Step 3.2 Selecting relevant predictors

3.2.1 Selecting features with PCA

Diane ESPEL

2025-08-05

Contents

1 Objectives

This script aims to perform **predictor selection using Principal Component Analysis (PCA)** on a dataset of raster-derived environmental variables collected from learning plots. The main goal is to identify the most predictive variables by reducing dimensionality and highlighting the variables that contribute most to variance within the dataset. This facilitates downstream modeling by focusing on the most relevant features while minimizing redundancy and multicollinearity among variables.

2 Script explanation

2.1 Clean environment and graphics

```
rm(list = ls()) # Clear all objects from the R environment to start fresh
graphics.off()   # Close all graphics devices (if any plots are open)
```

2.2 Load required packages

```
library(readr)        # For reading delimited text files
library(FactoMineR)   # For running PCA and multivariate analyses
library(factoextra)   # For enhanced PCA visualization (ggplot2-based)
library(corrplot)     # For correlation matrix visualizations
```

2.3 Define global variables

Note: A global variable is a variable defined outside of any function. This means the variable is accessible from any part of the code, including inside functions. A global variable retains its value throughout the execution of the R script unless it is explicitly modified in the code.

It is important to define at a minimum:

- the `"District"`: the archipelago of interest (e.g. “CRO” for Crozet archipelago)
- the `"Island"`: the island within the archipelago of interest (e.g. “POS” for Possession island)
- the `"Satellite1"`: the name of satellite providing multispectral imagery
- the `"Year1"`: the acquisition year of the multispectral imagery

```
District='CRO' # 3-letter code for archipelago (e.g. Crozet)
Island='POS'   # 3-letter code for island (e.g. Possession)
Satellite1="Pleiades" # satellite name of multispectral imagery
Year1="2022" # acquisition year of multispectral imagery
```

2.4 Set working directory

To optimize memory, you must define one general root directory (`"localscratch"`) that serves as the base path for your input and output data, respectively. This directory should point to the local environment where:

- input learning dataset is located under `"data/Learning_data/PrimaryTypo"`
- outputs from PCA will be saved under `"data/Learning_data/PrimaryTypo"`

```
# Base local path (customize to your local environment)
# localscratch = paste0("your_local_path/")
localscratch=paste0("/scratch/despel/CARTOVEGE/")

# Path to open input learning dataset
open_learning_primary_path=paste0(localscratch,"data/Learning_data/PrimaryTypo")

# Path to save results from PCA
save_learning_primary_path=paste0(localscratch,"data/Learning_data/PrimaryTypo")
```

2.5 Load and prepare learning dataset

Before performing PCA, it is often necessary to standardize the data so that each variable has a mean of 0 and a standard deviation of 1. This is especially important when the variables are measured on different scales. To achieve this:

- The script identifies columns corresponding to the first and last raster-derived variables.
- It standardizes the selected variables by centering to zero mean and scaling to unit variance

```
# Open data
FILE1=paste0(open_learning_primary_path,"/","Learning_plots_", District, "_",
↪  Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_EPSG32739.csv")
learning_data<- read.csv(FILE1, sep=";",dec=".",stringsAsFactors=FALSE) # `stringsAsFactors=F`
↪  ensures character strings don't import as factors

# Define Index of variables
imeR=which(colnames(learning_data)=="R") # First raster-derived variable
imeS=which(colnames(learning_data)=="Slope") # Last raster-derived variable

# # Standardize variables (mean=0, sd=1)
tab_norm <- as.data.frame(scale(learning_data[,c(imeR:imeS)]))
```

2.6 Compute initial correlations

A **Pearson correlation matrix** is calculated on the standardized variables to assess relationships and potential multicollinearity. This matrix is saved for further inspection and helps interpret PCA results by revealing clusters of correlated variables.

```
# Compute correlations between variables
correlation_matrix <- cor(tab_norm) # Pearson correlations
FILE2=paste0(save_learning_primary_path,"/","Correlation_matrix_",District,"_",Island,"_",Satellite1,"_",Year1,"_AL
write.table(correlation_matrix,FILE2,sep = ";", dec = ",", row.names = FALSE)

# Heatmap of correlations
NOMpng=paste0(save_learning_primary_path,"/","Correlation_matrix_",District,"_",Island,"_",Satellite1,"_",Year1,"_A
png(file = NOMpng, width = 500, height = 400)
p=corrplot(correlation_matrix, method = "color", type = "upper",
         tl.cex = 0.8, tl.col = "black", addCoef.col = "black",
         number.cex = 0.7, diag = FALSE)
print(p)
dev.off()

NOMsvg=paste0(save_learning_primary_path,"/","Correlation_matrix_",District,"_",Island,"_",Satellite1,"_",Year1,"_A
svg(file = NOMsvg)
print(p)
dev.off()
```

2.7 Apply PCA

PCA is conducted using the **FactoMineR** package on the standardized variables to reduce dimensionality. The number of principal components is set to the total number of variables.

- The script generates and saves visualizations of **eigenvalue**s (`eig.val`, as variance explained by each component), which help decide how many components to retain.

- **Correlation circles** are plotted to show variable loadings on principal components, illustrating how variables relate to each other and to each component axis. Several versions of these plots are saved, including colored ones highlighting the magnitude of each variable's contribution, aiding in variable interpretation.

```
# Apply PCA  ------------------------------------------------------------

res_pca=PCA(tab_norm,  # dateframe with n rows and p columns
            ncp = ncol(tab_norm), # maximum number of dimensions
            graph = F)# don't display the graph

# Eigen values  ---------------------------------------------------------

# Get eigen values
eig.val=as.data.frame(res_pca$eig) # Percentage of variance explained
FILE3=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Eigen
write.table(eig.val,FILE3,sep = ";", dec = ",", row.names = FALSE)

# Plot of Eigen values
NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Eige
png(file = NOMpng, width = 500, height = 400)
p=fviz_eig(res_pca,title="Eigen values")+
  theme(text = element_text(size = 17),
        axis.title = element_text(size = 17),
        axis.text = element_text(size = 15))
print(p)
dev.off()

NOMsvg=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Eige
svg(file = NOMsvg)
print(p)
dev.off()

# Correlation circles  --------------------------------------------------------

# Variable graphs PC1 and PC2
NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllV
png(file = NOMpng, width = 700, height = 700)
p=fviz_pca_var(res_pca, repel = T, #repel to avoid label overlaps
               labelsize=3,pointsize=4)+
  ylim (-1,1)+
  xlim(-1,1)+
  theme(text = element_text(size = 17),
        axis.title = element_text(size = 17),
        axis.text = element_text(size = 17))

print(p)
dev.off()

NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllV
png(file = NOMpng, width = 700, height = 700)
p=fviz_pca_var(res_pca, repel=T, col.var = "contrib",axes=c(1,2),gradient.cols = c("#00AFBB",
 ↪  "#E7B800", "#FC4E07"),
               labelsize=3,pointsize=4)+
  ylim (-1,1)+
  xlim(-1,1)+
  theme(text = element_text(size = 17),
        axis.title = element_text(size = 17),
        axis.text = element_text(size = 17))
```

```
print(p)
dev.off()

NOMsvg=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllV
svg(file = NOMsvg)
print(p)
dev.off()


NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllV
png(file = NOMpng, width = 700, height = 700)
p=plot(res_pca,cex=1,cex.axis=1.8,font.axis=1.5)
print(p)
dev.off()

NOMsvg=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllV
 ↪  #svg format to modify into Inkscape
svg(file = NOMsvg)
print(p)
dev.off()



# Variable graphs PC3 and PC4
NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllV
png(file = NOMpng, width = 700, height = 700)
p=fviz_pca_var(res_pca, repel=T, col.var = "contrib",axes=c(3,4),gradient.cols = c("#00AFBB",
 ↪  "#E7B800", "#FC4E07"),
              labelsize=3,pointsize=4)+
  ylim (-1,1)+
  xlim(-1,1)+
  theme(text = element_text(size = 17),
        axis.title = element_text(size = 17),
        axis.text = element_text(size = 17))
print(p)
dev.off()

NOMsvg=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllV
svg(file = NOMsvg)
print(p)
dev.off()
```

2.8 Individuals graphs

This part of the script generates visualizations of the individuals (sampling plots) in the PCA space, with point colors reflecting their quality of representation ($\mathbf{cos^2}$), and exports the plots in PNG and SVG formats for further interpretation.

```
# Plot individuals with cos2 coloring
NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Indi
png(file = NOMpng, width = 700, height = 700)
p=fviz_pca_ind(res_pca, repel=TRUE, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800",
 ↪  "#FC4E07"),labelsize=5,pointsize=4,ggrepel = TRUE, geom = c("text","point"))+
  ylim (-7.5, 7.5)  + xlim(-10,10) +
  theme(text = element_text(size = 17),
        axis.title = element_text(size = 17),
        axis.text = element_text(size = 15))
```

```
print(p)

dev.off()


NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Indi
png(file = NOMpng, width = 700, height = 700)
p=plot(res_pca, choix = "ind",ylim=c(-7.5,7.5),xlim=c(-10,10))
print(p)
dev.off()

NOMsvg=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Indi
 ↪  #svg format
svg(file = NOMsvg)
print(p)
dev.off()
```

2.9 Variable metrics

This part of the script extracts the PCA metrics for the variables (e.g., $\cos^2$, contributions, correlations) from the PCA results object for further analysis and visualization.

```
# Get results for variables metrics (cos2, contribution, etc.)
var=get_pca_var(res_pca) # for both quantitative and qualitative variables
```

2.10 Variable quality

This part of the script :

- extracts the $\cos^2$ values (quality of representation) for each variable across PCA dimensions (**Var_quality**). It then exports these values and visualizes them.
- computes cumulative quality over the first four dimensions (**total_quality_df**) to identify the best-represented variables.

```
#All variables quality to dimensions
Var_quality=as.data.frame(var$cos2)
FILE4=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllVa
write.table(Var_quality,FILE4,sep = ";", dec = ",", row.names = FALSE)

NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Corr
png(file = NOMpng, width = 700, height = 700)
corrplot(na.omit(var$cos2), is.corr=FALSE,tl.cex=1.4,tl.col = "black",cl.cex=1.4,cl.align.text="l")
 ↪  # show the link between variables , representation quality and correlations with variables and
 ↪  dimensions
dev.off()


# Compute cumulative Quality of the representation of each variable to the first 4 PCA dimensions
total_quality=rowSums(var$cos2[, 1:4])
total_quality_df <- data.frame(Variable = names(total_quality),
                               Cumulative_Quality = total_quality)
FILE4bis=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_Al
write.table(total_quality_df,FILE4bis,sep = ";", dec = ",", row.names = FALSE)
```

```
print(total_quality_df)
```

2.11 Variable contribution

This part of the script :

- extracts the contribution values for each variable across PCA dimensions (**Var_contrib**). It then exports these values and visualizes them.
- computes cumulative contribution over the first four dimensions (**total_contribution_df**) to identify the most influential variables overall.

```
#All variables Contributions to dimensions
Var_contrib=as.data.frame(var$contrib)
Var_contrib$Variable <- rownames(Var_contrib)
FILE5=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllVa
write.table(Var_contrib,FILE5,sep = ";", dec = ",", row.names = FALSE)

for (dim in seq(1:4)){

 ↪  NOMpng=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_
  png(file = NOMpng, width = 500, height = 400) # doted line show average value if contributions
   ↪  were uniform
  p=fviz_contrib(res_pca, "var", axes = dim)+ # Contribution to 1st dimension (DIM1)
    theme(text = element_text(size = 17),
          axis.title = element_text(size = 17),
          axis.text = element_text(size = 10))
  print(p)
  dev.off()
}


# Compute cumulative contribution of each variable to the first 4 PCA dimensions
total_contrib <- rowSums(res_pca$var$contrib[, 1:4])
total_contrib_df <- data.frame(Variable = names(total_contrib),
                               Cumulative_Contribution = total_contrib)
FILE6=paste0(save_learning_primary_path,"/","PCA_",District,"_",Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_AllVa
write.table(total_contrib_df,FILE6,sep = ";", dec = ",", row.names = FALSE)


print(total_contrib_df)
```

2.12 Variable selection

This script performs a refined variable selection based on PCA results and correlation analysis:

- It first preselects variables with a **high quality of representation** (cumulative $\cos^2 > 0.5$) and those with a **contribution above a set threshold** (i.e. 100/total number of variables), identifying variables that are both well represented and contribute meaningfully across the first four principal components.
- It computes the **Pearson correlation matrix** among these preselected variables to assess redundancy.
- For pairs of variables with high correlation ($|r| > 0.9$), it removes one variable to reduce multicollinearity. The choice of which variable to remove is based primarily on their cumulative contribution to the PCA

axes, keeping the variable with the higher contribution. If contributions are equal, it uses the quality of representation to decide.

The final selected variables are those preselected by quality and contribution, minus those removed due to high correlation. This ensures the final set of predictors is informative, non-redundant, and optimally representative for further modeling steps.

```
# Preselect variables with quality above threshold
preselected_vars_with_high_quality <- total_quality_df$Variable[total_quality_df$Cumulative_Quality
 ↪  > 0.5]

# Preselect variables with contribution above threshold
print("Selecting variables above contribution threshold")
threshold=100 / ncol(tab_norm) # Define contribution threshold (100/number of variables)
preselected_vars_with_high_contribution<-
 ↪  total_contrib_df$Variable[total_contrib_df$Cumulative_Contribution > threshold] # fibd the
 ↪  mostcontributive variables on the 4 PC

# Get variables that are both high quality and high contribution
selected_vars <- intersect(preselected_vars_with_high_quality,
 ↪  preselected_vars_with_high_contribution)


# Compute correlation matrix among selected variables
cor_matrix <- cor(tab_norm[, selected_vars])

NOMpng=paste0(save_learning_primary_path,"/","Correlation_matrix_",District,"_",Island,"_",Satellite1,"_",Year1,"_A
png(file = NOMpng, width = 500, height = 400)
p=corrplot(correlation_matrix, method = "color", type = "upper",
           tl.cex = 0.8, tl.col = "black", addCoef.col = "black",
           number.cex = 0.7, diag = FALSE)
print(p)
dev.off()

NOMsvg=paste0(save_learning_primary_path,"/","Correlation_matrix_",District,"_",Island,"_",Satellite1,"_",Year1,"_A
svg(file = NOMsvg)
print(p)
dev.off()

# Identify highly correlated pairs (|r| > 0.9)
highly_corr_pairs <- which(abs(cor_matrix) > 0.9 & upper.tri(cor_matrix), arr.ind = TRUE)

# Remove redundancy by keeping only one variable per highly correlated pair
vars_to_remove <- c()

for (k in seq_len(nrow(highly_corr_pairs))) {
  i <- rownames(cor_matrix)[highly_corr_pairs[k, 1]]
  j <- colnames(cor_matrix)[highly_corr_pairs[k, 2]]

  # Get contribution values
  contrib_i <- total_contrib_df$Cumulative_Contribution[total_contrib_df$Variable == i]
  contrib_j <- total_contrib_df$Cumulative_Contribution[total_contrib_df$Variable == j]

  # If equal, compare quality
  if (contrib_i == contrib_j) {
    qual_i <- total_quality_df$Cumulative_Quality[total_quality_df$Variable == i]
    qual_j <- total_quality_df$Cumulative_Quality[total_quality_df$Variable == j]

    if (qual_i < qual_j) {
```

```
      vars_to_remove <- c(vars_to_remove, i)
    } else {
      vars_to_remove <- c(vars_to_remove, j)
    }
  } else if (contrib_i < contrib_j) {
    vars_to_remove <- c(vars_to_remove, i)
  } else {
    vars_to_remove <- c(vars_to_remove, j)
  }
}

# Final selection (remove duplicates in case of overlap)
final_selected_vars <- setdiff(selected_vars, unique(vars_to_remove))

# Print results
cat("Initial selected variables:", length(selected_vars), "\n")
cat("Removed due to correlation:", length(unique(vars_to_remove)), "\n")
cat("Final selected variables:", length(final_selected_vars), "\n")
print(final_selected_vars)
```

2.13 Final learning dataset

This section generates the final learning dataset using only the selected, non-redundant predictors:

- It filters the original dataset to retain only the variables selected after PCA-based evaluation and correlation filtering (**learning_data_selected**)
- A second version of the dataset is also created by excluding observations from the "PHOTO-INTERPRETATION" source, retaining only the "true" field- or sensor-based learning plots, and saved separately (**true_learning_plots_selected**).

These outputs ensure that both a complete and a source-filtered version of the optimized training data are available for downstream modeling.

```
# Create final dataset with the selected variables
learning_data_selected <- learning_data[, !(colnames(learning_data) %in% vars_to_remove)]
FILE7=paste0(save_learning_primary_path,"/Selected_learning_plots_", District, "_",
 ↪  Island,"_",Satellite1,"_",Year1,"_ALL_SOURCES_EPSG32739.csv")
write.table(learning_data_selected ,file =FILE7, sep = ";", dec = ".", row.names = FALSE)

# Save filtered dataset
true_learning_plots_selected=subset(learning_data_selected,learning_data_selected$Source!="PHOTO-INTERPRETATION")
FILE8=paste0(save_learning_primary_path,"/Selected_learning_plots_", District, "_",
 ↪  Island,"_",Satellite1,"_",Year1,"_TRUE_SOURCES_EPSG32739.csv")
write.table(true_learning_plots_selected,file =FILE8, sep = ";", dec = ".", row.names = FALSE)
```