SCchicbackanalysis

Laurens-Willem Janssen

2023-07-13

This is an R script made to perform a single-cell analysis on chicken skin back data. Made by Laurens-Willem Janssen (s1114416) from Hogeschool Leiden. For Leiden University, guided by Michael Richardson and Luthfi Nurhidayat.

Loading in the necessary packages.

```
library(Seurat, scater)
library(scran)
library(Matrix, magrittr)
lapply(c("dplyr", "HGNChelper", "openxlsx", "tidyverse", "data.tree", "igraph", "ggraph"), library, character.onl
y = T)
```

```
## [[1]]
##  [1] "dplyr"                "Matrix"               "scran"
##  [4] "scuttle"              "SingleCellExperiment" "SummarizedExperiment"
##  [7] "Biobase"              "GenomicRanges"        "GenomeInfoDb"
## [10] "IRanges"              "S4Vectors"            "BiocGenerics"
## [13] "stats4"               "MatrixGenerics"       "matrixStats"
## [16] "SeuratObject"         "Seurat"               "stats"
## [19] "graphics"             "grDevices"            "utils"
## [22] "datasets"             "methods"              "base"
##
## [[2]]
##  [1] "HGNChelper"           "dplyr"                "Matrix"
##  [4] "scran"                "scuttle"              "SingleCellExperiment"
##  [7] "SummarizedExperiment" "Biobase"              "GenomicRanges"
## [10] "GenomeInfoDb"         "IRanges"              "S4Vectors"
## [13] "BiocGenerics"         "stats4"               "MatrixGenerics"
## [16] "matrixStats"          "SeuratObject"         "Seurat"
## [19] "stats"                "graphics"             "grDevices"
## [22] "utils"                "datasets"             "methods"
## [25] "base"
##
## [[3]]
##  [1] "openxlsx"             "HGNChelper"           "dplyr"
##  [4] "Matrix"               "scran"                "scuttle"
##  [7] "SingleCellExperiment" "SummarizedExperiment" "Biobase"
## [10] "GenomicRanges"        "GenomeInfoDb"         "IRanges"
## [13] "S4Vectors"            "BiocGenerics"         "stats4"
## [16] "MatrixGenerics"       "matrixStats"          "SeuratObject"
## [19] "Seurat"               "stats"                "graphics"
## [22] "grDevices"            "utils"                "datasets"
## [25] "methods"              "base"
##
## [[4]]
##  [1] "lubridate"            "forcats"              "stringr"
##  [4] "purrr"                "readr"                "tidyr"
##  [7] "tibble"               "ggplot2"              "tidyverse"
## [10] "openxlsx"             "HGNChelper"           "dplyr"
## [13] "Matrix"               "scran"                "scuttle"
## [16] "SingleCellExperiment" "SummarizedExperiment" "Biobase"
## [19] "GenomicRanges"        "GenomeInfoDb"         "IRanges"
## [22] "S4Vectors"            "BiocGenerics"         "stats4"
## [25] "MatrixGenerics"       "matrixStats"          "SeuratObject"
## [28] "Seurat"               "stats"                "graphics"
## [31] "grDevices"            "utils"                "datasets"
## [34] "methods"              "base"
##
## [[5]]
##  [1] "data.tree"            "lubridate"            "forcats"
##  [4] "stringr"              "purrr"                "readr"
##  [7] "tidyr"                "tibble"               "ggplot2"
## [10] "tidyverse"            "openxlsx"             "HGNChelper"
## [13] "dplyr"                "Matrix"               "scran"
## [16] "scuttle"              "SingleCellExperiment" "SummarizedExperiment"
## [19] "Biobase"              "GenomicRanges"        "GenomeInfoDb"
## [22] "IRanges"              "S4Vectors"            "BiocGenerics"
## [25] "stats4"               "MatrixGenerics"       "matrixStats"
## [28] "SeuratObject"         "Seurat"               "stats"
## [31] "graphics"             "grDevices"            "utils"
## [34] "datasets"             "methods"              "base"
##
## [[6]]
##  [1] "igraph"               "data.tree"            "lubridate"
##  [4] "forcats"              "stringr"              "purrr"
##  [7] "readr"                "tidyr"                "tibble"
## [10] "ggplot2"              "tidyverse"            "openxlsx"
## [13] "HGNChelper"           "dplyr"                "Matrix"
## [16] "scran"                "scuttle"              "SingleCellExperiment"
## [19] "SummarizedExperiment" "Biobase"              "GenomicRanges"
## [22] "GenomeInfoDb"         "IRanges"              "S4Vectors"
## [25] "BiocGenerics"         "stats4"               "MatrixGenerics"
## [28] "matrixStats"          "SeuratObject"         "Seurat"
## [31] "stats"                "graphics"             "grDevices"
## [34] "utils"                "datasets"             "methods"
## [37] "base"
##
## [[7]]
##  [1] "ggraph"               "igraph"               "data.tree"
##  [4] "lubridate"            "forcats"              "stringr"
##  [7] "purrr"                "readr"                "tidyr"
## [10] "tibble"               "ggplot2"              "tidyverse"
## [13] "openxlsx"             "HGNChelper"           "dplyr"
## [16] "Matrix"               "scran"                "scuttle"
## [19] "SingleCellExperiment" "SummarizedExperiment" "Biobase"
## [22] "GenomicRanges"        "GenomeInfoDb"         "IRanges"
## [25] "S4Vectors"            "BiocGenerics"         "stats4"
## [28] "MatrixGenerics"       "matrixStats"          "SeuratObject"
## [31] "Seurat"               "stats"                "graphics"
## [34] "grDevices"            "utils"                "datasets"
## [37] "methods"              "base"
```

Reading the Cell Ranger data.

```
back <- Read10X(data.dir = "filtered_feature_bc_matrixB")
back <- CreateSeuratObject(counts = back)
```

Adding the marker for the mitochondrial genes, as it is not present in the raw Cell Ranger output.

```
mtlist = c("^ND1", "^ND2", "^COX1", "^COX2", "^ATP8", "^ATP6", "^COX3", "^ND3",
           "^ND4L", "^ND4", "^ND5", "^CYTB", "^ND6")
mtgene <- grep(paste(mtlist, collapse = "|"),
               back@assays[["RNA"]]@counts@Dimnames[[1]], value=TRUE)
mtgene <- paste(mtgene, collapse = "|")
```

Create and add the percentage of mitochondrial and ribosomal genes

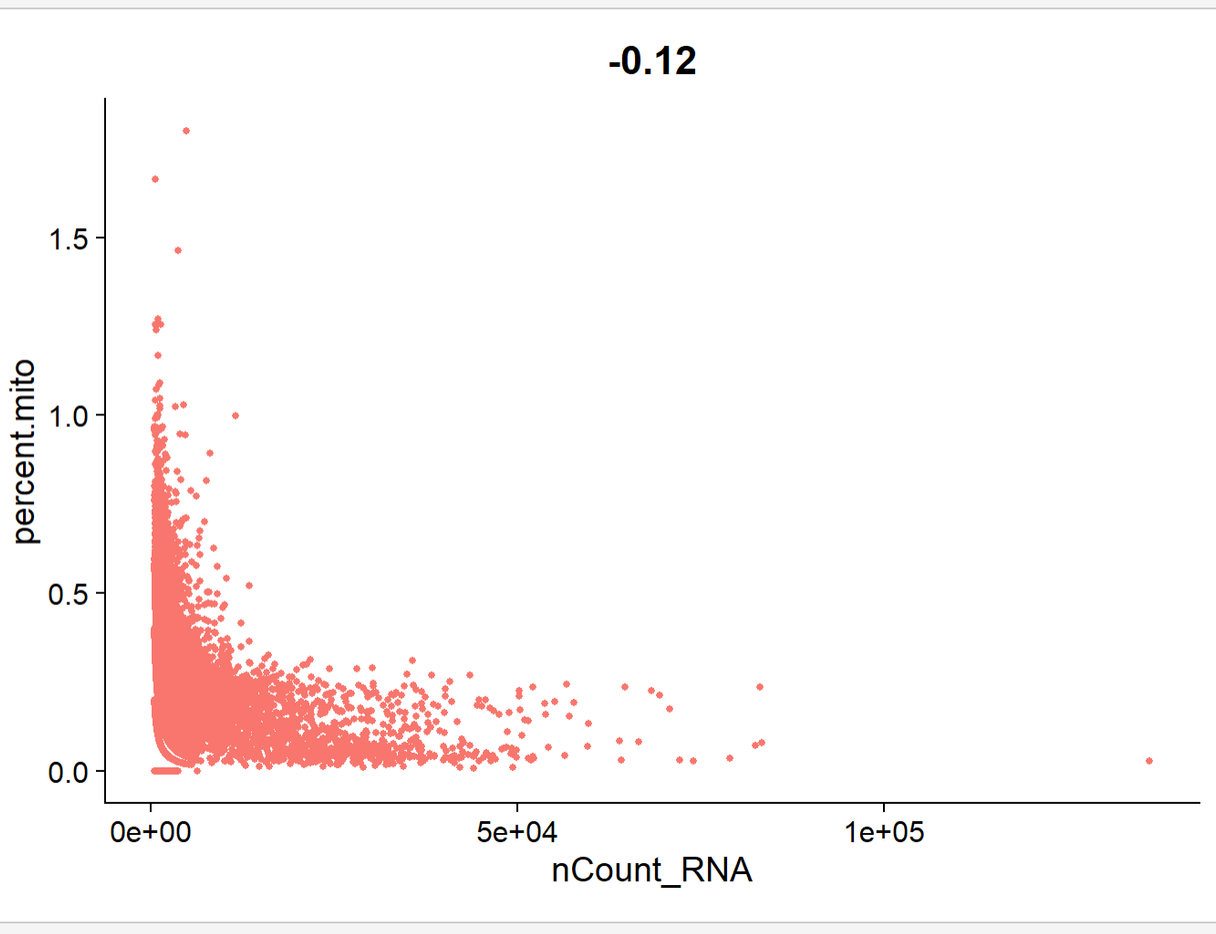
```
percent.mito <- PercentageFeatureSet(back, pattern = mtgene)
back <- AddMetaData(back, percent.mito, col.name = "percent.mito")
percent.ribo <- PercentageFeatureSet(back, pattern = "^RP[SL]")
back <- AddMetaData(back, percent.ribo, col.name = "percent.ribo")
```

Deleting the percent.mito/ribo objects to save RAM & creating the scatter plots

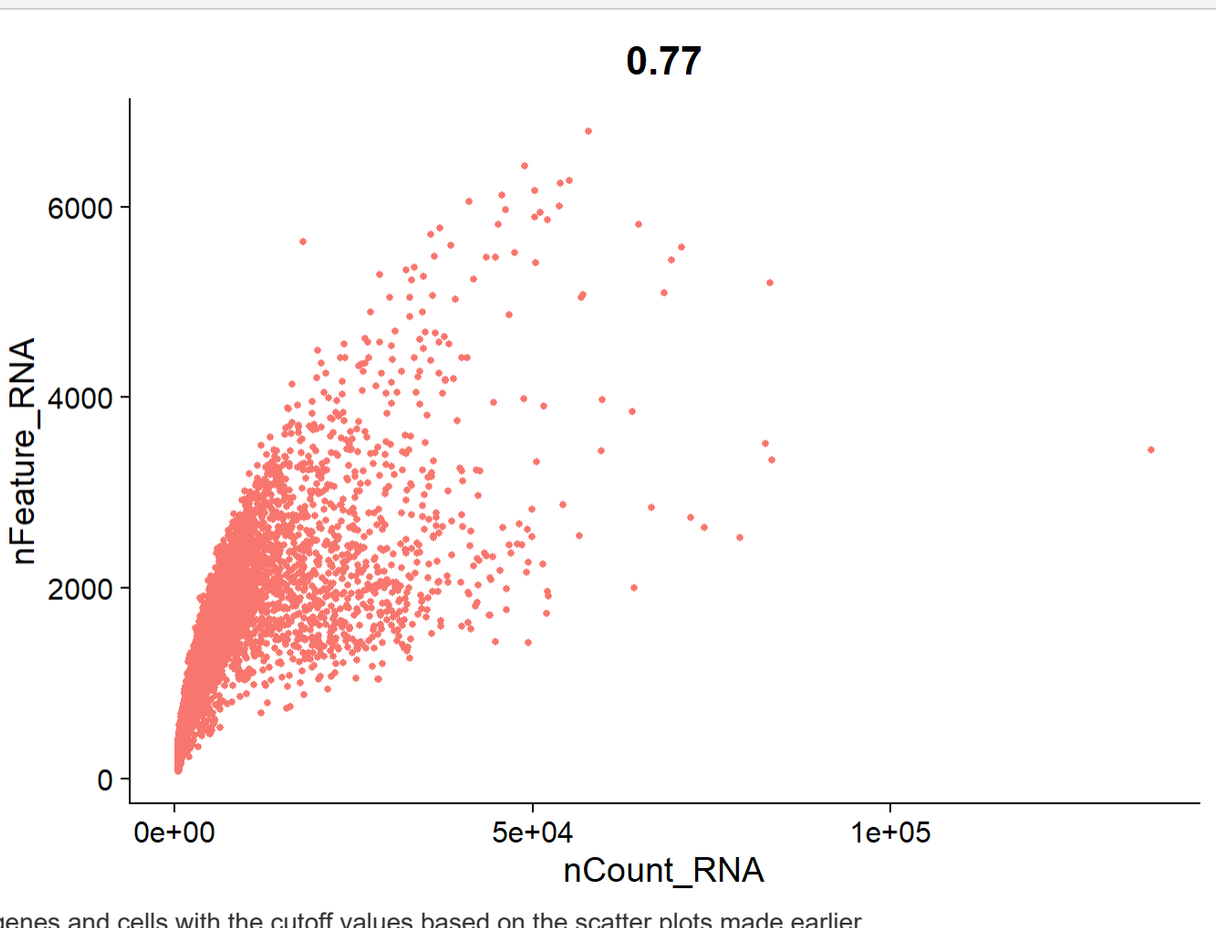
```
remove(percent.mito)
remove(percent.ribo)
remove(mtgene)
remove(mtlist)
table(Idents(back))
```

```
##
## SeuratProject
##         25700
```

```
FeatureScatter(back, feature1 = "nCount_RNA", feature2 = "percent.mito") + NoLegend()
```




```
FeatureScatter(back, feature1 = "nCount_RNA", feature2 = "nFeature_RNA") + NoLegend()
```



Cutting off over or under expressed genes and cells with the cutoff values based on the scatter plots made earlier.

```
back <- subset(back, subset = nFeature_RNA < 5000 & nFeature_RNA > 200 & nCount_RNA < 50000)
table(Idents(back))
```

```
##
## SeuratProject
##         25608
```

Normalization and finding variable features.

```
back <- NormalizeData(back, normalization.method = "LogNormalize", scale.factor = 10000)
back <- FindVariableFeatures(back, selection.method = "vst", nfeatures = 2000)
```

Scale the data and perform the PCA.

```
allgenes <- rownames(back)
back <- ScaleData(back, features = allgenes)
```

```
## Centering and scaling data matrix
```

```
remove(allgenes)
back <- RunPCA(back, features = VariableFeatures(object = back))
```

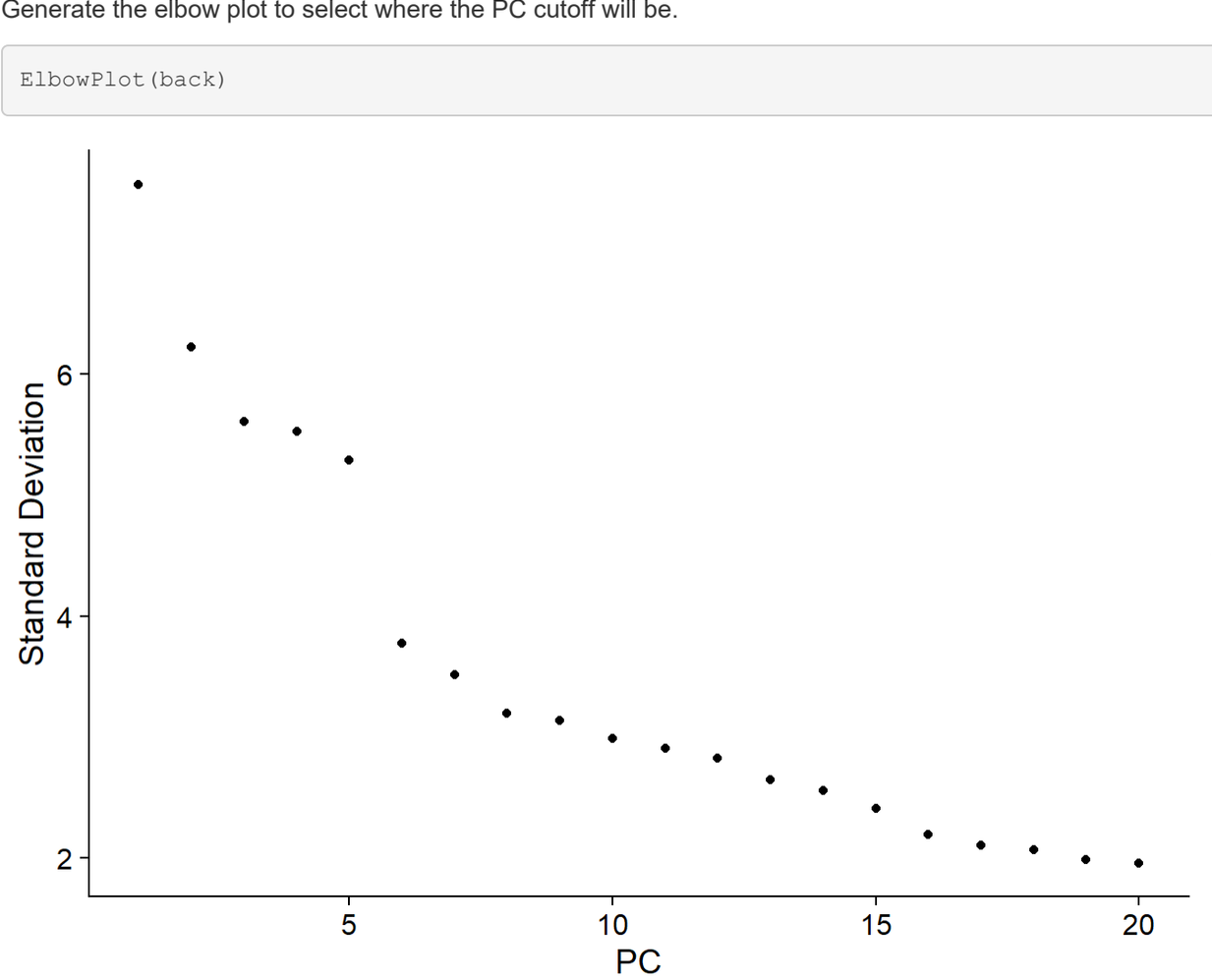
```
## PC_ 1
## Positive:  S100A16, S100A14, SFN, DYNLL1, KRT75L1, DSP, LY6CLEL, KRT8, CSTA, LOC772080
##     CLDN4, LOC423719, CD9, POF1B, PHYH, ELOVL4, LOC101751691, LOC420039, F3, TMEM254
##     FADS6, EPCAM, KRT71, LOC124417319, KRT10, LOC107055100, KRT9, LOC107049127, FABP5, SLC25A17
## Negative:  SPARC, COL3A1, DCN, COL1A2, GPX3, LOC121107581, POSTN, SERPINF1, IGFBP4, LOC107053353
##     VIM, CRIP1, COL6A2, COL1A1, MFAP5, C2orf40, FBN1, HTRA1, FN1, MUSTN1
##     RGCC, FTH1, GFPT2, PTGDS, COL6A1, SOD3, CTSK, PCOLCE2, DPT, FSTL1
## PC_ 2
## Positive:  COL1A2, DCN, COL3A1, SERPINF1, SPARC, GPX3, IGFBP4, MFAP5, POSTN, COL1A1
##     COL6A2, FBN1, SOD3, CALD1, RGCC, C2orf40, CPM, FN1, HTRA1, LY6E
##     MUSTN1, TUBA1A, GFPT2, LOC107053353, PCOLCE2, COL6A1, KRT8, DPT, FSTL1, TIMP3
## Negative:  RSFR, C1QA, C1QC, BLB1, IFI30, C1QB, LY86, LAPTM5, CD74, BLB2
##     DMA, CSF1R, RNASE6, MS4A4A, DMB2, CCL26, LBP, LOC107050532, CTSS, F13A1
##     PINLYP, TGFBI, CD83, SPI1, IL2RG, KRABZFP, FCER1G, LOC426820, LOC771876, STAB1
## PC_ 3
## Positive:  F3, KRT10, CSRP2, COL17A1, ID1, ELOVL4, NTM, AHNAK2, KRT75L1, LOC101751691
##     LOC107055100, LOC124417319, SERPINB2, KRT17, LOC423119, PI3, FGFBP1, KRT71, S100A14, SPTSSB
##     LOC107054916, CST3, PHYH, LOC107052763, DSP, GSTA4, LOC421125, LOC408038, SFN, POF1B
## Negative:  RSFR, C1QC, C1QA, BLB1, CD74, C1QB, IFI30, DMB2, LY86, RNASE6
##     LOC100857858, DMA, BLB2, CSF1R, CCL26, MS4A4A, CTSS, LBP, LOC107050532, LAPTM5
##     TUBB3, SELENOP1, KRT75L4, LOC428499, F13A1, TGFBI, PINLYP, LOC768978, TUBA1A, CD83
## PC_ 4
## Positive:  BAG2, KRT75L4, LOC768978, LOC431321, TUBB3, CSRP1, LOC769139, LMOD1, DES, FHL2
##     ACTG2, TAGLN, P2RX1, LOC100857858, LOC426913, F-KER, ACTA1, PCP4, MYLK, LOC431324
##     LOC769121, SMTN, LOC428499, LOC429492, KCNMB1, LOC426914, SYNPO2, DLX4, AK1, HSPB7
## Negative:  LOC101749151, KRT9, LOC424473, LOC431300, LOC107055100, LY6CLEL, PHYH, LY6EL, TMEM45L, LOC769512
##     LOC107051160, LOC422609, LOC408038, AvBD14, EREG, LOC121106479, CTSA, LOC101751691, LOC124417397, LOC77199
4
##     LOC101747554, LOC770450, ALDH3B1, LOC124417319, LOC101747995, TMEM254, ELOVL4, ABHD12B, PPIF, RNF222
## PC_ 5
## Positive:  LOC769139, LOC768978, KRT75L4, LOC431321, LOC426913, TUBB3, LOC100857858, LOC769121, F-KER, LOC4294
92
##     LOC431324, LOC428499, LOC426914, DLX4, LOC418813, LOC121107570, BAMBI, LOC100859249, LOC101747367, LOC1211
07568
##     LOC771995, LOC121107564, LOC100859427, UPK3BL, APOD, BMP8A, TLCD1, NPM3, LOC420039, KITLG
## Negative:  LMOD1, DES, FHL2, CSRP1, ACTG2, TAGLN, P2RX1, SMTN, ACTA1, PCP4
##     MYLK, KCNMB1, SYNPO2, MYH11, C11orf96, AK1, HSPB7, PGM5, SYNM, CAP2
##     MYL9, LOC112533410, TPM1, ACTG1, PDLIM3, EEF1A2, ADIPOQ, EXFABP, CUTAL, CCDC69
```

Performing the Jack Straw test on the PCA result.

```
back <- JackStraw(back, num.replicate = 100)
back <- ScoreJackStraw(back, dims = 1:20)
```

Generate the elbow plot to select where the PC cutoff will be.

```
ElbowPlot(back)
```



The elbow plot above tells that a cutoff around 10 PC would be the most suitable. After some trying a cutoff of 11 ended up producing the best result. A low resolution of 0.3 was chosen also, after some trial and error.

```
back <- FindNeighbors(back, dims = 1:11)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```
back <- FindClusters(back, resolution = 0.3)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 25608
## Number of edges: 810972
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9434
## Number of communities: 14
## Elapsed time: 6 seconds
```

```
back <- RunUMAP(back, dims = 1:11)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R-native UW
OT using the cosine metric
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session
```

```
## 12:53:39 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 12:53:39 Read 25608 rows and found 11 numeric columns
```

```
## 12:53:39 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 12:53:39 Building Annoy index with metric = cosine, n_trees = 50
```

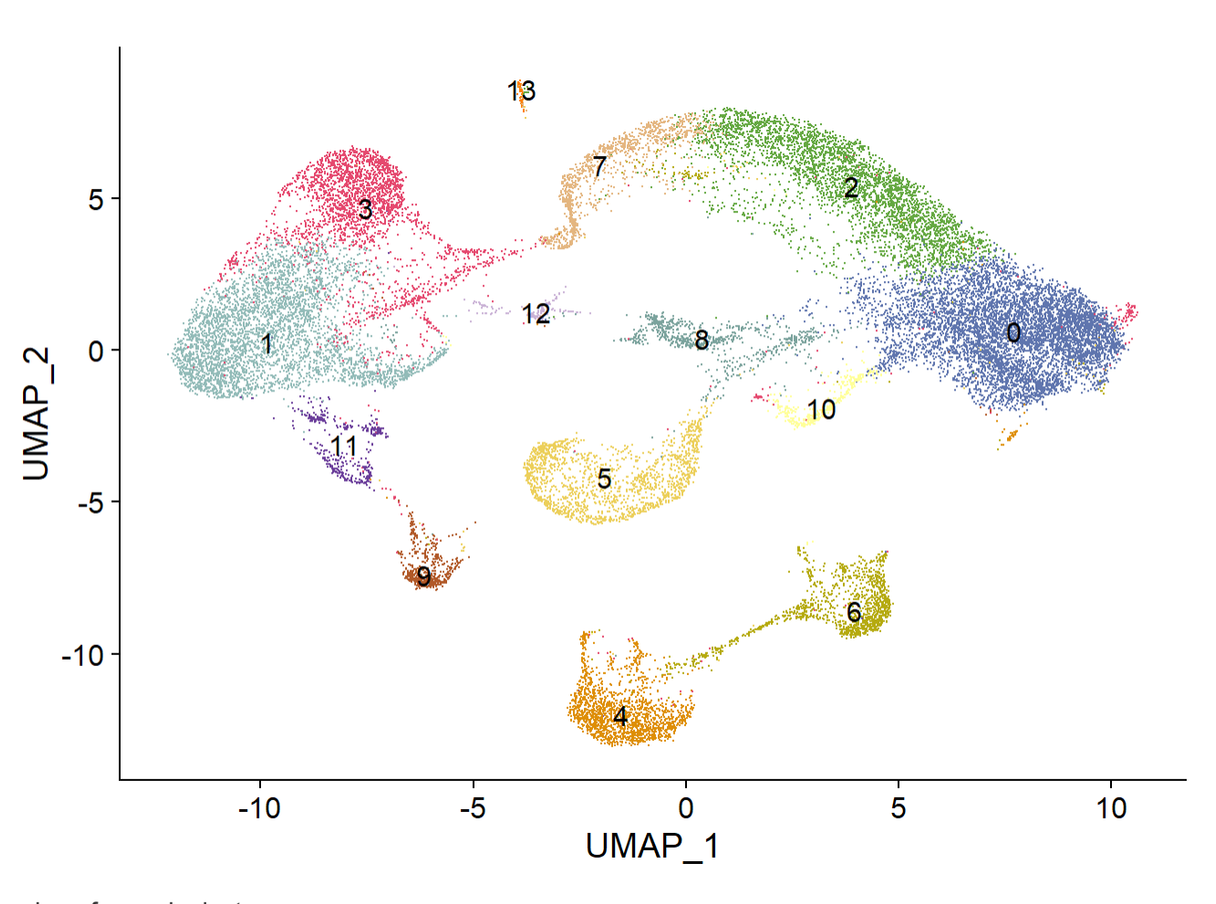
```
## 0%   10   20   30   40   50   60   70   80   90   100%
```

```
## [----|----|----|----|----|----|----|----|----|----|
```

```
## **************************************************|
## 12:53:44 Writing NN index file to temp file C:\Users\bestb\AppData\Local\Temp\Rtmps9ndFD\file230426c9726
## 12:53:44 Searching Annoy index using 1 thread, search_k = 3000
## 12:53:56 Annoy recall = 100%
## 12:53:58 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 12:54:01 Initializing from normalized Laplacian + noise (using irlba)
## 12:54:03 Commencing optimization for 200 epochs, with 1103996 positive edges
## 12:54:37 Optimization finished
```

Creating the list of colors that will be used for visualizing the fourteen generated clusters, and generating the UMAP showing the clusters.

```
ccolss= c("#5f75ae","#92bbb8","#64a841","#e5486e","#de8e06","#eccf5a","#b5aa0f","#e4b680","#7ba39d","#b15928","#f
fff99", "#6a3d9a","#cab2d6","#ff7f00","#fdbf6f","#e31a1c","#fb9a99","#33a02c","#b2df8a","#1f78b4","#a6cee3")
DimPlot(back, reduction = "umap", label = TRUE, cols = ccolss) + NoLegend()
```



Calculates what genes are the best markers for each cluster.

```
markers <- FindAllMarkers(back, only.pos = TRUE, min.pct = 0, logfc.threshold = 0.25)
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```
## Calculating cluster 3
```

```
## Calculating cluster 4
```

```
## Calculating cluster 5
```

```
## Calculating cluster 6
```

```
## Calculating cluster 7
```

```
## Calculating cluster 8
```

```
## Calculating cluster 9
```

```
## Calculating cluster 10
```

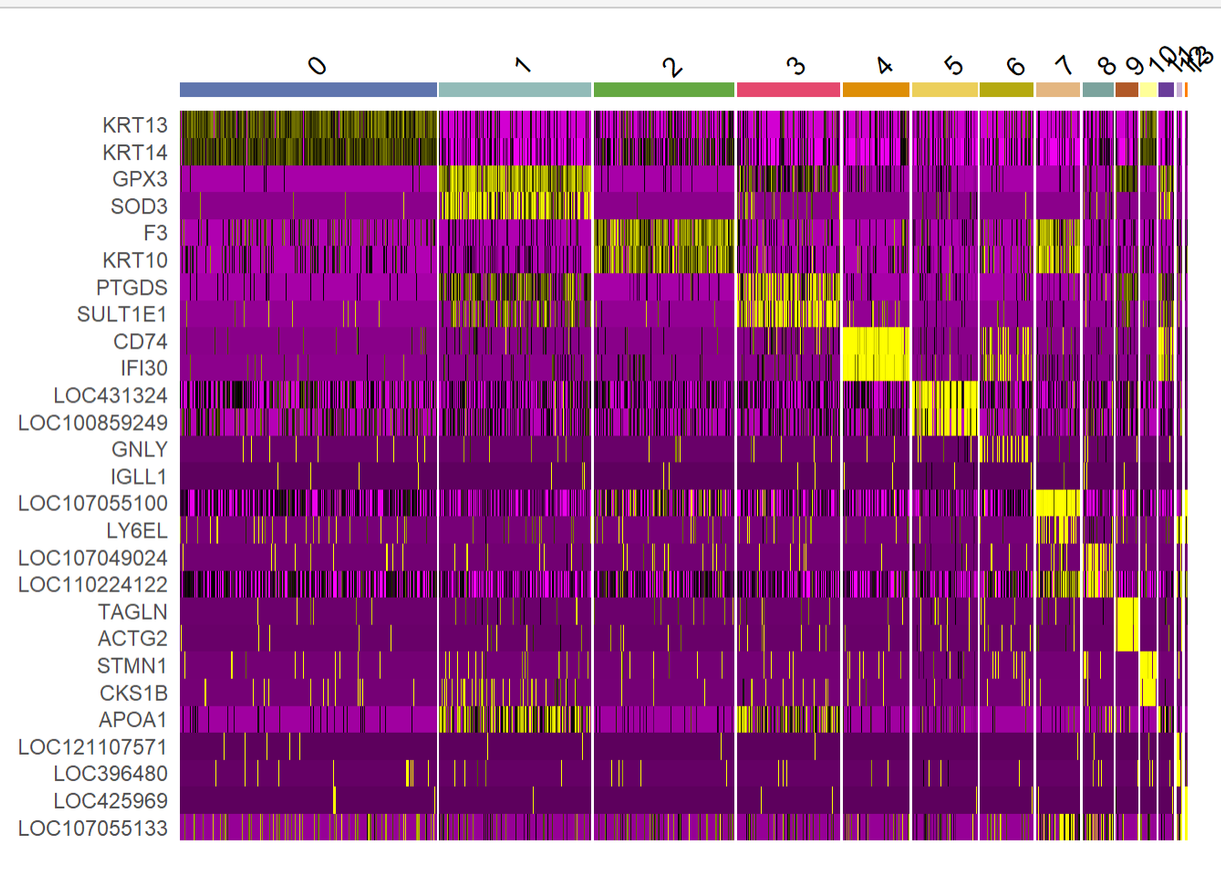
```
## Calculating cluster 11
```

```
## Calculating cluster 12
```

```
## Calculating cluster 13
```

Making the heat map that shows the expression levels of the discovered marker genes for each cluster.

```
markers %>%
  group_by(cluster) %>%
  top_n(n = 2, wt = avg_log2FC) -> topgene
DoHeatmap(back, features = topgene$gene, group.colors = ccolss, size = 4) + NoLegend()
```



Load the training and reference data needed for the cell type annotation with ScType.

```
source("https://raw.githubusercontent.com/IanevskiAleksandr/sc-type/master/R/gene_sets_prepare.R"); source("http
s://raw.githubusercontent.com/IanevskiAleksandr/sc-type/master/R/sctype_score_.R")
gs_list = gene_sets_prepare("ScTypeDB_full.xlsx", "Chic")
```

```
## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols

## Warning in checkGeneSymbols(markers_all): x contains non-approved gene symbols
```

```
db_ = "https://raw.githubusercontent.com/IanevskiAleksandr/sc-type/master/ScTypeDB_full.xlsx";
```

The next commands are part of the ScType ‘pipeline’ for annotating the most characteristic cell type in each cluster.

```
es = sctype_score(scRNAseqData = back[["RNA"]]@scale.data, scaled = TRUE,
                  gs = gs_list$gs_positive, gs2 = gs_list$gs_negative)
```

```
cL_resutls = do.call("rbind", lapply(unique(back@meta.data$seurat_clusters), function(cl){
  es = sort(rowSums(es[ ,rownames(back@meta.data[back@meta.data$seurat_clusters==cl, ])]), decreasing = !0)
  head(data.frame(cluster = cl, type = names(es), scores = es, ncells = sum(back@meta.data$seurat_clusters==cl)),
10)}))
sctype_scores = cL_resutls %>% group_by(cluster) %>% top_n(n = 1, wt = scores)
```

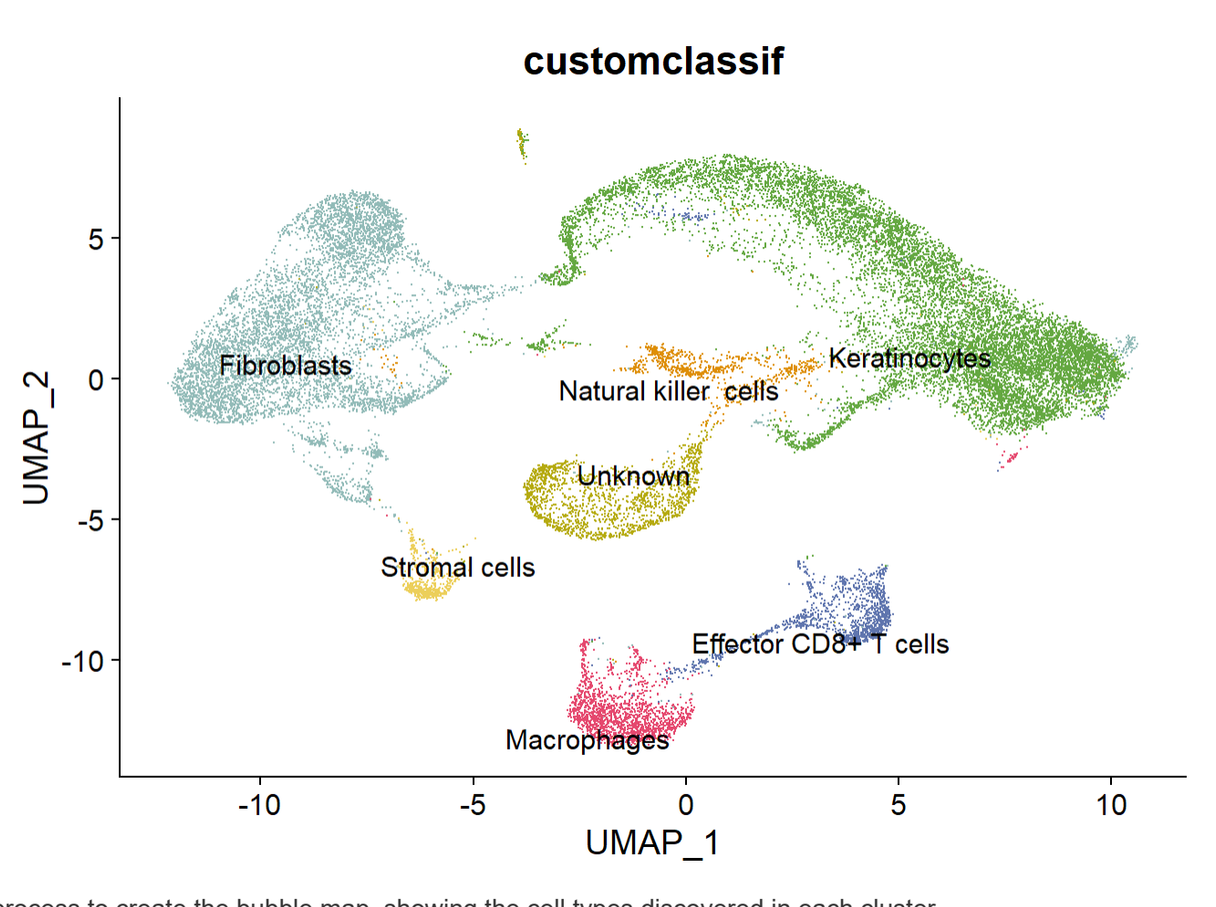
```
sctype_scores$type[as.numeric(as.character(sctype_scores$scores)) < sctype_scores$ncells/4] = "Unknown"
print(sctype_scores[,1:3])
```

```
## # A tibble: 14 × 3
## # Groups:   cluster [14]
##    cluster type                   scores
##    <fct>   <chr>                   <dbl>
##  1 3       Fibroblasts            6950.
##  2 1       Fibroblasts           22386.
##  3 4       Macrophages            3052.
##  4 0       Keratinocytes          2839.
##  5 10      Keratinocytes           241.
##  6 12      Keratinocytes            56.2
##  7 6       Effector CD8+ T cells  2997.
##  8 5       Unknown                 313.
##  9 7       Keratinocytes          1507.
## 10 11      Fibroblasts            1719.
## 11 2       Keratinocytes          4792.
## 12 8       Natural killer  cells   494.
## 13 9       Stromal cells           638.
## 14 13      Unknown                   9.58
```

```
back@meta.data$customclassif = ""
for(j in unique(sctype_scores$cluster)){
  cl_type = sctype_scores[sctype_scores$cluster==j,];
  back@meta.data$customclassif[back@meta.data$seurat_clusters == j] = as.character(cl_type$type[1])
}
```

Generate the UMAP showing the cell type for each data point.

```
DimPlot(back, reduction = "umap", label = TRUE, repel = TRUE, group.by = 'customclassif', cols = ccolss) + NoLege
nd()
```



The next commands are part of the process to create the bubble map, showing the cell types discovered in each cluster.

```
cL_resutls=cL_resutls[order(cL_resutls$cluster),]; edges = cL_resutls; edges$type = paste0(edges$type,"_",edges$c
luster); edges$cluster = paste0("cluster ", edges$cluster); edges = edges[,c("cluster", "type")]; colnames(edges)
= c("from", "to"); rownames(edges) <- NULL
```

```
nodes_lvl1 = sctype_scores[,c("cluster", "ncells")]; nodes_lvl1$cluster = paste0("cluster ", nodes_lvl1$cluster);
nodes_lvl1$Colour = "#f1f1ef"; nodes_lvl1$ord = 1; nodes_lvl1$realname = nodes_lvl1$cluster; nodes_lvl1 = as.dat
a.frame(nodes_lvl1); nodes_lvl2 = c()
```

```
for (i in 1:length(unique(cL_resutls$cluster))){
  dt_tmp = cL_resutls[cL_resutls$cluster == unique(cL_resutls$cluster)[i], ]; nodes_lvl2 = rbind(nodes_lvl2, dat
a.frame(cluster = paste0(dt_tmp$type,"_",dt_tmp$cluster), ncells = dt_tmp$scores, Colour = ccolss[i], ord = 2, re
alname = dt_tmp$type))
}
```

```
nodes <- rbind(nodes_lvl1, nodes_lvl2); nodes$ncells[nodes$ncells<1] = 1;
files_db = openxlsx::read.xlsx(db_)[,c("cellName", "shortName")]; files_db = unique(files_db); nodes = merge(node
s, files_db, all.x = T, all.y = F, by.x = "realname", by.y = "cellName", sort = F)
```

```
nodes$shortName[is.na(nodes$shortName)] = nodes$realname[is.na(nodes$shortName)]; nodes = nodes[,c("cluster", "nc
ells", "Colour", "ord", "shortName", "realname")]
```

```
mygraph <- graph_from_data_frame(edges, vertices=nodes)
```

generate the bubble plot.

```
ggraph(mygraph, layout = 'circlepack', weight=I(ncells)) +
  geom_node_circle(aes(filter=ord==1,fill=I("#F5F5F5"), colour=I("#D3D3D3")), alpha=0.9) + geom_node_circle(aes(f
ilter=ord==2,fill=I(Colour), colour=I("#D3D3D3")), alpha=0.9) +
  theme_void() + geom_node_text(aes(filter=ord==2, label=shortName, colour=I("#ffffff"), fill="white", repel = !1
, parse = T, size = I(log(ncells,25)*1.5)))+ geom_node_label(aes(filter=ord==1,  label=shortName, colour=I("#0000
00"), size = I(3), fill="white", parse = T), repel = !0, segment.linetype="dotted")
```

```
## Non-leaf weights ignored
```

```
## Warning in geom_node_text(aes(filter = ord == 2, label = shortName, colour =
## I("#ffffff"), : Ignoring unknown aesthetics: fill, repel, and parse
```

```
## Warning in geom_node_label(aes(filter = ord == 1, label = shortName, colour =
## I("#000000"), : Ignoring unknown aesthetics: parse
```