ST scRNAseq 2e partie : Prétraitement Visualisation & Visualisation

Guillaume SOUEDE

24 mai 2023

1. Chargement des Packages

Démarrer les packages préalablement installés.

```
genomics = "data/brain"
bc_matrix.h5 = "Visium_FFPE_Mouse_Brain_filtered_feature_bc_matrix.h5"
objet_seurat <- Load10X_Spatial(
  genomics,
  bc_matrix.h5,
)
objet_seurat <- SCTransform(objet_seurat, assay = "Spatial", verbose = FALSE)
objet_seurat <- RunPCA(objet_seurat, assay = "SCT", verbose = FALSE)
objet_seurat <- FindNeighbors(objet_seurat, reduction = "pca", dims = 1:30)
objet_seurat <- FindClusters(objet_seurat, verbose = FALSE)
objet_seurat <- RunUMAP(objet_seurat, reduction = "pca", dims = 1:30)
```

Ensemble de données utilisé | Visium

2. Prétraitement visualisation - Single-Cell Signature SCORER

2.1 Préparation

Single-Cell Signature Explorer regroupe 4 logiciels, dont le SCORER. Télécharger SingleCellSignatureScorer dans un dossier.

Le Choix des Databases se fait en fonction de l'organisme étudié (humain ou souris) et du but de l'expérience. Les collections appropriées pour l'étude Single-Cell sont C8 pour l'humain et M8 pour la souris. Ce sont des ensembles de gènes de signature des type cellulaire élaborés à partir de marqueurs de clusters identifiés dans le cadre d'études de séquençage de single-cell des tissus respectifs.

Télécharger la Database appropriée : C8 (humain) ou M8 (souris).

Dans SingleCellSignatureScorer/data : remplacer la table .tsv par celle créée dans la vignette précédente. Dans SingleCellSignatureScorer/databases : placer la database choisie, en ne conservant qu'un seul niveau de répertoires à l'intérieur du dossier databases.

2.2 Exécuter le SCORER

Ouvrir le dossier du SCORER dans un terminal. Faire :

Dans le terminal, choisir la database à utiliser avec le pavé numérique.

Une fois les calculs terminés : Dans SingleCellSignatureScorer/results, récupérer la table .tsv obtenue. Par exemple : M8_table_brain.tsv

3. Retour sur R

3.1 Importer la sortie du SCORER

Cliquer sur "Upload", choisir la table .tsv issue du SCORER. Faire de même pour les meta_coordinates (vignette précédente, ST scRNAseq 1ère partie). Lecture des deux tables.

```
scorer = read_tsv("mydata/seurat/M8_table_objet_seurat.tsv", col_names = T)
```

3.2 Traitement du fichier

Cet étape permet de récupérer les données liées à l'objet_seurat.

```
# Relier les metadata de l'objet aux coordonnées X et Y de l'objet_seurat
meta_coordinates <- cbind(objet_seurat@meta.data,x_image,y_image)
# Ajouter sur la gauche le terme "id" en titre de la 1ière colonne
meta_coordinates_scorer <- meta_coordinates %>% tibble::rownames_to_column("id") %>% left_join(scorer)
# Supprimer la colonne orig.ident qui ne contient que des caractères
meta_coordinates_scorer$orig.ident <- NULL
# Possibilité de récupérer la table avant le filtrage
# meta_coordinates_scorer %>% write_tsv("meta_coordinates_scorer.tsv")
```

3.3 Tri des Pathways

Choix des valeurs seuil

Dans cette étape, il faut filtrer les Pathways avec des valeurs déterminées. Le filtre doit correspondre à la biologie de l'objet_seurat utilisé.

```
A <- 5 # Expression dans au moins A cellules
B <- 10 # Score total = ou > à B
C <- 0.1 # Score moyen = ou > à C
D <- 0 # Dans les cellules qui l'expriment, Score total = ou > à D
E <- 0.5 # Dans les cellules qui l'expriment, Score moyen = ou > à E
```

Tri

```
# Filtre 0 : enlever les éventuelles valeurs NA, +Inf et -Inf.
# filtre_0 <- meta_coordinates_scorer
filtre_0 [is.na(filtre_0)] <- 0
filtre_0[filtre_0 == +Inf] <- 100
filtre_0[filtre_0 == -Inf] <- 0

# et enlever les Pathways qui ont un score d'expression nul pour toutes les cellules étudiées
filtre_0 <- filtre_0[, colSums(filtre_0 != 0) > 0]

# Filtre A : Expression dans au moins A cellules
filtre_A <- filtre_0[, colSums(filtre_0 != 0) >= A]

# Filtrage B : Score total d'au moins B
filtre_B <- filtre_A[, colSums(filtre_A[, sapply(filtre_A, is.numeric)]) >= B]

# Filtrage C : Score moyen d'au moins C
filtre_C <- filtre_B[, colMeans(filtre_B[sapply(filtre_B, is.numeric) > 0]) >= C]

# Filtrage D : Score total dans les cellules exprimant d'au moins D
filtre_D <- filtre_C[, colSums(filtre_C[, sapply(filtre_C, is.numeric)]) >= D]

# Filtrage E : Score moyen dans les cellules exprimant d'au moins E
filtre_E <- filtre_D[, colMeans(filtre_D[sapply(filtre_D, is.numeric) > 0]) >= E]

# Sortie du filtre :
sortie_filtre <- cbind(x_image, y_image, clusters, filtre_E)
sortie_filtre %>% write_tsv("sortie_filtre.tsv")
```

En sortie, on récupère un fichier sortie_filtre.tsv.

4. Single-Cell spatial Explorer

4.1 Préparer les fichiers

Dans spatial-main/data : effacer le .tsv de test, et placer sortie_filtre.tsv. Dans spatial-main/image : effacer l'image de test, et placer l'image hires ou lowres (au choix) issue du dossier Spatial imaging data.

4.2 Lancer le logiciel

4.2.1 Paramètres au 1ier démarrage

- Ne pas démarrer plusieurs instances du logiciel.
- Définir les préférences avant d'utiliser le logiciel : cliquer sur la roue.
- Si on change de tissu/coupe, il faudra refaire ces étapes et resélectionner chacun des éléments du menu déroulant même si ils portent le même nom.

Scaling factor :

```
# Afficher les facteurs d'échelle de l'objet_seurat
cat("Hires factor =", objet_seurat@images[[image]]@scale.factors[["hires"]], "\n")
# cat("Lowres factor =", objet_seurat@images[[image]]@scale.factors[["lowres"]], "\n")
```

Rotate : +90° pour les Data issues de 10x Genomics data.
X coordinates : x_image. Y coordinates : y_image. Cluster column : seurat_clusters. Cluster dots diameter : 10.

4.2.2 Affichage des clusters

Show Clusters ! Permet de vérifier si la table .tsv est conforme, sinon plantage !

4.2.3 Expression

Choisir une pathway, une palette, et cliquer sur “Plot Expression”. OU utiliser la fonction “Slide Show”.