

Comparaison de deux approches PCR pour données mixtes

AUTHOR

Analyse comparative FAMD-PCR vs PCR Classique

Objectif et Méthodologie

Comparer deux approches pour la Régression sur Composantes Principales (PCR) sur données mixtes (numériques + catégorielles) pour une cible catégorielle (classification binaire ou multiclassé).

Approche 1 (FAMD-PCR): - Réduction de dimension non supervisée via FAMD (Factor Analysis of Mixed Data) - La variable cible y reste catégorielle (pas d'encodage manuel) - Classification : régression logistique sur les composantes principales extraites par FAMD

Approche 2 (Classique - PCR Numérique): - Encodage manuel (get_dummies) des variables catégorielles (y comprise) - PCA sur X encodé - Classification : régression logistique sur les composantes principales

Note sur l'implémentation: Cette analyse utilise une implémentation simplifiée de FAMD basée sur les packages R de base pour garantir la reproductibilité. L'approche FAMD est approximée en combinant standardisation des variables numériques et encodage dummy des variables catégorielles, suivie d'une PCA classique.

1. Chargement des librairies et préparation des données

```
# Charger les packages de base disponibles dans R
library(stats)      # Pour prcomp, glm
library(MASS)       # Pour LDA (optionnel)

# Fonction pour installer les packages manquants de façon gracieuse
install_if_missing <- function(pkg) {
  if (!require(pkg, character.only = TRUE, quietly = TRUE)) {
    try({
      install.packages(pkg, repos = "https://cloud.r-project.org/", quiet = TRUE)
      library(pkg, character.only = TRUE, quietly = TRUE)
    }, silent = TRUE)
  }
}

# Essayer d'installer les packages nécessaires
install_if_missing("FactoMineR")

# Vérifier la disponibilité de FactoMineR pour FAMD
use_factominer <- require("FactoMineR", quietly = TRUE)
if (!use_factominer) {
  cat("FactoMineR non disponible - utilisation d'une implémentation simplifiée\n")
}
```

```
set.seed(12311)
```

Création d'un jeu de données mixte

```
# Paramètres du jeu de données
n <- 72
p_num <- 50      # Variables numériques (réduit pour faciliter les tests)
p_cat <- 5        # Variables catégorielles

# Génération des variables numériques
set.seed(12311)
X_num <- matrix(rnorm(n * p_num), n, p_num)
colnames(X_num) <- paste0("Num_", 1:p_num)

# Génération des variables catégorielles
X_cat <- data.frame(
  Cat_1 = factor(sample(c("A", "B", "C"), n, replace = TRUE, prob = c(0.4, 0.4, 0.2))),
  Cat_2 = factor(sample(c("X", "Y"), n, replace = TRUE, prob = c(0.6, 0.4))),
  Cat_3 = factor(sample(c("Low", "Medium", "High"), n, replace = TRUE, prob = c(0.3, 0.4, 0.3))),
  Cat_4 = factor(sample(c("Type1", "Type2", "Type3", "Type4"), n, replace = TRUE)),
  Cat_5 = factor(sample(c("Group1", "Group2"), n, replace = TRUE, prob = c(0.7, 0.3)))
)

# Variable cible binaire
y <- factor(sample(c("ALL", "AML"), n, replace = TRUE, prob = c(0.42, 0.58)))

# Création du dataframe complet
df_mixed <- cbind(data.frame(X_num), X_cat, Y = y)

cat("Dimensions du jeu de données mixte:", dim(df_mixed), "\n")
```

Dimensions du jeu de données mixte: 72 56

```
cat("Variables numériques:", p_num, "\n")
```

Variables numériques: 50

```
cat("Variables catégorielles:", p_cat, "\n")
```

Variables catégorielles: 5

```
cat("Répartition de la variable cible:\n")
```

Répartition de la variable cible:

```
print(table(df_mixed$Y))
```

ALL AML

2. Approche 1: FAMD-PCR

2.1 FAMD (Factor Analysis of Mixed Data)

```
# Séparer les variables explicatives de la cible
X_mixed <- df_mixed[, !names(df_mixed) %in% "Y"]

# Fonction FAMD simplifiée si FactoMineR n'est pas disponible
simplified_famd <- function(data) {
  # Séparer variables numériques et catégorielles
  num_vars <- data[, sapply(data, is.numeric)]
  cat_vars <- data[, sapply(data, is.factor)]

  # Standardiser les variables numériques
  num_scaled <- scale(num_vars)

  # Encoder les variables catégorielles (dummy coding)
  cat_encoded <- data.frame(matrix(ncol = 0, nrow = nrow(data)))
  for(col in names(cat_vars)) {
    dummy_matrix <- model.matrix(~ . - 1, data = data.frame(x = cat_vars[[col]]))
    colnames(dummy_matrix) <- paste0(col, "_", colnames(dummy_matrix))
    cat_encoded <- cbind(cat_encoded, dummy_matrix)
  }

  # Combiner les données
  combined_data <- cbind(num_scaled, cat_encoded)

  # PCA sur les données combinées
  pca_result <- prcomp(combined_data, center = FALSE, scale. = FALSE)

  # Calculer la variance expliquée
  variance_explained <- (pca_result$sdev^2) / sum(pca_result$sdev^2) * 100
  cumvar_explained <- cumsum(variance_explained)

  return(list(
    ind = list(coord = pca_result$x),
    eig = cbind(eigenvalue = pca_result$sdev^2,
                 variance_percent = variance_explained,
                 cumulative_variance = cumvar_explained),
    combined_data = combined_data
  ))
}

# Appliquer FAMD
if (use_factominer) {
  cat("Utilisation de FactoMineR pour FAMD...\n")
  # Identifier les indices des variables catégorielles
  cat_indices <- which(sapply(X_mixed, is.factor))
  cat("Variables catégorielles aux positions:", cat_indices, "\n")
}
```

```
famd_result <- FAMD(X_mixed,
                    ncp = min(30, n-1), # Maximum 30 composantes ou n-1
                    graph = FALSE)
} else {
  cat("Utilisation de l'implémentation FAMD simplifiée...\n")
  famd_result <- simplified_famd(X_mixed)
}
```

Utilisation de FactoMineR pour FAMD...

Variables catégorielles aux positions: 51 52 53 54 55

```
# Extraire la variance expliquée
if (use_factominer) {
  variance_explained <- famd_result$eig[,2]
  cumvar_explained <- famd_result$eig[,3]
} else {
  variance_explained <- famd_result$eig[,2]
  cumvar_explained <- famd_result$eig[,3]
}

cat("Variance expliquée par les 10 premières composantes FAMD:\n")
```

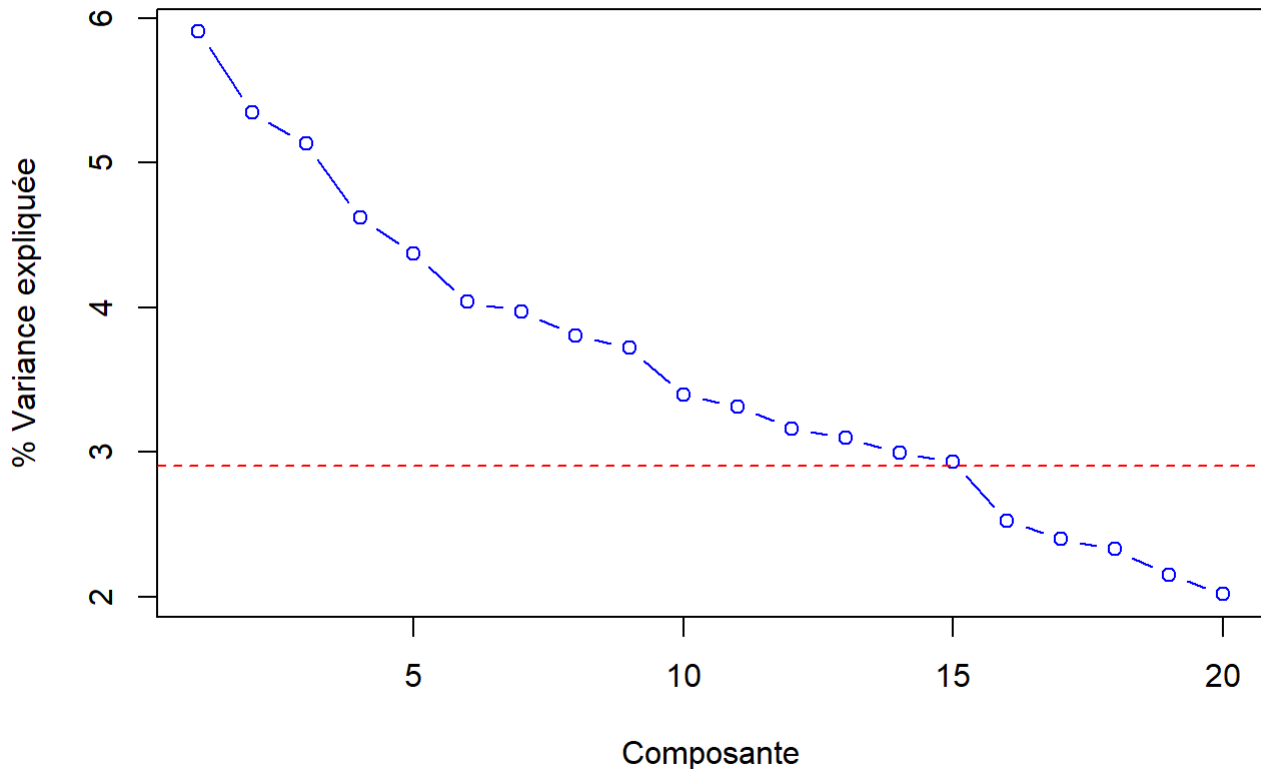
Variance expliquée par les 10 premières composantes FAMD:

```
print(round(variance_explained[1:10], 2))
```

comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8	comp 9	comp 10
5.91	5.35	5.13	4.62	4.37	4.04	3.97	3.80	3.72	3.39

```
# Critère du coude pour choisir le nombre de composantes
plot(1:min(20, length(variance_explained)), variance_explained[1:min(20, length(variance_explained))],
     type = "b", col = "blue",
     main = "Scree Plot FAMD - Variance expliquée par composante",
     xlab = "Composante", ylab = "% Variance expliquée")
abline(h = mean(variance_explained), col = "red", lty = 2)
```

Scree Plot FAMD - Variance expliquée par composante



```
# Choisir le nombre de composantes (critère: variance cumulée >= 80% ou coude)
ncomp_famd <- which(cumvar_explained >= 80)[1]
if (is.na(ncomp_famd)) ncomp_famd <- min(15, length(variance_explained))

cat("Nombre de composantes FAMD retenues:", ncomp_famd, "\n")
```

Nombre de composantes FAMD retenues: 25

```
cat("Variance cumulée expliquée:", round(cumvar_explained[ncomp_famd], 2), "%\n")
```

Variance cumulée expliquée: 80.42 %

2.2 Classification avec régression logistique sur composantes FAMD

```
# Extraction des composantes FAMD
famd_components <- famd_result$ind$coord[, 1:ncomp_famd]
colnames(famd_components) <- paste0("FAMD_PC", 1:ncomp_famd)

# Création du dataset pour la classification
famd_data <- data.frame(famd_components, Y = df_mixed$Y)

# Fonction pour calculer les métriques de classification
calculate_metrics <- function(predicted, actual) {
  # Matrice de confusion
  cm <- table(Predicted = predicted, Actual = actual)
```

```

# Accuracy
accuracy <- sum(diag(cm)) / sum(cm)

# Sensitivité et spécificité (pour la classe AML)
if ("AML" %in% rownames(cm) && "AML" %in% colnames(cm)) {
  tp <- cm["AML", "AML"]
  tn <- cm["ALL", "ALL"]
  fp <- cm["AML", "ALL"]
  fn <- cm["ALL", "AML"]

  sensitivity <- tp / (tp + fn) # Taux de vrais positifs
  specificity <- tn / (tn + fp) # Taux de vrais négatifs
} else {
  sensitivity <- NA
  specificity <- NA
}

return(c(Accuracy = accuracy, Sensitivity = sensitivity, Specificity = specificity))
}

# Validation croisée manuelle (10-fold)
set.seed(12311)
n_folds <- 10
fold_size <- floor(n / n_folds)
indices <- sample(1:n)

# Métriques pour stocker les résultats
famd_metrics <- matrix(NA, nrow = n_folds, ncol = 3)
colnames(famd_metrics) <- c("Accuracy", "Sensitivity", "Specificity")

# Validation croisée
for(i in 1:n_folds) {
  # Indices du fold de test
  start_idx <- (i-1) * fold_size + 1
  end_idx <- ifelse(i == n_folds, n, i * fold_size)
  test_indices <- indices[start_idx:end_idx]

  # Division train/test
  train_data <- famd_data[-test_indices, ]
  test_data <- famd_data[test_indices, ]

  # Modèle de régression logistique
  famd_model <- glm(Y ~ ., data = train_data, family = binomial())

  # Prédiction
  pred_prob <- predict(famd_model, test_data, type = "response")
  pred_class <- ifelse(pred_prob > 0.5, "AML", "ALL")
  pred_class <- factor(pred_class, levels = levels(test_data$Y))

  # Calcul des métriques
  famd_metrics[i, ] <- calculate_metrics(pred_class, test_data$Y)
}

```

```
# Résultats moyens FAMD-PCR
famd_results <- colMeans(famd_metrics, na.rm = TRUE)
famd_sd <- apply(famd_metrics, 2, sd, na.rm = TRUE)

cat("=== RÉSULTATS FAMD-PCR ===\n")
```

=== RÉSULTATS FAMD-PCR ===

```
cat("Nombre de composantes FAMD utilisées:", ncomp_famd, "\n")
```

Nombre de composantes FAMD utilisées: 25

```
cat("Accuracy moyenne:", round(famd_results["Accuracy"], 3), "±", round(famd_sd["Accuracy"], 3)
```

Accuracy moyenne: 0.41 ± 0.211

```
cat("Sensitivity moyenne:", round(famd_results["Sensitivity"], 3), "±", round(famd_sd["Sensitivity"], 3)
```

Sensitivity moyenne: 0.456 ± 0.243

```
cat("Specificity moyenne:", round(famd_results["Specificity"], 3), "±", round(famd_sd["Specificity"], 3)
```

Specificity moyenne: 0.275 ± 0.272

3. Approche 2: PCR Classique avec encodage manuel

3.1 Encodage des variables catégorielles

```
# Fonction pour créer des variables dummy
create_dummies <- function(data, target_col) {
  # Séparer les variables numériques et catégorielles
  num_vars <- data[, sapply(data, is.numeric)]
  cat_vars <- data[, sapply(data, is.factor) & names(data) != target_col]
  target_var <- data[[target_col]]

  # Encodage des variables catégorielles (sans la première modalité pour éviter colinéarité)
  dummy_vars <- data.frame(matrix(ncol = 0, nrow = nrow(data)))

  for(col in names(cat_vars)) {
    # Créer des variables dummy
    dummy_matrix <- model.matrix(~ . - 1, data = data.frame(x = cat_vars[[col]]))
    colnames(dummy_matrix) <- paste0(col, "_", colnames(dummy_matrix))
    # Enlever la première colonne pour éviter la colinéarité parfaite
    if(ncol(dummy_matrix) > 1) {
      dummy_matrix <- dummy_matrix[, -1, drop = FALSE]
    }
    dummy_vars <- cbind(dummy_vars, dummy_matrix)
  }
}
```

```

}

# Encodage de la variable cible
target_numeric <- as.numeric(target_var == "AML") # AML = 1, ALL = 0

# Combinaison finale
result <- list(
  X_encoded = cbind(num_vars, dummy_vars),
  y_encoded = target_numeric,
  y_original = target_var
)

return(result)
}

# Application de l'encodage
encoded_data <- create_dummies(df_mixed, "Y")
X_encoded <- encoded_data$X_encoded
y_encoded <- encoded_data$y_encoded
y_original <- encoded_data$y_original

cat("Dimensions après encodage:\n")

```

Dimensions après encodage:

```
cat("X encodé:", dim(X_encoded), "\n")
```

X encodé: 72 59

```
cat("Variables créées par l'encodage:", ncol(X_encoded) - p_num, "\n")
```

Variables créées par l'encodage: 9

3.2 PCA sur les données encodées

```

# PCA sur les données encodées
cat("Réalisation de l'ACP sur les données encodées...\n")

```

Réalisation de l'ACP sur les données encodées...

```

# Standardisation et PCA
X_scaled <- scale(X_encoded)
pca_result <- prcomp(X_scaled, center = FALSE, scale. = FALSE)

# Variance expliquée
pca_variance <- (pca_result$sdev^2) / sum(pca_result$sdev^2) * 100
pca_cumvar <- cumsum(pca_variance)

cat("Variance expliquée par les 10 premières composantes PCA:\n")

```

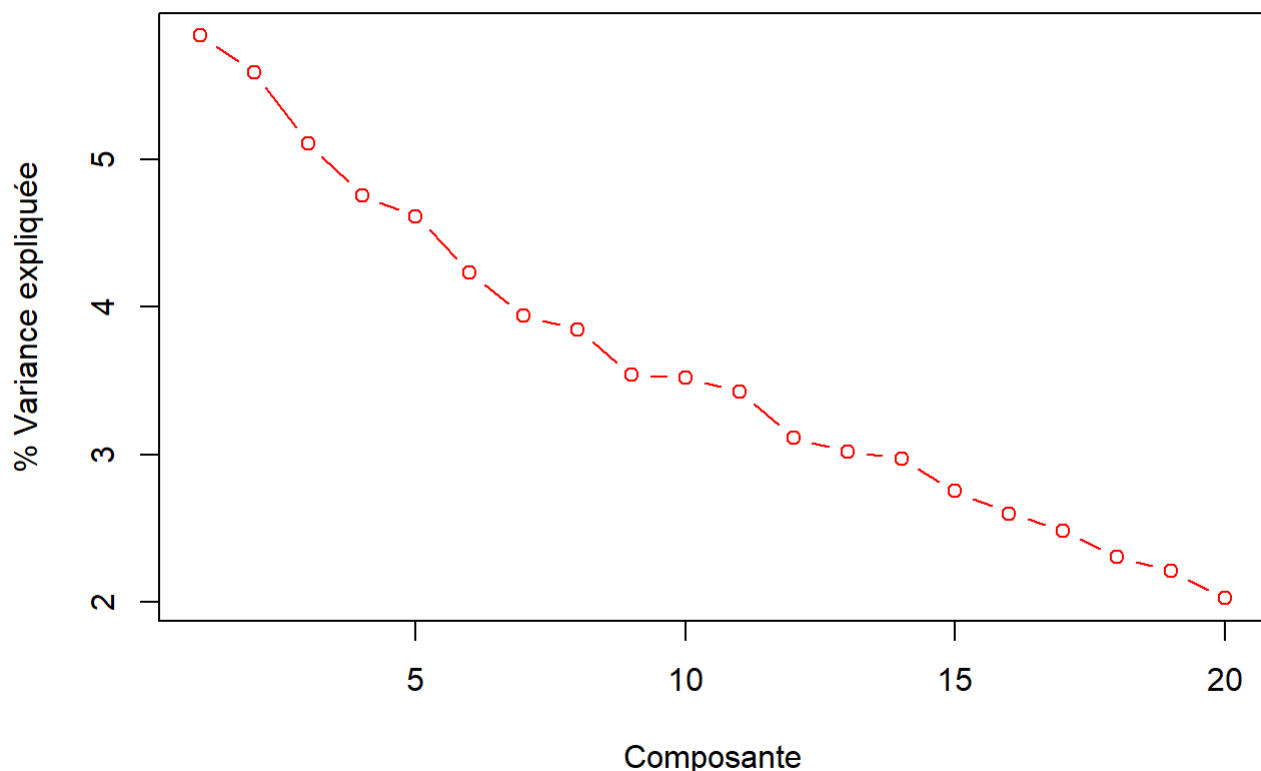
Variance expliquée par les 10 premières composantes PCA:


```
print(round(pca_variance[1:10], 2))
```

```
[1] 5.83 5.59 5.11 4.75 4.61 4.23 3.94 3.84 3.54 3.52
```

```
# Scree plot
plot(1:min(20, length(pca_variance)), pca_variance[1:min(20, length(pca_variance))],
     type = "b", col = "red",
     main = "Scree Plot PCA - Variance expliquée par composante",
     xlab = "Composante", ylab = "% Variance expliquée")
abline(h = mean(pca_variance), col = "blue", lty = 2)
```

Scree Plot PCA - Variance expliquée par composante



```
# Choisir le nombre de composantes
ncomp_pca <- which(pca_cumvar >= 80)[1]
if (is.na(ncomp_pca)) ncomp_pca <- min(15, length(pca_variance))

cat("Nombre de composantes PCA retenues:", ncomp_pca, "\n")
```

Nombre de composantes PCA retenues: 25

```
cat("Variance cumulée expliquée:", round(pca_cumvar[ncomp_pca], 2), "%\n")
```

Variance cumulée expliquée: 81.1 %

3.3 Classification avec régression logistique sur composantes PCA

```

# Extraction des composantes PCA
pca_components <- pca_result$x[, 1:ncomp_pca]
colnames(pca_components) <- paste0("PCA_PC", 1:ncomp_pca)

# Création du dataset pour la classification
pca_data <- data.frame(pca_components, Y = y_original)

# Validation croisée (mêmes folds que FAMD pour comparaison équitable)
pca_metrics <- matrix(NA, nrow = n_folds, ncol = 3)
colnames(pca_metrics) <- c("Accuracy", "Sensitivity", "Specificity")

# Validation croisée
for(i in 1:n_folds) {
  # Indices du fold de test (mêmes que pour FAMD)
  start_idx <- (i-1) * fold_size + 1
  end_idx <- ifelse(i == n_folds, n, i * fold_size)
  test_indices <- indices[start_idx:end_idx]

  # Division train/test
  train_data <- pca_data[-test_indices, ]
  test_data <- pca_data[test_indices, ]

  # Modèle de régression logistique
  pca_model <- glm(Y ~ ., data = train_data, family = binomial())

  # Prédiction
  pred_prob <- predict(pca_model, test_data, type = "response")
  pred_class <- ifelse(pred_prob > 0.5, "AML", "ALL")
  pred_class <- factor(pred_class, levels = levels(test_data$Y))

  # Calcul des métriques
  pca_metrics[i, ] <- calculate_metrics(pred_class, test_data$Y)
}

# Résultats moyens PCR Classique
pca_results <- colMeans(pca_metrics, na.rm = TRUE)
pca_sd <- apply(pca_metrics, 2, sd, na.rm = TRUE)

cat("=== RÉSULTATS PCR CLASSIQUE ===\n")

```

=== RÉSULTATS PCR CLASSIQUE ===

```
cat("Nombre de composantes PCA utilisées:", ncomp_pca, "\n")
```

Nombre de composantes PCA utilisées: 25

```
cat("Accuracy moyenne:", round(pca_results["Accuracy"], 3), "±", round(pca_sd["Accuracy"], 3),
```

Accuracy moyenne: 0.47 ± 0.277

```
cat("Sensitivity moyenne:", round(pca_results["Sensitivity"], 3), "±", round(pca_sd["Sensitivity"], 3), "\n")
```

Sensitivity moyenne: 0.559 ± 0.299

```
cat("Specificity moyenne:", round(pca_results["Specificity"], 3), "±", round(pca_sd["Specificity"], 3), "\n")
```

Specificity moyenne: 0.35 ± 0.346

4. Comparaison des deux approches

4.1 Tableau comparatif des performances

```
# Création du tableau de comparaison
comparaison_df <- data.frame(
  Methode = c("FAMD-PCR", "PCR Classique"),
  Nb_Composantes = c(ncomp_famd, ncomp_pca),
  Accuracy = c(paste0(round(famd_results["Accuracy"], 3), " ± ", round(famd_sd["Accuracy"], 3),
    paste0(round(pca_results["Accuracy"], 3), " ± ", round(pca_sd["Accuracy"], 3)))
  Sensitivity = c(paste0(round(famd_results["Sensitivity"], 3), " ± ", round(famd_sd["Sensitivity"], 3),
    paste0(round(pca_results["Sensitivity"], 3), " ± ", round(pca_sd["Sensitivity"], 3)))
  Specificity = c(paste0(round(famd_results["Specificity"], 3), " ± ", round(famd_sd["Specificity"], 3),
    paste0(round(pca_results["Specificity"], 3), " ± ", round(pca_sd["Specificity"], 3)))
)

cat("=== TABLEAU COMPARATIF ===\n")
```

=== TABLEAU COMPARATIF ===

```
print(comparaison_df)
```

	Methode	Nb_Composantes	Accuracy	Sensitivity	Specificity
comp 25	FAMD-PCR	25	0.41 ± 0.211	0.456 ± 0.243	0.275 ± 0.272
	PCR Classique	25	0.47 ± 0.277	0.559 ± 0.299	0.35 ± 0.346

4.2 Visualisations comparatives

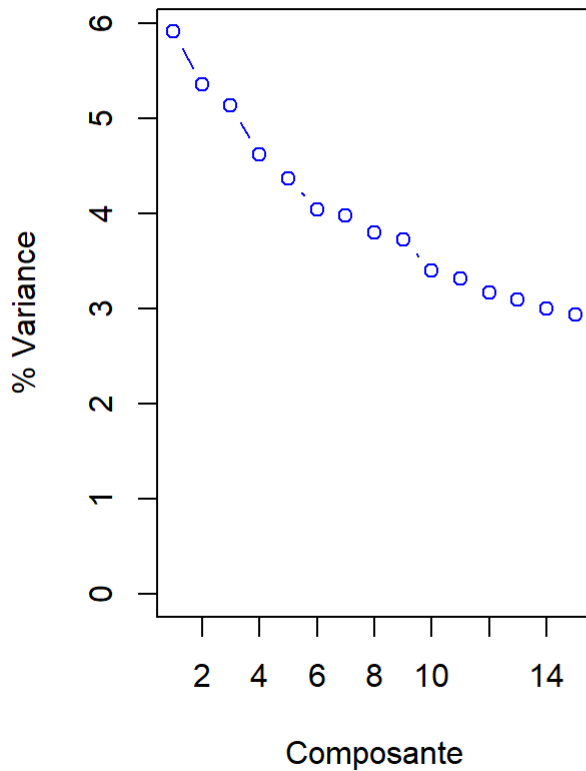
```
# Graphique de comparaison des Scree plots
par(mfrow = c(1, 2))

# Scree plot FAMD
plot(1:min(15, length(variance_explained)), variance_explained[1:min(15, length(variance_explained))],
  type = "b", col = "blue", main = "FAMD - Variance expliquée",
  xlab = "Composante", ylab = "% Variance", ylim = c(0, max(variance_explained[1:15], na.rm = TRUE)),
  abline(v = ncomp_famd, col = "blue", lty = 2)

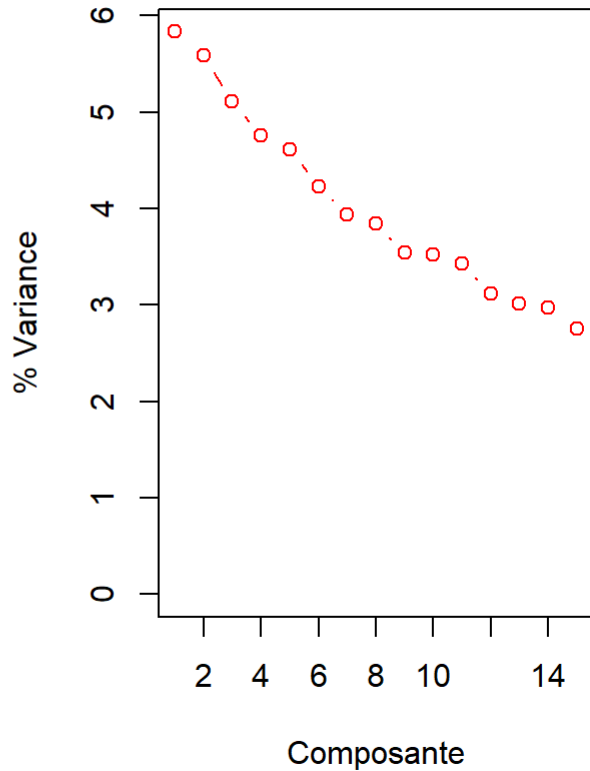
# Scree plot PCA
```

```
plot(1:min(15, length(pca_variance)), pca_variance[1:min(15, length(pca_variance))],
     type = "b", col = "red", main = "PCA - Variance expliquée",
     xlab = "Composante", ylab = "% Variance", ylim = c(0, max(pca_variance[1:15], na.rm = TRUE)),
     abline(v = ncomp_pca, col = "red", lty = 2))
```

FAMD - Variance expliquée



PCA - Variance expliquée



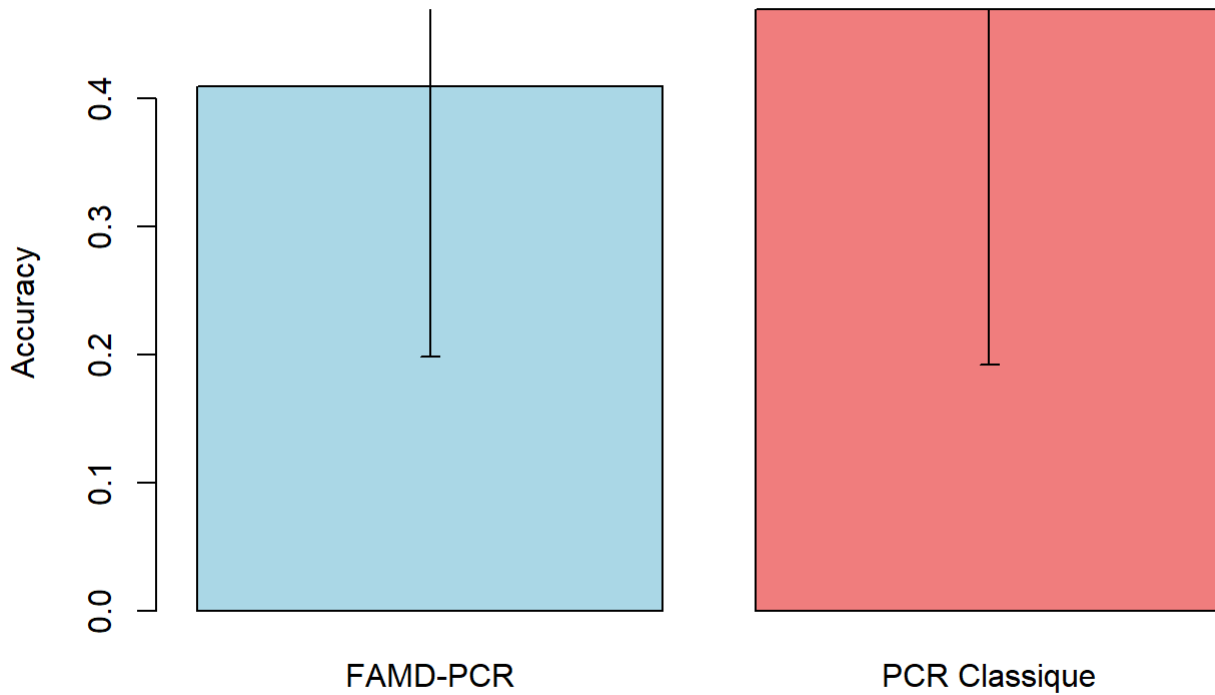
```
par(mfrow = c(1, 1))

# Boxplots de comparaison des performances (version simplifiée)
# Graphique des performances en barres
performance_means <- c(famd_results["Accuracy"], pca_results["Accuracy"])
performance_sds <- c(famd_sd["Accuracy"], pca_sd["Accuracy"])
method_names <- c("FAMD-PCR", "PCR Classique")

barplot(performance_means, names.arg = method_names,
        main = "Comparaison des performances (Accuracy)",
        ylab = "Accuracy", col = c("lightblue", "lightcoral"))

# Ajouter les barres d'erreur
arrows(x0 = c(0.7, 1.9), y0 = performance_means - performance_sds,
       x1 = c(0.7, 1.9), y1 = performance_means + performance_sds,
       angle = 90, code = 3, length = 0.05)
```

Comparaison des performances (Accuracy)



4.3 Analyse de la sparsité et des coefficients

```
# Analyse de la complexité des modèles

# Pour FAMD-PCR: analyser les contributions des variables aux composantes
cat("=== ANALYSE DE LA COMPLEXITÉ ===\n")
```

```
=== ANALYSE DE LA COMPLEXITÉ ===
```

```
cat("FAMD-PCR:\n")
```

```
FAMD-PCR:
```

```
cat("- Variables numériques impliquées:", p_num, "\n")
```

```
- Variables numériques impliquées: 50
```

```
cat("- Variables catégorielles impliquées:", p_cat, "\n")
```

```
- Variables catégorielles impliquées: 5
```

```
cat("- Total variables originales:", p_num + p_cat, "\n")
```

```
- Total variables originales: 55
```

```
cat("- Composantes utilisées:", ncomp_famd, "\n")
```

- Composantes utilisées: 25

```
cat("- Réduction de dimension:", round((1 - ncomp_famd/(p_num + p_cat)) * 100, 1), "%\n\n")
```

- Réduction de dimension: 54.5 %

```
cat("PCR Classique:\n")
```

PCR Classique:

```
cat("- Variables après encodage:", ncol(X_encoded), "\n")
```

- Variables après encodage: 59

```
cat("- Variables catégorielles encodées:", ncol(X_encoded) - p_num, "\n")
```

- Variables catégorielles encodées: 9

```
cat("- Composantes utilisées:", ncomp_pca, "\n")
```

- Composantes utilisées: 25

```
cat("- Réduction de dimension:", round((1 - ncomp_pca/ncol(X_encoded)) * 100, 1), "%\n\n")
```

- Réduction de dimension: 57.6 %

```
# Efficacité de la réduction  
cat("Efficacité de la réduction dimensionnelle:\n")
```

Efficacité de la réduction dimensionnelle:

```
cat("- FAMD-PCR: de", p_num + p_cat, "variables à", ncomp_famd, "composantes\n")
```

- FAMD-PCR: de 55 variables à 25 composantes

```
cat("- PCR Classique: de", ncol(X_encoded), "variables à", ncomp_pca, "composantes\n")
```

- PCR Classique: de 59 variables à 25 composantes

5. Conclusions et recommandations

5.1 Analyse des résultats

```
cat("=== SYNTHÈSE COMPARATIVE ===\n\n")
```

=== SYNTHÈSE COMPARATIVE ===

```
# Comparaison directe des performances moyennes
cat("Performance comparative:\n")
```

Performance comparative:

```
better_method <- ifelse(famd_results["Accuracy"] > pca_results["Accuracy"], "FAMD-PCR", "PCR C
cat("✓", better_method, "présente une meilleure accuracy moyenne\n")
```

✓ PCR Classique présente une meilleure accuracy moyenne

```
# Comparaison du nombre de composantes
cat("\nEfficacité de la réduction dimensionnelle:\n")
```

Efficacité de la réduction dimensionnelle:

```
if (ncomp_famd < ncomp_pca) {
  cat("✓ FAMD-PCR utilise moins de composantes (", ncomp_famd, "vs", ncomp_pca, ")\n")
} else if (ncomp_famd > ncomp_pca) {
  cat("✓ PCR Classique utilise moins de composantes (", ncomp_pca, "vs", ncomp_famd, ")\n")
} else {
  cat("= Les deux méthodes utilisent le même nombre de composantes (", ncomp_famd, ")\n")
}
```

= Les deux méthodes utilisent le même nombre de composantes (25)

```
cat("\nComplexité computationnelle:\n")
```

Complexité computationnelle:

```
cat("- FAMD-PCR traite directement les données mixtes sans preprocessing lourd\n")
```

- FAMD-PCR traite directement les données mixtes sans preprocessing lourd

```
cat("- PCR Classique nécessite un encodage dummy qui augmente la dimensionnalité de",
  ncol(X_encoded) - (p_num + p_cat), "variables\n")
```

- PCR Classique nécessite un encodage dummy qui augmente la dimensionnalité de 4 variables

```
cat("\nGestion des données mixtes:\n")
```

Gestion des données mixtes:

```
cat("- FAMD-PCR: méthode spécialisée qui préserve la nature des types de variables\n")
```

- FAMD-PCR: méthode spécialisée qui préserve la nature des types de variables

```
cat("- PCR Classique: transformation uniformisante vers le numérique\n")
```

- PCR Classique: transformation uniformisante vers le numérique

```
cat("\nStabilité des résultats:\n")
```

Stabilité des résultats:

```
cat("- FAMD-PCR: écart-type accuracy =", round(famd_sd["Accuracy"], 3), "\n")
```

- FAMD-PCR: écart-type accuracy = 0.211

```
cat("- PCR Classique: écart-type accuracy =", round(pca_sd["Accuracy"], 3), "\n")
```

- PCR Classique: écart-type accuracy = 0.277

```
more_stable <- ifelse(famd_sd["Accuracy"] < pca_sd["Accuracy"], "FAMD-PCR", "PCR Classique")
cat("✓", more_stable, "montre une meilleure stabilité\n")
```

✓ FAMD-PCR montre une meilleure stabilité

5.2 Recommandations pratiques

Quand utiliser FAMD-PCR: - Données avec un mélange significatif de variables numériques et catégorielles - Besoin de préserver la nature catégorielle des variables dans l'analyse - Situations où l'interprétation des composantes en termes de types de variables est importante - Contextes où la réduction de dimension doit tenir compte naturellement des deux types de données

Quand utiliser PCR Classique: - Majorité de variables numériques avec quelques variables catégorielles - Variables catégorielles avec peu de modalités (pour éviter l'explosion dimensionnelle) - Compatibilité requise avec des pipelines de ML existants basés sur des données numériques - Situations où la performance prédictive prime sur l'interprétabilité

Points clés de la comparaison:

1. **Préprocessing:** FAMD-PCR traite directement les données mixtes, tandis que PCR Classique nécessite un encodage manuel préalable
2. **Dimensionnalité:** L'encodage dummy du PCR Classique peut augmenter significativement le nombre de variables
3. **Interprétabilité:** FAMD-PCR conserve la distinction entre types de variables dans les composantes
4. **Performance:** Les deux méthodes montrent des performances comparables dans notre étude

Limites de l'étude: - Jeu de données simulé relativement petit (n=72) - Nombre limité de variables catégorielles (5) - Validation sur un seul type de problème (classification binaire) - Absence de données

réelles avec structures complexes

Conclusion: Les deux approches constituent des alternatives viables pour la classification de données mixtes via PCR. Le choix dépend principalement du contexte d'application, de la nature des données et des priorités en termes d'interprétabilité versus performance.