Chapter 2 - Preprocessing and Annotation of Spatial Transcriptomics

The preprocessing of spatial transcriptomics followed a relative straight forward pipeline inspired by the Seurat spatial transcriptomics analysis pipeline.

Methods

Spatial Transcriptomics Data Processing and Integration Spatial transcriptomics data from Xenium and Visium platforms were processed using the Seurat (v5.0) pipeline. For **Xenium data**, a merged Seurat object was reconstructed and updated to Seurat v5 compatibility. Quality control (QC) metrics, including mitochondrial gene percentage (`percent.mt`) and log10 genes per UMI, were calculated without cell filtering to retain spatial context. Variable features were identified across all ~6,000 genes to account for Xenium's high molecular resolution. For **Visium data**, raw count matrices from eight samples were merged, normalized (LogNormalize, scale factor = 10,000), and QC metrics (mitochondrial/ribosomal gene percentages) were visualised. Variable features (n = 2,000) were selected using variance-stabilizing transformation (vst).

Batch Correction and Dimensionality Reduction Both datasets underwent harmonised dimensionality reduction by the `harmony` algorithm. Principal component analysis (PCA) was applied to variable features, followed by Harmony integration (λ = 1) to correct for batch effects across samples. Harmony-corrected embeddings were used for neighbourhood graph construction (k = 20 dimensions for Xenium, 15 for Visium, determined by `ElbowPlot`), Leiden clustering (resolutions: 0.1–0.7), and UMAP visualisation.

Cell Cycle Scoring Cell cycle phase (S/G2M) was inferred using Regev Lab marker genes. Scores were regressed out during scaling to minimise cell cycle-driven variation. Phase distributions were visualised via heatmaps to confirm balanced representation across samples.

Spatial Deconvolution with CARD Cell-type proportions were resolved using the CARD package, referencing a single-cell RNA-seq atlas (GSE180661). The reference was subsampled to include 40% of each cell type (excluding "Other") to ensure diversity. We extracted the raw count matrix and metadata (cell id, cell type, sample id) from the single cell dataset. We then extracted the raw-count matrix from spatial transcriptomics dataset as well as the spatial coordinates of each spot, note that the `row` and `col` columns were used instead of `imagerow` and `imagecol` . Everything was then integrated into CARD objects, and deconvolution was performed with default

parameters (minCountGene = 100, minCountSpot = 5). Proportions of key populations (e.g., ovarian cancer cells, fibroblasts) were mapped spatially and visualised using pie charts and heat gradients. The proportions were rounded to their 2nd digits to facilitate a more biological meaningful estimate of the cell identity, the logic behind is that in spatial transcriptomics, each spot approximately contains 4-5 cells, and so the expression of each spot is instead an average of these 4-5 cells –– and hence the necessity of deconvolution. Sometimes the spots have very similar proportion scores across two or three cell types –– like 0.34 for T cell and 0.35 for cancer cell. In this case, because our focus is on fibroblast, we want to accurately identify the fibroblasts in our cohort, we implemented a customised function to first check if the highest likelihood (proportion) of cell X is Fibroblast or not. If it is, the function labelled it as fibroblast and moved to next cell, if not, the function checks if that cell is likely cancer cell, if it is, the function return cancer cell. If it's not cancer cell, then the algorithm will pick the most likely cell, if there is a tie, then the function will randomly pick from the tie.

Integration of Xenium and Visium Datasets A merged dataset was created by intersecting ~6,000 genes common to both platforms. Cell types were also aligned to make sure a unified cell type labelling is implemented in merged dataset. The combined object underwent identical normalization, variable feature selection, and Harmony-based batch correction. Cell-type proportions from CARD were appended to metadata, enabling cross-platform comparison of spatial niches.

Codes:

Xenium Preprocess

```
## reconstructing seurat object –– updating to v5 object
assay5 <- as(spatial[["RNA"]], Class = "Assay5")
spatial <- CreateSeuratObject(assay5, meta.data = spatial@meta.data)

## Join layer to prevent partially merged seurat object
spatial <- JoinLayers(spatial)

## calculate basic quality control scores, note that for spatial
transcriptomics, NO CELLS WERE FILTERED
spatial[["percent.mt"]] <- PercentageFeatureSet(spatial, pattern =
"^MT-")
spatial[["mitoRatio"]] <- spatial@meta.data$percent.mt/100
spatial[["log10GenesPerUMI"]] <-
log10(spatial$nFeature_RNA)/log10(spatial$nCount_RNA)
```

```
lower resolution -- so about 6000 genes.
spatial <- FindVariableFeatures(spatial, selection.method = "vst",
nfeatures = nrow(spatial))

## Cell stages were added using marker downloaded from Seurat Git Hub
Repository
regev_lab_cell_cycle_genes <-
read.delim("~/Desktop/regev_lab_cell_cycle_genes.txt", header=F) %>%
unlist
s.genes <- intersect(rownames(spatial),
regev_lab_cell_cycle_genes[1:43])
g2m.genes <- intersect(rownames(spatial),
regev_lab_cell_cycle_genes[44:97])
spatial <- CellCycleScoring(object = spatial,
s.features = s.genes,
g2m.features = g2m.genes,
set.ident = TRUE)

## We then perform the regular reduction by PCA and Harmony to remove
sample batch effects
spatial <- RunPCA(spatial, features = VariableFeatures(object =
spatial))
spatial <- harmony::RunHarmony(spatial,
group.by.vars = "samples",
reduction = "pca",
assay.use = "RNA",
reduction.save = "harmony",
lambda = 1)

## Following the removal of batch effect, we calculated the
clustering and run UMAP
spatial <- FindNeighbors(spatial, dims = 1:20, reduction = "harmony")
spatial <- FindClusters(object = spatial, resolution = c(0.1, 0.3,
0.5, 0.7), verbose = TRUE)
spatial <- RunUMAP(spatial, dims = 1:20, reduction="harmony")
```

Visium Preprocess

```
library(Seurat)
library(dplyr)
library(magrittr)
library(tidyr)
library(ggplot2)
```

```
library(stringr)

## Load and Build Seurat Object
id = list.dirs("data/spatial/GSE211956/", full.names = F, recursive =
F)
lst = list()
for (i in 1:length(id)){
temp = Seurat::Read10X(data.dir = paste0("data/spatial/GSE211956/",
id[i], "/count_matrix"))
lst[[i]] = CreateSeuratObject(counts = temp)
}
spatial = merge(lst[[1]], y =
c(lst[[2]],lst[[3]],lst[[4]],lst[[5]],lst[[6]],lst[[7]],lst[[8]]),
add.cell.ids = c(id[1],id[2],id[3],id[4],id[5],id[6],id[7],id[8]),
project = "HGSOC_spatial")
rm(lst)
rm(temp)

## Some small modifications
spatial = JoinLayers(spatial)
spatial$orig.ident =
unname(unlist(lapply(strsplit(rownames(spatial@meta.data), split =
"_"), function(x){x[[1]]})))
Idents(spatial) = spatial$orig.ident

## Add QC Metrics
spatial[["percent.mito"]] = PercentageFeatureSet(spatial, pattern =
"^MT-")
spatial[["percent.ribo"]] = PercentageFeatureSet(spatial, pattern =
"^RPL|^RPS|^MRP-")
spatial[["log10GenesPerUMI"]] =
log10(spatial$nFeature_RNA)/log10(spatial$nCount_RNA)
VlnPlot(spatial, features = c("nCount_RNA", "nFeature_RNA",
"percent.ribo", "percent.mito"), ncol = 1, pt.size = 0)
ggsave("plot/merged_spatial/preprocess/QC.png", width = 4, height =
12, units = "in", dpi = 600)
pie(table(spatial$orig.ident))

## Update Metadata, add Sample Names
spatial$cells = rownames(spatial)
spatial$samples = spatial$orig.ident

## Normalize, no subset because we're using spatial data
```

```
spatial = NormalizeData(spatial, normalization.method =
"LogNormalize", scale.factor = 10000)
spatial = FindVariableFeatures(spatial, selection.method = "vst",
nfeatures = 2000)
p1 = VariableFeaturePlot(spatial)
LabelPoints(plot = p1, points = head(VariableFeatures(spatial), 10),
repel = TRUE)

## Add Cell Cycle
regev_lab_cell_cycle_genes =
read.delim("data/regev_lab_cell_cycle_genes.txt", header=F) %>%
unlist
s.genes = regev_lab_cell_cycle_genes[1:43]
g2m.genes = regev_lab_cell_cycle_genes[44:97]
spatial = CellCycleScoring(object = spatial,
s.features = s.genes,
g2m.features = g2m.genes,
set.ident = TRUE)
cell_cycles = table(spatial$orig.ident, spatial$Phase) %>%
as.data.frame.matrix()
cell_cycles = as.data.frame(t(apply(cell_cycles, 1, function(x) x /
sum(x) * 100)))
pheatmap::pheatmap(cell_cycles, cluster_cols = F, cluster_rows = F)

## Scale, run PCA, run UMAP, and find Cluster
spatial = ScaleData(spatial, features = rownames(spatial),
vars.to.regress = c("nCount_RNA", "percent.ribo", "percent.mito",
"S.Score", "G2M.Score"))
spatial = RunPCA(spatial, features = VariableFeatures(object =
spatial))

## Run Harmony
spatial = harmony::RunHarmony(spatial,
group.by.vars = "samples",
reduction = "pca",
assay.use = "RNA",
reduction.save = "harmony",
lambda = 1)
spatial = FindNeighbors(spatial, dims = 1:15, reduction = "harmony")
spatial = FindClusters(object = spatial, resolution = c(0.1, 0.3,
0.5, 0.7), verbose = TRUE)
spatial = RunUMAP(spatial, dims = 1:15, reduction="harmony")
```

Visium Annotation by CARD

```
library(Seurat)
library(dplyr)
library(magrittr)
library(tidyr)
library(ggplot2)
library(stringr)
library(CARD)
library(MuSiC)
library(RColorBrewer)

set.seed(20020208)

directories <- list.dirs("~/Desktop/YAP1/data/spatial/GSE211956",
recursive = F)

# ## Read in Reference
# if(!file.exists("data/reference/CARD_REF.RData")){
# sc_count <- readRDS("data/reference/GSE180661.rds")
# sc_meta <- read.delim("data/reference/refDB/9606_map.tsv") %>%
# dplyr::select(cell_id, cell_type, patient_id) %>%
# group_by(cell_type) %>% slice_sample(prop = 0.4) %>% ungroup %>%
# dplyr::filter(cell_type != "Other") %>% as.data.frame() ## Slice
40%
# rownames(sc_meta) <- sc_meta$cell_id
# sc_count <- sc_count[, sc_meta$cell_id]
# } else {
# load("data/reference/CARD_REF.RData")
# }

## Create CARD object for each sample ------------------------------
------------
for(i in directories){
setwd(i)
## Extract Spatial Objects
spatial <- readRDS("output/seurat.rds")
spatial_count <- spatial@assays$Spatial$counts
spatial_location <- spatial@images$slice1@coordinates %>%
dplyr::select(row, col)
colnames(spatial_location) <- c("x", "y")
## Build CARD
CARD_obj = createCARDObject(sc_count = sc_count,
sc_meta = sc_meta,
```

```
spatial_count = spatial_count,
spatial_location = spatial_location,
ct.varname = "cell_type",
ct.select = unique(sc_meta$cell_type),
sample.varname = "patient_id",
minCountGene = 100,
minCountSpot = 5)
CARD_obj = CARD_deconvolution(CARD_object = CARD_obj)
## Plot
colors = c("grey","grey","grey","#F0027F", "grey", "#7FC97F",
"#386CB0", "grey", "#FFD92F")
p1 <- CARD.visualize.pie(proportion = CARD_obj@Proportion_CARD,
spatial_location = CARD_obj@spatial_location,
colors = colors, radius = 0.7)
png("plots/card-deconvolute.png", width = 10, height = 10, units =
"in", res = 300)
print(p1)
dev.off()
p2 <- CARD.visualize.prop(proportion = CARD_obj@Proportion_CARD,
spatial_location = CARD_obj@spatial_location,
ct.visualize = c("Ovarian.cancer.cell", "Fibroblast"),
colors = c("lightblue","lightyellow","red"),
NumCols = 4,
pointSize = 0.5)
png("plots/card-deconvolute-cancer&fibro.png", width = 10, height =
5, units = "in", res = 300)
print(p2)
dev.off()
## save CARD
saveRDS(CARD_obj, "output/CARD.rds")
## Clean up
rm(list = setdiff(ls(), c("sc_count", "sc_meta", "directories")))
gc()
}


## Work on Color ----------------------------------------------------
------
i <- directories[[8]]
# for(i in directories){
setwd(i)
spatial <- readRDS("output/seurat.rds")
CARD <- readRDS("output/CARD.rds")
```

```
CARD <- CARD@Proportion_CARD %>% as.data.frame()
CARD.round <- round(CARD, digits = 2)
CARD.round <- CARD.round[rownames(spatial@meta.data), ]
colnames(CARD.round) <- paste0("CARD.", colnames(CARD.round))

colour <- brewer.pal(n = ncol(CARD.round), name = "Set3")
names(colour) <- colnames(CARD.round)

spatial@meta.data <- merge(spatial@meta.data, CARD.round, by =
"row.names") %>%
tibble::column_to_rownames(var = "Row.names")
SpatialFeaturePlot(spatial, features = colnames(CARD.round),
image.alpha = 0,
alpha = 1, pt.size.factor = 2.4, keep.scale = "all")
ggsave("plots/card-deconvolute-mapped-selected.png", width = 12,
height = 12, units = "in", dpi = 600)
saveRDS(spatial, "output/labelled.rds")
# }
```

Visium Refined Annotation by In-House Function

```
library(Seurat)
library(dplyr)
library(magrittr)
library(tidyr)
library(ggplot2)
library(stringr)
library(CARD)
library(MuSiC)
library(RColorBrewer)

set.seed(20020208)

directories <- list.dirs("~/Desktop/YAP1/data/spatial/GSE211956",
recursive = F)

get_cell_type <- function(x) {
max_value <- max(x)
max_indices <- which(x == max_value)
max_names <- names(card_columns)[max_indices]
# Check for ties and apply rules
if (length(max_names) > 1) {
if (any(grepl("Fibroblast", max_names))) {
```

```
return("CARD.Fibroblast")
} else if (any(grepl("cancer.cell", max_names))) {
return(max_names[grepl("cancer.cell", max_names)][1]) # Pick the
first cancer cell type
} else {
return(sample(max_names, 1) ) # Randomly pick one of the tied cell
types
}
}
return(max_names[1]) # No ties, return the single max name
}


## Work on Color ----------------------------------------------------
------
for(i in directories){
setwd(i)
spatial <- readRDS("output/labelled.rds")
## View the Distribution of CARD proportion
plot_df <- spatial@meta.data %>% select(starts_with("CARD.")) %>%
pivot_longer(cols = everything(), names_to = "Cell_Type", values_to =
"Likelihood")
ggplot(plot_df, aes(x = Cell_Type, y = Likelihood)) + geom_boxplot()
+
labs(title = "Boxplot of Cell Type Likelihoods", x = "Cell Type", y =
"Likelihood") +
theme_bw() + theme(axis.text.x = element_text(angle = 45, hjust = 1))
ggsave("plots/dist-CARD-proportion.png", width = 4, height = 4, units
= "in", dpi = 600)
## add label
card_cols <- grepl("CARD", colnames(spatial@meta.data))
spatial@meta.data[, card_cols] <- round(spatial@meta.data[,
card_cols], digits = 2)
card_columns <- spatial@meta.data %>% select(starts_with("CARD"))
spatial@meta.data <- spatial@meta.data %>% mutate(cell_type =
apply(card_columns, 1, get_cell_type))
## Plot
colour <- brewer.pal(9, name = "Pastel1")
names(colour) <- c("CARD.Fibroblast", "CARD.Ovarian.cancer.cell",
"CARD.Endothelial.cell",
"CARD.T.cell", "CARD.B.cell", "CARD.Myeloid.cell",
"CARD.Plasma.cell")
SpatialDimPlot(spatial, group.by = "cell_type", pt.size.factor = 2.6,
image.alpha = 0, cols = colour)
```

```
ggsave("plots/card-deconvolute-mapped.png", width = 6, height = 4,
units = "in", dpi = 600)

## Save RDS
saveRDS(spatial, "output/labelled.rds")
}


## Extract metadata
meta <- list()

meta <- lapply(directories, function(x){
spatial <- readRDS(file.path(x, "output/labelled.rds"))
spatial <- spatial@meta.data
spatial$cells <- paste0(spatial$sample, "_", spatial$cells)
rownames(spatial) <- spatial$cells
return(spatial)
})

meta_df <- do.call(rbind, meta)
write.csv(meta_df, "data/spatial/collapsed/labelled_metadata.csv")
```

Merging Visium and Xenium

```
library(Seurat)
library(dplyr)
library(magrittr)
library(tidyr)
library(ggplot2)
library(stringr)
library(scales)
setwd("~/Desktop/YAP1/")

## Load in Data and Cleave other stuff ------------------------------
-----------
visium <- readRDS("data/spatial/collapsed/spatial.rds")
xenium <- readRDS("~/Desktop/Collection/spatial-
seq/10xXenium/Seurat/ST_xenium1_so.rds")
genes <- intersect(rownames(visium), rownames(xenium))
counts <- visium@assays$RNA$counts[genes, ]
visium <- CreateSeuratObject(counts, meta.data = visium@meta.data)
visium <- JoinLayers(visium)
```

```
counts <- xenium@assays$RNA$counts[genes, ]
xenium <- CreateSeuratObject(counts, meta.data = xenium@meta.data)
xenium <- JoinLayers(xenium)
rm(counts)

## Prepare Metadata ------------------------------------------------
-----------
xenium@meta.data$cells <- rownames(xenium@meta.data)
xenium@meta.data$sample <- xenium@meta.data$samples
visium_meta <-
read.csv("data/spatial/collapsed/labelled_metadata.csv", row.names =
1)
visium_meta$cell.types <- visium_meta$cell_type
visium_meta$nCount_RNA <- visium_meta$nCount_Spatial
visium_meta$nFeature_RNA <- visium_meta$nFeature_Spatial
visium_meta$YAP1 <- NULL
visium_meta$patients <- visium_meta$sample
meta_cols <- intersect(colnames(xenium@meta.data),
colnames(visium_meta))
visium_meta <- visium_meta[, meta_cols]
xenium_meta <- xenium@meta.data[, meta_cols]
metadata <- rbind(visium_meta, xenium_meta)

## Merge Two Objects -----------------------------------------------
---
merged <- merge(visium, xenium, project = "merged")
cells_id <- intersect(rownames(metadata), colnames(merged))
merged <- subset(merged, cells = cells_id)
metadata <- metadata[colnames(merged), ]
merged@meta.data <- metadata
merged@meta.data$cell.types <- gsub("CARD.", "",
merged@meta.data$cell.types)
merged@meta.data$cell.types <- ifelse(merged@meta.data$cell.types ==
"T.cell", "TNK.cell", merged@meta.data$cell.types)
merged@meta.data$cell.types <- ifelse(merged@meta.data$cell.types ==
"Myeloid.cell", "Monocyte", merged@meta.data$cell.types)
merged@meta.data$cell.types <- ifelse(merged@meta.data$cell.types ==
"Malignant", "Ovarian.cancer.cell", merged@meta.data$cell.types)
merged@meta.data$cell.types <- ifelse(merged@meta.data$cell.types ==
"Endothelial.cell", "Endothelial", merged@meta.data$cell.types)
merged <- JoinLayers(merged)
saveRDS(merged, "data/merged_visium_xenium.rds")
```

```
## Preprocessing ------------------------------------------------------
-----------
merged[["percent.mt"]] <- PercentageFeatureSet(merged, pattern =
"^MT-")
merged <- NormalizeData(merged, normalization.method =
"LogNormalize", scale.factor = 10000)
merged <- FindVariableFeatures(merged, selection.method = "vst",
nfeatures = 3000)
regev_lab_cell_cycle_genes <-
read.delim("data/regev_lab_cell_cycle_genes.txt", header=F) %>%
unlist
s.genes <- regev_lab_cell_cycle_genes[1:43]
g2m.genes <- regev_lab_cell_cycle_genes[44:97]
merged <- CellCycleScoring(object = merged, s.features = s.genes,
g2m.features = g2m.genes, set.ident = TRUE)
merged <- ScaleData(merged, vars.to.regress = c("Phase",
"orig.ident"), features = VariableFeatures(merged))
merged <- RunPCA(merged, features = VariableFeatures(merged))
merged <- harmony::RunHarmony(merged,
group.by.vars = "sample",
reduction = "pca",
assay.use = "RNA",
reduction.save = "harmony",
lambda = 1)
merged <- FindNeighbors(merged, dims = 1:10, reduction = "harmony")
merged <- FindClusters(merged, resolution = 0.5, verbose = TRUE)
merged <- RunUMAP(merged, dims = 1:10, reduction="harmony")
DimPlot(merged) + NoLegend()

## Save Object
saveRDS(merged, "data/merged_visium_xenium.rds")
```