ACP T-SNE UMAP

1. Importer les données:

```
data <- read.csv('C:/Users/asus/OneDrive/Desktop/Projet/data/breast_cancer.csv',
                 sep=';', row.names='id_sample')
dim(data)
```

```
## [1] 1016   51
```

```
head(data)
```

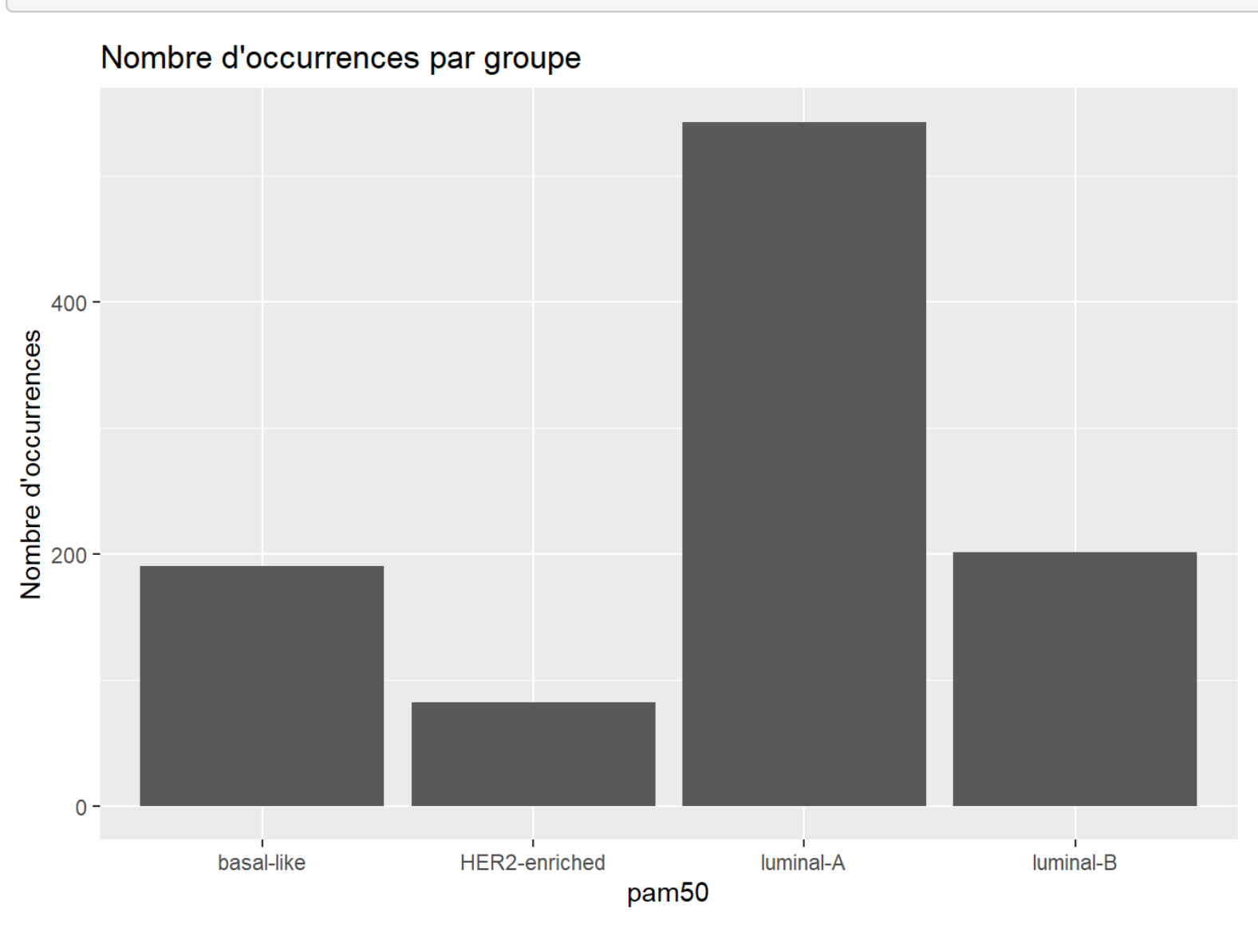
```
##                       NAT1    BIRC5     BAG1     BCL2    BLVRA    CCNB1    CCNE1
## TCGA-3C-AAAU-01A 7.100449 3.361004 3.972581 4.145669 4.765233 4.788987 2.164814
## TCGA-3C-AALI-01A 3.453640 4.501040 2.720929 1.493020 5.823480 5.281003 2.535437
## TCGA-3C-AALJ-01A 4.455574 4.164643 3.911511 4.191457 5.987255 5.229446 2.267963
## TCGA-3C-AALK-01A 4.297961 3.920234 3.688335 3.894904 5.211594 4.014641 0.951107
## TCGA-4H-AAAK-01A 1.695378 2.950846 4.110014 3.572843 4.317856 3.772768 1.103958
## TCGA-5L-AAT0-01A 3.839230 2.030192 3.649085 3.958096 5.186208 3.130406 1.138433
##                       CDC6    CDC20     CDH3    CENPF     EGFR    ERBB2     ESR1
## TCGA-3C-AAAU-01A 2.633598 4.131205 0.133455 3.182165 0.689091 5.239199 3.877698
## TCGA-3C-AALI-01A 2.734157 4.176553 0.110023 3.392215 0.954435 9.927166 0.382603
## TCGA-3C-AALJ-01A 3.379961 4.592752 0.236786 2.984169 0.821807 5.804556 4.859469
## TCGA-3C-AALK-01A 1.472950 3.806552 0.062392 2.893521 2.044761 7.382023 3.284898
## TCGA-4H-AAAK-01A 2.338953 3.473484 0.098773 2.514709 1.527022 5.942930 4.450677
## TCGA-5L-AAT0-01A 1.494393 2.720610 0.061163 2.014191 1.584587 5.579183 4.690124
##                      FGFR4    FOXC1     GRB7    FOXA1     KRT5    KRT14    KRT17
## TCGA-3C-AAAU-01A 0.923478 0.342602 3.845602 6.198849 0.034380 0.000000 0.009777
## TCGA-3C-AALI-01A 4.653488 0.794085 8.780099 6.303003 2.539948 2.954742 3.755362
## TCGA-3C-AALJ-01A 0.605796 1.628313 4.467610 5.905421 0.098767 0.105929 0.310048
## TCGA-3C-AALK-01A 1.462492 2.098339 4.968695 6.470010 5.303344 6.553266 6.691324
## TCGA-4H-AAAK-01A 3.292628 1.559426 4.087059 6.456412 5.081349 5.968474 5.939790
## TCGA-5L-AAT0-01A 0.853090 1.740289 3.603108 6.353737 5.114444 6.002268 5.181969
##                       MAPT     MDM2    MKI67    MMP11    MYBL2      MYC      PGR
## TCGA-3C-AAAU-01A 4.917417 5.880554 3.496616 6.383997 3.961628 5.324133 3.152494
## TCGA-3C-AALI-01A 0.767498 2.169065 3.293928 6.031836 6.328959 1.997876 0.127529
## TCGA-3C-AALJ-01A 3.530899 3.002628 2.222538 6.646212 5.776590 4.428182 1.399339
## TCGA-3C-AALK-01A 3.180234 2.629040 2.355445 5.776565 3.277660 4.540685 2.991844
## TCGA-4H-AAAK-01A 4.834376 2.414163 2.498481 7.791991 3.300091 6.156276 3.961307
## TCGA-5L-AAT0-01A 4.596884 2.475144 1.568097 7.717721 2.301077 4.805902 1.792283
##                       RRM2    SFRP1     TYMS      MIA     EXO1    PTTG1     MELK
## TCGA-3C-AAAU-01A 4.069746 2.610955 3.316276 0.056151 1.851411 3.551024 2.541704
## TCGA-3C-AALI-01A 5.110849 1.028520 3.660515 0.239768 2.893140 3.976765 3.232838
## TCGA-3C-AALJ-01A 3.858266 0.767785 3.567142 0.047283 1.735396 4.473512 3.361018
## TCGA-3C-AALK-01A 3.293315 3.701924 3.479655 1.252874 1.311587 3.117028 2.326996
## TCGA-4H-AAAK-01A 2.302735 4.090581 3.474526 1.224086 1.119937 2.956249 2.377493
## TCGA-5L-AAT0-01A 2.092475 4.301228 2.611648 1.128081 0.877216 2.300325 1.536737
##                      NDC80     KIF2C    UBE2C     ORC6    SLC39A6    PHGDH
## TCGA-3C-AAAU-01A 2.687899 3.422050 4.678216 1.837654 10.319962 2.991554
## TCGA-3C-AALI-01A 2.481391 3.674568 5.883007 2.762908  4.579932 2.973480
## TCGA-3C-AALJ-01A 3.015086 3.505261 5.706425 2.500333  7.747377 1.003328
## TCGA-3C-AALK-01A 1.788017 2.322912 4.378114 0.811602  7.921403 2.781136
## TCGA-4H-AAAK-01A 1.830427 2.509486 3.878444 0.816464  5.725175 2.728038
## TCGA-5L-AAT0-01A 1.253309 1.582389 3.057444 0.621261  5.251726 2.413841
##                     GPR160    UBE2T    CXXC5     ANLN    CEP55   ACTR3B     MLPH
## TCGA-3C-AAAU-01A 4.150233 4.106918 5.528618 3.073409 2.669860 1.928460 5.567999
## TCGA-3C-AALI-01A 5.561226 5.648057 4.711309 3.881110 3.357553 1.168684 7.064176
## TCGA-3C-AALJ-01A 2.859309 5.213461 6.152875 2.697093 2.599436 1.177678 5.222420
## TCGA-3C-AALK-01A 3.063807 4.166154 5.612184 2.645664 2.448027 1.026535 6.225590
## TCGA-4H-AAAK-01A 3.289418 3.437585 4.299617 2.068516 2.152652 1.513181 5.485277
## TCGA-5L-AAT0-01A 3.236713 3.555052 5.281731 1.050713 1.284239 1.413346 6.059373
##                       NUF2  TMEM45B         pam50
## TCGA-3C-AAAU-01A 2.536764 0.213597     luminal-A
## TCGA-3C-AALI-01A 3.124620 3.946538 HER2-enriched
## TCGA-3C-AALJ-01A 3.053335 0.281303     luminal-B
## TCGA-3C-AALK-01A 1.717959 3.289543     luminal-A
## TCGA-4H-AAAK-01A 1.537125 2.976903     luminal-A
## TCGA-5L-AAT0-01A 0.947315 2.884543     luminal-A
```

Afficher les occurences de chaque niveau de la colonne **pam50**:

```
group_sizes <- table(data$pam50)
print(group_sizes)
```

```
##
##    basal-like HER2-enriched     luminal-A     luminal-B
##           190            82           543           201
```

```
library(ggplot2)
ggplot(data, aes(x = pam50)) +
  geom_bar() +
  labs(title = "Nombre d'occurrences par groupe",
       x = "pam50",
       y = "Nombre d'occurrences")
```



2.Séparer les données d'expression et les étiquettes

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#Données d'expression de 50 gènes
X <- select_if(data, is.numeric)
cat('X', dim(X), '\n')
```

```
## X 1016 50
```

```
#Etiquettes correspondantes (sous-types moléculaires)
y <- data$pam50
cat('y', length(y), '\n')
```

```
## y 1016
```

3.Afficher les valeurs d'expression:

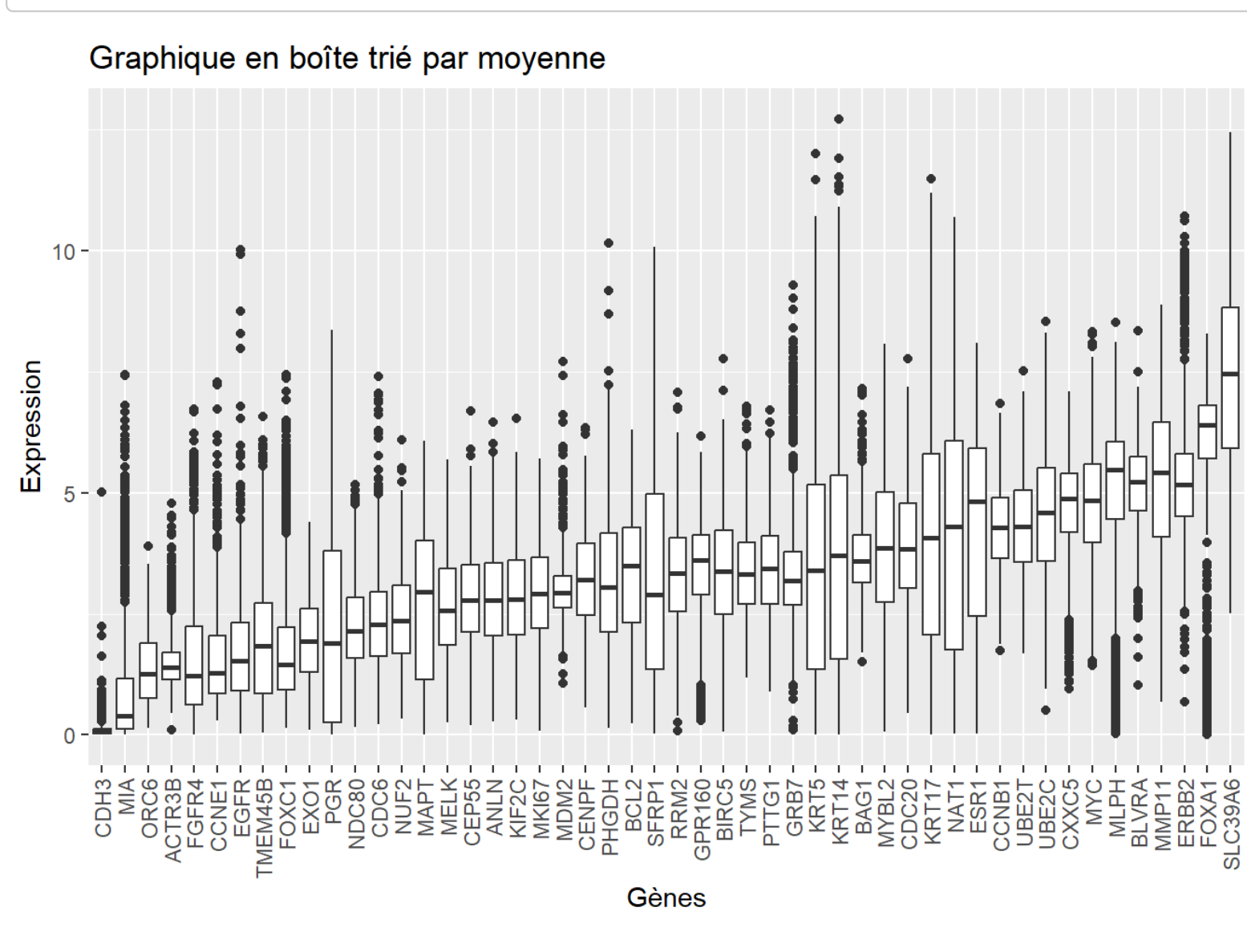
```
library(reshape2)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:reshape2':
##
##     smiths
```

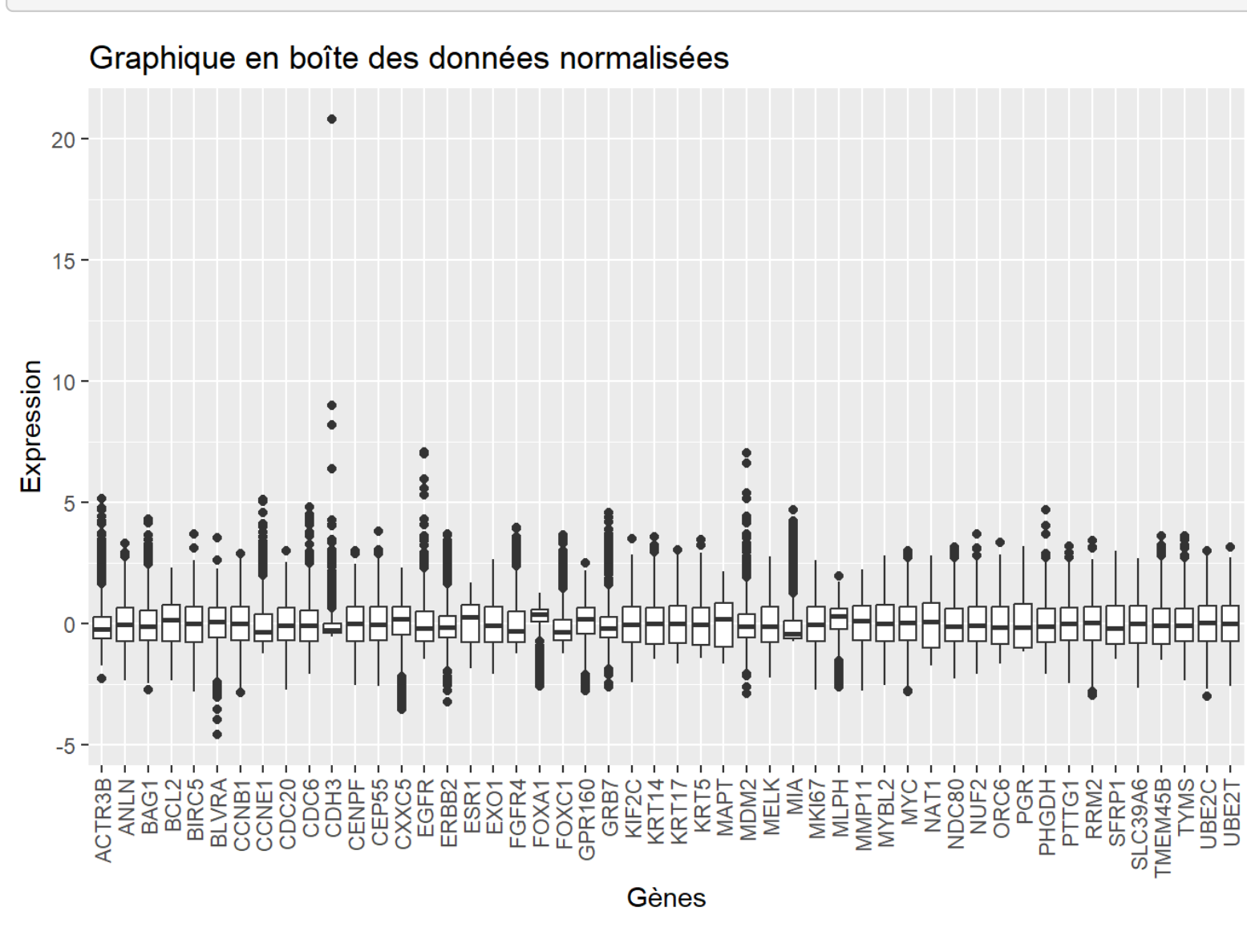
```
#Triez les colonnes par moyenne
sort_by_mean <- colMeans(X)
sort_by_mean <- sort_by_mean[order(sort_by_mean)]
#Créez un data.frame trié
X_sorted <- X[, names(sort_by_mean)]
#Créez le graphique en boîte avec ggplot2
ggplot(melt(X_sorted), aes(x = variable, y = value)) +
  geom_boxplot() +
  labs(title = "Graphique en boîte trié par moyenne",
       x = "Gènes",
       y = "Expression") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## No id variables; using all as measure variables
```



4.Appliquer une normalisation centrée-réduite aux données:

```
#Normalisation centrée-réduite (z-score)
X_scaled <- scale(X)
#Convertir la matrice résultante en un data.frame avec les mêmes noms de colonnes et indices
X_scaled <- as.data.frame(X_scaled)
colnames(X_scaled) <- colnames(X)
rownames(X_scaled) <- rownames(X)
#head(X_scaled)
ggplot(data = gather(X_scaled), aes(x = key, y = value)) +
  geom_boxplot() +
  labs(title = "Graphique en boîte des données normalisées",
       x = "Gènes",
       y = "Expression") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



5. Faire une analyse en composantes principales (ACP)

5.1 Calcul de l'ACP:

```
#Installez et chargez les bibliothèques FactoMineR et factoextra si ce n'est pas déjà fait
#install.packages("FactoMineR")
#install.packages("factoextra")
library(FactoMineR)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
#Réalisez l'ACP sur les données X_scaled
pca_result <- PCA(X_scaled, graph = FALSE)

#Obtenez les composantes principales (PC)
X_pca <- as.data.frame(pca_result$ind$coord)

#Renommez les colonnes des PC
colnames(X_pca) <- paste0("PC", 1:ncol(X_pca))

#Affichez les premières lignes du data.frame avec les composantes principales
head(X_pca)
```

```
##                         PC1        PC2        PC3        PC4         PC5
## TCGA-3C-AAAU-01A -0.8262713  3.4336500  2.5106320 -0.7383054  1.06999830
## TCGA-3C-AALI-01A  2.9028393  3.9221698 -6.6983824  1.5316700  0.13862034
## TCGA-3C-AALJ-01A  0.8692634  3.7491997  0.4344589 -0.6897723 -1.92845117
## TCGA-3C-AALK-01A -1.7659214 -0.7604501 -1.3398320  2.3319832 -0.07452753
## TCGA-4H-AAAK-01A -2.0171204 -1.7409448 -1.1634389  1.3857245 -0.17175045
## TCGA-5L-AAT0-01A -4.5929983 -2.7013014 -1.1158453  0.5633696 -0.79099750
```

5.2 Calcul de la variance expliquée:

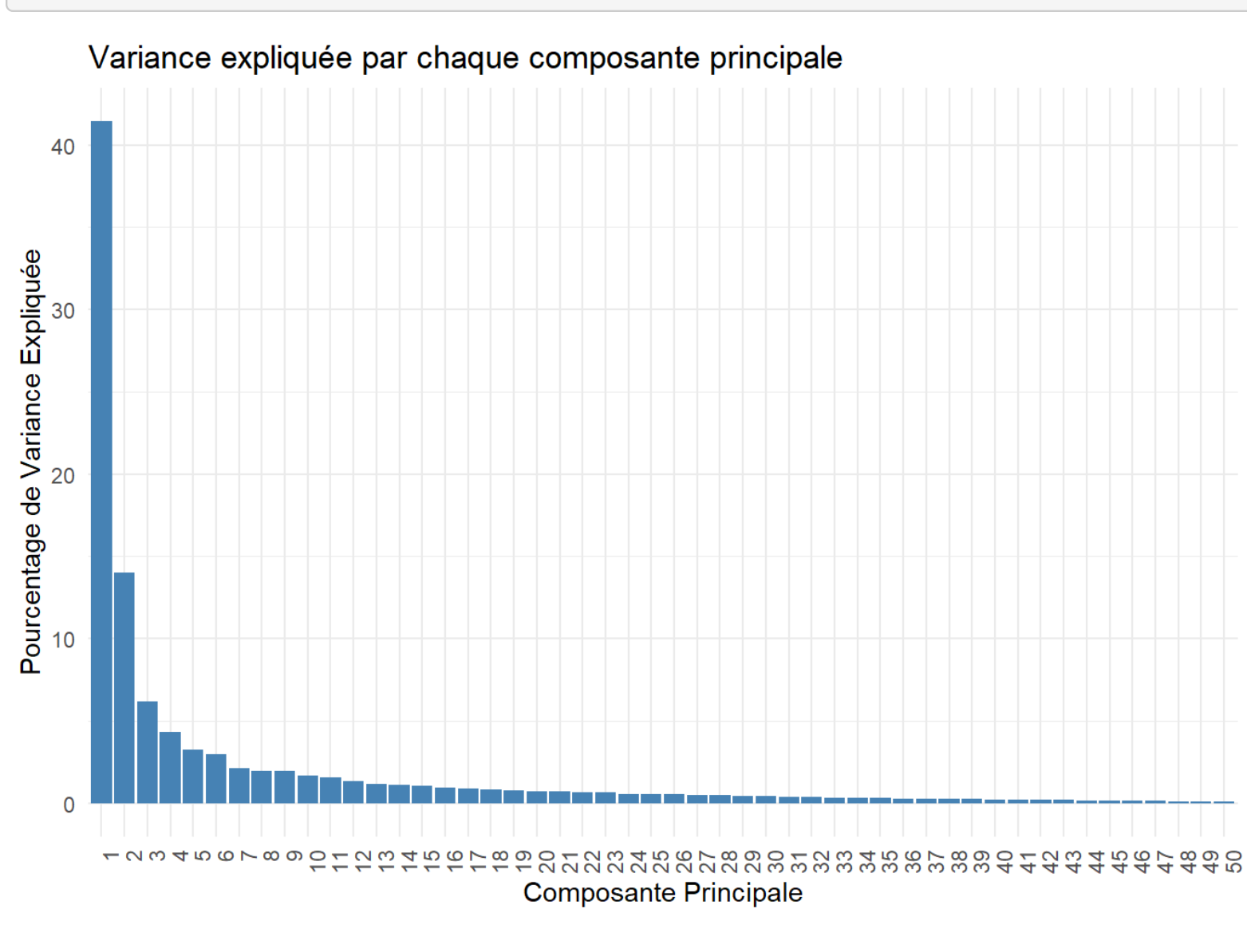
```
pca <- prcomp(X_scaled)
explained_variance <- tibble(
  PC = seq_along(pca$sdev),
  Explained_Variance_Ratio = pca$sdev^2 / sum(pca$sdev^2) * 100
)
# Afficher la variance expliquée
print(explained_variance)
```

```
## # A tibble: 50 × 2
##       PC Explained_Variance_Ratio
##    <int>                    <dbl>
##  1     1                    41.4
##  2     2                    14.0
##  3     3                     6.15
##  4     4                     4.30
##  5     5                     3.23
##  6     6                     2.95
##  7     7                     2.14
##  8     8                     1.93
##  9     9                     1.93
## 10    10                     1.65
## # ℹ 40 more rows
```

```
head(explained_variance)
```

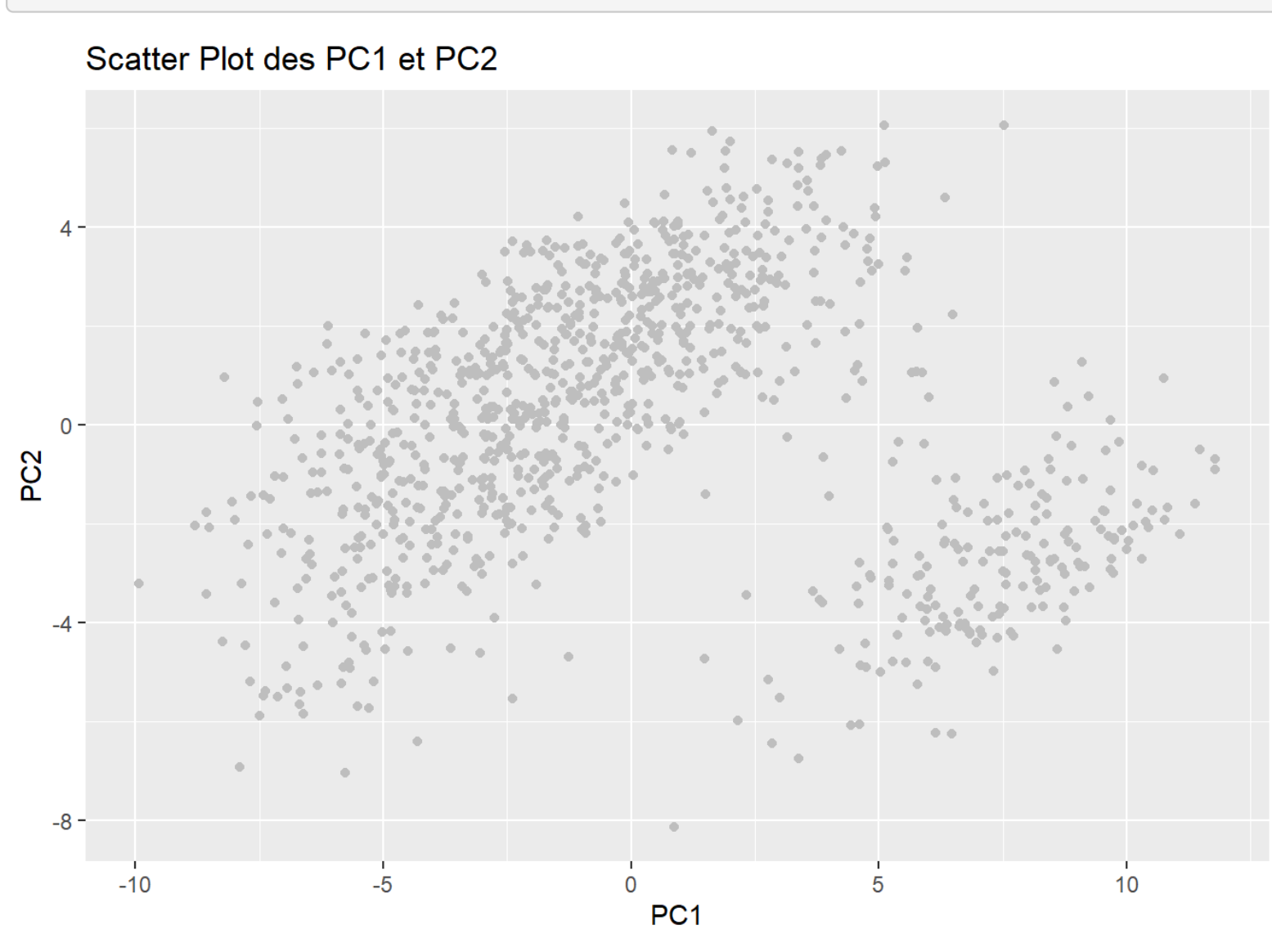
```
## # A tibble: 6 × 2
##      PC Explained_Variance_Ratio
##   <int>                    <dbl>
## 1     1                    41.4
## 2     2                    14.0
## 3     3                     6.15
## 4     4                     4.30
## 5     5                     3.23
## 6     6                     2.95
```

```
#Créer le graphique à barres avec ggplot2
ggplot(data = explained_variance, aes(x = factor(PC), y = Explained_Variance_Ratio)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Variance expliquée par chaque composante principale",
       x = "Composante Principale",
       y = "Pourcentage de Variance Expliquée") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



5.3 Visualisation des deux premières composantes principales de l'ACP:

```
library(ggplot2)
# Create a scatter plot of PC1 vs PC2
ggplot(X_pca, aes(x = PC1, y = PC2)) +
  geom_point(color = 'gray') +
  labs(title = 'Scatter Plot des PC1 et PC2', x = 'PC1', y = 'PC2')
```



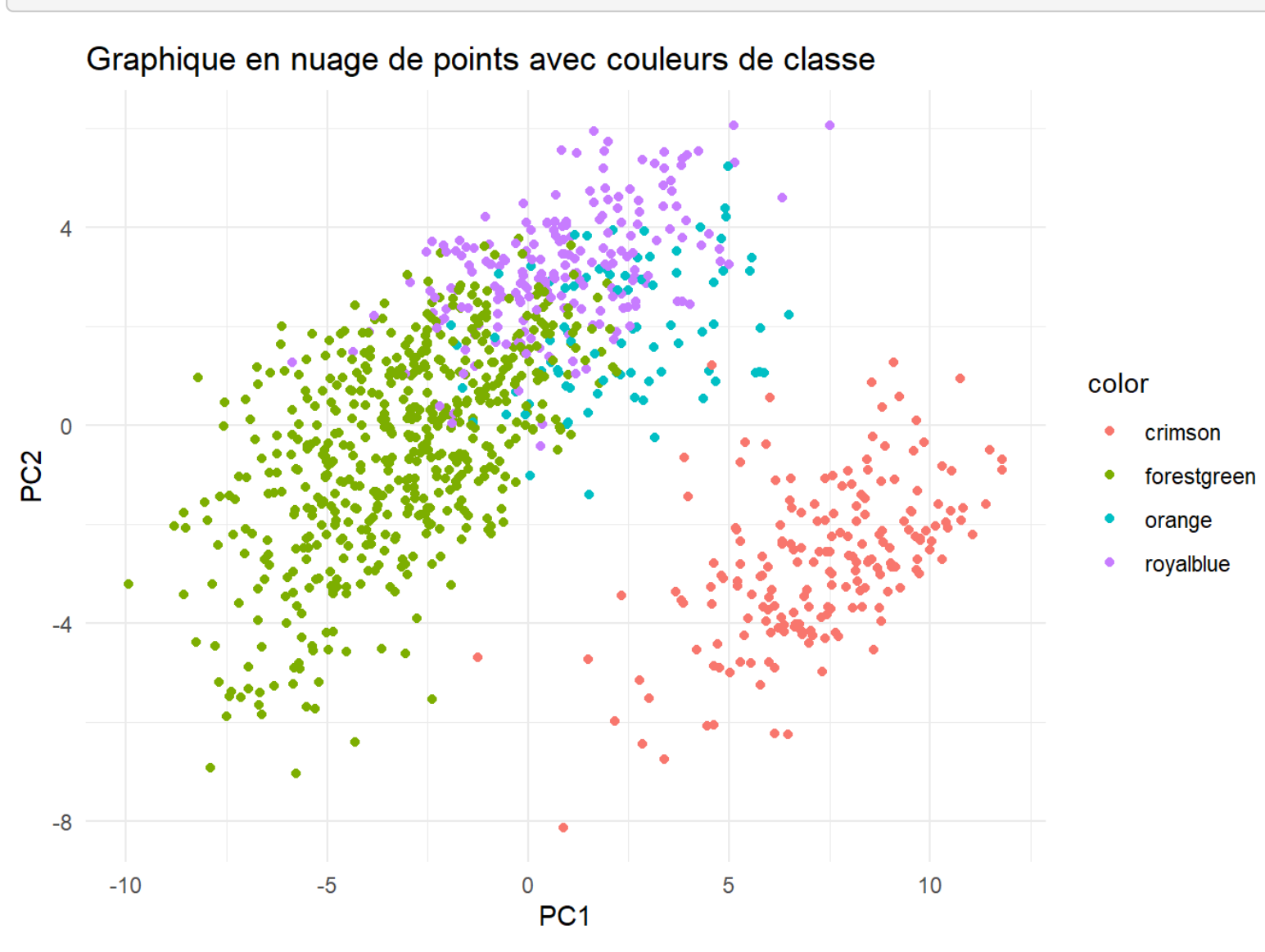
Les points se rassemblent naturellement en clusters, notamment deux clusters sont nettement visibles. Présentons chaque sous-type moléculaire en différente couleur pour comprendre à quoi correspondent ces clusters.

```
library(ggplot2)

# Ajouter les composantes principales au DataFrame data
data_with_pca <- cbind(data, PC1 = X_pca$PC1, PC2 = X_pca$PC2)

# Créer une palette de couleurs pour les classes
dict_colors <- c('luminal-A' = 'forestgreen', 'luminal-B' = 'royalblue', 'HER2-enriched' = 'orange', 'basal-like'
= 'crimson')
data_with_pca$color <- dict_colors[data_with_pca$pam50]

# Créer le graphique en nuage de points
ggplot(data_with_pca, aes(x = PC1, y = PC2, color = color)) +
  geom_point() +
  labs(title = "Graphique en nuage de points avec couleurs de classe", x = "PC1", y = "PC2") +
  theme_minimal()
```



5.4 Visualisation des trois premières composantes principales de l'ACP:

```
#Create 3D scatter plot
#install.packages("rgl")
#library(rgl)
#plot3d(X_pca[, 1], X_pca[, 2], X_pca[, 3], col = y_colors, size = 30, xlab = paste0("PC1 - ", explained_variance
[1], "%"),
#ylab = paste0("PC2 - ", explained_variance[2,], "%"), zlab = paste0("PC3 - ", explained_variance[3,], "%"))
#rgl.viewpoint(theta = 45, phi = 15)
```

La figure est enregistrée dans le fichier **images**

6. Visualiser les données avec la méthode t-SNE:

La méthode **t-SNE** (t-distributed stochastic neighbor embedding) fait partie de la librairie **Rtsne** ## Projection 2D:

```
#Installer la bibliothèque si ce n'est pas déjà fait
if (!requireNamespace("Rtsne", quietly = TRUE)) {
  install.packages("Rtsne")
}

#Charger la bibliothèque
library(Rtsne)

#Définir les paramètres de t-SNE
tsne_params <- Rtsne(X_scaled, check_duplicates = FALSE, pca = TRUE)

#Obtenir les coordonnées t-SNE
X_tsne <- tsne_params$Y

#Renommer les colonnes
colnames(X_tsne) <- paste0("DIM", 1:ncol(X_tsne))

#Afficher les premières lignes
head(X_tsne)
```

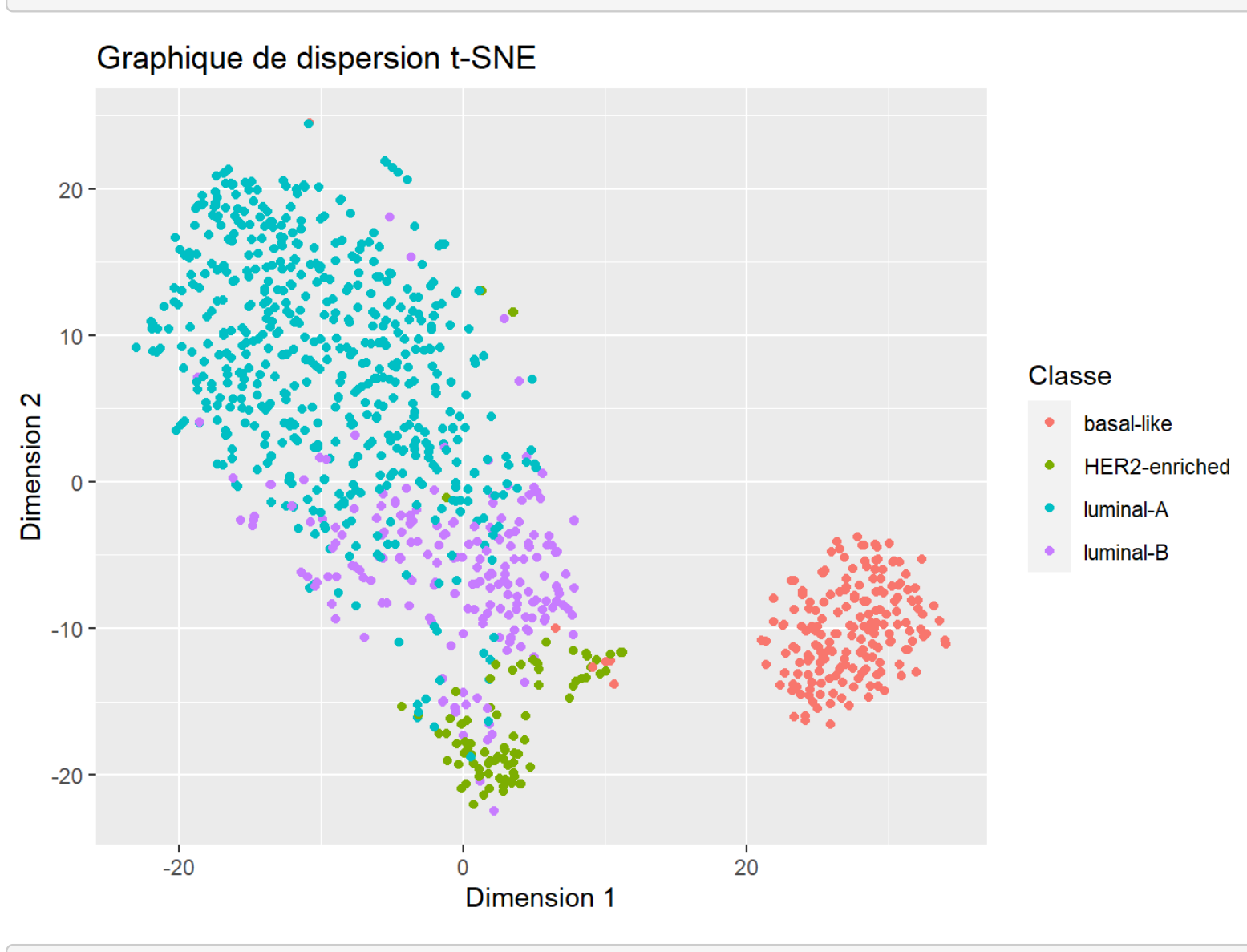
```
##            DIM1        DIM2
## [1,]  -9.348016  -4.5882210
## [2,]   3.416022 -20.5699950
## [3,]   5.207790  -0.6777247
## [4,]  -1.881259  12.0325107
## [5,]  -3.126303  12.5908889
## [6,] -13.286008  15.8917203
```

```
#Convertir la matrice en data.frame
X_tsne_df <- as.data.frame(X_tsne)

#Ajouter la classe y à X_tsne_df
X_tsne_df$y <- y

#Créer le graphique de dispersion t-SNE en 2D avec ggplot2
tsne_plot <- ggplot(X_tsne_df, aes(x = DIM1, y = DIM2, color = y)) +
  geom_point() +
  labs(title = "Graphique de dispersion t-SNE",
       x = "Dimension 1",
       y = "Dimension 2",
       color = "Classe")

#Afficher le graphique
print(tsne_plot)
```



```
#Charger le package Rtsne
library(Rtsne)

tsne_result <- Rtsne(X_scaled, dims = 2)

#Accéder à la divergence de Kullback-Leibler
kl_divergence <- tsne_result$itercost

#Obtenir le résultat de la dernière itération
resultat_derniere_iteration <- kl_divergence[length(kl_divergence)]

#Afficher le résultat de la dernière itération
print(resultat_derniere_iteration)
```

```
## [1] 1.233822
```

7. Visualiser les données avec la méthode UMAP:

Projection 2D:

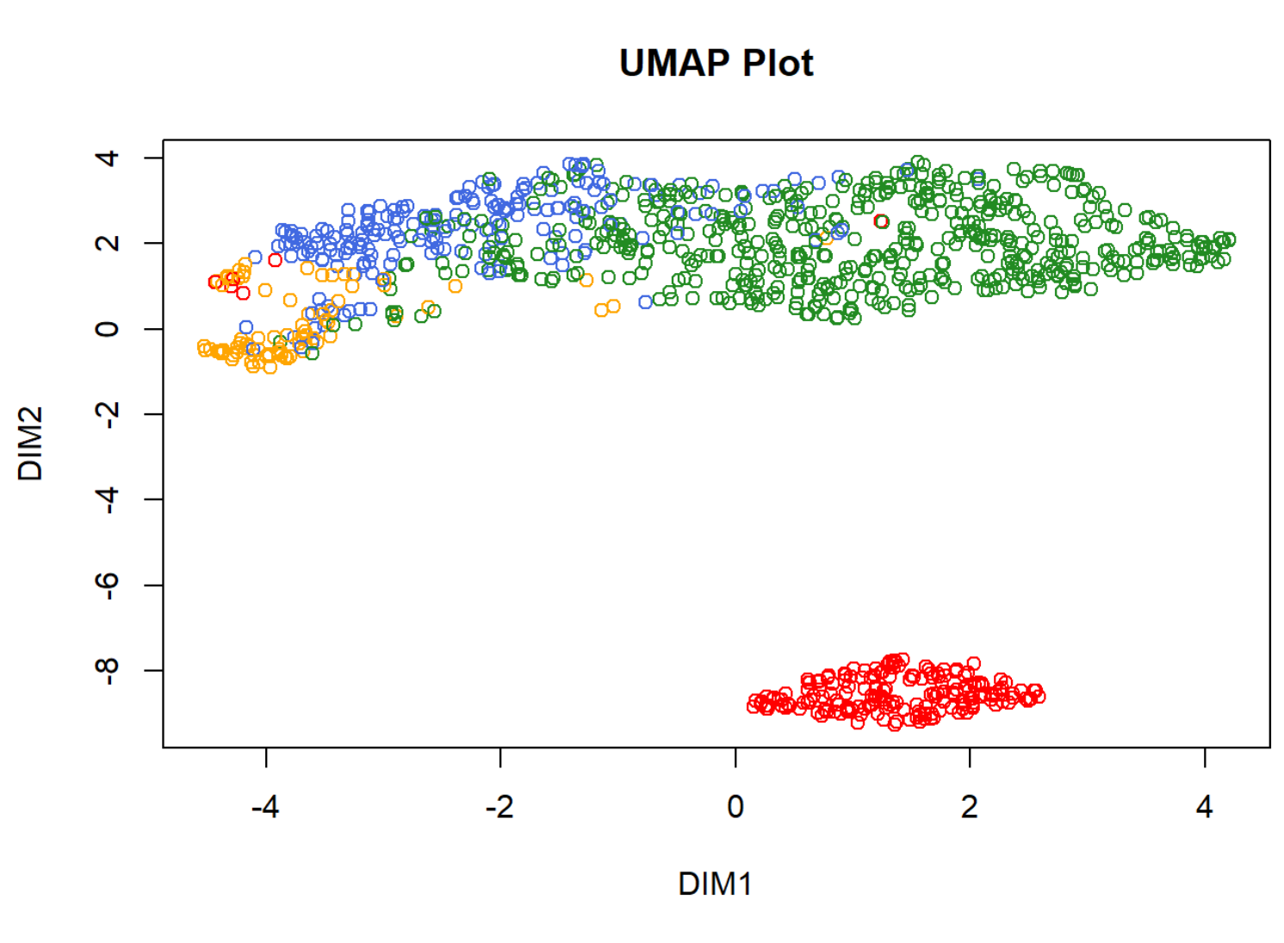
```
library(umap)

# Appliquer UMAP
set.seed(0)  # Pour la reproductibilité
embedding <- umap(X_scaled, n_components = 2, random_state = 0, n_threads = -1)

# Créer un dataframe
X_umap <- data.frame(DIM1 = embedding$layout[, 1], DIM2 = embedding$layout[, 2])

dict_colors <- c('luminal-A' = 'forestgreen', 'luminal-B' = 'royalblue', 'HER2-enriched' = 'orange', 'basal-like'
= 'red')
y_colors <- as.character(sapply(y, function(yi) dict_colors[yi]))

# Tracer le graphique de dispersion
plot(X_umap$DIM1, X_umap$DIM2, col = y_colors, xlab = "DIM1", ylab = "DIM2", main = "UMAP Plot")
```



```
ggplot(X_umap, aes(x = DIM1, y = DIM2, color = as.factor(y))) +
  geom_point() +
  labs(title = "UMAP Plot",
       x = "DIM1",
      y = "DIM2")
```