

CAF Single cell Transcriptomics Analysis

Mickael Coquerelle & Loïk Galtier

2025-12-08

Table des matières

1 Résumé	2
2 Préparation des données.	2
2.1 QC et filtrage des données	3
3 Normalisation avec SC Transform.	4
3.1 Le modèle mathématique derrière la normalisation SCTransform	5
3.2 Application de la méthode	5
4 Réduction de dimension, clustering et visualisation	5
4.1 Interprétation PCA\	5
4.2 Interprétation de l'ElbowPlot\	6
4.3 Interprétation TSNE \	7
4.4 Interprétation UMAP et clustering : \	8

```
# Setup global chunk options and libraries.
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE,
                      fig.width = 8, fig.height = 6)

# Tidyverse-like utilities
library(dplyr)
library(ggplot2)
# Seurat and single-cell specific tools
library(Seurat)
# Parallelization for Seurat (kept sequential by default in this report for reproducibility)
library(future)
# Annotation via Ensembl
library(AnnotationDbi)
library(EnsDb.Hsapiens.v86)
sessionInfo()

# Parallelisation:
plan(sequential)
options(future.globals.maxSize=10*1024^3) # 10 GB

# Resolve namespace conflicts (ensemblDb::filter masks dplyr::filter)
filter <- dplyr::filter
arrange <- dplyr::arrange
mutate <- dplyr::mutate
select <- dplyr::select
```

```

" data loading : we load the three datasets corresponding to three different patients' CAF single-cell

'p5 <- read.csv("~/Projets_GIT/Single_Cell_Project/Data/GSM4805570_CountsMatrix_20G00953M_TN.txt.gz",
sep="\t")
p4a <- read.csv("~/Projets_GIT/Single_Cell_Project/Data/GSM4805568_CountsMatrix_19G02977B_TN.txt.gz",
sep="\t")
p4b <- read.csv("~/Projets_GIT/Single_Cell_Project/Data/GSM4805566_CountsMatrix_19G02977A_TN.txt.gz",
sep="\t")'

p5 <- read.csv("~/Documents/git/Single_Cell_Project/Data/GSM4805570_CountsMatrix_20G00953M_TN.txt.gz"
sep="\t")
p4a <- read.csv("~/Documents/git/Single_Cell_Project/Data/GSM4805568_CountsMatrix_19G02977B_TN.txt.gz"
sep="\t")
p4b <- read.csv("~/Documents/git/Single_Cell_Project/Data/GSM4805566_CountsMatrix_19G02977A_TN.txt.gz"
sep="\t")

```

1 Résumé

L'objet de ce travail et de se familiariser avec l'analyse transcriptomique Single-Cell de CAF (Cancer Associated Fibroblasts) à partir d'une matrice de comptages issues des trois fichiers proposé. Notre pipeline d'analyse en se basant sur les éléments donnés en cours couvre le nettoyage (filtrage cellules/gènes), normalisation, réduction de dimension (PCA, t-SNE, UMPAP), clustering, identification de(s) signature(s) transcriptomique(s) pour marquer les clusters et enfin nous proposons une visualisation ciblée par heatmap et features plots.

2 Préparation des données.

We combine the three datasets and map Ensembl IDs to gene symbols and types. the objectif is to filter for protein-coding genes with unique symbols and no missing values. ## Pré-processing des données

```

caf.data <- data.matrix(cbind(p5, p4a, p4b))

# Ensembl mapping
'

ens <- read.csv("~/Projets_GIT/Single_Cell_Project/Data/ensembl-38-108-genes.txt", sep="\t")
ens <- read.csv("~/Documents/git/Single_Cell_Project/Data/ensembl-38-108-genes.txt", sep="\t")
ens2symb <- setNames(ens$Gene.name, ens$Gene.stable.ID)
ens2type <- setNames(ens$Gene.type, ens$Gene.stable.ID)

symbols <- ens2symb[rownames(caf.data)]
types <- ens2type[rownames(caf.data)]

good <- types == "protein_coding" & !is.na(symbols) & !duplicated(symbols)
symb <- symbols[good]

caf.data <- caf.data[good, ]
rownames(caf.data) <- symb

```

On crée un premier objet SeuratObject pour tirer profit des routines intégrées à la librairie (FindVariableFeatures, ScaleData, RunPCA, etc.) :

```
caf <- CreateSeuratObject(  
  counts = caf.data,  
  project = "CAF_raw",  
  min.cells = 0.01 * ncol(caf.data),  
  min.features = 1000  
)  
#caf
```

2.1 QC et filtrage des données

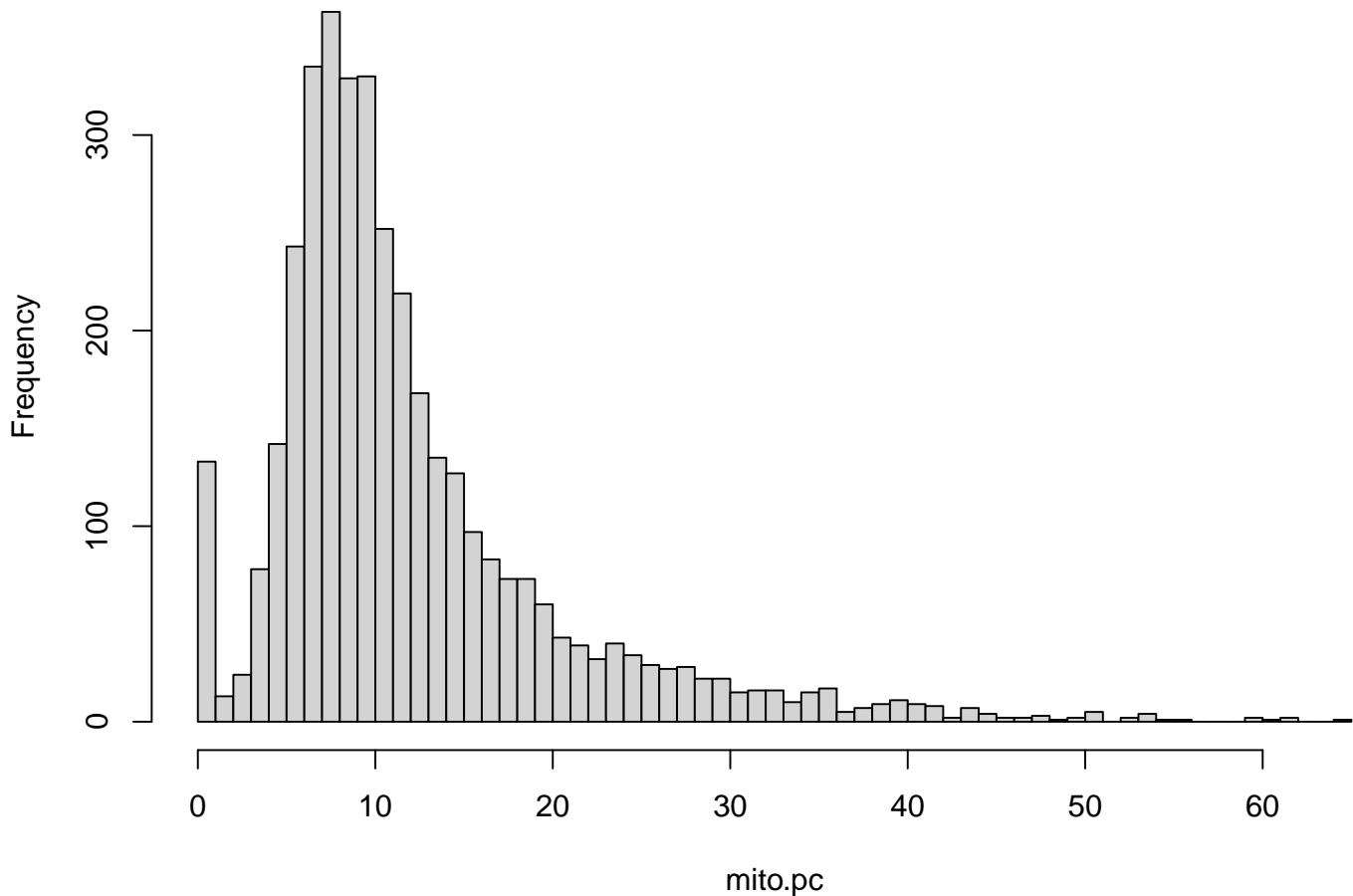
Puis on va définir des filtres. La détermination de ces derniers se fait sur des règles d'usage en *single-cell*. *min.cells = 0.01* au premier passage pour exclure les gènes exprimés dans très peu de cellules (ici adapté au CreateSeuratObject). On applique ensuite des QC aux cellules : - Les UMIs élevés (>50000) indiquent des dupliques il faut donc les gérer. - Les cellules avec peu de gènes détectés (< 1000) sont souvent de mauvaise qualité, il y'a donc nécessité de les filtrer. - Ensuite on ajoute une condition pour gérer la proportion de gènes mitochondriaux élevée (> 50), qui est un indicateur de cellules mortes. En combinant ces filtres, on nettoie notre dataset de manière avoir un jeu de cellule de haute qualité.

```
# Identify mitochondrial genes  
  
ensdb.genes <- genes(EnsDb.Hsapiens.v86)  
MT.names <- ensdb.genes[seqnames(ensdb.genes) == "MT"]$gene_name  
counts <- GetAssayData(caf, "RNA")  
umi.tot <- colSums(counts)  
gene.tot <- colSums(counts > 0)  
mito.pc <- colSums(counts[rownames(counts) %in% MT.names, ]) / umi.tot * 100  
  
# Quality filters  
bad.high <- umi.tot > 50000  
bad.low <- gene.tot < 1000  
bad.mito <- mito.pc > 50  
bad <- bad.high | bad.low | bad.mito  
  
# Cleaning  
counts <- counts[, !bad]  
good.genes <- rowSums(counts > 1) >= 0.01 * ncol(counts)  
counts <- counts[good.genes, ]  
dim(counts)
```

Nous pouvons visualiser la distribution de la proportion de gènes mitochondriaux, avant nettoyage :

```
hist(mito.pc, breaks=50)
```

Histogram of mito.pc



Nous mettons à jour notre objet Seurat avec les données nettoyées :

```
caf <- CreateSeuratObject(counts = counts, project = "CAF_clean")
caf[["percent.mt"]] <- PercentageFeatureSet(caf, pattern = "^\$MT-")
```

3 Normalisation avec SC Transform.

La normalisation étant une étape charnière et pouvant grandement changer les résultats finaux en Single-Cell. Nous avons choisi ici d'exploiter les fonctionnalités de normalisation de la librairie Seurat. Plutôt qu'une normalisation CPM (par millions) ou LogNormalize, nous appliquons ici **SCTransform**, une méthode plus moderne et robuste, qui est construite sur un modèle négatif binomial (Hafemeister & Satija, 2019). De ce que nous en avons compris, une telle approche va agir sur plusieurs aspects : - La normalisation par profondeur de séquençage,
- La stabilisation de variance,
- La sélection des gènes variables,
- La correction des effets techniques (ici regression des % mitochondriaux). Cette méthode est d'après la littérature **la référence Seurat** pour le pré-traitement.

3.1 Le modèle mathématique derrière la normalisation SCTransform

La normalisation SCTransform repose sur un modèle de **régression négative binomiale**.

Pour chaque gène g et chaque cellule i , on suppose : $y_{ig} \sim NB(\mu_{ig}, \theta_g)$ où y_{ig} est le nombre de transcripts observés, μ_{ig} la moyenne attendue de nos comptes et θ_g : le paramètre de dispersion du gène g . La moyenne μ_{ig} est modélisée via un lien log-linéaire.

$$\log(\mu_{ig}) = \beta_{0g} + \beta_{1g} \cdot \log(\text{UMI}_i) + \sum_k \gamma_{kg} x_{ik}$$

avec : - β_{0g} : intercept spécifique au gène g

- UMI_i : total de counts de la cellule i (size factor)

- x_{ik} : variables techniques à régresser (ex. % mitochondriaux, batch)

- γ_{kg} : coefficients associés aux covariables La normalisation finale utilisée pour l'analyse (PCA, clustering) est obtenue via les **résidus Pearson standardisés** :

$$r_{ig} = \frac{y_{ig} - \hat{\mu}_{ig}}{\sqrt{\hat{\mu}_{ig} + \frac{\hat{\mu}_{ig}^2}{\theta_g}}}$$

Ces résidus sont **variance-stabilisés** et permettent d'atténuer l'effet de la profondeur de séquençage et du bruit technique. Avec cette approche, on à *a priori* un modèle individuel pour chaque gène, et une normalisation « cellule spécifique ».

3.2 Application de la méthode

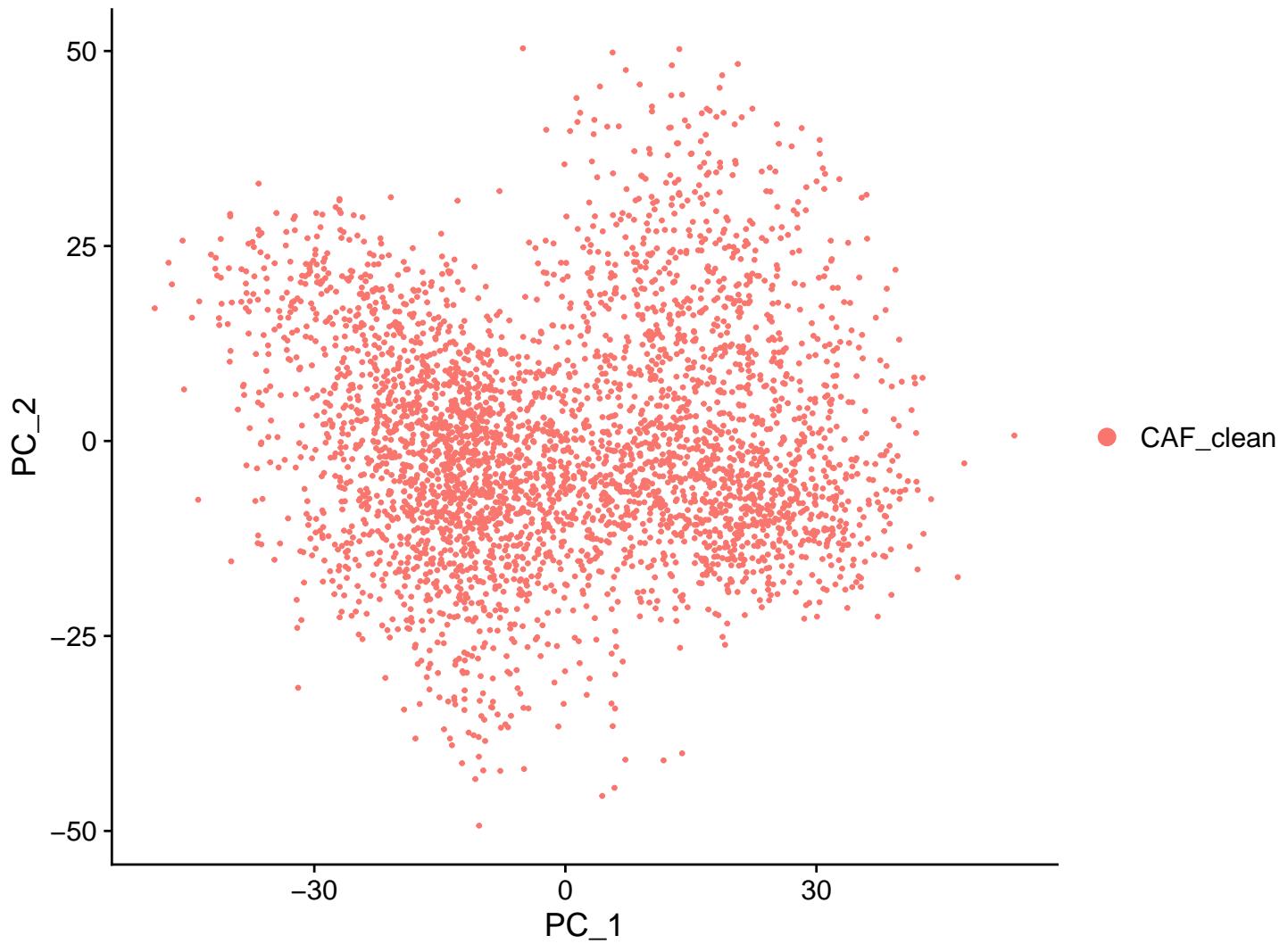
```
caf <- SCTransform(  
  caf,  
  vars.to.regress = "percent.mt",  
  verbose = TRUE)
```

4 Réduction de dimension, clustering et visualisation

Nous procédons à la réduction de dimension via PCA, puis UMAP pour la visualisation. Nous effectuons ensuite le clustering des cellules et identifions les marqueurs de chaque cluster.\

4.1 Interprétation PCA\

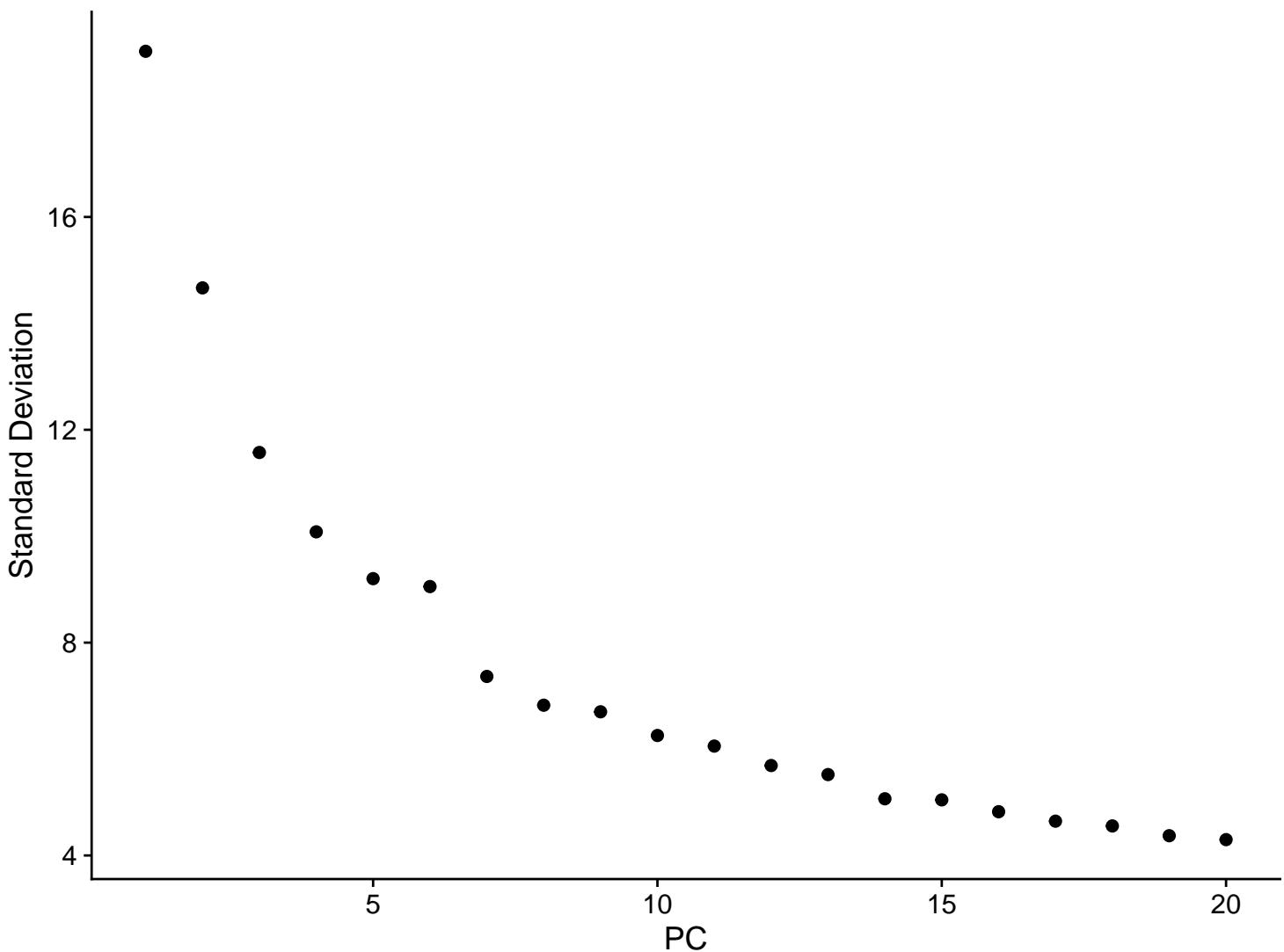
```
caf <- RunPCA(caf, verbose = FALSE)  
DimPlot(caf, reduction = "pca")
```



La PCA des cellules CAF nettoyées nous montre ici un nuage diffus sans clusters franchement identifiable. C'est le reflet d'une grande variabilité plutôt que de sous-populations clairement séparées. On peut expliquer cela en partie par la diversité des CAF, mais aussi par le fait que la PCA capture surtout la variance linéaire qui est dominante. Avant SCTransform, des effets techniques comme la profondeur de séquençage ou le % mitochondrial peuvent encore masquer la structure biologique réelle. Les signatures transcriptionnelles typiques des CAF sont ici illustrés par cette dispersion dans l'espace PCA. Cette première exploration nous confirme la qualité que des méthodes non linéaires (UMAP, t-SNE) sont nécessaires pour que nous allions plus loin dans les investigations.

4.2 Interprétation de l'ElbowPlot\

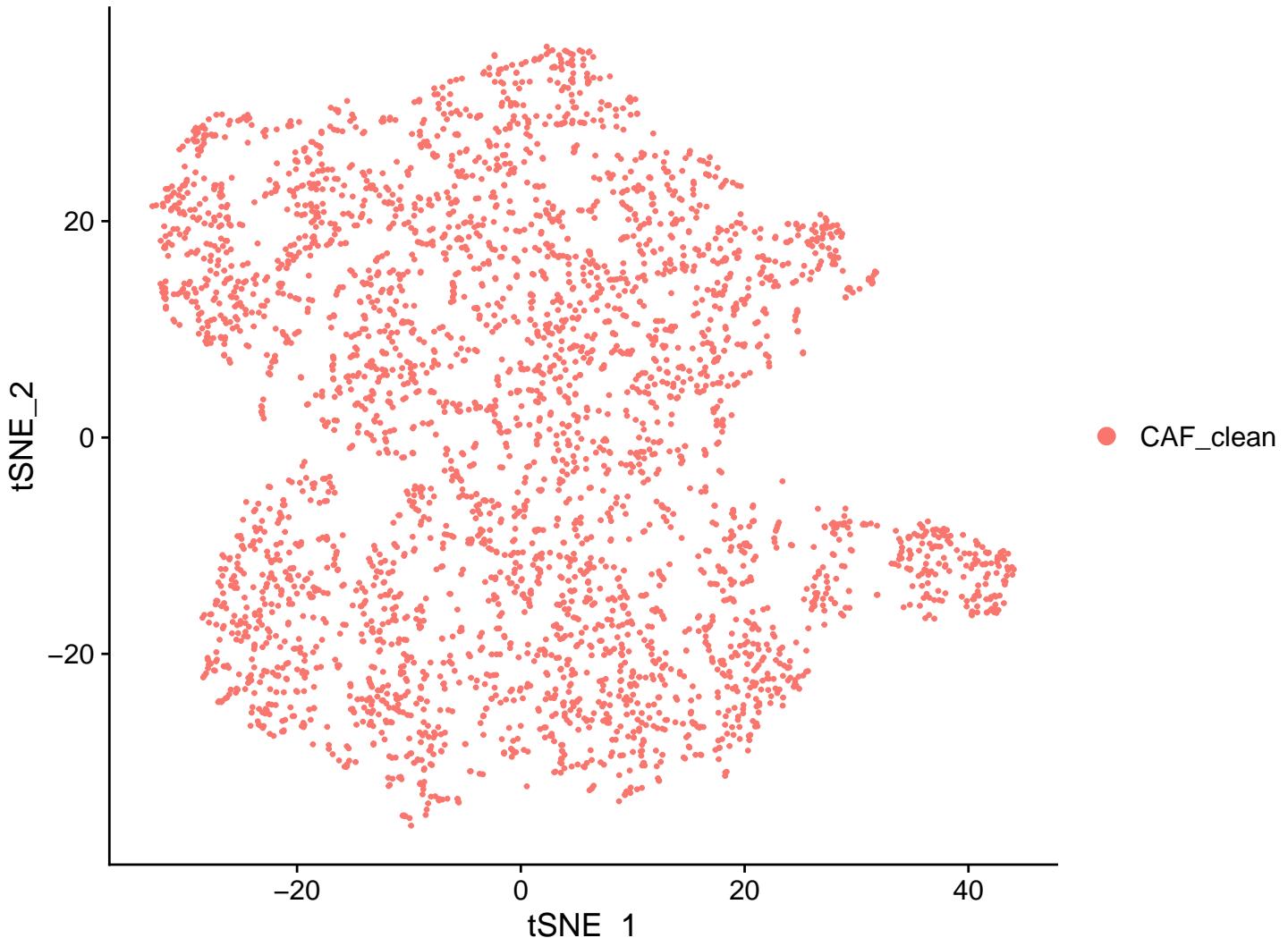
```
ElbowPlot(caf)
```



Le graphique ElbowPlot nous montre que les 10 premières composantes principales capturent la majorité de la variance significative dans les données (ce qui paraît cohérent avec les notions expliquées en cours). Au-delà de ce seuil, l'ajout de composantes supplémentaires contribue peu à l'explication de la variance totale, on a donc un point d'inflexion clair. Par conséquent, nous choisissons d'utiliser les 10 premières dimensions pour les analyses ultérieures telles que UMAP et t-SNE. Cela nous permet donc de capturer la structure transcriptionnelle des CAF sans intégrer trop de bruit.

4.3 Interprétation TSNE \

```
caf_tsne <- RunTSNE(caf, dims = 1:10, perplexity = 30)
DimPlot(caf_tsne, reduction = "tsne")
```



Nous avons vu que le t-SNE est une méthode de réduction non linéaire qui va nous aider à révéler la structure locale des données en essayant de préserver les voisinages cellulaires, contrairement à la PCA qui capture surtout la variance globale (van der Maaten & Hinton, 2008).

Dans nos caf nettoyés , il va nous permettent d'explorer d'éventuels sous-types tels que les myofibroblast-like ou immunomodulatory CAF (Elyada et al., 2019), ainsi que d'identifier d'éventuels signatures transcriptionnelles de cette population très plastique. En faisant un peu de bibliographie, nous avons cru comprendre que les CAF forment souvent des trajectoires continues plutôt que des clusters séparés, le t-SNE diffus que nous avons ici avec des amas pas clairement délimités est cohérent biologiquement avec ce que nous avons pu lire (Ohlund et al., 2017 - Helms et al., 2022). Cette absence de clusters bien définis n'est donc à priori pas un problème technique, mais la nature graduelle et hétérogène des CAF dans l'environnement tumoral. En complément de notre PCA et de l'elbow plot, le t-SNE consolide cette idée d'une organisation continue de cette population fibroblastique. Dans notre démarche pour distinguer des vrais clusters il faut aller donc plus loin.

4.4 Interprétation UMAP et clustering :

```
caf_umap <- RunUMAP(caf, dims = 1:10)
caf_umap <- FindNeighbors(caf_umap, dims = 1:10)
caf_umap <- FindClusters(caf_umap, resolution = 0.15)

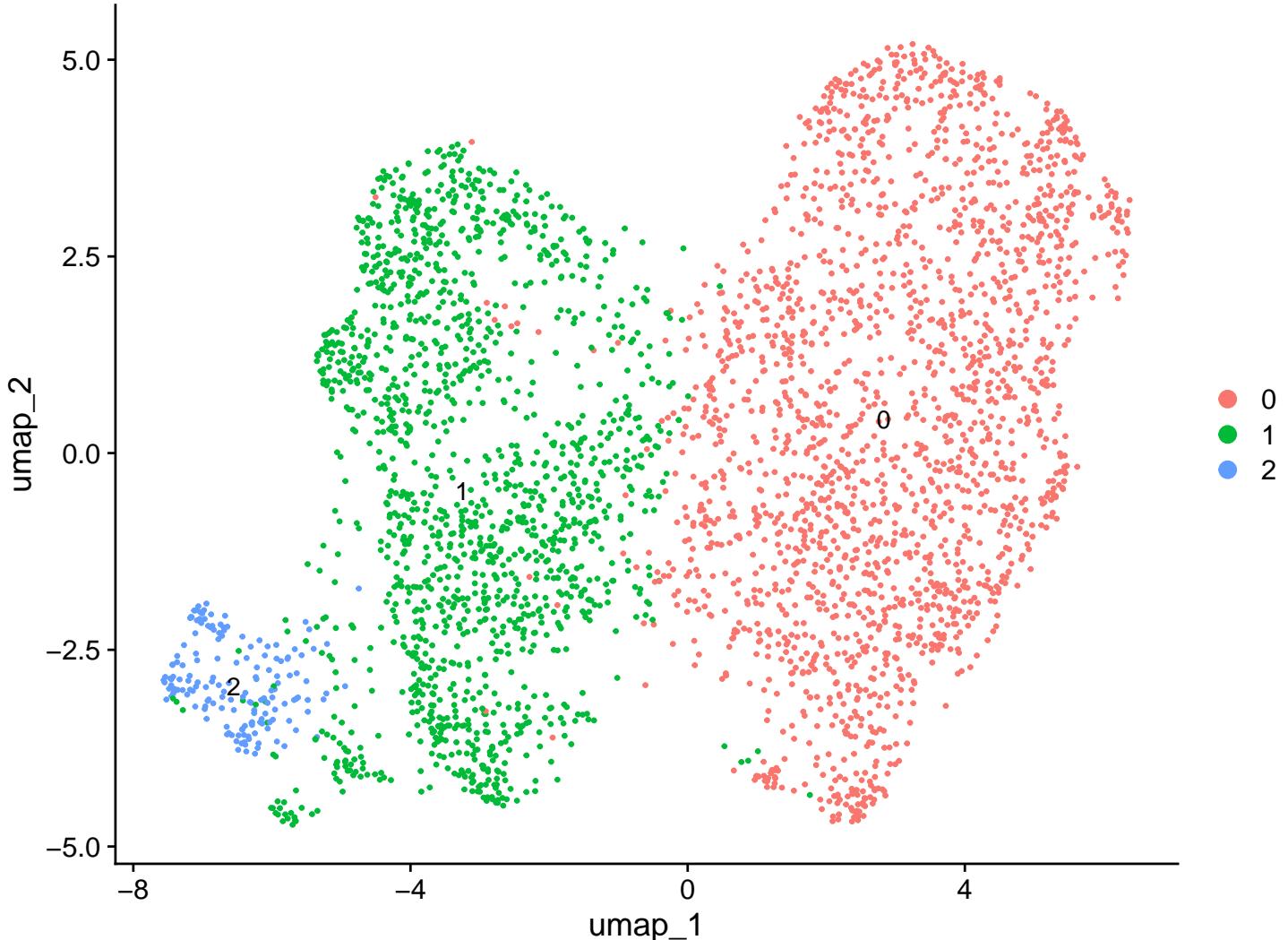
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
```

```

## 
## Number of nodes: 3728
## Number of edges: 118178
## 
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9054
## Number of communities: 3
## Elapsed time: 0 seconds

DimPlot(caf_umap, reduction = "umap", label = TRUE)

```



La dernière méthode que nous avons utiliser est UMAP. “Uniform Manifold Approximation and Projection” (UMAP) se base aussi sur la PCA et permet la réduction de la dimension comme T-sne, mais dans de nombreux cas, en mieux.

Interprétation Clustering :

Chaque ligne correspond à une famille d'origine différente. La première représente les VSMC, la seconde ligne correspond à SAMes, la troisième ligne représente les HSC

```

signatureGene <- c(
  "PLN", "SORBS2", "PHLDA2", "SNCG", "MT1M", "MYH11",
  "PTGDS", "FBLN1", "DCN", "LUM", "COL1A1", "LTBP2",

```

```
"FABP5", "HIGD1B", "AGT", "RGS5", "CPE", "SSTR2")
```

Les VSMC (“*Vascular Smooth Muscle Cells*”) sont donc les cellules musculaire lisse vasculaire, constituant la paroi des vaisseaux sanguin et du coeur. Si l’expression des VSMC est forte, cela suppose une CAF myofibroblastiques d’origine (peri)vasculaire (Chen et al., 2021). Les HSC (“*Hepatic Stellate Cells*”) sont des cellules ‘étoilé’ présentes dans le foie, dans l’espace de Disse. Lors d’un traumatisme, elles peuvent produire du collagén. Si l’expression est forte, cela peut indiquer un cancer du foie et un CAF pro-tumorigéniques d’origine HSC (Affò et al., 2013) Les SAMes sont des cellules regroupant les fibroplastes et les mesenchymales qui se trouvent dans le foie. Si l’expression est forte, cela suppose une origine cicatriciel (<https://doi.org/10.1073/pnas.1400062111>) ([#a revoir](https://www.mdpi.com/1422-0067/20/7/1723)

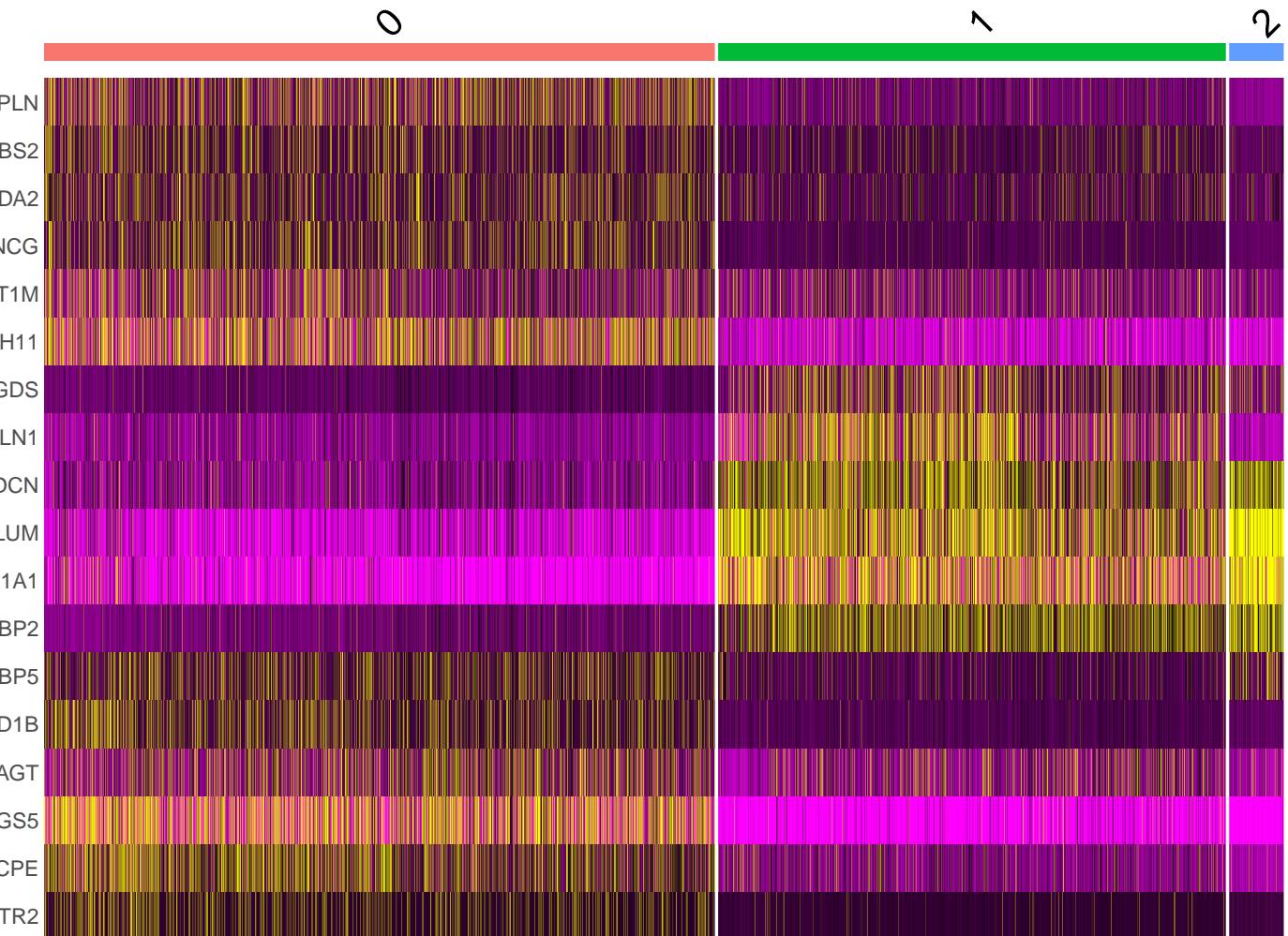
```
caf_markers <- FindAllMarkers(  
  caf_umap,  
  only.pos = TRUE,  
  min.pct = 0.25,  
  logfc.threshold = 0.25)  
  
caf_markers %>% group_by(cluster) %>% slice_max(n = 2, order_by = avg_log2FC)
```

```
## # A tibble: 6 x 7  
## # Groups:   cluster [3]  
##       p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene  
##       <dbl>     <dbl> <dbl>    <dbl>    <dbl> <fct> <chr>  
## 1 8.50e-124      6.63 0.294 0.006 7.62e-120 0    COX4I2  
## 2 2.17e-106      6.12 0.265 0.009 1.94e-102 0    RERGL  
## 3 6.34e-183      7.28 0.379 0.023 5.68e-179 1    C3  
## 4 8.13e-151      4.56 0.321 0.017 7.28e-147 1    SCN7A  
## 5 1.55e-151      7.37 0.297 0.007 1.39e-147 2    PCSK6  
## 6 6.37e-173      7.09 0.267 0.003 5.71e-169 2    NSG1
```

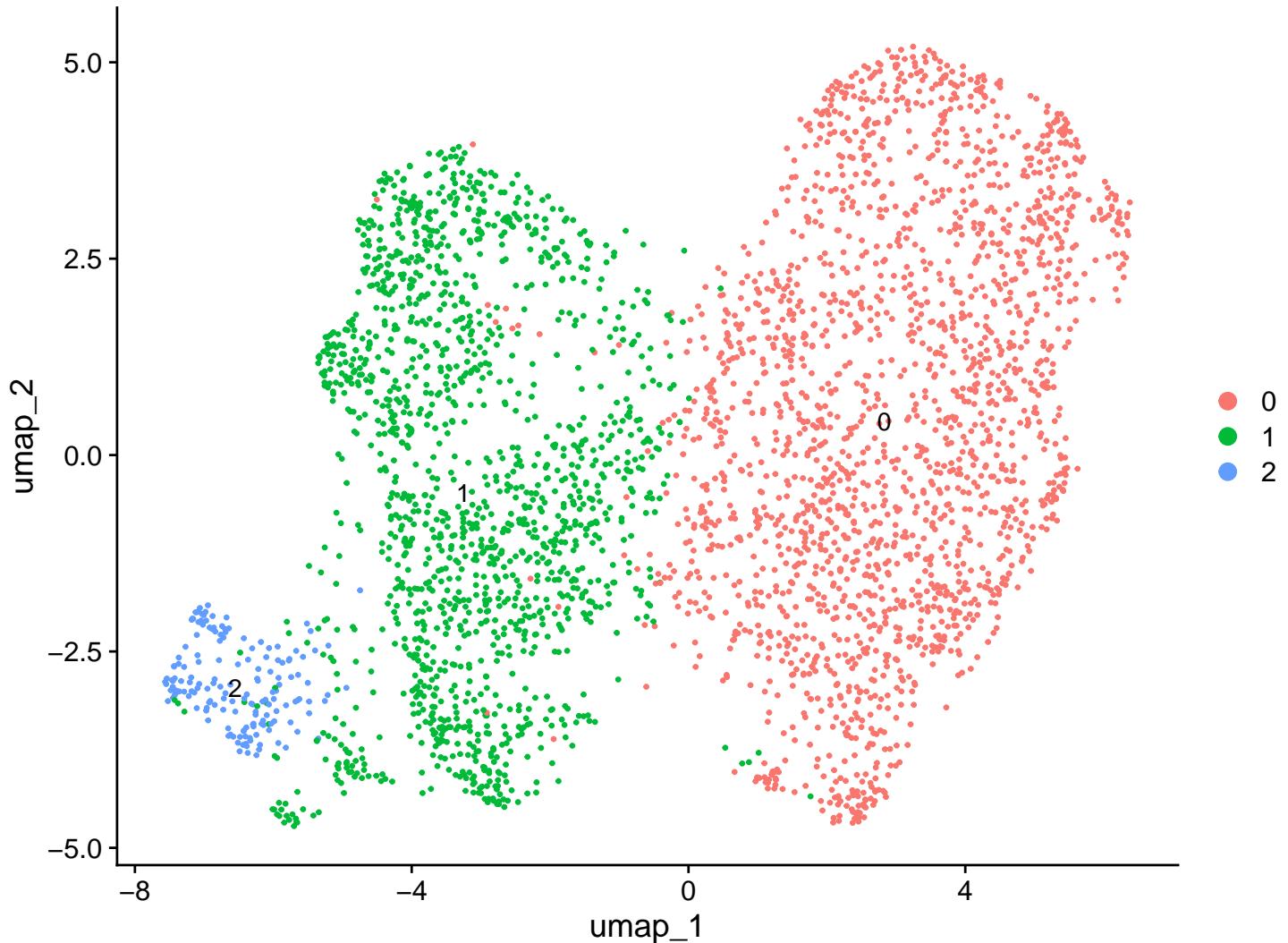
```
signatureGene <- c(  
  "PLN", "SORBS2", "PHLDA2", "SNCG", "MT1M", "MYH11",  
  "PTGDS", "FBLN1", "DCN", "LUM", "COL1A1", "LTBP2",  
  "FABP5", "HIGD1B", "AGT", "RGS5", "CPE", "SSTR2")
```

```
signatureGeneMarker <- caf_markers %>%  
  filter(gene %in% signatureGene) %>%  
  mutate(gene = factor(gene, levels = signatureGene)) %>%  
  arrange(gene)
```

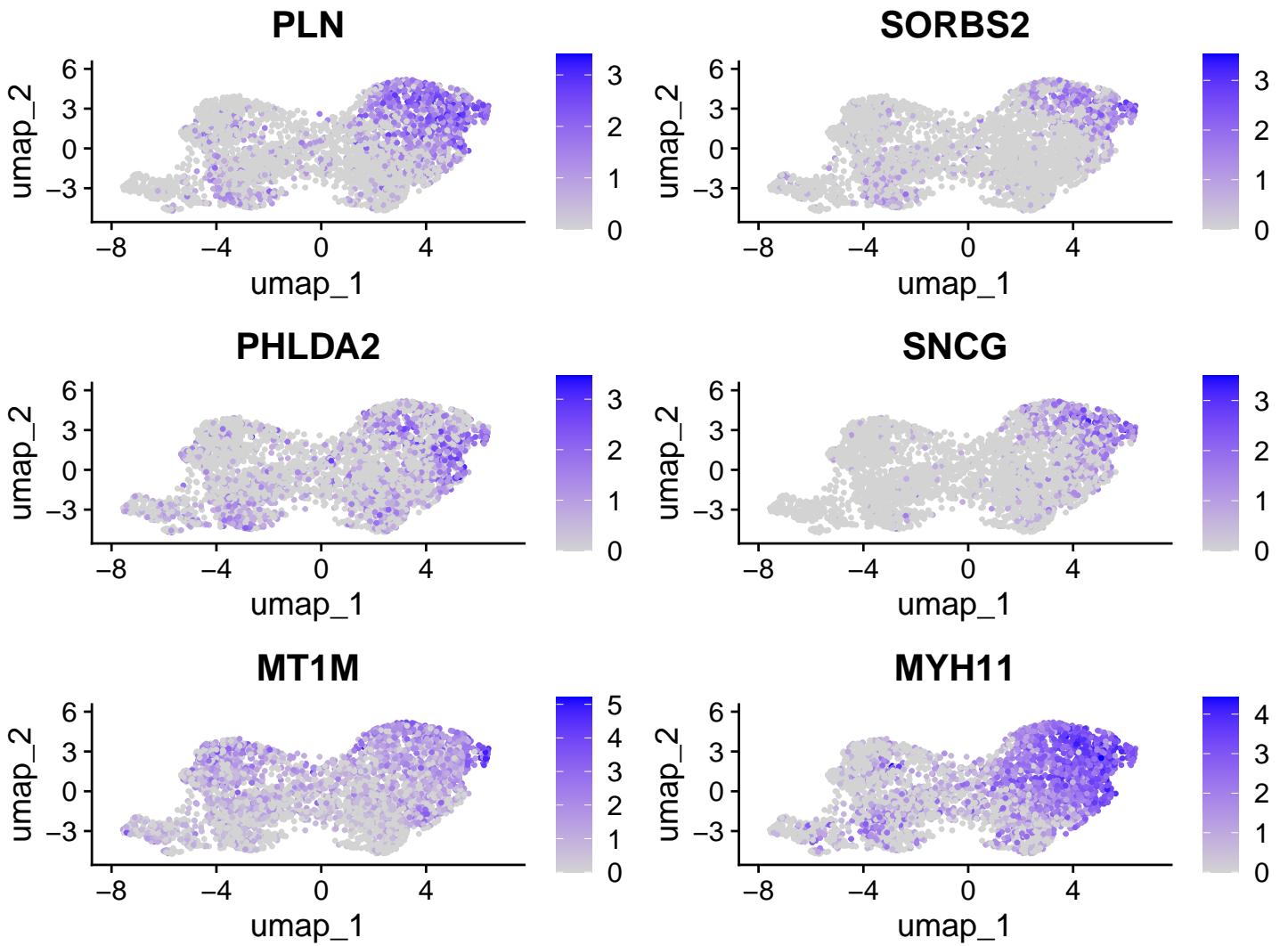
```
DoHeatmap(caf_umap, features = signatureGene) + NoLegend()
```



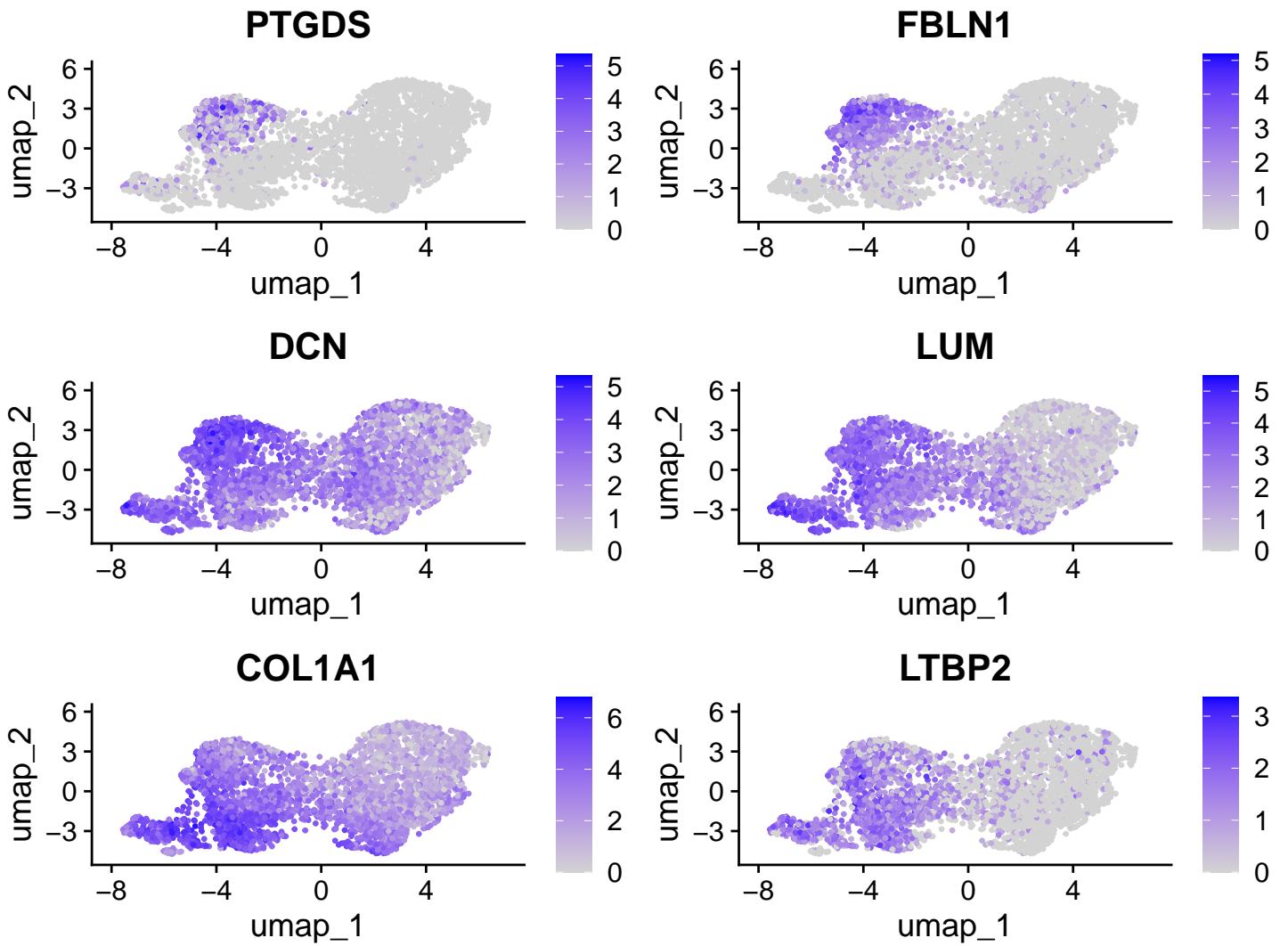
```
# UMAP clusters  
DimPlot(caf_umap, reduction = "umap", label = TRUE)
```



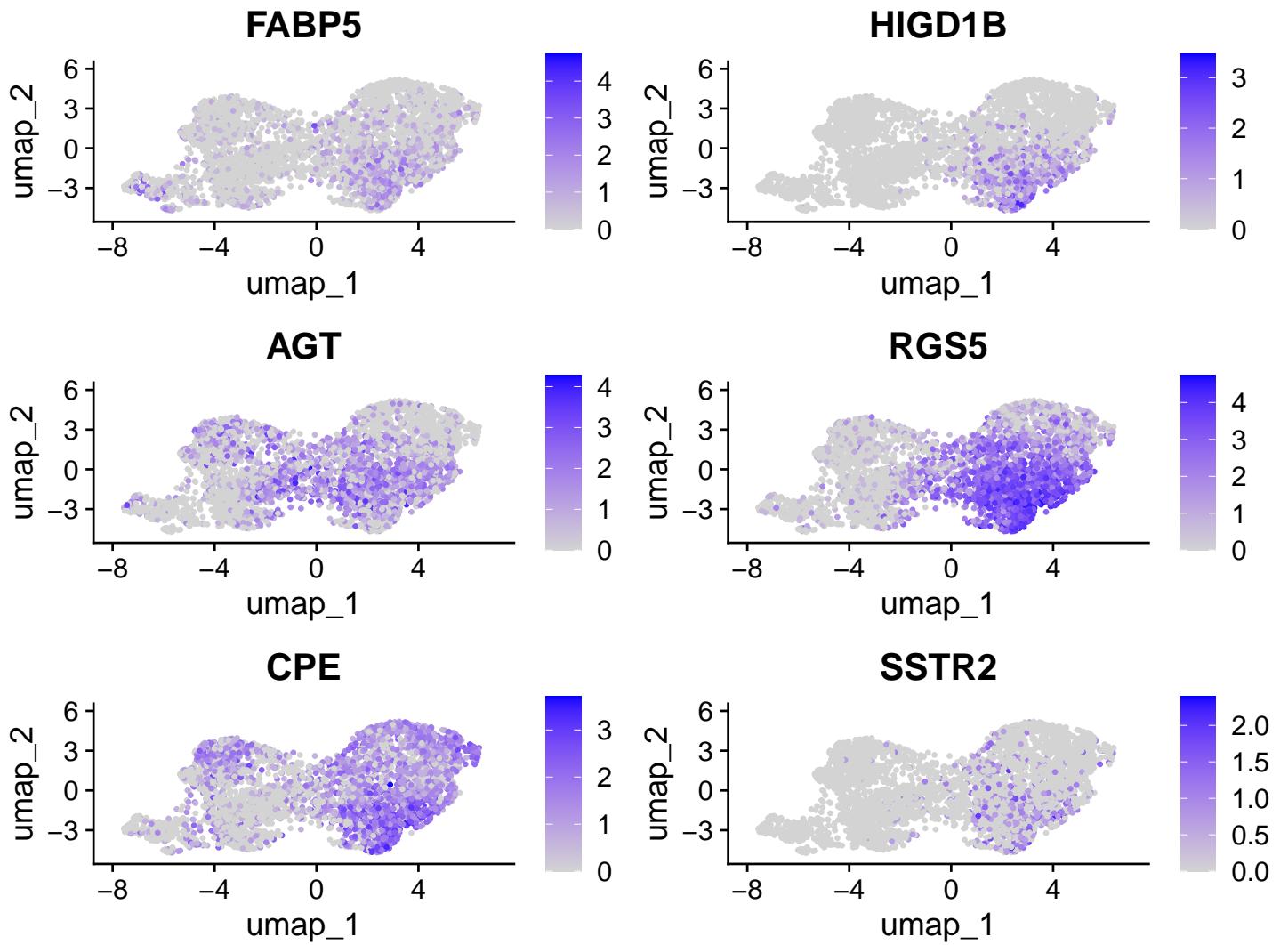
```
# FeaturePlots par famille  
FeaturePlot(caf_umap, features = c("PLN", "SORBS2", "PHLDA2", "SNCG", "MT1M", "MYH11"))
```



```
FeaturePlot(caf_umap, features = c("PTGDS", "FBLN1", "DCN", "LUM", "COL1A1", "LTBP2"))
```



```
FeaturePlot(caf_umap, features = c("FABP5", "HIGD1B", "AGT", "RGS5", "CPE", "SSTR2"))
```



Dans le cluster 0, la présence d'expression pour les gènes VSMC et du HSC, mais pas de SAMes suppose qu'il y a une double origine ? Dans le cluster 1, la présence d'expression pour les gènes SAMes mais pas des autres suppose une origine des fibroplastes et des mesenchymateux.