Cell annotation

Material

https://youtu.be/dNO7W7qkKBM?si=xkLBHjT8V25AEzIx

- Review on automated cell annotation

Exercises

Code starts here:

```
seu <- readRDS("day2/seu_day2-4.rds")
library(Seurat)
```

```
Loading required package: SeuratObject
```

```
Warning: package 'SeuratObject' was built under R version 4.3.3
```

```
Loading required package: sp
```

```

Attaching package: 'SeuratObject'
```

```
The following objects are masked from 'package:base':

    intersect, t
```

Code ends here

Load the following packages:

Code starts here:

```
library(celldex)
```

```
Loading required package: SummarizedExperiment
```

```
Loading required package: MatrixGenerics
```

```
Loading required package: matrixStats
```

```
Warning: package 'matrixStats' was built under R version 4.3.3
```

```

Attaching package: 'MatrixGenerics'
```

```
The following objects are masked from 'package:matrixStats':

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
```

```
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

Loading required package: GenomicRanges

Loading required package: stats4

Loading required package: BiocGenerics


Attaching package: 'BiocGenerics'

The following object is masked from 'package:SeuratObject':

    intersect

The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs

The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min

Loading required package: S4Vectors


Attaching package: 'S4Vectors'

The following object is masked from 'package:utils':

    findMatches
```

```
The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges


Attaching package: 'IRanges'

The following object is masked from 'package:sp':

    %over%

Loading required package: GenomeInfoDb

Warning: package 'GenomeInfoDb' was built under R version 4.3.3

Loading required package: Biobase

Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.


Attaching package: 'Biobase'

The following object is masked from 'package:MatrixGenerics':

    rowMedians

The following objects are masked from 'package:matrixStats':

    anyMissing, rowMedians


Attaching package: 'SummarizedExperiment'

The following object is masked from 'package:Seurat':

    Assays

The following object is masked from 'package:SeuratObject':

    Assays
```

```
library(SingleR)
```

```
Attaching package: 'SingleR'
```

```
The following objects are masked from 'package:celldex':

    BlueprintEncodeData, DatabaseImmuneCellExpressionData,
    HumanPrimaryCellAtlasData, ImmGenData, MonacoImmuneData,
    MouseRNAseqData, NovershternHematopoieticData
```

Code ends here

In the last exercise we saw that probably clustering at a resolution of 0.3 gave the most sensible results. Let's therefore set the default identity of each cell based on this clustering:

Code starts here:

```
seu <- Seurat::SetIdent(seu, value = seu$RNA_snn_res.0.3)
```

Code ends here

> **Note**
>
> From now on, grouping (e.g. for plotting) is done by the active identity (set at `@active.ident`) by default.

During cell annotation we will use the original count data (not the integrated data):

Code starts here:

```
DefaultAssay(seu) <- "RNA"
```

Code ends here

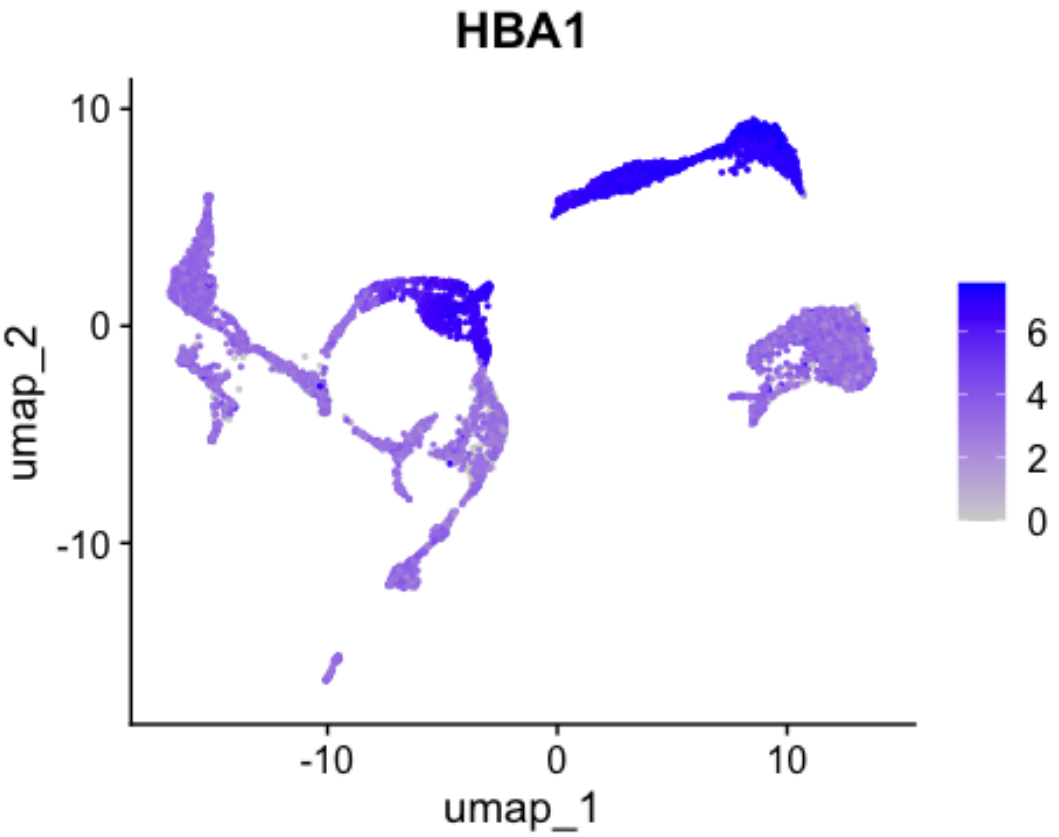
Based on the UMAP we have generated, we can visualize expression for a gene in each cluster:

Code starts here:

```
Seurat::FeaturePlot(seu, "HBA1")
```

```
Warning: The `slot` argument of `FetchData()` is deprecated as of SeuratObjec
t 5.0.0.
ℹ Please use the `layer` argument instead.
ℹ The deprecated feature was likely used in the Seurat package.
  Please report the issue at <https://github.com/satijalab/seurat/issues>.
```

Code ends here



Based on expression of sets of genes you can do a manual cell type annotation. If you know the marker genes for some cell types, you can check whether they are up-regulated in one or the other cluster. Here we have some marker genes for two different cell types:

Code starts here:

```
tcell_genes <- c("IL7R", "LTB", "TRAC", "CD3D")
monocyte_genes <- c("CD14", "CST3", "CD68", "CTSS")
```

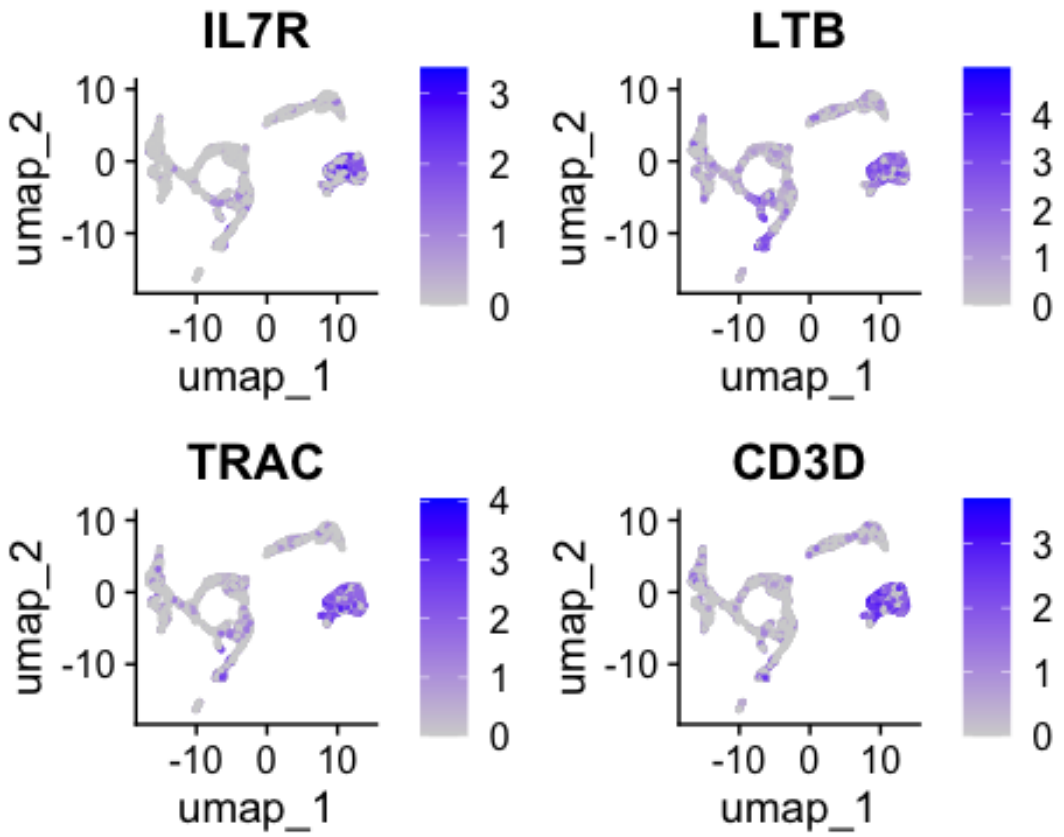
Code ends here

Let's have a look at the expression of the four T cell genes:

Code starts here:

```
Seurat::FeaturePlot(seu, tcell_genes, ncol=2)
```

Code ends here



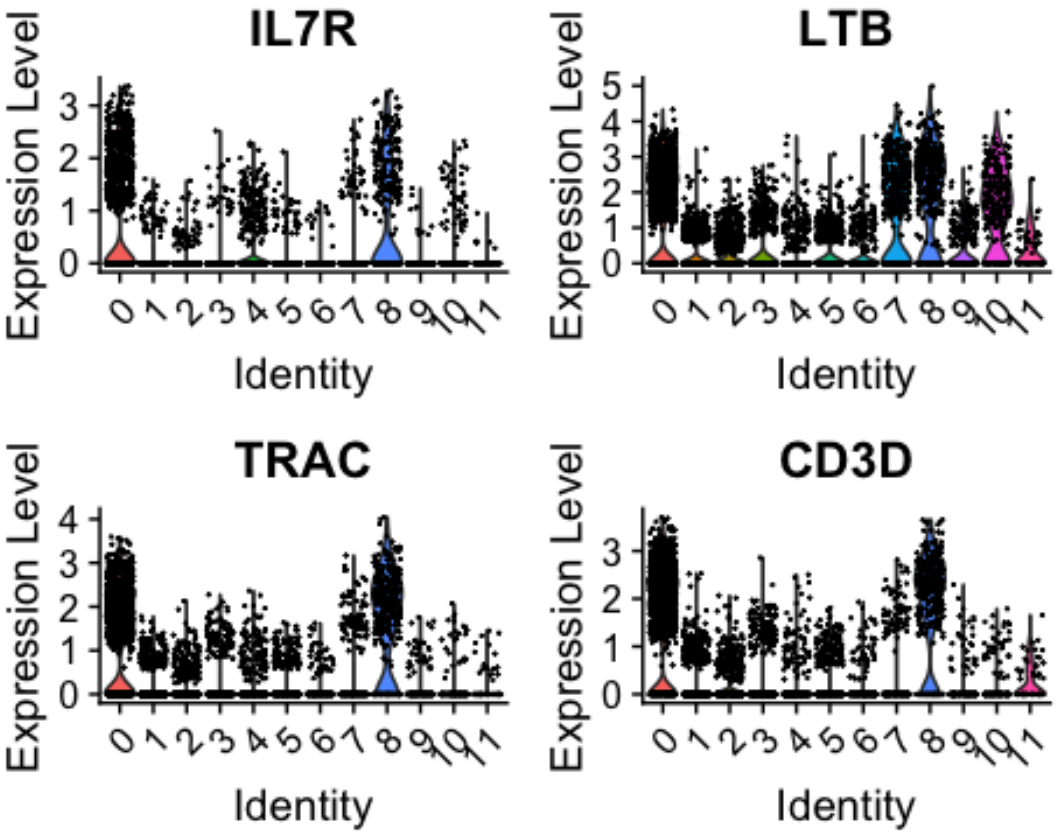
These cells are almost all in cluster 0 and 8. Which becomes clearer when looking at the violin plot:

Code starts here:

```
Seurat::VlnPlot(seu,
                features = tcell_genes,
                ncol = 2)
```

```
Warning: `PackageCheck()` was deprecated in SeuratObject 5.0.0.
ℹ Please use `rlang::check_installed()` instead.
ℹ The deprecated feature was likely used in the Seurat package.
  Please report the issue at <https://github.com/satijalab/seurat/issues>.
```

Code ends here



Exercise

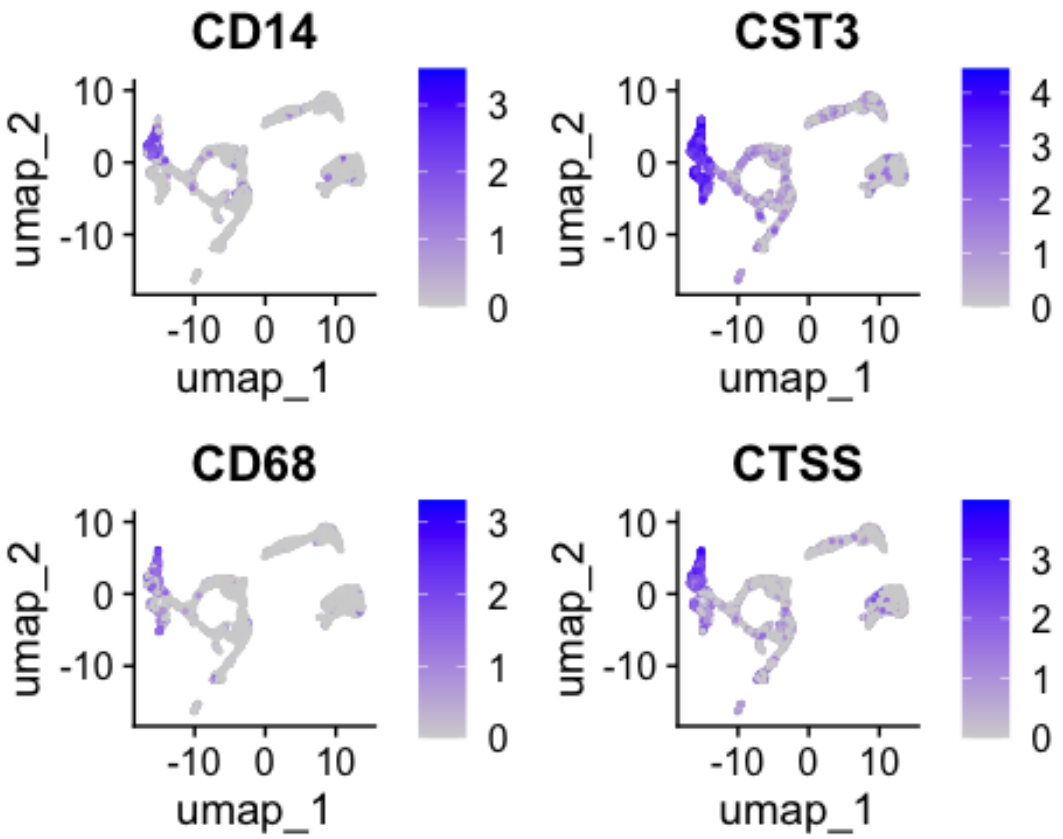
Have a look at the monocyte genes as well. Which clusters contain probably monocytes?

Answer

Code starts here:

```
Seurat::FeaturePlot(seu, monocyte_genes, ncol=2)
```

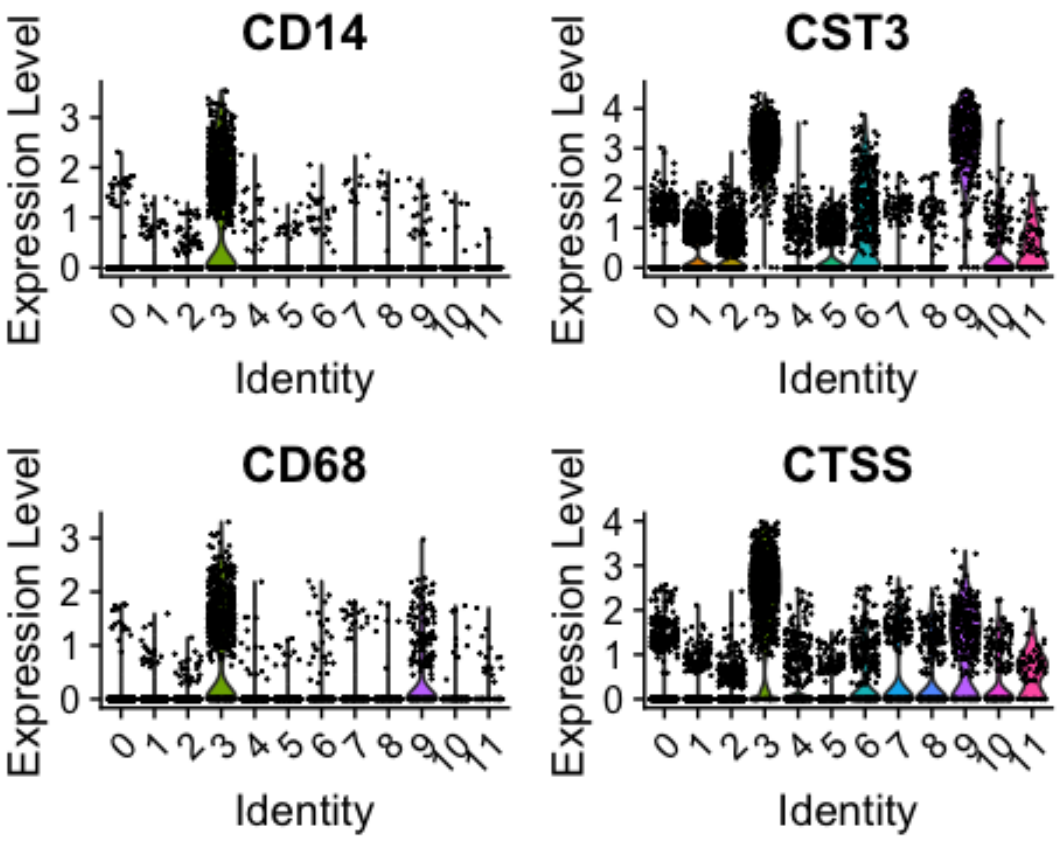
Code ends here



Code starts here

```
Seurat::VlnPlot(seu,
                features = monocyte_genes,
                ncol = 2)
```

Code ends here



We can also automate this with the function `AddModuleScore`. For each cell, an expression score for a group of genes is calculated:

Code starts here:

```
seu <- Seurat::AddModuleScore(seu,
                              features = list(tcell_genes),
                              name = "tcell_genes")
```

Code ends here

Exercise

After running `AddModuleScore`, a column was added to `seu@meta.data`.

A. What is the name of that column? What kind of data is in there?

B. Generate a UMAP with color accoding to this column and a violinplot grouped by cluster. Is this according to what we saw in the previous exercise?

Answer

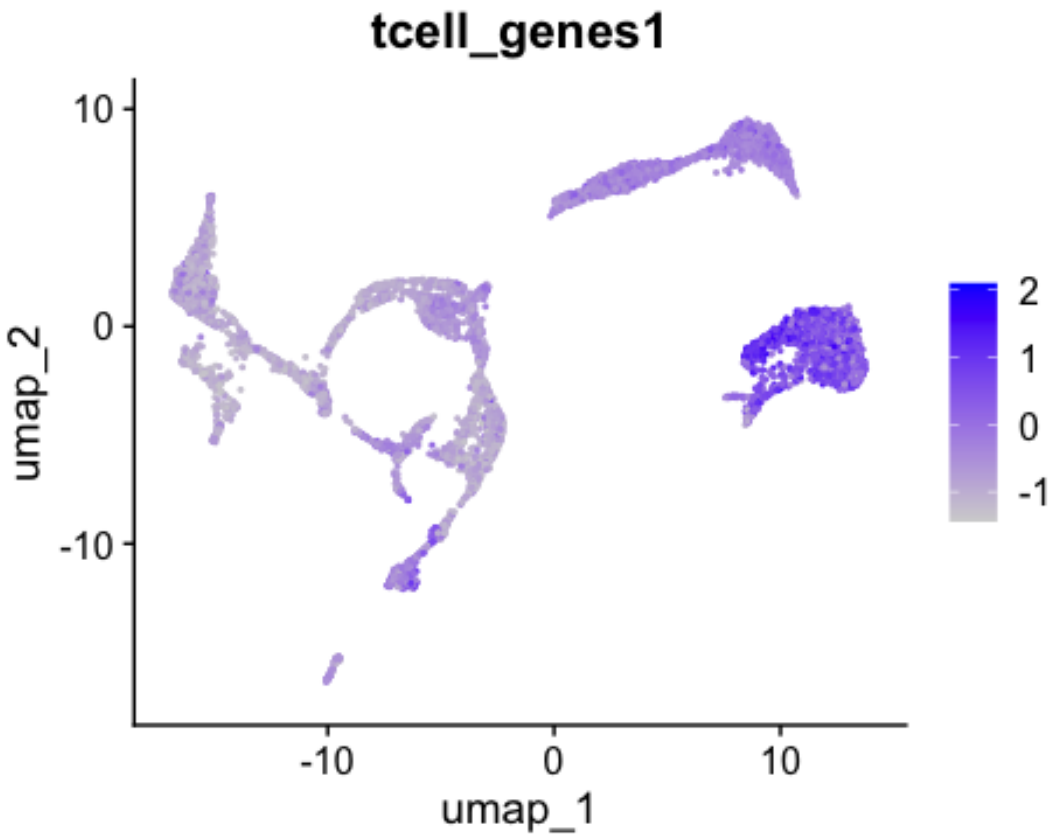
A. The new column is called `tcell_genes1`. It contains the module score for each cell (which is basically the average expression of the set of genes).

B. You can plot the UMAP with

Code starts here:

```
Seurat::FeaturePlot(seu, "tcell_genes1")
```

Code ends here

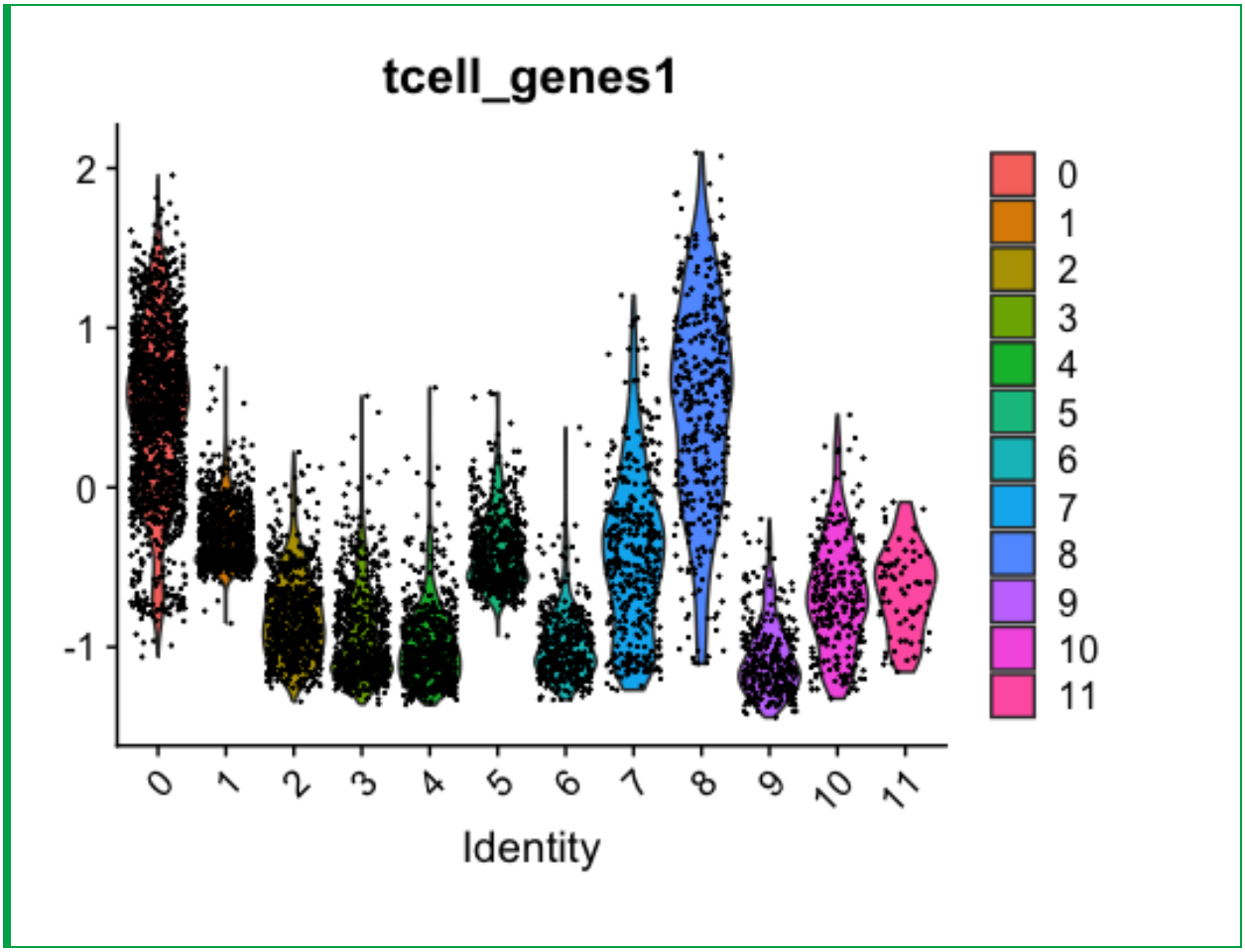


Which indeed shows these genes are mainly expressed in clusters 0 and 8:

Code starts here:

```
Seurat::VlnPlot(seu,
                "tcell_genes1")
```

Code ends here



Annotating cells according to cycling phase

Based on the same principle, we can also annotate cell cycling state. The function `CellCycleScore` uses `AddModuleScore` to get a score for the G2/M and S phase (the mitotic phases in which cell is cycling). In addition, `CellCycleScore` assigns each cell to either the G2/M, S or G1 phase.

First we extract the built-in genes for cell cycling:

Code starts here:

```
s.genes <- Seurat::cc.genes.updated.2019$s.genes
g2m.genes <- Seurat::cc.genes.updated.2019$g2m.genes
```

Code ends here

Now we run the function:

Code starts here:

```
seu <- Seurat::CellCycleScoring(seu,
                                s.features = s.genes,
                                g2m.features = g2m.genes)
```

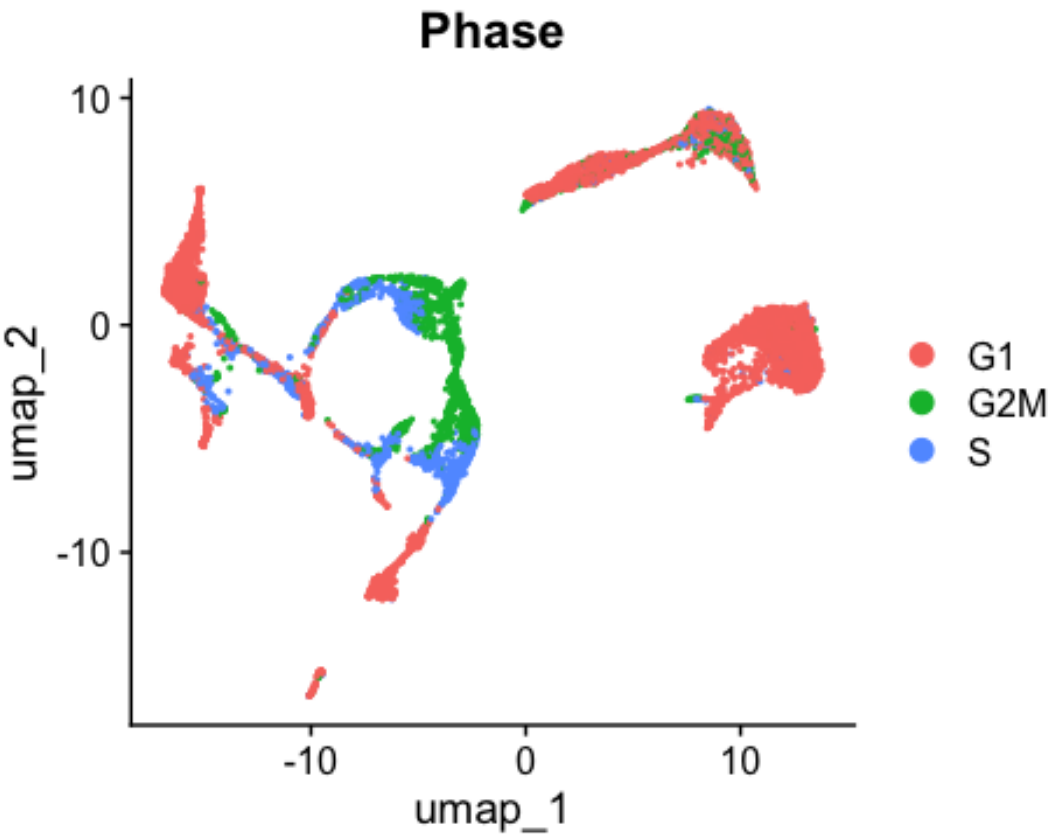
Code ends here

And we can visualize the annotations:

Code starts here:

```
Seurat::DimPlot(seu, group.by = "Phase")
```

Code ends here



Based on your application, you can try to regress out the cell cycling scores at the step of scaling. Reasons for doing that could be:

- Merging cycling and non-cycling cells of the same type in one cluster
- Merging G2/M and S phase in one cluster

Note

> Note that correcting for cell cycling is performed at the scaling step. It will therefore only influence analyses that use scaled data, like dimensionality reduction and clustering. For e.g. differential gene expression testing, we use the raw original counts (not scaled).

Here, we choose not to regress out either of them. Because we are looking at developing cells, we might be interested to keep cycling cells seperated. In addition, the G2/M and S phases seem to be in the same clusters. More info on correcting for cell cycling here.

Cell type annotation using `SingleR`

To do a fully automated annoation, we need a reference dataset of primary cells. Any reference could be used. The package `scRNAseq` in Bioconductor includes several scRNAseq datasets that can be used as reference to `SingleR`. One could also use a reference made of bulk RNA seq data. Here we are using the a hematopoietic reference dataset from `celldex`. Check out what's in there:

Code starts here:

```
ref <- celldex::NovershternHematopoieticData()
class(ref)
table(ref$label.main)
```

Code ends here

> **Note**
>
> You will be asked whether to create the directory `/home/rstudio/.cache/R/ExperimentHub`. Type `yes` as a response.

> **Note**
>
> You can find more information on different reference datasets at the `celldex` documentation

Now `SingleR` compares our normalized count data to a reference set, and finds the most probable annation:

Code starts here:

```
seu_SingleR <- SingleR::SingleR(test = Seurat::GetAssayData(seu),
                                ref = ref,
                                labels = ref$label.main)
```

Code ends here

See what's in there by using head:

Code starts here:

```
head(seu_SingleR)
```

```
DataFrame with 6 rows and 4 columns
                                                    scores          labels
                                                  <matrix>     <character>
PBMMC-1_AAACCTGCAGACGCAA-1 0.216202:0.197296:0.086435:...         B cells
PBMMC-1_AAACCTGTCATCACCC-1 0.143005:0.129582:0.170521:...    CD8+ T cells
PBMMC-1_AAAGATGCATAAAGGT-1 0.113423:0.196264:0.111341:...       Monocytes
PBMMC-1_AAAGCAAAGCAGCGTA-1 0.166749:0.168504:0.239303:...    CD4+ T cells
PBMMC-1_AAAGCAACAATAACGA-1 0.102549:0.103979:0.178762:...    CD8+ T cells
PBMMC-1_AAAGCAACATCAGTCA-1 0.181526:0.147693:0.115841:... Erythroid cells
                           delta.next   pruned.labels
                            <numeric>     <character>
PBMMC-1_AAACCTGCAGACGCAA-1  0.1471065         B cells
PBMMC-1_AAACCTGTCATCACCC-1  0.0753696    CD8+ T cells
PBMMC-1_AAAGATGCATAAAGGT-1  0.1274524       Monocytes
PBMMC-1_AAAGCAAAGCAGCGTA-1  0.0845267    CD4+ T cells
PBMMC-1_AAAGCAACAATAACGA-1  0.0607683    CD8+ T cells
PBMMC-1_AAAGCAACATCAGTCA-1  0.1057135 Erythroid cells
```

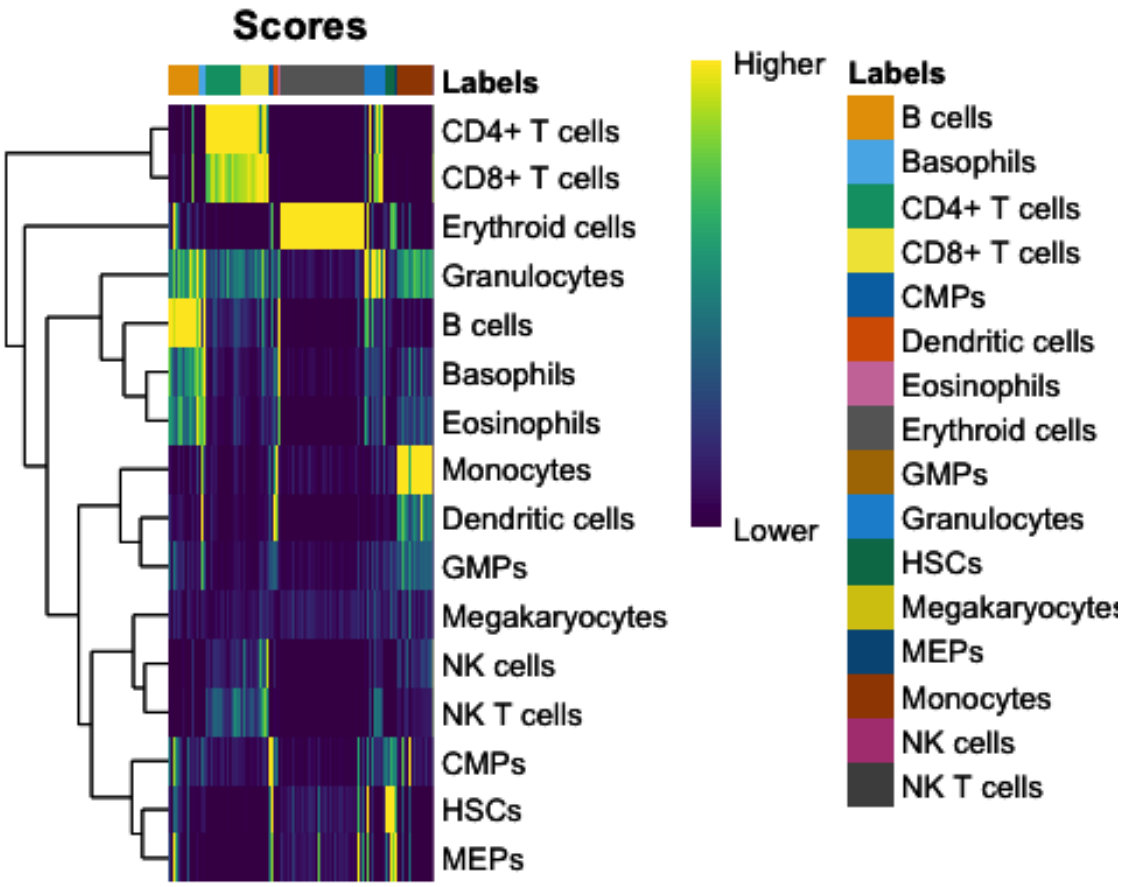
Code ends here

Visualize singleR score quality scores:

Code starts here:

```
SingleR::plotScoreHeatmap(seu_SingleR)
```

Code ends here



Code starts here:

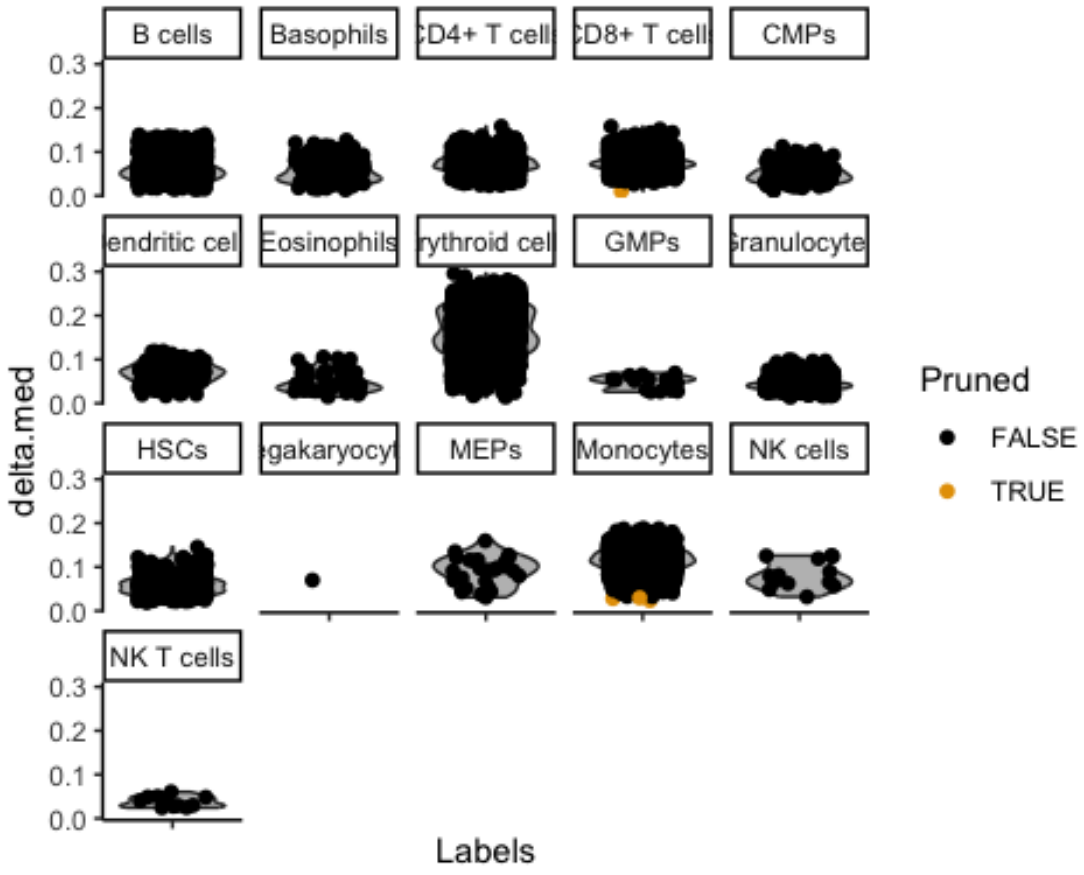
```
SingleR::plotDeltaDistribution(seu_SingleR)
```

```
Warning: Groups with fewer than two datapoints have been dropped.
ℹ Set `drop = FALSE` to consider such groups for position adjustment purposes
.

Warning in max(data$density, na.rm = TRUE): no non-missing arguments to max;
returning -Inf

Warning: Computation failed in `stat_ydensity()`.
Caused by error in `$<-.data.frame`:
! replacement has 1 row, data has 0
```

Code ends here



There are some annotations that contain only a few cells. They are usually not of interest, and they clogg our plots. Therefore we remove them from the annotation:

Code starts here:

```
singleR_labels <- seu_SingleR$labels
t <- table(singleR_labels)
other <- names(t)[t < 10]
singleR_labels[singleR_labels %in% other] <- "none"
```

Code ends here

In order to visualize it in our UMAP, we have to add the annotation to `seu@meta.data`:

Code starts here:

```
seu$SingleR_annot <- singleR_labels
```

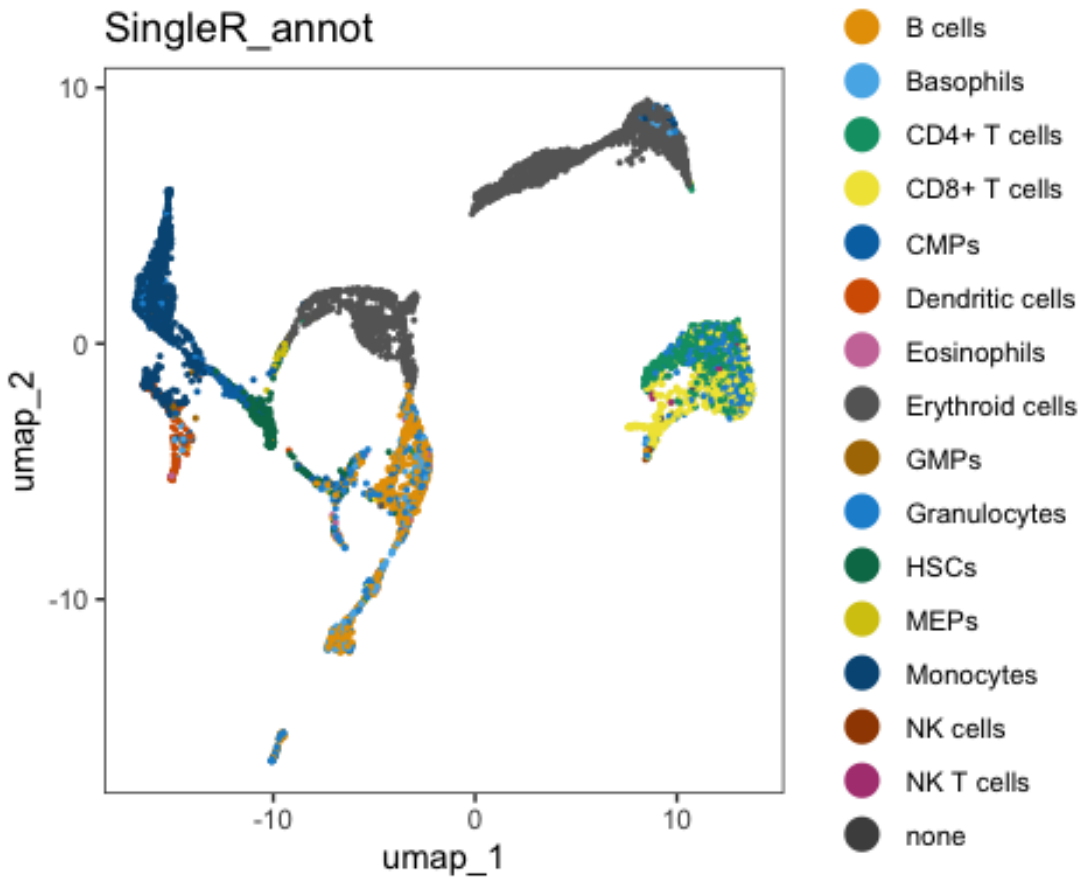
Code ends here

We can plot the annotations in the UMAP. Here, we use a different package for plotting (`dittoSeq`) as it has a bit better default coloring, and some other plotting functionality we will use later on.

Code starts here:

```
dittoSeq::dittoDimPlot(seu, "SingleR_annot", size = 0.7)
```

Code ends here

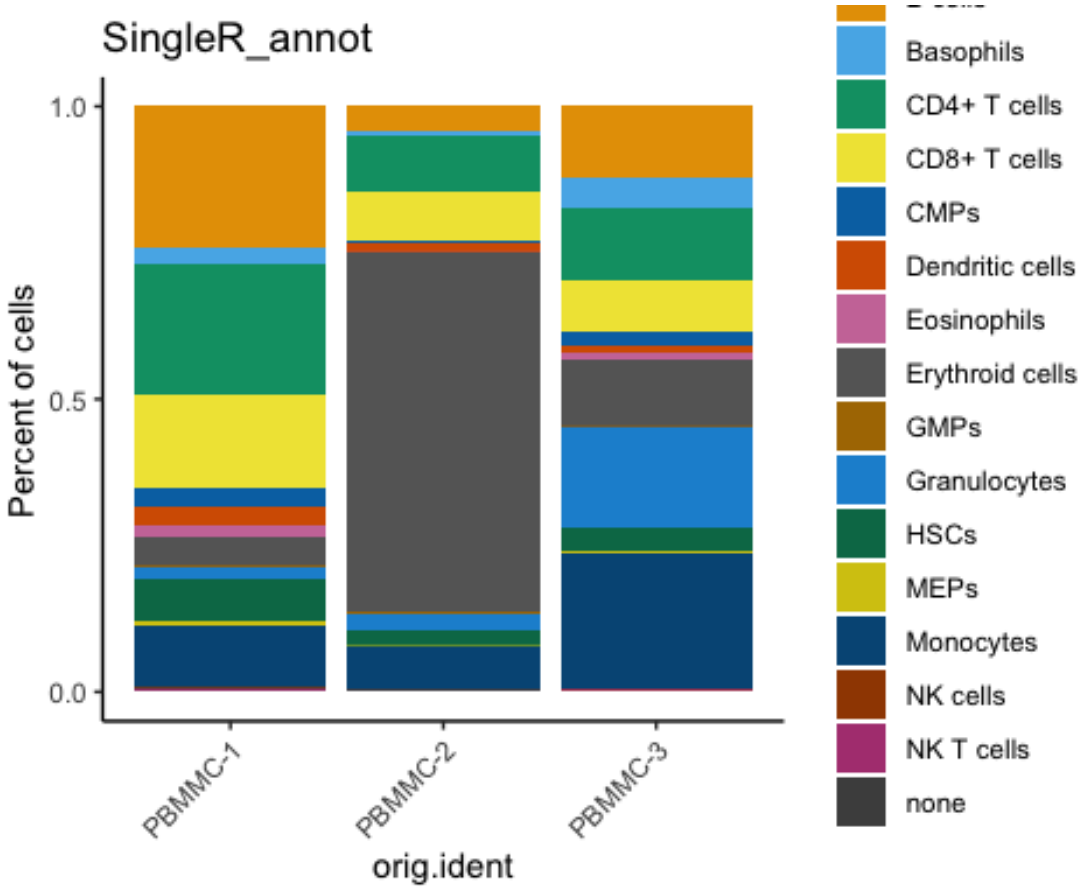


We can check out how many cells per sample we have for each annotated cell type:

Code starts here:

```
dittoSeq::dittoBarPlot(seu, var = "SingleR_annot", group.by = "orig.ident")
```

Code ends here



Exercise

Compare our manual annotation (based on the set of T cell genes) to the annotation with `SingleR`. Do they correspond?

You can for example use the plotting function `dittoBarPlot` to visualize the cell types according to cluster (use `RNA_snn_res.0.3` in stead of `orig.ident`))

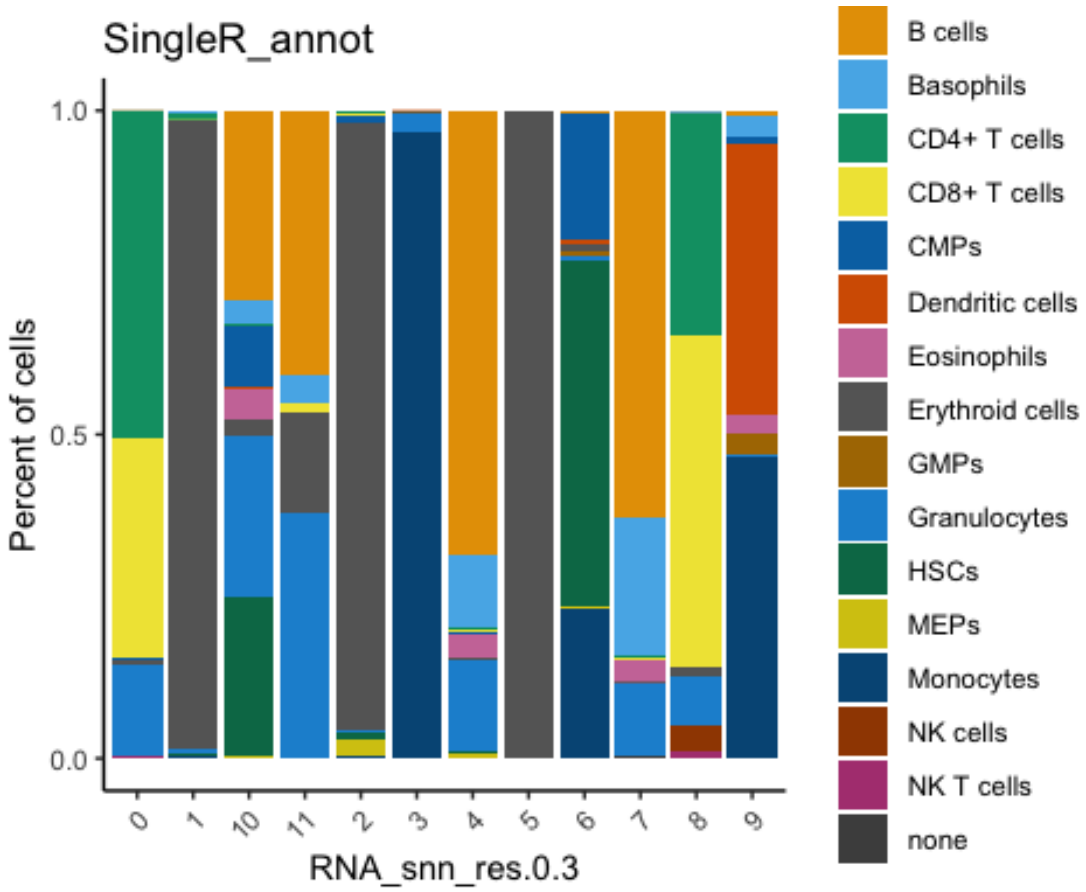
Answer

We can have a look at the mean module score for each `SingleR` annotation like this:

Code starts here:

```
dittoSeq::dittoBarPlot(seu,
                       var = "SingleR_annot",
                       group.by = "RNA_snn_res.0.3")
```

Code ends here



Here, you can see that cluster 0 and 8 contain cells annotated as T cells (CD4+ and CD8+).

Save the dataset and clear environment

Now, save the dataset so you can use it tomorrow:

Code starts here:

```
saveRDS(seu, "day3/seu_day3-1.rds")
```

Code ends here

Clear your environment:

Code starts here:

```
rm(list = ls())
gc()
.rs.restartR()
```

Code ends here