

Lecture 2c

Latent variable models

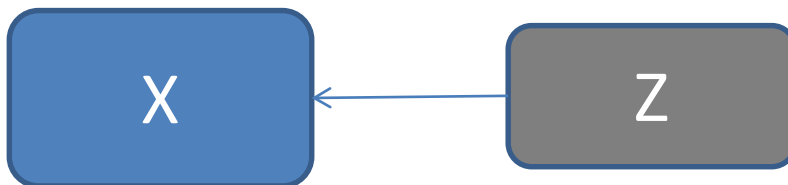


Overview

- Principal Component Analysis (PCA)
- Probabilistic PCA
- Independent component analysis (ICA)

Latent variables

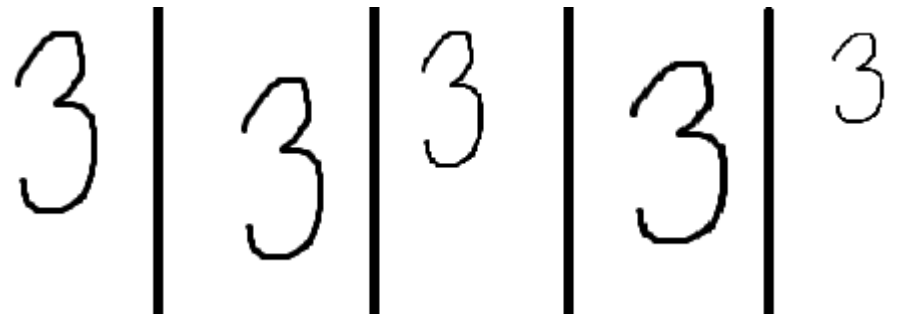
- Sometimes data depends on the variables we can not measure (hard to measure)
 - Answers on the test depend on Intelligence
 - Brain activity in the brain is measured by sensors
 - Stock prices depend on market confidence



Source: Leadliaison.com

Latent variables

- Latent factor discovered → data storage may decrease a lot



- Latent factors
 - Center
 - Scaling
- Original vs compressed
 - $100 \times 100 \times 5 = 50000$
 - $100 \times 100 + 2 \times 5 + 2 \times 5 = 10020$



Principal Component Analysis (PCA)

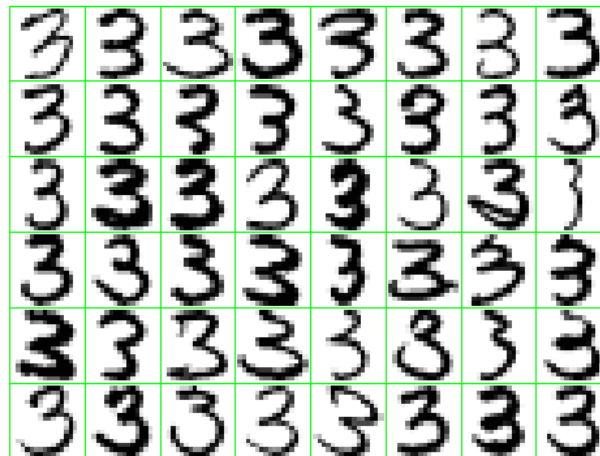
- *PCA* is a technique for reducing the complexity of high dimensional data
- It can be used to approximate high dimensional data with a few dimensions (latent features) → much less data to store
- New variables might have a special interpretation

Applications

- Image recognition
- Information compression
- Subspace clustering
- ...

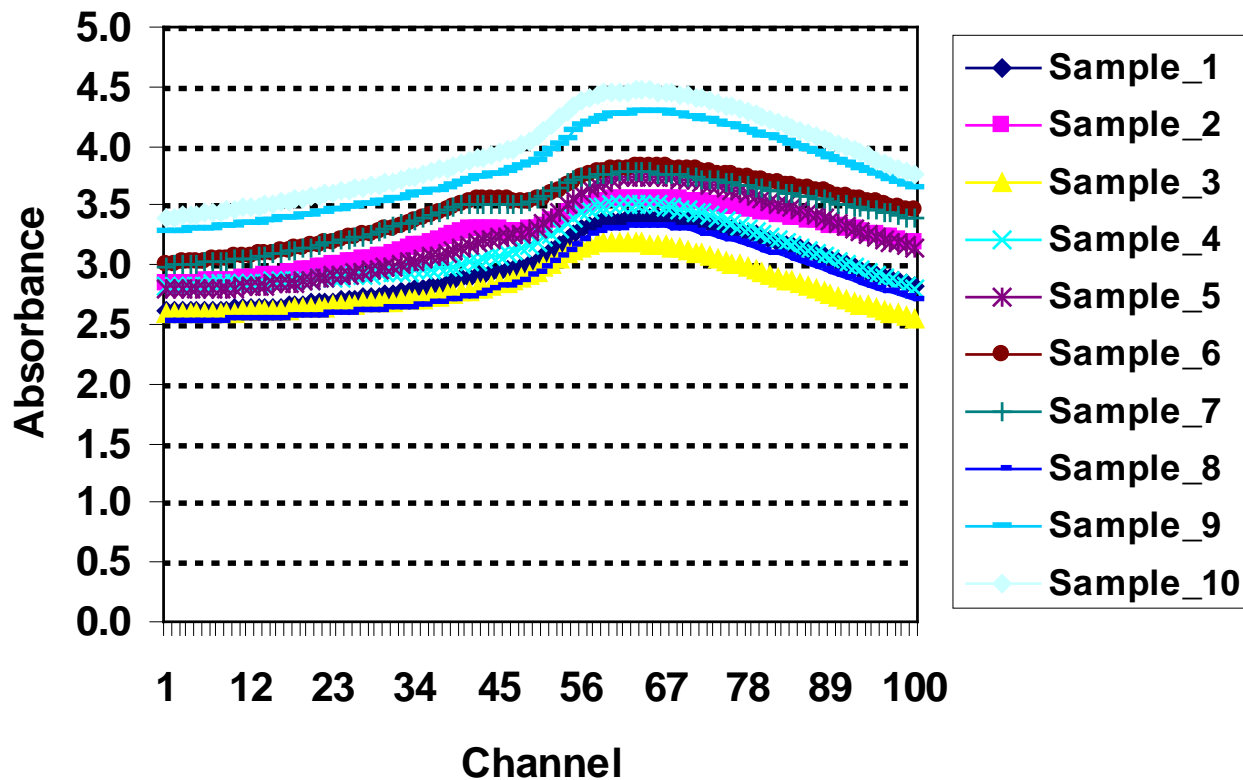
Principal Component Analysis (PCA)

- Example 1: Handwritten digits
 - Can we get a more compact summary?



Absorbance records for ten samples of chopped meat

Parallel coordinate plot for “FAT”



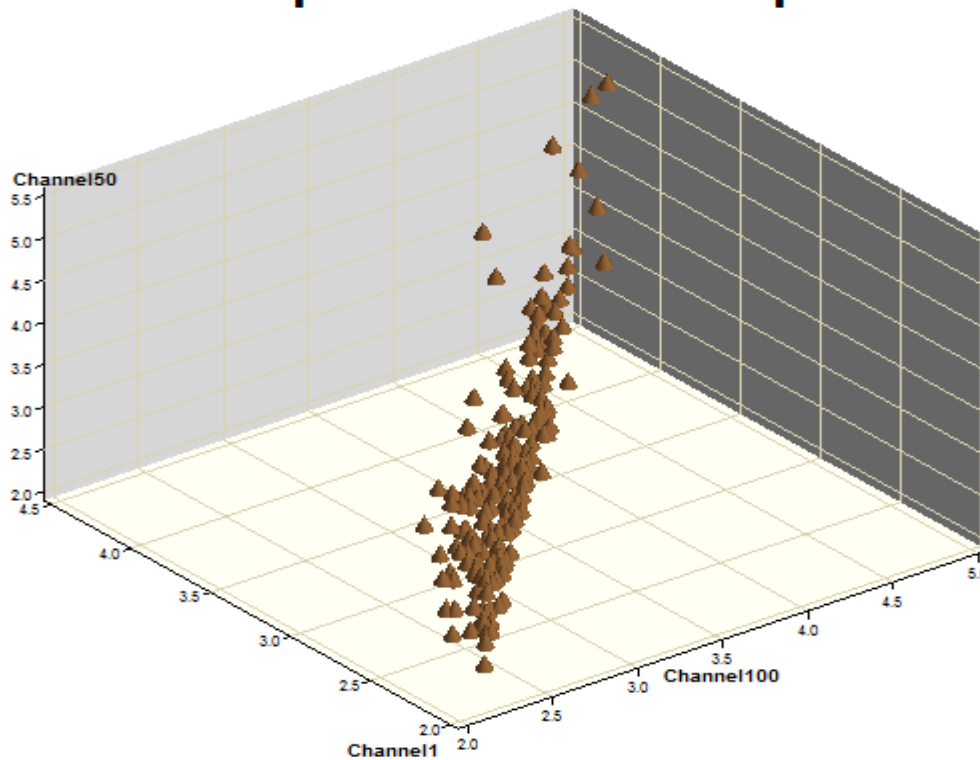
1 target (fat)

100 features
(absorbance at 100
wavelengths or
channels)

The features are
strongly correlated
to each other

3-D plots of absorbance records for samples of meat - channels 1, 50 and 100

Scatterplot for three components



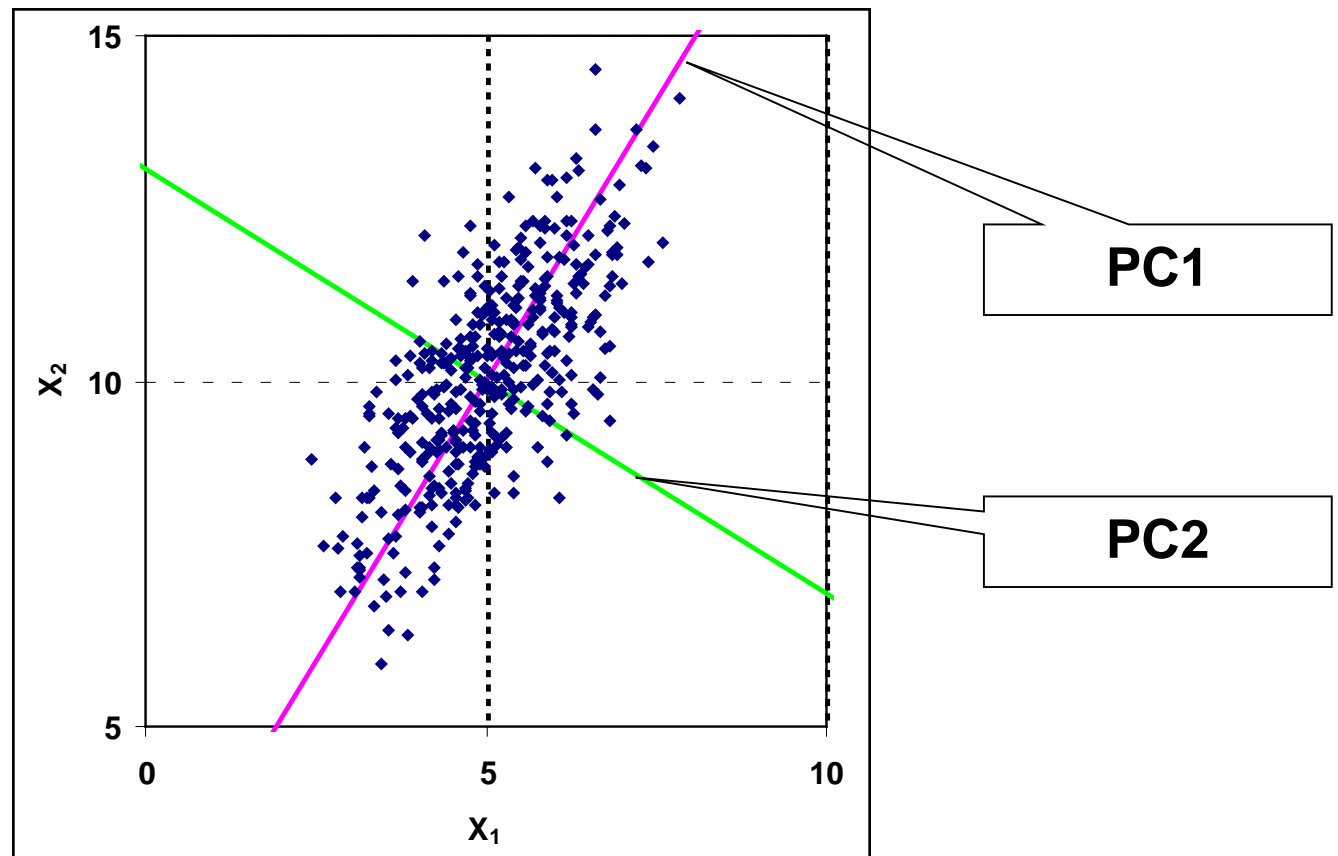
Principal components analysis

Idea: Introduce a new coordinate system (PC1, PC2, ...) where

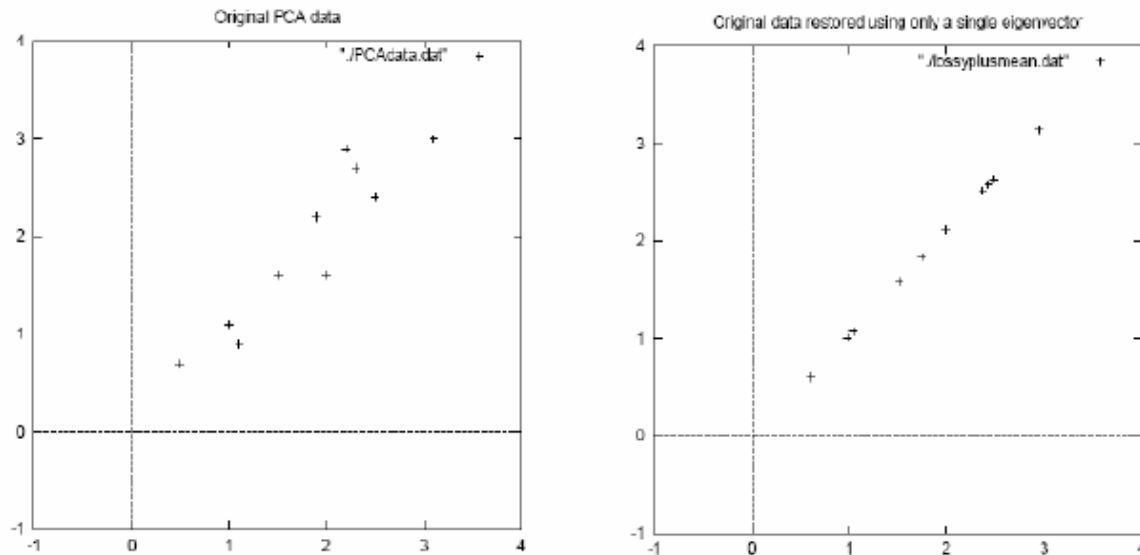
- The first principal component (PC1) is the direction that maximizes the variance of the projected data
- The second principal component (PC2) is the direction that maximizes the variance of the projected data after the variation along PC1 has been removed
- The third principal component (PC3) is the direction that maximizes the variance of the projected data after the variation along PC1 and PC2 has been removed
-

In the new coordinate system, coordinates corresponding to the last principal components are very small → can take away these columns

Principal Component Analysis - two inputs



PCA- after reducing dimensionality

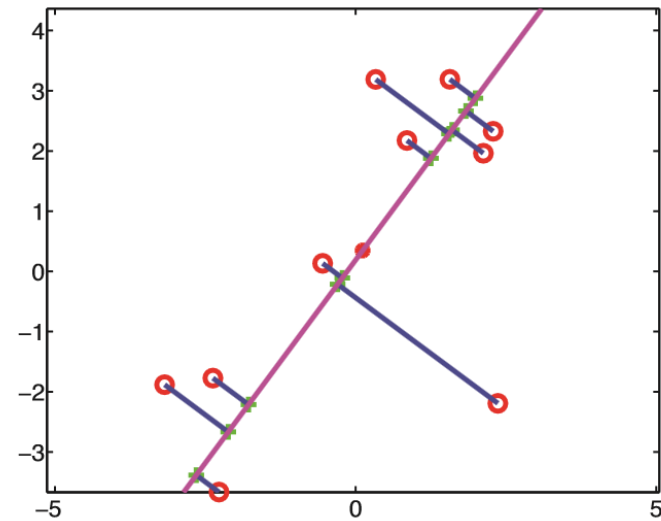


- Data became approximate (but less data to store)
- PC_1, \dots, PC_M are actually **eigenvectors of sample covariance** (first largest eigenvalue, ..., Mth largest eigenvalue)

PCA: another view

- Aim: minimize the distance between the original and projected data

$$\min_V \sum_{i=1}^N \|x_n - \tilde{x}_n\|^2$$



Source: Murphy

PCA: computations

Data $D = \left\| \mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_p \right\|, \quad \mathbf{x}_i = (x_{i1}, \dots, x_{in})$

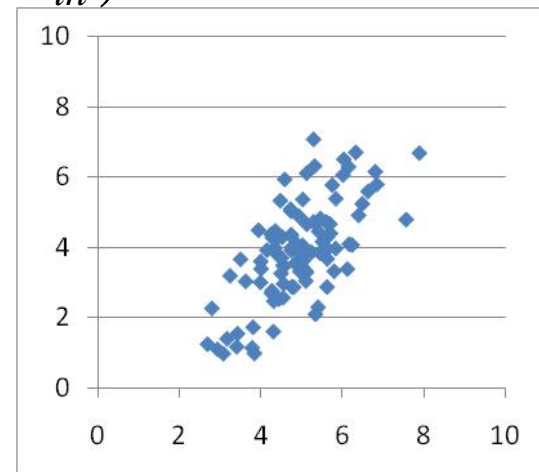
1. Centred data

$$X = \left\| \mathbf{x}_1 - \bar{\mathbf{x}}_1 \ \mathbf{x}_2 - \bar{\mathbf{x}}_2 \ \dots \ \mathbf{x}_p - \bar{\mathbf{x}}_p \right\|,$$

2. Covariance matrix

$$\mathbf{S} = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

3. Search for eigenvectors and eigenvalues of \mathbf{S}

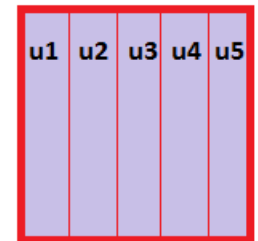
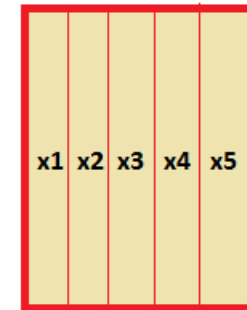


| | Column 1 | Column 2 |
|----------|----------|----------|
| Column 1 | 0.951 | 0.905 |
| Column 2 | 0.905 | 1.883 |

PCA: computations

4. Coordinates of any data point $x=(x_1...x_p)$ in the new coordinate system:

$$z = (z_1, \dots z_n), z_i = x^T u_i$$



Matrix form: $Z = X U$

5. Discard principle components after some M
6. New data will have dimensions $N \times M$ instead of $N \times p$

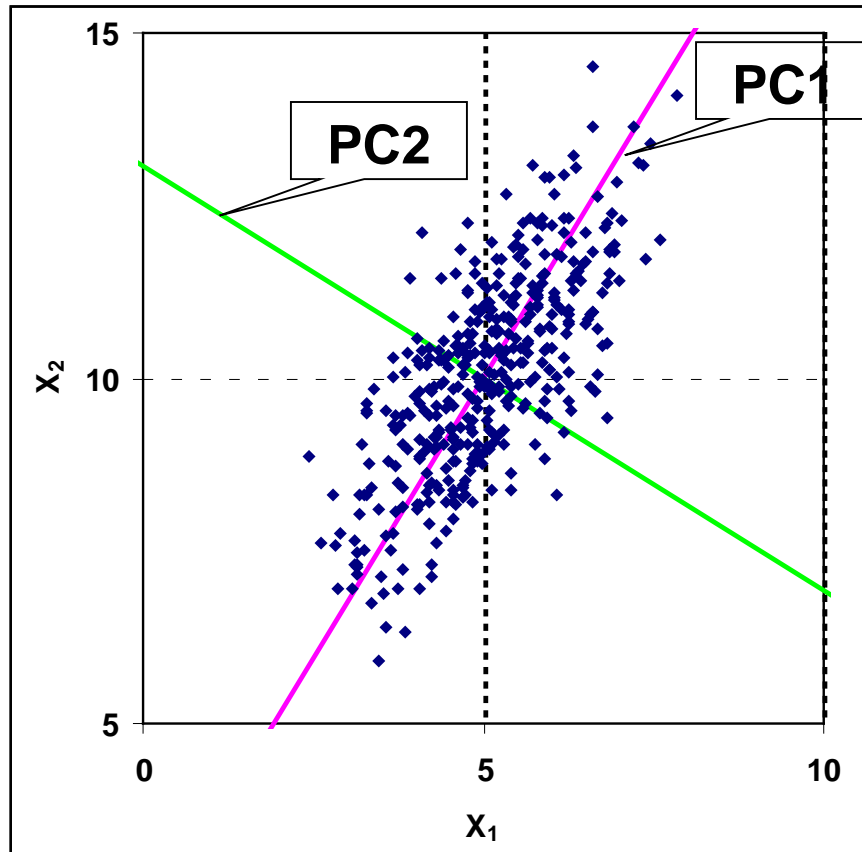
Store: $N \times M + p \times M$
instead $N \times p$

100*50 vs
100*4+50*4

Getting approximate original data:

$$X' = Z U_M^T$$

Principal Component Analysis



Eigenanalysis of the Covariance Matrix

| | | |
|------------|--------|--------|
| Eigenvalue | 2.8162 | 0.3835 |
| Proportion | 0.880 | 0.120 |
| Cumulative | 0.880 | 1.000 |

| Variable | PC1 | PC2 |
|----------|-------|--------|
| X1 | 0.523 | 0.852 |
| X2 | 0.852 | -0.523 |

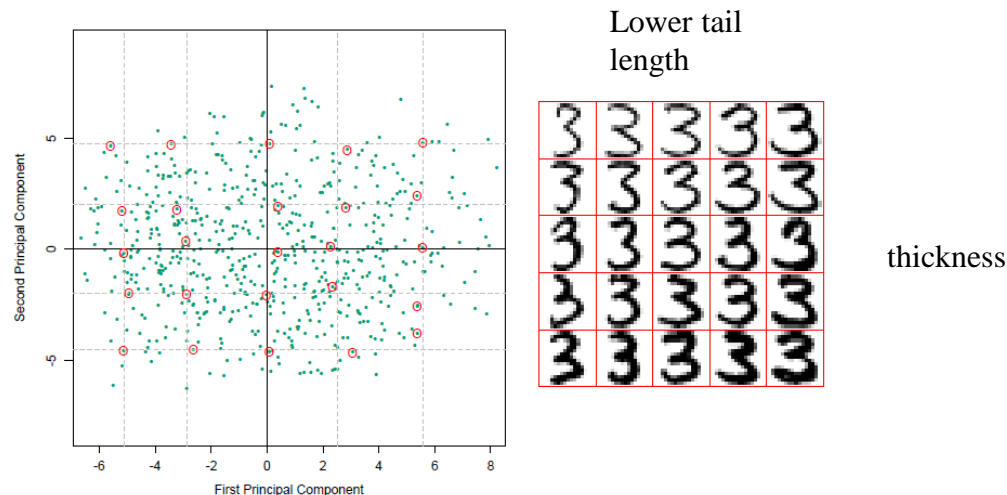
Loadings (U)

Principal Component Analysis

- Digits: two eigenvectors extracted

$$\mathbf{x} = \begin{bmatrix} 3 \end{bmatrix} + \mathbf{z1} \cdot \begin{bmatrix} 3 \end{bmatrix} + \mathbf{z2} \cdot \begin{bmatrix} 3 \end{bmatrix}.$$

- Interpretation of eigenvectors



PCA in R

- `Prcomp()`, `biplot()`, `screeplot()`

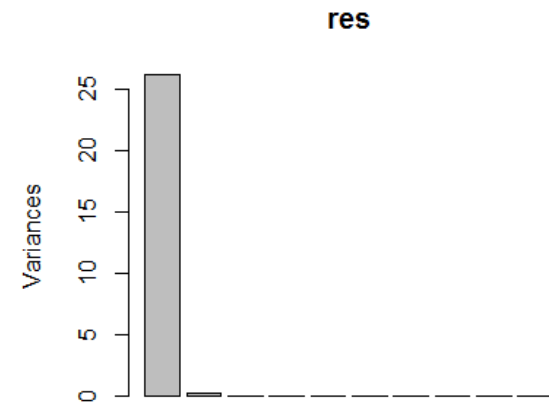
```
mydata=read.csv2("tecator.csv")
data1=mydata
data1$Fat=c()
res=prcomp(data1)
lambda=res$sdev^2
#eigenvalues
lambda
#proportion of variation
sprintf("%2.3f",lambda/sum(lambda)*100)
screeplot(res)
```

```
> lambda
```

```
[1] 2.612713e+01 2.385369e-01 7.844883e-02 3.018501e-01
[7] 2.052212e-04 1.084213e-04 2.077326e-05 1.150359e-04
```

```
> sprintf("%2.3f",lambda/sum(lambda)*100)
```

```
[1] "98.679" "0.901" "0.296" "0.114" "0.006"
[9] "0.000" "0.000" "0.000" "0.000" "0.000"
```



Only 1 component captures the
99% of variation!

PCA in R

- Principal component **loadings (U)**

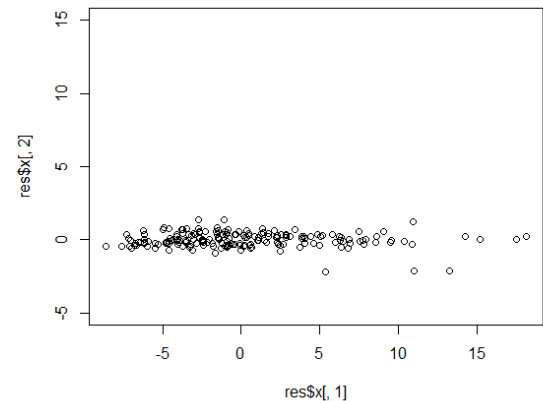
```
U=res$rotation  
head(U)
```

```
> head(U)
```

| | PC1 | PC2 | PC3 | |
|-----------|------------|-----------|------------|---------|
| Channel11 | 0.07938192 | 0.1156228 | 0.08073156 | -0.0927 |
| Channel12 | 0.07987445 | 0.1170972 | 0.07887873 | -0.0981 |
| Channel13 | 0.08036498 | 0.1185571 | 0.07702127 | -0.1031 |
| Channel14 | 0.08085611 | 0.1200006 | 0.07515015 | -0.1077 |
| Channel15 | 0.08135022 | 0.1214075 | 0.07323819 | -0.1119 |
| Channel16 | 0.08184806 | 0.1227401 | 0.07125048 | -0.1156 |

- Data in (PC1, PC2) – **scores (Z)**

```
plot(res$x[,1], res$x[,2], ylim=c(-5,15))
```

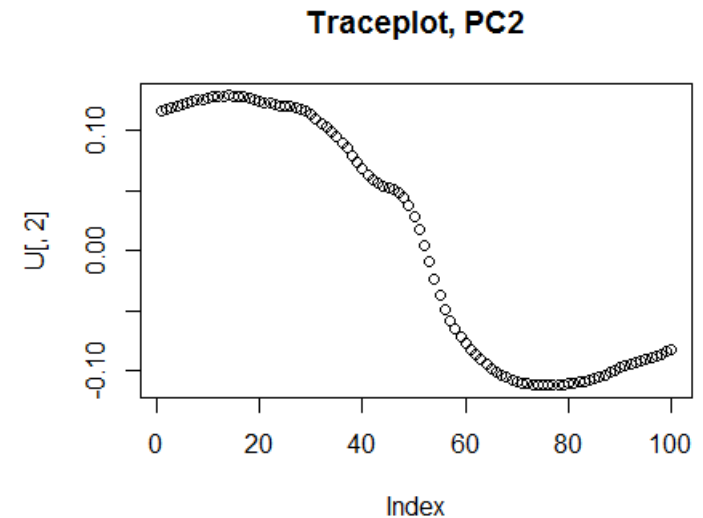
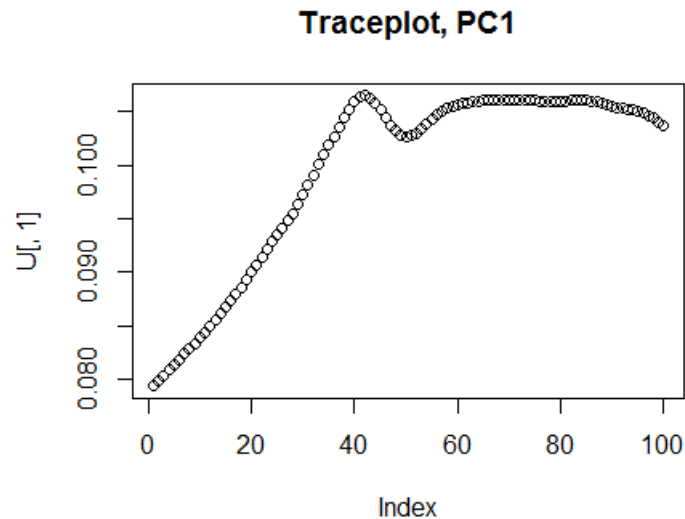


Do we need second dimension?

PCA in R

- Trace plots

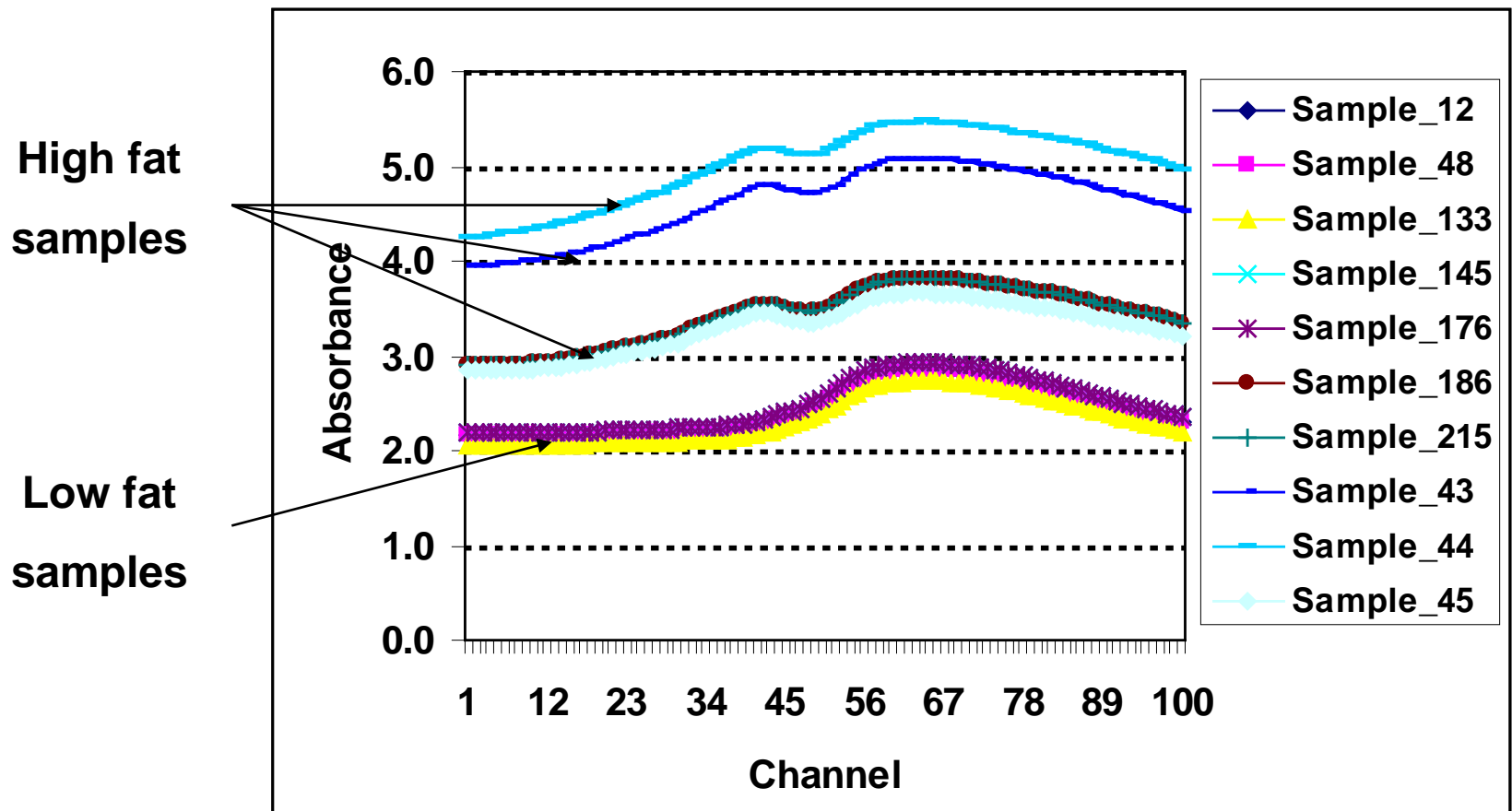
```
U=loadings(res)  
plot(U[,1], main="Traceplot, PC1")  
plot(U[,2],main="Traceplot, PC2")
```



Which components
contribute to PC1-2?

Absorbance records for ten samples of chopped meat

PCA2 captures the most of remaining variation



PCA for high-dimensional data

- Standard PCA for $p \gg N$
 - At most N eigenvalues are nonzero
 - Running time is $O(p^3)$
- High-dimensional PCA
 1. Use $S' = \frac{1}{N}XX^T$ (instead of $S = \frac{1}{N}X^TX$)
 2. Eigenvalues do not change
 3. Eigenvectors of S are $X^T v_i$

Probabilistic PCA

- z_i -latent variables, x_i - observed variables

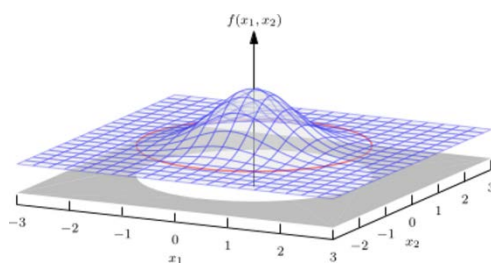
$$\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$$

$$\mathbf{x} | \mathbf{z} \sim N(\mathbf{x} | \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$$

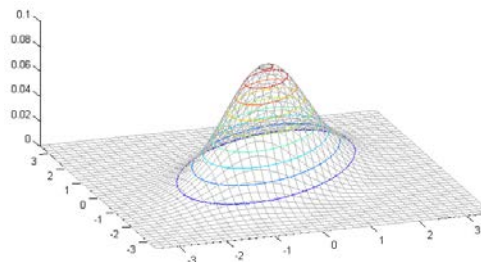
- Alternatively

$$\mathbf{z} \sim N(\mathbf{0}, \mathbf{I}), \mathbf{x} = \boldsymbol{\mu} + \mathbf{W}\mathbf{z} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I})$$

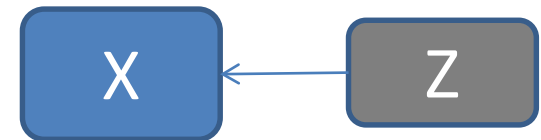
- **Interpretation:** Observed data (X) is obtained by rotation, scaling and translation of standard normal distribution (Z) and adding some noise.



Z



X



Probabilistic PCA

- **Aim:** extract Z from X
- Distribution of x :

$$x \sim N(\mu, C)$$
$$C = WW^T + \sigma^2 I$$

- Rotation invariance

- Assume that x was generated from $z' = Rz, RR^T = I$, $p(x)$ does not change!

$$x|z' \sim N(x|Wz' + \mu, \sigma^2 I)$$

- **Model will not be able find latent factors uniquely!** ☹
 - It does not distinguish z from z'

Probabilistic PCA

- Estimation of parameters: ML

Theorem. ML estimates are given by

$$\begin{aligned}\mu_{ML} &= \bar{x} \\ W_{ML} &= U_M (L_M - \sigma_{ML}^2 I)^{\frac{1}{2}} R \\ \sigma_{ML}^2 &= \frac{1}{p-M} \sum_{i=M+1}^p \lambda_i\end{aligned}$$

- U_M matrix of M eigenvectors
- L_M diagonal matrix of M eigenvalues
- R any orthogonal matrix

Probabilistic PCA

- Estimation of Z

- Use mean of posterior

$$\hat{Z} = (W_{ML}^T W_{ML} + \sigma_{ML}^2 I)^{-1} W_{ML}^T (x - \mu)$$

- Connection to standard PCA

- Assume $R = I, \sigma^2 = 0 \rightarrow$ get standard PCA components scaled by inverse root of eigenvalues

$$Z = XUL^{-\frac{1}{2}}$$

Advantages of probabilistic PCA

- More settings to specify → more flexible
- Can be faster when $M \ll p$
- Missing values can be handled
- M can be derived if a Bayesian version is used
- Probabilistic PCA can be applied to classification problems directly
- Probabilistic PCA can generate new data

Probabilistic PCA in R

- Use **pcaMethods** from Bioconductor
- Install
 - `source("https://bioconductor.org/biocLite.R")`
 - `biocLite("pcaMethods")`

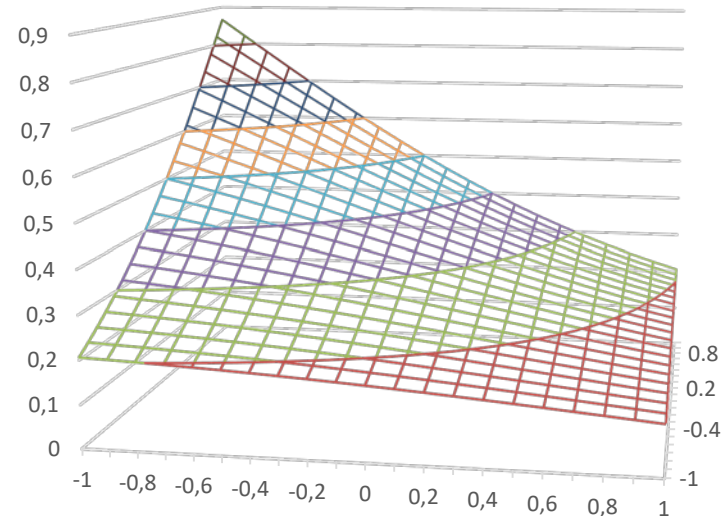
`Ppca(data, nPcs,...)`

Results: scores, loadings...

Independent component analysis (ICA)

- Probabilistic PCA does not capture latent factors
 - Rotation invariance
- Let's choose distribution which is not rotation invariant → will get unique latent factors
- Choose non-Gaussian $p(z)$

$$p(z) = \prod_{i=1}^M p(z_i)$$



ICA

- Model

$$x = \mu + Wz + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \Sigma)$$

- Maximum likelihood ($V = W^{-1}$)

$$\max_V \sum_{i=1}^n \sum_{j=1}^p \log(p_i(v_i^T x_j))$$

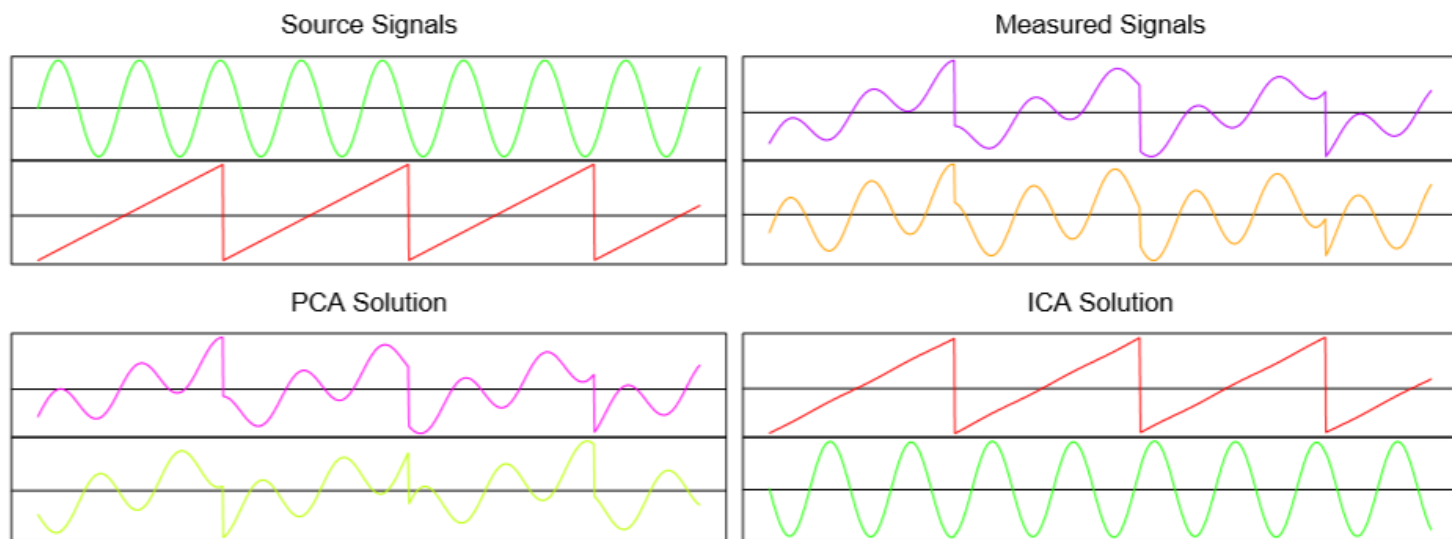
Subject to $\|v_i\| = 1$

- Another equivalent form \rightarrow maximize negentropy
 - ICA looks for model which is as much non-Gaussian as possible

$$\text{negentropy}(z) \triangleq \mathbb{H}(\mathcal{N}(\mu, \sigma^2)) - \mathbb{H}(z)$$

ICA

- Example



Source: Elem of stat learn by Hastie

Independent component analysis: R

```
S <- cbind(sin((1:1000)/20), rep((((1:200)-100)/100), 5)) #Generate data
A <- matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)
X <- S %*% A
a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
  method = "R", row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE) #ICA
```

