Lecture notes about Dimension Reduction, Visualization (draft)

Christophe

2026-01-12

Table of contents

Preface

This booklet is a collection of slides, proofs and exercises for the lecture "Dimension reduction and visualization". In no way, it is a full textbook on the topic, but rather a supplement to the lecture.

Outline of the course

We will consider the generic problem of reducing the dimension of a dataset, while preserving as much information as possible. We consider model of the type:

$$x = f(z) + \varepsilon,$$

where x is the observed high-dimensional data, z is a low-dimensional latent variable, f is a function linking the two variables and ε is a noise term. Different choices of f and of the distribution of ε lead to different models for dimension reduction.

We will also study the so-called decoding problem, which consists in estimating x from \$ z:

$$\hat{x} = g(z).$$

Other approaches to dimension reduction, such as multidimensional scaling (MDS) or t-SNE, will not be studied in this course.

The course is organized as follows:

1. Linear models:

- Factor Analysis (FA),
- Probabilistic Principal Component Analysis (PPCA),
- Independent Component Analysis (ICA).

2. Non-linear models:

- Auto-Encoders (AE),
- Variational Auto-Encoders (VAE).

3. Multidimensional scaling (MDS) and t-SNE, UMAP (if time permits).

Each chapter will contain slides, proofs and exercises related to the models studied.

1 Factor Analysis

1.1 Principles

- Using discrete latent variables provides limited summary (clustering)
- An alternative is to use a vector of real-valued latent variables, $z \in \mathbb{R}^L$.
- "Factor analysis (FA) describes variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors." Wikipedia quote.

1.1.1 PCA and FA are similar but different

PCA focuses on capturing overall variance, while FA aims to uncover underlying latent factors explaining correlations among observed variables.

1.2 Difference between PCA and FA

Aspect	PCA (Principal Component Analysis)	Factor Analysis (FA)
Objective	Maximise total variance explained	Explain correlations using latent variables
Model	Pure algebraic decomposition	Statistical latent variable model
Equation	$Z = XW$	$X = \Lambda F + \varepsilon$
Variance used	Total variance (shared + unique + error)	Common (shared) variance only
Measurement error	Not modelled	Explicitly modelled via ε
Components / Factors	Mathematical constructs	Latent causes
Uniqueness	Unique solution	Non-unique $\rightarrow$ requires rotation

Aspect	PCA (Principal Component Analysis)	Factor Analysis (FA)
Estimation	Eigen-decomposition of covariance / correlation	ML, GLS, Principal Factor, etc.
Main usage	Dimension reduction, compression, prediction	Construct identification, theory building

1.3 The model of factor analysis

We consider the observation $x \in \mathbb{R}^D$

$$x = Wz + \mu + \epsilon$$

where

- the noise $\epsilon \sim \mathcal{N}_D(0, \Psi)$
- the hidden (latent) vector $z \sim \mathcal{N}_L(0, I_L)$

$$p(x|z, \theta) = \mathcal{N}(Wz + \mu, \Psi)$$

the mean is a linear function of the (hidden) inputs

- W is a $D \times L$ matrix, known as the factor loading matrix,
- Ψ is a $D \times D$ covariance matrix that we take to be diagonal

The special case in which $\Psi = \sigma^2 I$ is called probabilistic principal components analysis or PPCA.

1.4 Reminder: Joint and conditional Gaussian distribution

Let us recall that if (see Murphy chapter 4)

- $z \sim \mathcal{N}(\mu_z, \Sigma_{zz})$ and $x \sim \mathcal{N}(\mu_x, \Sigma_{xx})$ and $cov(z, x) = \Sigma_{zx}$

then

$$p(z, x) = N(\begin{bmatrix} z \\ x \end{bmatrix} | \begin{bmatrix} \mu_z \\ \mu_x \end{bmatrix}, \begin{bmatrix} \Sigma_{zz} & \Sigma_{zx} \\ \Sigma_{xz} & \Sigma_{xx} \end{bmatrix})$$

and

$$p(z,x) = p(z|x)p(x) = \mathcal{N}(z|\mu_{z|x}, \Sigma_{z|x})\mathcal{N}(x|\mu_x, \Sigma_{xx})$$

where

- $\mu_{z|x} = \mu_z + \Sigma_{zx}\Sigma_{xx}^{-1}(x - \mu_x)$
- $\Sigma_{z|x} = \Sigma_{zz} - \Sigma_{zx}\Sigma_{xx}^{-1}\Sigma_{xz}$

1.5 Marginal and posterior distribution

1.5.1 Marginal distribution

$$x \sim \mathcal{N}_D(\mu, \Sigma_{xx} = WW^T + \Psi)$$

1.5.2 Posterior distribution

$$z|x \sim \mathcal{N}_L(\mu_{z|x}, \Sigma_{z|x})$$

where

- $\Sigma_{z|x} = (I_L + W^T\Psi^{-1}W)^{-1} = S$
- $\mu_{z|x} = \Sigma_{z|x}\Sigma_{xx}^{-1}(x - \mu) = SW^T\Psi^{-1}(x - \mu)$

1.5.3 Exercice

Demonstrate the above formulas

1.6 Estimation

1.6.1 The mean μ

can be estimated by maximum likelihood

$$\mu_{mle} = \bar{x}$$

1.6.2 W and Ψ

are estimated using an EM algorithm

1.7 EM algorithm

1.7.1 Data

- Observed data : $x_{1:n}$
- Missing (or hidden) data : : $z_{1:n}$

1.7.2 Principle

- Starting from θ^0
- At step q
  - E(xpectation) step: $Q(\theta, \theta^q) = E_{Z_{1:n}|x_{1:n}}[\log P(x_{1:n}, z_{1:n}, \theta)]$
  - M(aximisation) step: $\theta^{q+1} = argmax_\theta Q(\theta, \theta^q)$

1.8 EM for factor analysis

Let us assume that $\mu = 0$ (centering of the x_i), the complete log-likelihood is

$$\begin{aligned}\log p(X, Z|\mu, W, \Psi) &= \sum_i \log \mathcal{N}_L(z_i; 0, I) + \log \mathcal{N}_D(x_i; Wz_i, \Psi) \\ &= -\frac{n}{2}\log|I_L| - \frac{n}{2}Tr(\hat{\Sigma}_{zz}) \\ &\quad -\frac{n}{2}\log|\Psi| - \frac{n}{2}Tr(\hat{\Sigma}_{xx}\Psi^{-1}) + Cst\end{aligned}$$

where

$$\hat{\Sigma}_{xx} = \frac{1}{n}\sum_i (x_i - Wz_i)(x_i - Wz_i)^T$$

1.8.1 Exercice

Demonstrate the above formula

1.9 E step

The expectation of the complete log-likelihood requires

1. $\mathbb{E}_{z|x}[z_i] = SW^T\Psi^{-1}(x_i - \mu)$ where $S = (I_L + W^T\Psi^{-1}W)^{-1}$
2. $\mathbb{E}_{z|x}[z_i z_i^T] = E_{z|x}[z_i]E_{z|x}[z_i^T] + S$

1.10 M step

Reminders

$$\frac{\partial(b^T a)}{\partial a} = b$$
$$\frac{\partial(a^T A a)}{\partial a} = (A + A^T)a$$
$$\frac{\partial}{\partial A} tr(BA) = B^T$$
$$\frac{\partial}{\partial A} \log |A| = (A^{-1})^T$$
$$tr(ABC) = tr(CAB) = tra(BCA)$$

Thus if x is a vector

$$x^T A x = tr(x^T A x) = tr(A x x^T)$$

1.11 M step for Ψ

$$\mathbb{E}_{z|x}\left[\frac{\partial L(W,\Psi)}{\partial \Psi^{-1}}\right] = \mathbb{E}_{z|x}\left[\frac{n}{2}(\Psi - \hat{\Sigma}_{xx})\right] = \frac{n}{2}(\Psi - \mathbb{E}_{z|x}\left[\hat{\Sigma}_{xx}\right]) = 0$$

where

$$\begin{aligned}
\mathbb{E}_{z|x}\left[\hat{\Sigma}_{xx}\right] &= \frac{1}{n}(\sum_i x_i x_i^T + W(\sum_i \mathbb{E}_{z|x}\left[z_i z_i^T\right])W^T - 2W\sum_i E_{z|x}\left[z_i\right] x_i^T) \\
&= \frac{1}{n}(\sum_i x_i x_i^T + W(\sum_i \mathbb{E}_{z|x}\left[z_i x_i^T\right]) - 2W\sum_i E_{z|x}\left[z_i\right] x_i^T) \\
&= \frac{1}{n}(\sum_i x_i x_i^T - W\sum_i E_{z|x}\left[z_i\right] x_i^T)
\end{aligned}$$

1.12 M step for W

$$\mathbb{E}_{z|x}\left[\frac{\partial L(W,\Psi)}{\partial W}\right] = \mathbb{E}_{z|x}\left[-\Psi^{-1}\sum_i x_i z_i^T + \Psi^{-1} W \sum_i z_i z_i^T\right] = 0$$

1.13 M Step summary

1.13.1 Loading matrix

$$W^{q+1} = \left(\sum_i (x_i - \bar{x}) \mathbb{E}_{z|x}[z_i]^T \right) \left(\sum_i \mathbb{E}_{z|x}[z_i z_i^T] \right)^{-1}$$

1.13.2 Noise covariance matrix

$$\Psi^{q+1} = \frac{1}{N} diag\{ \sum_i x_i x_i^T - W^{q+1} \mathbb{E}_{z|x}[z_i] x_i^T \}$$

1.13.3 Log-likelihood

The log-likelihood can be computed using the EM decomposition

$$\log P(X; \Theta) = E_{Z_{1:n}|x_{1:n}}[\log P(x_{1:n}, z_{1:n}; \theta)] - E_{Z_{1:n}|x_{1:n}}[\log P(z_{1:n}|x_{1:n}; \theta)]$$

1.14 Implementation of the algorithm

1.14.1 Initialisation via a PCA

```
initialisation.FA<-function(X,L=1){
  # Return W and Psi
  d<-ncol(X)
  Sigmaxx<-var(X)
  W<-eigen(Sigmaxx)$vectors[,1:L]
  if (L==1) W<-cbind(W)
  Psi<-rep(1,d)
  return(list(W=W,Psi=Psi))
}
```

1.14.2 E step

```r
FA.E.step<-function(X,W,Psi){
  # X is assumed to be centered
  # M contain the contionnal expectation of the latent factor
  # S contains the covariance of the latent factor
  L<-ncol(W)
  S <- solve(diag(L) + t(W)%*%diag(1/Psi)%*%W)
  M<- X%*%diag(1/Psi)%*%W%*%S
  return(list(S=S,M=M))
}
```

1.14.3 M Step

```r
FA.M.step<-function(X,S,M,W,Psi){
  n<-nrow(X)
  Psi<-1/n*diag(t(X)%*%X -W%*%t(M)%*%X)
  W<- (t(X)%*%M)%*%solve(n*S+t(M)%*%M)
  return(list(Psi=Psi,W=W))
}
```

1.14.4 Computation of the criterion

```r
log.likelihood.FA<-function(X,S,M,Psi,W){
  n<-nrow(X)
  Sigmax<-(t(X)%*%X-W%*%t(M)%*%X)
  return(-(sum(diag(S+t(M)%*%M/n))
          +log(det(diag(Psi)))+
            log(det(S))+
            1/n*sum(diag(Sigmax%*%diag(1/Psi)))))
}
```

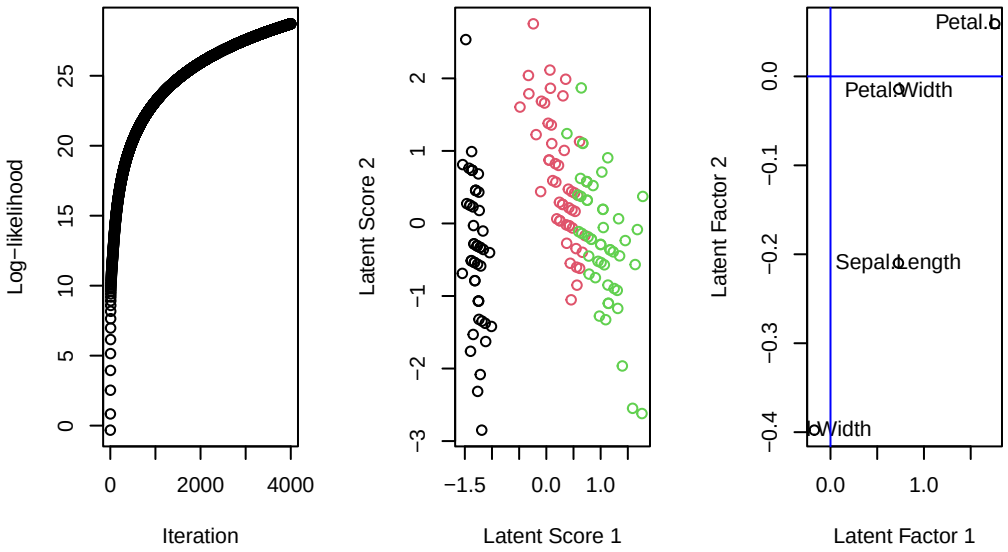
1.14.5 Putting it all together

```r
FA.EM<-function(X,L=1,max.iter=50){
  X<-scale(X,scale=FALSE);mu<-attr(X,"scaled:center")
  log.likelihood<-NULL; init<-initialisation.FA(X,L)
  W<-init$W;  Psi<-init$Psi; criterion<- Inf; iteration<-1;
```

```
  log.likelihood[iteration]<--Inf
  while ((criterion>1e-6)&&(iteration<=max.iter)){
    E.step<-FA.E.step(X,W,Psi); E.step$S->S; E.step$M->M
    M.step<-FA.M.step(X,S,M,W,Psi); M.step$Psi->Psi; M.step$W->W
    iteration<-iteration+1
    log.likelihood[iteration]<-log.likelihood.FA(X,S,M,Psi,W)
    criterion<-abs((log.likelihood[iteration] - log.likelihood[iteration-1])/max(log.likeliho
  }
  return(list( W=data.frame(W), Psi=Psi,
               M=data.frame(M), S=S,mu=mu,
               log.likelihood= log.likelihood[-1]))
}
```

1.15 Example with the Iris

```
data(iris); L=2
X<-scale(iris[,1:4],center=TRUE,scale=FALSE)
FA.result<-FA.EM(X,L=L,max.iter=4000)
names(FA.result$W)<-paste("Latent Factor",1:L)
names(FA.result$M)<-paste("Latent Score",1:L)
par(mfrow=c(1,3))
plot(FA.result$log.likelihood,ylab="Log-likelihood",xlab="Iteration")
plot(FA.result$M,col=iris$Species)
plot(FA.result$W)
text(FA.result$W,labels = names(iris)[1:4])
abline(h=0,v=0,col="blue")
```



1.16 Unidentifiability

- If we consider R an orthogonal rotation matrix such that

$$RR^T = I$$

It appears that $\tilde{W} = WR$ produces the same log-likelihood.

- W cannot be uniquely identified.

1.17 Possible rotations

1. Forcing W to be orthogonal with columns ordered by deacresing variance
2. Forcing W to be lower triangular (problem of founder variables)
3. Choosing an informative rotation matrix. For example the varimax rotation.
4. ...

Rotation	Type	Factors correlated?	Optimization criterion	Effect on loadings	Typical use case
None	–	–	None	Hard to interpret, mixed loadings	Never for interpretation

Rotation	Type	Factors correlated?	Optimization criterion	Effect on loadings	Typical use case
Varimax	Orthogonal	No	Maximise variance of squared loadings per factor	Each factor defined by few strong variables	General exploratory FA
Quartimax	Orthogonal	No	Simplify variables	One dominant general factor	When expecting one main latent trait
Equamax	Orthogonal	No	Compromise varimax/quartimax	Mixed behaviour	Rarely used
Oblimin	Oblique	Yes	Minimise cross-loadings with tunable correlation	Clean structure with correlated factors	Psychology, social sciences
Promax	Oblique	Yes	Varimax + power transformation	Fast, realistic latent traits	Large datasets
Geomin	Oblique	Yes	Minimise geometric mean of loadings	Smooth, stable solution	SEM / Mplus style models

1.17.1 Varimax

Varimax rotation maximizes the sum of the variance of the squared correlations between variables and factors

$$R_{\text{VARIMAX}} = \arg\max_{R} \left(\frac{1}{p} \sum_{j=1}^{k} \sum_{i=1}^{p} (WR)_{ij}^{4} - \sum_{j=1}^{k} \left(\frac{1}{p} \sum_{i=1}^{p} (WR)_{ij}^{2} \right)^{2} \right)$$

This results in high factor loadings for a small number of variables and low factor loadings for the rest.

1.18 Varimax

```
W.FA<-FA.result$W
W.Varimax<-varimax(as.matrix(FA.result$W))$loadings
print(W.Varimax)
```

```
Loadings:
             Latent Factor 1 Latent Factor 2
Sepal.Length  0.756
Sepal.Width                  -0.429
Petal.Length  1.683           0.509
Petal.Width   0.711           0.174

               Latent Factor 1 Latent Factor 2
SS loadings              3.916           0.473
Proportion Var           0.979           0.118
Cumulative Var           0.979           1.097
```

1.19 PCA vs FA decomposition

1.19.1 PCA vs FA loadings comparison

- PCA loadings are obtained by the eigen-decomposition of the covariance or correlation matrix
- FA loadings are obtained by the EM algorithm presented above
- PCA loadings are orthogonal, FA loadings are not
- PCA loadings explain total variance, FA loadings explain common variance only

1.19.2 Decompostion of PCA vs FA

PCA decomposes the covariance matrix Σ as

$$\Sigma = W_{PCA} W_{PCA}^T$$

FA decomposes the covariance matrix Σ as

$$\Sigma = W_{FA} W_{FA}^T + \Psi$$

where Ψ is a diagonal matrix

1.20 Communalities and Uniqueness

Communalities in FA represent the proportion of each variable's variance that can be explained by the common factors. They provide insight into how well the factors capture the underlying structure of the data.

Communality for variable j is defined as:

$$\text{Communality}_j = \sum_{k=1}^{L} \lambda_{jk}^2,$$

where λ_{jk} is the loading of variable j on factor k, and L is the number of factors.

Uniqueness is the portion of variance that is unique to each variable and not explained by the common factors. It reflects the specific variance of each variable that is not shared with others.

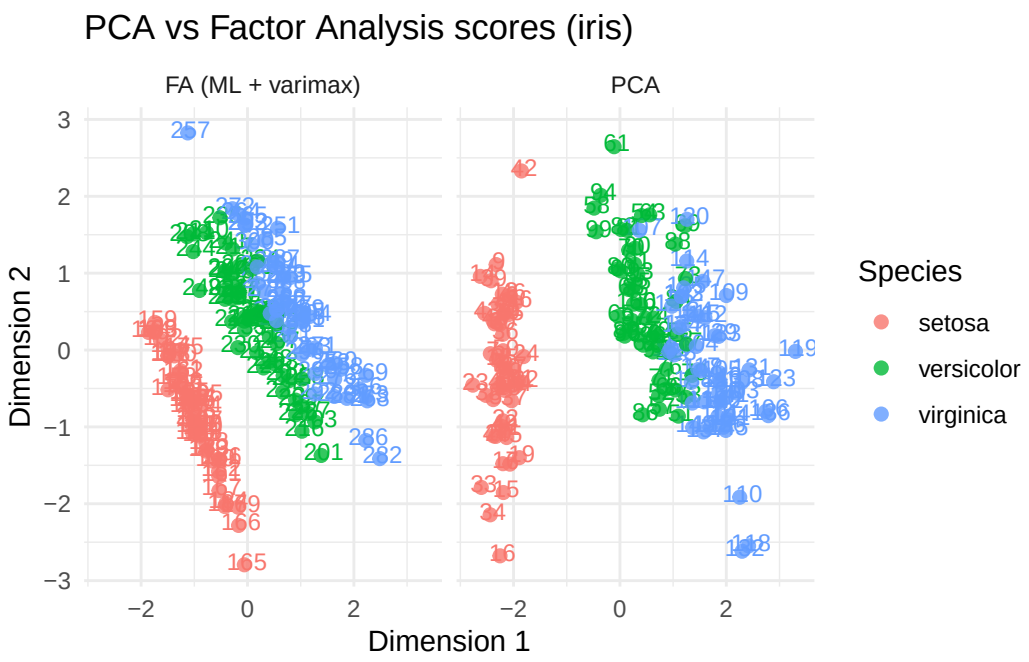
Uniqueness and communalities are related as follows:

$$\text{Communality}_j = 1 - \text{Uniqueness}_j$$

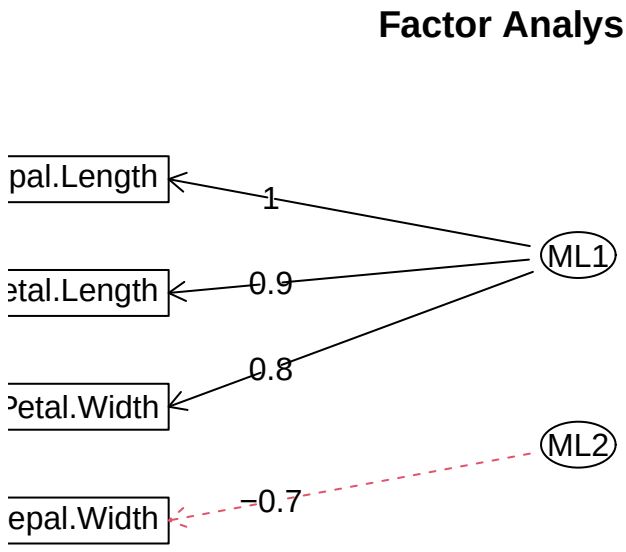
1.21 Iris dataset FA results

```r
# Packages (install only if missing)
pkgs <- c("psych", "knitr")
to_install <- pkgs[!pkgs %in% rownames(installed.packages())]
if (length(to_install) > 0) install.packages(to_install, dependencies = TRUE)
library(ggplot2)
library(psych)
library(knitr)

# ----------------------------
# Data: iris (numeric columns)
# ----------------------------
X <- iris[, 1:4]
Xsc <- scale(X)                 # standardize
R <- cor(Xsc)                   # correlation matrix (PCA/FA comparable)

# ----------------------------
# PCA (2 components)
# ----------------------------
pca <- prcomp(Xsc, center = FALSE, scale. = FALSE)
pca_load <- pca$rotation[, 1:2, drop = FALSE]
pca_var  <- (pca$sdev^2) / sum(pca$sdev^2)  # proportion var explained

# ----------------------------
# FA (2 factors, ML, varimax)
# ----------------------------
```

```r
fa2 <- fa(Xsc, nfactors = 2, fm = "ml", rotate = "varimax",scores = "regression")
fa_load <- unclass(fa2$loadings)[, 1:2, drop = FALSE]
comm <- fa2$communality
uniq <- fa2$uniquenesses


#------------------------------
# Compare projections of the data
#------------------------------

# PCA scores
pca_scores <- as.data.frame(pca$x[, 1:2, drop = FALSE])
colnames(pca_scores) <- c("Dim1", "Dim2")
pca_scores$Method <- "PCA"
fa_scores <- as.data.frame(fa2$scores[, 1:2, drop = FALSE])
colnames(fa_scores) <- c("Dim1", "Dim2")
fa_scores$Method <- "FA (ML + varimax)"
scores <- rbind(pca_scores, fa_scores)
scores$Ind <- seq_len(nrow(scores))
scores$Species <- rep(iris$Species, 2)
ggplot(scores, aes(x = Dim1, y = Dim2, color = Species)) +
  geom_text(aes(label = Ind),
            size = 3,
            nudge_x = 0.05,
            nudge_y = 0.05,
            show.legend = FALSE) +
  geom_point(size = 2, alpha = 0.8) +  facet_wrap(~ Method) +
  labs(title = "PCA vs Factor Analysis scores (iris)",
       x = "Dimension 1", y = "Dimension 2") +
  theme_minimal()
```




```
# Vizualize FA results with diagram
fa.diagram(fa2)
```




```
# ----------------------------
# Compare in one table
# ----------------------------
tab <- data.frame(
```

```
  variable = rownames(pca_load),
  PCA1 = pca_load[,1],
  PCA2 = pca_load[,2],
  FA1 = fa_load[,1],
  FA2 = fa_load[,2],
  communality = comm,
  uniqueness = uniq
)

# pretty rounding for print
tab_print <- within(tab, {
  PCA1 <- round(PCA1, 3); PCA2 <- round(PCA2, 3)
  FA1 <- round(FA1, 3); FA2 <- round(FA2, 3)
  communality <- round(communality, 3); uniqueness <- round(uniqueness, 3)
})

kable(
  tab_print,
  caption = "Iris: PCA vs FA (ML + varimax)."
)
```

Table 1.3: Iris: PCA vs FA (ML + varimax).

	variable	PCA1	PCA2	FA1	FA2	communal-ity	uniqueness
Sepal.Length	Sepal.Length	0.521	-0.377	0.997	0.006	0.995	0.005
Sepal.Width	Sepal.Width	-0.269	-0.923	-0.115	-0.665	0.455	0.545
Petal.Length	Petal.Length	0.580	-0.024	0.871	0.486	0.995	0.005
Petal.Width	Petal.Width	0.565	-0.067	0.818	0.514	0.932	0.068

```
cat("\n\n")
```

```
cat(sprintf("*PCA variance explained:* PC1 = %.1f%%, PC2 = %.1f%% (cum. %.1f%%)\n",
            100*pca_var[1], 100*pca_var[2], 100*sum(pca_var[1:2])))
```

PCA variance explained: PC1 = 73.0%, PC2 = 22.9% (cum. 95.8%)

```
cat(sprintf("*FA variance explained (SS loadings):* F1 = %.2f, F2 = %.2f (cum. %.2f)\n",
            fa2$Vaccounted["SS loadings",1],
            fa2$Vaccounted["SS loadings",2],
            sum(fa2$Vaccounted["SS loadings",1:2])))
```

FA variance explained (SS loadings): F1 = 2.44, F2 = 0.94 (cum. 3.38)

1.22 PCA vs FA on the *iris* dataset

This table is a textbook example of when **PCA and Factor Analysis almost coincide**.

1.22.1 PCA loadings

- **PC1** is dominated by *Petal.Length*, *Petal.Width*, *Sepal.Length* → a **global size axis**.
- **PC2** is almost entirely *Sepal.Width* → a nearly pure morphological dimension.

Already, PCA provides a very clean structure.

1.22.2 FA loadings (ML + varimax)

- **FA1** petal length + petal width + sepal length
- **FA2** sepal width (with a small secondary petal effect)

The factors are **almost identical** to PC1 and PC2.

1.23 PCA vs FA on the *iris* dataset (contd.)

1.23.1 Communalities: the key diagnostic

- *Petal.Length*, *Petal.Width*, *Sepal.Length*:
  communalities $1 \rightarrow$ almost pure common variance → **ideal latent indicators**.
- *Sepal.Width*:
  communality $= 0.46 \rightarrow$ more than half its variance is idiosyncratic.

This is the **only variable where FA truly disagrees with PCA**.

1.23.2 Conclusion

On *iris*:

- The latent structure is extremely clean.
- Almost all variance is shared.
- Measurement noise is minimal.

> **Note**
>
> When communalities are near 1, **PCA** **FA**.
> This dataset is therefore a *good possible case for PCA*

1.24 Relation to principal component analysis

1.24.1 Assumption

If

- $\Psi = \sigma^2 I$
- W is orthogonal

and

- $\sigma^2 \to 0$

Then

Tipping, M. and C. Bishop (1999, Probabilistic principal component analysis. J. of Royal Stat. Soc. Series B 21(3), 611–622) showed that FA is equivalent to PCA

1.24.2 Criterion

$$J(W, Z) = \|X - ZW^T\|_F^2$$

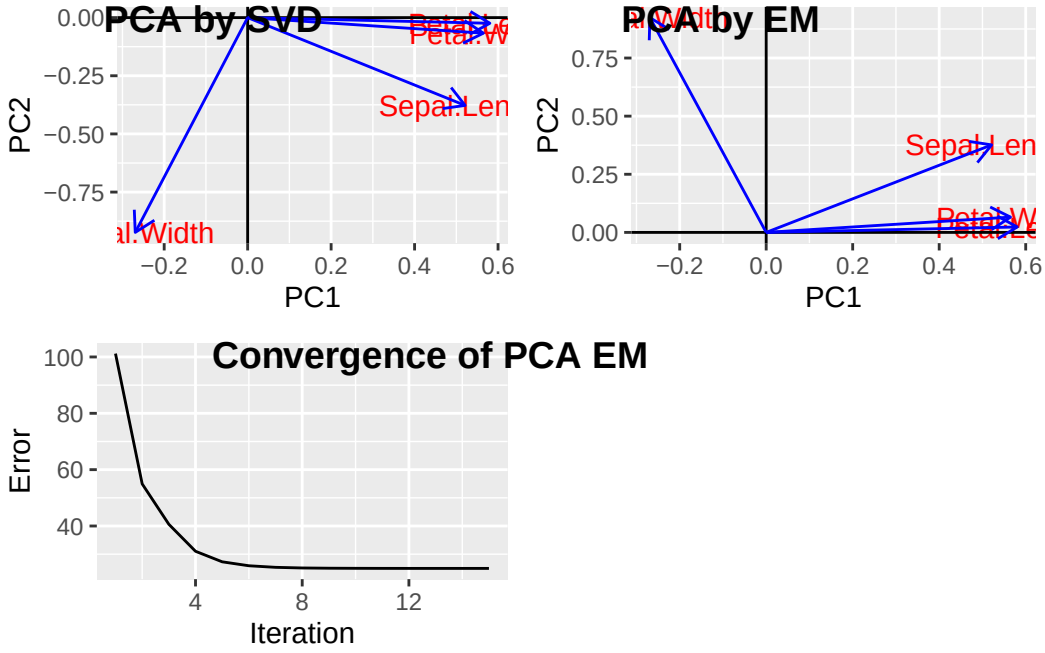
where $W^T W = I$

1.25 A Constrained EM Algorithm for PCA (from Ahn, J.-H. and J.-H. Oh, 2003)

```
upper<-function(A){A[lower.tri(A,diag=FALSE)]<-0;return(A)}
lower<-function(A){A[upper.tri(A,diag=FALSE)]<-0;return(A)}
PCA.EM<-function(X,q=2){
  p<-ncol(X); n<-nrow(X)
  W<-diag(p)[,1:q]; M<-X%*%W # Initialisation
  Jold<-0; J<-1; iteration<-0; Error<-NULL
  while ((abs(J - Jold)>1e-3)){
    Jold<-sum((X-M%*%t(W))^2)
    S <- solve(upper(t(W)%*%W)); M<- X%*%W%*%S # E-step
    W<- (t(X)%*%M)%*%solve(lower(n*S+t(M)%*%M))# M-step
    W<-apply(W,2,function(x){x/sqrt(sum(x^2))})#orthogonalisation
    J<-sum((X-M%*%t(W))^2); Error[iteration<-iteration+1]<-J
  }
  return(list(W=data.frame(W),M=data.frame(M),Error=Error))}
```

```
library(ggplot2)
library(cowplot)
#library(ggfortify)
library(metR)
X<-scale(iris[,1:4])
W <- data.frame(prcomp(X, scale. = TRUE)$rotation)

iris.pca<-ggplot(data=W,aes(PC1,PC2,label=rownames(W)))+geom_text(col="red")+
  geom_vline(aes(xintercept=0))+
  geom_hline(aes(yintercept=0))+
  geom_segment(aes(x=0, y=0, xend=PC1, yend=PC2 ),      arrow=arrow(length=unit(0.3,"cm")),c

pca.em.res<-PCA.EM(X)
W<-pca.em.res$W
names(W)<-paste("PC",1:2,sep="")

iris.em.pca<-ggplot(data=W,aes(PC1,PC2,label=rownames(W)))+geom_text(col="red")+
  geom_vline(aes(xintercept=0))+
  geom_hline(aes(yintercept=0))+
  geom_segment(aes(x=0, y=0, xend=PC1, yend=PC2 ),      arrow=arrow(length=unit(0.3,"cm")),c

Error=data.frame(Error=pca.em.res$Error,Iteration=1:length(pca.em.res$Error))
iris.em.pca.error<-ggplot(data=Error,          aes(x=Iteration,y=Error))+geom_line()
```

```
plot_grid(iris.pca,iris.em.pca,
          iris.em.pca.error,
          labels = c("PCA by SVD","PCA by EM","Convergence of PCA EM"))
```



1.26 Mixture of factor analysers

- Factor analyses is a way to estimate a variance matrix with few parameters
- This property can be used in the context of Gaussian mixture model assuming the following parameterization for component densities:

$$p(x_i|z_i, q_i = k) = \mathcal{N}(x_i|\mu_k + W_k z_i, \Psi)$$

  where k is the component number and W_k is a loading matrix defining the relation between the observation x_i and the latent vector z_i
- This approach is simular to the Banfield-Raftery idea of decomposing the component variance matrix k in volume, form et direction.

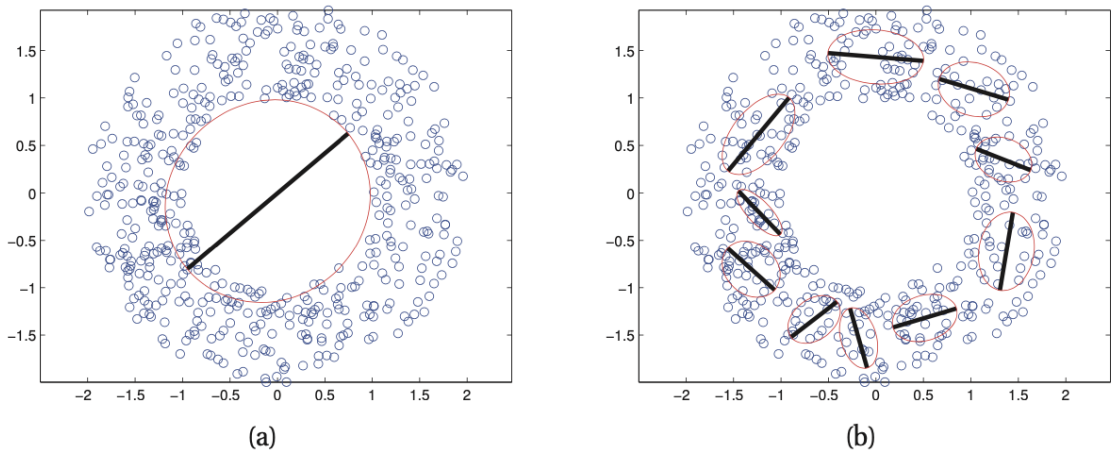


Figure 1.1: Mixture of 1d PPCAs with 1 and 10 components (from Murphy Chapter 12)

2 Exercices on Factor Analysis

We are in the Factor Analysis model

$$x = Wz + \mu + \varepsilon, \qquad z \sim \mathcal{N}_L(0, I_L), \qquad \varepsilon \sim \mathcal{N}_D(0, \Psi),$$

with Ψ diagonal (and invertible). Hence

$$x \mid z \sim \mathcal{N}_D(Wz + \mu, \Psi).$$

The woodbury identity that will be useful is

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

2.1 Exercice: Joint distribution of (z, x) and conditional distribution $z \mid x$

1. Compute moments mean and covariance of the joint vector

Solution

We have to compute the mean and covariance of the joint vector

`- Mean:`

$$\mathbb{E}[z] = 0, \qquad \mathbb{E}[x] = \mathbb{E}[Wz + \mu + \varepsilon] = \mu.$$

- Covariances:

$$\Sigma_{zz} = \mathrm{Var}(z) = I_L.$$
$$\Sigma_{xx} = \mathrm{Var}(Wz + \varepsilon) = W\mathrm{Var}(z)\,W^T + \mathrm{Var}(\varepsilon) = WW^T + \Psi.$$
$$\Sigma_{zx} = \mathrm{Cov}(z, x) = \mathrm{Cov}(z, Wz + \varepsilon) = \mathrm{Cov}(z, Wz) = \mathrm{Var}(z)\,W^T = W^T.$$
$$\Sigma_{xz} = W.$$

So

$$\begin{pmatrix} z \\ x \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ \mu \end{pmatrix}, \begin{pmatrix} I_L & W^T \\ W & WW^T + \Psi \end{pmatrix} \right).$$

2. Show that the mean vector and covariance matrix $z \mid x$ can be simply written as

$$z \mid x \sim \mathcal{N}\left(\mu_{z|x}, \Sigma_{z|x}\right)$$

with

$$\begin{aligned} \mu_{z|x} &= \mu_z + \Sigma_{zx}\Sigma_{xx}^{-1}(x - \mu_x) = W^T(WW^T + \Psi)^{-1}(x - \mu) \\ \Sigma_{z|x} &= \Sigma_{zz} - \Sigma_{zx}\Sigma_{xx}^{-1}\Sigma_{xz} = I_L - W^T(WW^T + \Psi)^{-1}W. \end{aligned}$$

a. Covariance matrix: show that $\Sigma_{z|x} = (I_L + W^T\Psi^{-1}W)^{-1}$

Solution

Now we rewrite these into your two requested forms.
First show that $\Sigma_{z|x} = (I_L + W^T\Psi^{-1}W)^{-1}$
It is a direct application of the Woodbury identity.
Identify $A = I_L$, $U = W$, $C = \Psi^{-1}$, $V = W^T$. Then $(A + UCV)^{-1} = (I_L + W^T\Psi^{-1}W)^{-1}$, and from Woodbury we have

$$(I_L + W^T\Psi^{-1}W)^{-1} = I_L - W^T(\Psi + WW^T)^{-1}W.$$

Then

$$\boxed{\Sigma_{z|x} = (I_L + W^T\Psi^{-1}W)^{-1} \equiv S}$$

which is exactly what we want.

b. Mean vector: show that

$$\mu_{z|x} = SW^T\Psi^{-1}(x - \mu)$$

where $S = (I_L + W^T\Psi^{-1}W)^{-1}$.

Solution

We already have

$$\mu_{z|x} = W^T(WW^T + \Psi)^{-1}(x - \mu).$$

Apply Woodbury to $(WW^T + \Psi)^{-1}$:

$$(WW^T + \Psi)^{-1} = \Psi^{-1} - \Psi^{-1}W(I_L + W^T\Psi^{-1}W)^{-1}W^T\Psi^{-1}.$$

Let $S = (I_L + W^T\Psi^{-1}W)^{-1}$. Then

$$W^T(WW^T + \Psi)^{-1} = W^T\Psi^{-1} - (W^T\Psi^{-1}W)\,S\,W^T\Psi^{-1}.$$

Since $S = (I_L + W^T\Psi^{-1}W)^{-1}$, we have

$$(I_L + W^T\Psi^{-1}W)S = I_L \quad \Longrightarrow \quad I_L - (W^T\Psi^{-1}W)S = S.$$

Therefore

$$W^T(WW^T + \Psi)^{-1} = [I_L - (W^T\Psi^{-1}W)S]W^T\Psi^{-1} = S\,W^T\Psi^{-1}.$$

Multiplying by $(x - \mu)$ yields

$$\boxed{\mu_{z|x} = S\,W^T\Psi^{-1}(x - \mu)}.$$

2.2 Exercise: EM algorithm for FA

1. Write down the complete data log-likelihood $\log p(x, z \mid \theta)$, where $\theta = (W, \mu, \Psi)$.

Solution

Let us assume that $\mu = 0$ (centering of the x_i), the complete log-likelihood is

$$\begin{aligned}\log p(X, Z|\mu, W, \Psi) = & \sum_i \log \mathcal{N}_L(z_i; 0, I) + \log \mathcal{N}_D(x_i; Wz_i, \Psi) \\ = & -\frac{n}{2}\log|I_L| - \frac{n}{2}Tr(\hat{\Sigma}_{zz}) \\ & -\frac{n}{2}\log|\Psi| - \frac{n}{2}Tr(\hat{\Sigma}_{xx}\Psi^{-1}) + Cst\end{aligned}$$

where

$$\hat{\Sigma}_{xx} = \frac{1}{n}\sum_i (x_i - Wz_i)(x_i - Wz_i)^T$$

2. E-step: Compute the sufficient statistics $\mathbb{E}[z \mid x]$ and $\mathbb{E}[zz^T \mid x]$.

Solution

From the previous exercise, we have

$$\mathbb{E}[z \mid x] = \mu_{z|x} = SW^T\Psi^{-1}(x - \mu),$$

and

$$\mathbb{E}[zz^T \mid x] = \Sigma_{z|x} + \mu_{z|x}\mu_{z|x}^T = S + \mu_{z|x}\mu_{z|x}^T.$$

3. M-step: Derive the update equations for W, μ and Ψ.

Solution

To derive the M-step updates, we maximize the expected complete data log-likelihood with respect to the parameters. Taking derivatives and setting them to zero, we obtain the following updates: - Update for W:

$$W = \left(\sum_{n=1}^{N}(x_n - \mu)\mathbb{E}[z_n^T \mid x_n]\right)\left(\sum_{n=1}^{N}\mathbb{E}[z_n z_n^T \mid x_n]\right)^{-1}.$$

- Update for μ:

$$\mu = \frac{1}{N}\sum_{n=1}^{N}x_n - W\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[z_n \mid x_n].$$

- Update for Ψ:

$$\Psi = \frac{1}{N}\sum_{n=1}^{N}\left[(x_n - W\mathbb{E}[z_n \mid x_n] - \mu)(x_n - W\mathbb{E}[z_n \mid x_n] - \mu)^T + W\mathbb{E}[z_n z_n^T \mid x_n]W^T\right].$$

2.3 Exercise: Program EM for Factor Analysis

Solution

2.3.1 Initialisation via a PCA

```r
initialisation.FA<-function(X,L=1){
  # Return W and Psi
  d<-ncol(X)
  Sigmaxx<-var(X)
  W<-eigen(Sigmaxx)$vectors[,1:L]
  if (L==1) W<-cbind(W)
  Psi<-rep(1,d)
  return(list(W=W,Psi=Psi))
}
```

Solution

2.3.2 E step

```r
FA.E.step<-function(X,W,Psi){
  # X is assumed to be centered
  # M contain the contionnal expectation of the latent factor
  # S contains the covariance of the latent factor
  L<-ncol(W)
  S <- solve(diag(L) + t(W)%*%diag(1/Psi)%*%W)
  M<- X%*%diag(1/Psi)%*%W%*%S
  return(list(S=S,M=M))
}
```

Solution

2.3.3 M Step

```r
FA.M.step<-function(X,S,M,W,Psi){
  n<-nrow(X)
  Psi<-1/n*diag(t(X)%*%X -W%*%t(M)%*%X)
  W<- (t(X)%*%M)%*%solve(n*S+t(M)%*%M)
  return(list(Psi=Psi,W=W))
}
```

Solution

2.3.4 Computation of the criterion

```
log.likelihood.FA<-function(X,S,M,Psi,W){
  n<-nrow(X)
  Sigmax<-(t(X)%*%X-W%*%t(M)%*%X)
  return(-(sum(diag(S+t(M)%*%M/n))
           +log(det(diag(Psi)))+
             log(det(S))+
             1/n*sum(diag(Sigmax%*%diag(1/Psi)))))
}
```

Solution

2.3.5 Putting it all together

```
FA.EM<-function(X,L=1,max.iter=50){
  X<-scale(X,scale=FALSE);mu<-attr(X,"scaled:center")
  log.likelihood<-NULL; init<-initialisation.FA(X,L)
  W<-init$W;  Psi<-init$Psi; criterion<- Inf; iteration<-1;
  log.likelihood[iteration]<--Inf
  while ((criterion>1e-6)&&(iteration<=max.iter)){
    E.step<-FA.E.step(X,W,Psi); E.step$S->S; E.step$M->M
    M.step<-FA.M.step(X,S,M,W,Psi); M.step$Psi->Psi; M.step$W->W
    iteration<-iteration+1
    log.likelihood[iteration]<-log.likelihood.FA(X,S,M,Psi,W)
    criterion<-abs((log.likelihood[iteration] - log.likelihood[iteration-1])/max(log.likeli
  }
  return(list( W=data.frame(W), Psi=Psi,
               M=data.frame(M), S=S,mu=mu,
               log.likelihood= log.likelihood[-1]))
}
```

2.4 Exercise: mtcars dataset with Factor Analysis

`mtcars` is a classic dataset in R, containing various attributes of different car models:

- Miles per gallon (mpg)

- Number of cylinders (cyl)
- Displacement (disp)
- Horsepower (hp)
- Rear axle ratio (drat)
- Weight (wt)
- Quarter mile time (qsec)
- Engine type (vs)
- Transmission (am)
- Number of forward gears (gear)
- Number of carburetors (carb)

In this exercise, we will compare the results of Principal Component Analysis (PCA) and Factor Analysis (FA) on the `mtcars` dataset.

1. Load the `mtcars` dataset and select a subset of continuous variables for analysis (e.g., `mpg`, `disp`, `hp`, `drat`, `wt`, `qsec`).

Solution

```
# ============================================================
# PCA vs Factor Analysis on a real dataset (mtcars)
# Goal: illustrate different loadings and the role of communalities
# ============================================================


# Packages to install if needed
pkgs <- c("psych", "ggplot2", "reshape2")

to_install <- pkgs[!pkgs %in% installed.packages()[, "Package"]]

if (length(to_install) > 0) {
  install.packages(to_install, dependencies = TRUE)
}

library(psych)
library(ggplot2)
```

```
Attachement du package : 'ggplot2'

Les objets suivants sont masqués depuis 'package:psych':

    %+%, alpha
```

```r
library(reshape2)

# ------------------------------------------------------------
# 1) Data: choose a small set of continuous-ish variables
# ------------------------------------------------------------
vars <- c("mpg","disp","hp","drat","wt","qsec")
X <- mtcars[, vars]

# Use correlation matrix (standard in PCA/FA comparisons)
Xsc <- scale(X)
R <- cor(Xsc)
knitr::kable(round(head(mtcars), 2), caption = "mtcars dataset (first 6 rows)")
```

Table 2.1: mtcars dataset (first 6 rows)

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

2. Perform PCA on the standardized data and extract the first two principal components. Then, perform Factor Analysis with 2 factors using maximum likelihood estimation and varimax rotation. Compare the loadings obtained from PCA and FA, and compute communalities and uniqueness for FA. Use the `psych` package for Factor Analysis. psych

Communalities in FA represent the proportion of each variable's variance that can be explained by the common factors. They provide insight into how well the factors capture the underlying structure of the data.

Communality for variable j is defined as:

$$\text{Communality}_j = \sum_{k=1}^{L} \lambda_{jk}^2,$$

where λ_{jk} is the loading of variable j on factor k, and L is the number of factors.

Uniqueness is the portion of variance that is unique to each variable and not explained by

the common factors. It reflects the specific variance of each variable that is not shared with others.

Uniqueness and communalities are related as follows:

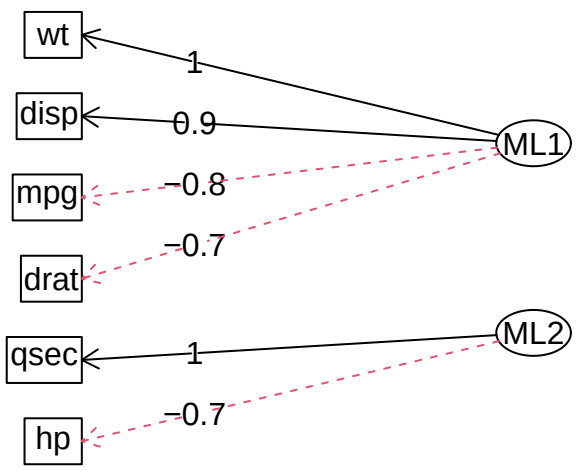
$$\text{Communality}_j = 1 - \text{Uniqueness}_j$$

Solution

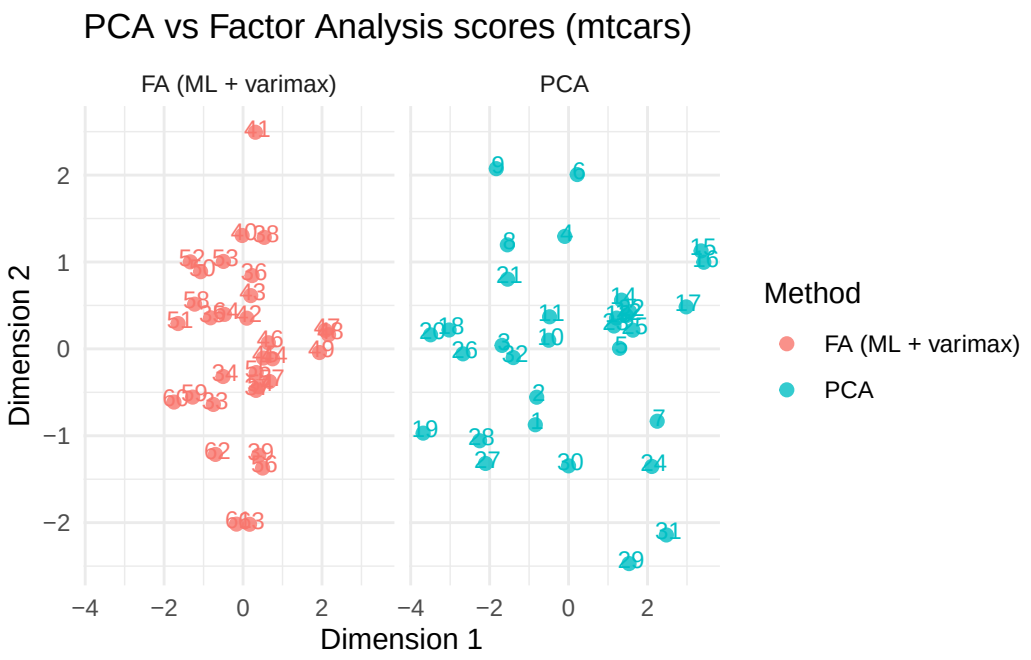
```
# ------------------------------------------------------------
# 2) PCA (on standardized data)
# ------------------------------------------------------------
pca <- prcomp(Xsc, center = FALSE, scale. = FALSE)

# PCA loadings = eigenvectors (columns) of correlation matrix
pca_load <- pca$rotation[, 1:2, drop = FALSE]

# Variance explained (for context)
pca_var <- (pca$sdev^2) / sum(pca$sdev^2)
pca_var[1:2]
```

```
[1] 0.6978994 0.1913520
```

```
# ------------------------------------------------------------
# 3) Factor Analysis: 2 factors, ML, varimax rotation
# ------------------------------------------------------------
fa2 <- fa(Xsc, nfactors = 2, fm = "ml", rotate = "varimax",scores="regression")

# FA rotated loadings
fa_load <- as.matrix(unclass(fa2$loadings))[, 1:2, drop = FALSE]

# Communalities (how well each variable is explained by the factors)
comm <- fa2$communality

## Visualize FA results with diagram
fa.diagram(fa2)
```




```
# ------------------------------------------------------------
# 4) Plot projection side-by-side (PCA vs FA)
# ------------------------------------------------------------
library(ggplot2)
pca_scores <- as.data.frame(pca$x[, 1:2])
colnames(pca_scores) <- c("Dim1", "Dim2")
pca_scores$Method <- "PCA"
fa_scores <- as.data.frame(fa2$scores)
colnames(fa_scores) <- c("Dim1", "Dim2")
fa_scores$Method <- "FA (ML + varimax)"
scores <- rbind(pca_scores, fa_scores)
scores$Ind <- seq_len(nrow(scores))
ggplot(scores, aes(x = Dim1, y = Dim2, color = Method)) +
geom_text(aes(label = Ind),
                  size = 3,
                  nudge_x = 0.05,
                  nudge_y = 0.05,
                  show.legend = FALSE) +
   geom_point(size = 2, alpha = 0.8) +  facet_wrap(~ Method) +
  labs(title = "PCA vs Factor Analysis scores (mtcars)",
       x = "Dimension 1", y = "Dimension 2") +
  theme_minimal()
```




```
# ------------------------------------------------------------
# 5) Put loadings side-by-side (table)
# ------------------------------------------------------------
tab <- data.frame(
  variable = rownames(pca_load),
  PCA1 = round(pca_load[,1], 3),
  PCA2 = round(pca_load[,2], 3),
  FA1  = round(fa_load[,1], 3),
  FA2  = round(fa_load[,2], 3),
  communality = round(comm, 3),
  uniqueness  = round(1 - comm, 3)
)

knitr::kable(tab, caption = "PCA vs Factor Analysis loadings (mtcars)")
```

Table 2.2: PCA vs Factor Analysis loadings (mtcars)

	variable	PCA1	PCA2	FA1	FA2	communality	uniqueness
mpg	mpg	-0.459	-0.059	-0.842	0.376	0.850	0.150
disp	disp	0.466	0.061	0.864	-0.389	0.898	0.102
hp	hp	0.426	-0.361	0.596	-0.698	0.843	0.157
drat	drat	-0.367	-0.437	-0.745	0.050	0.558	0.442

wt	wt	0.439	0.300	0.974	-0.116	0.962	0.038
qsec	qsec	-0.253	0.763	-0.065	0.962	0.930	0.070

```r
# ------------------------------------------------------------
# 5) Visualization: compare loadings (PCA vs FA) as heatmaps
# ------------------------------------------------------------
pca_long <- melt(pca_load)
colnames(pca_long) <- c("variable","component","loading")
pca_long$method <- "PCA"

fa_long <- melt(fa_load)
colnames(fa_long) <- c("variable","component","loading")
fa_long$method <- "FA (ML + varimax)"

long <- rbind(pca_long, fa_long)

# Order variables by absolute loading on the first dimension of each method
# (optional: keeps plots readable)
long$variable <- factor(long$variable, levels = rev(vars))

ggplot(long, aes(x = component, y = variable, fill = loading)) +
  geom_tile(color = "white") +
  facet_wrap(~ method) +
  geom_text(aes(label = sprintf("%.2f", loading)), size = 3) +
  labs(title = "Loadings comparison: PCA vs Factor Analysis (mtcars)",
       x = "", y = "") +
  theme_minimal()
```

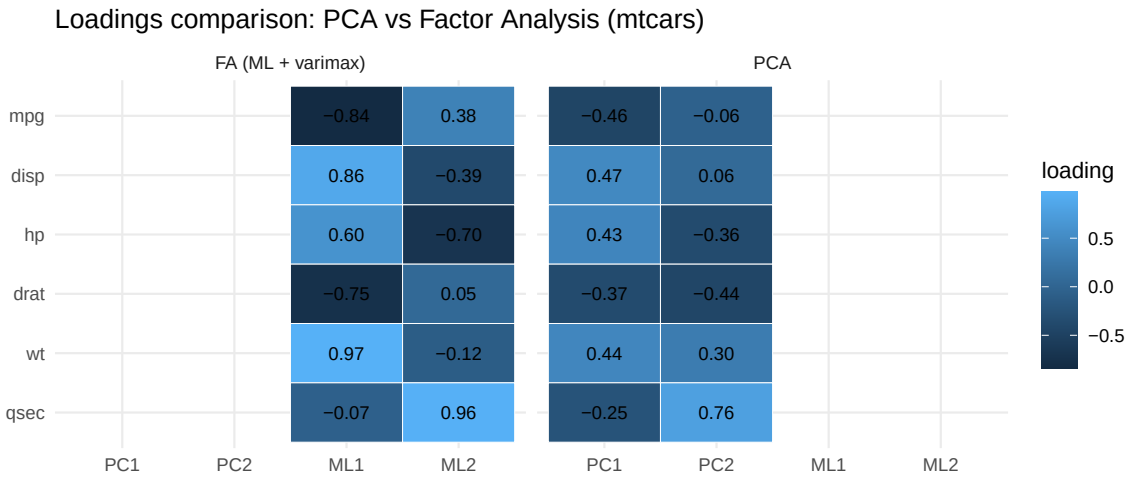


Figure 2.1: Loadings comparison: PCA vs Factor Analysis (mtcars)

3. Interpret the results: Discuss the differences in loadings between PCA and FA, and explain the significance of communalities and uniqueness in the context of FA.

	variable	PCA1	PCA2	FA1	FA2	communality	uniqueness
mpg	mpg	-0.459	-0.059	-0.842	0.376	0.850	0.150
disp	disp	0.466	0.061	0.864	-0.389	0.898	0.102
hp	hp	0.426	-0.361	0.596	-0.698	0.843	0.157
drat	drat	-0.367	-0.437	-0.745	0.050	0.558	0.442
wt	wt	0.439	0.300	0.974	-0.116	0.962	0.038
qsec	qsec	-0.253	0.763	-0.065	0.962	0.930	0.070

Solution

2.4.1 PCA loadings

Variance-optimal but conceptually blurred

variable	PCA1	PCA2
mpg	-0.46	-0.06
disp	0.47	0.06
hp	0.43	-0.36
drat	-0.37	-0.44
wt	0.44	0.30
qsec	-0.25	0.76

PC1 mixes together

- *Size & power*: `disp`, `hp`, `wt`
- *Efficiency*: `mpg`
- *Gearing*: `drat`

→ PC1 is a **mathematical axis of maximal variance**, not a clean concept.
PC2 is dominated by `qsec` but still polluted by `hp`, `drat`, `wt`.
PCA issues

- Components may be hard to name
- Cross-loadings everywhere
- No way to detect poorly represented variables

2.4.2 FA loadings

Latent structure appears

variable	FA1	FA2
mpg	-0.84	0.38
disp	0.86	-0.39
hp	0.60	-0.70
drat	-0.75	0.05
wt	0.97	-0.12
qsec	-0.07	0.96

2.4.2.1 Factor interpretation

Factor 1 – Mass / Power / Inefficiency

- `wt`, `disp`, `hp` strongly positive
- `mpg`, `drat` strongly negative

→ Heavy, powerful cars vs light efficient cars.
Factor 2 – Acceleration / Transmission dynamics

- `qsec` 1
- `hp` moderately negative

→ Clean acceleration–performance dimension.

2.4.3 Communalities and uniqueness

What FA adds that PCA cannot

variable	communality	uniqueness
mpg	0.850	0.150
disp	0.898	0.102
hp	0.843	0.157
drat	0.558	0.442
wt	0.962	0.038
qsec	0.930	0.070

Definitions

- h_i^2 (communality): proportion of variance explained by latent factors
- u_i^2 (uniqueness): variable-specific variance + noise

Examples

- `wt` – communality 0.962
  → almost entirely explained → *excellent indicator*
- `qsec` – 0.93
  → nearly pure expression of Factor 2
- `drat` – communality 0.558, uniqueness 0.442
  → almost half its variance is idiosyncratic → *poor latent measurement*

PCA would force this variance into components and hide the problem.

PCA	FA
Blurs concepts	Reveals latent mechanisms
Forces all variance into components	Separates signal from noise
No notion of measurement quality	Communalities help to diagnose variables
Components are algebraic	Factors are meaningful constructs

- **PCA asks**: *How can I compress the data?*
- **FA asks**: *What invisible causes generate the correlations?*

3 Independent Component analysis

Independent component analysis attempts to decompose a multivariate signal into

- independent
- non-Gaussian signals.

> **Outline**
>
> In this chapter we will:
>
> 1. Illustrate ICA through the cocktail party problem.
> 2. Introduce the ICA generative model and its ambiguities.
> 3. Explain preprocessing: centering and whitening.
> 4. Derive ICA from a maximum likelihood perspective.
> 5. Show how non-Gaussianity leads to contrast functions.
> 6. Derive FastICA as a fixed-point algorithm.

3.1 Cocktail party problem

3.1.1 The cocktail party problem

The cocktail party problem illustrates the goal of Independent Component Analysis (ICA).

Several microphones record simultaneous mixtures of multiple speakers talking in the same room, but only the mixed signals are observed.

Assuming that the original sources are statistically independent and non-Gaussian, ICA aims to recover the individual source signals by finding linear projections of the observed data that maximize non-Gaussianity.

The number of microphones must be at least equal to the number of sources for ICA to work effectively.

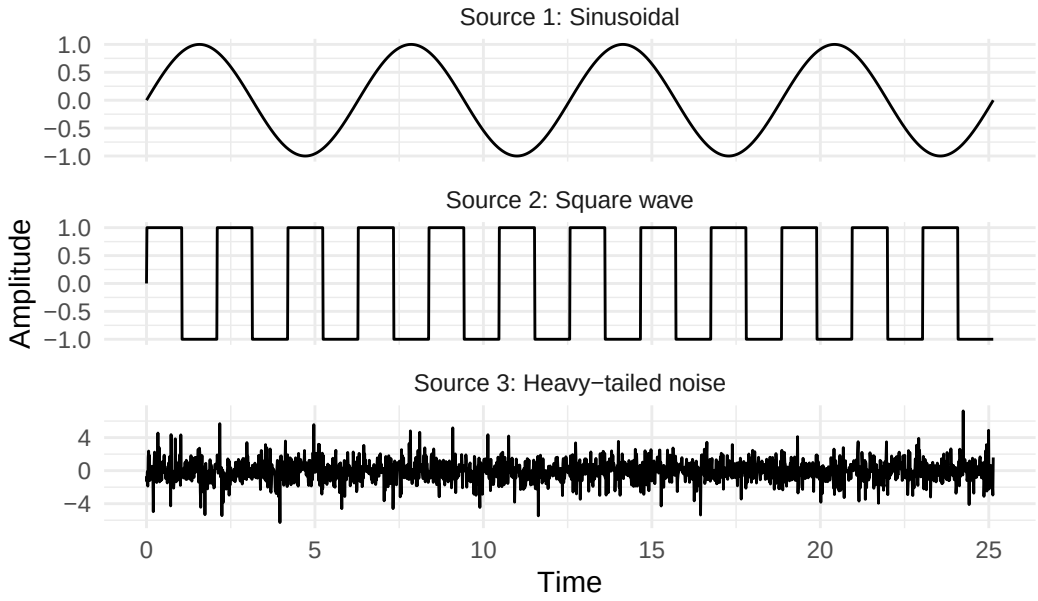


Figure: Three independent source signals (sinusoid, square wave, heavy−tailed noise).

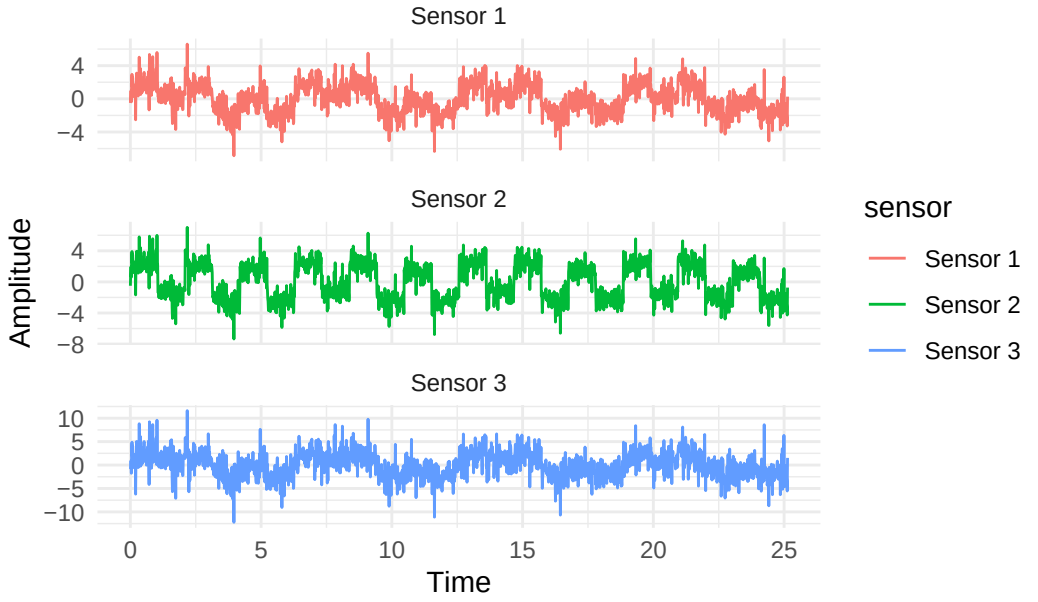
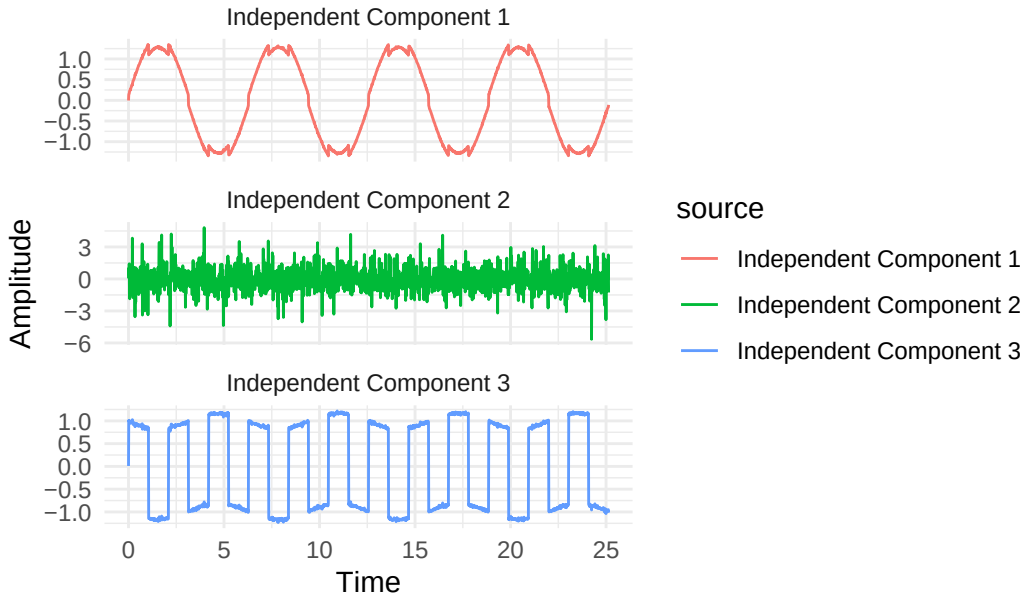


Figure: Observed mixtures at the three sensors after linear mixing.



: Estimated independent components recovered by FastICA.

3.2 ICA Historical context (French Wikipedia)

3.2.1 Blind source separation

The first formulation was carried out in 1984 by Jeanny Hérault and Bernard Ans, two researchers in neuroscience and signal processing, to model in the form of a neuromimetic network self-adaptive encoding and decoding of movement in humans.

3.2.2 France and Finland

- The French signal processing community adopted a statistical formalism
- While Finnish researchers aimed to extend principal component analysis by means of a connectionist formalism (1985)

3.2.3 Formalisation

- A first formalism of the blind source separation problem, as well as an algorithm making it possible to obtain a solution, was proposed by C. Jutten and J. Hérault in 1991.
- Mathematical formalization in the simplest case was carried out in 1994 by P. Comon

3.3 ICA Model

Let $x_t \in \mathbb{R}^D$ be the observed signal at the sensors at time t, and $z_t \in \mathbb{R}^L$ be the vector of source signals:

$$x_t = Wz_t + \epsilon_t$$

- W is an $D \times L$ matrix,
- $\epsilon_t \sim \mathcal{N}(0, \Psi)$.

The model is identical to factor analysis.

However, we will use a **different prior** for $p(z_t)$.

In the classical formulation the noise term is often omitted.

In FA, we assume all sources are uncorrelated, and has a Gaussian distribution. In this Gaussian case, the model is not identifiable: any orthogonal transformation of the sources yields an equivalent solution.

3.3.1 Relax this Gaussian assumption on latent variables (sources)

ICA assumes independence, which is stronger than uncorrelatedness,

$$p(z_t) = \prod_j p_j(z_{tj}).$$

3.3.2 Identifiability and ICA ambiguities

Two fundamental ambiguities remain even with perfect estimation:

- **Scaling/sign ambiguity**: if one source is scaled by a constant and the corresponding column of W is rescaled inversely, the mixture x is unchanged. Thus, source variances cannot be identified. A common convention is to fix each source to have unit variance, leaving only a sign ambiguity.
- **Permutation ambiguity**: the order of sources is not identifiable. Any permutation of components yields an equivalent model.

3.3.3 Additionnal assumptions

- Without loss of generality the variance of the source distributions is contrained to unity
- W is assumed square and hence invertible.

3.4 Preprocessing: centering and whitening

ICA is almost always preceded by preprocessing, which simplifies the estimation.

3.4.1 Centering

Subtract the mean so that $\mathbb{E}[x] = 0$. This makes the model zero-mean and simplifies equations. After estimation, the original mean can be recovered.

3.4.2 Whitening

Transform x so that its components are uncorrelated and have unit variance:

$$\mathbb{E}[\tilde{x}\,\tilde{x}^T] = I.$$

Using the eigenvalue decomposition of the covariance $\mathbb{E}[xx^T] = EDE^T$, a whitening transform is

$$\tilde{x} = D^{-1/2}E^T x.$$

Whitening turns the mixing matrix into an orthogonal matrix, which reduces the number of parameters to estimate and simplifies the optimization landscape.

3.5 Maximum likelihood estimation of ICA

If the data is centered and whitened, we have

$$\mathbb{E}[xx^t] = I = W\mathbb{E}[zz^T]W^T = WW^T$$

also have

Hence we see that W must be orthogonal. This reduces the number of parameters we have to estimate from D^2 to $D(D-1)/2$.

3.5.1 Recognition weights/Generative weights

Let $V = W^{-1}$ these are often called the recognition weights, as opposed to W, which are the generative weights

3.5.2 Log-likelihood

Since $x = Wz$, we have

$$p_x(Wz_t) = p_z(z_t)|det(W^{-1})| = p_z(Vx_t)|det(V)|$$

The log -likelihood of the data (on the whole population) is therefore

$$\ell(V) = \log|\det V| + \mathbb{E}\left[\sum_{j=1}^{D} \log p_j(y_j)\right], \qquad y_j = v_j^\top x.$$

3.6 ICA gradient ascent

3.6.1 Notation

- Data point: $x \in \mathbb{R}^D$ (assume centered and whitened, so $\mathbb{E}[x] = 0$ and $\mathbb{E}[xx^\top] = I$).
- Demixing matrix: $V \in \mathbb{R}^{D\times D}$, rows $v_j^\top$.
- Component (projection):
$$y_j = v_j^\top x, \qquad y = Vx.$$

Remark:

- $g(\cdot)$ below is a *scalar nonlinearity*, applied componentwise to y.

3.6.2 Log-likelihood objective (generic ICA)

Assume independent sources with densities p_j:

$$\ell(V) = \log|\det V| + \mathbb{E}\left[\sum_{j=1}^{D} \log p_j(y_j)\right], \qquad y_j = v_j^\top x.$$

Define the **score function** (derivative of log-density):

$$g_j(y_j) \;=\; \frac{d}{dy_j} \log p_j(y_j).$$

Thus

$$g(y) = (g_1(y_1), \dots, g_D(y_D))^\top.$$

where $g(y)$ is the vector of score functions applied componentwise to $y = Vx$.

3.6.3 Plain gradient (batch form)

The gradient of $\ell(V)$ is

$$\nabla_V \ell(V) = sign(det(V))V^{-\top} + \mathbb{E}[g(y)\, x^\top], \qquad y = Vx.$$

Two remarks:

- V is orhtogonal after whitening, $V^{-\top} = V$.
- The $sign(det(V))$ term is often dropped in practice, as it does not affect the direction of the gradient. Thus, the **gradient ascent** step is

$$V \leftarrow V + \eta\,(V + \mathbb{E}[g(Vx)\, x^\top]),$$

where $\eta > 0$ is the learning rate.

3.6.4 Online (one-sample) approximation

Replacing $\mathbb{E}[\cdot]$ by a single sample (x) gives:

$$V \leftarrow V + \eta\,(V + g(Vx)\, x^\top).$$

3.6.5 Practical note: whitening and orthogonality constraint

After whitening, one typically **constrains** V to be orthogonal:

$$VV^\top = I,$$

and then $\log|\det V|$ is constant (often dropped).

In practice, we enforce the constraint by **re-orthogonalizing** V after an update (e.g., symmetric decorrelation).

A central question is how to choose the nonlinearity g (or equivalently the source densities p_j) to achieve good performance and convergence properties. This leads us to measures of non-Gaussianity, which are crucial for ICA estimation.

3.7 Measures of non-Gaussianity (contrast functions)

ICA estimation is transformed into an optimization problem by choosing a measure of non-Gaussianity (a **contrast function**).

3.7.1 Kurtosis

For a zero-mean, unit-variance variable y, kurtosis is

$$\text{kurt}(y) = \mathbb{E}[y^4] - 3.$$

- **Gaussian variables have zero kurtosis**.
- **Super-Gaussian** distributions have positive kurtosis (spiky, heavy-tailed).
- **Sub-Gaussian** distributions have negative kurtosis (flat distributions).

Kurtosis is simple and has additive properties for independent variables, but it is sensitive to outliers, making it less robust in practice.

3.7.2 Negentropy

Negentropy uses information theory. Differential entropy is maximized by a Gaussian distribution for fixed covariance, so a measure of non-Gaussianity is

$$J(y) = H(y_{\text{gauss}}) - H(y) \geq 0.$$

Negentropy is zero iff the distribution is Gaussian. It is theoretically well-justified but hard to compute directly, so we rely to approximations.

3.7.3 Approximations of negentropy

A practical approximation uses

$$J(y) \approx \left(\mathbb{E}[G(y)] - \mathbb{E}[G(\nu)]\right)^2, \quad \nu \sim \mathcal{N}(0,1).$$

Common choices include:

- $G(u) = \log\cosh(u)$ for robustness with super-Gaussian sources.
- $G(u) = -\exp(-u^2/2)$ for sub-Gaussian sources.

3.7.4 Common choice of nonlinearity (example)

i log cosh nonlinearity

thus for a super-Gaussian prior:

$$G(u) = \log\cosh(u), \qquad g(u) = G'(u) = \tanh(u).$$

3.8 From gradient ascent to a fixed-point strategy (FastICA)

The gradient ascent approach derived above is statistically correct but leads to a slow algorithm due to the presence of matrix inverses and the need to tune a learning rate. FastICA overcomes these difficulties by **directly solving the stationarity condition** of the likelihood using a **fixed-point iteration**.

3.8.1 Stationarity condition (one component)

Consider a single row $v^\top$ of the demixing matrix V, and assume the data are centered and whitened:

$$\mathbb{E}[x] = 0, \qquad \mathbb{E}[xx^\top] = I.$$

We seek to maximize the contrast

$$J(v) = \mathbb{E}\big[G(v^\top x)\big] \quad \text{subject to } v^\top v = 1,$$

where $g(u) = G'(u)$.

Introducing a Lagrange multiplier λ, the stationarity condition is

$$\nabla_v \left(\mathbb{E}[G(v^\top x)] - \lambda(v^\top v - 1)\right) = 0,$$

which yields

$$\mathbb{E}[x\, g(v^\top x)] = 2\lambda v.$$

This equation already has the form of a **fixed-point condition**: the optimal direction v is proportional to $\mathbb{E}[x\, g(v^\top x)]$.

This can be rewritten as

$$v = h(v) \equiv \frac{1}{2\lambda}\mathbb{E}[x\, g(v^\top x)]$$

Let us Define **the Residual Function**

$$R(v) = v - h(v) = v - \frac{1}{2\lambda}\mathbb{E}[x\, g(v^\top x)]$$

We want to find v^* such that $R(v^*) = 0$.

We construct an **objective function**

$$F(v) = \frac{1}{2}\|R(v)\|^2 = \frac{1}{2}\left\|v - \frac{1}{2\lambda}\mathbb{E}[x\, g(v^\top x)]\right\|^2$$

and compute the gradient of F:

$$\nabla_v F(v) = (I - J_h(v))^\top R(v)$$

where $J_h(v)$ is the Jacobian of $h(v)$.

Computing the Jacobian:

$$J_h(v) = \frac{\partial h}{\partial v} = \frac{1}{2\lambda}\frac{\partial}{\partial v}\mathbb{E}[x\, g(v^\top x)]$$

Using the chain rule:

$$J_h(v) = \frac{1}{2\lambda}\mathbb{E}[x\, g'(v^\top x)x^\top]$$

Under the **whitening assumption** $\mathbb{E}[xx^\top] = I$:

$$J_h(v) \approx \frac{1}{2\lambda}\mathbb{E}[g'(v^\top x)]\, I$$

Newton's method for minimizing $F(v)$ gives:

$$v_{k+1} = v_k - [H_F(v_k)]^{-1}\nabla_v F(v_k)$$

Using the **Gauss-Newton approximation** for the Hessian:

$$H_F(v) \approx (I - J_h(v))^\top (I - J_h(v))$$

Substituting our Jacobian approximation:

$$H_F(v) \approx \left(I - \frac{1}{2\lambda}\mathbb{E}[g'(v^\top x)]I\right)^\top \left(I - \frac{1}{2\lambda}\mathbb{E}[g'(v^\top x)]I\right)$$

$$= \left(1 - \frac{1}{2\lambda}\mathbb{E}[g'(v^\top x)]\right)^2 I$$

The Newton iteration becomes:

$$v_{k+1} = v_k - \frac{1}{\left(1 - \frac{1}{2\lambda}\mathbb{E}[g'(v^\top x)]\right)^2}\left(1 - \frac{1}{2\lambda}\mathbb{E}[g'(v^\top x)]\right) R(v_k)$$

Simplifying:

$$v_{k+1} = v_k - \frac{1}{1 - \frac{1}{2\lambda}\mathbb{E}[g'(v^\top x)]} R(v_k)$$

Absorbing the Lagrange Multiplier

Since we enforce $\|v\| = 1$ by normalization after each step, we can absorb the constant 2λ into the iteration. Setting the iteration to directly solve:

$$v_{k+1} = \mathbb{E}[x\, g(v_k^\top x)] - \mathbb{E}[g'(v_k^\top x)]v_k$$

The final FastICA update becomes:

$$\boxed{v \leftarrow \mathbb{E}[x\, g(v^\top x)] - \mathbb{E}[g'(v^\top x)]v}$$

Followed by normalization:

$$\boxed{v \leftarrow \frac{v}{\|v\|}}$$

3.8.2 Remarks on FastICA derivation

1. **From residual minimization to fixed-point**: The framework $F(x) = \frac{1}{2}\|x - h(x)\|^2$ transforms the stationarity condition into an optimization problem.
2. **Newton acceleration**: Rather than simple gradient descent, Newton's method uses second-order information (Hessian) for faster convergence.
3. **Whitening simplification**: The whitening assumption allows the Jacobian to be approximated as a scalar multiple of identity, greatly simplifying the Hessian.
4. **No learning rate needed**: Unlike gradient descent, the Newton update has an adaptive step size built in, eliminating the need for hyperparameter tuning.

FastICA algorithm for one component

1. Center and whiten the data.
2. Initialize v with $\|v\| = 1$.
3. Repeat until convergence:

$$v \leftarrow \mathbb{E}[xg(v^\top x)] - \mathbb{E}[g'(v^\top x)]v$$

$$v \leftarrow v/\|v\|$$

4. Output $z = v^\top x$.

4 Exercices on independent component analyses

4.1 Exercice: Gaussian distribution and entropy

Show that among all continuous distributions with a given variance, the Gaussian distribution has the largest differential entropy.

Solution

We want to show that for a random variable X with variance σ^2, the differential entropy $H(X)$ is maximized when X follows a Gaussian distribution. The differential entropy of a continuous random variable X with probability density function $p(x)$ is defined as:

$$H(X) = -\int p(x) \log p(x)\, dx.$$

To prove that the Gaussian distribution maximizes the entropy for a given variance, we can use the method of Lagrange multipliers. We want to maximize $H(X)$ subject to the constraints that the variance is fixed: 1. $\int p(x)\, dx = 1$ (normalization constraint) 2. $\int x^2 p(x)\, dx = \sigma^2$ (variance constraint) We set up the Lagrangian:

$$\mathcal{L} = -\int p(x) \log p(x)\, dx + \lambda_1 \left(\int p(x)\, dx - 1 \right) + \lambda_2 \left(\int x^2 p(x)\, dx - \sigma^2 \right).$$

Taking the functional derivative of $\mathcal{L}$ with respect to $p(x)$ and setting it to zero gives:

$$-\log p(x) - 1 + \lambda_1 + \lambda_2 x^2 = 0.$$

Solving for $p(x)$, we find:

$$p(x) = \exp(\lambda_1 - 1) \exp(-\lambda_2 x^2).$$

This is the form of a Gaussian distribution. The constants λ_1 and λ_2 can be determined by enforcing the normalization and variance constraints. Thus, we conclude that the Gaussian distribution maximizes the differential entropy for a given variance.

4.2 Exercice: Orthogonality of the mixing matrix in ICA

Show that for centered and whitened data, the maximum likelihood estimate of the ICA model leads to an orthogonal mixing matrix.

Solution

For centered and whitened data, we have

$$\mathbb{E}[xx^t] = I = W\mathbb{E}[zz^T]W^T = WW^T$$

hence we see that W must be orthogonal. This reduces the number of parameters we have to estimate from D^2 to $D(D-1)/2$.

4.3 Exercice: Blind source separation with ICA

Each question below can be implemented in R or Python, using the FastICA algorithm from the `fastICA` package in R or `sklearn.decomposition.FastICA` in Python. Illustrate your results with appropriate plots.

1. Simulate three independent source signals:

- A sinusoidal signal
- A square wave signal
- A heavy-tailed noise signal (e.g., Student's t-distribution with low degrees of freedom)

2. Mix these sources using a the following mixing matrix:

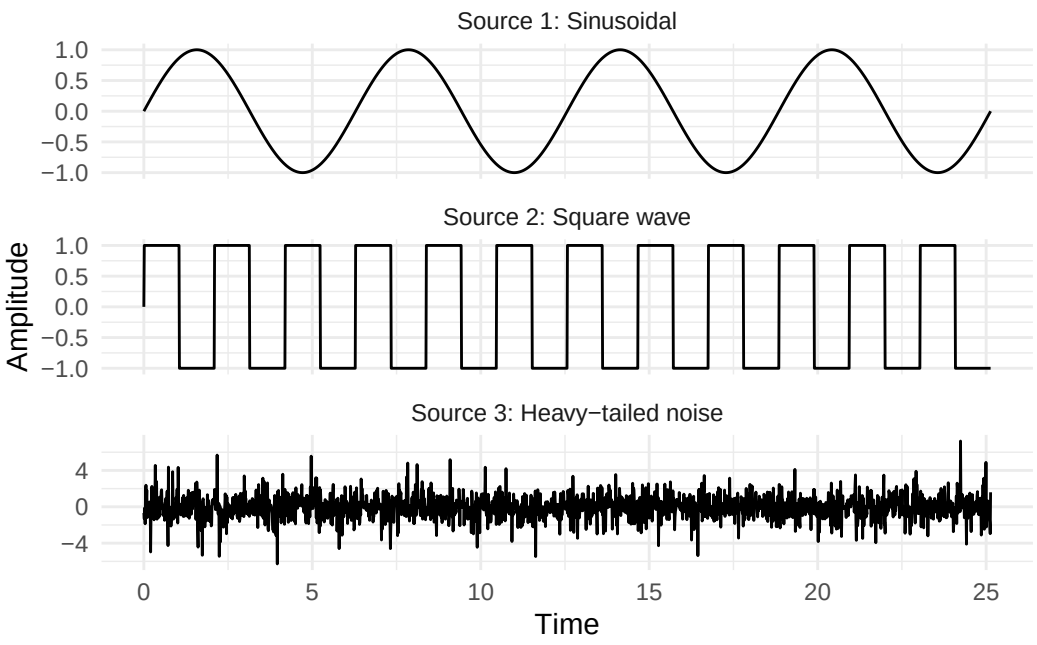
$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0.5 & 2 & 1 \\ 1.5 & 1 & 2 \end{pmatrix}$$

3. Apply the FastICA algorithm to the mixed signals to recover the independent components.

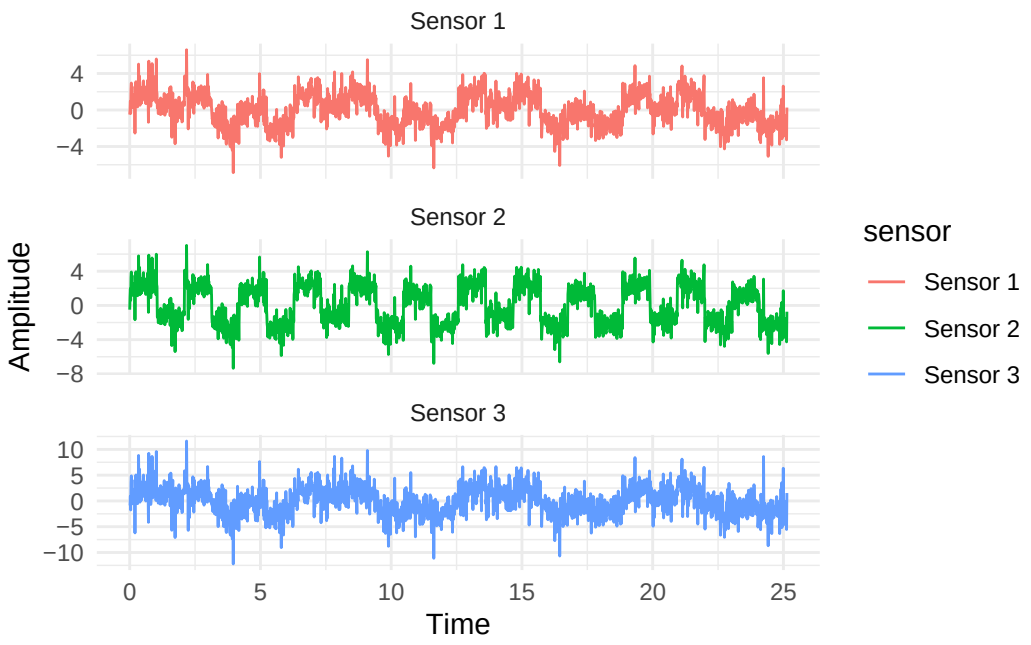
Solution

4.3.1 fastICA in R

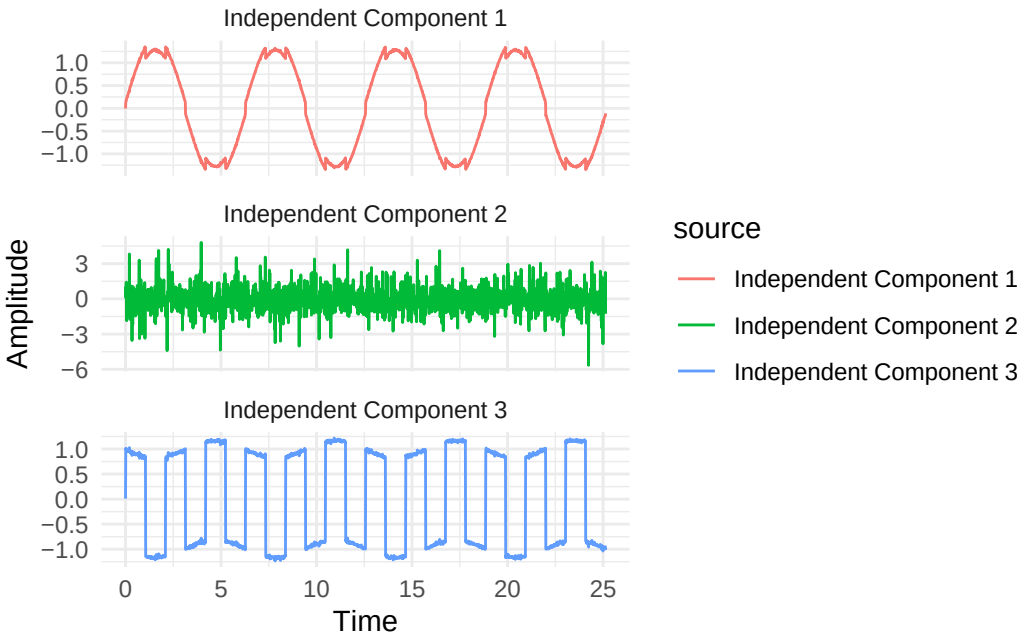
```r
library(ggplot2)
library(fastICA)

set.seed(123)
n <- 2000
t <- seq(0, 8*pi, length.out = n)

S <- cbind(
  t,
  S1=sin(t),                  # sinusoidal
  S2=sign(sin(3 * t)),        # square wave
  S3=rt(n,df=5)                    # heavy-tailed noise
)

S_df <- as.data.frame(S)

S_long <- data.frame(
  t = S_df$t,
  value = c(S_df$S1, S_df$S2, S_df$S3),
  source = factor(
    rep(c("Source 1: Sinusoidal",
          "Source 2: Square wave",
          "Source 3: Heavy-tailed noise"),
        each = n)
  )
)
ggplot(S_long, aes(x = t, y = value)) +
  geom_line() +
  facet_wrap(~ source, ncol = 1, scales = "free_y") +
  labs(x = "Time", y = "Amplitude") +
  theme_minimal()
```




```
S <- S[, -1]  # remove time
S <- scale(S)  # standardization
# mix the sources
A <- matrix(c(1, 1, 1,
              0.5, 2, 1,
              1.5, 1, 2),
            nrow = 3, byrow = TRUE)

X <- S %*% t(A)

ggplot(data.frame(t = rep(t, 3),
                  value = c(X[,1], X[,2], X[,3]),
                  sensor = factor(
                    rep(c("Sensor 1", "Sensor 2", "Sensor 3"),
                        each = n)
                  )),
       aes(x = t, y = value, color = sensor)) +
  geom_line() +
  facet_wrap(~ sensor, ncol = 1, scales = "free_y") +
  labs(x = "Time", y = "Amplitude") +
  theme_minimal()
```



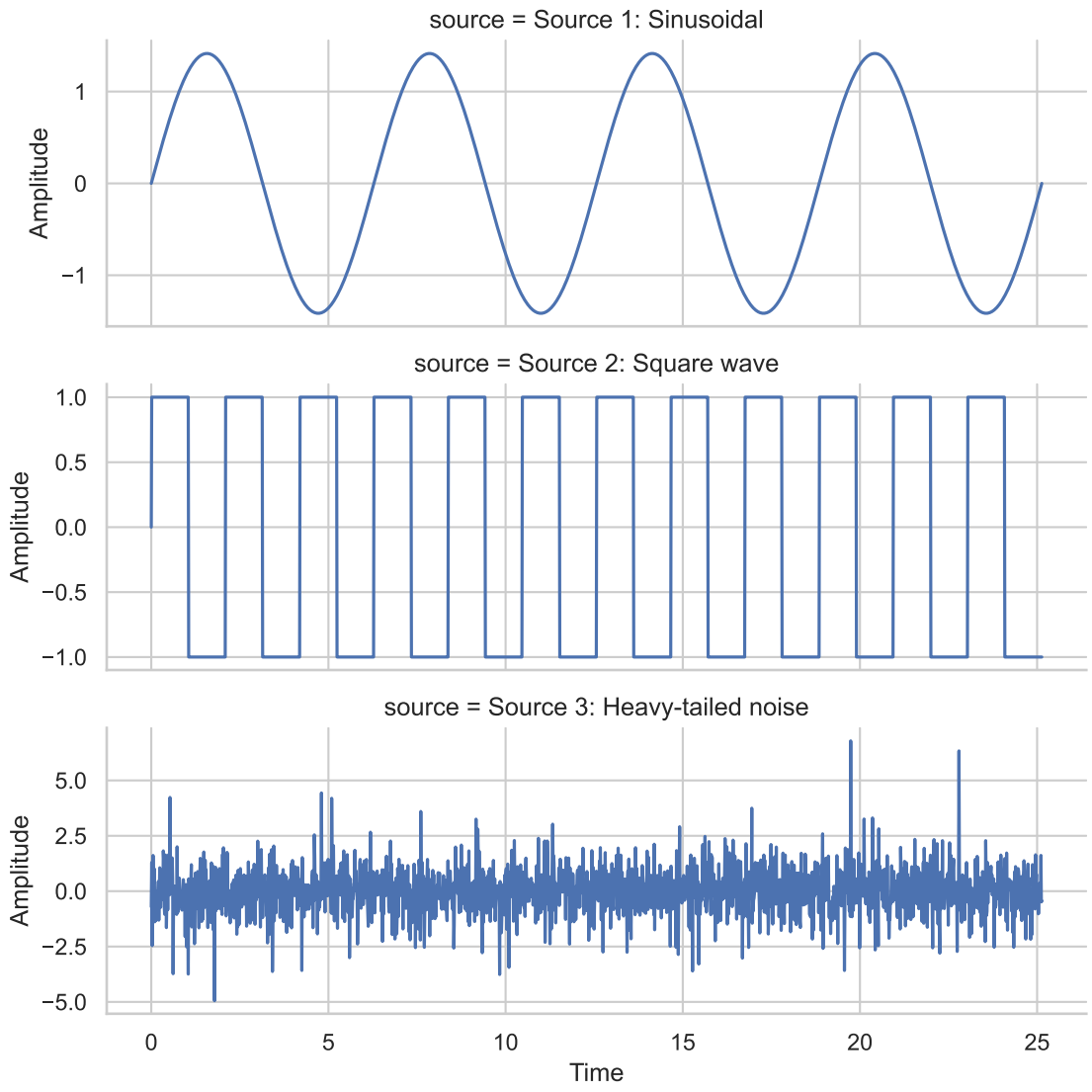

```
# apply ICA
ica <- fastICA(X, n.comp = 3)
Sica<-data.frame(t=t,ica$S)
Sica_long=data.frame(
  t = rep(Sica$t, 3),
  value = c(Sica$X1, Sica$X2, Sica$X3),
  source = factor(
    rep(c("Independent Component 1",
          "Independent Component 2",
          "Independent Component 3"),
        each = n)
  )
)

ggplot(Sica_long, aes(x = t, y = value, color = source)) +
  geom_line() +
  facet_wrap(~ source, ncol = 1, scales = "free_y") +
  labs(x = "Time", y = "Amplitude") +
  theme_minimal()
```



Solution

4.3.2 fastICA in python

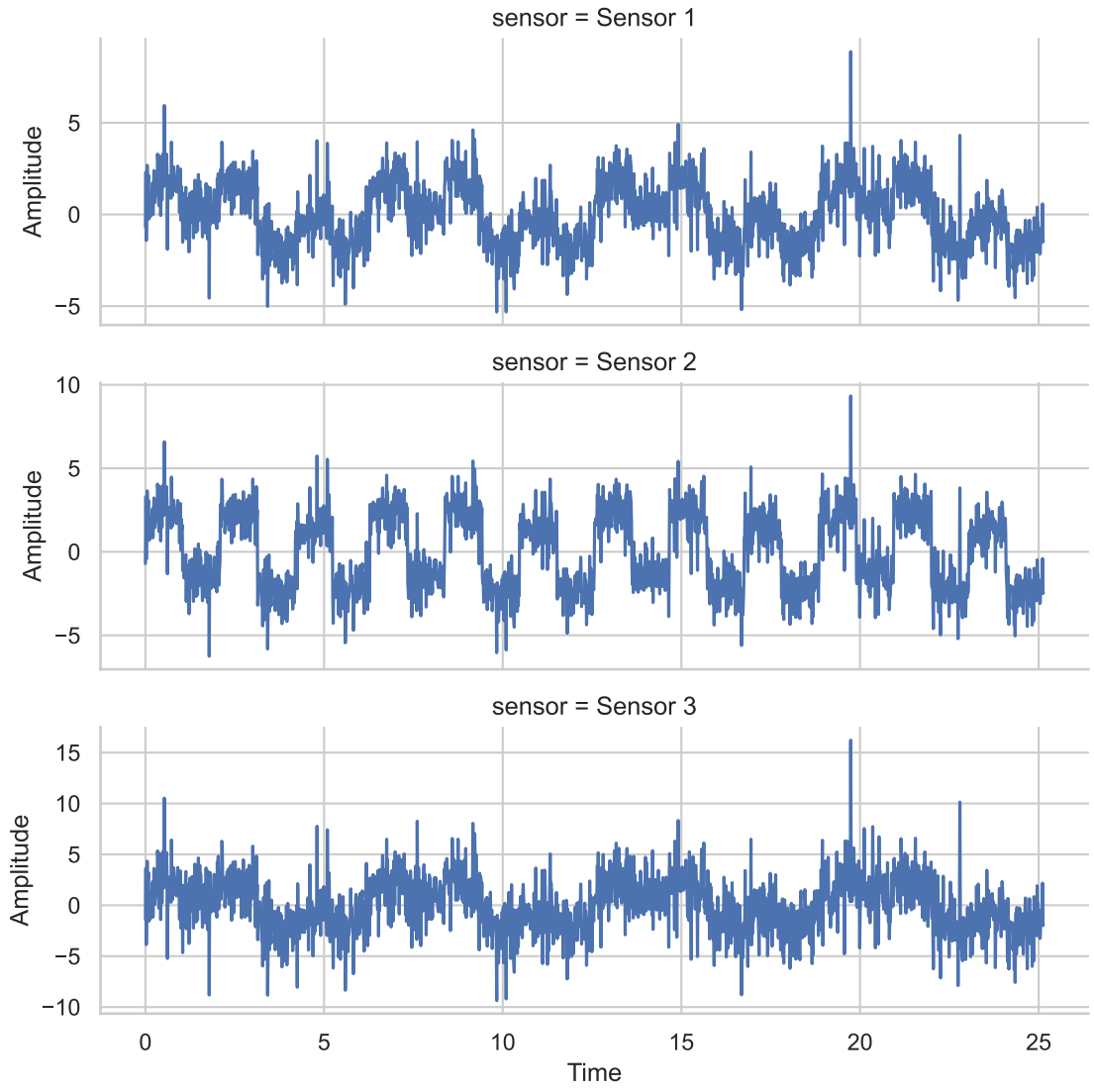
```
# === ICA demo: sources → mixing → FastICA → recovery ===

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import t as student_t
from sklearn.decomposition import FastICA

# Setup
np.random.seed(123)
sns.set_theme(style="whitegrid")

# Generate independent sources
n = 2000
t = np.linspace(0, 8*np.pi, n)

S = np.column_stack([
    np.sin(t),                    # Source 1: sinusoidal
    np.sign(np.sin(3 * t)),        # Source 2: square wave
    student_t.rvs(df=5, size=n)    # Source 3: heavy-tailed noise
])

# Standardize sources
S = (S - S.mean(axis=0)) / S.std(axis=0)

# Plot true sources
S_df = pd.DataFrame({
    "t": np.tile(t, 3),
    "value": np.concatenate([S[:,0], S[:,1], S[:,2]]),
    "source": np.repeat([
        "Source 1: Sinusoidal",
        "Source 2: Square wave",
        "Source 3: Heavy-tailed noise"
    ], n)
})

g = sns.FacetGrid(S_df, row="source", sharey=False, height=2.5, aspect=3);
g.map_dataframe(sns.lineplot, x="t", y="value");
g.set_axis_labels("Time", "Amplitude")
```



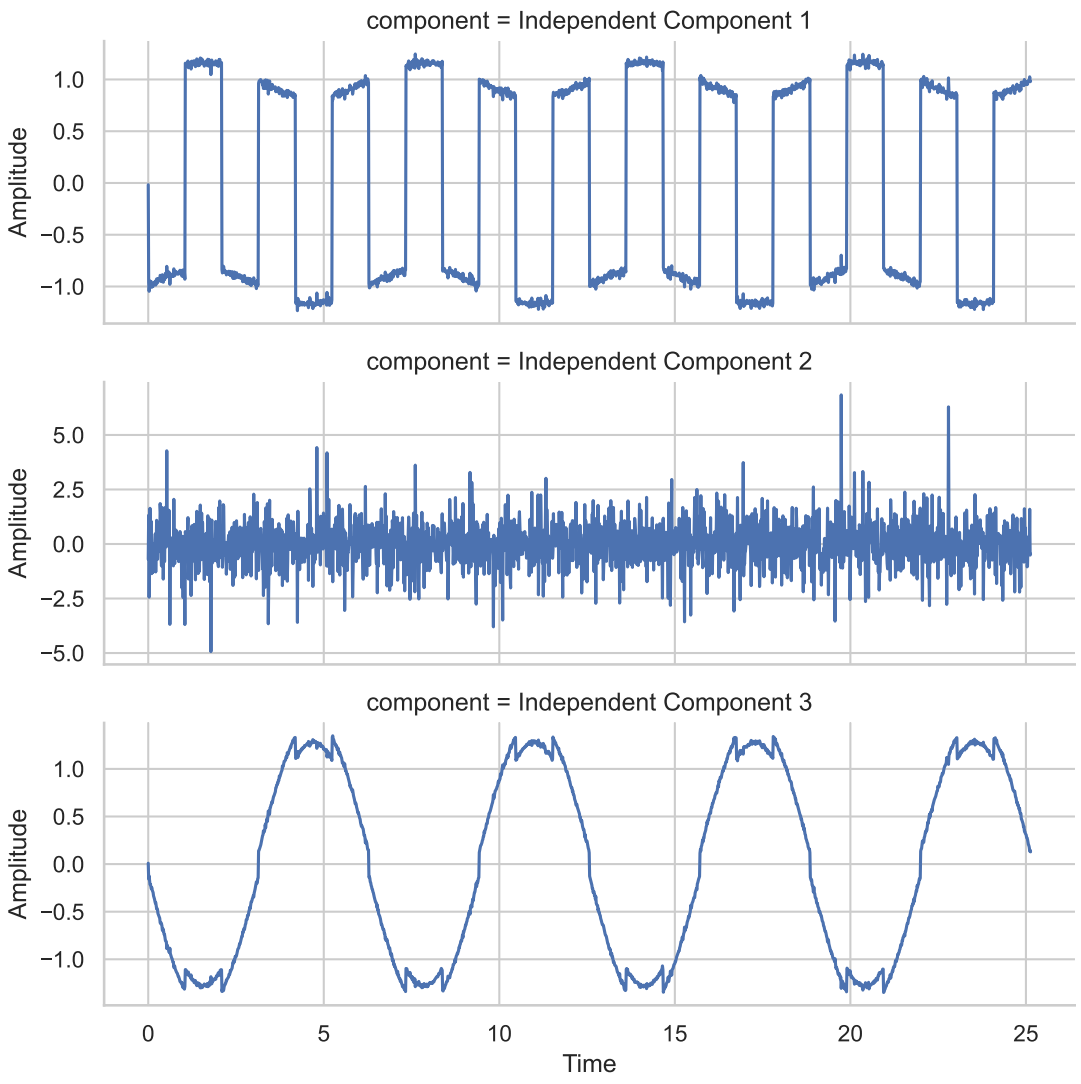

```
# Mixing matrix
A = np.array([
    [1,   1, 1],
    [0.5, 2, 1],
    [1.5, 1, 2]
])

# Mix sources
X = S @ A.T
```

```
<string>:3: RuntimeWarning: divide by zero encountered in matmul
<string>:3: RuntimeWarning: overflow encountered in matmul
<string>:3: RuntimeWarning: invalid value encountered in matmul
```

```
# Plot observed signals
X_df = pd.DataFrame({
    "t": np.tile(t, 3),
    "value": np.concatenate([X[:,0], X[:,1], X[:,2]]),
    "sensor": np.repeat([
        "Sensor 1",
        "Sensor 2",
        "Sensor 3"
    ], n)
})

g = sns.FacetGrid(X_df, row="sensor", sharey=False, height=2.5, aspect=3);
g.map_dataframe(sns.lineplot, x="t", y="value");
g.set_axis_labels("Time", "Amplitude")
```


sensor = Sensor 1
sensor = Sensor 2
sensor = Sensor 3
Amplitude
Time
5
0
−5
10
15
−10
0
5
10
15
20
25

```
# Apply FastICA
ica = FastICA(n_components=3, random_state=123)
S_ica = ica.fit_transform(X)

# Plot recovered independent components
Sica_df = pd.DataFrame({
    "t": np.tile(t, 3),
    "value": np.concatenate([S_ica[:,0], S_ica[:,1], S_ica[:,2]]),
    "component": np.repeat([
        "Independent Component 1",
        "Independent Component 2",
        "Independent Component 3"
    ], n)
})

g = sns.FacetGrid(Sica_df, row="component", sharey=False, height=2.5, aspect=3);
g.map_dataframe(sns.lineplot, x="t", y="value");
g.set_axis_labels("Time", "Amplitude");
None
```


component = Independent Component 1
Amplitude
1.0
0.5
0.0
−0.5
−1.0
component = Independent Component 2
Amplitude
5.0
2.5
0.0
−2.5
−5.0
component = Independent Component 3
Amplitude
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
Time

References