*#!/usr/bin/env nextflow*

*/\**

\* Single-Cell RNA-seq Analysis Pipeline with Cell Ranger and Seurat

\* Author: Your Name

\* Version: 1.0.0

*\*/*

nextflow.enable.dsl = 2

*// Pipeline parameters*

params {

*// Input parameters*

input\_dir = null

samplesheet = null

fastq\_dir = null

*// Reference genome*

reference\_genome = null

*// Cell Ranger parameters*

expect\_cells = 5000

localcores = 8

localmem = 64

chemistry = "auto"

*// Analysis parameters*

min\_cells = 3

min\_features = 200

max\_features = 5000

max\_mt\_percent = 20

min\_umi = 500

max\_umi = 25000

resolution = 0.5

n\_variable\_features = 2000

*// Output directory*

outdir = "./results"

*// Resource parameters*

max\_memory = '128.GB'

max\_cpus = 16

max\_time = '240.h'

*// Skip steps*

skip\_cellranger = false

skip\_seurat = false

skip\_integration = false

*// Help*

help = false

}

*// Help message*

def helpMessage() {

log.info"""

Single-Cell RNA-seq Analysis Pipeline

=====================================

Usage:

nextflow run main.nf --samplesheet samples.csv --reference\_genome /path/to/reference --outdir results

Required Arguments:

--samplesheet Path to sample sheet CSV file

--reference\_genome Path to Cell Ranger reference genome directory

--outdir Output directory for results

Optional Arguments:

--fastq\_dir Directory containing FASTQ files (if not in samplesheet)

--expect\_cells Expected number of cells per sample (default: 5000)

--localcores Number of cores for Cell Ranger (default: 8)

--localmem Memory in GB for Cell Ranger (default: 64)

--chemistry Chemistry version for Cell Ranger (default: auto)

--resolution Clustering resolution for Seurat (default: 0.5)

--skip\_cellranger Skip Cell Ranger processing

--skip\_seurat Skip Seurat analysis

--skip\_integration Skip sample integration

--help Show this help message

Sample Sheet Format:

sample\_id,fastq\_path,sample\_name,condition

Sample1,/path/to/fastqs,Sample1,Control

Sample2,/path/to/fastqs,Sample2,Treatment

""".stripIndent()

}

*// Show help message*

if (params.help) {

helpMessage()

exit 0

}

*// Parameter validation*

if (!params.samplesheet) {

error "Please provide a sample sheet with --samplesheet"

}

if (!params.reference\_genome) {

error "Please provide a reference genome with --reference\_genome"

}

if (!file(params.samplesheet).exists()) {

error "Sample sheet file not found: ${params.samplesheet}"

}

if (!file(params.reference\_genome).exists()) {

error "Reference genome directory not found: ${params.reference\_genome}"

}

*/\**

\* WORKFLOW DEFINITION

*\*/*

workflow {

*// Parse sample sheet*

ch\_samples = parse\_samplesheet(params.samplesheet)

*// Cell Ranger workflow*

if (!params.skip\_cellranger) {

CELLRANGER\_COUNT(ch\_samples, params.reference\_genome)

ch\_cellranger\_results = CELLRANGER\_COUNT.out.results

ch\_metrics = CELLRANGER\_COUNT.out.metrics

} else {

*// If skipping Cell Ranger, assume results already exist*

ch\_cellranger\_results = ch\_samples.map { sample ->

def result\_dir = file("${params.outdir}/cellranger/${sample.sample\_id}")

[sample, result\_dir]

}

ch\_metrics = Channel.empty()

}

*// Seurat analysis workflow*

if (!params.skip\_seurat) {

*// Individual sample analysis*

SEURAT\_QC\_AND\_FILTER(ch\_cellranger\_results)

ch\_filtered\_objects = SEURAT\_QC\_AND\_FILTER.out.filtered\_objects

ch\_qc\_plots = SEURAT\_QC\_AND\_FILTER.out.qc\_plots

*// Integration (if multiple samples and not skipped)*

if (!params.skip\_integration) {

ch\_all\_samples = ch\_filtered\_objects.collect()

SEURAT\_INTEGRATION(ch\_all\_samples)

ch\_integrated = SEURAT\_INTEGRATION.out.integrated\_object

*// Downstream analysis on integrated data*

SEURAT\_CLUSTERING(ch\_integrated)

SEURAT\_MARKERS(SEURAT\_CLUSTERING.out.clustered\_object)

SEURAT\_VISUALIZATION(SEURAT\_CLUSTERING.out.clustered\_object, SEURAT\_MARKERS.out.markers)

} else {

*// Process each sample individually*

SEURAT\_CLUSTERING(ch\_filtered\_objects)

SEURAT\_MARKERS(SEURAT\_CLUSTERING.out.clustered\_object)

SEURAT\_VISUALIZATION(SEURAT\_CLUSTERING.out.clustered\_object, SEURAT\_MARKERS.out.markers)

}

}

*// Generate final report*

if (!params.skip\_cellranger && !params.skip\_seurat) {

GENERATE\_REPORT(

ch\_metrics.collect(),

ch\_qc\_plots.collect(),

SEURAT\_VISUALIZATION.out.plots.collect()

)

}

}

*/\**

\* FUNCTIONS

*\*/*

def parse\_samplesheet(samplesheet\_path) {

return Channel

.fromPath(samplesheet\_path)

.splitCsv(header: true)

.map { row ->

def sample = [:]

sample.sample\_id = row.sample\_id

sample.fastq\_path = row.fastq\_path ?: params.fastq\_dir

sample.sample\_name = row.sample\_name ?: row.sample\_id

sample.condition = row.condition ?: "Unknown"

return sample

}

}

*/\**

\* PROCESSES

*\*/*

process CELLRANGER\_COUNT {

tag "$sample.sample\_id"

label 'process\_high'

publishDir "${params.outdir}/cellranger", mode: 'copy'

input:

val sample

path reference\_genome

output:

tuple val(sample), path("${sample.sample\_id}"), emit: results

path "${sample.sample\_id}/outs/metrics\_summary.csv", emit: metrics

script:

"""

cellranger count \\

--id=${sample.sample\_id} \\

--transcriptome=${reference\_genome} \\

--fastqs=${sample.fastq\_path} \\

--sample=${sample.sample\_id} \\

--expect-cells=${params.expect\_cells} \\

--localcores=${params.localcores} \\

--localmem=${params.localmem} \\

--chemistry=${params.chemistry} \\

--disable-ui

"""

}

process SEURAT\_QC\_AND\_FILTER {

tag "$sample.sample\_id"

label 'process\_medium'

publishDir "${params.outdir}/seurat/qc", mode: 'copy'

input:

tuple val(sample), path(cellranger\_results)

output:

tuple val(sample), path("${sample.sample\_id}\_filtered.rds"), emit: filtered\_objects

path "${sample.sample\_id}\_qc\_plots.pdf", emit: qc\_plots

script:

"""

#!/usr/bin/env Rscript

library(Seurat)

library(dplyr)

library(ggplot2)

library(patchwork)

# Load 10X data

data <- Read10X(data.dir = "${cellranger\_results}/outs/filtered\_feature\_bc\_matrix")

# Create Seurat object

seurat\_obj <- CreateSeuratObject(

counts = data,

project = "${sample.sample\_id}",

min.cells = ${params.min\_cells},

min.features = ${params.min\_features}

)

# Add metadata

seurat\_obj\$sample <- "${sample.sample\_id}"

seurat\_obj\$condition <- "${sample.condition}"

# Calculate QC metrics

seurat\_obj[["percent.mt"]] <- PercentageFeatureSet(seurat\_obj, pattern = "^MT-")

seurat\_obj[["percent.ribo"]] <- PercentageFeatureSet(seurat\_obj, pattern = "^RP[SL]")

# Create QC plots

p1 <- VlnPlot(seurat\_obj, features = c("nFeature\_RNA", "nCount\_RNA", "percent.mt"), ncol = 3)

p2 <- FeatureScatter(seurat\_obj, feature1 = "nCount\_RNA", feature2 = "percent.mt")

p3 <- FeatureScatter(seurat\_obj, feature1 = "nCount\_RNA", feature2 = "nFeature\_RNA")

combined\_plot <- p1 / (p2 | p3)

ggsave("${sample.sample\_id}\_qc\_plots.pdf", combined\_plot, width = 15, height = 10)

# Filter cells

filtered\_obj <- subset(seurat\_obj,

subset = nFeature\_RNA > ${params.min\_features} &

nFeature\_RNA < ${params.max\_features} &

percent.mt < ${params.max\_mt\_percent} &

nCount\_RNA > ${params.min\_umi} &

nCount\_RNA < ${params.max\_umi}

)

cat("Original cells:", ncol(seurat\_obj), "\\n")

cat("Filtered cells:", ncol(filtered\_obj), "\\n")

# Save filtered object

saveRDS(filtered\_obj, "${sample.sample\_id}\_filtered.rds")

"""

}

process SEURAT\_INTEGRATION {

label 'process\_high'

publishDir "${params.outdir}/seurat/integration", mode: 'copy'

input:

path filtered\_objects

output:

path "integrated\_seurat.rds", emit: integrated\_object

script:

"""

#!/usr/bin/env Rscript

library(Seurat)

library(dplyr)

# Load all filtered objects

object\_files <- list.files(pattern = "\_filtered.rds")

seurat\_list <- lapply(object\_files, readRDS)

names(seurat\_list) <- gsub("\_filtered.rds", "", object\_files)

if (length(seurat\_list) > 1) {

# Normalize and find variable features

seurat\_list <- lapply(seurat\_list, function(x) {

x <- NormalizeData(x)

x <- FindVariableFeatures(x, selection.method = "vst", nfeatures = ${params.n\_variable\_features})

return(x)

})

# Find integration anchors

anchors <- FindIntegrationAnchors(object.list = seurat\_list, dims = 1:30)

# Integrate data

integrated <- IntegrateData(anchorset = anchors, dims = 1:30)

DefaultAssay(integrated) <- "integrated"

} else {

# Single sample

integrated <- seurat\_list[[1]]

integrated <- NormalizeData(integrated)

integrated <- FindVariableFeatures(integrated, selection.method = "vst",

nfeatures = ${params.n\_variable\_features})

}

# Save integrated object

saveRDS(integrated, "integrated\_seurat.rds")

"""

}

process SEURAT\_CLUSTERING {

tag "$sample\_info"

label 'process\_medium'

publishDir "${params.outdir}/seurat/clustering", mode: 'copy'

input:

tuple val(sample), path(seurat\_object)

output:

tuple val(sample), path("\*\_clustered.rds"), emit: clustered\_object

script:

sample\_info = sample ? sample.sample\_id : "integrated"

"""

#!/usr/bin/env Rscript

library(Seurat)

# Load Seurat object

seurat\_obj <- readRDS("${seurat\_object}")

# Scale data and run PCA

seurat\_obj <- ScaleData(seurat\_obj)

seurat\_obj <- RunPCA(seurat\_obj, features = VariableFeatures(object = seurat\_obj))

# Determine number of PCs

pct <- seurat\_obj[["pca"]]@stdev / sum(seurat\_obj[["pca"]]@stdev) \* 100

cumu <- cumsum(pct)

co1 <- which(cumu > 90 & pct < 5)[1]

co2 <- sort(which((pct[1:length(pct) - 1] - pct[2:length(pct)]) > 0.1), decreasing = T)[1] + 1

pcs <- min(co1, co2, 30)

# Clustering

seurat\_obj <- FindNeighbors(seurat\_obj, dims = 1:pcs)

seurat\_obj <- FindClusters(seurat\_obj, resolution = ${params.resolution})

# UMAP and tSNE

seurat\_obj <- RunUMAP(seurat\_obj, dims = 1:pcs)

seurat\_obj <- RunTSNE(seurat\_obj, dims = 1:pcs)

# Save clustered object

saveRDS(seurat\_obj, "${sample\_info}\_clustered.rds")

"""

}

process SEURAT\_MARKERS {

tag "$sample\_info"

label 'process\_high'

publishDir "${params.outdir}/seurat/markers", mode: 'copy'

input:

tuple val(sample), path(clustered\_object)

output:

tuple val(sample), path("\*\_markers.csv"), emit: markers

script:

sample\_info = sample ? sample.sample\_id : "integrated"

"""

#!/usr/bin/env Rscript

library(Seurat)

library(dplyr)

# Load clustered object

seurat\_obj <- readRDS("${clustered\_object}")

# Set assay to RNA for marker detection

DefaultAssay(seurat\_obj) <- "RNA"

# Find all markers

all\_markers <- FindAllMarkers(seurat\_obj,

only.pos = TRUE,

min.pct = 0.25,

logfc.threshold = 0.25)

# Save markers

write.csv(all\_markers, "${sample\_info}\_markers.csv", row.names = FALSE)

"""

}

process SEURAT\_VISUALIZATION {

tag "$sample\_info"

label 'process\_medium'

publishDir "${params.outdir}/seurat/plots", mode: 'copy'

input:

tuple val(sample), path(clustered\_object)

tuple val(sample\_markers), path(markers\_file)

output:

path "\*.pdf", emit: plots

script:

sample\_info = sample ? sample.sample\_id : "integrated"

"""

#!/usr/bin/env Rscript

library(Seurat)

library(ggplot2)

library(dplyr)

library(patchwork)

# Load data

seurat\_obj <- readRDS("${clustered\_object}")

markers <- read.csv("${markers\_file}")

# UMAP plots

p1 <- DimPlot(seurat\_obj, reduction = "umap", label = TRUE, pt.size = 0.5) +

ggtitle("Clusters")

if ("sample" %in% colnames(seurat\_obj@meta.data)) {

p2 <- DimPlot(seurat\_obj, reduction = "umap", group.by = "sample", pt.size = 0.5) +

ggtitle("Samples")

umap\_combined <- p1 | p2

} else {

umap\_combined <- p1

}

ggsave("${sample\_info}\_umap\_plots.pdf", umap\_combined, width = 16, height = 8)

# Feature plots for top markers

if (nrow(markers) > 0) {

top\_markers <- markers %>%

group\_by(cluster) %>%

top\_n(n = 2, wt = avg\_log2FC) %>%

head(9)

if (nrow(top\_markers) > 0) {

feature\_plot <- FeaturePlot(seurat\_obj, features = top\_markers\$gene, ncol = 3)

ggsave("${sample\_info}\_feature\_plots.pdf", feature\_plot, width = 15, height = 15)

}

# Heatmap of top markers

top5\_markers <- markers %>%

group\_by(cluster) %>%

top\_n(n = 5, wt = avg\_log2FC)

if (nrow(top5\_markers) > 0) {

heatmap\_plot <- DoHeatmap(seurat\_obj, features = top5\_markers\$gene) +

theme(axis.text.y = element\_text(size = 6))

ggsave("${sample\_info}\_heatmap.pdf", heatmap\_plot, width = 15, height = 12)

}

}

# QC by cluster

qc\_plot <- VlnPlot(seurat\_obj, features = c("nFeature\_RNA", "nCount\_RNA", "percent.mt"),

ncol = 3, pt.size = 0)

ggsave("${sample\_info}\_qc\_by\_cluster.pdf", qc\_plot, width = 15, height = 8)

"""

}

process GENERATE\_REPORT {

label 'process\_low'

publishDir "${params.outdir}/report", mode: 'copy'

input:

path metrics\_files

path qc\_plots

path analysis\_plots

output:

path "pipeline\_report.html"

script:

"""

#!/usr/bin/env Rscript

library(rmarkdown)

library(knitr)

# Create report template

report\_template <- '

---

title: "Single-Cell RNA-seq Analysis Report"

date: "`r Sys.Date()`"

output:

html\_document:

toc: true

toc\_float: true

theme: bootstrap

---

# Pipeline Summary

This report summarizes the results of the single-cell RNA-seq analysis pipeline.

## Parameters Used

- Expected cells per sample: ${params.expect\_cells}

- Clustering resolution: ${params.resolution}

- Min features per cell: ${params.min\_features}

- Max mitochondrial percentage: ${params.max\_mt\_percent}

## Sample Metrics

```{r metrics, echo=FALSE}

# Load and display metrics

metrics\_files <- list.files(pattern = "metrics\_summary.csv", full.names = TRUE)

if (length(metrics\_files) > 0) {

metrics\_list <- lapply(metrics\_files, read.csv)

names(metrics\_list) <- gsub("\_metrics\_summary.csv", "", basename(metrics\_files))

print("Sample processing completed successfully")

}

```

## Analysis Results

Analysis plots and results are available in the respective output directories.

'

# Write template

writeLines(report\_template, "report.Rmd")

# render report

render("report.Rmd", output\_file = "pipeline\_report.html")

"""

}