Proposal Analysis: Step-by-Step Guide

Simplified Code for Combined Distance Species Delimitation

Practice Guide for Prospectus

2026-02-23

Contents

1 OVERVIEW

1.1 What This Guide Covers

This guide provides **simplified, step-by-step code** for your proposal analysis. Instead of complex functions, each step is written out explicitly so you can:

- **Understand** what each line does
- **Practice** running the code yourself
- **Modify** parameters to see what happens
- **Learn** the statistical methods hands-on

1.2 The Five Simulations

You will create and analyze **five simulation scenarios**:

1. **SIM1: Easy Separation** - Three clearly distinct species
2. **SIM2: Moderate Separation** - Realistic Australopithecus-like overlap
3. **SIM3: Oversplit** - One species artificially split into three "taxa"
4. **SIM4: Chronospecies** - One lineage evolving through time
5. **SIM5: Geographic Variation** - One widespread species across regions

1.3 Learning Objectives

By working through this guide, you will learn:

- How to generate realistic morphological data
- How to calculate Mahalanobis distance (accounts for correlations)
- How to calculate Gower distance (handles discrete traits)
- How to combine them into combined distance
- How to classify specimens (k-nearest neighbors)
- How to cluster specimens (PAM clustering)
- How to partition variance (temporal and geographic)
- How to make taxonomic decisions

2 SETUP

2.1 Install and Load Packages

What you need:

```
# Run this section once to install packages
install.packages("MASS")        # For multivariate normal data
install.packages("cluster")     # For Gower distance and clustering
install.packages("ggplot2")     # For plotting
install.packages("viridis")     # For color scales
install.packages("lme4")        # For hierarchical models
```

Load the packages:

```
# Run this at the start of every session
library(MASS)         # mvrnorm() for generating correlated data
library(cluster)      # daisy() for Gower, pam() for clustering
library(ggplot2)      # For all plots
library(viridis)      # Color-blind friendly colors
library(lme4)         # lmer() for hierarchical models

# Set random seed for reproducibility
set.seed(2024)
```

Why we need each package:

- `MASS`: Generates data with realistic correlations (teeth dimensions correlate)
- `cluster`: Calculates Gower distance for discrete traits
- `ggplot2`: Creates publication-quality figures
- `viridis`: Color palettes that work for color-blind readers
- `lme4`: Fits hierarchical models to partition variance

2.2 Create Output Directories

```
# Create folders to save your work
dir.create("data", showWarnings = FALSE)
dir.create("results", showWarnings = FALSE)
dir.create("figures", showWarnings = FALSE)

# Check they were created
list.dirs()
```

What this does: Creates three folders where you will save: - `data/` - Your simulated datasets - `results/` - Statistical output and tables - `figures/` - Plots and visualizations

3 CORRELATION STRUCTURE

3.1 Understanding Dental Correlations

Key concept: Dental measurements are NOT independent. If a specimen has large M1, it probably also has large M2.

Real primate data shows correlations: - M1 buccolingual M1 mesiodistal: r = 0.75 (highly correlated) - M1 M2 dimensions: r = 0.65-0.70 (moderately correlated) - Upper lower molars: r = 0.60 (correlated)

Why this matters: Euclidean distance treats variables as independent (WRONG). Mahalanobis distance accounts for correlations (CORRECT).

3.2 Define the Correlation Matrix

```
# Create a 5x5 correlation matrix
# Variables: M1_BL, M1_MD, M2_BL, M2_MD, P4_BL

correlation_matrix <- matrix(c(
  # M1_BL  M1_MD  M2_BL  M2_MD  P4_BL
    1.00,  0.75,  0.65,  0.60,  0.55,  # M1_BL row
    0.75,  1.00,  0.60,  0.65,  0.60,  # M1_MD row
    0.65,  0.60,  1.00,  0.75,  0.55,  # M2_BL row
    0.60,  0.65,  0.75,  1.00,  0.60,  # M2_MD row
    0.55,  0.60,  0.55,  0.60,  1.00   # P4_BL row
), nrow = 5, ncol = 5)

# Add variable names
rownames(correlation_matrix) <- c("M1_BL", "M1_MD", "M2_BL", "M2_MD", "P4_BL")
colnames(correlation_matrix) <- c("M1_BL", "M1_MD", "M2_BL", "M2_MD", "P4_BL")

# View the matrix
print(correlation_matrix)
```

Understanding the output: - Diagonal = 1.00 (perfect correlation with itself) - Off-diagonal = correlations between variables - Matrix is symmetric (M1_BL M1_MD = M1_MD M1_BL)

Example interpretation: - M1_BL and M1_MD have correlation 0.75 - This means: larger M1 buccolingual → larger M1 mesiodistal (75% correlation)

4 SIMULATION 1: EASY SEPARATION

4.1 Overview

What we are simulating: - **3 species** that are clearly distinct - **25 specimens** per species - **Large differences** between species (easy to tell apart) - **Like:** *Australopithecus africanus* vs. *Paranthropus boisei* vs. *Homo erectus*

4.2 Species Parameters

4.2.1 Species A (LARGE teeth - like P. boisei)

```
# Sample size
n_A <- 25

# Mean tooth sizes (in mm)
mean_A <- c(20, 18, 21, 19, 12)  # Large teeth
names(mean_A) <- c("M1_BL", "M1_MD", "M2_BL", "M2_MD", "P4_BL")

# Standard deviations (variation within species)
sd_A <- c(0.7, 0.6, 0.8, 0.7, 0.5)

# Discrete trait probabilities
# Cusp pattern: 90% Y5, 8% Y4, 2% Plus5
# Hypocone size: 10% small, 10% medium, 80% large
# Cingulum: 10% absent, 20% weak, 70% strong

print("Species A parameters:")
print(mean_A)
print(sd_A)
```

What these numbers mean: - `mean_A`: Average tooth size for Species A - M1 buccolingual = 20mm (very large, like *P. boisei*) - `sd_A`: How much variation within the species - M1_BL has SD = 0.7mm (specimens range roughly 18.6-21.4mm) - Discrete traits mostly primitive/robust (90% Y5, 80% large hypocone)

4.2.2 Generate Species A Data

```
# Step 1: Convert correlation matrix to covariance matrix
# Covariance = SD × Correlation × SD
covariance_A <- diag(sd_A) %*% correlation_matrix %*% diag(sd_A)
```

```r
# Step 2: Generate correlated measurements
# mvrnorm = multivariate normal random numbers
continuous_A <- mvrnorm(n = n_A, mu = mean_A, Sigma = covariance_A)

# Step 3: Convert to data frame and add specimen IDs
data_A <- as.data.frame(continuous_A)
data_A$specimen_id <- sprintf("Species_A_%03d", 1:n_A)
data_A$taxon <- "Species_A"

# Step 4: Add discrete traits
# Sample from categorical distributions with specified probabilities
data_A$cusp_pattern <- sample(
  c("Y5", "Y4", "Plus5"),
  size = n_A,
  replace = TRUE,
  prob = c(0.90, 0.08, 0.02)
)

data_A$hypocone_size <- sample(
  c("small", "medium", "large"),
  size = n_A,
  replace = TRUE,
  prob = c(0.10, 0.10, 0.80)
)

data_A$cingulum <- sample(
  c("absent", "weak", "strong"),
  size = n_A,
  replace = TRUE,
  prob = c(0.10, 0.20, 0.70)
)

# Step 5: Convert discrete variables to factors
data_A$cusp_pattern <- factor(data_A$cusp_pattern,
                              levels = c("Y5", "Y4", "Plus5"))
data_A$hypocone_size <- factor(data_A$hypocone_size,
                              levels = c("small", "medium", "large"))
data_A$cingulum <- factor(data_A$cingulum,
                          levels = c("absent", "weak", "strong"))

# View first few rows
head(data_A)
```

Understanding the code:

Step 1 explanation:

```r
covariance_A <- diag(sd_A) %*% correlation_matrix %*% diag(sd_A)
```

- `diag(sd_A)` creates diagonal matrix of standard deviations
- `%*%` is matrix multiplication
- Formula: Cov = SD × Corr × SD
- This converts correlations (0-1) to covariances (actual units)

Step 2 explanation:

```
continuous_A <- mvrnorm(n = n_A, mu = mean_A, Sigma = covariance_A)
```

- `mvrnorm()` = multivariate normal random generator
- `n = n_A` = how many specimens (25)
- `mu = mean_A` = means for each variable
- `Sigma = covariance_A` = covariance matrix (includes correlations)
- **Result:** 25 rows (specimens) × 5 columns (measurements) with realistic correlations

Step 4 explanation:

```
sample(c("Y5", "Y4", "Plus5"), size = n_A, replace = TRUE, prob = c(0.90, 0.08, 0.02))
```

- `sample()` randomly picks values
- `c("Y5", "Y4", "Plus5")` = possible values
- `size = n_A` = how many to pick (25)
- `replace = TRUE` = can pick same value multiple times
- `prob = c(0.90, 0.08, 0.02)` = probabilities (90% Y5, 8% Y4, 2% Plus5)

4.2.3 Species B (INTERMEDIATE teeth - like Au. afarensis)

```r
# Species B parameters
n_B <- 25
mean_B <- c(15, 14, 16, 15, 10)  # Intermediate size
sd_B <- c(0.6, 0.5, 0.7, 0.6, 0.5)

# Generate data (same process as Species A)
covariance_B <- diag(sd_B) %*% correlation_matrix %*% diag(sd_B)
continuous_B <- mvrnorm(n = n_B, mu = mean_B, Sigma = covariance_B)

data_B <- as.data.frame(continuous_B)
data_B$specimen_id <- sprintf("Species_B_%03d", 1:n_B)
data_B$taxon <- "Species_B"

# Discrete traits - more variable
data_B$cusp_pattern <- sample(c("Y5", "Y4", "Plus5"), n_B, TRUE,
```

```
                                c(0.60, 0.30, 0.10))
data_B$hypocone_size <- sample(c("small", "medium", "large"), n_B, TRUE,
                                c(0.20, 0.60, 0.20))
data_B$cingulum <- sample(c("absent", "weak", "strong"), n_B, TRUE,
                          c(0.20, 0.50, 0.30))

# Convert to factors
data_B$cusp_pattern <- factor(data_B$cusp_pattern,
                                levels = c("Y5", "Y4", "Plus5"))
data_B$hypocone_size <- factor(data_B$hypocone_size,
                                levels = c("small", "medium", "large"))
data_B$cingulum <- factor(data_B$cingulum,
                          levels = c("absent", "weak", "strong"))

head(data_B)
```

Species B characteristics: - **Smaller** than Species A (mean M1_BL = 15mm vs. 20mm) - **More variable** discrete traits (60% Y5 vs. 90% for Species A) - **Intermediate** between large (A) and small (C)

4.2.4 Species C (SMALL teeth - like H. erectus)

```
# Species C parameters
n_C <- 25
mean_C <- c(12, 11, 13, 12, 8)  # Small teeth
sd_C <- c(0.5, 0.4, 0.6, 0.5, 0.4)

# Generate data
covariance_C <- diag(sd_C) %*% correlation_matrix %*% diag(sd_C)
continuous_C <- mvrnorm(n = n_C, mu = mean_C, Sigma = covariance_C)

data_C <- as.data.frame(continuous_C)
data_C$specimen_id <- sprintf("Species_C_%03d", 1:n_C)
data_C$taxon <- "Species_C"

# Discrete traits - derived
data_C$cusp_pattern <- sample(c("Y5", "Y4", "Plus5"), n_C, TRUE,
                                c(0.30, 0.70, 0.00))  # Mostly Y4
data_C$hypocone_size <- sample(c("small", "medium", "large"), n_C, TRUE,
                                c(0.70, 0.20, 0.10))  # Mostly small
data_C$cingulum <- sample(c("absent", "weak", "strong"), n_C, TRUE,
                          c(0.80, 0.15, 0.05))  # Mostly absent
```

```r
# Convert to factors
data_C$cusp_pattern <- factor(data_C$cusp_pattern,
                              levels = c("Y5", "Y4", "Plus5"))
data_C$hypocone_size <- factor(data_C$hypocone_size,
                              levels = c("small", "medium", "large"))
data_C$cingulum <- factor(data_C$cingulum,
                          levels = c("absent", "weak", "strong"))

head(data_C)
```

Species C characteristics: - **Smallest** teeth (M1_BL = 12mm, like *H. erectus*) - **Derived** discrete traits (70% Y4, 70% small hypocone) - **Less variable** than other species (smaller SD)

4.2.5 Combine All Three Species

```r
# Stack all three species into one dataset
sim1_data <- rbind(data_A, data_B, data_C)

# Verify sample sizes
table(sim1_data$taxon)

# Check structure
str(sim1_data)

# Summary statistics by species
aggregate(M1_BL ~ taxon, data = sim1_data, FUN = mean)
aggregate(M1_BL ~ taxon, data = sim1_data, FUN = sd)

# Save the dataset
write.csv(sim1_data, "data/sim1_data.csv", row.names = FALSE)
```

What you should see:

```
Species_A Species_B Species_C
       25        25        25

M1_BL means:
Species_A: ~20mm
Species_B: ~15mm
Species_C: ~12mm

Differences: 5mm and 3mm (very distinct!)
```

5 CALCULATING DISTANCES

5.1 Part 1: Mahalanobis Distance

5.1.1 What is Mahalanobis Distance?

Simple explanation:

Regular Euclidean distance treats all variables as independent:

$$\text{Distance} = \sqrt{(x1-x2)^2 + (y1-y2)^2}$$

Mahalanobis distance accounts for correlations:

$$\text{Distance} = \sqrt{(x-y) \times S^{-1} \times (x-y)}$$

where S^{-1} is the inverse covariance matrix.

Why it matters: - If M1 and M2 are correlated, large differences in BOTH are less surprising - Mahalanobis "rotates" the space to account for this - Result: More accurate distance measurement

---

5.1.2 Calculate Mahalanobis Distance Step-by-Step

```
# Define which columns are continuous measurements
continuous_vars <- c("M1_BL", "M1_MD", "M2_BL", "M2_MD", "P4_BL")

# Extract just the continuous data
continuous_data <- sim1_data[, continuous_vars]

# STEP 1: Center the data (subtract means)
# This moves the "origin" to the center of the data
centered_data <- scale(continuous_data, center = TRUE, scale = FALSE)

# Look at before and after
head(continuous_data)  # Original data
head(centered_data)    # Centered (means = 0)

# STEP 2: Calculate covariance matrix
# This captures correlations between variables
S <- cov(centered_data)
print("Covariance matrix:")
print(round(S, 3))

# STEP 3: Check if matrix is singular
```

```r
# Singular = cannot be inverted (determinant near zero)
det_S <- det(S)
cat("Determinant:", det_S, "
")

# If determinant is very small, add small value to diagonal (ridge)
if (det_S < 1e-10) {
  cat("Matrix near-singular, adding ridge
")
  S <- S + diag(0.001, ncol(S))
}

# STEP 4: Invert the covariance matrix
# This is like "dividing" but for matrices
S_inv <- solve(S)
print("Inverse covariance matrix:")
print(round(S_inv, 3))

# STEP 5: Calculate all pairwise distances
n <- nrow(continuous_data)

# Create empty matrix to store distances
mahal_dist_matrix <- matrix(0, nrow = n, ncol = n)

# Loop through all pairs of specimens
for (i in 1:(n-1)) {
  for (j in (i+1):n) {

    # Get the two specimens
    specimen_i <- as.numeric(continuous_data[i, ])
    specimen_j <- as.numeric(continuous_data[j, ])

    # Calculate difference
    diff <- specimen_i - specimen_j

    # Mahalanobis formula: sqrt(diff^T × S_inv × diff)
    D_squared <- t(diff) %*% S_inv %*% diff
    mahal_dist_matrix[i, j] <- sqrt(D_squared)

    # Matrix is symmetric, so copy to lower triangle
    mahal_dist_matrix[j, i] <- mahal_dist_matrix[i, j]
  }
}

# Convert to distance object
mahal_dist <- as.dist(mahal_dist_matrix)
```

```r
# View first few distances
print("First 10 Mahalanobis distances:")
print(head(as.vector(mahal_dist), 10))
```

Understanding each step:

Step 1: Centering

```
Before: M1_BL values like 20.3, 19.8, 15.2, 12.1
After:  M1_BL values like 4.8, 4.3, -0.3, -3.4 (mean = 0)
Why: Makes calculations simpler, focuses on deviations from mean
```

Step 2: Covariance

```
Covariance matrix shows how variables co-vary:
- Large positive = tend to increase together
- Near zero = independent
- Negative = one increases when other decreases
```

Step 4: Inversion

```
S_inv "undoes" the correlations
Like converting correlated space back to independent axes
Necessary for Mahalanobis formula
```

Step 5: Distance calculation

```
For each pair:
1. Calculate difference: diff = specimen1 - specimen2
2. "Rotate" by covariance: diff^T × S_inv × diff
3. Take square root
Result: Distance accounting for correlations
```

5.1.3 Standardize Mahalanobis to 0-1 Scale

```r
# Find maximum distance
max_mahal <- max(mahal_dist_matrix[mahal_dist_matrix < Inf])

cat("Maximum Mahalanobis distance:", max_mahal, "
")

# Divide all distances by maximum
```

```r
# Now range is 0 (identical) to 1 (most different)
mahal_scaled <- mahal_dist_matrix / max_mahal

cat("Scaled Mahalanobis - Min:", min(mahal_scaled[mahal_scaled > 0]), "
")
cat("Scaled Mahalanobis - Max:", max(mahal_scaled), "
")

# Calculate mean distances within vs. between species
taxa <- sim1_data$taxon

# Within-species distances
within_A <- mean(mahal_scaled[taxa == "Species_A", taxa == "Species_A"][
  upper.tri(mahal_scaled[taxa == "Species_A", taxa == "Species_A"])
])
within_B <- mean(mahal_scaled[taxa == "Species_B", taxa == "Species_B"][
  upper.tri(mahal_scaled[taxa == "Species_B", taxa == "Species_B"])
])
within_C <- mean(mahal_scaled[taxa == "Species_C", taxa == "Species_C"][
  upper.tri(mahal_scaled[taxa == "Species_C", taxa == "Species_C"])
])

# Between-species distances
between_AB <- mean(mahal_scaled[taxa == "Species_A", taxa == "Species_B"])
between_AC <- mean(mahal_scaled[taxa == "Species_A", taxa == "Species_C"])
between_BC <- mean(mahal_scaled[taxa == "Species_B", taxa == "Species_C"])

# Print results
cat("
Mean distances (scaled):
")
cat("Within Species A:", round(within_A, 3), "
")
cat("Within Species B:", round(within_B, 3), "
")
cat("Within Species C:", round(within_C, 3), "
")
cat("
")
cat("Between A-B:", round(between_AB, 3), "
")
cat("Between A-C:", round(between_AC, 3), "
")
cat("Between B-C:", round(between_BC, 3), "
")

# Separation ratios
```

```
cat("
Separation ratios (between / within):
")
cat("A vs B:", round(between_AB / mean(c(within_A, within_B)), 2), "
")
cat("A vs C:", round(between_AC / mean(c(within_A, within_C)), 2), "
")
cat("B vs C:", round(between_BC / mean(c(within_B, within_C)), 2), "
")
```

Why standardize? - Mahalanobis distances can be any positive number (0 to infinity) - Gower distances are always 0 to 1 - To combine them, need same scale - Divide by max → all distances now 0 to 1

What the output means:

```
Within-species: ~0.15-0.25 (specimens within species are similar)
Between-species: ~0.60-0.85 (different species are far apart)
Separation ratio: ~3-4 (between is 3-4× larger than within)

Good separation ratio = >2.0
Excellent separation = >3.0
```

5.2 Part 2: Gower Distance

5.2.1 What is Gower Distance?

For discrete (categorical) variables:

Gower calculates similarity for each trait: - Match (Y5 and Y5) = similarity 1 - Mismatch (Y5 and Y4) = similarity 0

Then averages across all traits and converts to distance:

```
Distance = 1 - average_similarity
```

Example:

```
Specimen 1: Y5 cusp, large hypocone, strong cingulum
Specimen 2: Y5 cusp, medium hypocone, strong cingulum

Cusp: match (1)
Hypocone: mismatch (0)
Cingulum: match (1)

Average similarity = (1 + 0 + 1) / 3 = 0.667
Gower distance = 1 - 0.667 = 0.333
```

5.2.2 Calculate Gower Distance

```r
# Define discrete variables
discrete_vars <- c("cusp_pattern", "hypocone_size", "cingulum")

# Extract discrete data
discrete_data <- sim1_data[, discrete_vars]

# Check structure
str(discrete_data)  # Should all be factors

# Calculate Gower distance
# daisy() with metric="gower" handles categorical data
gower_dist <- daisy(discrete_data, metric = "gower", stand = FALSE)

# Convert to matrix for easier viewing
gower_matrix <- as.matrix(gower_dist)

# View first few
```

```r
print("First 5x5 of Gower distance matrix:")
print(round(gower_matrix[1:5, 1:5], 3))

# Summary statistics
cat("
Gower distance summary:
")
cat("Min:", min(gower_matrix[gower_matrix > 0]), "
")
cat("Max:", max(gower_matrix), "
")
cat("Mean:", mean(gower_matrix[upper.tri(gower_matrix)]), "
")

# Within vs. between
gower_within_A <- mean(gower_matrix[taxa == "Species_A", taxa == "Species_A"][
  upper.tri(gower_matrix[taxa == "Species_A", taxa == "Species_A"])
])
gower_within_B <- mean(gower_matrix[taxa == "Species_B", taxa == "Species_B"][
  upper.tri(gower_matrix[taxa == "Species_B", taxa == "Species_B"])
])
gower_within_C <- mean(gower_matrix[taxa == "Species_C", taxa == "Species_C"][
  upper.tri(gower_matrix[taxa == "Species_C", taxa == "Species_C"])
])

gower_between_AB <- mean(gower_matrix[taxa == "Species_A", taxa == "Species_B"])
gower_between_AC <- mean(gower_matrix[taxa == "Species_A", taxa == "Species_C"])
gower_between_BC <- mean(gower_matrix[taxa == "Species_B", taxa == "Species_C"])

cat("
Gower distances:
")
cat("Within A:", round(gower_within_A, 3), "
")
cat("Within B:", round(gower_within_B, 3), "
")
cat("Within C:", round(gower_within_C, 3), "
")
cat("
")
cat("Between A-B:", round(gower_between_AB, 3), "
")
cat("Between A-C:", round(gower_between_AC, 3), "
")
cat("Between B-C:", round(gower_between_BC, 3), "
")
```

Understanding Gower output:

```
Distance 0.0 = all traits match (identical)
Distance 0.33 = 1 out of 3 traits differ
Distance 0.67 = 2 out of 3 traits differ
Distance 1.0 = all traits differ (completely different)
```

Typical patterns for SIM1:

```
Within-species: ~0.20-0.30 (some variation but mostly similar)
Between-species: ~0.50-0.70 (many trait differences)
```

5.3 Part 3: Combine into Combined Distance

5.3.1 The Combined Formula

```
D_combined =   × D_Mahalanobis(scaled) + (1- ) × D_Gower

where   = 0.65 (from simulations)
```

What this means: - 65% weight to continuous measurements (Mahalanobis) - 35% weight to discrete traits (Gower) - Result: Integrated distance using all data

---

5.3.2 Calculate Combined Distance

```
# Set alpha (weight for continuous data)
alpha <- 0.65

cat("Alpha parameter:", alpha, "
")
cat("This means:
")
cat("  ", alpha * 100, "% weight to continuous measurements
")
cat("  ", (1-alpha) * 100, "% weight to discrete traits

")

# Combine the distances
combined_matrix <- alpha * mahal_scaled + (1 - alpha) * gower_matrix

# Convert to distance object
combined_dist <- as.dist(combined_matrix)

# Summary
cat("Combined distance summary:
")
cat("Min:", min(combined_matrix[combined_matrix > 0]), "
")
cat("Max:", max(combined_matrix), "
")
cat("Mean:", mean(combined_matrix[upper.tri(combined_matrix)]), "
")

# Compare all three metrics side-by-side
comparison_df <- data.frame(
```

```
  Metric = c("Mahalanobis", "Gower", "Combined"),
  Within_Mean = c(
    mean(c(within_A, within_B, within_C)),
    mean(c(gower_within_A, gower_within_B, gower_within_C)),
    mean(combined_matrix[taxa[row(combined_matrix)] == taxa[col(combined_matrix)] &
                         row(combined_matrix) != col(combined_matrix)])
  ),
  Between_Mean = c(
    mean(c(between_AB, between_AC, between_BC)),
    mean(c(gower_between_AB, gower_between_AC, gower_between_BC)),
    mean(combined_matrix[taxa[row(combined_matrix)] != taxa[col(combined_matrix)]])
  )
)

comparison_df$Separation_Ratio <- comparison_df$Between_Mean / comparison_df$Within_Mean

print(comparison_df)
```

Example output interpretation:

```
               Metric  Within  Between  Ratio
1        Mahalanobis   0.180    0.720   4.00
2              Gower   0.250    0.600   2.40
3            Combined   0.205    0.678   3.31

Interpretation:
- All three show clear separation (ratio > 2.0)
- Mahalanobis has highest separation (ratio = 4.0)
- Combined balances both sources (ratio = 3.31)
- Between-group distances 3-4× larger than within-group
```

6 CLASSIFICATION ANALYSIS

6.1 K-Nearest Neighbors with Cross-Validation

6.1.1 What is k-NN?

Simple explanation:

To classify a specimen: 1. Find the k=5 nearest neighbors (shortest distances) 2. See what species they are 3. Majority vote wins

Example:

```
Specimen Xs 5 nearest neighbors:
- Species_A (distance 0.12)
- Species_A (distance 0.15)
- Species_A (distance 0.18)
- Species_B (distance 0.22)
- Species_A (distance 0.24)

Vote: 4 Species_A, 1 Species_B
Prediction: Species_A
Confidence: 80% (4/5)
```

6.1.2 Cross-Validation

Why? - Testing on training data gives falsely high accuracy - Cross-validation uses separate test/train sets

5-fold CV:

```
Split 75 specimens into 5 groups of 15
Round 1: Test group 1, train on groups 2-5
Round 2: Test group 2, train on groups 1,3-5
...
Round 5: Test group 5, train on groups 1-4

Each specimen tested exactly once
```

6.1.3 Run k-NN Classification

```
# Set parameters
k_folds <- 5        # Number of cross-validation folds
k_neighbors <- 5    # Number of neighbors to use
```

```r
# Get true species labels
true_taxa <- sim1_data$taxon
n_specimens <- nrow(sim1_data)

# Create fold assignments (random)
fold_ids <- sample(rep(1:k_folds, length.out = n_specimens))

cat("Fold assignments:
")
table(fold_ids)

# Initialize storage for predictions
predictions <- rep(NA, n_specimens)
posterior_probs <- matrix(NA, nrow = n_specimens, ncol = 3)
colnames(posterior_probs) <- c("Species_A", "Species_B", "Species_C")

# Cross-validation loop
cat("
Running", k_folds, "-fold cross-validation...
")

for (fold in 1:k_folds) {

  cat("  Fold", fold, "of", k_folds, "
")

  # Split into test and train
  test_indices <- which(fold_ids == fold)
  train_indices <- which(fold_ids != fold)

  cat("    Test specimens:", length(test_indices), "
")
  cat("    Train specimens:", length(train_indices), "
")

  # For each test specimen
  for (i in test_indices) {

    # Get distances to all training specimens
    dists_to_train <- combined_matrix[i, train_indices]

    # Find k nearest neighbors
    nearest_indices <- order(dists_to_train)[1:k_neighbors]
    nearest_specimens <- train_indices[nearest_indices]

    # Get their species labels
    nearest_taxa <- true_taxa[nearest_specimens]
```

```
    # Count votes
    vote_table <- table(nearest_taxa)

    # Posterior probabilities (proportion of each species)
    for (sp in names(vote_table)) {
      posterior_probs[i, sp] <- vote_table[sp] / k_neighbors
    }

    # Prediction: majority vote
    predictions[i] <- names(which.max(vote_table))
  }
}

# Calculate accuracy
accuracy <- mean(predictions == true_taxa)

cat("
=== CLASSIFICATION RESULTS ===
")
cat("Overall accuracy:", round(accuracy * 100, 1), "%
")

# Confusion matrix
confusion <- table(Predicted = predictions, True = true_taxa)
print(confusion)

# Per-species accuracy
for (sp in levels(true_taxa)) {
  sp_correct <- sum(predictions[true_taxa == sp] == sp)
  sp_total <- sum(true_taxa == sp)
  sp_acc <- sp_correct / sp_total
  cat("
", sp, "accuracy:", round(sp_acc * 100, 1), "%
")
}

# Mean confidence (average of max posterior probability)
mean_confidence <- mean(apply(posterior_probs, 1, max, na.rm = TRUE))
cat("
Mean confidence:", round(mean_confidence * 100, 1), "%
")

# Save results
classification_results <- list(
  accuracy = accuracy,
  predictions = predictions,
  posterior_probs = posterior_probs,
```

```
  confusion_matrix = confusion,
  mean_confidence = mean_confidence
)

saveRDS(classification_results, "results/sim1_classification.rds")
```

Understanding the output:

Confusion matrix:

```
             True
Predicted    Species_A  Species_B  Species_C
  Species_A      24          1          0
  Species_B       1         23          0
  Species_C       0          1         25

Reading: Diagonal = correct, off-diagonal = errors
- 24/25 Species_A correctly classified (96%)
- 23/25 Species_B correctly classified (92%)
- 25/25 Species_C correctly classified (100%)
```

Expected for SIM1:

```
Accuracy: >95% (very easy to classify)
Confidence: >90% (very certain predictions)
Few misclassifications
```

7 CLUSTERING ANALYSIS

7.1 PAM Clustering with Silhouette

7.1.1 What is PAM?

PAM = Partitioning Around Medoids

Like k-means but works with distance matrices: 1. Choose k medoids (representative specimens) 2. Assign each specimen to nearest medoid 3. Repeat until stable

Silhouette score measures cluster quality: - +1: Perfectly clustered - 0: On the border between clusters - -1: Assigned to wrong cluster

Average silhouette >0.70 = strong clusters

7.1.2 Try Different Values of k

```
# Try k = 2, 3, 4, 5, 6 clusters
max_k <- 6

# Storage
silhouette_scores <- numeric(max_k - 1)
pam_models <- list()

cat("Testing different numbers of clusters...
")

for (k in 2:max_k) {

  cat("
  k =", k, "clusters
")

  # Run PAM clustering
  pam_result <- pam(combined_dist, k = k, diss = TRUE)

  # Store model
  pam_models[[k]] <- pam_result

  # Extract silhouette score
  sil_score <- pam_result$silinfo$avg.width
  silhouette_scores[k - 1] <- sil_score

  cat("    Silhouette:", round(sil_score, 3), "
```

```
")
}

# Find optimal k (highest silhouette)
optimal_k <- which.max(silhouette_scores) + 1
best_silhouette <- max(silhouette_scores)

cat("
=== CLUSTERING RESULTS ===
")
cat("Optimal k:", optimal_k, "
")
cat("Best silhouette:", round(best_silhouette, 3), "
")
cat("True k:", length(unique(true_taxa)), "
")

# Get cluster assignments for optimal k
best_pam <- pam_models[[optimal_k]]
cluster_assignments <- best_pam$clustering

# Compare clusters to true species
comparison_table <- table(Cluster = cluster_assignments, True = true_taxa)
print(comparison_table)

# Plot silhouette scores
plot_data <- data.frame(
  k = 2:max_k,
  Silhouette = silhouette_scores
)

ggplot(plot_data, aes(x = k, y = Silhouette)) +
  geom_line(linewidth = 1.5, color = "steelblue") +
  geom_point(size = 4, color = "steelblue") +
  geom_vline(xintercept = optimal_k, linetype = "dashed", color = "red") +
  geom_hline(yintercept = 0.60, linetype = "dashed", color = "orange") +
  annotate("text", x = optimal_k, y = 0.55,
           label = paste("Optimal k =", optimal_k), color = "red") +
  annotate("text", x = 5, y = 0.62,
           label = "Strong clusters (>0.60)", color = "orange") +
  labs(
    title = "Silhouette Scores for Different k",
    subtitle = "SIM1: Easy Separation",
    x = "Number of Clusters (k)",
    y = "Average Silhouette Score"
  ) +
  theme_minimal(base_size = 14)
```

```r
ggsave("figures/sim1_silhouette_plot.png", width = 8, height = 6, dpi = 300)

# Save results
clustering_results <- list(
  optimal_k = optimal_k,
  true_k = length(unique(true_taxa)),
  best_silhouette = best_silhouette,
  silhouette_scores = silhouette_scores,
  cluster_assignments = cluster_assignments
)

saveRDS(clustering_results, "results/sim1_clustering.rds")
```

Understanding the output:

For SIM1 (easy separation):

```
Expected optimal k: 3 (matches true k)
Expected silhouette: >0.75 (strong clusters)

If clustering finds wrong k:
- k < 3: Under-splitting (lumping species)
- k > 3: Over-splitting (finding structure that isnt there)
```

Silhouette interpretation:

```
>0.70: Strong, well-separated clusters
0.50-0.70: Reasonable structure
0.25-0.50: Weak structure
<0.25: No substantial structure (possible oversplitting)
```

8 VISUALIZATION

8.1 PCA Morphospace Plot

8.1.1 What is PCA?

Principal Component Analysis: - Reduces 5 measurements to 2 dimensions - PC1 captures most variation (usually size) - PC2 captures second-most variation (usually shape) - Can visualize in 2D plot

---

8.1.2 Calculate PCA

```
# Run PCA on continuous measurements
pca_result <- prcomp(continuous_data, scale. = TRUE)

# Extract PC scores
pca_scores <- as.data.frame(pca_result$x)
pca_scores$taxon <- true_taxa
pca_scores$cluster <- factor(cluster_assignments)
pca_scores$predicted <- predictions

# Variance explained
variance_explained <- summary(pca_result)$importance[2, ] * 100

cat("Variance explained by each PC:
")
print(round(variance_explained[1:3], 1))

cat("
PC1 explains", round(variance_explained[1], 1), "% of variation
")
cat("PC2 explains", round(variance_explained[2], 1), "% of variation
")
cat("Together:", round(sum(variance_explained[1:2]), 1), "%
")

# Loadings (which variables contribute to each PC)
cat("
PC1 loadings (what PC1 represents):
")
print(round(pca_result$rotation[, 1], 3))

cat("
PC2 loadings:
```

```
")
print(round(pca_result$rotation[, 2], 3))
```

Understanding loadings:

```
PC1 loadings all positive → PC1 = overall size
High PC1 = large teeth, Low PC1 = small teeth

PC2 loadings mixed signs → PC2 = shape differences
Might represent M1 vs M2 differences, etc.
```

8.1.3 Create Morphospace Plot

```
# Create plot with ellipses
ggplot(pca_scores, aes(x = PC1, y = PC2, color = taxon)) +

  # Points
  geom_point(size = 3, alpha = 0.7) +

  # 95% confidence ellipses
  stat_ellipse(level = 0.95, linewidth = 1) +

  # Colors
  scale_color_viridis_d(option = "D", end = 0.9,
                        name = "Species") +

  # Labels
  labs(
    title = "SIM1: Morphospace Analysis (Easy Separation)",
    subtitle = paste0("Accuracy: ", round(accuracy * 100, 1),
                      "% | Silhouette: ", round(best_silhouette, 3)),
    x = paste0("PC1 (", round(variance_explained[1], 1), "%)"),
    y = paste0("PC2 (", round(variance_explained[2], 1), "%)")
  ) +

  # Theme
  theme_bw(base_size = 14) +
  theme(
    legend.position = "bottom",
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 12)
  )

ggsave("figures/sim1_morphospace.png", width = 10, height = 8, dpi = 300)
```

What to look for: - **Clear separation**: Ellipses dont overlap much - **Few misclassifications**: Points match their ellipse color - **Good clustering**: Clusters match species

For SIM1: - Expect **clear separation** (no overlap) - Expect **high accuracy** (>95%) - Expect **tight clusters** (small ellipses)

9 TAXONOMIC DECISION

9.1 Apply Decision Criteria

9.1.1 Decision Thresholds (from Aim 2 simulations)

```
# Define thresholds
thresholds <- list(
  mahalanobis = 4.0,      # Species threshold
  accuracy = 0.80,        # 80% classification
  silhouette = 0.60,      # Good cluster quality
  confidence = 0.85       # High posterior probability
)

cat("Decision thresholds:
")
cat("  Mahalanobis D² > 4.0 → Distinct species
")
cat("  Accuracy > 80% → Reliably diagnosable
")
cat("  Silhouette > 0.60 → Strong clusters
")
cat("  Confidence > 85% → High certainty

")

# Calculate metrics
mean_mahal_scaled <- mean(mahal_scaled[upper.tri(mahal_scaled)])

# Count how many criteria met
criteria_met <- 0

if (mean_mahal_scaled > (thresholds$mahalanobis / 10)) criteria_met <- criteria_met + 1
if (accuracy > thresholds$accuracy) criteria_met <- criteria_met + 1
if (best_silhouette > thresholds$silhouette) criteria_met <- criteria_met + 1
if (mean_confidence > thresholds$confidence) criteria_met <- criteria_met + 1

cat("=== DECISION SUMMARY ===
")
cat("Criteria met:", criteria_met, "out of 4

")

# Make decision
if (criteria_met >= 3) {
  decision <- "RECOGNIZE AS DISTINCT SPECIES"
  confidence_level <- "HIGH"
```

```
} else if (criteria_met >= 2) {
  decision <- "TENTATIVELY DISTINCT"
  confidence_level <- "MODERATE"
} else {
  decision <- "INSUFFICIENT EVIDENCE"
  confidence_level <- "LOW"
}

cat("DECISION:", decision, "
")
cat("CONFIDENCE:", confidence_level, "

")

# Summary report
cat("=== SIM1 FINAL REPORT ===
")
cat("Scenario: Easy Separation (3 distinct species)
")
cat("Sample size:", n_specimens, "(", n_A, "per species )
")
cat("
Results:
")
cat("  Mean Mahalanobis D² (scaled):", round(mean_mahal_scaled, 3), "
")
cat("  Classification accuracy:", round(accuracy * 100, 1), "%
")
cat("  Optimal k found:", optimal_k, "(true k =", length(unique(true_taxa)), ")
")
cat("  Silhouette score:", round(best_silhouette, 3), "
")
cat("  Mean confidence:", round(mean_confidence * 100, 1), "%
")
cat("
Decision:", decision, "
")
cat("Confidence:", confidence_level, "
")
cat("
Interpretation:
")
cat("All three species are clearly morphologically distinct and
")
cat("can be reliably diagnosed. Method performs excellently with
")
cat("well-separated species.
```

```
")

# Save report
report <- list(
  scenario = "SIM1: Easy Separation",
  n = n_specimens,
  metrics = list(
    mahalanobis = mean_mahal_scaled,
    accuracy = accuracy,
    optimal_k = optimal_k,
    true_k = length(unique(true_taxa)),
    silhouette = best_silhouette,
    confidence = mean_confidence
  ),
  decision = decision,
  confidence_level = confidence_level
)

saveRDS(report, "results/sim1_report.rds")
```

Expected for SIM1:

```
  Mahalanobis: High (well separated)
  Accuracy: >95% (well above 80%)
  Silhouette: >0.75 (well above 0.60)
  Confidence: >90% (well above 85%)

Criteria met: 4/4
Decision: RECOGNIZE AS DISTINCT SPECIES
Confidence: HIGH

This is exactly what we want for clearly distinct species!
```

10 SIMULATION 2: MODERATE SEPARATION

10.1 Overview

What changes from SIM1: - **Smaller gaps** between species (1.5mm instead of 5mm) - **More overlap** in morphospace (~20%) - **More similar** discrete traits - **Like:** *Au. afarensis* vs. *Au. africanus* vs. *Au. garhi* (realistic!)

10.2 Species Parameters

10.2.1 Key Differences from SIM1

```
# Species A: Still largest but closer to B
mean_A_sim2 <- c(16.5, 15.0, 17.0, 16.0, 11.0)  # Reduced from 20mm
sd_A_sim2 <- c(0.9, 0.8, 1.0, 0.9, 0.7)         # More variable

# Species B: Intermediate
mean_B_sim2 <- c(15.0, 13.5, 15.5, 14.5, 10.0)
sd_B_sim2 <- c(0.9, 0.8, 1.0, 0.9, 0.7)

# Species C: Smallest but closer to B
mean_C_sim2 <- c(14.0, 12.5, 14.5, 13.5, 9.5)   # Increased from 12mm
sd_C_sim2 <- c(0.9, 0.8, 1.0, 0.9, 0.7)

# Calculate gaps
gap_AB <- mean_A_sim2[1] - mean_B_sim2[1]  # 1.5mm
gap_BC <- mean_B_sim2[1] - mean_C_sim2[1]  # 1.0mm

cat("Species separations:
")
cat("  A vs B gap:", gap_AB, "mm (cf. 5mm in SIM1)
")
cat("  B vs C gap:", gap_BC, "mm (cf. 3mm in SIM1)
")
cat("  More overlap expected!

")

# Discrete traits also more similar
# Species A: 70% Y5 (vs. 90% in SIM1)
# Species B: 50% Y5 (vs. 60% in SIM1)
# Species C: 40% Y5 (vs. 30% in SIM1)
cat("Discrete trait overlap:
")
cat("  More similar cusp patterns across species
")
```

```
cat("  Harder to distinguish based on discrete traits alone
")
```

Why this matters:

```
SIM1: 5mm gap = very easy (like Au. africanus vs P. boisei)
SIM2: 1.5mm gap = realistic (like Au. afarensis vs Au. africanus)

Expected results:
- Lower accuracy: ~85% (vs. >95% in SIM1)
- More misclassifications
- Lower silhouette: ~0.60 (vs. >0.75 in SIM1)
- This tests method on realistic scenarios!
```

10.2.2 Generate SIM2 Data

```
# Use same process as SIM1, just different parameters

# Species A
n <- 20  # Slightly smaller sample (more realistic)
covariance_A <- diag(sd_A_sim2) %*% correlation_matrix %*% diag(sd_A_sim2)
continuous_A <- mvrnorm(n = n, mu = mean_A_sim2, Sigma = covariance_A)

data_A <- as.data.frame(continuous_A)
data_A$specimen_id <- sprintf("Species_A_%03d", 1:n)
data_A$taxon <- "Species_A"

data_A$cusp_pattern <- sample(c("Y5", "Y4", "Plus5"), n, TRUE, c(0.70, 0.25, 0.05))
data_A$hypocone_size <- sample(c("small", "medium", "large"), n, TRUE, c(0.20, 0.20, 0.60))
data_A$cingulum <- sample(c("absent", "weak", "strong"), n, TRUE, c(0.20, 0.30, 0.50))

data_A$cusp_pattern <- factor(data_A$cusp_pattern, levels = c("Y5", "Y4", "Plus5"))
data_A$hypocone_size <- factor(data_A$hypocone_size, levels = c("small", "medium", "large"))
data_A$cingulum <- factor(data_A$cingulum, levels = c("absent", "weak", "strong"))

# Species B (repeat same structure)
continuous_B <- mvrnorm(n = n, mu = mean_B_sim2,
                        Sigma = diag(sd_B_sim2) %*% correlation_matrix %*% diag(sd_B_sim2))
data_B <- as.data.frame(continuous_B)
data_B$specimen_id <- sprintf("Species_B_%03d", 1:n)
data_B$taxon <- "Species_B"
data_B$cusp_pattern <- sample(c("Y5", "Y4", "Plus5"), n, TRUE, c(0.50, 0.40, 0.10))
data_B$hypocone_size <- sample(c("small", "medium", "large"), n, TRUE, c(0.30, 0.50, 0.20))
```

```r
data_B$cingulum <- sample(c("absent", "weak", "strong"), n, TRUE, c(0.30, 0.50, 0.20))
data_B$cusp_pattern <- factor(data_B$cusp_pattern, levels = c("Y5", "Y4", "Plus5"))
data_B$hypocone_size <- factor(data_B$hypocone_size, levels = c("small", "medium", "large"))
data_B$cingulum <- factor(data_B$cingulum, levels = c("absent", "weak", "strong"))

# Species C
continuous_C <- mvrnorm(n = n, mu = mean_C_sim2,
                        Sigma = diag(sd_C_sim2) %*% correlation_matrix %*% diag(sd_C_sim2))
data_C <- as.data.frame(continuous_C)
data_C$specimen_id <- sprintf("Species_C_%03d", 1:n)
data_C$taxon <- "Species_C"
data_C$cusp_pattern <- sample(c("Y5", "Y4", "Plus5"), n, TRUE, c(0.40, 0.55, 0.05))
data_C$hypocone_size <- sample(c("small", "medium", "large"), n, TRUE, c(0.50, 0.40, 0.10))
data_C$cingulum <- sample(c("absent", "weak", "strong"), n, TRUE, c(0.50, 0.40, 0.10))
data_C$cusp_pattern <- factor(data_C$cusp_pattern, levels = c("Y5", "Y4", "Plus5"))
data_C$hypocone_size <- factor(data_C$hypocone_size, levels = c("small", "medium", "large"))
data_C$cingulum <- factor(data_C$cingulum, levels = c("absent", "weak", "strong"))

# Combine
sim2_data <- rbind(data_A, data_B, data_C)

# Save
write.csv(sim2_data, "data/sim2_data.csv", row.names = FALSE)

cat("SIM2 data generated:
")
cat("  Total specimens:", nrow(sim2_data), "
")
cat("  Per species:", n, "
")
cat("  Expected accuracy: 80-90% (vs. >95% in SIM1)
")
```

10.2.3 Run Complete Analysis on SIM2

```r
# Follow exact same steps as SIM1:
# 1. Calculate Mahalanobis distance
# 2. Calculate Gower distance
# 3. Combine into combined distance
# 4. Run classification
# 5. Run clustering
# 6. Create plots
# 7. Make decision
```

```
# [Insert same code structure as SIM1, just using sim2_data]

# Expected results:
# Accuracy: 85-88%
# Silhouette: 0.60-0.65
# Some misclassifications visible in morphospace
# Still above thresholds → species distinct
```

Key learning points for SIM2:

1. Lower accuracy (85%) is still good for realistic data
2. Some overlap expected and OK
3. Method still works with moderate separation
4. This represents real Australopithecus-like scenarios

11 REMAINING SIMULATIONS (OVERVIEW)

11.1 SIM3: Oversplit Detection

Setup: - Generate **ONE species** (n=60) - Artificially label as **three "taxa"** (20 each) - Add tiny site effects (~0.3mm)

Expected: - Accuracy: <70% (cant tell them apart) - Silhouette: <0.40 (weak structure) - **This is how we detect oversplitting!**

Code structure: Same as SIM1/SIM2, just:

```
# Generate one species
true_species <- mvrnorm(60, mu = c(15, 14, 16, 15, 10),
                        Sigma = ..., high_SD)

# Artificially split
data$taxon <- rep(c("Taxon_A", "Taxon_B", "Taxon_C"), each = 20)

# Expect: Method shows uncertainty!
```

11.2 SIM4: Chronospecies

Setup: - Generate **five time periods** - Morphology **increases linearly** with time (0.4mm per period) - Discrete traits **evolve** (Y5 → Y4 transition)

Analysis additions:

```
# Add time variable
data$time_period <- rep(1:5, each = 15)

# Fit hierarchical model
library(lme4)
model <- lmer(M1_BL ~ time_period + (1|taxon), data = data)

# Extract variance components
temporal_variance <- ...  # Should be <30%

# Test discrete evolution
chisq.test(table(data$cusp_pattern, data$time_period))
# Should be significant (p < 0.05)
```

Expected: - Temporal variance: 18% (below 30% threshold) - Linear trend: $R^2 > 0.75$ - **Conclusion: ONE evolving lineage (chronospecies)**

11.3 SIM5: Geographic Variation

Setup: - Generate **three regions** - Slight size differences by latitude - **One species**, regional variation

Analysis additions:

```
# Add geographic data
data$region <- rep(c("Region_1", "Region_2", "Region_3"), each = 20)
data$latitude <- rep(c(0, 15, 30), each = 20)

# Fit hierarchical model
model <- lmer(M1_BL ~ (1|region), data = data)

# Extract ICC (intraclass correlation)
geographic_variance <- ...  # Should be <15%

# Test for clinal variation
cor.test(data$M1_BL, data$latitude)
# Should be significant
```

Expected: - Geographic variance: 12% (below 15% threshold) - Clinal pattern: $r = 0.60$, $p < 0.001$ - **Conclusion: Single widespread species**

12 COMPLETE WORKFLOW SUMMARY

12.1 Step-by-Step Checklist

For each simulation, follow this workflow:

12.1.1 Step 1: Generate Data

- ☐ Set species parameters (means, SDs)
- ☐ Generate correlated continuous data (mvrnorm)
- ☐ Add discrete characters (sample with probabilities)
- ☐ Combine species into one dataset
- ☐ Save as CSV

12.1.2 Step 2: Calculate Distances

- ☐ Mahalanobis distance (continuous)
- ☐ Standardize to 0-1 scale
- ☐ Gower distance (discrete)
- ☐ Combine with = 0.65

12.1.3 Step 3: Classification

- ☐ k-NN with 5-fold cross-validation
- ☐ Calculate accuracy
- ☐ Create confusion matrix
- ☐ Calculate mean confidence

12.1.4 Step 4: Clustering

- ☐ Try k = 2 to 6
- ☐ Calculate silhouette scores
- ☐ Find optimal k
- ☐ Compare to true k

12.1.5 Step 5: Visualization

- ☐ Run PCA
- ☐ Create morphospace plot
- ☐ Add 95% ellipses
- ☐ Mark misclassifications

12.1.6 Step 6: Special Analyses

- ☐ SIM4: Temporal variance partitioning
- ☐ SIM5: Geographic variance partitioning
- ☐ Hierarchical models if applicable

12.1.7 Step 7: Decision

- ☐ Check all criteria (D^2, accuracy, silhouette, confidence)
- ☐ Count criteria met
- ☐ Make taxonomic decision
- ☐ Assign confidence level
- ☐ Write interpretation

12.1.8 Step 8: Save Everything

- ☐ Data: `data/simX_data.csv`
- ☐ Results: `results/simX_*.rds`
- ☐ Figures: `figures/simX_*.png`
- ☐ Report: `results/simX_report.rds`

13 PRACTICE EXERCISES

13.1 Exercise 1: Modify SIM1

Task: Change Species A to be SMALLER instead of LARGER

Steps: 1. Change `mean_A` to `c(12, 11, 13, 12, 8)` (smallest) 2. Keep Species B and C the same 3. Run complete analysis 4. Does it still find 3 species?

Expected: Yes! Order doesnt matter, separation does.

13.2 Exercise 2: Test Different Alpha Values

Task: See how affects results

Steps: 1. Run SIM2 with = 0.5 (equal weighting) 2. Run SIM2 with = 0.8 (more continuous weight) 3. Run SIM2 with = 0.3 (more discrete weight) 4. Compare accuracies

Expected: = 0.65 should be best (from optimization)

13.3 Exercise 3: Investigate Misclassifications

Task: Find which specimens are misclassified and why

Steps:

```
# Find misclassified specimens
misclassified <- which(predictions != true_taxa)

cat("Misclassified specimens:
")
for (i in misclassified) {
  cat("
Specimen:", sim1_data$specimen_id[i], "
")
  cat("  True:", true_taxa[i], "
")
  cat("  Predicted:", predictions[i], "
")
  cat("  Measurements:", as.numeric(sim1_data[i, continuous_vars]), "
")
  cat("  Posteriors:", round(posterior_probs[i, ], 3), "
")
}
```

Questions to explore: - Are they borderline specimens (low confidence)? - Do they have unusual measurements? - Are they near cluster boundaries in PCA?

13.4 Exercise 4: Create Your Own Simulation

Task: Design a custom scenario

Ideas: - Two very similar species (test lower threshold) - Four species with different patterns of separation - Sexual dimorphism (males larger than females) - Measurement error (add random noise)

Template:

```
# Your custom parameters
n_custom <- 30
mean_custom <- c(?, ?, ?, ?, ?)
sd_custom <- c(?, ?, ?, ?, ?)

# Generate data
# ... follow SIM1 template ...

# Run analysis
# ... follow standard workflow ...

# Interpret results
# ... apply decision criteria ...
```

14 TROUBLESHOOTING

14.1 Common Issues and Solutions

14.1.1 Issue 1: Singular Matrix Error

Error: `Error in solve.default(S): system is computationally singular`

Cause: Covariance matrix cannot be inverted (variables too correlated)

Solution:

```
# Add small value to diagonal
if (det(S) < 1e-10) {
  S <- S + diag(0.001, ncol(S))
}
```

14.1.2 Issue 2: Low Accuracy for All

Symptom: Even SIM1 shows <80% accuracy

Possible causes: 1. Sample size too small 2. Wrong variables used 3. Bug in code

Check:

```
# Verify data was generated correctly
aggregate(M1_BL ~ taxon, data = sim1_data, mean)
# Should show clear differences (20, 15, 12)

# Check if distances calculated
summary(as.vector(combined_dist))
# Should show range 0 to ~1
```

14.1.3 Issue 3: Wrong Optimal k

Symptom: Clustering finds k 3 for SIM1

Causes: - Too little separation - Too much noise - Algorithm issue

Solutions:

```
# Try different clustering method
hc <- hclust(combined_dist, method = "ward.D2")
plot(hc)  # Visual inspection
```

```
# Check silhouette scores
print(silhouette_scores)
# Are they all low? (<0.5)
```

15 CONCLUSION

15.1 What Youve Learned

By working through this guide, you now understand:

1. **How to generate realistic morphological data**
    - Correlated continuous measurements
    - Discrete categorical traits
    - Species-specific parameters
2. **How distances work**
    - Mahalanobis: Accounts for correlations
    - Gower: Handles categorical data
    - Combined: Combines both optimally
3. **How to classify specimens**
    - k-Nearest neighbors algorithm
    - Cross-validation prevents overfitting
    - Posterior probabilities quantify uncertainty
4. **How to cluster data**
    - PAM finds natural groups
    - Silhouette scores assess quality
    - Optimal k selection
5. **How to make taxonomic decisions**
    - Multiple criteria (D^2, accuracy, silhouette)
    - Thresholds from simulations
    - Uncertainty quantification

15.2 Next Steps

15.2.1 Practice with the Code

1. Run each simulation completely
2. Modify parameters and observe effects
3. Try the practice exercises
4. Create your own scenarios

15.2.2 Apply to Real Data

Once comfortable with simulations: 1. Load your Australopithecus data 2. Follow same workflow 3. Compare results across species pairs 4. Make taxonomic decisions

15.2.3 Prepare Proposal Figures

The code generates all figures you need: - Morphospace plots (PCA) - Silhouette curves - Confusion matrices - Variance partitioning

15.3 Additional Resources

R Documentation: - `?mvrnorm` - Multivariate normal generation - `?daisy` - Gower distance - `?pam` - PAM clustering - `?lmer` - Hierarchical models

Statistical Concepts: - Mahalanobis distance - Cross-validation - Silhouette analysis - Variance partitioning

---

You now have all the code and knowledge to complete your proposal analysis!

Good luck with your practice and implementation!