Hyperspectral Data Analysis Tutorial: From Spectral Reflectance to Plant Classification

Giorgi Kozhoridze

2025-10-24

Table of contents

1 Overview

This tutorial demonstrates how to analyze hyperspectral data for plant classification and biomarker estimation. We'll work with spectral reflectance data from plant samples collected in different months (March and July) across various plant groups and classes. The analysis includes data preprocessing, visualization, statistical testing, and machine learning classification using both raw spectral data and derived vegetation indices.

2 Prerequisites

Before starting, you'll need to install and load several R packages for data manipulation, visualization, and machine learning.

```
# Install required packages (uncomment if needed)
# install.packages(c("tidyverse", "ggpubr", "dplyr", "caret", "ggplot2",
#                    "patchwork", "ggforce", "pls"))
#
# # Install BiocManager and mixOmics for advanced multivariate analysis
# install.packages("BiocManager")
# BiocManager::install("mixOmics")

# Load libraries
library(tidyverse)
library(ggpubr)
library(dplyr)
library(caret)
library(ggplot2)
library(patchwork)
library(ggforce)
library(pls)
library(mixOmics)
```

3 Part 1: Data Preparation and Exploration

3.1 Loading and Inspecting the Data

First, we load our hyperspectral dataset and examine its structure:

```
# Load the hyperspectral data
data <- read.csv("SamplesForWorkshop.csv")

# Inspect the basic structure
head(data[,1:21])
```

```
  Month     Group SampleN Class       Chla       Chlb          C    Antho
1 March Deciduous     101   SP1 0.04112532 0.01990401 0.02018104 0.001240
2 March Deciduous     103   SP1 0.03786589 0.01835909 0.01925522 0.000829
3 March Deciduous     106   SP1 0.06864117 0.03369560 0.02865739 0.000932
4 March Deciduous     107   SP1 0.09533804 0.04758828 0.04024224 0.000903
5 March Deciduous     108   SP1 0.05198037 0.02378042 0.02842488 0.000441
6 March Deciduous     109   SP1 0.03009055 0.01016909 0.01941967 0.000037
   Cellulose     Wax       X400       X401       X402       X403       X404
1         NA      NA 0.01935004 0.01953985 0.01971527 0.01990466 0.02013160
2 0.02858881 0.00138 0.02714292 0.02732804 0.02751218 0.02771258 0.02794536
3         NA      NA 0.03921675 0.03947513 0.03973155 0.04001420 0.04034739
4 0.02790119 0.00132 0.07315494 0.07358253 0.07399735 0.07442525 0.07488811
5 0.02567766 0.00159 0.04725425 0.04754305 0.04782556 0.04812899 0.04847630
6         NA      NA 0.12793647 0.12843469 0.12890088 0.12936994 0.12988197
```

```
        X405       X406       X407       X408       X409       X410
1 0.02035488 0.02059546 0.02085226 0.02109311 0.02135383 0.02163444
2 0.02818707 0.02844890 0.02872564 0.02899116 0.02927687 0.02958125
3 0.04068215 0.04104167 0.04142650 0.04180553 0.04220256 0.04262228
4 0.07534337 0.07581604 0.07630594 0.07677775 0.07726508 0.07776967
5 0.04883025 0.04920400 0.04959861 0.04999560 0.05041397 0.05085308
6 0.13043007 0.13095214 0.13144691 0.13196342 0.13249848 0.13304018
```

ℹ Dataset Structure

The dataset contains:

- **Month**: Sampling time (March, July)
- **Group**: Plant functional groups (Deciduous, Evergreen)

- **Class**: Plant species (SP0, SP1, SP2)
- **Biomarkers**: Chemical measurements in leaves: Chlorophylus A, Chlorophylus B, Carotenoids, Anthocyanins, Cellulose, Wax)
- **Spectral bands**: Reflectance values at different wavelengths (X400, X401, …, X2400)

For more details about the dataset, including measurement units, see Kozhoridze et al. (2016).

3.2 Data Type Conversion

Next, we convert categorical variables to factors and identify spectral columns:

```r
# Convert categorical variables to factors
data$Month <- factor(data$Month, levels = c("March", "July"))
data$Group <- as.factor(data$Group)
data$Class <- as.factor(data$Class)

# Identify spectral band columns (wavelengths from 400-2400 nm)
spectral_cols <- colnames(data)[grepl("^X\\d+$", colnames(data))]
cat("Number of spectral band columns found:", length(spectral_cols), "\n")
```

```
Number of spectral band columns found: 1753
```

! Why Factor Conversion Matters

Converting to factors ensures proper statistical analysis and visualization. The spectral columns represent reflectance measurements at specific wavelengths, forming the hyperspectral signature of each sample.

3.3 Creating Long Format for Analysis

Transform the wide spectral data into long format for easier plotting and analysis:

```r
# Reshape spectral data to long format
data_long <- data %>%
  pivot_longer(cols = all_of(spectral_cols),
               names_to = "Wavelength",
               values_to = "Reflectance") %>%
  mutate(Wavelength = as.numeric(sub("X", "", Wavelength)))

# Check the data structure
print(table(data$Class, data$Group, data$Month))
```

```
, ,  = March


      Deciduous Evergreen
  SP0         0         6
  SP1         6         0
  SP2         3         0

, ,  = July


      Deciduous Evergreen
  SP0         0         6
  SP1         6         0
  SP2         3         0
```

4 Part 2: Biomarker Analysis and Visualization

4.1 Comparing Biomarkers Across Groups

Create comprehensive visualizations to compare biomarker levels across different plant species and traits:

```r
# Prepare data for biomarker comparison
df_long <- data %>%
  pivot_longer(cols = c(Chla, Chlb, C, Antho, Cellulose, Wax),
               names_to = "Biomarker",
               values_to = "Value")

# Create boxplots comparing biomarkers
ggplot(df_long, aes(x = Group, y = Value, fill = Class)) +
  geom_boxplot() +
  facet_grid(Biomarker ~ Month, scales = "free_y") +
  labs(title = "Comparison of Biomarkers by Group and Class",
       y = "Value", x = "Group") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

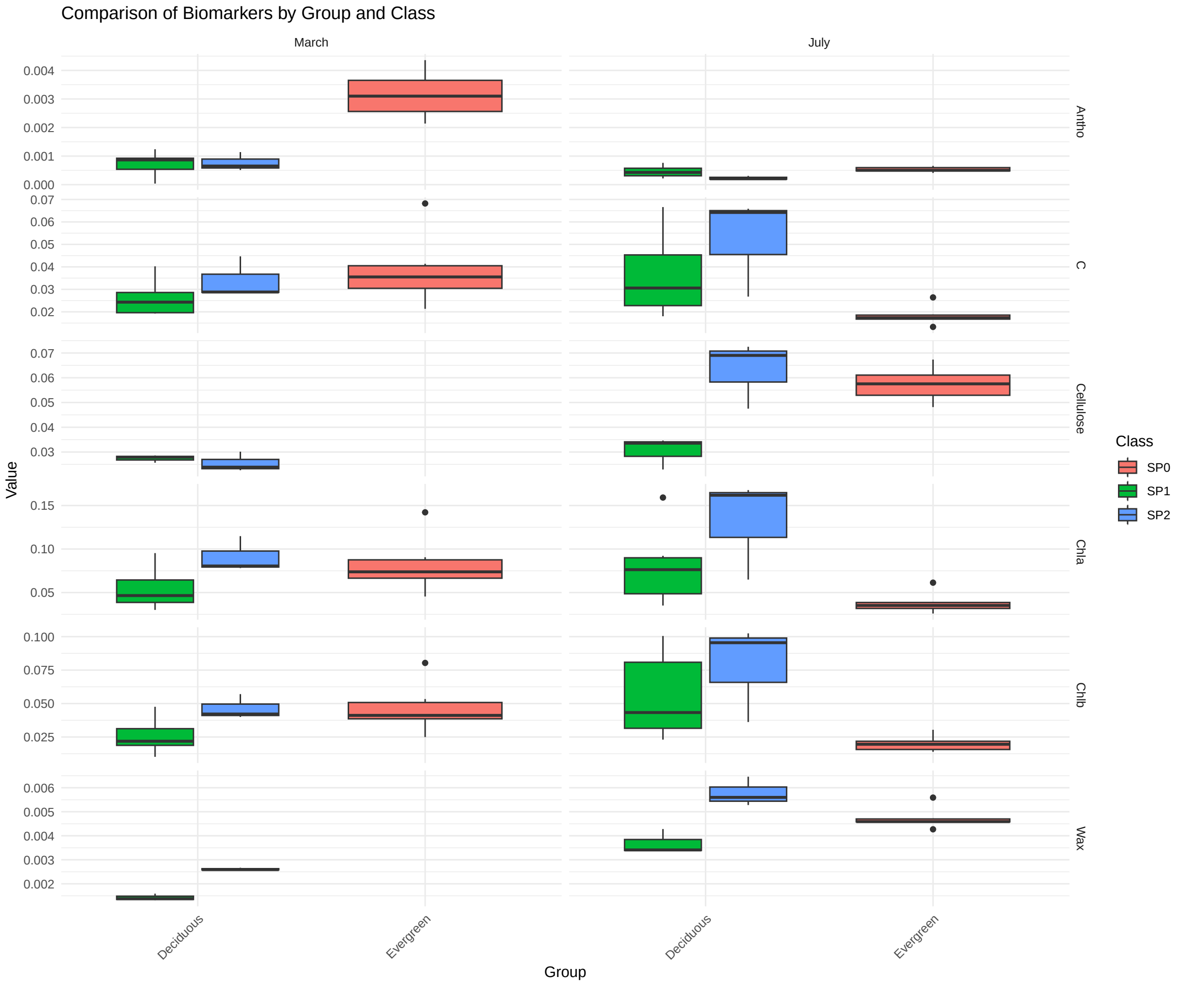

Figure 1: Comparison of biomarkers by group, class, and month

Understanding Biomarkers

Each biomarker represents different plant physiological processes:

- **Chla/Chlb**: Chlorophyll content (photosynthetic capacity)
- **C**: Carotenoid content (accessory pigments for light harvesting and photoprotection)
- **Antho**: Anthocyanin content (stress response pigments)
- **Cellulose**: Structural carbohydrates
- **Wax**: Protective leaf surface compounds

4.2 Statistical Testing

Perform pairwise comparisons to identify significant differences:

```r
# Calculate pairwise comparisons for each month and biomarker
valid_comparisons <- df_long %>%
  group_by(Month, Biomarker) %>%
  filter(!is.na(Value)) %>%
  filter(Class %in% c("SP0", "SP1", "SP2")) %>%
  compare_means(formula = Value ~ Class,
                group.by = c("Month", "Biomarker"),
                method = "t.test",
                p.adjust.method = "none",
                comparisons = list(c("SP0", "SP1"), c("SP0", "SP2"), c("SP1", "SP2")),
                na.rm = TRUE)

# Add position information for significance bars
global_max_y <- max(df_long$Value, na.rm = TRUE)
valid_comparisons <- valid_comparisons %>%
  group_by(Month, Biomarker) %>%
  arrange(p) %>%
  mutate(y.position = global_max_y * 1.05 + (row_number() - 1) * global_max_y * 0.1) %>%
  ungroup()

# Create plot with significance indicators
p <- ggboxplot(df_long, x = "Class", y = "Value", fill = "Class",
               facet.by = c("Month", "Biomarker"),
               short.panel.labs = TRUE,
               add = "jitter",
               palette = "jco") +
  stat_pvalue_manual(valid_comparisons,
                     label = "p.signif",
                     y.position = "y.position",
                     tip.length = 0.01) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(p)
```

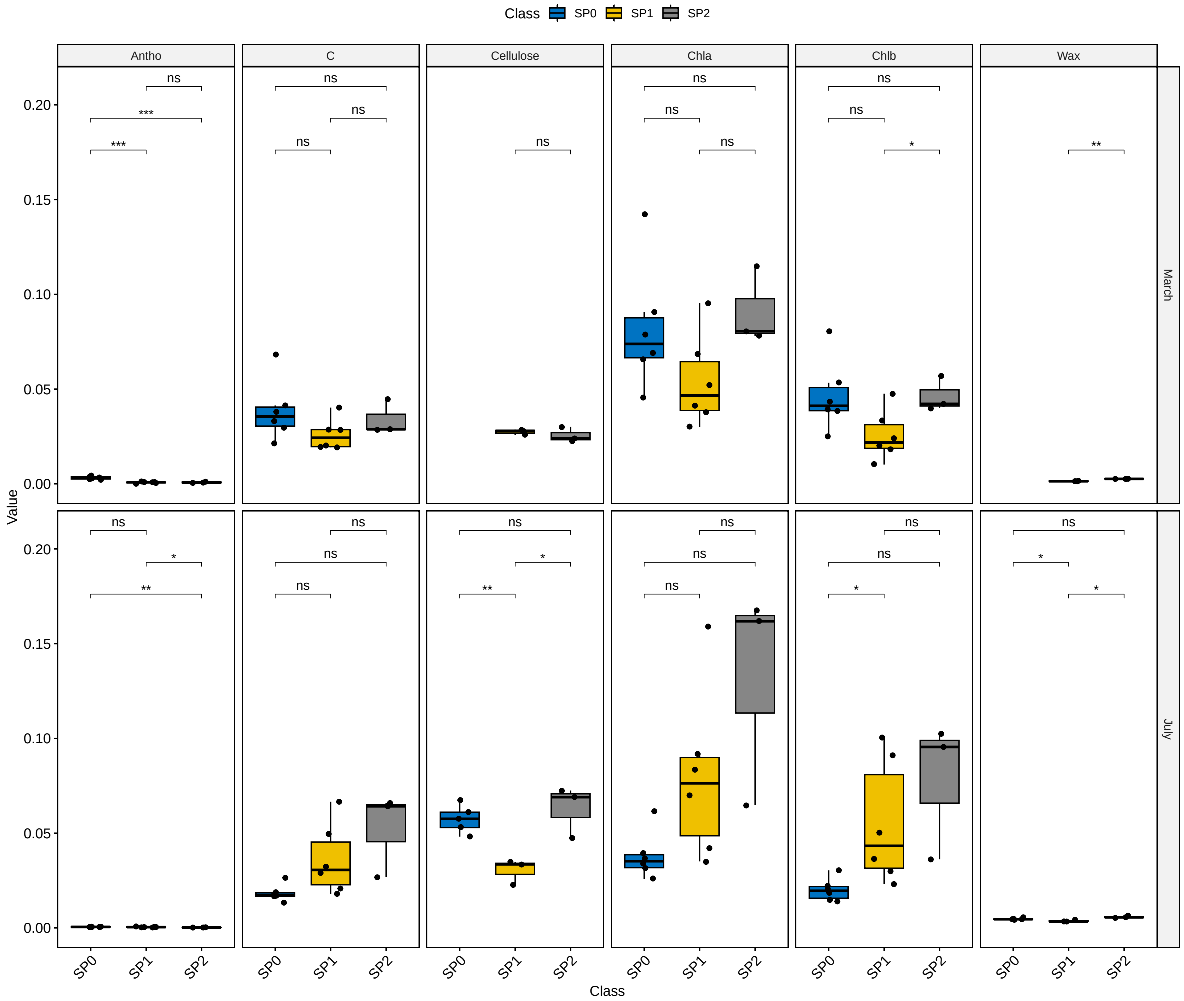

Figure 2: Biomarker comparisons with significance testing

Statistical Significance

Statistical testing helps identify which differences are statistically significant rather than due to random variation. The significance indicators (, ,) show the strength of evidence against the null hypothesis.

5 Part 3: Spectral Analysis by Biomarker Levels

This section explores how spectral signatures relate to biomarker concentrations.

5.1 Creating High/Low Groups Based on Quantiles

> Quantile-Based Biomarker Grouping
>
> To identify samples with distinctly high versus low biomarker concentrations, we use the 75th percentile (third quartile) as a threshold within each sampling month. Samples with biomarker values at or above the 75th percentile are classified as “High” (representing the top 25% of samples), while all remaining samples are classified as “Low” (bottom 75%). This approach ensures we’re comparing the most extreme cases - plants with genuinely elevated biomarker levels against the majority with typical or lower concentrations - making spectral differences more pronounced and biologically meaningful.

The following function implements this quantile-based approach to analyze spectral differences between high and low biomarker groups:

```r
analyze_spectral_by_biomarker <- function(biomarker_name) {
  # Filter data to specific plant groups
  data1 <- subset(data, Group == "Deciduous" | Group == "Evergreen")
  data1 <- data1 %>%
    mutate(Class = as.character(Class),
           Class = str_trim(Class)) %>%
    filter(!is.na(.data[[biomarker_name]]))

  # Group samples by 3rd quantile (top 25% vs. bottom 75%)
  data1 <- data1 %>%
    group_by(Month) %>%
    mutate(q3_val = quantile(.data[[biomarker_name]], 0.75, na.rm = TRUE),
           BioGroup = if_else(.data[[biomarker_name]] >= q3_val, "High", "Low")) %>%
    ungroup()

  # Extract spectral columns
  spectral_cols <- colnames(data1)[str_detect(colnames(data1), "^X\\d{3,4}$")]

  # Reshape to long format
  spectral_long <- data1 %>%
    dplyr::select(Month, BioGroup, all_of(spectral_cols)) %>%
    pivot_longer(cols = all_of(spectral_cols),
                 names_to = "Wavelength",
                 values_to = "Reflectance") %>%
    mutate(Wavelength = as.numeric(str_remove(Wavelength, "^X"))) %>%
    filter(!is.na(Reflectance))
  # Filter to visible and near-infrared range
  spectral_filtered <- spectral_long %>%
    filter(Wavelength >= 400, Wavelength <= 2400)

  # Calculate mean reflectance for each group
  avg_spectral <- spectral_filtered %>%
    group_by(Month, BioGroup, Wavelength) %>%
    summarise(MeanReflectance = mean(Reflectance, na.rm = TRUE), .groups = "drop")

  # Create plot
  ggplot(avg_spectral, aes(x = Wavelength, y = MeanReflectance, color = BioGroup)) +
    geom_line(linewidth = 1) +
    facet_wrap(~ Month) +
    theme_minimal() +
    labs(title = paste("Spectral Reflectance by", biomarker_name, "Level"),
         x = "Wavelength (nm)",
         y = "Mean Reflectance",
         color = paste(biomarker_name, "Level"))
}
```

5.2 Applying the Analysis to Different Biomarkers

```
# Analyze different biomarkers
biomarkers <- c("Antho", "C", "Chlb", "Chla", "Cellulose", "Wax")

for(biomarker in biomarkers) {
  plot <- analyze_spectral_by_biomarker(biomarker)
  print(plot)
}
```

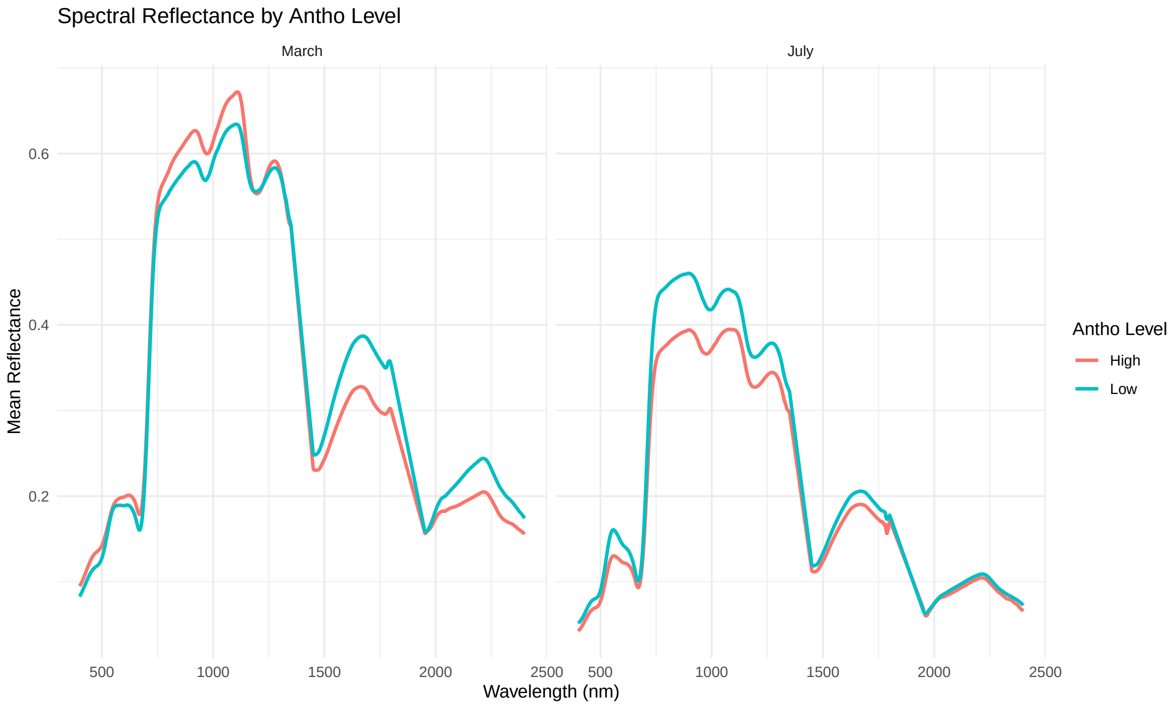

Figure 3: Spectral signatures for different biomarker levels

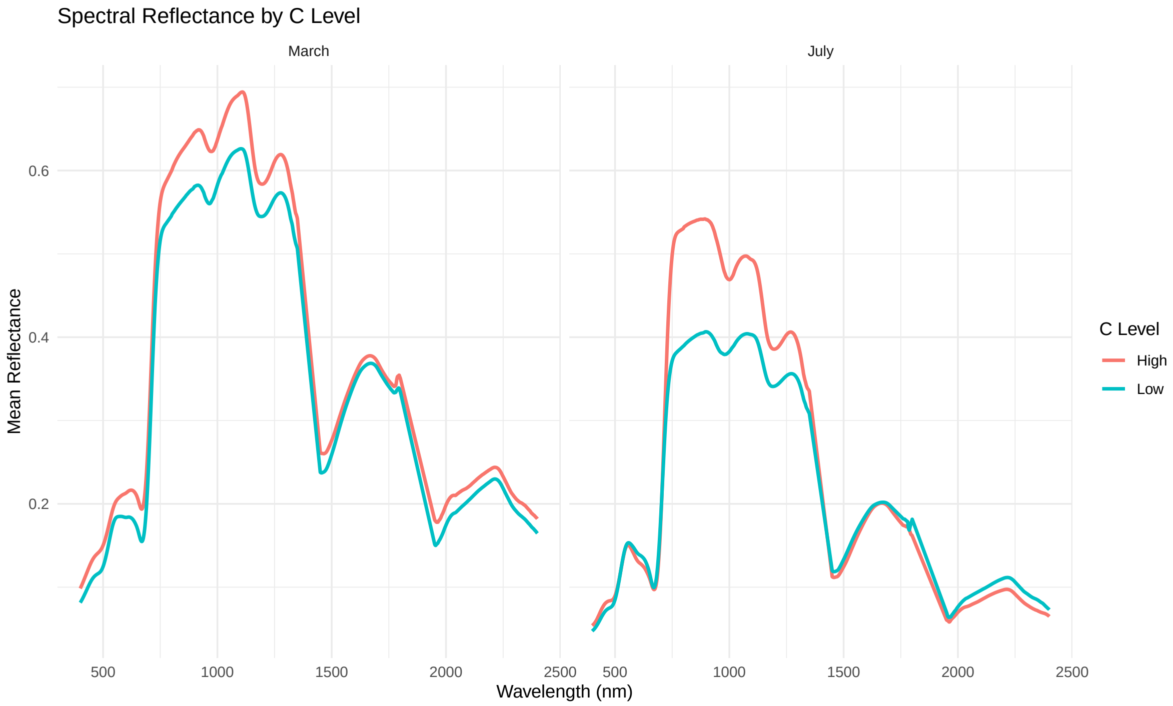

Figure 4: Spectral signatures for different biomarker levels

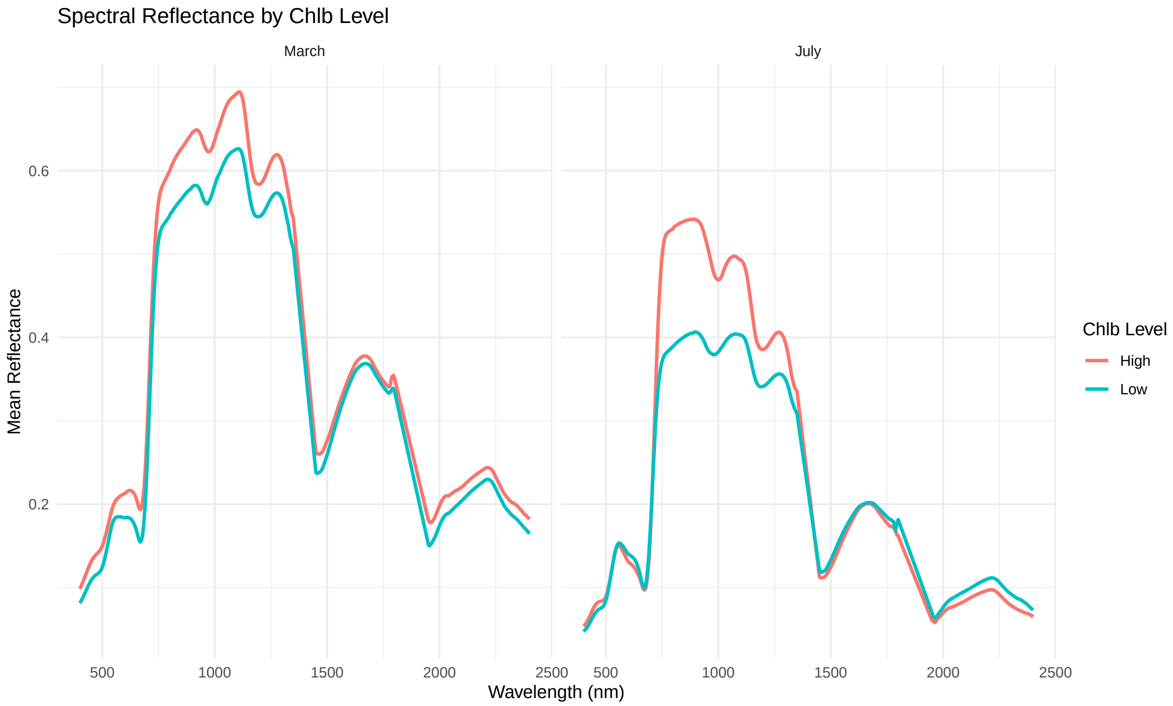

Figure 5: Spectral signatures for different biomarker levels

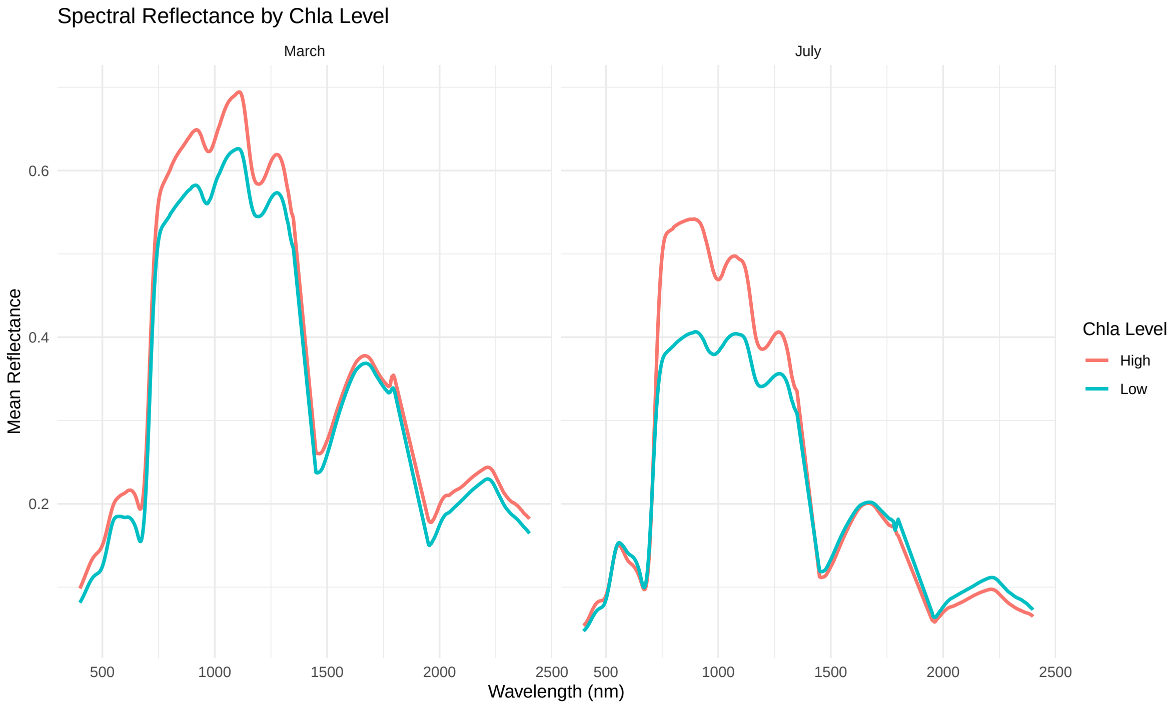

Figure 6: Spectral signatures for different biomarker levels

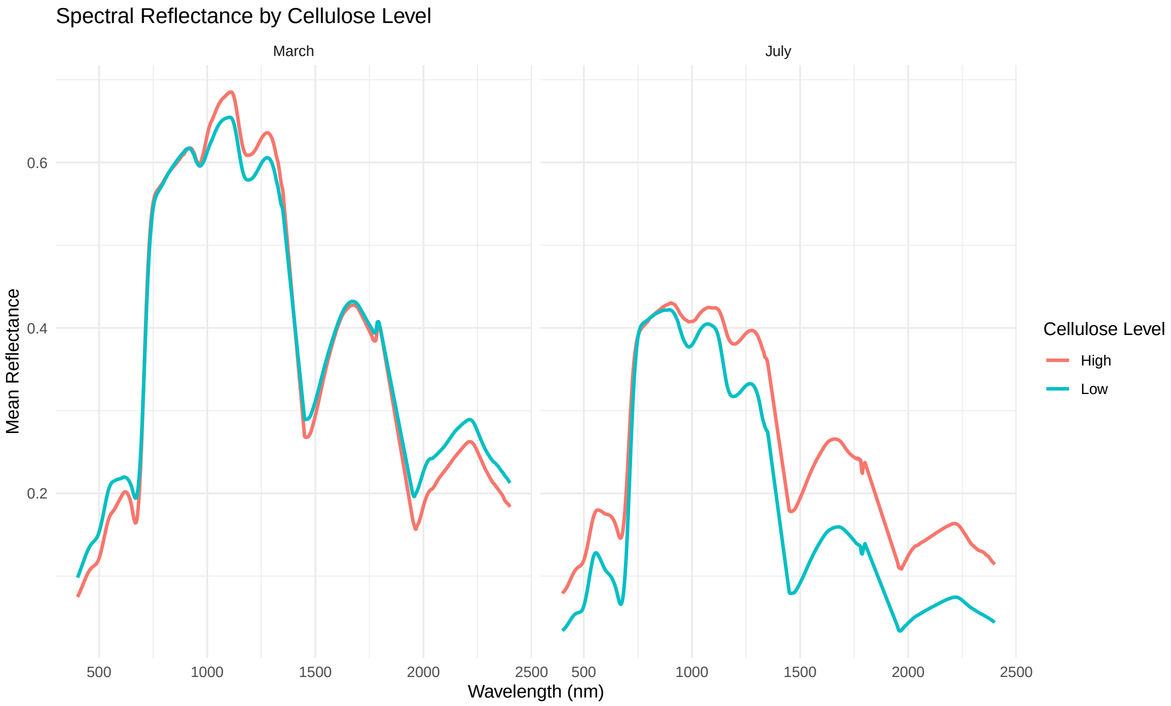

Figure 7: Spectral signatures for different biomarker levels

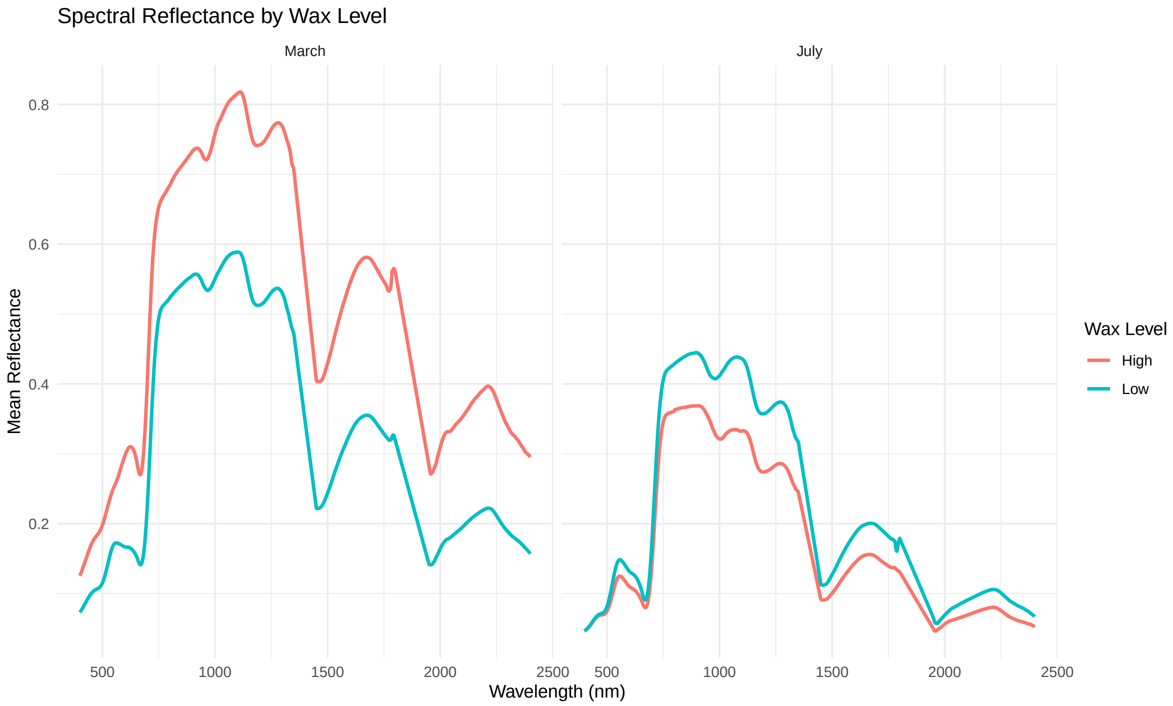

Figure 8: Spectral signatures for different biomarker levels

Understanding Spectral Signatures

Different biomarkers create distinct spectral patterns:

- **Chlorophyll**: Strong absorption around 680 nm (red) and 430 nm (blue)
- **Carotenoids**: Absorption around 480 nm (blue) and 500-550 nm (green), with reflectance peak around 550 nm (yellow)
- **Anthocyanins**: Absorption in green-yellow region (500-600 nm)
- **Cellulose**: Absorption features in near-infrared (1400-1500 nm, 1900-2000 nm)
- **Wax**: Affects overall reflectance levels, especially in near-infrared

6 Part 4: Machine Learning Classification

6.1 Classification Using Full Spectral Data

We'll use Partial Least Squares Discriminant Analysis (PLS-DA) to classify plant samples based on their spectral signatures.

7 What is PLS-DA?

Partial Least Squares Discriminant Analysis (PLS-DA) is a supervised classification technique that combines dimensionality reduction with discriminant analysis, making it particularly well-suited for high-dimensional data like hyperspectral datasets. Unlike traditional discriminant analysis methods that can fail when the number of variables exceeds the number of samples, PLS-DA first projects the data onto a lower-dimensional space of latent variables (components) that maximize the covariance between the predictor variables (spectral bands) and the class labels. This projection simultaneously reduces noise, handles multicollinearity among spectral bands, and identifies the most discriminative spectral features. The method then performs classification in this reduced space, making it robust for datasets with hundreds or thousands of correlated variables and relatively few samples - a common scenario in remote sensing applications.

7.0.1 Data Preparation for Classification

We now extract spectral data columns, scale them separately for each month to ensure equal variance across wavelengths, then combine the standardized data with class labels to create a machine learning-ready dataset.

```r
# Prepare classification dataset
spectral_names <- grep("^X\\d{3,4}$", names(data), value = TRUE)
ClassData <- subset(data, select = c("Month", "Group", "Class", spectral_names))

# Scale the spectral data separately for each month
spectral_cols_indices <- which(names(ClassData) %in% spectral_names)
MarchData <- scale(ClassData[1:15, spectral_cols_indices])
JulyData <- scale(ClassData[16:30, spectral_cols_indices])

# Combine scaled data
Class_Data_Scaled <- rbind(MarchData, JulyData)
Class_Data_Scaled <- cbind(ClassData[, 1:3], Class_Data_Scaled)
```

! Why Scale the Data

Scaling ensures that all spectral bands contribute equally to the analysis, preventing bands with higher absolute values from dominating the classification.

7.0.2 PLS-DA Function

This function applies the mixOmics::plsda() function to fit a PLS-DA model with 3 components, makes self-predictions on the training data, calculates accuracy metrics, and visualizes class separation in the first two component space.

The key point is that it uses plsda(X, Y, ncomp = 3) from the mixOmics package to perform the core discriminant analysis, then extracts predictions and component scores from the resulting model object for evaluation and plotting.

```r
run_plsda_for_month <- function(data, month) {
  # Subset data for specific month
  df <- subset(data, Month == month)
  df <- df[, !duplicated(names(df))]

  # Extract spectral columns and class labels
  spec_cols <- grep("^X\\d{3,4}$", names(df))
  X <- as.matrix(df[, spec_cols])  # Predictor matrix (spectral data)
  Y <- factor(df$Class)           # Response vector (plant classes)

  # Fit PLS-DA model
  plsda_model <- plsda(X, Y, ncomp = 3)

  # Make predictions
  pred_out <- predict(plsda_model, X)
  pred <- if (is.list(pred_out) && "class" %in% names(pred_out)) {
    pred_out$class$max.dist[, 2]  # Use component 2
  } else {
    as.factor(pred_out)
  }

  # Calculate accuracy metrics
  cm <- confusionMatrix(factor(pred, levels = levels(Y)), Y)
  OA <- cm$overall["Accuracy"]      # Overall Accuracy
  Kappa <- cm$overall["Kappa"]      # Cohen's Kappa

  # Create visualization of classification space
  scores_all <- plsda_model$variates$X[, 1:2]  # First two components
  plot_df <- data.frame(scores_all, Class = Y)
  colnames(plot_df)[1:2] <- c("Comp.1", "Comp.2")

  # Generate plot
  p <- ggplot(plot_df, aes(x = Comp.1, y = Comp.2, color = Class)) +
    geom_point(size = 3) +
    geom_mark_ellipse(aes(fill = Class), alpha = 0.2, show.legend = FALSE) +
    theme_minimal() +
    ggtitle(sprintf("%s - OA=%.3f, Kappa=%.3f", month, OA, Kappa)) +
    theme(plot.title = element_text(hjust = 0.5))

  return(list(plot = p, OA = OA, Kappa = Kappa))
}
```

7.0.3 Running Classification Analysis

```
# Subset data by month
Class_Data_Scaled_M <- subset(Class_Data_Scaled, Month == "March")
Class_Data_Scaled_J <- subset(Class_Data_Scaled, Month == "July")

# Run PLS-DA for each month
march_res <- run_plsda_for_month(Class_Data_Scaled_M, "March")
july_res <- run_plsda_for_month(Class_Data_Scaled_J, "July")

# Display results
combined_plot <- march_res$plot + july_res$plot + plot_layout(ncol = 2)
print(combined_plot)
```

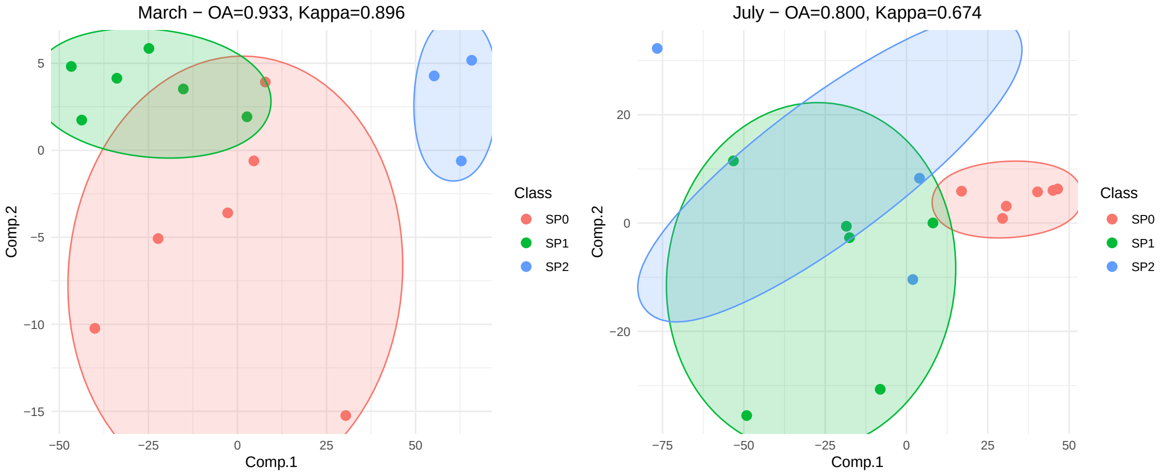

Figure 9: PLS-DA classification results using full spectral data

ℹ Interpreting PLS-DA Component Plots

Accuracy metrics evaluate the overall classification success:

- **Overall Accuracy (OA)**: Percentage of correctly classified samples
- **Kappa coefficient**: Agreement between predicted and actual classes, corrected for chance

The PLS-DA scatter plot shows classification results projected onto the first two discriminant components, revealing how well spectral signatures separate plant classes:

- **Tight clusters** (like SP1-green and SP2-blue) indicate classes with distinct, consistent spectral signatures
- **Dispersed patterns** (like SP0-red) suggest greater within-class spectral variability
- **Component axes** represent the most discriminative spectral directions: Component 1 typically provides primary class separation, while Component 2 refines distinctions between remaining classes
- **Elliptical boundaries** show 95% confidence regions; limited overlap indicates successful spectral discrimination
- **High accuracy metrics** (OA=93.3%, Kappa=89.6%) confirm that the model found reliable spectral features distinguishing the classes

This visualization helps assess both model performance and the underlying spectral separability of your plant classes.

7.1 Variable Importance Analysis (VIP Scores)

Identify which spectral bands are most important for classification:

```r
# Function to extract VIP scores
run_plsda_and_get_vip <- function(data, month, ncomp = 3) {
  df <- subset(data, Month == month)
  df <- df[, !duplicated(names(df))]

  spec_cols <- grep("^X\\d{3,4}$", names(df))
  X <- as.matrix(df[, spec_cols])
  Y <- factor(df$Class)

  plsda_model <- mixOmics::plsda(X, Y, ncomp = ncomp)
  vip_scores <- mixOmics::vip(plsda_model)[,1]  # VIP for first component

  return(list(model = plsda_model, VIP = vip_scores))
}

# Extract VIP scores for both months
march_VIP <- run_plsda_and_get_vip(Class_Data_Scaled_M, "March")
july_VIP <- run_plsda_and_get_vip(Class_Data_Scaled_J, "July")

# Prepare VIP data for visualization
vip_df <- data.frame(
  Variable = names(march_VIP$VIP),
  March = as.numeric(march_VIP$VIP),
  July = as.numeric(july_VIP$VIP)
)

vip_long <- vip_df %>%
  pivot_longer(cols = c("March", "July"), names_to = "Month", values_to = "VIP")

# Add wavelength information and spectral regions
vip_long$Wavelength <- as.numeric(gsub("X", "", vip_long$Variable))
vip_long$WavelengthGroup <- cut(
  vip_long$Wavelength,
  breaks = c(399, 700, 1300, 2400),
  labels = c("VIS (400-700)", "NIR (701-1300)", "SWIR (1301-2400)")
)

# Create labels for every 100 nm
vip_long$Label <- ifelse(vip_long$Wavelength %% 100 == 0,
                        paste0(vip_long$Wavelength, " nm"), "")

# Order variables by wavelength
ordered_vars <- vip_long %>%
  distinct(Variable, Wavelength) %>%
  arrange(Wavelength) %>%
  pull(Variable)
vip_long$Variable <- factor(vip_long$Variable, levels = ordered_vars)

# Plot VIP scores
vip_plot <- ggplot(vip_long, aes(x = Variable, y = VIP, fill = Month)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "red") +
  theme_minimal() +
  labs(title = "VIP Scores Comparison: March vs July",
       y = "VIP Score", x = NULL, fill = "Month") +
  scale_x_discrete(labels = vip_long$Label) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 7)) +
  facet_grid(. ~ WavelengthGroup, scales = "free_x", space = "free_x")

print(vip_plot)
```

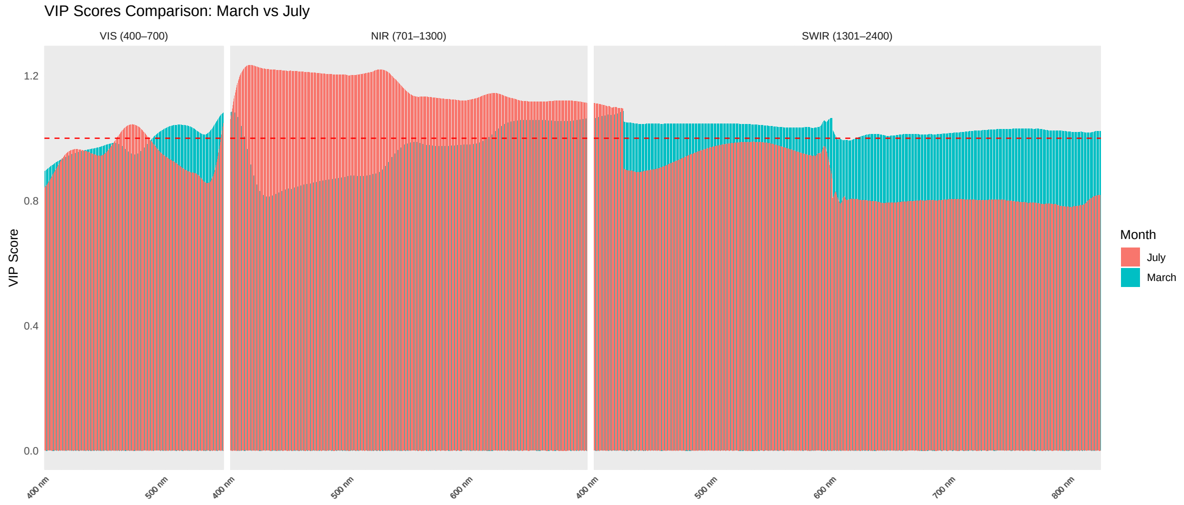

Figure 10: Variable Importance in Projection (VIP) scores for spectral bands

Understanding VIP Scores

- **VIP $>$ 1**: Important for classification (above red dashed line)
- **VIP $<$ 1**: Less important
- Higher VIP scores indicate wavelengths that contribute more to class discrimination

8 Part 5: Classification Using Vegetation Indices

Instead of using all spectral bands, we can create vegetation indices that capture specific plant properties.

8.1 Calculating Vegetation Indices

```
# Define wavelength ranges for index calculations
cols_750_770 <- paste0("X", 750:770)
cols_490_500 <- paste0("X", 490:500)
cols_660_690 <- paste0("X", 660:690)
cols_690_720 <- paste0("X", 690:720)
cols_760_800 <- paste0("X", 760:800)
cols_510_520 <- paste0("X", 510:520)
cols_690_710 <- paste0("X", 690:710)
cols_530_570 <- paste0("X", 540:560)

# Calculate mean reflectance for each range
mean_750_770 <- rowMeans(data[, cols_750_770], na.rm = TRUE)
mean_490_500 <- rowMeans(data[, cols_490_500], na.rm = TRUE)
mean_660_690 <- rowMeans(data[, cols_660_690], na.rm = TRUE)
mean_690_720 <- rowMeans(data[, cols_690_720], na.rm = TRUE)
mean_760_800 <- rowMeans(data[, cols_760_800], na.rm = TRUE)
mean_510_520 <- rowMeans(data[, cols_510_520], na.rm = TRUE)
mean_690_710 <- rowMeans(data[, cols_690_710], na.rm = TRUE)
mean_530_570 <- rowMeans(data[, cols_530_570], na.rm = TRUE)

# Calculate vegetation indices
VI_Chl <- (1/mean_690_720 - 1/mean_760_800) * mean_760_800      # Chlorophyll index
VI_C <- (1/mean_510_520 - 1/mean_690_710) * mean_760_800        # Carbon index
VI_Antho <- (1/mean_530_570 - 1/mean_690_710) * mean_760_800    # Anthocyanin index
VI_Cell <- with(data, 100 * (0.5 * (X2030 + X2210) - X2100))   # Cellulose index
VI_Wax <- 1 / sqrt(mean_750_770 - mean_490_500 - mean_660_690)  # Wax index

# Add indices to dataset
data$VI_Chl <- VI_Chl
data$VI_C <- VI_C
data$VI_Antho <- VI_Antho
data$VI_Cell <- VI_Cell
data$VI_Wax <- VI_Wax
```

Why Use Vegetation Indices

These indices are designed to enhance specific plant properties while reducing the effects of:

- Atmospheric conditions
- Soil background
- Illumination variations

- Noise in individual spectral bands

8.2 Classification Using Indices

```
# Prepare index-based classification data
ClassData <- subset(data, select = c("Month", "Group", "Class",
                                    "VI_Chl", "VI_C", "VI_Antho", "VI_Cell", "VI_Wax"))

# Scale indices separately for each month
MarchData <- scale(ClassData[1:15, 4:8])
JulyData <- scale(ClassData[16:30, 4:8])
Class_Data_Scaled <- rbind(MarchData, JulyData)
Class_Data_Scaled <- cbind(ClassData[, 1:3], Class_Data_Scaled)

# Modified PLS-DA function for vegetation indices
run_plsda_for_month_indices <- function(data, month) {
  df <- subset(data, Month == month)
  df <- df[, !duplicated(names(df))]

  vi_cols <- grep("^VI_", names(df))
  X <- as.matrix(df[, vi_cols])
  Y <- factor(df$Class)

  plsda_model <- plsda(X, Y, ncomp = 3)

  pred_out <- predict(plsda_model, X)
  pred <- if (is.list(pred_out) && "class" %in% names(pred_out)) {
    pred_out$class$max.dist[, 2]
  } else {
    as.factor(pred_out)
  }

  cm <- confusionMatrix(factor(pred, levels = levels(Y)), Y)
  OA <- cm$overall["Accuracy"]
  Kappa <- cm$overall["Kappa"]

  scores_all <- plsda_model$variates$X[, 1:2]
  plot_df <- data.frame(scores_all, Class = Y)
  colnames(plot_df)[1:2] <- c("Comp.1", "Comp.2")
  p <- ggplot(plot_df, aes(x = Comp.1, y = Comp.2, color = Class)) +
    geom_point(size = 3) +
    geom_mark_ellipse(aes(fill = Class), alpha = 0.2, show.legend = FALSE) +
    theme_minimal() +
    ggtitle(sprintf("%s - OA=%.3f, Kappa=%.3f", month, OA, Kappa)) +
    theme(plot.title = element_text(hjust = 0.5))

  return(list(plot = p, OA = OA, Kappa = Kappa))
}

# Run classification with indices
Class_Data_Scaled_M <- subset(Class_Data_Scaled, Month == "March")
Class_Data_Scaled_J <- subset(Class_Data_Scaled, Month == "July")

march_res <- run_plsda_for_month_indices(Class_Data_Scaled_M, "March")
july_res <- run_plsda_for_month_indices(Class_Data_Scaled_J, "July")

combined_plot <- march_res$plot + july_res$plot + plot_layout(ncol = 2)
print(combined_plot)
```

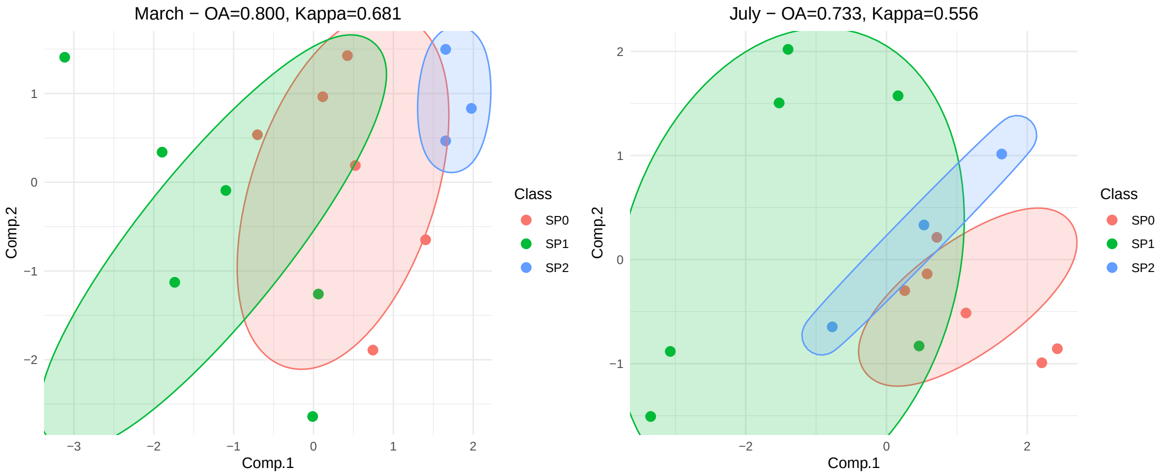

Figure 11: PLS-DA classification results using vegetation indices

8.3 VIP Analysis for Vegetation Indices

```r
# VIP analysis for vegetation indices
run_plsda_and_get_vip_indices <- function(data, month, ncomp = 3) {
  df <- subset(data, Month == month)
  df <- df[, !duplicated(names(df))]

  vi_cols <- grep("^VI_", names(df))
  X <- as.matrix(df[, vi_cols])
  Y <- factor(df$Class)

  plsda_model <- mixOmics::plsda(X, Y, ncomp = ncomp)
  vip_scores <- mixOmics::vip(plsda_model)[,1]

  return(list(model = plsda_model, VIP = vip_scores))
}

# Extract VIP scores
march_VIP <- run_plsda_and_get_vip_indices(Class_Data_Scaled_M, "March")
july_VIP <- run_plsda_and_get_vip_indices(Class_Data_Scaled_J, "July")

# Prepare and plot VIP data
vip_df <- data.frame(
  Variable = names(march_VIP$VIP),
  March = as.numeric(march_VIP$VIP),
  July = as.numeric(july_VIP$VIP)
)

vip_long <- vip_df %>%
  pivot_longer(cols = c("March", "July"), names_to = "Month", values_to = "VIP")

vip_plot <- ggplot(vip_long, aes(x = Variable, y = VIP, fill = Month)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "red") +
  geom_text(aes(label = round(VIP, 2)), position = position_dodge(width = 0.8),
            vjust = -0.5, size = 3) +
  theme_minimal() +
  labs(title = "VIP Scores Comparison: March vs July (Vegetation Indices)",
       y = "VIP Score", x = NULL, fill = "Month") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(vip_plot)
```

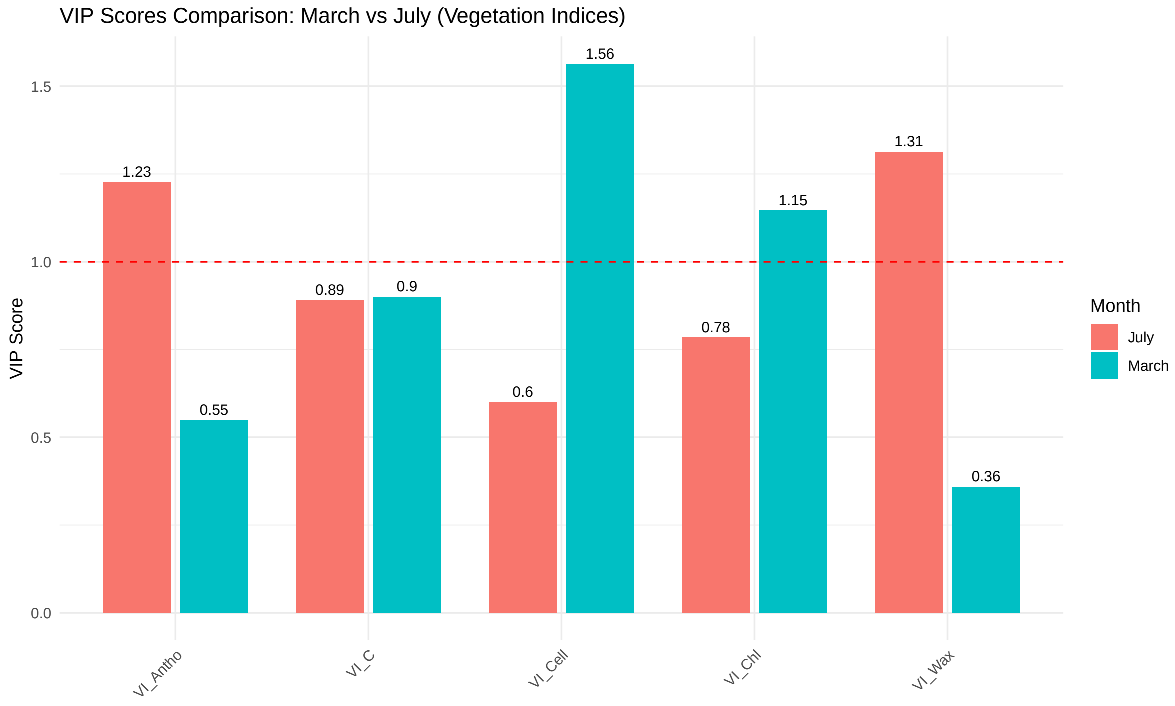

Figure 12: VIP scores for vegetation indices

9 Summary and Key Takeaways

10 Data Preprocessing

- Always scale spectral data before analysis
- Handle missing values appropriately
- Convert categorical variables to factors
- Consider temporal effects when designing analysis

11 Exploratory Analysis

- Visualize biomarker distributions across groups
- Examine spectral signatures for different conditions
- Use statistical tests to identify significant differences
- Look for patterns across different wavelength regions

12 Classification Approaches

Full spectral data: - Pros: More information, captures subtle spectral features - Cons: High dimensionality, potential overfitting, computational complexity

Vegetation indices:
- Pros: Reduced dimensionality, interpretable, robust to noise - Cons: May lose some spectral information, limited to pre-defined indices

13 Model Validation

- Use appropriate cross-validation strategies
- Report multiple accuracy metrics (OA, Kappa)
- Interpret VIP scores to understand important features
- Consider seasonal/temporal variations in model performance

14 Best Practices

- Always validate results across different time periods
- Consider biological relevance when interpreting results
- Use multiple approaches to confirm findings
- Document preprocessing steps for reproducibility

15 Advanced Topics and Extensions

15.1 Cross-Validation Strategies

For more robust model evaluation, implement proper cross-validation:

```r
# Example of k-fold cross-validation for PLS-DA
perform_cv_plsda <- function(data, k_folds = 5) {
  # Prepare data
  ind_cols <- grep("VI", names(data))
  X <- as.matrix(data[, ind_cols])
  Y <- factor(data$Class)

  # Create folds
  folds <- createFolds(Y, k = k_folds, list = TRUE)

  cv_results <- map_dfr(names(folds), function(fold_name) {
    test_idx <- folds[[fold_name]]
    train_idx <- setdiff(1:nrow(X), test_idx)

    # Train model
    plsda_model <- plsda(X[train_idx, ], Y[train_idx], ncomp = 3)

    # Predict on test set
    pred <- predict(plsda_model, X[test_idx, ])
    pred_class <- pred$class$max.dist[, 2]  # Use component 2

    # Calculate metrics
    cm <- confusionMatrix(factor(pred_class, levels = levels(Y)), Y[test_idx])

    tibble(
      fold = fold_name,
      accuracy = cm$overall["Accuracy"],
      kappa = cm$overall["Kappa"]
    )
  })

  return(cv_results)
}

# Example usage (uncomment to run)
march_data <- subset(Class_Data_Scaled, Month == "March")
cv_results <- perform_cv_plsda(march_data)
print(paste("Mean CV Accuracy:", round(mean(cv_results$accuracy), 3)))
```

```
[1] "Mean CV Accuracy: 0.8"
```

16 Why Cross-Validation Matters

Cross-validation prevents overly optimistic performance estimates by testing models on data they haven't seen during training. The earlier classification results used the same data for both training and testing (self-prediction), which typically inflates accuracy metrics since models can memorize training patterns rather than learn generalizable features. K-fold cross-validation splits data into multiple train-test partitions, providing more realistic estimates of how well the model will perform on new, unseen samples - the true test of a classification algorithm's practical utility.

17 Conclusion

This tutorial provides a comprehensive workflow for analyzing hyperspectral data in plant science applications. The methods demonstrated here can be adapted to various remote sensing and precision agriculture applications. The combination of statistical analysis, machine learning, and domain knowledge creates a robust framework for understanding plant spectral properties and their relationship to physiological characteristics.

The integration of multiple analytical approaches—from basic statistical comparisons to advanced machine learning techniques—provides researchers with a complete toolkit for hyperspectral data analysis. The emphasis on both full spectral analysis and vegetation indices demonstrates the trade-offs between detailed spectral information and practical, interpretable measures.

ℹ Next Steps

To further develop your hyperspectral analysis skills:

- **Cross-validation**: Implement robust validation strategies with your own datasets
- **Alternative algorithms**: Experiment with Random Forest, SVM, or deep learning approaches
- **Feature selection**: Apply dimensionality reduction techniques to optimize model performance
- **Temporal analysis**: Explore how spectral signatures change over time or growing seasons
- **Scale integration**: Connect lab measurements to field and satellite observations
- **Operational applications**: Develop automated pipelines for routine monitoring tasks

18 Additional Resources

- **Spectral libraries**: USGS, ECOSIS for reference spectra
- **Software packages**: `hsdar`, `RStoolbox` for specialized hyperspectral analysis
- **Remote sensing**: Integration with Google Earth Engine or other platforms
- **Field validation**: Best practices for ground-truthing remote sensing predictions

⚠ Important Considerations

When applying these methods to your own data:

- **Sample size**: Ensure adequate samples per class for reliable statistics
- **Data quality**: Check for instrument calibration and atmospheric corrections

- **Biological relevance**: Validate that spectral differences align with known plant physiology

- **Temporal effects**: Account for seasonal, phenological, and environmental variations
- **Scale effects**: Consider how lab results translate to field and landscape scales
- **Model generalization**: Test models across different sites, sensors, and conditions

---

This tutorial was designed to provide both theoretical understanding and practical implementation guidance for hyperspectral data analysis. The modular structure allows researchers to adapt specific components to their unique research questions and datasets.