Integration analysis of scRNA-seq

Pang-Kuo Lo

11/5/2019

Integration Analysis of Single-Cell RNA-seq Datasets

The “Seurat” library package developed by Satija Lab is a useful R-based tool for single-cell genomics analysis. Seurat is used here for integration analysis of single-cell RNA-seq datasets from two different BMMC samples (aml027 and aml035) with two different conditions (pre-transplantation and post-transplantation). These four datasets can be downloaded from the resources of the 10X genomics website.

```
## Load required library packages
library(Seurat)
library(cowplot)
library(ggplot2)
library(rafalib)
```

The first step is to load 10X datasets to create their respective Seurat objects using the Read10X() function syntax. Each single-cell RNA-seq (scRNA-seq) dataset contain three files, including barcodes.tsv, genes.tsv and matrix.mtx. These three files need to be placed in the same folder for being read by Read10X(). These files need to be decompressed before they can be read if they are compressed as tar or gz.

```
AML027_pre.data <- Read10X(data.dir="/Users/Pang-Kuo/Desktop/NGS_ML_Analysis/single_cell_RNA-seq_analysis/AML027_Pre-trans
plant_BMMCs/hg19", gene.column = 2)
AML027_post.data <- Read10X(data.dir="/Users/Pang-Kuo/Desktop/NGS_ML_Analysis/single_cell_RNA-seq_analysis/AML027_Post-tra
nsplant_BMMCs/hg19", gene.column = 2)
AML035_pre.data <- Read10X(data.dir="/Users/Pang-Kuo/Desktop/NGS_ML_Analysis/single_cell_RNA-seq_analysis/AML035_Pre-trans
plant_BMMCs/hg19", gene.column = 2)
AML035_post.data <- Read10X(data.dir="/Users/Pang-Kuo/Desktop/NGS_ML_Analysis/single_cell_RNA-seq_analysis/AML035_Post-tra
nsplant_BMMCs/hg19", gene.column = 2)
```

The second step is to create their respective Seurat objects from their matrix data.

```
# Create a function for converting matrix data into a Seurat object
Seurat_object <- function(matrix, object_name) {
  object <- CreateSeuratObject(counts = matrix, project = object_name, min.cells = 5)
  object$INT <- object_name
  object <- subset(object, subset = nFeature_RNA > 500)
  object <- NormalizeData(object, verbose = FALSE)
  object <- FindVariableFeatures(object, selection.method = "vst", nfeatures = 2000)
  return(object)
}

# Use the Seurat_object function to create Seurat objects
AML027_pre <- Seurat_object(AML027_pre.data, "AML027_pre")
AML027_post <- Seurat_object(AML027_post.data, "AML027_post")
AML035_pre <- Seurat_object(AML035_pre.data, "AML035_pre")
AML035_post <- Seurat_object(AML035_post.data, "AML035_post")
```

The third step is to perform integration for creating an integrated Seurat object.

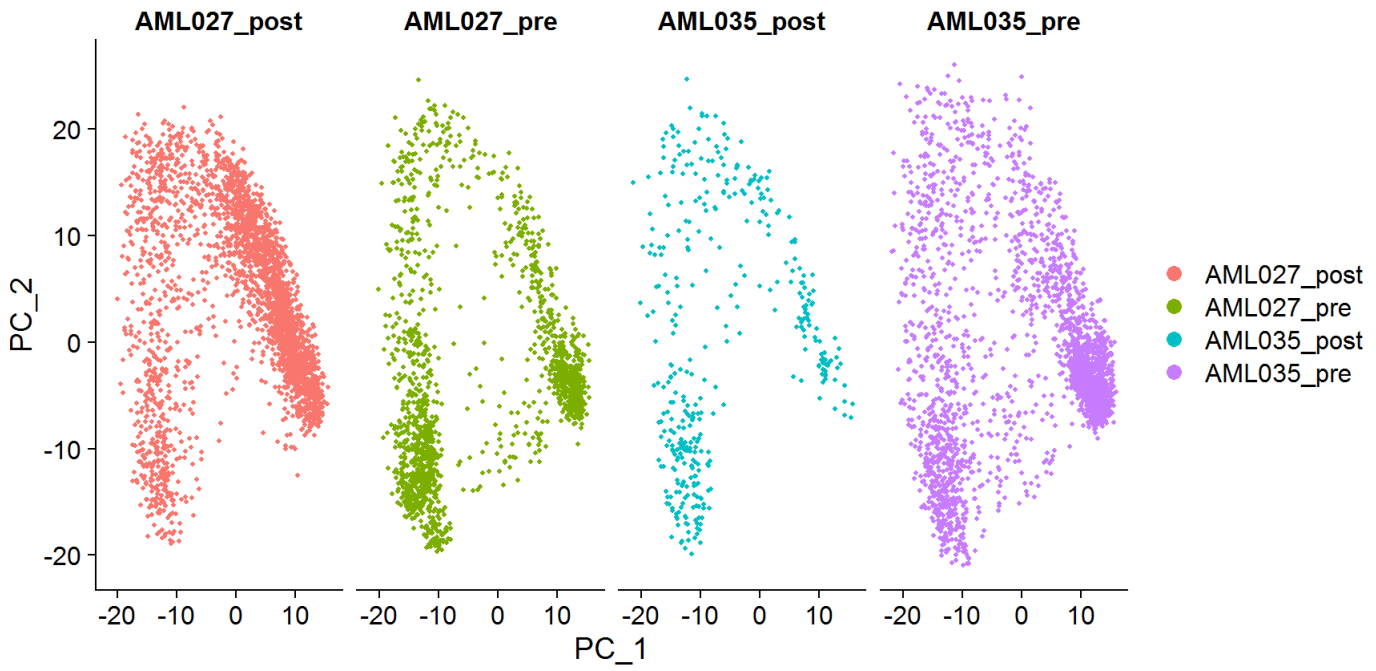
```
#Perform integration
AML.anchors <- FindIntegrationAnchors(object.list = list(AML027_pre, AML027_post, AML035_pre, AML035_post), dims = 1:15)
AML.combined <- IntegrateData(anchorset = AML.anchors, dims = 1:15)
```

The fourth step is to scale and center the data and perform a PCA analysis.

```
DefaultAssay(AML.combined) <- "integrated"
AML.combined <- ScaleData(AML.combined, verbose = FALSE)
AML.combined <- RunPCA(AML.combined, npcs = 30, verbose = FALSE)
```

PCA plots can be created using the DimPlot function syntax.

```
DimPlot(AML.combined, reduction = "pca", split.by = "INT", pt.size = 0.8)
```




```
# Or
# PCAPlot(AML.combined, split.by = "INT", pt.size = 0.8)
```

The fifth step is to perform clustering.

```
AML.combined <- FindNeighbors(AML.combined, reduction = "pca", dims = 1:15)
AML.combined <- FindClusters(AML.combined, resolution = 0.5)
```

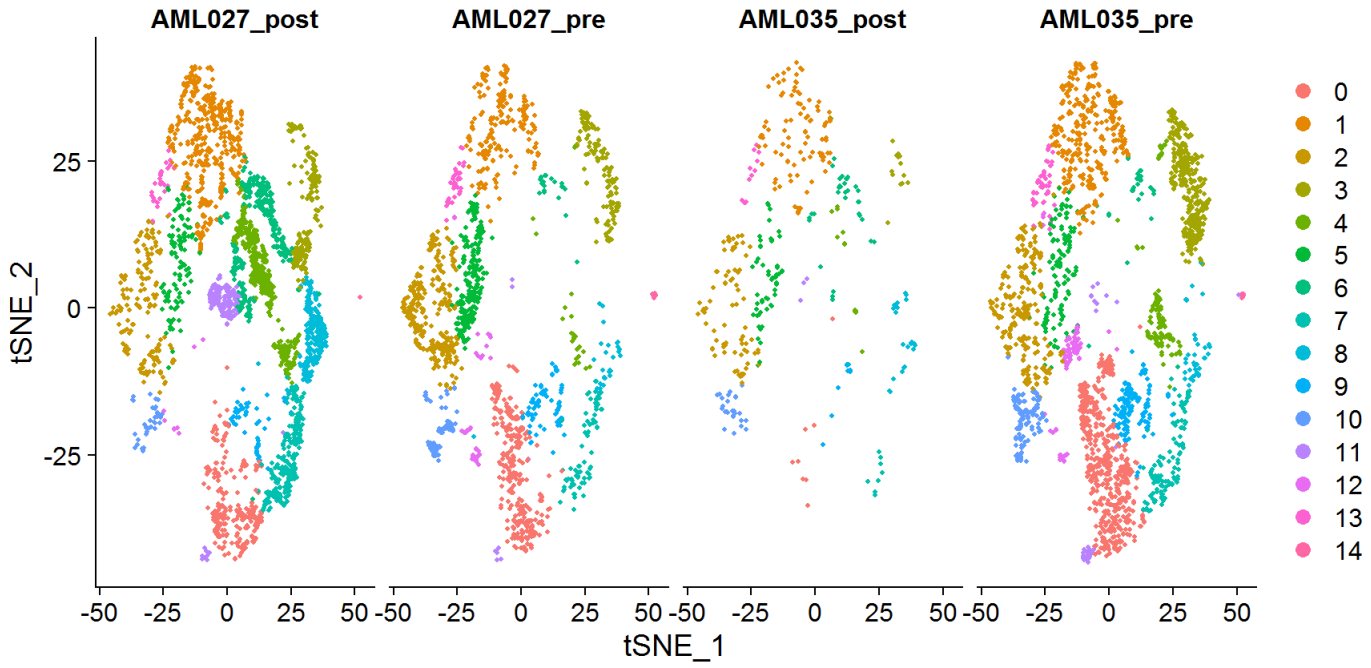
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 7425
## Number of edges: 269528
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9078
## Number of communities: 15
## Elapsed time: 1 seconds
```

The sixth step is to perform t-SNE analysis.

```
AML.combined <- RunTSNE(AML.combined, reduction = "pca", dims = 1:15)
```

The t-SNE plots can be created using DimPlot or TSNEPlot function syntaxes.

```
DimPlot(AML.combined, reduction = "tsne", split.by = "INT", pt.size = 0.8)
```



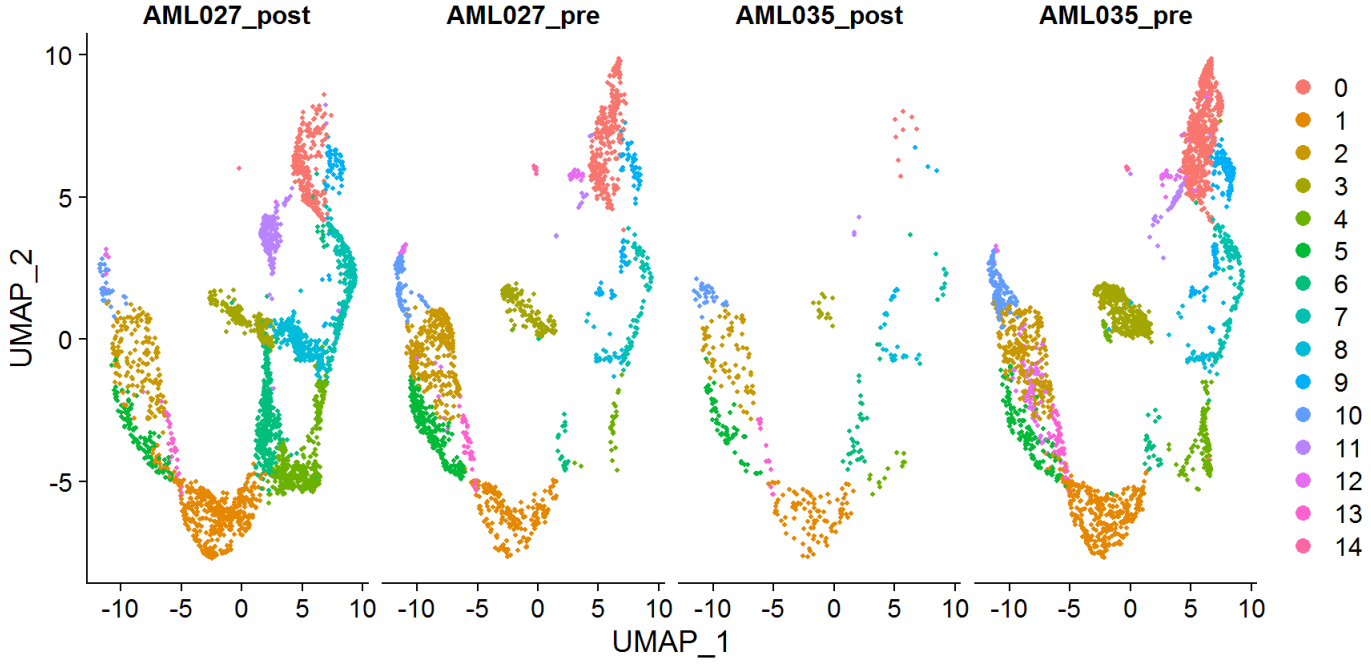

```
# Or
# TSNEPlot(AML.combined, split.by = "INT", pt.size = 0.8)
```

The seventh step is to perform UMAP analysis.

```
AML.combined <- RunUMAP(AML.combined, reduction = "pca", dims = 1:15)
```

The UMAP plots can be created using DimPlot or UMAPPlot function syntaxes.

```
DimPlot(AML.combined, reduction = "umap", split.by = "INT", pt.size = 0.8)
```




```
# Or
# UMAPPlot(AML.combined, split.by = "INT", pt.size = 0.8)
```

The eighth step is to identify conserved cell-type-specific gene markers.

```
DefaultAssay(AML.combined) <- "RNA"
## Create an empty list for storing the output data of differentially expressed gene markers
gene_markers <- vector("list", 15)
clust_num <- c(seq(1:15)-1)
names(gene_markers) <- paste0("c", c(seq(1:15)-1), "_markers")

# Store differentially expressed genes in each cell cluster in the gene_markers list (Only cell clusters 0 - 4 are analyze
d here as examples)
for (i in 1:5) {
  gene_markers[[i]] <- FindConservedMarkers(AML.combined, ident.1 = clust_num[i], grouping.var = "INT", verbose = FALSE)
}
```

Print the top 5 representative marker genes for cell clusters 0 - 4

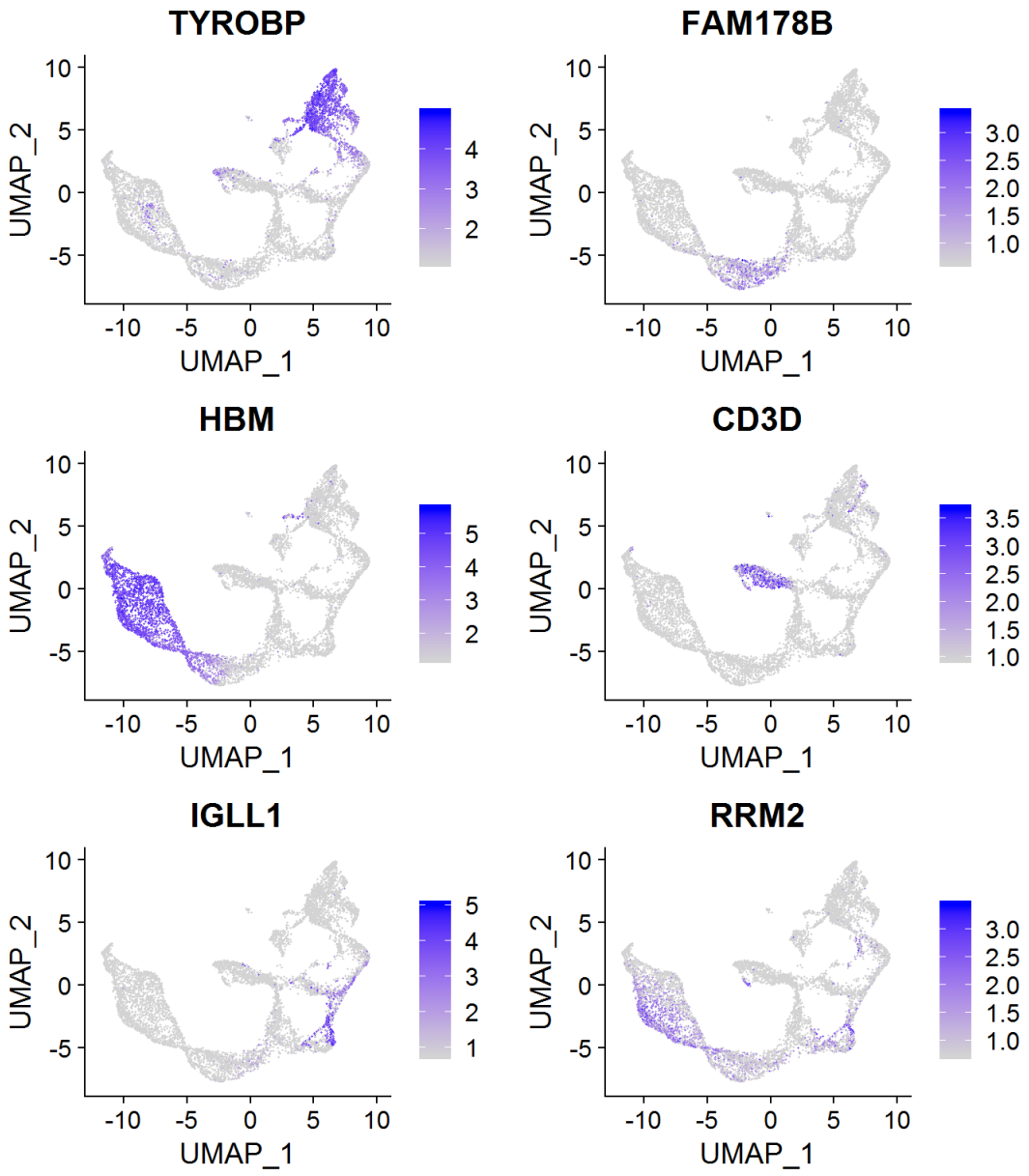
```
for (i in 1:5) {
  print(names(gene_markers[i]));print(head(gene_markers[[i]],5)[,c(2,5,7,10,12,15,17,20)])
}
```

```
## [1] "c0_markers"
##        AML035_post_avg_logFC AML035_post_p_val_adj AML027_pre_avg_logFC
## TYROBP             2.7448239          3.945659e-20             2.433887
## CTSS               1.5719726          2.887940e-05             2.460065
## LST1               1.7182461          3.292903e-04             2.260340
## FTL                2.1507628          2.276731e-01             1.945386
## AIF1               0.9132507          1.000000e+00             2.338426
##        AML027_pre_p_val_adj AML035_pre_avg_logFC AML035_pre_p_val_adj
## TYROBP        3.647359e-194             2.139878         0.000000e+00
## CTSS          5.583204e-198             2.057758        4.911329e-277
## LST1          1.933649e-188             2.033927        6.617928e-273
## FTL           3.364691e-147             1.941649        2.974006e-272
## AIF1          4.543306e-190             1.908322        1.202626e-252
##        AML027_post_avg_logFC AML027_post_p_val_adj
## TYROBP             2.1896494         5.192242e-188
## CTSS               0.8474198          4.942137e-07
## LST1               1.0552910          8.775128e-40
## FTL                1.0035732          3.388461e-89
## AIF1               0.3832754          8.589001e-03
## [1] "c1_markers"
##         AML035_post_avg_logFC AML035_post_p_val_adj AML027_pre_avg_logFC
## FAM178B             1.5534075          3.398802e-52             1.441816
## APOC1               1.0806722          1.333125e-25             1.229159
## KCNH2               1.0367480          2.457230e-25             1.215353
## SYNGR1              1.2357021          4.332460e-37             1.528121
## AKR1C3              0.7673205          1.170952e-29             1.109178
##         AML027_pre_p_val_adj AML035_pre_avg_logFC AML035_pre_p_val_adj
## FAM178B        5.332282e-234             1.207616         0.000000e+00
## APOC1          3.522437e-138             1.298342        1.389776e-264
## KCNH2          3.700710e-116             1.319148        2.525188e-251
## SYNGR1         1.853304e-139             1.622014        1.533163e-244
## AKR1C3         1.098943e-154             1.261837        3.577456e-242
##         AML027_post_avg_logFC AML027_post_p_val_adj
## FAM178B             0.5413052          8.420264e-90
## APOC1               1.1976836         3.370181e-215
## KCNH2               0.5378647          3.144586e-76
## SYNGR1              0.9006224          3.300552e-94
## AKR1C3              0.2876318          3.572936e-20
## [1] "c2_markers"
##       AML035_post_avg_logFC AML035_post_p_val_adj AML027_pre_avg_logFC
## TSPO2             0.7712366          4.746315e-12            1.1635718
## HBM               1.2061167          9.385538e-23            1.3287420
## HEMGN             0.9011395          3.668879e-18            1.3025588
## GYPA              0.9016375          1.737597e-14            0.9069064
## AHSP              0.8467724          1.852941e-17            0.9090798
##       AML027_pre_p_val_adj AML035_pre_avg_logFC AML035_pre_p_val_adj
## TSPO2         9.616885e-96             1.022451        4.732569e-163
## HBM          1.697229e-105             1.805603        7.903608e-161
## HEMGN        1.763949e-117             1.441080        3.935436e-150
## GYPA          1.110094e-76             1.361713        8.300169e-145
## AHSP          9.653263e-73             1.286348        6.421956e-118
##       AML027_post_avg_logFC AML027_post_p_val_adj
## TSPO2              1.131273         1.859195e-135
## HBM                2.403444         4.563608e-120
## HEMGN              1.336840          9.181634e-60
## GYPA               1.852953         3.001329e-122
## AHSP               2.216127         1.136299e-136
## [1] "c3_markers"
##         AML035_post_avg_logFC AML035_post_p_val_adj AML027_pre_avg_logFC
## CD3D                1.6040051          8.551412e-32             2.132558
## LTB                 1.0786259          7.300513e-27             2.408045
## PTPRCAP             1.0602053          8.468700e-04             2.292773
## CD3E                0.9277859          5.300909e-15             1.869362
## CD52                1.0816220          1.523180e-04             1.513704
##         AML027_pre_p_val_adj AML035_pre_avg_logFC AML035_pre_p_val_adj
## CD3D          1.588274e-147             2.078507        2.409102e-259
## LTB            9.791347e-74             2.467561        3.776746e-214
```

```
## PTPRCAP        1.243634e-175             1.404613        1.775080e-102
## CD3E           2.548705e-169             1.285028        7.767770e-113
## CD52            1.530059e-56             1.688550        1.061956e-160
##         AML027_post_avg_logFC AML027_post_p_val_adj
## CD3D                0.9927434          3.463103e-76
## LTB                 1.0209048          1.633977e-46
## PTPRCAP             1.0720331          4.139250e-45
## CD3E                0.7979811          1.529861e-81
## CD52                0.6256404          1.077084e-08
## [1] "c4_markers"
##        AML035_post_avg_logFC AML035_post_p_val_adj AML027_pre_avg_logFC
## IGLL1              0.5305418          9.970261e-02             2.970287
## STMN1              1.0444289          7.393521e-03             2.154919
## TYMS               0.2680974          1.000000e+00             1.085404
## PDLIM1             0.7644226          4.338690e-06             0.702984
## GGH                0.4368772          1.000000e+00             0.587439
##        AML027_pre_p_val_adj AML035_pre_avg_logFC AML035_pre_p_val_adj
## IGLL1          1.708561e-34            3.0971833        6.373197e-208
## STMN1          5.861482e-14            2.6739623        1.545194e-108
## TYMS           5.094455e-02            1.4699160         3.382697e-53
## PDLIM1         2.347806e-05            1.3569098         2.287523e-99
## GGH            1.000000e+00            0.6392683         2.561256e-07
##        AML027_post_avg_logFC AML027_post_p_val_adj
## IGLL1              0.3351756          5.694963e-04
## STMN1              0.6690276          5.231816e-49
## TYMS               0.9531129         3.254359e-106
## PDLIM1             0.7832588          2.328389e-72
## GGH                0.6745250          2.445212e-82
```

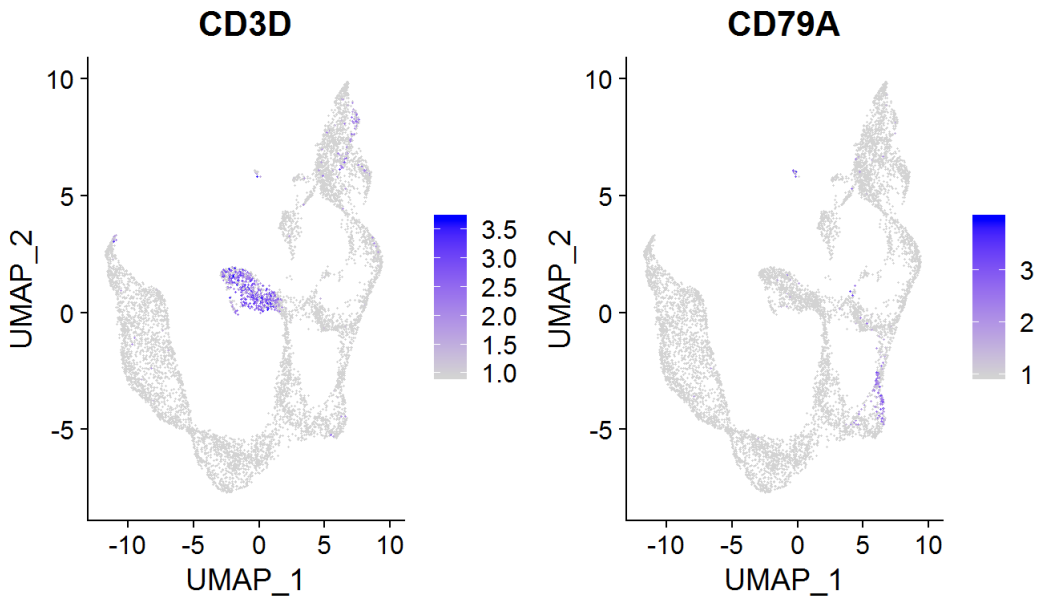
The FeaturePlot function syntax can be used to visualize expression patterns of cell-type-specific gene markers in UMAP plots (c0 cluster:“TYROBP”, c1 cluster:“FAM178B”, c2 cluster:“HBM”, c3 cluster:“CD3D”, c4 cluster:“IGLL1”, c5 cluster:“RRM2”)

```
FeaturePlot(AML.combined, features = c("TYROBP", "FAM178B","HBM", "CD3D", "IGLL1", "RRM2"), min.cutoff = "q9")
```



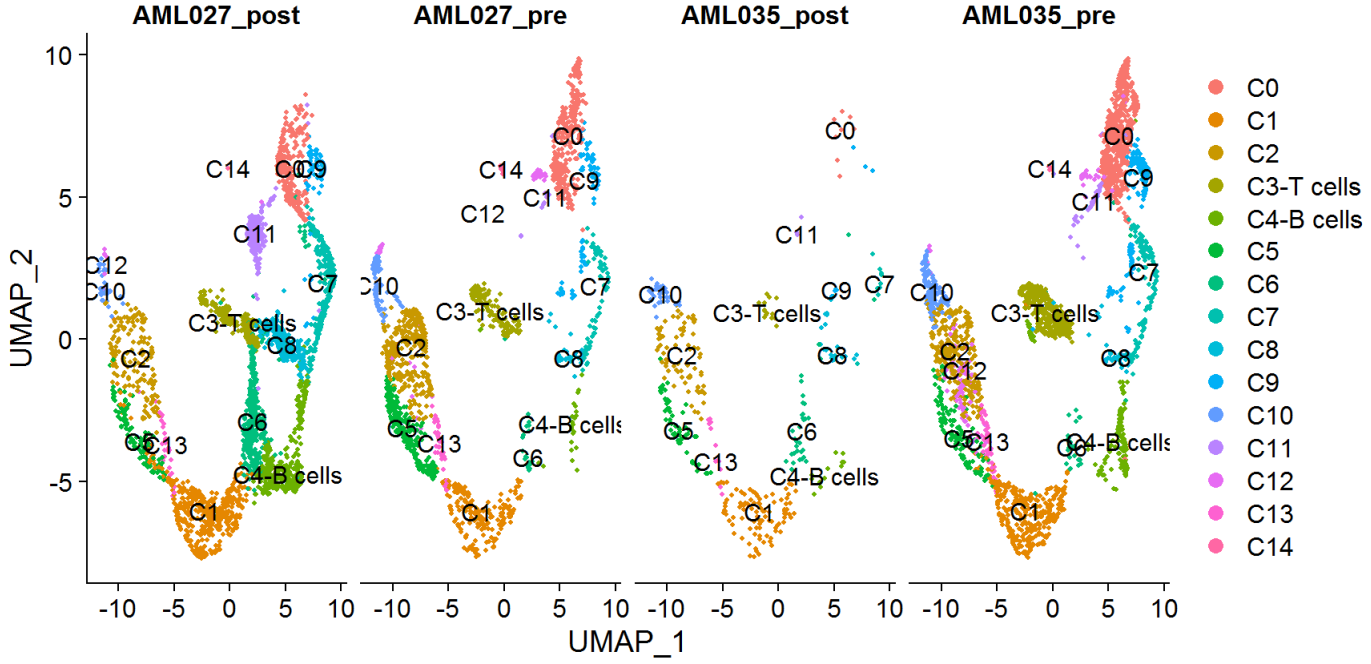
Check some well-known cell-type-specific marker (T cell marker: CD3D, B cell marker: CD79A)

```
FeaturePlot(AML.combined, features = c("CD3D", "CD79A"), min.cutoff = "q9")
```



To be easy visualization and comparison, identified cell clusters can be labeled in UMAP plots.

```
AML.combined <- RenameIdents(AML.combined, `0` = "C0", `1` = "C1", `2` = "C2", `3` = "C3-T cells", `4` = "C4-B cells",
                             `5` = "C5", `6` = "C6", `7` = "C7", `8` = "C8", `9` = "C9", `10` = "C10", `11` = "C11",
                            `12` = "C12", `13` = "C13", `14` = "C14")
DimPlot(AML.combined, split.by = "INT", label = TRUE, pt.size = 0.8)
```



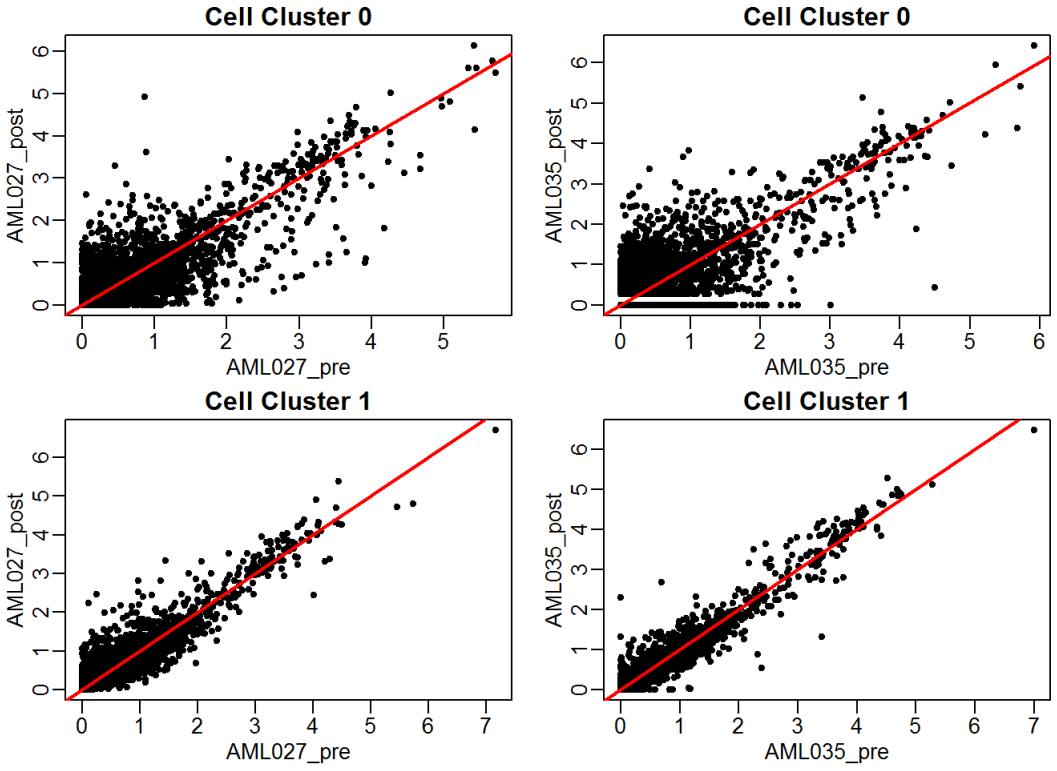
The AverageExpression syntax can be used to compute the average expression of genes in a specific cell cluster accross different conditions. Clusters 0 and 1 are analyzed here as examples.

```
C0_cluster <- subset(AML.combined, idents = "C0")
Idents(C0_cluster) <- "INT"
avg.C0_cluster <- log1p(AverageExpression(C0_cluster, verbose = FALSE)$RNA)
avg.C0_cluster$gene <- rownames(avg.C0_cluster)

C1_cluster <- subset(AML.combined, idents = "C1")
Idents(C1_cluster) <- "INT"
avg.C1_cluster <- log1p(AverageExpression(C1_cluster, verbose = FALSE)$RNA)
avg.C1_cluster$gene <- rownames(avg.C1_cluster)
```

Differentially expressed genes can be displayed in scatter plots using output data from above.

```
mypar(2,2)
with(avg.C0_cluster, plot(AML027_pre, AML027_post, pch=19, cex=0.7, main="Cell Cluster 0")) ; abline(0,1, lwd=2, col="red"
)
with(avg.C0_cluster, plot(AML035_pre, AML035_post, pch=19, cex=0.7, main="Cell Cluster 0")) ; abline(0,1, lwd=2, col="red"
)
with(avg.C1_cluster, plot(AML027_pre, AML027_post, pch=19, cex=0.7, main="Cell Cluster 1")) ; abline(0,1, lwd=2, col="red"
)
with(avg.C1_cluster, plot(AML035_pre, AML035_post, pch=19, cex=0.7, main="Cell Cluster 1")) ; abline(0,1, lwd=2, col="red"
)
```



The final step is to analyze differential expressed genes in a specific cell cluster/type across conditions using the FindMarkers syntax. Cell cluster C1 is analyzed here as an example. HSPA5 is identified as an upregulated gene in both AML027 and AML035 cell samples under the post-transplantation condition when compared to the pre-transplantation condition.

```
AML.combined$celltype.INT <- paste(Idents(AML.combined), AML.combined$INT, sep = "_")
AML.combined$celltype <- Idents(AML.combined)
AML.combined_2 <- AML.combined
Idents(AML.combined_2) <- "celltype.INT"

C1_AML027_DEG <- FindMarkers(AML.combined_2, ident.1 = "C1_AML027_post", ident.2 = "C1_AML027_pre", verbose = FALSE)
C1_AML027_DEG.sorted <- C1_AML027_DEG[order(C1_AML027_DEG$avg_logFC, decreasing = TRUE),]
head(C1_AML027_DEG.sorted, 8)
```

```
##                p_val avg_logFC pct.1 pct.2    p_val_adj
## NFKBIA   1.747228e-47  2.202256 0.747 0.170 2.455729e-43
## ZFP36    1.053281e-51  2.117927 0.752 0.108 1.480386e-47
## TMSB4X   4.824679e-69  1.885517 0.978 0.722 6.781086e-65
## HSPA5    6.736204e-50  1.832553 0.900 0.649 9.467734e-46
## CCNL1    7.219248e-51  1.590184 0.829 0.325 1.014665e-46
## PPP1R15A 2.171074e-51  1.553731 0.909 0.675 3.051445e-47
## IER2     2.256732e-36  1.512779 0.803 0.418 3.171837e-32
## HLA-B    2.057930e-45  1.512186 0.827 0.376 2.892420e-41
```

```
tail(C1_AML027_DEG.sorted, 8)
```

```
##              p_val  avg_logFC pct.1 pct.2    p_val_adj
## HBG1  7.128858e-46 -0.9179637 0.175 0.830 1.001961e-41
## RGCC  1.152116e-46 -0.9205402 0.193 0.763 1.619299e-42
## MYL4  2.834529e-46 -0.9387783 0.208 0.794 3.983930e-42
## HBA2  2.296885e-13 -0.9407781 0.958 1.000 3.228272e-09
## IFI27 1.322299e-43 -0.9606629 0.193 0.763 1.858491e-39
## STMN1 6.762280e-46 -1.0805062 0.550 0.959 9.504384e-42
## MGST3 4.628940e-45 -1.2811060 0.282 0.825 6.505976e-41
## AHSP  6.898711e-66 -1.5796961 0.650 1.000 9.696138e-62
```

```
C1_AML035_DEG <- FindMarkers(AML.combined_2, ident.1 = "C1_AML035_post", ident.2 = "C1_AML035_pre", verbose = FALSE)
C1_AML035_DEG.sorted <- C1_AML035_DEG[order(C1_AML035_DEG$avg_logFC, decreasing = TRUE),]
head(C1_AML035_DEG.sorted, 8)
```

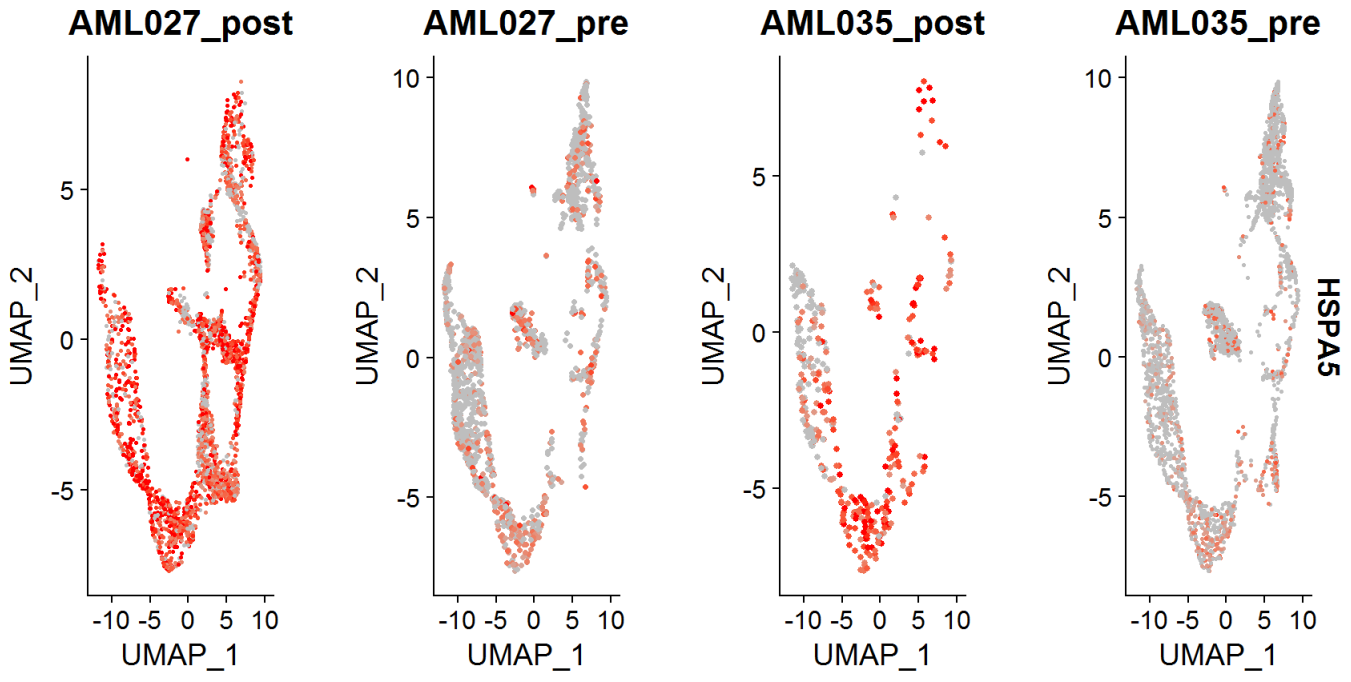
```
##               p_val avg_logFC pct.1 pct.2    p_val_adj
## RPS4Y1  8.842151e-85 2.2894587 0.919 0.000 1.242764e-80
## HSPA5   6.468616e-44 1.9919423 0.960 0.464 9.091640e-40
## EIF1AY  1.660416e-65 1.3222253 0.737 0.000 2.333715e-61
## RPS29   5.163357e-46 1.2464066 1.000 0.982 7.257098e-42
## RPL36A  8.209129e-49 1.1709177 1.000 0.988 1.153793e-44
## RPL17   7.787407e-38 1.0472376 0.980 0.796 1.094520e-33
## JUN     5.377391e-19 1.0277637 0.556 0.147 7.557923e-15
## TSC22D3 1.811644e-22 0.9876815 0.505 0.090 2.546265e-18
```

```
tail(C1_AML035_DEG.sorted, 8)
```

```
##               p_val  avg_logFC pct.1 pct.2    p_val_adj
## LYZ    7.546175e-06 -0.8787207 0.000 0.177 1.060615e-01
## MT-CO2 1.403205e-26 -0.9258513 1.000 0.970 1.972205e-22
## HBD    2.197989e-13 -0.9691496 0.859 0.958 3.089274e-09
## S100A8 2.080685e-04 -1.1156741 0.030 0.177 1.000000e+00
## S100A9 2.010699e-04 -1.1606336 0.020 0.162 1.000000e+00
## TMSB4X 1.588449e-03 -1.4524421 0.434 0.563 1.000000e+00
## HBG2   1.593157e-05 -1.8541596 0.071 0.269 2.239183e-01
## HBG1   1.185122e-05 -2.0883761 0.091 0.305 1.665689e-01
```

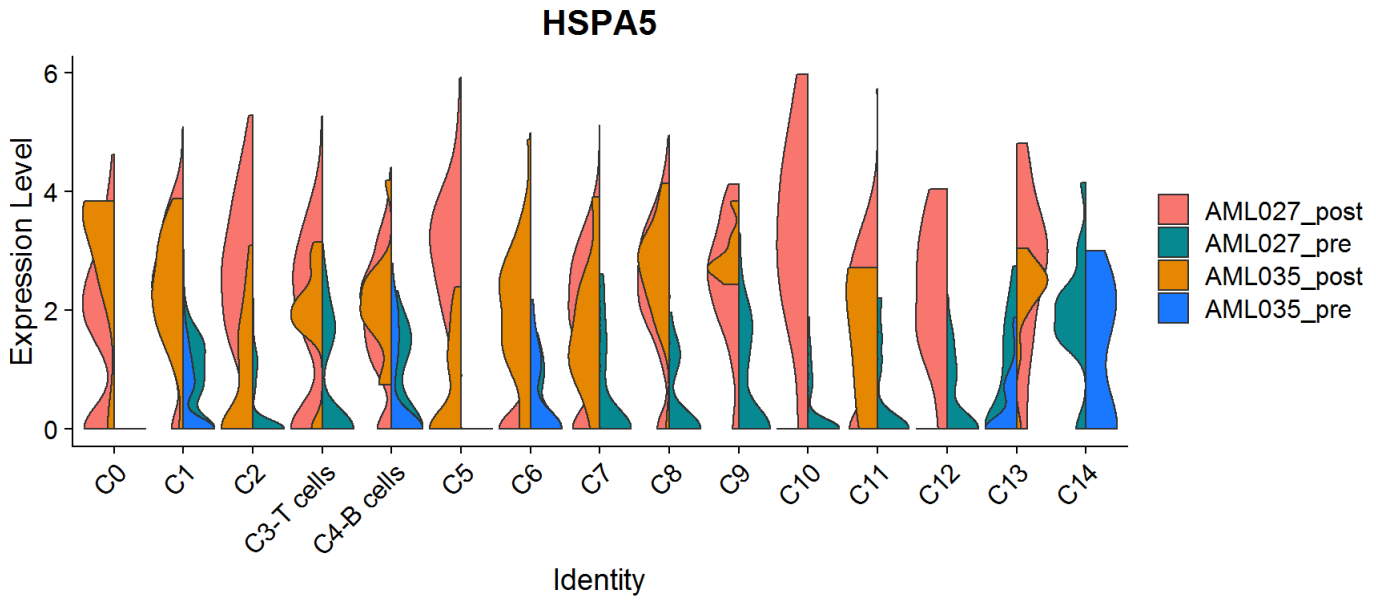
Visualization of differentially expressed genes in UMAP plots.

```
FeaturePlot(AML.combined, features = "HSPA5", split.by = "INT", max.cutoff = 3, cols = c("grey", "red"))
```



Visualization of differentially expressed genes in VInplots.

```
plots <- VlnPlot(AML.combined, features = "HSPA5", split.by = "INT", group.by = "celltype", pt.size = 0, combine = FALSE)
CombinePlots(plots = plots, ncol = 1)
```



Here is the output of sessionInfo() on the system on which this document was compiled:

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## 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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] rafalib_1.0.0 ggplot2_3.2.1 cowplot_1.0.0 Seurat_3.1.1
##
## loaded via a namespace (and not attached):
##  [1] tsne_0.1-3          nlme_3.1-140        bitops_1.0-6
##  [4] RcppAnnoy_0.0.13    RColorBrewer_1.1-2  httr_1.4.1
##  [7] sctransform_0.2.0   tools_3.6.1         backports_1.1.5
## [10] R6_2.4.0            irlba_2.3.3         KernSmooth_2.23-15
## [13] uwot_0.1.4          lazyeval_0.2.2      colorspace_1.4-1
## [16] withr_2.1.2         npsurv_0.4-0        gridExtra_2.3
## [19] tidyselect_0.2.5    compiler_3.6.1      plotly_4.9.1
## [22] labeling_0.3        caTools_1.17.1.2    scales_1.0.0
## [25] lmtest_0.9-37       ggridges_0.5.1      pbapply_1.4-2
## [28] stringr_1.4.0       digest_0.6.22       rmarkdown_1.16
## [31] R.utils_2.9.0       pkgconfig_2.0.3     htmltools_0.4.0
## [34] bibtex_0.4.2        htmlwidgets_1.5.1   rlang_0.4.1
## [37] zoo_1.8-6           jsonlite_1.6        ica_1.0-2
## [40] gtools_3.8.1        dplyr_0.8.3         R.oo_1.23.0
## [43] magrittr_1.5        Matrix_1.2-17       Rcpp_1.0.3
## [46] munsell_0.5.0       ape_5.3             reticulate_1.13
## [49] lifecycle_0.1.0     R.methodsS3_1.7.1   stringi_1.4.3
## [52] yaml_2.2.0          gbRd_0.4-11         MASS_7.3-51.4
## [55] gplots_3.0.1.1      Rtsne_0.15          plyr_1.8.4
## [58] grid_3.6.1          parallel_3.6.1      gdata_2.18.0
## [61] listenv_0.7.0       ggrepel_0.8.1       crayon_1.3.4
## [64] lattice_0.20-38     splines_3.6.1       SDMTools_1.1-221.1
## [67] zeallot_0.1.0       knitr_1.25          pillar_1.4.2
## [70] igraph_1.2.4.1      future.apply_1.3.0  reshape2_1.4.3
## [73] codetools_0.2-16    leiden_0.3.1        glue_1.3.1
## [76] evaluate_0.14       lsei_1.2-0          metap_1.1
## [79] RcppParallel_4.4.4  data.table_1.12.6   vctrs_0.2.0
## [82] png_0.1-7           Rdpack_0.11-0       gtable_0.3.0
## [85] RANN_2.6.1          purrr_0.3.3         tidyr_1.0.0
## [88] future_1.15.0       assertthat_0.2.1    xfun_0.10
## [91] rsvd_1.0.2          RSpectra_0.15-0     survival_2.44-1.1
## [94] viridisLite_0.3.0   tibble_2.1.3        cluster_2.1.0
## [97] globals_0.12.4      fitdistrplus_1.0-14 ROCR_1.0-7
```