1-Preprocessing_workflow

Joan Abinet

2025-06-25 14:11:26 +0200

Contents

Introduction

Lung endothelial cells (CD45-, EpCAM-, CD31+) were FACS-sorted as living singlet from lung single-cell suspensions pooled from 4 mice per condition: control, d3 a-Ly6G, d3 a-CXCR2, d14 a-Ly6G, d14 a-CXCR2. For each sample, an aliquot of Trypan blue-treated cells was examined under the microscope for counting, viability and aggregate assessment following FACS sorting. Viability was above 90% for all samples and no aggregates were observed. Cell preparations were centrifuged and pellets were resuspended in calcium- and magnesium-free PBS containing 0.4 mg ml-1 UltraPure BSA (Thermo Fisher Scientific)

Single-cell RNA sequencing was performed using the 10x Genomics Chromium Next GEM Single Cell 3' v3.1 kit. A total of 13,000 cells were loaded per sample, with eight samples processed on a single chip using the Chromium Controller instrument. The manufacturer's protocol was strictly followed throughout the procedure. cDNA quality and fragment size distribution were assessed using the Agilent High Sensitivity DNA Kit on a Fragment Analyzer. Final libraries were quality-checked on a QIAxcel system (Qiagen), quantified by qPCR (Roche), and pooled in equimolar amounts.

Sequencing was performed on an Illumina NovaSeq 6000 system using an S4 flow cell with paired-end 2×150 bp reads (Read1: 150 cy, read2: 150 cy, index1:10cy, index2: 10cy), targeting a depth of 30,000 reads per cell for each sample. Raw sequencing data were demultiplexed using sample-specific indexes, quality-filtered, and converted into FASTQ files using bcl2fastq. Sequencing data quality was evaluated per sample using FastQC, and summarized across all samples using MultiQC.

The Cell Ranger (v6.1.2) application (10x Genomics) was then used to demultiplex the BCL files into FASTQ files (cellranger mkfastq), to perform alignment (using Cell Ranger human genome references 6.1.2 GRCm38/release 102), filtering and unique molecular identifier counting and to produce gene-barcode matrices (cellranger count).

Loading packages

```
suppressMessages({
    library(dplyr)
    library(Seurat)
    library(patchwork)
    library(ggplot2)
    library(formatR)
    library(SingleR)
})
```

Loading data

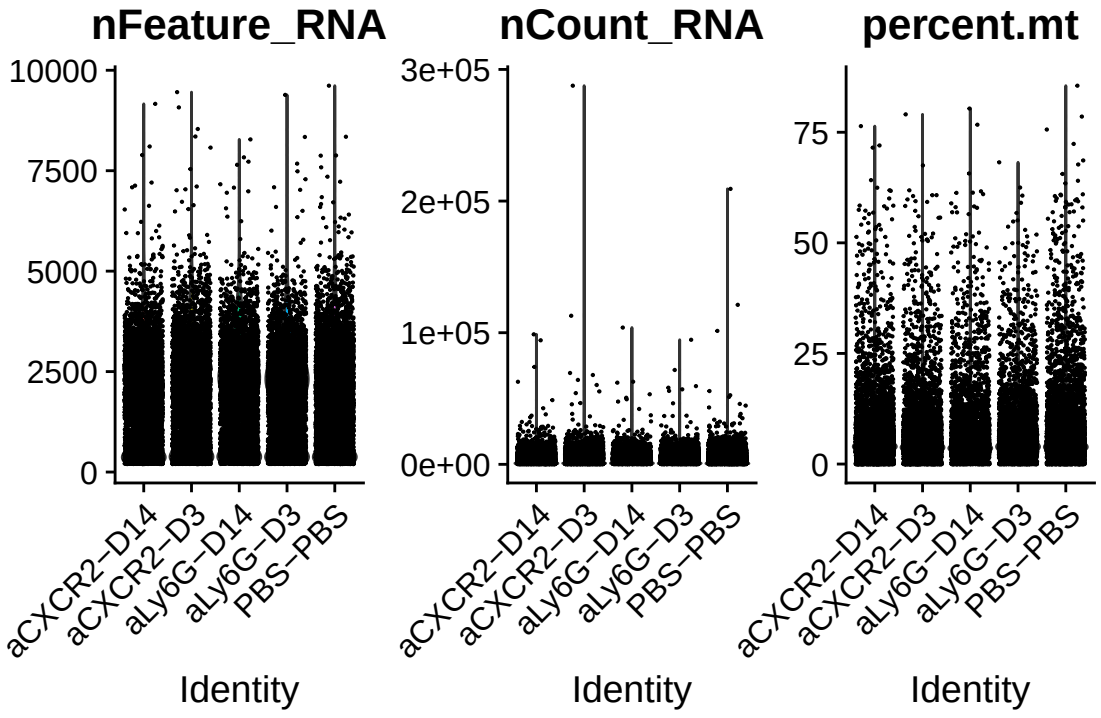
The count files can be download from GEO

```
# Specify your 10x directory
all_dirs <- list.dirs(path = "Count", full.names = TRUE, recursive = F)

list_sample_name <- c("aCXCR2-D14", "aCXCR2-D3", "aLy6G-D14", "aLy6G-D3",
    "PBS-PBS")

list_sample <- list()
for (i in 1:length(all_dirs)) {

    Seq_raw_file <- Read10X(data.dir = all_dirs[i])
    Seurat_file <- CreateSeuratObject(counts = Seq_raw_file, project = list_sample_name[i],
        min.cells = 3, min.features = 200)
```

```
    list_sample <- append(list_sample, Seurat_file)
}
list_sample

endothelial_cells <- merge(list_sample[[1]], y = list_sample[-1],
    add.cell.ids = c("1", "2", "3", "4", "5"), project = "Endothelial_Cells")
```

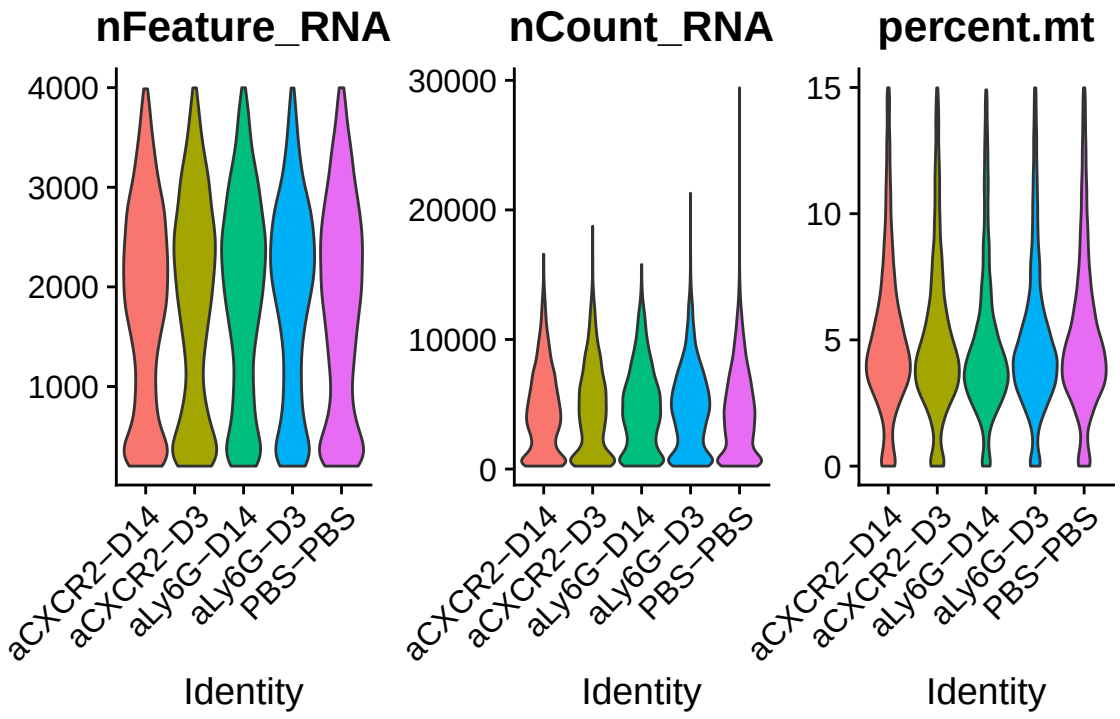
Quality Control

```
endothelial_cells[["percent.mt"]] <- PercentageFeatureSet(endothelial_cells,
    pattern = "^mt-")
VlnPlot(endothelial_cells, features = c("nFeature_RNA", "nCount_RNA",
    "percent.mt"), ncol = 3)
```



Filtering

```
endothelial_cells <- subset(endothelial_cells, subset = nFeature_RNA >
    200 & nFeature_RNA < 4000 & percent.mt < 15)
p1 <- VlnPlot(endothelial_cells, features = c("nFeature_RNA", "nCount_RNA",
    "percent.mt"), ncol = 3, pt.size = 0)
p1
```

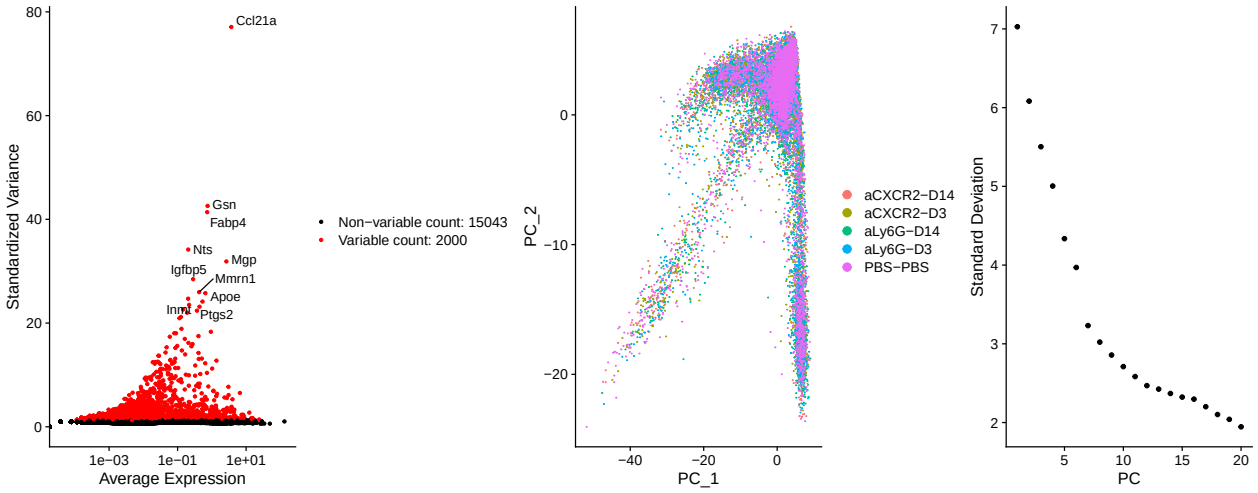


Pre-processing workflow

```
# Data Normalization
endothelial_cells <- NormalizeData(endothelial_cells, normalization.method = "LogNormalize",
    scale.factor = 10000)

# Feature selection
endothelial_cells <- FindVariableFeatures(endothelial_cells, selection.method = "vst",
    nfeatures = 2000)
top10 <- head(VariableFeatures(endothelial_cells), 10)
p2 <- VariableFeaturePlot(endothelial_cells)
p2 <- LabelPoints(plot = p2, points = top10, repel = TRUE)

# Scaling the data
all.genes <- rownames(endothelial_cells)
endothelial_cells <- ScaleData(endothelial_cells, features = all.genes)

# Linear dimensional reduction
endothelial_cells <- RunPCA(endothelial_cells, features = VariableFeatures(object = endothelial_cells))
p3 <- DimPlot(endothelial_cells, reduction = "pca")

# Determining the 'dimensionality' of the dataset
p4 <- ElbowPlot(endothelial_cells)
```

```
p2 + p3 + p4
```

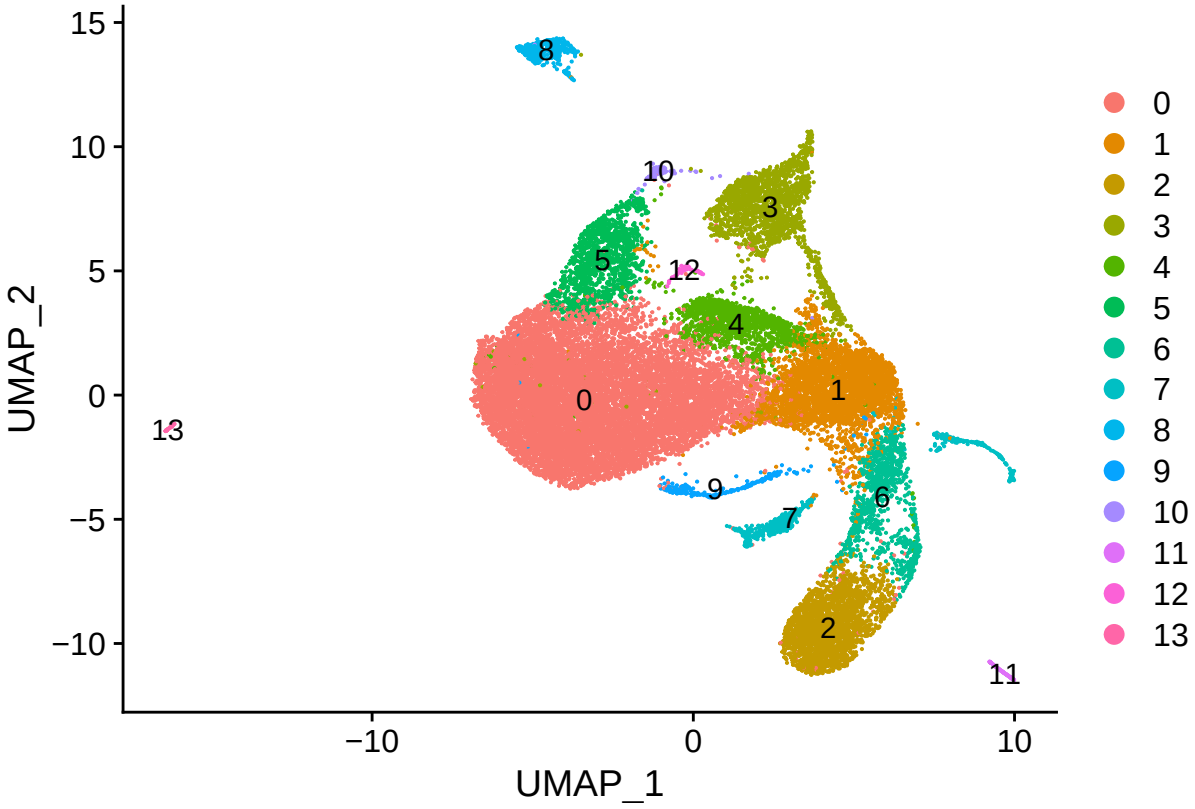
```
## Warning: Transformation introduced infinite values in continuous x-axis
```



Clustering

```
endothelial_cells <- FindNeighbors(endothelial_cells, dims = 1:15)
endothelial_cells <- FindClusters(endothelial_cells, resolution = 0.25)

endothelial_cells <- RunUMAP(endothelial_cells, dims = 1:15)
```

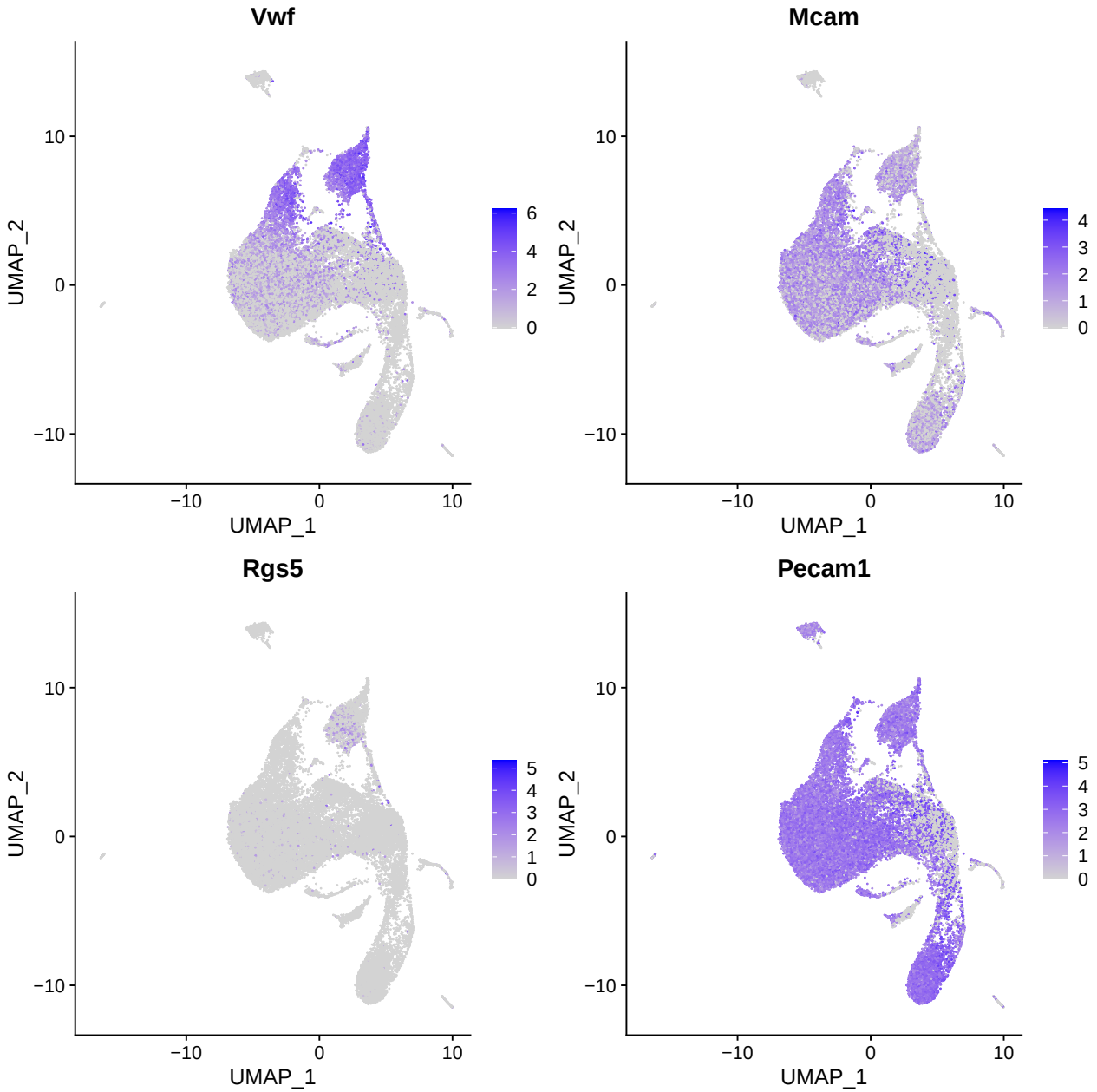
```
DimPlot(endothelial_cells, reduction = "umap", label = T)
```



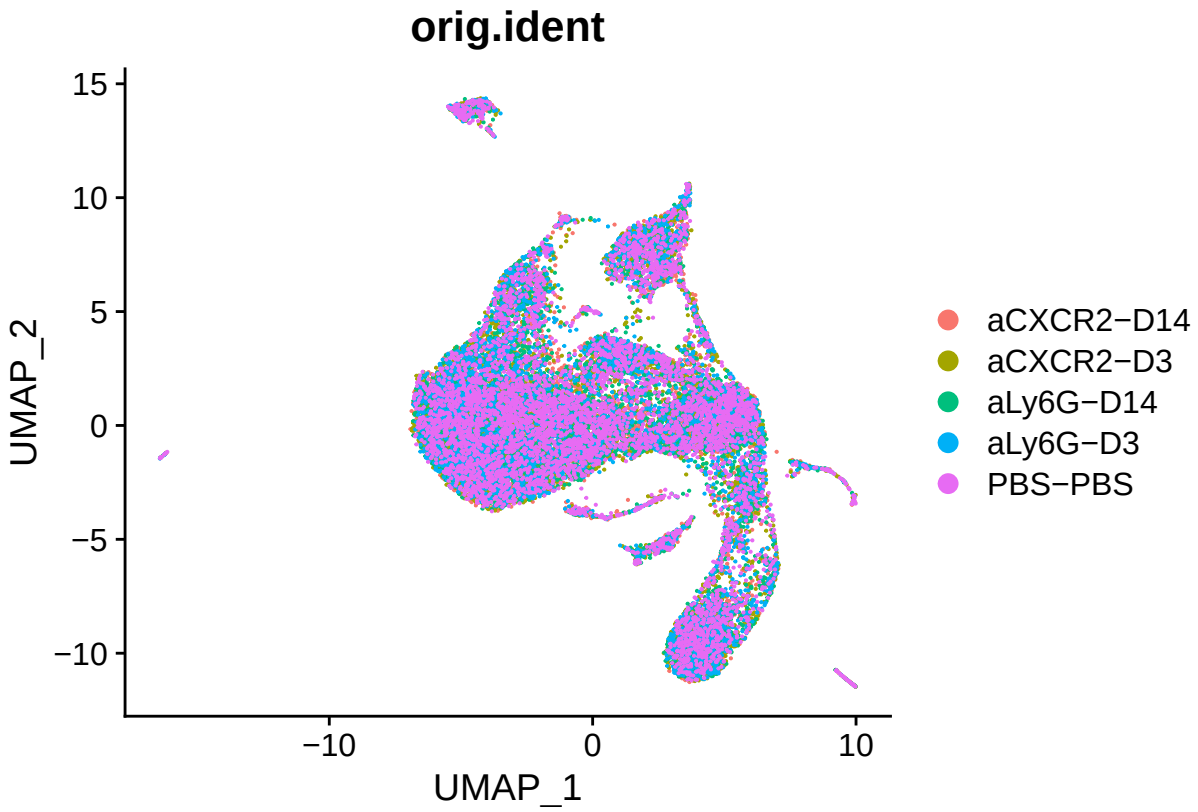

```
# ggsave('Umap_1.png', height = 8, width=12)
```

Endothelial genes

```
FeaturePlot(endothelial_cells, features = c("Vwf", "Mcam", "Rgs5",
    "Pecam1"))
```




```
DimPlot(endothelial_cells, group.by = "orig.ident", reduction = "umap")
```



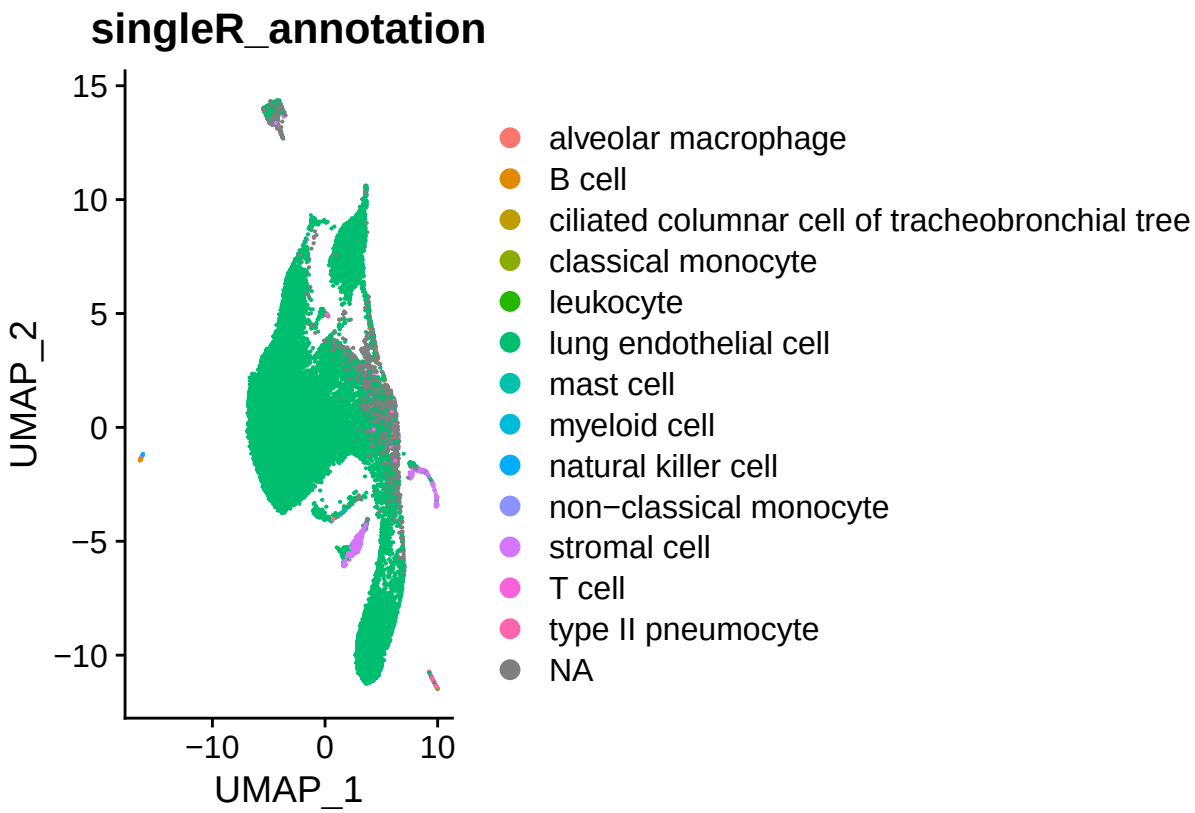
Automatic annotation

Annotation with TabulaMurisData

```
library(ExperimentHub)
require(scuttle)

eh <- ExperimentHub()
query(eh, "TabulaMurisData")
ref <- eh[["EH1617"]]

lung_ref <- ref[, !is.na(ref$cell_ontology_class)]
lung_ref <- lung_ref[, lung_ref$tissue == "Lung"]

lung_ref <- logNormCounts(lung_ref)

tested_data <- as.SingleCellExperiment(endothelial_cells)
tested_data <- logNormCounts(tested_data)

results <- SingleR(test = tested_data, ref = lung_ref, labels = lung_ref$cell_ontology_class)
cell_annotations <- results
table(cell_annotations$labels)

rownames(endothelial_cells@meta.data)
rownames(cell_annotations)
setequal(rownames(endothelial_cells@meta.data), rownames(cell_annotations))
```

```
endothelial_cells[["singleR_annotation"]] <- cell_annotations[, c(4)]
```

```
DimPlot(endothelial_cells, reduction = "umap", group.by = "singleR_annotation",
    cols = c())
```

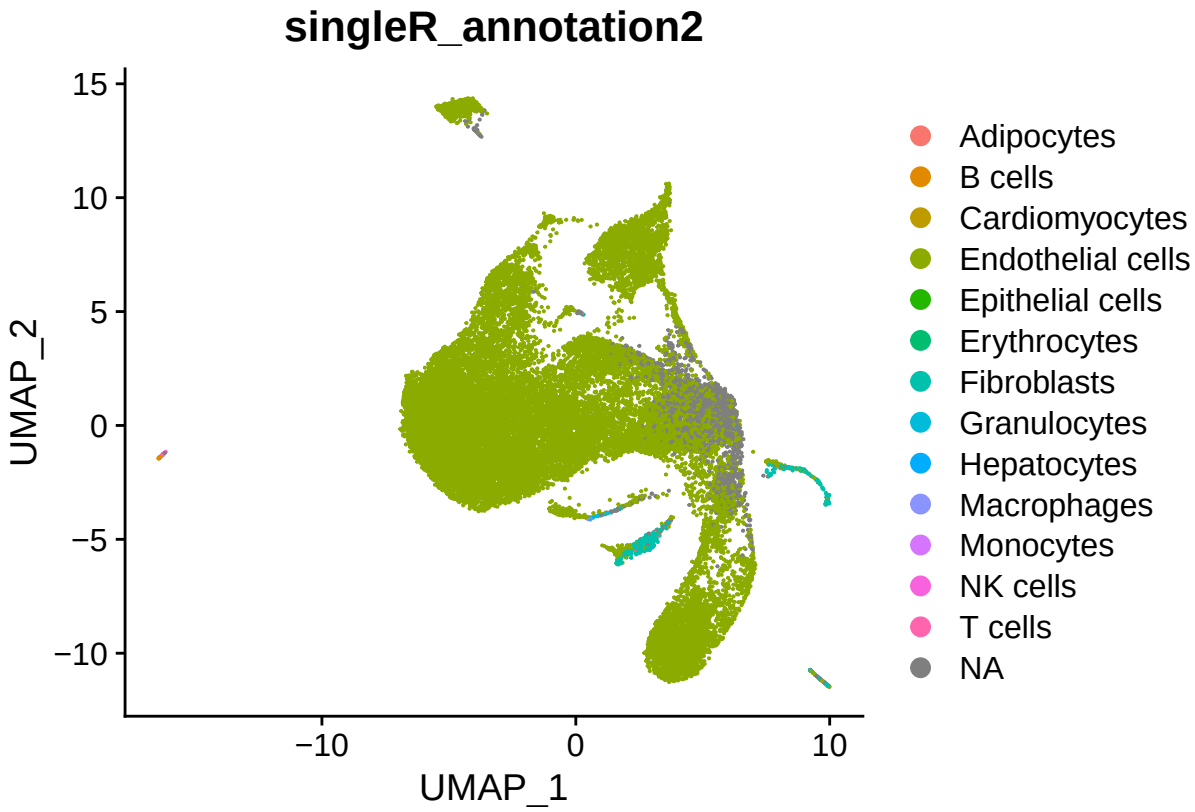



```
# ggsave('Umap_singleR1.png', height = 8, width=12)
```

Annotation with MouseRNAseqData

```
library(celldex)

ref.bulk <- MouseRNAseqData()

results_celldex <- SingleR(test = tested_data, ref = ref.bulk, labels = ref.bulk$label.main)
endothelial_cells[["singleR_annotation2"]] <- results_celldex[, c(4)]
```

```
DimPlot(endothelial_cells, reduction = "umap", group.by = "singleR_annotation2")
```




```
# ggsave('Umap_singleR2.png', height = 8, width=12)
```

Clusters 7, 11 and 13 were identified as contamination were removed.

```
endothelial_cells <- subset(endothelial_cells, seurat_clusters %in%
    c(7, 11, 13), invert = T)
saveRDS(endothelial_cells, "endothelial_cells.rds")
```

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 24.04.2 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK version 3.12.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=fr_BE.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=fr_BE.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=fr_BE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_BE.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Europe/Brussels
## tzcode source: system (glibc)
##
## attached base packages:
```

```
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] celldex_1.10.1              TabulaMurisData_1.18.0
##  [3] scuttle_1.10.1              SingleCellExperiment_1.22.0
##  [5] ExperimentHub_2.8.0         AnnotationHub_3.8.0
##  [7] BiocFileCache_2.8.0         dbplyr_2.3.2
##  [9] SingleR_2.2.0               SummarizedExperiment_1.30.2
## [11] Biobase_2.60.0              GenomicRanges_1.52.0
## [13] GenomeInfoDb_1.36.0         IRanges_2.34.0
## [15] S4Vectors_0.38.1            BiocGenerics_0.46.0
## [17] MatrixGenerics_1.12.2       matrixStats_1.0.0
## [19] formatR_1.14                ggplot2_3.4.2
## [21] patchwork_1.1.2             SeuratObject_4.1.3
## [23] Seurat_4.3.0                dplyr_1.1.2
##
## loaded via a namespace (and not attached):
##   [1] RcppAnnoy_0.0.21              splines_4.3.3
##   [3] later_1.3.1                   filelock_1.0.2
##   [5] bitops_1.0-7                  tibble_3.2.1
##   [7] R.oo_1.25.0                   polyclip_1.10-4
##   [9] lifecycle_1.0.3               globals_0.16.2
##  [11] lattice_0.22-5                MASS_7.3-60.0.1
##  [13] magrittr_2.0.3                plotly_4.10.2
##  [15] rmarkdown_2.23                yaml_2.3.7
##  [17] httpuv_1.6.11                 sctransform_0.3.5
##  [19] spam_2.9-1                    sp_2.2-0
##  [21] spatstat.sparse_3.0-2         reticulate_1.30
##  [23] cowplot_1.1.1                 pbapply_1.7-2
##  [25] DBI_1.1.3                     RColorBrewer_1.1-3
##  [27] abind_1.4-5                   zlibbioc_1.46.0
##  [29] Rtsne_0.16                    purrr_1.0.1
##  [31] R.utils_2.12.2                RCurl_1.98-1.12
##  [33] rappdirs_0.3.3                GenomeInfoDbData_1.2.10
##  [35] ggrepel_0.9.3                 irlba_2.3.5.1
##  [37] listenv_0.9.0                 spatstat.utils_3.0-3
##  [39] goftest_1.2-3                 spatstat.random_3.1-5
##  [41] fitdistrplus_1.1-11           parallelly_1.36.0
##  [43] DelayedMatrixStats_1.22.1     leiden_0.4.3
##  [45] codetools_0.2-19              DelayedArray_0.26.3
##  [47] tidyselect_1.2.0              farver_2.1.1
##  [49] ScaledMatrix_1.8.1            spatstat.explore_3.2-1
##  [51] jsonlite_1.8.7                ellipsis_0.3.2
##  [53] progressr_0.13.0              ggridges_0.5.4
##  [55] survival_3.5-8                tools_4.3.3
##  [57] ica_1.0-3                     Rcpp_1.0.11
##  [59] glue_1.6.2                    gridExtra_2.3
##  [61] xfun_0.39                     withr_2.5.0
##  [63] BiocManager_1.30.21           fastmap_1.1.1
##  [65] fansi_1.0.4                   digest_0.6.33
##  [67] rsvd_1.0.5                    R6_2.5.1
##  [69] mime_0.12                     colorspace_2.1-0
##  [71] scattermore_1.2               tensor_1.5
```

```
##  [73] RSQLite_2.3.1                spatstat.data_3.0-1
##  [75] R.methodsS3_1.8.2            utf8_1.2.3
##  [77] tidyr_1.3.0                  generics_0.1.3
##  [79] data.table_1.14.8            httr_1.4.6
##  [81] htmlwidgets_1.6.2            S4Arrays_1.2.1
##  [83] uwot_0.1.16                  pkgconfig_2.0.3
##  [85] gtable_0.3.3                 blob_1.2.4
##  [87] lmtest_0.9-40                XVector_0.40.0
##  [89] htmltools_0.5.5              dotCall64_1.0-2
##  [91] scales_1.2.1                 png_0.1-8
##  [93] knitr_1.43                   rstudioapi_0.14
##  [95] reshape2_1.4.4               curl_5.0.1
##  [97] nlme_3.1-164                 cachem_1.0.8
##  [99] zoo_1.8-12                   stringr_1.5.0
## [101] BiocVersion_3.17.1           KernSmooth_2.23-22
## [103] parallel_4.3.3               miniUI_0.1.1.1
## [105] vipor_0.4.5                  AnnotationDbi_1.62.1
## [107] ggrastr_1.0.2                pillar_1.9.0
## [109] grid_4.3.3                   vctrs_0.6.3
## [111] RANN_2.6.1                   promises_1.2.0.1
## [113] BiocSingular_1.16.0          beachmat_2.16.0
## [115] xtable_1.8-4                 cluster_2.1.6
## [117] beeswarm_0.4.0               evaluate_0.21
## [119] cli_3.6.1                    compiler_4.3.3
## [121] rlang_1.1.1                  crayon_1.5.2
## [123] future.apply_1.11.0          labeling_0.4.2
## [125] plyr_1.8.8                   ggbeeswarm_0.7.2
## [127] stringi_1.8.4                viridisLite_0.4.2
## [129] deldir_1.0-9                 BiocParallel_1.34.2
## [131] Biostrings_2.68.1            munsell_0.5.0
## [133] lazyeval_0.2.2               spatstat.geom_3.2-4
## [135] Matrix_1.6-1                 bit64_4.0.5
## [137] sparseMatrixStats_1.12.0     future_1.33.0
## [139] KEGGREST_1.40.0              shiny_1.7.4.1
## [141] interactiveDisplayBase_1.38.0 highr_0.10
## [143] ROCR_1.0-11                  memoise_2.0.1
## [145] igraph_1.5.0.1               bit_4.0.5
```