

# Bioinformatics figures for publication

David Chisanga and Wei Shi

03 August, 2022

# Contents

<b>1</b>	<b>Preface</b>	<b>3</b>
<b>2</b>	<b>Prerequisites</b>	<b>4</b>
2.1	Data . . . . .	4
2.2	SOFTWARE . . . . .	5
<b>3</b>	<b>Bulk RNA-seq analysis</b>	<b>7</b>
3.1	Background . . . . .	7
3.2	Multi-Dimensional scaling plot (MDS) or PCA . . . . .	7
3.3	Mean-difference plot . . . . .	8
3.4	Volcano plots . . . . .	9
3.5	Heatmaps . . . . .	11
3.6	Dendrogram . . . . .	13
3.7	Pathway Analysis . . . . .	13
<b>4</b>	<b>scRNA-seq data</b>	<b>21</b>
4.1	Background . . . . .	21
4.2	tSNE plots . . . . .	21
4.3	uMAP . . . . .	24
4.4	Differential expression analysis between clusters . . . . .	26
4.5	Heatmaps . . . . .	27
4.6	Feature plots . . . . .	29
4.7	Cell annotation . . . . .	33
4.8	Trajectory analysis . . . . .	35

# **Chapter 1**

## **Preface**

One of the many questions biologists ask Bioinformaticians is “How can I best represent these results in the form of a figure for my publication?”. This workshop seeks to address this question by focusing on the figures that can be developed to address a biological question and provide explanations of how to interpret such figures. This is aimed at improving understanding of these figures generated in a bioinformatics analysis and help biologists present their discoveries/observations in the best way. This workshop will be very helpful for biologists to prepare bioinf figures for their publications and also for looking into their data in a better way. This will help improve the communications between them and us as well.

Here we will go through some of the most common used figures in the bioinformatics analysis of bulk and single cell RNA-seq data.

# Chapter 2

## Prerequisites

Before you get started with the rest of the analysis, it is important that you have the necessary data and software that will be used in this analysis.

### 2.1 Data

#### 2.1.1 Bulk RNA-seq data

An RNA-seq dataset generated in a published study (Delconte et al., 2016) is used as an example dataset in this protocol.

Two samples (wild-type and Cish-/ natural killer cells) are included in this dataset and each sample has two biological replicates. These data were already deposited in the Gene Expression Omnibus database (GSE79409). However for the convenience of this analysis, processed counts from the Bulk RNA-seq analysis workshop.

#### 2.1.2 scRNA-seq data

A subset of the scRNA-seq dataset that was generated in a published study (Chen et al., 2020) is used as an example dataset. The data consists of two well-characterized cellular reference samples (human breast cancer cell line (HCC1395, sample A) and the matched normal B lymphocyte line (HCC1395BL, sample B)) that were captured using the 10X platform. The data is available in the SRA repository under accession code no. PRJNA504037. However, for the convenience of this analysis, processed counts from the scRNA-seq analysis workshop.

## 2.2 SOFTWARE

The following software tools should be installed on your computer:

- R
- Rstudio
- Rsubread
- limma
- edgeR
- org.Hs.eg.db
- statmod
- Seurat
- SingleR
- monocle

Consult the R Project website for the installation of R (<https://www.r-project.org/>) (R Core Team, 2022). Make sure the latest release version of R is downloaded and installed. After R is installed, launch R and type the following commands to install *Rsubread*, *limma*, *edgeR*, *org.Hs.eg.db*, *statmod*, *Seurat*, *SingleR* and *Monocle*:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(
  c(
    'Rsubread',
    "org.Hs.eg.db",
    "SingleR",
    'BiocGenerics',
    'DelayedArray',
    'DelayedMatrixStats',
    'S4Vectors',
    'SingleCellExperiment',
    'limma',
    'edgeR',
    'SummarizedExperiment',
    'batchelor',
    'Matrix.utils',
    'monocle',
    'celldex'
  ),
  update = T
)

if (!requireNamespace("statmod", quietly = TRUE))
```

```
install.packages("statmod")
if (!requireNamespace("Seurat", quietly = TRUE))
  install.packages("Seurat")
```

# Chapter 3

## Bulk RNA-seq analysis

### 3.1 Background

Previous results from the Bulk RNA-seq analysis workshop were saved in the “Workshop\_RNAseq” directory on All staff share as an R-object “Counts2.RData” or can also be downloaded from here. Import this into R to run the rest of the analysis

```
#load the required package
library(limma)
load("Counts2.RData") # loads the counts dataset
```

There are a number of ways in which you can visualize the results from RNA-seq, while in this chapter we attempt to cover the most common ways of visualizing the data, they are by no means exhaustive.

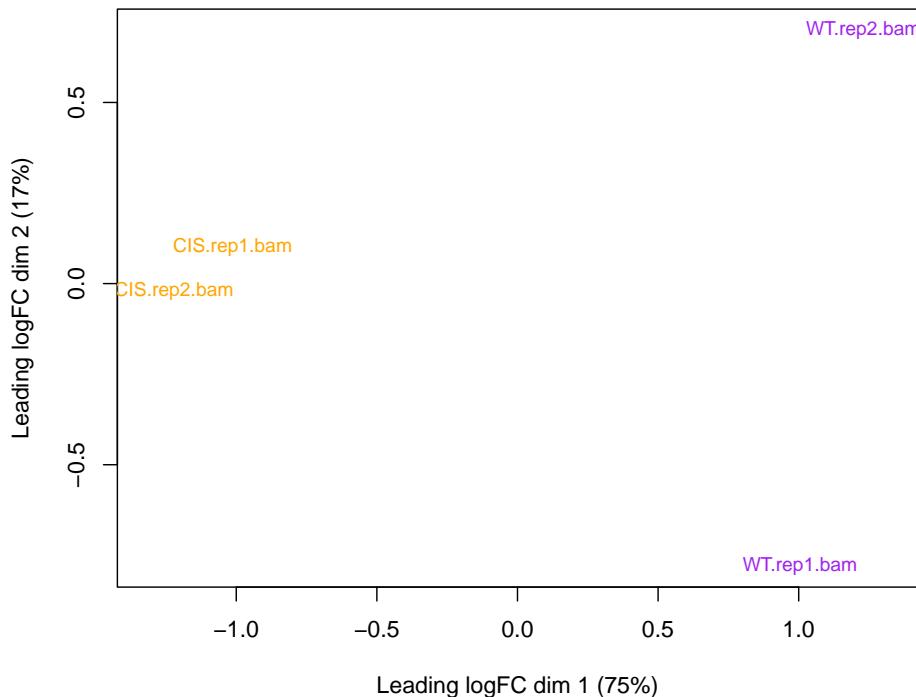
### 3.2 Multi-Dimensional scaling plot (MDS) or PCA

The Multi-Dimensional scaling (MDS) plot is a means of visualizing the level of similarity of individual samples in a dataset. In the plot, the distance between each pair of samples is the root-mean-square deviation (Euclidean distance) for the top genes. Distances on the plot can be interpreted as leading log2-fold-change, meaning the typical (root-mean-square) log2-fold-change between the samples for the genes that distinguish those samples.

```

col<-c("orange", "purple")[factor(targets$CellType)]
plotMDS(
  y,
  cex = 0.8,
  col=col
)

```



```
## Venn diagram
```

If differential expression analysis is performed on more than 1 comparison, a Venn diagram can be used to quickly compare the number of differentially expressed genes that are either unique to one comparison or DE in other comparisons. This can be easily done using the `vennDiagram` function in **limma**.

### 3.3 Mean-difference plot

A mean-difference plot or MD-plot is a plot that can be used to show the fold-change differences against the average expression values of all genes used in the analysis. This can be generated easily using the `plotMD` function within the **limma** package.

```
plotMD(fit.contr, column = 1,status = dt,cex=0.8)
```

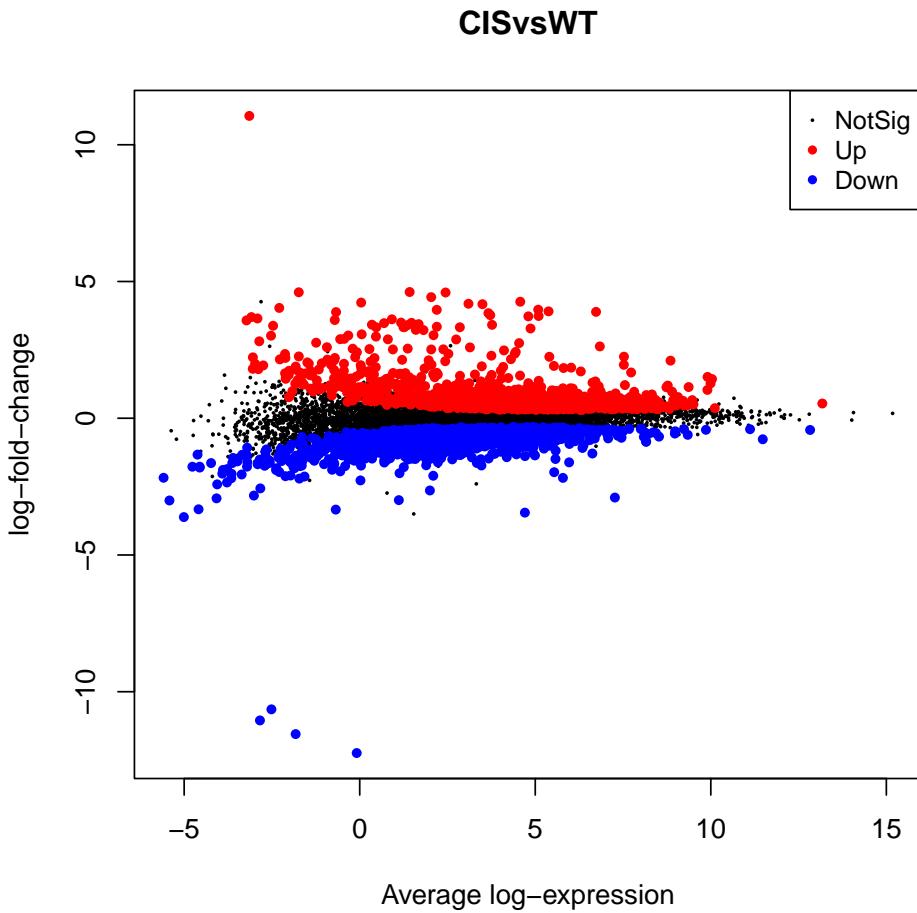


Figure 3.1: Mean-Difference plot. Represents relationship between mean expression and fold-changes

### 3.4 Volcano plots

A volcano plot shows the relationship between the log fold-change on the x-axis and a measure of statistical significance on the y-axis. The measure of significance can be  $-\log(p\text{-value})$  or the B-statistics. Here, we will use the *volcanoplot* function within the **limma** package. In the volcano plot below, differentially expressed genes are highlighted with red for up-regulated and blue for down-regulated genes.

```
#highlight de genes
col<-c("blue","black","red")[factor(dt[, "CISvsWT"])]
volcanoplot(fit.contr,col=col)
```

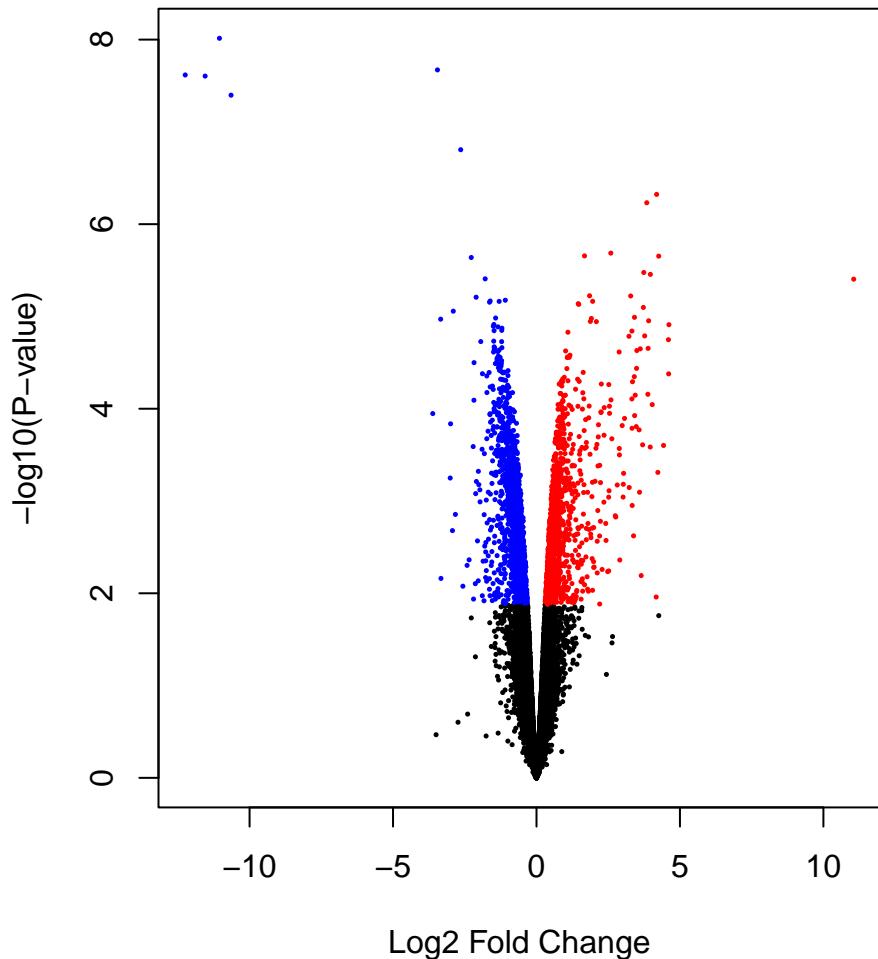


Figure 3.2: Volcano plot. Shows the relationship between fold-change and significance of change

Alternatively, an interactive mean-difference plot can be generated using *glMD-Plot* in **Glimma** which outputs an interactive html page with results displayed in the left panel and the expression values for the selected gene. A gene within the dataset can be searched through a search bar.

```

if(!require(Glimma))
{
  BiocManager::install("Glimma")
  library(Glimma)
}

## Loading required package: Glimma

glMDPlot(fit.contr, coef=1, status=dt, main=colnames(fit.contr)[1],
          side.main="ENTREZID", counts=y$E, groups=colnames(contr),
          launch = F)

```

[Click here for interactive version](#)

Hover over points to see sample-wise expression

- Click on column names to sort by column
- Click rows on tables to highlight gene

## 3.5 Heatmaps

Heatmaps are a great way of demonstrating the differences in the expression patterns between different conditions in your dataset. Here we use the *coolmap* function within the **limma** package which is an extension of the *heatmap.2* function within the **gplots** package. The heatmap below shows the expression pattern of the top 100 differentially expressed genes between CIS vs WT.

```

heat.data <- y$E[head(rownames(de.genes), n = 100), ]
colnames(heat.data) <- targets$Sample
rownames(heat.data) <- head(de.genes$Symbol, n = 100)
coolmap(
  heat.data,
  margin = c(5, 4),
  cexCol = 0.8,
  lhei = c(0.8, 4),
  cexRow = 0.6
)

```

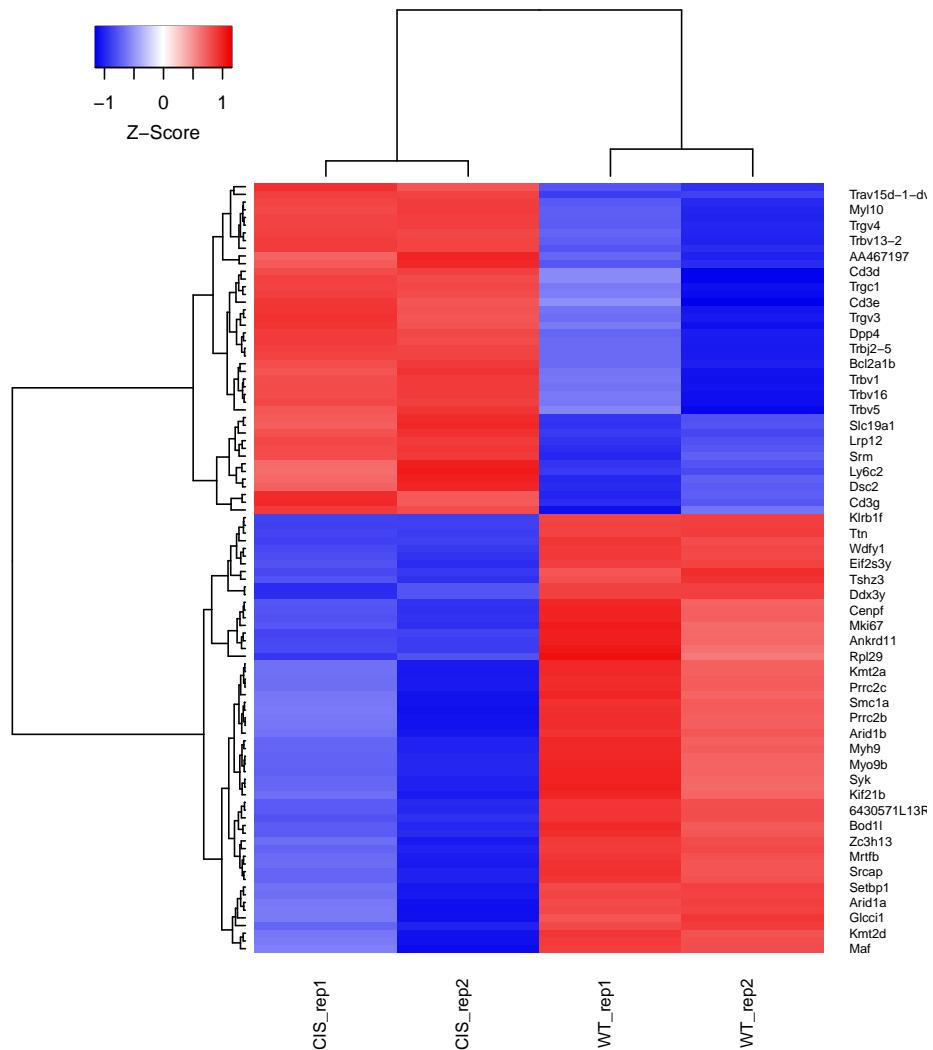


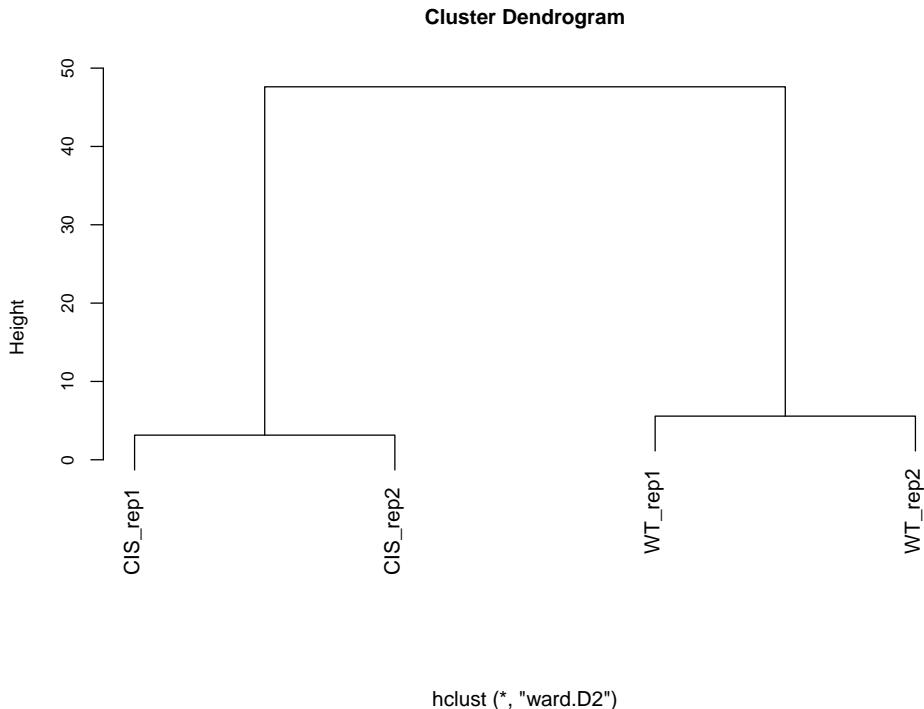
Figure 3.3: Heatmap plot of differentially expressed genes in CIS vs WT

### 3.6 Dendrogram

Similarly we can also use cluster dendrogram to show the relationship between samples.

```
d<-dist(t(heat.data))
hclst<-hclust(d,method = "ward.D2")
```

```
plot(
  hclst,
  cex.main = 1,
  cex.lab = 0.9,
  xlab ="",
  cex.axis = 0.8)
```



`hclust (*, "ward.D2")`

### 3.7 Pathway Analysis

From the differential expression analysis there were some 3181 differentially expressed genes. This is quiet a long list of genes to interpret and nearly impossible to go through a gene at a time to gain any meaningful biological understanding.

A common downstream analysis step is to understand pathways/gene networks that the genes are implicated in.

### 3.7.1 Gene Ontology enrichment

Gene ontology (<http://www.geneontology.org/>) provides a controlled vocabulary for describing biological processes (BP ontology), molecular functions (MF ontology) and cellular components (CC ontology).

The GO ontologies themselves are organism-independent; terms are associated with genes for a specific organism through direct experimentation or through sequence homology with another organism and its GO annotation.

Terms are related to other terms through parent-child relationships in a directed acyclic graph.

You can use enrichment analysis as another way of drawing conclusions from your set of differentially expressed genes.

```
enrich.pvalue<-0.00001
```

Here, we use the *goana* function within the **limma** package to test for enrichment of differentially expressed genes between the CIS vs WT. Note that the *p-values* returned by *goana* are **unadjusted for multiple testing**. It is therefore, advisable that if the results are to be published, only terms with very small p-values should be included. For instance, in the example below, only terms with a p-value  $< 10^{-5}$  are retained.

We use the output from the differential expression analysis and test for over-representation for the Up and Down regulated genes separately.

```
if(!requireNamespace("GO.db"))
  BiocManager::install("GO.db")
library(GO.db)
go.rst<-goana(fit.contr,species="Mm")
#exclude terms not meeting our cut-off
go.rst<-go.rst[rowSums(go.rst[,c("P.Up","P.Down")]<enrich.pvalue)==1,]
```

A simple way to visualize the enrichment results is through a bar plot as shown below which shows the top 10 enriched terms in each direction.

```
top.up <- head(go.rst[order(go.rst$P.Up, decreasing = F),], n = 10)
top.down <-
  head(go.rst[order(go.rst$P.Down, decreasing = F),], n = 10)
bar.data <-
  rbind(
```

```

data.frame(
  Term = top.up$Term,
  P.value = top.up$P.Up,
  Dir = "up"
),
data.frame(
  Term = top.down$Term,
  P.value = top.down$P.Down,
  Dir = "down"
)
)
bar.data <- bar.data[order(bar.data$P.value, decreasing = F),]
par(mai = c(0.8, 2, 0.5, 0.5))
bb <- barplot(
  -log10(bar.data$P.value),
  horiz = T,
  xlab = " -log10(p-value)",
  cex.names = 0.5,
  col = c("red", "blue")[factor(bar.data$Dir)])
)
axis(
  2,
  line = -0.8,
  at = bb,
  labels = bar.data$Term,
  tick = F,
  las = 2,
  cex.axis = 0.5
)
legend("topright", legend = c("Up", "Down"), pch = 22, pt.bg = c("red", "blue"))

```

### Add gene information (optional)

Sometimes you may want to know what genes are enriched in each of the terms, this section allows you to add gene details.

First, we extract the genes that are associated with the catalytic activity GO terms. This results in a list mapping the GO term and the associated genes.

```

if(!require(GO.db))
  BiocManager::install("GO.db")
if(!require(org.Mm.eg.db))
  BiocManager::install("org.Mm.eg.db")

## Loading required package: org.Mm.eg.db

```

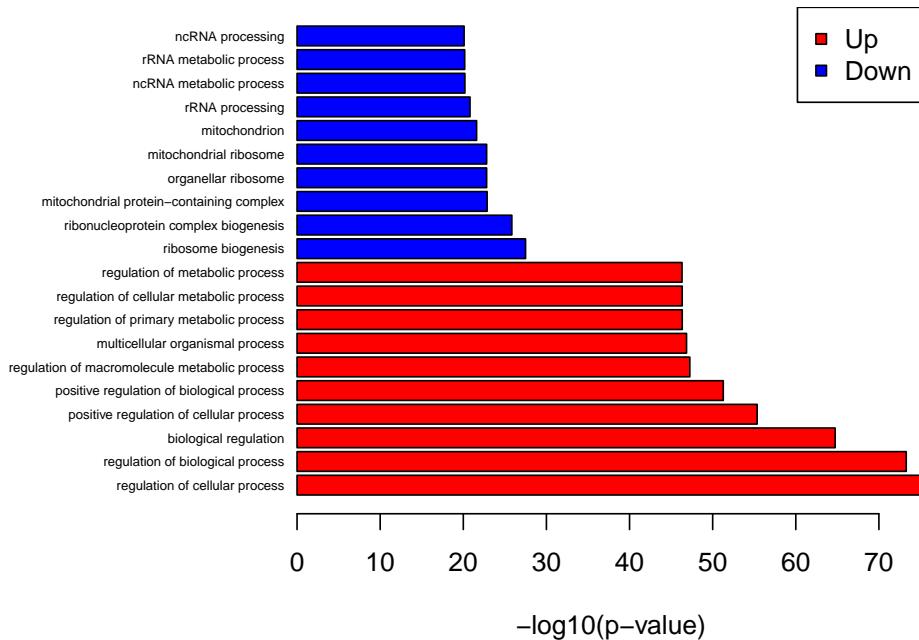


Figure 3.4: Enrichment plot. Top enrichments in either direction

```
go.EntrezID <- as.list(org.Mm.egGO2ALLEGES)
go.rst <- tibble::rownames_to_column(go.rst, var = "GO.ID")
```

```
#Add gene details
go.rst$Genes.Up <- unlist(lapply(go.rst$GO.ID, function(x) {
  xx <- go.EntrezID[[x]]
  if (is.null(xx) | 
      with(go.rst[go.rst$GO.ID == x,], Up == 0 |
          P.Up > enrich.pvalue))
    return("—")
  x <- intersect(with(de.genes,EntrezID[logFC>0]),xx)
  x <-
    paste0(with(fit.contr$genes, Symbol[EntrezID %in% xx]), collapse = " | ")
  return(x)
})) 
go.rst$Genes.Down <- unlist(lapply(go.rst$GO.ID, function(x) {
  xx <- go.EntrezID[[x]]
  if (is.null(xx) | 
      with(go.rst[go.rst$GO.ID == x,], Down == 0 |
          P.Down > enrich.pvalue))
    return("—")})
```

```

x <- intersect(with(de.genes,EntrezID[logFC<0]),xx)
x <-
  paste0(with(fit.contr$genes, Symbol[EntrezID %in% xx]), collapse = "|")
return(x)
}())

```

### 3.7.2 KEGG pathway enrichment

Pathway enrichment can also be performed against the Kyoto Encyclopedia of Genes and Genomes (KEGG) (<https://www.genome.jp/kegg/>) database. The KEGG database contains curated molecular pathways and disease signatures. To perform this analysis we use the *kegga* function within the **limma** which is similar to the *goana* function we used above.

```

kegg.rst<-kegga(fit.contr,species="Mm")
topKEGG(kegg.rst)

```

	Pathway	N	Up	Down
## path:mmu03008	Ribosome biogenesis in eukaryotes	75	37	4
## path:mmu01100	Metabolic pathways	1038	206	75
## path:mmu04810	Regulation of actin cytoskeleton	143	6	45
## path:mmu05206	MicroRNAs in cancer	121	6	40
## path:mmu04919	Thyroid hormone signaling pathway	93	3	32
## path:mmu03010	Ribosome	137	39	1
## path:mmu01240	Biosynthesis of cofactors	111	32	2
## path:mmu04330	Notch signaling pathway	43	2	18
## path:mmu00310	Lysine degradation	52	1	20
## path:mmu03020	RNA polymerase	32	14	1
## path:mmu00190	Oxidative phosphorylation	118	32	0
## path:mmu04062	Chemokine signaling pathway	119	5	34
## path:mmu04611	Platelet activation	77	0	25
## path:mmu05200	Pathways in cancer	323	26	71
## path:mmu05205	Proteoglycans in cancer	126	8	34
## path:mmu04658	Th1 and Th2 cell differentiation	69	6	22
## path:mmu05224	Breast cancer	81	8	24
## path:mmu05202	Transcriptional misregulation in cancer	121	16	32
## path:mmu03050	Proteasome	44	15	0
## path:mmu01232	Nucleotide metabolism	65	19	0
	P.Up	P.Down		
## path:mmu03008	1.182585e-14	9.951210e-01		
## path:mmu01100	5.674601e-12	1.000000e+00		
## path:mmu04810	9.998630e-01	4.679254e-08		
## path:mmu05206	9.987434e-01	5.746746e-08		
## path:mmu04919	9.996663e-01	4.319019e-07		

```

## path:mmu03010 6.123733e-07 1.000000e+00
## path:mmu01240 4.561006e-06 9.999989e-01
## path:mmu04330 9.789811e-01 6.418221e-06
## path:mmu00310 9.991527e-01 9.325271e-06
## path:mmu03020 1.341355e-05 9.916924e-01
## path:mmu00190 1.832128e-05 1.000000e+00
## path:mmu04062 9.995796e-01 2.081620e-05
## path:mmu04611 1.000000e+00 2.435669e-05
## path:mmu05200 9.971966e-01 4.089375e-05
## path:mmu05205 9.933378e-01 7.494761e-05
## path:mmu04658 8.861202e-01 1.000116e-04
## path:mmu05224 8.230146e-01 1.784375e-04
## path:mmu05202 4.708773e-01 1.811724e-04
## path:mmu03050 2.086057e-04 1.000000e+00
## path:mmu01232 3.146027e-04 1.000000e+00

```

### 3.7.3 Gene Set Enrichment Analysis

GO and KEGG enrichment analysis uses the list of differentially expressed genes to test for over-representation of these genes. As such the results typically depend on the cut-offs used to get the differentially expressed genes.

Gene Set Enrichment Analysis is performed using *roast*, a gene set enrichment function within the **limma** package. Unlike above where we focused on a subset of genes(differentially genes), here all the genes in the set are used to test for gene expression signatures or pathways. A test is performed against the GO terms for demonstrative purposes, alternatively, if you wish to test against a specific gene set, this can quickly be done by importing that list.

We then use *roast* to test for enrichment between CIS and WT. We use the design matrix ‘design’, the expression values ‘y’ and the contrast matrix ‘contr’ from the imported results to perform the gene set test and the GO term and gene map we created above.

```

go.all<-go.EntreZID[go.rst$GO.ID]
names(go.all)<-go.rst$Term
roast.GSEA<-roast(y, index = go.all, design = design, contrast = contr)
head(roast.GSEA)

```

	NGenes	PropDown	PropUp	Direction
## 90S preribosome	28	0.07142857	0.7142857	Up
## RNA processing	809	0.26205192	0.4066749	Up
## preribosome	79	0.06329114	0.6708861	Up
## meiotic chromosome segregation	63	0.55555556	0.0952381	Down
## mitotic sister chromatid segregation	150	0.52000000	0.1666667	Down

```

## regulation of chromosome separation      69 0.47826087 0.1884058      Down
##                                         PValue      FDR PValue.Mixed  FDR.Mixed
## 90S preribosome                      0.0005 0.1000000 0.0060 0.06227709
## RNA processing                       0.0005 0.1000000 0.0555 0.06227709
## preribosome                          0.0020 0.1253906 0.0130 0.06227709
## meiotic chromosome segregation       0.0035 0.1253906 0.0210 0.06227709
## mitotic sister chromatid segregation 0.0035 0.1253906 0.0135 0.06227709
## regulation of chromosome separation   0.0035 0.1253906 0.0160 0.06227709

term.test<-rownames(roast.GSEA)[1]

```

Next, we can visualize the gene set enrichment results using a barcode plot. The barcode plot is used to plot the positions of genes within a set in ranked list of statistics. The statistics are ranked from left to right ie. from smallest to largest.

To do so, the gene identifiers for the term should first be mapped to the index position of each gene in the expression matrix;

```

index <- rownames(fit.contr) %in% go.all[[rownames(roast.GSEA)[1]]]
barcodeplot(
  fit.contr$coefficients[, "CISvsWT"],
  index = index,
  labels = c("CS", "WT"),
  main = rownames(roast.GSEA)[1]
)

```

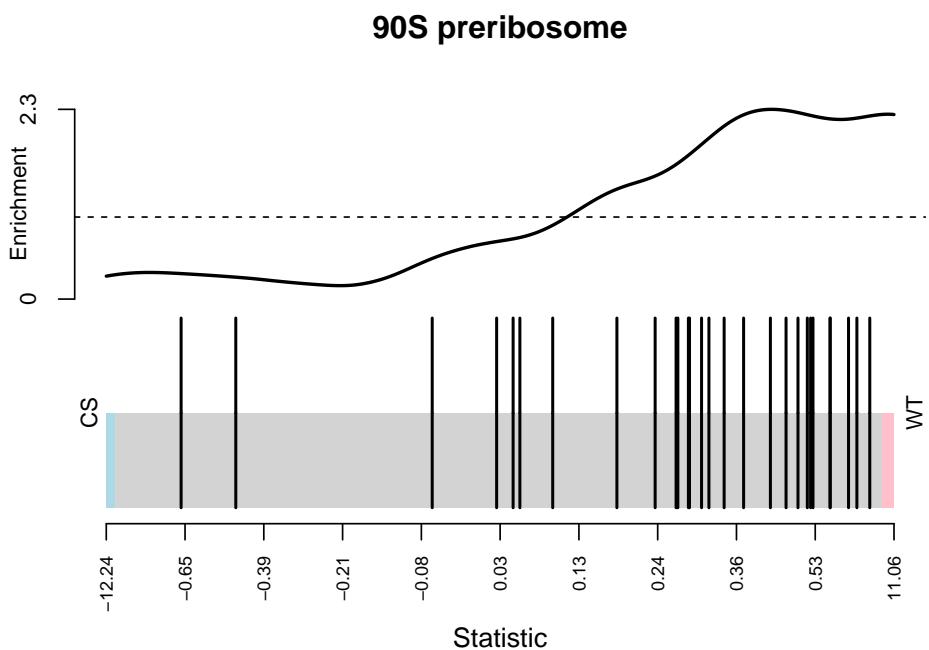


Figure 3.5: Barcode plot showing the enrichment of 90S preribosome in CS vs WT

# Chapter 4

## scRNA-seq data

### 4.1 Background

As discussed previously, processed counts from the scRNA-seq workshop are used here to demonstrate the different types of figures that can be generated for publications.

### 4.2 tSNE plots

t-distributed stochastic neighbor embedding (t-SNE) is a statistical method for visualizing high-dimensional data by giving each data point a location in a two or three-dimensional map. It is a dimensionality reduction technique which is a way to graphically simplify very large datasets.

```
library(Seurat)

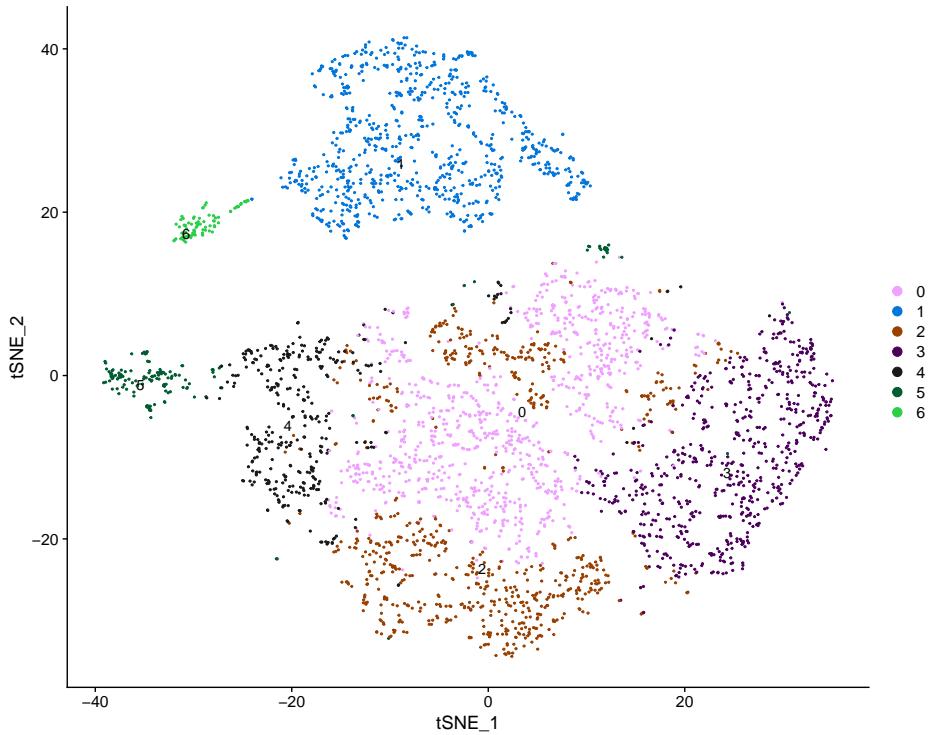
## Attaching SeuratObject

## Attaching sp

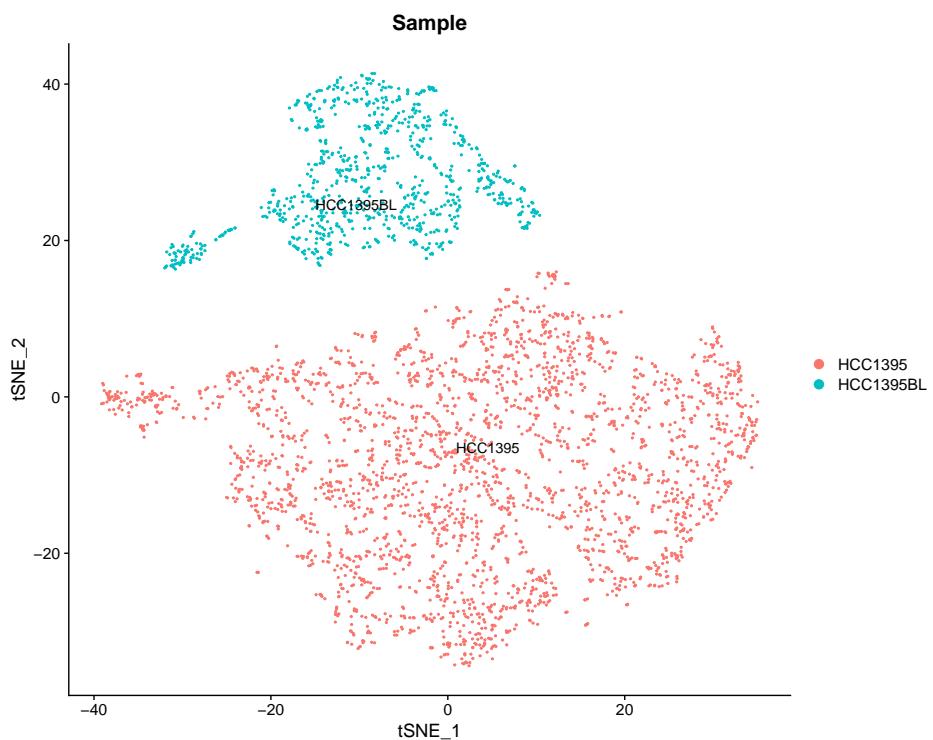
library(ggplot2)
load("Counts_scRNA-norm.RData")

DimPlot(
  counts_st,
  reduction = "tsne",
```

```
label = T,  
size = 0.5,  
repel = T,  
cols = DiscretePalette(length(levels(Idents(  
    counts_st  
))))  
)
```



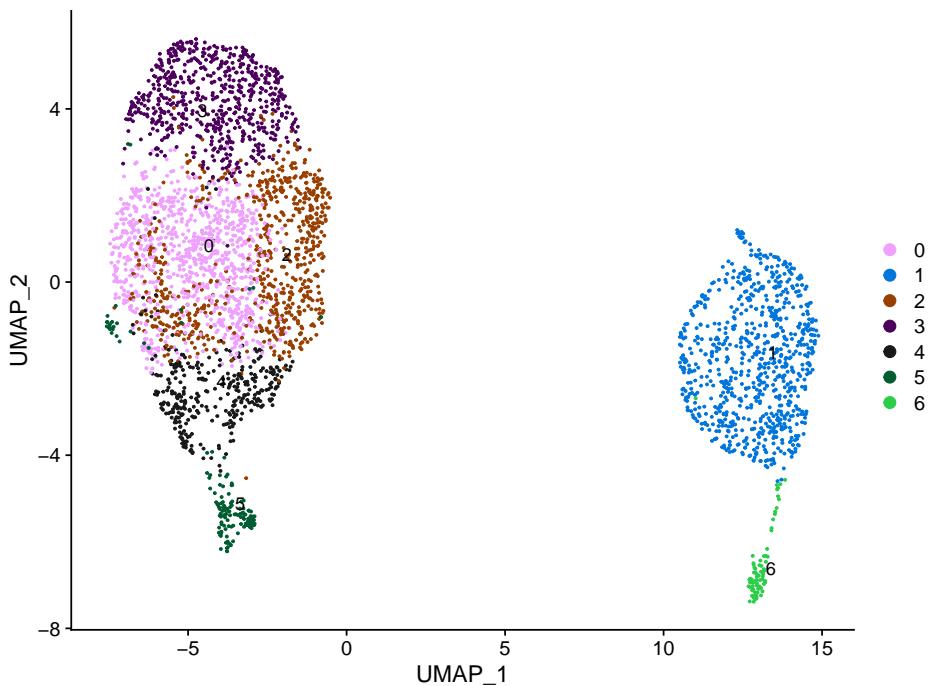
```
DimPlot(  
  counts_st,  
  reduction = "tsne",  
  label = T,  
  group.by = "Sample",  
  size = 0.5,  
  repel = T  
)
```



### 4.3 uMAP

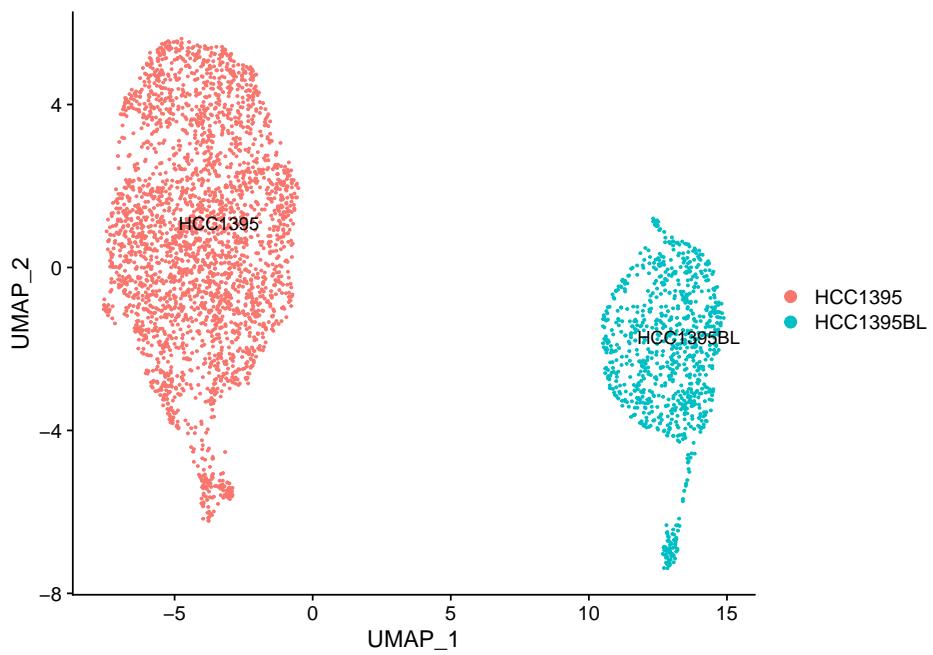
Optionally, we can also perform dimension reducibility reduction using UMAP

```
DimPlot(
  counts_st,
  reduction = "umap",
  label = T,
  size = 0.5,
  repel = T,
  cols = DiscretePalette(length(levels(Idents(
    counts_st
  ))))
)
```



```
DimPlot(
  counts_st,
  reduction = "umap",
  label = T,
  group_by = "Sample",
  size = 0.5,
```

```
repel = T  
)+ggtitle("")
```



## 4.4 Differential expression analysis between clusters

To identify differentially expressed genes between clusters we can use the *FindMarker* function in Seurat. As an example, DE genes between clusters 0 and 1 are computed below.

```
DE.cluster0_1 <- FindMarkers(
  counts_st,
  ident.1 = 0,
  ident.2 = 1,
  verbose = F)
head(DE.cluster0_1[order(abs(DE.cluster0_1$avg_log2FC),
                           decreasing = T),])
```

	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
## CD74	0.000000e+00	-5.891577	0.008	1.000	0.000000e+00
## IGHM	0.000000e+00	-5.729445	0.000	0.998	0.000000e+00
## S100A6	0.000000e+00	5.598281	1.000	0.047	0.000000e+00
## TMSB4X	1.6575656e-278	-4.911549	0.970	1.000	2.935713e-274
## CCL3	6.580286e-308	-4.861987	0.000	0.945	1.165434e-303
## KRT81	4.273972e-296	4.593045	0.962	0.000	7.569632e-292

## 4.5 Heatmaps

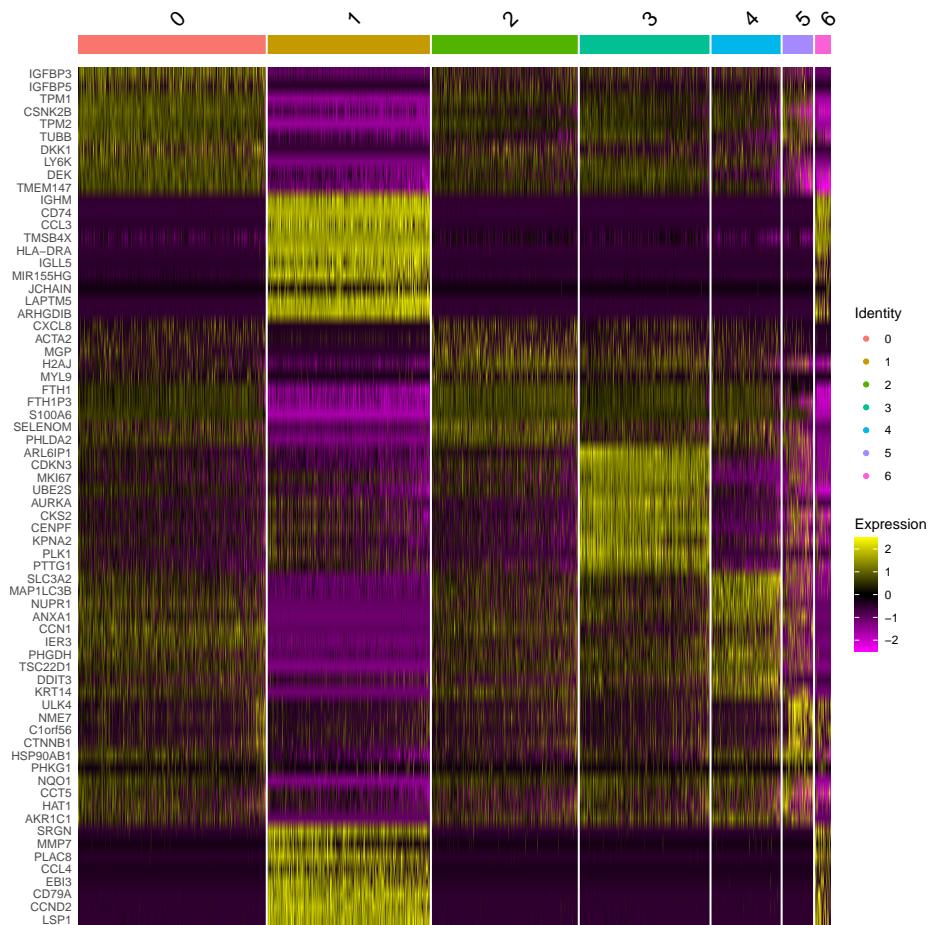
Marker genes which can be used to uniquely identify each of the clusters are identified using the *FindAllMarkers* function.

```
library(dplyr)
all.markers %>%
  group_by(cluster) %>%
  slice_max(n = 5, order_by = avg_log2FC)

## # A tibble: 35 x 7
## # Groups:   cluster [7]
##       p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##       <dbl>     <dbl> <dbl> <dbl>    <dbl> <fct>  <chr>
## 1 1.61e-155      1.23  0.94  0.525 2.86e-151 0     IGFBP3
## 2 4.07e- 65      1.17  0.417  0.149 7.21e- 61 0     IGFBP5
## 3 2.12e-147      1.06  0.998  0.714 3.75e-143 0     TPM1
## 4 2.31e-202      0.938 0.994  0.869 4.09e-198 0     CSNK2B
## 5 1.78e-131      0.901 0.998  0.665 3.15e-127 0     TPM2
## 6 0               4.79   0.998  0.021 0           1     IGHM
## 7 0               4.63   1      0.037 0           1     CD74
## 8 0               4.22   0.945  0.015 0           1     CCL3
## 9 0               4.08   1      0.959 0           1     TMSB4X
## 10 0              3.48   0.999  0.033 0           1     HLA-DRA
## # ... with 25 more rows
## # i Use `print(n = ...)` to see more rows

top10.markers<-all.markers %>%
  group_by(cluster) %>%
  slice_max(n = 10, order_by = avg_log2FC)

DoHeatmap(counts_st, features = top10.markers$gene)
```

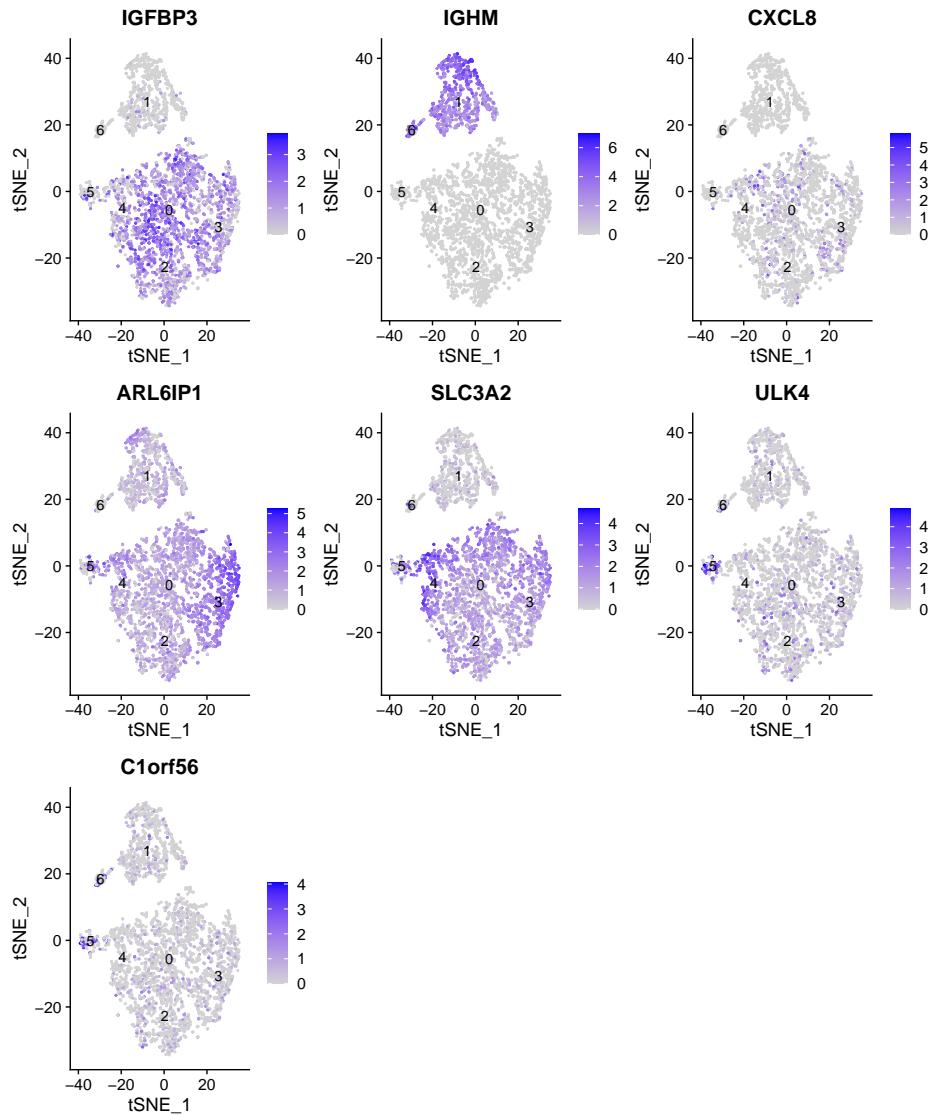


## 4.6 Feature plots

We can also highlight the expression of genes of interest on the clusters by way of a tSNE plot.

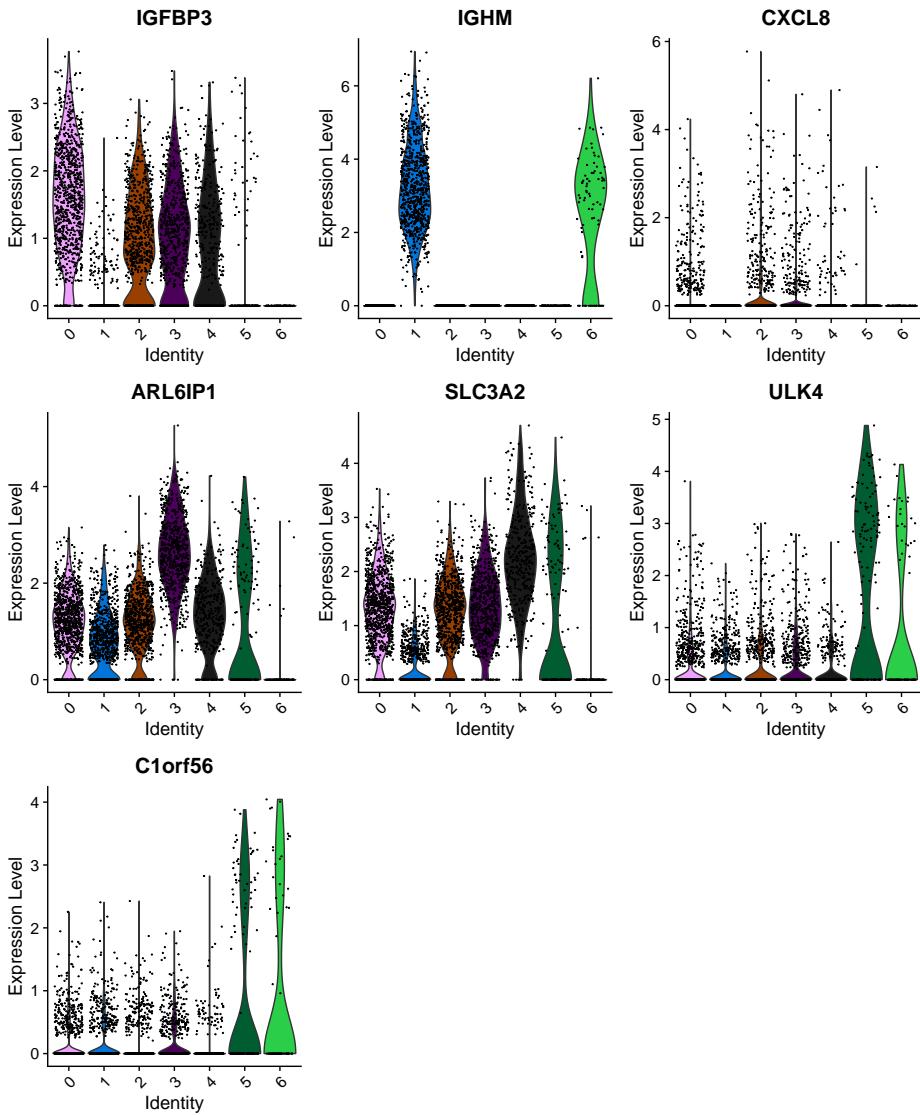
```
#Get the top marker gene per cluster
top.markers<-all.markers %>%
  group_by(cluster) %>%
  slice_max(n = 1, order_by = avg_log2FC)

FeaturePlot(
  counts_st,
  features = top.markers$gene,
  ncol = 3,
  label = T,
  reduction = "tsne"
)
```



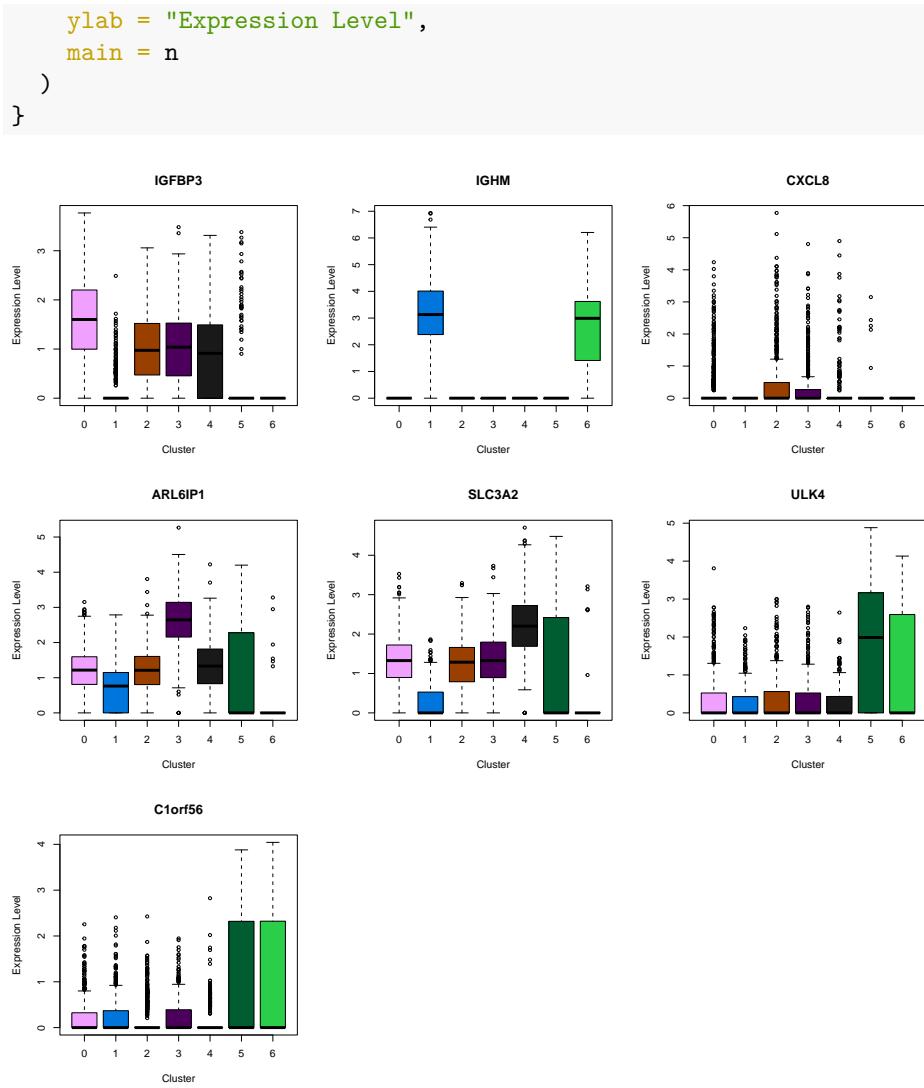
Alternatively, we can also use violin plots or boxplots to show the expression profile of genes of interest across cells by cluster or any other combination of cells

```
VlnPlot(counts_st,
        features = top.markers$gene,
        cols = DiscretePalette(length(unique(
        counts_st$seurat_clusters
        ))))
```



```

par(mfrow = c(3, 3))
for (n in top.markers$gene)
{
  n.data <- counts_st[n, ]
  boxplot(
    as.numeric(n.data@assays$RNA@data) ~ as.character(n.data$seurat_clusters),
    col = DiscretePalette(length(unique(
      counts_st$seurat_clusters
))), 
    xlab = "Cluster",
  )
}
  
```



## 4.7 Cell annotation

An unbiased cell type recognition is performed using **SingleR**. **celldex** has a range of annotations derived from Bulk RNA-seq data that can be used to annotate the identified clusters above. Here, we use the Human Primary Cell Atlas database as an example.

```
library(SingleR)
library(SingleCellExperiment)

cell.sce<- as.SingleCellExperiment(counts_st)
annot<-celldex::HumanPrimaryCellAtlasData()
cell.annots <- SingleR(
  test = cell.sce,
  ref = annot,
  clusters = cell.sce$seurat_clusters,
  labels = annot$label.main)

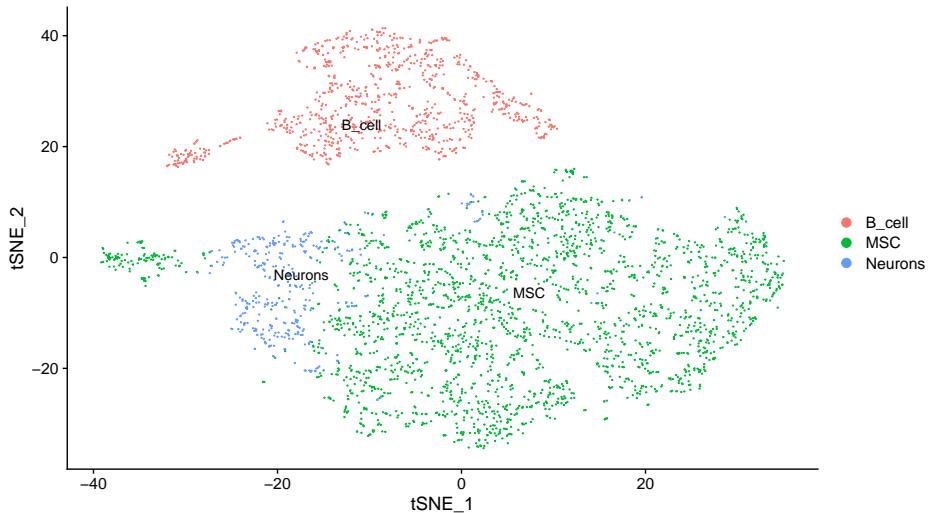
cell.annots.fine<-SingleR(
  test = cell.sce,
  ref = annot,
  clusters = cell.sce$seurat_clusters,
  labels = annot$label.fine)

save(cell.annots,cell.annots.fine,file = "Annotations.RData")

counts_st <-
  AddMetaData(counts_st, cell.annots[match(counts_st@meta.data$seurat_clusters,rownames(cell.annots)),])

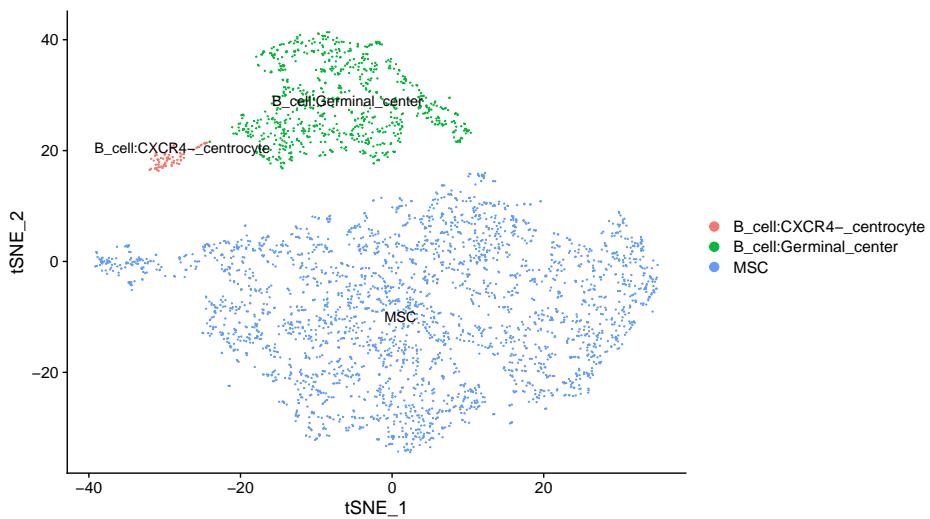
library(ggplot2)

DimPlot(
  counts_st,
  reduction = "tsne",
  group.by = "Annot.main",
  label = T,
  repel = T,
  pt.size = 0.1,
) + ggtitle(label = "")
```



```
counts_st <- AddMetaData(counts_st, cell.annots.fine[match(counts_st@meta.data$seurat_
```

```
DimPlot(
  counts_st,
  reduction = "tsne",
  group.by = "Annot.fine",
  label = T,
  repel = T,
  pt.size = 0.1
) + ggtitle(label = "")
```



## 4.8 Trajectory analysis

Pseudotime analysis of the cells identified in the dataset was performed using **Monocle3**.

```
mono2.learn.traject <-
  function(X_counts,
          these.cell.types) {
    library(monocle3)
    rds.fname <- "Trajectory-cds.rds"
    gsndf <- data.frame(gene_short_name = rownames(X_counts))
    csndf <- data.frame(cell.type = these.cell.types)
    rownames(gsndf) <- rownames(X_counts)
    rownames(csndf) <- colnames(X_counts)
    cds <- new_cell_data_set(
      expression_data = X_counts,
      cell_metadata = csndf,
      gene_metadata = gsndf
    )
    cds <- preprocess_cds(cds, num.dim = 50)
    #cds <- align_cds(cds)
    cds <- reduce_dimension(cds)
    cds <- cluster_cells(cds)
    cds <- learn_graph(cds)
    #cds <- order_cells(cds)
    saveRDS(cds , rds.fname)
  }

mono2.learn.traject(
  X_counts = as.matrix(counts_st@assays$RNA@counts),
  these.cell.types = counts_st$Sample
)

mono.rds <- readRDS("Trajectory-cds.rds")
monocle3::plot_cells(
  mono.rds,
  color_cells_by = "cell.type",
  label_groups_by_cluster = FALSE,
  label_leaves = FALSE,
  label_branch_points = FALSE
)
```

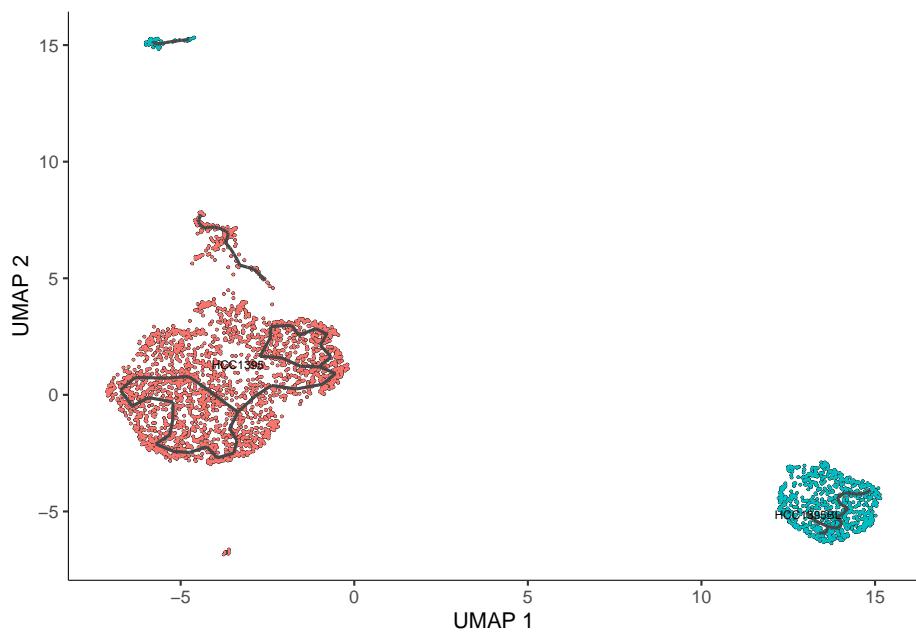


Figure 4.1: Pseudo-time plot showing how individual cells progress through the development

**Session info**

```
sessionInfo()

## R version 4.2.0 (2022-04-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur/Monterey 10.16
##
## Matrix products: default
## BLAS:    /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK:  /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8
##
## attached base packages:
## [1] stats4      stats       graphics    grDevices   utils       datasets    methods
## [8] base
##
## other attached packages:
##  [1] SingleCellExperiment_1.18.0 SingleR_1.10.0
##  [3] SummarizedExperiment_1.26.1 GenomicRanges_1.48.0
##  [5] GenomeInfoDb_1.32.2        MatrixGenerics_1.8.1
##  [7] matrixStats_0.62.0         dplyr_1.0.9
##  [9] ggplot2_3.3.6              sp_1.5-0
## [11] SeuratObject_4.1.0         Seurat_4.1.1
## [13] org.Mm.eg.db_3.15.0        GO.db_3.15.0
## [15] AnnotationDbi_1.58.0       IRanges_2.30.0
## [17] S4Vectors_0.34.0           Biobase_2.56.0
## [19] BiocGenerics_0.42.0        Glimma_2.6.0
## [21] limma_3.52.2
##
## loaded via a namespace (and not attached):
##  [1] utf8_1.2.2                  reticulate_1.25
##  [3] lme4_1.1-30                 tidyselect_1.1.2
##  [5] RSQLite_2.2.15               htmlwidgets_1.5.4
##  [7] grid_4.2.0                  BiocParallel_1.30.3
##  [9] Rtsne_0.16                  munsell_0.5.0
## [11] ScaledMatrix_1.4.0          codetools_0.2-18
## [13] ica_1.0-3                  future_1.27.0
## [15] miniUI_0.1.1.1              withr_2.5.0
## [17] spatstat.random_2.2-0       colorspace_2.0-3
## [19] progressr_0.10.1            highr_0.9
## [21] knitr_1.39                  rstudioapi_0.13
## [23] ROCR_1.0-11                 tensor_1.5
```

```

## [25] listenv_0.8.0           labeling_0.4.2
## [27] GenomeInfoDbData_1.2.8 polyclip_1.10-0
## [29] bit64_4.0.5            farver_2.1.1
## [31] parallelly_1.32.1       vctrs_0.4.1
## [33] generics_0.1.3          xfun_0.31
## [35] R6_2.5.1                ggbeeswarm_0.6.0
## [37] rsvd_1.0.5              locfit_1.5-9.6
## [39] bitops_1.0-7             spatstat.utils_2.3-1
## [41] cachem_1.0.6            DelayedArray_0.22.0
## [43] assertthat_0.2.1         promises_1.2.0.1
## [45] scales_1.2.0             rgeos_0.5-9
## [47] beeswarm_0.4.0           gtable_0.3.0
## [49] beachmat_2.12.0          globals_0.15.1
## [51] goftest_1.2-3            rlang_1.0.4
## [53] genefilter_1.78.0         splines_4.2.0
## [55] lazyeval_0.2.2            spatstat.geom_2.4-0
## [57] yaml_2.3.5                reshape2_1.4.4
## [59] abind_1.4-5              httpuv_1.6.5
## [61] tools_4.2.0                bookdown_0.27
## [63] ellipsis_0.3.2            gplots_3.1.3
## [65] spatstat.core_2.4-4        RColorBrewer_1.1-3
## [67] gggridges_0.5.3           Rcpp_1.0.9
## [69] plyr_1.8.7                sparseMatrixStats_1.8.0
## [71] zlibbioc_1.42.0            purrr_0.3.4
## [73] RCurl_1.98-1.8            rpart_4.1.16
## [75] deldir_1.0-6               pbapply_1.5-0
## [77] cowplot_1.1.1              zoo_1.8-10
## [79] ggrepel_0.9.1              cluster_2.1.3
## [81] magrittr_2.0.3              data.table_1.14.2
## [83] scattermore_0.8             lmtest_0.9-40
## [85] RANN_2.6.1                 fitdistrplus_1.1-8
## [87] patchwork_1.1.1            mime_0.12
## [89] evaluate_0.15              xtable_1.8-4
## [91] XML_3.99-0.10             gridExtra_2.3
## [93] compiler_4.2.0              monocle3_1.2.9
## [95] tibble_3.1.8                KernSmooth_2.23-20
## [97] crayon_1.5.1              minqa_1.2.4
## [99] htmltools_0.5.3            mgcv_1.8-40
## [101] later_1.3.0              tidyverse_1.2.0
## [103] geneplotter_1.74.0          DBI_1.1.3
## [105] MASS_7.3-58                boot_1.3-28
## [107] Matrix_1.4-1              cli_3.3.0
## [109] parallel_4.2.0             igraph_1.3.4
## [111] pkgconfig_2.0.3            terra_1.5-21
## [113] plotly_4.10.0              spatstat.sparse_2.1-1
## [115] annotate_1.74.0            vipor_0.4.5

```

```
## [117] XVector_0.36.0
## [119] digest_0.6.29
## [121] RcppAnnoy_0.0.19
## [123] Biostrings_2.64.0
## [125] leiden_0.4.2
## [127] edgeR_3.38.2
## [129] shiny_1.7.2
## [131] nloptr_2.0.3
## [133] nlme_3.1-158
## [135] BiocNeighbors_1.14.0
## [137] fansi_1.0.3
## [139] lattice_0.20-45
## [141] KEGGREST_1.36.3
## [143] httr_1.4.3
## [145] glue_1.6.2
## [147] bit_4.0.4
## [149] blob_1.2.3
## [151] BiocSingular_1.12.0
## [153] memoise_2.0.1
## [155] future.apply_1.9.0
stringr_1.4.0
sctransform_0.3.3
spatstat.data_2.2-0
rmarkdown_2.14
uwot_0.1.11
DelayedMatrixStats_1.18.0
gtools_3.9.3
lifecycle_1.0.1
jsonlite_1.8.0
viridisLite_0.4.0
pillar_1.8.0
ggrastr_1.0.1
fastmap_1.1.0
survival_3.3-1
png_0.1-7
stringi_1.7.8
DESeq2_1.36.0
caTools_1.18.2
irlba_2.3.5
```

# Bibliography

- Chen, W., Zhao, Y., Chen, X., Yang, Z., Xu, X., Bi, Y., Chen, V., Li, J., Choi, H., Ernest, B., et al. (2020). A multicenter study benchmarking single-cell rna sequencing technologies using reference samples. *Nature Biotechnology*, pages 1–12.
- Delconte, R. B., Kolesnik, T. B., Dagley, L. F., Rautela, J., Shi, W., Putz, E. M., Stannard, K., Zhang, J.-G., Teh, C., Firth, M., et al. (2016). Cis is a potent checkpoint in nk cell-mediated tumor immunity. *Nature Immunology*, 17(7):816–824.
- R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.