Integration with 10X v1.0

Honeycomb Bioinformatics Team

2023-08-25

1 Introduction

This code document outlines how to integrate HIVE data with other single-cell data. In this example, the same cryopreserved neonatal blood sample was loaded onto the HIVE and 10X Genomics platforms for scRNAseq. This document outlines how to integrate these two data sets for combined analysis.

Please contact support@honeycomb.bio (mailto:support@honeycomb.bio) for all questions.

1.1 Package installation and loading

The below packages need to be installed prior to running this vignette. Code for package installation is below. There may be additional dependencies required by these packages you may have to install depending on your R environment.

If needed, install the packages using the code below.

```
install.packages("tidyverse")
install.packages("Seurat")
install.packages("HGNChelper")
install.packages("openxlsx")
install.packages("DT")
```

Load the libraries with the code below.

```
library(tidyverse)
library(Seurat)
library(HGNChelper)
library(openxlsx)
library(DT)
```

1.2 Copying BeeNet/BeeNetPLUS output files from Google Cloud

The code below loads your R object from the Google Cloud Platform Bucket. You will need to replace code where outlined below:

```
system("mkdir -p data")

#Replace the URL with the link to the R object output by BeeNetPLUS or your own downstream R
        object.
#Navigate to your Google Cloud output directory and locate the .tgz object or R object and c
        opy the path into the code below:
system("gsutil -m cp gs://path-to-data ./data")

#Unzip the file
system("tar -zxf data/*tgz -C ./data")
```

2 Load data

2.1 HIVE scRNAseq data

Load your R object into the environment. For BeeNetPLUSv1.0 users, change the name of the object using the code below.

The R object below called `umi_seurat` contains data from HIVE scRNAseq data from blood.

```
#Replace the file path for the R object and load in the R object
load("./data/Name_of_File.Rdata")
```

```
# Add a platform metadata column
umi_seurat$platform <- "HIVE"

# Changing name of object
HIVE <- umi_seurat
remove(umi_seurat)
```

To generate an R object from count matrices from BeeNet, see our Seurat Analysis Tutorial (https://honeycombbio.zendesk.com/hc/en-us/articles/4412967724699-Seurat-Analysis-Tutorial).

2.2 10X data

Load matching 10X data and create an R object. The count matrix for 10x data is typically in a folder named filtered_featured_bc_matrix. This code below reads in the data and creates one Seurat object containing the 10x data.

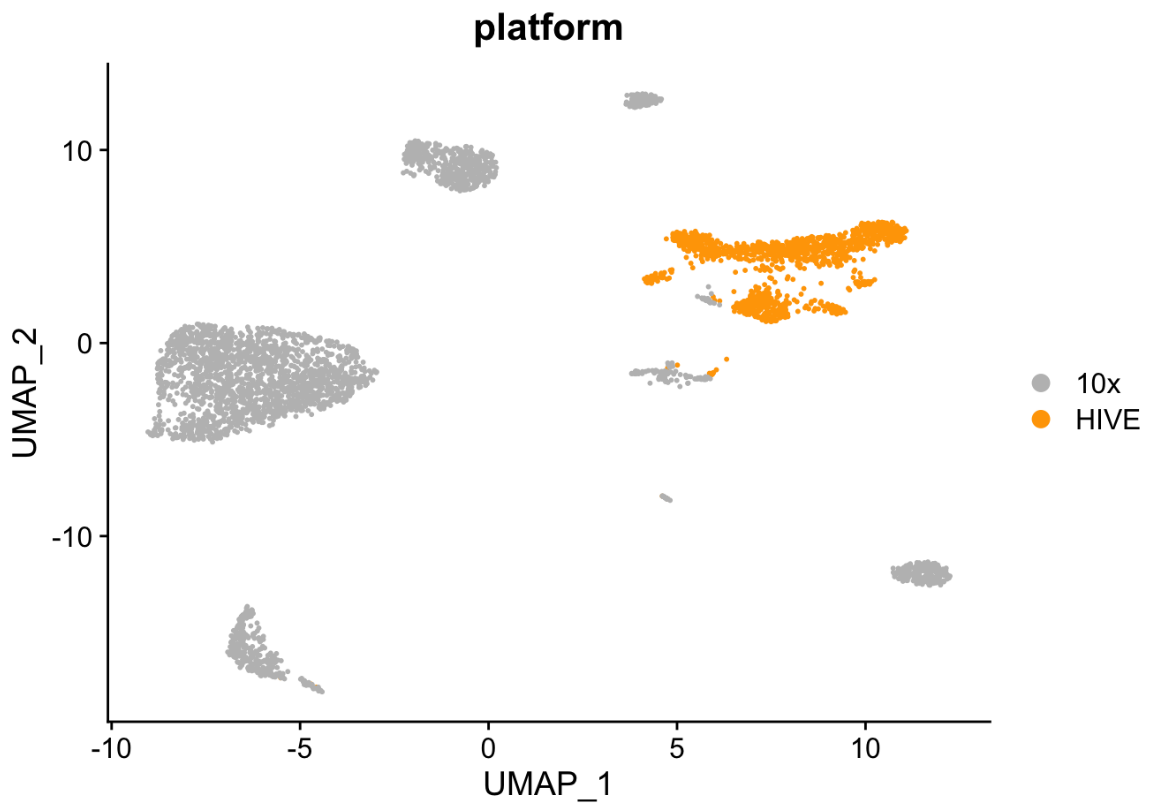
```
# list the filtered_featured_bc_matrix folders for each sample
count_folder_10x <- "path_to_10x_featured_filtered_bc_matrix_folder/"
```

```
# Create Seurat objects for each sample
seurat_10x <- CreateSeuratObject(counts = Read10X(count_folder_10x))

# Add platform information
seurat_10x$platform <- "10x"
```

2.3 Merge data

Without any integration, you can merge the two Seurat objects to create one object. However, the differences in the two platforms platforms cause cells to cluster by platform instead of by cell type. Integration will allow you to analyze the data for biological, rather than technical, variation.

```
# merging 2 datasets without integration
obj <- merge(HIVE, seurat_10x)

# Cluster new object
obj <- NormalizeData(obj)
obj <- FindVariableFeatures(obj, selection.method = "vst", nfeatures = 2000)
obj <- ScaleData(obj)
obj <- RunPCA(obj, verbose = FALSE)
obj <- RunUMAP(obj, dims=1:30,verbose = FALSE)
obj <- FindNeighbors(obj,verbose = FALSE)
obj <- FindClusters(obj, verbose = FALSE)
obj <- BuildClusterTree(obj,reorder=TRUE,reorder.numeric=TRUE)
obj$seurat_clusters <- obj$tree.ident

# Plot results
DimPlot(obj, group.by = "platform", cols = c("grey", "orange"))
```



2.4 Integrating data

The two data sets can be integrated using Seurat functions. First, identify features for integration using `SelectIntegrationFeatures()`. Next, find integration anchors using `FindIntegrationAnchors()`. Finally, integrate the data using `IntegrateData()`. Integration creates a new assay called “integrated”. This assay automatically becomes the default assay in the object and contains variable features that have been integrated. The full data sets for each sample remain in the object in the RNA assay.

The resulting data shows that data from both platforms can integrate together and clustering occurs by cell type instead of platform.

```
# remove old object
remove(obj)

# Normalize and identify variable features in each data set
HIVE <- NormalizeData(HIVE)
HIVE <- FindVariableFeatures(HIVE, selection.method = "vst", nfeatures = 2000)
HIVE <- ScaleData(HIVE)

seurat_10x <- NormalizeData(seurat_10x)
seurat_10x <- FindVariableFeatures(seurat_10x, selection.method = "vst", nfeatures = 2000)
seurat_10x <- ScaleData(seurat_10x)

# Add percent.mito reads to 10x data - this metadata already exists in BeeNetPlus generated
        R objects
seurat_10x[["percent.mito"]] <- PercentageFeatureSet(seurat_10x, pattern = "^MT-") / 100

# Find features for integration
features <- SelectIntegrationFeatures(object.list = c(HIVE, seurat_10x))

# Find anchors for integration using the RNA assay for each object
anchors <- FindIntegrationAnchors(object.list = c(HIVE, seurat_10x), anchor.features = featu
        res,
                                  assay = c("RNA", "RNA"))

# Integrate data
obj <- IntegrateData(anchorset = anchors)
```

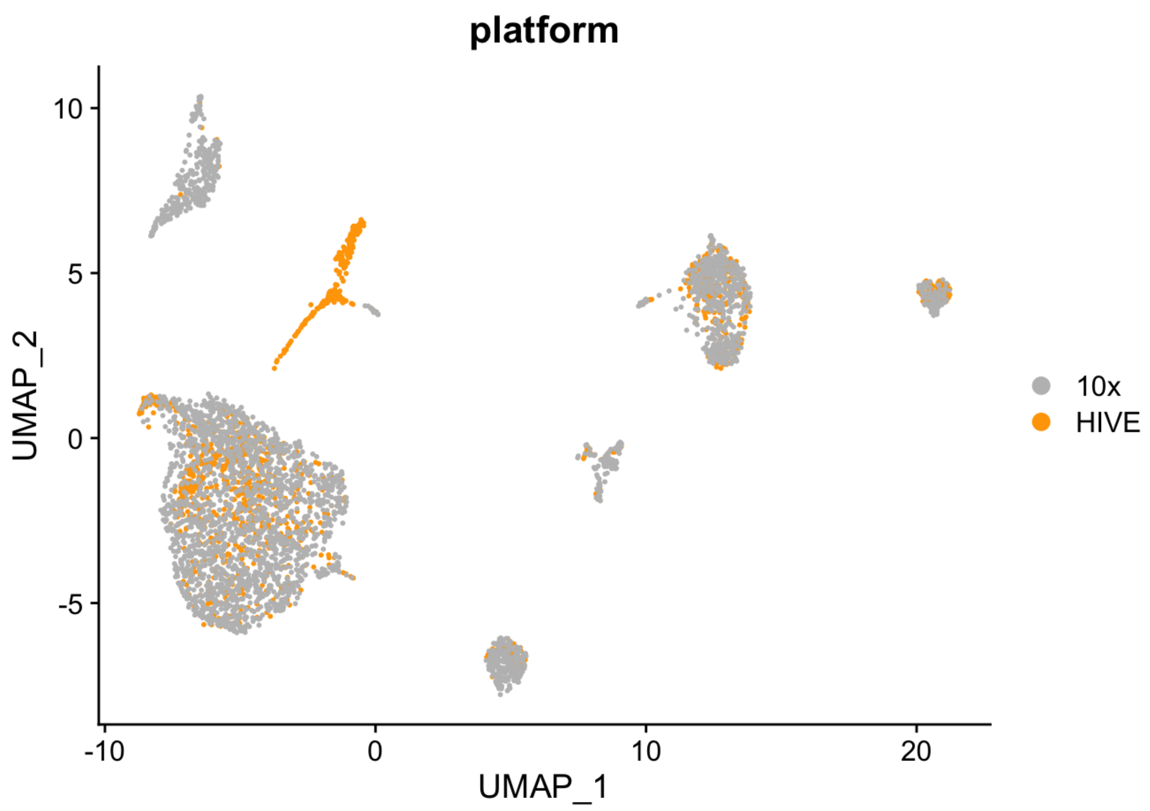
You can easily change the default assay with the code below.

```
# Cluster new data
DefaultAssay(obj) <- "integrated"

obj
```

```
## An object of class Seurat
## 81807 features across 4361 samples within 3 assays
## Active assay: integrated (2000 features, 2000 variable features)
##  2 other assays present: RNA, SCT
```

Following integration, the new assay “integration” needs to be reclustered.

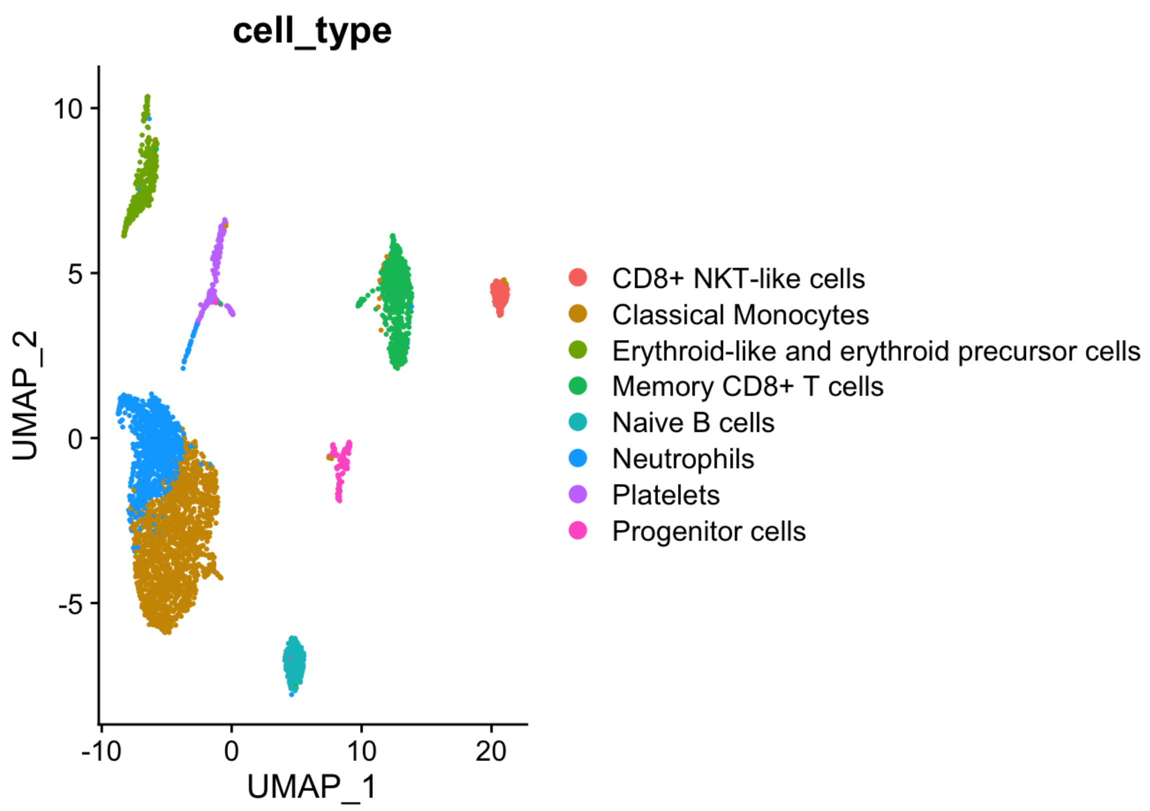
```
# Run the standard workflow for visualization and clustering
obj <- ScaleData(obj, verbose = FALSE)
obj <- RunPCA(obj, npcs = 30, verbose = FALSE)
obj <- RunUMAP(obj, reduction = "pca", dims = 1:30)
obj <- FindNeighbors(obj, reduction = "pca", dims = 1:30)
obj <- FindClusters(obj, resolution = 0.3, verbose = F)
obj <- BuildClusterTree(obj,reorder=TRUE,reorder.numeric=TRUE,
                        verbose = F)
obj$seurat_clusters <- obj$tree.ident

# Visualize results
DimPlot(obj, group.by = "platform", cols = c("grey", "orange"))
```



3 Annotating cell types

Below, we use scType (https://github.com/IanevskiAleksandr/sc-type) to automatically annotate cell types by cluster.

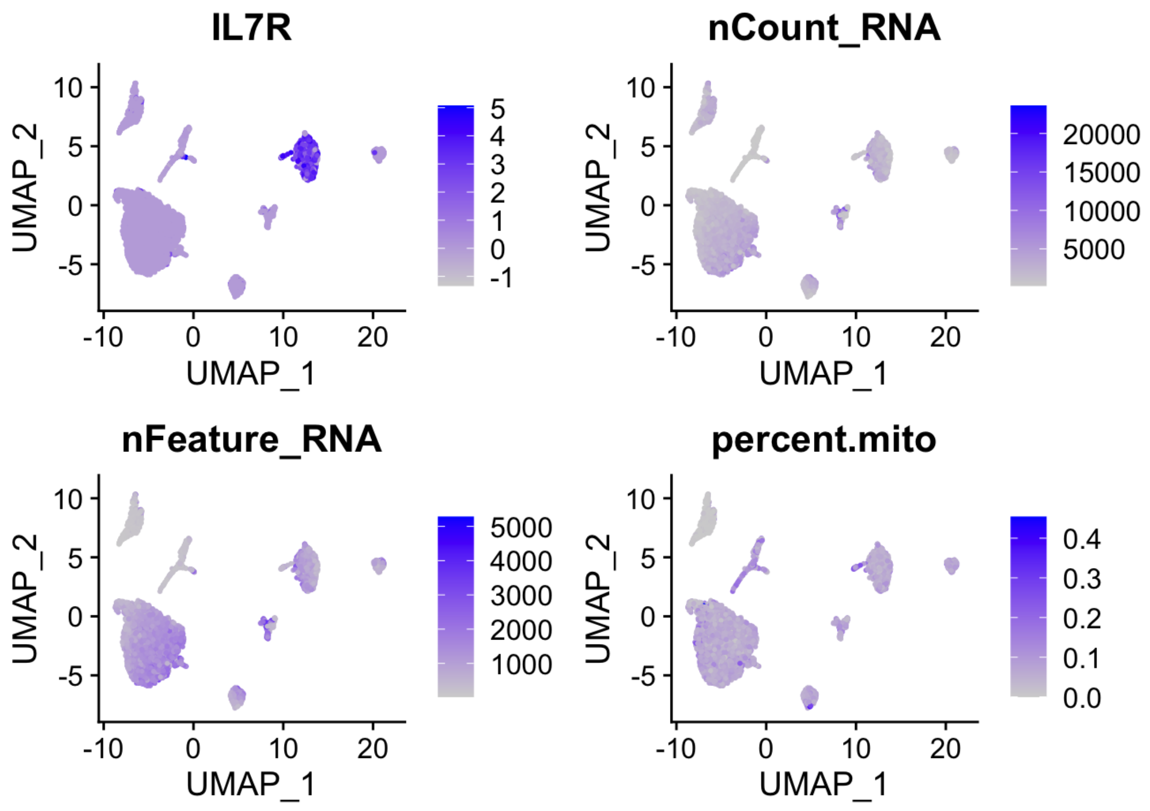
```r
# Using scType to annotate cell types

## Download the reference scripts and mar
source("https://raw.githubusercontent.com/IanevskiAleksandr/sc-type/master/R/gene_sets_prepa
        re.R")
source("https://raw.githubusercontent.com/IanevskiAleksandr/sc-type/master/R/sctype_score_.
        R")
source("https://raw.githubusercontent.com/IanevskiAleksandr/sc-type/master/R/auto_detect_tis
        sue_type.R")

# Download reference data for different tissue types
db_ = "https://raw.githubusercontent.com/IanevskiAleksandr/sc-type/master/ScTypeDB_full.xls
        x"

# Define the tissue. These are blood samples so Immune system is the right choice
tissue<- "Immune system"

# prepare gene sets
gs_list = gene_sets_prepare(db_, tissue)

es.max = sctype_score(scRNAseqData = obj[["integrated"]]@scale.data,
                      scaled = TRUE,
                      gs = gs_list$gs_positive,
                      gs2 = gs_list$gs_negative)

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

# set low-confident (low ScType score) clusters to "unknown"
sctype_scores$type[as.numeric(as.character(sctype_scores$scores)) < sctype_scores$ncells/4]
        = "Unknown"
#print(sctype_scores[,1:3])

obj_cluster_id <- sctype_scores

# save old idents for reference
obj[["old.ident"]] <- Idents(object = obj)



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

obj[["cell_type"]]<- obj$customclassif

# Visualize cell types
DimPlot(obj, group.by = "cell_type")
```



4 Feature Plots

Use the code below to look at feature plots. You can plot QC metrics from the metadata or marker genes of interest. BeeNetPLUSv1.0 users will have QC meta data in their Seurat objects. Some of this metadata is not present in the 10X count matrices. These variables will only plot in HIVE data.

```
# Look at marker genes and QC metrics
FeaturePlot(obj, c("IL7R", "nCount_RNA", "nFeature_RNA", "percent.mito"))
```



5 Find marker genes

You can find marker genes between clusters or cell types using the integrated data.

```
# Set cell idents to clusters
Idents(obj) <- obj$seurat_clusters

# Find marker genes between clusters 1 and 2
markers_clust <- FindMarkers(obj, ident.1 = 1, ident.2 = 2, min.pct = 0.25)

# Display top marker genes
datatable(markers_clust[1:5,])
```

Show 10 entries Search:

	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
AHSP	1.41131323117099e-296	4.1689339027403	0.779	0.004	2.82262646234199e-293
ALAS2	3.76696825848622e-261	4.20829137172602	0.893	0.033	7.53393651697244e-258
SELENBP1	2.97688207786748e-215	2.29410580701632	0.585	0.003	5.95376415573497e-212
HBB	1.99928790266377e-212	8.93808856866528	1	0.175	3.99857580532753e-209
HBM	5.54693188222017e-193	4.91280647725505	0.751	0.075	1.10938637644403e-189

Showing 1 to 5 of 5 entries Previous 1 Next

```
# Set cell idents to cell type
Idents(obj) <- obj$cell_type

# Find marker genes
markers_ct <- FindAllMarkers(obj, min.pct = 0.25)

# Display top marker gene for each cell type
markers_ct %>% group_by(cluster) %>%
        slice(1) %>% datatable()
```

Show 10 entries Search:

	p_val	avg_log2FC	pct.1	pct.2	p_val_adj	cluster	gene
1	4.68124116617039e-194	1.73946915549553	0.988	0.693	9.36248233234077e-191	Neutrophils	CTSS
2	0	3.92214115441325	0.95	0.043	0	Platelets	PPBP
3	0	1.57352124210773	0.97	0.528	0	Classical Monocytes	LGALS1
4	0	3.92992493057268	0.923	0.08	0	Memory CD8+ T cells	IL7R
5	0	4.33604356162276	0.843	0.011	0	CD8+ NKT-like cells	KLRD1

	p_val	avg_log2FC	pct.1	pct.2	p_val_adj	cluster	gene
6	0	3.71650991999123	0.905	0.039	0	Naive B cells	MS4A1
7	1.37223660092883e-282	0.690591734449282	0.414	0.004	2.74447320185767e-279	Progenitor cells	GATA2
8	0	4.1434389848535	0.893	0.076	0	Erythroid-like and erythroid precursor cells	ALAS2

Showing 1 to 8 of 8 entries

Previous 1 Next

You can also find marker genes from the RNA assay in the object.

```
# Find marker genes using the data in the RNA assay
RNA_markers <- FindAllMarkers(obj, assay = "RNA", min.pct = 0.25)

# Display top marker
RNA_markers %>% group_by(cluster) %>%
        slice(1) %>% datatable()
```

Show 10 entries

Search:

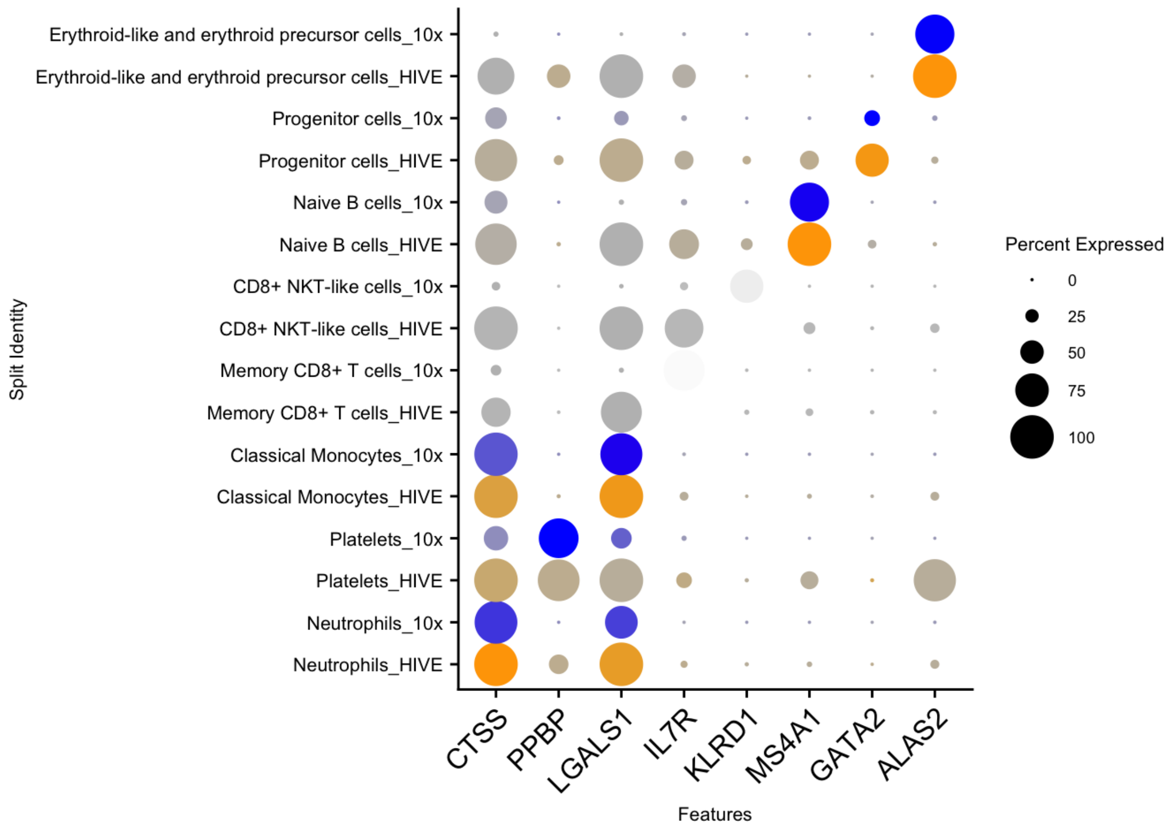
	p_val	avg_log2FC	pct.1	pct.2	p_val_adj	cluster	gene
1	3.07529698921627e-127	2.55633765003627	0.587	0.264	1.77902855529172e-122	Neutrophils	NAMPT
2	1.22898827482941e-63	-3.15399352985694	0.115	0.81	7.10957427106065e-59	Platelets	FTL
3	0	1.91131243720273	0.837	0.205	0	Classical Monocytes	LGALS1
4	0	4.02169409673142	0.905	0.044	0	Memory CD8+ T cells	IL7R
5	0	4.979492782193	0.633	0.008	0	CD8+ NKT-like cells	GNLY
6	0	5.41037746287107	0.519	0.014	0	Naive B cells	IGKC
7	0	1.68489361328921	0.445	0.003	0	Progenitor cells	PRSS57
8	0	9.38983491131701	0.997	0.128	0	Erythroid-like and erythroid precursor cells	HBG2

Showing 1 to 8 of 8 entries

Previous 1 Next

6 Visualize marker genes by platform

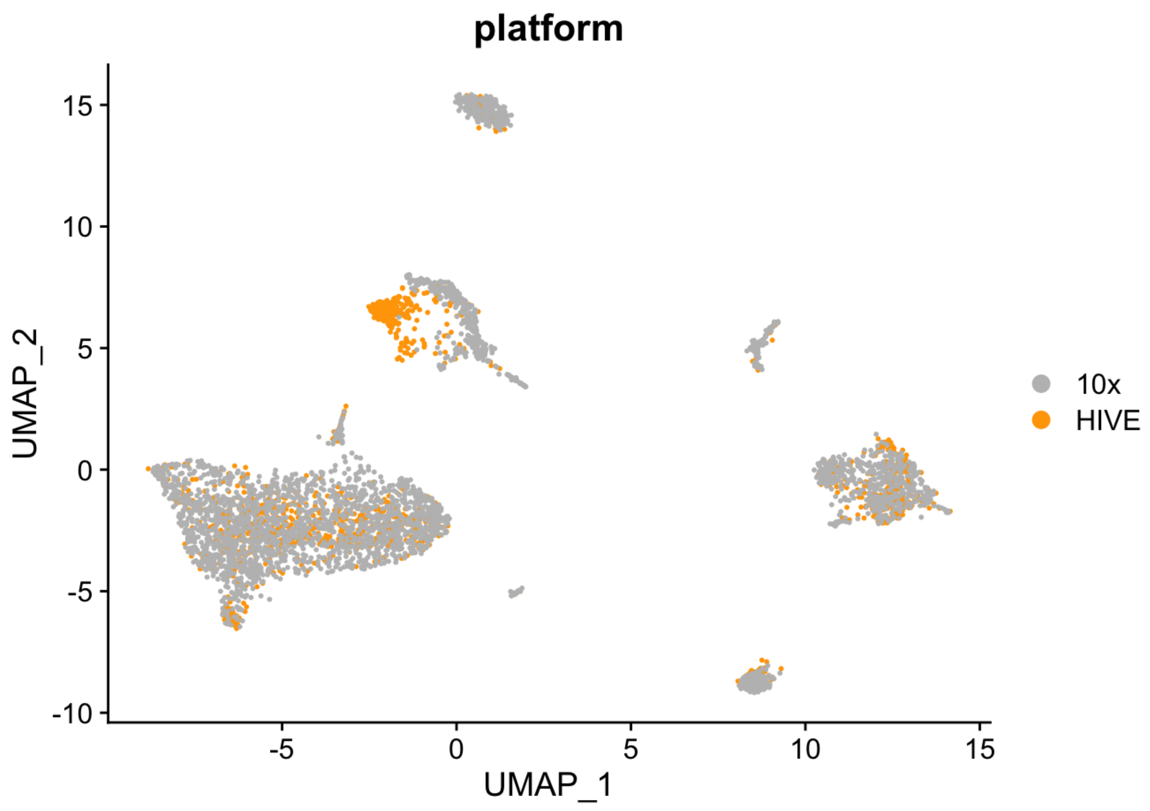
```
plot_marks <- markers_ct %>%
        group_by(cluster) %>%
        slice(1)

DotPlot(obj, plot_marks$gene, assay = "integrated", split.by = "platform",
        cols = c("orange", "blue"), dot.scale = 8) +
        RotatedAxis() +
        theme(text = element_text(size = 8),
              axis.text.y = element_text(size= 8))
```



7 Integration with SCTransform

You can also integrate using SCTransform to regress out variables like percent mitochondrial reads.

```
# Use SCTransform to regress out variation in percent.mito and normalize data
HIVE <- SCTransform(HIVE, vars.to.regress = "percent.mito", verbose = FALSE)
seurat_10x <- SCTransform(seurat_10x, vars.to.regress = "percent.mito", verbose = FALSE)

# Find markers for integration
features <- SelectIntegrationFeatures(object.list = c(HIVE, seurat_10x), nfeatures = 3000)
obj_list <- PrepSCTIntegration(object.list = c(HIVE, seurat_10x), anchor.features = feature
        s)

# Find anchors and integrate data using SCT normalization method
SCTanchors <- FindIntegrationAnchors(object.list = obj_list, normalization.method = "SCT",
                                        anchor.features = features)
obj_SCT <- IntegrateData(anchorset = SCTanchors, normalization.method = "SCT")

# Cluster and visualize
obj_SCT <- ScaleData(obj_SCT, verbose = FALSE)
obj_SCT <- RunPCA(obj_SCT, npcs = 30, verbose = FALSE)
obj_SCT <- RunUMAP(obj_SCT, reduction = "pca", dims = 1:30)
obj_SCT <- FindNeighbors(obj_SCT, reduction = "pca", dims = 1:30)
obj_SCT <- FindClusters(obj_SCT, resolution = 0.5, verbose = F)
obj_SCT <- BuildClusterTree(obj_SCT,reorder=TRUE,reorder.numeric=TRUE,
                        verbose = F)
obj_SCT$seurat_clusters <- obj_SCT$tree.ident

# Visualize results
DimPlot(obj_SCT, group.by = "platform", cols = c("grey", "orange"))
```



For more information on the data integration method used above see Stuart, Butler, et.al 2019 (https://www.cell.com/cell/fulltext/S0092-8674(19)30559-8) or visit the Seurat (https://satijalab.org/seurat/index.html) website for more data integration vignettes.

8 Package info

R Package Info

Package	Loaded version	Date
dplyr	1.1.2	2023-04-20
DT	0.28	2023-05-18
forcats	1.0.0	2023-01-29
ggplot2	3.4.2	2023-04-03
HGNChelper	0.8.1	2019-10-24
lubridate	1.9.2	2023-02-10
openxlsx	4.2.5.2	2023-02-06
purrr	1.0.1	2023-01-10
readr	2.1.4	2023-02-10
Seurat	4.3.0	2022-11-18
SeuratObject	4.1.3	2022-11-07
stringr	1.5.0	2022-12-02
tibble	3.2.1	2023-03-20
tidyr	1.3.0	2023-01-24
tidyverse	2.0.0	2023-02-22