Rapport de Projet

par Kévin Des Courières, Axel Kuehn et Florian Lock-Fat

Présentation du projet :

Voici notre rapport de projet sur les données météo. Nous avons travaillé à trois sur ce projet. Vous pouvez accèder aux codes sources du projet en R ainsi que le document RMarkdown et même ce rapport en PDF à l'adresse suivante : https://github.com/lockfatf/AnalyseDonneesMeteo.

I. Préparations des données :

I.1. Importation des librairies nécéssaires :

```
pacman::p_load(data.table,tidyverse,reshape2,FactoMineR,doParallel,Matrix,
               factoextra,rgl,caret,missMDA,dplyr,ggplot2,pls,glmnet,cowplot)
```

I.2. Imporation des données :

Commencons par importer les jeux de données mis à notre disposition.

```
# Chargement des donnees avec fread, plus intelligent que read.csv
# Climat
climat <- fread("Données/climat.201708.csv",data.table = F)
# Villes
villes <- fread("Données/postesSynop.csv",data.table=F)
```

Avant de faire une jointure des 2 jeux de données, Regardons les plus en détails avec un `summary()`.

```
# Summary
summary(climat)
summary(villes)
```

```
# Merge des 2 jeux de données
climat <- merge(climat,villes,by="NUM_POSTE",all.x = T)
# Une ville avec aucune correspondance
climat$Nom[is.na(climat$Nom)] <- "NOM INCONNU"
# Renomage de l'indice par la variable "Nom""
row.names(climat) <- climat$Nom
# Elimination des variables "Nom" et "NUM_POSTE"
climat <- climat[,-c(1,55)]
# Dimension de notre jeu de données
dim(climat)
```

```
## [1] 58 56
```

Suite à la jointure des 2 jeux de données, nous avons un jeu de données de 58 observations et 56 variables.

I.3. Nettoyage des données :

Premièrement, nous cherchons à isoler les variables avec plus de 10% de valeurs manquantes. Nous isolons ausi les variales avec une variance trop faible en prévention de la futur ACP.

```
# Recuperer variables avec plus de 10% de NA pour les enlever
idx_bad_col <- which(apply(climat,2,function(x){sum(is.na(x))})> (dim(climat)[1]/10))
# et les variables avec variance trop faible, probleme pour l'acp sinon
near_zero_var <- nzv(climat, freqCut = 95/5)
# et les enlever
climat_rm_na <- climat[,-c(idx_bad_col,near_zero_var)]
# Sélection des variables numériques pour PCA
climat_rm_na <- climat_rm_na %>% select_if(is_numeric)
# Dimension de notre jeu de données
dim(climat_rm_na)
```

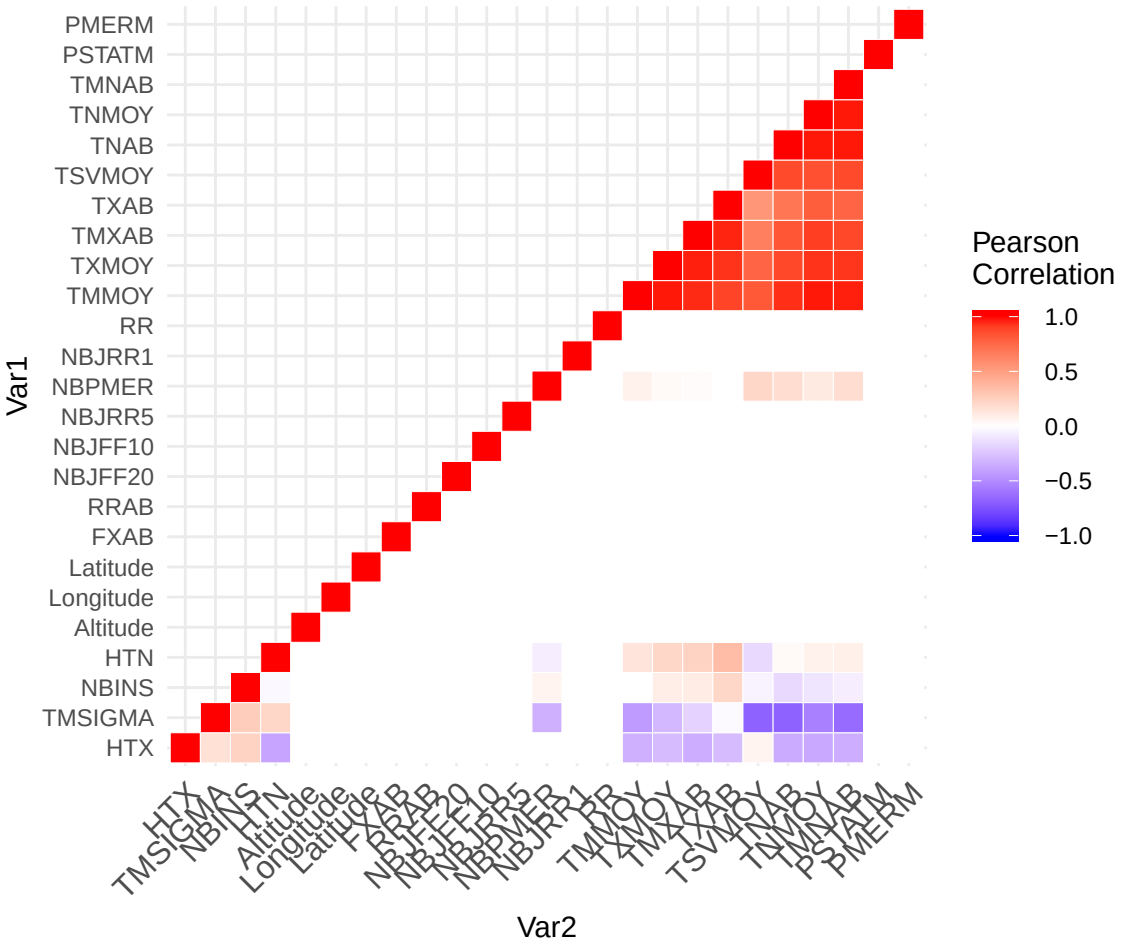
```
## [1] 58 25
```

Nous obtenons un jeu de données nettoyé avec 58 observations et 25 variables.

I.4. Matrice de corélation :

Avec ce dernier jeu de données obtenu, nous allons regarder la matrice de corrélation :

```
# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal() + # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1)) +
  coord_fixed()
# Print the heatmap
ggheatmap
```



La matrice de corrélation révèle de nombreuses corrélations entre nos variables que l'ont peut repérer et éliminer grâce à la fonction `findCorrelation()`:

```
# Annotation des variables fortement corrélées
idx_corVar = findCorrelation(cormat, cutoff = 0.5, verbose = FALSE, names = FALSE, exact = T)
```

On cherche maintenant à imputer les valeurs manquantes de notre jeu de données. Pour cela nous avons décidé d'utiliser le package `missMDA` qui a été spécifiquement implémenté pour imputer les données manquante en vue d'effectuer une analyse par ACP. Il nous faut tout d'abord estimer le nombre de composantes requises par la fonction `imputePCA` qui est la fonction permettan d'imputer ensuite les données par ACP itérative, avec le nombre de composantes optimal identifié par la fonction `estim_ncpPCA`.

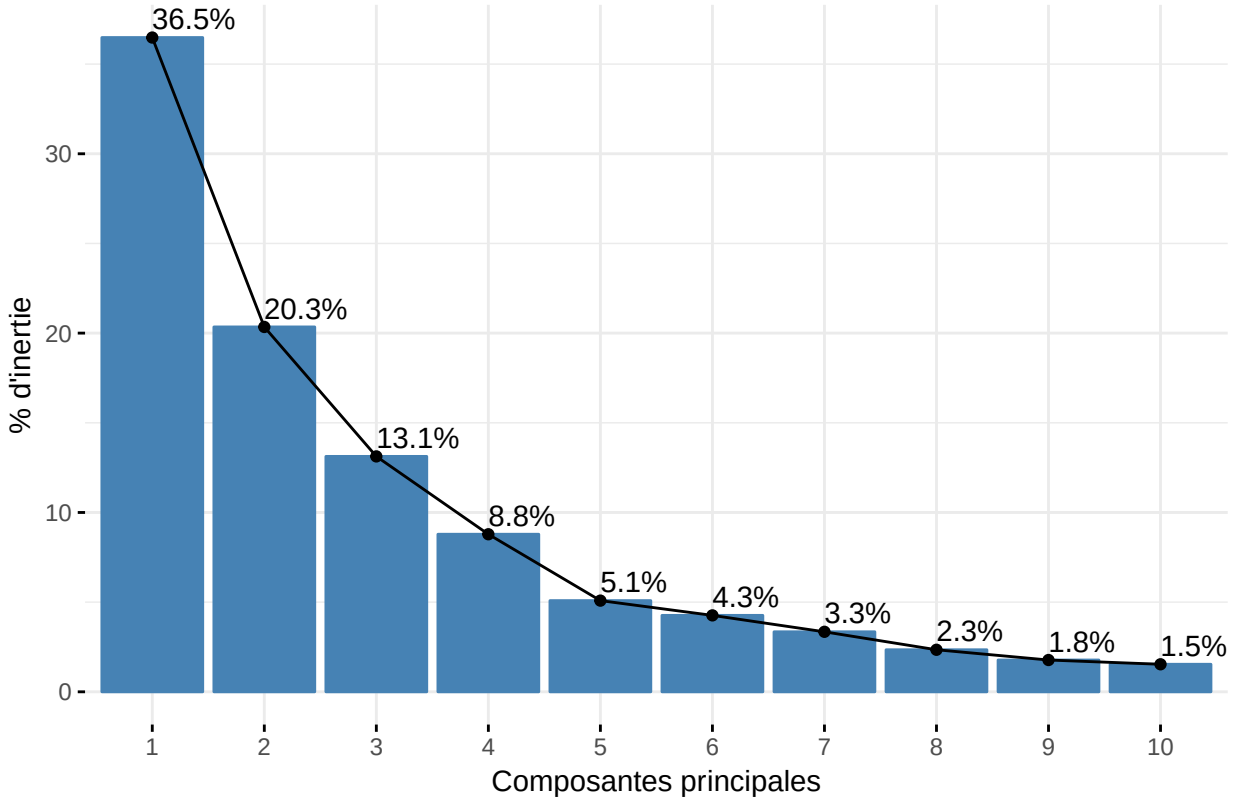
```
# Elimination des variables annotées
preproc <- climat_rm_na[,-idx_corVar]
# Estimation du nombre de composantes requises
set.seed(123)
nb <- estim_ncpPCA(preproc,scale=T)
preproc_impute <- imputePCA(preproc,ncp=nb$ncp,scale=T)$completeObs
```

II. Exploration :

II.1. Premier ACP :

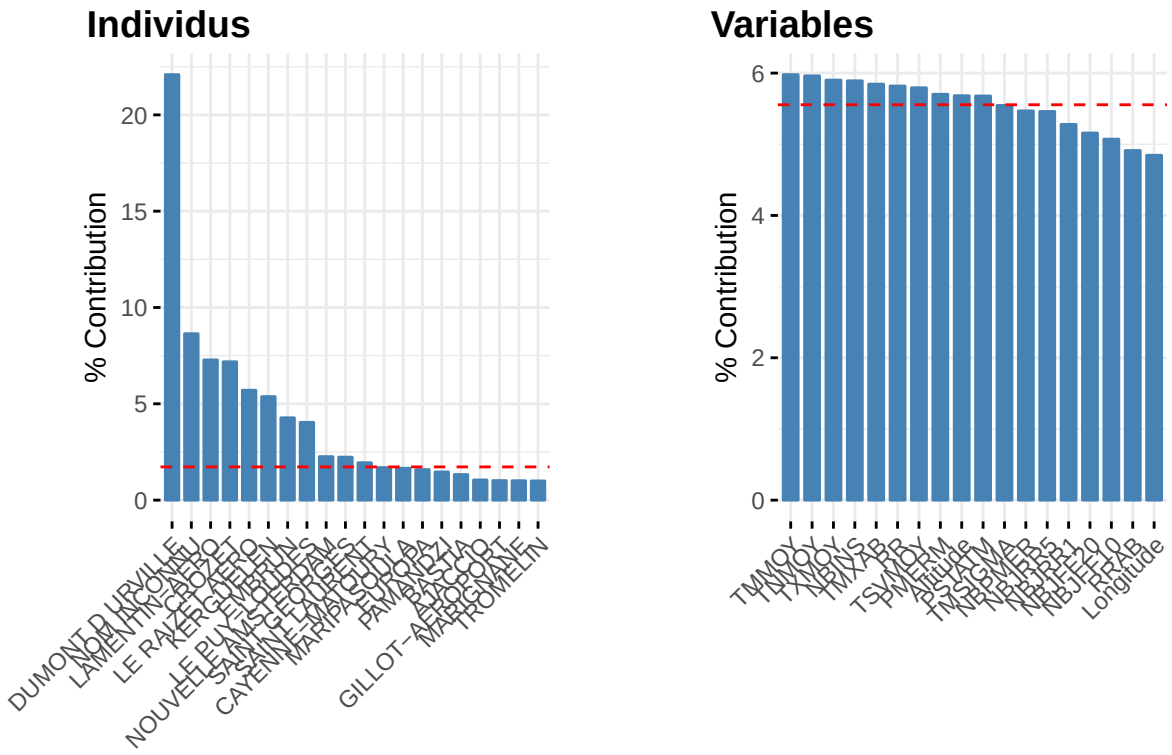
Dans un premier temps, nous effectuons une PCA exploratoire afin de découvrir les éventuels leviers que pourraient présenter certains individus et/ou variables.

```
# 1ère PCA
res_pca1 <- PCA(preproc_impute, ncp = dim(preproc_impute)[2], scale.unit = TRUE, graph=F)
#Affichage des valeurs propres pour sélection du nombre de composantes
fviz_eig(res_pca1, addlabels = TRUE) + ggtitle("") + xlab("Composantes principales") + ylab("% d'inerti
```



Nous récupérons 90% de la variance à partir de 7 composantes.

```
# Diagramme des contributions des individus
ind_plot = fviz_contrib(res_pca1,axes=c(1:7),choice="ind", top = 20) + ggtitle("") + ylab("% Contributi
# Diagramme des contributions des variables
var_plot = fviz_contrib(res_pca1,axes=c(1:7),choice="var") + ggtitle("") + ylab("% Contribution") + coo
# Affichage des 2 diagrammes :
plot_grid(ind_plot, var_plot, labels=c("Individus", "Variables"), ncol = 2, nrow = 1, align = 'h')
```



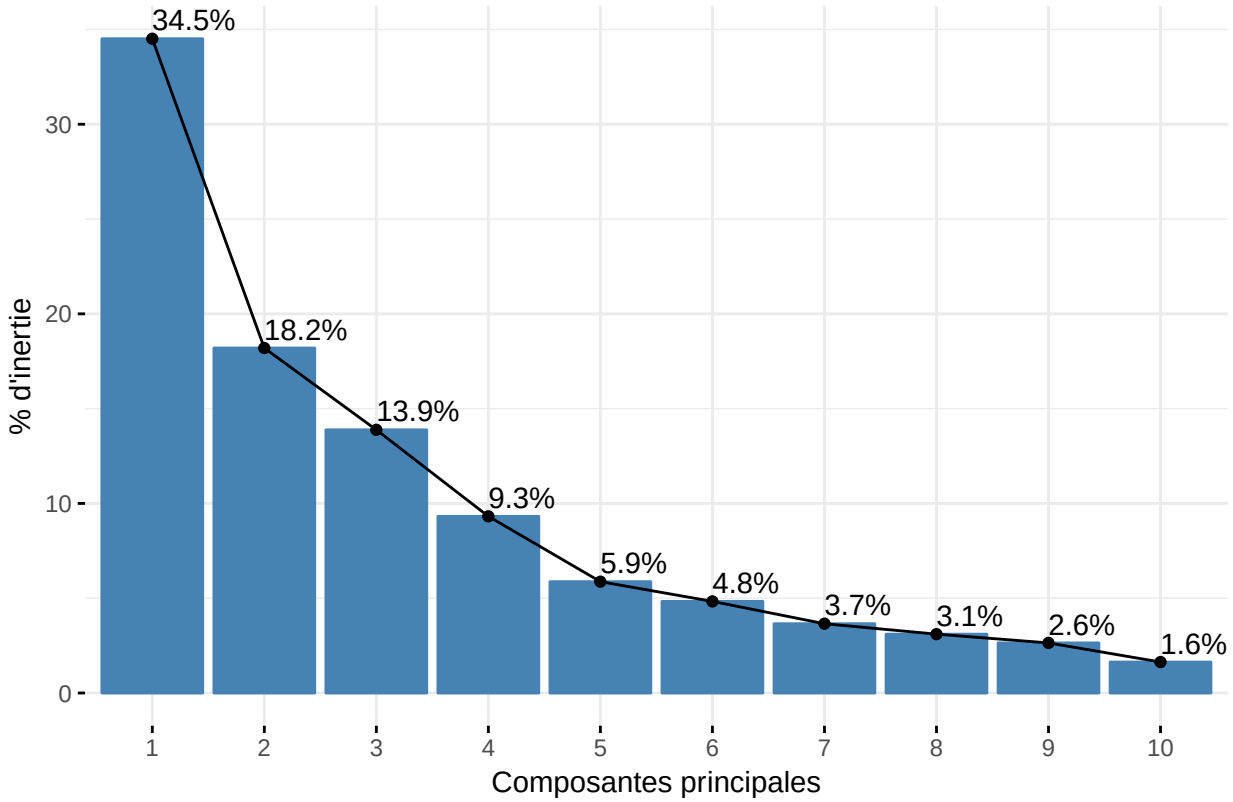
Sur ces 7 composante nous décidons d'ignorer les individus contribuant à plus de 5% pour la totalité de ces axes. Rien d'étonant, les stations qui contribuent fortement sont des stations hors France Métropolitaine, exemple avec Dumont d'Urville qui est une base en Antarctique.

```
# Marquage des individus contribuant fortement
contrib_ind <-fviz_contrib(res_pca1,axes=c(1:7),choice="ind")
ind_high_contrib <- rownames(contrib_ind$data[contrib_ind$data$contrib>5,])
```

Nous effectuons une nouvelle ACP en enlevant les indidus ciblés.

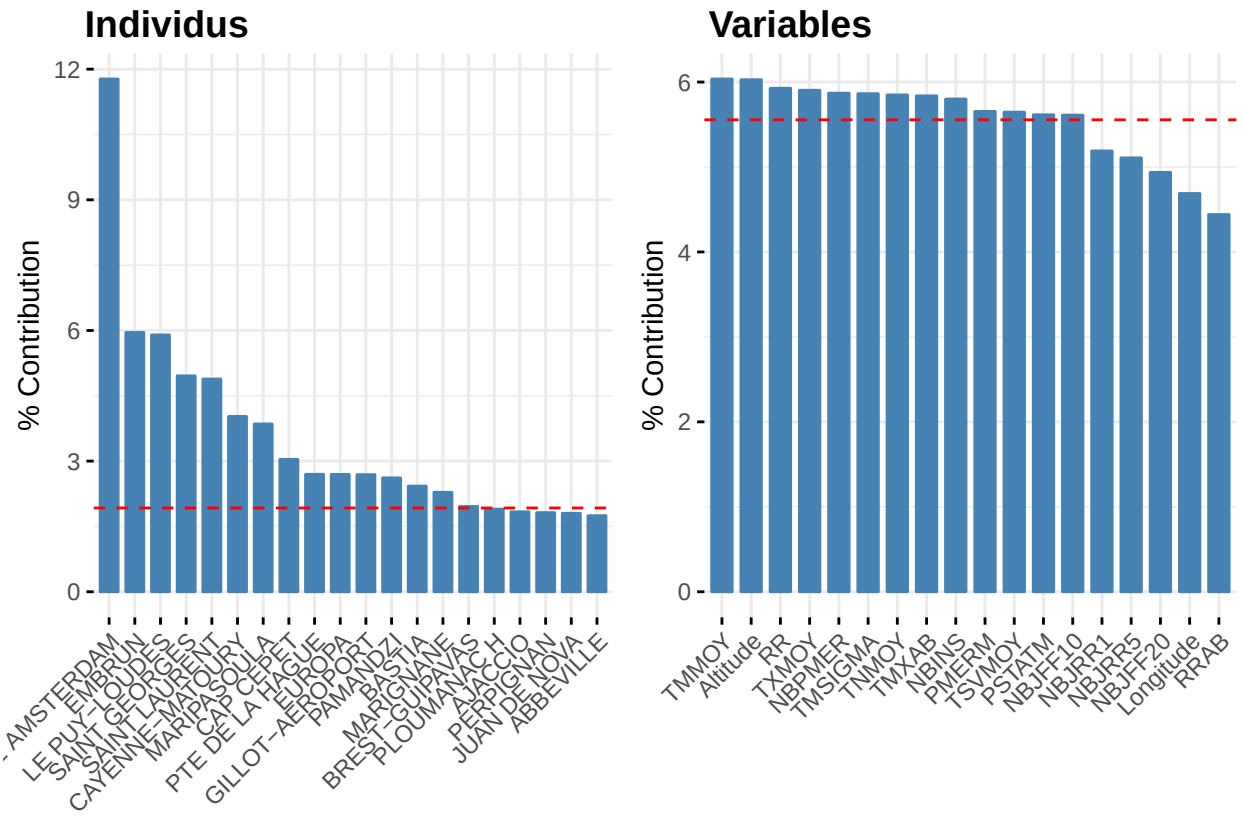
II.2. Deuxième ACP :

```
# 2e ACP moins les individus à forte contribution
res_pca2 <- PCA(preproc_impute, ncp = dim(preproc_impute)[2], scale.unit = TRUE, graph=F,ind.sup=which(i
#fviz_pca_biplot(res_pca2,axes=c(1,2),repel = T)
fviz_eig(res_pca2, addlabels = TRUE) + ggtitle("") + ylab("% d'inertie") + xlab("Composantes principales
```



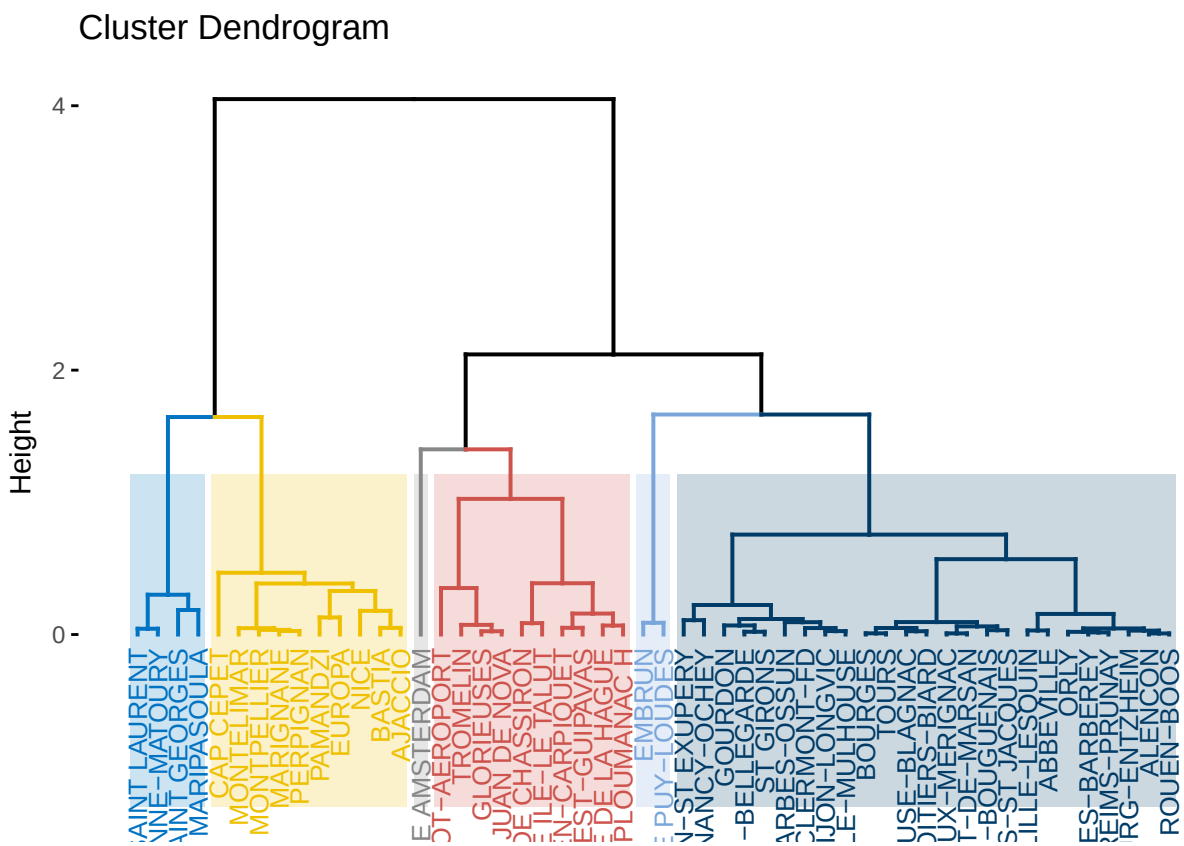
Nous obtenons le même résultat que la première ACP, c'est à dire, nous couvrons 90% à partir de la 7e composantes. Dans l'ensemble cela reste stable.

```
# Diagramme des contributions des individus
ind_plot2 = fviz_contrib(res_pca2,axes=c(1:7),choice="ind", top = 20) + ggtitle("") + ylab("% Contributi
# Diagramme des contributions des variables
var_plot2 = fviz_contrib(res_pca2,axes=c(1:7),choice="var") + ggtitle("") + ylab("% Contribution")
# Affichage des 2 diagrammes :
plot_grid(ind_plot2, var_plot2, labels=c("Individus", "Variables"), ncol = 2, nrow = 1, align = 'h')
```

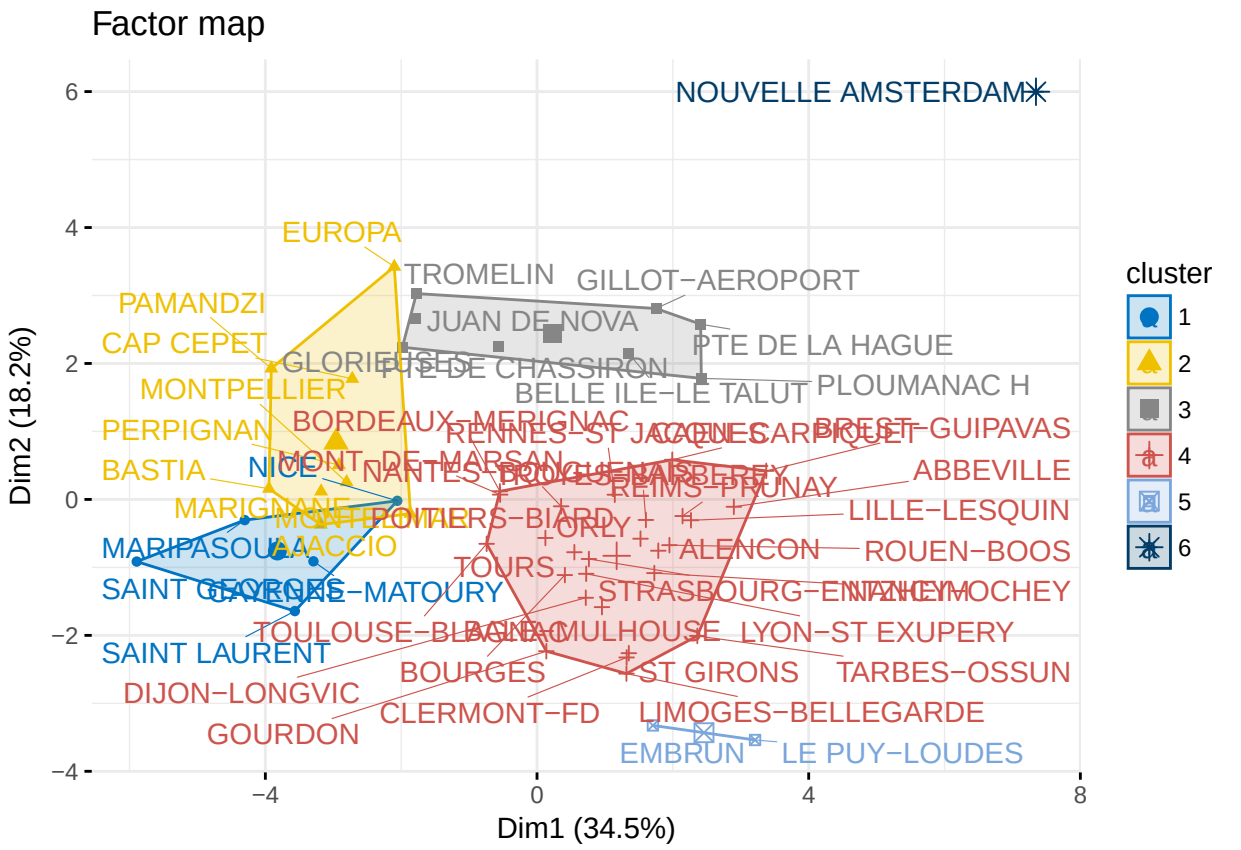


Nous décidons de ne plus éliminer d'individus.

```
# HCPC pour obtenir clusters à partir la PCA
res.hcpc <- HCPC(res_pca2, graph = FALSE)
fviz_dend(res.hcpc,
          cex = 0.7,                     # Taille du text
          palette = "jco",               # Palette de couleur ?ggpubr::ggpar
          rect = TRUE, rect_fill = TRUE, # Rectangle autour des groupes
          rect_border = "jco",           # Couleur du rectangle
          labels_track_height = 0.8      # Augment l'espace pour le texte
)
```




```
fviz_cluster(res.hcpc,
             repel = TRUE,            # Evite le chevauchement des textes
             show.clust.cent = TRUE, # Montre le centre des clusters
             palette = "jco",         # Palette de couleurs, voir ?ggpubr::ggpar
             ggtheme = theme_minimal(),
             main = "Factor map"
)
```



On observe 2 clusters principaux, un Centre-Nord France et un Sud France. Les individus contribuant fortement se retrouvent dans des clusters à part, notamment le cluster Océan Indien (Réunion + Mayotte).

III. Prédiction par PCR, PLSR et RandomForest

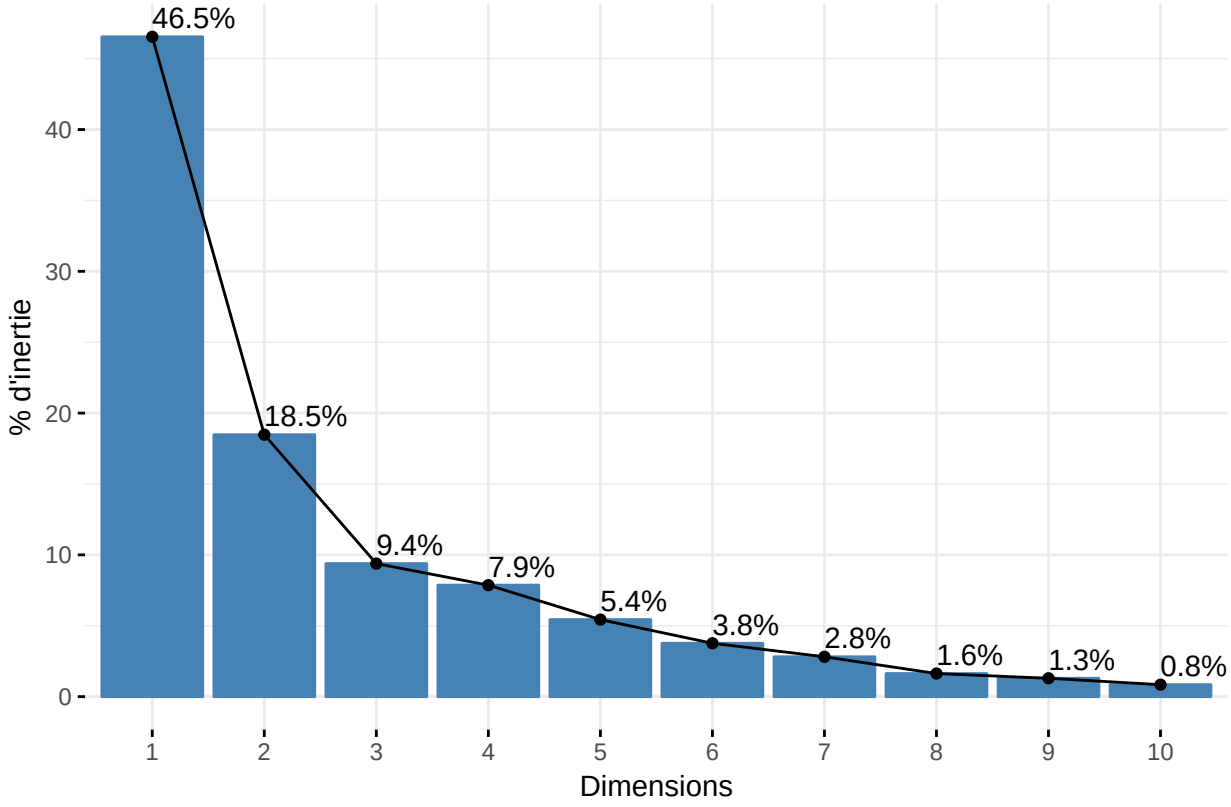
Tout d'abord, il conviendra de suivre le même cheminement que pour l'analyse exploratoire, toutefois en se restreignant au jeu d'apprentissage .

```
# Re-chargement des données
climat <- fread("Données/climat.201708.csv",data.table = F)
villes <- fread("Données/postesSynop.csv",data.table=F)
climat <- merge(climat,villes,by="NUM_POSTE",all.x = T)
climat$Nom[is.na(climat$Nom)] <- "Nom Inconnu"
row.names(climat) <- climat$Nom
climat <- climat[-which(is.na(climat$FXAB)),]
# Récupération des index et du nom des villes du jeu de test
idx_test <- c(7110,7149,7222,7481,7535,7591,7630,7747,7790)
test_villes <- rownames(climat)[climat$NUM_POSTE %in% idx_test]
# Création du jeu d'apprentissage et du jeu de test
training <- climat[-which(climat$NUM_POSTE %in% idx_test),-which(colnames(climat) %in% c("NUM_POSTE","F
test <- climat[climat$NUM_POSTE %in% idx_test,-which(colnames(climat) %in% c("NUM_POSTE","FXAB"))]
```

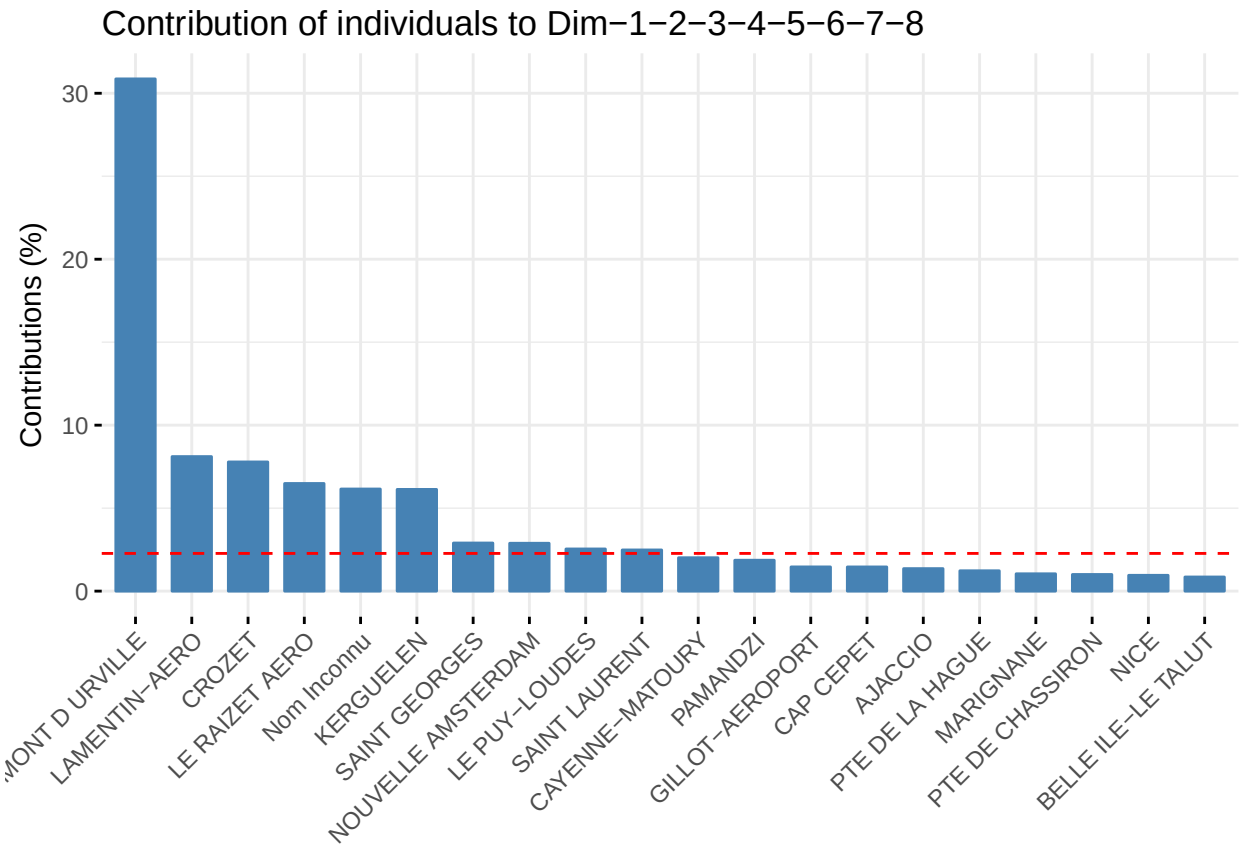
```
# Sélection des variables numérique
training <- training %>% select_if(is.numeric)
# Elimination des variables contenant plus de 5% de NA
idx_bad_col <- which(apply(training,2,function(x){sum(is.na(x))})> (dim(training)[1]/20))
```

```r
training <- training[,-idx_bad_col]
# Création d'un modèle de preprocessing: imputation des NA et élimination des variables à très faible v
set.seed(123)
training_bagImpute <- preProcess(training,method=c("bagImpute","nzv"))
training <- predict(training_bagImpute,training)
test <- test[,which(colnames(test) %in% colnames(training))]
test <- predict(training_bagImpute,test)
```

```r
# On effectue une 1ère ACP sur notre jeu d'apprentissage nettoyé
res_pca_explo <- PCA(training, ncp = dim(training)[2], scale.unit = TRUE, graph=F)
fviz_eig(res_pca_explo, addlabels = TRUE) + ggtitle("") + ylab("% d'inertie")
```

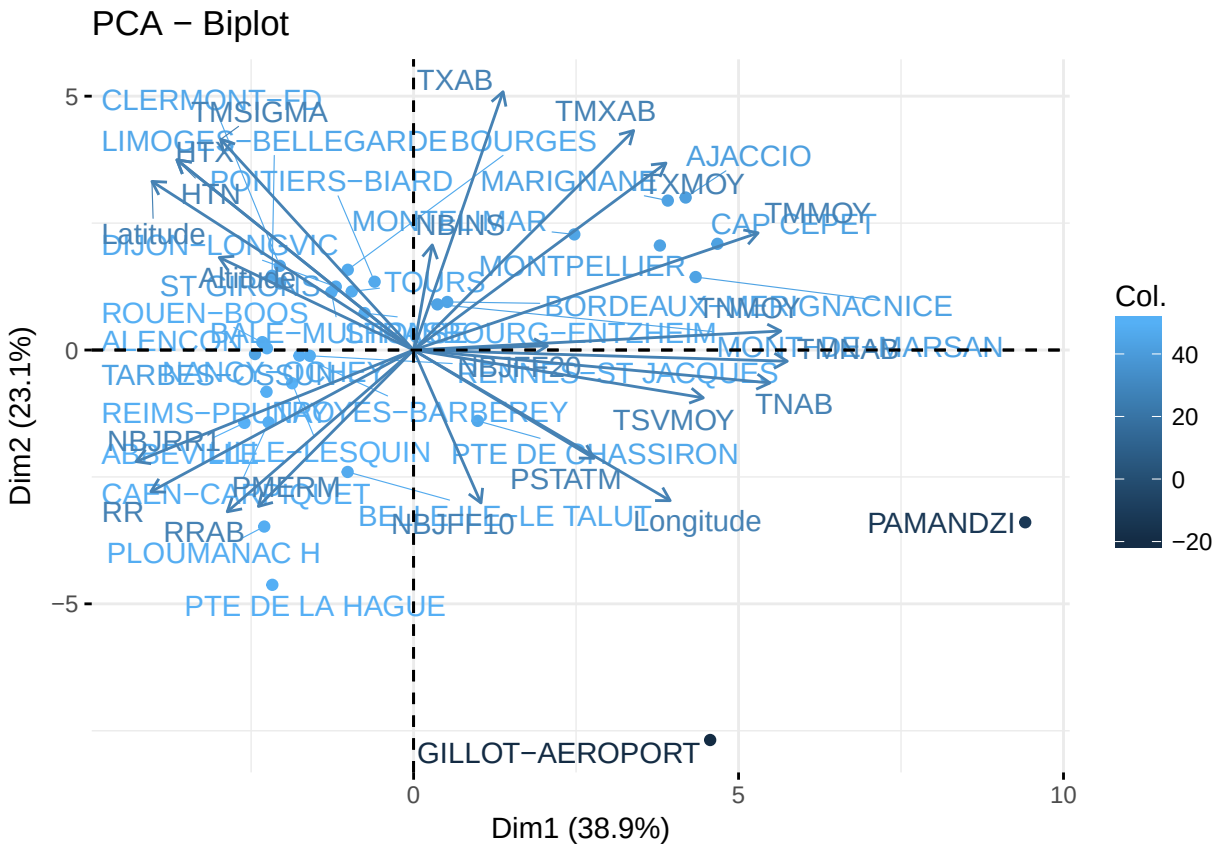



```r
# 8 composantes pour 95% de la variance
contrib_explo <- fviz_contrib(res_pca_explo,axes=c(1:8),choice="ind", top = 20)
contrib_explo
```

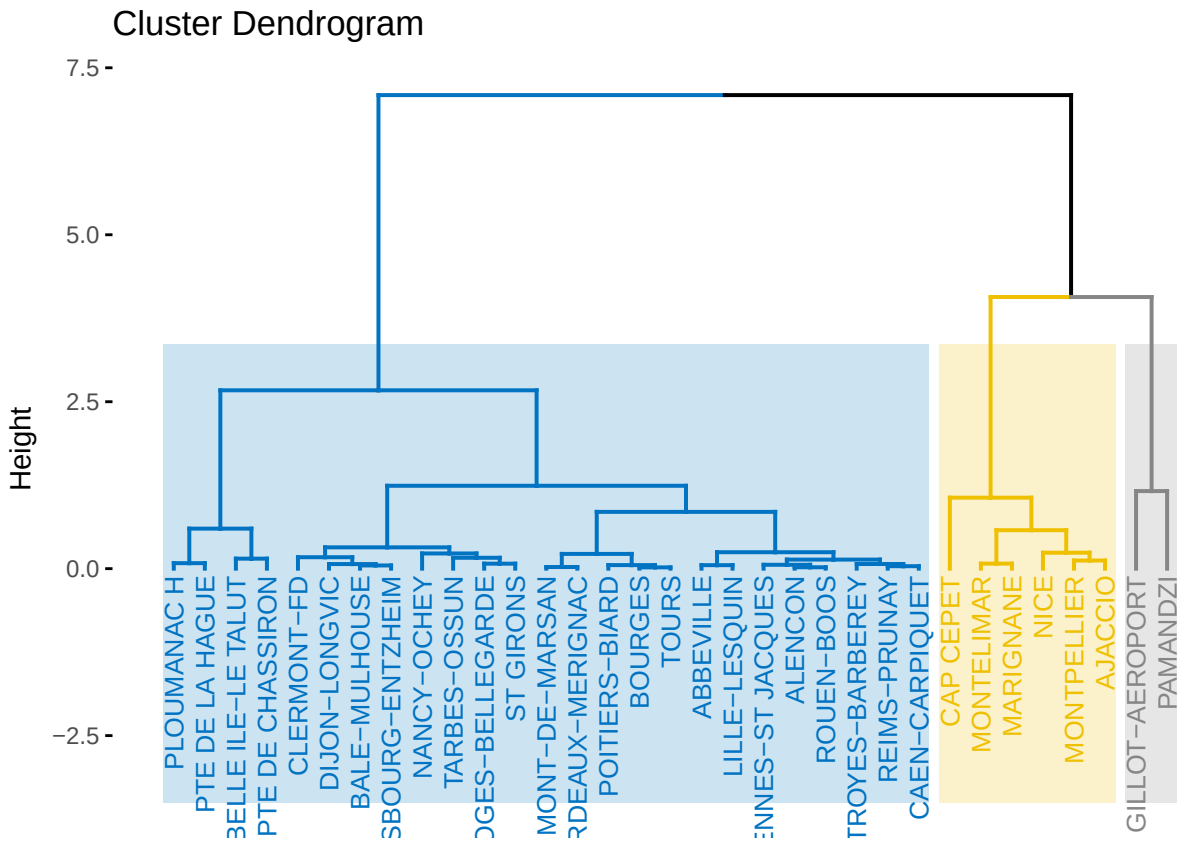



```
high_contrib_names <- as.character(contrib_explo$data$name[contrib_explo$data$contrib > 2])
```

```
# Création d'un jeu d'entrainement pour l'ACP ingorant les individus à forte contribution sur la totali
training_pca <- training[-which(rownames(training) %in% high_contrib_names),]
# ACP sur le jeu d'apprentissage
res_pca <- PCA(training_pca, ncp = dim(training_pca)[2], scale.unit = TRUE, graph=F)
fviz_pca_biplot(res_pca,axes=c(1,2),repel = T,col.ind = training_pca$Latitude)
```



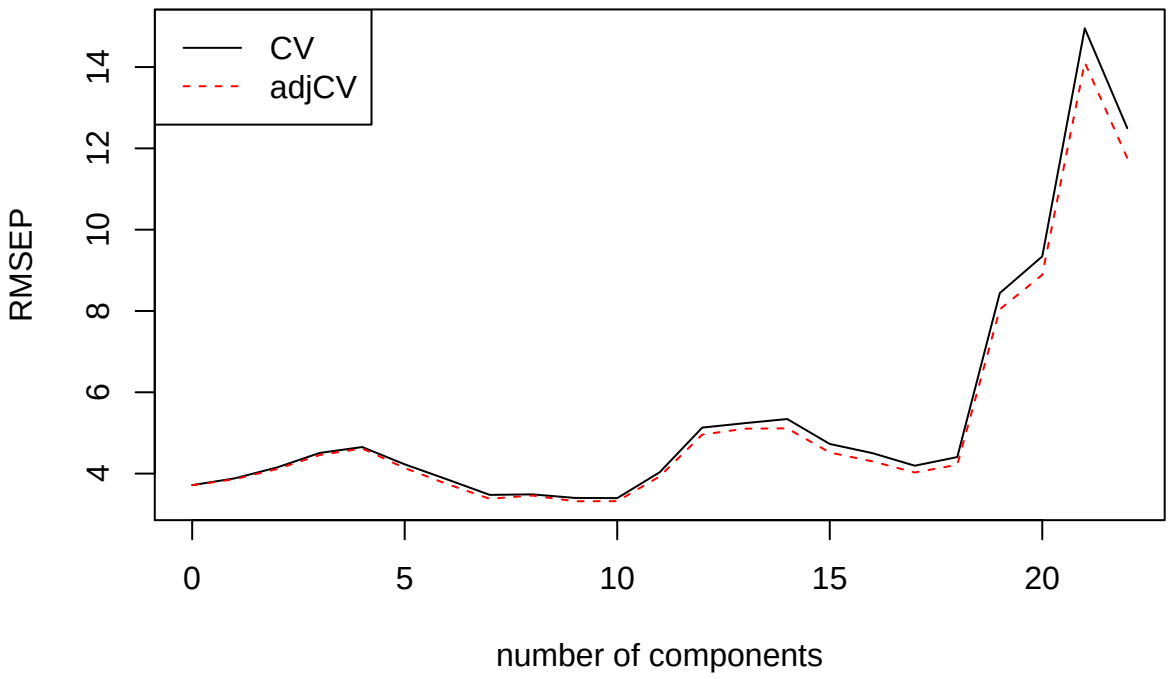

```
res_hcpc <- HCPC(res_pca, graph = F)
fviz_dend(res_hcpc,
          cex = 0.7,                     # Taille du text
          palette = "jco",               # Palette de couleur ?ggpubr::ggpar
          rect = TRUE, rect_fill = TRUE, # Rectangle autour des groupes
          rect_border = "jco",           # Couleur du rectangle
          labels_track_height = 3      # Augment l'espace pour le texte
)
```



L'ACP effectuée sur notre jeu de données d'apprentissage permet d'observer 3 clusters distincts. Un cluster Nord de la France, un cluster Sud de la France et un cluster Océan Indien.

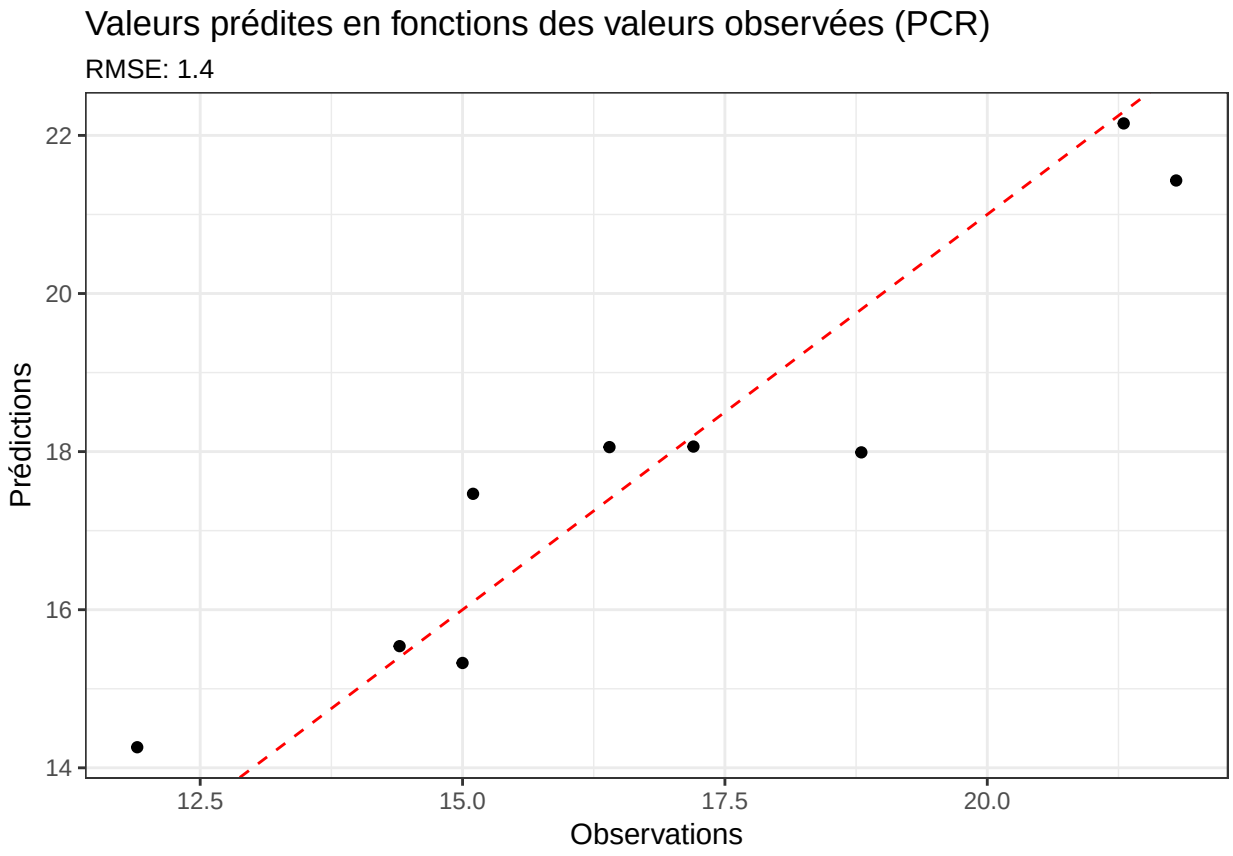
```
# Récupération de la matrice des composantes du jeu d'apprentissage
train_proj <- as.data.frame(res_pca$ind$coord)
# Standardisation du jeu de test avec les statistiques du jeu d'apprentissage
test_standard <- t(apply(test,1,function(x){(x-res_pca$call$centre)/res_pca$call$ecart.type} ))
# Projection du jeu de test standardisé sur les composantes identifiées par ACP sur le jeu d'apprentiss
test_proj <- as.data.frame(as.matrix(test_standard) %*% res_pca$svd$V)
# Récupération de la variable FXAB pour création du modèle et évaluation des prédictions sur le jeu de
train_proj$FXAB<- climat$FXAB[which(rownames(climat) %in% rownames(train_proj))]
test_proj$FXAB <- climat$FXAB[which(rownames(climat) %in% rownames(test_proj))]
# Formattage nécessaire
colnames(test_proj) <- colnames(train_proj)
```

```
set.seed(123)
pcr_fit <- pcr(FXAB ~ ., data = train_proj, scale = F, validation = "CV")
par(mfrow=c(1,1))
# Plot de la variation de la RMSE en fonction du nombre de composantes sélectionnées par le modèle PCR
plot(RMSEP(pcr_fit), legendpos = "topleft")
```

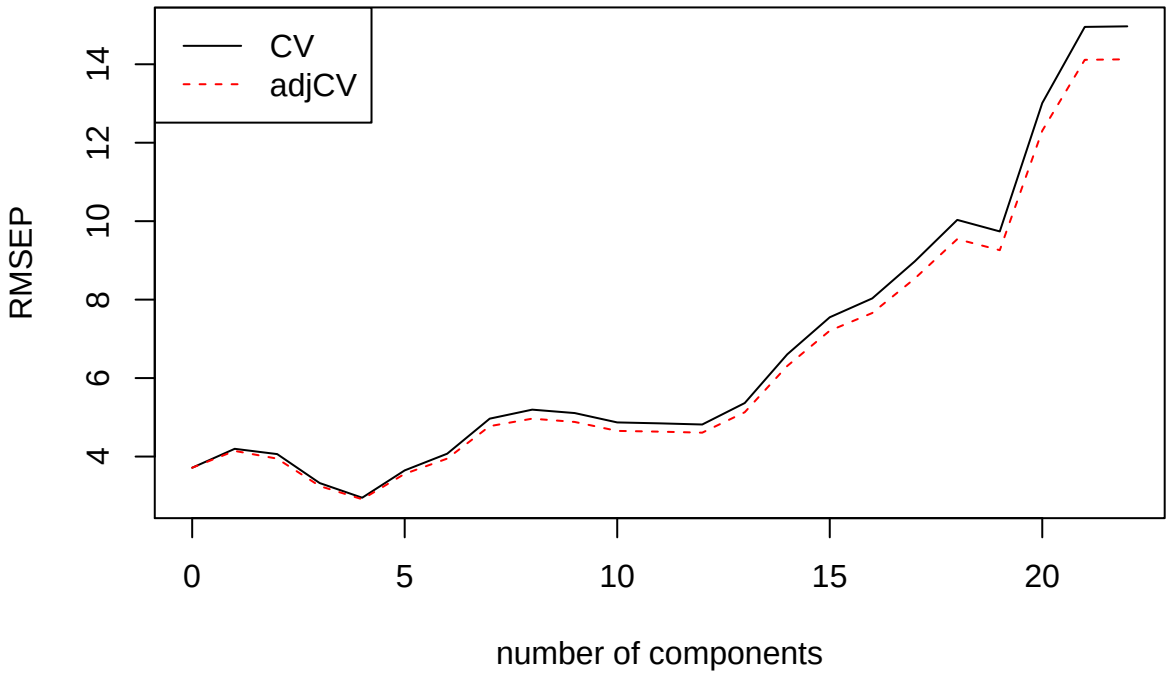



```
# Récupération du nombre optimal de composantes obtenu par cross-validation
ncomp.min <- which.min(sapply(1:dim(pcr_fit$validation$pred)[3],function(x){
  RMSE(pcr_fit$validation$pred[,,x],train_proj$FXAB)
}))
set.seed(123)
pcr_fit <- pcr(FXAB ~ ., data = train_proj,ncomp=ncomp.min, scale = F, validation = "CV")
```

```
pred_pcr <- predict(pcr_fit, ncp = ncomp.min, newdata = test_proj) %>% as.data.frame %>% pull()
rmse_pcr <- RMSE(pred_pcr,test_proj$FXAB)
as.data.frame(pred_pcr) %>% cbind(., measured = test_proj$FXAB) %>%
  ggplot(mapping = aes(x = measured, y = pred_pcr)) +
    geom_point() +
    geom_abline(mapping = aes(slope=1, intercept=1), color='red', linetype=2) +
    theme_bw() +
    labs(title = "Valeurs prédites en fonctions des valeurs observées (PCR)", subtitle = paste("RMSE:",
```

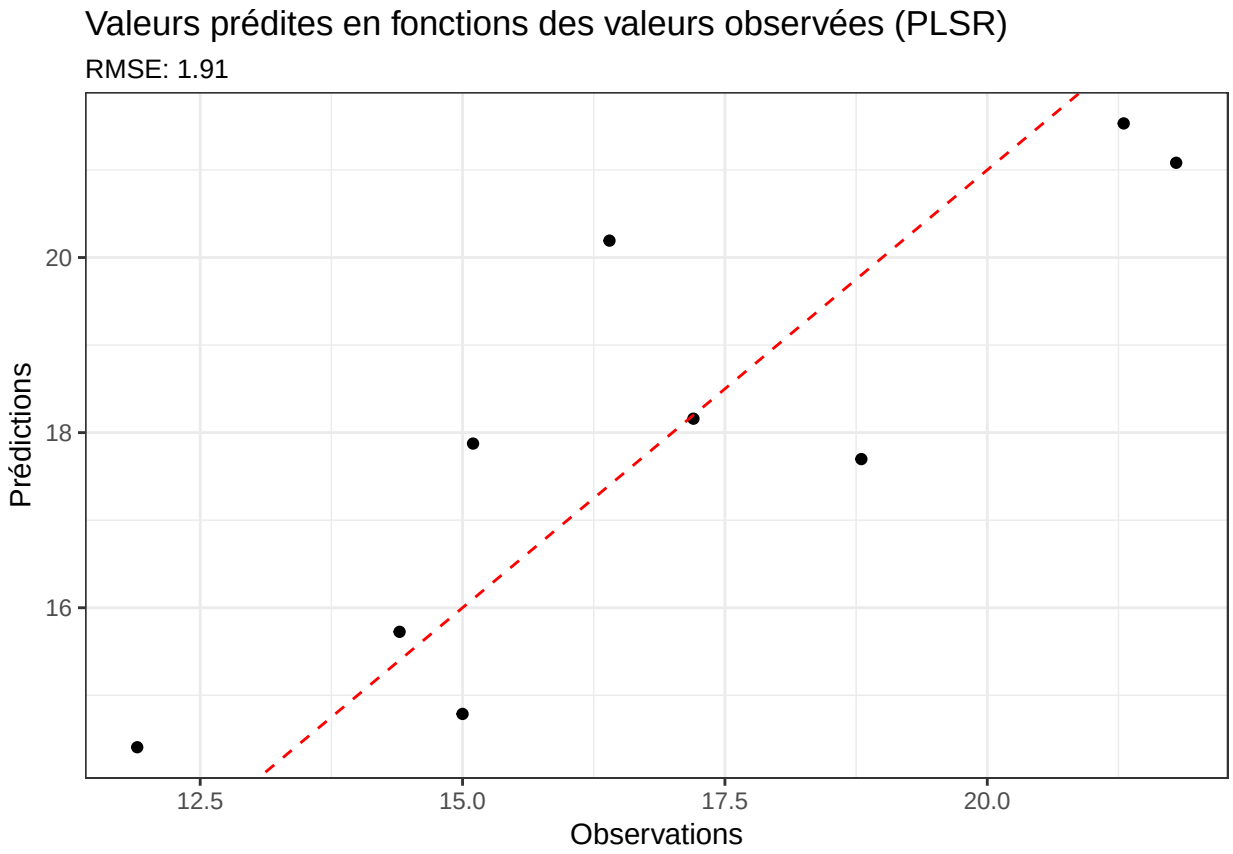



```r
set.seed(123)
plsr_fit <- plsr(FXAB ~ ., data = train_proj, scale = F, validation = "CV")
par(mfrow=c(1,1))
# Plot de la variation de la RMSE en fonction du nombre de composantes sélectionnées par le modèle PLSR
plot(RMSEP(plsr_fit), legendpos = "topleft")
```




```
# Récupération du nombre optimal de composantes obtenu par cross-validation
ncomp.min <- which.min(sapply(1:dim(plsr_fit$validation$pred)[3],function(x){
  RMSE(plsr_fit$validation$pred[,,x],train_proj$FXAB)
}))
set.seed(123)
plsr_fit <- plsr(FXAB ~ ., data = train_proj,ncomp=ncomp.min, scale = F, validation = "CV")
```

```
pred_plsr <- predict(plsr_fit, ncp = ncomp.min, newdata = test_proj) %>% as.data.frame %>% pull()
rmse_plsr <- RMSE(pred_plsr,test_proj$FXAB)
as.data.frame(pred_plsr) %>% cbind(., measured = test_proj$FXAB) %>%
  ggplot(mapping = aes(x = measured, y = pred_plsr)) +
    geom_point() +
    geom_abline(mapping = aes(slope=1, intercept=1), color='red', linetype=2) +
    theme_bw() +
    labs(title = "Valeurs prédites en fonctions des valeurs observées (PLSR)", subtitle = paste("RMSE:"
```




```
training_rf <- training[-which(rownames(training) %in% high_contrib_names),]
training_rf$FXAB <- climat$FXAB[which(rownames(climat) %in% rownames(training_rf))]
test_rf <- test
test_rf$FXAB <- climat$FXAB[which(rownames(climat) %in% rownames(test_rf))]
set.seed(123)
ctrl = trainControl(method = "repeatedcv",number=3,repeats=10)
mod_ranger <- train(FXAB~., data=training_rf, method="ranger")
mod_svm <- train(FXAB~., data=training_rf, method="svmLinear2")
mod_glmnet <- train(FXAB~., data=training_rf, method="glmnet")
RMSE(predict(mod_ranger,test_rf),test_rf$FXAB)
```

```
## [1] 1.531036
```

```
RMSE(predict(mod_svm,test_rf),test_rf$FXAB)
```

```
## [1] 2.027013
```

```
RMSE(predict(mod_glmnet,test_rf),test_rf$FXAB)
```

```
## [1] 2.165308
```