Demonstration of cell type annotation using CellRef

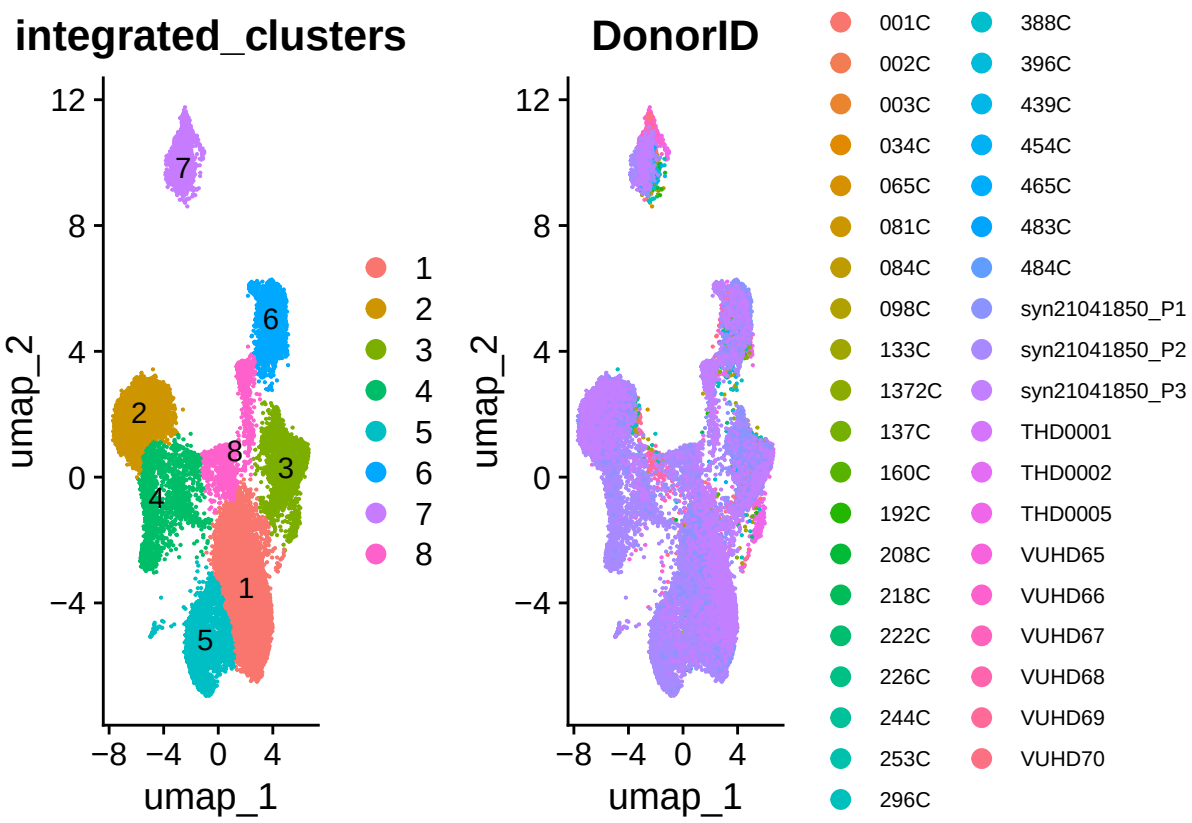
Minzhe Guo

Compiled: 2023-01-13

```
library(Seurat)
library(SeuratDisk)
library(ggplot2)
library(patchwork)
library(CellRef)
library(monocle3)
library(pheatmap)
```

First, let's create a Seurat object using the UMI count matrix (lung_endo.counts.rds) and cell information (lung_endo.cells.rds) of the demo data, which can be downloaded from the data folder of this CellRef github.

```
cells = readRDS(file = "./lung_endo.cells.rds")
counts = readRDS(file = "./lung_endo.counts.rds")
query = CreateSeuratObject(counts = counts, meta.data = cells)
query = NormalizeData(query)

# batch correction of data from different donors
query = doDataIntegration(query, integration.batch = "DonorID", method = "Monocle3-mnn", npcs = 200,
    do.clustering = T)

# after integration, the method also perform clustering using leiden algorithm
g1 = DimPlot(query, reduction = "umap", group.by = "integrated_clusters", label = T)
g2 = DimPlot(query, reduction = "umap", group.by = "DonorID", label = F)
g2 = g2 + theme(legend.text = element_text(size = 8))
g1 + g2
```



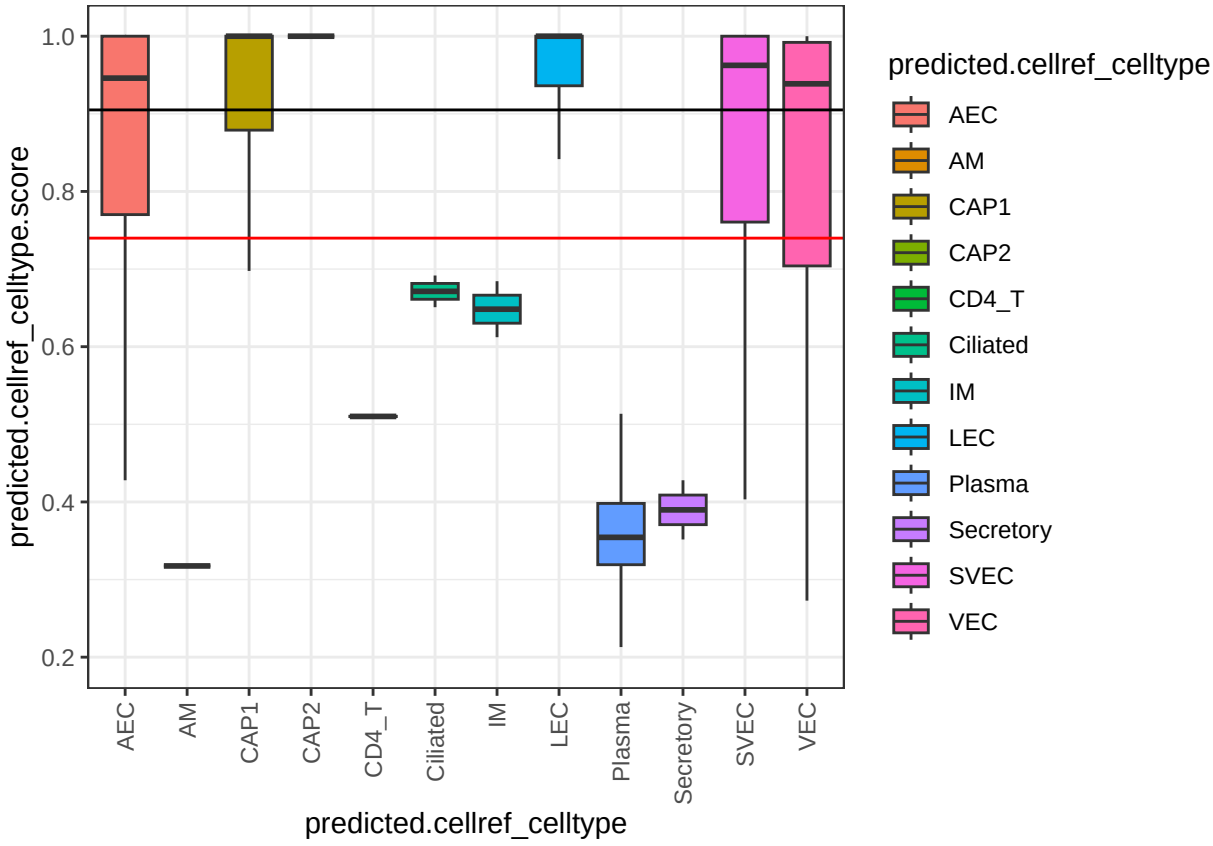
Next, we use LungMAP Human Lung CellRef to annotate the cells in the Seurat object that we just created

```
ref_ident = "celltype_level3"

ref = readRDS(file = "./LungMAP_HumanLung_CellRef_Seed.v1.rds")

refmap.anchors <- FindTransferAnchors(reference = ref, query = query, normalization.method = "SCT",
    reference.reduction = "pca", dims = 1:dim(ref@reductions$pca@cell.embeddings)[2])

mapped <- MapQuery(anchorset = refmap.anchors, reference = ref, query = query, refdata = list(cellref_c
    reference.reduction = "pca", reduction.model = "umap")
```

We can prune the predictions with low confidence. By default, we use the mean-sd as the cutoff of prediction scores.

```
score.mean = mean(mapped@meta.data$predicted.cellref_celltype.score)
score.sd = sd(mapped@meta.data$predicted.cellref_celltype.score)
score.cutoff = score.mean - score.sd

# We can see that while several non-Endo cell type were predicted, they were all have very low
# confidence, and will be pruned by our default cutoff (red) of the prediction score
g = ggplot(mapped@meta.data, aes(x = predicted.cellref_celltype, y = predicted.cellref_celltype.score,
    fill = predicted.cellref_celltype))
g = g + geom_boxplot(outlier.shape = NA)
g = g + geom_hline(yintercept = score.mean, color = "black")
g = g + geom_hline(yintercept = score.cutoff, color = "red")
```

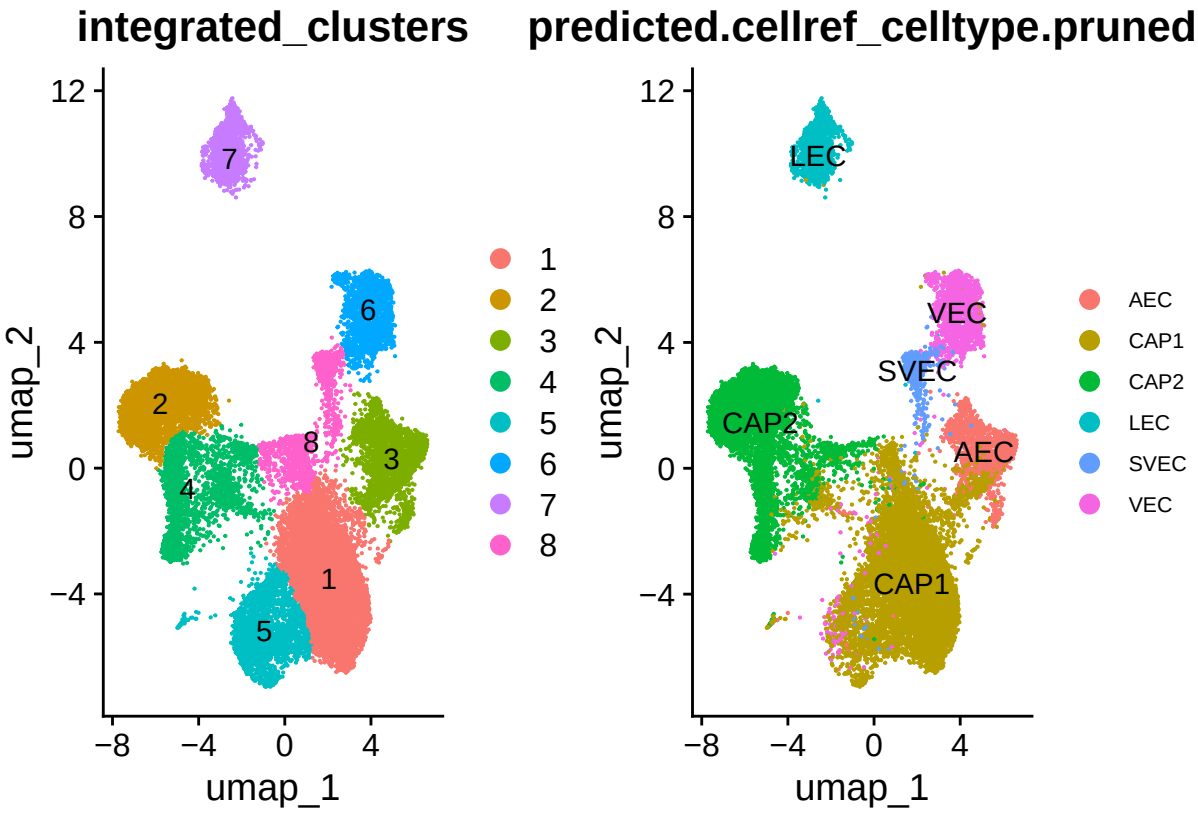
```
g = g + theme_bw()
g = g + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
g
```




```
# pruned the predictions lower than the cutoff of prediction score
mapped@meta.data$predicted.cellref_celltype.pruned = as.character(mapped@meta.data$predicted.cellref_cel
mapped@meta.data$predicted.cellref_celltype.pruned[which(mapped@meta.data$predicted.cellref_celltype.sco
    score.cutoff)] = "pruned"
```

UMAP visualization of the cell clusters and CellRef annotations

```
mapped.pruned = subset(mapped, predicted.cellref_celltype.pruned != "pruned")

g1 = DimPlot(mapped, reduction = "umap", group.by = "integrated_clusters", label = T)
g2 = DimPlot(mapped.pruned, reduction = "umap", group.by = "predicted.cellref_celltype.pruned",
    label = T)
g2 = g2 + theme(legend.text = element_text(size = 8))
g1 + g2
```



Session Info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] pheatmap_1.0.12             monocle3_1.0.0
##  [3] SingleCellExperiment_1.16.0 SummarizedExperiment_1.24.0
##  [5] GenomicRanges_1.46.1        GenomeInfoDb_1.30.1
##  [7] IRanges_2.28.0              S4Vectors_0.32.3
```

```
##  [9] MatrixGenerics_1.6.0        matrixStats_0.62.0
## [11] Biobase_2.54.0              BiocGenerics_0.40.0
## [13] CellRef_0.1.0               patchwork_1.1.2
## [15] ggplot2_3.4.0               SeuratDisk_0.0.0.9019
## [17] sp_1.5-0                    SeuratObject_4.1.2
## [19] Seurat_4.2.0
##
## loaded via a namespace (and not attached):
##   [1] plyr_1.8.7                igraph_1.3.5
##   [3] lazyeval_0.2.2            splines_4.1.0
##   [5] BiocParallel_1.26.2       listenv_0.8.0
##   [7] scattermore_0.8           digest_0.6.30
##   [9] htmltools_0.5.3           viridis_0.6.2
##  [11] fansi_1.0.3               magrittr_2.0.3
##  [13] ScaledMatrix_1.0.0        tensor_1.5
##  [15] cluster_2.1.4             ROCR_1.0-11
##  [17] limma_3.48.3              globals_0.16.1
##  [19] spatstat.sparse_3.0-0     colorspace_2.0-3
##  [21] ggrepel_0.9.1             xfun_0.34
##  [23] dplyr_1.0.10              crayon_1.5.2
##  [25] RCurl_1.98-1.9            jsonlite_1.8.3
##  [27] progressr_0.11.0          spatstat.data_3.0-0
##  [29] survival_3.4-0            zoo_1.8-11
##  [31] glue_1.6.2                polyclip_1.10-4
##  [33] gtable_0.3.1              zlibbioc_1.40.0
##  [35] XVector_0.34.0            leiden_0.4.3
##  [37] DelayedArray_0.20.0       BiocSingular_1.8.1
##  [39] future.apply_1.10.0       abind_1.4-5
##  [41] scales_1.2.1              DBI_1.1.3
##  [43] spatstat.random_3.0-1     miniUI_0.1.1.1
##  [45] Rcpp_1.0.9                viridisLite_0.4.1
##  [47] xtable_1.8-4              reticulate_1.26
##  [49] spatstat.core_2.4-4       rsvd_1.0.5
##  [51] bit_4.0.4                 ResidualMatrix_1.2.0
##  [53] htmlwidgets_1.5.4         httr_1.4.4
##  [55] RColorBrewer_1.1-3        ellipsis_0.3.2
##  [57] ica_1.0-3                 farver_2.1.1
##  [59] scuttle_1.2.1             pkgconfig_2.0.3
##  [61] uwot_0.1.14               deldir_1.0-6
##  [63] utf8_1.2.2                labeling_0.4.2
##  [65] tidyselect_1.2.0          rlang_1.0.6
##  [67] reshape2_1.4.4            later_1.3.0
##  [69] munsell_0.5.0             tools_4.1.0
##  [71] cli_3.4.1                 generics_0.1.3
##  [73] ggridges_0.5.4            batchelor_1.8.1
##  [75] evaluate_0.17             stringr_1.4.1
##  [77] fastmap_1.1.0             yaml_2.3.6
##  [79] goftest_1.2-3             knitr_1.40
##  [81] bit64_4.0.5               fitdistrplus_1.1-8
##  [83] purrr_0.3.5               RANN_2.6.1
##  [85] sparseMatrixStats_1.4.2   pbapply_1.5-0
##  [87] future_1.28.0             nlme_3.1-160
##  [89] mime_0.12                 formatR_1.12
##  [91] hdf5r_1.3.7               compiler_4.1.0
```

```
##  [93] rstudioapi_0.14           plotly_4.10.0
##  [95] png_0.1-7                 spatstat.utils_3.0-1
##  [97] tibble_3.1.8              stringi_1.7.6
##  [99] highr_0.9                 rgeos_0.5-9
## [101] lattice_0.20-45           Matrix_1.5-1
## [103] vctrs_0.5.0               pillar_1.8.1
## [105] lifecycle_1.0.3           spatstat.geom_3.0-3
## [107] lmtest_0.9-40             BiocNeighbors_1.10.0
## [109] RcppAnnoy_0.0.20          data.table_1.14.4
## [111] cowplot_1.1.1             bitops_1.0-7
## [113] irlba_2.3.5.1             httpuv_1.6.6
## [115] R6_2.5.1                  promises_1.2.0.1
## [117] KernSmooth_2.23-20        gridExtra_2.3
## [119] parallelly_1.32.1         codetools_0.2-18
## [121] MASS_7.3-58.1             assertthat_0.2.1
## [123] leidenbase_0.1.12         withr_2.5.0
## [125] sctransform_0.3.5         GenomeInfoDbData_1.2.7
## [127] mgcv_1.8-41               parallel_4.1.0
## [129] beachmat_2.8.1            grid_4.1.0
## [131] rpart_4.1.19              tidyr_1.2.1
## [133] DelayedMatrixStats_1.14.3 rmarkdown_2.17
## [135] Rtsne_0.16                shiny_1.7.3
```