8 Multivariate Methods

R. Heller · Date: 2026-05-11 · Last changed: 2026-11-10

Methods for high-dimensional data without an outcome variable: principal components, factor analysis, canonical correlation, linear discriminant analysis, hierarchical and partition clustering, UMAP, and t-SNE. Both the variance-explained and the visual-projection use cases are covered.

This chapter contains **30 method pages** and **3 labs**. If you are not sure which method to read, return to Chapter 0 and follow the decision tree to the right node.

8.1 Method pages

Method	Source slug
Bray-Curtis Dissimilarity	`bray-curtis`
Canonical Correlation Analysis	`canonical-correlation`
Cluster Validation: Silhouette	`cluster-validation-silhouette`
Confirmatory Factor Analysis	`factor-analysis-confirmatory`
Correspondence Analysis	`correspondence-analysis`
DBSCAN Clustering	`dbscan-clustering`
Distance and Dissimilarity Measures	`distance-measures`
Exploratory Factor Analysis	`factor-analysis-exploratory`
Factor Rotation	`factor-rotation`
Gaussian Mixture Models	`gaussian-mixture-models`
Hierarchical Clustering	`hierarchical-clustering`
Independent Component Analysis	`ica-independent-component-analysis`
Jaccard and Dice Dissimilarity	`jaccard-dissimilarity`
k-Means Clustering	`kmeans-clustering`
Kernel PCA	`kernel-pca`
Linear Discriminant Analysis	`linear-discriminant-analysis`
Mahalanobis Distance	`mahalanobis-distance`
MANOVA	`manova`
Metric Multidimensional Scaling	`mds-metric`
Multiple Correspondence Analysis	`multiple-correspondence-analysis`
Non-Metric Multidimensional Scaling	`mds-nonmetric`
Partial Least Squares Regression	`pls-regression`
Partitioning Around Medoids	`pam-kmedoids`
PLS Discriminant Analysis	`pls-da`

Method	Source slug
Principal Component Analysis	`principal-component-analysis`
Quadratic Discriminant Analysis	`quadratic-discriminant-analysis`
Structural Equation Models	`structural-equation-models`
The Elbow Method	`elbow-method`
The Gap Statistic	`gap-statistic`
UMAP and t-SNE: Overview	`umap-tsne-overview`

8.2 Labs

Lab
PCA, factor analysis, CCA, LDA
Clustering: k-means, hierarchical, model-based
UMAP and t-SNE

8.3 Bray-Curtis Dissimilarity

8.3.1 Introduction

Bray-Curtis dissimilarity is the default index for comparing ecological communities described by species-abundance data. It ranges from 0 when two samples share the same composition and abundances to 1 when they share no species at all, and unlike binary indices it makes use of the abundance information rather than collapsing it to presence/absence. Microbial ecology, plant community work, and any compositional dataset where rare and abundant species both contribute meaningful information typically reach for Bray-Curtis as the first dissimilarity to compute.

8.3.2 Prerequisites

A working knowledge of distance and dissimilarity matrices, abundance data structures, and ordination methods (NMDS, PCoA) that consume dissimilarity matrices.

8.3.3 Theory

For two samples i and j with abundances x_{ik} across species k:

$$\mathrm{BC}_{ij} = \frac{\sum_k |x_{ik} - x_{jk}|}{\sum_k (x_{ik} + x_{jk})}.$$

It is a dissimilarity, not a strict metric — the triangle inequality can fail, so Bray-Curtis matrices are best visualised with non-metric MDS (NMDS) rather than principal coordinates. The binary form (presence-absence input) reduces to the Sørensen dissimilarity. Bray-Curtis is sensitive to dominant species; a log or Hellinger transformation of the abundance matrix before computing the index down-weights dominance and lets rare species contribute more.

8.3.4 Assumptions

Count or abundance data, all entries non-negative, comparable sampling effort across samples (or normalisation to relative abundance to remove the effort difference).

8.3.5 R Implementation

```r
library(vegan)

data(dune)
d_bc ← vegdist(dune, method = "bray")
as.matrix(d_bc)[1:3, 1:3]

# Log-transform to down-weight dominant species
d_bc_log ← vegdist(log1p(dune), method = "bray")

# NMDS visualisation
nmds ← metaMDS(dune, distance = "bray", k = 2, trymax = 50)
plot(nmds, type = "t")
```

8.3.6 Output & Results

`vegdist()` returns a `dist` object containing the lower-triangular Bray-Curtis matrix. `metaMDS()` runs NMDS on the resulting matrix and returns coordinates, stress, and convergence diagnostics. Stress values below 0.1 indicate excellent representation; 0.1–0.2 acceptable; above 0.2 poor.

8.3.7 Interpretation

A reporting sentence: “Bray-Curtis dissimilarity averaged 0.55 across all pairs (range 0.32 to 0.84), indicating substantial community turnover; NMDS in two dimensions converged at stress 0.07, and PERMANOVA (`vegan::adonis2`) detected a significant management-type effect ($F = 3.6$, $p = 0.001$, $R^2 = 0.21$).” Always pair Bray-Curtis ordination with a formal group test such as PERMANOVA or ANOSIM.

8.3.8 Practical Tips

- Log-transform abundance (`log1p()`) or use Hellinger distance (`vegdist(..., method = "hellinger")`) to reduce dominance by abundant species.
- Bray-Curtis is the default distance in `vegan::metaMDS()` ; passing the data matrix directly is equivalent to `vegdist(data, "bray")` followed by `metaMDS()` on the dist.
- For phylogenetic-aware distances on microbial communities, use UniFrac (`GUniFrac::GUniFrac`) instead; it accounts for evolutionary relationships among

taxa.

- Pair NMDS with PERMANOVA (`adonis2`) for group comparison and PERMDISP (`betadisper`) to check homogeneity of multivariate dispersion; significant PERMDISP can confound PERMANOVA conclusions.
- `vegdist()` supports many alternatives — Jaccard, Manhattan, Euclidean, Canberra, Gower, Hellinger — change `method` to switch.
- For very large community matrices, sparse implementations in `parallelDist` scale Bray-Curtis computation across cores without code changes.

8.3.9 R Packages Used

`vegan` for `vegdist()`, `metaMDS()`, `adonis2()`, and `betadisper()`; `GUniFrac` for phylogenetic UniFrac variants on microbiome data; `parallelDist` for fast multi-core distance computation; `phyloseq` for microbiome workflows that integrate Bray-Curtis with sequence and tree data.

8.3.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.3.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.4 Canonical Correlation Analysis

8.4.1 Introduction

Canonical correlation analysis (CCA) finds pairs of linear combinations — one from each of two variable sets — that are maximally correlated. Where multiple regression has one outcome and many predictors, CCA generalises to many outcomes and many predictors simultaneously: the first canonical pair (U_1, V_1) is the linear combination of the predictor side and the linear combination of the outcome side whose correlation is highest. Successive pairs are constrained to be uncorrelated with previous ones, mirroring the orthogonality of principal components.

8.4.2 Prerequisites

A working understanding of multiple regression, principal components analysis, and the geometry of correlation in multivariate spaces.

8.4.3 Theory

Given two sets of variables X (p columns) and Y (q columns), CCA finds pairs of canonical variates $U_k = Xa_k$ and $V_k = Yb_k$ maximising $\text{cor}(U_k, V_k)$ subject to $\text{Var}(U_k) = \text{Var}(V_k) = 1$ and orthogonality with previous pairs. There are $\min(p, q)$ canonical pairs, indexed by canonical correlations $\rho_1 \geq \rho_2 \geq \ldots$ that are non-increasing.

Inference under multivariate Normality uses Wilks' lambda or Pillai's trace to test whether canonical correlations beyond a given index are zero. Canonical loadings (correlations between original variables and canonical variates) help interpret the structure of each canonical pair.

8.4.4 Assumptions

Multivariate Normality (for the asymptotic significance tests), linear associations between variable sets, and adequate sample size relative to $p + q$. CCA is sensitive to outliers; one or two extreme observations can dominate the canonical structure.

8.4.5 R Implementation

```r
library(CCA)

# Two sets: first two and last two iris variables (artificial split)
X ← scale(iris[, 1:2])
Y ← scale(iris[, 3:4])

cc_res ← cc(X, Y)
cc_res$cor      # canonical correlations
cc_res$xcoef    # canonical coefficients for X
cc_res$ycoef    # canonical coefficients for Y

# Significance via Wilks' lambda
wilks ← (1 - cc_res$cor^2)
wilks
```

8.4.6 Output & Results

`cc()` returns canonical correlations (sorted in decreasing order), canonical coefficients (weights for each variable set), and the canonical variate scores. Wilks' lambda statistic and its associated F-approximation test the joint significance of canonical correlations from a given index onwards.

8.4.7 Interpretation

A reporting sentence: "Canonical correlation analysis on the iris measurements found a first canonical correlation of $\rho_1 = 0.94$ ($p < 0.001$, Wilks' $\lambda = 0.077$); the canonical loadings showed petal dimensions dominating the Y-side combination, indicating that overall flower size is the primary axis of association between the sepal and petal measurements." Report canonical loadings (correlations) rather than raw weights for interpretation; the latter are scale-dependent.

8.4.8 Practical Tips

- Interpret only the first few canonical correlations; later pairs typically capture little signal.
- For high-dimensional settings ($p + q$ approaching or exceeding n), use regularised CCA (`CCA::rcc`); standard CCA breaks down with rank-deficient covariance matrices.
- Partial CCA (`CCA::pcc` or `CCP::p.asym`) controls for covariates that should be removed before testing the canonical structure.
- For predictive use, partial least squares (`pls::plsr` or `mixOmics::spls`) is often preferable; PLS optimises predictive covariance rather than mutual correlation.
- Canonical loadings (correlations between variables and canonical variates) interpret cleaner than raw weights because they are scale-independent.
- Always pre-screen outliers; CCA is highly sensitive to extreme observations, and a robust CCA variant (`rrcov::CcaClassic` vs `CcaRobust`) is worth comparing.

8.4.9 R Packages Used

`CCA` for `cc()`, `rcc()`, and visualisation helpers; `CCP` for asymptotic significance tests (Wilks' lambda, Pillai, Hotelling-Lawley); `vegan::CCorA()` for ecology-flavoured CCA; `mixOmics::cca()` for sparse and regularised variants integrated with omics workflows.

8.4.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.4.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.5 Cluster Validation: Silhouette

8.5.1 Introduction

The silhouette is the most widely used internal validation metric for clustering. For each observation, it compares the mean distance to others in its own cluster against the mean distance to the nearest competing cluster, returning a value between -1 and $+1$. Values near 1 indicate observations comfortably inside their assigned cluster; values near 0 indicate ambiguous assignments on a cluster boundary; negative values indicate

observations that would fit better in a different cluster. Averaging silhouettes across observations gives a single number summarising cluster quality without needing labels or external references.

8.5.2 Prerequisites

A working understanding of distance metrics, clustering algorithms (k-means, hierarchical, PAM), and the distinction between internal and external validation criteria.

8.5.3 Theory

For observation i in cluster C_k, define:

- a_i = mean distance from i to all other observations in C_k.
- b_i = minimum, over clusters $C_l \neq C_k$, of the mean distance from i to observations in C_l.

The silhouette width is

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \qquad s_i \in [-1, 1].$$

The cluster silhouette is the mean s_i across observations in that cluster; the overall silhouette is the mean across all observations. Conventional thresholds: > 0.7 indicates a strong cluster structure, 0.5–0.7 reasonable, 0.25–0.5 weak, < 0.25 no substantial structure.

8.5.4 Assumptions

A meaningful distance metric; unambiguous cluster assignments. Silhouette assumes convex clusters and can be misleading for density-defined or non-convex shapes (rings, half-moons), where DBSCAN-style validation is more appropriate.

8.5.5 R Implementation

```r
library(cluster); library(factoextra)

X <- scale(iris[, 1:4])

# k-means with silhouette
km <- kmeans(X, centers = 3, nstart = 25)
sil <- silhouette(km$cluster, dist(X))
summary(sil)

fviz_silhouette(sil)

# Choose k by silhouette
fviz_nbclust(X, kmeans, method = "silhouette")
```

8.5.6 Output & Results

`silhouette()` returns per-observation s_i, the assigned cluster, and the nearest-competing cluster. `summary()` aggregates by cluster and overall. `fviz_silhouette()` produces the canonical horizontal-bar silhouette plot. `fviz_nbclust()` automates k selection by maximising overall silhouette.

8.5.7 Interpretation

A reporting sentence: "k-means with $k = 3$ on the standardised iris data produced an overall silhouette width of 0.51 (cluster widths 0.79, 0.39, 0.45), no observations had negative silhouette, and 12 observations near the versicolor / virginica boundary had values below 0.20." Always report cluster-specific silhouettes, not just the overall mean — a poorly-fitting cluster can be hidden inside a high overall average.

8.5.8 Practical Tips

- Use silhouette to choose k alongside the elbow method and gap statistic; concordance across all three is the strongest argument for a chosen partition.
- Negative silhouettes flag misassigned observations; investigate whether they represent genuine intermediate cases, data errors, or an inadequate k.
- Silhouette is misleading for non-convex clusters (rings, manifolds); for density-based methods (DBSCAN, OPTICS), prefer density-aware validation indices.
- Plot silhouette per cluster and compare cluster-level means; a single low-silhouette cluster pulls the overall mean down without a per-cluster breakdown making it visible.
- Alternative internal indices (Davies-Bouldin, Calinski-Harabasz, Dunn) capture different aspects of cluster quality; `NbClust` computes 30+ and reports the consensus optimum.
- Silhouette computation is $O(n^2)$ in distance evaluations; for very large n, use sub-sample-based silhouette via `cluster::silhouette()` on a random subset.

8.5.9 R Packages Used

`cluster` for `silhouette()` and the underlying clustering algorithms; `factoextra` for `fviz_silhouette()` and `fviz_nbclust()` ggplot-style visualisation; `NbClust` for 30+ alternative internal indices in one call; `clusterCrit` for additional intrinsic and extrinsic validation indices.

8.5.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.5.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.6 Correspondence Analysis

8.6.1 Introduction

Correspondence analysis (CA) is the categorical analogue of principal components analysis. It takes a two-way contingency table and produces a low-dimensional map in which rows and columns become points whose proximity reflects association beyond what would be expected under independence. CA decomposes the chi-squared statistic into row and column contributions across a small number of dimensions, much as PCA decomposes covariance variance, and the resulting biplot is one of the most intuitive ways to visualise the association structure of a contingency table.

8.6.2 Prerequisites

A working understanding of contingency tables, the chi-squared statistic for independence, principal components analysis, and singular value decomposition.

8.6.3 Theory

For an $r \times c$ contingency table with cell counts n_{ij} and margins $n_{i\cdot}$, $n_{\cdot j}$, total n:

1. Compute expected counts under independence $e_{ij} = n_{i\cdot} n_{\cdot j} / n$.
2. Form the matrix of standardised residuals $z_{ij} = (n_{ij} - e_{ij}) / \sqrt{e_{ij}}$.
3. Apply singular value decomposition $Z = U D V^{\top}$.
4. Row coordinates and column coordinates come from U and V scaled by D and the row/column masses; the squared singular values sum to the total inertia χ^2 / n.

The biplot places row points close to column points with positive association; points close to the origin are near the average profile and contribute little to the structure. Multiple correspondence analysis (MCA) generalises CA to more than two categorical variables.

8.6.4 Assumptions

Count data in a rectangular contingency table with no zero marginal sums (all rows and columns have at least one observation), and a meaningful chi-squared interpretation of association.

8.6.5 R Implementation

```r
library(FactoMineR); library(factoextra)

data(smoke)
ca_res ← CA(smoke, graph = FALSE)
fviz_ca_biplot(ca_res)
summary(ca_res)
```

8.6.6 Output & Results

`CA()` returns row and column coordinates, contributions to each axis, total inertia, and inertia per axis (the categorical analogue of variance explained). `fviz_ca_biplot()` displays both row and column points on the first two principal axes, with axis labels showing the proportion of inertia captured.

8.6.7 Interpretation

A reporting sentence: "Correspondence analysis of the smoke dataset (5 rank levels × 4 smoking categories) decomposed total inertia of 0.085 into three principal axes; the first axis (88 % of inertia) placed senior employees close to the 'heavy smoker' category and junior managers near 'none', suggesting an age-rank-smoking association beyond chance ($\chi^2 = 16.4$, $p = 0.17$ — the visual association is strong but the statistical test is underpowered for this small table)." Always quote the proportion of inertia displayed and complement with the chi-squared test.

8.6.8 Practical Tips

- Report the proportion of inertia retained in the displayed dimensions; if the first two axes capture less than 70 %, much of the structure is not visible in the biplot.
- For more than two categorical variables, switch to multiple correspondence analysis (MCA, `FactoMineR::MCA`); supplementary continuous covariates can be projected on top.
- Remove or merge zero-count rows or columns before fitting; they break the chi-squared transformation and produce undefined coordinates.
- The symmetric biplot scales rows and columns comparably; the row-principal or column-principal alternatives are useful when one dimension is the focus but distort the joint geometry.
- Pair CA with a chi-squared test on the original table; the inertia decomposition shows where the association concentrates, while the test quantifies whether it exceeds chance.
- For supplementary categories that should not influence the axes (e.g. demographic markers on the original survey items), pass them through the `quali.sup` argument; they are projected after fitting but do not affect axis estimation.

8.6.9 R Packages Used

`FactoMineR` for `CA()`, `MCA()`, and the broader factor-analysis family; `factoextra` for `fviz_ca_biplot()`, `fviz_ca_row()`, `fviz_ca_col()` ggplot-style visualisation; `ca` for an alternative implementation with classic biplot styling; `ade4::dudi.coa()` for ecology-flavoured CA.

8.6.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.6.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.7 DBSCAN Clustering

8.7.1 Introduction

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is the most influential density-based clustering algorithm. Unlike k-means, it does not require specifying the number of clusters in advance, finds clusters of arbitrary shape (rings, half-moons, elongated bands), and explicitly flags low-density points as noise rather than forcing them into a cluster. These properties make it the right choice for spatial data, anomaly detection alongside clustering, and any context where the intuitive notion of "cluster" is "a dense region surrounded by low-density gaps".

8.7.2 Prerequisites

A working understanding of distance metrics, density-based reasoning, and the limitations of partitioning algorithms (k-means, PAM) for non-convex cluster shapes.

8.7.3 Theory

DBSCAN has two parameters: ϵ (eps), the neighbourhood radius, and minPts, the minimum number of points (including the centre) required for a point to be a *core* point.

- A **core point** has at least minPts points within distance ϵ.
- A **border point** is within ϵ of a core point but is not itself a core point.
- A **noise point** is neither.

Clusters are formed as maximal connected sets of core points (with their associated border points), with noise points left unassigned. The algorithm is order-invariant for core-point structure but border-point assignments can depend on processing order. Choosing ϵ is the practical challenge; the standard heuristic is the k-distance plot, which shows the distance to the k-th nearest neighbour for each point sorted in ascending order — a knee in the curve marks a sensible ϵ.

8.7.4 Assumptions

A meaningful distance metric; clusters have sufficient density separation to be distinguished from noise; standardised input so distances are comparable across variables.

8.7.5 R Implementation

```r
library(dbscan)

X ← scale(iris[, 1:4])

# Choose eps from k-distance plot
kNNdistplot(X, k = 5)
abline(h = 0.7, col = "red", lty = 2)

# DBSCAN
db ← dbscan(X, eps = 0.7, minPts = 5)
db$cluster

table(db$cluster, iris$Species)
```

8.7.6 Output & Results

`dbscan()` returns a vector of cluster labels with `0` indicating noise. Core points, border points, and noise are tagged separately in the result object. The k-distance plot guides ϵ selection; the table comparison against species labels shows how DBSCAN's discovered structure aligns with known classes.

8.7.7 Interpretation

A reporting sentence: "DBSCAN with $\epsilon = 0.7$ and $\text{minPts} = 5$ identified three clusters and flagged 10 observations as noise; the clusters reproduced the iris species structure with 88 % agreement, with the noise points concentrated at the versicolor / virginica boundary." Always state ϵ, minPts, and the number of noise points; clusters identified are sensitive to all three.

8.7.8 Practical Tips

- Use the k-distance plot (`kNNdistplot()`) to select ϵ; a clear knee in the sorted distance curve marks a sensible threshold.
- DBSCAN is sensitive to ϵ, especially with clusters of widely different densities; use OPTICS (`dbscan::optics()`) or HDBSCAN (`dbscan::hdbscan()`) when density

varies across clusters.
- Standardise variables before clustering; distance-based methods misbehave when scales differ across columns.
- Treat noise points explicitly in downstream analysis; do not force them into the nearest cluster, as that defeats DBSCAN's main advantage over k-means.
- For high-dimensional data, distances concentrate (the "curse of dimensionality") and DBSCAN loses discriminating power; reduce dimensionality first via PCA or UMAP.
- HDBSCAN is generally more robust than DBSCAN — it avoids the ϵ choice and handles varying densities — and is increasingly preferred for new applications.

8.7.9 R Packages Used

`dbscan` for `dbscan()`, `optics()`, `hdbscan()`, and the k-distance heuristic;
`fpc::dbscan()` for an alternative implementation with similar API;
`factoextra::fviz_cluster()` for ggplot-style cluster visualisation;
`cluster::clusterboot()` for stability-based validation of the resulting clusters.

8.7.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.7.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.8 Distance and Dissimilarity Measures

8.8.1 Introduction

The choice of distance or dissimilarity is the foundation of most multivariate methods: clustering algorithms (k-means, hierarchical, PAM, DBSCAN) consume a distance, multidimensional scaling and PCoA produce coordinates that approximate one, and k-nearest neighbours computes them at every prediction. Different metrics capture different notions of proximity, and the right choice depends on the data type — continuous vs. categorical, abundances vs. presence/absence, magnitudes vs. directions — and on the inferential question. Mismatched metrics produce results that look correct but answer the wrong question.

8.8.2 Prerequisites

A working understanding of vectors, the difference between distance and dissimilarity (metric vs. semi-metric), and the most common multivariate methods that consume distance matrices.

8.8.3 Theory

Standard continuous distances between p-dimensional vectors:

- **Euclidean**: $\sqrt{\sum_i (x_i - y_i)^2}$ — straight-line distance, the default for most uses.
- **Manhattan (L_1)**: $\sum_i |x_i - y_i|$ — robust to outliers, appropriate for grid-like geometry.
- **Minkowski (L_p)**: $(\sum_i |x_i - y_i|^p)^{1/p}$ — generalises Euclidean ($p = 2$) and Manhattan ($p = 1$).
- **Canberra**: $\sum_i |x_i - y_i| / (|x_i| + |y_i|)$ — sensitive to small values; useful for compositional data.
- **Cosine**: $1 - (x \cdot y) / (\|x\| \|y\|)$ — measures direction, not magnitude; standard in text and gene-expression contexts.
- **Mahalanobis**: scale- and correlation-aware, weights the deviation by the inverse covariance matrix.

For categorical or mixed data: Gower distance handles mixed types via per-variable scaling; Jaccard and Bray-Curtis suit presence-absence and abundance data respectively.

8.8.4 Assumptions

Metrics satisfy the triangle inequality and symmetry; dissimilarities may relax these. Some downstream methods (classical MDS, k-means) implicitly assume metric distances; others (NMDS, hierarchical clustering with non-Euclidean distances) handle dissimilarities gracefully.

8.8.5 R Implementation

```r
library(cluster)

x ← rbind(c(1, 2, 3), c(2, 4, 5), c(5, 0, 1))

dist(x, method = "euclidean")
dist(x, method = "manhattan")
dist(x, method = "minkowski", p = 3)
dist(x, method = "canberra")

# Gower for mixed data
d ← data.frame(num = 1:3, cat = c("A", "B", "A"))
daisy(d, metric = "gower")
```

8.8.6 Output & Results

`dist()` returns a `dist` object containing the lower-triangular pairwise distance matrix. Different methods produce numerically different matrices that emphasise different features of the geometry. `cluster::daisy()` extends this to mixed numeric-categorical data via Gower distance.

8.8.7 Interpretation

A reporting sentence: "Euclidean distances of 3.0 between rows 1 and 2 and 5.7 between rows 1 and 3 reflect continuous-valued separation; the Gower distance on mixed data scaled each variable to $[0, 1]$ before combining and produced a unitless dissimilarity in the same range." Always state the metric used; clustering or ordination output is uninterpretable without it.

8.8.8 Practical Tips

- Standardise variables (`scale()`) before Euclidean distance whenever scales differ across columns; otherwise large-variance variables dominate the distance and bias every downstream method.
- For mixed numeric and categorical data, Gower is the standard; it scales each variable to $[0, 1]$ via type-specific rules and combines them as a weighted average.
- For binary presence/absence, prefer Jaccard or Sørensen-Dice (which ignore joint absences) over Hamming (which counts all matches).
- For sparse high-dimensional data (text, gene expression), cosine distance is often more meaningful than Euclidean because magnitudes vary widely while direction encodes the signal.
- Mahalanobis is essential when predictor correlations matter; it underlies the discriminant function in LDA and Hotelling's T^2.
- Always match the metric to the data type and downstream method; Euclidean on mixed scales or on presence/absence data is a frequent mistake that invalidates clustering results.

8.8.9 R Packages Used

Base R `dist()` for the standard continuous metrics; `cluster::daisy()` for Gower distance; `vegan::vegdist()` for ecological distances (Bray-Curtis, Jaccard, Hellinger, Canberra); `philentropy::distance()` for an extensive catalogue of distance and divergence functions; `proxy::dist()` and `parallelDist::parDist()` for multi-core distance computation on large matrices.

8.8.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.8.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.9 The Elbow Method

8.9.1 Introduction

The elbow method is the simplest heuristic for choosing the number of clusters in k-means or any partitioning algorithm. The within-cluster sum of squares (WSS) decreases monotonically with k, but the rate of decrease slows once the natural cluster structure has been captured. Plotting WSS against k reveals a "bend" — the elbow — at the point where additional clusters bring diminishing returns. The method is unavoidably subjective but widely used because it is fast, intuitive, and works well when the data have well-separated clusters.

8.9.2 Prerequisites

A working understanding of k-means clustering, the within-cluster sum of squares as a quality criterion, and the relationship between k and within-cluster compactness.

8.9.3 Theory

For partition $C = \{C_1, \ldots, C_k\}$ of the data, $\mathrm{WSS}(k) = \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \bar{x}_j\|^2$ is the total within-cluster sum of squared distances to centroids. As k grows, WSS necessarily decreases — at $k = n$ it reaches zero. A genuine cluster structure produces a sharp bend at the true number of clusters, after which adding more k subdivides homogeneous regions and yields only modest reductions. The bend is identified visually or, more reproducibly, by algorithms such as Kneedle that compute a normalised distance from each point on the curve to a chord between the endpoints.

8.9.4 Assumptions

A genuine cluster structure exists in the data; if no clusters are present, no clear elbow appears. Variables are standardised so distances along each axis are comparable.

8.9.5 R Implementation

```r
library(factoextra)

X ← scale(iris[, 1:4])

# Elbow plot
fviz_nbclust(X, kmeans, method = "wss", k.max = 10)

# Manual: compute WSS across k
wss ← sapply(1:10, function(k)
  kmeans(X, centers = k, nstart = 25)$tot.withinss)
plot(1:10, wss, type = "b",
     xlab = "Number of clusters k",
     ylab = "Within-cluster sum of squares")
```

8.9.6 Output & Results

A monotone decreasing WSS curve. For the standardised iris data, the bend appears at $k = 3$, consistent with the three species. `fviz_nbclust()` automates the standard elbow plot with `ggplot2` styling and works with k-means, PAM, and CLARA via the `FUNcluster` argument.

8.9.7 Interpretation

A reporting sentence: "The elbow plot showed a clear bend at $k = 3$ (within-cluster sum of squares dropped from 681 at $k = 1$ to 140 at $k = 3$ and only marginally further to 96 at $k = 10$); we chose $k = 3$, consistent with the silhouette method's optimum and with the known species structure." Always couple the elbow visual with at least one quantitative criterion (silhouette, gap statistic).

8.9.8 Practical Tips

- Pair the elbow with the silhouette method (`fviz_nbclust(..., method = "silhouette")`) and the gap statistic (`method = "gap_stat"`); concordance across all three increases confidence in the chosen k.
- If no clear elbow appears, the data may not have a natural cluster structure; do not force a k where the WSS curve is smooth.
- Standardise variables before clustering; variables with larger variance dominate WSS without standardisation and distort the elbow.
- Use `nstart = 25` or higher when computing WSS; multiple random starts avoid local minima that produce noisy WSS curves.
- The Kneedle algorithm (R package `kneedle`) automates elbow detection by finding the point of maximum curvature; useful for batch processing many datasets.
- Elbow alone is insufficient justification for a chosen k in publications; report it alongside silhouette and gap-statistic results.

8.9.9 R Packages Used

`factoextra` for `fviz_nbclust()` ggplot-style elbow, silhouette, and gap-statistic plots; `cluster` for the underlying clustering algorithms; `NbClust` for 30+ cluster-validation indices in one call; `kneedle` for automated elbow detection.

8.9.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.9.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.10 Confirmatory Factor Analysis

8.10.1 Introduction

Confirmatory factor analysis (CFA) tests a pre-specified factor structure against data: which items load on which factors, which loadings are constrained to zero, and how factors correlate with each other. It is the measurement-model portion of structural equation modelling and the standard tool for validating questionnaire structures, latent-trait models in psychometrics, and multi-indicator latent constructs in clinical and behavioural research. Unlike exploratory factor analysis, CFA does not estimate which items belong to which factor — that decision is in the model — and instead estimates only the strength of the predetermined relationships and tests model-data fit.

8.10.2 Prerequisites

A working understanding of exploratory factor analysis, regression, and the `lavaan` model-syntax conventions for specifying latent variables.

8.10.3 Theory

CFA specifies a complete pattern matrix in advance: each item either loads freely on its assigned factor or is constrained to zero on every other factor. Fit is assessed via:

- **Chi-squared test**: a non-significant χ^2 formally accepts the model, but the test rejects too easily at large n.
- **Incremental fit indices**: CFI and TLI compare to a baseline null model; values above 0.95 indicate good fit, 0.90–0.95 acceptable.

- **Absolute fit**: RMSEA measures discrepancy per degree of freedom; below 0.06 is good, 0.06–0.08 acceptable. The accompanying 90 % CI quantifies precision.
- **SRMR**: standardised root-mean-square residual; below 0.08 indicates good fit.

A well-fitting model satisfies multiple criteria simultaneously — no single index is sufficient.

8.10.4 Assumptions

A correctly specified factor structure, multivariate Normal continuous indicators (or robust / WLSMV estimators for non-Normal or ordinal data), and sufficient sample size (rule of thumb: at least 10 observations per estimated parameter, or $n \geq 200$).

8.10.5 R Implementation

```r
library(lavaan); library(semTools)

# Three-factor model on HolzingerSwineford1939 dataset
model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- cfa(model, data = HolzingerSwineford1939)
summary(fit, fit.measures = TRUE, standardized = TRUE)

# Modification indices for potential improvements
head(modificationIndices(fit, sort = TRUE), 10)
```

8.10.6 Output & Results

`summary(fit, fit.measures = TRUE, standardized = TRUE)` reports unstandardised and standardised loadings, factor correlations, residual variances, and the full panel of fit indices (chi-squared, CFI, TLI, RMSEA with CI, SRMR). `modificationIndices()` lists potential parameter additions ordered by expected chi-squared improvement.

8.10.7 Interpretation

A reporting sentence: "The three-factor CFA fit the data adequately ($\chi^2(24) = 85.3$, $p < 0.001$; CFI 0.93; RMSEA 0.092 [90 % CI 0.071 to 0.114]; SRMR 0.065); all standardised loadings exceeded 0.50 and were statistically significant ($p < 0.001$). RMSEA exceeded the conventional 0.06 threshold, suggesting modest local misspecification — modification indices indicated a residual correlation between x_7 and x_9 that would improve fit if substantively justified." Always report multiple fit indices; no single one is decisive.

8.10.8 Practical Tips

- Do not follow modification indices blindly; each unconstrained parameter moves the analysis from confirmatory to exploratory and inflates Type I error if not pre-specified.

- For ordinal Likert items, use diagonally-weighted least squares (`cfa(..., estimator = "DWLS")` or `"WLSMV"`); ordinary ML on Likert items violates Normality assumptions.
- For non-Normal continuous data, use robust ML (`estimator = "MLR"`); standard errors and chi-squared are corrected for non-Normality.
- Measurement-invariance testing across groups (configural, metric, scalar invariance) is a standard follow-up before comparing latent means; use `semTools::measurementInvariance()`.
- Report chi-squared with degrees of freedom and the full set of fit indices; never report a single index alone.
- Standardised loadings (factor-loading correlations) communicate effect size more clearly than unstandardised; always include them.

8.10.9 R Packages Used

`lavaan` for `cfa()` and `sem()` with multiple estimators; `semTools` for measurement-invariance testing, reliability coefficients, and modification heuristics; `lavaanPlot` and `semPlot` for path-diagram visualisation; `blavaan` for Bayesian CFA when small samples or complex priors require it.

8.10.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.10.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.11 Exploratory Factor Analysis

8.11.1 Introduction

Exploratory factor analysis (EFA) searches for a small number of latent factors that explain the correlations among a larger set of observed variables. It is the cornerstone of psychometric scale development, the standard tool for validating questionnaire structures, and a useful exploratory technique in any context where many correlated indicators are believed to reflect a smaller number of latent constructs. The output — rotated factor loadings together with factor correlations — directly informs item-selection, scale-construction, and CFA model-specification decisions.

8.11.2 Prerequisites

A working understanding of correlation matrices, principal components analysis, and the conceptual difference between observed indicators and latent constructs.

8.11.3 Theory

The common factor model is

$$X = \Lambda F + U,$$

where X is the observed item matrix, Λ is the loading matrix, F are the common factors, and U is unique (specific) variance. Unlike PCA, which lumps common and specific variance together, EFA explicitly partitions the variance.

The standard EFA workflow has five steps:

1. **Check factorability**: Kaiser-Meyer-Olkin (KMO > 0.6 acceptable, > 0.8 good) and Bartlett's sphericity test confirm that the correlation matrix has enough structure to factor.
2. **Choose number of factors**: parallel analysis is the modern gold standard; the obsolete Kaiser eigenvalue-greater-than-1 rule and the subjective scree plot are still seen but should be supplemented.
3. **Extract**: maximum likelihood (ML) for inferential purposes, principal-axis factoring (PAF) for robustness, principal-components for quick exploration.
4. **Rotate**: varimax (orthogonal) or oblimin / promax (oblique) for interpretable loadings.
5. **Interpret**: examine rotated loadings, factor correlations, and item communalities.

8.11.4 Assumptions

Multivariate Normal continuous data for ML extraction; linearity of item-factor relationships; sufficient sample size (rule of thumb: at least 5–10 cases per item, with $n \geq 200$ as a floor).

8.11.5 R Implementation

```r
library(psych); library(GPArotation)

d ← bfi[, 1:25]   # 25-item big-five inventory

# Factorability
KMO(d)
cortest.bartlett(cor(d, use = "pairwise"), n = nrow(d))

# Number of factors
fa.parallel(d, fa = "fa")

# EFA with 5 factors, oblique rotation
efa ← fa(d, nfactors = 5, rotate = "oblimin", fm = "ml")
print(efa, cut = 0.3)
```

8.11.6 Output & Results

`KMO()` returns overall and per-item factorability indices; `cortest.bartlett()` returns the sphericity test. `fa.parallel()` overlays observed eigenvalues on simulated null eigenvalues to determine the number of factors. `fa()` returns rotated loadings, communalities, factor correlations (under oblique rotation), and explained variance per factor.

8.11.7 Interpretation

A reporting sentence: "EFA on 25 personality items (n = 2,800; KMO = 0.85, Bartlett's $\chi^2 = 18{,}245$, $p < 0.001$) supported a five-factor solution by parallel analysis; oblimin-rotated loadings aligned with the Big Five structure (extraversion, neuroticism, conscientiousness, agreeableness, openness), explaining 45 % of common variance, and factor correlations ranged from 0.15 to 0.42." Always report KMO, Bartlett, the number-of-factors method, the extraction method, and the rotation.

8.11.8 Practical Tips

- Use parallel analysis (`fa.parallel`) for the number-of-factors decision; avoid the obsolete "eigenvalue > 1" Kaiser rule which over-extracts in most settings.
- Prefer oblique rotation (oblimin, promax) by default; uncorrelated factors are rare in real psychological or biological data.
- Watch for Heywood cases — loadings above 1 or communalities above 1 — which indicate over-extraction or model misspecification.
- For ordinal Likert items, use polychoric correlations (`fa(..., cor = "poly")`); Pearson correlations on Likert items underestimate the true factor structure.
- For confirmatory tests of the resulting structure, run a separate CFA (`lavaan::cfa`) on a held-out sample; never confirm on the data used to extract.
- Report item communalities alongside loadings; communalities below 0.20 indicate items that contribute little to any factor and may need removal.

8.11.9 R Packages Used

`psych` for `fa()`, `KMO()`, `fa.parallel()`, and the comprehensive EFA workflow; `GPArotation` for the underlying rotation algorithms; `EFAtools` for parallel-analysis-aware extraction with bootstrap confidence intervals; `lavaan` for confirmatory factor analysis as the natural follow-up; `MBESS` for additional reliability and structural-equation utilities.

8.11.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.11.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.12 Factor Rotation

8.12.1 Introduction

After factor extraction, rotation transforms the loading matrix to make interpretation easier without changing model fit. The mathematical reason is that any rotation of the factor solution produces the same predicted covariance matrix; rotation merely chooses a particular representation among the infinitely many that fit equally well. The choice of rotation method — orthogonal versus oblique, varimax versus promax versus oblimin — substantially affects which interpretation a reader will reach, so disclosing it is part of any responsible EFA report.

8.12.2 Prerequisites

A working understanding of exploratory factor analysis, factor loadings, and the distinction between orthogonal and oblique factor structures.

8.12.3 Theory

Orthogonal rotations keep factors uncorrelated; the rotated loading matrix retains the property that factors do not share variance.

- **Varimax**: maximises the variance of squared loadings within each column, producing "simple structure" where each variable loads strongly on one factor and weakly on others.
- **Quartimax**: maximises variance per row; tends to produce a single general factor.

Oblique rotations allow factors to correlate; this is more realistic for most psychological, biological, and social phenomena where underlying constructs are not artificially independent.

- **Oblimin**: direct minimisation of cross-factor loadings under controlled obliquity.
- **Promax**: applies a power transformation to a varimax solution to produce an oblique structure.
- **Geomin**: alternative oblique rotation popular in IRT contexts.

Oblique rotations produce both a *pattern* matrix (regression-like loadings of variables on factors) and a *structure* matrix (correlations between variables and factors); the two coincide only under orthogonal rotation.

8.12.4 Assumptions

Rotation leaves fit unchanged; only interpretation changes. The chosen rotation respects the substantive theory about whether factors should correlate.

8.12.5 R Implementation

```r
library(psych); library(GPArotation)

d ← bfi[, 1:25]

# Varimax (orthogonal)
efa_v ← fa(d, nfactors = 5, rotate = "varimax", fm = "ml")
print(efa_v, cut = 0.3)

# Oblimin (oblique)
efa_ob ← fa(d, nfactors = 5, rotate = "oblimin", fm = "ml")
print(efa_ob, cut = 0.3)

# Factor correlations after oblique rotation
efa_ob$Phi
```

8.12.6 Output & Results

`fa()` returns a rotated loading matrix; under oblique rotation, `$Phi` contains the factor correlation matrix. Setting `cut = 0.3` in `print()` suppresses small loadings for cleaner display. Comparing varimax and oblimin loadings on the same data shows how dramatically the interpretation can shift.

8.12.7 Interpretation

A reporting sentence: “Oblimin-rotated factors showed moderate correlations (range 0.15 to 0.42), justifying oblique rotation; the resulting pattern matrix gave clear simple structure with loadings above 0.40 aligning with the hypothesised Big Five domains and cross-loadings predominantly below 0.20.” Always report factor correlations under oblique rotation; they justify the choice and constrain interpretation.

8.12.8 Practical Tips

- Oblique rotation (oblimin or promax) is the default in modern psychometrics; orthogonal rotation should be justified by genuine reason to expect uncorrelated factors.
- Report factor correlations from the Phi matrix; if all are below 0.20, an orthogonal solution may be acceptable.
- Varimax produces simpler-looking loading structures but can be misleading when factors truly correlate; the apparent simplicity may hide systematic cross-loadings.
- Quartimax tends to produce a strong general factor; useful when a hierarchical structure is expected (Spearman’s g, hierarchical personality models).
- Rotation is a numerical optimisation; start from multiple random initialisations to confirm the solution is stable.

- `psych::fa.diagram()` visualises the rotated structure as a path diagram, often easier to communicate than a loadings table.

8.12.9 R Packages Used

`psych` for `fa()` and the broad EFA / CFA workflow with rotation options; `GPArotation` for the underlying rotation algorithms (oblimin, promax, geomin, simplimax); `lavaan` for confirmatory factor analysis as the natural follow-up; `EFAtools` for parallel-analysis-aware EFA workflows.

8.12.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.12.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.13 The Gap Statistic

8.13.1 Introduction

The gap statistic, introduced by Tibshirani, Walther, and Hastie in 2001, is a principled way to choose the number of clusters k in any partitioning method. Where the elbow method relies on visual judgement and the silhouette method on per-observation neighbour comparisons, the gap statistic compares the within-cluster dispersion of the data to what would be expected under a null reference distribution with no cluster structure. The point at which the data deviate most from the null is the optimal k, with bootstrap-based standard errors that quantify the certainty of the choice.

8.13.2 Prerequisites

A working understanding of clustering algorithms, within-cluster sum of squares, and bootstrap resampling for standard error estimation.

8.13.3 Theory

For each candidate k, define W_k as the log of the within-cluster dispersion (sum of squared distances to centroids) on the actual data after running k-means or another partitioning algorithm. Generate B datasets from a null reference — uniformly distributed

over the data's bounding box, or over its principal-components box — and compute the average within-cluster dispersion $\mathbb{E}[W_k^*]$ under the null. The gap statistic is

$$\mathrm{Gap}(k) = \mathbb{E}[W_k^*] - W_k.$$

The optimal k is the smallest value satisfying $\mathrm{Gap}(k) \geq \mathrm{Gap}(k+1) - s_{k+1}$, where s_{k+1} is the standard deviation of the null reference at $k+1$ (the so-called "first-SE-max" criterion). This rule rewards parsimonious choices: it picks the smallest k that is not significantly worse than larger values.

8.13.4 Assumptions

A meaningful distance metric on the data; the null reference distribution is sensible (uniform over the bounding box is reasonable for low-to-moderate dimensions; PC-aligned bounding boxes are better in high dimensions).

8.13.5 R Implementation

```r
library(cluster); library(factoextra)

X ← scale(iris[, 1:4])

set.seed(2026)
gap ← clusGap(X, FUN = kmeans, nstart = 25, K.max = 10, B = 100)
print(gap, method = "firstSEmax")

fviz_gap_stat(gap)
```

8.13.6 Output & Results

`clusGap()` returns a matrix with W_k, $\mathbb{E}[W_k^*]$, the gap value, and its bootstrap standard error for each k. `print(..., method = "firstSEmax")` applies Tibshirani's selection rule to identify the optimal k. `fviz_gap_stat()` plots the gap curve with error bars.

8.13.7 Interpretation

A reporting sentence: "The gap statistic, computed with $B = 100$ bootstrap replicates against a uniform reference, identified $k = 3$ as the optimal number of clusters using the firstSEmax criterion ($\mathrm{Gap}(3) = 0.59$, $\mathrm{Gap}(4) - s_4 = 0.55$); this matches the underlying species structure in the iris data." Report the criterion explicitly (firstSEmax, Tibs2001SEmax, globalmax); they sometimes disagree.

8.13.8 Practical Tips

- Use $B = 100$ to 500 bootstrap replicates; fewer can produce unstable optima, more is rarely worth the runtime cost.
- The default uniform-bounding-box reference works well in low dimensions; in high dimensions, switch to a PC-aligned reference (`spaceH0 = "scaledPCA"`).
- Combine the gap statistic with elbow and silhouette methods; concordance across all three is the strongest argument for a chosen k.

- Gap performs best on well-separated clusters; for overlapping or hierarchical structures it can favour very large k or fail to find a clear optimum.
- For very large datasets, subsample before applying the gap statistic; the bootstrap step otherwise dominates runtime.
- Selection criteria differ: `firstSEmax` (Tibshirani's parsimonious choice), `globalmax` (the absolute maximum, often larger), and `Tibs2001SEmax` (the original 2001 rule). Quote the criterion in any report.

8.13.9 R Packages Used

`cluster` for `clusGap()` and the underlying clustering algorithms; `factoextra` for `fviz_gap_stat()` and `fviz_nbclust()` ggplot-style visualisation; `NbClust` for 30+ alternative cluster-validation indices in one call.

8.13.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.13.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.14 Gaussian Mixture Models

8.14.1 Introduction

Gaussian mixture models (GMMs) represent the data as a weighted sum of several multivariate Normal distributions, each component characterising a latent cluster. Unlike k-means, which is essentially a hard-assignment heuristic, GMMs are fully probabilistic generative models: each observation has a posterior probability of belonging to each cluster, the clusters can have different shapes (covariance structures), and model-selection criteria (BIC) handle the choice of cluster count without ad-hoc heuristics. The `mclust` package enumerates a small library of covariance structures and selects the best automatically.

8.14.2 Prerequisites

A working understanding of the multivariate Normal distribution, the EM algorithm for latent-variable models, and the BIC as a model-selection criterion.

8.14.3 Theory

The mixture density is

$$p(x) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k),$$

with mixing proportions $\pi_k \geq 0$ summing to 1. Maximum-likelihood estimation alternates an E-step (compute posterior responsibilities $\gamma_{ik} = \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) / p(x_i)$) and an M-step (update π_k, μ_k, Σ_k from the responsibilities). `mclust` parameterises the covariance via 14 structures from spherical-equal-volume to fully-unconstrained-different-shape and orientation, indexed by three letters: volume, shape, orientation, each Equal or Varying. BIC selects the best combination of structure and component count.

8.14.4 Assumptions

Multivariate Normal mixture components, independent observations, and a sample size large enough to estimate component covariances stably (the most flexible models require $\Omega(Kp^2)$ observations for K clusters in p dimensions).

8.14.5 R Implementation

```r
library(mclust)

X ← iris[, 1:4]
fit ← Mclust(X)
summary(fit)

# Soft assignments
head(fit$z)

# Plot
plot(fit, what = "classification")

# BIC across models and K
plot(fit, what = "BIC")
```

8.14.6 Output & Results

`Mclust()` returns the best model code (e.g. “VEV”), the number of components, BIC values across the model space, the fitted parameters, and the matrix of soft assignment probabilities `fit$z`. `plot(fit, what = "BIC")` shows BIC against component count for each covariance structure, making the selection rationale visible.

8.14.7 Interpretation

A reporting sentence: “`mclust` selected a VEV (varying volume, equal shape, varying orientation) model with three components ($\mathrm{BIC} = -560.4$, $\Delta\mathrm{BIC}$ to the second-best model 18.2); the three components closely match the iris species (overall accuracy 97 %),

and the soft assignments quantify uncertainty for the seven observations classified differently from their species labels." Always report the chosen covariance structure and BIC value.

8.14.8 Practical Tips

- Use BIC for model selection across K and covariance structure jointly; comparing K at fixed structure or vice versa misses the joint optimum.
- Soft assignments (`fit$z`) are useful for downstream uncertainty propagation; rather than committing to one cluster label, propagate the posterior probabilities into subsequent analyses.
- For high-dimensional data ($p > n$), regularised mixture models (`HDclassif`, `pgmm`) or variational EM are needed; standard `mclust` is unstable when Σ_k has more parameters than observations.
- Outliers can distort the Normal components; consider t-mixture models (`teigen::teigen`) for heavier-tailed components, or trimmed EM (`tclust`) for robustness.
- Log-likelihood can become unbounded for unconstrained covariances when a component covers a single observation; `mclust` adds a small prior to prevent this, but watch for warnings.
- For very large n, mini-batch EM and online variants scale to millions of observations, but the `mclust` defaults handle up to a few thousand comfortably.

8.14.9 R Packages Used

`mclust` for `Mclust()`, BIC selection across the 14-model library, and visualisation; `flexmix` for finite mixtures of regressions and other generalised mixture models; `teigen` for t-mixture models robust to outliers; `tclust` for trimmed mixture models with explicit contamination handling.

8.14.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.14.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.15 Hierarchical Clustering

8.15.1 Introduction

Hierarchical clustering builds a nested family of partitions: at the bottom every observation is its own cluster, at the top all observations form a single cluster, and intermediate levels correspond to all the partitions you might consider. Agglomerative methods (the practical default) merge the two closest clusters at each step until everything is connected; divisive methods reverse the process. The output is a dendrogram — a tree that can be cut at any height to produce a partition with the corresponding number of clusters. The dendrogram is the headline visual that distinguishes hierarchical clustering from partitioning methods.

8.15.2 Prerequisites

A working understanding of distance metrics, the difference between distance between observations and distance between clusters, and the dendrogram as a graphical representation of the merging sequence.

8.15.3 Theory

Linkage criteria define the distance between two clusters A and B:

- **Single linkage**: $\min_{a \in A, b \in B} d(a, b)$. Prone to chaining: it joins clusters by their nearest pair, producing long stringy structures.
- **Complete linkage**: $\max_{a \in A, b \in B} d(a, b)$. Produces compact, roughly equal-size clusters.
- **Average linkage**: average of pairwise distances; intermediate behaviour.
- **Ward's linkage**: at each merge, choose the pair that minimises the increase in within-cluster sum of squares. Produces compact spherical clusters and is the most popular default.

The dendrogram is cut at a user-chosen height (or at k clusters via `cutree()`); the height of the cut is itself a diagnostic — short merges below the cut indicate tight clusters, tall merges above suggest meaningful separation.

8.15.4 Assumptions

A meaningful distance metric on the data and roughly independent observations. The chosen linkage criterion matches the expected cluster geometry; Ward's assumes Euclidean distance.

8.15.5 R Implementation

```r
library(factoextra)

X <- scale(iris[, 1:4])
d <- dist(X)

hc_ward <- hclust(d, method = "ward.D2")
fviz_dend(hc_ward, k = 3, rect = TRUE, palette = "Set2")

# Compare linkages
par(mfrow = c(2, 2))
for (m in c("single", "complete", "average", "ward.D2")) {
  plot(hclust(d, method = m), main = m, hang = -1)
}

# Cluster assignments at k = 3
cutree(hc_ward, k = 3)
```

8.15.6 Output & Results

`hclust()` returns the merging sequence and per-merge heights as an `hclust` object. `fviz_dend()` plots the dendrogram with cluster boxes and a colour palette. `cutree()` extracts cluster assignments at a chosen k. The four-panel comparison shows how dramatically linkage choice changes the resulting tree.

8.15.7 Interpretation

A reporting sentence: "Ward-linkage agglomerative clustering on standardised iris data, cut at $k = 3$, produced three clusters matching the species labels with 95 % agreement; single linkage produced a chained 1-iris-vs-149 partition that is uninformative for this dataset, illustrating the linkage choice's importance." Always disclose the linkage method and the distance metric used.

8.15.8 Practical Tips

- Ward's linkage (`ward.D2`) is the most reliable default for spherical clusters of similar size; it agrees with k-means in spirit but produces a tree.
- Single linkage often produces stringy chains; reach for it only when chain-like structures are substantively meaningful (e.g. minimum spanning trees in network analysis).
- Visualise the dendrogram before choosing k; the height of merges above the cut suggests meaningful versus arbitrary splits.
- For very large datasets, hierarchical clustering is $O(n^2)$ in memory and time; switch to k-means or DBSCAN above tens of thousands of observations.
- For mixed numeric-categorical data, compute Gower distance with `cluster::daisy()` and feed it to `hclust()`; Ward's then needs `method = "ward.D"` (not D2).
- Consider `cluster::agnes()` over `hclust()` for an agglomerative coefficient that quantifies the strength of the clustering structure (close to 1 = strong structure).

8.15.9 R Packages Used

Base R `hclust()` and `cutree()`; `cluster::agnes()` and `diana()` for agglomerative and divisive variants with structure indices; `factoextra` for `fviz_dend()` and `fviz_cluster()` ggplot-style visualisation; `dendextend` for advanced dendrogram manipulation including tanglegrams comparing two trees.

8.15.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.15.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.16 Independent Component Analysis

8.16.1 Introduction

Independent component analysis (ICA) is the canonical approach for recovering statistically independent latent signals from observed linear mixtures. The classic application is the "cocktail-party problem" — separating individual conversations from microphone recordings of overlapping speakers — but the technique generalises to any setting where multiple independent sources mix linearly into a smaller set of observed channels. EEG, fMRI, gene-expression data, and audio source separation all use ICA as a primary decomposition method.

8.16.2 Prerequisites

A working understanding of principal components analysis, statistical independence (a stronger condition than uncorrelatedness), and the concept of a linear mixing model.

8.16.3 Theory

The ICA model assumes observations X are generated as $X = AS$, where A is an unknown $n \times n$ mixing matrix and S contains the independent latent sources. The goal is to estimate A (or its inverse, the unmixing matrix W) so that $\hat{S} = WX$ recovers the sources up to permutation and scale ambiguities.

Two principles drive ICA algorithms:

- **Non-Gaussianity maximisation**: by the central limit theorem, sums of independent variables become more Gaussian; an unmixing direction that maximises non-Gaussianity (kurtosis, negentropy) recovers an independent source.
- **Statistical independence maximisation**: minimise mutual information between recovered components.

FastICA is the most widely used algorithm and uses fixed-point iteration on kurtosis or negentropy. Infomax is the alternative principle implemented in domain-specific tools.

The key contrast with PCA: PCA produces *uncorrelated* components by maximising variance; ICA produces *independent* components by maximising non-Gaussianity. The two coincide only for Gaussian data, where ICA is undefined.

8.16.4 Assumptions

Sources are statistically independent and non-Gaussian (at most one Gaussian source can be recovered); mixing is linear and instantaneous; the number of sources equals the number of observed channels (or equals the chosen number of components).

8.16.5 R Implementation

```r
library(fastICA)

set.seed(2026)
# Simulate two sources: sine and sawtooth
t <- seq(0, 8, length.out = 1000)
s1 <- sin(2 * pi * t)
s2 <- ((2 * t) %% 2) - 1      # sawtooth

# Mix
A <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
S <- cbind(s1, s2)
X <- S %*% t(A)

# Run ICA
ic <- fastICA(X, n.comp = 2)

par(mfrow = c(3, 1))
matplot(S, type = "l", main = "True sources")
matplot(X, type = "l", main = "Mixed signals")
matplot(ic$S, type = "l", main = "Recovered ICA components")
par(mfrow = c(1, 1))
```

8.16.6 Output & Results

`fastICA()` returns the recovered sources `$S`, the estimated mixing matrix `$A`, and the unmixing matrix `$W`. The three-panel plot shows the original sine and sawtooth sources, the mixed signals as observed, and the ICA-recovered components — visually almost identical to the originals up to sign and permutation.

8.16.7 Interpretation

A reporting sentence: “FastICA on two channels of mixed sine and sawtooth signals recovered both sources cleanly (visual inspection), demonstrating blind source separation under the linear-mixing model; recovered components were rescaled to unit variance and permuted, reflecting the inherent ICA ambiguities.” Mention the algorithm and pre-processing (whitening, mean-centring) in the methods section.

8.16.8 Practical Tips

- Pre-whiten the data (PCA-based decorrelation followed by unit-variance scaling) before FastICA; the `fastICA` package does this automatically.
- ICA is essentially a rotation of whitened data — order and sign of components are arbitrary; do not interpret component indices as meaningful.
- At most one Gaussian source can be recovered; if all sources are Gaussian, ICA cannot separate them.
- For fMRI and EEG, specialised implementations (MELODIC in FSL, runica in EEGLAB) handle the domain-specific challenges of large channel counts and physiological artefacts.
- For non-square mixing problems (more sources than channels), compressive ICA and multi-modal extensions exist but require specialised tools.
- ICA is unsupervised; evaluating the result requires interpreting the recovered signals substantively, often through expert judgement.

8.16.9 R Packages Used

`fastICA` for the standard FastICA implementation; `ica` for additional ICA variants including extended Infomax; `mlbench` for synthetic mixing examples; `JADE` for joint approximate diagonalisation of eigen-matrices, an alternative algebraic approach to ICA.

8.16.10 For Reviewers

Use the shared reviewer checklist, together with this method’s assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript’s application of the method.

8.16.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.17 Jaccard and Dice Dissimilarity

8.17.1 Introduction

Jaccard and Dice dissimilarities are the standard distance measures for binary (presence/absence) data: ecological community composition, disease symptom panels, gene set membership, document term frequencies after thresholding. Both indices share a key property: they ignore joint absences and focus only on agreements and disagreements among presences. This is exactly what is needed when an absence carries little information — a species absent from two sites might be absent because the habitat is unsuitable, or simply not detected — but is wrong when absences are themselves diagnostic.

8.17.2 Prerequisites

A working understanding of binary data, set-based similarity reasoning, and the conventional 2×2 contingency table for two binary vectors.

8.17.3 Theory

For two binary vectors x, y with counts a (both present), b (only in x), c (only in y), d (both absent):

- **Jaccard index**: $J = a/(a+b+c)$, the ratio of the intersection to the union of the present-set. **Jaccard dissimilarity** is $1 - J$.
- **Dice / Sørensen coefficient**: $D = 2a/(2a+b+c)$, with **Dice dissimilarity** $1 - D$. Dice doubles the weight on joint presences relative to mismatches.
- **Sørensen** and **Dice** are equivalent for binary data.

Both ignore d (joint absences), making them appropriate when absences are uninformative. Indices that include d — simple matching, Hamming distance — are preferable when joint absences carry meaning (e.g., binary medical diagnoses where "neither has condition X" is informative).

8.17.4 Assumptions

Binary data (or count data thresholded to binary), and joint absences are not meaningful for the inferential question. The same data on different absence-meaning assumptions can yield very different distances.

8.17.5 R Implementation

```r
library(vegan)

# Species abundance data (reduce to presence/absence)
data(dune)
dune_pa ← (dune > 0) * 1
head(dune_pa)

# Jaccard dissimilarity
d_jac ← vegdist(dune_pa, method = "jaccard", binary = TRUE)
as.matrix(d_jac)[1:3, 1:3]

# Sorensen (= Dice)
d_sor ← vegdist(dune_pa, method = "bray", binary = TRUE)
```

8.17.6 Output & Results

Two pairwise dissimilarity matrices on the dune presence/absence data, one for Jaccard and one for Sørensen-Dice. Jaccard distances are systematically larger than Dice for the same data because Dice double-counts joint presences in its denominator.

8.17.7 Interpretation

A reporting sentence: "Jaccard dissimilarity averaged 0.40 across all pairs of sites (range 0.16 to 0.65), indicating that 40 % of species presences are not shared between any randomly chosen pair; the corresponding Sørensen-Dice dissimilarity averaged 0.25, reflecting Dice's higher weight on joint presences." Always state which index was used; the same data produce different conclusions under Jaccard versus Dice.

8.17.8 Practical Tips

- Use Jaccard or Dice when joint absences are biologically or ecologically uninformative (rare species, sparse term-document data, gene-set membership).
- For abundance rather than presence-absence data, Bray-Curtis is the standard; Bray-Curtis on binarised data reduces to Sørensen-Dice.
- For microbiome data with phylogenetic information, UniFrac (`GUniFrac::GUniFrac`) is preferable because it weights distances by evolutionary relationships, not just shared OTUs.
- For binary medical data where joint absences are meaningful (neither patient has the condition), use simple matching (`dist(x, method = "binary")` for Hamming-style) rather than Jaccard.
- For very large sparse binary matrices (text, single-cell sparse counts), use the sparse-matrix-aware implementations in `proxy` or `parallelDist` to avoid memory blow-up.
- Pair dissimilarity matrices with NMDS or PCoA for visualisation; both ordinations consume Jaccard or Dice matrices directly.

8.17.9 R Packages Used

`vegan` for `vegdist()` with `method = "jaccard"` or `"bray"` and the broader ecology workflow; `proxy` for general dissimilarity functions including binary metrics; `philentropy` for distance and similarity coefficients across many families; `GUniFrac` for phylogenetic UniFrac alternatives in microbiome analysis.

8.17.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.17.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.18 Kernel PCA

8.18.1 Introduction

Kernel principal component analysis performs PCA in a high-dimensional feature space implicitly defined by a kernel function, capturing non-linear structure that linear PCA misses. The classic example — two concentric circles — illustrates the point: linear PCA fails to separate them because no straight axis distinguishes the two classes, but kernel PCA with a radial-basis-function kernel finds the radial coordinate in an implicit infinite-dimensional space and separates the rings cleanly. The technique generalises PCA to data lying on curved manifolds while preserving its eigen-decomposition mathematical structure.

8.18.2 Prerequisites

A working understanding of principal components analysis, the kernel trick, Gram matrices, and the difference between linear and non-linear feature spaces.

8.18.3 Theory

A kernel function $K(x, y) = \langle \phi(x), \phi(y) \rangle$ defines an inner product in a feature space without ever computing ϕ explicitly (the "kernel trick"). Standard choices:

- **RBF / Gaussian**: $K(x, y) = \exp(-\gamma \|x - y\|^2)$, infinite-dimensional feature space.
- **Polynomial**: $K(x, y) = (x^\top y + c)^d$, polynomial feature space of degree d.
- **Sigmoid**: $K(x, y) = \tanh(\alpha x^\top y + c)$, related to neural-net activations.

Construct the $n \times n$ Gram matrix K, centre it in feature space, and compute its top eigenvectors. The projection of an observation onto a kernel principal component is a weighted sum of kernel evaluations against all training observations — making kernel PCA $O(n^2)$ in space and time, which becomes prohibitive for $n \gg 10{,}000$.

8.18.4 Assumptions

The kernel encodes a similarity meaningful for the inferential question; the Gram matrix is positive semi-definite (true for valid Mercer kernels); sufficient observations for stable eigendecomposition.

8.18.5 R Implementation

```
library(kernlab)

# Data on two concentric circles (linear PCA fails)
set.seed(2026)
theta ← runif(200, 0, 2 * pi)
r ← c(rep(1, 100), rep(3, 100))
X ← cbind(r * cos(theta), r * sin(theta)) + matrix(rnorm(400, 0, 0.1), 200)
class ← factor(r)

# Linear PCA
pca ← prcomp(X)
plot(pca$x[, 1:2], col = class, main = "Linear PCA")

# Kernel PCA with RBF
kpca ← kpca(X, kernel = "rbfdot", kpar = list(sigma = 0.1), features = 2)
plot(rotated(kpca), col = class, main = "Kernel PCA (RBF)")
```

8.18.6 Output & Results

Two side-by-side scatterplots: linear PCA leaves the two concentric rings overlapping in PC space, while kernel PCA with RBF kernel and $\sigma = 0.1$ separates them cleanly along the first kernel principal component.

8.18.7 Interpretation

A reporting sentence: “Kernel PCA with an RBF kernel ($\sigma = 0.1$) separated the two concentric rings (n = 100 each) that linear PCA could not, demonstrating that the underlying class structure is non-linear in the original feature space; the first kernel principal component captured the radial coordinate that distinguishes the classes.” When kernel PCA succeeds where linear PCA fails, name the geometric feature being captured.

8.18.8 Practical Tips

- Kernel choice and hyperparameters matter; RBF with median-heuristic bandwidth is a reasonable default, then tune by cross-validation on a downstream supervised task.
- Kernel PCA components are not directly interpretable in the original feature space; they are weighted sums of kernel evaluations against training points.

- Computation scales as $O(n^2)$ space and $O(n^3)$ time for the eigendecomposition; for $n > 10{,}000$, use Nyström approximation (`kernlab::kha`, `RSpectra` for partial eigendecomposition).
- Modern alternatives for non-linear visualisation — UMAP, t-SNE, autoencoders — often outperform kernel PCA for cluster separation in two dimensions.
- Kernel PCA is most useful as a feature engineering step for downstream supervised learning when p is very small; in $p \gg n$ regimes, sparse linear methods (PLS, lasso) are typically more efficient.
- For preserving local geometry, prefer Laplacian eigenmaps or diffusion maps; kernel PCA preserves variance in the implicit feature space, not local distances.

8.18.9 R Packages Used

`kernlab` for `kpca()` and a wide range of kernels; `RSpectra` for partial eigendecomposition of large Gram matrices; `uwot` (UMAP) and `Rtsne` for modern non-linear embedding alternatives; `dimRed` for a unified interface across kernel PCA, t-SNE, UMAP, and Isomap.

8.18.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.18.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.19 k-Means Clustering

8.19.1 Introduction

k-means is the workhorse of partitioning cluster analysis: assign each of n observations to one of k clusters, with the cluster centres as multivariate means and the objective being to minimise the total within-cluster sum of squared distances. It scales to large datasets, is conceptually simple, and produces interpretable centres in the original feature space. Its limitations — sensitivity to outliers, dependence on Euclidean geometry, the need to pre-specify k, and the assumption of approximately spherical equal-size clusters — are equally well known and motivate the alternatives (PAM, DBSCAN, Gaussian mixtures) in this category.

8.19.2 Prerequisites

A working understanding of distance metrics, multivariate means, and the trade-off between within-cluster compactness and between-cluster separation.

8.19.3 Theory

Lloyd's algorithm alternates two steps until convergence:

1. **Initialise** k centres, either randomly or via the k-means++ heuristic which spreads initial centres apart for better convergence.
2. **Assign** each observation to the cluster of its nearest centre.
3. **Update** each centre to the mean of its assigned observations.
4. **Repeat** until cluster assignments no longer change.

The objective is

$$\mathrm{WSS} = \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \bar{x}_j\|^2,$$

monotonically decreasing across iterations. The algorithm converges to a local optimum, so multiple random starts (`nstart ⩾ 25`) are needed to find a good solution. Choice of k uses the elbow method, silhouette analysis, gap statistic, or composite tools like `NbClust`.

8.19.4 Assumptions

Continuous variables on comparable scales (standardise first); roughly spherical clusters of similar size; Euclidean distance is meaningful. Outliers can drag means and produce unbalanced clusters; PAM is preferable when outliers are present.

8.19.5 R Implementation

```r
library(factoextra)

X ← scale(iris[, 1:4])

# 3 clusters
set.seed(2026)
km ← kmeans(X, centers = 3, nstart = 25)
km$size
km$centers

fviz_cluster(km, data = X, palette = "Set2", ggtheme = theme_minimal())

# Elbow and silhouette
fviz_nbclust(X, kmeans, method = "wss")
fviz_nbclust(X, kmeans, method = "silhouette")
```

8.19.6 Output & Results

`kmeans()` returns cluster assignments, cluster centres in the standardised feature space, total within-cluster sum of squares, between-cluster sum of squares, and convergence diagnostics. `fviz_cluster()` plots the clusters on the first two principal components with cluster ellipses; `fviz_nbclust()` automates elbow and silhouette diagnostics.

8.19.7 Interpretation

A reporting sentence: "k-means with $k = 3$ and 25 random starts partitioned the standardised iris data with total within-cluster sum of squares 140; the resulting clusters closely matched the three species (overall accuracy 89 %, with versicolor and virginica contributing the misclassifications)." Always state k, the number of starts, and a quality metric (silhouette or WSS).

8.19.8 Practical Tips

- Always standardise predictors before k-means; variables with larger variance dominate Euclidean distance and bias the partition.
- Use `nstart = 25` or higher to avoid poor local optima; the algorithm is non-convex and a single random start can converge to a sub-optimal solution.
- For non-spherical clusters, switch to hierarchical clustering, DBSCAN (density-based), or Gaussian mixture models (model-based with elliptical clusters).
- For categorical data, k-means does not apply directly; use k-modes (`klaR::kmodes`) or PAM with Gower distance.
- Cross-validate the chosen k via stability measures (`fpc::clusterboot`); clusters that recur under bootstrapping are more trustworthy than those that change with each resample.
- Outliers can drag means; for outlier-contaminated data, prefer PAM (`cluster::pam`), trimmed k-means (`tclust::tkmeans`), or robust mixture models.

8.19.9 R Packages Used

Base R `kmeans()` for the standard algorithm; `factoextra` for `fviz_cluster()`, `fviz_nbclust()`, and ggplot-style diagnostics; `cluster` for related partitioning algorithms; `klaR::kmodes()` for categorical data; `fpc::clusterboot()` for stability-based cluster validation.

8.19.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.19.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.20 Linear Discriminant Analysis

8.20.1 Introduction

Linear discriminant analysis (LDA) is the canonical Gaussian classifier for two or more classes. Under the assumption that within-class distributions are multivariate Normal with a common covariance matrix, the optimal Bayes classifier is linear in the predictors, and the corresponding posterior probabilities are simple closed-form functions of class means and the pooled covariance. Fisher's complementary derivation gives a projection onto $K-1$ axes that maximally separates the K class centroids; this is the most useful low-dimensional visualisation for class structure.

8.20.2 Prerequisites

A working understanding of multivariate Normal distributions, classification metrics, and the Bayes-optimal classifier under known generative assumptions.

8.20.3 Theory

Assume class k has prior π_k, mean μ_k, and common covariance Σ. The Bayes-optimal log-discriminant is

$$\delta_k(x) = x^\top \Sigma^{-1}\mu_k - \tfrac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \log \pi_k.$$

Assigning x to the class with the largest δ_k gives the LDA classifier. The decision boundary between any two classes is linear in x, which is what makes the model interpretable: classification depends on a small number of linear combinations of the predictors.

Fisher's LDA, conceptually identical, finds the directions in which the between-class variance is large relative to the within-class variance; projecting onto the top $K-1$ directions gives the canonical low-dimensional visualisation of class structure.

8.20.4 Assumptions

Multivariate Normal within each class, equal class covariance matrices (otherwise QDA is preferred), and independent observations. Outliers can pull class means and inflate the pooled covariance, distorting the linear boundary.

8.20.5 R Implementation

```r
library(MASS)

set.seed(2026)
train_idx <- sample(nrow(iris), 100)
train <- iris[train_idx, ]; test  <- iris[-train_idx, ]

fit_lda <- lda(Species ~ ., data = train)
pred <- predict(fit_lda, test)
table(pred$class, test$Species)

# Project onto LDA axes
plot(fit_lda, col = as.integer(train$Species))
```

8.20.6 Output & Results

`lda()` returns class priors, class means, scaling matrix (coefficients of linear discriminants), and the proportion of trace each axis explains. `predict()` provides class labels and posterior probabilities. The classifier projects test observations onto the LDA axes and assigns each to the nearest class centroid in the projected space.

8.20.7 Interpretation

A reporting sentence: “Linear discriminant analysis on the iris training set classified the held-out observations with 98 % accuracy (49/50 correct); the first discriminant axis accounted for 99 % of between-group variance, indicating that a single linear combination of the four measurements largely separates the three species.” Compare LDA to QDA when class covariances differ; otherwise LDA is the more efficient choice.

8.20.8 Practical Tips

- For class-specific covariance differences, switch to quadratic discriminant analysis (QDA); Box’s M test (`biotools::boxM`) provides a formal check, but it is sensitive to non-Normality.
- LDA is sensitive to outliers; screen for them or use robust variants such as `MASS::lda(... , method = "mve")` or `rrcov::Linda()`.
- Prior class probabilities affect the decision boundary; specify them via `prior = c(...)` when training-set proportions differ from population proportions.
- For high-dimensional data with p approaching or exceeding n, regularise via `klaR::rda()` (regularised discriminant analysis) or sparse LDA (`MASS::lda` with PCA pre-step, or `sparseLDA`).
- Multiclass alternatives include multinomial logistic regression (`nnet::multinom`) and random forests; LDA remains competitive in low-dimensional problems with Gaussian-like classes.
- Visualise the LDA projection with the first two discriminants; a 2D plot with class colours quickly shows whether classes are linearly separable.

8.20.9 R Packages Used

`MASS` for `lda()` and `qda()`; `klaR` for `rda()` (regularised) and `partimat()` for partition diagnostics; `caret` for cross-validated comparison across discriminant variants; `rrcov` for robust LDA; `sparseLDA` for high-dimensional sparse extensions.

8.20.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.20.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.21 Mahalanobis Distance

8.21.1 Introduction

Mahalanobis distance is the multivariate generalisation of "how many standard deviations from the mean" that respects scale and correlation between variables. Where Euclidean distance treats each axis equally and ignores covariance, Mahalanobis pre-multiplies the deviation by the inverse covariance matrix, so directions with high variance contribute less and correlated directions are decorrelated before measurement. It is the foundation of multivariate outlier detection, the discriminant function in LDA and QDA, and Hotelling's T^2 statistic.

8.21.2 Prerequisites

A working understanding of covariance matrices, multivariate Normal distributions, and the chi-squared distribution.

8.21.3 Theory

For a point $x \in \mathbb{R}^p$, mean vector μ, and covariance Σ:

$$D_M(x, \mu, \Sigma) = \sqrt{(x-\mu)^\top \Sigma^{-1} (x-\mu)}.$$

Under multivariate Normality, D_M^2 follows a chi-squared distribution with p degrees of freedom. This is the basis for probabilistic outlier flagging: points with D_M^2 exceeding the $1-\alpha$ chi-squared quantile are unlikely under the null. Substituting the sample mean and

sample covariance gives the standard estimator, but the sample covariance is itself influenced by outliers — leading to "masking" where extreme points hide behind each other. Robust estimators (Minimum Covariance Determinant, MCD; Minimum Volume Ellipsoid, MVE) trim the most-influential observations before estimating Σ.

8.21.4 Assumptions

Multivariate Normality for the chi-squared reference distribution; positive-definite Σ (or its pseudo-inverse if rank-deficient); sufficient sample size relative to p for stable covariance estimation.

8.21.5 R Implementation

```r
library(robustbase)

X ← iris[, 1:4]
mu ← colMeans(X)
Sigma ← cov(X)
MD2 ← mahalanobis(X, mu, Sigma)

# Threshold at 97.5% chi-squared quantile
cutoff ← qchisq(0.975, df = 4)
which(MD2 > cutoff)

# Robust version (MCD)
mcd ← covMcd(X)
MD2_rob ← mahalanobis(X, mcd$center, mcd$cov)
which(MD2_rob > cutoff)
```

8.21.6 Output & Results

`mahalanobis()` returns squared distances under the supplied centre and covariance. Indexing by the chi-squared cutoff identifies candidate outliers. The robust MCD-based version typically flags more outliers because the classical estimator is itself distorted by the very points it should flag.

8.21.7 Interpretation

A reporting sentence: "Classical Mahalanobis distance flagged 4 iris observations above the χ^2_4 97.5 % cutoff, while the robust MCD-based estimator flagged 12; the additional observations were masked by their joint influence on the classical covariance estimate, illustrating why robust methods are preferable for outlier diagnostics in non-trivial multivariate data." Always pair classical and robust Mahalanobis when reporting outliers; agreement increases confidence, disagreement signals masking.

8.21.8 Practical Tips

- Classical Mahalanobis suffers from masking — outliers distort the very covariance used to detect them. Always pair with MCD or MVE for any non-trivial outlier-detection task.

- For high-dimensional data ($p \gg n$), covariance is rank-deficient; use regularised estimators (`corpcor::cov.shrink`) or first reduce dimensionality.
- Mahalanobis distance underlies LDA and QDA: the discriminant function is the Mahalanobis distance to each class centre, weighted by class priors.
- For multivariate Normality assessment, plot ordered Mahalanobis distances against chi-squared quantiles; deviations from the diagonal indicate non-Normality or outliers.
- The cutoff χ^2_p assumes Normality; for non-Normal data, calibrate the cutoff via simulation or use a robust universal cutoff.
- Use the squared form D^2_M in reporting; it has the chi-squared reference distribution and avoids the square-root computation.

8.21.9 R Packages Used

Base R `mahalanobis()`; `robustbase::covMcd()` for MCD-based robust covariance; `MASS::cov.mve()` for MVE; `rrcov::CovMcd()` and `CovMve()` for additional robust estimators with formal classes; `mvoutlier::aq.plot()` for adjusted quantile plots that compare classical and robust Mahalanobis distances.

8.21.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.21.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.22 MANOVA

8.22.1 Introduction

MANOVA — multivariate analysis of variance — tests whether group means differ across several dependent variables simultaneously, accounting for the correlations among the outcomes. It is more powerful than running separate one-way ANOVAs when outcomes are correlated, because it pools information across the response set instead of testing each in isolation. It also controls Type I error at the family-wise level for the joint hypothesis, removing the need for multiple-comparison correction across a battery of univariate ANOVAs.

8.22.2 Prerequisites

A working understanding of one-way ANOVA, multivariate Normal distributions, and covariance matrices.

8.22.3 Theory

MANOVA decomposes the total cross-product matrix into hypothesis (H) and error (E) components and constructs four standard multivariate test statistics from their joint structure:

- **Wilks' lambda**: $\Lambda = |E|/|H+E|$, the classical choice; small values reject the null.
- **Pillai's trace**: $V = \mathrm{tr}(H(H+E)^{-1})$, robust to covariance heterogeneity and the recommended default.
- **Hotelling–Lawley trace**: $T = \mathrm{tr}(HE^{-1})$, more powerful when one dimension dominates.
- **Roy's greatest root**: $\theta =$ largest eigenvalue of HE^{-1}, most powerful when group differences are concentrated in one direction but liberal otherwise.

All four test the same omnibus null hypothesis but have slightly different operating characteristics under non-Normality and unequal covariances.

8.22.4 Assumptions

Multivariate Normal within groups, homogeneous covariance matrices across groups (Box's M test), and independent observations.

8.22.5 R Implementation

```
fit ← manova(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~
             Species, data = iris)
summary(fit)                     # Pillai default
summary(fit, test = "Wilks")
summary(fit, test = "Hotelling-Lawley")

# Follow-up univariate ANOVAs
summary.aov(fit)

# car Anova for Type II/III
library(car)
Manova(lm(cbind(Sepal.Length, Sepal.Width) ~ Species, data = iris))
```

8.22.6 Output & Results

`summary(fit)` returns the Pillai trace by default, with F approximation, degrees of freedom, and p-value. Specifying other tests via the `test` argument produces Wilks, Hotelling-Lawley, and Roy variants. `summary.aov(fit)` runs follow-up univariate ANOVAs on each outcome.

8.22.7 Interpretation

A reporting sentence: "MANOVA showed a significant multivariate species effect (Wilks' $\Lambda = 0.02$, $F(8, 288) = 199$, $p < 0.001$, multivariate $\eta^2 = 0.86$); univariate follow-up ANOVAs confirmed significant species differences on each of the four flower dimensions ($p < 0.001$ for each), with petal length carrying the strongest effect ($\eta^2 = 0.94$)." Always pair MANOVA with both univariate follow-ups and a multivariate effect-size measure.

8.22.8 Practical Tips

- Pillai's trace is the recommended default — it is the most robust to covariance heterogeneity and unequal sample sizes.
- For a significant MANOVA, follow up with univariate ANOVAs on each outcome to identify which dimensions drive the effect; corrected p-values (Bonferroni or Holm) are appropriate at this stage.
- Test homogeneity of covariance with Box's M (`biotools::boxM`) before relying on Wilks; a significant Box's M motivates Pillai.
- For unequal sample sizes and multifactorial designs, use `car::Manova()` with explicit Type II or Type III sums of squares; base R's `summary.manova` defaults to sequential (Type I).
- Discriminant analysis follows naturally as a post-MANOVA exploration; the linear discriminants visualise where the multivariate group differences live.
- For few groups with many highly-correlated outcomes, MANOVA is more powerful than Bonferroni-corrected univariate tests; for many groups with few outcomes, separate ANOVAs may be more interpretable.

8.22.9 R Packages Used

Base R `manova()` and `summary.manova()`; `car::Manova()` for Type II / III sums of squares; `biotools::boxM()` for the homogeneity-of-covariance test; `MASS::lda()` for discriminant-analysis follow-up; `heplots` for visualising hypothesis-error decompositions.

8.22.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.22.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.23 Metric Multidimensional Scaling

8.23.1 Introduction

Metric multidimensional scaling — also known as classical MDS or principal coordinates analysis (PCoA) — takes a pairwise distance matrix and produces a low-dimensional Euclidean configuration whose pairwise distances approximate the input as closely as possible. It is the workhorse for visualising similarity or distance data when raw coordinates are unavailable: think reconstructing a map from inter-city distances, projecting microbial-community Bray-Curtis dissimilarities, or visualising a kernel-derived similarity matrix in a 2D plot.

8.23.2 Prerequisites

A working understanding of distance and similarity matrices, principal components analysis, and basic linear algebra (eigendecomposition).

8.23.3 Theory

Given an $n \times n$ distance matrix D, form the doubly-centred inner-product matrix

$$B = -\tfrac{1}{2}\big(I - J/n\big) D^{(2)} \big(I - J/n\big),$$

where $D^{(2)}$ has elements d_{ij}^2 and J is the all-ones matrix. Eigendecompose $B = U \Lambda U^\top$. The top k eigenvectors scaled by $\sqrt{\lambda}$ give a k-dimensional configuration whose Euclidean inter-point distances approximate the original distances. When D comes from Euclidean distances on raw data, classical MDS is equivalent to PCA on the centred data — the eigenvalues match the squared singular values of the centred data matrix.

For non-Euclidean distances, B may have negative eigenvalues; the configuration still uses only the top positive eigenvalues, with the discarded negative mass quantifying the embedding error.

8.23.4 Assumptions

A pairwise distance matrix that is at least approximately Euclidean-embeddable. Severe violations (negative eigenvalues dominating) suggest non-metric MDS as a more appropriate alternative.

8.23.5 R Implementation

```r
# Classical MDS on US city distances
data(UScitiesD)
mds ← cmdscale(UScitiesD, k = 2, eig = TRUE)
plot(mds$points, type = "n")
text(mds$points, labels = labels(UScitiesD))

# Eigenvalues and variance explained
mds$eig[1:5]
cumsum(mds$eig[1:5] / sum(mds$eig[1:5]))
```

8.23.6 Output & Results

`cmdscale()` returns the low-dimensional configuration (`$points`), eigenvalues, and an indicator of how well the chosen k approximates the original distances. The plot of US-city MDS coordinates recovers the recognisable geographical structure with a near-perfect rotation and reflection of the actual map.

8.23.7 Interpretation

A reporting sentence: “Classical MDS on the 10 inter-city distances produced a two-dimensional configuration that captured 99 % of the eigen-mass; the resulting layout matches actual geography after a reflection, with east-west and north-south structure immediately visible.” Always report the proportion of eigen-mass retained in the displayed dimensions; a poor approximation signals that more dimensions or non-metric MDS would help.

8.23.8 Practical Tips

- For Euclidean distances, classical MDS is equivalent to PCA on the centred data; if you have raw data, run PCA directly for cleaner output.
- For non-Euclidean distances (Bray-Curtis, Jaccard, network distances), check the eigenvalue structure — large negative eigenvalues indicate a non-Euclidean metric and motivate non-metric MDS instead.
- In ecology and microbiome research, this method is called principal coordinates analysis (PCoA); the algorithm is identical.
- Add Procrustes analysis (`vegan::procrustes`) when comparing MDS solutions across related datasets; it aligns configurations up to rotation, reflection, and scaling.
- For configurations beyond two dimensions, plot pairs of axes; `factoextra::fviz_mds()` provides a ggplot-style alternative.
- Pair the configuration with a stress measure or the proportion of variance retained to communicate fit quality alongside the visual.

8.23.9 R Packages Used

Base R `cmdscale()` for classical MDS; `vegan::wcmdscale()` for weighted classical MDS in ecology; `MASS::isoMDS()` and `vegan::metaMDS()` for non-metric alternatives; `factoextra::fviz_mds()` for ggplot-style visualisation; `smacof` for advanced MDS

variants including weighted and constrained versions.

8.23.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.23.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.24 Non-Metric Multidimensional Scaling

8.24.1 Introduction

Non-metric multidimensional scaling (NMDS) preserves the *rank order* of pairwise dissimilarities rather than their absolute magnitudes. This robustness makes it the standard ordination method for ecological community data, where Bray-Curtis and Jaccard dissimilarities are not Euclidean and where distance values themselves carry less meaning than the ordering of similarities. Where classical MDS works on the squared distances directly, NMDS minimises a stress function that depends only on whether the configuration's distances respect the input ranking.

8.24.2 Prerequisites

A working understanding of metric MDS (classical / PCoA), rank-based statistics, and the distinction between metric and ordinal distance information.

8.24.3 Theory

NMDS finds a configuration Y in k dimensions whose pairwise Euclidean distances $d(Y_i, Y_j)$ are monotonically related to the input dissimilarities δ_{ij}. The optimisation minimises Kruskal stress:

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

where $\hat{d}_{ij}$ are *disparities* — values fitted via isotonic regression to be a non-decreasing function of δ_{ij}. The Shepard diagram plots δ_{ij} against d_{ij} with the isotonic fit overlaid; tight scatter around a smooth monotone curve indicates good ordination.

Stress interpretation (Kruskal): below 0.05 is excellent, 0.05–0.10 good, 0.10–0.20 fair, above 0.20 poor and the configuration should not be trusted.

8.24.4 Assumptions

Dissimilarity ranks are meaningful; sufficient observations for a stable configuration; the chosen number of dimensions (k) is large enough to accommodate the rank structure.

8.24.5 R Implementation

```r
library(MASS); library(vegan)

# Non-metric MDS on UScities
data(UScitiesD)
iso ← isoMDS(UScitiesD, k = 2)
iso$stress

# ecology example with vegan
data(varespec)
nmds ← metaMDS(varespec, distance = "bray", k = 2, trymax = 50)
nmds$stress
plot(nmds, type = "t")
```

8.24.6 Output & Results

`isoMDS()` and `metaMDS()` return a 2D configuration plus the stress value. `metaMDS()` also runs multiple random starts (`trymax`) and returns the best solution; this is essential because NMDS is non-convex and prone to local minima.

8.24.7 Interpretation

A reporting sentence: “NMDS of the vegetation community (Bray-Curtis dissimilarity, two dimensions, 50 random starts) achieved a stress of 0.18 — fair fit — and the configuration revealed two dominant environmental gradients corresponding to moisture and nutrient availability when site-level covariates were overlaid via `envfit()`.” Always report the stress value and the number of starts; a single-start fit can land in a poor local minimum.

8.24.8 Practical Tips

- Always check stress; values above 0.2 indicate the configuration is unreliable and a higher-dimensional solution may be needed.
- Use multiple random starts (`trymax = 50` or more in `metaMDS`); NMDS is non-convex and a single start often lands in a local minimum.
- NMDS is the standard for non-Euclidean ecological dissimilarities; metric MDS forces a Euclidean embedding that may not respect the underlying geometry.
- Procrustes-align configurations across related datasets (`vegan::procrustes`); NMDS solutions are defined up to rotation, reflection, and uniform scaling.
- Pair NMDS with `vegan::envfit()` or `ordisurf()` to overlay environmental covariates as arrows or contours, exposing the gradients the ordination has captured.

- Interpret relative positions only; absolute distances in the NMDS plot do not have a single fixed scale.

8.24.9 R Packages Used

`MASS::isoMDS()` for the classical Kruskal NMDS implementation; `vegan::metaMDS()` for the ecology-flavoured workflow with multiple starts and stress reporting; `smacof::smacofSym()` for advanced NMDS variants with weights and constraints; `ade4::nmds` for an alternative interface integrated with the dudi family of multivariate methods.

8.24.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.24.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.25 Multiple Correspondence Analysis

8.25.1 Introduction

Multiple correspondence analysis (MCA) is the categorical analogue of principal components analysis. Where PCA reduces a matrix of continuous variables to a small number of orthogonal axes capturing maximal variance, MCA reduces a set of categorical variables to axes capturing maximal chi-squared inertia. It is the natural exploratory tool for survey data, lifestyle patterns, ICD code panels, and any context in which several categorical variables jointly characterise units of observation.

8.25.2 Prerequisites

A working understanding of correspondence analysis on a two-way contingency table, indicator (disjunctive) coding of categorical variables, and the chi-squared distance interpretation.

8.25.3 Theory

MCA operates on the disjunctive-coded indicator matrix that represents each categorical variable by one binary column per category. Two equivalent computations exist:

- **Indicator-matrix CA**: apply standard correspondence analysis to the $n \times \sum K_q$ indicator matrix.
- **Burt-matrix CA**: apply CA to the Burt table, the cross-tabulation of all pairs of variables; this gives identical row and column structure with sometimes more intuitive geometry.

Each category becomes a point in the resulting low-dimensional space; each individual is also placed in the space (the "cloud of individuals"). Inertia per axis is typically low — much lower than typical PCA explained variance — because each indicator column contributes only one dimension of variation. Benzécri's or Greenacre's adjusted inertia formulas correct for this and give more meaningful variance-explained percentages.

8.25.4 Assumptions

Categorical variables (no strong zero cells), sufficient sample size relative to the number of categories, and a meaningful joint structure to extract.

8.25.5 R Implementation

```r
library(FactoMineR); library(factoextra)

data(hobbies)
mca ← MCA(hobbies[, 1:8], graph = FALSE)
fviz_mca_biplot(mca)
fviz_mca_var(mca)

summary(mca)
```

8.25.6 Output & Results

`MCA()` returns coordinates for individuals and category points, inertia per axis, contributions, and squared cosines (quality of representation). `fviz_mca_biplot()` displays both individuals and categories in the first two dimensions; `fviz_mca_var()` zooms in on the category cloud.

8.25.7 Interpretation

A reporting sentence: "MCA of 8 hobby variables (n = 8,403 respondents) extracted two interpretable dimensions: the first contrasts active vs sedentary hobbies (Benzécri-adjusted inertia 28 %), the second social vs solitary hobbies (adjusted inertia 19 %); supplementary projection of age and gender showed a clear age gradient on the first dimension and a weak gender effect on the second." Always report adjusted inertia rather than raw inertia; raw values systematically understate explained variation.

8.25.8 Practical Tips

- Use Benzécri's or Greenacre's adjusted inertia (`FactoMineR::MCA()` reports both) for variance-explained percentages; raw inertias are pessimistic and misleading.

- Supplementary categorical variables (`quali.sup`) project onto the axes without affecting their estimation; useful for adding demographic markers post-hoc.
- For mixed continuous-categorical data, switch to factor analysis of mixed data (FAMD); it generalises PCA and MCA jointly.
- Visualise the category cloud separately from the individual cloud when the latter is dense; `fviz_mca_var()` gives a cleaner view of category geometry.
- For ordinal categorical data, MCA loses the ordering — consider polychoric PCA or graded-response IRT models instead.
- Pair MCA with hierarchical clustering on the principal component scores (`HCPC()`) for a single workflow that combines exploration and partitioning.

8.25.9 R Packages Used

`FactoMineR` for `MCA()`, `FAMD()`, and `HCPC()`; `factoextra` for `fviz_mca_*` ggplot-style visualisation; `ca::mjca()` for an alternative Burt-table implementation; `ade4::dudi.acm()` for ecology-flavoured MCA.

8.25.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.25.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.26 Partitioning Around Medoids

8.26.1 Introduction

Partitioning around medoids (PAM) is the canonical k-medoids clustering algorithm. Like k-means, it partitions n observations into k clusters; unlike k-means, the cluster centres are actual observations (medoids) rather than coordinate means. This single change makes PAM robust to outliers (a single extreme value cannot drag a medoid away from the cluster bulk), enables clustering on arbitrary dissimilarity matrices including those derived from non-Euclidean distances or mixed numeric-categorical data, and produces interpretable centre observations that can be quoted in clinical or biological contexts ("the typical patient in cluster 2 is patient 47").

8.26.2 Prerequisites

A working understanding of k-means, distance and dissimilarity matrices, and the difference between mean-based and median-like cluster centres.

8.26.3 Theory

PAM alternates two steps:

1. **Build / swap**: for each cluster, choose the medoid that minimises the total dissimilarity to all other cluster members.
2. **Assign**: assign each observation to the cluster of its nearest medoid.

The objective is the sum of dissimilarities to the assigned medoid, summed over all observations. The algorithm converges to a local minimum; multiple random initialisations help find a better local optimum. Because PAM consumes a dissimilarity matrix, it works with Gower distance for mixed numeric-categorical data, Manhattan distance for outlier-robust clustering, correlation-based distances for gene-expression patterns, and any custom dissimilarity that the analyst can define.

8.26.4 Assumptions

A meaningful dissimilarity (or distance) on the input space. Approximately equal cluster sizes are preferable; PAM does not handle very imbalanced clusters as gracefully as model-based methods (Gaussian mixtures).

8.26.5 R Implementation

```r
library(cluster)

# Mixed-type data with Gower distance
d_iris ← iris
set.seed(2026)
d_iris$Category ← factor(sample(c("A", "B", "C"), 150, replace = TRUE))

g ← daisy(d_iris, metric = "gower")
pam_fit ← pam(g, k = 3)

pam_fit$medoids
table(pam_fit$clustering, iris$Species)

# Silhouette
summary(silhouette(pam_fit))
```

8.26.6 Output & Results

`pam()` returns medoid identifiers (actual case indices), cluster assignments, and within-cluster summaries. `silhouette()` returns per-observation silhouette widths and an average — a primary internal validation metric.

8.26.7 Interpretation

A reporting sentence: "Partitioning around medoids on Gower-distance dissimilarities produced three clusters that closely match the iris species (Cluster 1: setosa; Cluster 2: 92 % versicolor; Cluster 3: 86 % virginica) with average silhouette width 0.45 (range -0.10 to 0.78)." Always pair PAM with a silhouette analysis to verify that the chosen k produces well-separated clusters.

8.26.8 Practical Tips

- Choose PAM over k-means when outliers are likely or when the data are mixed-type or genuinely non-Euclidean; the medoid is robust where the mean is not.
- `cluster::pam()` accepts either a data matrix (computing Euclidean distance internally) or a pre-computed `dist` object via the `diss = TRUE` argument.
- For large datasets ($n > 1{,}000$), `cluster::clara()` runs PAM on sub-samples and combines results, scaling to tens of thousands of observations.
- Use `fpc::pamk()` to automatically select k via the silhouette method; it returns the optimal k along with the fit.
- For Gower distances on mixed numeric and categorical data, `cluster::daisy(metric = "gower")` is the standard input; categorical variables become indicator distances, numeric ones become Manhattan after range normalisation.
- For very large n, alternatives such as CLARANS, BIRCH, or mini-batch k-means scale better; the price is reduced robustness or interpretability.

8.26.9 R Packages Used

`cluster` for `pam()`, `clara()`, `daisy()`, and `silhouette()`; `fpc` for `pamk()` automatic k selection; `factoextra` for `fviz_cluster()` ggplot-style cluster visualisation; `klaR::kmodes()` for an analogous categorical-only k-modes algorithm.

8.26.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.26.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.27 PLS Discriminant Analysis

8.27.1 Introduction

PLS discriminant analysis (PLS-DA) extends partial least squares regression to classification by dummy-coding the class variable as a binary indicator matrix. It has become a standard tool in metabolomics, transcriptomics, and other omics disciplines because it handles the $p \gg n$ regime gracefully and, in its sparse variant, performs simultaneous classification and feature selection. The output — a small set of latent components plus the discriminant boundary in the projected space — is interpretable in a way that black-box classifiers (random forests, deep nets) often are not.

8.27.2 Prerequisites

A working understanding of PLS regression, classification metrics, and the high-dimensional ($p \gg n$) regime that motivates latent-component methods.

8.27.3 Theory

Encode the class variable as one-hot indicators $Y \in \{0, 1\}^{n \times K}$ for K classes. PLS regression on (X, Y) produces latent components that maximise covariance with the indicator matrix; the predicted indicators are continuous, and a new sample is assigned to the class with the largest prediction. The number of components is selected by cross-validation (`mixOmics::perf()`).

Sparse PLS-DA (sPLS-DA) imposes an L1 penalty per component, retaining only `keepX` predictors per component. This produces a signature — a small list of variables driving the classification — that can be reported alongside the classifier itself.

8.27.4 Assumptions

Linear relationships between latent components and outcome probabilities, sample sizes large enough per class to estimate the components stably (rule of thumb: at least 5–10 observations per class), and meaningful covariance between predictors and the encoded indicators.

8.27.5 R Implementation

```r
library(mixOmics)

data(srbct)
X ← srbct$gene
Y ← srbct$class

# PLS-DA
fit_plsda ← plsda(X, Y, ncomp = 3)
plotIndiv(fit_plsda, ind.names = FALSE, legend = TRUE)

# Sparse PLS-DA: select 50 variables per component
fit_splsda ← splsda(X, Y, ncomp = 3, keepX = c(50, 50, 50))
selectVar(fit_splsda, comp = 1)$name ▷ head()
```

8.27.6 Output & Results

`plsda()` returns latent component scores per sample, loadings for each predictor, and predicted class labels. `splsda()` adds variable-selection-per-component, with `selectVar()` extracting the chosen predictors. `plotIndiv()` displays samples in the latent component space, coloured by class.

8.27.7 Interpretation

A reporting sentence: “Sparse PLS-DA on the 2,308-gene SRBCT dataset selected 50 genes per component across three components, achieving 100 % balanced accuracy in 10-fold cross-validation; the resulting 150-gene signature substantially overlaps with the published transcriptomic markers of the four cancer subtypes.” Always report both classification performance (accuracy, balanced accuracy, AUC) and the selected feature signature.

8.27.8 Practical Tips

- Use `mixOmics::perf()` with cross-validation to select `ncomp` and `keepX`; a fixed choice without tuning over-fits in high-dimensional omics.
- For two-class problems, logistic regression with elastic net regularisation often performs comparably and is more familiar; reserve sPLS-DA for multi-class or when the covariance structure is strongly multivariate.
- Sparse PLS-DA can produce unstable variable selections in the presence of highly correlated predictors; report selection stability via bootstrap resampling (`mixOmics::perf()` with `nrepeat = 50`).
- The selected signature is the headline result for biological audiences; pair it with pathway-enrichment analysis to interpret functionally.
- Compare against alternative classifiers (random forest, elastic-net logistic, sPLS-DA) on identical CV folds; modern omics papers typically report multiple methods.
- For very imbalanced classes, use `near.zero.var = TRUE` and balanced cross-validation to prevent the majority class from dominating component construction.

8.27.9 R Packages Used

`mixOmics` for `plsda()`, `splsda()`, `perf()`, `selectVar()`, and the broader multi-block extensions; `caret` for unified cross-validated benchmarking against alternative classifiers; `ropls` for an alternative PLS-DA implementation popular in metabolomics; `glmnet` for elastic-net logistic regression as a complementary high-dimensional classifier.

8.27.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.27.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.28 Partial Least Squares Regression

8.28.1 Introduction

Partial least squares (PLS) regression is a latent-component method designed for the regime where predictors are numerous, correlated, and possibly outnumber observations — exactly the situation in which ordinary least squares fails. It extracts components from the predictor matrix that are simultaneously high-variance and strongly correlated with the outcome, then uses those components as the regression basis. Chemometrics and metabolomics use PLS as their default regression workhorse, and in bioinformatics it competes with elastic net and principal-components regression for high-dimensional prediction tasks.

8.28.2 Prerequisites

A working understanding of principal-components regression, multiple linear regression, cross-validation, and the trade-off between variance explanation and prediction.

8.28.3 Theory

PCA finds orthogonal directions in X that maximise variance of X. PLS finds directions that maximise *covariance* with Y — the latent components are simultaneously high-variance in X and predictive of Y. With a components, the model is

$$\hat{Y} = T_a Q_a^\top, \qquad T_a = X W_a,$$

where W_a are the PLS weights and T_a the resulting latent scores. The number of components is chosen by cross-validation, typically minimising root-mean-square error of prediction (RMSEP). PLS regression coefficients in the original predictor space are recovered as $\hat{\beta}_{\text{PLS}} = W_a (P_a^\top W_a)^{-1} Q_a^\top$.

8.28.4 Assumptions

Linear relationships between latent components and the outcome, independent observations, and (for inference) homoscedastic Normal residuals. PLS itself is fairly assumption-light for prediction; cross-validated RMSEP is the primary quality criterion.

8.28.5 R Implementation

```r
library(pls)

data(mtcars)
fit ← plsr(mpg ~ ., data = mtcars, validation = "CV", ncomp = 5)
summary(fit)
validationplot(fit, val.type = "RMSEP")

# Use the best number of components
n_comp ← 2
pred ← predict(fit, ncomp = n_comp)
```

8.28.6 Output & Results

`summary(fit)` reports cumulative explained variance in X and in Y for each component count, plus cross-validated RMSEP. `validationplot()` shows the RMSEP curve against component count, with a clear minimum at the optimal number of components.

8.28.7 Interpretation

A reporting sentence: "PLS regression with two latent components yielded a cross-validated RMSEP of 2.3 mpg on the 32-row `mtcars` dataset; the two components captured 85 % of variance in the predictors and 92 % of variance in the outcome, after which additional components gave diminishing returns." Report both X- and Y-variance explained; PLS optimises the latter, but readers familiar with PCA may expect the former.

8.28.8 Practical Tips

- Always use cross-validation to choose the number of components; in-sample fit is monotonically better with more components and selects too many.
- For classification, PLS-DA (`mixOmics::plsda()`) dummy-codes the outcome and applies PLS regression; it remains popular in metabolomics.
- Sparse PLS (`mixOmics::spls()`) adds variable selection at each component, useful when interpretability of individual predictors matters.
- For truly wide data ($p \gg n$), PLS and sparse PLS are core tools alongside ridge, lasso, and elastic net; cross-validate all four for benchmarking.

- Interpret latent components cautiously; they are linear combinations of all (or many) predictors and rarely have direct biological interpretations.
- For non-linear extensions, kernel PLS and PLS with spline-transformed predictors generalise to non-linear response surfaces.

8.28.9 R Packages Used

`pls` for `plsr()`, `mvr()`, and the standard cross-validation framework; `mixOmics` for PLS-DA, sparse PLS, and multi-block extensions popular in omics; `caret::train()` for unified cross-validated comparison against ridge, lasso, and elastic-net regression; `plsmod` for tidymodels-flavoured PLS interfaces.

8.28.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.28.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.29 Principal Component Analysis

8.29.1 Introduction

Principal component analysis (PCA) takes a data matrix with many correlated variables and re-expresses it in a smaller number of uncorrelated components that capture most of the variance. It is used for exploratory visualisation, for compression, as a preprocessing step for regression and classification, and as a quality-control tool in high-dimensional experiments such as RNA-seq. It is not a model for the data; it is a change of basis.

8.29.2 Prerequisites

The reader should be familiar with vectors, variance, covariance, and correlation. Some understanding of eigenvalues and eigenvectors helps but is not essential.

8.29.3 Theory

Given a data matrix $\mathbf{X}$ with n rows (observations) and p columns (variables), PCA finds orthogonal directions in the p-dimensional space of variables that successively maximise the projected variance. The first principal component (PC1) is the direction of greatest

variance; PC2 is the direction of greatest variance orthogonal to PC1; and so on up to $\min(n-1, p)$.

Algebraically, if $\mathbf{X}$ is centred (each column has mean zero), PCA is the eigendecomposition of the covariance matrix $\mathbf{S} = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$:

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top,$$

where the columns of $\mathbf{V}$ are the principal-component directions (loadings), and $\mathbf{\Lambda}$ is the diagonal matrix of variances along each direction (eigenvalues). The scores – coordinates of the observations in the new basis – are $\mathbf{Z} = \mathbf{X}\mathbf{V}$.

A practical issue: if the variables are on different scales, the variance-maximisation criterion is dominated by whichever variable happens to have the biggest numeric range. For heterogeneous data, PCA should be performed on the correlation matrix instead – equivalent to standardising each variable to unit variance before the decomposition.

8.29.4 Assumptions

PCA has no formal probabilistic assumptions, but its interpretation rests on:

1. The relationships of interest are linear in the variables.
2. Variance is a reasonable proxy for "importance".
3. The first few components contain most of the useful signal.

The first is violated when relationships are non-linear; alternative methods (kernel PCA, UMAP) may be needed. The second is violated when small-variance components encode the scientific signal (for example, subtle gene regulators in a bulk RNA-seq study).

8.29.5 R Implementation

```r
library(FactoMineR)
library(factoextra)

data(iris)
X ← iris[, 1:4]

pca ← PCA(X, scale.unit = TRUE, graph = FALSE)

fviz_eig(pca, addlabels = TRUE)
fviz_pca_biplot(pca,
                habillage = iris$Species,
                palette = "Set2",
                addEllipses = TRUE,
                label = "var")

summary(pca)
```

`PCA()` from `FactoMineR` handles centring and scaling by default. `fviz_eig()` draws the scree plot; `fviz_pca_biplot()` draws the biplot with observations coloured by species and variable-loading arrows overlaid. `summary()` produces a textual summary including variance explained, variable loadings, and contribution percentages.

8.29.6 Output & Results

For the `iris` data, the first component captures approximately 73% of the variance, the second another 23%; together the first two components retain 96% of the information. The biplot shows that the three species separate cleanly along PC1, which is heavily loaded on petal length, petal width, and sepal length.

8.29.7 Interpretation

In a manuscript, a PCA is usually reported in two pieces: the proportion of variance explained by the first k components, and the biological or physical interpretation of each component's loadings. "PC1 (73% of variance) is a size component, loading positively on all length measurements; PC2 (23%) contrasts sepal width against the other measurements and separates *I. setosa* from the other two species."

8.29.8 Practical Tips

- Always standardise (`scale.unit = TRUE`) unless all variables share the same units and you have a reason to preserve absolute-scale differences.
- Inspect the scree plot and keep components up to an elbow, or enough for 80-90% cumulative variance, or until an interpretable pattern emerges. There is no universal rule.
- The sign of each component is arbitrary (eigenvectors are unique up to sign), so "PC1 goes up" and "PC1 goes down" are equivalent representations; do not over-interpret the direction.
- For high-dimensional sparse data (e.g., scRNA-seq), use randomised SVD (`irlba`, `rsvd`) or the PCA implementations in `Seurat` or `scran` rather than dense eigendecomposition.
- PCA is sensitive to outliers; robust variants (ROBPCA, projection pursuit PCA) are available in the `rrcov` package.

8.29.9 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.29.10 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.30 Quadratic Discriminant Analysis

8.30.1 Introduction

Quadratic discriminant analysis (QDA) extends linear discriminant analysis (LDA) by allowing each class to have its own covariance matrix instead of pooling across classes. The relaxation transforms the decision boundary from a hyperplane (LDA) into a conic section (ellipse, parabola, or hyperbola) and gives the classifier flexibility to handle classes whose spread differs in shape or orientation. The cost is parameter count: K separate covariance matrices instead of one, which means more training data is required to estimate them precisely.

8.30.2 Prerequisites

A working understanding of linear discriminant analysis, multivariate Normal distributions, and the role of class covariance matrices in shaping decision boundaries.

8.30.3 Theory

The QDA discriminant for class k is

$$\delta_k(x) = -\tfrac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k) - \tfrac{1}{2}\log|\Sigma_k| + \log \pi_k,$$

where μ_k, Σ_k, π_k are the class mean, class covariance, and class prior probability respectively. A test observation is assigned to the class with the largest $\delta_k(x)$. The quadratic term in x is what produces non-linear decision boundaries; pooling covariances across classes (as LDA does) eliminates the quadratic term and recovers linearity.

8.30.4 Assumptions

Multivariate Normal within each class, with possibly different covariance matrices across classes; independent observations; sufficient observations per class to estimate Σ_k stably (rule of thumb: at least 5–10 observations per dimension per class).

8.30.5 R Implementation

```r
library(MASS)

set.seed(2026)
train_idx <- sample(nrow(iris), 100)
train <- iris[train_idx, ]; test <- iris[-train_idx, ]

fit_qda <- qda(Species ~ ., data = train)
pred <- predict(fit_qda, test)
table(pred$class, test$Species)
```

8.30.6 Output & Results

`qda()` returns class priors, class means, and class-specific group counts. `predict()` provides class labels and posterior probabilities. The confusion matrix (`table(pred$class, test$Species)`) summarises classification accuracy on the held-out fold.

8.30.7 Interpretation

A reporting sentence: "QDA classified the iris test set with 94 % accuracy (47/50 correct); LDA reached 98 % on the same split, suggesting the equal-covariance assumption was reasonable here and the additional flexibility of QDA is not warranted." Compare QDA against LDA whenever class covariances are not obviously different; the simpler model often wins.

8.30.8 Practical Tips

- Prefer QDA when within-class covariance structures clearly differ in scale or orientation; if pooled covariance is a good approximation, LDA generalises better.
- QDA requires more training data than LDA because of the per-class covariance estimation; in high dimensions or with small classes, LDA or regularised discriminant analysis (RDA) is more reliable.
- RDA (`klaR::rda`) interpolates between LDA and QDA via shrinkage parameters; cross-validate the shrinkage to find the data's preferred point on the spectrum.
- QDA decision boundaries are conic — ellipses, parabolas, or hyperbolas; visualise on a 2D projection (`klaR::partimat`) to see the curvature.
- For high-dimensional or non-Gaussian data, compare against flexible classifiers (random forests, SVMs with RBF kernels); QDA's quadratic boundary may not match the true structure.
- Diagnose the equal-covariance assumption with Box's M test (`biotools::boxM`); a significant test motivates QDA over LDA, but the test is itself sensitive to non-Normality.

8.30.9 R Packages Used

`MASS` for `qda()` and `lda()`; `klaR` for `rda()` (regularised discriminant analysis) and `partimat()` for partition diagnostics; `caret` for cross-validated comparison across discriminant variants; `biotools::boxM()` for Box's M test of homogeneity of covariance.

8.30.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.30.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.31 Structural Equation Models

8.31.1 Introduction

Structural equation modelling (SEM) is the unified framework that integrates measurement (how observed items reflect underlying latent constructs) and structure (how latent constructs relate to each other) into a single estimation problem. It generalises confirmatory factor analysis (measurement only), path analysis (regressions among observed variables only), and multivariate regression. Modern psychometrics, organisational research, and increasingly clinical-outcomes work depend on SEM as the primary analytical tool because it isolates relationships among true constructs from those distorted by measurement error.

8.31.2 Prerequisites

A working understanding of confirmatory factor analysis, multiple regression, latent-variable thinking, and basic familiarity with `lavaan` syntax.

8.31.3 Theory

SEM estimates measurement and structural parameters jointly by maximum likelihood (or robust ML, weighted least squares for ordinal indicators). Components in `lavaan` syntax:

- **Measurement model** (`=~`): latent factor defined by its observed indicators.
- **Structural regressions** (`~`): latent-on-latent or latent-on-observed regressions.
- **Variances / covariances** (`~~`): residual correlations or factor covariances.
- **Defined parameters** (`:=`): derived quantities such as indirect effects.

Standard fit indices (CFI, TLI, RMSEA, SRMR) parallel those of CFA. Indirect effects and confidence intervals are computed at the level of MCMC or bootstrap draws.

8.31.4 Assumptions

Correct specification of both the measurement and structural components, multivariate Normal continuous indicators (or robust estimators for non-Normal data), and sufficient sample size — at least 200 observations as a rough rule, more for complex models.

8.31.5 R Implementation

```r
library(lavaan); library(semPlot)

model ← '
  # Measurement
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

  # Structural
  dem60 ~ ind60
  dem65 ~ ind60 + dem60

  # Residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'
fit ← sem(model, data = PoliticalDemocracy)
summary(fit, standardized = TRUE, fit.measures = TRUE)

semPaths(fit, "std", layout = "tree2")
```

8.31.6 Output & Results

`summary(fit, standardized = TRUE)` reports unstandardised and standardised coefficients, factor loadings, residual covariances, and the full set of fit indices. `semPaths()` renders a path diagram with arrow widths proportional to standardised coefficients.

8.31.7 Interpretation

A reporting sentence: “The structural equation model fit the Political Democracy data well (CFI 0.99, RMSEA 0.04 [90 % CI 0.02 to 0.06], SRMR 0.05); industrialisation in 1960 predicted both 1960 and 1965 democracy ($\beta = 0.45$ and 0.39 respectively), and 1960 democracy strongly predicted 1965 democracy ($\beta = 0.84$, $p < 0.001$).” Always report standardised coefficients alongside fit indices.

8.31.8 Practical Tips

- Check the measurement model in isolation (run a CFA) before adding structural paths; weak factors produce uninterpretable structural relationships.
- Use `modificationIndices()` cautiously; data-driven modifications blur the confirmatory status of the analysis and inflate Type I error.
- For mediation analysis, define indirect effects with `:=` and request bootstrap CIs (`sem(..., se = "bootstrap")`); the indirect effect is a product of coefficients with non-Normal sampling distribution.
- For ordinal indicators, switch to weighted least squares (`estimator = "WLSMV"`); ordinary ML on Likert items violates Normality assumptions.

- Very complex models need very large samples; if $n < 5$ per estimated parameter, results are unstable and a simpler path model may be more robust.
- For cross-cultural or multi-group comparisons, fit measurement-invariance hierarchies (configural, metric, scalar) before comparing latent means across groups.

8.31.9 R Packages Used

`lavaan` for `sem()`, `cfa()`, `growth()`, and `lavInspect()`; `semPlot` for path diagrams; `semTools` for measurement-invariance tests, reliability coefficients, and modification heuristics; `blavaan` for Bayesian SEM that handles small samples and complex priors.

8.31.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

8.31.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.32 UMAP and t-SNE: Overview

8.32.1 Introduction

UMAP (uniform manifold approximation and projection) and t-SNE (t-distributed stochastic neighbour embedding) are the dominant non-linear dimensionality-reduction techniques for visualising very high-dimensional data — single-cell RNA sequencing, large image embeddings, language-model representations. They produce striking 2D or 3D scatter plots in which similar observations cluster together, and they are now near-default visualisations in many computational-biology and machine-learning pipelines. They also have well-documented pitfalls, and learning what they do *not* show is as important as learning to apply them.

8.32.2 Prerequisites

A working understanding of principal components analysis, distance metrics, and the trade-off between preserving local and global structure in a low-dimensional embedding.

8.32.3 Theory

Both methods optimise a 2D or 3D embedding such that pairwise similarities computed in the original high-dimensional space are preserved as well as possible.

- **t-SNE** computes Gaussian-weighted neighbour probabilities in the high-dimensional space, t-distributed neighbour probabilities in the low-dimensional space, and minimises the KL divergence between them. The Student-t distribution in low dimensions prevents the "crowding problem" (compressing many high-dimensional neighbours into a small low-dimensional area).
- **UMAP** constructs a fuzzy topological representation of the high-dimensional data using local distance scaling, then optimises a similar fuzzy structure in low dimensions via a cross-entropy loss. UMAP is faster, scales better to large n, and is often considered to preserve global structure better than t-SNE.

Both algorithms are stochastic; setting the random seed is required for reproducible figures.

8.32.4 Assumptions

A similarity-preserving 2D / 3D embedding is informative for the inferential question. Both algorithms assume meaningful local structure exists in the data; they do not assume Gaussianity, linearity, or any specific distance metric (UMAP's metric is configurable).

8.32.5 R Implementation

```r
library(umap); library(Rtsne)

set.seed(2026)
X ← as.matrix(iris[, 1:4])

# UMAP
u ← umap(X)
plot(u$layout, col = iris$Species, pch = 16,
     main = "UMAP of iris")

# t-SNE
t ← Rtsne(X, perplexity = 15, check_duplicates = FALSE)
plot(t$Y, col = iris$Species, pch = 16,
     main = "t-SNE of iris")
```

8.32.6 Output & Results

Both methods produce 2D coordinates per observation. UMAP and t-SNE both separate the three iris species clearly; differences in the layout (relative cluster positions, cluster sizes) reflect algorithm-specific choices rather than data-driven structure.

8.32.7 Interpretation

A reporting sentence: “UMAP and t-SNE embeddings of the four iris measurements both separate the three species, with setosa well-isolated and versicolor / virginica close but distinguishable; UMAP preserves the inter-cluster gap more faithfully than t-SNE, consistent with its better global-structure properties.” Always describe the algorithm and key hyperparameters in the figure caption; readers should not have to guess.

8.32.8 Practical Tips

- Treat distances in the embedding as relative neighbourhoods only; absolute distances and cluster *sizes* are not meaningful, despite how the visualisation looks.
- t-SNE’s `perplexity` parameter (typically 5–50) sets the effective neighbourhood scale; vary it across at least three values to confirm the structure is robust.
- UMAP’s `n_neighbors` and `min_dist` control the same trade-offs; defaults work for most data but should be tuned for very small or very large n.
- For very high-dimensional input, run PCA first to reduce noise to the top 50 components, then UMAP/t-SNE on the PCA scores; this is the standard pipeline in single-cell analysis.
- Cluster sizes in the embedding are meaningless; do not interpret a “small cluster” or “large cluster” as biologically real.
- Reproducibility requires a fixed seed *and* fixed software versions; UMAP results in particular can shift between releases.

8.32.9 R Packages Used

`umap` for the canonical UMAP implementation; `uwot` for a faster UMAP variant with more options; `Rtsne` for the standard t-SNE; `M3C` and `embed` for tidymodels-flavoured wrappers; `dimRed` for a unified interface across UMAP, t-SNE, Isomap, and other non-linear embeddings.

8.32.10 For Reviewers

Use the shared reviewer checklist, together with this method’s assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript’s application of the method.

8.32.11 See also — labs in this chapter

- PCA, factor analysis, CCA, LDA
- Clustering: k-means, hierarchical, model-based
- UMAP and t-SNE

8.33 Week 1, Session 4 — Clustering: k-means, hierarchical, model-based

Workflow labs use the variant template: Goal → Approach → Execution → Check → Report.

8.33.1 Learning objectives

- Fit k-means and hierarchical clusterings, and choose k by elbow and silhouette.
- Fit a model-based clustering with `mclust` and select the number of components by BIC.
- Explain why "number of clusters" is often not the right question.

8.33.2 Prerequisites

PCA; distance and linkage.

8.33.3 Background

Clustering is a search for structure in unlabelled data, and unlike classification it has no gold standard. Any algorithm will return a partition if asked to, and the partitions it returns depend on assumptions about what a cluster is. K-means assumes roughly spherical, equal-variance clusters and needs k in advance. Hierarchical clustering produces a nested sequence of partitions at every k, which you cut at a level you choose. Model-based clustering fits a Gaussian mixture and selects k by information criterion, which at least makes the assumption explicit.

In biomedicine, the honest version of "how many clusters?" is usually "is there more than one?" A good clustering lab focuses on stability and interpretability of the partition rather than on the number returned.

When you report a clustering, report the number of clusters, the criterion used to select it, a visualisation in a 2-D projection (PCA or UMAP), and the size and key features of each cluster. Do not report p-values for cluster labels against the data that produced the labels; that circularity is a classic error.

8.33.4 Setup

```
library(tidyverse)
library(mclust)
library(cluster)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

8.33.5 1. Goal

Cluster a simulated dataset with three known groups using three algorithms, and see which recovers the structure.

8.33.6 2. Approach

Three Gaussian blobs in 5 dimensions with different sizes.

```
make_blob ← function(n, mu, sd = 1) matrix(rnorm(n * 5, 0, sd), n, 5) +
  rep(mu, each = n)
X ← rbind(
  make_blob(60, c(0, 0, 0, 0, 0)),
  make_blob(40, c(3, 0, 0, 0, 0)),
  make_blob(30, c(0, 3, 0, 0, 0))
)
true_lbl ← rep(c("A", "B", "C"), c(60, 40, 30))

pc ← prcomp(X)$x[, 1:2]
tibble(PC1 = pc[, 1], PC2 = pc[, 2], group = true_lbl) ▷
  ggplot(aes(PC1, PC2, colour = group)) + geom_point(alpha = 0.7)
```

8.33.7 3. Execution

K-means with k sweep; silhouette.

```
wss ← sapply(2:8, function(k) kmeans(X, centers = k, nstart = 10)$tot.withinss)
sil ← sapply(2:8, function(k) {
  km ← kmeans(X, centers = k, nstart = 10)
  mean(silhouette(km$cluster, dist(X))[, 3])
})
tibble(k = 2:8, wss = wss, sil = sil) ▷
  pivot_longer(-k) ▷
  ggplot(aes(k, value)) + geom_line() + geom_point() +
  facet_wrap(~ name, scales = "free_y")
```

Hierarchical with Ward linkage.

```
hc ← hclust(dist(X), method = "ward.D2")
plot(hc, labels = FALSE, main = "Ward hierarchical clustering")
```

8.33.8 4. Check

Model-based clustering by BIC.

```
mc ← Mclust(X, G = 1:6)
summary(mc)
```

8.33.9 5. Report

> On three simulated Gaussian blobs (n = 130 in R^5), model-based clustering selected G = `mc$G` by BIC; k-means with k = `which.max(sil) + 1` maximised mean silhouette (`round(max(sil), 2)`). Hierarchical clustering with Ward linkage produced a dendrogram consistent with three primary branches.

When the generating model is actually Gaussian, `mclust` tends to win. On real data, no single method is best; the test is whether the partition is stable under resampling and interpretable in terms of the features.

8.33.10 Common pitfalls

- Treating k-means output as ground truth; it is an optimisation answer given k.
- Running hierarchical clustering on a very large dataset without noting the $O(n^2)$ memory cost.
- Testing post-hoc differences between clusters with the same data that produced them.

8.33.11 Further reading

- Fraley C, Raftery AE (2002), *Model-based clustering, discriminant analysis, and density estimation*.
- Hastie, Tibshirani, Friedman, *ESL*, §14.3.

8.33.12 Session info

```
sessionInfo()
```

8.33.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

8.34 Week 1, Session 3 — PCA, factor analysis, CCA, LDA

Workflow labs use the variant template: Goal → Approach → Execution → Check → Report.

8.34.1 Learning objectives

- Compute a PCA, read a biplot, and explain loadings and scores without resorting to magical reasoning about "directions of variance".
- Distinguish PCA, factor analysis, canonical correlation analysis, and linear discriminant analysis by the question each answers.
- Fit an LDA classifier and interpret the discriminant coefficients.

8.34.2 Prerequisites

Matrix algebra (eigendecomposition); basic multivariate intuition.

8.34.3 Background

Dimension reduction is often described as “finding structure”, but each of these methods has a concrete question behind it. PCA asks which directions in feature space have the largest variance; its axes are eigenvectors of the covariance matrix. Factor analysis asks which latent factors, plus unique variance per feature, reproduce the observed covariance; it is PCA’s statistical cousin with a proper error model. Canonical correlation analysis asks which linear combinations of two feature sets are maximally correlated. Linear discriminant analysis asks which direction separates known classes while collapsing within-class variance.

They share machinery — all four involve eigenvalues of something — but they are not interchangeable. Using PCA where LDA was called for will waste the label information; using LDA where PCA was called for will produce a single, class-dependent axis and nothing else.

When you report a PCA, report the variance explained by each of the first few components and a biplot with named observations and loadings. When you report an LDA, report the classification accuracy on held-out data and the direction of each discriminant, not just the confusion matrix.

8.34.4 Setup

```
library(tidyverse)
library(MASS)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

8.34.5 1. Goal

Use `iris` to illustrate PCA and LDA on the same data, compare their projections, and classify species with LDA.

8.34.6 2. Approach

Four morphometric variables, three species. PCA ignores the label; LDA uses it.

```
iris |>
  pivot_longer(-Species) |>
  ggplot(aes(value, fill = Species)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ name, scales = "free") +
  labs(x = NULL, y = "density")
```

8.34.7 3. Execution

PCA on the four numeric features, then LDA.

```r
pca <- prcomp(iris[, 1:4], scale. = TRUE)
summary(pca)$importance[, 1:3]

lda_fit <- lda(Species ~ ., data = iris)
lda_fit$scaling
```

```r
scores_pca <- as_tibble(pca$x[, 1:2]) |> mutate(Species = iris$Species)
scores_lda <- as_tibble(predict(lda_fit)$x) |> mutate(Species = iris$Species)
p1 <- ggplot(scores_pca, aes(PC1, PC2, colour = Species)) +
  geom_point(alpha = 0.7) + labs(title = "PCA (unsupervised)")
p2 <- ggplot(scores_lda, aes(LD1, LD2, colour = Species)) +
  geom_point(alpha = 0.7) + labs(title = "LDA (uses labels)")
gridExtra::grid.arrange(p1, p2, ncol = 2)
```

8.34.8 4. Check

Hold-out accuracy for LDA (simple 70/30 split).

```r
idx <- sample(nrow(iris), 0.7 * nrow(iris))
fit2 <- lda(Species ~ ., data = iris[idx, ])
pred <- predict(fit2, iris[-idx, ])$class
mean(pred == iris$Species[-idx])
```

8.34.9 5. Report

> On the *iris* data, the first two principal components explained `round(sum(summary(pca)$importance[2, 1:2]) * 100, 1)` % of the total variance. A linear discriminant classifier achieved hold-out accuracy of `round(mean(pred == iris$Species[-idx]) * 100, 1)` % on a 70/30 split.

In this low-dimensional problem, PCA already separates the species visually because the morphometric variables are correlated with species. In realistic biomedical data, that is rarely true: the supervised projection (LDA, or a classifier's decision function) is usually sharper.

8.34.10 Common pitfalls

- Reporting "PC1 is age" because age happens to correlate with it.
- Running LDA when classes are hugely imbalanced; use QDA or a proper classifier instead.
- Comparing variance-explained across datasets as if it were a quality score.

8.34.11 Further reading

- Jolliffe IT (2002), *Principal Component Analysis*, 2nd ed.
- Mardia, Kent, Bibby, *Multivariate Analysis*, chapters on LDA/CCA.

8.34.12 Session info

```
sessionInfo()
```

8.34.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

8.35 Week 1, Session 5 — UMAP and t-SNE

Workflow labs use the variant template: Goal → Approach → Execution → Check → Report.

8.35.1 Learning objectives

- Compute UMAP and t-SNE embeddings and describe what each optimisation is doing.
- Interpret global vs local structure in a UMAP/t-SNE plot and avoid over-reading cluster distances.
- Tune `n_neighbors` / `perplexity` and see the visual effect.

8.35.2 Prerequisites

PCA; basic notion of neighbour graphs.

8.35.3 Background

UMAP and t-SNE are nonlinear embedding methods widely used in single-cell and imaging biomedicine. Both build a neighbour graph in high dimensions and then try to match a lower-dimensional graph with similar neighbour structure. They are not clustering algorithms, and distances in the embedding are not Euclidean distances in the original space. The prominent visual clusters in a UMAP often exist in the data, but the distance between clusters is close to meaningless and the density inside a cluster is not comparable across clusters.

t-SNE emphasises local structure; UMAP, with default settings, is a little more faithful to global structure but can still distort. Both have knobs — `perplexity` for t-SNE, `n_neighbors` and `min_dist` for UMAP — that change the resulting picture substantially. A reviewer who sees only one embedding has no way to tell if the picture is a stable feature of the data or an artefact of the settings.

Treat UMAP/t-SNE as a visualisation of neighbourhood structure, not as a geometric truth. Do not run statistical tests on the coordinates. Do not claim a trajectory unless you used a trajectory method.

8.35.4 Setup

```
library(tidyverse)
library(uwot)
library(Rtsne)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

8.35.5 1. Goal

Embed a simulated high-dimensional dataset with three blobs into two dimensions using UMAP and t-SNE, and compare against PCA.

8.35.6 2. Approach

```
make_blob ← function(n, mu, sd = 1, d = 20) matrix(rnorm(n * d, 0, sd), n, d) +
  rep(mu, each = n)
X ← rbind(
  make_blob(100, c(0, 0, rep(0, 18))),
  make_blob(100, c(4, 0, rep(0, 18))),
  make_blob(100, c(0, 4, rep(0, 18)))
)
lbl ← rep(c("A", "B", "C"), each = 100)
```

8.35.7 3. Execution

```
emb_pca  ← prcomp(X)$x[, 1:2]
emb_umap ← umap(X, n_neighbors = 15, min_dist = 0.1)
emb_tsne ← Rtsne(X, perplexity = 30, check_duplicates = FALSE)$Y

df ← bind_rows(
  tibble(method = "PCA",  x = emb_pca[, 1],  y = emb_pca[, 2],  lbl = lbl),
  tibble(method = "UMAP", x = emb_umap[, 1], y = emb_umap[, 2], lbl = lbl),
  tibble(method = "tSNE", x = emb_tsne[, 1], y = emb_tsne[, 2], lbl = lbl)
)
ggplot(df, aes(x, y, colour = lbl)) +
  geom_point(alpha = 0.7, size = 0.8) +
  facet_wrap(~ method, scales = "free") +
  labs(x = NULL, y = NULL)
```

8.35.8 4. Check

Sensitivity to `n_neighbors`.

```
do_umap ← function(k) {
  e ← umap(X, n_neighbors = k, min_dist = 0.1)
  tibble(x = e[, 1], y = e[, 2], lbl = lbl, k = paste0("nn=", k))
}
bind_rows(do_umap(5), do_umap(15), do_umap(50)) ▷
  ggplot(aes(x, y, colour = lbl)) + geom_point(alpha = 0.7, size = 0.8) +
  facet_wrap(~ k, scales = "free")
```

8.35.9 5. Report

> Both UMAP and t-SNE recovered the three simulated blobs as distinct visual clusters; PCA separated them more weakly because the signal lay along two of 20 dimensions. UMAP embeddings with `n_neighbors` $\in \{5, 15, 50\}$ all identified the same three groups but differed substantially in their relative layout.

The message is not which method is best, but that the embedding is best used as a visual companion, with another method — clustering, classification, or differential analysis — doing the statistical work.

8.35.10 Common pitfalls

- Reading distances between clusters off a UMAP as if they were biological distances.
- Running one embedding with defaults and treating the result as the data.
- Using UMAP coordinates as features in a downstream supervised model.

8.35.11 Further reading

- McInnes L, Healy J, Melville J (2018), *UMAP: Uniform manifold approximation and projection*.
- Kobak D, Berens P (2019), *The art of using t-SNE for single-cell transcriptomics*.

8.35.12 Session info

```
sessionInfo()
```

8.35.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)