

Setup Complet R + TensorFlow + Keras + ClustR

Riad, Aya, Thibaud

2025-11-30

Introduction

Ce document présente le setup complet pour utiliser le package ClustR avec R 4.5.2 et Python 3.10, incluant TensorFlow et Keras. Trois méthodes de clustering de variables sont illustrées : Deep Learning, VARCLUS qualitative et K-means réallocatif.

Installation des packages R

```
install.packages(c(
  "devtools", "shiny", "R6", "ggplot2", "dplyr", "tidyr",
  "cluster", "factoextra", "plotly", "DT", "pheatmap",
  "reticulate", "keras3", "torch", "shinyWidgets"
))

devtools::install_github("riadshrn/ClustR")
```

Configuration Python 3.10 pour TensorFlow

```
library(reticulate)

# MODIFIER CE CHEMIN SI TON PYTHON 3.10 EST AILLEURS
python310 <- "C:/Users/thiba/AppData/Local/Programs/Python/Python310/python.exe"

# Création de l'environnement virtualenv r-tensorflow
if (!virtualenv_exists("r-tensorflow")) {
  virtualenv_create(
    envname = "r-tensorflow",
    python = python310
  )
}

use_virtualenv("r-tensorflow", required = TRUE)
```

Vérification de la configuration Python

```
print(py_config())
```

Installation des modules Python

```
virtualenv_install(  
  "r-tensorflow",  
  packages = c("tensorflow", "keras")  
)
```

Test de TensorFlow

```
library(tensorflow)  
library(keras3)  
  
keras3::use_backend("tensorflow")  
  
test <- try(tf$constant(1), silent = TRUE)  
print(test)
```

Chargement du package ClustR

```
library(shiny)  
library(ClustR)  
library(shinyWidgets)  
  
# Décommente cette ligne pour lancer automatiquement l'app :  
# shiny::runApp(system.file("shiny-app", package = "ClustR"))
```

Exemple 1 : Deep Learning (variables numériques)

Préparation des données

```
# Préparer les données (uniquement numériques)  
data_num <- iris[, sapply(iris, is.numeric)]  
head(data_num)  
  
##   Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1         5.1         3.5         1.4         0.2  
## 2         4.9         3.0         1.4         0.2
```

## 3	4.7	3.2	1.3	0.2
## 4	4.6	3.1	1.5	0.2
## 5	5.0	3.6	1.4	0.2
## 6	5.4	3.9	1.7	0.4

```
set.seed(42)
X <- matrix(rnorm(200 * 15), nrow = 200)
colnames(X) <- paste0("Var", 1:15)
```

Création et ajustement du modèle

```
# Créer le modèle Deep
model_deep <- VariableClustering$new(
  method      = "deep",
  n_clusters   = 3,
  latent_dim   = 5,
  hidden_layers = c(64, 32),
  epochs       = 50,
  dropout      = 0.1,
  l2_reg        = 1e-4,
  seed         = 42
)

# Ajuster le modèle
model_deep$fit(X)
```

Résultats du modèle Deep Learning

```
# Résumé et affichage
model_deep$print()
model_deep$summary()

# Récupérer les clusters
clusters_deep <- model_deep$get_clusters()
head(clusters_deep)

# Exporter en dataframe
df_deep <- model_deep$to_dataframe()
head(df_deep)
```

Visualisation

```
model_deep$plot()
```

Prédictions

```
# Prédire sur nouvelles données
pred_deep <- model_deep$predict(X)
head(pred_deep)

# Fit + predict en une fois
result_deep <- model_deep$fit_predict(X)
```

Exemple 2 : VARCLUS qualitative (variables catégorielles)

Préparation des données catégorielles

```
# Préparer des données catégorielles
iris_cat <- iris
iris_cat$Sepal.Length_cat <- cut(iris$Sepal.Length, breaks = 3,
                                labels = c("Petit", "Moyen", "Grand"))
iris_cat$Sepal.Width_cat <- cut(iris$Sepal.Width, breaks = 3,
                                labels = c("Etroit", "Moyen", "Large"))
iris_cat$Petal.Length_cat <- cut(iris$Petal.Length, breaks = 3,
                                labels = c("Court", "Moyen", "Long"))
iris_cat$Petal.Width_cat <- cut(iris$Petal.Width, breaks = 3,
                                labels = c("Fin", "Moyen", "Epais"))

# Sélectionner uniquement les variables catégorielles
iris_quali <- iris_cat[, c("Sepal.Length_cat", "Sepal.Width_cat",
                           "Petal.Length_cat", "Petal.Width_cat", "Species")]

head(iris_quali)
```

```
##   Sepal.Length_cat Sepal.Width_cat Petal.Length_cat Petal.Width_cat Species
## 1             Petit             Moyen             Court             Fin setosa
## 2             Petit             Moyen             Court             Fin setosa
## 3             Petit             Moyen             Court             Fin setosa
## 4             Petit             Moyen             Court             Fin setosa
## 5             Petit             Moyen             Court             Fin setosa
## 6             Petit             Large             Court             Fin setosa
```

Création et ajustement du modèle VARCLUS

```
# Créer le modèle VARCLUS Quali
model_quali <- VariableClustering$new(
  method      = "quali_varclus",
  n_clusters  = 3,      # ou NULL pour sélection automatique
  max_iter    = 30,
  seed        = 42
)
```

```
# Ajuster le modèle
model_quali$fit(iris_quali)
```

Résultats du modèle VARCLUS

```
# Résultats
model_quali$print()
model_quali$summary()

# Récupérer les clusters
clusters_quali <- model_quali$get_clusters()
print(clusters_quali)

# Exporter en dataframe
df_quali <- model_quali$to_dataframe()
print(df_quali)
```

Prédictions VARCLUS

```
# Prédictions
pred_quali <- model_quali$predict(iris_quali)
head(pred_quali)

# Fit + predict en une fois
result_quali <- model_quali$fit_predict(iris_quali)
```

Exemple 3 : K-means réallocatif (variables numériques)

Création et ajustement du modèle K-means

```
# Créer le modèle K-means
model_kmeans <- VariableClustering$new(
  method      = "kmeans",
  n_clusters  = 3,
  max_iter    = 100,
  tol         = 1e-6,
  scale       = TRUE,
  seed        = 42
)

# Ajuster le modèle
model_kmeans$fit(data_num)
```

Résultats du modèle K-means

```
# Résumé
model_kmeans$print()
model_kmeans$summary()

# Récupérer les assignations des clusters
clusters_kmeans <- model_kmeans$get_clusters()
print(clusters_kmeans)

# Exporter en dataframe
df_kmeans <- model_kmeans$to_dataframe()
head(df_kmeans)
```

Visualisation K-means

```
model_kmeans$plot()
```

Prédictions K-means

```
# Prédictions sur nouvelles données
pred_kmeans <- model_kmeans$predict(data_num)
head(pred_kmeans)

# Fit + predict en une fois
result_kmeans <- model_kmeans$fit_predict(data_num)
```

Comparaison des trois méthodes

```
cat("\n=== COMPARAISON DES MÉTHODES ===\n\n")

# Récupérer les paramètres de chaque modèle
params_deep <- model_deep$get_params()
params_quali <- model_quali$get_params()
params_kmeans <- model_kmeans$get_params()

cat("Deep Learning:\n")
print(params_deep)

cat("\nVARCLUS Quali:\n")
print(params_quali)

cat("\nK-means Réallocatif:\n")
print(params_kmeans)
```

Accès aux modèles sous-jacents

```
cat("\n=== MODÈLES INTERNES ===\n")
cat("Type du modèle Deep   :", class(model_deep$model)[1], "\n")
cat("Type du modèle Quali  :", class(model_quali$model)[1], "\n")
cat("Type du modèle K-means:", class(model_kmeans$model)[1], "\n")
```

Résumé des méthodes communes

Les méthodes suivantes sont disponibles pour toutes les approches de clustering :

- **initialize()** : Créer le modèle
- **fit(X)** : Ajuster le modèle
- **predict(X)** : Prédire sur nouvelles données
- **fit_predict(X)** : Fit + predict en une fois
- **get_clusters()** : Récupérer les assignations
- **summary()** : Résumé détaillé
- **print()** : Affichage simple du modèle

Conclusion

Ce document présente trois approches complémentaires pour le clustering de variables avec ClustR. Choisissez la méthode en fonction de vos données et objectifs :

- **Deep Learning** : pour des données complexes avec relations non-linéaires
- **VARCLUS qualitative** : pour des variables catégorielles
- **K-means réallocatif** : pour une approche simple et rapide sur données numériques