From Distances to Maps: Multidimensional Scaling

Unit 11 — Data Visualization

Livio Finos, University of Padova

Contents

Why work with distances? **2**

Learning goals . . . 2

Objects, variables and distance matrices **3**

Metric or general dissimilarity? . . . 3

Four useful ways to compare objects **3**

Euclidean distance: straight-line difference . . . 3

Manhattan distance: total coordinate-wise change . . . 4

Jaccard distance: presence and absence of features . . . 5

A Euclidean distance needs standardisation when units differ **6**

A two-dimensional scale-change example . . . 6

Worked example: Dutch dune-meadow communities **9**

A first description of the community data . . . 11

What does each distance discriminate? . . . 13

Bray–Curtis distance: composition with abundances . . . 13

Hellinger transformation: composition in Euclidean geometry . . . 14

Bray–Curtis versus standardised Euclidean distance . . . 14

Scaling is also a modelling choice . . . 16

Classical MDS: turn distances into coordinates **17**

Spectral decomposition, one matrix at a time . . . 17

Step 1 — distances and squared distances . . . 18

Step 2 — centre the squared distances . . . 19

Step 3 — spectral decomposition of B . . . 20

Step 4 — recover coordinates . . . 21

The same objects in a reusable function . . . 23

MDS map of the communities 24
Why MDS axes have no inherent interpretation 24

Choosing the number of dimensions **26**

Reading MDS eigenvalues and fit **28**
What changes under Hellinger–Euclidean distance? 31

Classical MDS and non-metric MDS **32**

Practical workflow **33**

Take-home messages **33**

Why work with distances?

Many datasets are naturally represented as a rectangular table: one row per person, site, product or image; one column per measured variable. In other problems, the scientifically meaningful starting point is not a variable but a **comparison between objects**. Examples include:

- ecological sites compared through their species composition;
- products compared through consumer ratings or ingredient profiles;
- patients compared through a set of symptoms or biomarkers;
- cities compared through travel time rather than straight-line geography.

In all these cases we can ask the same question:

> Which objects are close, which are far apart, and can that structure be shown in a useful low-dimensional map?

This unit introduces distances and **classical multidimensional scaling (MDS)**. It deliberately concerns non-textual data. Text documents will be treated later, when we can make the preprocessing and weighting choices explicit.

Learning goals

By the end of the unit, you should be able to:

- distinguish an object-by-variable table from an object-by-object distance matrix;
- state the properties of a metric and recognise a dissimilarity that is not one;
- calculate and compare Euclidean, Manhattan, Jaccard and Bray–Curtis distances;
- explain why standardisation and the choice of distance are modelling decisions;
- construct and interpret a classical MDS configuration in R;
- connect the MDS construction to spectral decomposition;
- use eigenvalues to diagnose whether a distance is exactly Euclidean and how well a two-dimensional map represents it.

Prerequisites: vectors and matrices, Euclidean distance, centring, and the spectral decomposition of a real symmetric matrix. You do not need prior knowledge of clustering or ordination.

Objects, variables and distance matrices

Let $x_i = (x_{i1}, \ldots, x_{ip})$ denote the measurements for object i. After choosing a comparison rule d, we obtain the distance matrix

$$D = (d_{ij}), \qquad d_{ij} = d(x_i, x_j).$$

The matrix is square, symmetric and has zeros on its diagonal. It contains $n(n-1)/2$ distinct pairwise comparisons, not np original measurements. This distinction matters: the original variables may disappear after distances are computed, so their units and preprocessing choices must be documented.

```
points3 <- rbind(A = c(0, 0), B = c(3, 4), C = c(3, 0))
points3
```

```
#>   [,1] [,2]
#> A    0    0
#> B    3    4
#> C    3    0
```

```
round(as.matrix(dist(points3)), 2)
```

```
#>   A B C
#> A 0 5 3
#> B 5 0 4
#> C 3 4 0
```

For these coordinates, $d(A, B) = 5$ under Euclidean distance. The matrix says that A is farthest from B and that C lies between them in one sense, but it no longer records that the original variables were horizontal and vertical position.

Metric or general dissimilarity?

A **metric** satisfies, for every pair or triple of objects,

$$\begin{aligned}
&d(x_i, x_j) \geq 0 && \text{(non-negativity)},\\
&d(x_i, x_j) = 0 \Leftrightarrow x_i = x_j && \text{(identity)},\\
&d(x_i, x_j) = d(x_j, x_i) && \text{(symmetry)},\\
&d(x_i, x_k) \leq d(x_i, x_j) + d(x_j, x_k) && \text{(triangle inequality)}.
\end{aligned}$$

In applications, we also use broader **dissimilarities**. They may be useful even when one of these properties fails. For example, a directed travel time may fail symmetry; a correlation-based dissimilarity may not have a direct physical unit. The important question is not whether a formula is fashionable, but whether it expresses the scientific comparison we need.

Four useful ways to compare objects

Euclidean distance: straight-line difference

For quantitative variables on commensurate scales,

$$d_E(x_i, x_j) = \left\{ \sum_{k=1}^{p} (x_{ik} - x_{jk})^2 \right\}^{1/2}.$$

Euclidean distance gives large differences disproportionate influence because they are squared. It is appropriate when the variables define meaningful Cartesian coordinates, or when a straight-line geometry is substantively useful.

```
pair_index <- combn(seq_len(nrow(points3)), 2)
euclidean_calculations <- apply(pair_index, 2, function(index) {
  first <- points3[index[1], ]
  second <- points3[index[2], ]
  tibble(
    pair = paste(rownames(points3)[index], collapse = "--"),
    calculation = sprintf("sqrt((%g - %g)^2 + (%g - %g)^2)",
                          first[1], second[1], first[2], second[2]),
    distance = sqrt(sum((first - second)^2))
  )
}) |> bind_rows()
knitr::kable(euclidean_calculations, digits = 2,
             caption = "Coordinate-by-coordinate Euclidean calculations.")
```

Table 1: Coordinate-by-coordinate Euclidean calculations.

pair	calculation	distance
A–B	sqrt((0 - 3)^2 + (0 - 4)^2)	5
A–C	sqrt((0 - 3)^2 + (0 - 0)^2)	3
B–C	sqrt((3 - 3)^2 + (4 - 0)^2)	4

```
D_euclidean_points3 <- matrix(0, nrow(points3), nrow(points3),
                              dimnames = list(rownames(points3), rownames(points3)))
for (i in seq_len(nrow(points3) - 1)) {
  for (j in (i + 1):nrow(points3)) {
    D_euclidean_points3[i, j] <- D_euclidean_points3[j, i] <-
      sqrt(sum((points3[i, ] - points3[j, ])^2))
  }
}
knitr::kable(D_euclidean_points3, digits = 2,
             caption = "Euclidean distance matrix, assembled from the displayed calculations.")
```

Table 2: Euclidean distance matrix, assembled from the displayed calculations.

	A	B	C
A	0	5	3
B	5	0	4
C	3	4	0

Manhattan distance: total coordinate-wise change

The Manhattan (or L_1) distance is

$$d_M(x_i, x_j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}|.$$

It measures the total absolute difference and is less dominated by one very large coordinate difference than Euclidean distance. It can be motivated by movement along a grid, but it is also a general robust alternative for numerical profiles.

```
manhattan_calculations <- apply(pair_index, 2, function(index) {
  first <- points3[index[1], ]
  second <- points3[index[2], ]
  tibble(
    pair = paste(rownames(points3)[index], collapse = "--"),
    calculation = sprintf("abs(%g - %g) + abs(%g - %g)",
                          first[1], second[1], first[2], second[2]),
    distance = sum(abs(first - second))
  )
}) |> bind_rows()
knitr::kable(manhattan_calculations, digits = 2,
             caption = "Coordinate-by-coordinate Manhattan calculations.")
```

Table 3: Coordinate-by-coordinate Manhattan calculations.

pair	calculation	distance
A–B	abs(0 - 3) + abs(0 - 4)	7
A–C	abs(0 - 3) + abs(0 - 0)	3
B–C	abs(3 - 3) + abs(4 - 0)	4

```
D_manhattan_points3 <- matrix(0, nrow(points3), nrow(points3),
                              dimnames = list(rownames(points3), rownames(points3)))
for (i in seq_len(nrow(points3) - 1)) {
  for (j in (i + 1):nrow(points3)) {
    D_manhattan_points3[i, j] <- D_manhattan_points3[j, i] <-
      sum(abs(points3[i, ] - points3[j, ]))
  }
}
knitr::kable(D_manhattan_points3, digits = 2,
             caption = "Manhattan distance matrix, assembled from the displayed calculations.")
```

Table 4: Manhattan distance matrix, assembled from the displayed calculations.

	A	B	C
A	0	7	3
B	7	0	4
C	3	4	0

Jaccard distance: presence and absence of features

For binary data, shared absences often carry little information. If two sites both lack a rare species, that does not necessarily make them similar. Let a be the number of shared presences, and b, c the mismatches. Jaccard similarity and distance are

$$s_J = \frac{a}{a+b+c}, \qquad d_J = 1 - s_J.$$

```
jaccard_distance <- function(a, b) {
  shared_present <- sum(a == 1 & b == 1)
  present_either <- sum(a == 1 | b == 1)
  1 - shared_present / present_either
}

site_a <- c(1, 1, 0, 0, 0, 1)
site_b <- c(1, 0, 0, 1, 0, 1)
jaccard_distance(site_a, site_b)
```

```
#> [1] 0.5
```

Use Jaccard for occurrence data, tags or sets. It is not a natural choice when the original measurements are abundances and their magnitude is meaningful.

A Euclidean distance needs standardisation when units differ

Euclidean distance is often exactly the right comparison rule, but only after we decide what a one-unit change in each variable should mean. Consider the course dataset on memory, hippocampal volume and background measurements. Age is measured in years, intracranial volume (ICV) in a much larger physical unit, motion on a small scale, and memory on a score scale. A raw Euclidean distance would be driven mainly by variables with the largest numerical spread; that is a statement about units, not about scientific importance.

When the aim is to compare profiles by *relative position on each measured variable*, a common choice is the column-wise z-score

$$z_{ij} = \frac{x_{ij} - \bar{x}_{\cdot j}}{s_j}.$$

Euclidean distance on these z-scores gives each selected variable variance one. This does **not** make it universally better: it says that one standard deviation of age, ICV, motion, sleep, stress, memory or hippocampal volume deserves equal weight. That is a substantive choice. Identifiers and binary/categorical fields must not be silently treated as continuous measurements.

A two-dimensional scale-change example

The following four illustrative profiles have age in years and ICV in cubic centimetres. In the raw plot, profile B differs greatly from A in age but only slightly in ICV; profile C has almost the same age as A but a much larger ICV. Raw Euclidean distance calls B closer to A because the 200-unit ICV difference to C dominates. After z-scoring, C becomes closer because the age difference between A and B is large relative to the observed age variation.

```
profile_2d <- rbind(A = c(age = 20, ICV = 1500),
                    B = c(age = 50, ICV = 1510),
                    C = c(age = 21, ICV = 1700),
                    D = c(age = 35, ICV = 2500))
profile_2d_z <- scale(profile_2d)
```

```r
D_raw_2d <- as.matrix(dist(profile_2d))
D_z_2d <- as.matrix(dist(profile_2d_z))

comparison_to_A <- tibble(
  profile = c("B", "C"),
  raw_distance_to_A = D_raw_2d["A", c("B", "C")],
  z_scored_distance_to_A = D_z_2d["A", c("B", "C")]
)
knitr::kable(comparison_to_A, digits = 2,
             caption = "The nearest profile to A changes from B to C after z-scoring.")
```

Table 5: The nearest profile to A changes from B to C after z-scoring.

profile	raw_distance_to_A	z_scored_distance_to_A
B	31.62	2.13
C	200.00	0.43

```r
knitr::kable(D_raw_2d, digits = 1, caption = "Raw Euclidean distance matrix.")
```

Table 6: Raw Euclidean distance matrix.

	A	B	C	D
A	0.0	31.6	200.0	1000.1
B	31.6	0.0	192.2	990.1
C	200.0	192.2	0.0	800.1
D	1000.1	990.1	800.1	0.0

```r
knitr::kable(D_z_2d, digits = 2, caption = "Euclidean distance matrix after z-scoring.")
```

Table 7: Euclidean distance matrix after z-scoring.

	A	B	C	D
A	0.00	2.13	0.43	2.36
B	2.13	0.00	2.09	2.34
C	0.43	2.09	0.00	1.96
D	2.36	2.34	1.96	0.00

```r
old_par <- par(no.readonly = TRUE)
par(mfrow = c(1, 2), mar = c(4, 4, 3, 1))
plot(profile_2d[, "age"], profile_2d[, "ICV"], pch = 19, col = blue,
     xlab = "Age (years)", ylab = "ICV (cm^3)", main = "Raw coordinates")
segments(profile_2d["A", "age"], profile_2d["A", "ICV"],
         profile_2d["B", "age"], profile_2d["B", "ICV"], col = orange, lwd = 2)
segments(profile_2d["A", "age"], profile_2d["A", "ICV"],
         profile_2d["C", "age"], profile_2d["C", "ICV"], col = green, lwd = 2)
text(profile_2d[, "age"], profile_2d[, "ICV"], labels = rownames(profile_2d), pos = 3)
plot(profile_2d_z[, "age"], profile_2d_z[, "ICV"], pch = 19, col = blue,
     xlab = "Age (z-score)", ylab = "ICV (z-score)", main = "After z-scoring")
```

```
segments(profile_2d_z["A", "age"], profile_2d_z["A", "ICV"],
         profile_2d_z["B", "age"], profile_2d_z["B", "ICV"], col = orange, lwd = 2)
segments(profile_2d_z["A", "age"], profile_2d_z["A", "ICV"],
         profile_2d_z["C", "age"], profile_2d_z["C", "ICV"], col = green, lwd = 2)
text(profile_2d_z[, "age"], profile_2d_z[, "ICV"], labels = rownames(profile_2d_z), pos = 3)
legend("topleft", legend = c("A--B", "A--C"), col = c(orange, green), lwd = 2, bty = "n")
```

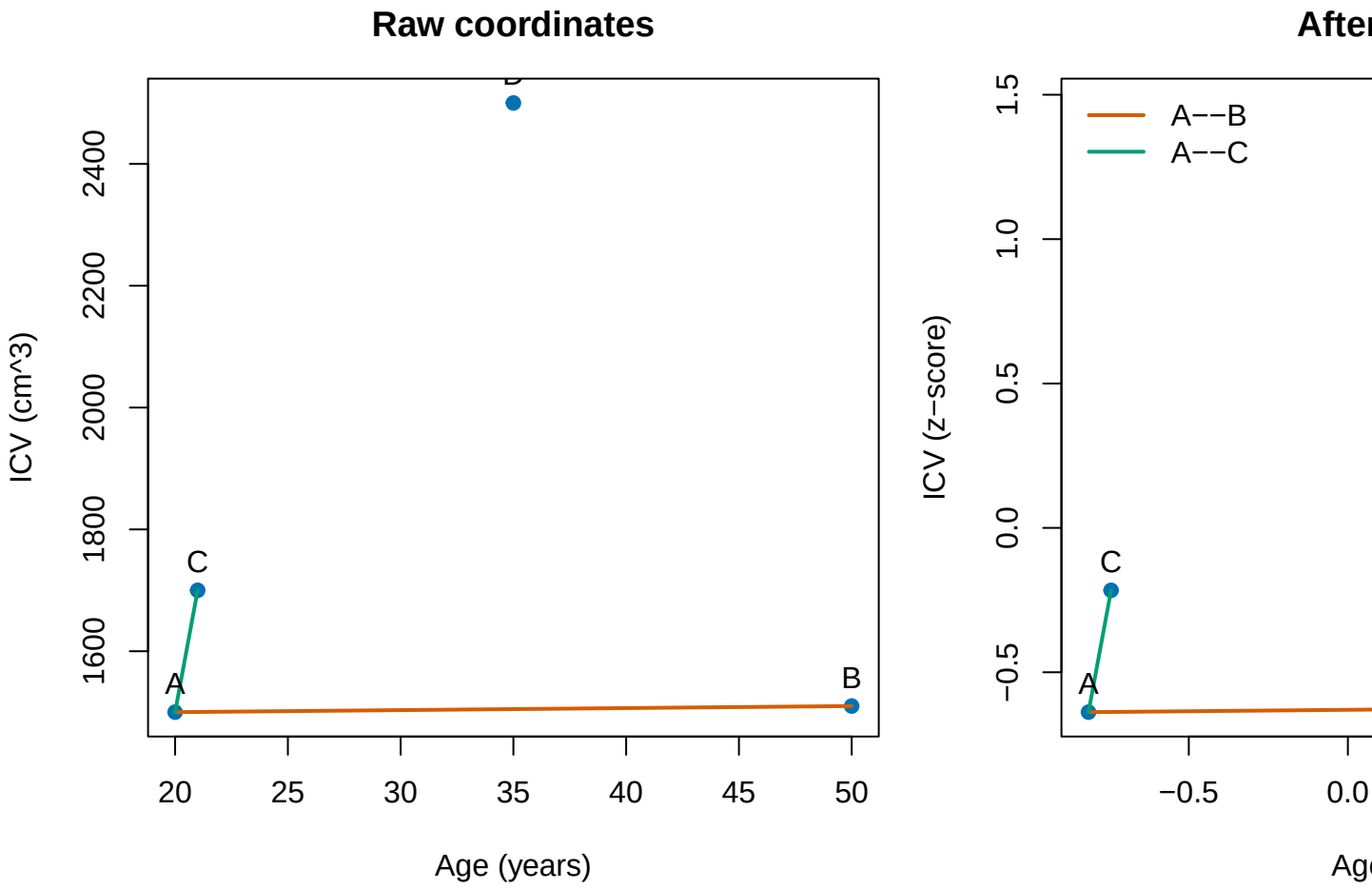


Figure 1: The nearest profile to A changes after both variables are put on their standard-deviation scale. Segments compare A–B (orange) and A–C (green).

```
par(old_par)
```

```
brain <- read_csv("data/hippocampus_memory_age_continuous.csv", show_col_types = FALSE)
brain_profile <- brain |> select(memory_score, hippocampal_volume, age, ICV,
                                 motion, sleep_hours, stress_score)

d_brain_raw <- dist(brain_profile)
d_brain_z <- dist(scale(brain_profile))

# Mean squared pairwise difference identifies which variables drive raw distance.
raw_contribution <- sapply(brain_profile, function(v) mean(outer(v, v, "-")^2))
contribution <- tibble(variable = names(raw_contribution), mean_squared_difference = raw_contribution)
```

```r
knitr::kable(arrange(contribution, desc(mean_squared_difference)), digits = 2,
             caption = "Raw Euclidean distance is dominated by variables with large numerical scales.")
```

Table 8: Raw Euclidean distance is dominated by variables with large numerical scales.

variable	mean_squared_difference
ICV	21092.30
age	596.82
memory_score	537.04
stress_score	237.14
sleep_hours	1.79
hippocampal_volume	0.97
motion	0.01

```r
comparison_brain <- tibble(raw = as.vector(d_brain_raw), z_scored = as.vector(d_brain_z))
ggplot(comparison_brain, aes(raw, z_scored)) +
  geom_point(colour = blue, alpha = .55) +
  geom_smooth(method = "loess", se = FALSE, colour = orange, linewidth = 1) +
  labs(title = "Raw and z-scored Euclidean distance answer different questions",
       x = "Raw Euclidean distance", y = "Euclidean distance after z-scoring")
```

Worked example: Dutch dune-meadow communities

We use the real `dune` data distributed with the R package `vegan`. It contains cover-class values for 30 plant species recorded at 20 Dutch dune-meadow sites. The companion metadata record soil A1-horizon thickness, moisture, management, land use and manure level. The observations come from the dune-meadow study reported by Jongman, ter Braak and van Tongeren (1987).

The course folder includes two redistributable CSV files extracted from that dataset: `ecological_communities.csv` contains the species-by-site table and `ecological_communities_metadata.csv` contains the site metadata. Species names are abbreviated as in the original teaching dataset. The cover classes are ordered abundance scores, not direct counts from equal-area sampling.

```r
community <- read_csv("data/ecological_communities.csv", show_col_types = FALSE)
site_metadata <- read_csv("data/ecological_communities_metadata.csv", show_col_types = FALSE)
abundance <- community |> select(-site) |> as.data.frame()
rownames(abundance) <- community$site
knitr::kable(head(site_metadata), caption = "First six rows of the site metadata. Full data are availabl
```

Table 9: First six rows of the site metadata. Full data are available in the CSV file.

site	A1	Moisture	Management	Use	Manure
site01	2.8	1	SF	Haypastu	4
site02	3.5	1	BF	Haypastu	2
site03	4.3	2	SF	Haypastu	4
site04	4.2	2	SF	Haypastu	4
site05	6.3	1	HF	Hayfield	2
site06	4.3	1	HF	Haypastu	2

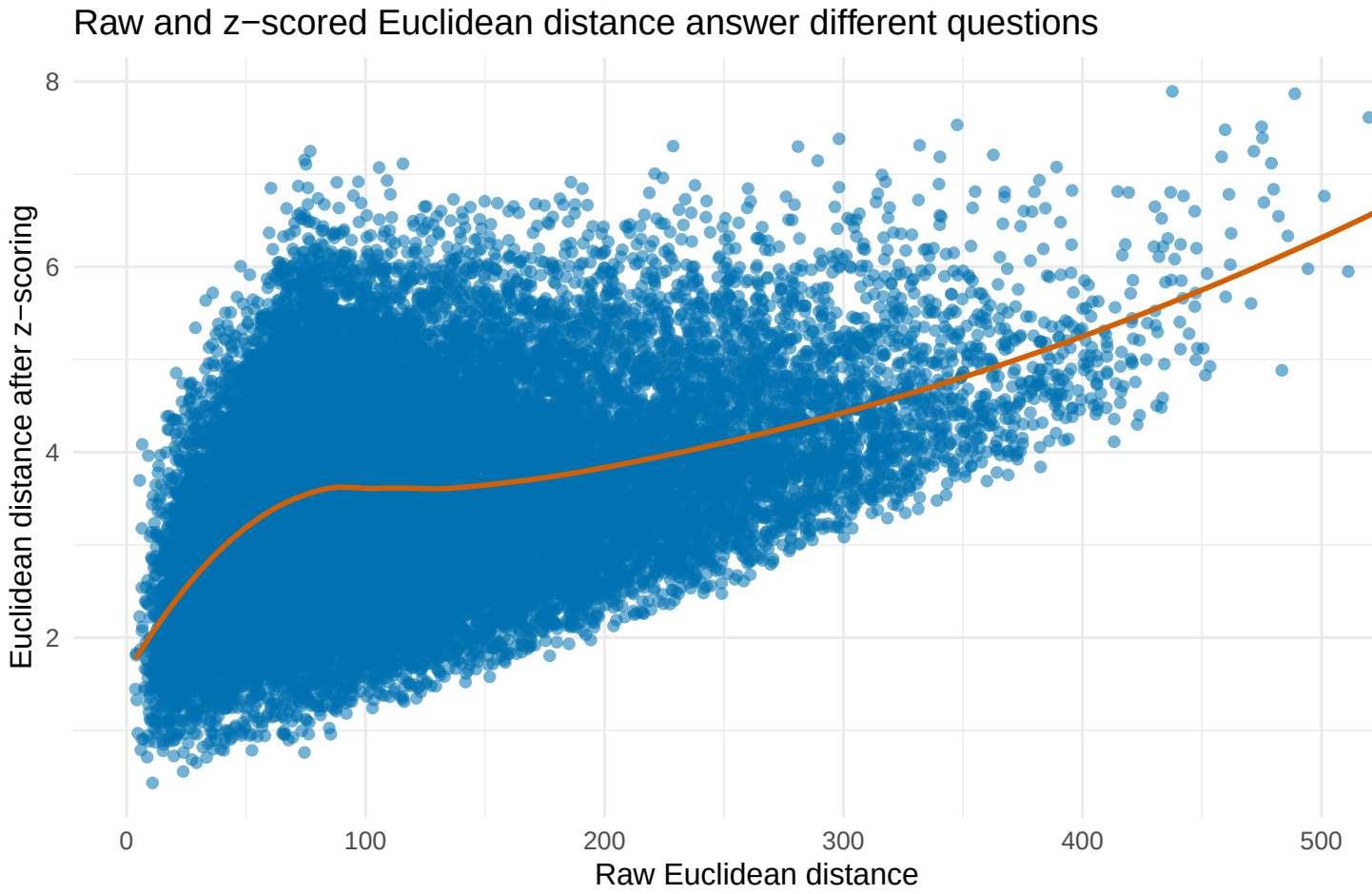


Figure 2: The two distances do not merely differ by a constant multiplier: standardisation can change which pairs of profiles are judged closest.

A first description of the community data

Each row is one meadow site and each of the 30 species columns is an ordinal cover-class score. Before computing any distances, ask how dense the site profiles are and whether the sites differ in their recorded management. The site total is a simple sum of cover scores, whereas species richness is the number of species with a non-zero score. Neither is a direct ecological abundance estimate, but both make the scale and sparsity of the input tangible.

```
site_description <- tibble(
  summary = c("Number of sites", "Number of species", "Site-total cover score: min / median / max",
              "Species richness: min / median / max"),
  value = c(
    nrow(abundance), ncol(abundance),
    sprintf("%s / %.1f / %s", min(rowSums(abundance)), median(rowSums(abundance)), max(rowSums(abundanc
    sprintf("%s / %.1f / %s", min(rowSums(abundance > 0)), median(rowSums(abundance > 0)), max(rowSums(a
  )
)
knitr::kable(site_description, caption = "Basic descriptive summary of the dune-meadow community matrix
```

Table 10: Basic descriptive summary of the dune-meadow community matrix.

summary	value
Number of sites	20
Number of species	30
Site-total cover score: min / median / max	15 / 34.0 / 48
Species richness: min / median / max	5 / 9.5 / 14

```
site_profiles <- tibble(
  site = rownames(abundance),
  total_cover_score = rowSums(abundance),
  species_richness = rowSums(abundance > 0)
) |> left_join(site_metadata |> select(site, Management), by = "site")
knitr::kable(count(site_profiles, Management, name = "sites"), caption = "Sites by recorded management
```

Table 11: Sites by recorded management category.

Management	sites
BF	3
HF	5
NM	6
SF	6

```
ggplot(site_profiles, aes(total_cover_score, species_richness, colour = Management, label = site)) +
  geom_point(size = 3) +
  geom_text(nudge_y = .35, show.legend = FALSE) +
  labs(title = "Density and richness of the 20 site profiles",
       x = "Sum of cover-class scores", y = "Number of species present")
```

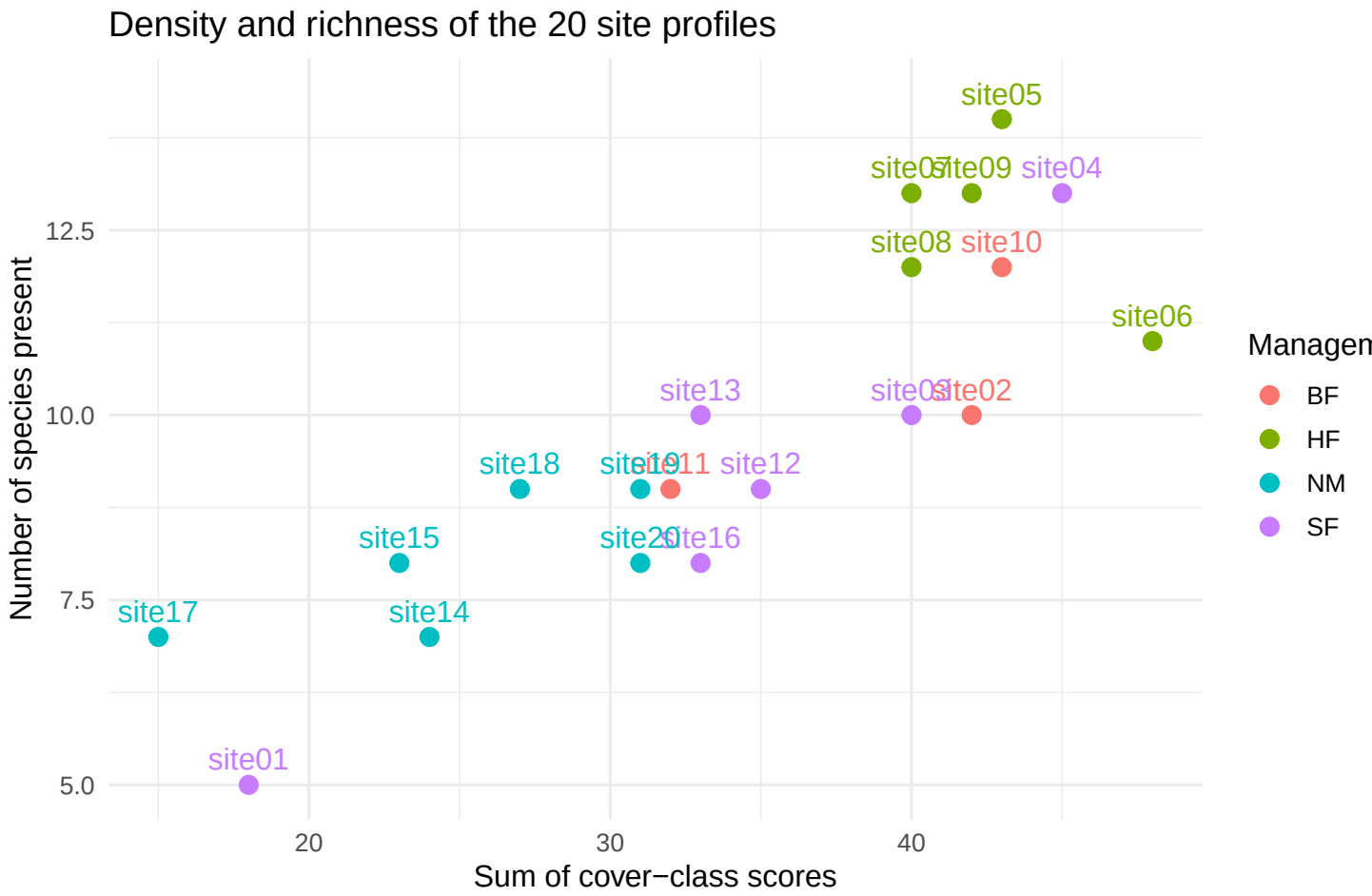


Figure 3: The sites differ both in total cover score and in the number of recorded species; these features are part of what a distance can respond to.

What does each distance discriminate?

The relevant question is not whether numerical values of two distances can be compared directly: they usually cannot. The question is whether a distance separates sites in a way that corresponds to a defensible notion of ecological difference.

Bray–Curtis compares the absolute difference in the species profiles with the combined abundance of the two sites. Its denominator is $\sum_k (x_{ik} + x_{jk})$, twice the average total score of the pair. Thus it creates a unitless *relative* discrepancy and prevents a uniformly larger abundance scale from mechanically creating arbitrarily large distances. It does not make totals irrelevant: two sites with the same composition but very different total abundance still have positive Bray–Curtis dissimilarity.

For count-like observations, independent Poisson sampling is a useful reminder: both the expected count and its variance grow with abundance. Raw Euclidean distance therefore gives highly abundant species a large role partly because they are numerically large and partly because their sampling variation is larger. It is a pragmatic composition-oriented dissimilarity. If total abundance is driven mainly by unequal sampling effort, the effort problem should be addressed before interpreting any distance.

```
bray_curtis <- function(x) {
  x <- as.matrix(x)
  n <- nrow(x)
  out <- matrix(0, n, n, dimnames = list(rownames(x), rownames(x)))
  for (i in seq_len(n - 1)) {
    for (j in (i + 1):n) {
      out[i, j] <- out[j, i] <- sum(abs(x[i, ] - x[j, ])) / sum(x[i, ] + x[j, ])
    }
  }
  as.dist(out)
}

d_euclidean <- dist(abundance, method = "euclidean")
d_bray <- bray_curtis(abundance)

relative_abundance <- sweep(as.matrix(abundance), 1, rowSums(abundance), "/")
hellinger <- sqrt(relative_abundance)
d_hellinger <- dist(hellinger, method = "euclidean")
```

Bray–Curtis distance: composition with abundances

For non-negative abundance vectors, Bray–Curtis distance is

$$d_{BC}(x_i, x_j) = \frac{\sum_{k=1}^{p} |x_{ik} - x_{jk}|}{\sum_{k=1}^{p} (x_{ik} + x_{jk})}.$$

It lies between zero and one. A feature absent from both objects contributes neither to the numerator nor the denominator. Consequently, it is popular for species abundance, microbial communities, market baskets and other composition data with many zeros.

Bray–Curtis is a valid and useful dissimilarity, but it is not generally an exact Euclidean distance. That fact will become visible in its MDS eigenvalues.

Hellinger transformation: composition in Euclidean geometry

For each site, Hellinger transformation first changes cover scores into a composition (each row sums to one), then takes the square root of each relative abundance. The resulting distance is

$$d_H(i,j) = \left[\sum_{k=1}^{p}\left(\sqrt{\frac{x_{ik}}{x_{i+}}} - \sqrt{\frac{x_{jk}}{x_{j+}}}\right)^2\right]^{1/2}.$$

It therefore compares composition rather than raw total cover, while the square root reduces the dominance of common species. Most importantly for this unit, the transformed rows lie in an ordinary Euclidean space: their Euclidean distances can be represented exactly by a sufficiently high-dimensional MDS configuration. Hellinger transformation is not a generic replacement for Bray–Curtis; it answers a closely related but distinct question.

```
distance_long <- bind_rows(
  as.data.frame(as.table(as.matrix(d_bray))) |> rename(site_1 = Var1, site_2 = Var2, distance = Freq) |
  as.data.frame(as.table(as.matrix(d_hellinger))) |> rename(site_1 = Var1, site_2 = Var2, distance = Fre
) |> filter(site_1 != site_2)

ggplot(distance_long, aes(site_1, site_2, fill = distance)) +
  geom_tile(colour = "white") +
  geom_text(aes(label = sprintf("%.2f", distance)), size = 3) +
  facet_wrap(~ metric) +
  scale_fill_viridis_c(option = "C") +
  coord_fixed() +
  labs(title = "Pairwise distances depend on the question", x = NULL, y = NULL, fill = "Distance") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Bray–Curtis versus standardised Euclidean distance

Hellinger standardisation first converts each site profile to relative abundance and then takes square roots. The square root reduces the dominance of common species and makes Euclidean geometry available. It is not the same as applying a column-wise z-score: z-scoring gives every species equal variance across sites and can make a rare species influential merely because it varies little.

```
comparison <- tibble(
  bray = as.vector(d_bray),
  hellinger_euclidean = as.vector(d_hellinger)
)

ggplot(comparison, aes(bray, hellinger_euclidean)) +
  geom_point(size = 2.4, colour = blue, alpha = .8) +
  geom_smooth(method = "loess", se = FALSE, colour = orange, linewidth = 1) +
  labs(
    title = "Do the two distances discriminate similar site pairs?",
    subtitle = paste("Spearman rank correlation =", round(cor(comparison$bray, comparison$hellinger_eucl
    x = "Bray--Curtis dissimilarity",
    y = "Euclidean distance after Hellinger standardisation"
  )
```

Neither distance is universally better.

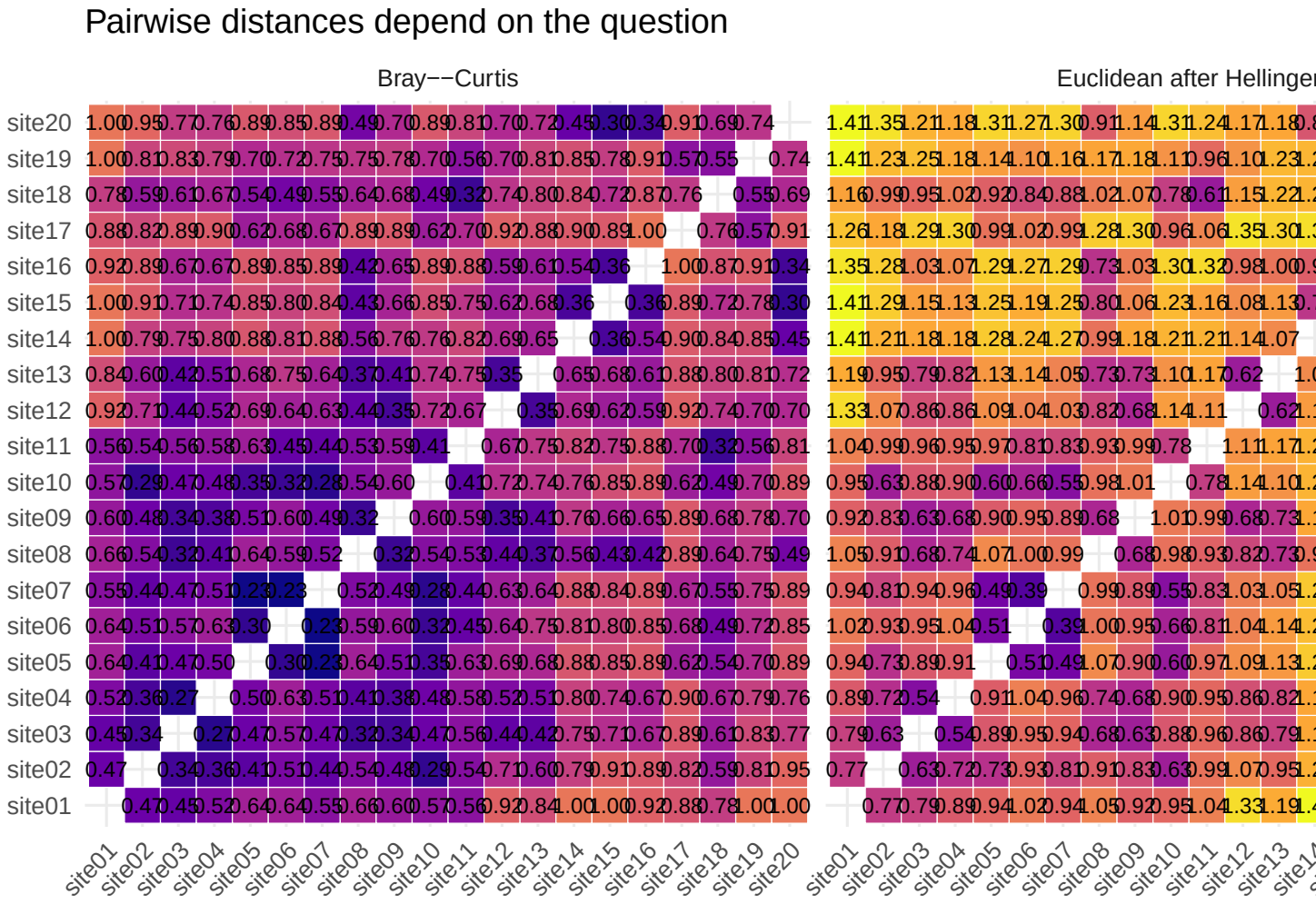


Figure 4: Bray–Curtis and Euclidean distance after Hellinger standardisation encode different, but related, composition contrasts. The numerical values use different scales; compare the pattern of close and distant pairs instead.

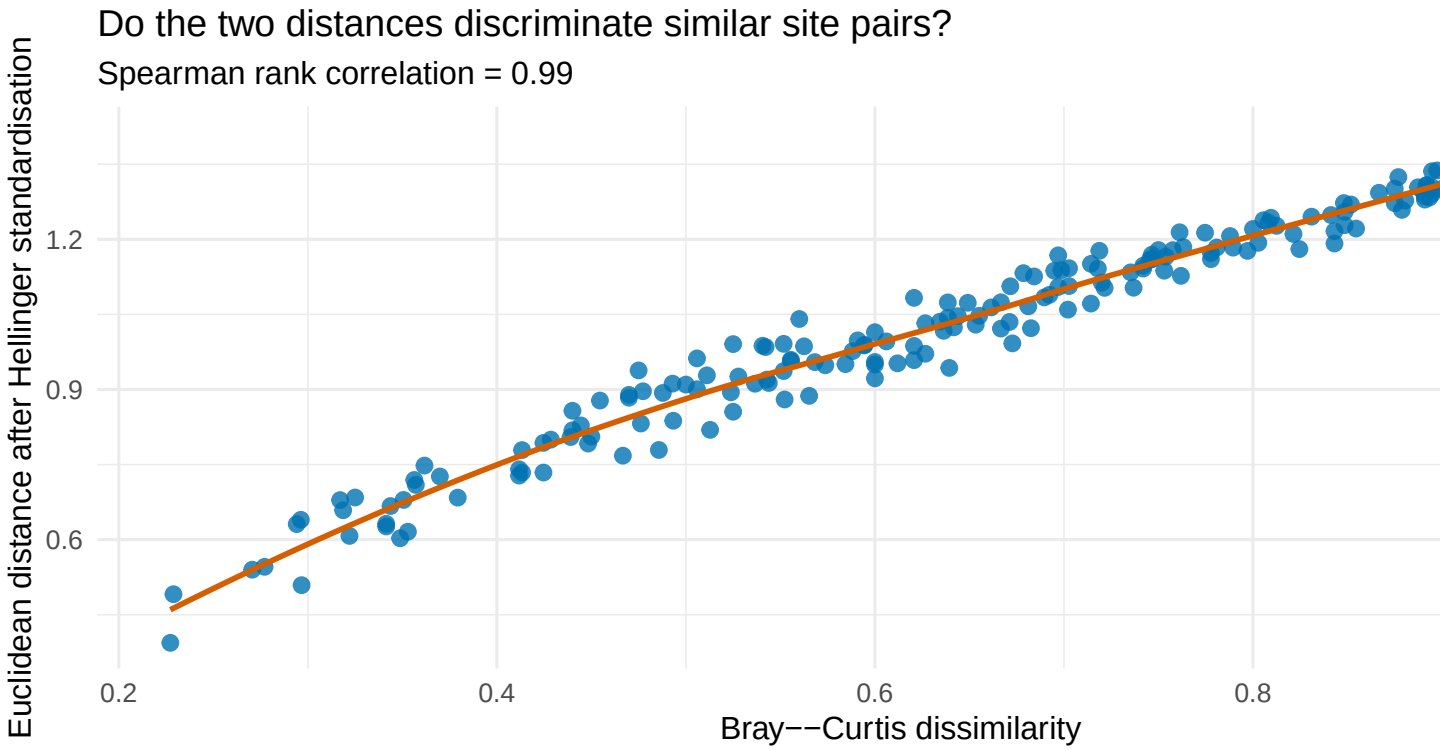


Figure 5: The metrics need not have the same numerical scale. Their rank agreement tells us whether they broadly discriminate the same pairs of sites.

If the scientific emphasis is…	Prefer…	Why
A traditional ecological comparison of abundance profiles, with shared zeros ignored	Bray–Curtis	Its denominator gives a relative, bounded discrepancy and it is widely interpretable for community composition.
A composition analysis requiring Euclidean geometry, PCA-like tools or an exactly Euclidean MDS input	Hellinger transformation + Euclidean distance	It compares transformed relative abundances and gives a Euclidean distance matrix.
Equal influence for every species after accounting for each species' variability	Column-standardised Euclidean distance	This is a different question; use it only when equal species variance is substantively intended.

Report the selected definition and show a sensitivity comparison when conclusions depend on it. A result is reproducible only if the transformations, distance and included species are documented.

Scaling is also a modelling choice

Column-wise standardisation is valuable for generic multivariate measurements in different units, but it is not a default repair for species-abundance tables. To make the contrast visible, we compare Hellinger Euclidean distance with Euclidean distance after species-wise z-scoring.

```r
scaled_distance <- as.matrix(dist(scale(abundance)))
hellinger_distance <- as.matrix(d_hellinger)

plot(hellinger_distance[upper.tri(hellinger_distance)], scaled_distance[upper.tri(scaled_distance)],
     pch = 19, col = adjustcolor(blue, .7),
     xlab = "Euclidean distance after Hellinger transformation",
     ylab = "Euclidean distance after species-wise z-scoring",
     main = "Different standardisations encode different questions")
abline(lm(scaled_distance[upper.tri(scaled_distance)] ~ hellinger_distance[upper.tri(hellinger_distance)]),
       col = orange, lwd = 2)
```

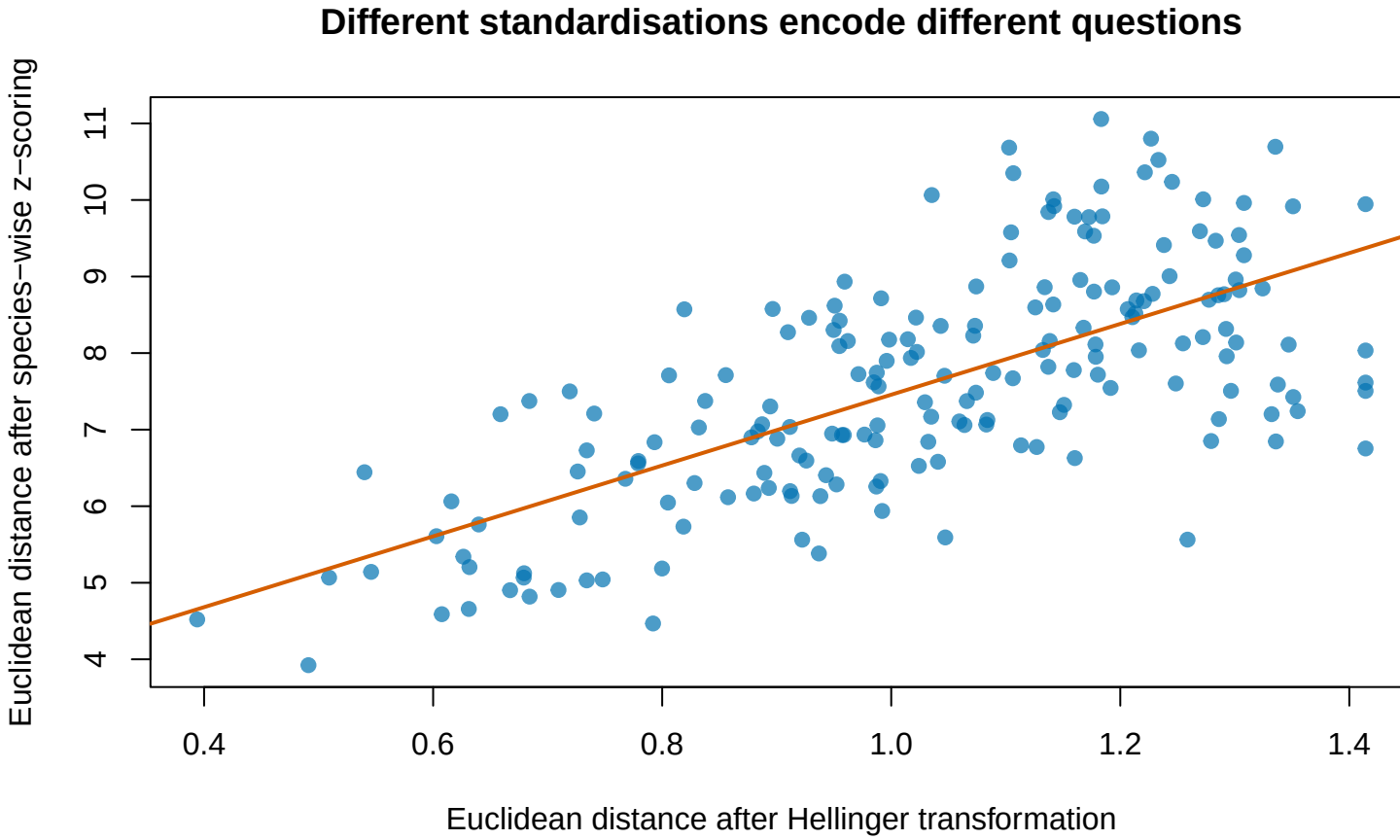


Figure 6: Standardisation changes Euclidean distances by changing the relative influence of variables.

Classical MDS: turn distances into coordinates

MDS receives a distance matrix and returns coordinates. Classical MDS finds a Euclidean configuration whose interpoint distances reproduce the input as closely as possible. The configuration can then be plotted just like ordinary coordinates.

Spectral decomposition, one matrix at a time

The construction is easiest to see with only three objects. Reuse the three points introduced earlier: A = (0, 0), B = (3, 4), and C = (3, 0). We know their geometry, so we can check whether MDS recovers it from

distances alone. The coordinates are used only to create the example; every following step uses the distance matrix.

Step 1 — distances and squared distances

Let D contain the pairwise distances and let $D^{(2)}$ contain their squares:

$$D = (d_{ij}), \qquad D^{(2)} = (d_{ij}^2).$$

```
X_original <- rbind(A = c(0, 0), B = c(3, 4), C = c(3, 0))
d_orig <- tibble(object = rownames(X_original), x = X_original[, 1], y = X_original[, 2])
D <- as.matrix(dist(X_original))
D2 <- D^2

knitr::kable(d_orig, digits = 2, caption = "Original two-dimensional coordinates used only to create thi
```

Table 12: Original two-dimensional coordinates used only to create this small example.

object	x	y
A	0	0
B	3	4
C	3	0

```
knitr::kable(D, digits = 2, caption = "Step 1a: distance matrix D.")
```

Table 13: Step 1a: distance matrix D.

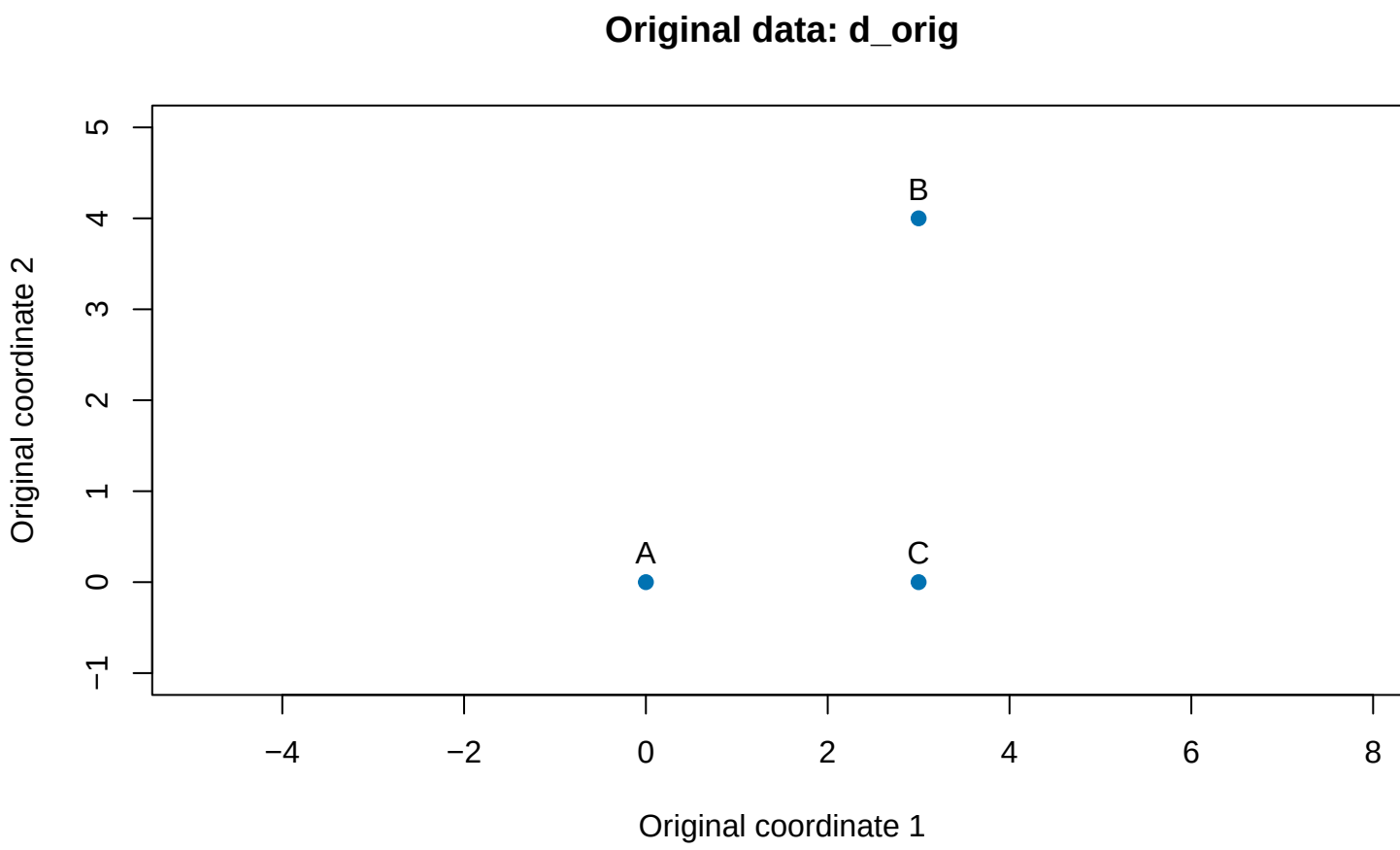
	A	B	C
A	0	5	3
B	5	0	4
C	3	4	0

```
knitr::kable(D2, digits = 2, caption = "Step 1b: squared-distance matrix D^(2).")
```

Table 14: Step 1b: squared-distance matrix D^(2).

	A	B	C
A	0	25	9
B	25	0	16
C	9	16	0

```
plot(d_orig$x, d_orig$y, asp = 1, pch = 19, col = blue, xlab = "Original coordinate 1",
     ylab = "Original coordinate 2", main = "Original data: d_orig",
     xlim = c(-1, 4), ylim = c(-1, 5))
text(d_orig$x, d_orig$y, labels = d_orig$object, pos = 3)
```



Step 2 — centre the squared distances

With $n = 3$, the centring matrix is

$$J = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

Double-centring converts squared distances into a Gram matrix:

$$B = -\frac{1}{2} J D^{(2)} J.$$

For exactly Euclidean distances, B is the centred inner-product matrix $X_c X_c^T$, where X_c denotes centred coordinates. It is symmetric and positive semidefinite.

```
n <- nrow(D)
J <- diag(n) - matrix(1 / n, n, n)
B <- -0.5 * J %*% D2 %*% J

knitr::kable(J, digits = 3, caption = "Step 2a: centring matrix J.")
```

Table 15: Step 2a: centring matrix J.

0.667	-0.333	-0.333

-0.333	0.667	-0.333
-0.333	-0.333	0.667

```
knitr::kable(B, digits = 3, caption = "Step 2b: centred Gram matrix B.")
```

Table 16: Step 2b: centred Gram matrix B.

5.778	-5.556	-0.222
-5.556	8.111	-2.556
-0.222	-2.556	2.778

Step 3 — spectral decomposition of B

Because B is real and symmetric, the spectral theorem gives

$$B = Q\Lambda Q^T,$$

where the columns of Q are orthonormal eigenvectors and Λ is a diagonal matrix of eigenvalues. The number of positive eigenvalues is the dimension needed for an exact Euclidean representation (apart from numerical rounding).

```
decomposition <- eigen(B, symmetric = TRUE)
Q <- decomposition$vectors
Lambda <- diag(decomposition$values)
rownames(Q) <- rownames(D)
rownames(Lambda) <- colnames(Lambda) <- paste0("lambda", seq_len(n))
colnames(Q) <- paste0("q", seq_len(n))

knitr::kable(Q, digits = 3, caption = "Step 3a: eigenvector matrix Q.")
```

Table 17: Step 3a: eigenvector matrix Q.

	q1	q2	q3
A	0.598	-0.556	0.577
B	-0.781	-0.240	0.577
C	0.183	0.796	0.577

```
knitr::kable(Lambda, digits = 3, caption = "Step 3b: diagonal eigenvalue matrix Lambda.")
```

Table 18: Step 3b: diagonal eigenvalue matrix Lambda.

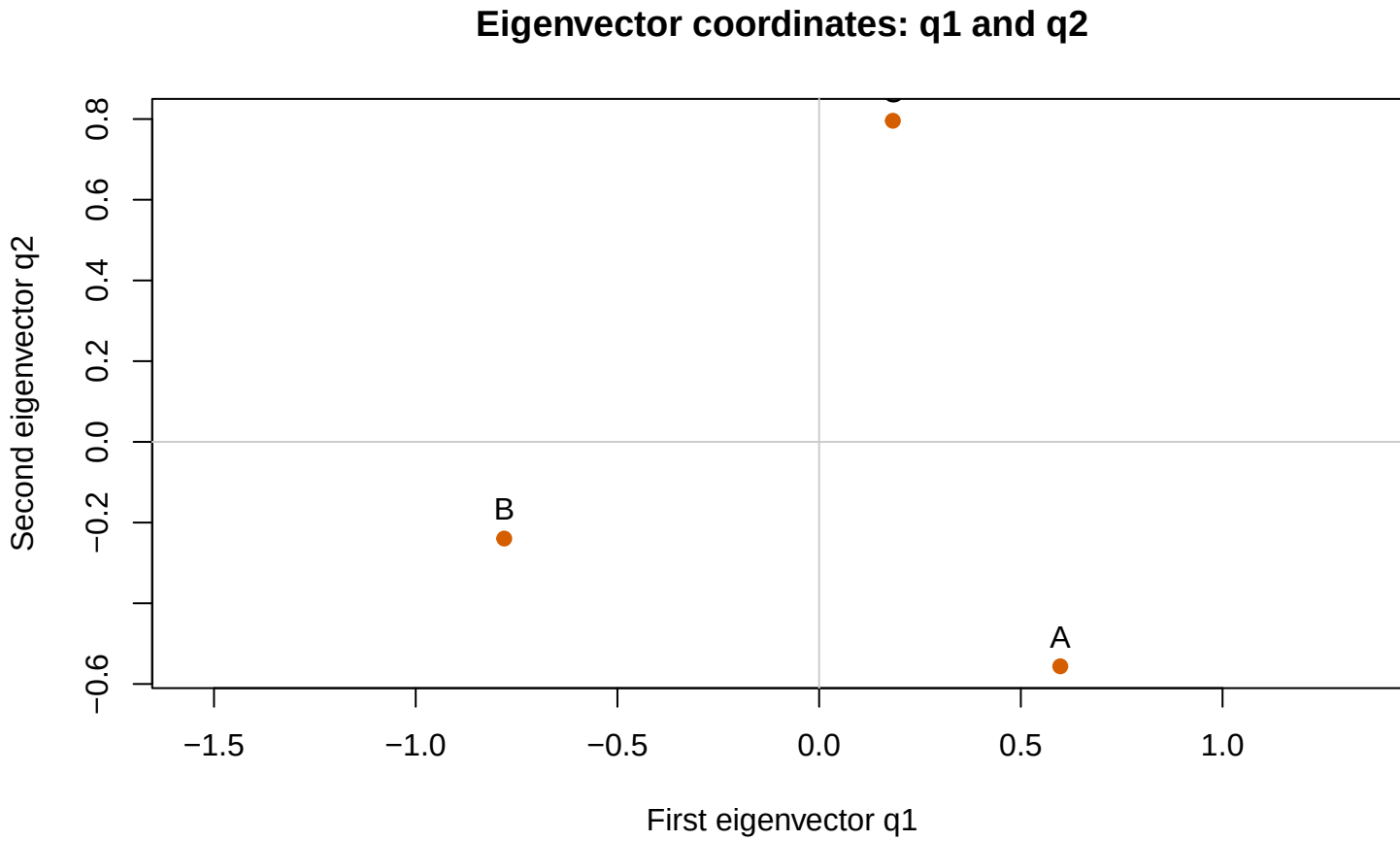
	lambda1	lambda2	lambda3
lambda1	12.964	0.000	0
lambda2	0.000	3.703	0
lambda3	0.000	0.000	0

```
knitr::kable(tibble(dimension = seq_len(n), eigenvalue = decomposition$values),
             digits = 3, caption = "Step 3c: eigenvalues in descending order.")
```

Table 19: Step 3c: eigenvalues in descending order.

dimension	eigenvalue
1	12.964
2	3.703
3	0.000

```
q_plot <- tibble(object = rownames(Q), q1 = Q[, 1], q2 = Q[, 2])
plot(q_plot$q1, q_plot$q2, asp = 1, pch = 19, col = orange,
     xlab = "First eigenvector q1", ylab = "Second eigenvector q2",
     main = "Eigenvector coordinates: q1 and q2")
abline(h = 0, v = 0, col = "grey80")
text(q_plot$q1, q_plot$q2, labels = q_plot$object, pos = 3)
```



Step 4 — recover coordinates

Retain the r positive eigenvalues and their eigenvectors. If Q_r and Λ_r are the retained matrices, the MDS coordinates are

$$X = Q_r \Lambda_r^{1/2}.$$

The recovered X can be translated, rotated or reflected relative to X_original; those transformations leave its distances unchanged.

```
positive <- which(decomposition$values > 1e-10)
Q_r <- Q[, positive, drop = FALSE]
Lambda_r <- diag(decomposition$values[positive], nrow = length(positive))
X <- Q_r %*% sqrt(Lambda_r)
rownames(X) <- rownames(D)
colnames(X) <- paste0("MDS", seq_len(ncol(X)))

X_table <- tibble(object = rownames(X), MDS1 = X[, 1], MDS2 = X[, 2])
knitr::kable(d_orig, digits = 3, caption = "Step 4a: original coordinates d_orig.")
```

Table 20: Step 4a: original coordinates d_orig.

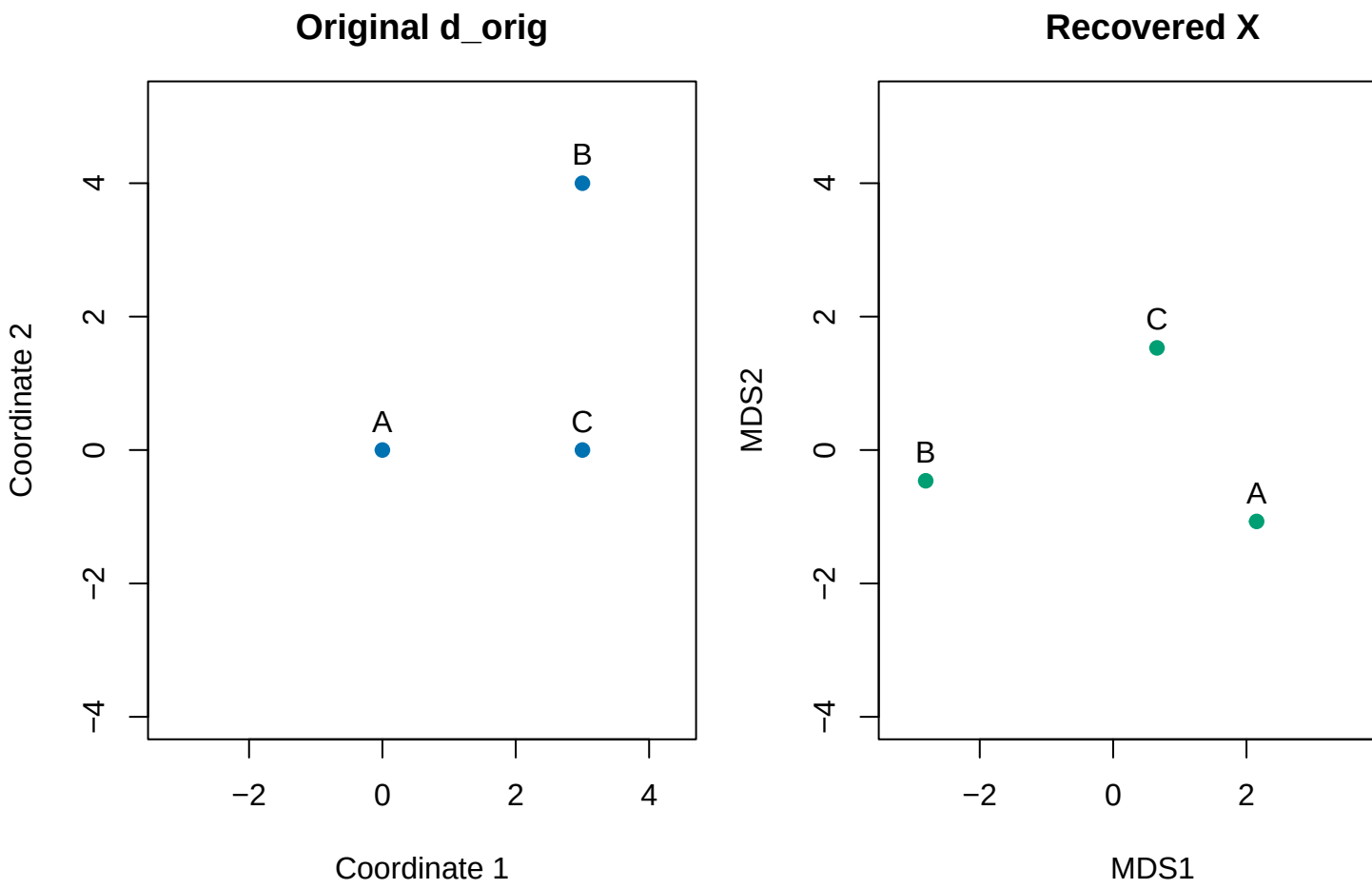
object	x	y
A	0	0
B	3	4
C	3	0

```
knitr::kable(X_table, digits = 3, caption = "Step 4b: coordinates X recovered by classical MDS.")
```

Table 21: Step 4b: coordinates X recovered by classical MDS.

object	MDS1	MDS2
A	2.152	-1.070
B	-2.810	-0.461
C	0.658	1.531

```
old_par <- par(no.readonly = TRUE)
par(mfrow = c(1, 2), mar = c(4, 4, 3, 1))
common_limits <- range(c(d_orig$x, d_orig$y, X[, 1], X[, 2]))
common_limits <- common_limits + c(-.4, .4)
plot(d_orig$x, d_orig$y, asp = 1, pch = 19, col = blue,
     xlab = "Coordinate 1", ylab = "Coordinate 2", main = "Original d_orig",
     xlim = common_limits, ylim = common_limits)
text(d_orig$x, d_orig$y, labels = d_orig$object, pos = 3)
plot(X[, 1], X[, 2], asp = 1, pch = 19, col = green,
     xlab = "MDS1", ylab = "MDS2", main = "Recovered X",
     xlim = common_limits, ylim = common_limits)
text(X[, 1], X[, 2], labels = rownames(X), pos = 3)
```




```
par(old_par)
```

This is the whole classical-MDS mechanism. It is not a black box: eigenvectors provide the coordinate directions, while eigenvalues quantify how much centred squared-distance structure those directions represent.

The same objects in a reusable function

The function below uses the same names as the derivation: D, D2, J, B, Q, Lambda and X. Its return value retains each object so that its calculation can be inspected after fitting a larger dataset.

If B has a materially negative eigenvalue, its associated diagonal entry in Lambda has no real square root. We therefore cannot turn that component into real Euclidean coordinates. The function retains only positive eigenvalues: its X is a Euclidean approximation, but it cannot reconstruct the original D exactly when negative eigenvalues are present.

```
classical_mds <- function(D, k = 2, tolerance = 1e-10) {
  D <- as.matrix(D)
  D2 <- D^2
  n <- nrow(D)
  J <- diag(n) - matrix(1 / n, n, n)
  B <- -0.5 * J %*% D2 %*% J

  decomposition <- eigen(B, symmetric = TRUE)
```

```
  Q <- decomposition$vectors
  Lambda <- diag(decomposition$values)
  positive <- which(decomposition$values > tolerance)
  keep <- positive[seq_len(min(k, length(positive)))]
  Q_r <- Q[, keep, drop = FALSE]
  Lambda_r <- diag(decomposition$values[keep], nrow = length(keep))
  X <- Q_r %*% sqrt(Lambda_r)
  rownames(X) <- rownames(D)

  list(D = D, D2 = D2, J = J, B = B, Q = Q, Lambda = Lambda,
       X = X, eigenvalues = decomposition$values)
}

mds_bray <- classical_mds(d_bray, k = 2)
mds_hellinger <- classical_mds(d_hellinger, k = 2)
round(mds_bray$eigenvalues, 4)
```

```
#>  [1]  1.7163  1.0224  0.4615  0.3822  0.2791  0.2366  0.1691  0.0962  0.0745
#> [10]  0.0617  0.0549  0.0192  0.0161  0.0040  0.0000 -0.0264 -0.0429 -0.0547
#> [19] -0.0741 -0.0968
```

MDS map of the communities

```
mds_df <- as.data.frame(mds_bray$X) |>
  setNames(c("MDS1", "MDS2")) |>
  mutate(site = rownames(mds_bray$X)) |>
  left_join(site_metadata |> select(site, Management, Moisture), by = "site")

ggplot(mds_df, aes(MDS1, MDS2, colour = Management, label = site)) +
  geom_hline(yintercept = 0, colour = "grey85") +
  geom_vline(xintercept = 0, colour = "grey85") +
  geom_point(size = 4) +
  geom_text(nudge_y = 0.025, show.legend = FALSE) +
  scale_colour_brewer(palette = "Dark2") +
  coord_equal() +
  labs(title = "Community composition: Bray--Curtis MDS", x = "MDS dimension 1", y = "MDS dimension 2",
```

Sites with similar species composition tend to lie close together; management is shown as external descriptive context, not as an input to the distance. Any visual association with management is exploratory and does not establish a causal effect.

Why MDS axes have no inherent interpretation

Multiplying every coordinate matrix by an orthogonal rotation or reflection leaves all interpoint Euclidean distances unchanged. Thus MDS1 is not “the management gradient” unless an external analysis supports that interpretation. The stable evidence is proximity, separation, groups and the fit of map distances to original distances.

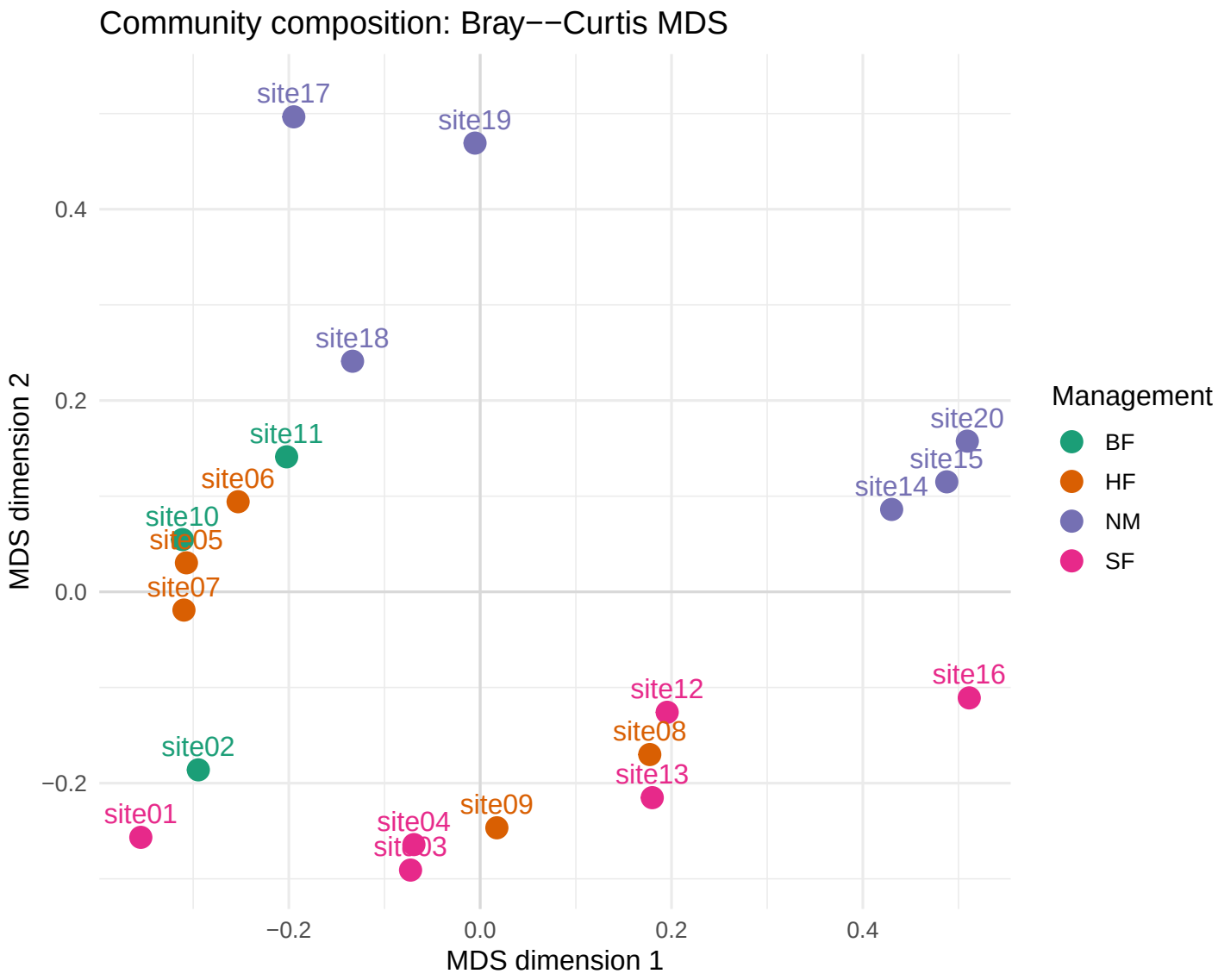


Figure 7: Classical MDS map based on Bray–Curtis dissimilarity. Colour records the observed management category; the map itself was computed using species cover classes only.

```r
rot90 <- matrix(c(0, -1, 1, 0), nrow = 2)
rotated <- as.matrix(mds_df[, c("MDS1", "MDS2")]) %*% rot90
rotated_df <- data.frame(MDS1 = rotated[, 1], MDS2 = rotated[, 2], site = mds_df$site, Management = mds

p1 <- ggplot(mds_df, aes(MDS1, MDS2, colour = Management, label = site)) + geom_point(size = 3) + geom_
p2 <- ggplot(rotated_df, aes(MDS1, MDS2, colour = Management, label = site)) + geom_point(size = 3) + g

print(p1)
```

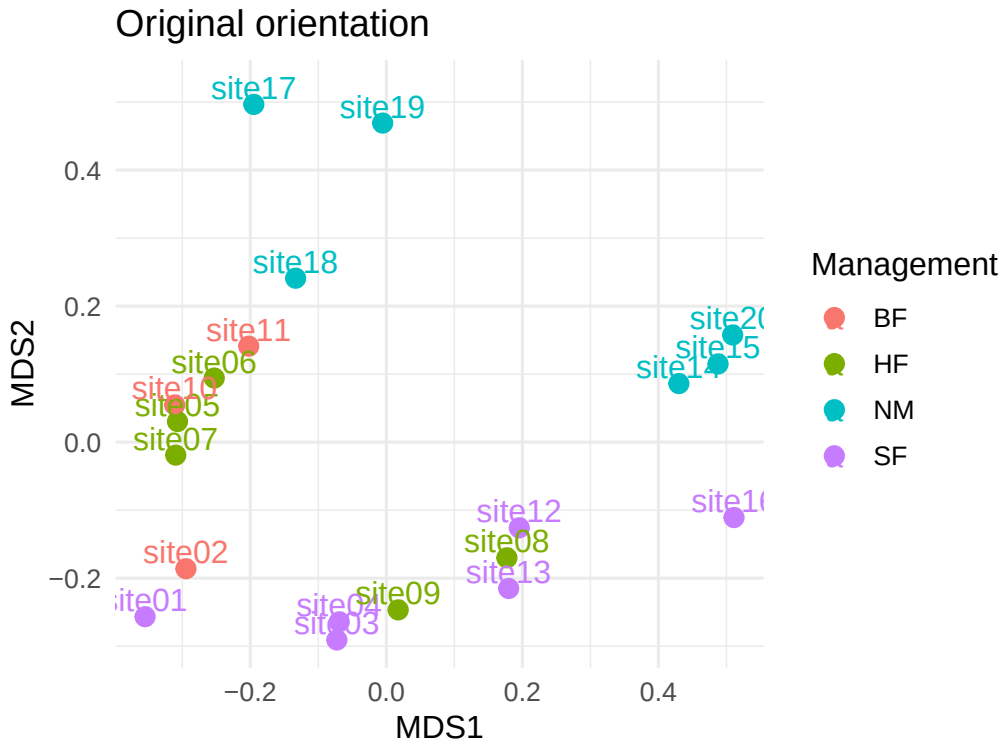


Figure 8: A 90-degree rotation changes axes but preserves all pairwise distances. Both panels are the same MDS solution.

```r
print(p2)
```

Choosing the number of dimensions

Two dimensions are chosen here because the intended final display is a static map. This is a communication constraint, not evidence that the data are intrinsically two-dimensional. In an exactly Euclidean problem, the number of positive eigenvalues is the dimension required for exact reconstruction. With non-Euclidean dissimilarities, negative eigenvalues add a separate warning about the geometry.

Choose k using all three pieces of evidence below:

1. **Eigenvalue spectrum:** is there an elbow after the first few positive eigenvalues?
2. **Distance reproduction:** does the configuration preserve the pairwise distances sufficiently for the intended claim?

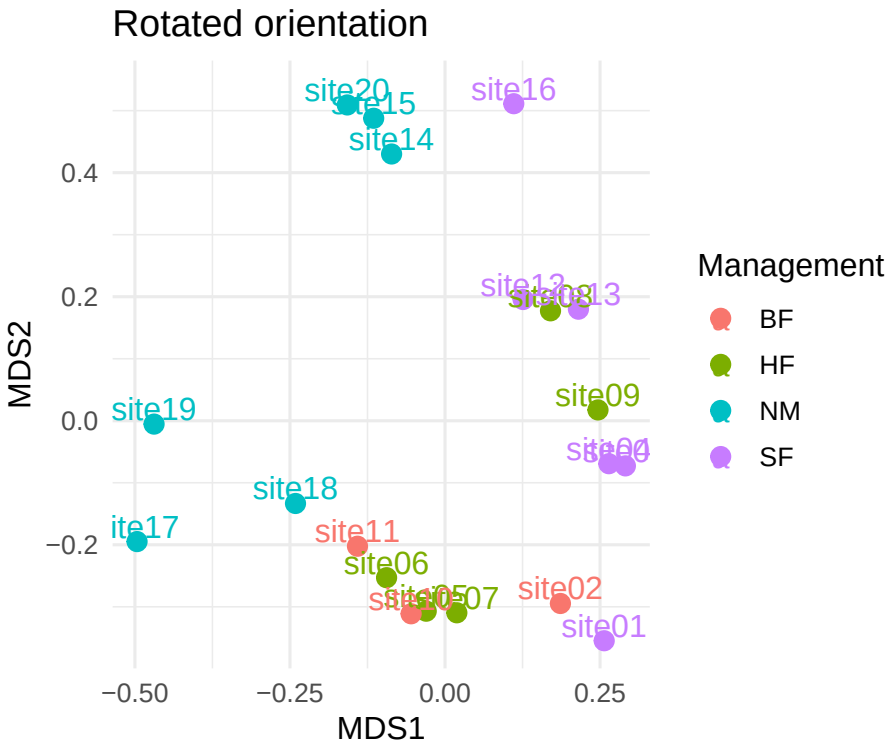


Figure 9: A 90-degree rotation changes axes but preserves all pairwise distances. Both panels are the same MDS solution.

3. **Communication purpose:** two dimensions are easy to print and annotate; three dimensions may be inspected interactively; additional dimensions may be retained for later analysis without being drawn as a single map.

Do not call the eigenvalue fraction "variance explained" without qualification: this is MDS of a distance matrix, not PCA of the original variables. Here we use the fraction of **absolute-eigenvalue structure**. Its denominator is $\sum_j |\lambda_j|$, including negative eigenvalues. Thus even retaining every positive eigenvalue need not reach 100%: the remaining fraction is non-Euclidean structure that no real Euclidean coordinate system can reproduce exactly.

```
positive_bray <- mds_bray$eigenvalues[mds_bray$eigenvalues > 1e-10]
total_absolute_structure <- sum(abs(mds_bray$eigenvalues))
negative_structure_fraction <- sum(abs(mds_bray$eigenvalues[mds_bray$eigenvalues < -1e-10])) /
  total_absolute_structure
dimension_summary <- tibble(
  dimension = seq_along(positive_bray),
  eigenvalue = positive_bray,
  cumulative_fraction = cumsum(positive_bray) / total_absolute_structure
)

p_scree <- ggplot(dimension_summary, aes(dimension, eigenvalue)) +
  geom_col(fill = blue) +
  labs(title = "Positive-eigenvalue spectrum", x = "Dimension", y = "Eigenvalue")

p_cumulative <- ggplot(dimension_summary, aes(dimension, cumulative_fraction)) +
  geom_line(colour = orange, linewidth = 1) +
```

```
  geom_point(colour = orange, size = 2.5) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1), limits = c(0, 1)) +
  labs(title = "Cumulative absolute-eigenvalue fraction", x = "Dimensions retained",
       y = "Fraction of total absolute structure")

print(p_scree)
```

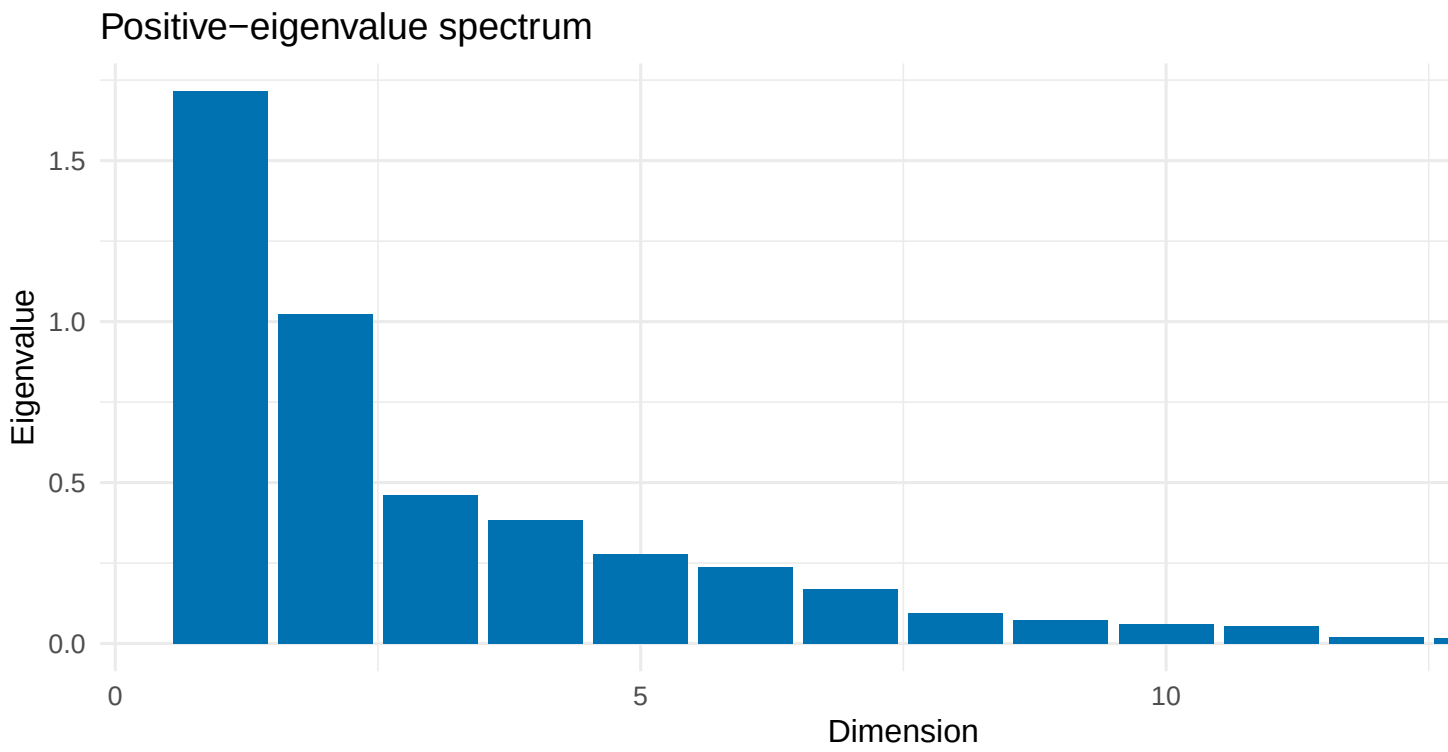


Figure 10: The positive-eigenvalue scree plot and the cumulative absolute-eigenvalue fraction separate represented Euclidean structure from non-Euclidean structure.

```
print(p_cumulative)
```

```
scales::percent(negative_structure_fraction, accuracy = .1)
```

```
#> [1] "6.0%"
```

The printed percentage is the share of total absolute-eigenvalue structure carried by negative eigenvalues. It is the portion that will remain unreconstructable in any real Euclidean MDS configuration, regardless of how many positive dimensions are retained.

Reading MDS eigenvalues and fit

For an exactly Euclidean distance, all non-zero eigenvalues of B are non-negative. Bray–Curtis is not guaranteed to have that property. Small negative eigenvalues tell us that no exact Euclidean configuration reproduces all of its distances. They are not an automatic failure: they are diagnostic evidence that the distance geometry is only approximately Euclidean.

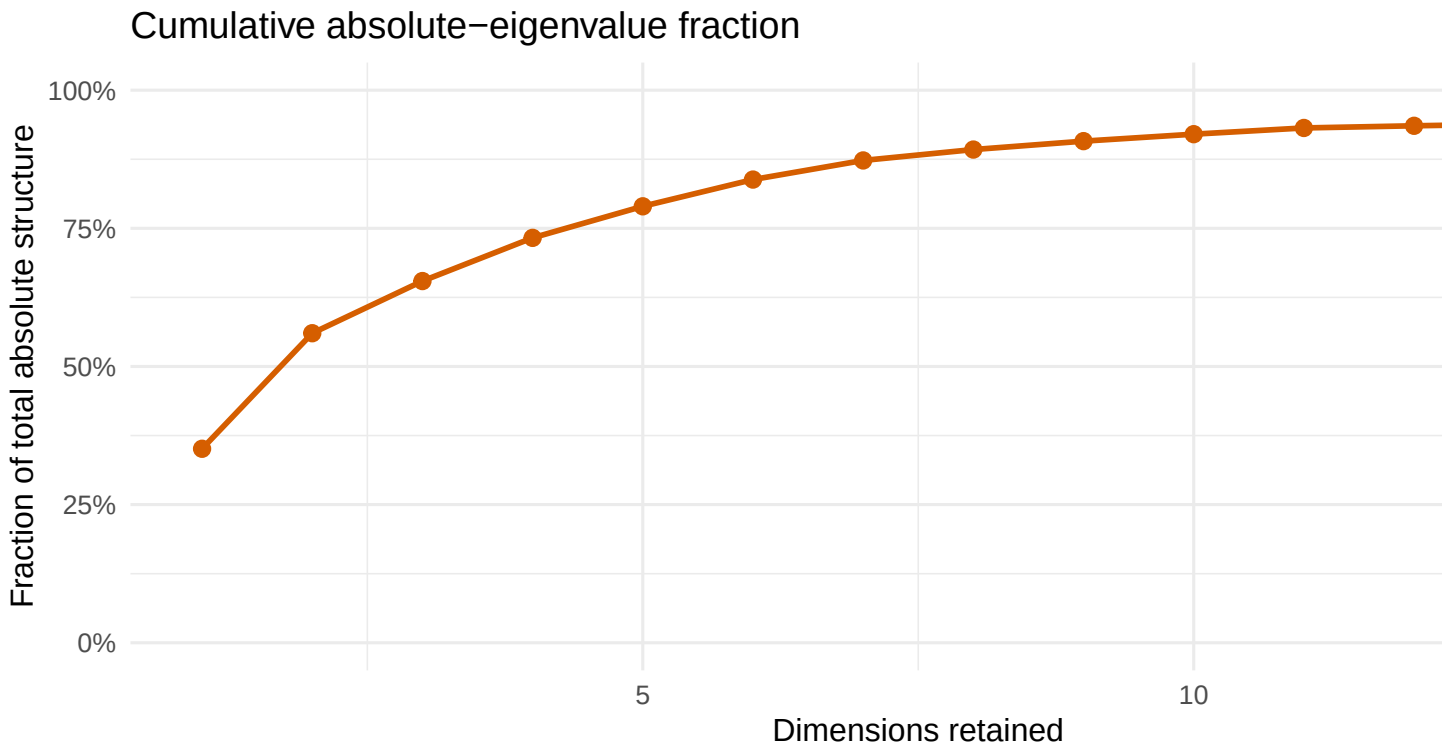


Figure 11: The positive-eigenvalue scree plot and the cumulative absolute-eigenvalue fraction separate represented Euclidean structure from non-Euclidean structure.

```
eig_df <- tibble(dimension = seq_along(mds_bray$eigenvalues), eigenvalue = mds_bray$eigenvalues)
ggplot(eig_df, aes(dimension, eigenvalue, fill = eigenvalue > 0)) +
  geom_col(show.legend = FALSE) +
  geom_hline(yintercept = 0) +
  scale_fill_manual(values = c(`TRUE` = blue, `FALSE` = orange)) +
  labs(title = "MDS eigenvalues", x = "Dimension", y = "Eigenvalue")
```

Two simple summaries are useful:

```
abs_eigen <- abs(mds_bray$eigenvalues)
variance_like_fit <- sum(abs_eigen[1:2]) / sum(abs_eigen)

map_d <- dist(mds_bray$X)
distance_correlation <- cor(as.vector(d_bray), as.vector(map_d))

c(two_dimension_positive_eigenvalue_fraction = variance_like_fit,
  correlation_original_vs_map_distance = distance_correlation)
```

```
#> two_dimension_positive_eigenvalue_fraction
#>                                  0.5601833
#>       correlation_original_vs_map_distance
#>                                  0.9163764
```

The first value reports how much of the **absolute-eigenvalue** structure is retained in two dimensions. The overall structure comprises the positive eigenvalues that can be represented in an Euclidean space (i.e. by

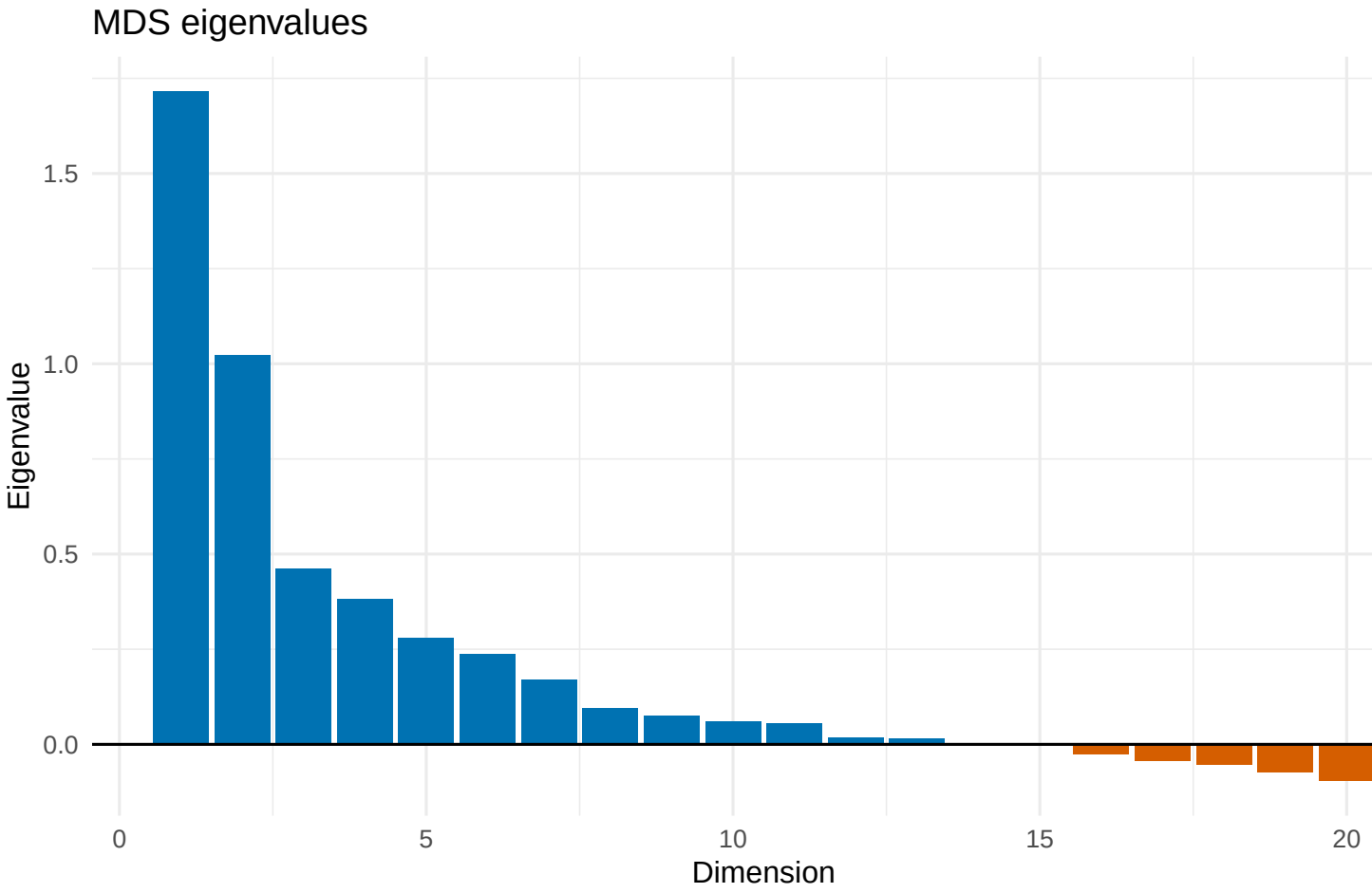


Figure 12: Positive and negative MDS eigenvalues for Bray–Curtis dissimilarity. Negative values reveal non-Euclidean structure.

increasing the number of dimensions), and the negative ones that cannot be described by an Euclidean distance, no matter the number of dimensions. The second reports whether pairs far apart in the original distance also tend to be far apart on the map. Neither is a universal threshold; inspect the map and its distance reproduction before treating it as a complete representation.

What changes under Hellinger–Euclidean distance?

The following comparison does not select a winner mechanically. It makes the trade-off explicit: Bray–Curtis is a familiar ecological dissimilarity but its classical MDS has negative eigenvalues; Hellinger–Euclidean distance gives an exactly Euclidean input but can emphasise a different ecological contrast.

```
mds_summary <- function(d, fit) {
  eig <- fit$eigenvalues
  total_absolute_structure <- sum(abs(eig))
  retained_absolute_structure <- sum(abs(eig[seq_len(min(2, length(eig)))]))
  map_distance <- dist(fit$X)
  c(
    negative_eigenvalues = sum(eig < -1e-10),
    negative_absolute_fraction = sum(abs(eig[eig < -1e-10])) / total_absolute_structure,
    two_dimension_absolute_fraction = retained_absolute_structure / total_absolute_structure,
    original_map_distance_correlation = cor(as.vector(d), as.vector(map_distance))
  )
}

comparison_table <- rbind(
  `Bray--Curtis` = mds_summary(d_bray, mds_bray),
  `Hellinger + Euclidean` = mds_summary(d_hellinger, mds_hellinger)
)
knitr::kable(round(comparison_table, 3), caption = "Diagnostics for two defensible distance choices. Hig
```

Table 22: Diagnostics for two defensible distance choices. Higher map-distance correlation indicates better two-dimensional distance reproduction; it is not a universal measure of scientific quality.

	negative_eigenvalues	negative_absolute_fraction	two_dimension_absolute_fraction	original_map_distance_correlation
Bray–Curtis	5	0.06	0.560	0.916
Hellinger + Euclidean	0	0.00	0.505	0.905

If the scientific message is stable across the two maps, confidence in the broad description increases. If it changes, do not choose the prettier map: return to the data-generating and sampling context, state the competing definitions of similarity, and report the sensitivity.

```
library(patchwork)
shepard <- tibble(original = as.vector(d_bray), map = as.vector(map_d))
p1=ggplot(shepard, aes(original, map)) +
  geom_point(size = 2.5, colour = blue, alpha = .8) +
  geom_abline(slope = 1, intercept = 0, linetype = 2, colour = orange) +
  coord_equal(xlim = c(0,1)) +
  labs(title = "How faithfully does the map reproduce distance?", x = "Original Bray--Curtis dissimilari
```

```
shepard <- tibble(original = as.vector(d_hellinger), map = as.vector(map_d))
p2=ggplot(shepard, aes(original, map)) +
  geom_point(size = 2.5, colour = blue, alpha = .8) +
  geom_abline(slope = 1, intercept = 0, linetype = 2, colour = orange) +
  coord_equal(xlim = c(0,1)) +
  labs(title = "How faithfully does the map reproduce distance?", x = "Hellinger distance", y = "Distan

p1+p2
```

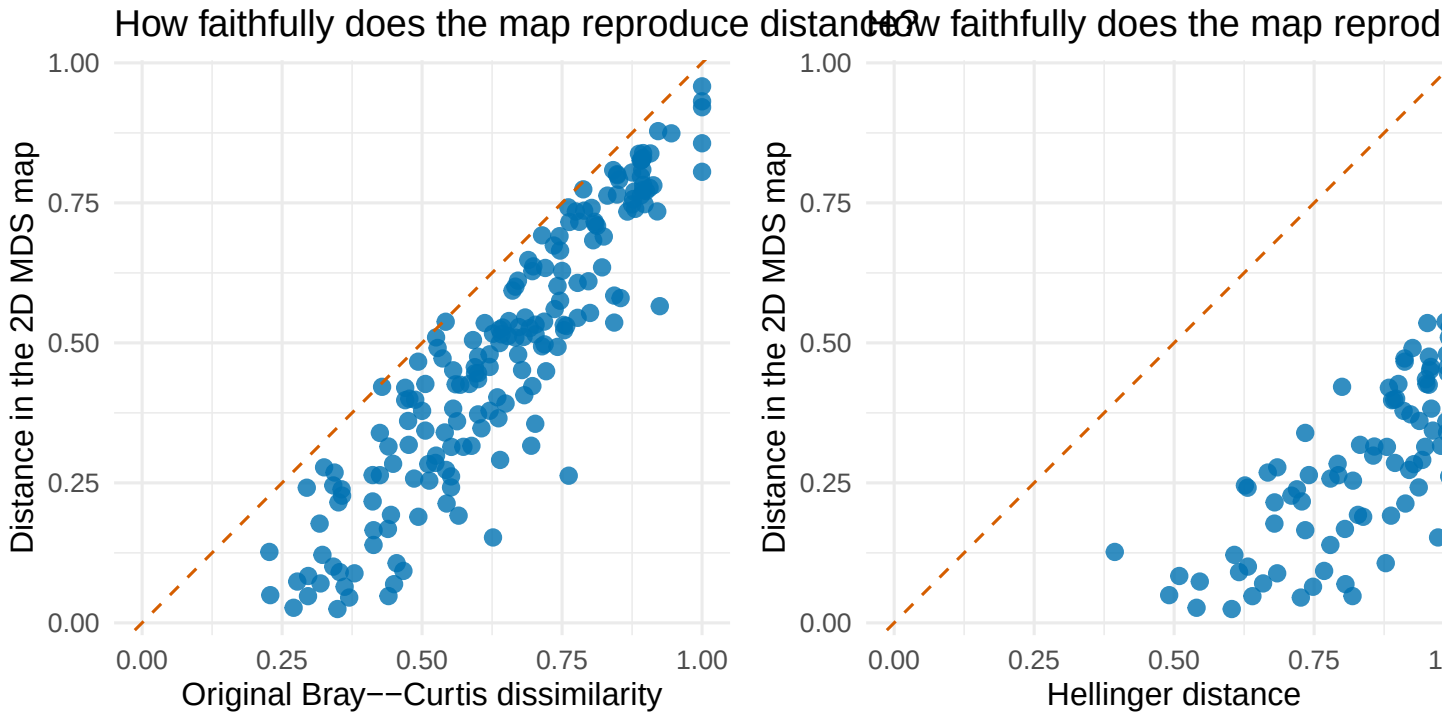


Figure 13: A Shepard-style plot compares original Bray–Curtis dissimilarities with distances on the two-dimensional MDS map. Points close to the diagonal would indicate perfect reproduction.

Classical MDS and non-metric MDS

Classical MDS tries to preserve the numerical values of distances. If only the rank order of dissimilarities is trusted, **non-metric MDS** is often preferable: it seeks a configuration whose map distances have the same ordering as the input dissimilarities. Non-metric MDS is especially common with ecological or ordinal dissimilarities, and is normally assessed with stress and repeated starts.

We postpone non-metric MDS implementation until students are comfortable with the classical construction. The conceptual contrast is important:

Method	Main aim	Diagnostic
Classical MDS	Preserve numerical distances as closely as possible	eigenvalues, distance-reproduction plot
Non-metric MDS	Preserve rank order of dissimilarities	stress, distance-rank plot, stability across starts

Practical workflow

When building a distance-based visualization, document these decisions:

1. What are the objects? What are the variables or features?
2. Have variables been transformed, standardised or weighted? Why?
3. Which distance was chosen, and what kind of difference does it represent?
4. Does a two-dimensional map reproduce the distances adequately?
5. Are map labels and graphical encodings accessible and interpretable?
6. Which conclusion is descriptive, and which would require additional design or modelling assumptions?

Take-home messages

- A distance is a model of what it means for two objects to differ.
- Standardisation is not a neutral default; it changes that model.
- Classical MDS is spectral decomposition applied to a double-centred squared distance matrix.
- MDS axes are arbitrary up to rotation and reflection. Interpret distances, neighbourhoods and separation instead.
- Negative eigenvalues are informative diagnostics for non-Euclidean distances.
- A beautiful MDS map is not sufficient evidence: report the input distance and inspect its reproduction.

Note on AI assistance

These teaching materials were prepared with assistance from ChatGPT (OpenAI) in drafting explanations, developing examples and generating R code. The author retains responsibility for the final content.