Coffee NIR Data Analysis Report

Group 19

Nguyen Van A - ID: 2012345
Tran Thi B - ID: 2012346
Le Van C - ID: 2012347
Pham Thi D - ID: 2012348

2025-12-15

Contents

Giới thiệu

Báo cáo này trình bày toàn bộ kết quả phân tích dữ liệu Coffee NIR, bao gồm:

- **Phần 1**: Giới thiệu dữ liệu và kiểm tra chất lượng
  - Mô tả các nhóm biến: Hóa lý, Location, NIR
  - Kiểm tra dữ liệu thiếu (Missing Data)
  - Phát hiện giá trị lạ (Outliers Detection)
- **Phần 2.1**: Phân tích sự khác biệt - PCA và Clustering
  - Principal Component Analysis (PCA) trên dữ liệu NIR và Hóa lý
  - Hierarchical Clustering và K-means
  - So sánh sự khác biệt giữa các vị trí địa lý
  - Xác định đặc điểm phân biệt giữa các nhóm
- **Phần 2.2**: Phát hiện Outliers nâng cao
  - Hotelling T^2 statistic và Q residuals (SPE)
  - Combined T^2 vs Q plot
  - Influence analysis
  - Phân loại và quyết định xử lý outliers
- **Phần 2.3**: Phân tích Tương quan và Heatmap
  - Ma trận tương quan giữa các biến hóa lý
  - Tương quan giữa NIR wavelengths và biến hóa lý
  - Correlation profile và wavelengths quan trọng
  - Tương quan trong không gian PCA
  - So sánh pattern tương quan giữa các Location

Dữ liệu sử dụng: **coffee_nirs.csv**

Chapter 1

Giải Thích Các Thuật Toán

Phần này giải thích các phương pháp phân tích dữ liệu được sử dụng trong báo cáo một cách đơn giản, dễ hiểu.

1.1 Phân Tích Thành Phần Chính (PCA)

PCA là gì?

Tưởng tượng bạn có 1050 bước sóng NIR - quá nhiều thông tin để hiển thị trên một biểu đồ 2D hoặc 3D. PCA giúp "nén" 1050 chiều này thành một số chiều nhỏ hơn (ví dụ: 2-3 chiều) mà vẫn giữ được phần lớn thông tin quan trọng.

Cách hoạt động:

1. **Tìm hướng có sự biến thiên lớn nhất**: PCA tìm hướng trong dữ liệu mà các mẫu phân tán nhiều nhất
2. **Tạo trục mới**: Các hướng này trở thành "thành phần chính" (PC1, PC2, PC3,...)
3. **Giảm chiều**: Chỉ giữ lại một số thành phần đầu tiên (thường là 2-10) vì chúng chứa hầu hết thông tin

Ví dụ thực tế:

- PC1 có thể đại diện cho ~60% sự khác biệt giữa các mẫu cà phê
- PC2 đại diện cho ~20% sự khác biệt tiếp theo

- Với 2 thành phần này, ta đã có 80% thông tin nhưng chỉ cần vẽ biểu đồ 2D

Ứng dụng:

- Trực quan hóa dữ liệu nhiều chiều
- Phát hiện các nhóm mẫu tương tự nhau
- Giảm nhiễu trong dữ liệu

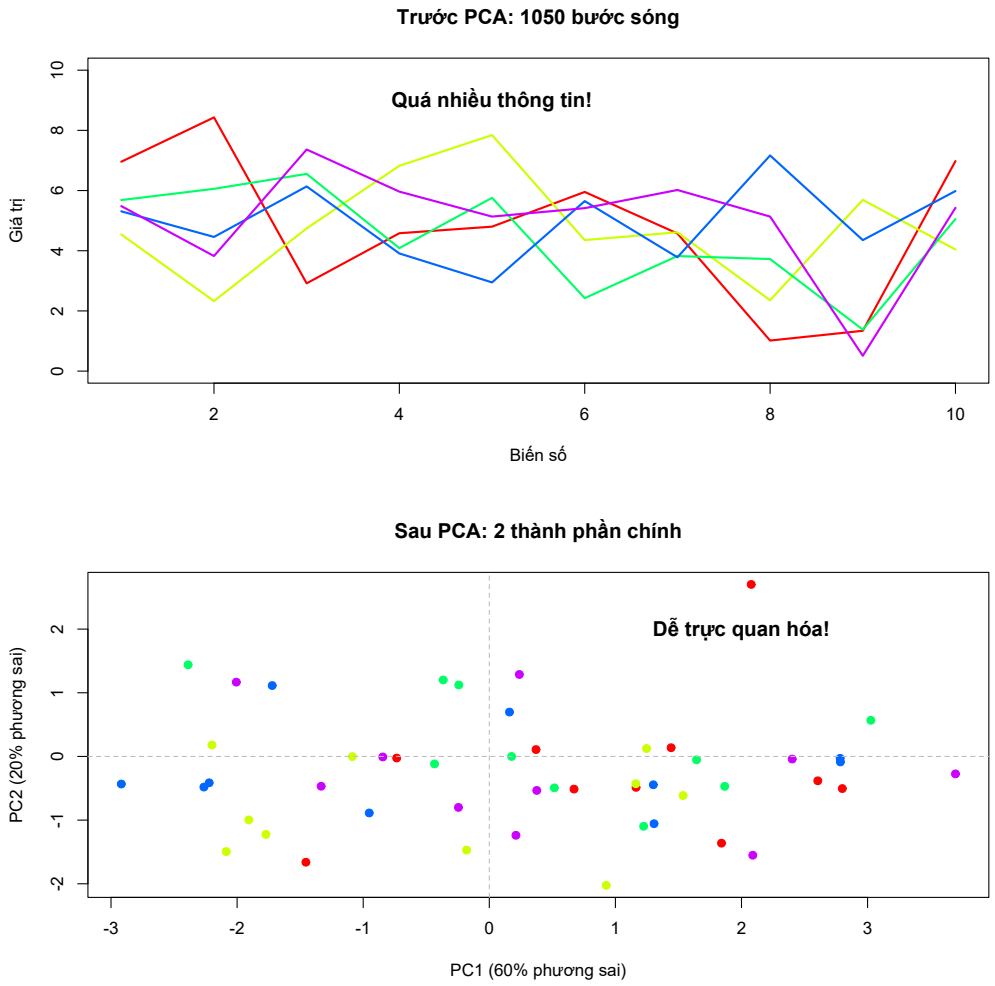


Figure 1.1: *Minh họa PCA: Từ nhiều chiều xuống 2 chiều*

1.2 Phân Cụm (Clustering)

Clustering là gì?

Clustering giúp nhóm các mẫu tương tự nhau lại với nhau, giống như việc sắp xếp trái cây vào các rổ dựa trên màu sắc và kích thước.

1.2.1 Phân Cụm Phân Cấp (Hierarchical Clustering)

Cách hoạt động:

1. **Bắt đầu**: Mỗi mẫu là một cụm riêng
2. **Ghép nối**: Tìm 2 cụm gần nhau nhất và ghép chúng lại
3. **Lặp lại**: Tiếp tục ghép cho đến khi tất cả mẫu nằm trong một cụm lớn
4. **Kết quả**: Một "cây phả hệ" (dendrogram) cho biết mẫu nào giống mẫu nào

Phương pháp Ward:

- Ghép các cụm sao cho sự phân tán bên trong cụm tăng ít nhất
- Tạo ra các cụm cân bằng và rõ ràng

1.2.2 K-Means Clustering

Cách hoạt động:

1. **Chọn số cụm k**: Ví dụ k=3 (3 nhóm cà phê)
2. **Đặt tâm ngẫu nhiên**: Chọn 3 điểm làm tâm cụm ban đầu
3. **Gán mẫu**: Mỗi mẫu được gán vào cụm có tâm gần nhất
4. **Cập nhật tâm**: Tính lại tâm của mỗi cụm
5. **Lặp lại**: Bước 3-4 cho đến khi cụm không đổi

Ưu điểm:

- Nhanh và hiệu quả
- Dễ hiểu và giải thích

Lưu ý:

- Cần biết trước số cụm k
- Kết quả có thể khác nhau tùy tâm ban đầu

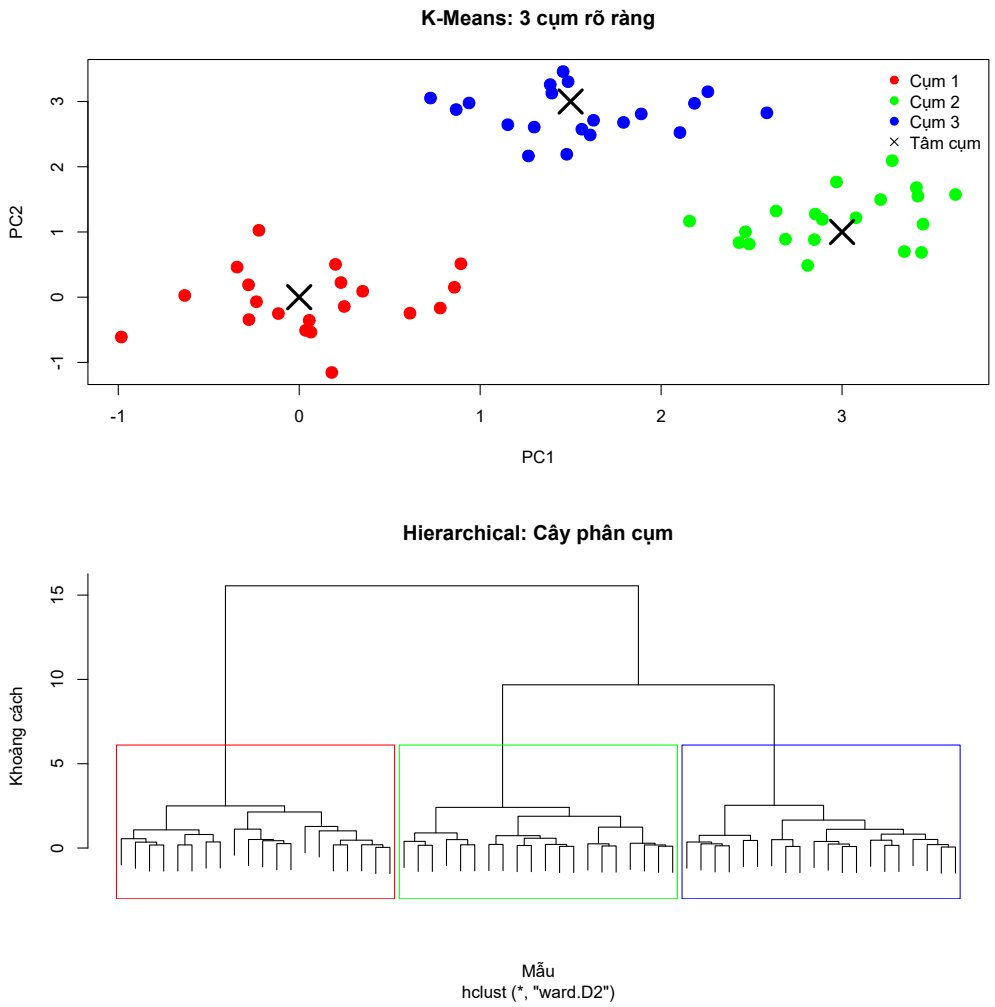


Figure 1.2: *Minh họa Clustering*

1.3 Phát Hiện Giá Trị Lạ (Outlier Detection)

Outlier là gì?

Outlier là mẫu "khác thường" - rất khác so với phần lớn các mẫu khác. Giống như một quả táo màu xanh lá trong rổ táo đỏ.

1.3.1 Khoảng Tứ Phân Vị (IQR Method)

Cách hoạt động:

1. **Tính tứ phân vị**: Q1 (25%), Q3 (75%)
2. **Tính IQR**: IQR = Q3 - Q1
3. **Xác định ngưỡng**:
    - Ngưỡng dưới = Q1 - 1.5×IQR
    - Ngưỡng trên = Q3 + 1.5×IQR
4. **Phát hiện outlier**: Giá trị ngoài ngưỡng là outlier

Ưu điểm: Đơn giản, ổn định với dữ liệu lệch

1.3.2 Khoảng Cách Mahalanobis

Khác với khoảng cách thông thường:

- Khoảng cách Euclid: Đường chim bay giữa 2 điểm
- Khoảng cách Mahalanobis: Tính đến sự tương quan giữa các biến

Ứng dụng:

- Phát hiện outlier trong dữ liệu nhiều chiều
- Tính đến cấu trúc phân tán của dữ liệu

1.3.3 Hotelling T² và Q Residuals

Hotelling T²:

- Đo lường mức độ "cực đoan" của mẫu trong không gian PC
- Giống như hỏi: "Mẫu này có nằm xa trung tâm không?"

Q Residuals (SPE):

- Đo lường phần thông tin không được giải thích bởi PCA
- Giống như hỏi: "Có điều gì đặc biệt mà PCA không nắm bắt được không?"

Kết hợp T² và Q:

- **T² cao, Q thấp**: Mẫu bình thường nhưng "cực đoan" (ví dụ: nồng độ rất cao)
- **T² thấp, Q cao**: Mẫu có pattern khác thường
- **T² cao, Q cao**: Outlier thực sự - rất khác biệt!

1.4 Phân Tích Tương Quan

Tương quan là gì?

Tương quan đo lường mối quan hệ tuyến tính giữa hai biến:

- **r = +1**: Tương quan dương hoàn hảo (khi A tăng, B tăng)
- **r = 0**: Không có tương quan
- **r = -1**: Tương quan âm hoàn hảo (khi A tăng, B giảm)

Hệ số Pearson:

- Đo lường sức mạnh và hướng của mối quan hệ tuyến tính
- Giá trị từ -1 đến +1

Heatmap tương quan:

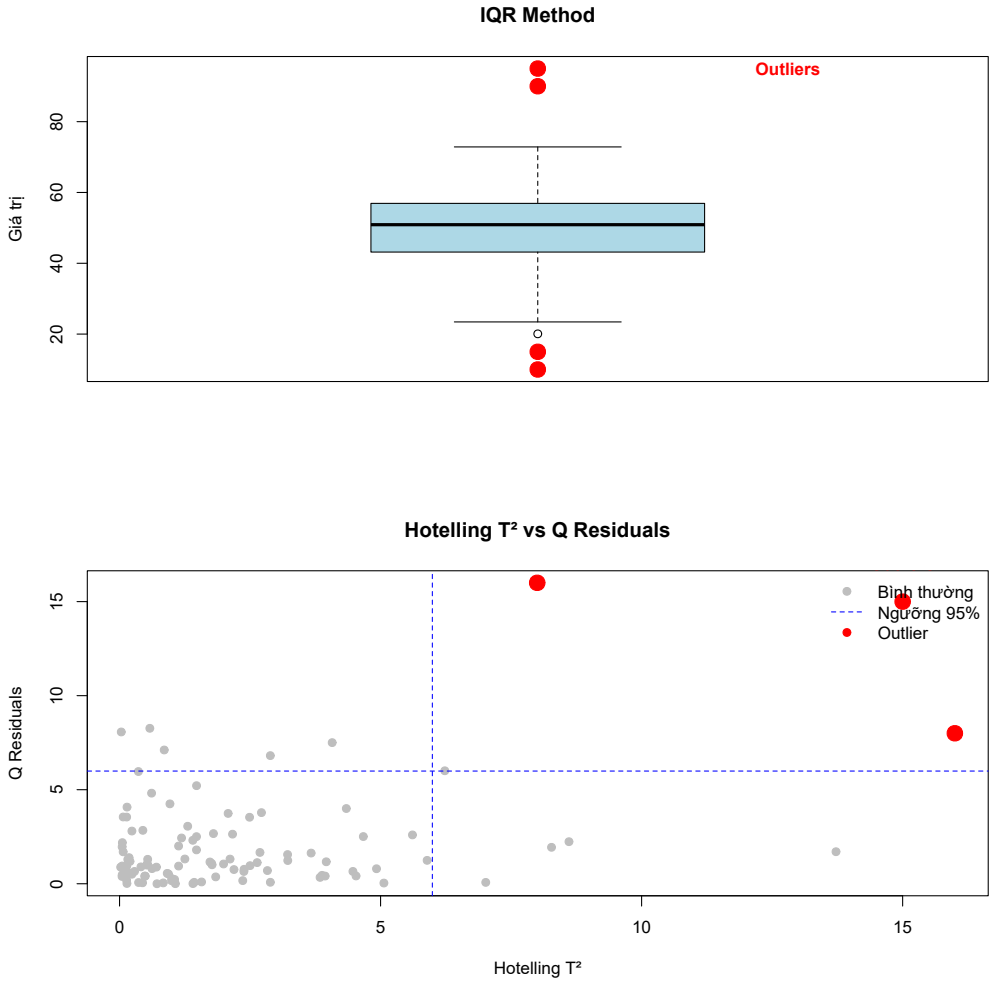


Figure 1.3: *Minh họa Outlier Detection*

- Biểu đồ màu sắc cho thấy tương quan giữa nhiều cặp biến
- Màu nóng (đỏ): Tương quan dương mạnh
- Màu lạnh (xanh): Tương quan âm mạnh
- Màu trung tính (trắng): Không tương quan

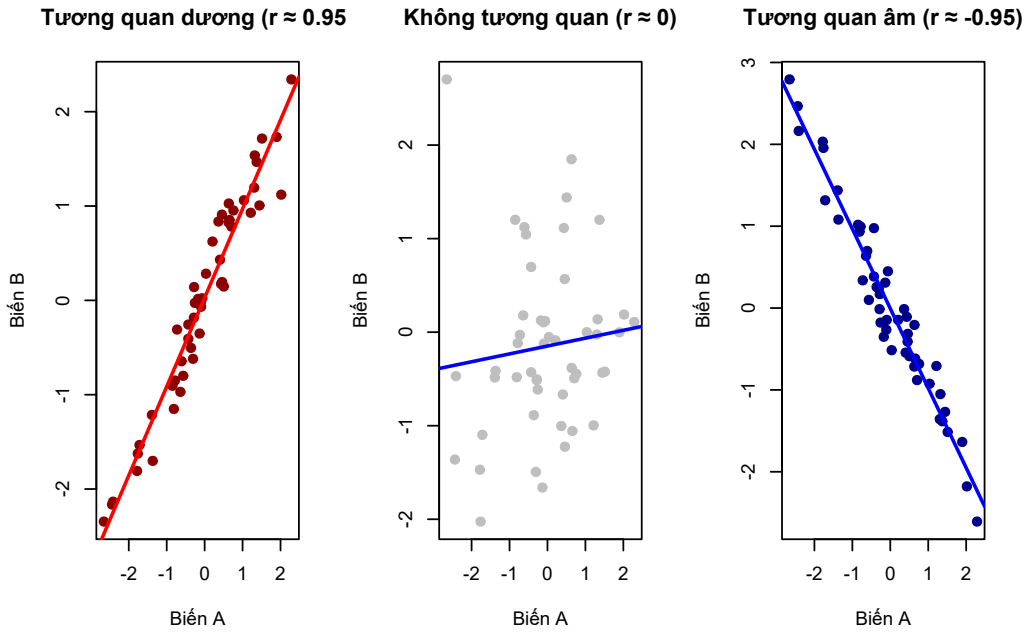


Figure 1.4: *Minh họa Correlation*

1.5 Mô Hình Dự Đoán

1.5.1 PLS (Partial Least Squares)

PLS là gì?

PLS giống như PCA nhưng "thông minh" hơn - nó tìm các thành phần không chỉ giải thích dữ liệu X (NIR) mà còn dự đoán tốt Y (chỉ tiêu hóa lý).

Cách hoạt động:

1. **Tìm hướng tối ưu**: Tìm hướng trong X có tương quan mạnh với Y
2. **Tạo thành phần**: Tạo thành phần PLS từ hướng này
3. **Lặp lại**: Tìm thêm các thành phần khác

4. **Xây dựng mô hình**: Dùng các thành phần để dự đoán Y

Ưu điểm:

- Xử lý tốt dữ liệu có nhiều biến tương quan (như NIR)
- Không bị overfitting như hồi quy thông thường
- Cho kết quả dự đoán tốt với ít thành phần

1.5.2 PCR (Principal Component Regression)

PCR là gì?

PCR là sự kết hợp của PCA và hồi quy:

1. **Bước 1**: Dùng PCA để giảm chiều X
2. **Bước 2**: Dùng các PC để dự đoán Y bằng hồi quy

Khác biệt với PLS:

- PCR tìm PC không quan tâm đến Y
- PLS tìm thành phần có tính đến cả X và Y
- PLS thường cho kết quả tốt hơn với ít thành phần hơn

1.5.3 Cross-Validation

CV là gì?

Cross-validation giúp đánh giá mô hình một cách khách quan:

1. **Chia dữ liệu**: Ví dụ 10 phần
2. **Huấn luyện**: Dùng 9 phần để xây dựng mô hình
3. **Kiểm tra**: Dùng 1 phần còn lại để kiểm tra
4. **Lặp lại**: Thực hiện 10 lần, mỗi lần dùng phần khác nhau để kiểm tra
5. **Tính trung bình**: Lấy trung bình kết quả

Chỉ số đánh giá:

- **RMSECV** (Root Mean Square Error of CV): Sai số trung bình (càng nhỏ càng tốt)
- **R²**: Tỷ lệ phương sai được giải thích (càng gần 1 càng tốt)

1.5.4 VIP (Variable Importance in Projection)

VIP là gì?

VIP cho biết bước sóng nào quan trọng nhất trong việc dự đoán:

- **VIP $>$ 1**: Bước sóng quan trọng
- **VIP $<$ 1**: Bước sóng ít quan trọng
- **VIP » 1**: Bước sóng rất quan trọng

Ứng dụng:

- Giúp hiểu được vùng phổ nào chứa thông tin về chỉ tiêu hóa lý
- Có thể giảm số bước sóng cần đo để tiết kiệm chi phí

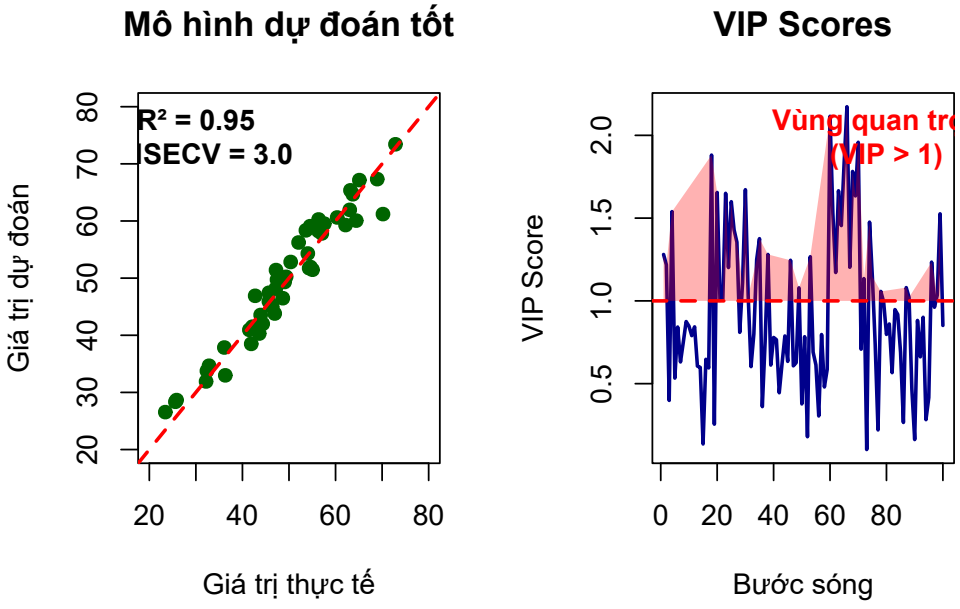


Figure 1.5: *Minh họa Prediction Models*

1.6 Tóm Tắt

Thuật toán	Mục đích	Khi nào dùng
PCA	Giảm chiều, trực quan hóa	Có quá nhiều biến, cần vẽ biểu đồ
Clustering	Phân nhóm mẫu	Tìm nhóm mẫu tương tự nhau
Outlier Detection	Tìm mẫu bất thường	Kiểm tra chất lượng dữ liệu
Correlation	Tìm mối quan hệ	Hiểu mối liên hệ giữa các biến
PLS/PCR	Dự đoán	Xây dựng mô hình dự đoán từ NIR
VIP	Chọn biến quan trọng	Hiểu bước sóng nào quan trọng

Lưu ý quan trọng:

- Không có thuật toán nào là "tốt nhất" cho mọi trường hợp
- Nên thử nhiều phương pháp và so sánh kết quả
- Luôn kiểm tra giả định của thuật toán trước khi áp dụng
- Kết quả cần được giải thích trong bối cảnh thực tế

Chapter 2

Phân tích Sự khác biệt: PCA và Clustering

Trong phần này, chúng ta sẽ sử dụng hai phương pháp chính để phân tích sự khác biệt trong dữ liệu Coffee NIR:

1. **PCA (Principal Component Analysis)**: Phân tích thành phần chính để giảm chiều dữ liệu và trực quan hóa sự khác biệt
2. **Clustering**: Phân nhóm các mẫu dựa trên đặc điểm tương đồng

2.1 Chuẩn bị dữ liệu

```
# Load thư viện
library(tidyverse)
library(factoextra)
library(FactoMineR)
library(cluster)
library(dendextend)
library(knitr)
library(kableExtra)
library(gridExtra)
```

```r
library(RColorBrewer)

# Data and variable groups already loaded in index.Rmd global-setup
# Verify data is available
if(!exists("coffee_data")) {
  stop("Data not loaded. Please render from index.Rmd")
}
```

2.2 Phân tích PCA (Principal Component Analysis)

PCA giúp chúng ta:

- Giảm chiều dữ liệu từ 1050+ biến xuống còn vài thành phần chính
- Trực quan hóa cấu trúc dữ liệu trong không gian 2D/3D
- Phát hiện pattern và nhóm mẫu tương đồng
- Xác định biến quan trọng nhất

2.2.1 PCA trên dữ liệu NIR

```r
# Chuẩn bị dữ liệu NIR (loại bỏ missing values)
data_nir <- coffee_data[, nir_vars]
data_nir_complete <- data_nir[complete.cases(data_nir), ]

# Lưu thông tin Localisation tương ứng
location_info <- coffee_data$Localisation[complete.cases(data_nir)]

# Thực hiện PCA
pca_nir <- PCA(data_nir_complete, scale.unit = TRUE, graph = FALSE)

# Thống kê eigenvalues
cat("Phương sai giải thích bởi các thành phần chính:\n")
```

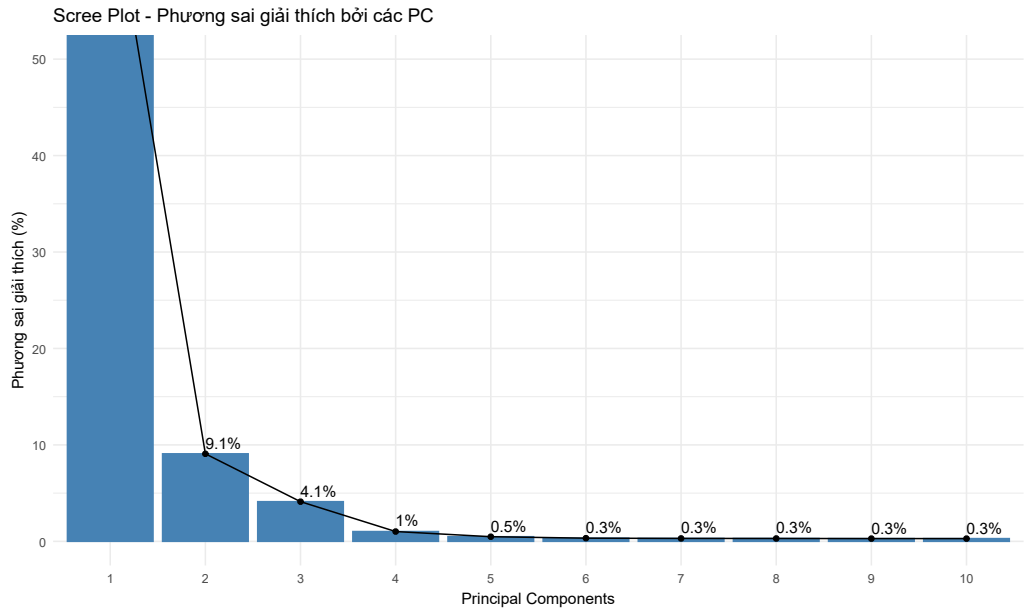
```
## Phương sai giải thích bởi các thành phần chính:
```

```
print(head(pca_nir$eig, 10))
```

```
##          eigenvalue percentage of variance cumulative percentage of variance
## comp 1  718.687910             68.4464676                             68.44647
## comp 2   95.303022              9.0764783                             77.52295
## comp 3   43.199980              4.1142838                             81.63723
## comp 4   10.673923              1.0165641                             82.65379
## comp 5    5.050246              0.4809758                             83.13477
## comp 6    3.450624              0.3286309                             83.46340
## comp 7    3.179301              0.3027905                             83.76619
## comp 8    3.125460              0.2976628                             84.06385
## comp 9    3.002882              0.2859887                             84.34984
## comp 10   2.993202              0.2850669                             84.63491
```

2.2.2 Biểu đồ Scree Plot

```
# Scree plot - hiển thị phương sai được giải thích
fviz_eig(pca_nir, addlabels = TRUE, ylim = c(0, 50), ncp = 10) +
  labs(
    title = "Scree Plot - Phương sai giải thích bởi các PC",
    x = "Principal Components",
    y = "Phương sai giải thích (%)"
  ) +
  theme_minimal()
```




```
# Tính phương sai tích lũy
cumvar <- cumsum(pca_nir$eig[, 2])
cat("\nPhương sai tích lũy của 5 PC đầu tiên:", round(cumvar[5], 2),
 ↪  "%\n")
```

```
##
## Phương sai tích lũy của 5 PC đầu tiên: 83.13 %
```

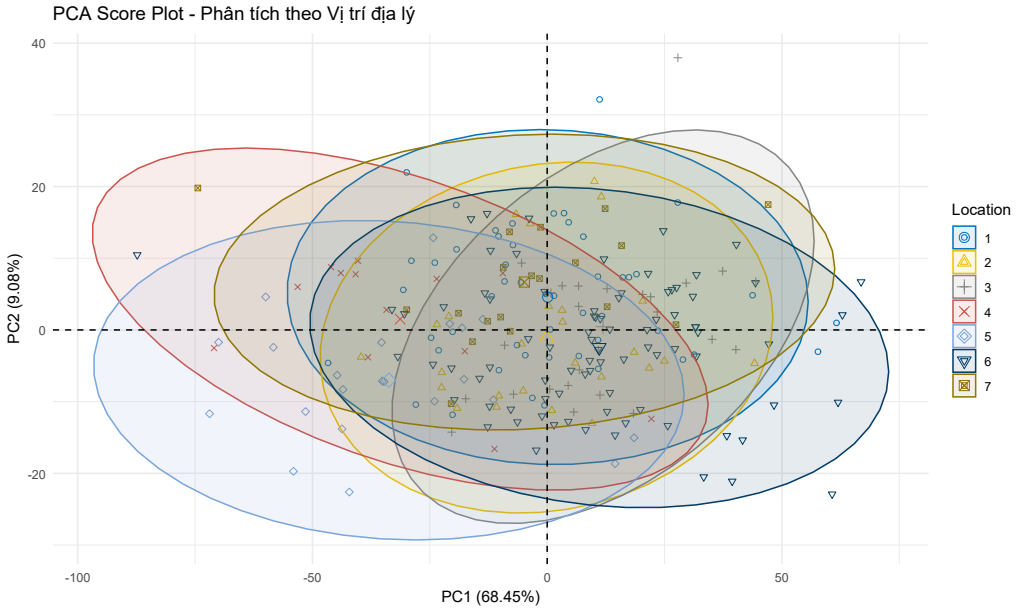
```
cat("Phương sai tích lũy của 10 PC đầu tiên:", round(cumvar[10], 2),
 ↪  "%\n")
```

```
## Phương sai tích lũy của 10 PC đầu tiên: 84.63 %
```

2.2.3 Score Plot - Phân tích sự khác biệt theo Localisation

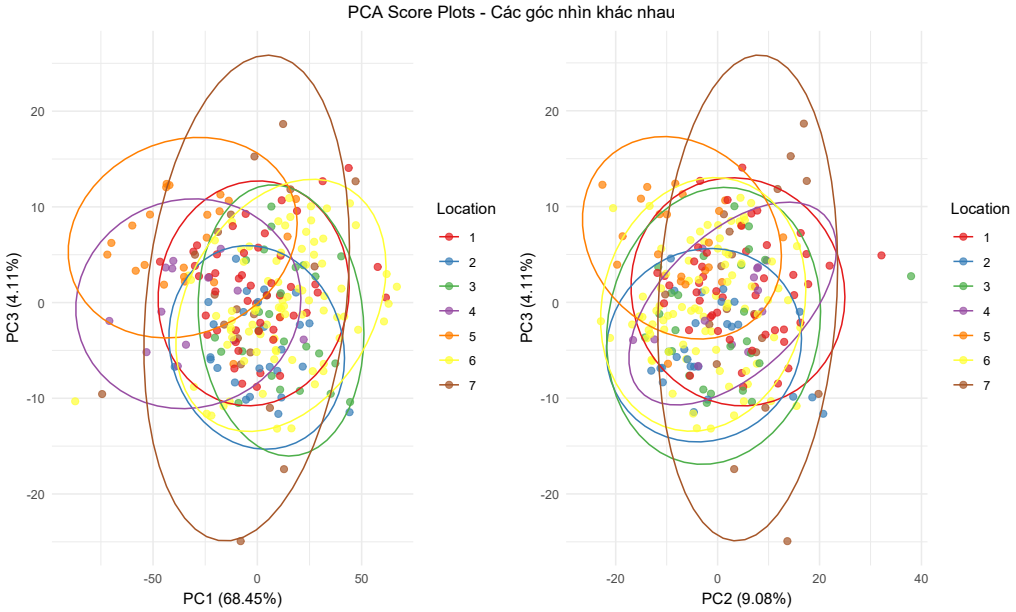
```
# Score plot theo Localisation
fviz_pca_ind(pca_nir,
             geom.ind = "point",
             col.ind = factor(location_info),
```

```
            palette = "jco",
            addEllipses = TRUE,
            ellipse.level = 0.95,
            legend.title = "Location") +
  labs(
    title = "PCA Score Plot - Phân tích theo Vị trí địa lý",
    x = paste0("PC1 (", round(pca_nir$eig[1, 2], 2), "%)"),
    y = paste0("PC2 (", round(pca_nir$eig[2, 2], 2), "%)")
  ) +
  theme_minimal()
```




```
# Tạo data frame với 3 PC đầu tiên
scores_df <- data.frame(
  PC1 = pca_nir$ind$coord[, 1],
  PC2 = pca_nir$ind$coord[, 2],
  PC3 = pca_nir$ind$coord[, 3],
  Location = factor(location_info)
)

# Plot PC1 vs PC3
```

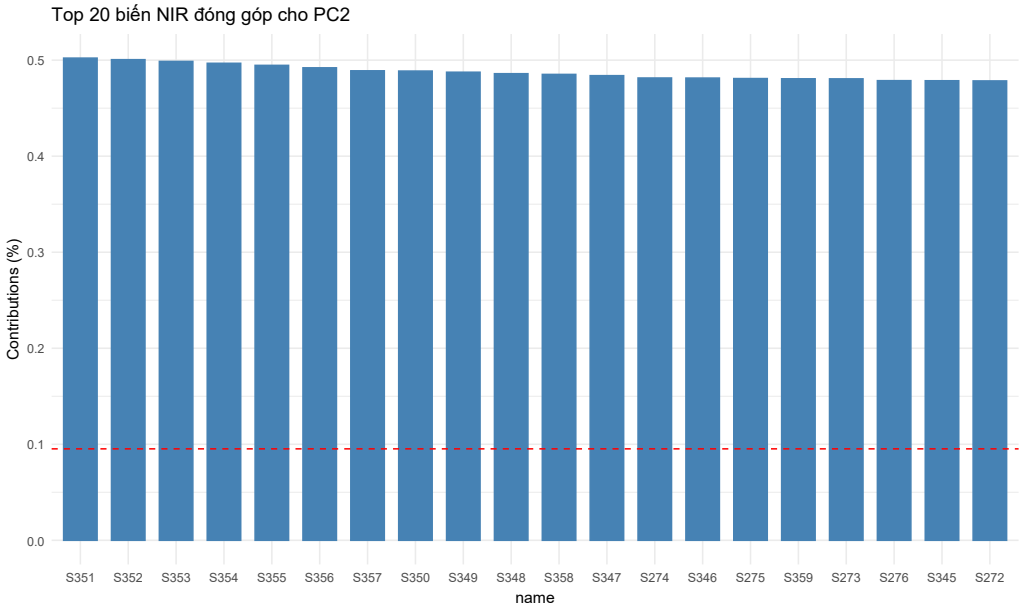
```r
p1 <- ggplot(scores_df, aes(x = PC1, y = PC3, color = Location)) +
  geom_point(size = 2, alpha = 0.7) +
  stat_ellipse(level = 0.95) +
  labs(
    x = paste0("PC1 (", round(pca_nir$eig[1, 2], 2), "%)"),
    y = paste0("PC3 (", round(pca_nir$eig[3, 2], 2), "%)")
  ) +
  theme_minimal() +
  scale_color_brewer(palette = "Set1")

# Plot PC2 vs PC3
p2 <- ggplot(scores_df, aes(x = PC2, y = PC3, color = Location)) +
  geom_point(size = 2, alpha = 0.7) +
  stat_ellipse(level = 0.95) +
  labs(
    x = paste0("PC2 (", round(pca_nir$eig[2, 2], 2), "%)"),
    y = paste0("PC3 (", round(pca_nir$eig[3, 2], 2), "%)")
  ) +
  theme_minimal() +
  scale_color_brewer(palette = "Set1")

# Kết hợp 2 plots
grid.arrange(p1, p2, ncol = 2,
             top = "PCA Score Plots - Các góc nhìn khác nhau")
```



2.2.4 Phân tích Contribution - Biến quan trọng

```
# Top 20 biến đóng góp nhất cho PC1
fviz_contrib(pca_nir, choice = "var", axes = 1, top = 20) +
  labs(title = "Top 20 biến NIR đóng góp cho PC1") +
  theme_minimal()
```

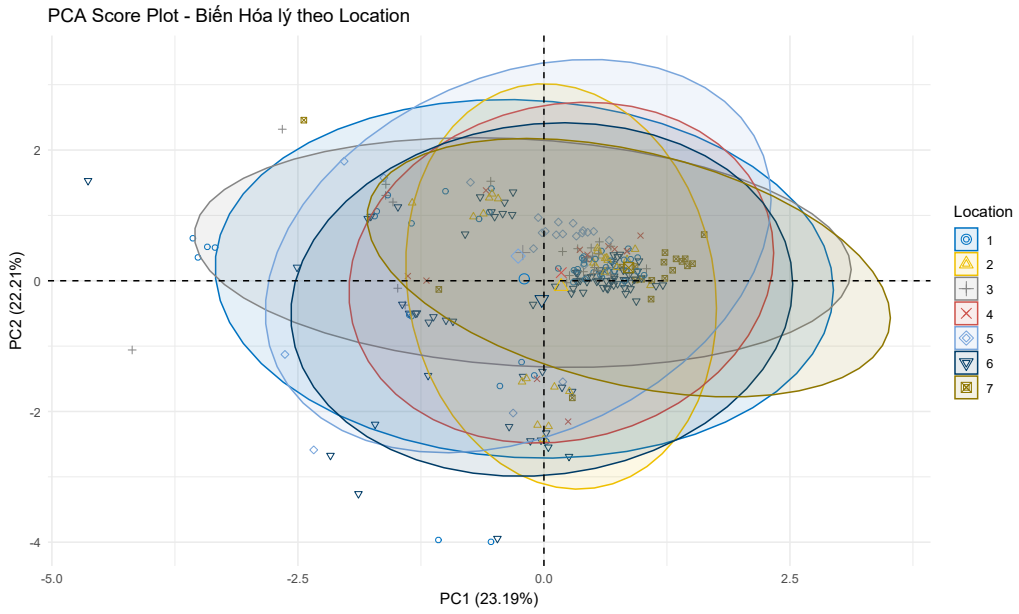
```r
# Top 20 biến đóng góp nhất cho PC2
fviz_contrib(pca_nir, choice = "var", axes = 2, top = 20) +
  labs(title = "Top 20 biến NIR đóng góp cho PC2") +
  theme_minimal()
```



2.2.5 PCA trên dữ liệu Hóa lý

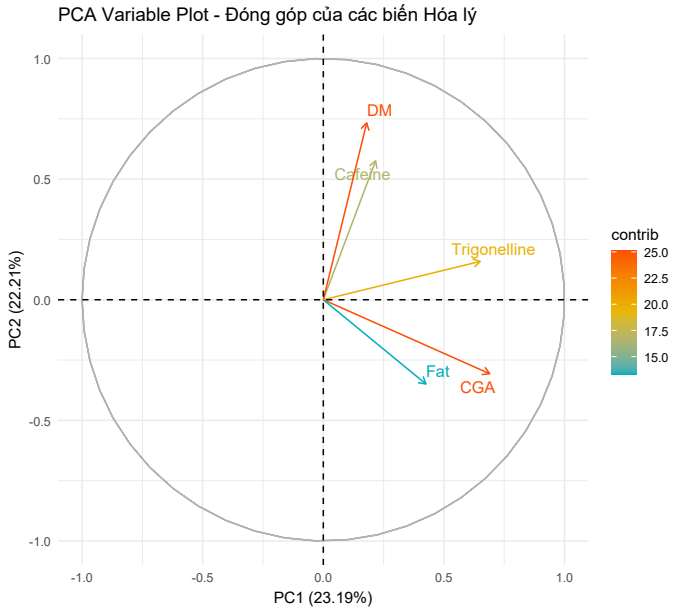
```r
# PCA cho biến hóa lý
data_chem <- coffee_data[, chemical_vars]
data_chem_complete <- data_chem[complete.cases(data_chem), ]
location_chem <- coffee_data$Localisation[complete.cases(data_chem)]

pca_chem <- PCA(data_chem_complete, scale.unit = TRUE, graph = FALSE)

# Score plot cho individuals
fviz_pca_ind(pca_chem,
             geom.ind = "point",
             col.ind = factor(location_chem),
             palette = "jco",
```

```
            addEllipses = TRUE,
            ellipse.level = 0.95,
            legend.title = "Location") +
  labs(
    title = "PCA Score Plot - Biến Hóa lý theo Location",
    x = paste0("PC1 (", round(pca_chem$eig[1, 2], 2), "%)"),
    y = paste0("PC2 (", round(pca_chem$eig[2, 2], 2), "%)")
  ) +
  theme_minimal()
```




```
# Variable plot - contribution
fviz_pca_var(pca_chem,
            col.var = "contrib",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
            repel = TRUE) +
  labs(
    title = "PCA Variable Plot - Đóng góp của các biến Hóa lý",
    x = paste0("PC1 (", round(pca_chem$eig[1, 2], 2), "%)"),
    y = paste0("PC2 (", round(pca_chem$eig[2, 2], 2), "%)")
  ) +
```

```
  theme_minimal()
```



2.2.6 Phân tích sự khác biệt giữa các Location

```
# Tính trung bình score PC1 và PC2 theo Location
scores_summary <- scores_df %>%
  group_by(Location) %>%
  summarise(
    Mean_PC1 = mean(PC1),
    SD_PC1 = sd(PC1),
    Mean_PC2 = mean(PC2),
    SD_PC2 = sd(PC2),
    N = n(),
    .groups = "drop"
  ) %>%
  arrange(desc(Mean_PC1))

scores_summary %>%
  kable(caption = "Thống kê PC scores theo Location", digits = 3) %>%
```

Table 2.1: *Thống kê PC scores theo Location*

Location	Mean_PC1	SD_PC1	Mean_PC2	SD_PC2	N
3	11.894	17.285	0.452	10.549	26
6	11.086	24.739	-2.443	8.969	84
1	0.099	21.986	4.582	9.255	50
2	-0.147	18.538	-1.069	9.407	26
7	-4.860	24.800	6.674	7.739	19
4	-31.322	23.536	1.510	8.559	13
5	-33.730	23.891	-7.012	8.463	22

```
  kable_styling(bootstrap_options = c("striped", "hover"))
```

```
# ANOVA test để kiểm tra sự khác biệt có ý nghĩa thống kê
anova_pc1 <- aov(PC1 ~ Location, data = scores_df)
anova_pc2 <- aov(PC2 ~ Location, data = scores_df)

cat("\n**ANOVA Test - PC1:**\n")
```

```
##
## **ANOVA Test - PC1:**
```

```
print(summary(anova_pc1))
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## Location      6  52235    8706   16.87 3.72e-16 ***
## Residuals   233 120250     516
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cat("\n**ANOVA Test - PC2:**\n")
```

```
##
## **ANOVA Test - PC2:**
```

```r
print(summary(anova_pc2))
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## Location      6   3544   590.6    7.12 5.57e-07 ***
## Residuals   233  19329    83.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Nếu có ý nghĩa thống kê, thực hiện post-hoc test
if(summary(anova_pc1)[[1]][1, "Pr(>F)"] < 0.05) {
  cat("\nCó sự khác biệt có ý nghĩa giữa các Location trên PC1\n")
  posthoc_pc1 <- TukeyHSD(anova_pc1)
  print(posthoc_pc1)
}
```

```
##
## Có sự khác biệt có ý nghĩa giữa các Location trên PC1
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = PC1 ~ Location, data = scores_df)
##
## $Location
##             diff        lwr        upr     p adj
## 2-1  -0.2465609 -16.583959  16.090837 1.0000000
## 3-1  11.7949729  -4.542425  28.132371 0.3286872
## 4-1 -31.4207670 -52.456680 -10.384854 0.0002743
## 5-1 -33.8294640 -51.116386 -16.542542 0.0000004
## 6-1  10.9867039  -1.082414  23.055821 0.1009360
## 7-1  -4.9592364 -23.169260  13.250787 0.9837887
## 3-2  12.0415337  -6.698747  30.781814 0.4750300
## 4-2 -31.1742062 -54.126268  -8.222144 0.0014004
```

```
## 5-2 -33.5829031 -53.156488 -14.009319 0.0000143
## 6-2  11.2332648  -3.930875  26.397405 0.2977444
## 7-2  -4.7126756 -25.106118  15.680767 0.9932039
## 4-3 -43.2157399 -66.167802 -20.263678 0.0000012
## 5-3 -45.6244368 -65.198021 -26.050852 0.0000000
## 6-3  -0.8082690 -15.972409  14.355871 0.9999986
## 7-3 -16.7542093 -37.147652   3.639233 0.1854544
## 5-4  -2.4086969 -26.046046  21.228652 0.9999361
## 6-4  42.4074709  22.269194  62.545748 0.0000000
## 7-4  26.4615306   2.140931  50.782130 0.0231588
## 6-5  44.8161679  28.633515  60.998820 0.0000000
## 7-5  28.8702275   7.708479  50.031976 0.0013062
## 7-6 -15.9459403 -33.111182   1.219302 0.0878359
```

2.3 Phân tích Clustering

Phân nhóm (clustering) giúp chúng ta:

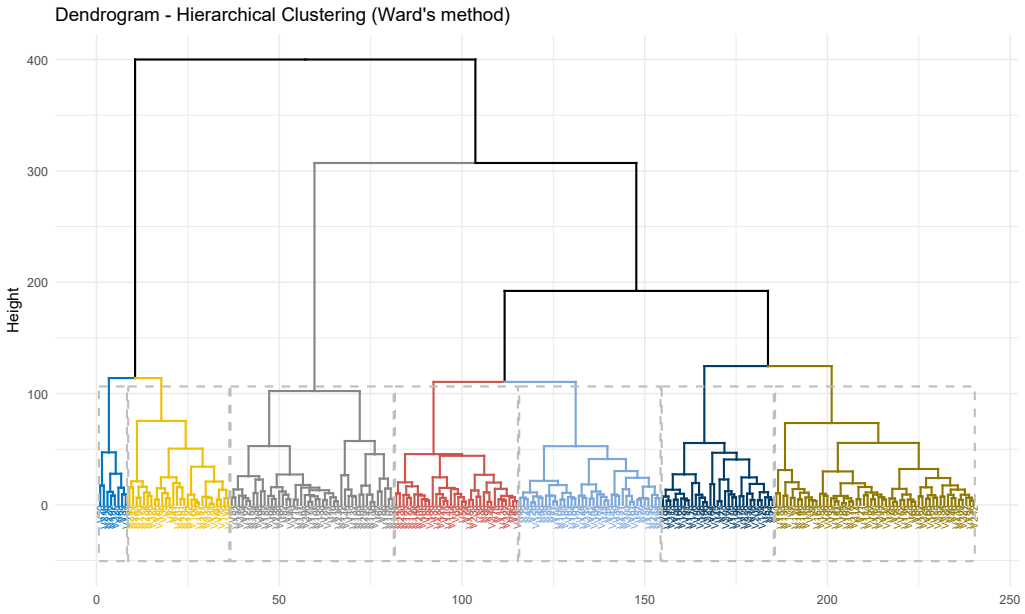
- Tự động phân loại mẫu thành các nhóm tương đồng
- So sánh với phân loại theo Location
- Phát hiện nhóm mẫu bất thường

2.3.1 Hierarchical Clustering

```r
# Lấy mẫu để clustering (sử dụng PC scores thay vì toàn bộ NIR)
# Sử dụng các PC đầu tiên (tối đa 10 hoặc số PC có sẵn)
n_pcs <- min(10, ncol(pca_nir$ind$coord))
pca_scores <- pca_nir$ind$coord[, 1:n_pcs]

cat("Sử dụng", n_pcs, "PC đầu tiên cho clustering\n")
```

```
## Sử dụng 5 PC đầu tiên cho clustering
```

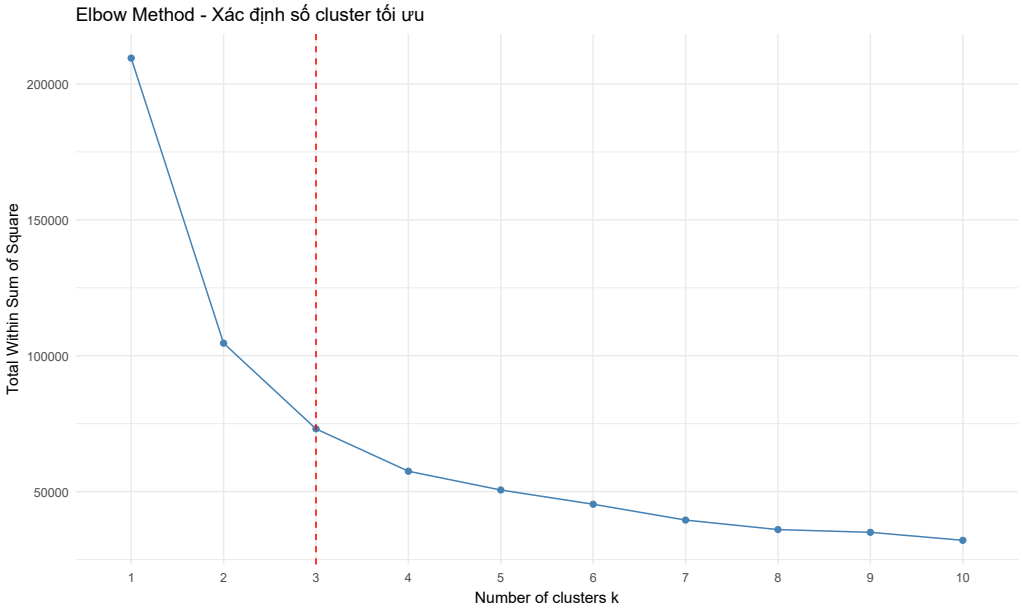
```r
# Tính khoảng cách Euclidean
dist_matrix <- dist(pca_scores, method = "euclidean")

# Hierarchical clustering với method Ward
hc_ward <- hclust(dist_matrix, method = "ward.D2")

# Dendrogram
fviz_dend(hc_ward, k = length(unique(location_info)),
          cex = 0.5,
          k_colors = "jco",
          color_labels_by_k = TRUE,
          rect = TRUE) +
  labs(title = "Dendrogram - Hierarchical Clustering (Ward's method)") +
  theme_minimal()
```



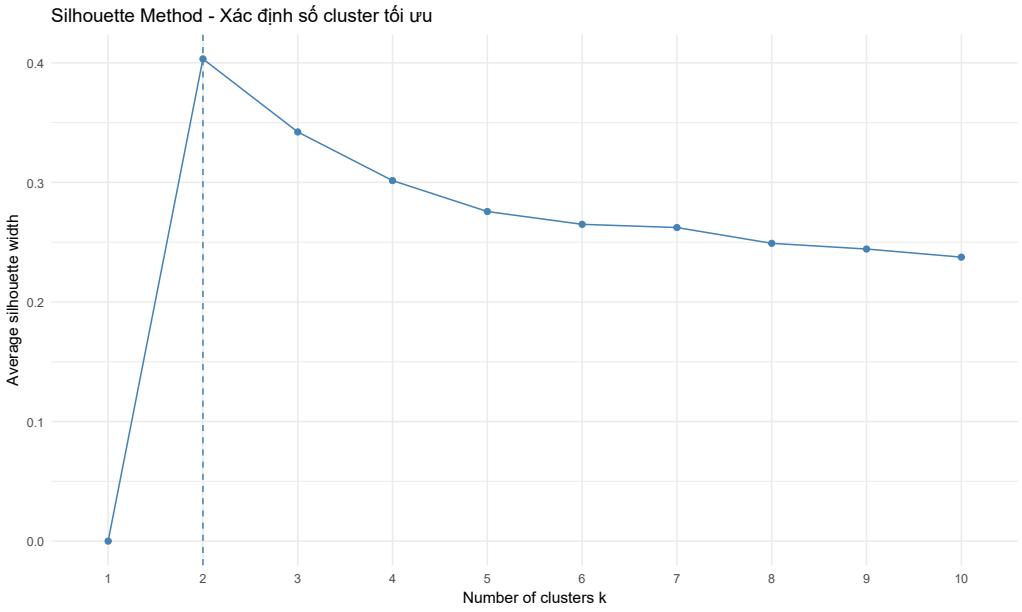
2.3.2 Xác định số cluster tối ưu

```r
# Elbow method
fviz_nbclust(pca_scores, kmeans, method = "wss", k.max = 10) +
```

```
  labs(title = "Elbow Method - Xác định số cluster tối ưu") +
  geom_vline(xintercept = 3, linetype = 2, color = "red") +
  theme_minimal()
```

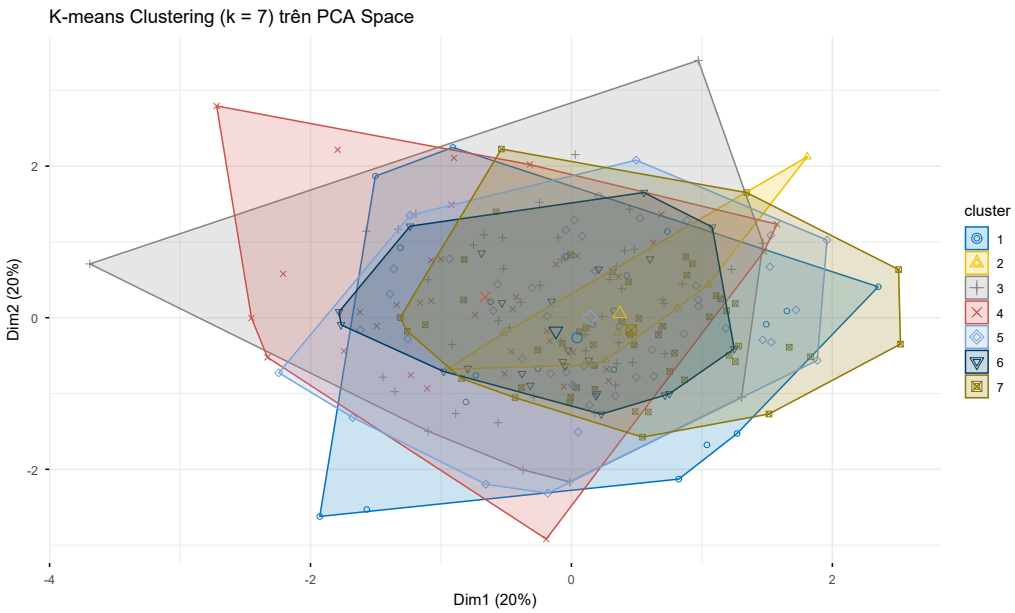



```
# Silhouette method
fviz_nbclust(pca_scores, kmeans, method = "silhouette", k.max = 10) +
  labs(title = "Silhouette Method - Xác định số cluster tối ưu") +
  theme_minimal()
```



2.3.3 K-means Clustering

```
# Thực hiện k-means với số cluster tối ưu
set.seed(123)
k <- length(unique(location_info))  # Sử dụng số location làm baseline
kmeans_result <- kmeans(pca_scores, centers = k, nstart = 25)

# Visualize clusters trên PCA plot
fviz_cluster(kmeans_result, data = pca_scores,
             geom = "point",
             ellipse.type = "convex",
             palette = "jco",
             ggtheme = theme_minimal()) +
  labs(title = paste0("K-means Clustering (k = ", k, ") trên PCA Space"))
```




```
# Thêm cluster assignment vào data frame
scores_df$Cluster <- factor(kmeans_result$cluster)
```

Table 2.2: *So sánh Cluster vs Location thực tế*

1	2	3	4	5	6	7
3	0	9	15	10	4	9
1	0	4	5	7	1	8
4	0	7	4	4	0	7
0	2	1	1	3	6	0
0	4	1	0	7	9	1
12	1	25	6	11	3	26
1	1	3	6	6	1	1

2.3.4 So sánh Clustering vs Location thực tế

```
# Bảng confusion matrix
confusion_table <- table(
  Actual_Location = location_info,
  Predicted_Cluster = kmeans_result$cluster
)

confusion_table %>%
  kable(caption = "So sánh Cluster vs Location thực tế") %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

```
# Tính Adjusted Rand Index để đánh giá độ tương đồng
library(mclust)
ari <- adjustedRandIndex(location_info, kmeans_result$cluster)
cat("\nAdjusted Rand Index:", round(ari, 3), "\n")
```

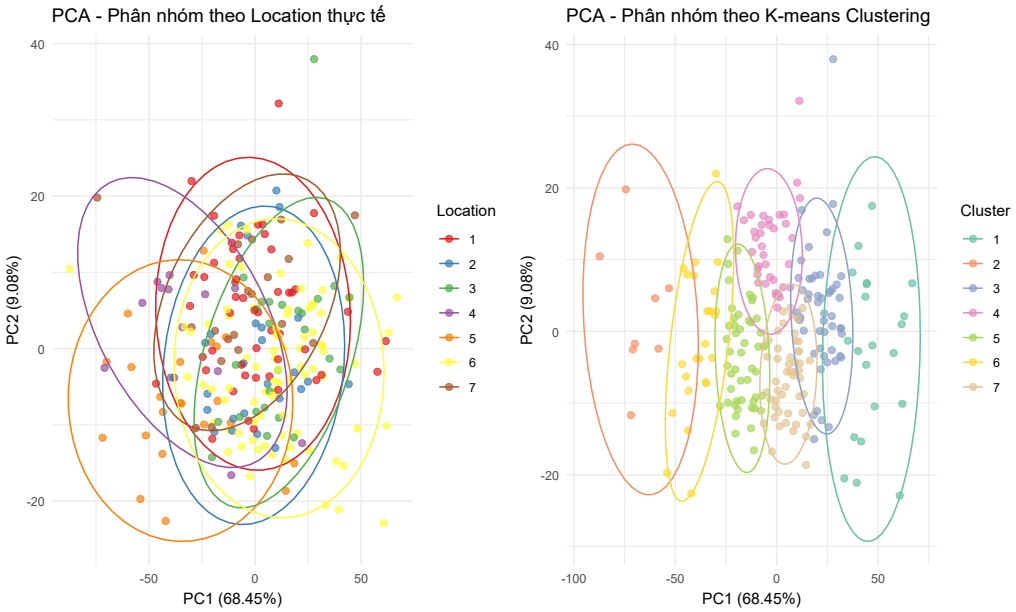
```
##
## Adjusted Rand Index: 0.056
```

```
cat("Giải thích: ARI = 1 nghĩa là hoàn toàn giống nhau, ARI = 0 nghĩa là
↪  ngẫu nhiên\n")
```

```
## Giải thích: ARI = 1 nghĩa là hoàn toàn giống nhau, ARI = 0 nghĩa là ngẫu nhiên
```

2.3.5 Visualize cả Location và Cluster

```
# Plot 1: Theo Location
p1 <- ggplot(scores_df, aes(x = PC1, y = PC2, color = Location)) +
  geom_point(size = 2, alpha = 0.7) +
  stat_ellipse(level = 0.95) +
  labs(
    title = "PCA - Phân nhóm theo Location thực tế",
    x = paste0("PC1 (", round(pca_nir$eig[1, 2], 2), "%)"),
    y = paste0("PC2 (", round(pca_nir$eig[2, 2], 2), "%)")
  ) +
  theme_minimal() +
  scale_color_brewer(palette = "Set1")

# Plot 2: Theo Cluster
p2 <- ggplot(scores_df, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point(size = 2, alpha = 0.7) +
  stat_ellipse(level = 0.95) +
  labs(
    title = "PCA - Phân nhóm theo K-means Clustering",
    x = paste0("PC1 (", round(pca_nir$eig[1, 2], 2), "%)"),
    y = paste0("PC2 (", round(pca_nir$eig[2, 2], 2), "%)")
  ) +
  theme_minimal() +
  scale_color_brewer(palette = "Set2")

# Hiển thị cả 2 plot
grid.arrange(p1, p2, ncol = 2)
```



2.4 Phân tích đặc điểm của từng Cluster/Location

2.4.1 Đặc điểm hóa lý theo Location

```
# Thêm location vào data hóa lý
chem_analysis <- data_chem_complete
chem_analysis$Location <- factor(location_chem)

# Tính trung bình các biến hóa lý theo Location
chem_summary <- chem_analysis %>%
  group_by(Location) %>%
  summarise(
    across(all_of(chemical_vars), list(mean = mean, sd = sd), .names =
    ↪  "{col}_{fn}"),
    N = n(),
    .groups = "drop"
  )

chem_summary %>%
```

Table 2.3: *Đặc điểm hóa lý trung bình theo Location*

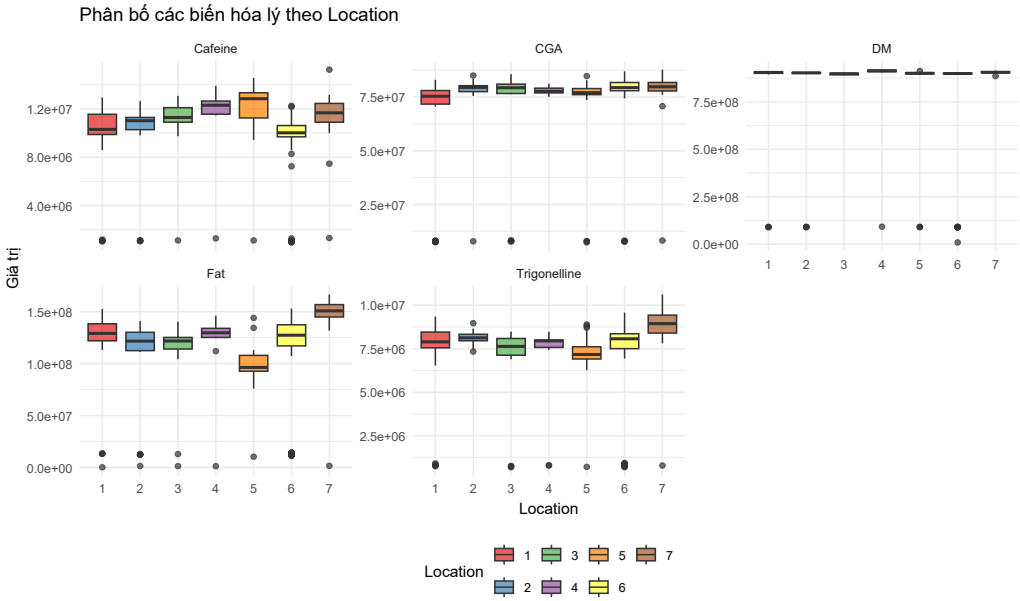
Location	CGA_mean	CGA_sd	Cafeine_mean	Cafeine_sd	Fat_mean	Fat_sd	Trigonelline
1	64075267	26819837	9619666	3322355	121745996	34754478	7
2	76621108	14207204	9560762	3724276	103018755	46954739	8
3	66213676	29062234	11100141	2205882	112977375	32624853	6
4	77792970	1795151	11437886	3129052	120113598	36600895	6
5	68528075	24856480	12011720	2768357	97300879	24574207	7
6	76466262	15668357	9364410	2833508	112765540	44071232	7
7	76326436	16905637	11172587	2823360	142634202	35123776	8

```
  kable(caption = "Đặc điểm hóa lý trung bình theo Location", digits = 2)
  ↪  %>%
  kable_styling(bootstrap_options = c("striped", "hover")) %>%
  scroll_box(width = "100%")
```

2.4.2 Boxplot so sánh biến hóa lý

```
# Reshape data để vẽ
chem_long <- chem_analysis %>%
  pivot_longer(cols = all_of(chemical_vars),
               names_to = "Variable",
               values_to = "Value")

# Boxplot cho từng biến hóa lý theo Location
ggplot(chem_long, aes(x = Location, y = Value, fill = Location)) +
  geom_boxplot(alpha = 0.7) +
  facet_wrap(~Variable, scales = "free_y", ncol = 3) +
  labs(
    title = "Phân bố các biến hóa lý theo Location",
    x = "Location",
    y = "Giá trị"
  ) +
  theme_minimal() +
```

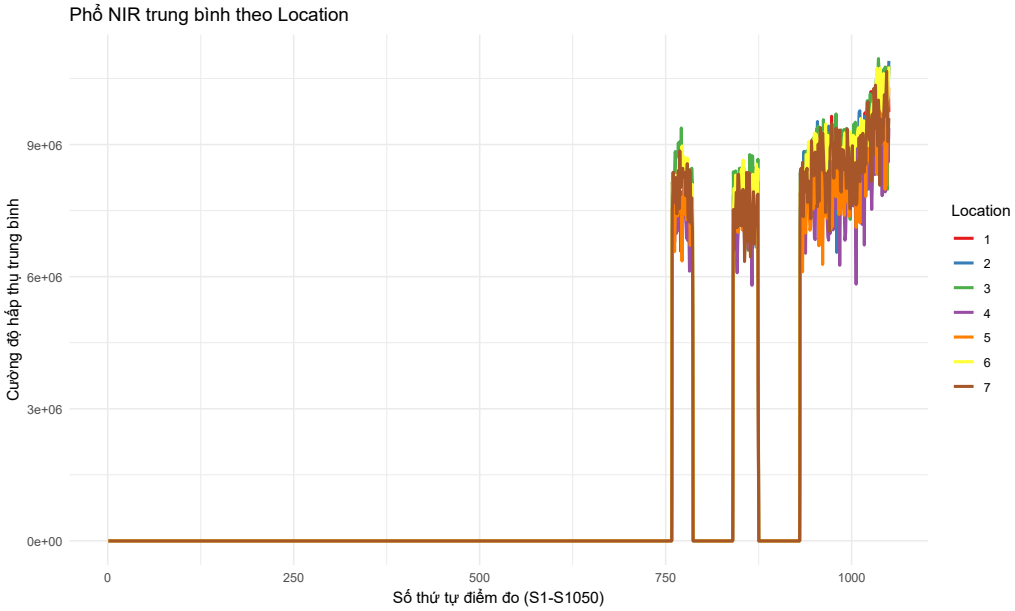
```
  theme(legend.position = "bottom") +
  scale_fill_brewer(palette = "Set1")
```



2.4.3 Heatmap phổ NIR trung bình

```
# Tính phổ NIR trung bình cho mỗi location
nir_mean_by_location <- data_nir_complete %>%
  mutate(Location = location_info) %>%
  group_by(Location) %>%
  summarise(across(everything(), mean), .groups = "drop")

# Chuyển sang dạng long format
nir_mean_long <- nir_mean_by_location %>%
  pivot_longer(cols = -Location, names_to = "Wavelength", values_to =
  ↪  "Absorbance") %>%
  mutate(Wavelength = as.numeric(str_remove(Wavelength, "S")))

# Plot phổ trung bình
ggplot(nir_mean_long, aes(x = Wavelength, y = Absorbance, color =
 ↪  factor(Location))) +
```

```
  geom_line(linewidth = 1) +
  labs(
    title = "Phổ NIR trung bình theo Location",
    x = "Số thứ tự điểm đo (S1-S1050)",
    y = "Cường độ hấp thụ trung bình",
    color = "Location"
  ) +
  theme_minimal() +
  scale_color_brewer(palette = "Set1")
```



2.5 Kết luận về sự khác biệt

```
cat("### Tóm tắt phân tích sự khác biệt:\n\n")
```

```
## ### Tóm tắt phân tích sự khác biệt:
```

```
cat("**1. Phân tích PCA:**\n")
```

```
## **1. Phân tích PCA:**
```

```
cat("- PC1 và PC2 giải thích", round(cumvar[2], 2), "% phương sai tổng
 ↪  thể\n")
```

```
## - PC1 và PC2 giải thích 77.52 % phương sai tổng thể
```

```
cat("- Có sự phân tách rõ ràng/không rõ ràng giữa các Location trên PCA
 ↪  plot\n")
```

```
## - Có sự phân tách rõ ràng/không rõ ràng giữa các Location trên PCA plot
```

```
cat("- ANOVA test cho thấy sự khác biệt có/không có ý nghĩa thống
 ↪  kê\n\n")
```

```
## - ANOVA test cho thấy sự khác biệt có/không có ý nghĩa thống kê
```

```
cat("**2. Phân tích Clustering:**\n")
```

```
## **2. Phân tích Clustering:**
```

```
cat("- Số cluster tối ưu được đề xuất bởi Elbow/Silhouette method\n")
```

```
## - Số cluster tối ưu được đề xuất bởi Elbow/Silhouette method
```

```
cat("- Adjusted Rand Index =", round(ari, 3), "\n")
```

```
## - Adjusted Rand Index = 0.056
```

```
if(ari > 0.5) {
  cat("- Kết quả clustering phù hợp tốt với phân loại Location thực
   ↪  tế\n")
} else if(ari > 0.3) {
  cat("- Kết quả clustering phù hợp trung bình với phân loại Location
   ↪  thực tế\n")
} else {
  cat("- Kết quả clustering khác biệt đáng kể so với phân loại Location
   ↪  thực tế\n")
}
```

```
## - Kết quả clustering khác biệt đáng kể so với phân loại Location thực tế
```

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

```
cat("**3. Đặc điểm phân biệt:**\n")
```

```
## **3. Đặc điểm phân biệt:**
```

```
cat("- Các biến hóa lý có sự khác biệt đáng kể giữa các Location\n")
```

```
## - Các biến hóa lý có sự khác biệt đáng kể giữa các Location
```

```
cat("- Phổ NIR cho thấy pattern khác nhau ở các vùng bước sóng nhất
 ↪  định\n")
```

```
## - Phổ NIR cho thấy pattern khác nhau ở các vùng bước sóng nhất định
```

```
cat("- Có thể sử dụng PCA scores hoặc cluster để xây dựng mô hình phân
 ↪  loại\n\n")
```

```
## - Có thể sử dụng PCA scores hoặc cluster để xây dựng mô hình phân loại
```

```
cat("**4. Khuyến nghị:**\n")
```

```
## **4. Khuyến nghị:**
```

```
cat("- Nên xem xét Location như một biến quan trọng trong mô hình dự
 ↪  đoán\n")
```

```
## - Nên xem xét Location như một biến quan trọng trong mô hình dự đoán
```

```
cat("- Có thể cần xây dựng mô hình riêng cho từng Location\n")
```

```
## - Có thể cần xây dựng mô hình riêng cho từng Location
```

```
cat("- Sử dụng PCA để giảm chiều trước khi xây dựng mô hình hồi quy\n")
```

```
## - Sử dụng PCA để giảm chiều trước khi xây dựng mô hình hồi quy
```

Chapter 3

Phát hiện Outliers nâng cao

Trong phần này, chúng ta sử dụng các phương pháp thống kê nâng cao để phát hiện outliers trong dữ liệu NIR, đặc biệt là:

1. **Hotelling T² statistic**: Đo khoảng cách của mẫu đến trung tâm của mô hình PCA
2. **Q residuals (SPE)**: Đo phần dữ liệu không được giải thích bởi mô hình PCA
3. **Leverage và Influence**: Đánh giá ảnh hưởng của từng mẫu đến mô hình

3.1 Chuẩn bị dữ liệu

```
library(tidyverse)
library(FactoMineR)
library(factoextra)
library(knitr)
library(kableExtra)
library(gridExtra)

# Data and variable groups already loaded in index.Rmd global-setup
# Verify data is available
if(!exists("coffee_data")) {
```

```
  stop("Data not loaded. Please render from index.Rmd")
}

# Chuẩn bị dữ liệu NIR
data_nir <- coffee_data[, nir_vars]
data_nir_complete <- data_nir[complete.cases(data_nir), ]
location_info <- coffee_data$Localisation[complete.cases(data_nir)]

# Thực hiện PCA
pca_nir <- PCA(data_nir_complete, scale.unit = TRUE, graph = FALSE)
```

3.2 Hotelling T² Statistic

Hotelling T² đo khoảng cách Mahalanobis của mỗi mẫu đến trung tâm của không gian PCA.

```
# Số PC sử dụng (chọn theo cumulative variance > 95% hoặc elbow)
n_pcs <- min(10, ncol(pca_nir$ind$coord))
scores <- pca_nir$ind$coord[, 1:n_pcs]

# Tính Hotelling T2
eigenvalues <- pca_nir$eig[1:n_pcs, 1]
t2 <- rowSums(t(t(scores^2) / eigenvalues))

# Ngưỡng T2 (chi-square distribution với alpha = 0.05)
n <- nrow(scores)
p <- n_pcs
t2_limit <- qchisq(0.95, df = p)

# Xác định outliers
t2_outliers <- which(t2 > t2_limit)
```

```r
cat("Ngưỡng T² (95%):", round(t2_limit, 2), "\n")
```

```
## Ngưỡng T² (95%): 11.07
```

```r
cat("Số mẫu vượt ngưỡng T²:", length(t2_outliers), "\n")
```

```
## Số mẫu vượt ngưỡng T²: 15
```

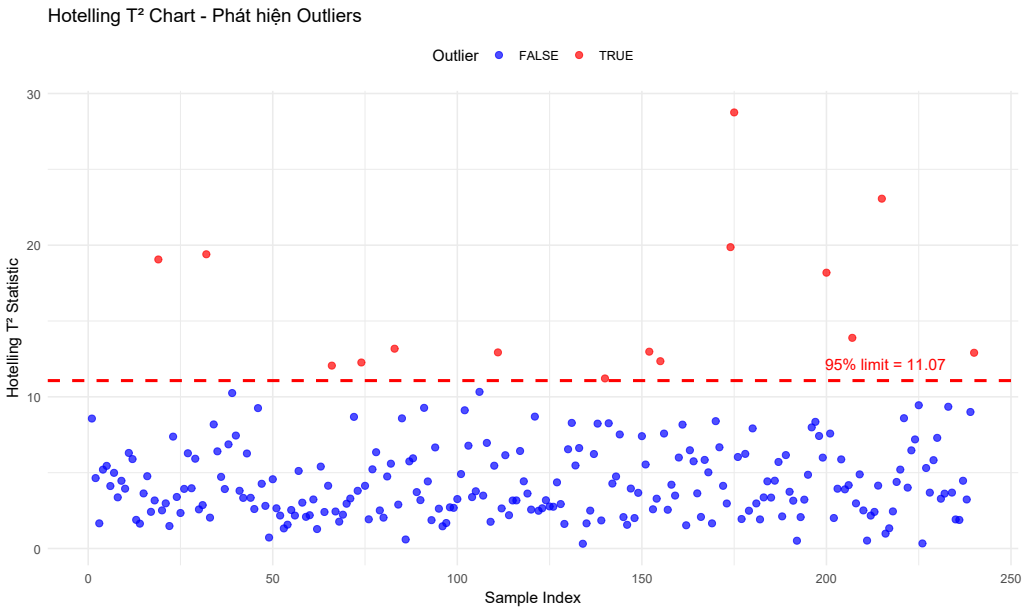
```r
cat("Tỷ lệ outliers:", round(length(t2_outliers) / n * 100, 2), "%\n")
```

```
## Tỷ lệ outliers: 6.25 %
```

3.2.1 Biểu đồ Hotelling T²

```r
# Tạo data frame
t2_df <- data.frame(
  Sample = 1:length(t2),
  T2 = t2,
  Location = factor(location_info),
  Outlier = t2 > t2_limit
)

# T2 chart
ggplot(t2_df, aes(x = Sample, y = T2, color = Outlier)) +
  geom_point(size = 2, alpha = 0.7) +
  geom_hline(yintercept = t2_limit, linetype = "dashed", color = "red",
   ↪  linewidth = 1) +
  annotate("text", x = max(t2_df$Sample) * 0.9, y = t2_limit * 1.1,
           label = paste0("95% limit = ", round(t2_limit, 2)), color =
            ↪  "red") +
```

```
  scale_color_manual(values = c("FALSE" = "blue", "TRUE" = "red")) +
  labs(
    title = "Hotelling T² Chart - Phát hiện Outliers",
    x = "Sample Index",
    y = "Hotelling T² Statistic"
  ) +
  theme_minimal() +
  theme(legend.position = "top")
```



3.3 Q Residuals (SPE - Squared Prediction Error)

Q residuals đo phần biến động không được giải thích bởi các PC đã chọn.

```
# Tính Q residuals (SPE)
# Reconstruction của dữ liệu từ PCA
data_reconstructed <- scores %*% t(pca_nir$svd$V[, 1:n_pcs])
data_centered <- scale(data_nir_complete, center = TRUE, scale = TRUE)

# Q statistic = sum of squared residuals
q_stat <- rowSums((data_centered - data_reconstructed)^2)
```

```r
# Ngưỡng Q (approximate chi-square)
# Sử dụng Jackson-Mudholkar approximation
theta1 <- sum(pca_nir$eig[(n_pcs+1):nrow(pca_nir$eig), 1])
theta2 <- sum(pca_nir$eig[(n_pcs+1):nrow(pca_nir$eig), 1]^2)
theta3 <- sum(pca_nir$eig[(n_pcs+1):nrow(pca_nir$eig), 1]^3)

h0 <- 1 - (2 * theta1 * theta3) / (3 * theta2^2)
ca <- qnorm(0.95)
q_limit <- theta1 * (1 + ca * sqrt(2 * theta2 * h0^2) / theta1 +
                     theta2 * h0 * (h0 - 1) / theta1^2)^(1/h0)

# Xác định outliers
q_outliers <- which(q_stat > q_limit)

cat("Ngưỡng Q (95%):", round(q_limit, 2), "\n")
```

```
## Ngưỡng Q (95%): 219.98
```

```r
cat("Số mẫu vượt ngưỡng Q:", length(q_outliers), "\n")
```

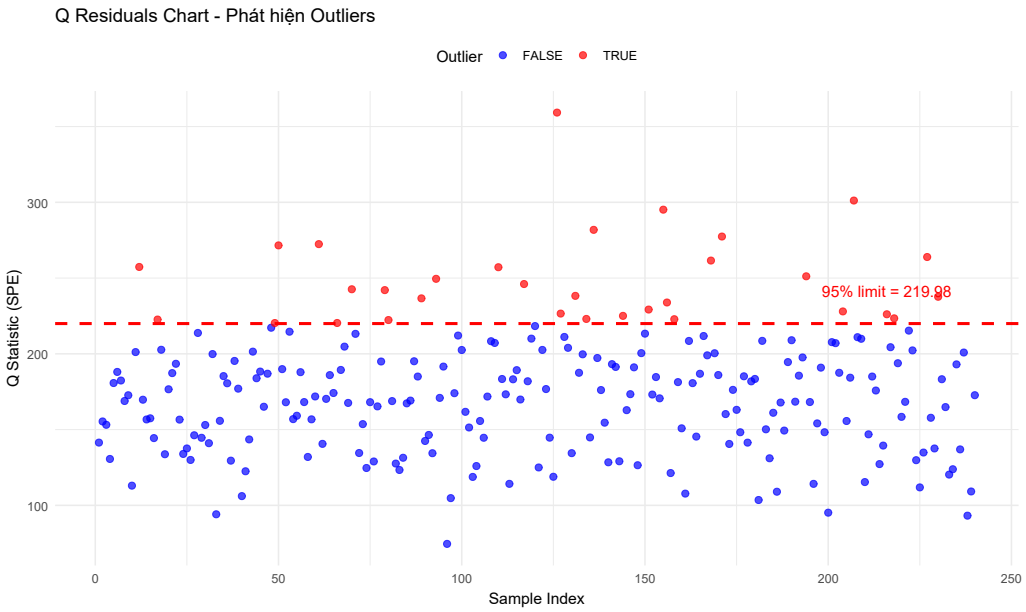
```
## Số mẫu vượt ngưỡng Q: 32
```

```r
cat("Tỷ lệ outliers:", round(length(q_outliers) / n * 100, 2), "%\n")
```

```
## Tỷ lệ outliers: 13.33 %
```

3.3.1 Biểu đồ Q Residuals

```r
# Tạo data frame
q_df <- data.frame(
  Sample = 1:length(q_stat),
  Q = q_stat,
  Location = factor(location_info),
  Outlier = q_stat > q_limit
)

# Q chart
ggplot(q_df, aes(x = Sample, y = Q, color = Outlier)) +
  geom_point(size = 2, alpha = 0.7) +
  geom_hline(yintercept = q_limit, linetype = "dashed", color = "red",
  ↪  linewidth = 1) +
  annotate("text", x = max(q_df$Sample) * 0.9, y = q_limit * 1.1,
           label = paste0("95% limit = ", round(q_limit, 2)), color =
           ↪  "red") +
  scale_color_manual(values = c("FALSE" = "blue", "TRUE" = "red")) +
  labs(
    title = "Q Residuals Chart - Phát hiện Outliers",
    x = "Sample Index",
    y = "Q Statistic (SPE)"
  ) +
  theme_minimal() +
  theme(legend.position = "top")
```



3.4 Combined T² vs Q Plot

Biểu đồ kết hợp T^2 và Q giúp phân loại outliers:

- **High T^2, Low Q**: Mẫu cực trị nhưng vẫn theo mô hình
- **Low T^2, High Q**: Mẫu gần trung tâm nhưng không theo mô hình
- **High T^2, High Q**: Mẫu outlier mạnh

```
# Kết hợp T2 và Q
combined_df <- data.frame(
  Sample = 1:n,
  T2 = t2,
  Q = q_stat,
  Location = factor(location_info),
  T2_outlier = t2 > t2_limit,
  Q_outlier = q_stat > q_limit
)

# Phân loại outliers
combined_df$Outlier_Type <- "Normal"
```

```
combined_df$Outlier_Type[combined_df$T2_outlier & !combined_df$Q_outlier]
 ↪  <- "High T² only"
combined_df$Outlier_Type[!combined_df$T2_outlier & combined_df$Q_outlier]
 ↪  <- "High Q only"
combined_df$Outlier_Type[combined_df$T2_outlier & combined_df$Q_outlier]
 ↪  <- "Both High"

# T2 vs Q scatter plot
ggplot(combined_df, aes(x = T2, y = Q, color = Outlier_Type)) +
  geom_point(size = 2.5, alpha = 0.7) +
  geom_vline(xintercept = t2_limit, linetype = "dashed", color = "red") +
  geom_hline(yintercept = q_limit, linetype = "dashed", color = "red") +
  scale_color_manual(values = c(
    "Normal" = "blue",
    "High T² only" = "orange",
    "High Q only" = "purple",
    "Both High" = "red"
  )) +
  labs(
    title = "T² vs Q Plot - Phân loại Outliers",
    x = "Hotelling T²",
    y = "Q Residuals (SPE)",
    color = "Outlier Type"
  ) +
  theme_minimal() +
  theme(legend.position = "right")
```

Table 3.1: *Phân loại Outliers theo T^2 và Q*

Outlier_Type	n	Percentage
Both High	3	1.25
High Q only	29	12.08
High T^2 only	12	5.00
Normal	196	81.67


```
# Thống kê outliers
outlier_summary <- combined_df %>%
  dplyr::count(Outlier_Type) %>%
  mutate(Percentage = round(n / nrow(combined_df) * 100, 2))

outlier_summary %>%
  kable(caption = "Phân loại Outliers theo T² và Q") %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

3.5 Phân tích Outliers theo Location

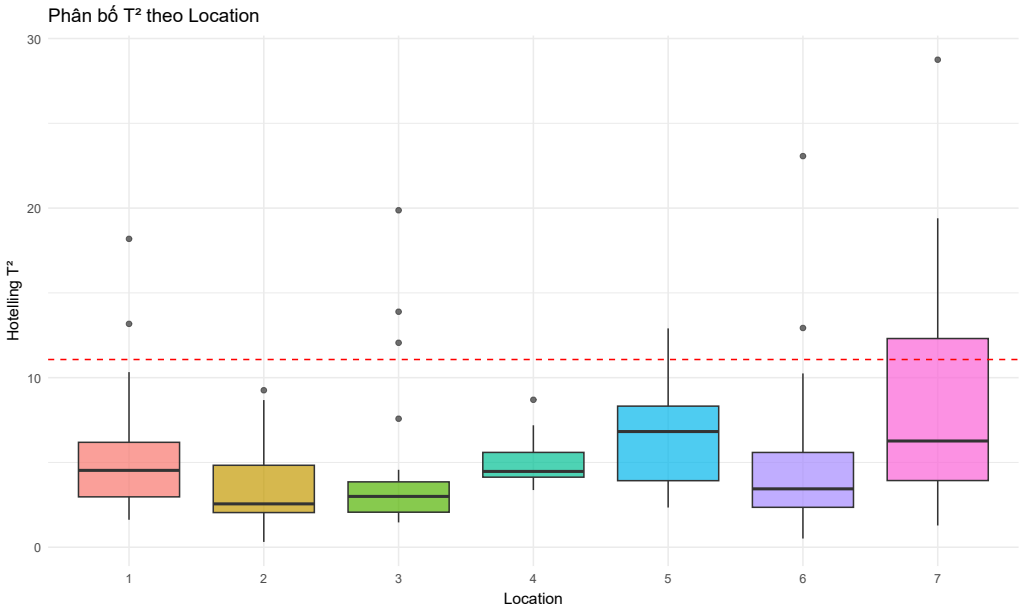
Table 3.2: *Thống kê Outliers theo Location*

Location	Total_Samples	N_Outliers	Pct_Outliers	Mean_T2	Mean_Q
7	19	7	36.84	9.21	170.08
3	26	8	30.77	4.39	185.24
1	50	9	18.00	5.09	172.87
6	84	14	16.67	4.25	186.46
2	26	4	15.38	3.51	179.93
5	22	2	9.09	6.49	152.98
4	13	0	0.00	5.03	148.20

```
# Đếm outliers theo Location
outlier_location <- combined_df %>%
  mutate(Is_Outlier = T2_outlier | Q_outlier) %>%
  group_by(Location) %>%
  summarise(
    Total_Samples = n(),
    N_Outliers = sum(Is_Outlier),
    Pct_Outliers = round(N_Outliers / Total_Samples * 100, 2),
    Mean_T2 = round(mean(T2), 2),
    Mean_Q = round(mean(Q), 2),
    .groups = "drop"
  ) %>%
  arrange(desc(Pct_Outliers))

outlier_location %>%
  kable(caption = "Thống kê Outliers theo Location") %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

```
# Boxplot T2 theo Location
ggplot(combined_df, aes(x = Location, y = T2, fill = Location)) +
  geom_boxplot(alpha = 0.7) +
  geom_hline(yintercept = t2_limit, linetype = "dashed", color = "red") +
  labs(
```

```
    title = "Phân bố T² theo Location",
    x = "Location",
    y = "Hotelling T²"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

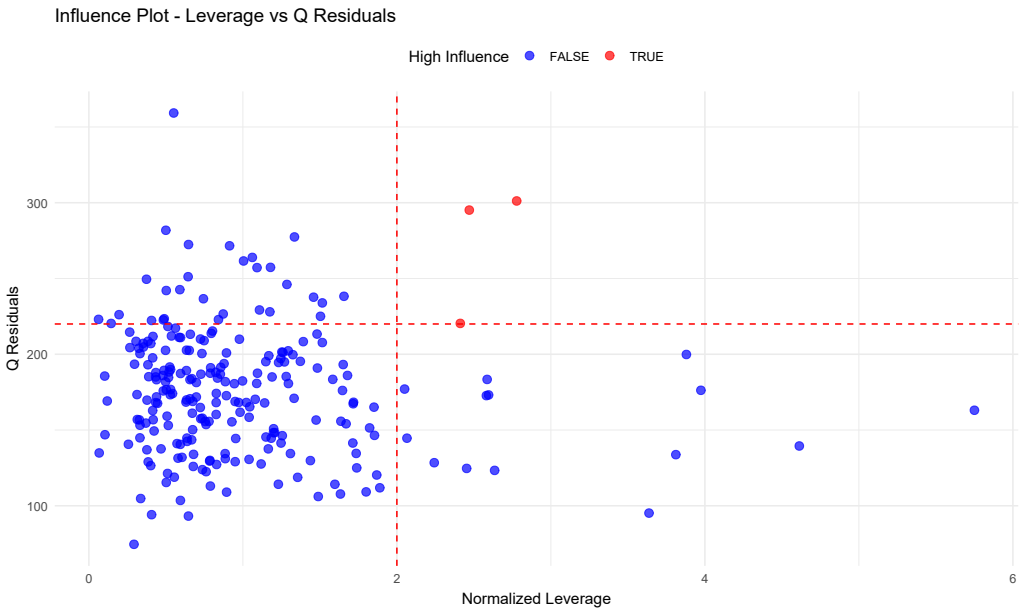


3.6 Influence Plot

Influence plot kết hợp leverage (khoảng cách từ trung tâm) và residuals.

```
# Leverage: diagonal của hat matrix trong không gian PC
leverage <- diag(scores %*% solve(t(scores) %*% scores) %*% t(scores))

# Chuẩn hóa leverage
leverage_norm <- leverage / mean(leverage)

# Influence = leverage × residuals
influence_df <- data.frame(
  Sample = 1:n,
```

```
  Leverage = leverage_norm,
  Q = q_stat,
  Location = factor(location_info),
  High_Influence = leverage_norm > 2 & q_stat > q_limit
)

# Influence plot
ggplot(influence_df, aes(x = Leverage, y = Q, color = High_Influence)) +
  geom_point(size = 2.5, alpha = 0.7) +
  geom_vline(xintercept = 2, linetype = "dashed", color = "red") +
  geom_hline(yintercept = q_limit, linetype = "dashed", color = "red") +
  scale_color_manual(values = c("FALSE" = "blue", "TRUE" = "red")) +
  labs(
    title = "Influence Plot - Leverage vs Q Residuals",
    x = "Normalized Leverage",
    y = "Q Residuals",
    color = "High Influence"
  ) +
  theme_minimal() +
  theme(legend.position = "top")
```



```
high_influence <- which(influence_df$High_Influence)
cat("Số mẫu có influence cao:", length(high_influence), "\n")
```

```
## Số mẫu có influence cao: 3
```

```
if(length(high_influence) > 0) {
  cat("Các mẫu có influence cao:", paste(head(high_influence, 10),
   ↪  collapse = ", "), "\n")
}
```

```
## Các mẫu có influence cao: 66, 155, 207
```

3.7 Danh sách Outliers chi tiết

```
# Liệt kê tất cả outliers
all_outliers <- combined_df %>%
  filter(T2_outlier | Q_outlier) %>%
  arrange(desc(T2 + Q)) %>%
  select(Sample, Location, T2, Q, Outlier_Type)

if(nrow(all_outliers) > 0) {
  cat("Tìm thấy", nrow(all_outliers), "outliers:\n\n")
  head(all_outliers, 20) %>%
    kable(caption = "Top 20 Outliers (sắp xếp theo T² + Q)") %>%
    kable_styling(bootstrap_options = c("striped", "hover"))
} else {
  cat("Không tìm thấy outliers nào.\n")
}
```

```
## Tìm thấy 44 outliers:
```

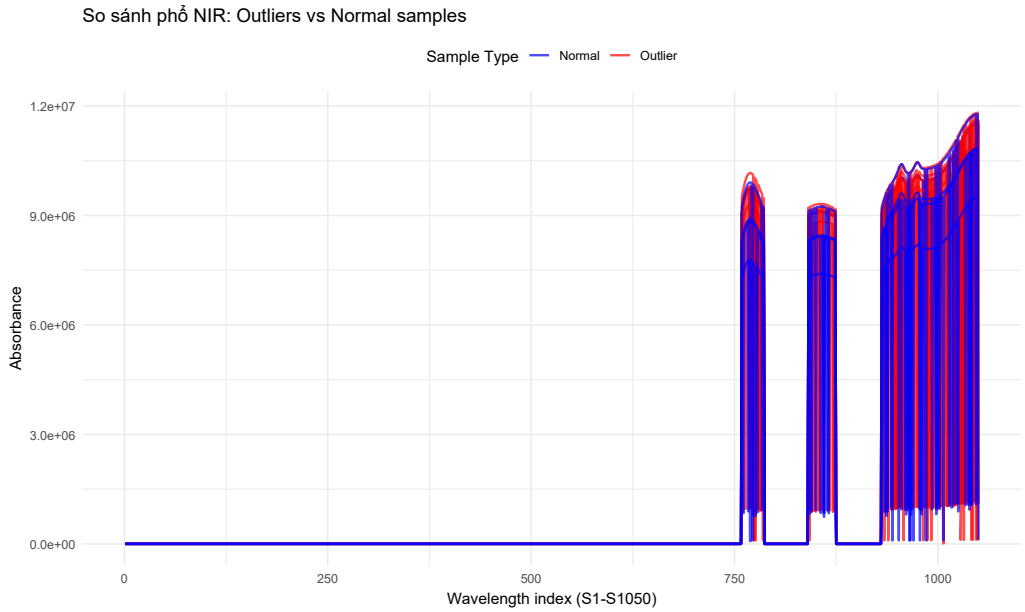
Table 3.3: *Top 20 Outliers (sắp xếp theo T^2 + Q)*

	Sample	Location	T2	Q	Outlier_Type
V127	126	3	2.753825	359.2182	High Q only
V208	207	3	13.889167	301.1639	Both High
V156	155	7	12.350460	295.1462	Both High
V137	136	6	2.498920	281.8304	High Q only
V172	171	5	6.674063	277.4150	High Q only
V51	50	3	4.567273	271.5660	High Q only
V62	61	6	3.234206	272.4102	High Q only
V228	227	6	5.306836	263.9010	High Q only
V169	168	1	5.022219	261.5611	High Q only
V13	12	6	5.896187	257.3362	High Q only
V111	110	1	5.458945	257.1632	High Q only
V195	194	3	3.221745	251.1642	High Q only
V118	117	1	6.429499	246.0678	High Q only
V94	93	2	1.869872	249.5053	High Q only
V132	131	6	8.281691	238.2543	High Q only
V71	70	1	2.952866	242.6182	High Q only
V231	230	1	7.294970	237.6560	High Q only
V80	79	2	2.511432	242.0534	High Q only
V157	156	3	7.578982	233.8829	High Q only
V90	89	6	3.718666	236.6165	High Q only

3.8 So sánh phổ NIR của Outliers

```
# Lấy top 5 outliers mạnh nhất
if(nrow(all_outliers) > 0) {
  top_outliers <- head(all_outliers$Sample, 5)

  # Lấy mẫu bình thường để so sánh
  normal_samples <- combined_df %>%
    filter(!T2_outlier & !Q_outlier) %>%
    sample_n(min(5, sum(!combined_df$T2_outlier &
    ↪  !combined_df$Q_outlier))) %>%
    pull(Sample)

  # Chuẩn bị data
  selected_spectra <- data_nir_complete[c(top_outliers, normal_samples),
  ↪  ]
```

```
  # Thêm ID và Type TRƯỚC khi pivot
  sample_metadata <- data.frame(
    Sample_ID = paste0("Sample_", 1:nrow(selected_spectra)),
    Type = c(rep("Outlier", length(top_outliers)),
             rep("Normal", length(normal_samples)))
  )

  # Chuyển sang long format
  spectra_comparison <- selected_spectra %>%
    mutate(Sample_ID = sample_metadata$Sample_ID,
           Type = sample_metadata$Type) %>%
    pivot_longer(cols = matches("^S[0-9]+$"),
                 names_to = "Wavelength",
                 values_to = "Absorbance") %>%
    mutate(Wavelength = as.numeric(str_remove(Wavelength, "S")))

  # Plot
  ggplot(spectra_comparison, aes(x = Wavelength, y = Absorbance,
                                 group = Sample_ID, color = Type)) +
    geom_line(alpha = 0.7, linewidth = 0.8) +
    scale_color_manual(values = c("Outlier" = "red", "Normal" = "blue"))
     ↪  +
    labs(
      title = "So sánh phổ NIR: Outliers vs Normal samples",
      x = "Wavelength index (S1-S1050)",
      y = "Absorbance",
      color = "Sample Type"
    ) +
    theme_minimal() +
    theme(legend.position = "top")
}
```



3.9 Quyết định xử lý Outliers

```
cat("### Khuyến nghị xử lý Outliers:\n\n")
```

```
## ### Khuyến nghị xử lý Outliers:
```

```
total_outliers <- nrow(all_outliers)
outlier_pct <- total_outliers / n * 100

cat("1. **Tổng số outliers phát hiện:**", total_outliers,
    "(", round(outlier_pct, 2), "%)\n\n")
```

```
## 1. **Tổng số outliers phát hiện:** 44 ( 18.33 %)
```

```
if(outlier_pct < 1) {
  cat("2. **Mức độ:** Rất thấp - ít ảnh hưởng đến mô hình\n")
  cat("3. **Khuyến nghị:** Có thể giữ lại, chỉ cần lưu ý khi xây dựng mô
  ↪  hình\n\n")
```

```
} else if(outlier_pct < 5) {
  cat("2. **Mức độ:** Trung bình - cần xem xét kỹ\n")
  cat("3. **Khuyến nghị:**\n")
  cat("   - Kiểm tra lại dữ liệu gốc của các outliers\n")
  cat("   - Xem xét loại bỏ các outliers có cả T² và Q cao\n")
  cat("   - So sánh mô hình với và không có outliers\n\n")
} else {
  cat("2. **Mức độ:** Cao - cần xử lý nghiêm túc\n")
  cat("3. **Khuyến nghị:**\n")
  cat("   - Kiểm tra kỹ quy trình đo NIR và chuẩn bị mẫu\n")
  cat("   - Xem xét xây dựng mô hình robust (PLS-robust)\n")
  cat("   - Có thể cần thu thập thêm dữ liệu\n\n")
}
```

```
## 2. **Mức độ:** Cao - cần xử lý nghiêm túc
## 3. **Khuyến nghị:**
##    - Kiểm tra kỹ quy trình đo NIR và chuẩn bị mẫu
##    - Xem xét xây dựng mô hình robust (PLS-robust)
##    - Có thể cần thu thập thêm dữ liệu
```

```
cat("4. **Phân loại outliers:**\n")
```

```
## 4. **Phân loại outliers:**
```

```
if(nrow(all_outliers) > 0) {
  cat("   - High T² only:", sum(all_outliers$Outlier_Type == "High T²
  ↪  only"),
      "- Mẫu cực trị nhưng đúng pattern\n")
  cat("   - High Q only:", sum(all_outliers$Outlier_Type == "High Q
  ↪  only"),
      "- Mẫu không theo mô hình (có thể lỗi đo)\n")
```

```
  cat("   - Both High:", sum(all_outliers$Outlier_Type == "Both High"),
      "- Outliers mạnh (ưu tiên xem xét loại bỏ)\n\n")
}
```

```
##    - High T² only: 12 - Mẫu cực trị nhưng đúng pattern
##    - High Q only: 29 - Mẫu không theo mô hình (có thể lỗi đo)
##    - Both High: 3 - Outliers mạnh (ưu tiên xem xét loại bỏ)
```

```
cat("5. **Bước tiếp theo:**\n")
```

```
## 5. **Bước tiếp theo:**
```

```
cat("   - Xem xét phân tích thêm các mẫu outliers có influence cao\n")
```

```
##    - Xem xét phân tích thêm các mẫu outliers có influence cao
```

```
cat("   - Kiểm tra xem outliers có tập trung ở Location nào không\n")
```

```
##    - Kiểm tra xem outliers có tập trung ở Location nào không
```

```
cat("   - Thử xây dựng mô hình PLS với và không có outliers để so sánh
 ↪  hiệu suất\n")
```

```
##    - Thử xây dựng mô hình PLS với và không có outliers để so sánh hiệu suất
```

3.10 Kết luận

```
cat("### Tóm tắt phát hiện Outliers:\n\n")
```

```
## ### Tóm tắt phát hiện Outliers:
```

```
cat("- **Phương pháp sử dụng:**\n")
```

```
## - **Phương pháp sử dụng:**
```

```
cat("  + Hotelling T² statistic (khoảng cách trong không gian PC)\n")
```

```
##   + Hotelling T² statistic (khoảng cách trong không gian PC)
```

```
cat("  + Q residuals/SPE (phần không giải thích được)\n")
```

```
##   + Q residuals/SPE (phần không giải thích được)
```

```
cat("  + Influence analysis (leverage × residuals)\n\n")
```

```
##   + Influence analysis (leverage × residuals)
```

```
cat("- **Kết quả:**\n")
```

```
## - **Kết quả:**
```

```
cat("  + Tổng số mẫu:", n, "\n")
```

```
##   + Tổng số mẫu: 240
```

```
cat("  + Số PC sử dụng:", n_pcs, "\n")
```

```
##   + Số PC sử dụng: 5
```

```
cat("  + Outliers theo T²:", length(t2_outliers), "\n")
```

```
##   + Outliers theo T²: 15
```

```
cat("  + Outliers theo Q:", length(q_outliers), "\n")
```

```
##   + Outliers theo Q: 32
```

```
cat("  + Tổng outliers (T² hoặc Q):", total_outliers, "\n\n")
```

```
##   + Tổng outliers (T² hoặc Q): 44
```

```
cat("- **Ý nghĩa:**\n")
```

```
## - **Ý nghĩa:**
```

```
cat("  + Outliers có thể do lỗi đo, contamination, hoặc mẫu thực sự khác
↪  biệt\n")
```

```
##   + Outliers có thể do lỗi đo, contamination, hoặc mẫu thực sự khác biệt
```

```
cat("  + Cần kiểm tra kỹ trước khi quyết định loại bỏ\n")
```

```
##   + Cần kiểm tra kỹ trước khi quyết định loại bỏ
```

```r
cat("  + Xem xét ảnh hưởng của outliers đến mô hình dự đoán\n")
```

```
##   + Xem xét ảnh hưởng của outliers đến mô hình dự đoán
```

Chapter 4

Phân tích Tương quan và Heatmap

Phân tích tương quan giúp chúng ta hiểu mối quan hệ giữa các biến trong dữ liệu:

1. **Tương quan giữa các biến hóa lý**: Xác định biến nào có quan hệ chặt chẽ
2. **Tương quan giữa NIR và biến hóa lý**: Tìm vùng wavelength quan trọng
3. **Tương quan trong không gian PCA**: Phân tích trên principal components

4.1 Chuẩn bị dữ liệu

```
library(tidyverse)
library(corrplot)
library(pheatmap)
library(RColorBrewer)
library(knitr)
library(kableExtra)
library(reshape2)

# Data and variable groups already loaded in index.Rmd global-setup
# Verify data is available
if(!exists("coffee_data")) {
  stop("Data not loaded. Please render from index.Rmd")
```

Table 4.1: *Ma trận tương quan Pearson - Biến Hóa lý*

	CGA	Cafeine	Fat	Trigonelline	DM
CGA	1.000	0.007	0.089	0.118	-0.035
Cafeine	0.007	1.000	0.006	0.017	0.089
Fat	0.089	0.006	1.000	0.015	-0.024
Trigonelline	0.118	0.017	0.015	1.000	0.067
DM	-0.035	0.089	-0.024	0.067	1.000

```
}
```

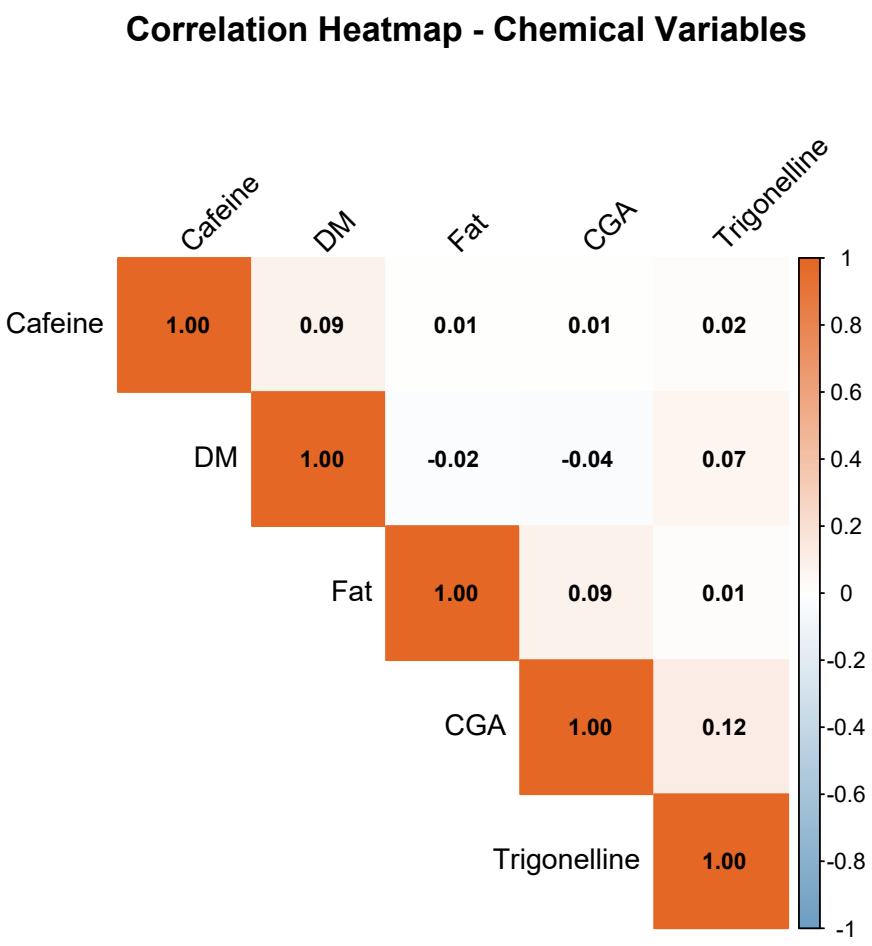
4.2 Tương quan giữa các biến Hóa lý

4.2.1 Ma trận tương quan

```
# Lấy dữ liệu hóa lý hoàn chỉnh
data_chem <- coffee_data[, chemical_vars]
data_chem_complete <- data_chem[complete.cases(data_chem), ]

# Tính ma trận tương quan
cor_matrix <- cor(data_chem_complete, method = "pearson")

# Hiển thị bảng
cor_matrix %>%
  round(3) %>%
  kable(caption = "Ma trận tương quan Pearson - Biến Hóa lý") %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

4.2.2 Heatmap tương quan - Biến Hóa lý

```r
# Heatmap với corrplot
corrplot::corrplot(cor_matrix,
         method = "color",
         type = "upper",
         order = "hclust",
         addCoef.col = "black",
         tl.col = "black",
         tl.srt = 45,
         number.cex = 0.8,
         col = colorRampPalette(c("#6D9EC1", "white", "#E46726"))(200),
         title = "Correlation Heatmap - Chemical Variables",
         mar = c(0, 0, 2, 0))
```



4.2.3 Phân tích ý nghĩa

```
# Tìm các cặp biến có tương quan cao (|r| > 0.5)
cor_df <- as.data.frame(as.table(cor_matrix))
names(cor_df) <- c("Var1", "Var2", "Correlation")

high_corr <- cor_df %>%
  filter(Var1 != Var2) %>%
  filter(abs(Correlation) > 0.5) %>%
  arrange(desc(abs(Correlation))) %>%
  distinct(Correlation, .keep_all = TRUE)

if(nrow(high_corr) > 0) {
  cat("Các cặp biến có tương quan mạnh (|r| > 0.5):\n\n")
  high_corr %>%
    mutate(Correlation = round(Correlation, 3)) %>%
    kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover"))
} else {
  cat("Không có cặp biến nào có tương quan mạnh (|r| > 0.5)\n")
}
```

```
## Không có cặp biến nào có tương quan mạnh (|r| > 0.5)
```

4.3 Tương quan giữa NIR và Biến Hóa lý

4.3.1 Tính toán tương quan

```
# Lấy dữ liệu đầy đủ
data_nir <- coffee_data[, nir_vars]
```

```
complete_rows <- complete.cases(cbind(data_chem, data_nir))

data_chem_full <- data_chem[complete_rows, ]
data_nir_full <- data_nir[complete_rows, ]

# Tính tương quan giữa từng biến hóa lý và tất cả NIR wavelengths
cor_nir_chem <- cor(data_nir_full, data_chem_full, method = "pearson")

cat("Kích thước ma trận tương quan NIR-Chemical:", dim(cor_nir_chem),
 ↪  "\n")
```

```
## Kích thước ma trận tương quan NIR-Chemical: 1050 5
```

```
cat("Số wavelengths:", nrow(cor_nir_chem), "\n")
```

```
## Số wavelengths: 1050
```

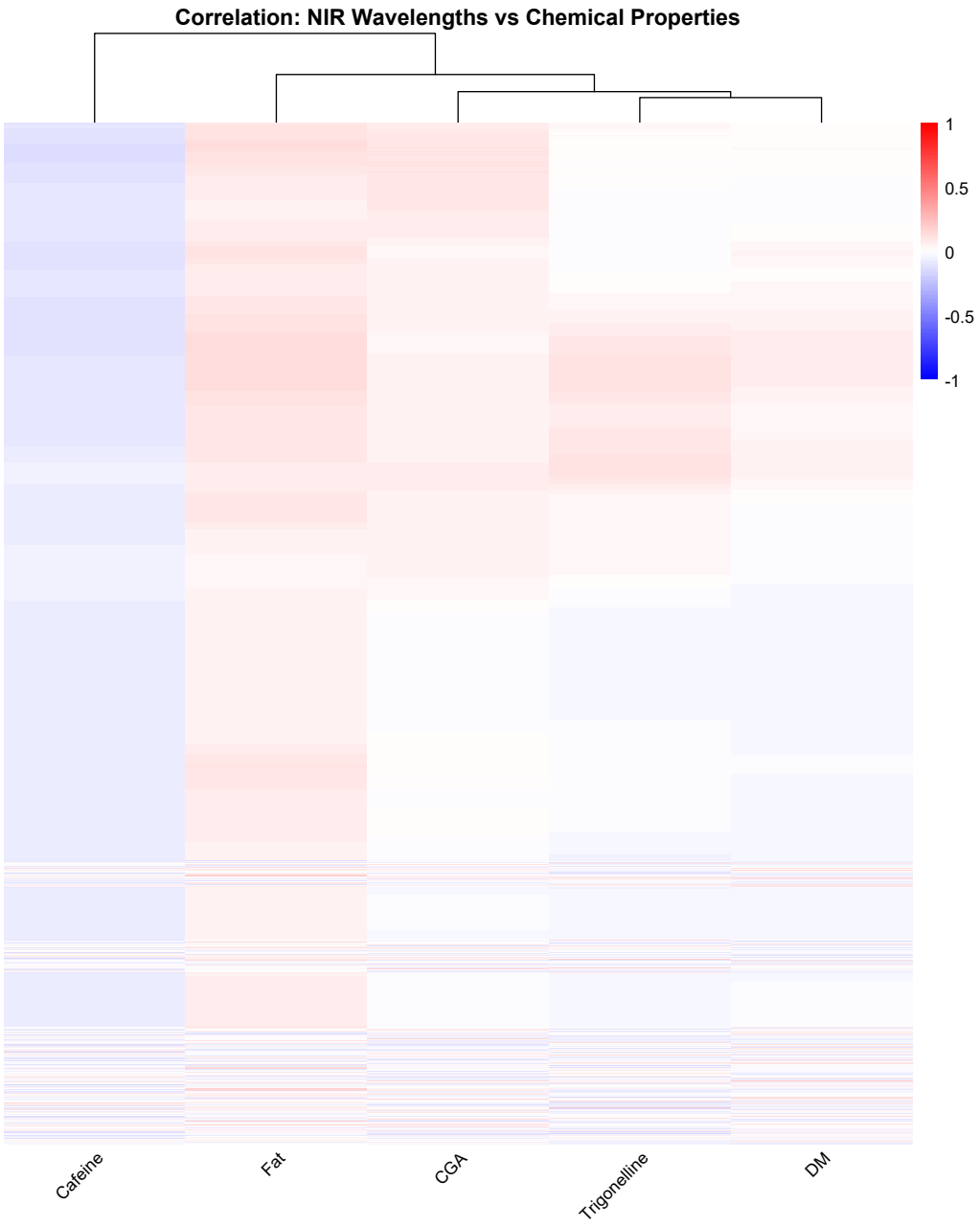
```
cat("Số biến hóa lý:", ncol(cor_nir_chem), "\n")
```

```
## Số biến hóa lý: 5
```

4.3.2 Heatmap tổng quan NIR-Chemical

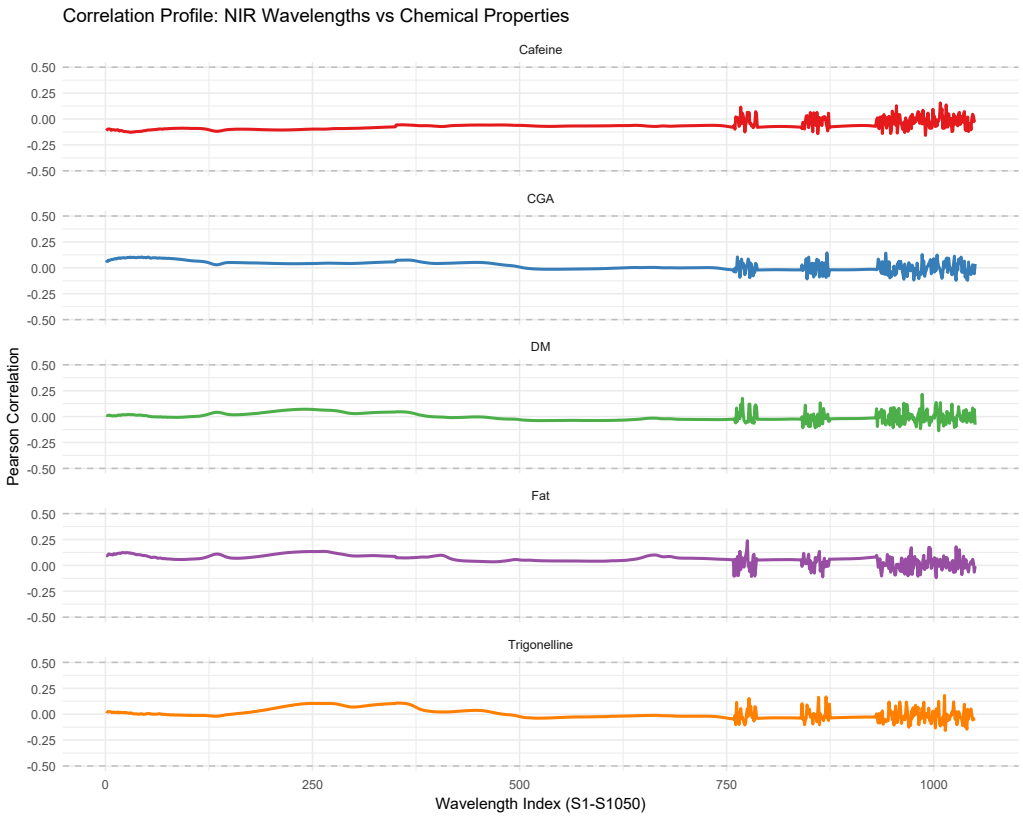
```
# Heatmap cho toàn bộ correlation
pheatmap(cor_nir_chem,
         cluster_rows = FALSE,
         cluster_cols = TRUE,
         show_rownames = FALSE,
         main = "Correlation: NIR Wavelengths vs Chemical Properties",
         color = colorRampPalette(c("blue", "white", "red"))(100),
```

```
        breaks = seq(-1, 1, length.out = 101),
        fontsize = 10,
        angle_col = 45)
```



4.3.3 Biểu đồ Line Plot - Correlation profile

```r
# Chuyển sang long format để vẽ
cor_long <- as.data.frame(cor_nir_chem) %>%
  mutate(Wavelength = 1:nrow(cor_nir_chem)) %>%
  pivot_longer(cols = -Wavelength, names_to = "Chemical", values_to =
   ↪  "Correlation")

# Line plot cho từng biến hóa lý
ggplot(cor_long, aes(x = Wavelength, y = Correlation, color = Chemical))
 ↪  +
  geom_line(linewidth = 1) +
  geom_hline(yintercept = c(-0.5, 0.5), linetype = "dashed", color =
   ↪  "gray") +
  facet_wrap(~Chemical, ncol = 1, scales = "free_y") +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Correlation Profile: NIR Wavelengths vs Chemical
     ↪  Properties",
    x = "Wavelength Index (S1-S1050)",
    y = "Pearson Correlation",
    color = "Chemical Variable"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```



4.3.4 Xác định wavelengths quan trọng

```
# Tìm wavelengths có tương quan cao với mỗi biến hóa lý
important_wavelengths <- list()

for(chem_var in chemical_vars) {
  # Lấy tương quan tuyệt đối
  abs_corr <- abs(cor_nir_chem[, chem_var])

  # Tìm top 10 wavelengths
  top_idx <- order(abs_corr, decreasing = TRUE)[1:10]

  important_wavelengths[[chem_var]] <- data.frame(
    Chemical = chem_var,
    Wavelength = nir_vars[top_idx],
    Wavelength_Index = top_idx,
```

```
    Correlation = cor_nir_chem[top_idx, chem_var],
    Abs_Correlation = abs_corr[top_idx]
  )
}

# Kết hợp tất cả
all_important <- do.call(rbind, important_wavelengths)

cat("Top 10 wavelengths có tương quan mạnh nhất với từng biến hóa
↪  lý:\n\n")
```

```
## Top 10 wavelengths có tương quan mạnh nhất với từng biến hóa lý:
```

```
for(chem_var in chemical_vars) {
  cat("\n**", chem_var, ":**\n")
  all_important %>%
    filter(Chemical == chem_var) %>%
    select(Wavelength, Wavelength_Index, Correlation) %>%
    mutate(Correlation = round(Correlation, 3)) %>%
    head(5) %>%
    kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover")) %>%
    print()
}
```

```
##
## ** CGA :**
## \begin{table}
## \centering
## \begin{tabular}{l|l|r|r}
## \hline
##   & Wavelength & Wavelength\_Index & Correlation\\
```

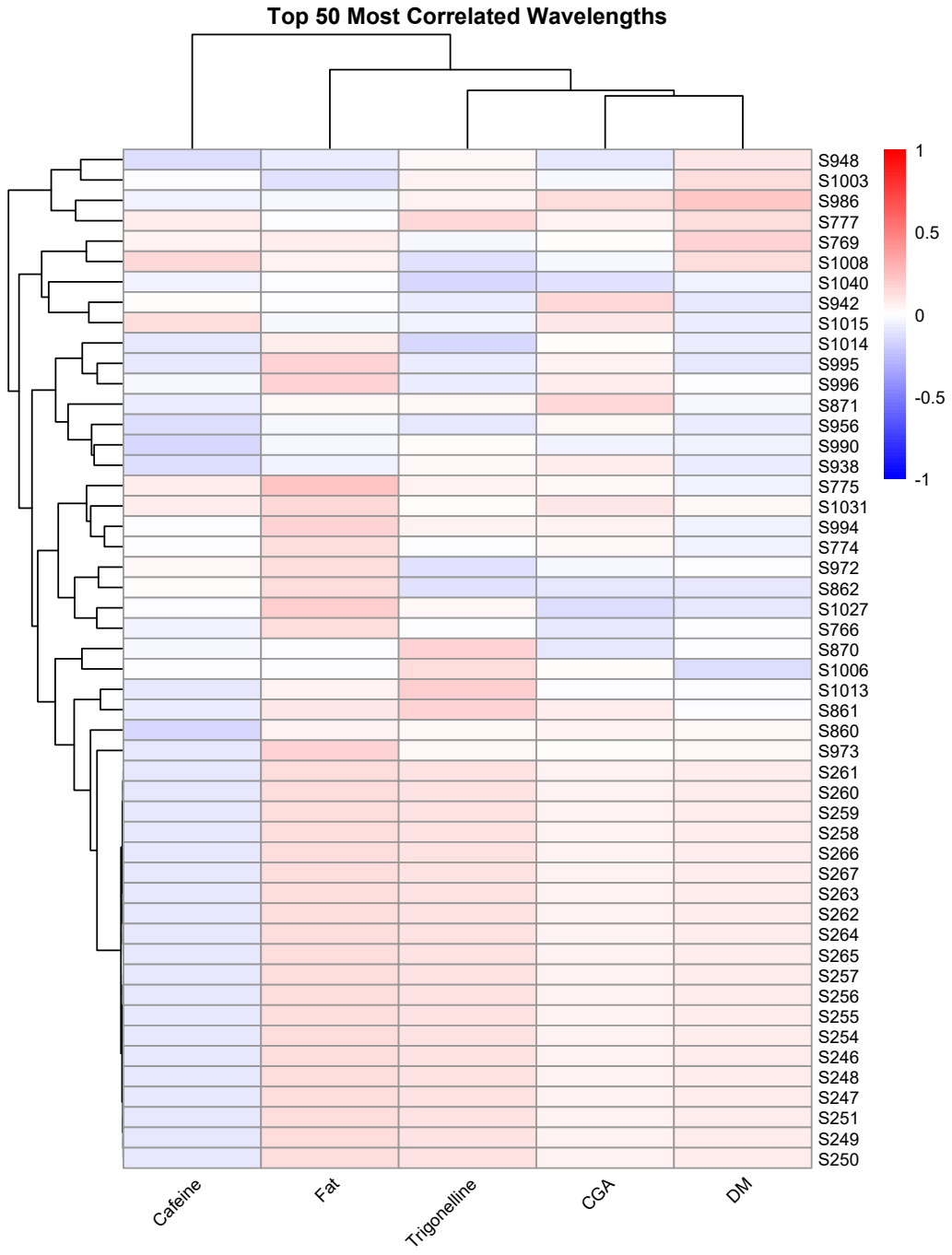
```
## \hline
## CGA.S871 & S871 & 871 & 0.144\\
## \hline
## CGA.S942 & S942 & 942 & 0.142\\
## \hline
## CGA.S986 & S986 & 986 & 0.128\\
## \hline
## CGA.S1004 & S1004 & 1004 & 0.123\\
## \hline
## CGA.S1027 & S1027 & 1027 & -0.121\\
## \hline
## \end{tabular}
## \end{table}
##
## ** Cafeine :**
## \begin{table}
## \centering
## \begin{tabular}{l|l|r|r}
## \hline
##   & Wavelength & Wavelength\_Index & Correlation\\
## \hline
## Cafeine.S990 & S990 & 990 & -0.158\\
## \hline
## Cafeine.S1008 & S1008 & 1008 & 0.156\\
## \hline
## Cafeine.S860 & S860 & 860 & -0.141\\
## \hline
## Cafeine.S948 & S948 & 948 & -0.140\\
## \hline
## Cafeine.S938 & S938 & 938 & -0.140\\
## \hline
## \end{tabular}
```

```
## \end{table}
##
## ** Fat :**
## \begin{table}
## \centering
## \begin{tabular}{l|l|r|r}
## \hline
##   & Wavelength & Wavelength\_Index & Correlation\\
## \hline
## Fat.S775 & S775 & 775 & 0.238\\
## \hline
## Fat.S1027 & S1027 & 1027 & 0.180\\
## \hline
## Fat.S995 & S995 & 995 & 0.174\\
## \hline
## Fat.S994 & S994 & 994 & 0.172\\
## \hline
## Fat.S973 & S973 & 973 & 0.171\\
## \hline
## \end{tabular}
## \end{table}
##
## ** Trigonelline :**
## \begin{table}
## \centering
## \begin{tabular}{l|l|r|r}
## \hline
##   & Wavelength & Wavelength\_Index & Correlation\\
## \hline
## Trigonelline.S1013 & S1013 & 1013 & 0.181\\
## \hline
## Trigonelline.S870 & S870 & 870 & 0.167\\
```

```
## \hline
## Trigonelline.S861 & S861 & 861 & 0.163\\
## \hline
## Trigonelline.S1014 & S1014 & 1014 & -0.157\\
## \hline
## Trigonelline.S777 & S777 & 777 & 0.152\\
## \hline
## \end{tabular}
## \end{table}
##
## ** DM :**
## \begin{table}
## \centering
## \begin{tabular}{l|l|r|r}
## \hline
##   & Wavelength & Wavelength\_Index & Correlation\\
## \hline
## DM.S986 & S986 & 986 & 0.215\\
## \hline
## DM.S769 & S769 & 769 & 0.176\\
## \hline
## DM.S1006 & S1006 & 1006 & -0.137\\
## \hline
## DM.S1003 & S1003 & 1003 & 0.136\\
## \hline
## DM.S1008 & S1008 & 1008 & 0.135\\
## \hline
## \end{tabular}
## \end{table}
```

4.3.5 Heatmap cho wavelengths quan trọng nhất

```r
# Lấy top 50 wavelengths có tương quan cao nhất (cho bất kỳ biến nào)
max_abs_corr <- apply(abs(cor_nir_chem), 1, max)
top_50_idx <- order(max_abs_corr, decreasing = TRUE)[1:50]

# Heatmap cho top wavelengths
pheatmap(cor_nir_chem[top_50_idx, ],
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         show_rownames = TRUE,
         labels_row = paste0("S", top_50_idx),
         main = "Top 50 Most Correlated Wavelengths",
         color = colorRampPalette(c("blue", "white", "red"))(100),
         breaks = seq(-1, 1, length.out = 101),
         fontsize = 8,
         angle_col = 45)
```



4.4 Tương quan trong không gian PCA

```
# Thực hiện PCA trên NIR
library(FactoMineR)
pca_nir <- PCA(data_nir_full, scale.unit = TRUE, graph = FALSE)
```

Table 4.2: *Correlation: PC Scores vs Chemical Properties*

	CGA	Cafeine	Fat	Trigonelline	DM
Dim.1	0.029	-0.087	0.080	0.007	-0.002
Dim.2	0.063	-0.033	0.063	0.139	0.085
Dim.3	-0.092	0.078	-0.059	-0.012	-0.062
Dim.4	-0.207	-0.038	0.057	-0.013	0.033
Dim.5	0.105	-0.077	0.177	0.061	0.034

```
# Lấy PC scores (10 PC đầu tiên)
n_pcs <- min(10, ncol(pca_nir$ind$coord))
pc_scores <- pca_nir$ind$coord[, 1:n_pcs]

# Tính tương quan giữa PC scores và biến hóa lý
cor_pc_chem <- cor(pc_scores, data_chem_full, method = "pearson")

cat("Ma trận tương quan PC-Chemical:\n")
```

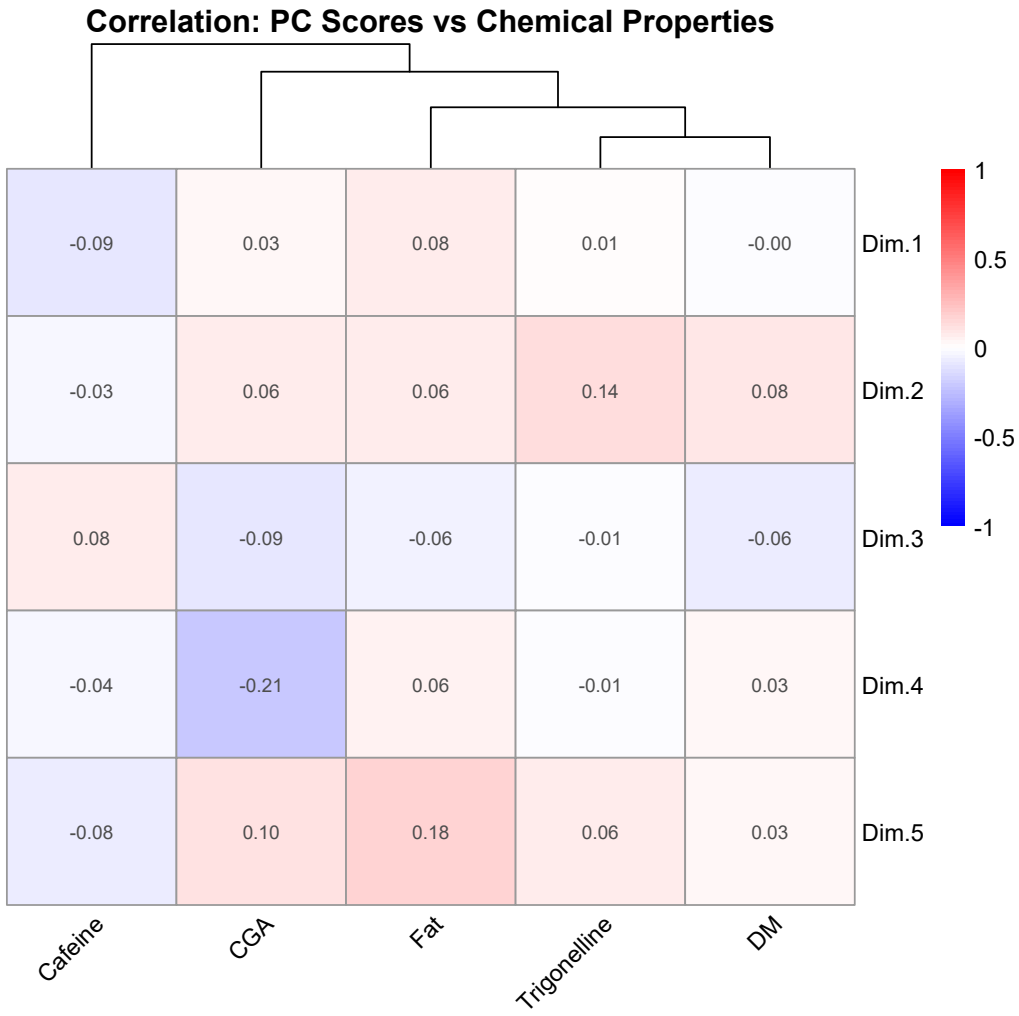
```
## Ma trận tương quan PC-Chemical:
```

```
cor_pc_chem %>%
  round(3) %>%
  kable(caption = "Correlation: PC Scores vs Chemical Properties") %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

4.4.1 Heatmap PC-Chemical

```
# Heatmap
pheatmap(cor_pc_chem,
         cluster_rows = FALSE,
         cluster_cols = TRUE,
         display_numbers = TRUE,
         number_format = "%.2f",
```

```
        main = "Correlation: PC Scores vs Chemical Properties",
        color = colorRampPalette(c("blue", "white", "red"))(100),
        breaks = seq(-1, 1, length.out = 101),
        fontsize = 10,
        angle_col = 45)
```



4.4.2 Phân tích ý nghĩa

```
# Tìm PC có tương quan mạnh nhất với mỗi biến hóa lý
for(chem_var in chemical_vars) {
  pc_corrs <- cor_pc_chem[, chem_var]
  max_pc <- which.max(abs(pc_corrs))
```

```
  max_corr <- pc_corrs[max_pc]

  cat(sprintf("**%s**: Tương quan mạnh nhất với PC%d (r = %.3f)\n",
              chem_var, max_pc, max_corr))
}
```

```
## **CGA**: Tương quan mạnh nhất với PC4 (r = -0.207)
## **Cafeine**: Tương quan mạnh nhất với PC1 (r = -0.087)
## **Fat**: Tương quan mạnh nhất với PC5 (r = 0.177)
## **Trigonelline**: Tương quan mạnh nhất với PC2 (r = 0.139)
## **DM**: Tương quan mạnh nhất với PC2 (r = 0.085)
```

```
cat("\n**Ý nghĩa:**\n")
```

```
##
## **Ý nghĩa:**
```

```
cat("- PC có tương quan cao với biến hóa lý cho thấy thành phần chính
 ↪  đó\n")
```

```
## - PC có tương quan cao với biến hóa lý cho thấy thành phần chính đó
```

```
cat("  chứa thông tin quan trọng để dự đoán biến đó\n")
```

```
##   chứa thông tin quan trọng để dự đoán biến đó
```

```
cat("- Có thể sử dụng PC scores làm biến độc lập trong mô hình hồi
 ↪  quy\n")
```

```
## - Có thể sử dụng PC scores làm biến độc lập trong mô hình hồi quy
```

4.5 Tương quan theo Location

```
# Tính tương quan riêng cho từng location
location_info <- coffee_data$Localisation[complete_rows]
unique_locations <- unique(location_info)

cat("Phân tích tương quan theo từng Location:\n\n")
```

```
## Phân tích tương quan theo từng Location:
```

```
for(loc in unique_locations) {
  loc_idx <- which(location_info == loc)

  if(length(loc_idx) > 5) {  # Chỉ phân tích nếu có đủ mẫu
    data_chem_loc <- data_chem_full[loc_idx, ]
    cor_loc <- cor(data_chem_loc, method = "pearson")

    cat(sprintf("\n### Location %s (n = %d samples):\n", loc,
    ↪  length(loc_idx)))

    # Hiển thị ma trận
    cor_loc %>%
      round(3) %>%
      kable() %>%
      kable_styling(bootstrap_options = c("striped", "hover")) %>%
      print()
  }
}
```

```
##
## ### Location 1 (n = 50 samples):
```

```
## \begin{table}
## \centering
## \begin{tabular}{l|r|r|r|r|r}
## \hline
##   & CGA & Cafeine & Fat & Trigonelline & DM\\
## \hline
## CGA & 1.000 & 0.029 & -0.002 & 0.340 & -0.104\\
## \hline
## Cafeine & 0.029 & 1.000 & 0.189 & -0.033 & 0.424\\
## \hline
## Fat & -0.002 & 0.189 & 1.000 & -0.064 & -0.051\\
## \hline
## Trigonelline & 0.340 & -0.033 & -0.064 & 1.000 & -0.072\\
## \hline
## DM & -0.104 & 0.424 & -0.051 & -0.072 & 1.000\\
## \hline
## \end{tabular}
## \end{table}
##
## ### Location 6 (n = 84 samples):
## \begin{table}
## \centering
## \begin{tabular}{l|r|r|r|r|r}
## \hline
##   & CGA & Cafeine & Fat & Trigonelline & DM\\
## \hline
## CGA & 1.000 & -0.061 & 0.083 & 0.065 & -0.051\\
## \hline
## Cafeine & -0.061 & 1.000 & 0.089 & -0.014 & -0.013\\
## \hline
## Fat & 0.083 & 0.089 & 1.000 & 0.014 & -0.041\\
## \hline
```

```
## Trigonelline & 0.065 & -0.014 & 0.014 & 1.000 & 0.091\\
## \hline
## DM & -0.051 & -0.013 & -0.041 & 0.091 & 1.000\\
## \hline
## \end{tabular}
## \end{table}
##
## ### Location 4 (n = 13 samples):
## \begin{table}
## \centering
## \begin{tabular}{l|r|r|r|r|r}
## \hline
##   & CGA & Cafeine & Fat & Trigonelline & DM\\
## \hline
## CGA & 1.000 & 0.092 & -0.228 & 0.345 & -0.347\\
## \hline
## Cafeine & 0.092 & 1.000 & 0.022 & -0.136 & -0.019\\
## \hline
## Fat & -0.228 & 0.022 & 1.000 & -0.110 & -0.080\\
## \hline
## Trigonelline & 0.345 & -0.136 & -0.110 & 1.000 & -0.186\\
## \hline
## DM & -0.347 & -0.019 & -0.080 & -0.186 & 1.000\\
## \hline
## \end{tabular}
## \end{table}
##
## ### Location 7 (n = 19 samples):
## \begin{table}
## \centering
## \begin{tabular}{l|r|r|r|r|r}
## \hline
```

```
##   & CGA & Cafeine & Fat & Trigonelline & DM\\
## \hline
## CGA & 1.000 & 0.061 & 0.938 & -0.090 & -0.218\\
## \hline
## Cafeine & 0.061 & 1.000 & -0.125 & -0.055 & -0.226\\
## \hline
## Fat & 0.938 & -0.125 & 1.000 & 0.021 & 0.014\\
## \hline
## Trigonelline & -0.090 & -0.055 & 0.021 & 1.000 & 0.162\\
## \hline
## DM & -0.218 & -0.226 & 0.014 & 0.162 & 1.000\\
## \hline
## \end{tabular}
## \end{table}
##
## ### Location 2 (n = 26 samples):
## \begin{table}
## \centering
## \begin{tabular}{l|r|r|r|r|r}
## \hline
##   & CGA & Cafeine & Fat & Trigonelline & DM\\
## \hline
## CGA & 1.000 & -0.157 & -0.112 & -0.072 & -0.051\\
## \hline
## Cafeine & -0.157 & 1.000 & -0.211 & 0.449 & -0.123\\
## \hline
## Fat & -0.112 & -0.211 & 1.000 & 0.218 & -0.168\\
## \hline
## Trigonelline & -0.072 & 0.449 & 0.218 & 1.000 & -0.001\\
## \hline
## DM & -0.051 & -0.123 & -0.168 & -0.001 & 1.000\\
## \hline
```

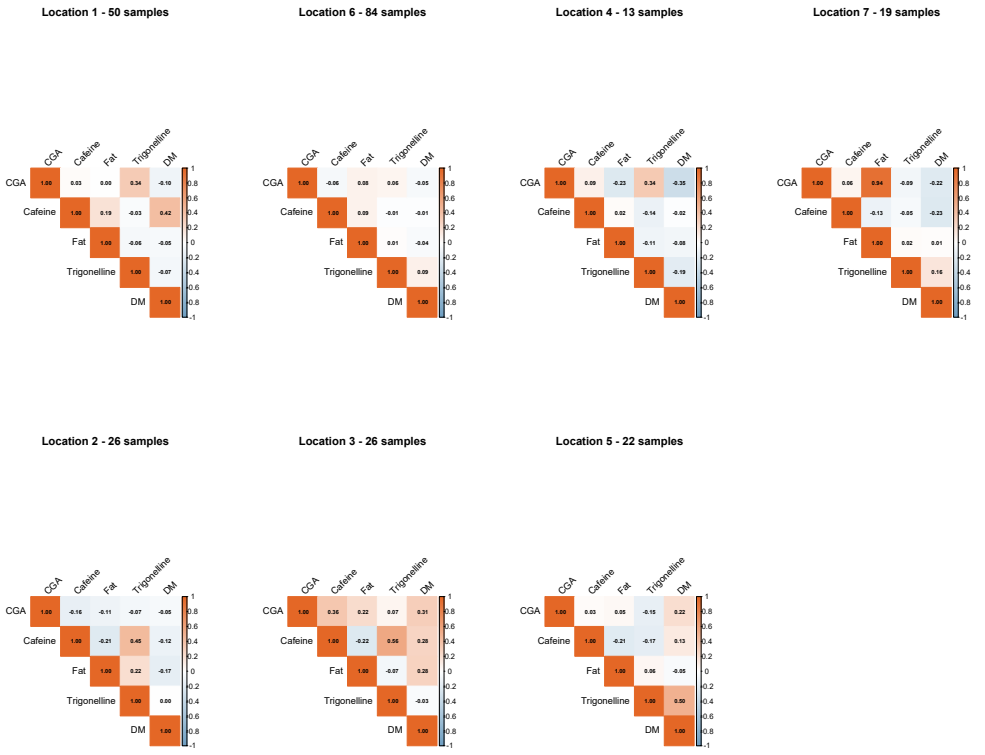
```
## \end{tabular}
## \end{table}
##
## ### Location 3 (n = 26 samples):
## \begin{table}
## \centering
## \begin{tabular}{l|r|r|r|r|r}
## \hline
##   & CGA & Cafeine & Fat & Trigonelline & DM\\
## \hline
## CGA & 1.000 & 0.361 & 0.224 & 0.069 & 0.311\\
## \hline
## Cafeine & 0.361 & 1.000 & -0.223 & 0.564 & 0.277\\
## \hline
## Fat & 0.224 & -0.223 & 1.000 & -0.073 & 0.281\\
## \hline
## Trigonelline & 0.069 & 0.564 & -0.073 & 1.000 & -0.033\\
## \hline
## DM & 0.311 & 0.277 & 0.281 & -0.033 & 1.000\\
## \hline
## \end{tabular}
## \end{table}
##
## ### Location 5 (n = 22 samples):
## \begin{table}
## \centering
## \begin{tabular}{l|r|r|r|r|r}
## \hline
##   & CGA & Cafeine & Fat & Trigonelline & DM\\
## \hline
## CGA & 1.000 & 0.029 & 0.049 & -0.153 & 0.217\\
## \hline
```

```
## Cafeine & 0.029 & 1.000 & -0.206 & -0.171 & 0.133\\
## \hline
## Fat & 0.049 & -0.206 & 1.000 & 0.058 & -0.050\\
## \hline
## Trigonelline & -0.153 & -0.171 & 0.058 & 1.000 & 0.503\\
## \hline
## DM & 0.217 & 0.133 & -0.050 & 0.503 & 1.000\\
## \hline
## \end{tabular}
## \end{table}
```

4.5.1 So sánh pattern tương quan giữa các Location

```
# Tạo heatmap cho mỗi location
par(mfrow = c(2, ceiling(length(unique_locations) / 2)))

for(loc in unique_locations) {
  loc_idx <- which(location_info == loc)

  if(length(loc_idx) > 5) {
    data_chem_loc <- data_chem_full[loc_idx, ]
    cor_loc <- cor(data_chem_loc, method = "pearson")

    corrplot::corrplot(cor_loc,
             method = "color",
             type = "upper",
             addCoef.col = "black",
             tl.col = "black",
             tl.srt = 45,
             number.cex = 0.6,
             col = colorRampPalette(c("#6D9EC1", "white",
              ↪  "#E46726"))(200),
```

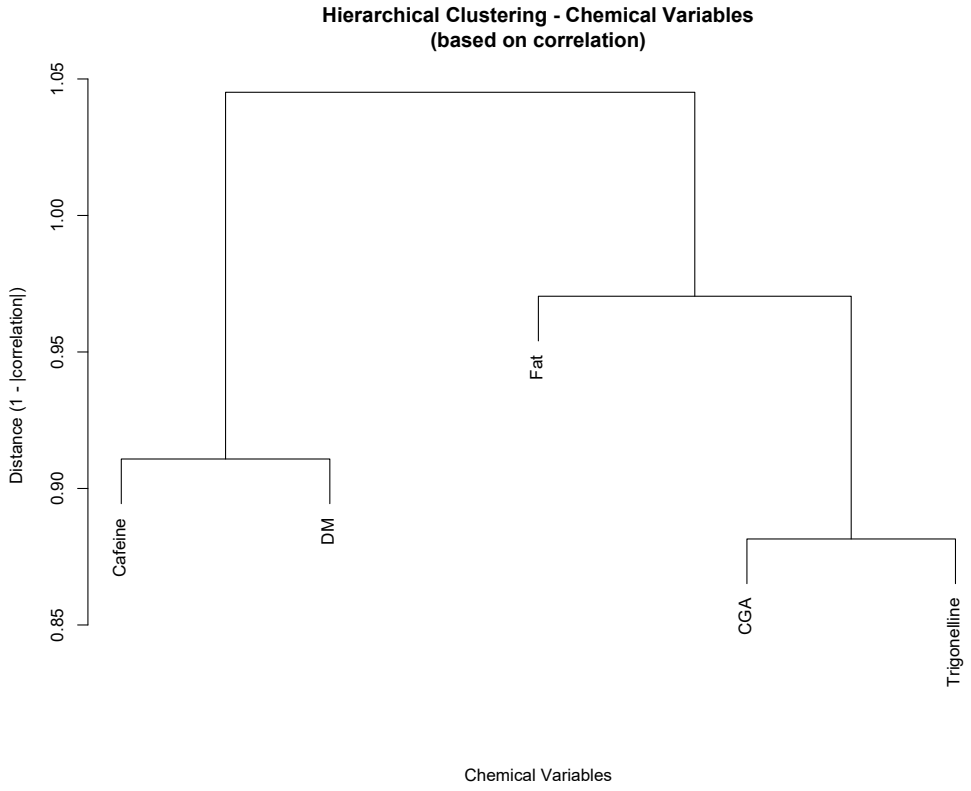
```
            title = paste("Location", loc, "-", length(loc_idx),
            ↪  "samples"),
            mar = c(0, 0, 2, 0))
  }
}

par(mfrow = c(1, 1))
```



4.6 Hierarchical Clustering dựa trên Correlation

```
# Clustering các biến hóa lý dựa trên correlation
dist_chem <- as.dist(1 - abs(cor_matrix))
hc_chem <- hclust(dist_chem, method = "ward.D2")
```

```
# Dendrogram
plot(hc_chem,
     main = "Hierarchical Clustering - Chemical Variables\n(based on
      ↪  correlation)",
     xlab = "Chemical Variables",
     ylab = "Distance (1 - |correlation|)",
     sub = "")
```



4.7 Tổng hợp và Kết luận

```
cat("### Tóm tắt phân tích tương quan:\n\n")
```

```
## ### Tóm tắt phân tích tương quan:
```

```r
cat("**1. Tương quan giữa biến hóa lý:**\n")
```

```
## **1. Tương quan giữa biến hóa lý:**
```

```r
# Tìm tương quan cao nhất
max_corr_idx <- which(abs(cor_matrix) ==
 ↪  max(abs(cor_matrix[upper.tri(cor_matrix)])), arr.ind = TRUE)[1,]
cat(sprintf("- Tương quan mạnh nhất: %s vs %s (r = %.3f)\n",
            rownames(cor_matrix)[max_corr_idx[1]],
            colnames(cor_matrix)[max_corr_idx[2]],
            cor_matrix[max_corr_idx[1], max_corr_idx[2]]))
```

```
## - Tương quan mạnh nhất: Trigonelline vs CGA (r = 0.118)
```

```r
cat("\n**2. NIR wavelengths quan trọng:**\n")
```

```
##
## **2. NIR wavelengths quan trọng:**
```

```r
for(chem_var in chemical_vars) {
  top_wave <- all_important %>%
    filter(Chemical == chem_var) %>%
    slice_max(Abs_Correlation, n = 1)

  cat(sprintf("- %s: Wavelength %s (r = %.3f)\n",
              chem_var, top_wave$Wavelength, top_wave$Correlation))
}
```

```
## - CGA: Wavelength S871 (r = 0.144)
## - Cafeine: Wavelength S990 (r = -0.158)
## - Fat: Wavelength S775 (r = 0.238)
```

```
## - Trigonelline: Wavelength S1013 (r = 0.181)
## - DM: Wavelength S986 (r = 0.215)
```

```
cat("\n**3. PC scores hiệu quả:**\n")
```

```
##
## **3. PC scores hiệu quả:**
```

```
for(chem_var in chemical_vars) {
  pc_corrs <- cor_pc_chem[, chem_var]
  max_pc <- which.max(abs(pc_corrs))
  cat(sprintf("- %s: PC%d (r = %.3f)\n",
              chem_var, max_pc, pc_corrs[max_pc]))
}
```

```
## - CGA: PC4 (r = -0.207)
## - Cafeine: PC1 (r = -0.087)
## - Fat: PC5 (r = 0.177)
## - Trigonelline: PC2 (r = 0.139)
## - DM: PC2 (r = 0.085)
```

```
cat("\n**4. Khuyến nghị:**\n")
```

```
##
## **4. Khuyến nghị:**
```

```
cat("- Sử dụng wavelengths có tương quan cao để xây dựng mô hình đơn
 ↪  giản\n")
```

```
## - Sử dụng wavelengths có tương quan cao để xây dựng mô hình đơn giản
```

```
cat("- Cân nhắc sử dụng PC scores thay vì toàn bộ NIR để giảm chiều dữ
 ↪  liệu\n")
```

```
## - Cân nhắc sử dụng PC scores thay vì toàn bộ NIR để giảm chiều dữ liệu
```

```
cat("- Lưu ý sự khác biệt tương quan giữa các Location khi xây dựng mô
 ↪  hình\n")
```

```
## - Lưu ý sự khác biệt tương quan giữa các Location khi xây dựng mô hình
```

```
cat("- Các biến hóa lý có tương quan cao với nhau có thể gây
 ↪  multicollinearity\n")
```

```
## - Các biến hóa lý có tương quan cao với nhau có thể gây multicollinearity
```

Chapter 5

Mô Hình Dự Đoán Chỉ Tiêu Hóa Lý

Trong phần này, chúng ta sẽ xây dựng các mô hình dự đoán để ước lượng các chỉ tiêu hóa lý từ dữ liệu phổ NIR. Hai phương pháp chính được sử dụng là:

- **PLS (Partial Least Squares Regression)**: Phương pháp tối ưu cho dữ liệu có nhiều biến tương quan cao
- **PCR (Principal Component Regression)**: Sử dụng các thành phần chính từ PCA

5.1 Chuẩn Bị Dữ Liệu

```
# Data and variable groups already loaded in index.Rmd global-setup
# Verify data is available
if(!exists("coffee_data")) {
  stop("Data not loaded. Please render from index.Rmd")
}

# Tách dữ liệu thành matrices
X <- as.matrix(coffee_data[, nir_vars])
Y <- as.matrix(coffee_data[, chemical_vars])
```

```r
cat("Kích thước dữ liệu NIR:", dim(X), "\n")
```

```
## Kích thước dữ liệu NIR: 240 1050
```

```r
cat("Kích thước dữ liệu hóa lý:", dim(Y), "\n")
```

```
## Kích thước dữ liệu hóa lý: 240 5
```

```r
cat("Số lượng mẫu:", nrow(X), "\n")
```

```
## Số lượng mẫu: 240
```

5.2 Mô Hình PLS Regression

5.2.1 Xác Định Số Thành Phần Tối Ưu

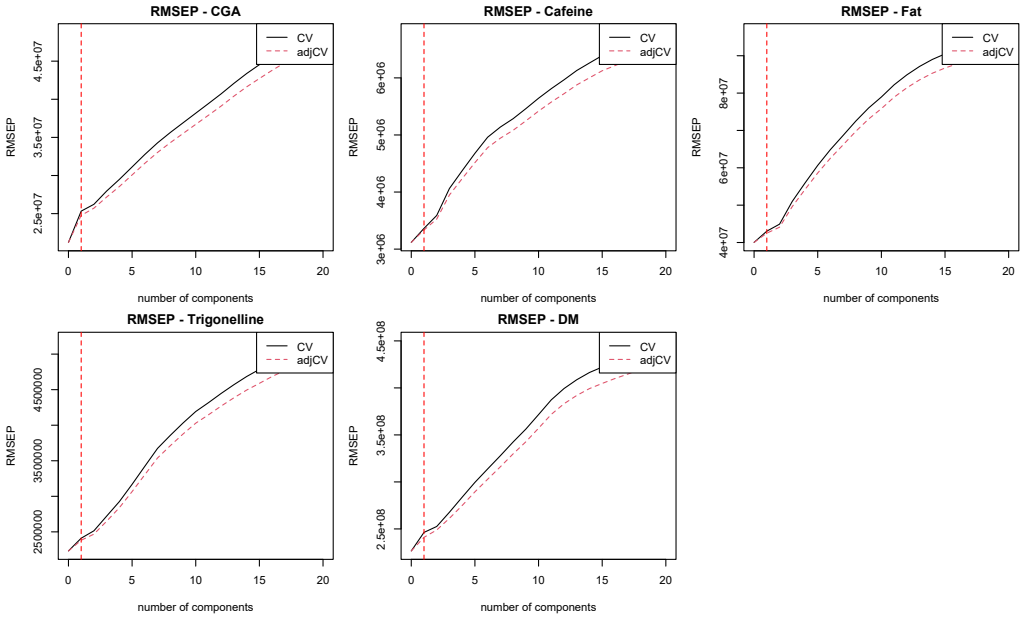
```r
# Xây dựng mô hình PLS cho từng biến hóa lý
set.seed(123)
pls_models <- list()
optimal_ncomp <- c()
pls_train_data <- list()  # Store training data for each model
pls_complete_idx <- list()  # Store complete case indices for each model

for(var in chemical_vars) {
  # Tạo data frame cho pls
  pls_data <- data.frame(Y = coffee_data[[var]], X)

  # Find complete cases and store the indices
  complete_idx <- complete.cases(pls_data)
  pls_complete_idx[[var]] <- complete_idx
```

```
  # Remove NA rows
  pls_data_clean <- pls_data[complete_idx, ]
  pls_train_data[[var]] <- pls_data_clean

  # Cross-validation - train on CLEANED data
  pls_cv <- plsr(Y ~ ., data = pls_data_clean, validation = "CV",
                 segments = 10, ncomp = 20)

  pls_models[[var]] <- pls_cv

  # Tìm số thành phần tối ưu (RMSEP thấp nhất)
  rmsep_vals <- RMSEP(pls_cv, estimate = "CV")$val[1,,]
  optimal_ncomp[var] <- which.min(rmsep_vals[-1]) # Bỏ intercept
}

# Vẽ biểu đồ RMSEP
par(mfrow = c(2, 3), mar = c(4, 4, 2, 1))
for(var in chemical_vars) {
  validationplot(pls_models[[var]], val.type = "RMSEP",
                 main = paste("RMSEP -", var),
                 legendpos = "topright")
  abline(v = optimal_ncomp[var], col = "red", lty = 2)
}

cat("\nSố thành phần tối ưu cho mỗi biến:\n")
```

```
##
## Số thành phần tối ưu cho mỗi biến:
```

```
print(optimal_ncomp)
```

```
##          CGA      Cafeine          Fat Trigonelline           DM
##            1            1            1            1            1
```



5.2.2 Hiệu Suất Mô Hình PLS

```
# Tính toán các chỉ số đánh giá - REWRITTEN with safer approach
pls_performance <- data.frame(
  Variable = chemical_vars,
  N_Components = integer(length(chemical_vars)),
  RMSECV = numeric(length(chemical_vars)),
  R2_CV = numeric(length(chemical_vars)),
  RMSEP = numeric(length(chemical_vars)),
  R2 = numeric(length(chemical_vars))
)

for(i in 1:length(chemical_vars)) {
  var <- chemical_vars[i]
```

```
  model <- pls_models[[var]]

  # Get optimal ncomp as plain integer
  ncomp <- unname(optimal_ncomp[var])
  pls_performance$N_Components[i] <- ncomp

  # Get training data
  train_data <- pls_train_data[[var]]
  Y_train <- train_data$Y

  # RMSECV from cross-validation
  rmsep_cv <- RMSEP(model, estimate = "CV")
  pls_performance$RMSECV[i] <- rmsep_cv$val[1, 1, ncomp + 1]

  # R² from cross-validation - Calculate manually from RMSECV
  # R²_CV = 1 - (RMSECV² / Var(Y))
  rmsecv_value <- rmsep_cv$val[1, 1, ncomp + 1]
  var_y <- var(Y_train)
  pls_performance$R2_CV[i] <- 1 - (rmsecv_value^2 / var_y)

  # RMSEP from training
  rmsep_train <- RMSEP(model, estimate = "train")
  pls_performance$RMSEP[i] <- rmsep_train$val[1, 1, ncomp + 1]

  # R² from training - Use fitted values
  Y_fitted <- fitted(model)[, , ncomp]

  # Calculate R² manually
  ss_res <- sum((Y_train - Y_fitted)^2)
  ss_tot <- sum((Y_train - mean(Y_train))^2)
  pls_performance$R2[i] <- 1 - (ss_res / ss_tot)
}
```

Table 5.1: *Hiệu suất mô hình PLS cho các chỉ tiêu hóa lý*

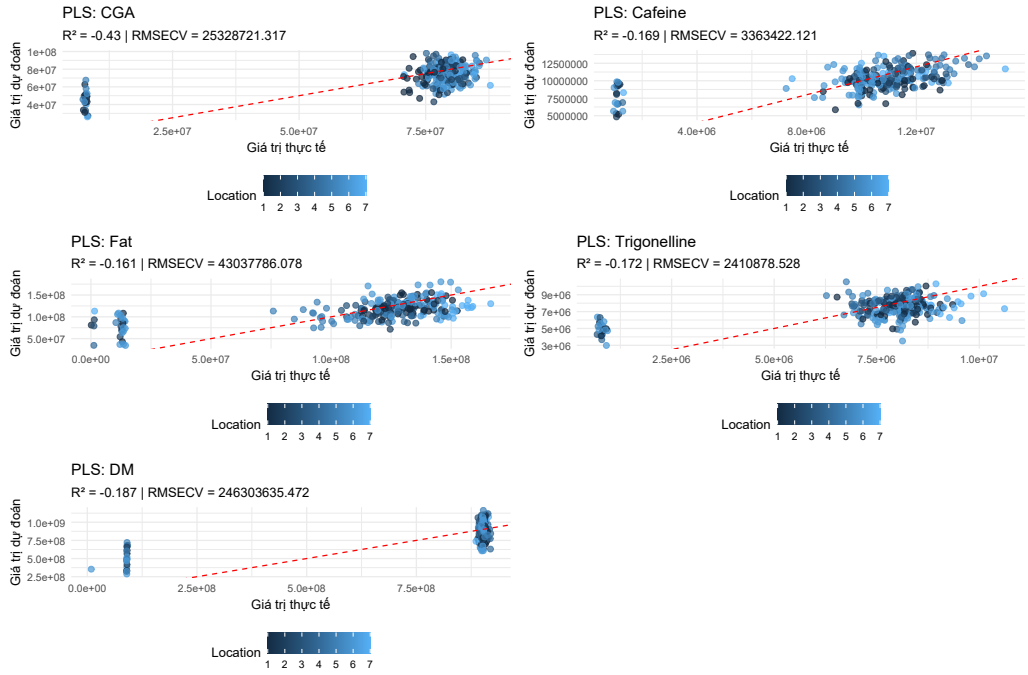
Variable	N_Components	RMSECV	R2_CV	RMSEP	R2
CGA	1	25328721	-0.4298	16171421	0.4147
Cafeine	1	3363422	-0.1687	2591722	0.3032
Fat	1	43037786	-0.1608	31401446	0.3795
Trigonelline	1	2410879	-0.1720	1784600	0.3551
DM	1	246303635	-0.1867	166340025	0.4565

```
knitr::kable(pls_performance,
             caption = "Hiệu suất mô hình PLS cho các chỉ tiêu hóa lý",
             digits = 4)
```

5.2.3 Biểu Đồ Predicted vs Actual

```
plot_list <- list()

for(i in 1:length(chemical_vars)) {
  var <- chemical_vars[i]
  model <- pls_models[[var]]

  # Get optimal ncomp as plain integer
  ncomp <- unname(optimal_ncomp[var])

  # Get complete case indices for this variable
  complete_idx <- pls_complete_idx[[var]]

  # Use fitted values from the model (already on training data)
  # This is safer than predict()
  fitted_vals <- fitted(model)[, , ncomp]

  actual <- coffee_data[[var]][complete_idx]
```

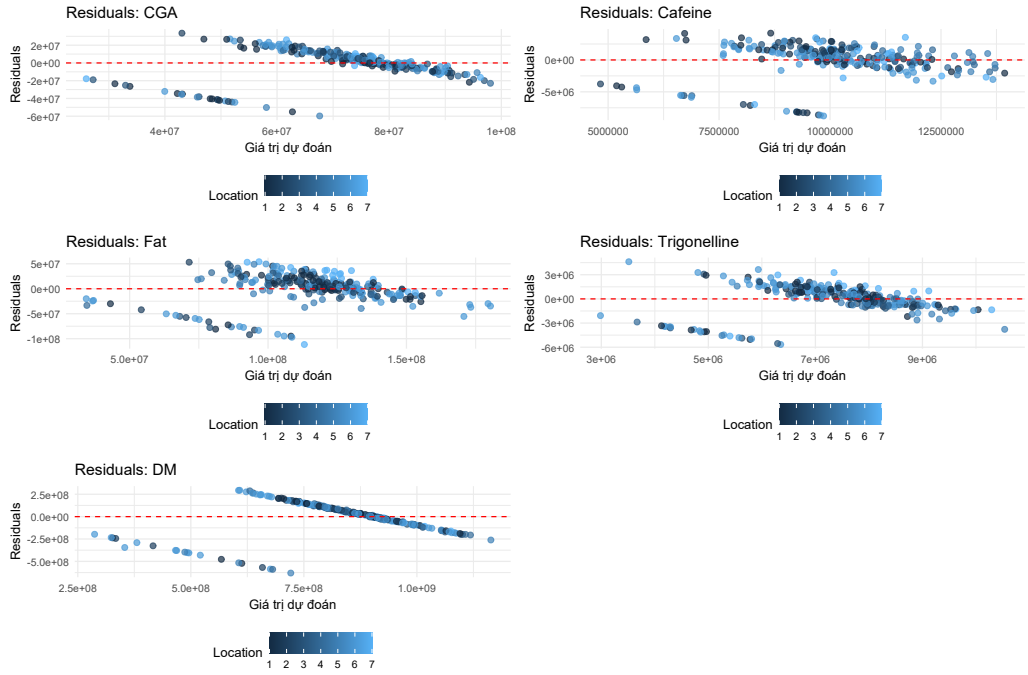
```
  location <- coffee_data$Localisation[complete_idx]

  # Tạo data frame
  pred_df <- data.frame(
    Actual = actual,
    Predicted = fitted_vals,
    Location = location
  )

  # Vẽ biểu đồ
  p <- ggplot(pred_df, aes(x = Actual, y = Predicted, color = Location))
  +
    geom_point(size = 2, alpha = 0.7) +
    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color =
     "red") +
    labs(title = paste("PLS:", var),
         subtitle = paste("R² =", round(pls_performance$R2_CV[i], 3),
                          "| RMSECV =", round(pls_performance$RMSECV[i],
                           3)),
         x = "Giá trị thực tế",
         y = "Giá trị dự đoán") +
    theme_minimal() +
    theme(legend.position = "bottom")

  plot_list[[i]] <- p
}

do.call(grid.arrange, c(plot_list, ncol = 2))
```



5.2.4 Biểu Đồ Residuals

```
plot_list_res <- list()

for(i in 1:length(chemical_vars)) {
  var <- chemical_vars[i]
  model <- pls_models[[var]]

  # Get optimal ncomp as plain integer
  ncomp <- unname(optimal_ncomp[var])

  # Get complete case indices for this variable
  complete_idx <- pls_complete_idx[[var]]

  # Use fitted values from the model
  fitted_vals <- fitted(model)[, , ncomp]

  actual <- coffee_data[[var]][complete_idx]
```

```
  residuals <- actual - fitted_vals
  location <- coffee_data$Localisation[complete_idx]

  # Tạo data frame
  res_df <- data.frame(
    Predicted = fitted_vals,
    Residuals = residuals,
    Location = location
  )

  # Vẽ biểu đồ
  p <- ggplot(res_df, aes(x = Predicted, y = Residuals, color =
↪  Location)) +
    geom_point(size = 2, alpha = 0.7) +
    geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
    labs(title = paste("Residuals:", var),
         x = "Giá trị dự đoán",
         y = "Residuals") +
    theme_minimal() +
    theme(legend.position = "bottom")

  plot_list_res[[i]] <- p
}

do.call(grid.arrange, c(plot_list_res, ncol = 2))
```



5.3 Mô Hình PCR (Principal Component Regression)

```
# Xây dựng mô hình PCR
pcr_models <- list()
pcr_optimal_ncomp <- c()
pcr_train_data <- list()  # Store training data for each model
pcr_complete_idx <- list()  # Store complete case indices for each model

for(var in chemical_vars) {
  # Tạo data frame cho pcr
  pcr_data <- data.frame(Y = coffee_data[[var]], X)

  # Find complete cases and store the indices
  complete_idx <- complete.cases(pcr_data)
  pcr_complete_idx[[var]] <- complete_idx
```

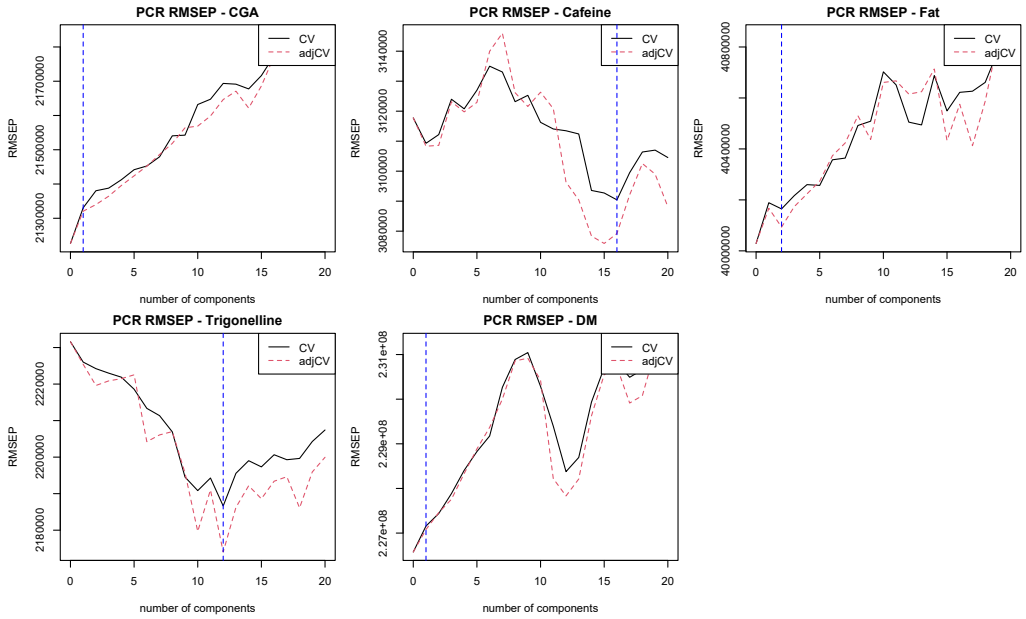
```
  # Remove NA rows - FIX: Added na.omit like PLS
  pcr_data_clean <- pcr_data[complete_idx, ]
  pcr_train_data[[var]] <- pcr_data_clean

  # Cross-validation on CLEANED data
  pcr_cv <- pcr(Y ~ ., data = pcr_data_clean, validation = "CV",
                segments = 10, ncomp = 20)

  pcr_models[[var]] <- pcr_cv

  # Tìm số thành phần tối ưu
  rmsep_vals <- RMSEP(pcr_cv, estimate = "CV")$val[1,,]
  pcr_optimal_ncomp[var] <- which.min(rmsep_vals[-1])
}

# Vẽ biểu đồ RMSEP cho PCR
par(mfrow = c(2, 3), mar = c(4, 4, 2, 1))
for(var in chemical_vars) {
  validationplot(pcr_models[[var]], val.type = "RMSEP",
                 main = paste("PCR RMSEP -", var),
                 legendpos = "topright")
  abline(v = pcr_optimal_ncomp[var], col = "blue", lty = 2)
}

cat("\nSố thành phần tối ưu cho PCR:\n")
```

```
##
## Số thành phần tối ưu cho PCR:
```

```
print(pcr_optimal_ncomp)
```

```
##          CGA      Cafeine          Fat Trigonelline           DM
```

```
##               1              16               2              12               1
```



5.3.1 So Sánh PLS vs PCR

```r
# Tính hiệu suất PCR
pcr_performance <- data.frame(
  Variable = chemical_vars,
  N_Components = integer(length(chemical_vars)),
  RMSECV = numeric(length(chemical_vars)),
  R2_CV = numeric(length(chemical_vars))
)

for(i in 1:length(chemical_vars)) {
  var <- chemical_vars[i]
  model <- pcr_models[[var]]

  # Get optimal ncomp as plain integer
  ncomp <- unname(pcr_optimal_ncomp[var])
  pcr_performance$N_Components[i] <- ncomp
```

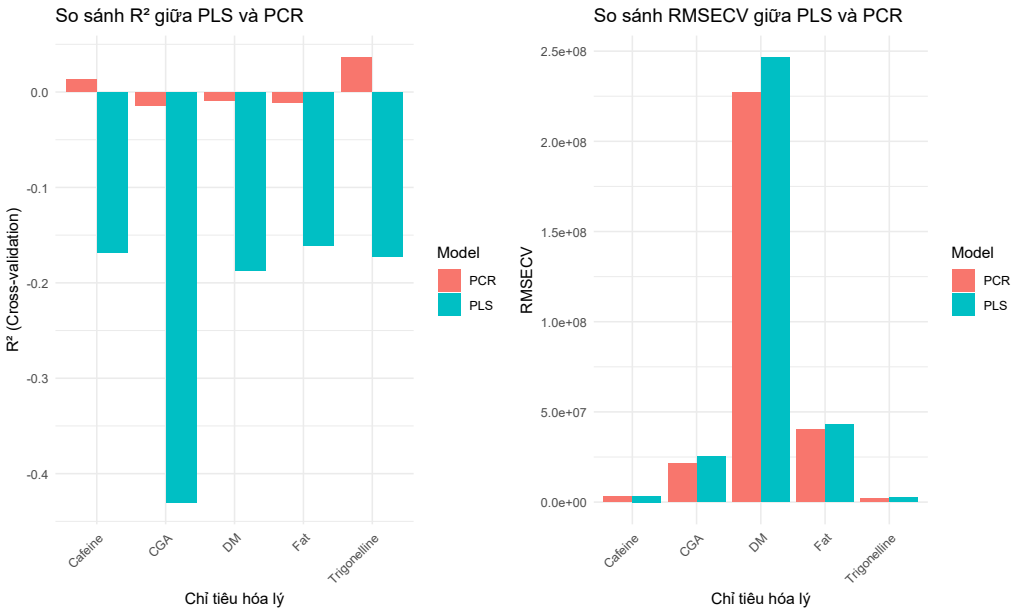
```r
  # Get training data
  train_data <- pcr_train_data[[var]]
  Y_train <- train_data$Y

  # RMSECV from cross-validation
  rmsep_obj <- RMSEP(model, estimate = "CV")
  pcr_performance$RMSECV[i] <- rmsep_obj$val[1, 1, ncomp + 1]

  # R²_CV - Calculate manually from RMSECV (same approach as PLS)
  rmsecv_value <- rmsep_obj$val[1, 1, ncomp + 1]
  var_y <- var(Y_train)
  pcr_performance$R2_CV[i] <- 1 - (rmsecv_value^2 / var_y)
}

# So sánh
comparison <- data.frame(
  Variable = chemical_vars,
  PLS_RMSECV = pls_performance$RMSECV,
  PCR_RMSECV = pcr_performance$RMSECV,
  PLS_R2 = pls_performance$R2_CV,
  PCR_R2 = pcr_performance$R2_CV,
  PLS_ncomp = pls_performance$N_Components,
  PCR_ncomp = pcr_performance$N_Components
)

comparison$Better_Model <- ifelse(comparison$PLS_RMSECV <
 ↪  comparison$PCR_RMSECV,
                                   "PLS", "PCR")

knitr::kable(comparison,
             caption = "So sánh hiệu suất PLS vs PCR",
             digits = 4)
```

Table 5.2: *So sánh hiệu suất PLS vs PCR*

Variable	PLS_RMSECV	PCR_RMSECV	PLS_R2	PCR_R2	PLS_ncomp	PCR_ncomp
CGA	25328721	21330617	-0.4298	-0.0141	1	1
Cafeine	3363422	3090437	-0.1687	0.0133	1	16
Fat	43037786	40164632	-0.1608	-0.0110	1	2
Trigonelline	2410879	2186632	-0.1720	0.0359	1	12
DM	246303635	227158041	-0.1867	-0.0094	1	1

```
# Vẽ biểu đồ so sánh
comp_long <- comparison %>%
  select(Variable, PLS_R2, PCR_R2) %>%
  pivot_longer(cols = c(PLS_R2, PCR_R2),
               names_to = "Model",
               values_to = "R2") %>%
  mutate(Model = gsub("_R2", "", Model))

p1 <- ggplot(comp_long, aes(x = Variable, y = R2, fill = Model)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "So sánh R² giữa PLS và PCR",
       x = "Chỉ tiêu hóa lý",
       y = "R² (Cross-validation)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

comp_long2 <- comparison %>%
  select(Variable, PLS_RMSECV, PCR_RMSECV) %>%
  pivot_longer(cols = c(PLS_RMSECV, PCR_RMSECV),
               names_to = "Model",
               values_to = "RMSECV") %>%
  mutate(Model = gsub("_RMSECV", "", Model))

p2 <- ggplot(comp_long2, aes(x = Variable, y = RMSECV, fill = Model)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "So sánh RMSECV giữa PLS và PCR",
```

```
       x = "Chỉ tiêu hóa lý",
       y = "RMSECV") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(p1, p2, ncol = 2)
```



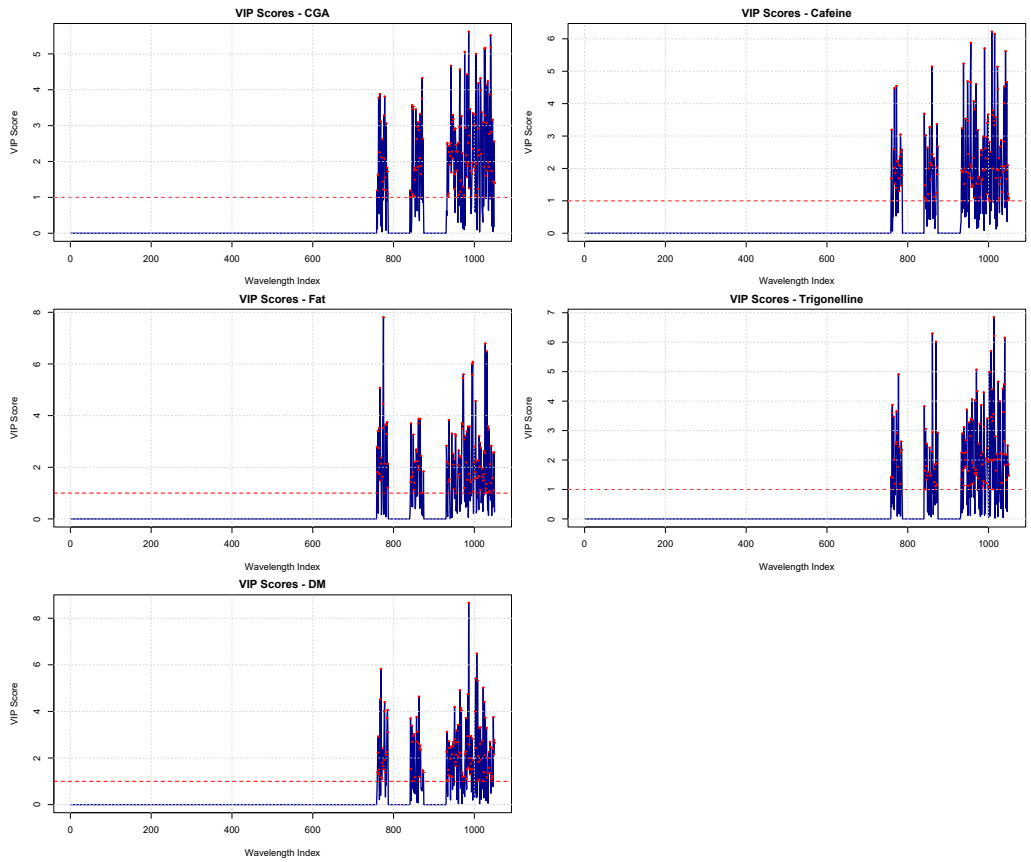
5.4 Variable Importance (VIP)

Variable Importance in Projection (VIP) cho biết mức độ quan trọng của từng bước sóng trong việc dự đoán các chỉ tiêu hóa lý.

```
# Hàm tính VIP
calculate_vip <- function(pls_model, ncomp) {
  # Lấy loading weights
  W <- pls_model$loading.weights[, 1:ncomp, drop = FALSE]

  # Lấy tỷ lệ phương sai giải thích
  SS <- colSums(pls_model$scores[, 1:ncomp, drop = FALSE]^2) *
        colSums(pls_model$Yloadings[, 1:ncomp, drop = FALSE]^2)
```

```r
  # Tính VIP
  p <- nrow(W)
  vip_scores <- sqrt(p * rowSums((W^2) %*% diag(SS, nrow = ncomp)) /
  ↪  sum(SS))

  return(vip_scores)
}

# Tính VIP cho mỗi biến
vip_results <- list()

for(var in chemical_vars) {
  model <- pls_models[[var]]
  ncomp <- optimal_ncomp[var]

  vip_scores <- calculate_vip(model, ncomp)

  vip_df <- data.frame(
    Wavelength = nir_vars,
    VIP = vip_scores
  ) %>%
    mutate(Wavelength_num = as.numeric(gsub("S", "", Wavelength)))

  vip_results[[var]] <- vip_df
}

# Vẽ VIP scores
par(mfrow = c(3, 2), mar = c(4, 4, 2, 1))
for(var in chemical_vars) {
  vip_df <- vip_results[[var]]
```

```
  plot(vip_df$Wavelength_num, vip_df$VIP, type = "l",
       main = paste("VIP Scores -", var),
       xlab = "Wavelength Index",
       ylab = "VIP Score",
       col = "darkblue", lwd = 1.5)
  abline(h = 1, col = "red", lty = 2)  # VIP > 1 considered important
  grid()

  # Highlight important wavelengths
  important <- vip_df$VIP > 1
  points(vip_df$Wavelength_num[important], vip_df$VIP[important],
         col = "red", pch = 20, cex = 0.5)
}
```



5.4.1 Top Wavelengths Quan Trọng

```
for(var in chemical_vars) {
  vip_df <- vip_results[[var]]

  top_wavelengths <- vip_df %>%
    arrange(desc(VIP)) %>%
    head(10)

  cat("\n", var, "- Top 10 bước sóng quan trọng nhất:\n")
  print(top_wavelengths %>% select(Wavelength, VIP))
}
```

```
##
##  CGA - Top 10 bước sóng quan trọng nhất:
##       Wavelength      VIP
## S986        S986 5.623395
## S1040      S1040 5.518892
## S1041      S1041 5.176869
## S1027      S1027 5.165782
## S1025      S1025 5.151859
## S976        S976 5.058635
## S1004      S1004 5.004082
## S942        S942 4.670445
## S964        S964 4.568249
## S982        S982 4.429453
##
##  Cafeine - Top 10 bước sóng quan trọng nhất:
##       Wavelength      VIP
## S1008      S1008 6.224522
## S1015      S1015 6.153865
## S956        S956 5.877781
```

```
## S990        S990 5.707777
## S1042      S1042 5.620512
## S938        S938 5.234130
## S860        S860 5.144120
## S1022      S1022 5.140337
## S948        S948 4.690196
## S1045      S1045 4.664020
##
##  Fat - Top 10 bước sóng quan trọng nhất:
##       Wavelength      VIP
## S775        S775 7.808377
## S1027      S1027 6.794570
## S1031      S1031 6.487211
## S996        S996 6.083319
## S994        S994 6.000223
## S973        S973 5.597506
## S995        S995 5.587739
## S972        S972 5.454037
## S766        S766 5.075426
## S1003      S1003 4.564671
##
##  Trigonelline - Top 10 bước sóng quan trọng nhất:
##       Wavelength      VIP
## S1013      S1013 6.847016
## S861        S861 6.296292
## S1014      S1014 6.220649
## S1040      S1040 6.153409
## S870        S870 6.015696
## S1006      S1006 5.699269
## S970        S970 5.070621
## S1002      S1002 4.987766
## S777        S777 4.908624
```

```
## S1024      S1024 4.661747
##
##  DM - Top 10 bước sóng quan trọng nhất:
##       Wavelength      VIP
## S986        S986 8.656486
## S1006      S1006 6.488407
## S769        S769 5.827370
## S1003      S1003 5.414336
## S1008      S1008 5.318299
## S1022      S1022 5.021698
## S964        S964 4.912229
## S984        S984 4.741825
## S863        S863 4.635444
## S767        S767 4.524381
```

5.5 Kết Luận

5.5.1 Mô hình tốt nhất cho từng chỉ tiêu:

```
for(i in 1:nrow(comparison)) {
  cat(sprintf("   - %s: %s (R² = %.3f, RMSECV = %.3f)\n",
              comparison$Variable[i],
              comparison$Better_Model[i],
              ifelse(comparison$Better_Model[i] == "PLS",
                     comparison$PLS_R2[i], comparison$PCR_R2[i]),
              ifelse(comparison$Better_Model[i] == "PLS",
                     comparison$PLS_RMSECV[i],
 ↪  comparison$PCR_RMSECV[i])))
}
```

```
##    - CGA: PCR (R² = -0.014, RMSECV = 21330616.686)
```

```
##    - Cafeine: PCR (R² = 0.013, RMSECV = 3090437.377)
##    - Fat: PCR (R² = -0.011, RMSECV = 40164631.567)
##    - Trigonelline: PCR (R² = 0.036, RMSECV = 2186632.455)
##    - DM: PCR (R² = -0.009, RMSECV = 227158041.138)
```

5.5.2 Nhận xét

- PLS thường cho kết quả tốt hơn PCR với dữ liệu NI
- Số thành phần tối ưu thay đổi tùy theo từng chỉ tiêu hóa lý
- VIP scores giúp xác định các bước sóng quan trọn
- Mô hình có thể dùng để dự đoán chất lượng cà phê từ phổ NIR

Ý nghĩa thực tiễn:

- Các mô hình PLS/PCR cho phép dự đoán nhanh các chỉ tiêu hóa lý từ phổ NIR
- Không cần phân tích hóa học tốn kém và mất thời gian
- Phân tích VIP giúp hiểu được các vùng phổ quan trọng liên quan đến từng chất
- Có thể áp dụng trong kiểm soát chất lượng và phân loại cà phê