Clustering

Ahmed Mahfouz and Indu Khatri

11/16/2021

Overview

In this tutorial we will look at different approaches to cluster scRNA-seq datasets in order to characterize the different subgroups of cells. Using unsupervised clustering, we will try to identify groups of cells based on the similarities of the transcriptomes without any prior knowledge of the labels.

Load required packages:

```
suppressMessages(require(Seurat))
```

Datasets

Again, we will continue with dataset the you have preprocessed and visualized in the previous practicals. Let's start by loading the data again.

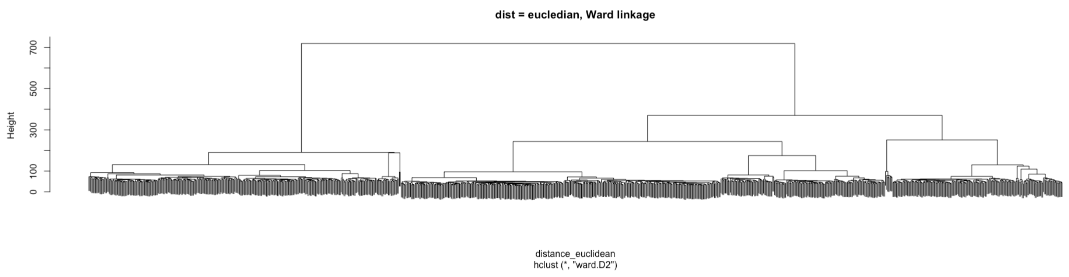
During the previous practical, we have already selected highly variable genes. This step is also to decide which genes to use when clustering the cells. Single cell RNA-seq can profile a huge number of genes in a lot of cells. But most of the genes are not expressed enough to provide a meaningful signal and are often driven by technical noise. Including them could potentially add some unwanted signal that would blur the biological variation. Moreover gene filtering can also speed up the computational time for downstream analysis.

```
pbmc <- readRDS('pbmc3k_DR.rds')
```

Clustering

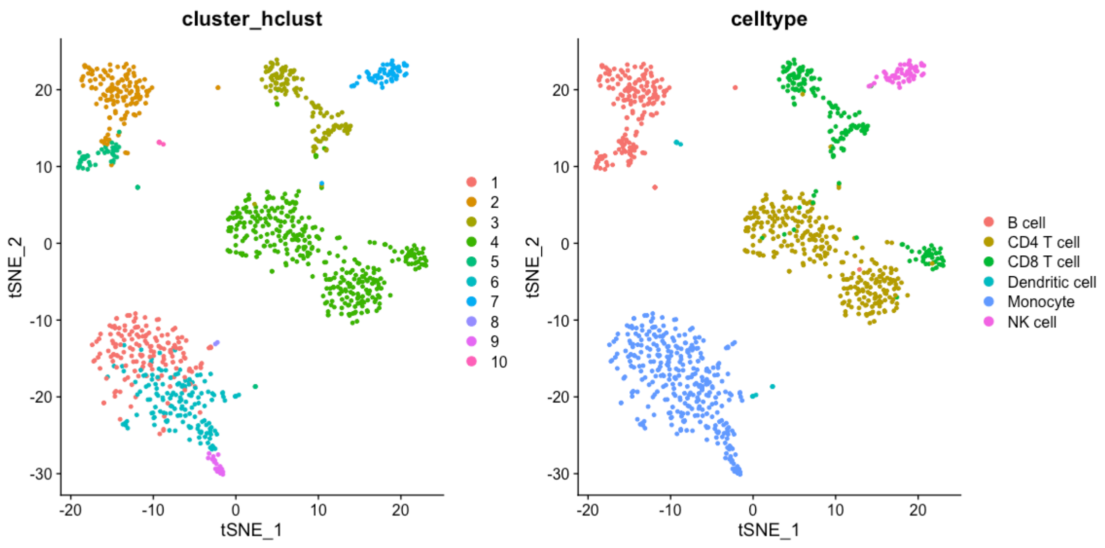
Hierarchical clustering

```
# Get scaled counts from the Seurat object
scaled_pbmc <- pbmc@assays$RNA@scale.data

# Calculate Distances (default: Euclidean distance)
distance_euclidean <- dist(t(scaled_pbmc))

#Perform hierarchical clustering using ward linkage
ward_hclust_euclidean <- hclust(distance_euclidean,method = "ward.D2")
plot(ward_hclust_euclidean, main = "dist = eucledian, Ward linkage", labels=F
ALSE)
```



Now cut the dendrogram to generate 10 clusters and plot the cluster labels and the previously given celltype labels on the t-SNE plot. For now, we just pick 10, but you can of course vary this number to see how it influences your results.

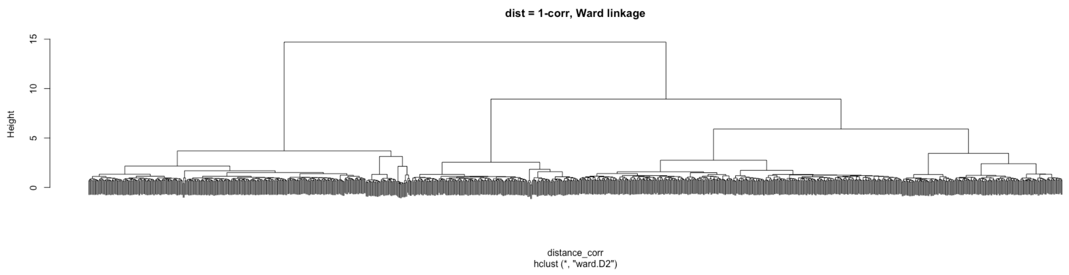
```
#Cutting the cluster tree to make 10 groups
cluster_hclust <- cutree(ward_hclust_euclidean,k = 10)
pbmc@meta.data$cluster_hclust <- factor(cluster_hclust)

p1 <- DimPlot(pbmc, reduction="tsne", group.by = "cluster_hclust")
p2 <- DimPlot(pbmc, reduction="tsne", group.by = "celltype")

p1+p2
```



Now let's try a different distance measure. A commonly used distance measure is 1 - correlation.

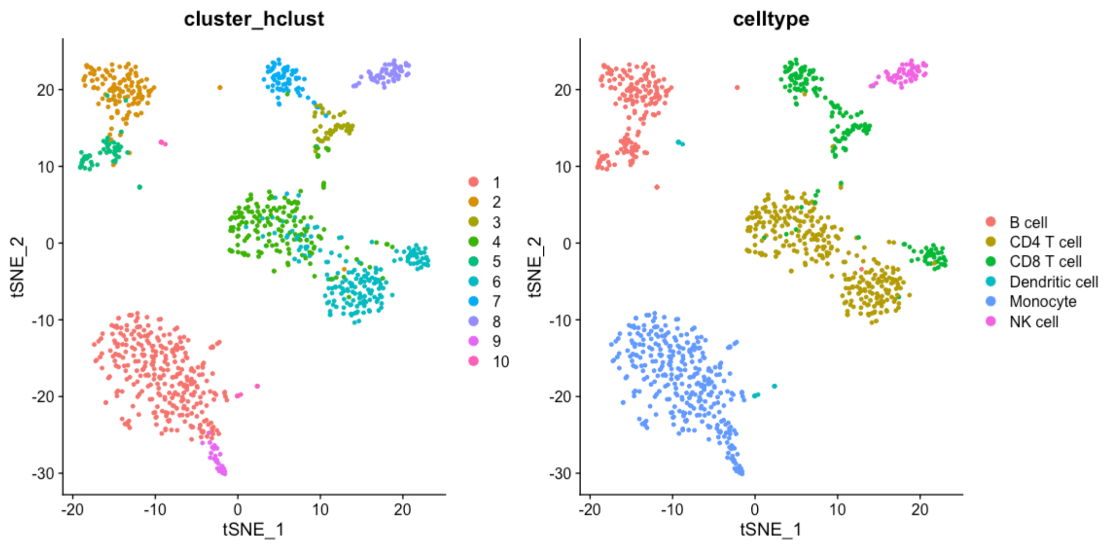
```
# Calculate Distances (1 - correlation)
C <- cor(scaled_pbmc)

# Run clustering based on the correlations, where the distance will
# be 1-correlation, e.g. higher distance with lower correlation.
distance_corr <- as.dist(1-C)

#Perform hierarchical clustering using ward linkage
```

```
ward_hclust_corr <- hclust(distance_corr,method="ward.D2")
plot(ward_hclust_corr, main = "dist = 1-corr, Ward linkage", labels=FALSE)
```



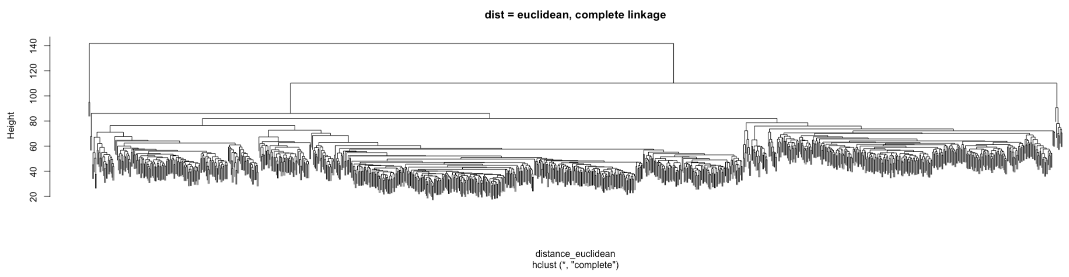
Again, let's cut the dendrogram to generate 10 clusters and plot the cluster labels on the t-SNE plot.

```
#Cutting the cluster tree to make 10 groups
cluster_hclust <- cutree(ward_hclust_corr,k = 10)
pbmc@meta.data$cluster_hclust <- factor(cluster_hclust)

p1 <- DimPlot(pbmc, reduction="tsne", group.by = "cluster_hclust")
p2 <- DimPlot(pbmc, reduction="tsne", group.by = "celltype")

p1+p2
```



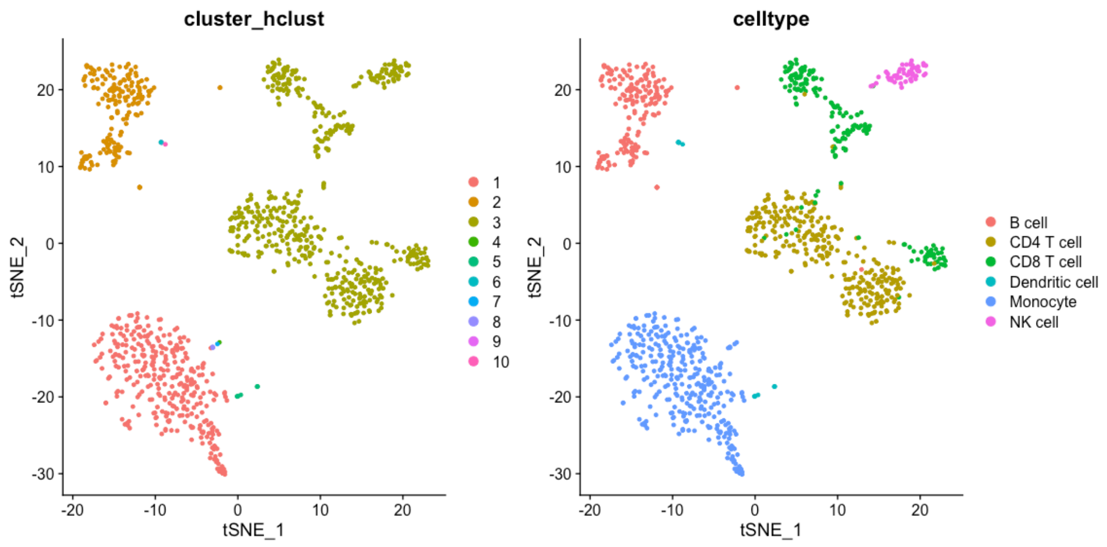
Instead of changing the distance metric, we can change the linkage method. Instead of using Ward's method, let's use complete linkage.

```
#Perform hierarchical clustering using complete linkage & euclidean distance
comp_hclust_eucledian <- hclust(distance_euclidean,method = "complete")
plot(comp_hclust_eucledian, main = "dist = euclidean, complete linkage", labe
ls=FALSE)
```



Once more, let's cut the dendrogram to generate 10 clusters and plot the cluster labels on the t-SNE plot.

```
#Cutting the cluster tree to make 10 groups
cluster_hclust <- cutree(comp_hclust_eucledian,k = 10)
pbmc@meta.data$cluster_hclust <- factor(cluster_hclust)

p1 <- DimPlot(pbmc, reduction="tsne", group.by = "cluster_hclust")
p2 <- DimPlot(pbmc, reduction="tsne", group.by = "celltype")

p1+p2
```



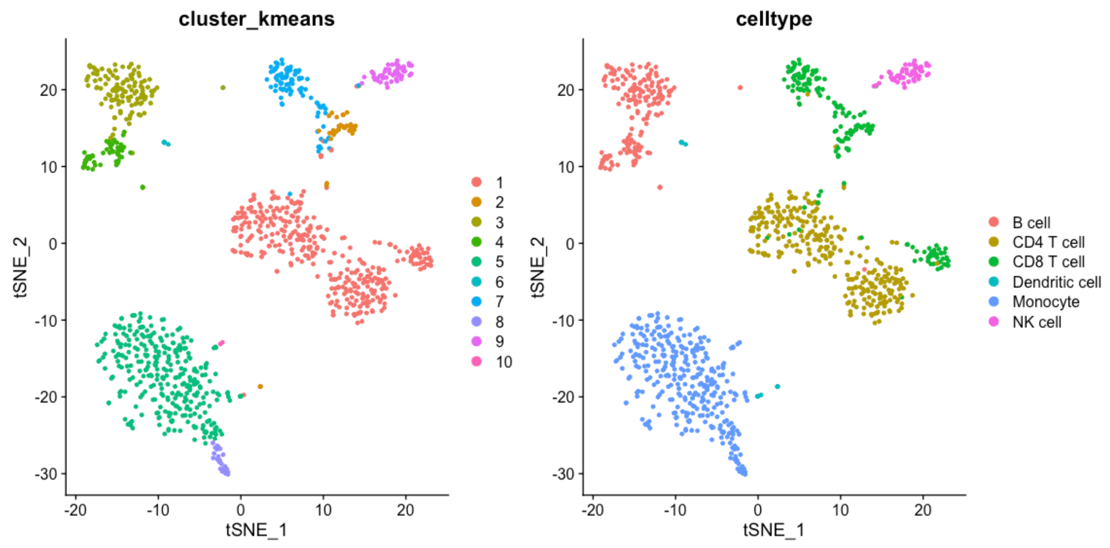
As you can see, these linkage methods and distances cluster the data differently. If you want, there are even more distance measures and linkage methods to play around with.

K-means

Next, we will try the k-means algorithm on the scaled data.

```
pbmc_kmeans <- kmeans(x = t(scaled_pbmc), centers = 10)
pbmc@meta.data$cluster_kmeans <- factor(pbmc_kmeans$cluster)

p1 <- DimPlot(pbmc, reduction="tsne", group.by = "cluster_kmeans")
p2 <- DimPlot(pbmc, reduction="tsne", group.by = "celltype")
```

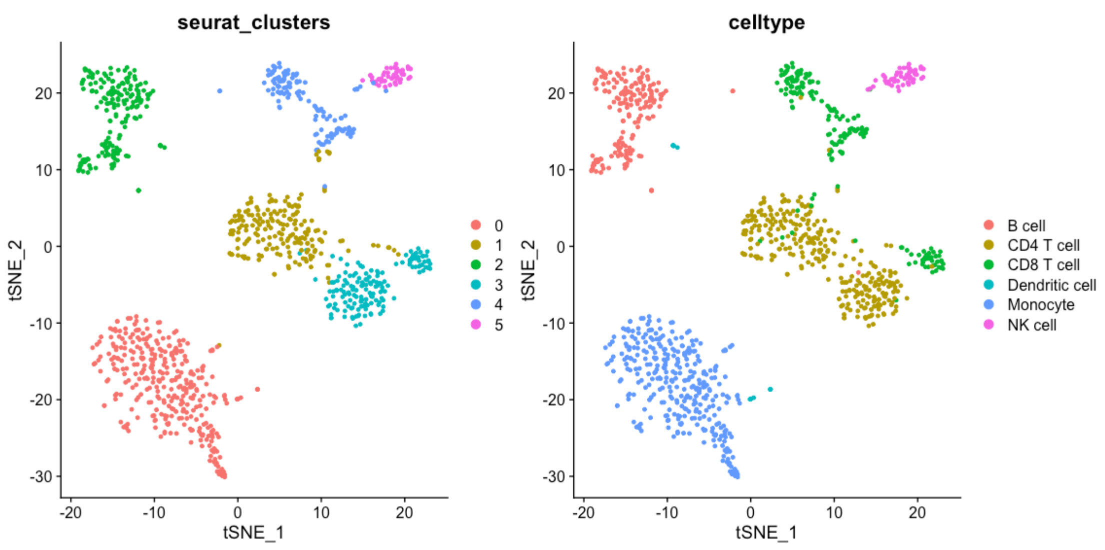
```
p1+p2
```



Graph based clustering

The clustering algorithm of Seurat itself is based on graph based clustering. The output of the clustering, will be saved automatically in the metadata as 'seurat_clusters'. As explained in the lecture, the resolution parameter is related to the number of clusters. You can play around with this parameters to see how it influences the results.

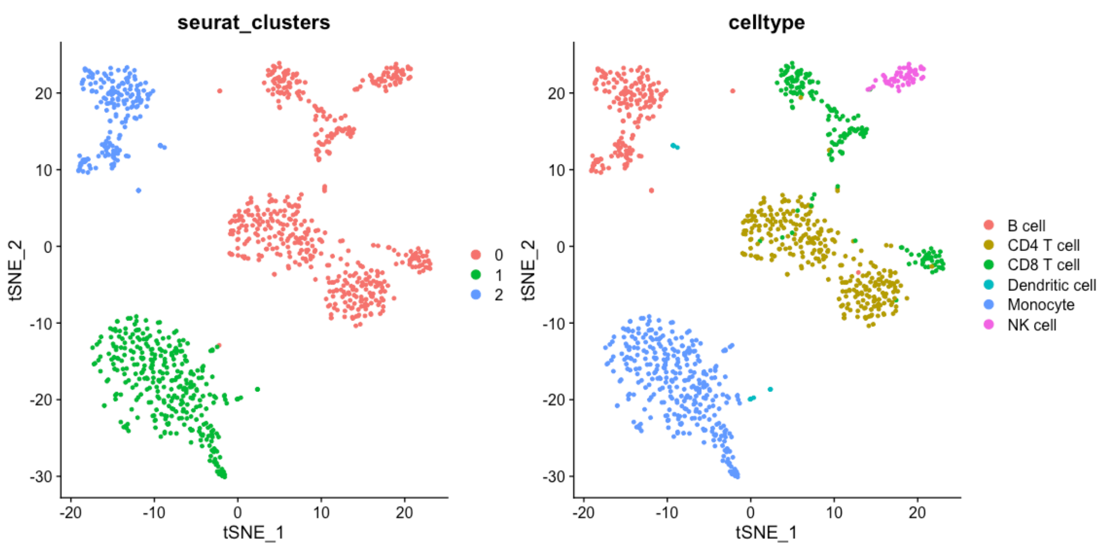
```
pbmc <- FindNeighbors(pbmc, dims = 1:10, verbose = FALSE)
pbmc <- FindClusters(pbmc, resolution = 0.25, verbose = FALSE)

p1 <- DimPlot(pbmc, reduction="tsne", group.by = "seurat_clusters")
p2 <- DimPlot(pbmc, reduction="tsne", group.by = "celltype")

p1+p2
```



Visualizing marker genes and annotating the cells

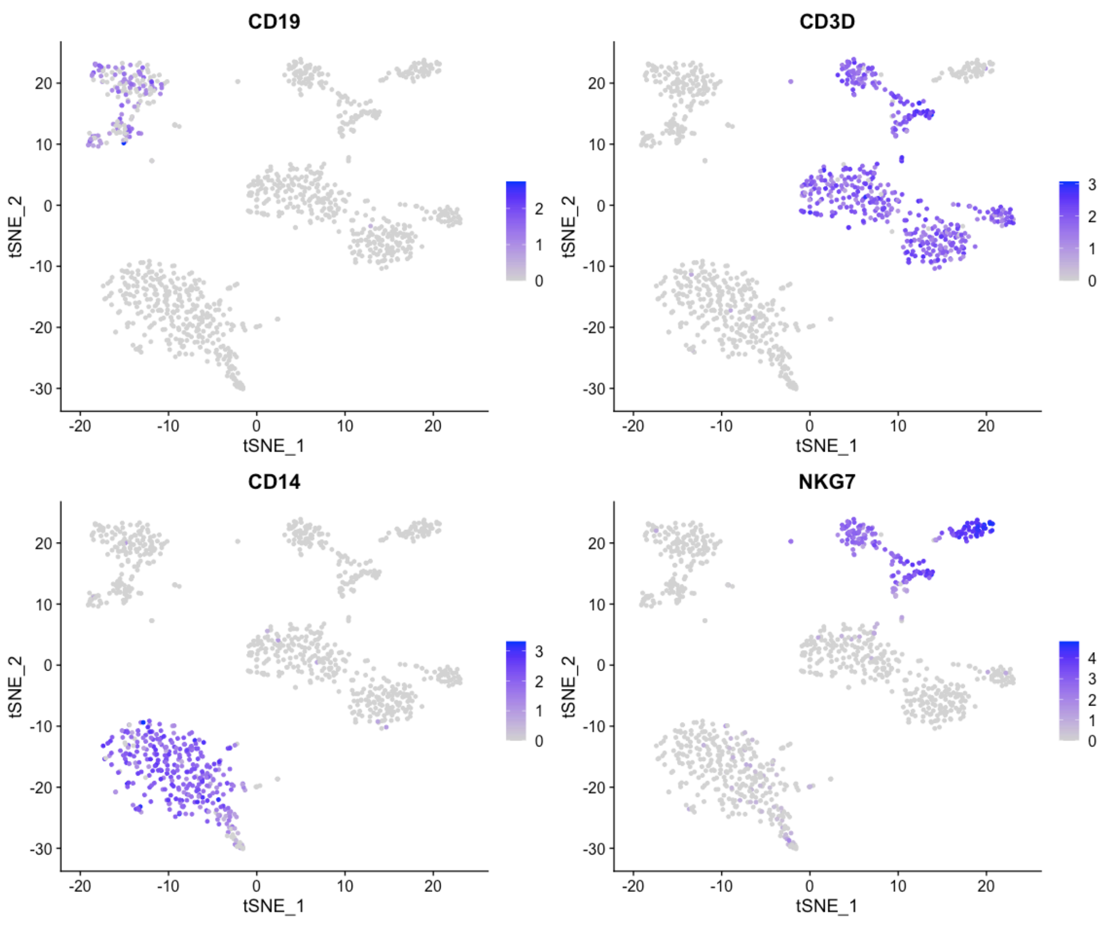
Once, you are satisfied with the clusters, these can be annotated by visualizing known marker genes or by looking at differentially expressed genes. In a later practical, you will learn how to select these, for now we will just focus on known marker genes. A commonly used approach is that the data is annotated in a hierarchical fashion. First the data is annotated at a low resolution (e.g. only 2-3 cell types) and afterwards each cluster is subsetted from the data, clustered and annotated again. This process can continue until you're satisfied with the resolution.

```
pbmc <- FindNeighbors(pbmc, dims = 1:10, verbose = FALSE)
pbmc <- FindClusters(pbmc, resolution = 0.01, verbose = FALSE)

p1 <- DimPlot(pbmc, reduction="tsne", group.by = "seurat_clusters")
p2 <- DimPlot(pbmc, reduction="tsne", group.by = "celltype")

p1+p2
```



So now that we have clustered the data at a low resolution, we can visualize some marker genes: CD19 (B cells), CD3D (T cells), CD14 (Monocytes), NKG7 (NK cells).

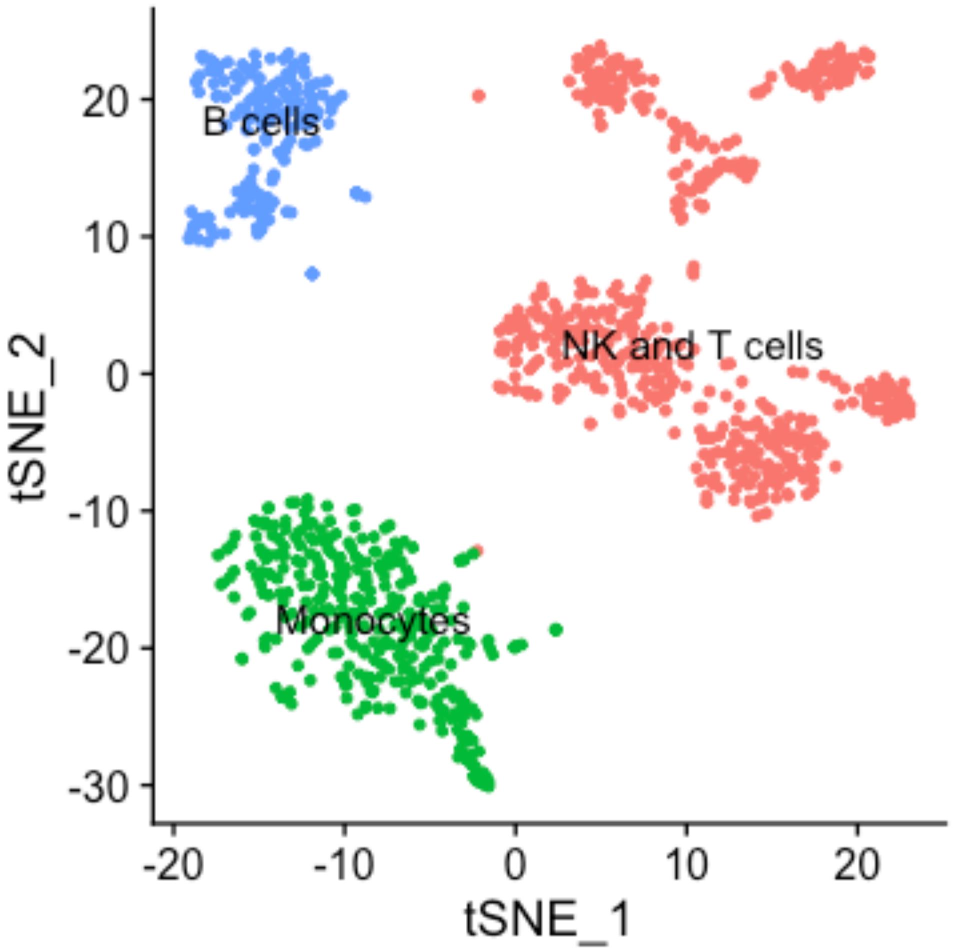
```
FeaturePlot(pbmc, reduction='tsne', features=c('CD19', 'CD3D', 'CD14', 'NKG7'
))
```



***Question 6** Find clusters at different resolutions: Find clusters at resolution e.g. 0.1, 0.4 and 0.6. Save the plots in p2, p3 and p4 plots and thereafter plot all four plots together. Find what differs when changing these resolutions.*

For a new, more complex dataset, you will probably need to visualize more genes before you can label a cluster. For now, we will assume that cluster 0 are NK and T cells, cluster 1 are Monocytes and cluster 2 are B cells. In the code below, you will assign these labels to your cluster.

```
new.cluster.ids <- c("NK and T cells", "Monocytes", "B cells")
names(new.cluster.ids) <- levels(pbmc)
pbmc <- RenameIdents(pbmc, new.cluster.ids)
DimPlot(pbmc, reduction = "tsne", label = TRUE) + NoLegend()
```



If you want to cluster the cells at a higher resolution, you could for instance subset the data now and repeat these steps. For now, we will just save the object for the next practicals.

```
saveRDS(pbmc, file = "pbmc3k_Clust.rds")
```

Session info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.6
## 
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRbla
s.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlap
ack.dylib
## 
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
## 
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base     
## 
## other attached packages:
## [1] SeuratObject_4.0.2 Seurat_4.0.4      
## 
## loaded via a namespace (and not attached):
##   [1] nlme_3.1-153          spatstat.sparse_2.0-0 matrixStats_0.60.1   
##   [4] RcppAnnoy_0.0.19      RColorBrewer_1.1-2    httr_1.4.2           
##   [7] sctransform_0.3.2     tools_4.1.0           utf8_1.2.2           
##  [10] R6_2.5.1              irlba_2.3.3           rpart_4.1-15         
##  [13] KernSmooth_2.23-20    uwot_0.1.10           mgcv_1.8-36          
##  [16] DBI_1.1.1             lazyeval_0.2.2        colorspace_2.0-2     
##  [19] tidyselect_1.1.1      gridExtra_2.3         compiler_4.1.0       
##  [22] plotly_4.9.4.1        labeling_0.4.2        scales_1.1.1         
##  [25] spatstat.data_2.1-0   lmtest_0.9-38         ggridges_0.5.3       
##  [28] pbapply_1.5-0         goftest_1.2-2         stringr_1.4.0        
##  [31] digest_0.6.27         spatstat.utils_2.2-0  rmarkdown_2.11       
##  [34] pkgconfig_2.0.3       htmltools_0.5.2       parallelly_1.28.1    
##  [37] highr_0.9             fastmap_1.1.0         htmlwidgets_1.5.4    
##  [40] rlang_0.4.11          shiny_1.6.0           farver_2.1.0         
##  [43] generics_0.1.0        zoo_1.8-9             jsonlite_1.7.2       
##  [46] ica_1.0-2             dplyr_1.0.7           magrittr_2.0.1       
##  [49] patchwork_1.1.1       Matrix_1.3-4          Rcpp_1.0.7           
##  [52] munsell_0.5.0         fansi_0.5.0           abind_1.4-5          
##  [55] reticulate_1.22       lifecycle_1.0.0       stringi_1.7.4        
##  [58] yaml_2.2.1            MASS_7.3-54           Rtsne_0.15           
##  [61] plyr_1.8.6            grid_4.1.0            parallel_4.1.0       
##  [64] listenv_0.8.0         promises_1.2.0.1      ggrepel_0.9.1        
##  [67] crayon_1.4.1          deldir_0.2-10         miniUI_0.1.1.1       
##  [70] lattice_0.20-44       cowplot_1.1.1         splines_4.1.0        
##  [73] tensor_1.5            knitr_1.34            pillar_1.6.2         
##  [76] igraph_1.2.6          spatstat.geom_2.2-2   future.apply_1.8.1   
##  [79] reshape2_1.4.4        codetools_0.2-18      leiden_0.3.9         
##  [82] glue_1.4.2            evaluate_0.14         data.table_1.14.0
##  [85] png_0.1-7             vctrs_0.3.8           httpuv_1.6.3
##  [88] polyclip_1.10-0       gtable_0.3.0          RANN_2.6.1
##  [91] purrr_0.3.4           spatstat.core_2.3-0   tidyr_1.1.3
##  [94] scattermore_0.7       future_1.22.1         assertthat_0.2.1
##  [97] ggplot2_3.3.5         xfun_0.26             mime_0.11
## [100] xtable_1.8-4          later_1.3.0           survival_3.2-13
## [103] viridisLite_0.4.0     tibble_3.1.4          cluster_2.1.2
## [106] globals_0.14.0        fitdistrplus_1.1-5    ellipsis_0.3.2
## [109] ROCR_1.0-11
```