5162707
[2,] -0.0424340   0.1899794  -0.3296564  -0.1216394
[3,]  1.2743154 -0.3296564   3.1162779   1.2956094
[4,]  0.5162707 -0.1216394   1.2956094   0.5810063

```

```

> # Eigen-decomposition
> S%*%C[,1]
      [,1]
Sepal.Length  1.5280299
Sepal.Width   -0.3573816
Petal.Length  3.6222104
Petal.Width   1.5149333
> D[1]*C[,1]
[1]  1.5280299 -0.3573816
[3]  3.6222104  1.5149333

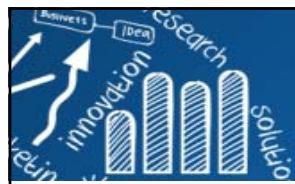
```



# Some Decompositions in R

- Eigen-decomposition
  - `eigen {base}`: Spectral Decomposition of a Matrix
- LU decomposition
  - `LU-class {Matrix}`: LU (dense) Matrix Decompositions
  - `lu {Matrix}`: (Generalized) Triangular Decomposition of a Matrix
  - `LU.decompose {optR}`: LU decomposition
- `svd {base}`: Singular Value Decomposition of a Matrix
- `qr {base}`: The QR Decomposition of a Matrix
  - If  $A$  is a  $m \times n$  matrix with linearly independent columns, then  $A$  can be decomposed as,  $A=QR$ , where  $Q$  is a  $m \times n$  matrix whose columns form an orthonormal basis for the column space of  $A$  and  $R$  is an nonsingular upper triangular matrix.
- `chol {base}`: The Choleski Decomposition
  - Cholesky Decomposition: If  $A$  is a real, symmetric and positive definite matrix then there exists a unique lower triangular matrix  $L$  with positive diagonal element such that  $A=LL^T$ .

```
> eigen(cov(iris[,1:4]))
```



## (2) Singular Value Decomposition

Singular Value Decomposition (SVD) factorizes a general  $m \times n$  (assume  $m \geq n$ ) matrix in the form

奇異值分解

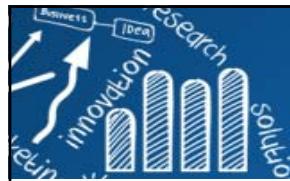
$$A = \underline{USV^t},$$

where  $U$  is an  $m \times m$  orthogonal matrix,  $V$  is an  $n \times n$  orthogonal matrix, and  $S$  is an  $m \times n$  matrix whose only nonzero elements lie along the main diagonal.

$$\begin{matrix} A \\ \text{---} \\ m \text{ rows} \\ n \text{ columns} \end{matrix} = \begin{matrix} U \\ \text{---} \\ m \text{ rows} \\ m \text{ columns} \end{matrix} \cdot \begin{matrix} S \\ \text{---} \\ m \text{ rows} \\ n \text{ columns} \end{matrix} \cdot \begin{matrix} V^t \\ \text{---} \\ n \text{ rows} \\ n \text{ columns} \end{matrix}$$

$$\begin{matrix} A_k \doteq A \\ \text{---} \\ m \text{ rows} \\ n \text{ columns} \end{matrix} = \begin{matrix} U_k \\ \text{---} \\ m \text{ rows} \\ k \text{ columns} \end{matrix} \cdot \begin{matrix} S_k \\ \text{---} \\ k \text{ rows} \\ k \text{ columns} \end{matrix} \cdot \begin{matrix} V_k^t \\ \text{---} \\ k \text{ rows} \\ n \text{ columns} \end{matrix}$$

- The  $m$  columns of  $U$  are called the **left singular vectors**. The  $n$  columns of  $V$  are called the **right singular vectors**.
  - The right singular vectors are eigenvectors of  $A^T A$ .
  - The left singular vectors are eigenvectors of  $A A^T$ .
  - The non-zero values of  $S$  are the square root of the eigenvalues of  $A A^T$  and  $A^T A$  are called the singular values
- $U$  and  $V$  are unitary matrixes, i.e.  $U U^T = I$ ,  $V V^T = I$ .



# SVD Examples

## SVD in R

- **svd{base}**: Singular Value Decomposition of a Matrix  
<https://stat.ethz.ch/R-manual/R-devel/library/base/html/svd.html>
- **fast.svd{corpcor}**: Fast Singular Value Decomposition  
<http://svtsrv25.epfl.ch/R-doc/library/corpcor/html/fast.svd.html>
- R Package '**svd**': Interfaces to various state-of-art SVD and eigensolvers  
<https://cran.r-project.org/web/packages/svd/index.html>
- R Package '**svdvisual**' : SVD visualization tools  
<https://cran.r-project.org/web/packages/svdvisual/index.html>

## R Code Examples

- Data Mining Algorithms In R/Dimensionality Reduction/Singular Value Decomposition  
[https://en.wikibooks.org/wiki/Data\\_Mining\\_Algorithms\\_In\\_R/Dimensionality\\_Reduction/Singular\\_Value\\_Decomposition](https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Dimensionality_Reduction/Singular_Value_Decomposition)
- R Code Fragments, Examples of Singular Value Decomposition  
[http://www.ats.ucla.edu/stat/r/pages/svd\\_demos.htm](http://www.ats.ucla.edu/stat/r/pages/svd_demos.htm)
- Running PCA and SVD in R  
[http://genomicsclass.github.io/book/pages/pca\\_svd.html](http://genomicsclass.github.io/book/pages/pca_svd.html)
- R-bloggers: Using the SVD to find the needle in the haystack  
<http://www.r-bloggers.com/using-the-svd-to-find-the-needle-in-the-haystack/>
- Image Compression with the SVD in R  
<http://www.johnmyleswhite.com/notebook/2009/12/17/image-compression-with-the-svd-in-r/>
- The Netflix Prize, Big Data, SVD and R  
<http://blog.revolutionanalytics.com/2011/05/the-netflix-prize-big-data-svd-and-r.html>
- Singular Value Decomposition in R  
<http://www.di.fc.ul.pt/~jpn/r/svd/svd.html>



# SVD: Making Approximations

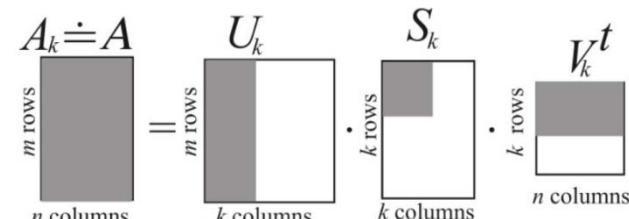
```

> iris.sub <- iris[sample(1:150, 8),1:4]
> iris.sub
  Sepal.Length Sepal.Width Petal.Length Petal.Width
58          4.9       2.4        3.3        1.0
59          6.6       2.9        4.6        1.3
77          6.8       2.8        4.8        1.4
84          6.0       2.7        5.1        1.6
54          5.5       2.3        4.0        1.3
39          4.4       3.0        1.3        0.2
71          5.9       3.2        4.8        1.8
87          6.7       3.1        4.7        1.5
> M.svd <- svd(iris.sub)
> M.svd
$d
[1] 22.2553197  2.2360595  0.7611307  0.1773813

$u
      [,1]      [,2]      [,3]      [,4]
[1,] -0.2898553 -0.096733080 -0.007319459 -0.05013322
[2,] -0.3885172 -0.006908608  0.305429166 -0.28018114
[3,] -0.3992219  0.066823762  0.450619905  0.06954991
[4,] -0.3793864  0.327663490 -0.174978375 -0.69121320
[5,] -0.3275406  0.106859076  0.218964907  0.50428246
[6,] -0.2285027 -0.918407065 -0.136688043 -0.13248141
[7,] -0.3782377  0.154232634 -0.776519488  0.27425130
[8,] -0.3989561 -0.009386179  0.058068514  0.29884180

$v
      [,1]      [,2]      [,3]      [,4]
[1,] -0.7497994 -0.3154600  0.5318218  0.2354811
[2,] -0.3527468 -0.5480151 -0.7276317 -0.2140122
[3,] -0.5343751  0.7023730 -0.1376906 -0.4496184
[4,] -0.1667746  0.3268587 -0.4108028  0.8346201

```



```

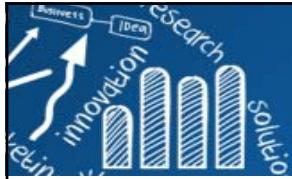
> M.svd$u %*% (diag(M.svd$d) %*% t(M.svd$v))
      [,1] [,2] [,3] [,4]
[1,] 4.9  2.4  3.3  1.0
[2,] 6.6  2.9  4.6  1.3
[3,] 6.8  2.8  4.8  1.4
[4,] 6.0  2.7  5.1  1.6
[5,] 5.5  2.3  4.0  1.3
[6,] 4.4  3.0  1.3  0.2
[7,] 5.9  3.2  4.8  1.8
[8,] 6.7  3.1  4.7  1.5

>
> # use the first two values to approximate
> d.sub <- diag(M.svd$d[1:2])
> u.sub <- as.matrix(M.svd$u[, 1:2])
> v.sub <- as.matrix(M.svd$v[, 1:2])
> iris.sub.approx <- u.sub %*% d.sub %*% t(v.sub)
> iris.sub.approx
      [,1]      [,2]      [,3]      [,4]
[1,] 4.905057  2.394043  3.295235  1.0051334
[2,] 6.488070  3.058517  4.609664  1.4369796
[3,] 6.614690  3.052204  4.852772  1.5306008
[4,] 6.099701  2.576853  5.026535  1.6476201
[5,] 5.390302  2.440411  4.063166  1.2938078
[6,] 4.460863  2.919270  1.275109  0.1768745
[7,] 6.202869  2.780357  4.740493  1.5166003
[8,] 6.664012  3.143504  4.729919  1.4739142

>
> # compute the sum of squared errors
> sum((iris.sub - iris.sub.approx)^2)
[1] 0.610784

```

# SVD: Image Compression (1/3)

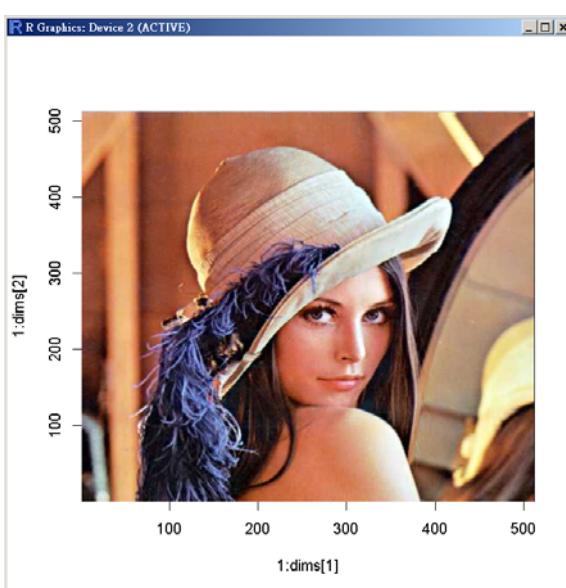


Lenna 97: A Complete Story of Lenna

<http://www.ee.cityu.edu.hk/~lmpo/lenna/Lenna97.html>

<https://en.wikipedia.org/wiki/Lenna>

This scan became one of the most used images in computer history.[4] In a 1999 issue of *IEEE Transactions on Image Processing* "Lena" was used in three separate articles,[5] and the picture continued to appear in scientific journals throughout the beginning of the 21st century. ... To explain Lena's popularity, David C. Munson, editor-in-chief of IEEE Transactions on Image Processing, noted that it was a good test image because of its **detail, flat regions, shading, and texture**. However, he also noted that its popularity was largely because an image of an attractive woman appealed to the males in a male-dominated field

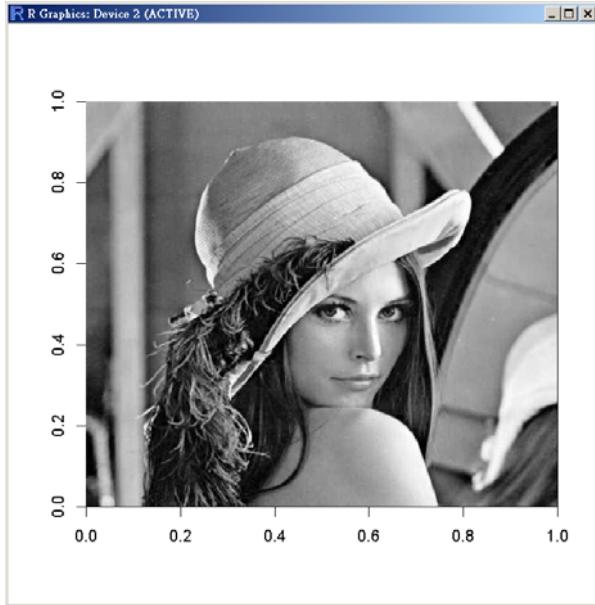
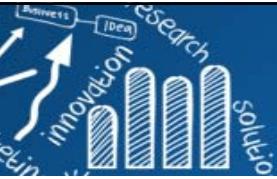


```
> # require packages: locfit, tiff, fftwtools
> library(EBImage) # (Repositories: BioC Software)
> lena <- readImage("lena.jpg")
> dims <- dim(lena)
> dims
[1] 512 512    3
>
> plot(c(0, dims[1]), c(0, dims[2]), type='n', xlab="", ylab="")
> rasterImage(lena, 0, 0, dims[1], dims[2])
```

```
> source("https://bioconductor.org/biocLite.R")
> biocLite("EBImage")
```

```
> #install.packages("jpeg")
> library(jpeg)
> lena <- readJPEG("lena.jpg")
```

# SVD: Image Compression (2/3)



```

> lena.flip <- Image(flip(lena))
> # convert RGB to grayscale
> red.weight   <- .2989
> green.weight <- .587
> blue.weight  <- 0.114
>
> lena.gray <- red.weight * imageData(lena.flip)[,,1] +
+               green.weight * imageData(lena.flip)[,,2] +
+               blue.weight  * imageData(lena.flip)[,,3]
> dim(lena.gray)
[1] 512 512
> lena.gray[1:5, 1:5]
 [,1]      [,2]      [,3]      [,4]      [,5]
 [1,] 0.07128863 0.06344627 0.05952510 0.06736745 0.07913098
 [2,] 0.07913098 0.06736745 0.06344627 0.07128863 0.08305216
 [3,] 0.08697333 0.07520980 0.07128863 0.07913098 0.09089451
 [4,] 0.09089451 0.08305216 0.07913098 0.08305216 0.09873686
 [5,] 0.09570980 0.08786745 0.08002510 0.08786745 0.10355216
> image(lena.gray, col = grey(seq(0, 1, length = 256)))

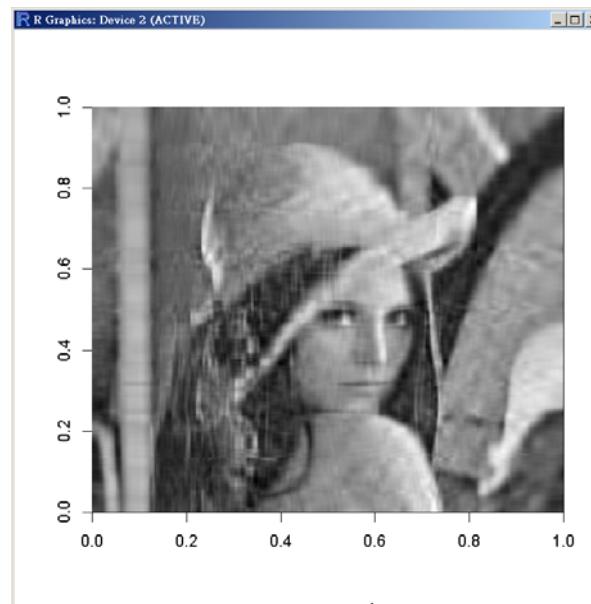
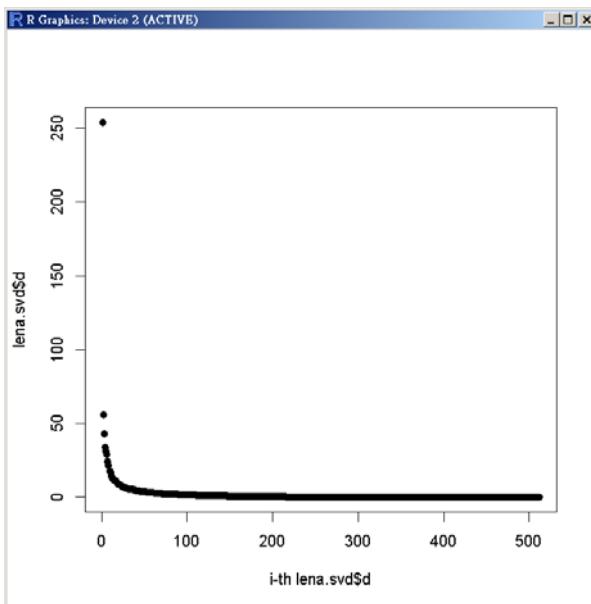
```

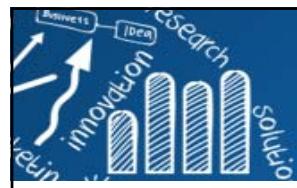
Converting RGB to grayscale/intensity

<http://stackoverflow.com/questions/687261/converting-rgb-to-grayscale-intensity>

# SVD: Image Compression (3/3)

```
> lena.svd <- svd(lena.gray)
> d <- diag(lena.svd$d)
> dim(d)
[1] 512 512
> u <- lena.svd$u
> v <- lena.svd$v
> plot(1:length(lena.svd$d), lena.svd$d, pch=19, xlab="i-th lena.svd$d", ylab="lena.svd$d")
>
> used.no <- 20
> u.sub <- as.matrix(u[, 1:used.no])
> v.sub <- as.matrix(v[, 1:used.no])
> d.sub <- as.matrix(d[1:used.no, 1:used.no])
> lena.approx <- u.sub %*% d.sub %*% t(v.sub)
> image(lena.approx, col = grey(seq(0, 1, length = 256)))
```

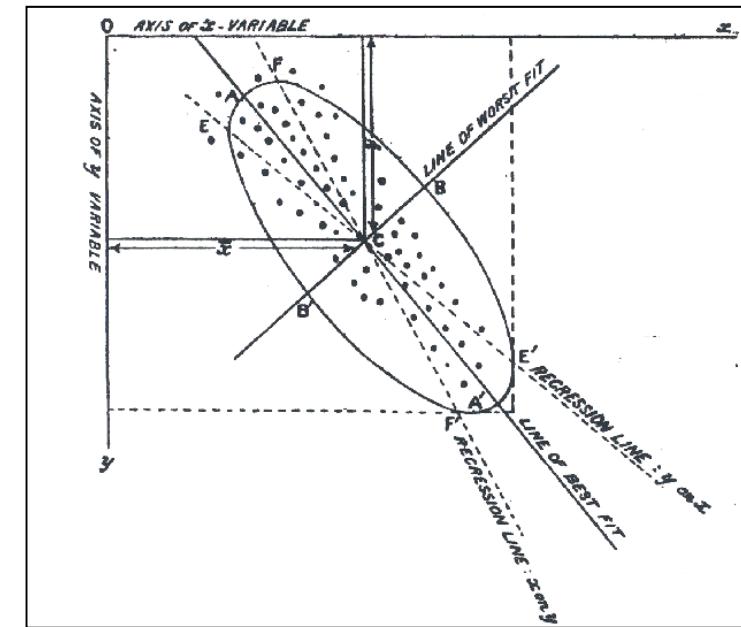
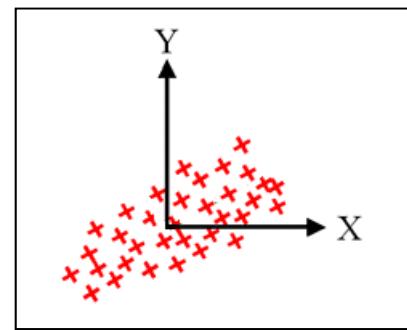




### (3) Principal Component Analysis

MA Table	exp01	exp02	exp03	exp04	exp05	exp...	exp P
gene001	-0.48	-0.2	0.87	0.92	0.67		-0.35
gene002	-0.39	-0.38	1.03	1.21	0.52		-0.58
gene003	0.87	0.38	-0.17	0.18	-0.13		-0.13
gene004	1.57	1.03	1.22	0.31	0.16		-1.02
gene005	-1.15	-0.68	1.21	1.62	1.12		-0.44
gene006	0.01	-0.2	0.3	0.16	0.17		0.08
gene007	2.95	0.5	-0.40	-0.66	-0.59		-0.76
gene008	-1.22	-0.14	1.34	1.50	0.63		-0.55
gene009	-0.78	-1.06	-0.70	-0.02	0.16		0.03
gene010	-0.53	-0.40	0.10	0.58	-0.09		-0.45
gene011	-0.50	-0.22	0.66	1.05	0.68		0.01
gene012	-0.86	-0.19	0.42	0.46	0.30		-0.63
gene013	-0.16	0.19	0.17	-0.28	-0.02		-0.04
gene014	-0.36	-0.15	-0.08	-0.08	-0.23		-0.21
gene015	-0.72	-0.08	0.54	1.04	0.84		-0.64
gene016	-0.78	-0.12	0.26	0.20	0.48		0.27
gene017	0.60	-0.46	0.41	0.45	0.18		-1.02
gene018	-0.20	-0.47	0.18	0.10	0.38		0.05
gene019	-2.29	-0.44	0.77	1.60	0.53		-0.38
gene020	-1.46	-0.18	1.03	1.50	0.74		-0.70
gene021	-0.57	0.22	1.03	1.35	0.64		-0.40
gene022	-0.1	0.3	0.40	0.60	0.23		0.19
gene n	-1.79	0.94	2.13	1.75	0.23		-0.66

Karl Pearson 1901;  
Hotelling 1933;  
Jolliffe 2002



On lines and planes of closest fit to systems of points in space.

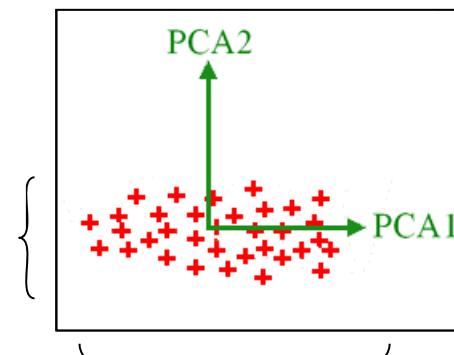
The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, 6 (2),  
559-572, 1901.

- As an **exploratory** tool to uncover unknown trends in the data.
- Make a **visual inspection** of the relationship between observations in a multi-dimensional matrix.
- PCA is a method that **reduces** data dimensionality by performing a **covariance analysis** between factors.
- PCA is to ‘**summarize**’ the data, it is not considered a clustering tool.



# PCA

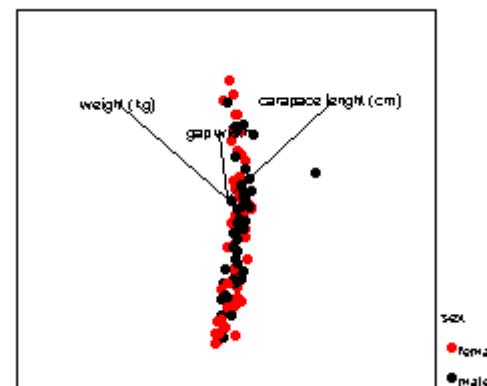
**PCA** is a method that reduces data dimensionality by finding the new variables (major axes, principal components).



$$\text{PCA}_1 = a_1 \mathbf{X} + b_1 \mathbf{Y}$$

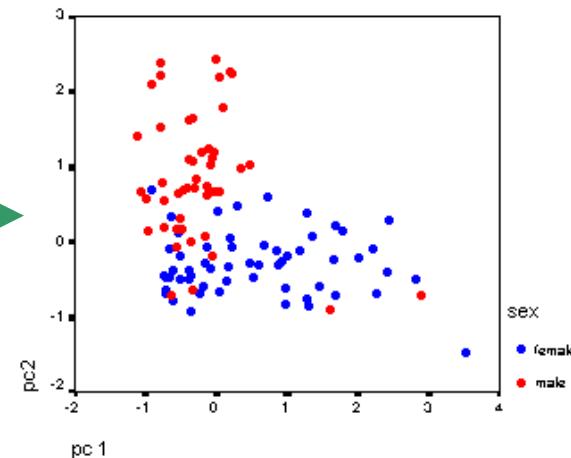
$$\text{PCA}_2 = a_2 \mathbf{X} + b_2 \mathbf{Y}$$

Image source: 61BL4165 Multivariate Statistics, Department of Biological Sciences, Manchester Metropolitan University



$$\text{PCA}_1 = a_1 \mathbf{X} + b_1 \mathbf{Y} + c_1 \mathbf{Z}$$

$$\text{PCA}_2 = a_2 \mathbf{X} + b_2 \mathbf{Y} + c_2 \mathbf{Z}$$



$$\text{PCA}_1 = a_{11} \mathbf{X}_1 + a_{12} \mathbf{X}_2 + \cdots + a_{1p} \mathbf{X}_p$$

$$\text{PCA}_2 = a_{21} \mathbf{X}_1 + a_{22} \mathbf{X}_2 + \cdots + a_{2p} \mathbf{X}_p$$

Amongst all possible projections, PCA finds the projections so that the maximum amount of information, measured in terms of variability, is retained in the smallest number of dimensions.



# PCA: Loadings and Scores

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

**Scores Matrix**

	1	2	...	k	...	p
1	1	2	...	k	...	p
2						
3						
.						
.						
.						
n						

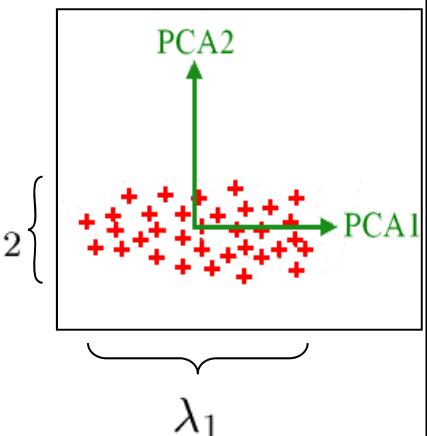
**Data Matrix**

	x1	x2	...	xp
1	1	2	...	p
2				
3				
.				
.				
.				
n				

**Loadings Matrix**

	1	2	...	k	...	p
1	1	2	...	k	...	p
2						
3						
.						
.						
.						
p						

$\mathbf{X}$

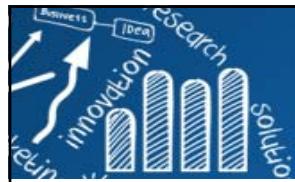


The  $i$ th principal component of  $\mathbf{X}$  is  $\mathbf{X}\mathbf{w}_i$ , where  $\mathbf{w}_i$  is the  $i$ th normalized eigenvector of  $\Sigma_{\mathbf{x}}$  corresponding to the  $i$ th largest eigenvalues.

Eigenvalues  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$

$$\text{proportion} = \frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i}$$

PCA is the spectral decomposition of the variance covariance matrix,  $\mathbf{S}$ , of the data matrix,  $\mathbf{X}$ . we can write  $\mathbf{S}=\mathbf{CDC}^T$ .



## PCA: General Methodology (1/3)

- Find a low-dimensional space such that when  $\mathbf{x}$  is projected there, information loss is minimized.
- PCA **centers** the sample and then **rotates** the axes to line up with the directions of **highest** variance.

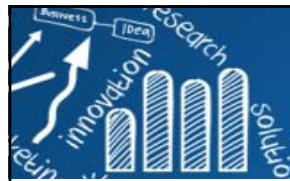
The projection of  $\mathbf{x}$  on the direction of  $\mathbf{w}$  is:  $z = \mathbf{w}^T \mathbf{x}$

Find  $\mathbf{w}$  such that  $\text{var}(z)$  is maximized

$$\begin{aligned}\text{var}(z) &= \text{var}(\mathbf{w}^T \mathbf{x}) \\ &= E[(\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mu)^2] \\ &= E[\mathbf{w}^T (\mathbf{x} - \mu)^2 \mathbf{w}] \\ &= E[\mathbf{w}^T (\mathbf{x} - \mu)(\mathbf{x} - \mu)^T \mathbf{w}] \\ &= \mathbf{w}^T E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T] \mathbf{w}\end{aligned}$$

$$\begin{aligned}\text{var}(\mathbf{x}) &= E[(\mathbf{x} - \mu)^2] \\ &= E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T] \\ &= \Sigma\end{aligned}$$

```
?cov  
x <- iris[, 1:4]  
cov(x)
```



## PCA: General Methodology (2/3)

$$\text{var}(z) = \mathbf{w}^T \Sigma \mathbf{w} \quad \text{Maximize } \text{var}(z) \text{ subject to } \|\mathbf{w}\|=1$$

$$\mathcal{L}(\lambda, \mathbf{w}_1) = \mathbf{w}_1^T \Sigma \mathbf{w}_1 - \lambda (\mathbf{w}_1^T \mathbf{w}_1 - 1) \quad \text{Lagrangian method}$$

$$\max_{\mathbf{w}_1} \mathcal{L}(\lambda, \mathbf{w}_1) \rightarrow$$

$$\frac{\partial \mathcal{L}(\lambda, \mathbf{w}_1)}{\partial \lambda} = 0 \rightarrow \mathbf{w}_1^T \mathbf{w}_1 - 1 = 0$$

$$\frac{\partial \mathcal{L}(\lambda, \mathbf{w}_1)}{\partial \mathbf{w}_1} = 0 \rightarrow 2 \Sigma \mathbf{w}_1 - 2\lambda \mathbf{w}_1 = 0$$

$$\rightarrow \Sigma \mathbf{w}_1 = \lambda \mathbf{w}_1$$

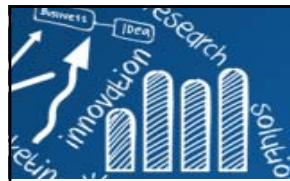
$\rightarrow \mathbf{w}_1$  is an eigenvector of  $\Sigma$

$\lambda$  is an eigenvalue associated with  $\mathbf{w}_1$

```
x <- iris[, 1:4]
(covx <- cov(x))
e <- eigen(covx)
v <- e$vectors
v.inverse <- solve(e$vectors)
covx.hat <- v %*% diag(e$values) %*% v.inverse
covx.hat # same with covx
```

$$\text{var}(z) = \text{var}(\mathbf{w}_1^T \mathbf{x}) = \mathbf{w}_1^T \Sigma \mathbf{w}_1 = \mathbf{w}_1^T \lambda \mathbf{w}_1 = \lambda \mathbf{w}_1^T \mathbf{w}_1 = \lambda$$

$$\rightarrow \max \text{var}(z) = \max \lambda$$



## PCA: General Methodology (3/3)

Second principal component:

Max  $\text{var}(z_2)$ , s.t.,  $\|\mathbf{w}_2\|=1$  and orthogonal to  $\mathbf{w}_1$

$$\mathcal{L}(\lambda_2, \beta, \mathbf{w}_1, \mathbf{w}_2) = \mathbf{w}_2^T \Sigma \mathbf{w}_2 - \lambda_2 (\mathbf{w}_2^T \mathbf{w}_2 - 1) - \beta (\mathbf{w}_2^T \mathbf{w}_1 - 0)$$

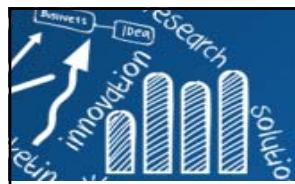
$$\max_{\mathbf{w}_2} \mathcal{L}(\lambda_2, \beta, \mathbf{w}_1, \mathbf{w}_2) \rightarrow \Sigma \mathbf{w}_2 = \lambda_2 \mathbf{w}_2$$

$\mathbf{w}_i$  is the eigenvector of  $\Sigma$  associated with the  $i^{th}$  largest eigenvalue.

$$\text{var}(z_i) = \lambda_i$$

**Proof (by induction – Assume true for  $i-1$ , then prove true for  $i$ )**

- $\mathbf{w}_i$ 's are uncorrelated (orthogonal)
- $\mathbf{w}_1$  explains as much as possible of original variance in data set
- $\mathbf{w}_2$  explains as much as possible of remaining variance, etc.



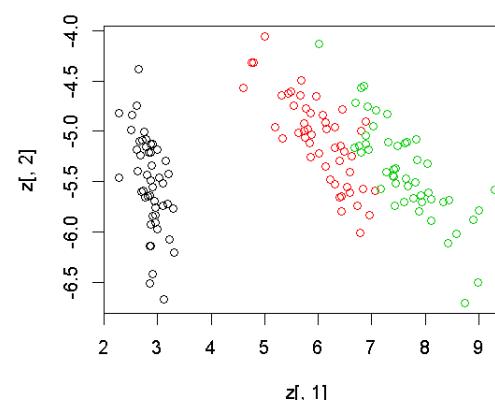
# PCA: Solutions

- If the dimensions are **highly correlated**, there will be a **small number of eigenvectors** with **large eigenvalues**, and  $k$  will be much smaller than  $d$  and a large reduction in dimensionality may be attained.
- If the dimensions are not correlated,  $k$  will be as large as  $d$  and there is no gain through PCA.

$$\text{var}(z) = w^T \Sigma w$$

- **$S$  is symmetric:** for two different eigenvalues, the eigenvectors are orthogonal.
- **If  $S$  is positive definite:** ( $x^T S x > 0, x \neq \text{null}$ ): all eigenvalues are positive.
- **If  $S$  is singular:** then its rank, the effective dimension is  $k < d$ , and  $\lambda_i = 0, i > k$ .

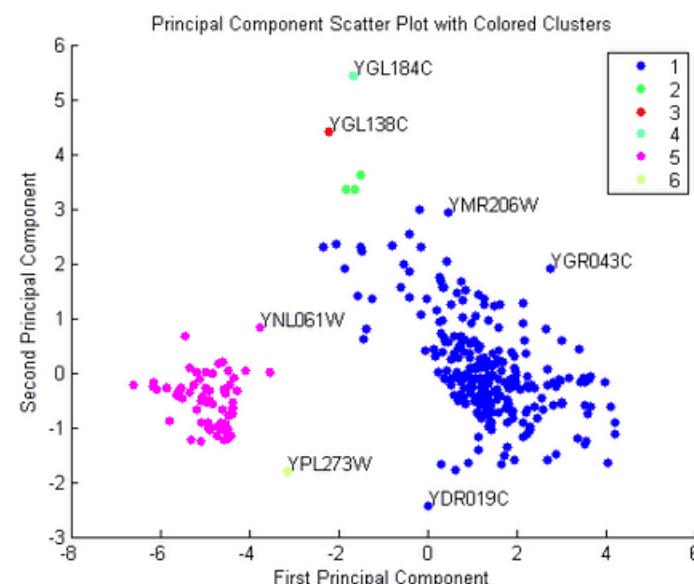
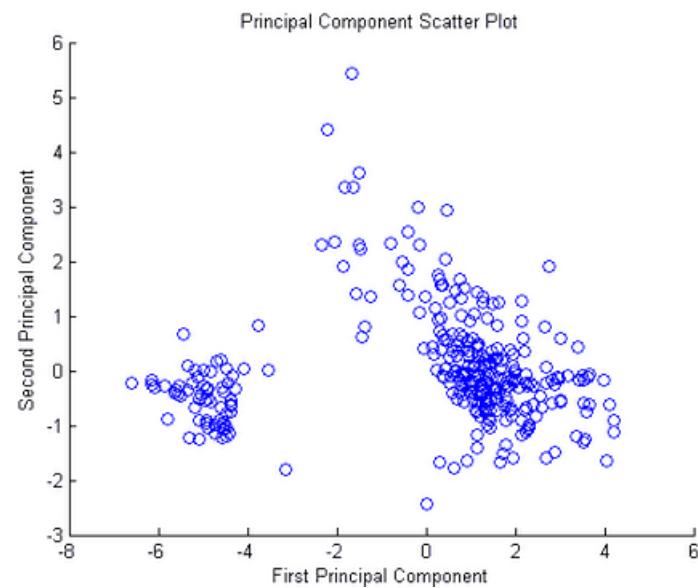
```
# PCA for iris data
z <- as.matrix(x) %*% e$vectors[, 1:2]
plot(z[, 1], z[, 2], col=iris[, 5])
```





# When to use PCA?

- As an exploratory tool to uncover unknown trends in the data.
- PCA on genes provide a way to identify predominant gene expression patterns.
- PCA on conditions explore correlations between samples or conditions.
- PCA is to ‘summarize’ the data, it is not considered a clustering tool.



*Yeast Microarray Data is from*

DeRisi, JL, Iyer, VR, and Brown, PO.(1997). "Exploring the metabolic and genetic control of gene expression on a genomic scale"; Science, Oct 24;278(5338):680-6.



# How Many Components to Use?

- The proportion of Variance explained by the first  $k$  principal components

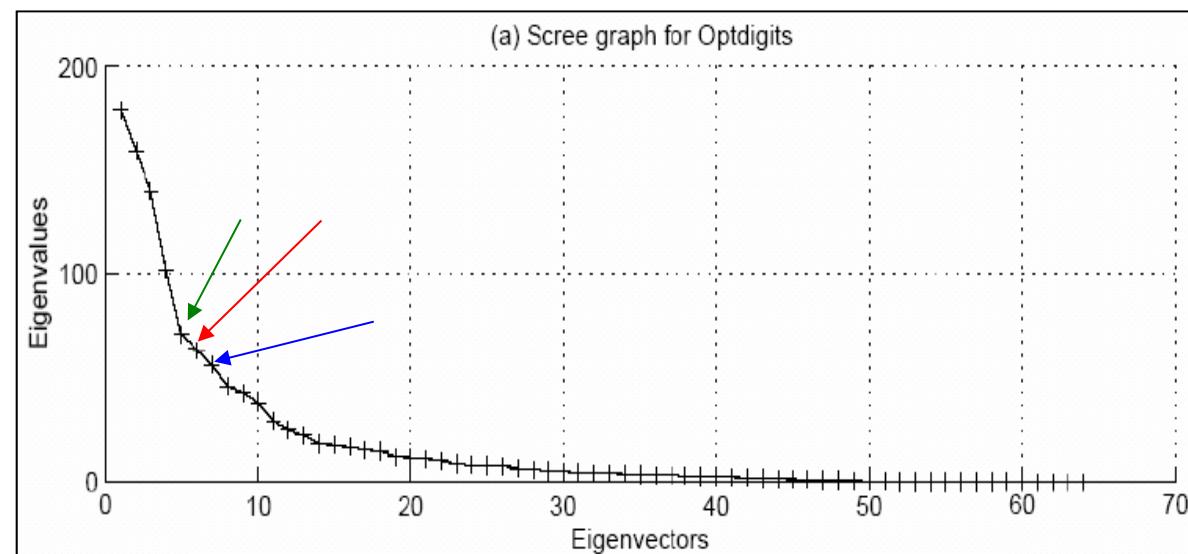
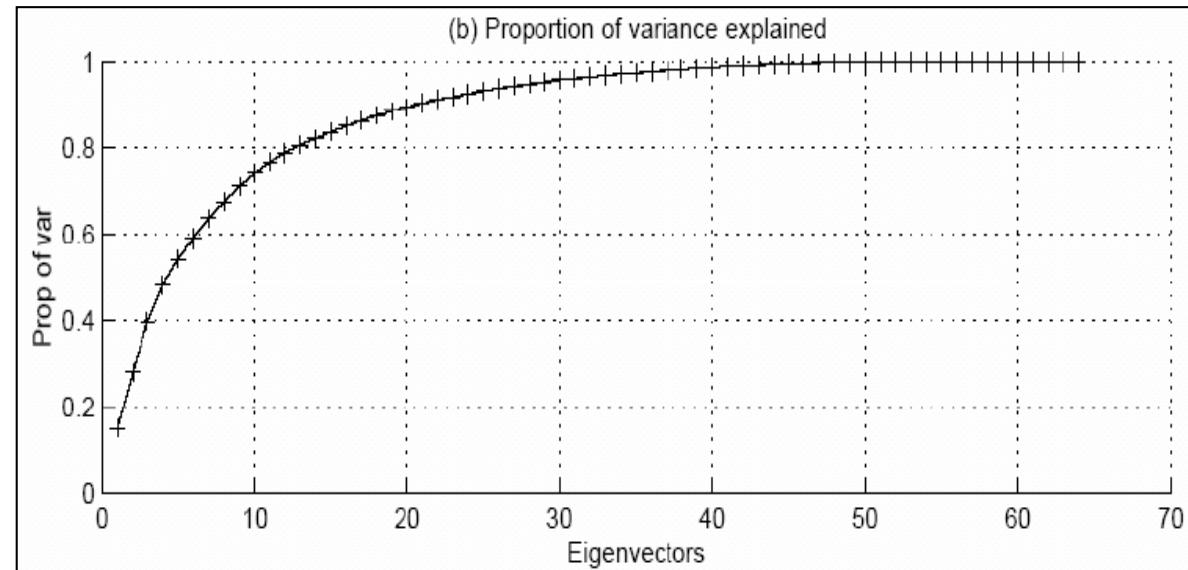
$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d} = \frac{\sum_{j=1}^r \lambda_j}{\text{tr}(\Sigma)}.$$

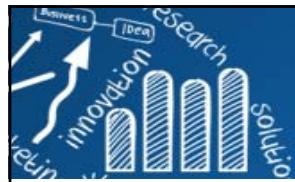
- If  $\lambda_j < 1$  then component explains less variance than original variable (correlation matrix).
- Use 2 components (or 3) for visual ease.
- Keep only the eigenvectors with eigenvalues greater than average eigenvalue.
- Keep only those which have higher than the average input variance.



# Scree Diagram

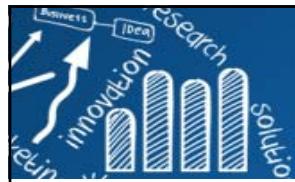
At the **elbow**, adding another eigenvector does not significantly increase the variance explained.





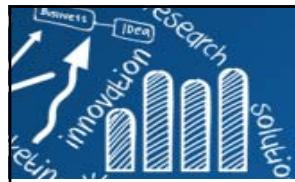
## PCA: Scaling (1/2)

- Different variables have completely different scaling.
  - Eigenvalues of the matrix is scale dependent.
  - (If we would multiply one column of the data matrix  $X$  by some scale factor (say  $s$ ) then variance of this variable would increase by  $s^2$  and this variable can dominate whole covariance matrix and hence whole eigenvalue and eigenvectors.)
- Covariance Matrix
  - Variables must be in same units.
  - Emphasizes variables with most variance.
  - Mean eigenvalue > 1.0
- Correlation Matrix
  - Variables are standardized (mean=0.0, SD=1.0)
  - Variables can be in different units.
  - All variables have same impact on analysis.
  - Mean eigenvalue = 1.0



## PCA: Scaling (2/2)

- Bring all data to the **same scale**: If scale of the data is unknown then it is better to use **correlation matrix** instead of the covariance matrix.
- The interpretation of the principal components derived by these two methods (**Covariance Matrix and Correlation Matrix**) can be completely different.
- If the variance of the original  $x_i$  dimensions vary considerably, they affect the directions of the principal components more than the correlations.
  - Standardized data by columns, mean=0, sd=1,
  - PCA on correlation matrix, for the correlations to be effective and not the individual variances.



# PCA: Potential Problems

- Sensitive to outliers
  - Robust estimation
  - Discarding the isolated data points that are far away.

- Lack of Independence
  - NO PROBLEM

- Lack of Normality
  - Normality desirable but not essential

- Lack of Precision
  - Precision desirable but not essential

- Many Zeroes in Data Matrix
  - Problem  
(use Correspondence Analysis)

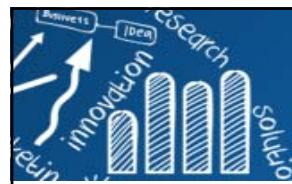
This works better than taking the average because we take into account **correlations** and **differences in variances**.

Example:

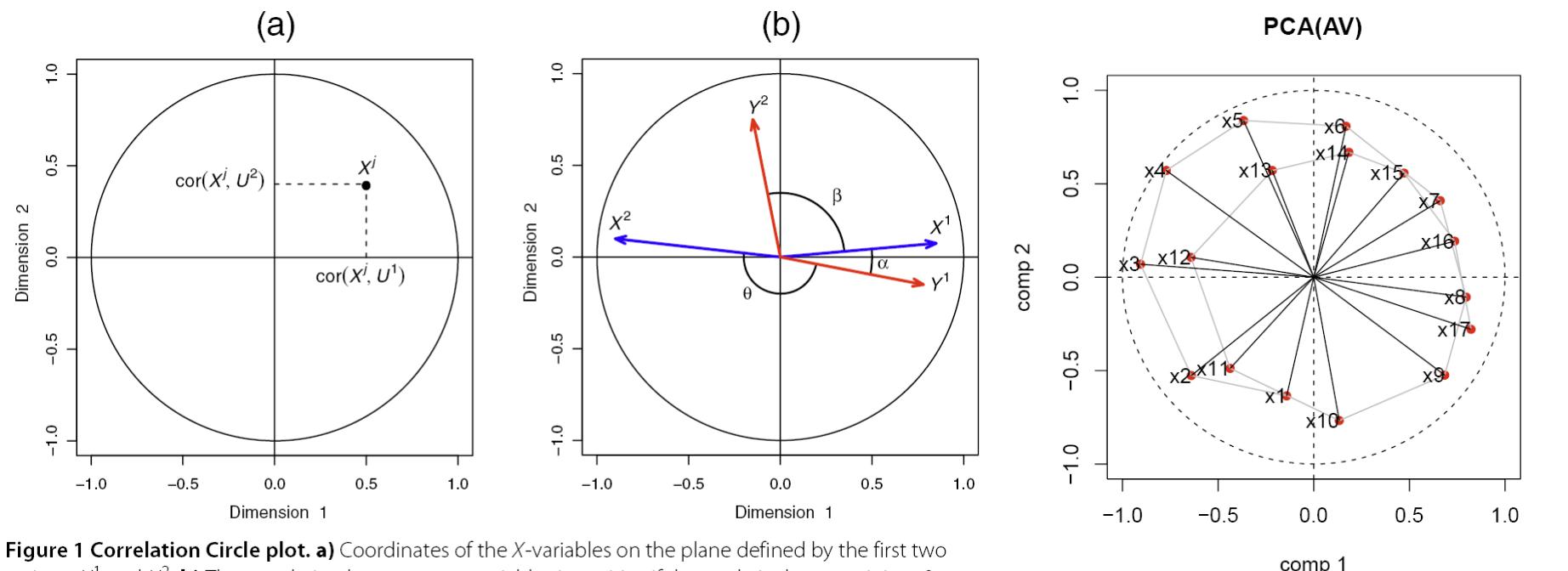
	A	B	C	D	E	F
1	學號	小考1	期中考	小考2	期末考	報告
2	A01	69	92	85	45	62
3	A02	66	90	83	36	90
4	A03	72	92	80	62	70
5	A04	68	90	60	37	95
6	A05	74	60	86	54	70
7	A06	77	90	88	88	95
8	A07	73	88	77	51	95
9	A08	61	90	84	40	82
10	A09	66	88	82	39	80
11	A10	76	75	87	72	80
12	A11	64	90	90	26	95
13	A12	75	90	60	55	70
14	A13	92	90	83	90	95

average

平均	PCA1



# Circle of Correlations



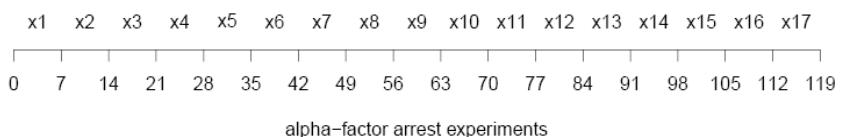
**Figure 1 Correlation Circle plot.** **a)** Coordinates of the  $X$ -variables on the plane defined by the first two variates  $U^1$  and  $U^2$ . **b)** The correlation between two variables is positive if the angle is sharp  $\cos(\alpha) > 0$ , negative if the angle is obtuse  $\cos(\theta) < 0$ , and null if the vectors are perpendicular  $\cos(\beta) \approx 0$ .

González et al. BioData Mining 2012, 5:19  
<http://www.biodatamining.org/content/5/1/19>

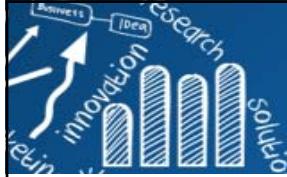
the input variables,  $X_j$ ,

the DR components,  $Z_k$ ,

$$\rho_{jk} = \text{Corr}(X_j; Z_k) = \frac{\text{Cov}(X_j, Z_k)}{\sqrt{\text{Var}(X_j)\text{Var}(Z_k)}}.$$



Circles of correlations associated with the first two DR components based on the applied vector and path point approaches applied to the microarray data of the yeast cell cycle.



# Principal Components Analysis and SVD

The principal components are equal to the **right singular values** if the variables of data is scaled. (subtract the mean, divide by the standard deviation).

```
> m1 <- matrix(sample(1:16,16),4,4)
> m1
      [,1] [,2] [,3] [,4]
[1,]    15     1    16     7
[2,]     8     9     2    10
[3,]    13     4    12     3
[4,]     5    14    11     6
> m1.scale.svd <- svd(scale(m1))
> m1.scale.svd
$d
[1] 2.827057e+00 1.759769e+00 9.544443e-01 4.258869e-17

$u
      [,1]      [,2]      [,3]      [,4]
[1,] -0.5580839 -0.3507292  0.5617218   0.5
[2,]  0.5716314 -0.5999038 -0.2517003   0.5
[3,] -0.4320314  0.2946885 -0.6902952   0.5
[4,]  0.4184839  0.6559445  0.3802737   0.5

$v
      [,1]      [,2]      [,3]      [,4]
[1,] -0.5663183 -0.3664762 -0.1512544  0.72256547
[2,]  0.5395013  0.4585555  0.1574801  0.68837872
[3,] -0.5008709  0.3789166  0.7772586 -0.03767775
[4,]  0.3706081 -0.7154329  0.5900773  0.05112996
```

```
> m1.pca <- prcomp(m1, scale=T)
> m1.pca
Standard deviations:
[1] 1.6322e+00 1.016003e+00 5.510487e-01 2.458859e-17

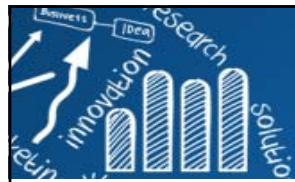
Rotation:
          PC1        PC2        PC3        PC4
[1,] -0.5663183 -0.3664762 -0.1512544  0.72256547
[2,]  0.5395013  0.4585555  0.1574801  0.68837872
[3,] -0.5008709  0.3789166  0.7772586 -0.03767775
[4,]  0.3706081 -0.7154329  0.5900773  0.05112996

> pca2 <- princomp(m1, cor=T)
> pca2$scores
  Comp.1       Comp.2       Comp.3       Comp.4
[1,] 1.821811  0.7126839  0.6190721  6.227657e-16
[2,] -1.866036  1.2190081 -0.2773982 -9.020562e-16
[3,]  1.410325 -0.5988089 -0.7607725 -1.804112e-16
[4,] -1.366101 -1.3328832  0.4190986  4.649059e-16
> pca2$loadings

Loadings:
  Comp.1       Comp.2       Comp.3       Comp.4
[1,]  0.566   0.366  -0.151   0.723
[2,] -0.540  -0.459   0.157   0.688
[3,]  0.501  -0.379   0.777
[4,] -0.371   0.715   0.590

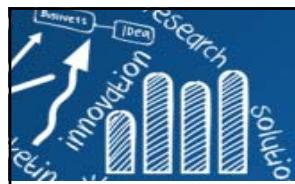
          Comp.1       Comp.2       Comp.3       Comp.4
ss loadings      1.00      1.00      1.00      1.00
Proportion Var  0.25      0.25      0.25      0.25
Cumulative Var  0.25      0.50      0.75      1.00
```

Source: Singular Value Decomposition, João Neto, <http://www.di.fc.ul.pt/~jpn/r/svd/svd.html>



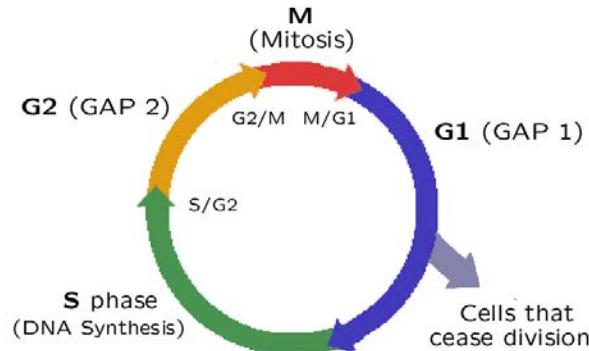
# PCA in R

- 5 functions to do Principal Components Analysis in R (by Gaston Sanchez): `prcomp{stats}`; `princomp{stats}`; `PCA{FactoMineR}`; `dudi.pca{ade4}`; `acp{amap}`; `epPCA{ExPosition}`  
[\(http://gastonsanchez.com/blog/how-to/2012/06/17/PCA-in-R.html\)](http://gastonsanchez.com/blog/how-to/2012/06/17/PCA-in-R.html)
- STHDA: Statistical tools for high-throughput data analysis
  - Principal component analysis in R : `prcomp()` vs. `princomp()` - R software and data mining  
[\(http://www.sthda.com/english/wiki/principal-component-analysis-in-r-prcomp-vs-princomp-r-software-and-data-mining\)](http://www.sthda.com/english/wiki/principal-component-analysis-in-r-prcomp-vs-princomp-r-software-and-data-mining)
  - `FactoMineR` and `factoextra` : Principal Component Analysis Visualization - R software and data mining  
[\(http://www.sthda.com/english/wiki/factominer-and-factoextra-principal-component-analysis-visualization-r-software-and-data-mining\)](http://www.sthda.com/english/wiki/factominer-and-factoextra-principal-component-analysis-visualization-r-software-and-data-mining)
- Gregory B. Anderson, principal component analysis in R  
[\(https://www.ime.usp.br/~pavan/pdf/MAE0330-PCA-R-2013\)](https://www.ime.usp.br/~pavan/pdf/MAE0330-PCA-R-2013)
- R Package "`bigpca`": Title PCA, Transpose and Multicore Functionality for '`big.matrix`'  
[\(https://cran.r-project.org/web/packages/bigpca/\)](https://cran.r-project.org/web/packages/bigpca/)



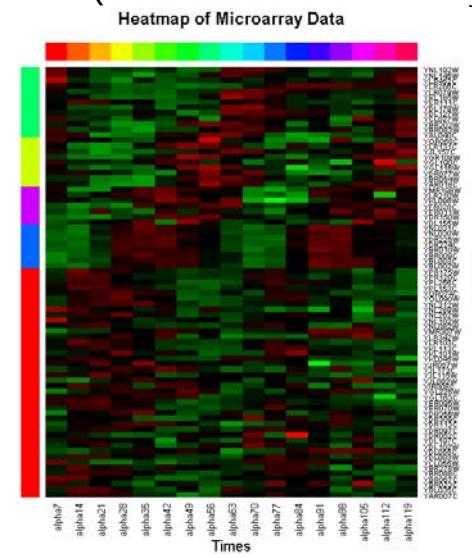
# 範例: Microarray Data of Yeast Cell Cycle

- Synchronized by alpha factor arrest method (Spellman et al. 1998; Chu et al. 1998)
- 103** known genes: every 7 minutes and totally **18** time points. (remove NA's: 79 genes)



Microarray Data Matrix

MA Table	exp01	exp02	exp03	exp04	exp05	exp...*	p
gene001	-0.48	-0.42	0.87	0.92	0.67		-0.35
gene002	-0.39	-0.58	1.08	1.21	0.52		-0.58
gene003	0.87	0.25	-0.17	0.18	-0.13		-0.13
gene004	1.57	1.03	1.22	0.31	0.16		-1.02
gene005	-1.15	-0.86	1.21	1.62	1.12		-0.44
gene006	0.04	-0.12	0.31	0.16	0.17		0.08
gene007	2.95	0.45	-0.40	-0.66	-0.59		-0.76
gene008	-1.22	-0.74	1.34	1.50	0.63		-0.55
gene009	-0.73	-1.06	-0.79	-0.02	0.16		0.03
gene010	-0.58	-0.40	0.13	0.58	-0.09		-0.45
gene011	-0.50	-0.42	0.66	1.05	0.68		0.01
gene012	-0.86	-0.29	0.42	0.46	0.30		-0.63
gene013	-0.16	0.29	0.17	-0.28	-0.02		-0.04
gene014	-0.36	-0.03	-0.03	-0.08	-0.23		-0.21
gene015	-0.72	-0.85	0.54	1.04	0.84		-0.64
gene016	-0.78	-0.52	0.26	0.20	0.48		0.27
gene017	0.60	-0.55	0.41	0.45	0.18		-1.02
gene018	-0.20	-0.67	0.13	0.10	0.38		0.05
gene019	-2.29	-0.64	0.77	1.60	0.53		-0.38
gene020	-1.46	-0.76	1.08	1.50	0.74		-0.70
gene021	-0.57	0.42	1.03	1.35	0.64		-0.40
gene022	-0.11	0.13	0.41	0.60	0.23		0.19
gene...*							-0.66
gene n	-1.79	0.94	2.13	1.75	0.23		



```

cell.matrix <- read.table("YeastCellCycle_alpha.txt", header=TRUE, row.names=1)
n <- dim(cell.matrix)[1]
p <- dim(cell.matrix)[2]-1
cell.data <- cell.matrix[,2:p+1]
gene.phase <- cell.matrix[,1]
phase <- unique(gene.phase)
phase.name <- c("G1", "S", "S/G2", "G2/M", "M/G1")
cell.sdata <- t(scale(t(cell.data)))
rc <- rainbow(5)[as.integer(gene.phase)]
cc <- rainbow(ncol(cell.sdata))
hv <- heatmap(cell.sdata, col = GBRcol, scale = "column", Colv=NA, Rowv=NA,
              RowSideColors = rc, ColSideColors = cc, margins = c(5,10),
              xlab = "Times", ylab = "Genes", main = "Heatmap of Microarray Data")

```



# PCA on Conditions

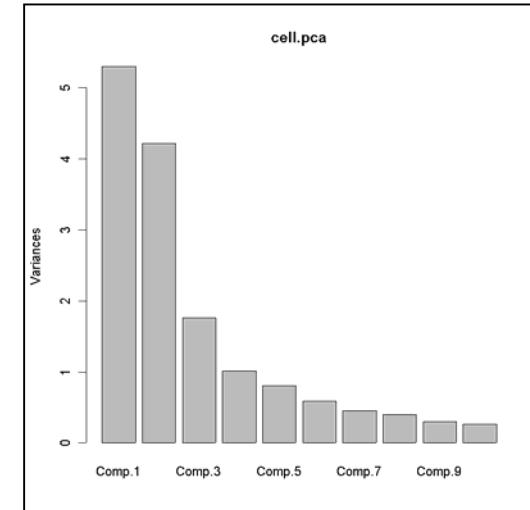
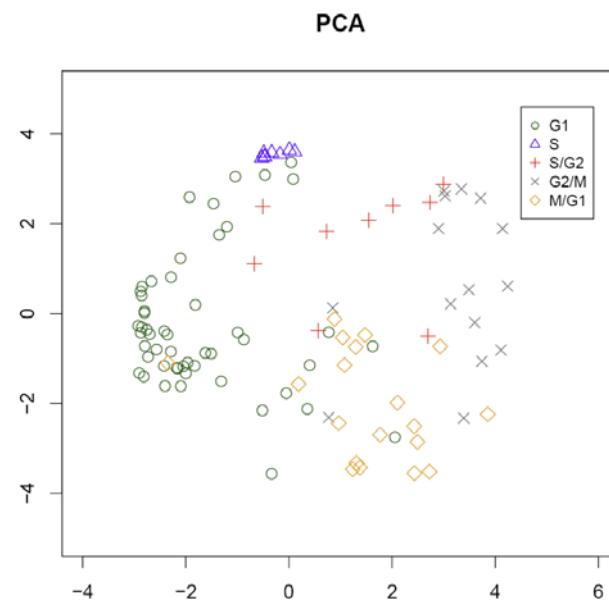
```

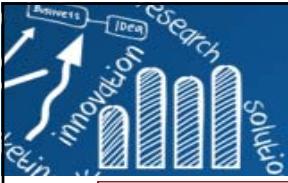
cell.pca <- princomp(cell.sdata, cor=TRUE, scores=TRUE)
# 2D plot for first two components
pca.dim1 <- cell.pca$scores[,1]
pca.dim2 <- cell.pca$scores[,2]
plot(pca.dim1, pca.dim2, main="PCA for Cell Cycle Data on Genes", xlab="1st PCA
Component", ylab="2nd PCA Component", col=c(phase), pch=c(phase))
legend(3, 4, phase.name, pch=c(phase), col=c(phase))

# shows a screeplot.
plot(cell.pca)
biplot(cell.pca)

```

MA Table	PCA-1	PCA-2	PCA-3
gene001	-0.18	-0.11	-0.03
gene002	0.51	-0.53	0.54
gene003	-0.35	-0.39	0.26
gene004	-0.18	-1.08	0.41
gene005	-0.62	-0.8	0.13
gene006	-0.09	-0.23	0.77
gene007	-0.38	-0.32	1.08
gene008	-0.88	-0.55	1.03
gene009	-1.26	0.45	0.41
gene010	0.12	-0.36	-0.16
gene011	-0.28	-0.44	2.13
gene012	-0.45	-0.23	0.82
gene013	-0.2	-0.43	0.44
gene014	0.03	-0.26	-0.68
gene015	-0.7	-0.76	0.5
gene016	-0.61	0.07	-0.04
gene017	-0.23	-0.71	0.01
gene018	0.1	0.1	0.11
gene019	-0.94	-0.97	0.24
gene020	-0.55	-0.53	0.86
gene021	-0.47	-0.87	-0.02
gene022	-0.34	-1.1	0.51
gene***	-0.49	-0.2	0.91
gene n	-0.15	-1.04	-0.01



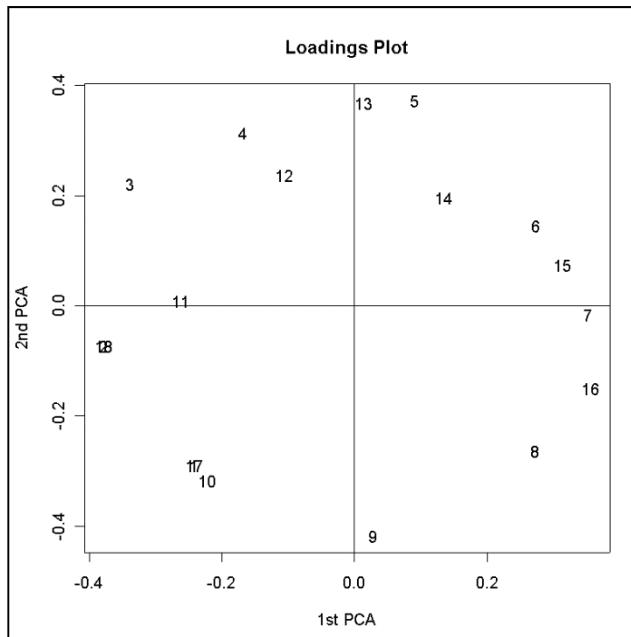


# Loadings Plot

```
# loadings plot
plot(loadings(cell.pca)[,1], loadings(cell.pca)[,2], xlab="1st PCA",
ylab="2nd PCA", main="Loadings Plot", type="n")
text(loadings(cell.pca)[,1], loadings(cell.pca)[,2], labels=paste(1:p))
abline(h=0)
abline(v=0)
```

```
> summary(cell.pca)
Importance of components:
                                         Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.15
Standard deviation     2.3012110 2.0542795 1.3300507 1.00895544 0.90053289 0.308577283
Proportion of Variance 0.3309732 0.2637540 0.1105647 0.06362444 0.05068497 *** 0.005951246
Cumulative Proportion  0.3309732 0.5947272 0.7052919 0.76891637 0.81960134 1.000000000
```

```
# print loadings  
loadings(cell.pca)  
summary(cell.pca)
```



Loadings:	Comp.1	Comp.2	Comp.3	Comp.4
alpha14	-0.283	-0.21	0.283	0.136
alpha21	-0.374	0.211	-0.135	-0.16
alpha28	-0.26	0.298	0.161	-0.168
alpha35	-0.102	0.372	0.165	-0.321
alpha42	0.161	0.355	0.2	-0.317
alpha49	0.287	0.167	0.116	-0.515
alpha56	0.35	0.172	-0.274	-0.115
alpha63	0.251	-0.258	-0.275	-0.37
alpha70	-0.372	-0.217	-0.382	-0.159
alpha77	-0.253	-0.221	-0.321	-0.32
alpha84	-0.249	-0.437	-0.309	-0.256
alpha91	-0.115	0.279	-0.436	0.114
alpha98	0.36	-0.284	0.186	-0.138
alpha105	0.16	0.257	-0.283	-0.125
alpha112	0.347	0.319	-0.178	-0.276
alpha119	0.348	-0.164	-0.201	0.11

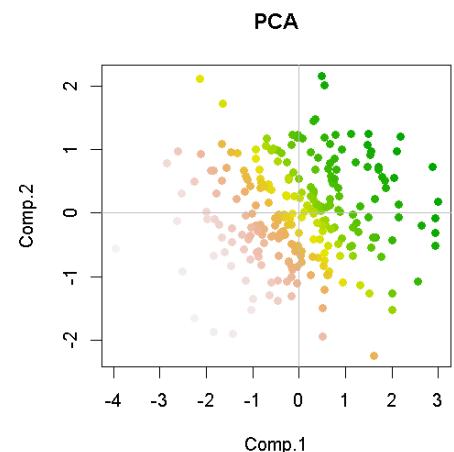
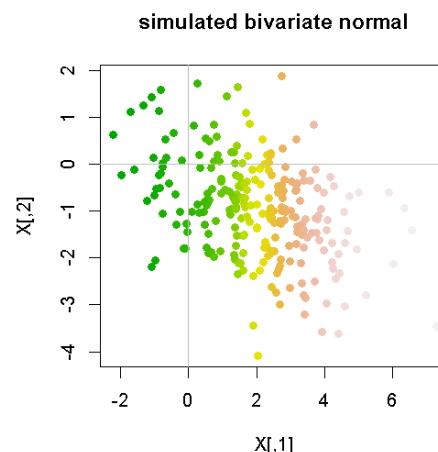


## 範例: two-dimensional simulated data

**Example 2.3** The two-dimensional **simulated data** consist of 250 vectors  $\mathbf{X}_i$  from the bivariate normal distribution with mean  $[2, -1]^\top$  and covariance matrix  $\Sigma$  as

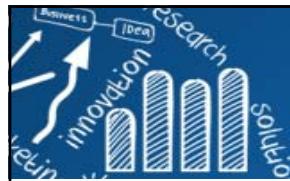
$$\Sigma = \begin{pmatrix} 2.4 & -0.5 \\ -0.5 & 1 \end{pmatrix}.$$

```
> library(MASS)
> mu <- c(2, -1)
> Sigma <- matrix(c(2.4, -0.5, -0.5, 1), 2)
> n <- 250
> X <- mvrnorm(n, mu, Sigma)
>
> mycol <- terrain.colors(n)
> sorted.x1 <- sort(X[,1])
> order.x1 <- order(X[,1])
> id <- 1:n
> sorted.id <- id[order.x1]
> x1.col <- mycol[order(sorted.id)]
>
> par(mfrow=c(1, 2))
> plot(X, col=x1.col, pch=16,
+ main="simulated bivariate normal")
> abline(h=0, v=0, col="gray")
> X.pca <- princomp(X, cor = TRUE)
> X.pca$sdev
  Comp.1    Comp.2
1.1875339 0.7679605
```



```
> X.pca$loadings
Loadings:
  Comp.1  Comp.2
[1,] -0.707 -0.707
[2,]  0.707 -0.707

  Comp.1  Comp.2
SS loadings     1.0   1.0
Proportion Var  0.5   0.5
Cumulative Var 0.5   1.0
> plot(X.pca$scores, col=x1.col, pch=16, main="PCA")
> abline(h=0, v=0, col="gray")
```



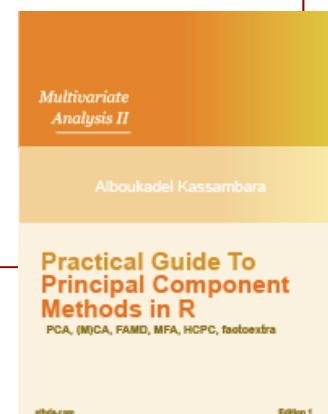
# 範例: **decathlon2 {factoextra}**: Athletes' performance in decathlon

37/144

use **FactoMineR** for the analysis and **factoextra** for ggplot2-based visualization.

```
> pca.pkg <- c("FactoMineR", "factoextra", "corrplot")
> install.packages(pca.pkg)
> lapply(pca.pkg, library, character.only=TRUE)
> data(decathlon2) # 十項全能
> head(decathlon2) # 100米, 跳遠, 鉛球, 跳高, 400米, 110米跨欄, 鐵餅, 撑竿跳高, 標槍, 1500米
   X100m Long.jump Shot.put High.jump X400m X110m.hurdle Discus
SEBRLE    11.04      7.58    14.83     2.07 49.81      14.69    43.75
CLAY       10.76      7.40    14.26     1.86 49.37      14.05    50.72
BERNARD    11.02      7.23    14.25     1.92 48.93      14.99    40.87
YURKOV     11.34      7.09    15.19     2.10 50.42      15.31    46.26
ZSIVOCZKY  11.13      7.30    13.48     2.01 48.62      14.17    45.67
McMULLEN   10.83      7.31    13.76     2.13 49.91      14.38    44.41
   Pole.vault Javeline X1500m Rank Points Competition
SEBRLE      5.02    63.19   291.7      1    8217 Decastar
CLAY        4.92    60.15   301.5      2    8122 Decastar
BERNARD     5.32    62.77   280.1      4    8067 Decastar
YURKOV      4.72    63.44   276.4      5    8036 Decastar
ZSIVOCZKY   4.42    55.37   268.0      7    8004 Decastar
McMULLEN    4.42    56.37   285.1      8    7995 Decastar
> dim(decathlon2)
[1] 27 13
> # standardization # x <- scale(x)
> x <- decathlon2[,1:10]
> # (default, PCA {FactoMineR} standardizes the data automatically)
```

**FactoMineR:** Multivariate Exploratory  
Data Analysis and Data Mining



Practical Guide to Principal Component Methods in R

<http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/>

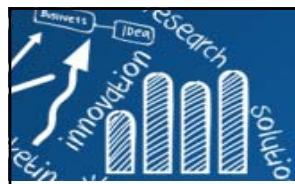


## 範例: PCA {FactoMineR}

```
> # PCA(X, scale.unit = TRUE, ncp = 5, graph = TRUE)
> x.pca <- PCA(x, graph = FALSE)
> class(x.pca)
[1] "PCA"    "list"
> str(x.pca)
List of 5
 $ eig : num [1:10, 1:3] 3.75 1.745 1.518 1.032 0.618 ...
 ...- attr(*, "dimnames")=List of 2
 ...
 ...
 ..$ call      : language PCA(X = x, graph = FALSE)
 - attr(*, "class")= chr [1:2] "PCA" "list"
> print(x.pca)
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 27 individuals, described by 10 variables
*The results are available in the following objects:

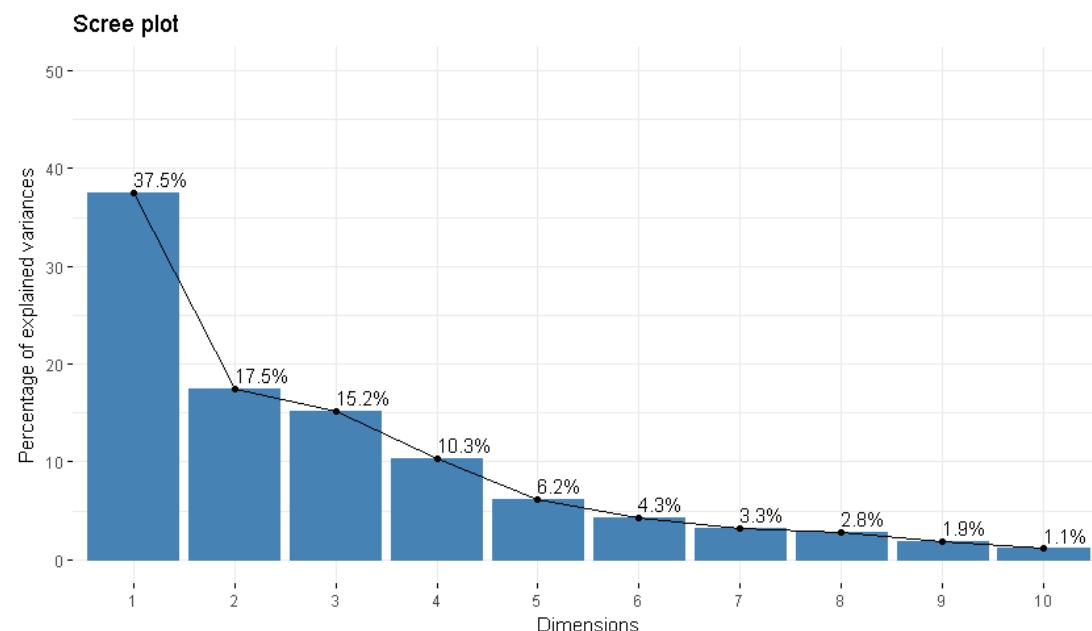
      name           description
1  "$eig"         "eigenvalues"
2  "$var"         "results for the variables"
3  "$var$coord"   "coord. for the variables"
4  "$var$cor"     "correlations variables - dimensions"
...
15  "$call$col.w" "weights for the variables"
```

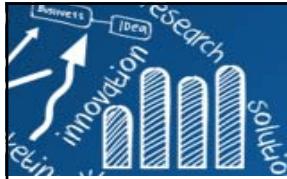
```
# get_eigenvalue(x.pca): Extract the eigenvalues/variances of principal components
# fviz_eig(x.pca): Visualize the eigenvalues
# get_pca_ind(x.pca), get_pca_var(x.pca): Extract the results for individuals and variables.
# fviz_pca_ind(x.pca), fviz_pca_var(x.pca): Visualize the results individuals and variables.
# fviz_pca_biplot(x.pca): Make a biplot of individuals and variables.
```



# Eigenvalues/Variances, scree plot

```
> # Eigenvalues/Variances
> eig.val <- get_eigenvalue(x.pca)
> eig.val
  eigenvalue variance.percent cumulative.variance.percent
Dim.1    3.7499727      37.499727                37.49973
Dim.2    1.7451681      17.451681                54.95141
Dim.3    1.5178280      15.178280                70.12969
Dim.4    1.0322001      10.322001                80.45169
Dim.5    0.6178387      6.178387                 86.63008
...
Dim.10   0.1122959      1.122959                 100.00000
> # scree plot
> fviz_eig(x.pca, addlabels = TRUE, ylim = c(0, 50))
```





# Correlation Circles

```

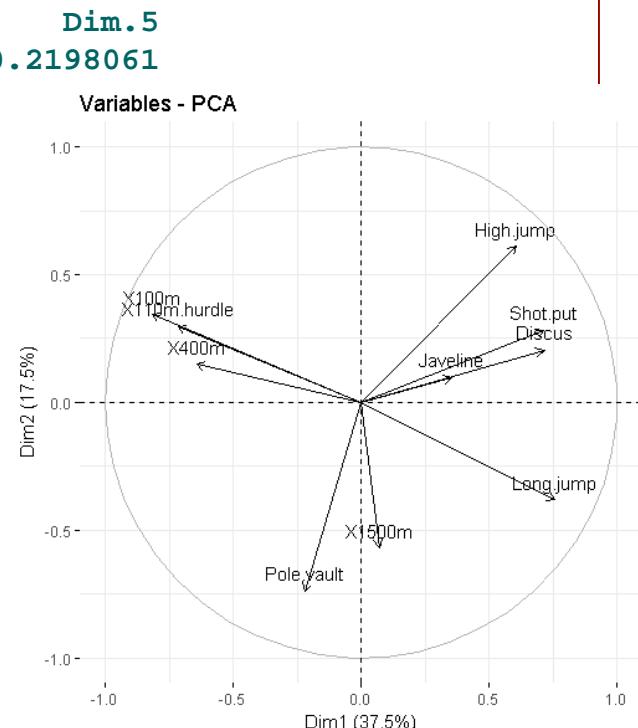
> var <- get_pca_var(x.pca)
> var
Principal Component Analysis Results for variables
=====
  Name      Description
1 "$coord" "Coordinates for the variables"
2 "$cor"    "Correlations between variables and dimensions"
3 "$cos2"   "Cos2 for the variables"
4 "$contrib" "contributions of the variables"
> # Coordinates of variables
> head(var$coord, 4)
          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
x100m -0.8189521 0.3427787 0.100864539 0.10134200 -0.2198061
Long.jump 0.7588985 -0.3814931 -0.006261254 -0.18542415
Shot.put  0.7150783  0.2821167  0.473854591  0.03610404
High.jump 0.6084933  0.6113542  0.004605966  0.07124353
> # Correlation circle, variable correlation plots
> fviz_pca_var(x.pca, col.var = "black")

```

- # Positively correlated variables are grouped together.
- # Negatively correlated variables are positioned on opposite sides of the plot origin (opposed quadrants).
- # The distance between variables and the origin measures the quality of the variables on the factor map. Variables that are away from the origin are well represented on the factor map.

# **var\$cos2**: quality of representation for variables on the factor map.  
 $\text{var.cos2} = \text{var.coord} * \text{var.coord}$ .

# **var\$contrib**: the contributions of the variables to the principal components.  
 $(\text{var.cos2} * 100) / (\text{total cos2 of the component})$ .

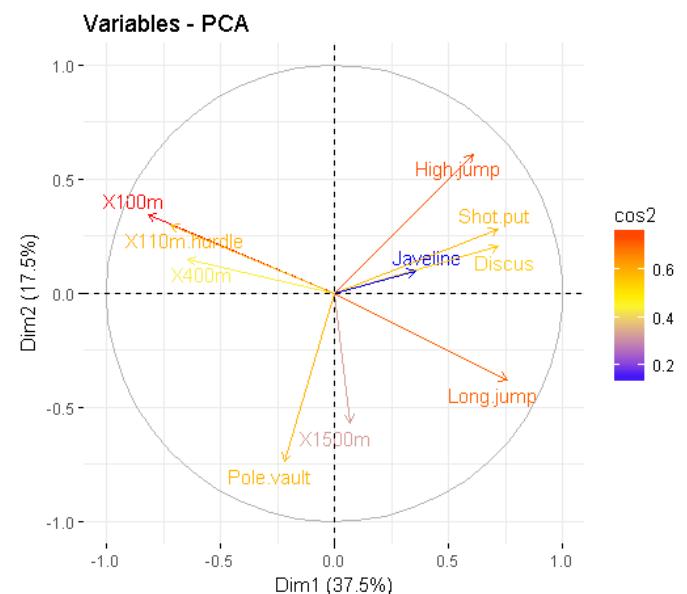
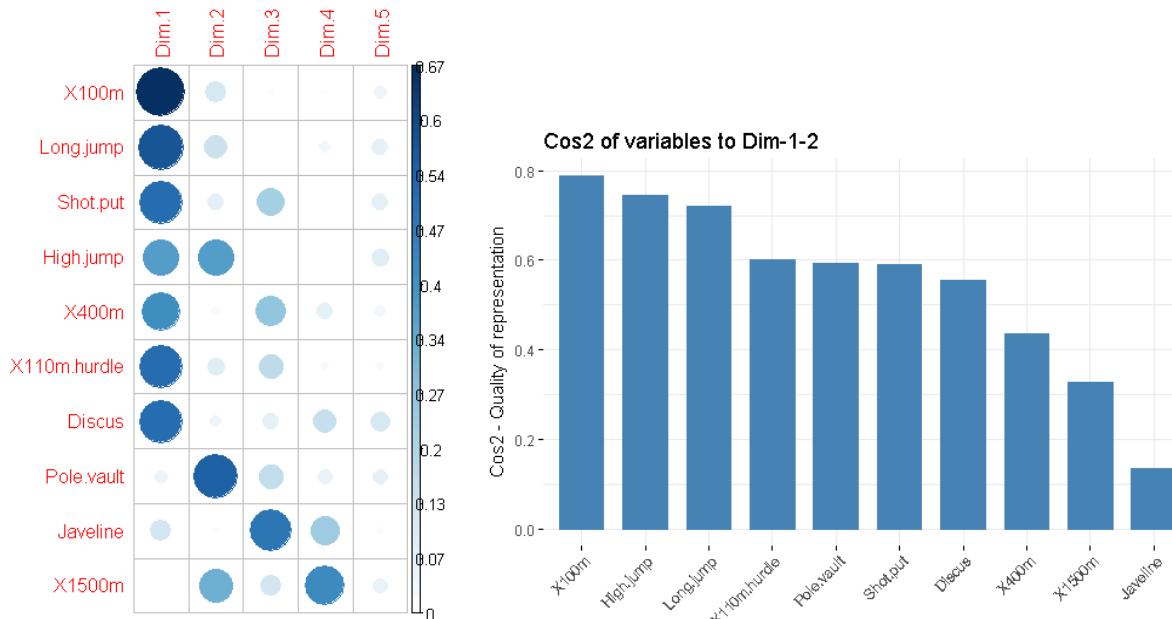


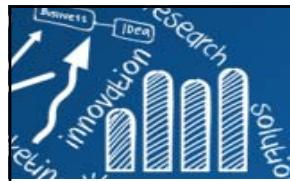


# Quality of representation: square cosine (cos2 values) plot

41/144

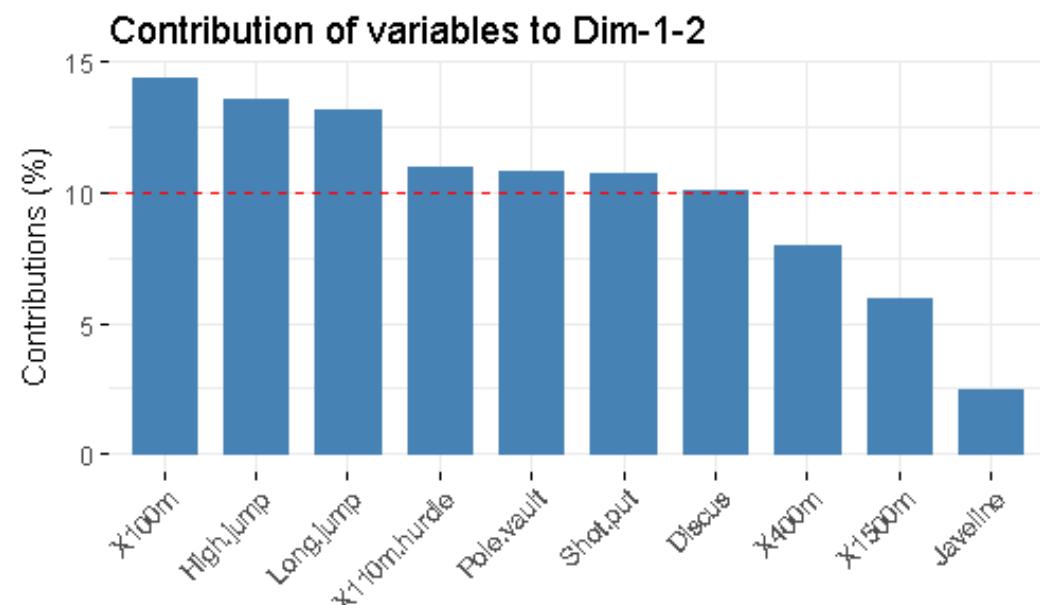
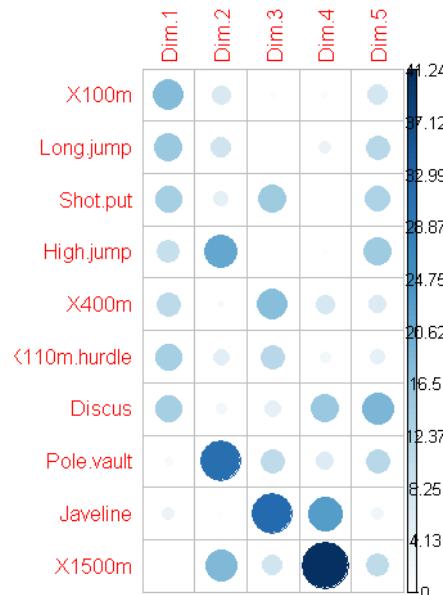
```
> head(var$cos2, 4)
      Dim.1      Dim.2      Dim.3      Dim.4      Dim.5
X100m  0.6706825 0.11749725 1.017366e-02 0.010270201 0.04831474
Long.jump 0.5759270 0.14553701 3.920330e-05 0.034382115 0.06954513
Shot.put  0.5113370 0.07958983 2.245382e-01 0.001303502 0.07763857
High.jump 0.3702640 0.37375400 2.121493e-05 0.005075641 0.09035233
> corrplot(var$cos2, is.corr=FALSE)
> fviz_cos2(x.pca, choice = "var", axes = 1:2)
> # variables with low/mid/high cos2 values will be colored in blue/yellow/red
> fviz_pca_var(x.pca, col.var = "cos2",
+                 gradient.cols = c("blue", "yellow", "red"),
+                 repel = TRUE) # Avoid text overlapping
```

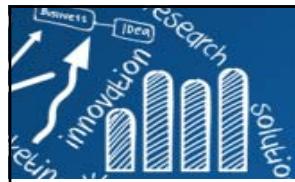




# Contributions of variables to PCs

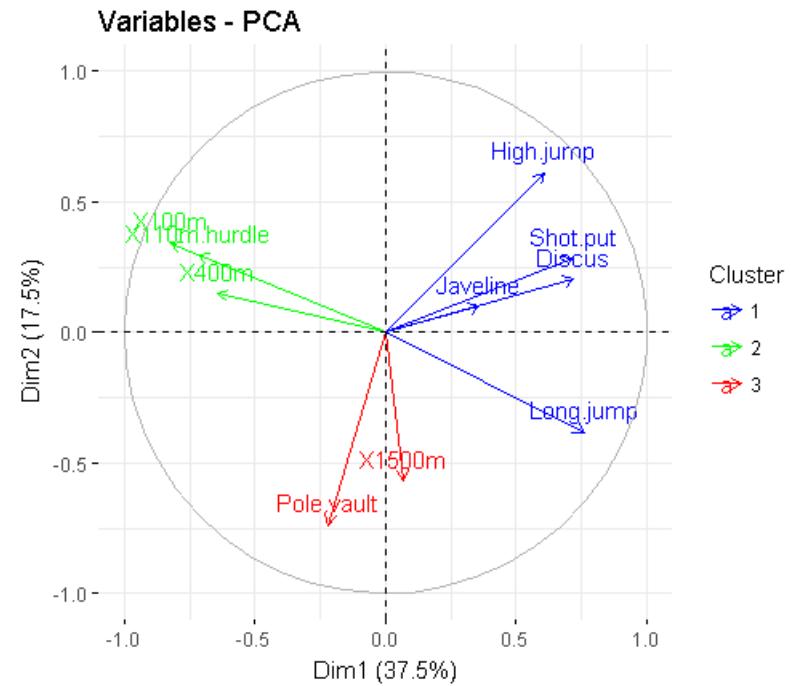
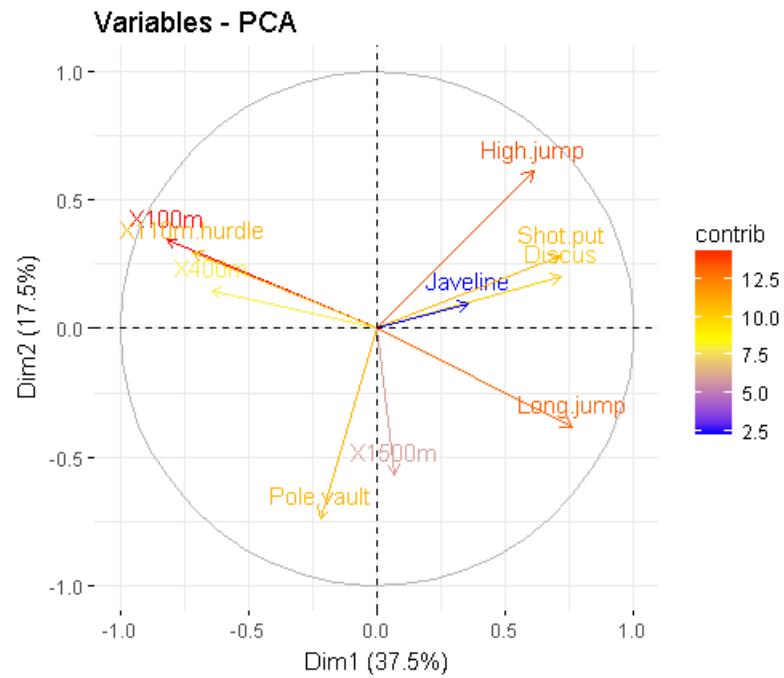
```
> head(var$contrib, 4)
      Dim.1    Dim.2    Dim.3    Dim.4    Dim.5
X100m  17.88500  6.732718  0.670277238  0.9949816  7.81996
Long.jump 15.35817  8.339426  0.002582856  3.3309545 11.25620
Shot.put  13.63575  4.560583 14.793387670  0.1262838 12.56616
High.jump  9.87378 21.416504  0.001397716  0.4917303 14.62394
> corrplot(var$contrib, is.corr=FALSE)
> # Contributions of variables to PC1
> fviz_contrib(x.pca, choice = "var", axes = 1, top = 10)
> # Contributions of variables to PC2
> fviz_contrib(x.pca, choice = "var", axes = 2, top = 10)
> # The total contribution to PC1 and PC2:
> fviz_contrib(x.pca, choice = "var", axes = 1:2, top = 10)
```

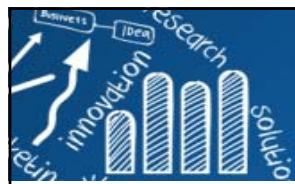




# Highlight the most important variables

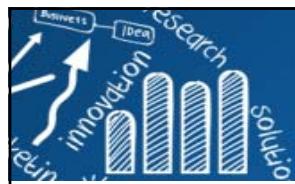
```
> fviz_pca_var(x.pca, col.var = "contrib",
+               gradient.cols = c("blue", "yellow", "red"))
> set.seed(123)
> var.kms <- kmeans(var$coord, centers = 3, nstart = 25)
> kms.grp <- as.factor(var.kms$cluster)
> # Color variables by kmeans' result
> fviz_pca_var(x.pca, col.var = kms.grp, palette = c("blue", "green", "red"),
+               legend.title = "Cluster")
```





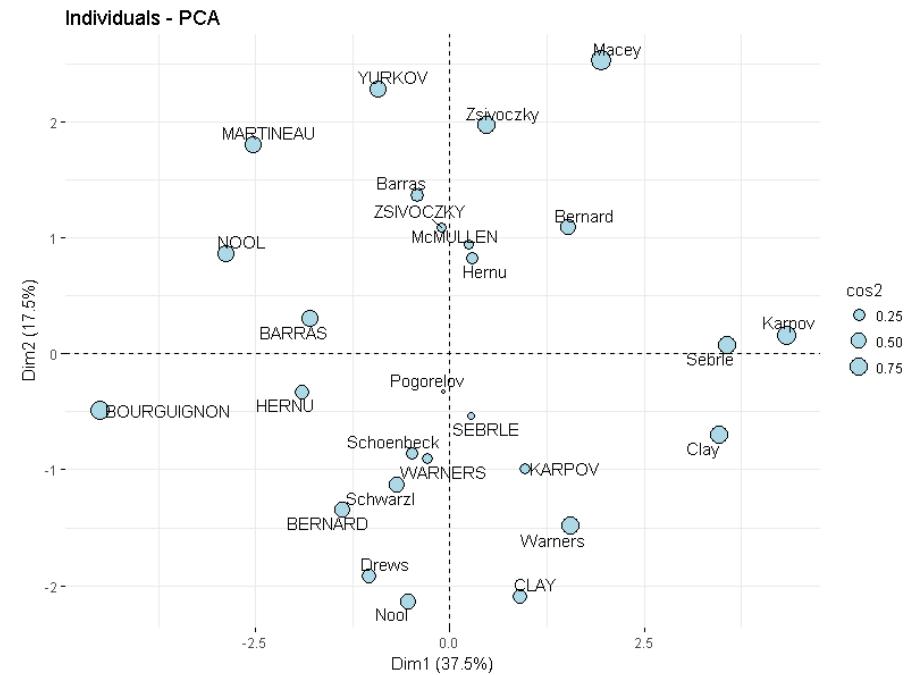
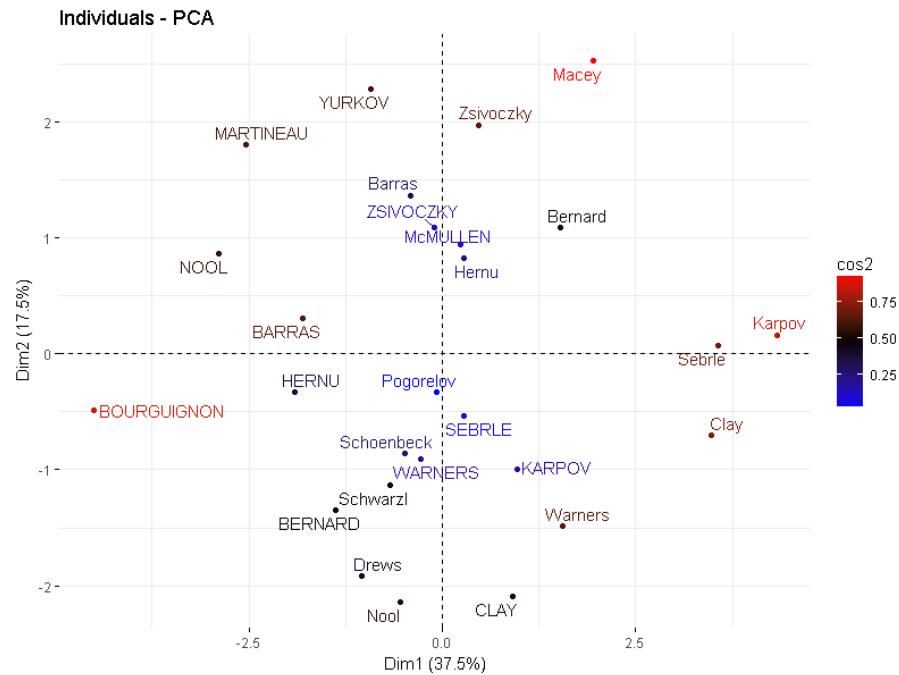
# Obtain PCA results for individuals

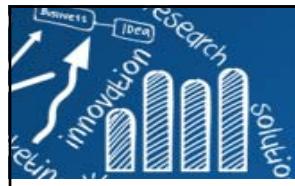
```
> ind <- get_pca_ind(x.pca)
> ind
Principal Component Analysis Results for individuals
=====
  Name      Description
1 "$coord" "Coordinates for the individuals"
2 "$cos2"   "Cos2 for the individuals"
3 "$contrib" "contributions of the individuals"
> # Coordinates of individuals
> head(ind$coord, 3)
          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
SEBRLE    0.2779582 -0.5364345  1.5852390  0.10582254  1.07462350
CLAY      0.9048536 -2.0942803  0.8406848  1.85071783 -0.40864484
BERNARD   -1.3722659 -1.3481155  0.9619317 -1.49307179 -0.18266695
> # Quality of individuals
> head(ind$cos2, 3)
          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
SEBRLE    0.015446507 0.05753138  0.50241310  0.0022388646  0.2308788213
CLAY      0.065574226 0.35127414  0.05660346  0.2743196867  0.0133742243
BERNARD   0.232236641 0.22413434  0.11411498  0.2749258382  0.0041150408
> # Contributions of individuals
> head(ind$contrib, 3)
          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
SEBRLE    0.07630746  0.6107062  6.132015  0.04018174  6.92267277
CLAY      0.80865774  9.3082618  1.724567 12.29002506  1.00104404
BERNARD   1.85987901  3.8570314  2.257887  7.99896372  0.20002354
```



# Graph of individuals

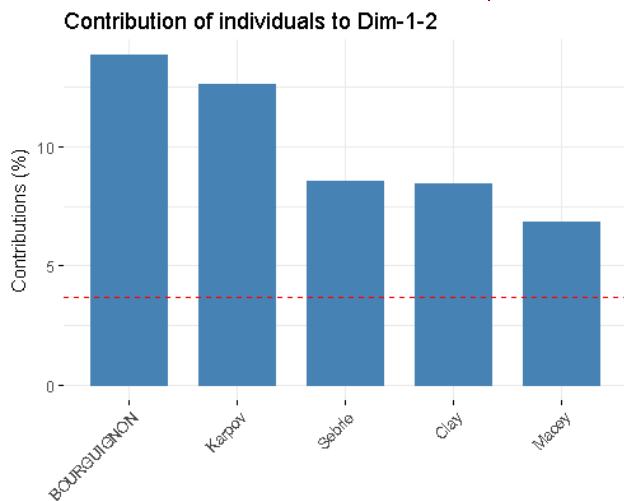
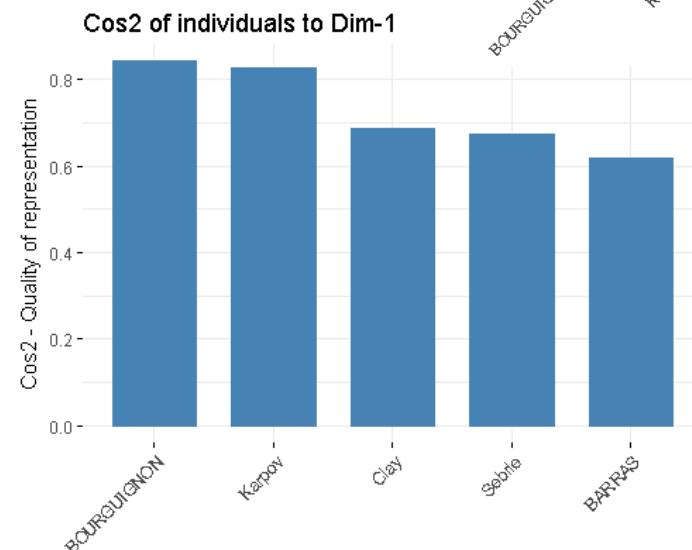
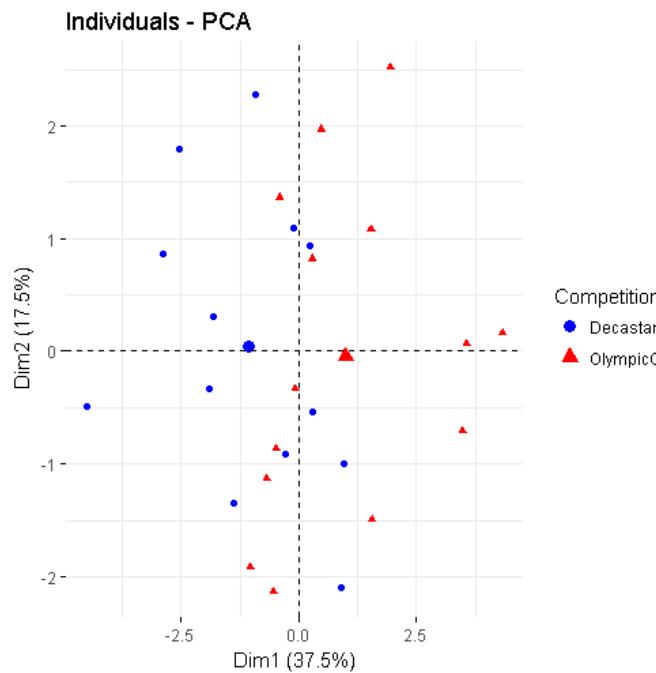
```
> # color individuals by their cos2 values  
> fviz_pca_ind(x.pca, col.ind = "cos2", gradient.cols = c("blue", "black", "red"),  
+                 repel = TRUE)  
>  
> # change the point size according the cos2 of the corresponding individuals  
> fviz_pca_ind(x.pca, pointsize = "cos2", pointshape = 21, fill = "lightblue",  
+                 repel = TRUE)
```





# Graph of individuals

```
> fviz_pca_ind(x.pca, geom.ind = "point", col.ind = decathlon2[,13],  
+                 palette = c("blue", "red"), legend.title = "Competition")  
>  
> # quality of representation (cos2) of individuals on the factor map  
> fviz_cos2(x.pca, choice = "ind", top = 5)  
>  
> # Total contribution of individuals on PC1 and PC2  
> fviz_contrib(x.pca, choice = "ind", axes = 1:2, top = 5)
```



get eigenvectors  
**x.pca\$svd\$V**



## (4) The Biplot

The data matrix can be factored:

$$\mathbf{X} = \mathbf{AB}'$$

$\mathbf{X}_{n \times p}$ : data matrix.

$\mathbf{A}_{n \times k}$ : the coordinates for the  $n$  observations points along  $k$  rectangular axes.

$\mathbf{B}_{p \times k}$ : the coordinates for the  $p$  variables along the same  $k$  axes.

To obtain  $\mathbf{A}$  and  $\mathbf{B}$ , using Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{UDV}'$$

$\mathbf{A}_{[2]}$ : the  $n \times 2$  matrix of biplot coordinates for the observation points.

$\mathbf{B}_{[2]}$ : the  $p \times 2$  matrix of biplot coordinates for the variables.

$$\mathbf{A}_{[2]} = \mathbf{U}_{[2]} \mathbf{D}_{[2]}^c$$

$$\mathbf{B}_{[2]} = \mathbf{V}_{[2]} \mathbf{D}_{[2]}^{1-c}$$

$\mathbf{U}_{[2]}$ : the first two columns of  $\mathbf{U}$ .

$\mathbf{V}_{[2]}$ : the first two columns of  $\mathbf{V}$ .

$\mathbf{D}_{[2]}$ : the diagonal matrix formed by the first two singular values.

$$\mathbf{X}_{[2]} = \mathbf{A}_{[2]} \mathbf{B}_{[2]}'$$

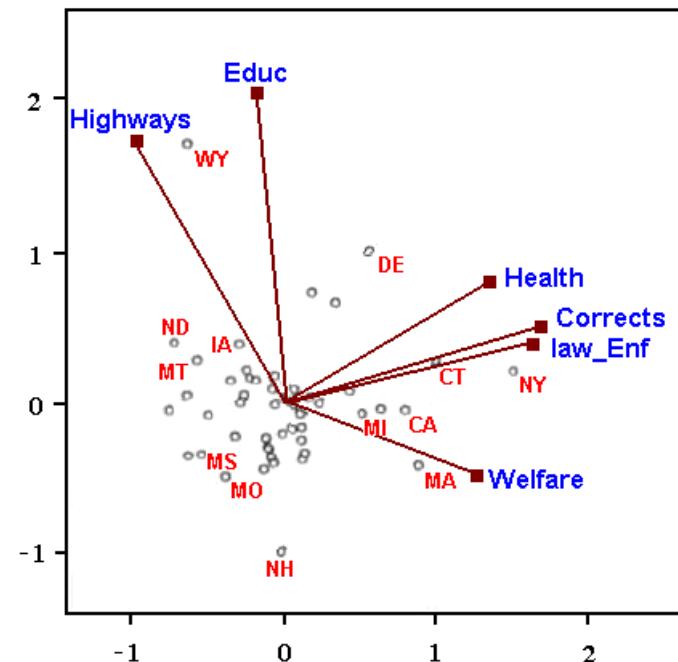
Each row of  $\mathbf{A}_{[2]}$  is plotted as a point in a two-axis coordinate system.

The rows of  $\mathbf{B}_{[2]}$  are also plotted within the same space.

Goodness of fit measure  $R$  ( $s_r$  : singular values)

$$R = \frac{s_1^2 + s_2^2}{\sum_{s=1}^p s_s^2}$$

- K. R. Gabriel (1971). The biplot graphical display of matrices with application to principal component analysis. *Biometrika* 58, 453–467.
- J.C. Gower and D. J. Hand (1996). Biplots. Chapman & Hall.

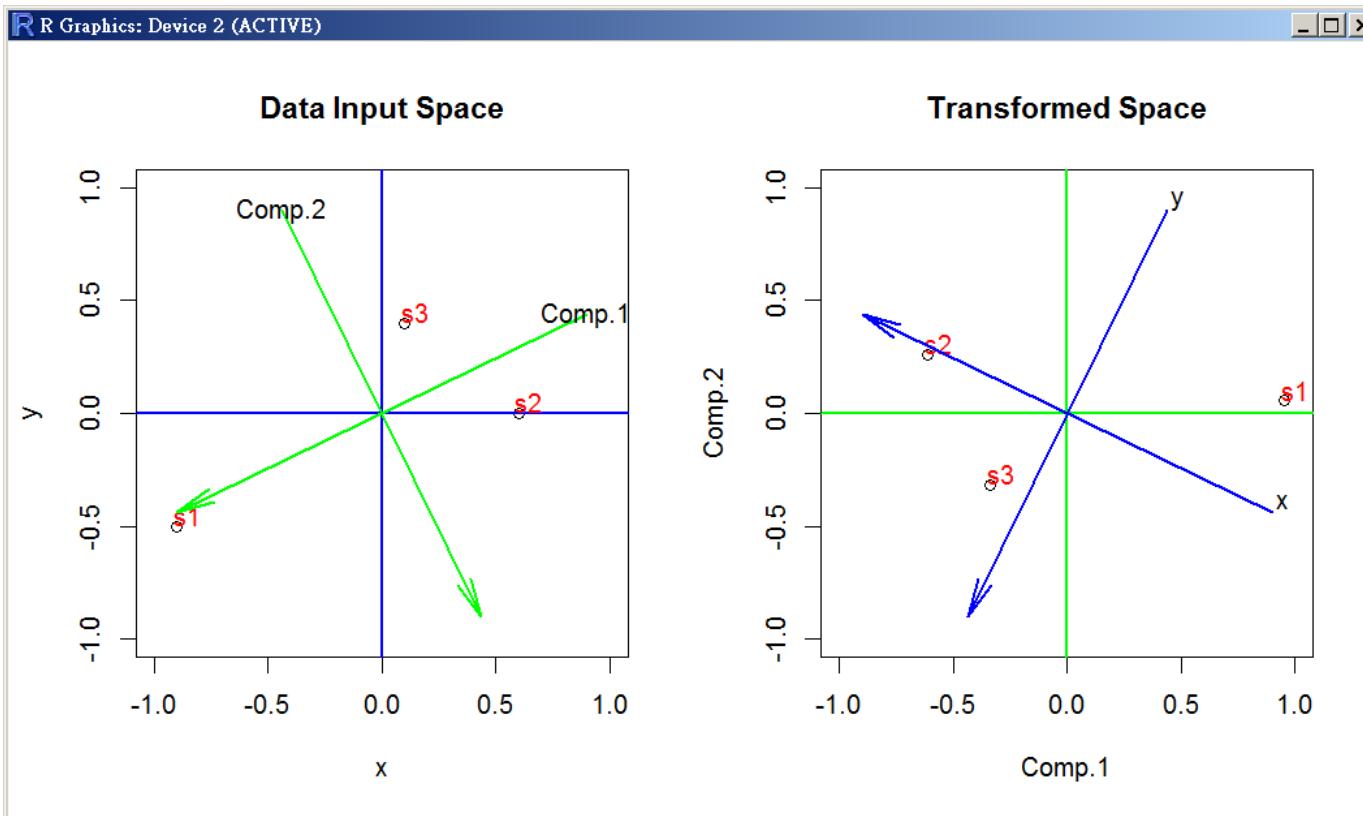


Biplot of 1992 State Policy Spending

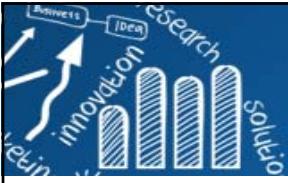


# Moving Towards the Biplot: Rotation of axes

48/144



```
x <- c(-0.9, 0.6, 0.1)
y <- c(-0.5, 0, 0.4)
plot(x, y, xlim=c(-1, 1), ylim=c(-1, 1), main="Data Input Space")
abline(h=0, v=0, col="blue", lwd=2)
text(x+0.05, y+0.05, c("s1", "s2", "s3"), col="red")
pca <- princomp(cbind(x, y)); ab <- pca$loadings
arrows(-ab[1,1], -ab[2,1], ab[1,1], ab[2,1], col="green", angle = 10, lwd=2)
text(-ab[1,1], -ab[2,1], "Comp.1")
arrows(-ab[1,2], -ab[2,2], ab[1,2], ab[2,2], col="green", angle = 10, lwd=2)
text(-ab[1,2], -ab[2,2], "Comp.2")
```



# Graphical Display of Matrix Multiplication

49/144

$$A(4, 2)$$

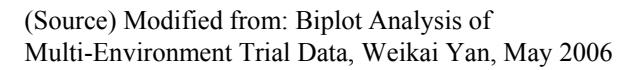
$$B(2, \, 3)$$

$$P(4, \, 3)$$

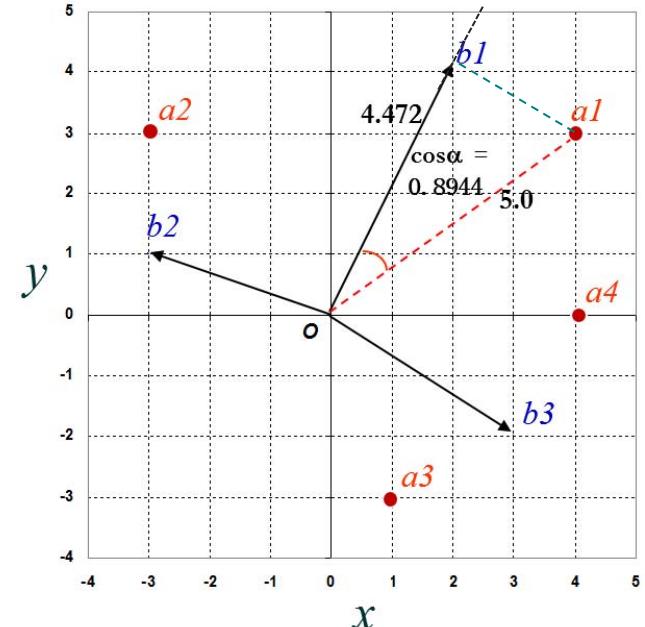
$$\begin{array}{cc} x & y \end{array} \begin{array}{c} b1 \\ a1 \\ a2 \\ a3 \\ a4 \end{array} \begin{array}{c} b2 \\ 4 \\ -3 \\ 1 \\ 4 \end{array} \begin{array}{c} b3 \\ 3 \\ 3 \\ -3 \\ 0 \end{array} \times \begin{array}{cc} x & y \end{array} \begin{array}{c} b1 \\ 2 \\ 4 \\ 1 \end{array} \begin{array}{c} b2 \\ -3 \\ 1 \\ -2 \end{array} \begin{array}{c} b3 \\ 3 \\ -2 \\ -12 \end{array} = \begin{array}{cc} b1 & b2 & b3 \end{array} \begin{array}{c} a1 \\ a2 \\ a3 \\ a4 \end{array} \begin{array}{c} 20 \\ 6 \\ -10 \\ 8 \end{array} \begin{array}{c} -9 \\ 12 \\ -6 \\ -12 \end{array} \begin{array}{c} 6 \\ -15 \\ 9 \\ 12 \end{array}$$

Inner product property:  $\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\| \cdot \|\mathbf{y}\| \cdot \cos(\theta)$

- $P_{ij} = OA_i \times OB_j \times \cos\alpha_{ij}$
  - Implies the product matrix



$$P_{11} = 5 \times 4.472 \times 0.8944 = 20$$



- The *biplot points* can be projected onto the *biplot axes*, and their projections, multiplied by the length of the corresponding biplot vectors, give the scalar products which are equal to the elements of the target matrix.
  - The cosines of the angles between the variables approximate the pairwise correlation coefficients.

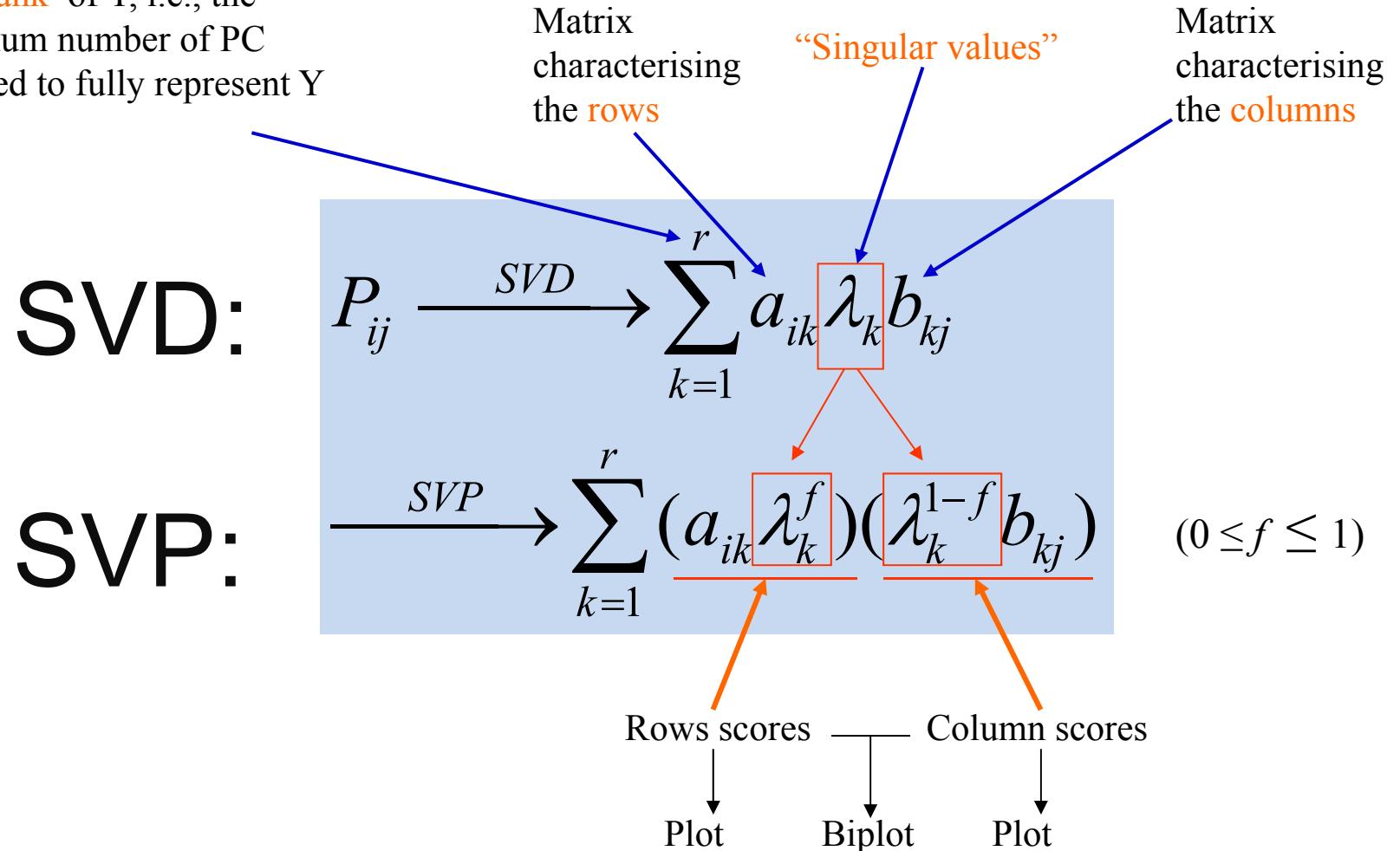
Source: Biplots in Practice, Michael Greenacre.



# Singular Value Decomposition (SVD) & 50/144

## Singular Value Partitioning (SVP)

The ‘rank’ of Y, i.e., the minimum number of PC required to fully represent Y



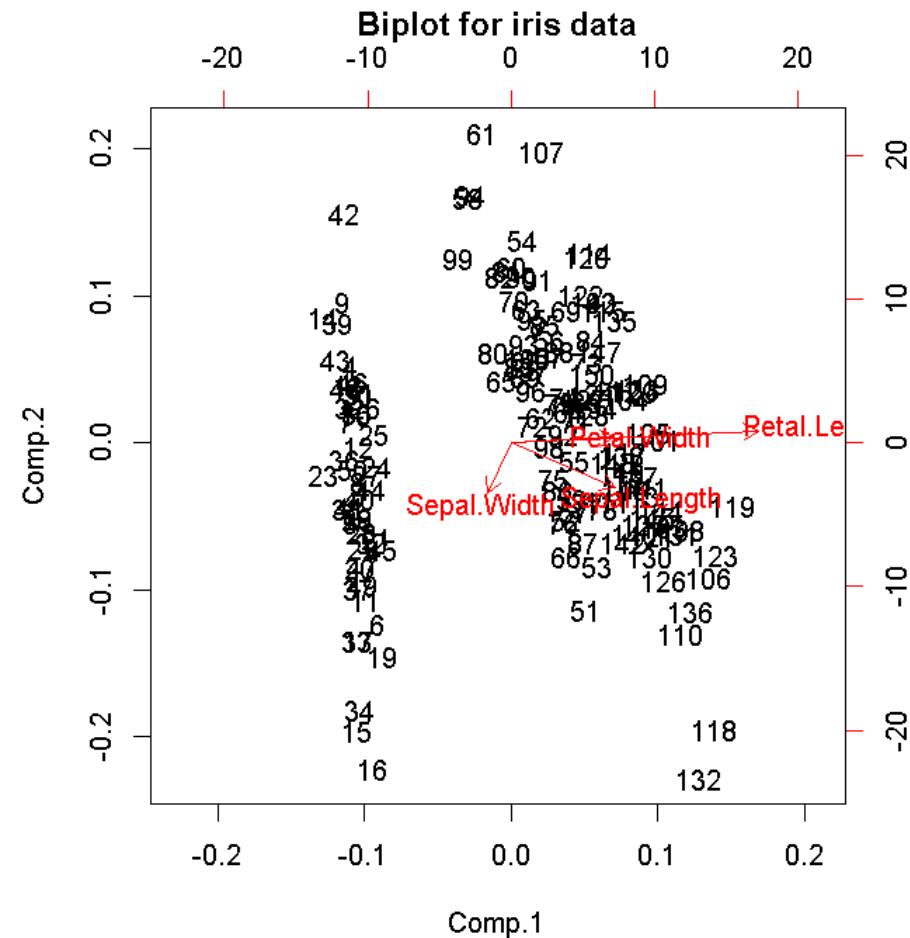
(Source) from: Biplot Analysis of Multi-Environment Trial Data, Weikai Yan, May 2006

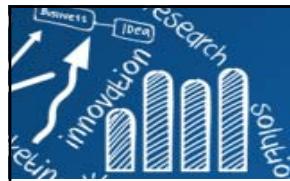


# biplot {stats}: Biplot of Multivariate Data

51/144

```
iris.pca <- princomp(iris[,1:4])
biplot(iris.pca, main="Biplot for iris data")
```



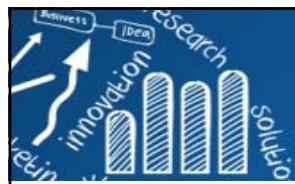


# Interpretation of Biplot

- **Interpreting Points:** the **relative location** of the points can be interpreted.
  - Points that are close together correspond to observations that have similar scores on the components displayed in the plot.
  - To the extent that these components fit the data well, the points also correspond to observations that have similar values on the variables.
- **Interpreting Vectors:** both the **direction** and **length** of the vectors can be interpreted.
  - A vector points in the direction which is most like the variable represented by the vector. This is the direction which has the **highest squared multiple correlation** with the principal components.
  - The length of the vector is proportional to the squared multiple correlation between the fitted values for the variable and the variable itself.
  - *The fitted values for a variable are the result of projecting the points in the space orthogonally onto the variable's vector (to do this, you must imagine extending the vector in both directions).*
  - The observations whose points project furthest in the direction in which the vector points are the observations that have the most of whatever the variable measures. Those points that project at the other end have the least. Those projecting in the middle have an average amount.
  - Thus, vectors that point in the same direction correspond to variables that have similar response profiles, and can be interpreted as having similar meaning in the context set by the data.

Source: Principal Components: BiPlots, Forrest Young's Notes

<http://forrest.psych.unc.edu/research/vista-frames/help/lecturenotes/lecture13/biplot.html>



# Biplot in R

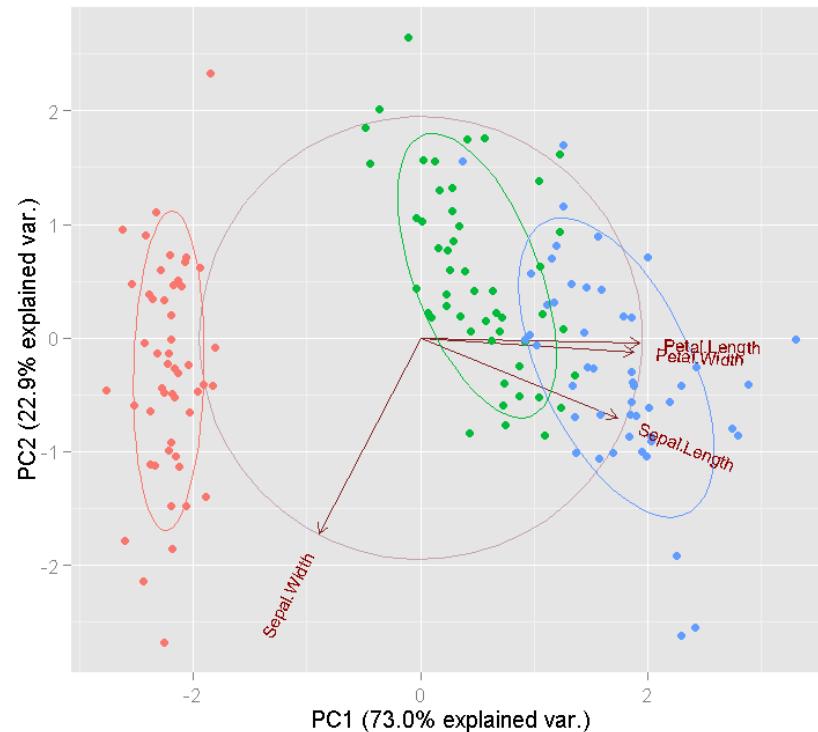
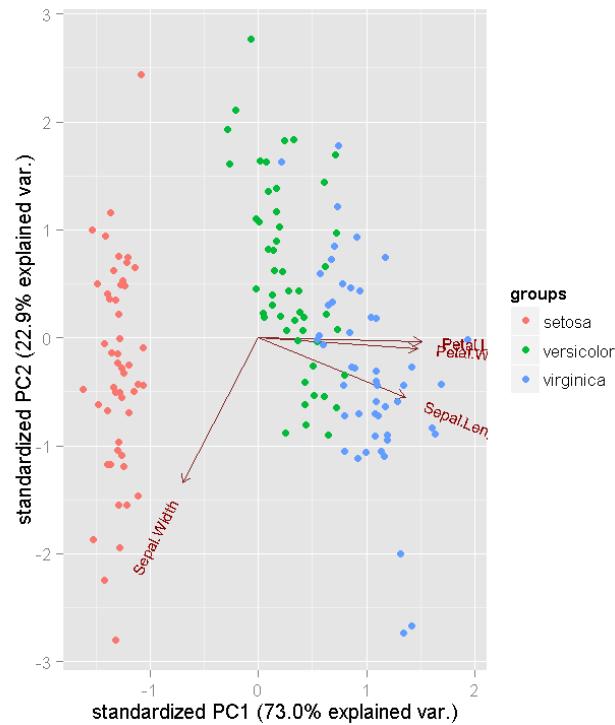
- **biplot {stats}**: Biplot of Multivariate Data  
<https://stat.ethz.ch/R-manual/R-devel/library/stats/html/biplot.html>
- **biplot.princomp {stats}**: Biplot for Principal Components:  
<https://stat.ethz.ch/R-manual/R-devel/library/stats/html/biplot.princomp.html>
- ggbiopt: <https://github.com/vqv/ggbiopt>
- **biplot.rda {vegan}**: <http://cc.oulu.fi/~jarioksa/softhelp/vegan/html/biplot.rda.html>
- **biplot.psych {psych}**: <http://www.personality-project.org/r/html/biplot.psych.html>  
<https://cran.r-project.org/web/packages/psych/>
- **bPCA**: Biplot of Multivariate Data Based on Principal Components Analysis  
<https://cran.r-project.org/web/packages/bPCA/index.html>
- **dynBiplotGUI**: Full Interactive GUI for Dynamic Biplot in R  
<https://cran.r-project.org/web/packages/dynBiplotGUI/index.html>
- The BiplotGUI package homepage: <http://biplotgui.r-forge.r-project.org/>
- **BiplotGUI**: Interactive Biplots in R: <https://cran.r-project.org/web/packages/BiplotGUI/index.html>
- **biplotbootGUI**: Bootstrap on Classical Biplots and Clustering Disjoint Biplot:  
<https://cran.r-project.org/web/packages/biplotbootGUI/index.html>
- **GGEbiplotGUI**: GGEbiplotGUI: Interactive GGE Biplots in R:  
<https://cran.r-project.org/web/packages/GGEbiplotGUI/index.html>
- **multibiplotGUI**: Multibiplot Analysis in R:  
<https://cran.r-project.org/web/packages/multibiplotGUI/index.html>
- **MultBiplot** In R: <http://biplot.dep.usal.es/classicalbiplot/multibiplot-in-r/>
- **NominalLogisticBiplot**: Biplot representations of categorical data  
<https://cran.r-project.org/web/packages/NominalLogisticBiplot/index.html>
- **OrdinalLogisticBiplot**: Biplot representations of ordinal variables  
<https://cran.r-project.org/web/packages/OrdinalLogisticBiplot/index.html>
- **PLSbiplot1**: The Partial Least Squares (PLS) Biplot  
<https://cran.r-project.org/web/packages/PLSbiplot1/index.html>
- **PCABiplot**: Principal Components Analysis Biplots  
<http://biplot.usal.es/ClassicalBiplot/multibiplot-in-r/pcabiplot-manual.pdf>

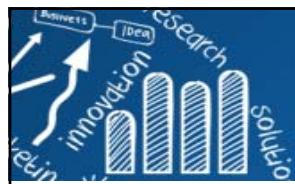
# ggbiplots for Iris Data

```

> library(devtools)
> install_github("vqv/ggbiplots")
>
> library(ggbiplots)
> iris.prcomp <- prcomp(iris[,1:4], scale. = TRUE)
> ggbiplots(iris.prcomp, groups = iris[,5])
> ggbiplots(iris.prcomp, obs.scale = 1, var.scale = 1,
+   groups = iris[,5], ellipse = TRUE, circle = TRUE) +
+   scale_color_discrete(name = '') +
+   theme(legend.direction = 'horizontal', legend.position = 'top')

```

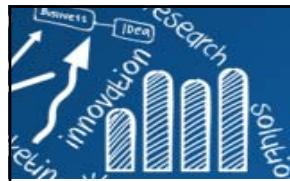




# Creating A Biplot Using SVD (1/3)

```
> library(lattice)
> state.spending <- read.table("StatePolicySpending.txt", header = T, row.names=1)
> head(state.spending)
   Education Welfare Highways HealthCare Corrections LawEnforcement
AL     0.863   0.448    0.168      0.284       0.044        0.183
AR     0.887   0.492    0.241      0.150       0.044        0.141
AZ     0.765   0.473    0.249      0.112       0.090        0.364
CA     0.916   0.817    0.142      0.170       0.104        0.425
CO     0.782   0.415    0.188      0.108       0.076        0.282
CT     0.764   0.742    0.251      0.323       0.128        0.347
> spend <- scale(state.spending)
> spend.svd <- svd(spend, nu=2, nv=2)
>
> D <- spend.svd$d[1:2]
> V <- spend.svd$v[,1:2]
> U <- spend.svd$u[,1:2]
>
> # Create matrices for variables and observations by weighting singular vectors with
> # the square roots of the first two singular values. These will be used to construct
> # a symmetric biplot.
>
> spend.var <- V * (rep(1, length(V[,1])) %*% t(D^.5))
> spend.obs <- U * (rep(1, length(U[,1])) %*% t(D^.5))
> row.names(spend.var) <- colnames(spend)
> row.names(spend.obs) <- row.names(spend)
```

Source: Measurement, Scaling, and Dimensional Analysis, Bill Jacoby Summer 2015



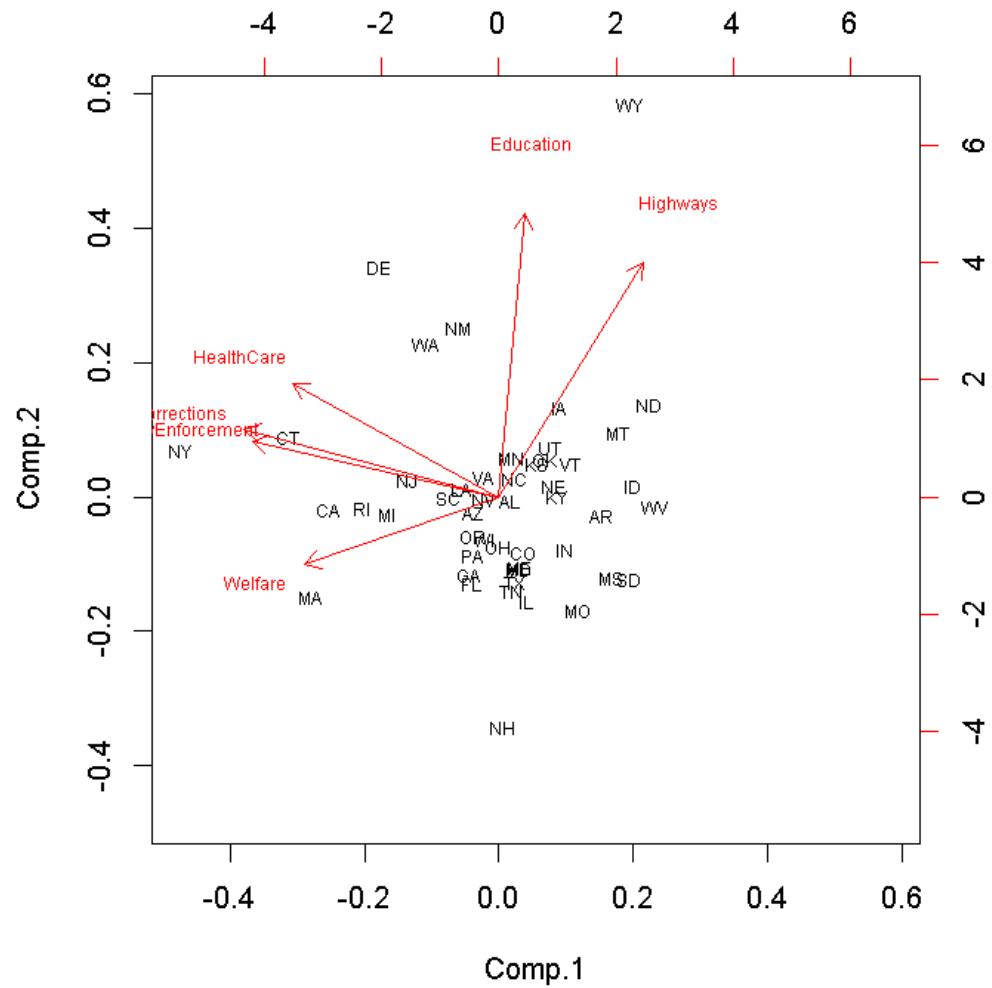
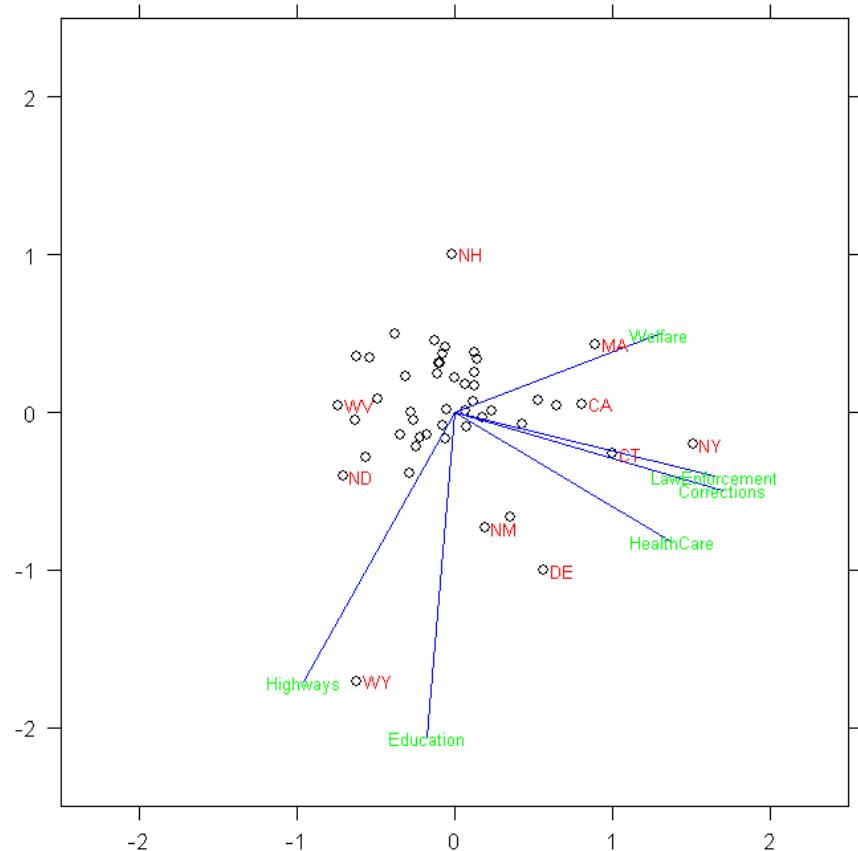
# Creating A Biplot Using SVD (2/3)

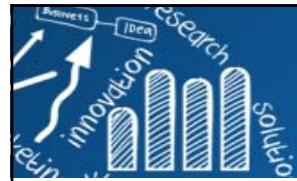
```
> # Within the panel function, "panel.xyplot" draws the observation points and
> # "panel.segments" draws the variable vectors. The first "panel.text" labels the vectors,
> # and the second "panel.text" provides labels for relatively extreme observation points.
> xyplot(spend.obs[,2] ~ spend.obs[,1],
+         aspect = 1,
+         panel = function(x, y) {
+             panel.xyplot(x, y, col = "black")
+             panel.segments(rep(0, length(spend.var[,1])),
+                            rep(0, length(spend.var[,1])),
+                            spend.var[,1], spend.var[,2], lty = 1, col = "blue")
+             panel.text(spend.var[,1], spend.var[,2],
+                        row.names(spend.var), cex = .7, col = "green")
+             panel.text(x[abs(x)>.7 | abs(y)>.7]+.04, y[abs(x)>.7 | abs(y)>.7],
+                        row.names(spend.obs)[abs(x)>.7 | abs(y)>.7], cex = .7,
+                        adj = 0, col= "red")
+         },
+         xlim = c(-2.5, 2.5), ylim = c(-2.5, 2.5),
+         xlab = " ", ylab = " ")
>
> # Calculate proportion of variance explained by
> # first two pairs of singular vectors
> var.explained <- sum(D^2)/sum(spend.svd$d^2)
> var.explained
[1] 0.6233372
```



# Creating A Biplot Using SVD (3/3)

```
biplot(princomp(scale(state.spending)), cex=0.6)
```

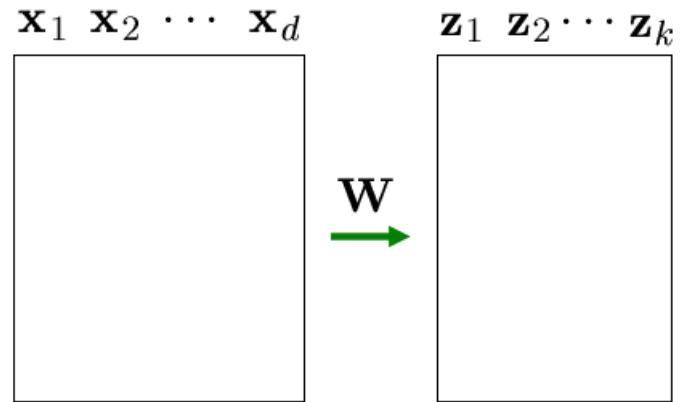




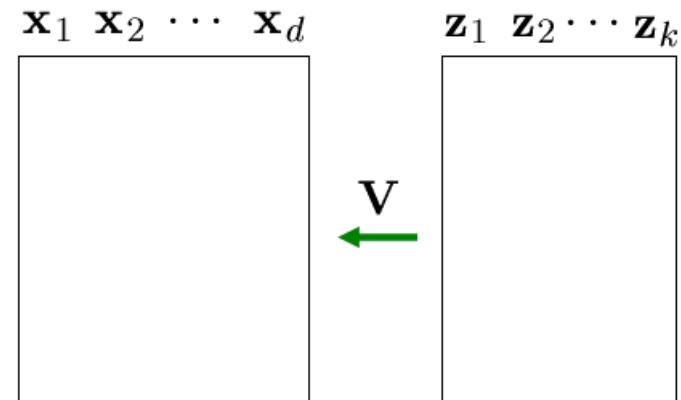
## (5) Factor Analysis (1/4)

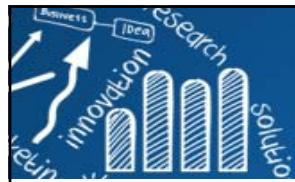
- There is a set of unobservable, latent factors  $z_j, j=1,\dots,k$ , which when acting in combination generate  $x$ .
- **Goal:** to characterize the dependency among the observed variables by means of a smaller number of factors (groupings).
- A few factors can represent the groups of variables.
- FA always partitions the variables into factor clusters.

**PCA**  $z = W^T(x - \mu)$



**FA**  $x - \mu = Vz + e$





## Factor Analysis (2/4)

- FA assumes that each input dimension can be written as a weighted sum of the ***k* factors**, plus the residual term.

$$x_i - \mu_i = \sum_{j=1}^k v_{ij} z_j + \epsilon_i, \quad i = 1, \dots, d \quad \rightarrow \quad \text{Var}(x_i) = \sum_{j=1}^k v_{ij}^2 + \psi_i$$

**Factor loadings**  
↓  
**Model:**  $\mathbf{x} - \boldsymbol{\mu} = \mathbf{V}\mathbf{z} + \mathbf{E}$   
 $[d \times 1] \quad [d \times k] \quad [k \times 1] \quad [d \times 1]$

The variance explained by the common factor and unique factor

### Sample

$$\mathbf{X} = \{\mathbf{x}^t\}_{t=1}^N$$

$$E[\mathbf{x}^t] = \boldsymbol{\mu}$$

$$\text{Cov}[\mathbf{x}^t] = \boldsymbol{\Sigma}$$

### Common Factors

$$E[z_i] = 0, \quad i = 1, \dots, k$$

$$\text{Var}(z_i) = 1$$

$$\text{Cov}(z_i, z_j) = 0, \quad i \neq j$$

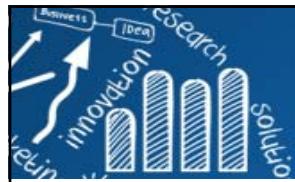
### Unique Factors

$$E[\epsilon_i] = 0, \quad i = 1, \dots, k$$

$$\text{Var}(\epsilon_i) = \psi_i$$

$$\text{Cov}(\epsilon_i, \epsilon_j) = 0, \quad i \neq j$$

$$\text{Cov}(\epsilon_i, z_j) = 0, \quad \forall i, j$$



## Factor Analysis (3/4)

$$\mathbf{x} - \mu = \mathbf{V}\mathbf{z} + \mathbf{E}$$

$$\Sigma = \text{Cov}(\mathbf{x})$$

$$= \text{Cov}(\mathbf{V}\mathbf{z} + \mathbf{E})$$

$$= \text{Cov}(\mathbf{V}\mathbf{z}) + \text{Cov}(\mathbf{E})$$

$$= \mathbf{V}\text{Cov}(\mathbf{z})\mathbf{V}^T + \Psi$$

$$= \mathbf{V}\mathbf{V}^T + \Psi$$

$$\Psi = \text{diag}[\psi_i]_{i=1}^d$$

### Variance Component:

$$x_1 = v_{11} z_1 + v_{12} z_2 + \epsilon_1$$

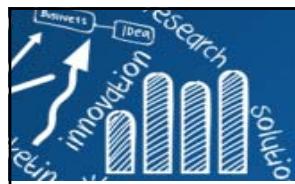
$$\text{Cov}(x_1, z_2) = \text{Cov}(v_{12}z_2, z_2) = v_{12}\text{Var}(z_2) = v_{12} \quad \rightarrow \quad \text{Cov}(\mathbf{x}, \mathbf{z}) = \mathbf{V}$$

The loadings represent the correlations of variables with the factors.

**Example:** two factors

$$\text{Cov}(x_1, x_2) = v_{11}v_{21} + v_{12}v_{22}$$

- If  $x_1$  and  $x_2$  have **high covariance**, then they are related through **a factor**.
- 1st factor  $\Rightarrow v_{11}, v_{21}$  high
- 2nd factor  $\Rightarrow v_{12}, v_{22}$  high
- In either case, the sum will be high.
- If  $x_1$  and  $x_2$  have **low covariance**, then they depend on **different factors** and in the products in the sum.
- One term will be high, the other will be low, the sum will be low.



## Factor Analysis (4/4)

**Model:**  $\mathbf{x} - \mu = \mathbf{V}\mathbf{z} + \mathbf{E}$

**Variance Component:**  $\Sigma = \mathbf{V}\mathbf{V}^T + \Psi \rightarrow \Psi = \text{diag}[\psi_i]_{i=1}^d$

$$\mathbf{S} = \mathbf{V}\mathbf{V}^T + \Psi$$



To find  $\mathbf{V} = [d \times k]$

$\Psi = \psi \mathbf{I}$  Probabilistic PCA

$\Psi = 0 \mathbf{I}$  Conventional PCA

- If there are only a few factors ( $V$  has few columns), then we have a simplified structure for  $S$ .
- Finding factor loadings and the specific variances.

$$\mathbf{S} = \mathbf{C}\mathbf{D}\mathbf{C}^T = \mathbf{C}\mathbf{D}^{1/2}\mathbf{D}^{1/2}\mathbf{C}^T = (\mathbf{C}\mathbf{D}^{1/2})(\mathbf{C}\mathbf{D}^{1/2})^T$$

$$\text{Var}(x_i) = \sum_{j=1}^k v_{ij}^2 + \psi_i$$

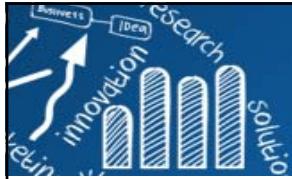


$$\psi_i = s_i^2 - \sum_{j=1}^k v_{ij}^2$$

$$\mathbf{C} = [d \times k]$$

$$\mathbf{D} = [k \times k]$$

$$\mathbf{V} = \mathbf{C}\mathbf{D}^{1/2}$$



## Two Uses of Factor Analysis

- It can be used for knowledge extraction when we find the **loadings** and try to express the variables using fewer **factors**.
- It can also be used for dimension reduction when  $k < d$ .
- Find the **factor scores**  $\mathbf{z}_j$  from  $\mathbf{x}_i$ .
- Find the **loadings**  $\mathbf{w}_{ii}$  such that

$$z_j = \sum_{j=1}^d w_{ji} x_i + \epsilon_i, \quad j = 1, \dots, k$$

$$\mathbf{z}^t = \mathbf{W}^T \mathbf{x}^t + \mathbf{e}, \quad \forall t = 1, \dots, N$$

$$x_i - \mu_i = \sum_{j=1}^k v_{ij} z_j + \epsilon_i, \quad i = 1, \dots, d$$

**NOTE:** For dimension reduction, FA offers no advantage over PCA except the **interpretability** of factors allowing the identification of **common causes**, a simple explanation, and knowledge extraction.

$$\mathbf{Z} = \mathbf{X}\mathbf{W} + \mathbf{E} \quad \rightarrow \quad \mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Z}$$

$$\mathbf{Z} = [N \times k]$$

$$\mathbf{W} = (N-1)(\mathbf{X}^T \mathbf{X})^{-1} \frac{\mathbf{X}^T \mathbf{Z}}{N-1}$$

$$\mathbf{X} = [N \times d]$$

$$\mathbf{E} = [N \times k]$$

$$\mathbf{W} = \left( \frac{\mathbf{X}^T \mathbf{X}}{N-1} \right)^{-1} \frac{\mathbf{X}^T \mathbf{Z}}{N-1} \quad \text{Cov}(\mathbf{x}, \mathbf{z}) = \mathbf{V}$$

$$\mathbf{W} = \mathbf{S}^{-1} \mathbf{V} \quad \rightarrow \quad \widehat{\mathbf{Z}} = \mathbf{X}\mathbf{W} = \mathbf{X}\mathbf{S}^{-1} \mathbf{V}$$



## FA: Rotation

- When  $\mathbf{V}$  is multiplied with any orthogonal matrix ( $\mathbf{T}^T = \mathbf{I}$ ), the solution is not unique.

$$\mathbf{S} = \mathbf{V}\mathbf{V}^T + \Psi \quad \longrightarrow \quad \mathbf{S} = (\mathbf{V}\mathbf{T})(\mathbf{V}\mathbf{T})^T = \mathbf{V}\mathbf{T}\mathbf{T}^T\mathbf{V}^T = \mathbf{V}\mathbf{I}\mathbf{V}^T = \mathbf{V}\mathbf{V}^T$$

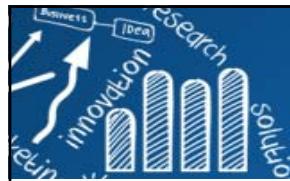
- If  $\mathbf{T}$  is an orthogonal matrix, the distance to the origin does not change.

$$\mathbf{z}^T \mathbf{z} = (\mathbf{T}\mathbf{x})^T (\mathbf{T}\mathbf{x}) = \mathbf{x}^T \mathbf{x}$$

- Multiplying with an orthogonal matrix has the effect of rotating the axes which allows us to choose the set of axes most interpretable.

$$\mathbf{T} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \quad \text{Rotates the axes by phi}$$

- Orthogonal rotation:** the factors are still orthogonal after the rotation.
  - ◆ **Varimax:** tries to maximize variance of squared loadings for each factor (orthogonal):
    - lines up factors with original variables.
    - improves interpretability of factors.
  - ◆ **Quartimax:** maximizes the variance of the squared loadings within the variables.
- Oblique rotation:** the factors are allowed to become correlated.



## R: factanal

```
data(iris3)
Iris <- data.frame(rbind(iris3[,1], iris3[,2], iris3[,3]),
Species = rep(c("setosa","versicolor","virginica"), rep(50,3)))

## FA
data <- Iris[,1:4]
class <- Iris[,5]
iris.fa <- factanal(data, factors=1)
fa.dim1 <- as.matrix(data) %*% iris.fa$loadings[,1]
```

```
> library(stats)
> help(factanal)
```

```
Call:
factanal(x = data, factors = 1)

Uniquenesses:
Sepal.L. Sepal.W. Petal.L. Petal.W.
 0.240    0.822    0.005    0.069

Loadings:
          Factor1
Sepal.L.  0.872
Sepal.W. -0.422
Petal.L.  0.998
Petal.W.  0.965

          Factor1
SS loadings   2.864
Proportion Var  0.716

Test of the hypothesis that 1 factor is sufficient.
The chi square statistic is 85.51 on 2 degrees of freedom.
The p-value is 2.7e-19
```

```
> factanal(data, factors=2)
Error in factanal(data, factors = 2) : 2 factors is too many for 4 variables
```



# (6) Multidimensional Scaling (MDS)

(Torgerson 1952; Cox and Cox 2001)

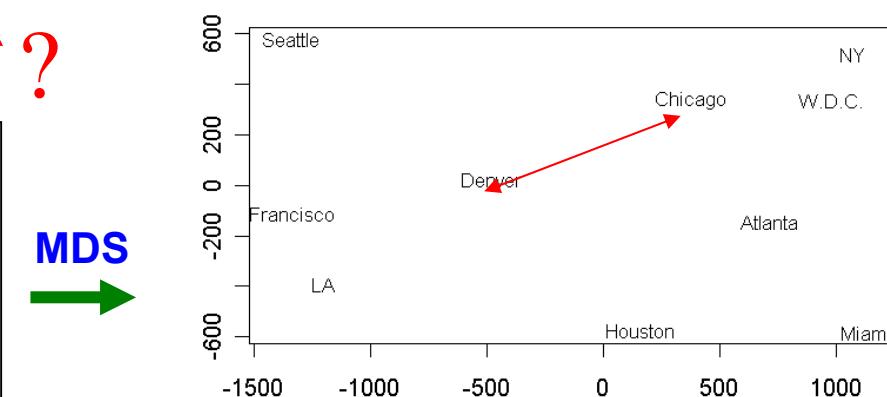
65/144



[http://www.lib.utexas.edu/maps/united\\_states.html](http://www.lib.utexas.edu/maps/united_states.html)

Flying Mileages Between Ten U.S. Cities

	Atlanta	Chicago	Denver	Houston	Los Angeles	Miami	New York	San Francisco	Seattle	Washington D.C.
Seattle	0	587	920	0	1212	79	0	701	940	1936
NY	587	0	920	0	1212	79	0	701	940	1936
W.D.C.	587	920	0	1212	79	0	701	940	1936	0
Atlanta	1745	831	1374	0	1188	1726	968	2339	0	1737
Chicago	1936	0	1745	831	1374	0	1188	1726	968	2339
Denver	701	940	0	1212	79	0	587	920	1745	1936
Houston	748	1631	1420	2451	1092	0	713	1631	1420	2451
Los Angeles	2139	1858	949	1645	347	2594	2571	0	1891	959
Miami	2182	1737	1021	1891	959	2734	2408	678	0	2329
New York	543	597	1494	1220	2300	923	205	2442	2329	0
San Francisco	543	597	1494	1220	2300	923	205	2442	2329	0
Seattle	543	597	1494	1220	2300	923	205	2442	2329	0
Washington D.C.	543	597	1494	1220	2300	923	205	2442	2329	0



MDS



# MDS: Metric and Non-Metric Scaling

- Given the distance between pairs of points, we **don't know** the exact coordinates of the points, or their dimensionality, or how the distances are calculated.
- MDS is the method for placing these points in a low space such that the **Euclidean distance** between them in the two-dimensional space is as close as possible to the given distances in the original space.

**Question:** Given a *dissimilarity matrix* D of certain objects, can we construct points in k-dimensional (often 2-dimensional) space such that

**Goal of metric scaling**

the Euclidean distances between these points approximate the entries in the dissimilarity matrix?

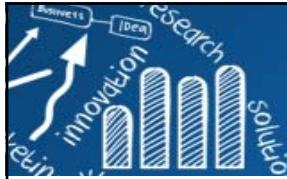
$$S = \sum_{i,j} (\hat{d}_{ij} - d_{ij})^2$$

**Goal of non-metric scaling**

the order in distances coincides with the order in the entries of the dissimilarity matrix approximately?

$$\text{Stress} = \sqrt{\frac{\sum_{i,j} (\hat{d}_{ij} - d_{ij})^2}{\sum_{i,j} d_{ij}^2}}$$

Mathematically: for given  $k$ , compute points  $x_1, \dots, x_n$  in  $k$ -dimensional space such that the object function is minimized.



## MDS: General Methodology (1/3)

**Constraint:** center the data at the origin and assume

$$\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^N \quad \mathbf{x} \in \mathcal{R}^d$$

$$\sum_{i=1}^N x_{ij} = 0, \quad \forall j = 1, \dots, d$$

$$\begin{aligned} d_{rs}^2 &= \| \mathbf{x}_r - \mathbf{x}_s \|^2 \\ &= \sum_{j=1}^d (x_{rj} - x_{sj})^2 \\ &= \sum_{j=1}^d x_{rj}^2 + 2 \sum_{j=1}^d x_{rj} x_{sj} + \sum_{j=1}^d x_{sj}^2 \\ &= b_{rr} + b_{ss} - 2b_{rs} \end{aligned}$$

$$\begin{aligned} \sum_{r=1}^N d_{rs}^2 &= T + Nb_{ss} \\ \sum_{s=1}^N d_{rs}^2 &= T + Nb_{rr} \\ \sum_{r=1}^N \sum_{s=1}^N d_{rs}^2 &= 2NT \end{aligned}$$

$$T \equiv \sum_{i=1}^N b_{ii} = \sum_{i=1}^N \sum_{j=1}^d x_{ij}^2$$

$$d_{..s}^2 \equiv \frac{1}{N} \sum_{r=1}^N d_{rs}^2$$

$$d_{r..}^2 \equiv \frac{1}{N} \sum_{s=1}^N d_{rs}^2$$

$$d_{..}^2 \equiv \frac{1}{N^2} \sum_{r=1}^N \sum_{s=1}^N d_{rs}^2$$

$$b_{rs} \equiv \sum_{j=1}^d x_{rj} x_{sj}$$



$$b_{rs} = \frac{1}{2}(d_{r..}^2 + d_{..s}^2 - d_{..}^2 - d_{rs}^2)$$



# MDS: General Methodology (2/3)

- The new coordinates of instance  $t$  are given by the  $t$ -th elements of the eigenvectors,  $c_j, j=1,\dots,k$ , after normalization.

$$d_{rs}^2 = b_{rr} + b_{ss} - 2b_{rs}$$

$$b_{rs} \equiv \sum_{j=1}^d x_{rj}x_{sj}$$



Only Distance Matrix is Needed

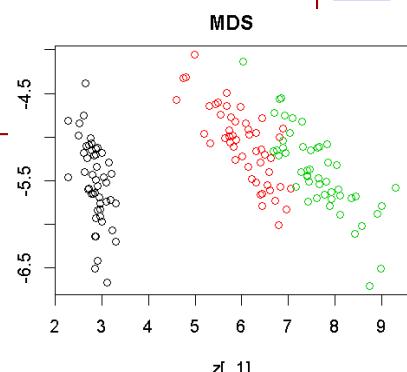
$$b_{rs} = \frac{1}{2}(d_{r\cdot}^2 + d_{\cdot s}^2 - d_{..}^2 - d_{rs}^2)$$

$$\mathbf{B} = \mathbf{X}\mathbf{X}^T \quad [N \times N]$$

Spectral Decomposition

$$\mathbf{X} = \mathbf{CD}^{1/2}$$

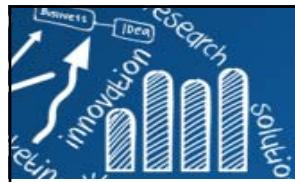
```
> x <- as.matrix(iris[, 1:4])
> B <- x %*% t(x)
> d <- 2
> eB <- eigen(B)
> C <- eB$vectors
> D <- eB$values
> Z <- C[, 1:d] %*% diag(D[1:d])
> plot(z[,1], z[,2],
+ col=iris[,5], main="MDS")
```



- C: columns are the eigenvectors of  $\mathbf{B}$ .  
D: diagonals are the eigenvalues of  $\mathbf{B}$ .

$$\mathbf{Z} \equiv \widehat{\mathbf{X}} = \begin{matrix} c_1 & \dots & c_k & \dots & c_N \\ \boxed{\mathbf{C}} & & & & \boxed{\mathbf{D}} \\ \sqrt{\lambda_1} & \ddots & \sqrt{\lambda_k} & \ddots & \sqrt{\lambda_N} \end{matrix}$$

$$z_{tj} = \sqrt{\lambda_j} c_{tj} \quad j = 1, \dots, k \\ t = 1, \dots, N$$



## MDS: General Methodology (3/3)

- The map is **centered** on the origin, we can get **any rotated** or mirror image version.
- MDS can be used for dimension reduction by calculating pairwise Euclidean distances in the d-dimensional  $x$  space and giving this as input to MDS, which then projects it to a lower-dimensional space so as to **preserve** these distance.

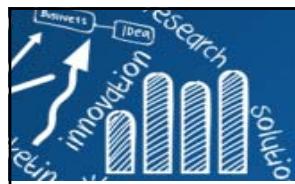
$$\mathbf{B} = \mathbf{XX}^T$$

$[N \times N]$

$$\mathbf{C} = \mathbf{X}^T \mathbf{X}$$

$[d \times d]$

- Chatfield and Collins, 1980: Eigenvalues of  $(\mathbf{XX}^T)$  are the same as those of  $(\mathbf{X}^T \mathbf{X})$  and eigenvectors are related by a simple linear transformation.
- PCA does the same work with MDS.
- PCA done on the correlation matrix equals doing MDS with standardized Euclidean distance where each variable has unit variance.



## General Case of MDS

- The linear mapping

$$\mathbf{z} = g(\mathbf{x}|\theta) \quad \mathbf{z} \in \mathcal{R}^k \quad \mathbf{x} \in \mathcal{R}^d$$

$g(\mathbf{x}|\theta)$ : the mapping function from  $d$  to  $k$  dimensions

Classical MDS:  $\mathbf{z} = g(\mathbf{x}|\theta) = \mathbf{W}^T \mathbf{x}$

- The nonlinear mapping: **Sammon mapping**

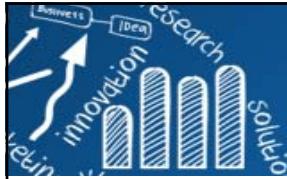
- Sammon stress: the normalized error in the mapping

$$\begin{aligned} E[\theta|\mathbf{X}] &= \sum_{r,s}^N \frac{(\| \mathbf{z}_r - \mathbf{z}_s \| - \| \mathbf{x}_r - \mathbf{x}_s \|)^2}{\| \mathbf{x}_r - \mathbf{x}_s \|^2} \\ &= \sum_{r,s}^N \frac{(\| g(\mathbf{x}_r|\theta) - g(\mathbf{x}_s|\theta) \| - \| \mathbf{x}_r - \mathbf{x}_s \|)^2}{\| \mathbf{x}_r - \mathbf{x}_s \|^2} \end{aligned}$$

- Nonlinear dimensionality reduction
- Use regression method for  $g(\cdot|\theta)$

- In the case of classification, one can include class information in the distance (Webb, 1999) as  $d'_{rs} = (1 - \alpha)d_{rs} + \alpha c_{rs}$

$c_{rs}$ : the distance between the classes  $\mathbf{x}_r$  and  $\mathbf{x}_s$  belong to.

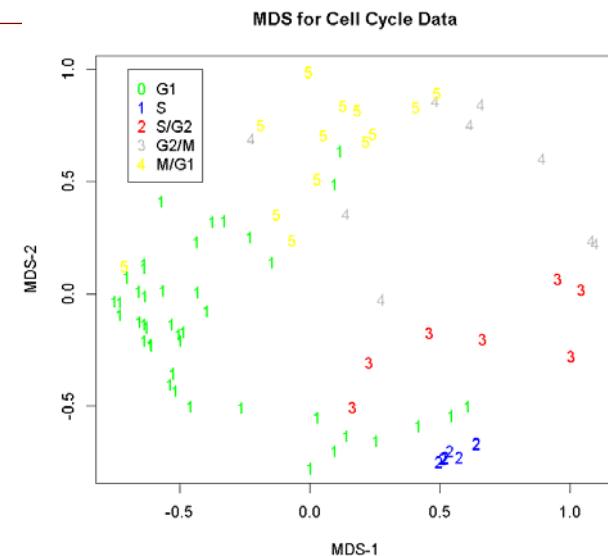


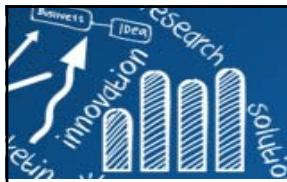
# MDS to Microarray Data

```

cell.matrix <- read.table("YeastCellCycle_alpha.txt", header=TRUE, row.names=1)
n <- dim(cell.matrix)[1]
p <- dim(cell.matrix)[2]-1
cell.data <- cell.matrix[,2:p+1]
gene.phase <- cell.matrix[,1]
phase.name <- c("G1", "S", "S/G2", "G2/M", "M/G1")
cell.sdata <- t(scale(t(cell.data)))
cell.cor <- cor(t(cell.sdata))
cell.dist <- sqrt(2*(1-cell.cor))
cell.mds <- cmdscale(cell.dist)
plot(cell.mds[,1], cell.mds[,2], type="n", xlab="MDS-1", ylab="MDS-2", main="MDS for
Cell Cycle Data")
number <- c(1, 4, 5, 2, 3)[as.integer(gene.phase)]
phase.color <- c("green", "blue", "red", "gray", "yellow")
text(cell.mds[,1], cell.mds[,2], number, col= phase.color[number])
legend(-0.7, 1.0, phase.name, pch="01234", col=phase.color)

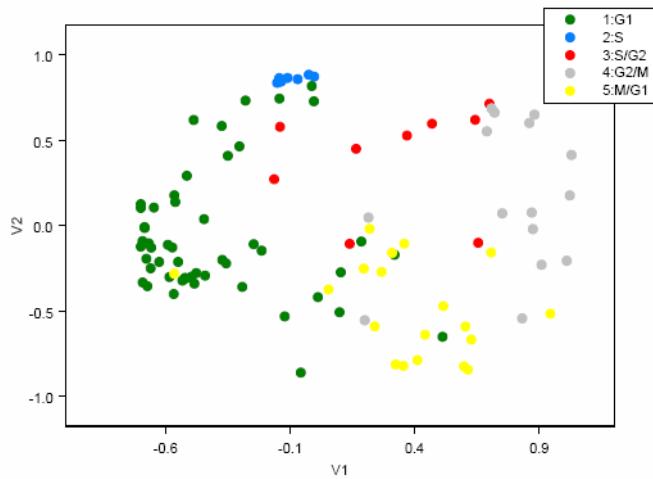
```



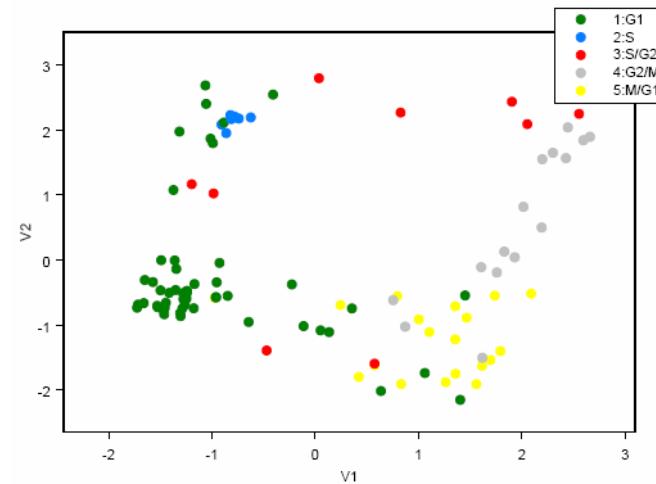


# Using Different Features

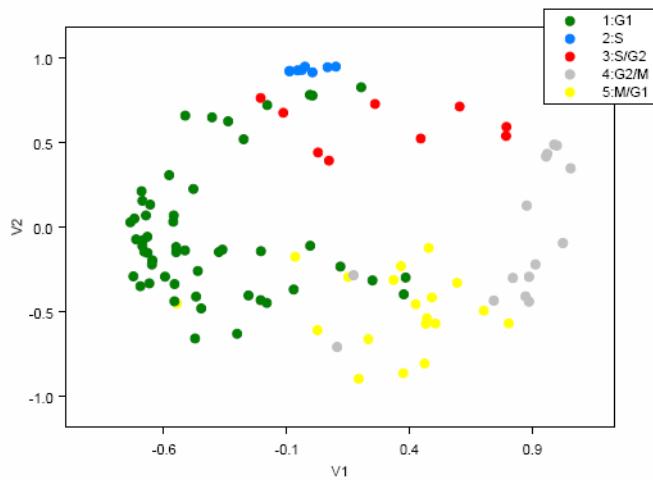
(a) MDS



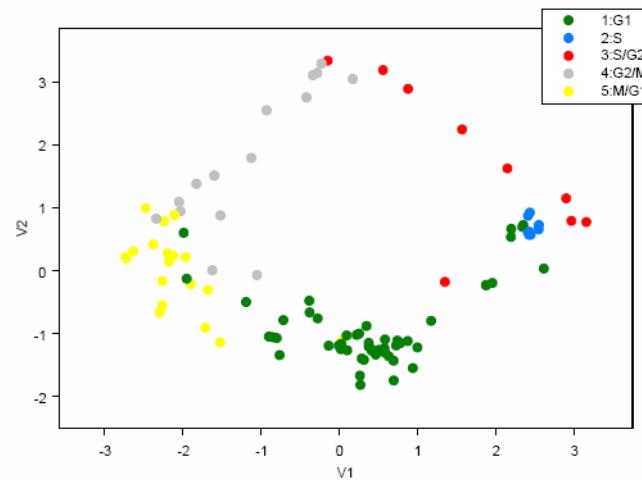
(c) Isomap



(b) MDS+DWT



(d) Isomap+DWT





# R: PCA, MDS with K-means

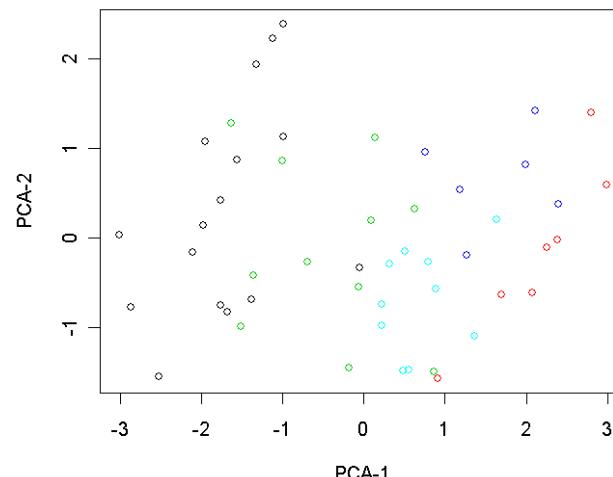
```
> library(stats)

no.group <- 5
no.iter <- 20
USArrests.kmeans <- kmeans(USArrests, no.group, no.iter)
plot(USArrests, col = USArrests.kmeans$cluster, main = "K-means: USArrests data")

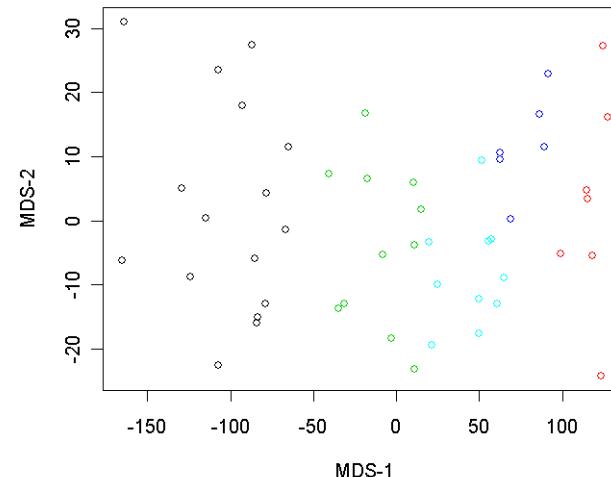
# PCA
USArrests.pca <- princomp(USArrests, cor=TRUE, scores=TRUE)
pca.dim1 <- USArrests.pca$scores[,1]; pca.dim2 <- USArrests.pca$scores[,2]
plot(pca.dim1, pca.dim2, main="PCA for USArrests Data with K-means", xlab="PCA-1",
     ylab="PCA-2", col=USArrests.kmeans$cluster)

# MDS
USArrests.mds<- cmdscale(dist(USArrests))
mds.dim1 <- USArrests.mds[,1]; mds.dim2 <- USArrests.mds[,2]
plot(mds.dim1, mds.dim2, xlab="MDS-1", ylab="MDS-2", main="MDS for USArrests Data with K-
means", col = USArrests.kmeans$cluster)
```

PCA for USArrests Data with K-means



MDS for USArrests Data with K-means



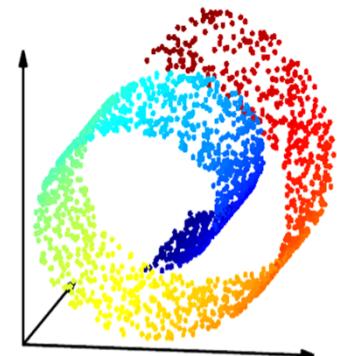
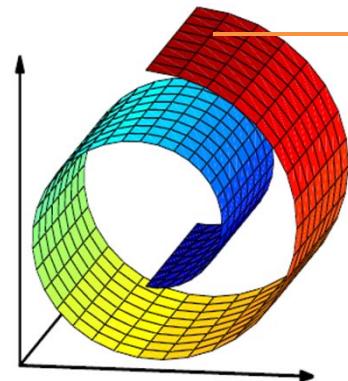


## (7) Isometric Mapping (Isomap)

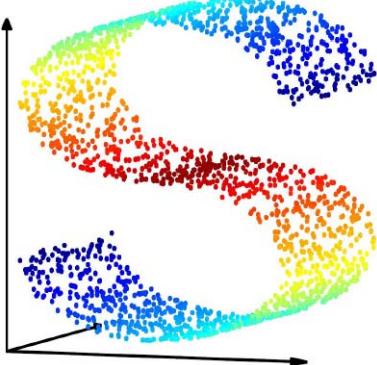
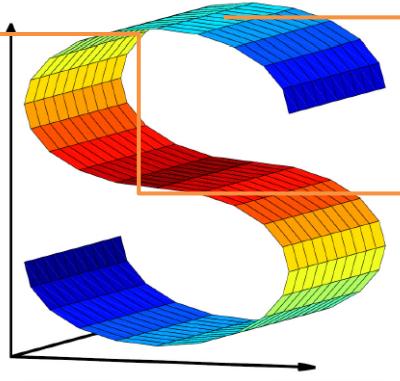
Manifolds (流形) and Nonlinearity:

A **manifold** is a mathematical geometric space and in which the **local** space is Euclidean space.

Swissroll

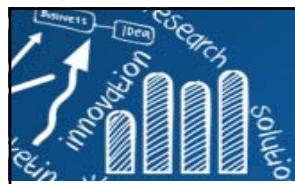


S-curve



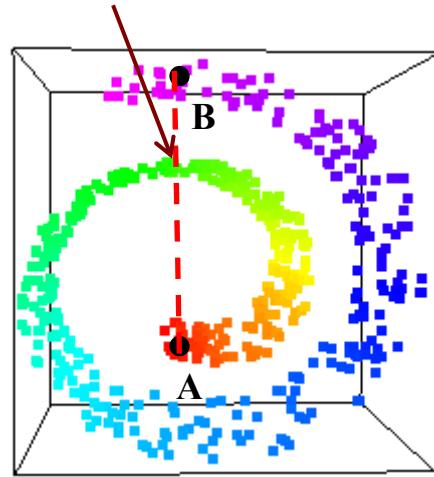
Locally Euclidean



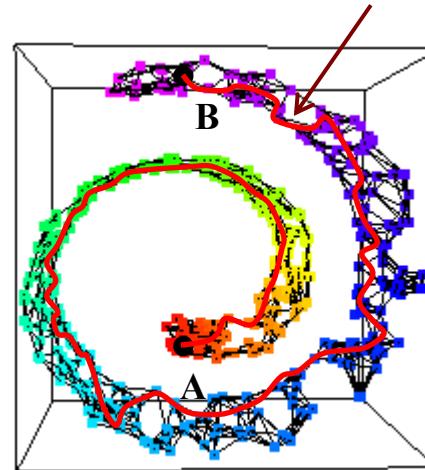


# Nonlinear Structure

linear distance



nonlinear path



**Machine learning for  
nonlinear dimension reduction (NLDR) and  
manifold learning**

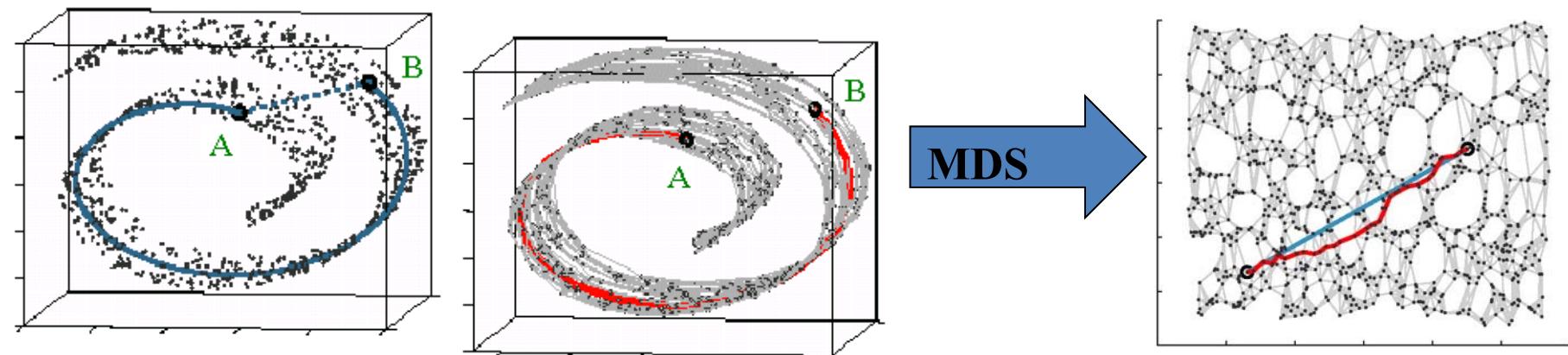
- **ISOMAP** (Tenenbaum, de Silva and Langford, 2000)
- Local linear embedding (**LLE**)(Roweis and Saul, 2000)
- **Hessian eigenmaps** (Donoho and Grimes, 2003)
- **Laplacian eigenmaps** (Belkin and Niyogi, 2003)
- **Diffusion maps** (Coifman et al., 2005) and their variants.



# Isometric Mapping (Isomap)

Tenenbaum , J. B., Silva, V. de, and Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction, *Science* 290, 2319-2323.

Isomap finds the **projection** that **preserves the global, nonlinear geometry** of the data by preserving the geodesic manifold interpoint distances.



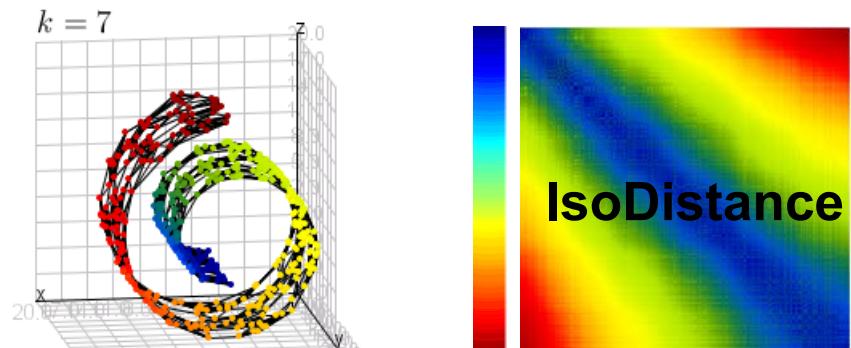
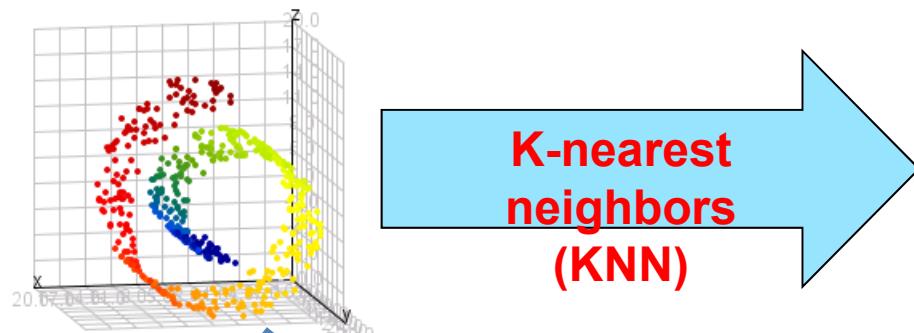
**What is important is the geodesic distance!**



# Isometric Feature Mapping (ISOMAP)<sup>77/144</sup>

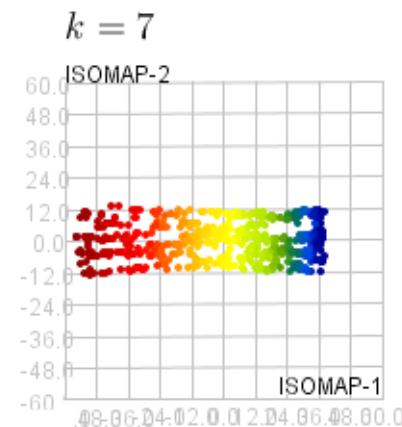
Tenenbaum, J.B., Silva, V.d., Langford, J.C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science **290** (2000) 2319–2323

## Geodesic Distance Approximation



nxn

Nonlinear Dimension Reduction



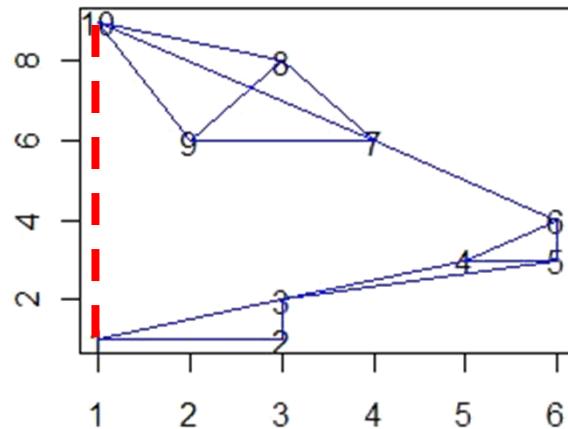
Multidimensional Scaling (MDS)



# Geodesic Distance

## Euclidean Distance

samples = 10, neighbors = 3



$$d_G(i, j) = \min\{ d_G(i, l) + d_G(l, j) \}$$

IsoDistance

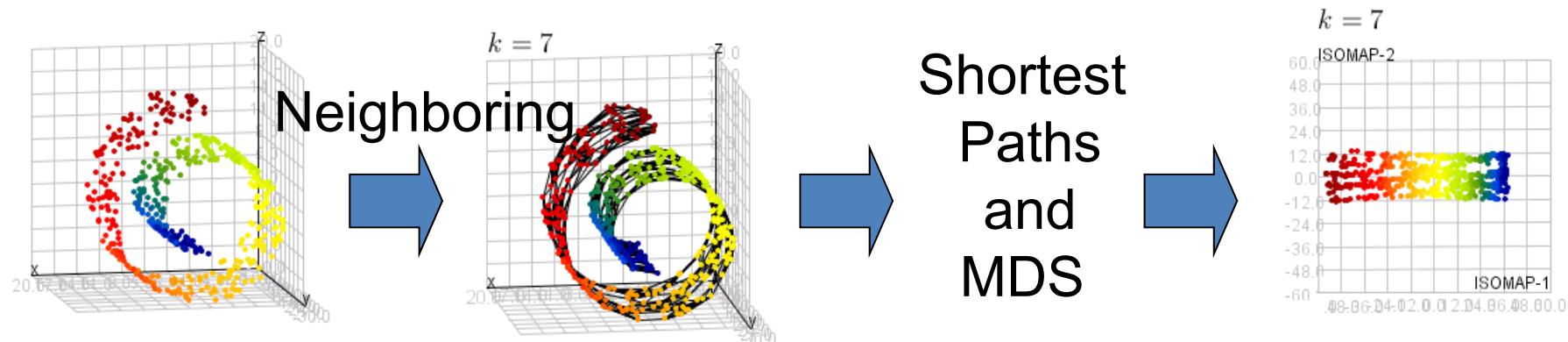
1	2	3	4	5	6	7	8	9	10
1	0	2	2.236	4.472					?
2	2	0	1	2.828					
3	2.236	1	0	2.236					
4		2.236	0	1	1.414				
5			3.162	1	0	1			
6				1.414	1	0	2.828		
7					2.828	0	2.236	2	
8						2.236	0	2.236	2.236
9							2	2.236	0
10								3.162	0

The shortest path by point 1 to point 10: 1 → 3 → 4 → 6 → 7 → 8 → 10

1	2	3	4	5	6	7	8	9	10	
1	0	2	2.236	4.472	5.398	5.886	8.715	10.951	10.715	12.957
2	2	0	1	2.828	3.828	4.243	7.071	9.307	9.071	11.314
3	2.236	1	0	2.236	3.162	3.65	6.479	8.715	8.479	10.721
4	4.472	2.828	2.236	0	1	1.414	4.243	6.479	6.243	8.485
5	5.398	3.828	3.162	1	0	1	3.828	6.064	5.828	8.071
6	5.886	4.243	3.65	1.414	1	0	2.828	5.064	4.828	7.071
7	8.715	7.071	6.479	4.243	3.828	2.828	0	2.236	2	4.243
8	10.951	9.307	8.715	6.479	6.064	5.064	2.236	0	2.236	2.236
9	10.715	9.071	8.479	6.243	5.828	4.828	2	2.236	0	3.162
10	12.957	11.314	10.721	8.485	8.071	7.071	4.243	2.236	3.162	0

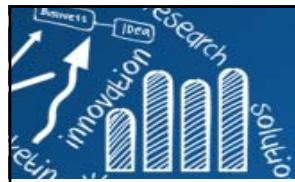


# Algorithm of Isomap



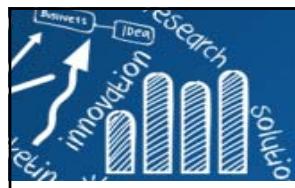
## Algorithm of Isomap (Tenenbaum et al., 2000)

1. Calculate the distance  $d_X(i, j)$  between all pairs  $i, j$  from  $n$  data points in the  $p$ -dimensional input space.
2. Construct the graph by determining the neighbors for each data point with  $\epsilon$ -Isomap or  $k$ -Isomap.
3. Pursue the shortest paths in the graph  $G$ .  
Initialize  $d_G(i, j) = d_X(i, j)$  if  $i, j$  are neighbors; otherwise, set  $d_G(i, j) = \infty$ . For each value of  $l = 1, 2, \dots, n$  and for all  $i, j$ ,  $d_G(i, j)$  are replaced by  $\min\{d_G(i, j), d_G(i, l) + d_G(l, j)\}$ .
4. Apply classical MDS to  $D_G$ .

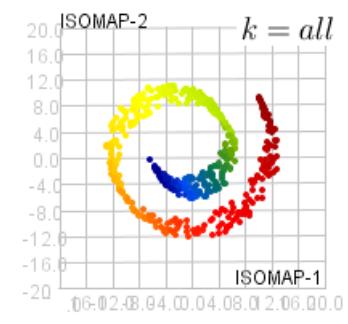
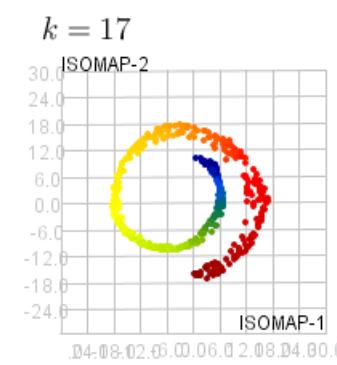
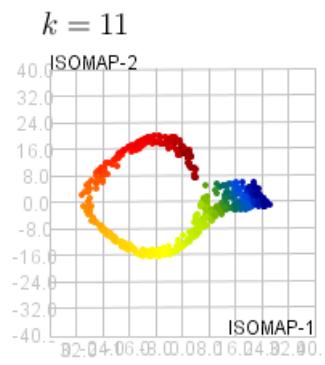
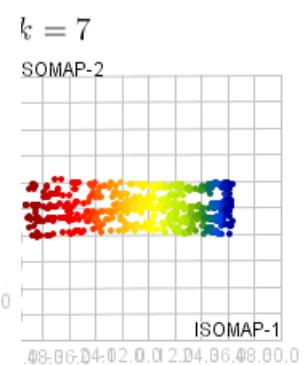
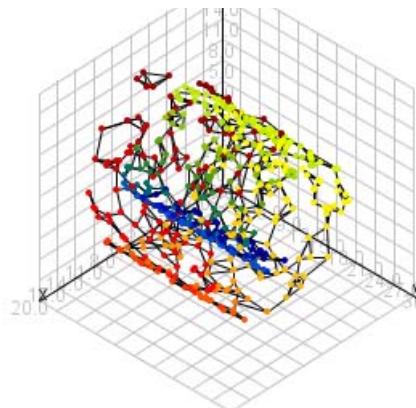
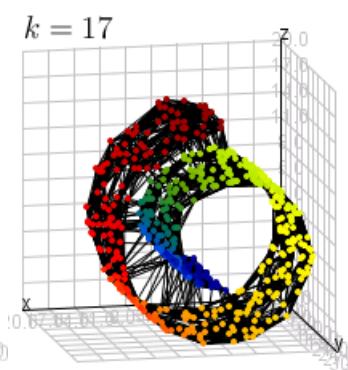
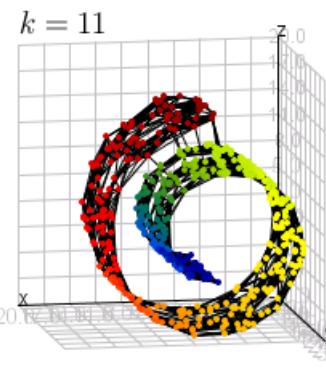
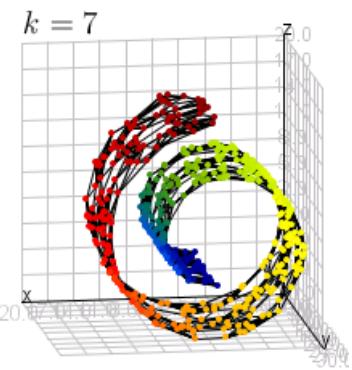
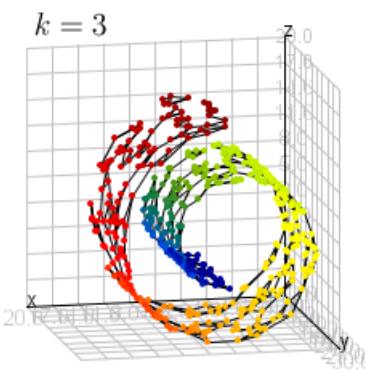
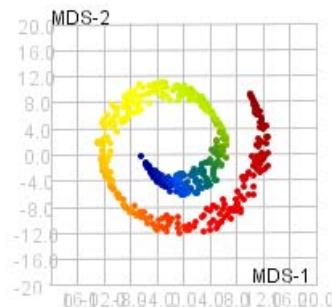
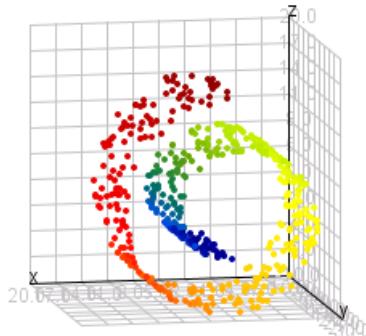


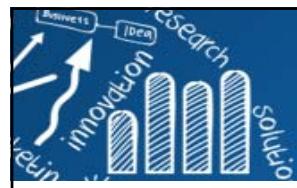
## Notes

- For neighboring points **Euclidean distance** is a good approximation to the geodesic distance.
  - For faraway points estimate the distance by **a series of short hops** between neighboring points.
  - Find **shortest paths** in a graph with edges connecting neighboring data points.
  - Once we have all **pairwise geodesic distances** use classical metric MDS.
- Balasubramanian et al. (2002), *Science*:  
"What is new about the Isomap algorithm is how it defines the **connectivity** of each data point via **its nearest Euclidean neighbors** in the high-dimensional space."

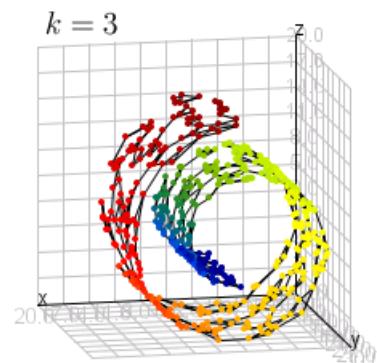
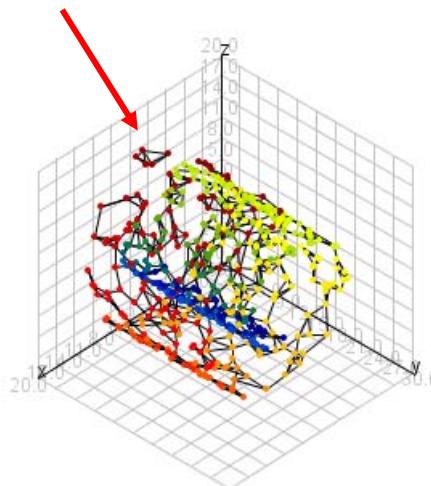


# The Pinch and Short-Circuit Problem

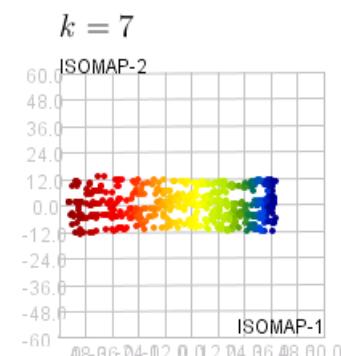
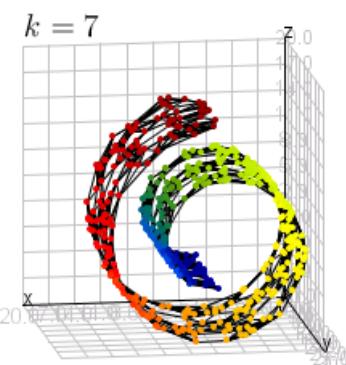
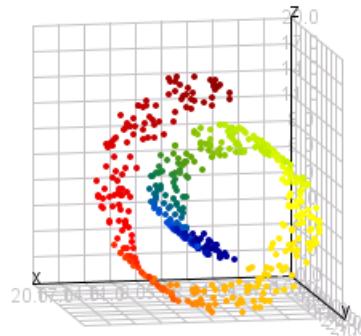




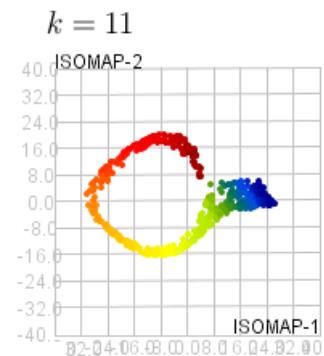
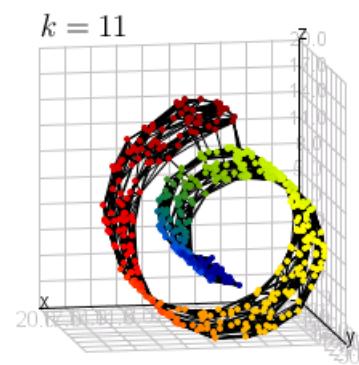
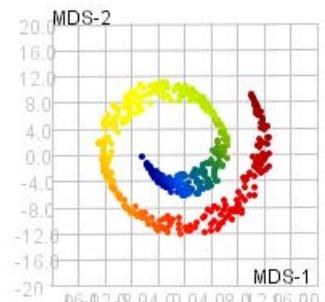
# Short-Circuit Problem



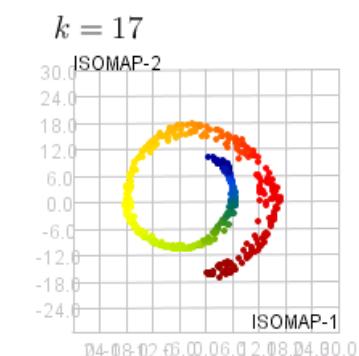
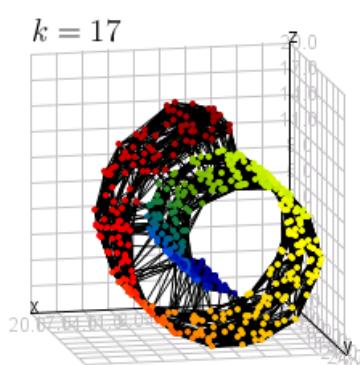
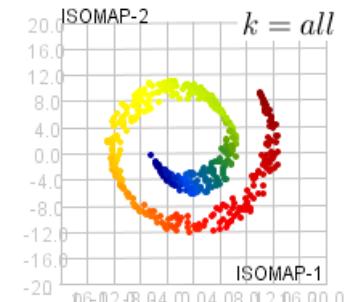
3D ScatterPlot



2D MDS



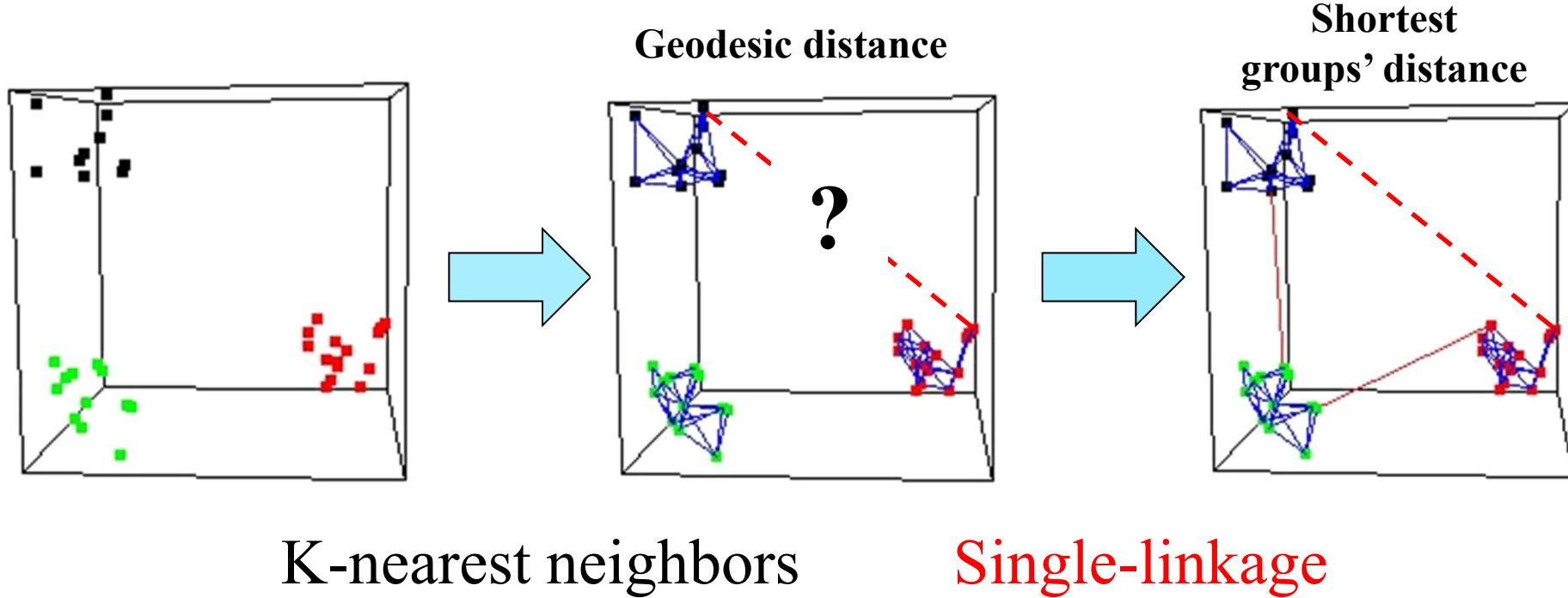
2D ISOAMP



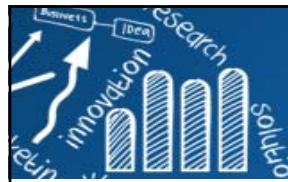


# Solving The Pinch and Short-Circuit Problem

83/144



**Benefit:** A small number of neighbors in computing geodesic distance.

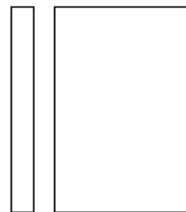


# Example

## lymphoma dataset

Alizadeh *et al.* (2000)

96 samples



854 genes

9 diagnostic classes

defined by Alizadeh *et al.* (2000).

- DLBCL
- Germinal Centre B
- NI Lymph Node/Tonsil
- Activated blood B
- Resting/activated T
- Transformed cell lines
- Follicular lymphoma
- Resting blood B
- CLL

BIOINFORMATICS

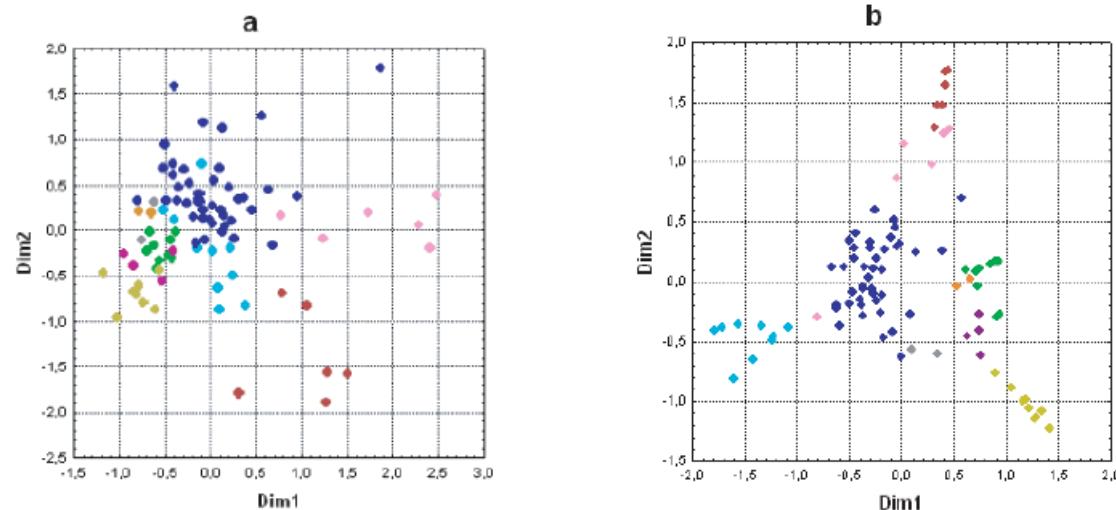
Vol. 20 no. 6 2004, pages 874–880  
DOI: 10.1093/bioinformatics/btg496

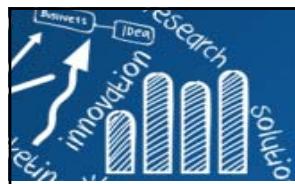


### Approximate geodesic distances reveal biologically relevant structures in microarray data

Jens Nilsson<sup>1,\*</sup>, Thoas Fioretos<sup>2</sup>, Mattias Höglund<sup>2</sup> and Magnus Fontes<sup>1</sup>

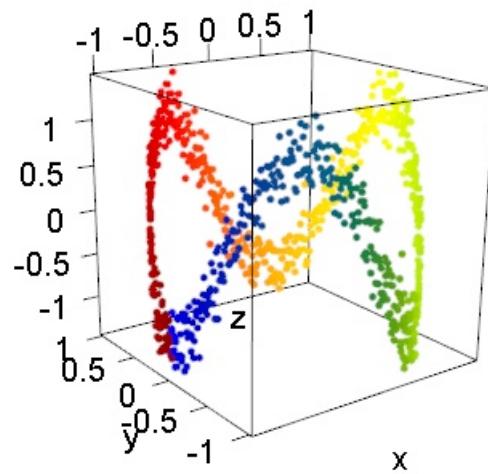
<sup>1</sup>Centre for Mathematical Sciences, Lund University, Box 118, SE-221 00 Lund, Sweden and <sup>2</sup>Department of Clinical Genetics, Lund University Hospital, SE-221 85 Lund, Sweden



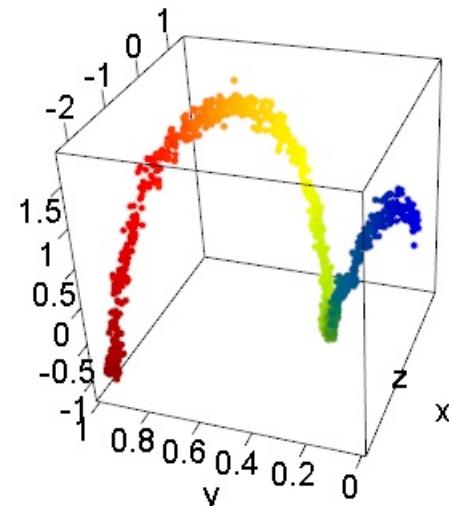


# Some Nonlinear Manifold Data Sets

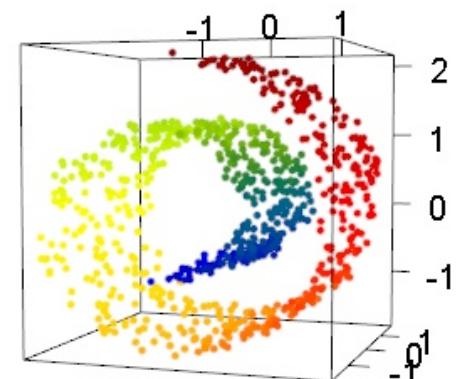
85/144



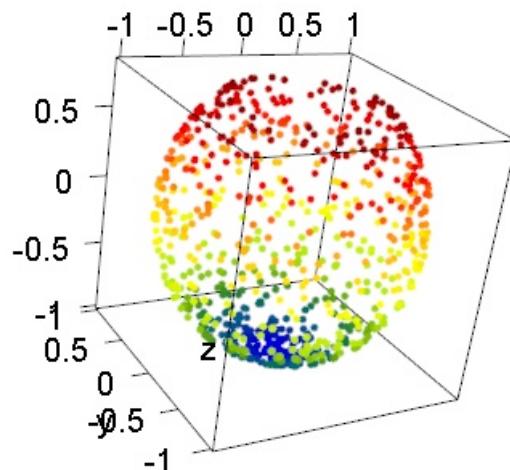
Trigcircle



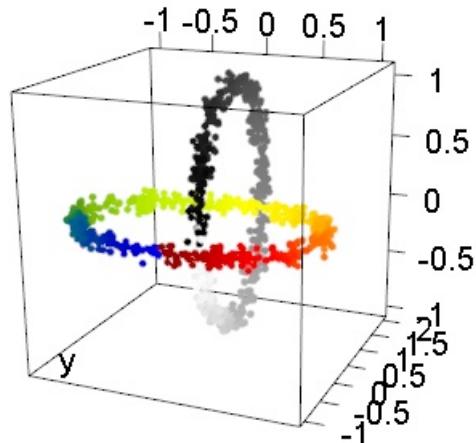
Spiral



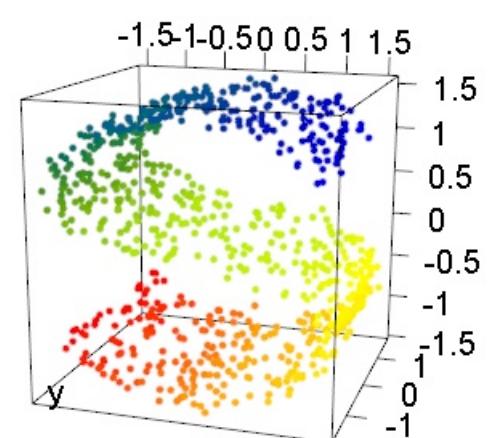
Swiss roll



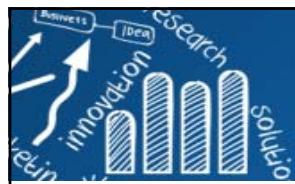
Fishbowl



Chainlink

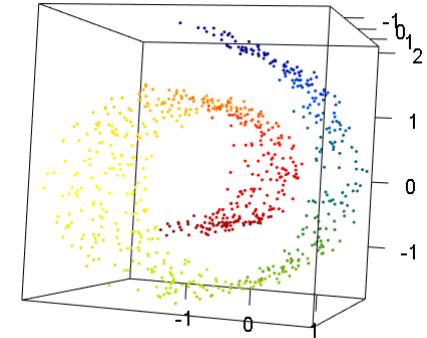


S curve

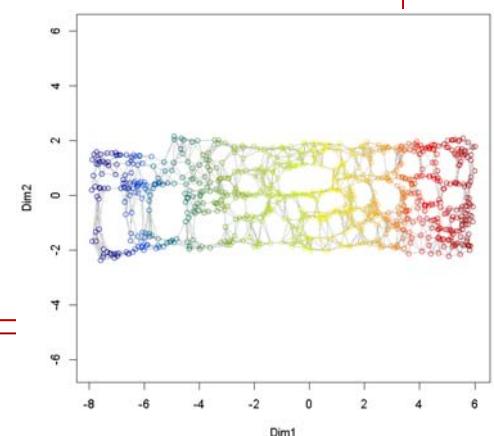


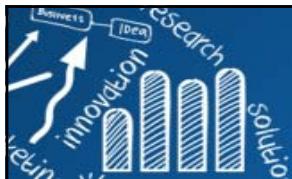
# Simulation of Swiss Roll Data Set

```
swissroll <- function(n, sigma=0.05){  
  
    angle <- (3*pi/2)*(1+2*runif(n));  
    height <- runif(n);  
    xdata <- cbind(angle*cos(angle), height, angle*sin(angle))  
  
    # angle <- seq((1.5*pi): (4.5*pi), length.out=n);  
    # height <- sample(0:21, n, replace = TRUE);  
    # xdata <- cbind(angle*cos(angle), height, angle*sin(angle))  
  
    xdata <- scale(xdata) + matrix(rnorm(n*3, 0, sigma), n, 3)  
  
    order.angle <- order(angle)  
    sort.angle <- sort(order.angle, index.return=TRUE)  
    col.id <- rainbow130(n)  
    my.color <- col.id[sort.angle$ix]  
  
    colnames(xdata) <- paste("x", 1:3, sep="")  
  
    return(list(xdata=xdata, angle=angle, color=my.color))  
}  
  
source("rainbow130.r")  
sr <- swissroll(800)
```



```
library(rgl); open3d()  
plot3d(sr$xdata[,1], sr$xdata[,2], sr$xdata[,3], col=sr$color, size=3,  
       xlab="", ylab="", zlab="", axes = T)  
library(vegan)  
sr.isomap <- isomap(dist(sr$xdata), ndim=2, k=7) # try different k  
plot(sr.isomap, col=sr$color)
```





# Simulation of S Curve Data Set

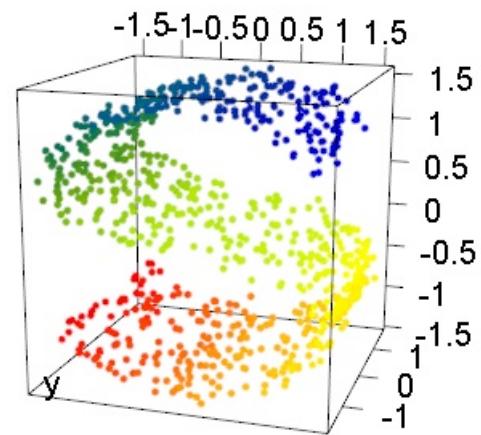
```

Scurve <- function(n){
  # upper half S curve
  theta1 <- runif((n/2), min=-0.9*(pi), max=(pi)/2)
  z1 <- runif((n/2), min=0, max=5)
  x1 <- -cos(theta1)
  y1 <- 2-sin(theta1)
  side.upper <- matrix(0, ncol=4, nrow=(n/2))
  for(i in 1:(n/2)){
    side.upper[i,1] <- x1[i]
    side.upper[i,2] <- y1[i]
    side.upper[i,3] <- z1[i]
    side.upper[i,4] <- theta1[i]
  }
  order.theta1 <- order(theta1)
  sort.theta1 <- sort(order.theta1, method="qu", index=TRUE)
  upper.color <- sort.theta1$ix

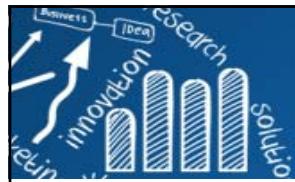
  # lower half S curve
  theta2 <- runif((n/2), min=(pi)/2, max=1.9*(pi))
  z2 <- runif((n/2), min=0, max=5)
  x2 <- cos((pi)-theta2)
  y2 <- sin((pi)-theta2)
  side.lower <- matrix(0, ncol=4, nrow=(n/2))

  for(i in 1:(n/2)){
    side.lower[i,1] <- x2[i]
    side.lower[i,2] <- y2[i]
    side.lower[i,3] <- z2[i]
    side.lower[i,4] <- theta2[i]
  }
  order.theta2 <- order(theta2)
  sort.theta2 <- sort(order.theta2, method="qu", index=TRUE)
  lower.color <- sort.theta2$ix
}

```



接續



# Simulation of S Curve Data Set

```
xdata <- rbind(side.upper[,c(1,3,2)], side.lower[,c(1,3,2)])
xdata <- scale(xdata) + matrix(rnorm(n*3,0, 0.05), n, 3)

angle <- c(side.upper[,4], side.lower[,4])
S.color <- c(upper.color, (n/2 + lower.color))
my.color <- (rainbow130((1.2)*n)[S.color])[1:n]

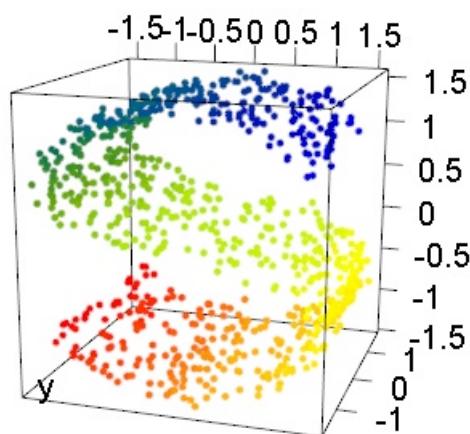
scatterplot3d(xdata, color=my.color,
               xlab="x", ylab="y", zlab="z",
               pch=20, angle=30)

open3d()
plot3d(xdata[,1], xdata[,2], xdata[,3], col=my.color, size=5,
       xlab="x", ylab="y", zlab="z")

return(list(xdata=xdata, angle=angle, color=my.color))

}
```

Use *rgl* package to plot the simulated data.

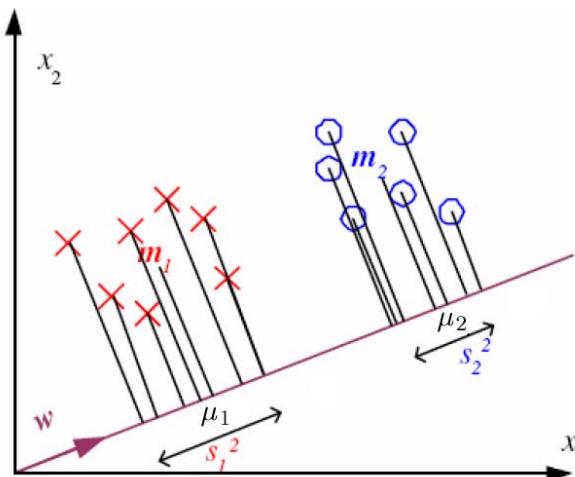




## (8) Linear Discriminant Analysis (LDA)

89/144

- LDA (Fisher, 1936) finds the linear combinations  $w^T x$  of  $x = (x_1, \dots, x_p)$  with large ratios of between-groups to within-groups sum of squares.
- LDA is a supervised method for dimension reduction for classification problem.
- Given samples from two classes  $C1$  and  $C2$ , we want to find the direction, as defined by a vector  $w$ , such that when the data are projected onto  $w$ , the examples from the two classes are as well separated as possible.





## LDA: Methodology (1/3)

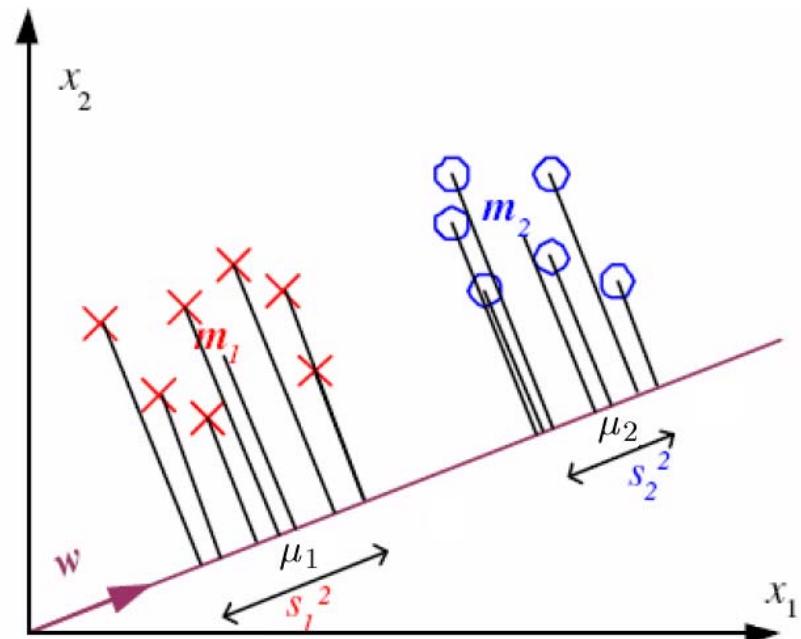
$$z = \mathbf{w}^T \mathbf{x}$$

the projection of  $\mathbf{x}$  onto  $\mathbf{w}$  and thus is a dimension reduction from  $d$  to 1.

$$X = \{\mathbf{x}^t, r^t\}, r^t = 1, \text{ if } \mathbf{x}^t \in C_1, r^t = 0, \text{ if } \mathbf{x}^t \in C_2$$

$\mathbf{m}_1 \in \mathcal{R}^d$ : the means of samples from  $C_1$   
before projection

$\mu_1 \in \mathcal{R}^1$ : the means of samples from  $C_1$   
after projection





## LDA: Methodology (2/3)

$$\mu_1 = \frac{\sum_t^N \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t^N r^t} = \mathbf{w}^T \mathbf{m}_1 \quad \mu_2 = \frac{\sum_t^N \mathbf{w}^T \mathbf{x}^t (1 - r^t)}{\sum_t^N (1 - r^t)} = \mathbf{w}^T \mathbf{m}_2$$

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

$$s_1^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - \mu_1)^2 r^t \quad s_2^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - \mu_2)^2 (1 - r^t)$$

Want  $|\mu_1 - \mu_2|$  to be large and  $s_1^2 + s_2^2$  to be small

$$J(\mathbf{w}) = \frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2}$$

$$\begin{aligned} (\mu_1 - \mu_2)^2 &= (\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{m}_2)^2 \\ &= \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2) (\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w} \\ &= \mathbf{w}^T \mathbf{S}_B \mathbf{w} \end{aligned}$$

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$

Between-class scatter matrix

$$s_1^2 + s_2^2 = \mathbf{w}^T \mathbf{S}_W \mathbf{w}$$

$$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

Total within-class scatter matrix

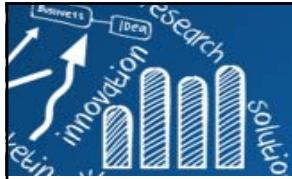
$$\begin{aligned} s_1^2 &= \sum_t (\mathbf{w}^T \mathbf{x}^t - \mu_1)^2 r^t \\ &= \sum_t \mathbf{w}^T (\mathbf{x}^t - \mathbf{m}_1) (\mathbf{x}^t - \mathbf{m}_1)^T \mathbf{w} r^t \\ &= \mathbf{w}^T \mathbf{S}_1 \mathbf{w} \end{aligned}$$

$$\mathbf{S}_1 = \sum_t r^t (\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T$$

Within-class scatter matrix for  $C_1$

$$s_2^2 = \mathbf{w}^T \mathbf{S}_2 \mathbf{w}$$

$$\mathbf{S}_2 = \sum_t (1 - r^t) (\mathbf{x}^t - \mathbf{m}_2)(\mathbf{x}^t - \mathbf{m}_2)^T$$



## LDA: Methodology (3/3)

$$J(\mathbf{w}) = \frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2} \quad \rightarrow \quad J(\mathbf{w}) = \frac{\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2) |^2}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$



$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$



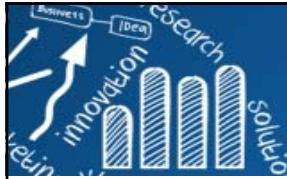
$$\frac{\partial J}{\partial \mathbf{w}} = \frac{\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} (2(\mathbf{m}_1 - \mathbf{m}_2) - \frac{\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \mathbf{S}_W \mathbf{w}) = 0$$

$$\mathbf{w} = c \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

- Because it is the direction that is important for us and not the magnitude, set  $c=1$  and find  $\mathbf{w}$

**NOTE:**  $p(\mathbf{x}|C_i) \sim N(\mu_i, \Sigma)$

Linear discriminant  $\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$



# LDA: More Than 2 Classes

$$\mathbf{z} = \mathbf{W}^T \mathbf{x} \quad \mathbf{z} \in \mathcal{R}^k \quad \mathbf{W} = [d \times k]$$

The within-class scatter matrix for  $C_i$

$$\mathbf{S}_i = \sum_t r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T$$

$r^t = 1$  if  $\mathbf{x}^t \in C_i$  and 0 otherwise

The total within-class scatter

$$\mathbf{S}_W = \sum_{i=1}^K \mathbf{S}_i$$

The between-class scatter matrix

$$\mathbf{S}_B = \sum_{i=1}^K N_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

$$\mathbf{m} = \frac{1}{K} \sum_{i=1}^K \mathbf{m}_i \quad N_i = \sum_t r_i^t$$

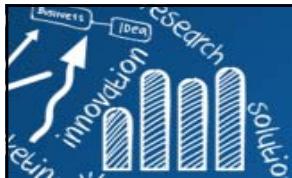
$$J(\mathbf{W}) = \frac{\mathbf{W}^T \mathbf{S}_B \mathbf{W}}{\mathbf{W}^T \mathbf{S}_W \mathbf{W}}$$

► The largest eigenvectors of  $\mathbf{S}_W^{-1} \mathbf{S}_B$  are the solution.

►  $\mathbf{S}_B$  is the sum of  $K$  matrices of rank 1,  $(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$ , and only  $K - 1$  of them are independent.

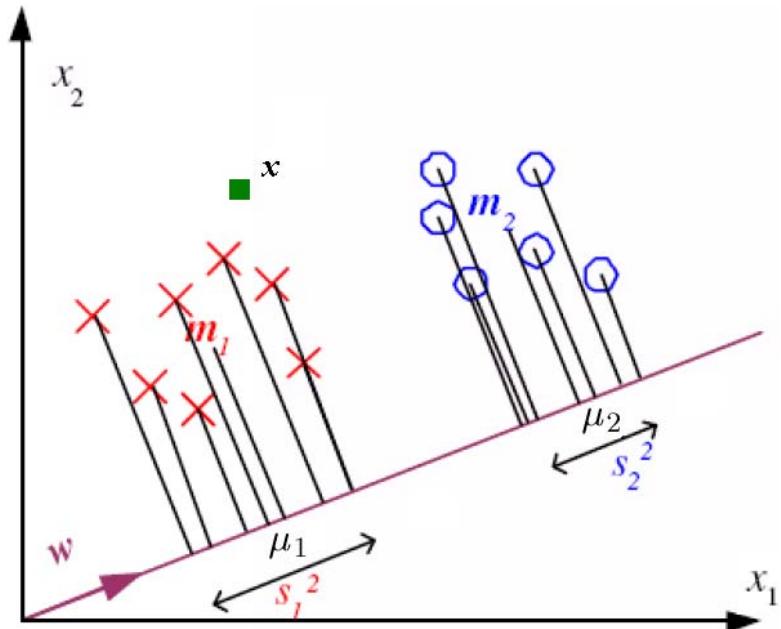
►  $\mathbf{S}_B$  has a maximum rank of  $K - 1$  and we take  $k = K - 1$ .

► define a new lower  $(K - 1)$  dimensional space, where the discriminant is then to be constructed.



# LDA: Classification

- Fisher's linear discriminant is optimal for normally distributed.
- After projection, for the two classes to be well separated we would like **the means to be as far apart as possible** and the examples of classes be scattered in as small regions as possible.



### **Classification Rules:**

For an observation  $\mathbf{x} = (x_1, \dots, x_d)$

$$d_k(\mathbf{x}) = ((\mathbf{x} - \bar{\mathbf{x}}_k) \mathbf{w})^2$$

denote its (squared) Euclidean distance, in terms of the discriminant variables, from the  $1 \times d$  vector of class  $k$  averages  $\bar{\mathbf{x}}$  for the learning set  $\mathcal{L}$ .

The predicted class for observation  $\mathbf{x}$  is  
 $\mathcal{C}(\mathbf{x}, \mathcal{L}) = \operatorname{argmin}_k d_k(\mathbf{x}),$   
 the class whose mean vector is closest to  $\mathbf{x}$  in the space of discriminant variables.



# R: LDA

```

> library(MASS)

> help(lda)

data(iris3)
Iris <- data.frame(rbind(iris3[,1], iris3[,2], iris3[,3]),
                     Species = rep(c("setosa","versicolor","virginica"), rep(50,3)))

## LDA
> library(MASS)
data <- Iris[,1:4]
class <- Iris[,5]
Iris.lda <- lda(x=data, grouping=class)
Iris.lda <- lda(Species ~ ., Iris)
lda.dim1 <- as.matrix(data) %*% iris.lda$scaling[,1]
lda.dim2 <- as.matrix(data) %*% iris.lda$scaling[,2]

## LDA for classification
trainingIndex <- sample(1:150, 75)
trainingSample <- Iris[trainingIndex, ]
testSample <- Iris[-trainingIndex, ]
table(Iris$Species[trainingIndex])

ldaRule <- lda(Species ~ ., Iris,
                subset = trainingIndex)
predict(ldaRule, testSample)$class

```

```

lda(Species ~ ., data = Iris)

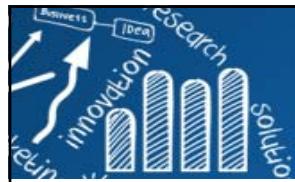
Prior probabilities of groups:
  setosa versicolor virginica
  0.3333333  0.3333333  0.3333333

Group means:
          Sepal.L. Sepal.W. Petal.L. Petal.W.
setosa      5.006   3.428   1.462   0.246
versicolor  5.936   2.770   4.260   1.326
virginica   6.588   2.974   5.552   2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.L.  0.8293776  0.02410215
Sepal.W.  1.5344731  2.16452123
Petal.L. -2.2012117 -0.93192121
Petal.W. -2.8104603  2.83918785

Proportion of trace:
    LD1     LD2
0.9912 0.0088

```



## (9) Sliced Inverse Regression (SIR)

96/144

K.C. Li, (1991),

Sliced inverse regression for dimension reduction, *JASA* **86**, 316–342.

$y$  is a univariate variable.

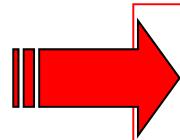
$f$  is an arbitrary function.

$$y = f(\beta'_1 \mathbf{x}, \dots, \beta'_K \mathbf{x}, \epsilon)$$

$\epsilon$  is a random variable  
independent of  $\mathbf{x}$ .

The  $\beta$ 's are referred to  
effective dimension reduction (*e.d.r.*)  
or projection directions.

$\mathbf{x}$  is a random vector  
with dimension  $p \times 1$ ,  
 $p \geq K$ .



Sliced inverse regression (SIR) is a method for  
estimating the *e.d.r.* directions based on  $y$  and  $\mathbf{x}$ .



# SIR: Algorithm

$$\mathbf{X} = \begin{array}{c|ccc} y & X_1 & X_2 & \cdots & X_p \\ \hline \mathbf{x}_1 & 10 & 1 & 4 & 2 \\ \mathbf{x}_2 & 13 & 3 & 5 & 3 \\ \mathbf{x}_3 & 9 & 0 & 3 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_N & 11 & 2 & 5 & 2 \end{array}$$

$$\bar{\mathbf{X}} = \begin{array}{c|ccc} y & X_1 & X_2 & \cdots & X_p \\ \hline \bar{\mathbf{x}}(1) & 9 & 0 & 3 & 2 \\ \bar{\mathbf{x}}(2) & 10 & 1 & 4 & 2 \\ \bar{\mathbf{x}}(3) & 11 & 2 & 5 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{\mathbf{x}}(H) & 12 & 2 & 4 & \cdots & 3 \\ & \vdots & \vdots & \vdots & \vdots & \vdots \\ & 13 & 3 & 5 & & 3 \\ & 17 & 5 & 5 & & 6 \end{array}$$

$$\bar{\mathbf{X}} = \begin{array}{c|ccc} X_1 & X_2 & \cdots & X_p \\ \hline \bar{\mathbf{x}}_1 & 0.5 & 3.5 & \cdots & 2.0 \\ \bar{\mathbf{x}}_2 & 2.0 & 4.5 & & 2.5 \\ \vdots & \vdots & \vdots & & \vdots \\ \bar{\mathbf{x}}_H & 4.0 & 5.0 & & 4.5 \end{array}$$

$$\hat{\Sigma}_W \hat{\beta}_i = \hat{\lambda}_i \hat{\Sigma}_X \hat{\beta}_i$$

$$\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_p$$

Sample Mean  
 $\bar{\mathbf{x}} = \frac{\sum_{i=1}^N \mathbf{x}_i}{N}$

Sample Covariance Matrix  
 $\hat{\Sigma}_X = \sum_{i=1}^N N^{-1} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$

Sliced Mean

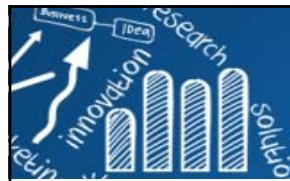
$$\bar{\mathbf{x}}_h = n_h^{-1} \sum_{(i) \in \text{slice}_h} \mathbf{x}_{(i)}$$

Weighted Covariance

Matrix for the Slice Means

$$\hat{\Sigma}_W = \sum_{h=1}^H \frac{n_h}{N} (\bar{\mathbf{x}}_h - \bar{\mathbf{x}})(\bar{\mathbf{x}}_h - \bar{\mathbf{x}})^T$$

$$\begin{array}{c|ccccc} y & & & & & \\ \hline 10 & \hat{\beta}_1 \mathbf{x}_1 & \hat{\beta}_2 \mathbf{x}_1 & \cdots & \hat{\beta}_K \mathbf{x}_1 \\ 13 & \hat{\beta}_1 \mathbf{x}_2 & \hat{\beta}_2 \mathbf{x}_2 & \cdots & \hat{\beta}_K \mathbf{x}_2 \\ 9 & & & & & \\ 17 & & & & & \\ 12 & \hat{\beta}_1 \mathbf{x}_N & \hat{\beta}_2 \mathbf{x}_N & \cdots & \hat{\beta}_K \mathbf{x}_N \\ 11 & & & & & \end{array}$$



# SIR: Theorem

## Linear Design Condition (L.D.C.)

For any  $b$  in  $R^p$ ,

the conditional expectation  $E(b' \mathbf{x} | \beta_1' \mathbf{x}, \dots, \beta_K' \mathbf{x})$  is linear in  $\beta_1' \mathbf{x}, \dots, \beta_K' \mathbf{x}$ ;

► that is, for some constants  $c_0, c_1, \dots, c_k$ ,

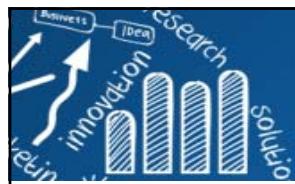
$$E(b' \mathbf{x} | \beta_1' \mathbf{x}, \dots, \beta_K' \mathbf{x}) = c_0 + c_1 \beta_1' \mathbf{x} + \dots + c_k \beta_K' \mathbf{x}.$$

### THEOREM:

under regular conditions, the centered inverse regression curve  $E[\mathbf{x}|y] - E[\mathbf{x}]$  is contained in the linear subspace spanned by  $\beta_k \Sigma_{\mathbf{X}}$  ( $k = 1, \dots, K$ ).

### COROLLARY 3.1 (Li, 1991)

Assume that  $\mathbf{x}$  has been standardized to  $\mathbf{z}$ . Then under the model and (3.1), the standardized inverse regression curve  $E(\mathbf{z}|y)$  is contained in the linear space generated by the standardized e.d.r. directions  $\theta_1 \ \theta_2 \ \dots \ \theta_K$



# SIR: Theorem

The SIR directions  $\mathbf{v}_i$  falls into the *e.d.r* space.

$$\text{Cov}(E[\mathbf{z}|y]) = E_y[E[\mathbf{z}|y]E[\mathbf{z}|y]^T]$$

$$E[\mathbf{z}|y] = c_1\theta_1 + \cdots + c_K\theta_K$$

$$= E_y[(c_1\theta_1 + \cdots + c_K\theta_K)(c_1\theta_1 + \cdots + c_K\theta_K)^T]$$

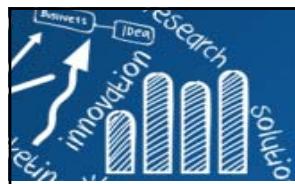
$$= (\theta_1 \ \theta_2 \ \cdots \ \theta_K) \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_K \end{pmatrix} (c_1 \ c_2 \ \cdots \ c_K) \begin{pmatrix} \theta_1^T \\ \theta_2^T \\ \vdots \\ \theta_K^T \end{pmatrix} = \Theta \mathbf{C} \Theta^T$$

$$= \Theta(\mathbf{U} \mathbf{D} \mathbf{U}^T) \Theta^T$$

$$= (\theta_1 \ \theta_2 \ \cdots \ \theta_K) \begin{pmatrix} c_1^2 & c_1c_2 & \cdots & c_1c_K \\ c_2c_1 & c_2^2 & \cdots & \\ c_3c_1 & c_3c_2 & c_3^2 & \cdots \\ \vdots & & & \\ c_Kc_1 & & & c_K^2 \end{pmatrix} \begin{pmatrix} \theta_1^T \\ \theta_2^T \\ \vdots \\ \theta_K^T \end{pmatrix} = (\Theta \mathbf{U}) \mathbf{D}^T (\Theta \mathbf{U})^T$$

$$\text{Cov}(E[\mathbf{z}|y])\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

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



## R: dr package

### ■ dr: Methods for Dimension Reduction for Regression

```
dr(formula, data, subset, group=NULL, na.action = na.fail,  
  weights, method = "sir", chi2approx="bx", ...)
```

Table 1: Methods implemented in dr. “Multivariate” is yes if the response can be multivariate. “Partial” is yes if conditioning on categorical predictors is allowed using the **group** argument. “Coord” is yes if coordinate tests are available.

Name	Reference	Multivariate	Partial	Coord
sir	Li (1991), Chiaromonte, Cook and Li (2002)	Yes	Yes	Yes
save	Cook and Weisberg (1991); Shao, Cook and Weisberg (2007)	Yes	Yes	Yes
phdy	Li (1992), Cook (1998)	No	No	No
phdres	Cook (1998)	No	No	No
phdq	Li (1992)	No	No	No
ire	Cook and Ni (2006), Wen and Cook (2007)	No	Yes	Yes

Source: The dr package, Vignettes, Sanford Weisberg



# SIR to Iris Data

```
> iris.sir <- dr(as.integer(Species) ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width , data=iris, nslices=3, chi2approx="wood", method="sir")
> summary(iris.sir)
Call:
dr(formula = as.integer(Species) ~ Sepal.Length + Sepal.Width +
    Petal.Length + Petal.Width, data = iris, nslices = 3, chi2approx = "wood",
    method = "sir")
Method:
sir with 3 slices, n = 150.
```

Slice Sizes:

50 50 50

Estimated Basis Vectors for Central Subspace:

	Dir1	Dir2
Sepal.Length	-0.2087	-0.006532
Sepal.Width	-0.3862	-0.586611
Petal.Length	0.5540	0.252562
Petal.Width	0.7074	-0.769453

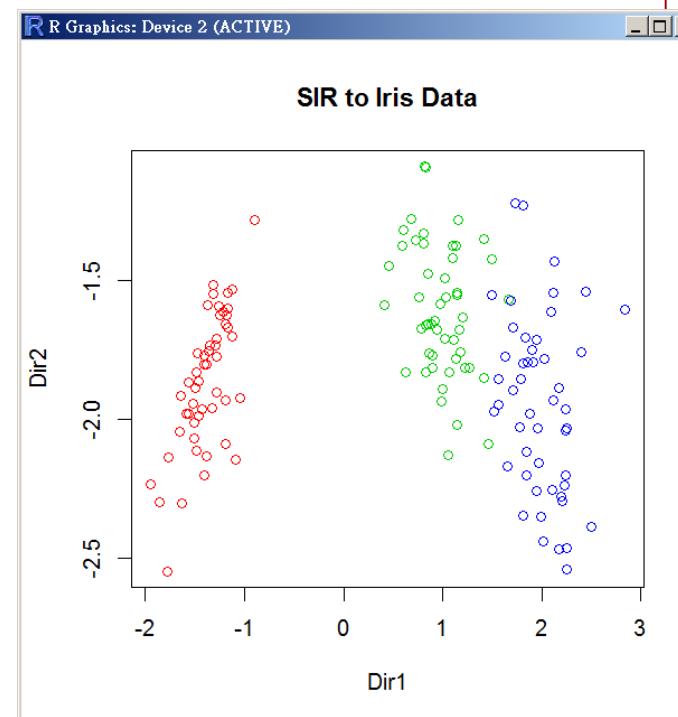
	Dir1	Dir2
Eigenvalues	0.9699	0.222
R^2(OLS dr)	0.9874	1.000

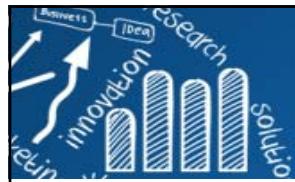
Large-sample Marginal Dimension Tests:

	Stat	df	p.value
0D vs >= 1D	178.8	8	0.000e+00
1D vs >= 2D	33.3	3	2.779e-07

```
> comp.sir <- as.matrix(iris[,1:4]) %*% as.matrix(iris.sir$evecs[,1:2])
> plot(comp.sir, col=as.integer(iris$Species)+1, main="SIR to Iris Data")
```

```
> install.packages("dr")
> library(dr)
```





# 模擬(1): SIR to the Simulated Data from the 102/144 Linear Model

```
Li6.1 <- function(n){  
  p <- 5  
  x <- matrix(0, ncol=p, nrow=n)  
  for(i in 1:p){  
    x[,i] <- rnorm(n, mean=0, sd=1)  
  }  
  
  epsilon <- rnorm(n, mean=0, sd=1)  
  y <- x[,1] + x[,2] + x[,3]+ x[,4] + epsilon  
  colnames(x) <- c("x1","x2","x3","x4","x5")  
  
  as.data.frame(cbind(y, x))  
}  
  
mydata <- Li6.1(200)  
attach(mydata)  
library(dr)  
my.sir <- dr(y~x1+x2+x3+x4+x5, method="sir",  
nslices=10)  
my.sir
```

## 1 Linear Model (6.1)

- $y = x_1 + x_2 + x_3 + x_4 + 0x_5 + \epsilon$
- $x_1, x_2, \dots, x_5 \sim N(0, 1), \epsilon \sim N(0, 1), x_i$ 's  $\perp \epsilon$
- $X_{[100 \times 5]}$
- $\beta = (1, 1, 1, 1, 0)$

```
> my.sir  
  
dr(formula = y ~ x1 + x2 + x3 + x4 + x5, method =  
  "sir",  
  nslices = 10)  
Estimated Basis Vectors for Central Subspace:  
          Dir1        Dir2        Dir3        Dir4  
x1 -0.52809049 -0.4830910  0.2286112 -0.03295919  
x2 -0.47623158  0.6938396  0.3784483  0.32234068  
x3 -0.49866200 -0.3134047 -0.4832488  0.62730069  
x4 -0.49551956  0.1774727 -0.3674617 -0.68542820  
x5 -0.01097735 -0.3943230  0.6602699 -0.17802302  
Eigenvalues:  
[1] 0.81108317 0.07944489 0.03748905 0.01976887
```



## 模擬(2): SIR to the Simulated Data from the 103/144 Quadratic Function Model

### 2 Quadratic Model (6.2)

- $y = x_1(x_1 + x_2 + 1) + \sigma \cdot \epsilon$
- $X_{[400 \times 10]}$
- $x_1, x_2, \dots, x_{10} \sim N(0, 1), \epsilon \sim N(0, 1), x_i$ 's  $\perp \epsilon$
- $\sigma = 0.5, \sigma = 1$
- $\beta_1 = (1, 0, \dots, 0), \beta_2 = (0, 1, 0, \dots, 0)$

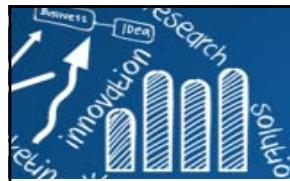
$$R^2(b) = \max_{\beta \in B} \frac{(b \Sigma_{xx} \beta')^2}{b \Sigma_{xx} b \cdot \beta \Sigma_{xx} \beta'}, \quad (2.2)$$

the squared multiple correlation coefficient between the projected variable  $b\mathbf{x}$  and the ideally reduced variables  $\beta_1\mathbf{x}, \dots, \beta_K\mathbf{x}$ .

$\Delta(\hat{\beta})$  = canonical correlations between  $(\beta_1, \beta_2; \hat{\beta}_1, \hat{\beta}_2)$ .

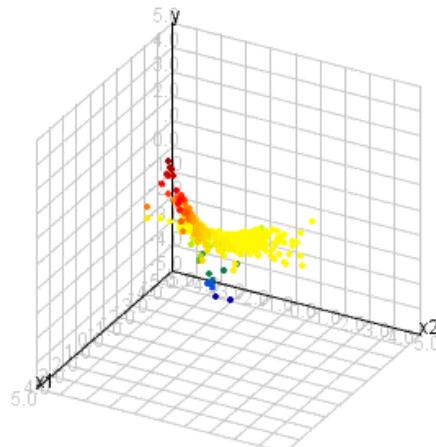
Table 2. Mean and Standard Deviation of  $R^2(\hat{\beta}_1)$  and  $R^2(\hat{\beta}_2)$  for the Quadratic Model (6.2),  $p = 10, n = 400$

$H$	$\sigma = 0.5$		$\sigma = 1$	
	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$
5	.91 (.05)	.75 (.15)	.88 (.07)	.52 (.21)
10	.92 (.04)	.80 (.13)	.89 (.08)	.55 (.24)
20	.93 (.04)	.77 (.15)	.88 (.08)	.49 (.26)

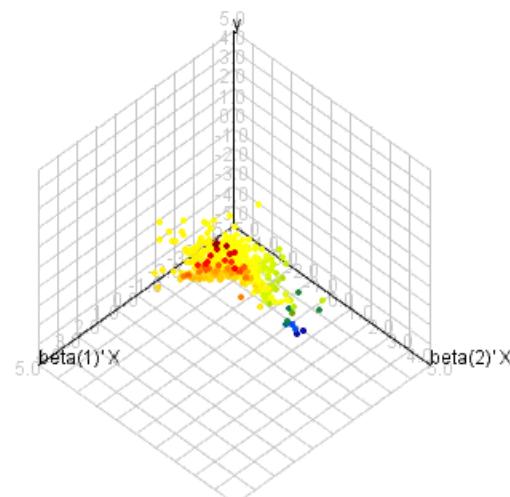


## 模擬(3): SIR to the Simulated Data from the 104/144 Rational Function Model

<http://www.hmwu.idv.tw/KSIR/Data/Li-DataIII/index.htm>



best view of the data

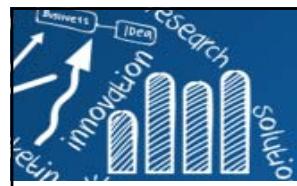


### 3 Rational Function Model (6.3)

- $y = x_1/(0.5 + (x_2 + 1.5)^2) + \sigma \cdot \epsilon$
- $x_1, x_2, \dots, x_{10} \sim N(0, 1), \epsilon \sim N(0, 1), x_i$ 's  $\perp \epsilon$
- $X_{[400 \times 10]}, \sigma = 0.5, \sigma = 1$
- $\beta_1 = (1, 0, \dots, 0), \beta_2 = (0, 1, 0, \dots, 0)$

Table 3. Mean and Standard Deviation of  $R^2(\hat{\beta}_1)$  and  $R^2(\hat{\beta}_2)$  for the Rational Function Model (6.3),  $p = 10, n = 400$

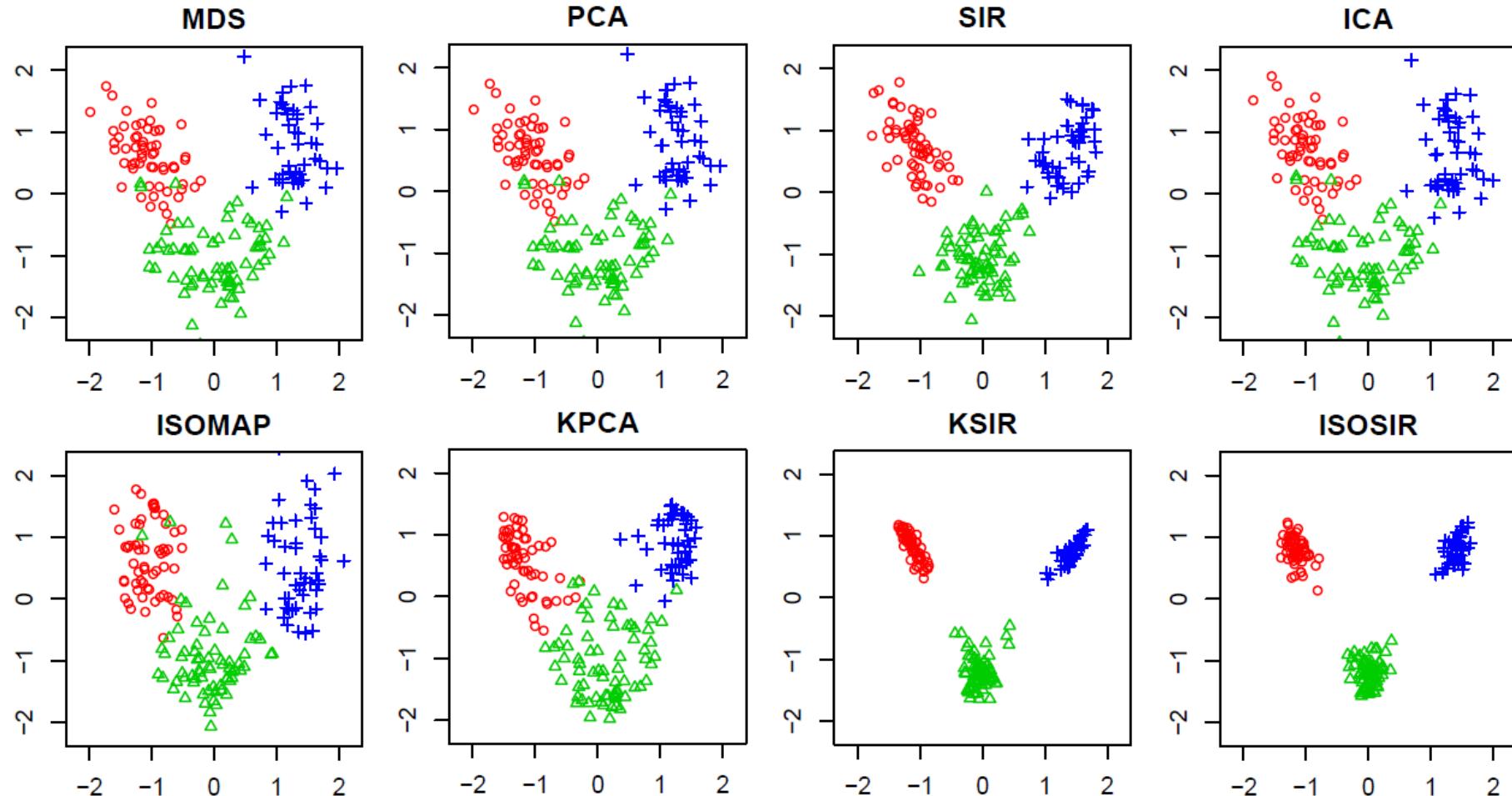
H	$\sigma = 0.5$		$\sigma = 1$	
	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$
5	.96 (.02)	.83 (.08)	.89 (.06)	.51 (.23)
10	.96 (.02)	.88 (.06)	.90 (.06)	.56 (.23)
20	.96 (.02)	.89 (.06)	.90 (.06)	.53 (.24)



## ISOSIR for Nonlinear Dimension Reduction and Data Visualization: Wine Data (178x13)

105/144

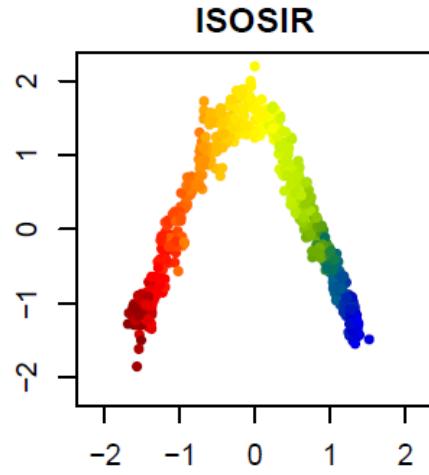
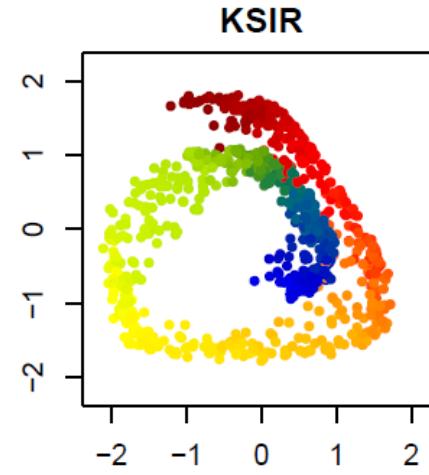
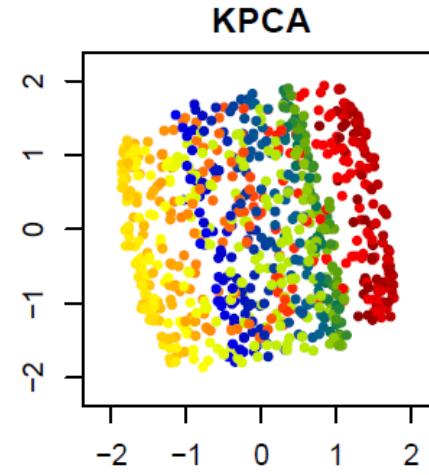
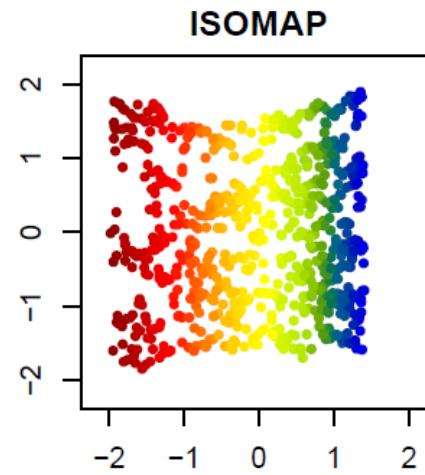
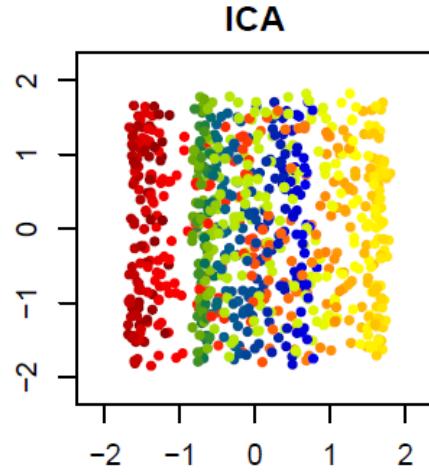
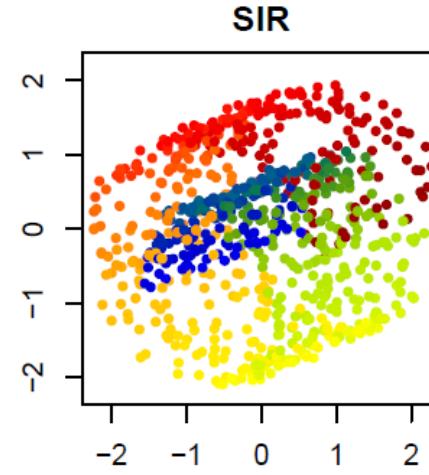
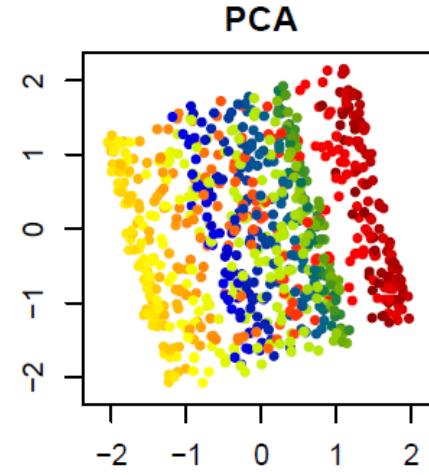
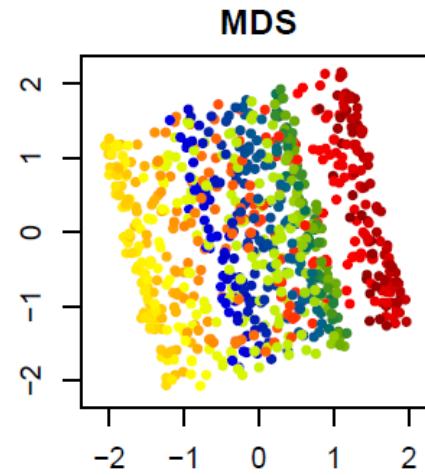
<https://archive.ics.uci.edu/ml/datasets/Wine>





## ISOSIR for Nonlinear Dimension Reduction and Data Visualization: Swissroll Data (800x3)

106/144



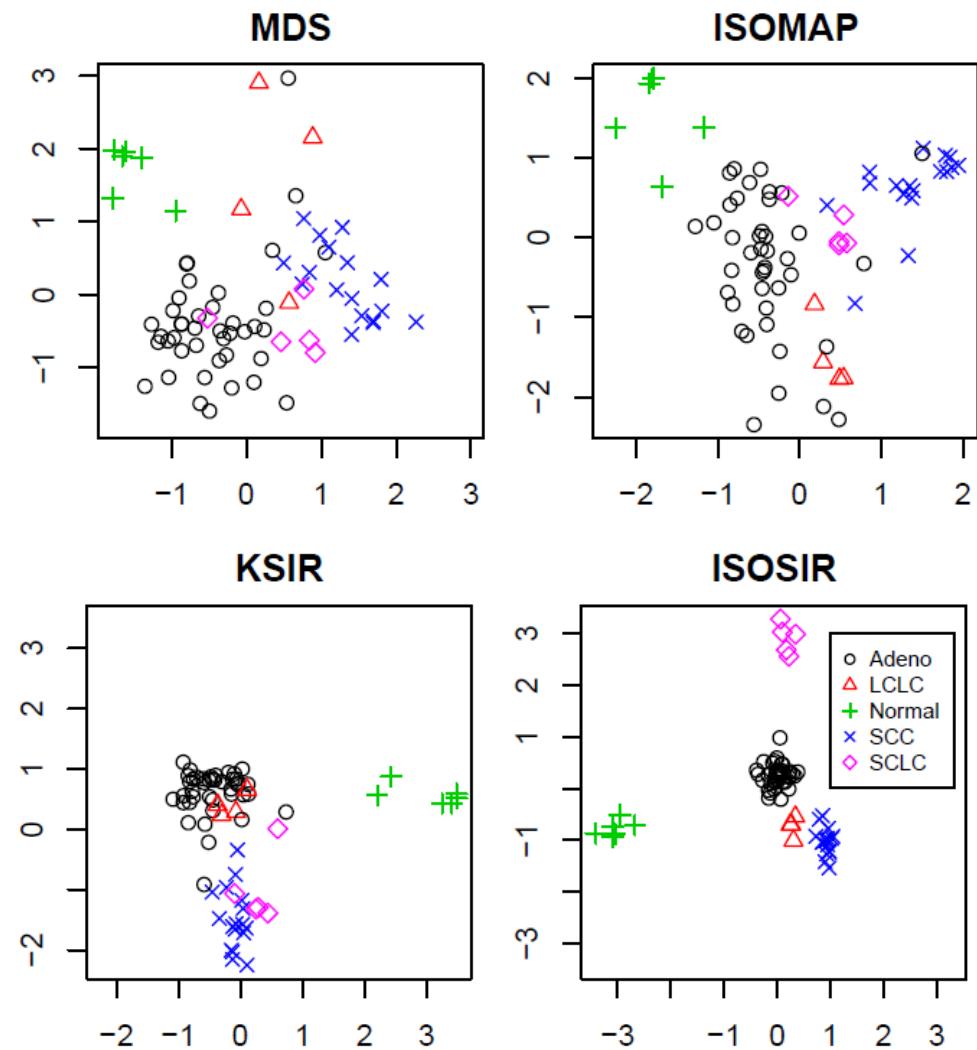


## ISOSIR for Nonlinear Dimension Reduction and Data Visualization: Lung Cancer Microarray Data (73x831) 107/144

**Aim:** find differentially expressed genes that can provide a basis for the classification of lung cancer subtypes (Garber et al. 2001).

**Samples:** 73 samples were divided into five diagnostic classes: adenocarcinoma (Adeno, 41), large-cell lung cancer (LCLC, 4), Normal (6), small-cell lung cancer (SCLC, 5), and squamous cell carcinoma (SCC, 17).

**Genes:** Nilsson et al. (2004) selected a subset of 831 genes and performed MDS and ISOMAP to visualize these samples.



小細胞肺癌 (SCLC) | 非小細胞肺癌(NSCLC): 包括腺癌(adenocarcinoma)、鱗狀細胞癌(SCC)、大細胞癌(LCLC)。



# High-dimensional data (HDD)

- Three different groups of HDD:
  - $d$  is large but smaller than  $n$ ;
  - $d$  is large and larger than  $n$ : **the high-dimension low sample size data (HDLSS)**; and
  - the data are functions of a continuous variable  $d$ : the **functional data**.
- In high dimension, the space becomes emptier as the dimension increases
  - when  $p > n$ , the rank  $r$  of the covariance matrix  $S$  satisfies  $r \leq \min\{p, n\}$ .
  - For HDLSS data, one cannot obtain more than  $n$  principal components.
  - Either PCA needs to be adjusted, or other methods such as ICA or Projection Pursuit could be used.

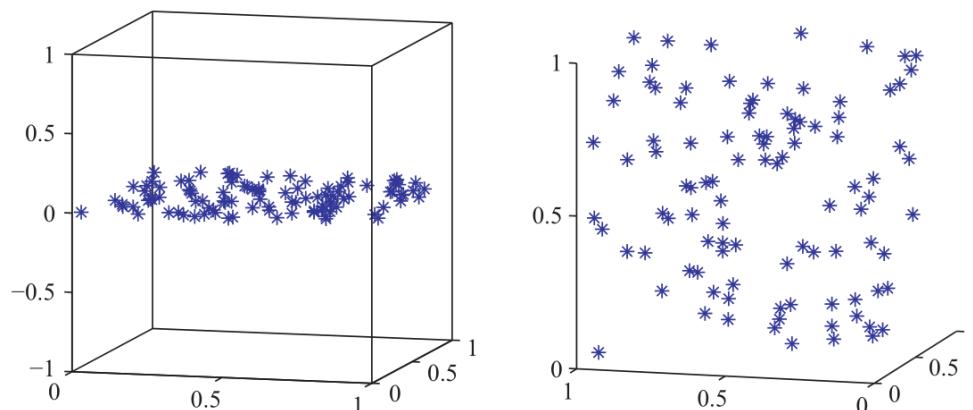
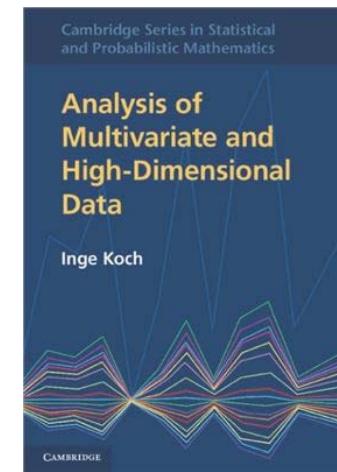


Figure 2.12 Distribution of 100 points in 2D and 3D unit space.





# HDLSS examples

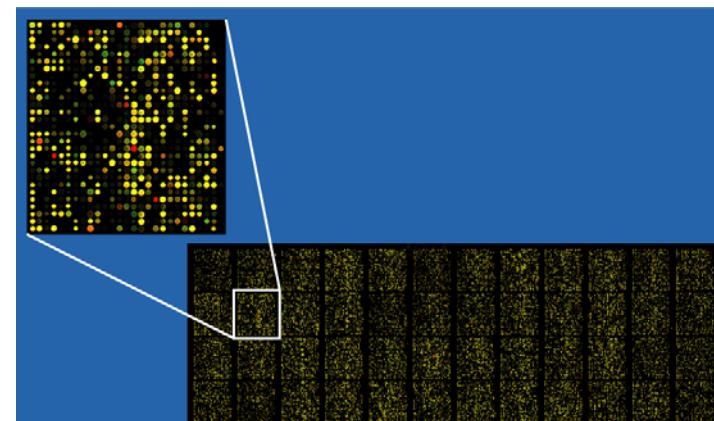
Sungkyu Jung and J. S. Marro, 2009, Pca Consistency In High Dimension, Low Sample Size Context, The Annals of Statistics 37(6B), 4104–4130.

- Examples:
  - in face recognition (**images**) we have many thousands of variables (pixels), the number of training samples defining a class (person) is usually small (usually less than 10).
  - **Microarray** experiments is unusual for there to be more than 50 repeats ( data points) for several thousand variables (genes).
- The covariance matrix will be singular, and therefore cannot be inverted. In these cases we need to find some method of estimating a full rank covariance matrix to calculate an inverse.



Face recognition using PCA

<https://www.mathworks.com/matlabcentral/fileexchange/45750-face-recognition-using-pca>



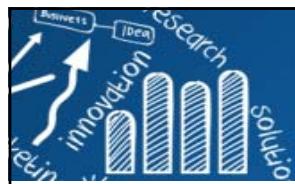
<https://zh.wikipedia.org/wiki/DNA微陣列>



# eigen/cov/PCA of a data set when n<<p

```
> x <- matrix(rnorm(100), ncol=20)
> dim(x)
[1] 5 20
> princomp(x)
Error in princomp.default(x) :
  'princomp' can only be used with more units than variables
```

```
> princomp
function (x, ...)
UseMethod("princomp")
<bytecode: 0x00000000aa0a3e8>
<environment: namespace:stats>
> methods(princomp)
[1] princomp.default* princomp.formula*
see '?methods' for accessing help and source code
> getAnywhere('princomp.default') # or stats:::princomp.default
A single object matching 'princomp.default' was found
...
function (x, cor = FALSE, scores = TRUE, covmat = NULL, subset = rep_len(TRUE,
    nrow(as.matrix(x))), ...)
{
...
  dn <- dim(z)
  if (dn[1L] < dn[2L])
    stop("'princomp' can only be used with more units than variables")
  covmat <- cov.wt(z)
...
<environment: namespace:stats>
>
> ?cov.wt
```



# eigen/cov/PCA of a data set when n<<p

```
> e.cor <- eigen(cor(x, use="pairwise.complete.obs"))
> e.cor$values
[1] 6.263117e+00 5.443693e+00 4.702980e+00 3.590210e+00
[5] 3.644515e-15 1.258361e-15 3.437109e-16 2.431887e-16
[9] 1.508096e-16 6.521571e-17 4.539563e-17 2.806356e-17
[13] 1.782844e-17 2.135321e-18 -9.004648e-18 -1.546332e-17
[17] -5.765008e-17 -6.563774e-17 -9.621985e-17 -5.282082e-16
> # e.cor$vectors
```

```
> pca.pkg <- c("FactoMineR", "factoextra", "corrplot")
> lapply(pca.pkg, library, character.only=TRUE)
> x.pca <- PCA(x, graph = FALSE) # works, why?
> PCA # check the code
function (X, scale.unit = TRUE, ncp = 5, ind.sup = NULL, quanti.sup = NULL,
...
  tmp <- svd.triplet(X, row.w = row.w, col.w = col.w, ncp = ncp)
  ...
<environment: namespace:FactoMineR>
> str(x.pca)
List of 5
 $ eig :'data.frame':   4 obs. of  3 variables:
...
 $ svd :List of 3
 ..$ vs: num [1:4] 2.5 2.33 2.17 1.89
 ..$ U : num [1:5, 1:4] 1 -1.256 -0.129 1.272 -0.888 ...
 ..$ V : num [1:20, 1:4] 0.1651 0.3414 -0.0531 -0.1638 0.2618 ...
...
 - attr(*, "class")= chr [1:2] "PCA" "list "
```

```
> get_eigenvalue(x.pca)
> x.pca$svd$V
```



## The breast tumour (gene expression) data of van't Veer et al

112/144

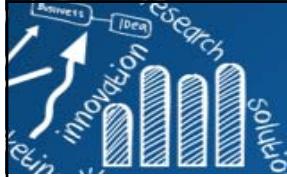
- The breast tumour (gene expression) data of van't Veer et al. (2002): 78 observations and 4,751 gene expressions, log 10 transformations.
- Typically, gene expression data contain intensity levels or expression indices of genes which are measured for a large number of genes
- In gene microarray experiments, the chips are the observations, and the genes are the variables.
- The data contain survival times in months as well as **binary responses regarding survival**.
- Patients who left the study or metastasised before the end of five years were grouped into the first class (**DM**), and those who survived five years formed the second class. Of the 78 patients, 44 survived the critical five years (**NODM**).



# PCA for The breast tumour data

```
> library(cancerdata)
> data(VEER1) # Breast cancer gene expression data (van't Veer)
> # load("VEER1.RData")
> VEER1
ExpressionSet (storageMode: lockedEnvironment)
assayData: 4948 features, 78 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: X1 X2 ... X79 (78 total)
  varLabels: info class
  varMetadata: labelDescription
featureData
  featureNames: J00129 Contig29982_RC ... AF067420 (4948 total)
  fvarLabels: symbol
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: Hu25K
> dim(exprs(VEER1))
[1] 4948 78
> str(VEER1)
Formal class 'ExpressionSet' [package "Biobase"] with 7 slots
..@ experimentData :Formal class 'MIAME' [package "Biobase"] with 13 slots
...
..@ featureData :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots
...
> exprs(VEER1)[1:3, 1:5] x1      x2      x3      x4      x5
J00129      -0.448 -0.480 -0.568 -0.819 -0.112
Contig29982_RC -0.296 -0.512 -0.411 -0.267 -0.670
Contig42854     -0.100 -0.031 -0.398  0.023  0.421
```

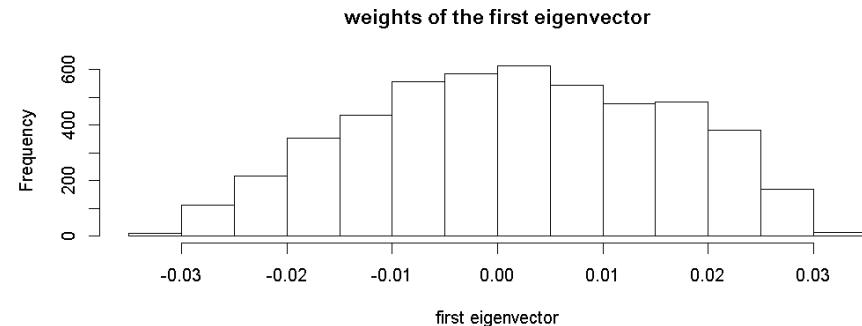
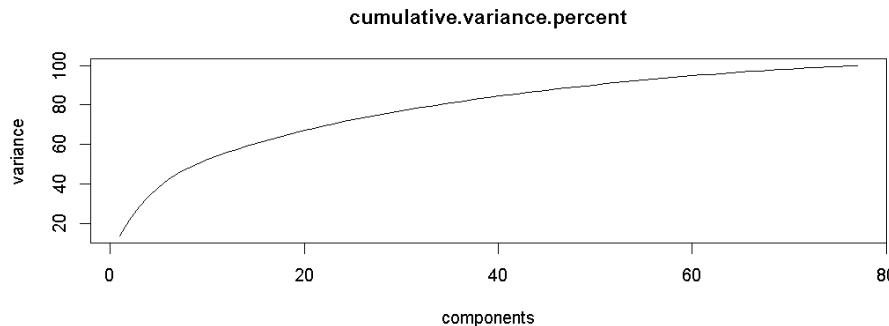
```
> VEER1@phenoData@data$info
[1] Sample.1..5.yr.survival.150...
...
97 Levels: Sample.1..5.yr.survival...
> table(VEER1@phenoData@data$class)
DM NODM
34   44
```

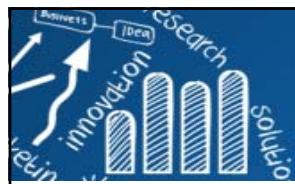


# PCA for The breast tumour data

```
> library(FactoMineR); library(factoextra); library(corrplot)
> x <- t(exprs(VEER1))
> x.pca <- PCA(x, graph = FALSE)
> x.pca
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 78 individuals, described by 4948 variables
*The results are available in the following objects:

      name           description
1  "$eig"         "eigenvalues"
2  "$var"         "results for the variables"
...
15 "$call$col.w" "weights for the variables"
> eig.val <- get_eigenvalue(x.pca)
> plot(eig.val[,3], type="l", xlab="components", ylab="variance",
+       main="cumulative.variance.percent")
> hist(x.pca$svd$V[,1], xlab="first eigenvector", main="weights of the first eigenvector")
```

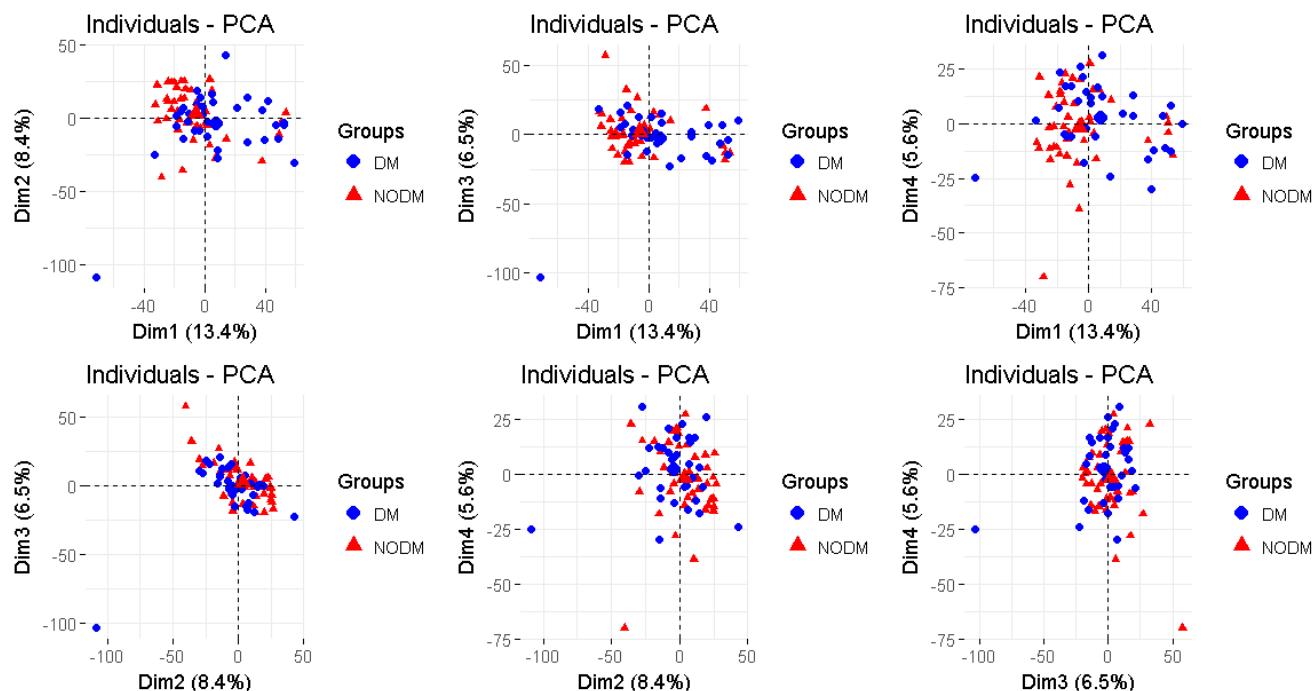




# PCA for The Breast Tumour Data

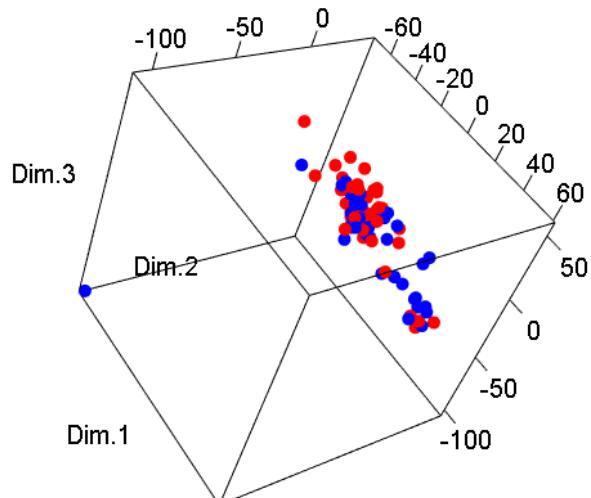
```
> x.pc <- rep(1:3, 3:1)
> y.pc <- unlist(lapply(2:4, function(x) seq(x, 4)))
>
> mypca.plot <- function(used.pc){
+   fviz_pca_ind(x.pc, axes = used.pc,
+                 geom.ind = "point",
+                 col.ind = VEER1@phenoData$data$class,
+                 palette = c("blue", "red"),
+                 legend.title = "Groups")
+ }
> plot.list <- apply(cbind(x.pc, y.pc), 1, mypca.plot)
```

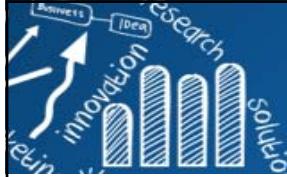
```
> library(gridGraphics)
> library(grid)
> grid.newpage()
> library(gridExtra)
> grid.arrange(grobs=plot.list,
+               nrow=2,
+               ncol=3)
```



# PCA for The Breast Tumour Data

```
> x.pca.scores <- get_pca_ind(x.pca)
> dim(x.pca.scores$coord)
[1] 78  5
> group.col <- c("blue", "red")[as.integer(VEER1@phenoData@data$class)]
> library("rgl")
> open3d()
> plot3d(x.pca.scores$coord[,1:3],
+         col=group.col,
+         type ="p", size=10)
> play3d(spin3d(), duration=10)
```





# Efficient Estimation of Covariance: a Shrinkage Approach

117/144

$$s_{ij} = \frac{1}{n-1} \sum_{k=1}^n (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j),$$

a shrinkage estimator  
 $\hat{\Sigma}_{\text{LW}} = \alpha_1 \mathbf{I} + \alpha_2 \mathbf{S}$ .

“Small  $n$ , Large  $p$ ”

Covariance and Correlation Estimators  $S^*$  and  $R^*$ :

$$s_{ij}^* = \begin{cases} s_{ii} & \text{if } i = j \\ r_{ij}^* \sqrt{s_{ii}s_{jj}} & \text{if } i \neq j \end{cases}$$

$$r_{ij}^* = \begin{cases} 1 & \text{if } i = j \\ r_{ij} \min(1, \max(0, 1 - \hat{\lambda}^*)) & \text{if } i \neq j \end{cases}$$

$$\text{with } \hat{\lambda}^* = \frac{\sum_{i \neq j} \widehat{\text{Var}}(r_{ij})}{\sum_{i \neq j} r_{ij}^2}$$

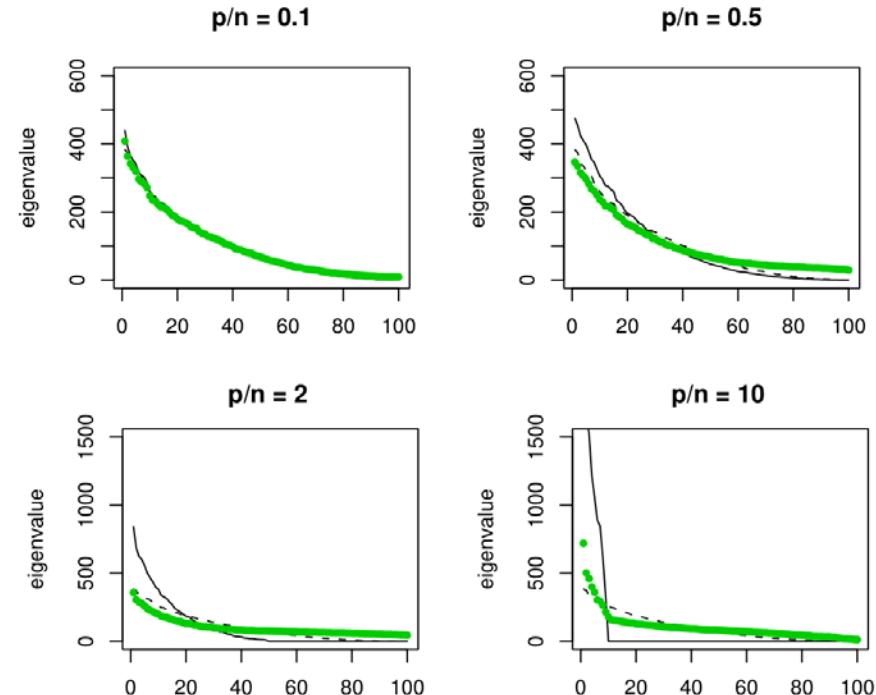


Figure 1: Ordered eigenvalues of the sample covariance matrix  $\mathbf{S}$  (thin black line) and that of an alternative estimator  $\mathbf{S}^*$  (fat green line, for definition see Tab. 1), calculated from simulated data with underlying  $p$ -variate normal distribution, for  $p = 100$  and various ratios  $p/n$ . The true eigenvalues are indicated by a thin black dashed line.

Schäfer, J., and K. Strimmer. 2005. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology* . 4: 32.

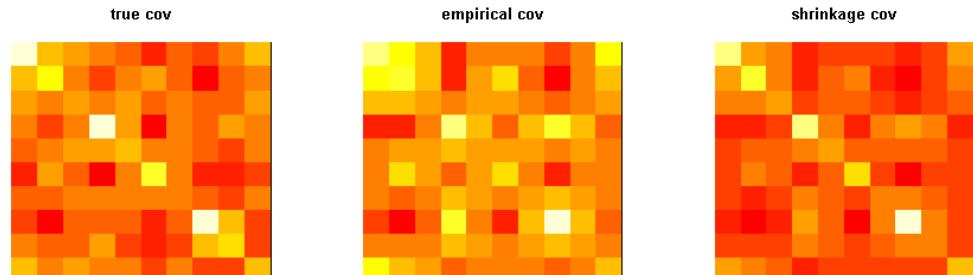


# Example Script from corpcor Package

```
> library("corpcor")
>
> n <- 6 # try 20, 500
> p <- 10 # try 100, 10
> set.seed(123456)
> # generate random pxp covariance matrix
> sigma <- matrix(rnorm(p * p), ncol = p)
> sigma <- crossprod(sigma) + diag(rep(0.1, p)) # t(x) %*% x
>
> # simulate multinormal data of sample size n
> sigsvd <- svd(sigma)
> y <- t(sigsvd$v %*% (t(sigsvd$u) * sqrt(sigsvd$d)))
> x <- matrix(rnorm(n * ncol(sigma)), nrow = n) %*% y # problem here! use mvrnorm {MASS}
> # x <- mvrnorm(n, mu=rep(0, p), Sigma=sigma)
> # estimate covariance matrix
> s1 <- cov(x)
> s2 <- cov.shrink(x)
Estimating optimal shrinkage intensity lambda.var (variance vector): 0.4378
Estimating optimal shrinkage intensity lambda (correlation matrix): 0.6494
> par(mfrow=c(1,3))
> image(t(sigma)[,p:1], main="true cov", xaxt="n", yaxt="n")
> image(t(s1)[,p:1], main="empirical cov", xaxt="n", yaxt="n")
> image(t(s2)[,p:1], main="shrinkage cov", xaxt="n", yaxt="n")
>
> # squared error
> sum((s1 - sigma) ^ 2)
[1] 4427.215
> sum((s2 - sigma) ^ 2)
[1] 850.2443
```

**mvrnorm {MASS}:**

Simulate from a Multivariate Normal Distribution  
`mvrnorm(n = 1, mu, Sigma, ...)`





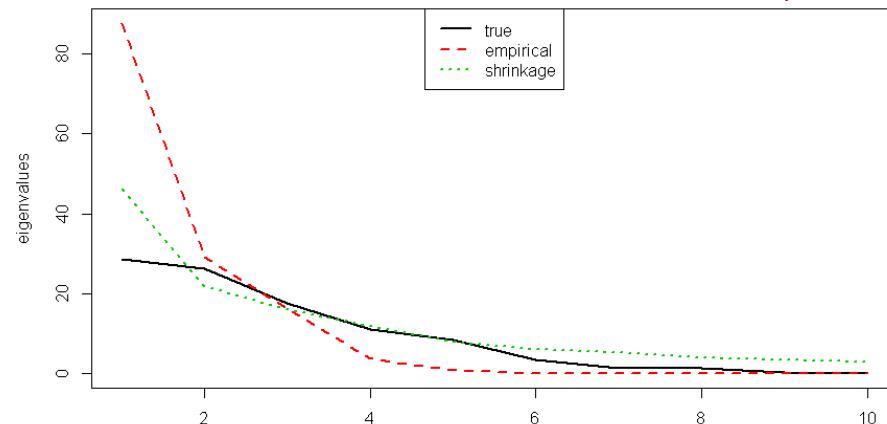
# Compare Eigenvalues

```
> # compare positive definiteness
> is.positive.definite(sigma)
[1] TRUE
> is.positive.definite(s1)
[1] FALSE
> is.positive.definite(s2)
[1] TRUE
>
> # compare ranks and condition
> rc <- rbind(
+   data.frame(rank.condition(sigma)),
+   data.frame(rank.condition(s1)),
+   data.frame(rank.condition(s2)))
> rownames(rc) <- c("true", "empirical", "shrinkage")
> rc
      rank condition      tol
true     10 256.35819 6.376444e-14
empirical 5      Inf 1.947290e-13
shrinkage 10 15.31643 1.022819e-13
>
>
>
> # compare eigenvalues
> e0 <- eigen(sigma, symmetric = TRUE)$values
> e1 <- eigen(s1, symmetric = TRUE)$values
> e2 <- eigen(s2, symmetric = TRUE)$values
>
> matplot(data.frame(e0, e1, e2), type = "l", ylab="eigenvalues", lwd=2)
> legend("top", legend=c("true", "empirical", "shrinkage"), lwd=2, lty=1:3, col=1:3)
```

## Shrinkage estimation of covariance matrix:

- `cov.shrink` {`corpcor`}
- `shrinkcovmat.identity` {`ShrinkCovMat`}
- `covEstimation` {`RiskPortfolios`}

Re-analyze "The breast tumour of van't Veer et al" data by a shrinkage method





# Canonical Correlation Analysis (CCA)<sup>120/144</sup>

- The main purpose of CCA (Hotelling, 1936) is the exploration of sample correlations between two sets of quantitative variables observed on the same experimental units.

$$X = [X_1, X_2, \dots, X_p]^T \quad Y = [Y_1, Y_2, \dots, Y_q]^T$$

$n \times p$

$n \times q$

assumed that  $p \leq q$

assumed that the columns of  $X$  and  $Y$  are standardized

Classical CCA assumes first  $p \leq n$  and  $q \leq n$

$$U_1 = \mathbf{a}_1^T X = a_{11}X_1 + a_{12}X_2 + \dots + a_{1p}X_p \quad V_1 = \mathbf{b}_1^T Y = b_{11}Y_1 + b_{12}Y_2 + \dots + b_{1q}Y_q$$

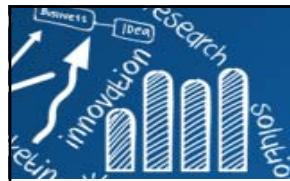
$$U_2 = \mathbf{a}_2^T X = a_{21}X_1 + a_{22}X_2 + \dots + a_{2p}X_p \quad V_2 = \mathbf{b}_2^T Y = b_{21}Y_1 + b_{22}Y_2 + \dots + b_{2q}Y_q$$

$\vdots \quad \vdots$

$\vdots \quad \vdots$

$$U_p = \mathbf{a}_p^T X = a_{p1}X_1 + a_{p2}X_2 + \dots + a_{pp}X_p \quad V_q = \mathbf{b}_q^T Y = b_{q1}Y_1 + b_{q2}Y_2 + \dots + b_{qq}Y_q$$

$$\rho_1 = \text{corr}(U_1, V_1) = \max \text{corr}(\mathbf{a}_1^T X, \mathbf{b}_1^T Y) \text{ subject to } \text{Var}(\mathbf{a}_1^T X) = \text{Var}(\mathbf{b}_1^T Y) = 1.$$



## Mathematical Aspects

$$\text{Corr}(U_i, V_i) = \frac{\text{Cov}(U_i, V_i)}{\sqrt{\text{Var}(U_i)} \sqrt{\text{Var}V_i}}, \quad i = 1, 2, \dots, r = \min(p, q).$$

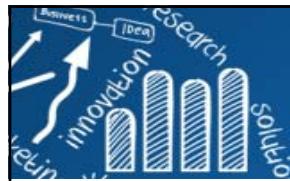
The first pair canonical variables is defined by  $\text{Corr}(U_1, V_1) = \rho_1$ .

- $\rho_1^2 \geq \rho_2^2 \geq \dots \geq \rho_r^2$ : the eigenvalues of the matrix  $\Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{XY}^T \Sigma_{XX}^{-1/2}$ .
- $\Sigma_{XX}$ : the variance-covariance of  $X$ .
- $\Sigma_{YY}$ : the variance-covariance of  $Y$ .
- $\Sigma_{XY}$ : the covariance matrix of the random vector  $X$  and  $Y$ .
- $\rho_1$ : the first canonical correlation (the square root of the largest eigenvalue).

$$\frac{\mathbf{a}' \boldsymbol{\Sigma}_{XY} \mathbf{b}}{\sqrt{\mathbf{a}' \boldsymbol{\Sigma}_X \mathbf{a}} \sqrt{\mathbf{b}' \boldsymbol{\Sigma}_Y \mathbf{b}}}$$

The second pair canonical variable is  $\text{Corr}(U_2, V_2) = \rho_2$  and so on.

$$\rho_1 = \text{corr}(U_1, V_1) = \max \text{corr}(\mathbf{a}_1^T X, \mathbf{b}_1^T Y) \text{ subject to } \text{Var}(\mathbf{a}_1^T X) = \text{Var}(\mathbf{b}_1^T Y) = 1.$$



## Mathematical Aspects

$$\Sigma_{XX} = \text{cov}(X, X) \quad \Sigma_{YY} = \text{cov}(Y, Y)$$

$$\text{maximize} \quad \rho = \frac{a' \Sigma_{XY} b}{\sqrt{a' \Sigma_{XX} a} \sqrt{b' \Sigma_{YY} b}}.$$

define

$$c = \Sigma_{XX}^{1/2} a,$$
$$d = \Sigma_{YY}^{1/2} b.$$

$$\rho = \frac{c' \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2} d}{\sqrt{c' c} \sqrt{d' d}}.$$

By the Cauchy–Schwarz inequality,  $|\langle \mathbf{u}, \mathbf{v} \rangle|^2 \leq \langle \mathbf{u}, \mathbf{u} \rangle \cdot \langle \mathbf{v}, \mathbf{v} \rangle$ ,

$$\left( c' \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2} \right) d \leq \left( c' \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2} \Sigma_{YY}^{-1/2} \Sigma_{YX} \Sigma_{XX}^{-1/2} c \right)^{1/2} (d' d)^{1/2},$$

$$\rho \leq \frac{\left( c' \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX} \Sigma_{XX}^{-1/2} c \right)^{1/2}}{(c' c)^{1/2}}.$$

[https://en.wikipedia.org/wiki/Canonical\\_correlation](https://en.wikipedia.org/wiki/Canonical_correlation)



## Mathematical Aspects

$$\rho \leq \frac{\left( c' \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX} \Sigma_{XX}^{-1/2} c \right)^{1/2}}{(c' c)^{1/2}}.$$

There is equality if the vectors  $d$  and  $\Sigma_{YY}^{-1/2} \Sigma_{YX} \Sigma_{XX}^{-1/2} c$  are collinear.

$c$  is an eigenvector of  $\Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX} \Sigma_{XX}^{-1/2}$

$d$  is an eigenvector of  $\Sigma_{YY}^{-1/2} \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XY} \Sigma_{YY}^{-1/2}$

- $a$  is an eigenvector of  $\Sigma_{XX}^{-1} \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX}$
- $b$  is an eigenvector of  $\Sigma_{YY}^{-1} \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XY}$

$$c = \Sigma_{XX}^{1/2} a,$$

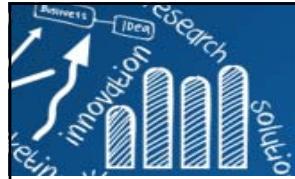
The canonical variables are defined by:

$$U = c' \Sigma_{XX}^{-1/2} X = a' X$$

$$d = \Sigma_{YY}^{1/2} b.$$

$$V = d' \Sigma_{YY}^{-1/2} Y = b' Y$$

[https://en.wikipedia.org/wiki/Canonical\\_correlation](https://en.wikipedia.org/wiki/Canonical_correlation)



# LifeCycleSavings {datasets}: 124/144

## Intercountry Life-Cycle Savings Ratio Data

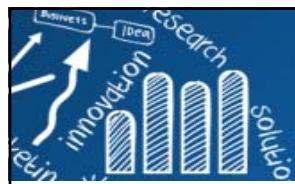
- Under the life-cycle savings hypothesis as developed by Franco Modigliani, the savings ratio (aggregate personal saving divided by disposable income (可支配收入)) is explained by per-capita disposable income (平均每人可支配所得), the percentage rate of change in per-capita disposable income, and two demographic variables: the percentage of population less than 15 years old and the percentage of the population over 75 years old. The data are averaged over the decade 1960–1970 to remove the business cycle or other short-term fluctuations.

A data frame with 50 observations on 5 variables.

```
[,1] sr : aggregate personal savings  
[,2] pop15: % of population under 15  
[,3] pop75: % of population over 75  
[,4] dpi: real per-capita disposable income  
[,5] ddpi: % growth rate of dpi
```

```
> head(LifeCycleSavings)  
      sr pop15 pop75      dpi ddpi  
Australia 11.43 29.35  2.87 2329.68 2.87  
Austria   12.07 23.32  4.41 1507.99 3.93  
Belgium   13.17 23.80  4.43 2108.47 3.82  
Bolivia    5.75 41.89  1.67 189.13 0.22  
Brazil    12.88 42.19  0.83  728.47 4.56  
Canada     8.79 31.72  2.85 2982.88 2.43  
> pop <- LifeCycleSavings[, 2:3]  
> oec <- LifeCycleSavings[, -(2:3)]
```

```
> cancor(pop, oec)  
$cor  
[1] 0.8247966112 0.3652761515  
  
$xcoef  
[ ,1] [ ,2]  
pop15 -0.009110856229 -0.03622206049  
pop75  0.048647513750 -0.26031158157  
  
$ycoef  
[ ,1] [ ,2] [ ,3]  
sr    0.00847102 0.0333793558 -0.00515712977  
dpi   0.00013073 -0.0000758823 0.00000454370  
ddpi  0.00417059 -0.0122678964 0.05188323606  
  
$xcenter  
pop15  pop75  
35.0896 2.2930  
  
$ycenter  
sr      dpi      ddpi  
9.6710 1106.7584 3.7576
```



## Regularized CCA

- CCA cannot be performed when the  $n \leq \max(p, q)$ .
- When the number of variables increases, greatest canonical correlations are nearly 1 because of recovering of canonical subspaces that do not provide any meaningful information.
- A standard condition usually advocated for CCA (Eaton and Perlman 1973) is  $n \geq p + q + 1$ .
- A regularization step in the data processing (Bickel and Li 2006) to perform a regularized canonical correlation analysis (RCCA).

$S_{XX}$  and  $S_{YY}$  are replaced respectively by  $\Sigma_{XX}(\lambda_1)$  and  $\Sigma_{YY}(\lambda_2)$  defined by

$$\Sigma_{XX}(\lambda_1) = S_{XX} + \lambda_1 I_p \quad \text{and} \quad \Sigma_{YY}(\lambda_2) = S_{YY} + \lambda_2 I_q.$$

How to get the "good" values for the regularization parameters?

Bickel PJ, Li B (2006). "Regularization in Statistics." Sociedad de Estadistica e Investigacion Operativa, Test, 15(2), 271–344.



# Cross-validation for tuning regularization parameters

126/144

- Denote  $\alpha = (\alpha_1, \alpha_2)$ . For a given value of  $\alpha$ , do  $i = 1, \dots, n$ :
  - $\rho_\alpha^{(-i)}$ : the first canonical correlation computed from the units with rows  $X^i$  and  $Y^i$  removed.
  - $\mathbf{a}_\alpha^{(-i)}$  and  $\mathbf{b}_\alpha^{(-i)}$ : the first canonical variates vectors.

- Obtain  $n$  pairs of vectors

$$(\mathbf{a}_\alpha^{(-1)}, \mathbf{b}_\alpha^{(-1)}), \dots, (\mathbf{a}_\alpha^{(-n)}, \mathbf{b}_\alpha^{(-n)}).$$

$$\Sigma_{XX}(\alpha_1) = S_{XX} + \alpha_1 \mathbf{I}_p$$

$$\Sigma_{YY}(\alpha_2) = S_{YY} + \alpha_2 \mathbf{I}_q$$

- The leave-one-out cross validation score for  $\alpha = (\alpha_1, \alpha_2)$  is then defined by Leurgans et al. (1993):

$$CV(\alpha_1, \alpha_2) = \text{corr}(\{(\mathbf{a}_\alpha^{(-i)})^T X^i\}_{i=1}^n, \{(\mathbf{b}_\alpha^{(-i)})^T Y^i\}_{i=1}^n).$$

- Choose the value of  $\alpha_1$  and  $\alpha_2$  that maximizes this correlation:

$$\hat{\alpha} = (\hat{\alpha}_1, \hat{\alpha}_2) = \arg \max_{\alpha_1, \alpha_2} CV(\alpha_1, \alpha_2).$$

Ignacio González, Sébastien Déjean, Pascal G. P. Martin, Alain Baccini, 2008, CCA: An R Package to Extend Canonical Correlation Analysis, Journal of Statistical Software, Vol 23, Issue 12.

- Note that  $\hat{\alpha}_1$  and  $\hat{\alpha}_2$  are chosen with respect to the first canonical variates and are then fixed for higher order canonical variates.

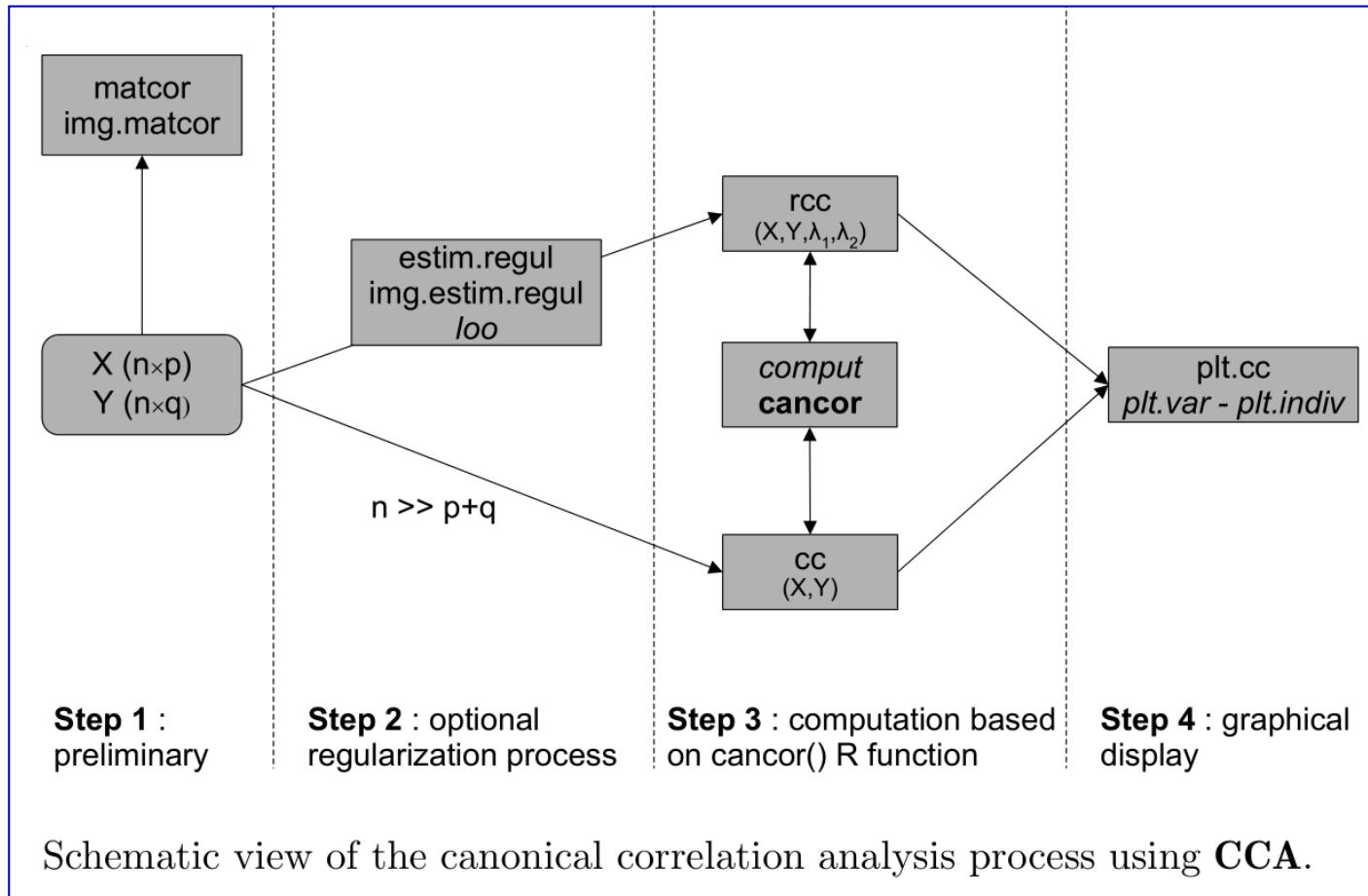
Two tuning parameters in the regularized CCA, so the CV is performed on a 2D surface.

- (1) Directly searching for a maximum on the 2D parameter surface. or
- (2) A relatively small grid of reasonable values for  $\alpha_1$  and  $\alpha_2$ .

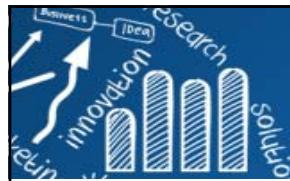


# CCA: An R Package to Extend Canonical Correlation Analysis

127/144



Ignacio González, Sébastien Déjean, Pascal G. P. Martin, Alain Baccini, 2008, CCA: An R Package to Extend Canonical Correlation Analysis, Journal of Statistical Software, Vol 23, Issue 12.



# a Nutrition Study in the Mouse

Two sets of variables were measured on 40 mice:

- **gene** (40 x 120): expressions of 120 genes measured in liver cells potentially involved in nutritional problems.
- **lipid** (40 x 21): concentrations of 21 hepatic fatty acids (肝脂肪酸).

The 40 mice were distributed in a 2-factors experimental design (4 replicates):

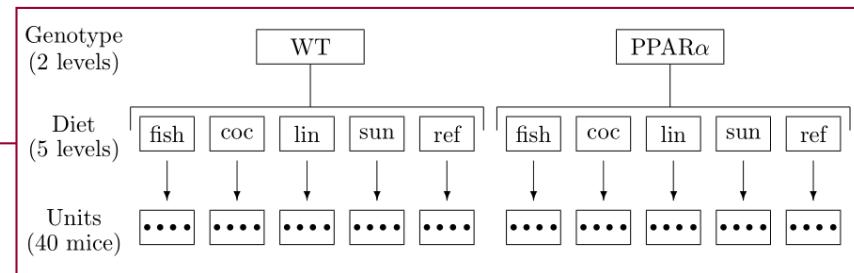
- genotype (2-levels factor): wild-type and PPARalpha -/-
- diet (5-levels factor): Oils used

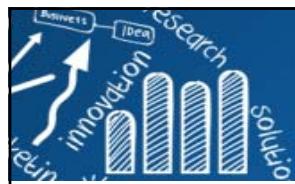
```
> library(CCA)
> data(nutrimouse)
> str(nutrimouse)

List of 4
 $ gene    :'data.frame':      40 obs. of  120 variables:
 ..$ X36b4    : num [1:40] -0.42 -0.44 -0.48 -0.45 -0.42 ...
 ...
 ..$ Tpbeta   : num [1:40] -1.11 -1.09 -1.14 -1.04 -1.2 ...
 ... [list output truncated]
 $ lipid    :'data.frame':      40 obs. of  21 variables:
 ..$ C14.0    : num [1:40] 0.34 0.38 0.36 0.22 0.37 ...
 ...
 ..$ C22.6n.3: num [1:40] 10.39 2.61 2.51 14.99 6.69 ...
 $ diet     : Factor w/ 5 levels "coc","fish","lin",...
 $ genotype: Factor w/ 2 levels "wt","ppar": 1 1 1 1 1 1 1 1 1 ...

>
> table(nutrimouse$genotype, nutrimouse$diet)

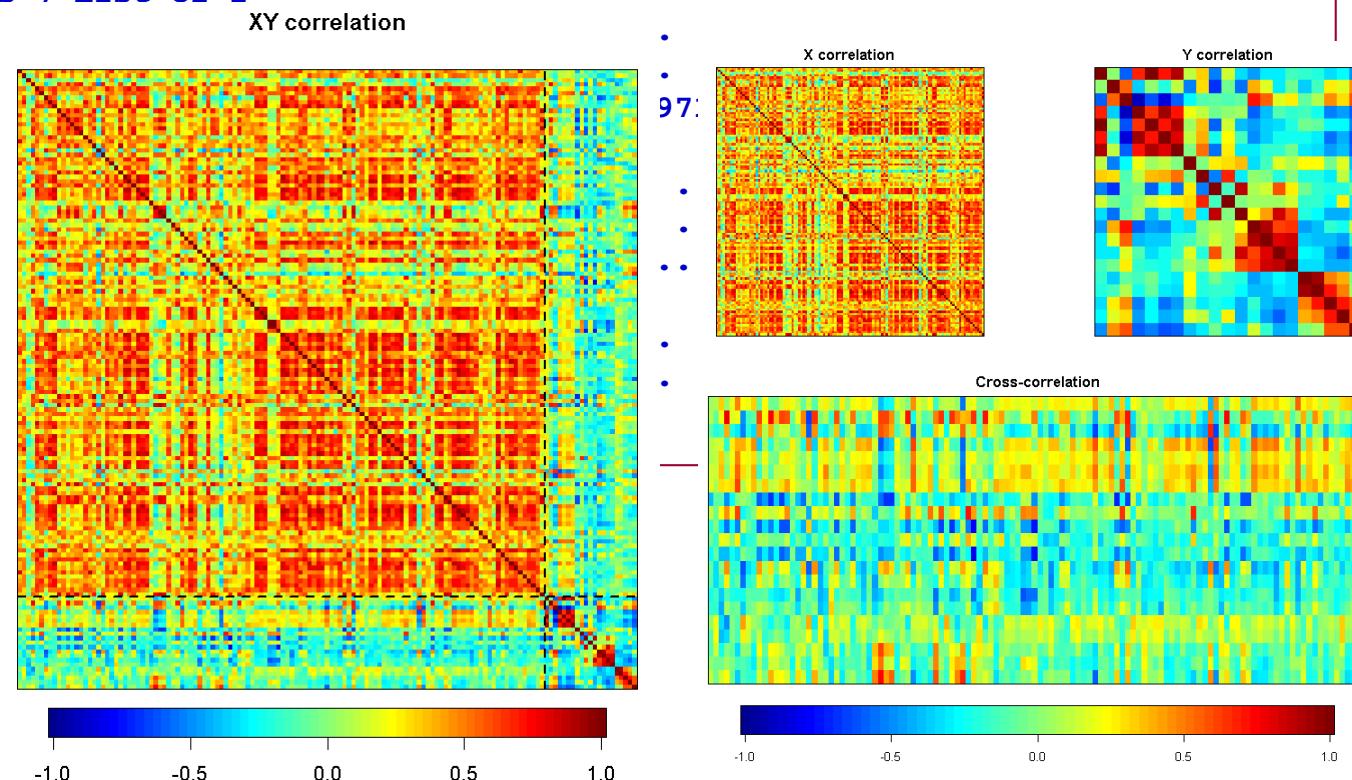
coc fish lin ref sun
wt    4    4    4    4    4
ppar  4    4    4    4    4
```

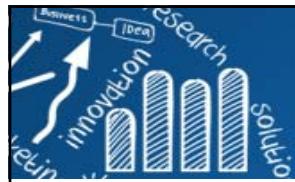




# Visualizing the Correlation Matrices

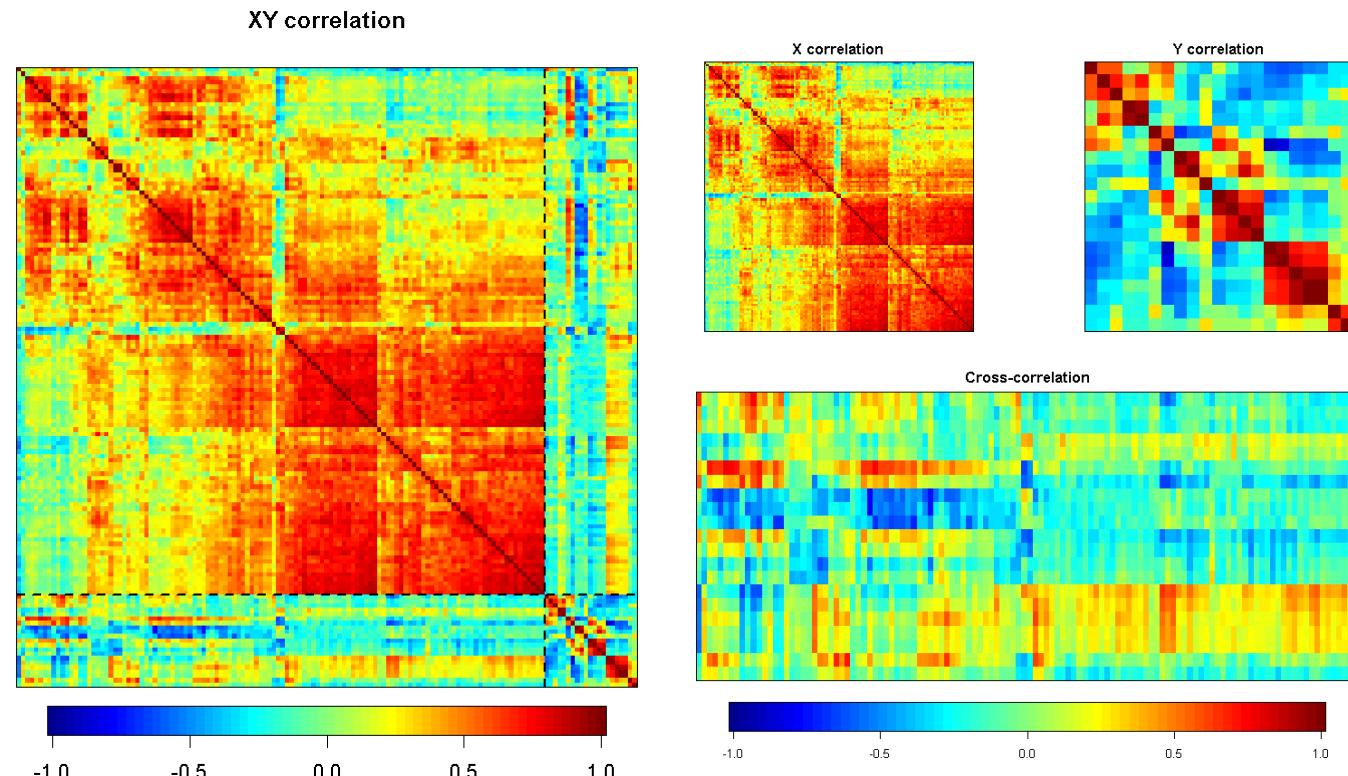
```
> # convert the data into the matrix format before performing CCA.  
> X <- as.matrix(nutrimouse$gene)  
> Y <- as.matrix(nutrimouse$lipid)  
> correl <- matcor(X, Y)  
> img.matcor(correl)  
> img.matcor(correl, type = 2)  
> str(correl)  
List of 3  
 $ Xcor : num [1:120, 1:120] 1 0.328 0.107 0.262 0.491 ...  
 ...- attr(*, "dimnames")=List of 2  
 ... ..$ : chr [1:120]  
 ... ..$ : chr [1:120]  
 $ Ycor : num [1:21, 1]  
 ...- attr(*, "dimname")  
 ... ..$ : chr [1:21]  
 ... ..$ : chr [1:21]  
 $ XYcor: num [1:141,  
 ...- attr(*, "dimname")  
 ... ..$ : chr [1:141]  
 ... ..$ : chr [1:141]
```





# Visualizing the Correlation Matrices

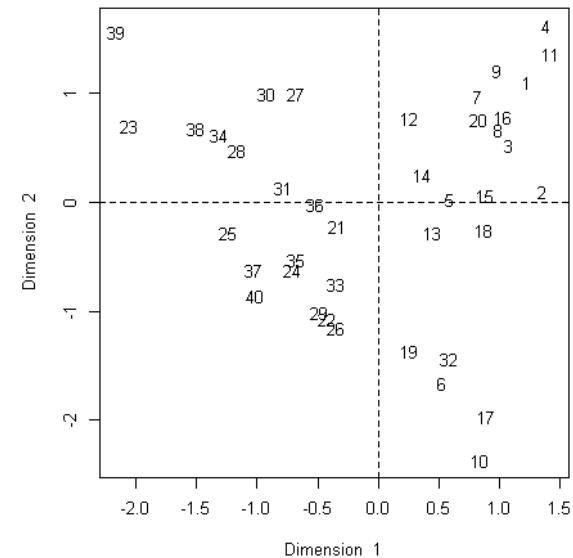
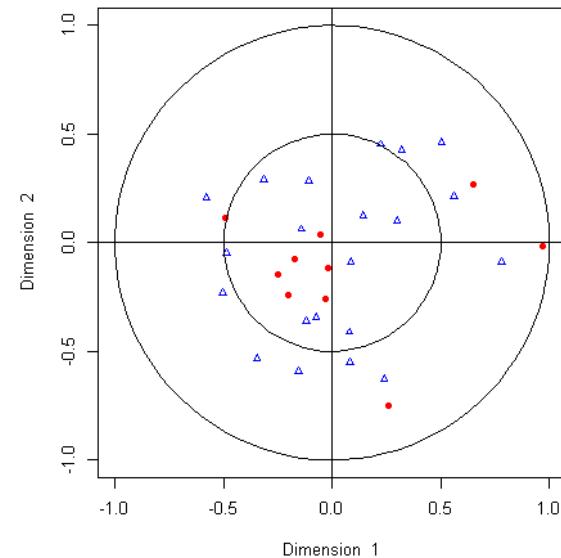
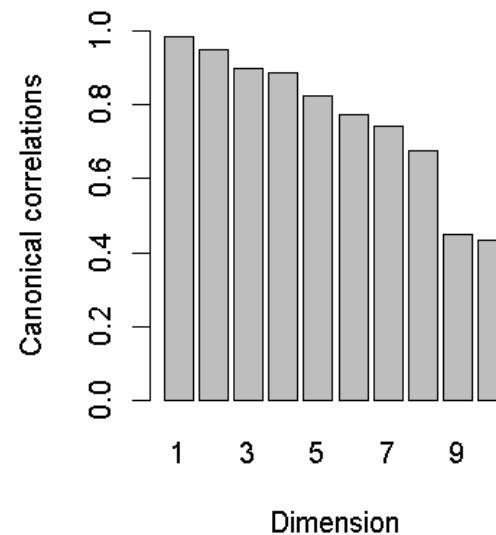
```
> x.hm <- heatmap(correl$Xcor)
> y.hm <- heatmap(correl$Ycor)
> ordering <- c(x.hm$rowInd, y.hm$rowInd + 120)
> correl.2 <- list(Xcor=correl$Xcor[x.hm$rowInd, x.hm$rowInd],
+                     Ycor=correl$Ycor[y.hm$rowInd, y.hm$rowInd],
+                     XYcor=correl$XYcor[ordering, ordering])
> img.matcor(correl.2)
> img.matcor(correl.2, type = 2)
```

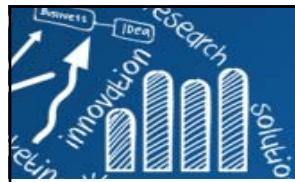




# Classical CCA

```
> X.subset <- as.matrix(nutrimouse$gene[, sample(1:120, size = 10)])
> my.cca <- cc(X.subset, Y)
> barplot(my.cca$cor, xlab="Dimension",
+         ylab="Canonical correlations",
+         names.arg=1:10, ylim=c(0,1))
> plt.cc(my.cca)
> names(my.cca)
[1] "cor"      "names"    "xcoef"   "ycoef"   "scores"
> my.cca$cor
[1] 0.9851407 0.9494411 0.8996597 0.8882256 0.8228657 0.7720856 0.7430616 0.6739105 0.4497435
[10] 0.4339681
```





# Regularized CCA

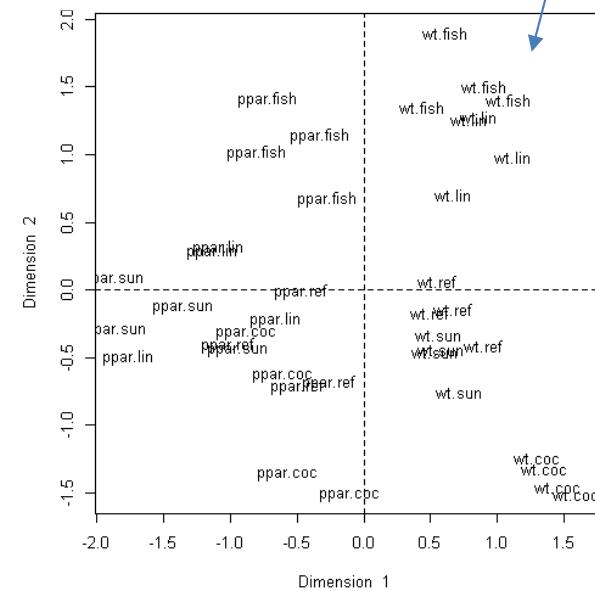
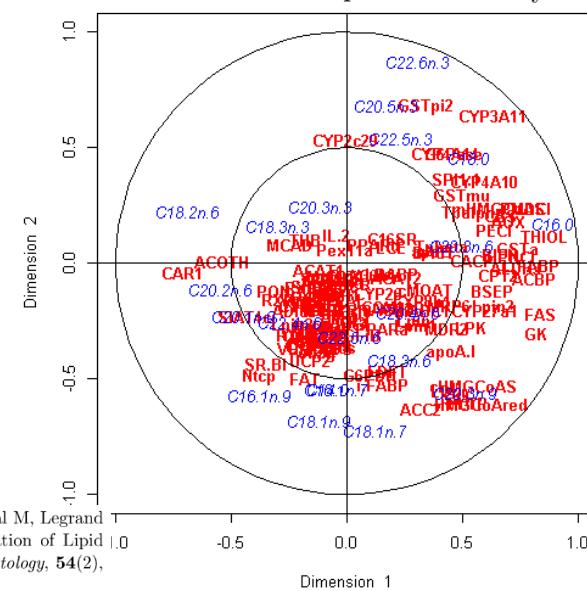
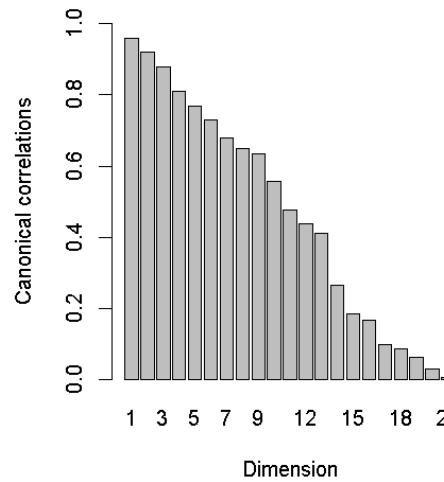
```

> regul.par <- estim.regul(X, Y, plt = TRUE,
+                             grid1 = seq(0.0001, 0.01, l=5), # l=50
+                             grid2 = seq(0, 0.1, l=5)) # l=50
+
lambda1 = 0.01
lambda2 = 0.075
CV-score = 0.884716
> my.rcca <- rcc(X, Y, regul.par$lambda1, regul.par$lambda2)
> names(my.rcca)
[1] "cor"      "names"    "xcoef"   "ycoef"   "scores"
> barplot(my.rcca$cor, xlab = "Dimension",
+           ylab = "Canonical correlations", names.arg = 1:21, ylim = c(0,1))
> plt.cc(my.rcca, var.label = TRUE,
+         ind.names = paste(nutrimouse$genotype, nutrimouse$diet, sep = "."))

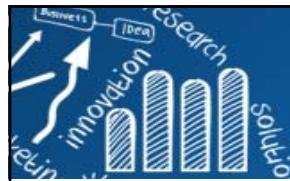
```

`> names(my.rcca$scores)`  
`[1] "xscores" "yscores" "corr.X.xscores"`  
`[4] "corr.Y.xscores" "corr.X.yscores" "corr.Y.yscores"`

my.rcca\$scores\$xscores[,1:2]



Martin PG, Guillou H, Lasserre F, Déjean S, Lan A, Pascussi JM, SanCristobal M, Legrand T, P. Besse P, Pineau T (2007). "Novel Aspects of PPAR $\alpha$ -mediated Regulation of Lipid and Xenobiotic Metabolism Revealed through a Nutrigenomic Study." *Hepatology*, 54(2), 767-777.

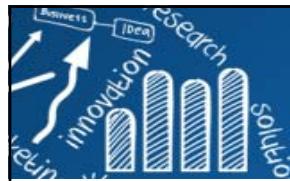


## plt.cc

```
> plt.cc
function (res, d1 = 1, d2 = 2, int = 0.5, type = "b", ind.names = NULL,
  var.label = FALSE, Xnames = NULL, Ynames = NULL)
{
  par(mfrow = c(1, 1), pty = "s")
  if (type == "v")
    plt.var(res, d1, d2, int, var.label, Xnames, Ynames)
  if (type == "i")
    plt.indiv(res, d1, d2, ind.names)
  if (type == "b") {
    def.par <- par(no.readonly = TRUE)
    layout(matrix(c(0, 0, 1, 2, 0, 0), ncol = 2, nrow = 3,
      byrow = TRUE), widths = 1, heights = c(0.1, 1, 0.1))
    par(pty = "s", mar = c(4, 4.5, 0, 1))
    plt.var(res, d1, d2, int, var.label, Xnames, Ynames)
    plt.indiv(res, d1, d2, ind.names)
    par(def.par)
  }
}
<environment: namespace:CCA>
```

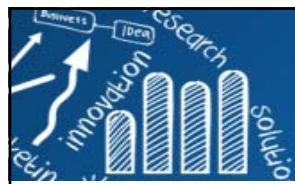
```
> plt.indiv
function (res, d1, d2, ind.names = NULL)
{
  if (is.null(ind.names))
    ind.names = res$names$ind.names
  if (is.null(ind.names))
    ind.names = 1:nrow(res$scores$xscores)
  plot(res$scores$xscores[, d1], res$scores$xscores[, d2],
    type = "n", main = "", xlab = paste("Dimension ", d1),
    ylab = paste("Dimension ", d2))
  text(res$scores$xscores[, d1], res$scores$xscores[, d2],
    ind.names)
  abline(v = 0, h = 0, lty = 2)
}
<environment: namespace:CCA>
```

```
> plt.var
function (res, d1, d2, int = 0.5, var.label = FALSE, Xnames = NULL,
  Ynames = NULL)
{
  if (!var.label) {
    plot(0, type = "n", xlim = c(-1, 1), ylim = c(-1, 1),
      xlab = paste("Dimension ", d1), ylab = paste("Dimension ",
        d2))
    points(res$scores$corr.X.xscores[, d1], res$scores$corr.X.xscores[, d2], pch = 20, cex = 1.2, col = "red")
    points(res$scores$corr.Y.xscores[, d1], res$scores$corr.Y.xscores[, d2], pch = 24, cex = 0.7, col = "blue")
  } else {
    if (is.null(Xnames))
      Xnames = res$names$Xnames
    if (is.null(Ynames))
      Ynames = res$names$Ynames
    plot(0, type = "n", xlim = c(-1, 1), ylim = c(-1, 1),
      xlab = paste("Dimension ", d1), ylab = paste("Dimension ",
        d2))
    text(res$scores$corr.X.xscores[, d1], res$scores$corr.X.xscores[, d2], Xnames, col = "red", font = 2)
    text(res$scores$corr.Y.xscores[, d1], res$scores$corr.Y.xscores[, d2], Ynames, col = "blue", font = 3)
  }
  abline(v = 0, h = 0)
  lines(cos(seq(0, 2 * pi, l = 100)), sin(seq(0, 2 * pi, l = 100)))
  lines(int * cos(seq(0, 2 * pi, l = 100)), int * sin(seq(0, 2 * pi, l = 100)))
}
<environment: namespace:CCA>
```



# Check Data and its Covariance

```
> library("corpcor")
> options(digits=4)
>
> test <- function(x, s){
+   image(t(s)[,nrow(s):1], main="cov(x)", col=terrain.colors(100))
+   image(t(x)[,nrow(x):1], main="x", col=terrain.colors(100))
+   cat("is.positive.definite:", is.positive.definite(s), "\n")
+   cat("eigen:\n")
+   print(eigen(s))
+   cat("inverse:\n")
+   print(solve(s))
+ }
>
> layout(matrix(1:2, ncol=1), height=c(1,2))
> # set 1: regular data
> n <- 100
> p <- 4
> x1 <- matrix(rnorm(n*p), ncol=p)
> summary(x1)
      V1          V2          V3          V4
Min. :-2.4964  Min. :-2.5184  Min. :-2.182  Min. :-2.9044
1st Qu.:-0.7516 1st Qu.:-0.5753 1st Qu.:-0.393 1st Qu.:-0.7751
Median : 0.0941 Median : 0.0726 Median : 0.221 Median :-0.0412
Mean   : 0.0382 Mean   : 0.1075 Mean   : 0.167 Mean   : 0.0181
3rd Qu.: 0.6850 3rd Qu.: 0.9046 3rd Qu.: 0.770 3rd Qu.: 0.7272
Max.   : 2.3993 Max.   : 2.3687 Max.   : 3.369 Max.   : 3.0475
```

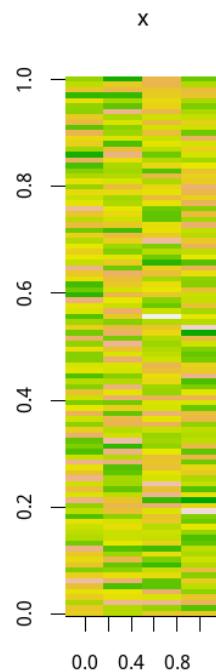
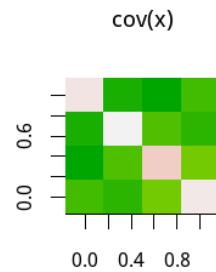


# Check Data and its Covariance

```
> s1 <- cov(x1)
> test(x1, s1)
is.positive.definite: TRUE
eigen:
$values
[1] 1.4157 1.2531 1.0118 0.7773

$vectors
 [,1]      [,2]      [,3]      [,4]
[1,]  0.6637   0.1010   0.4884   0.55746
[2,] -0.4384  -0.5896   0.6775   0.03517
[3,] -0.5707   0.3055  -0.1423   0.74879
[4,] -0.2040   0.7408   0.5313  -0.35681

inverse:
 [,1]      [,2]      [,3]      [,4]
[1,]  0.95483  0.09922  0.22535 -0.03536
[2,]  0.09922  0.86835 -0.02841  0.05419
[3,]  0.22535 -0.02841  1.04587 -0.15555
[4,] -0.03536  0.05419 -0.15555  0.91009
```





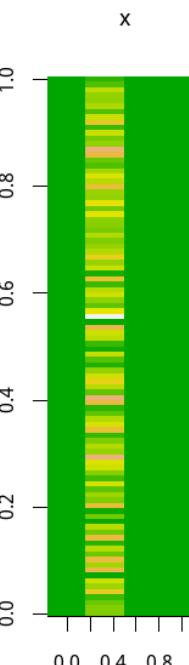
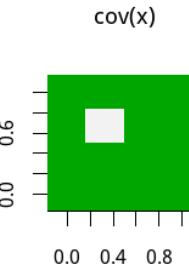
# Check Data and its Covariance

```
> # set 2: add some outlier variables
> x2 <- matrix(rnorm(n*p, sd=0.0001), ncol=p)
> id <- sample(1:p, floor(p/3))
> x2[, id] <- x2[, id] + abs(rnorm(n*length(id), m=10000, sd=5000))
> summary(x2)
      V1          V2          V3          V4
Min. :-0.00019325  Min. : 318  Min. :-0.0002198  Min. :-0.00026808
1st Qu.:-0.00004618 1st Qu.: 7273 1st Qu.:-0.0000667 1st Qu.:-0.00007213
Median :-0.00000283 Median : 9900 Median : 0.0000107 Median : 0.00000007
Mean   : 0.00000367 Mean   :10201  Mean   : 0.0000103 Mean   : 0.00000063
3rd Qu.: 0.00005736 3rd Qu.:12617 3rd Qu.: 0.0000998 3rd Qu.: 0.00007487
Max.   : 0.00021909 Max.   :27439  Max.   : 0.0003177 Max.   : 0.00027794

> s2 <- cov(x2)
> test(x2, s2)
is.positive.definite: FALSE
eigen:
$values
[1] 25301262.378715548664          0.000000012360          0.000000010255          0.000000007155

$vectors
[,1]          [,2]          [,3]          [,4]
[1,] -0.0000000006300 -0.2198484423160 -0.123788454460  0.9676482216971
[2,] -1.0000000000000 -0.0000000004066  0.000000004536 -0.0000000001631
[3,] -0.0000000002447 -0.9600601615329 -0.148515578779 -0.2371236156468
[4,]  0.0000000045072 -0.1730640015964  0.981131765566  0.0861934449310

inverse:
[,1]          [,2]          [,3]          [,4]
[1,] 133713761.94777 -0.06915255563 -12572701.82036  2664291.0791
[2,]      -0.06915  0.00000004155       -0.02882        0.4378
[3,] -12572701.82036 -0.02881648352  84429162.38724 -3566776.2275
[4,]  2664291.07907  0.43778259944 -3566776.22748 97309029.4158
```



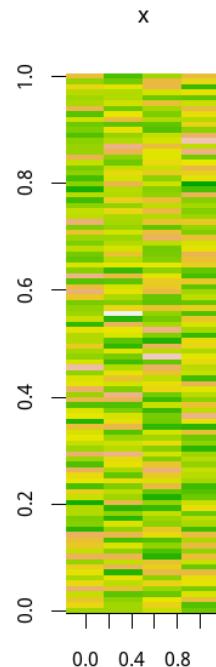
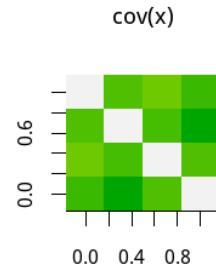


# Check Data and its Covariance

```
> # set 3: do the scaling
> x3 <- scale(x2)
> s3 <- cov(x3)
> test(x3, s3)
is.positive.definite: TRUE
eigen:
$values
[1] 1.2270 1.1165 0.8797 0.7768

$vectors
      [,1]     [,2]     [,3]     [,4]
[1,] 0.21350  0.66557  0.71239  0.06273
[2,] 0.69471 -0.06814 -0.20495  0.68610
[3,] 0.04397  0.71997 -0.67057 -0.17333
[4,] -0.68547  0.18442 -0.02884  0.70377

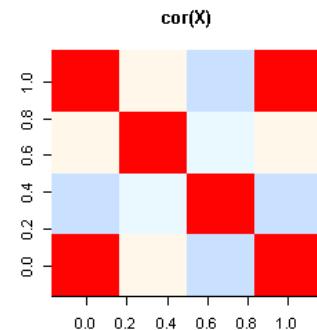
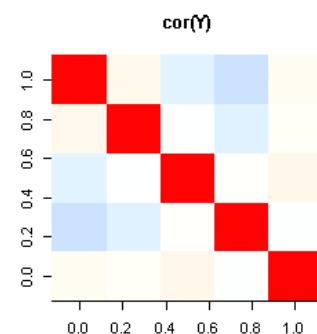
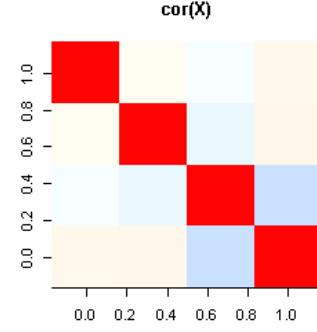
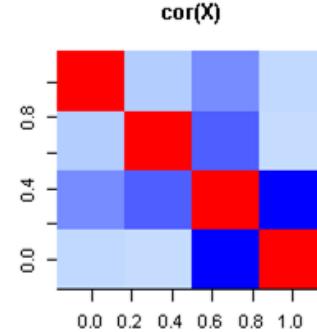
inverse:
      [,1]     [,2]     [,3]     [,4]
[1,] 1.01590 -0.03032 -0.12020  0.02415
[2,] -0.03032  1.05120 -0.01590  0.22895
[3,] -0.12020 -0.01590  1.01571 -0.04068
[4,]  0.02415  0.22895 -0.04068  1.05192
```

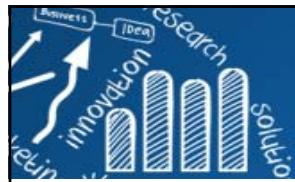




# Some Columns are Linear Dependent

```
> library(fields); library("corpcor"); library(CCA)
> cor.col <- two.colors(start="blue", middle="white",
                           end="red") # length=255
> par(mfrow=c(3,1))
> n <- 100
> p <- 4
> q <- 5
> set.seed(12345)
> X <- matrix(rnorm(n*p), ncol=p); rX <- cor(X)
> range.col <- floor((1+range(rX))*127+1)
> # wrong: image(t(cor(Y))[,q:1], main="cor(Y)", col=cor.col)
> image(t(rY)[,p:1], main="cor(X)", col=cor.col[range.col[1]: range.col[2]])
> is.positive.definite(cov(X))
[1] TRUE
> Y <- matrix(rnorm(n*q), ncol=q); rY <- cor(Y)
> range.col <- floor((1+range(rY))*127+1)
> image(t(rY)[,q:1], main="cor(Y)", col=cor.col[range.col[1]: range.col[2]])
> is.positive.definite(cov(Y))
[1] TRUE
> xy.cca <- cc(X, Y)
>
> X[,1] <- 3*X[,4] + 1; rX <- cor(X)
> range.col <- floor((1+range(rX))*127+1)
> image(t(rX)[,p:1], main="cor(X)", col=cor.col[range.col[1]: range.col[2]])
> xy.cca <- cc(X, Y)
Error in chol.default(Bmat) :
  the leading minor of order 4 is not positive definite
> is.positive.definite(cov(X))
[1] FALSE
```





## DR Quality Assessment: local continuity meta-criterion

139/144

$$\mathcal{N}_{K'}^D(i) = \{j_1, \dots, j_{K'}\} \quad K'\text{-NNs with regard to } D_{i,j}$$

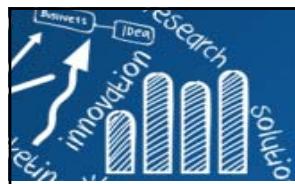
$$\mathcal{N}_{K'}^X(i) = \{k_1, \dots, k_{K'}\} \quad K'\text{-NNs with regard to } \|\mathbf{x}_i - \mathbf{x}_k\| \text{ (excluding } i\text{).}$$

$$N_{K'}(i) = |\mathcal{N}_{K'}^D(i) \cap \mathcal{N}_{K'}^X(i)|, \quad N_{K'} = \frac{1}{N} \sum_{i=1}^N N_{K'}(i).$$

normalize overlap to the [0, 1] interval:  $M_{K'} = \frac{1}{K'} N_{K'}$

*adjusted LC meta-criteria* (Chen 2006)  $M_{K'}^{\text{adj}} = M_{K'} - \frac{K'}{N-1}$ .

Lee, J.A., Lee, J.A., Verleysen, M., 2009. Quality assessment of dimensionality reduction: Rank-based criteria. Neurocomputing 72.  
Chen, L., Buja, A., 2009. Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Layout and Proximity Analysis , JASA 104(485), 209-219.



# Co-ranking Matrix and LCMC

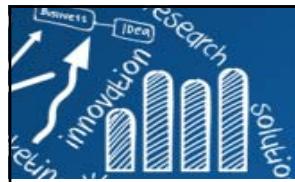
```
> LCMC
function (Q, K = 1:nrow(Q))
{
...
  nQ <- nrow(Q)
  nK <- length(K)
  N <- nQ + 1
  if (nK < 0.2 * nQ) {
    lcmc <- numeric(nK)
    for (i in 1:nK) {
      k <- K[i]
      lcmc[i] <- k/(1 - N) + sum(Q[cm.UL_K(k, nQ)]) / N / k
    }
  } else {
    lcmc_ <- diag(apply(apply(Q, 2, cumsum), 1, cumsum)) / (1:nQ) / N - (1:nQ) / nQ
    lcmc <- lcmc_[K]
  }
  lcmc
}
<environment: namespace:coRanking>
```

```
my.coranking <- function(X.high, X.low) {
  # X.high <- iris[1:10, 1:4]
  # X.low <- princomp(X.high)$score[, 1:2]

  D.high <- as.matrix(dist(X.high))
  D.low <- as.matrix(dist(X.low))
  f <- function(x){
    rank(x, ties.method = 'first', na.last = FALSE)
  }
  diag(D.high) <- NA # NA is rank 1
  diag(D.low) <- NA
  R.high <- apply(D.high, 1, f)
  R.low <- apply(D.low, 1, f)
  table(R.high, R.low)[-1, -1]
```

The *co-ranking matrix* [23] can then be defined as

$$\mathbf{Q} = [q_{kl}]_{1 \leq k, l \leq N-1} \quad \text{with} \quad q_{kl} = |\{(i, j) : \rho_{ij} = k \text{ and } r_{ij} = l\}|.$$



# coRanking: Co-Ranking Matrix

Calculate the co-ranking matrix to assess the quality of a dimensionality reduction.

```
coranking(Xi, X, input = c("data", "dist", "rank"), use = "C")
```

## Arguments

**Xi**: high dimensional data

**X**: low dimensional data

**input**: type of input

Plots the co-ranking matrix nicely

```
imageplot(Q, lwd = 2, bty = "n", main = "co-ranking matrix",
          xlab = expression(R), ylab = expression(Ro),
          col = colorRampPalette(colors = c("gray85", "red", "yellow",
          "green", "blue"))(100), axes = FALSE, legend = TRUE, ...)
```

Calculate the local continuity meta-criterion from a co-ranking matrix.

```
LCMC(Q, K = 1:nrow(Q))
```

## Arguments

**Q**: a co-ranking matrix

**K**: vector of integers describing neighborhood size

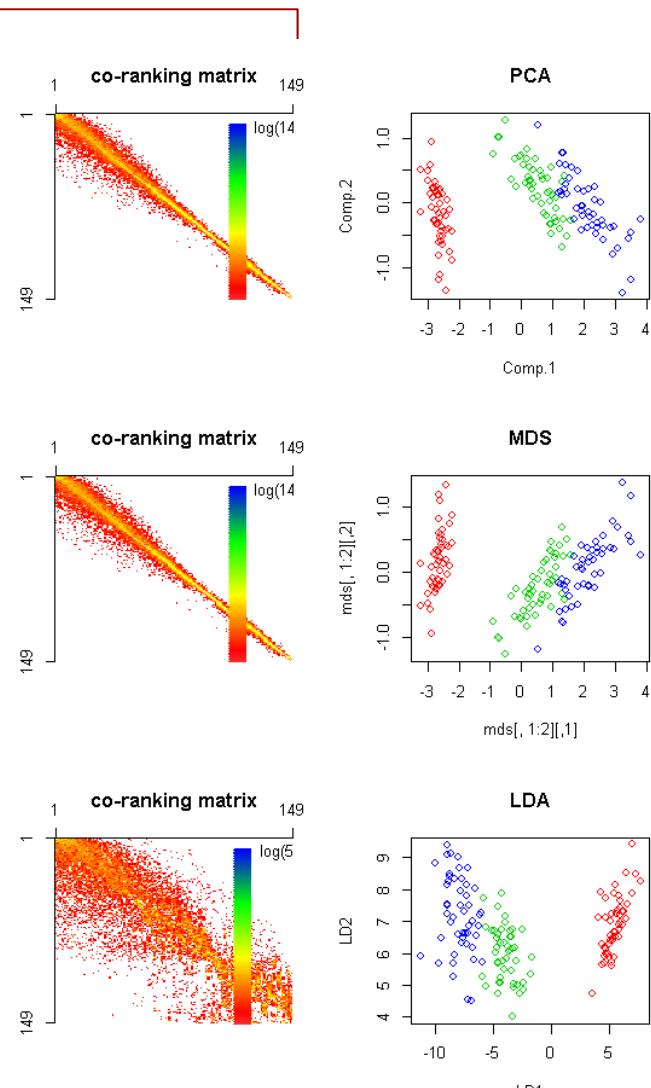


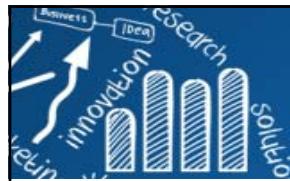
# Example

```

> library(coRanking) # install.packages("coRanking")
> library(MASS)
> par(mfrow=c(2, 3))
> x <- iris[,1:4]
> y <- iris[,5]
> pca <- princomp(x)
> Q.pca <- coranking(x, pca$score[,1:2], input = "data")
> imageplot(Q.pca)
> lcmc.pca <- LCMC(Q.pca, K = 5:10)
>
> mds <- cmdscale(dist(x))
> Q.mds <- coranking(x, mds[,1:2], input = "data")
> imageplot(Q.mds)
> lcmc.mds <- LCMC(Q.pca, K = 5:10)
>
> mylda <- lda(x, grouping=y)
> lda.dim <- as.matrix(x) %*% mylda$scaling[,1:2]
> Q.lda <- coranking(x, lda.dim, input = "data")
> imageplot(Q.lda)
> lcmc.lda <- LCMC(Q.lda, K = 5:10)
>
> names(lcmc.pca) <- paste0("K=", 5:10)
> rbind(lcmc.pca, lcmc.mds, lcmc.lda)
      K=5        K=6        K=7        K=8        K=9
lcmc.pca 0.6077763 0.6108427 0.6292106 0.6521421 0.6581158
lcmc.mds 0.6077763 0.6108427 0.6292106 0.6521421 0.6581158
lcmc.lda 0.3171096 0.3286204 0.3396868 0.3638087 0.3781158
> plot(pca$score[,1:2], col=as.integer(y)+1, main="PCA")
> plot(mds[,1:2], col=as.integer(y)+1, main="MDS")
> plot(lda.dim[,1:2], col=as.integer(y)+1, main="LDA")

```





# Evaluate DR by Classification (n>>p)

## UCI Datasets

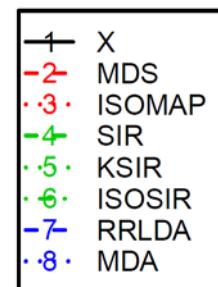
143/144

DR as a tool of **feature extraction**.

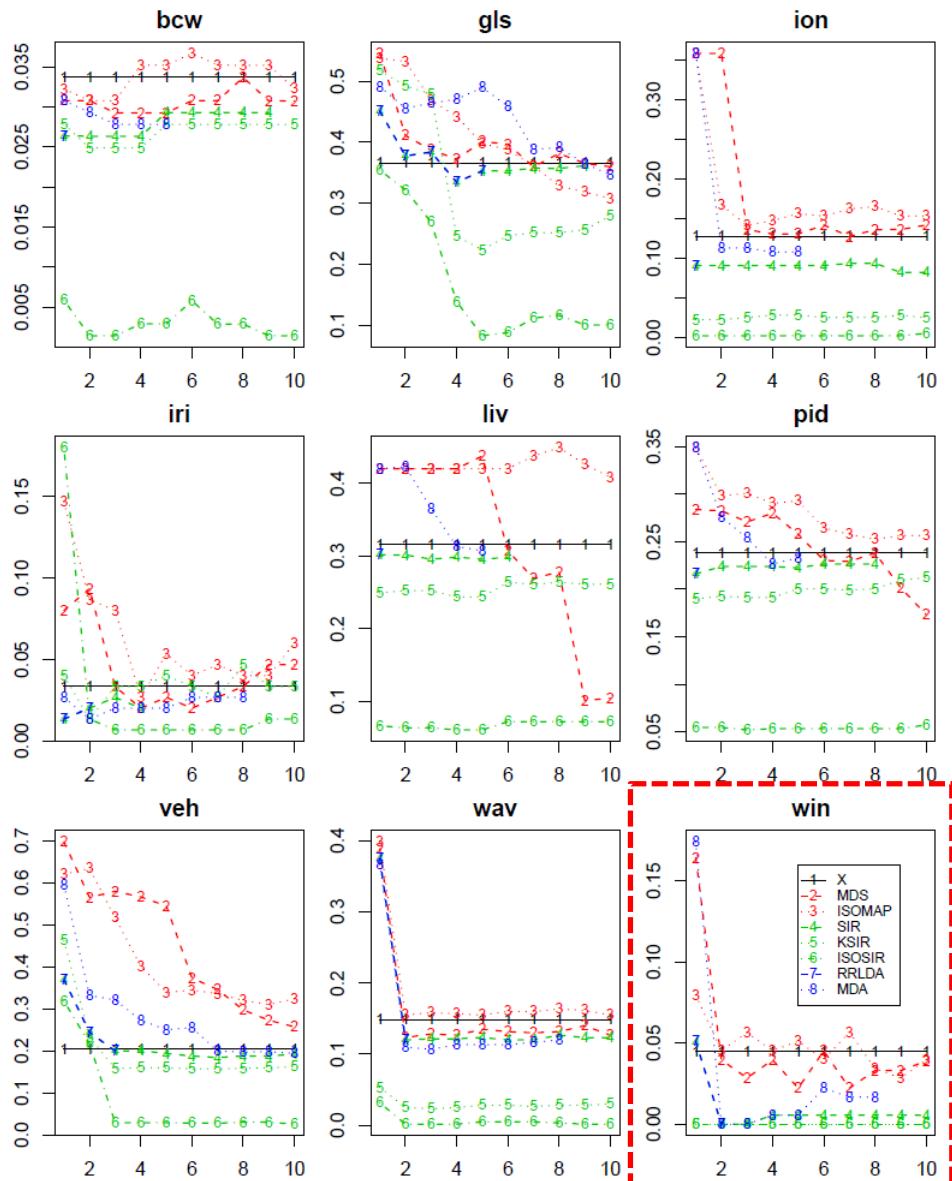
UCI machine learning repository datasets

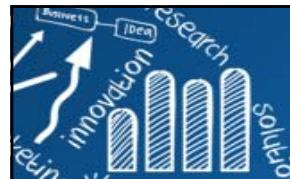
Data set	$n$	$p$	$G(n_h)$
Wisconsin breast cancer (bcw)	683	9	2 (444, 239)
Glass identification (gls)	214	9	6 (70, 76, 17, 13, 9, 29)
Ionosphere (ion)	351	33	2 (225, 126)
Iris plants (iri)	150	4	3 (50×3)
BUPA liver disorders (liv)	345	6	2 (145, 200)
Pima Indians diabetes (pid)	768	8	2 (500, 268)
StatLog vehicle silhouettes (veh)	846	18	4 (212, 217, 218, 199)
Waveform database generator (wav)	600	21	3 (200×3)
Wine recognition data (win)	178	13	3 (59, 71, 48 )

- Classifier : linear SVM
- 10-fold cross-validation error rate
- Gaussian kernel with scale 0.05



See 105/138, 106/138,  
107/138





# Evaluate DR by Classification (n<<p) Microarray Datasets

144/144

Datasets	Publication	$n$	$p$	$G(n_h)$	Response
Brain	Pomeroy et al. 2002	42	5597	5(10, 10, 10, 4, 8)	Different tumor types
Colon	Alon et al. 1999	62	2000	2(22, 40)	Tumor/normal tissue
Leukemia	Golub et al. 1999	72	3571	2(47, 25)	Subtypes of leukemia
Lymphoma	Alizadeh et al. 2000	62	4026	3(42, 9, 11)	Subtypes of lymphoma
Prostate	Singh et al. 2002	102	6033	2(50, 52)	Tumor/normal tissue
SRBCT	Khan et al. 2001	63	2308	4(23, 20, 12, 8)	Different tumor types

$n \times p; n << p$

$$\Sigma_{XX} \rightarrow p \times p$$

$$\text{IsoDistance} \rightarrow n \times n$$

- Classifier : linear SVM
- leave-one-out cross-validation error rate
- Gaussian kernel with scale 0.05

