Recreating ´Single-cell transcriptomics and epigenomics unravel the role of monocytes in neuroblastoma bone marrow metastasis'

Lucía Castelli

Contents

Introduction

What are the goals of this project?

This project serves as a training initiative in single-cell transcriptomics and epigenomics, an exercise in project management, and an educational tool for anyone interested. As part of this effort, I will reproduce the findings of the 2023 study titled "Single-cell transcriptomics and epigenomics unravel the role of monocytes in neuroblastoma bone marrow metastasis" by Fetahu *et al.*, which is largely aligned with my research interests. The abstract from the original article describes the project as follows:

> Metastasis is the major cause of cancer-related deaths. Neuroblastoma (NB), a childhood tumor has been molecularly defined at the primary cancer site, however, the bone marrow (BM) as the metastatic niche of NB is poorly characterized. Here we perform single-cell transcriptomic and epigenomic profiling of BM aspirates from 11 subjects spanning three major NB subtypes and compare these to five age-matched and metastasis-free BM, followed by in-depth single cell

> analyses of tissue diversity and cell-cell interactions, as well as functional validation. We show that cellular plasticity of NB tumor cells is conserved upon metastasis and tumor cell type composition is NB subtype-dependent. NB cells signal to the BM microenvironment, rewiring via macrophage mgration inhibitory factor and midkine signaling specifically monocytes, which exhibit M1 and M2 features, are marked by activation of pro- and anti-inflammatory programs, and express tumor-promoting factors, reminiscent of tumor-associated macrophages. The interactions and pathways characterized in our study provide the basis for therapeutic approaches that target tumor-to-microenvironment interactions.

The primary aim of this project educational, so I am committed to providing a thorough and accessible breakdown of the biological concepts, R code, and rationale behind each step. My goal is to present this information clearly and sequentially, using straightforward language to support understanding among fellow researchers and students. You can think of this project as both (1) my personal notebook as a student, and (2) the manual I would have liked to have as a student learning these concepts. Self-teaching myself these pipelines required many hours of work and dedication; I hope this helps someone in a similar situation as me when I started this project.

The entire project is also available as a downloadable pdf file here.

Disclaimer: Please note that this is an independent project, and any errors or misinterpretations are solely my own. For questions or feedback, feel free to contact me.

Credit where credit is due: a note on the original code and software versions

The code for performing the analyses and generating all figures from the original article has been made available at GitHub by the original researchers, and is used as a guide for this project. The pipeline I followed included parts of their pipeline, and parts of Daniel Beiting, *PhD.*'s, as shown in his online course, DIY.transcriptomics.

The results in this project may show slight differences from those reported in the original paper due to potential discrepancies in methodologies or in R or package versions. To ensure reproducibility, I will export the R project and use `renv` to capture the exact package environment. This will allow you to recreate my results using the same software versions I used.

Biological background

Neuroblastoma (NB) is a type of childhood cancer that arises from neuroblasts, immature cells of the sympathetic nervous system, typically originating in the adrenal medulla, the inner portion of the adrenal glands, right above the kidneys. NB has the potential to metastasize to the **bone marrow (BM)**, a progression associated with significantly poorer prognosis. It is the most common extracranial (outside the brain) solid (not in the blood, like leukemia or lymphoma) pediatric cancer, and accounts for 15% of cancer-associated childhood deaths. Neuroblasts, the cells where NB originates from, are pluripotent cells that migrate after arising in the neural crest, a transient structure of multipotent embryonic stem cells. When neuroblasts undergo differentiation, they might generating several types of cells and tissues, including NB cells.

NB tumors are very heterogeneous: different cell subpopulations present different prognostic markers and therapeutic targets, which explains different levels of response to treatment. More heterogeneity (inter- and intratumor) is caused by metastasis and adaptation to new tissue microenvironments. In particular, tumor cells that disseminate to the bone marrow (which happens on most metastatic cases even before diagnosis) have a substantially different genetic makeup and expression programs from the tumor they originated from.

NB patients are also very diverse, both genetically and in their clinical outcomes: some patients have spontaneous tumor regression, and some present malignant metastatic disease, relapses, and poor response to therapy. Because of this, patient stratification is important to identify high-risk patients (who have survival rates lower than 40%). Therefore, the aim of researchers tends to be understanding cancer cells and the tumor microenvironment (e.g. by molecular profiling) at the primary and metastatic sites, in order to identify

new biomarkers (e.g. in the blood, to monitor patients and detect relapses earlier), and develop targeted treatments.

Sources: https://pmc.ncbi.nlm.nih.gov/articles/PMC6558004/, https://ccri.at/research-group/sabine-taschner-mandl-group/, https://www.sciencedirect.com/topics/medicine-and-dentistry/neuroblast.

The study I will be recreating in this project focuses on 11 patients representing three major subtypes of NB with bone marrow (BM) metastasis, alongside 5 individuals without BM metastasis. The NB subtypes analyzed include:

1. **Amplified MYCN (MNA, n = 4)**: Amplification of this gene, transcriptional regulator associated to growth and development, is associated with a variety of tumors, most notably neuroblastomas.
2. **Mutated ATRX ($\mathbf{ATRX^{mut}}$, n = 2)**: The protein encoded by this gene contains an ATPase/helicase domain, and it is involved in transcriptional regulation and chromatin remodelation. Mutations are associated to telomere maintenance via recombination, as opposed to telomerase activity, in ALT (alternative lengthening of telomeres).
3. **Sporadic (n = 3)**: this subgroup is defined by large segmental chromosomal aberrations, instead of any small and recurrent somatic alteration.

What are the main contributions of the original research project?

This research poses and seeks to answer three questions:

1. **What are the differences in cellular plasticity in primary and metastastic tumors, across all NB subtypes?** The researchers conclude that NB tumors conserve their plasticity upon metastasis, meaning that they can access different genetic programs, allowing adaptation to new niches. Tumor cell type composition is NB subtype-dependent.
2. **What are the interactions between tumor cells and the metastastic BM microenvironment?:** The researchers conclude that NB tumor cells modify the BM microenvironment by signalling to monocytes, which express tumor-promoting factors and obtain either a M1 (pro-inflammatory) or M2 (anti-inflammatory) phenotype. This signalling, from NB cells to BM monocytes, occurs mainly via two molecules: *midkine* (MDK), a growth factor that promotes cell growth, migration, and angiogenesis, and the *macrophage migration inhibitory factor* (MIF), a lymphokine involved in cell-mediated immunity, immunoregulation, and inflammation.
3. **How is the BM altered by metastasis?** The BM microenvironment is altered in metastasis, significantly affecting the immune system. The consequence is the presence of pro- and anti-inflammatory monocytes that express tumor-promoting factors, similarly to tumor-associated macrophages.

In my view, the primary strength of this study lies in its application of single-cell resolution analysis:

- across distinct subtypes of neuroblastoma patients, rather than employing a generalized, unstratified approach, and
- within the bone marrow, the most common site of metastasis and a key determinant of poor prognosis, rather than the more frequently studied, but less lethal, primary site in the adrenal glands.
- to illuminate the ability cancer cells have to take on new traits, in order to understand BM metastasis better and treat it.

Data: download and description

- `Adrenal_medulla_Seurat.RDS`: reference expression data for adrenal medullary cells; downloaded from https://adrenal.kitz-heidelberg.de/developmental_programs_NB_viz/ (Download data -> Download Adrenal medulla data -> Seurat object (RDS))
- `GSE216155_RAW.tar`: main raw data used in the scRNA-seq experiment; downloaded from GEO Series GSE216155. Extracting this file (`$ tar -xvf GSE216155_RAW.tar`) should result in 51 new files.

```
ls ~/projects/neuroblastoma/data_raw/GSE216155/GSE216155_RAW
```

```
## GSM6659414_C1_filtered_feature_bc_matrix.h5
## GSM6659414_C1_metrics_summary.csv
## GSM6659414_C1_raw_feature_bc_matrix.h5
## GSM6659415_C2_filtered_feature_bc_matrix.h5
## GSM6659415_C2_metrics_summary.csv
## GSM6659415_C2_raw_feature_bc_matrix.h5
## GSM6659416_C3_filtered_feature_bc_matrix.h5
## GSM6659416_C3_metrics_summary.csv
## GSM6659416_C3_raw_feature_bc_matrix.h5
## GSM6659417_C4_filtered_feature_bc_matrix.h5
## GSM6659417_C4_metrics_summary.csv
## GSM6659417_C4_raw_feature_bc_matrix.h5
## GSM6659418_C5_filtered_feature_bc_matrix.h5
## GSM6659418_C5_metrics_summary.csv
## GSM6659418_C5_raw_feature_bc_matrix.h5
## GSM6659419_M1_filtered_feature_bc_matrix.h5
## GSM6659419_M1_metrics_summary.csv
## GSM6659419_M1_raw_feature_bc_matrix.h5
## GSM6659420_M2a_filtered_feature_bc_matrix.h5
## GSM6659420_M2a_metrics_summary.csv
## GSM6659420_M2a_raw_feature_bc_matrix.h5
## GSM6659421_M2b_filtered_feature_bc_matrix.h5
## GSM6659421_M2b_metrics_summary.csv
## GSM6659421_M2b_raw_feature_bc_matrix.h5
## GSM6659422_M3_filtered_feature_bc_matrix.h5
## GSM6659422_M3_metrics_summary.csv
## GSM6659422_M3_raw_feature_bc_matrix.h5
## GSM6659423_M4_filtered_feature_bc_matrix.h5
## GSM6659423_M4_metrics_summary.csv
## GSM6659423_M4_raw_feature_bc_matrix.h5
## GSM6659424_A1_filtered_feature_bc_matrix.h5
## GSM6659424_A1_metrics_summary.csv
## GSM6659424_A1_raw_feature_bc_matrix.h5
## GSM6659425_A2_filtered_feature_bc_matrix.h5
## GSM6659425_A2_metrics_summary.csv
## GSM6659425_A2_raw_feature_bc_matrix.h5
## GSM6659426_S1_filtered_feature_bc_matrix.h5
## GSM6659426_S1_metrics_summary.csv
## GSM6659426_S1_raw_feature_bc_matrix.h5
## GSM6659427_S2_filtered_feature_bc_matrix.h5
## GSM6659427_S2_metrics_summary.csv
## GSM6659427_S2_raw_feature_bc_matrix.h5
## GSM6659428_S3_filtered_feature_bc_matrix.h5
## GSM6659428_S3_metrics_summary.csv
## GSM6659428_S3_raw_feature_bc_matrix.h5
## GSM6659429_S4_filtered_feature_bc_matrix.h5
## GSM6659429_S4_metrics_summary.csv
## GSM6659429_S4_raw_feature_bc_matrix.h5
## GSM6659430_S5_filtered_feature_bc_matrix.h5
## GSM6659430_S5_metrics_summary.csv
## GSM6659430_S5_raw_feature_bc_matrix.h5
```

Notice how there are three files per sample:

- **Raw feature barcode matrix** (`*_raw_feature_bc_matrix.h5`): *"includes all valid barcodes from GEMs (Gel Bead-In EMulsions) captured in the data. However, because most GEMs do not actually contain cells, it follows that most barcodes in the data do not correspond to cells, but rather background noise (e.g. GEMs with free-floating mRNA from lysed or dead cells)* (source: 10X Genomics). The extension `.h5` means this file is in Hierarchical Data Format 5 (HDF5), made to store large amount of data.
- **Filtered feature barcode matrix** (`*_filtered_feature_bc_matrix.h5`): excludes barcodes that correspond to background noise (source: 10X Genomics). This should be used in the analysis.
- **Metrics summary** (`metrics_summary.csv.gz`): after extraction, shows a table with summarized quality metrix, the transposed version of which looks as follows for sample *C1*:

Estimated Number of Cells	12,417
Mean Reads per Cell	15,616
Median Genes per Cell	544
Number of Reads	193,915,006
Valid Barcodes	98.3%
Sequencing Saturation	46.6%
Q30 Bases in Barcode	96.0%
Q30 Bases in RNA Read	92.9%
Q30 Bases in Sample Index	94.5%
Q30 Bases in UMI	95.6%
Reads Mapped to Genome	95.1%
Reads Mapped Confidently to Genome	89.8%
Reads Mapped Confidently to Intergenic Regions	3.3%
Reads Mapped Confidently to Intronic Regions	27.6%
Reads Mapped Confidently to Exonic Regions	58.9%
Reads Mapped Confidently to Transcriptome	53.9%
Reads Mapped Antisense to Gene	2.3%
Fraction Reads in Cells	54.9%
Total Genes Detected	20,923
Median UMI Counts per Cell	1,035

scRNA-seq analysis

Load data

After downloading the data, the first thing to do is loading it into our R environment. I load libraries and metadata before that.

```r
# Load libraries
library(tidyverse)
library(GEOquery)
library(Seurat)
library(cowplot)
library(scds)
#library(UCell)
#library(fs)
#library(monocle3)
```

`tidyverse` is a collection of R packages for data manipulation, visualization, and analysis, built around consistent grammar and syntax; `GEOquery` enables access to gene expression and metadata directly from the NCBI Gene Expression Omnibus (GEO); `Seurat` is widely used for single-cell RNA-seq analysis, providing tools for data normalization, clustering, visualization, and integration; `cowplot` extends `ggplot2` to improve plots; `scds` provides methods for detecting and filtering doublets (artificial cell multiplets) in single-cell RNA-seq datasets.

```r
# Get metadata from list of ExpressionSets and generate study_design table
gse_ID <- "GSE216155"
directory = "~/projects/neuroblastoma/data_raw/GSE216155/"
gse <- getGEO(GEO = gse_ID, destdir = directory)
metadata <- t(Biobase::pData(gse[[1]])) # also: exprs (ematrix), fData (gene/probe anno)
study_design <- data.frame(row.names = colnames(metadata),
                           sample_code = metadata["sample id:ch1",],
                           phenotype = metadata["sample type:ch1",],
                           diagnosis = metadata["diagnosis:ch1",])
# sapply() returns a vector, & `[` is the name of the indexing function, e.g. x[1]:
study_design$phenotype <- sapply(str_split(study_design$phenotype, " "), FUN = `[`, 1)
saveRDS(object = study_design,
        file = "~/projects/neuroblastoma/data_generated/study_design.RDS")
```

```r
study_design
```

```
##            sample_code      phenotype            diagnosis
## GSM6659414          C1        Control ganglioneuroblastoma
## GSM6659415          C2        Control ganglioneuroblastoma
## GSM6659416          C3        Control       ganglioneuroma
## GSM6659417          C4        Control       ganglioneuroma
## GSM6659418          C5        Control       ganglioneuroma
## GSM6659419          M1 MYCN-amplified        neuroblastoma
## GSM6659420         M2a MYCN-amplified        neuroblastoma
## GSM6659421         M2b MYCN-amplified        neuroblastoma
## GSM6659422          M3 MYCN-amplified        neuroblastoma
## GSM6659423          M4 MYCN-amplified        neuroblastoma
## GSM6659424          A1   ATRX-deleted        neuroblastoma
## GSM6659425          A2   ATRX-deleted        neuroblastoma
## GSM6659426          S1       Sporadic        neuroblastoma
## GSM6659427          S2       Sporadic        neuroblastoma
## GSM6659428          S3       Sporadic        neuroblastoma
## GSM6659429          S4       Sporadic        neuroblastoma
## GSM6659430          S5       Sporadic        neuroblastoma
```

Now I will get the filtered .h5 files instead of the raw ones. I will loop through each filename to produce all Seurat objects and save them together in a list of Seurat objects with all my samples (`datasets`). In the expression matrix (`data`), the `rows` are the genes, the `cols` are the cells, the numbers (or dots, which

represent '0') are the amount of UMIs (so, the mRNA counts).

As opposed to bulk RNA-seq analyses, where you get to normalize data only after filtering it, when working in at a single-cell level this can be done directly after generating the Seurat object (according to Daniel Beiting in his course "DIY Transcriptomics"). `NormalizeData()` makes expression values comparable across cells despite their different total number of reads/UMIs, by performing library size normalization. `FindVariableFeatures()` identifies the most variable genes across cells, reducing dimensionality by focusing on the most informative genes.

```
path = "~/projects/neuroblastoma/data_raw/GSE216155/GSE216155_RAW/"
setwd(path)
filenames <- list.files(path = path)
filenames <- filenames[grepl("filtered", filenames)]
datasets <- list()

# Load filtered data for each sample
for (filename in filenames) {
  sample_id <- strsplit(filename, split = "_")[[1]][2]
  data <- Read10X_h5(filename)
  datasets[[sample_id]] <- CreateSeuratObject(counts = data,
                                              project = sample_id,
                                              min.cells = 3,
                                              min.features = 200) %>%
    NormalizeData(verbose = FALSE) %>%
    FindVariableFeatures(verbose = FALSE)

  # Clean up barcode names: remove default -1 and add sample suffix
  old_barcodes <- sub("-1$", "", colnames(datasets[[sample_id]]))
  colnames(datasets[[sample_id]]) <- paste0(old_barcodes, "-", sample_id)
  }
```

```
datasets$C1 # I access the sample labeled "C1"
```

```
## An object of class Seurat
## 17732 features across 11001 samples within 1 assay
## Active assay: RNA (17732 features, 2000 variable features)
##  2 layers present: counts, data
```

The data has been successfully loaded! The Seurat object is described with "Active assay: RNA", because it is a transcriptomics experiment; but it could just as well include a different type of experiment, such as *ATACseq* for chromatin accessibility.

Quality Control of scRNA-seq data.

The uploaded data has is not the raw sequencing data. This has been demultiplexed, aligned to a reference genome (GRCh38) and counted using Cell Ranger v3.0.2 (10x Genomics) by the researchers. We must perform QC over this data, a process described in the original article as follows:

> Only cells matching the following criteria (as calculated by Seurat v4.0.079) were included for downstream analyses: >200 and <500 features, as well as <10% reads mapped to mitochondrial genes. Moreover, doublets with a binary classification-based doublet score >0.8 (as calculated by scds v1.6.080) were discarded. Eventually, quality control filtering yielded a final dataset consisting of 80,789 cells. Cell-free mRNA contamination was removed via SoupX (v1.5.0).

Let us now do Quality control for the samples. First, I will calculate the percentage of detected genes per cell that are mitochondrial (`%MT`). This percentage should not be too high, as that would suggest cell damage: when a cell's membrane is compromised, cytoplasmic RNA (which is linear) is more likely to degrade or be

lost compared to the more stable mitochondrial RNA (which is circular). This results in a higher proportion of reads mapping to mitochondrial genes.

After this, and in order to detect outliers, the data can be employed to make the following plots per sample, where each dot represents a theoretical single cell:

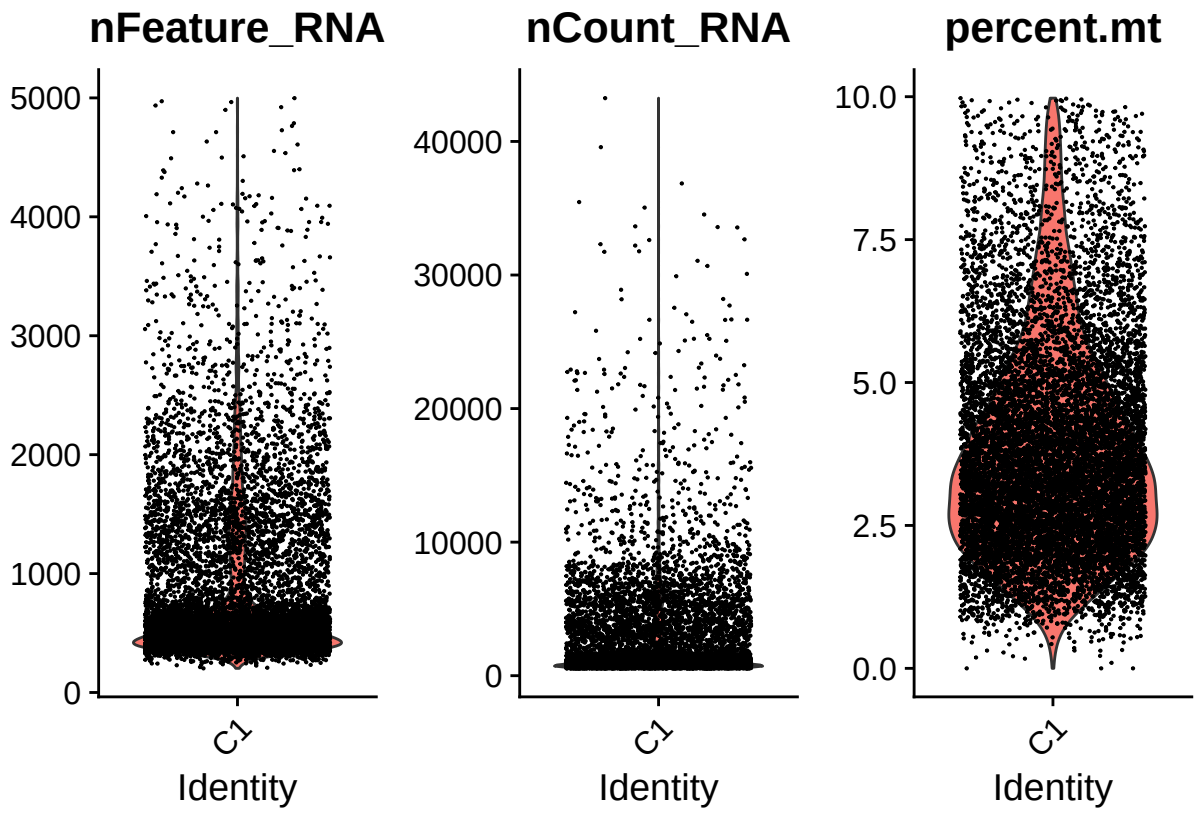
1. **Violin plots:** three violin plots to we analyze the `nCounts` (number of UMIs or individual RNA molecules), `nFeatures` (number of genes), and the `%MT` per cell. How can each of these three values help us determine low-quality cells?
   - `nCounts` (total UMIs per cell): a high number of RNA molecules could indicate the presence of doublets/multiplets, whereas a low number could represent the absence of a cell, and mRNA molecules from dying cells (ambient RNA) captured in a droplet instead.
   - `nFeatures` (total genes per cell): similarly to `nCounts`, a high number of genes could indicate doublets/multiplets, and a low number likely corresponds to dead/damaged cells.
   - `%MT` (percent mitochondrial genes): as explained above, a low percentage of mitochondrial genes is to be expected in a healthy cell.
2. **Scatter plot:** this is a graphic way to relate two variables to each other, with the `nCounts` in the *x axis* and `nFeatures` in the *y axis*. The result should look somewhat linear, so it can be useful to include a diagonal line to represent the general trend the data follows.

We can use these plots to 'eyeball' the cutoffs to filter our data. These cutoffs are somewhat arbitrary, and depend on your own experiment, data, and criterium. Adaptability is advised. For example, is the cell type you're looking for known to have little expression activity? Then you shouldn't filter those cells with low gene expression.

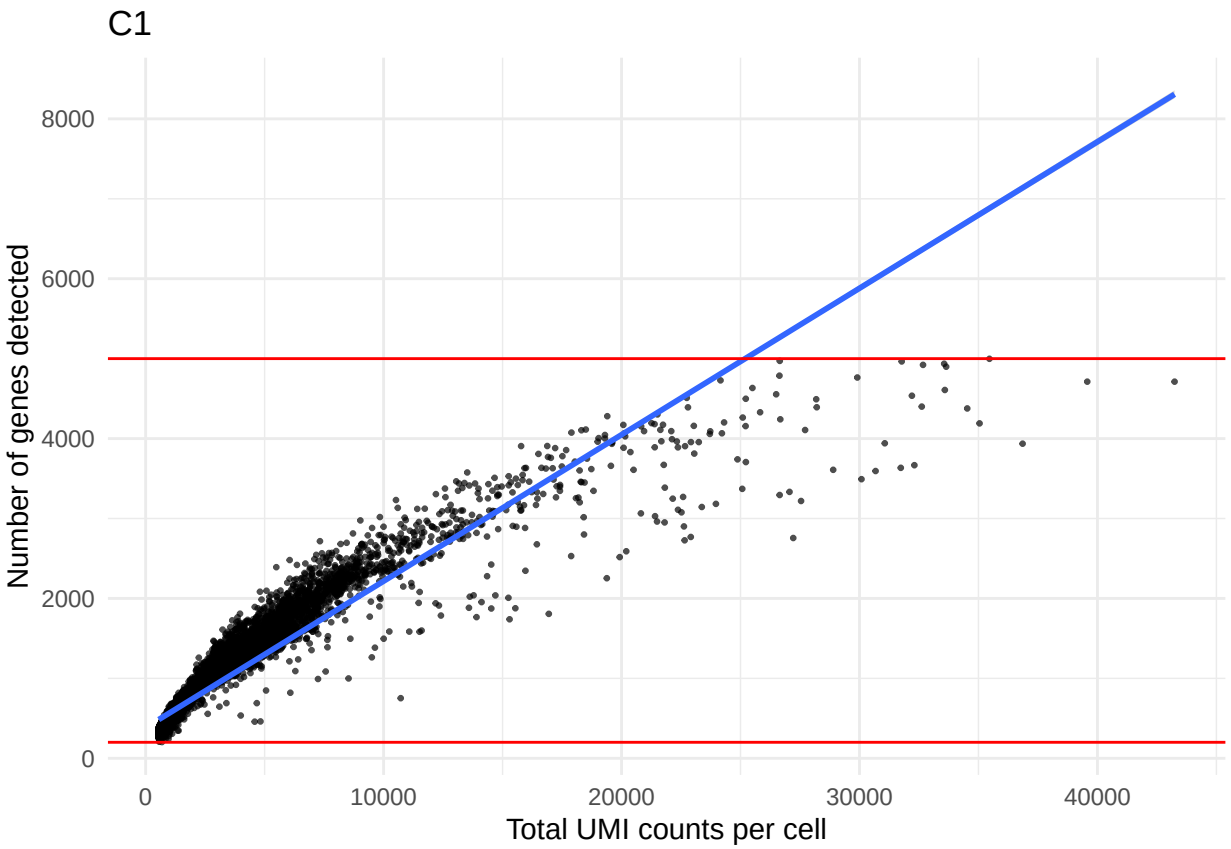
In this case, I will use the QC parameters used by the original researchers to include cells for downstream analyses. These include cells with less than 10% reads mapped to mitochondrial genes, and with more than 200 and less than 5000 features/genes (*note: paper says less than 500, but it must have been a typo, because the code from github uses 5000*). In my example scatter plot, I decided to include two red, horizontal lines, to show these cutoffs. Even though no filter is applied to the number of counts directly as such, an additional filter is calculated with the `bcds` ("Binary Classification Decision Score) function from the `scds` library, which detects doublets/multiplets using a binary classification approach to discriminate artificial doublets from original data: doublets/multiplets with a score > 0.8 were discarded.

```
violin_plots = list()
scatter_plots = list()

for (name in names(datasets)) {
  datasets[[name]][["percent.mt"]] <- PercentageFeatureSet(datasets[[name]],
                                                           pattern = "^MT-")

  violin_plots[[name]] <- VlnPlot(datasets[[name]],
                                  features = c("nFeature_RNA",
                                               "nCount_RNA",
                                               "percent.mt"),
                                  ncol = 3)

  scatter_plots[[name]] <- ggplot(datasets[[name]]@meta.data,
                                  aes(nCount_RNA, nFeature_RNA)) +
    geom_point(alpha = 0.7, size = 0.5) +
    labs(x = "Total UMI counts per cell",
         y = "Number of genes detected",
         title = name) +
    geom_smooth(method = "lm", formula = y ~ x) +
    geom_hline(yintercept = 200, color = "red") +
```

```
    geom_hline(yintercept = 5000, color = "red") +
    theme_minimal()
  }

# plot_grid(plotlist = scatter_plots, nrow = 3) # this would plot all samples!
selection <- c("C1")
violin_plots[[selection]]
```




```
scatter_plots[[selection]]
```



Now you could skip a lot of the previous code by running:

After filtering, you might want to plot the QC details again to verify your cutoffs and potentially iterate until you are satisfied. The researchers mentioned that after quality control filtering, the final dataset included a total of 80,789 cells. Let us verify if out number is close to that:

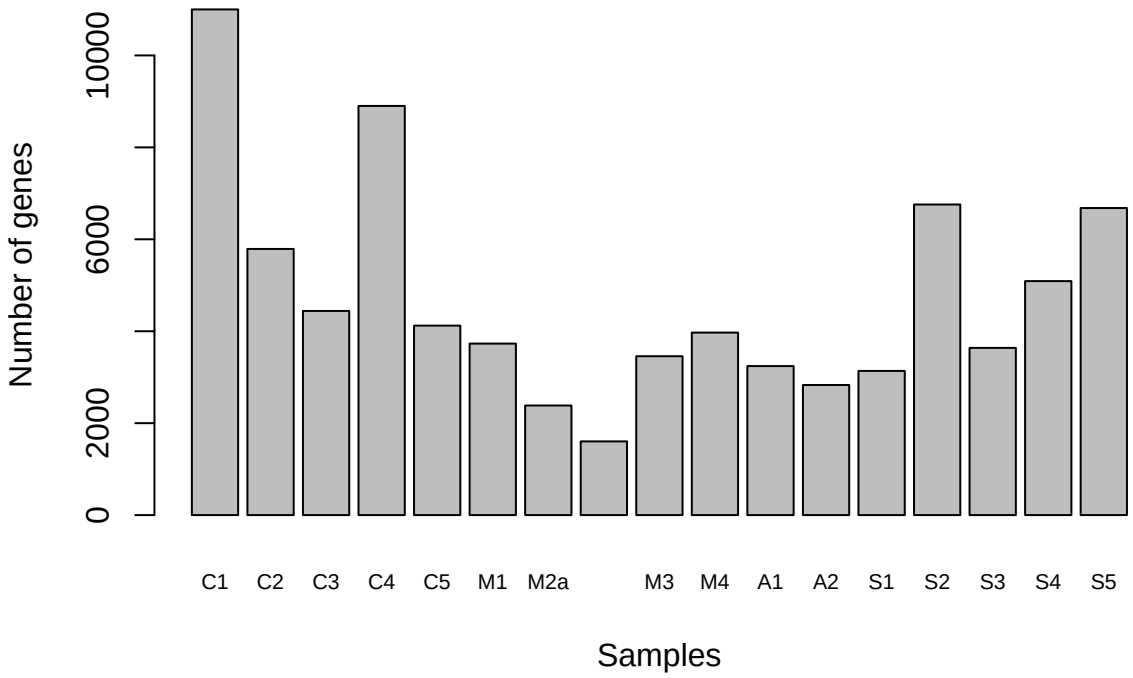
```
# Number of genes
sapply(datasets, ncol)
```

```
##    C1    C2    C3    C4    C5    M1   M2a   M2b    M3    M4    A1    A2    S1
## 11001  5790  4441  8901  4122  3731  2384  1603  3456  3970  3242  2830  3136
##    S2    S3    S4    S5
##  6757  3637  5090  6680
```

```
sum(sapply(datasets, ncol))
```

```
## [1] 80771
```

```
barplot(sapply(datasets, ncol), cex.names = .7, ylab = "Number of genes", xlab = "Samples")
```



The final QC step is removing cell-free mRNA contamination with SoupX, which will be done after sample integration.

`TODO`: There is clearly something up with the M2 samples, which correspond to amplified MYCN expression (n_patients = 4, but n_samples = 5). This might mean M2a and M2b, which have low numbers of genes, together make the M2 sample. Is this due to a quality issue?

Integrate scRNA-seq samples with Seurat, perform basic analyses and extract resulting metadata

I was unable to use Monocle3, for now. But I am interested in performing this analysis with Seurat, as taught by Daniel Beiting in DIYTranscriptomics, his online course.

Right now, our Seurat objects are completely blind to our experimental design. We can inform every object to what phenotype they belong, and this information will be added in the object's `@meta.data` as a new category, called `phenotype`.

```
# Right now Seurat object is blind to phenotype, so I will add data from study_design
for (name in names(datasets)) {
  datasets[[name]]$phenotype <- study_design %>%
    filter(sample_code == name) %>%
    pull(phenotype)
}

# Save data
saveRDS(object = datasets,
        file = "~/projects/neuroblastoma/data_generated/RNA_filtered.RDS")
```

```
# Now you could skip a lot of the previous code by running:
# datasets <- readRDS("~/projects/neuroblastoma/data_generated/RNA_filtered.RDS") # :)
```

To access the phenotype:

```
unique(datasets$C1$phenotype)
```

```
## [1] "Control"
```

```
unique(datasets$M3$phenotype)
```

```
## [1] "MYCN-amplified"
```

Now I want to integrate all of the samples into a single Seurat object. When you have multiple scRNA-seq datasets (e.g., different patients, or experimental conditions), they each have their own technical noise and batch effects. If you just "merge" them, you risk clustering by batch instead of biology. Integration aims to align the datasets into a shared space by correcting for these batch effects while preserving biological differences.

The integration will be done using three functions: 1. `SelectIntegrationFeatures()`: this finds genes that are variable across all samples. These are the most informative ones to align the datasets. It's different from `FindVariableFeatures()`, which works within a single dataset. 2. `FindIntegrationAnchors()`: this uses the genes we just selected to detect pairs of cells between datasets (samples) that are similar to each other. This function does the heavy lifting — it can take hours to run. I recommend saving the output as a RDS just in case, after it runs. 3. `IntegrateData()`: This uses the anchors to adjust expression values so that cells from all different samples are comparable.

```
anchor_features <- SelectIntegrationFeatures(object.list = datasets)
anchors <- FindIntegrationAnchors(object.list = datasets, # can take hours!
                                  anchor.features = anchor_features)
integrated <- IntegrateData(anchorset = anchors)
saveRDS(object = integrated,
        file = "~/projects/neuroblastoma/data_generated/integrated.RDS")
```

Now, the "active assay" is not "RNA", but "integrated". This can be easily changed, which is useful, for example, when working with multi-omics data:

```
# Before:
integrated
```

```
## An object of class Seurat
## 26072 features across 80771 samples within 2 assays
## Active assay: integrated (2000 features, 2000 variable features)
##  2 layers present: data, scale.data
##  1 other assay present: RNA
##  2 dimensional reductions calculated: pca, umap
```

```
DefaultAssay(integrated) <- "RNA"

# After:
integrated
```

```
## An object of class Seurat
## 26072 features across 80771 samples within 2 assays
## Active assay: RNA (24072 features, 2000 variable features)
##  3 layers present: scale.data, data, counts
##  1 other assay present: integrated
##  2 dimensional reductions calculated: pca, umap
```

Now you can apply dimensionality reduction as if you were working with a single sample.

Dimensional reduction and clustering of scRNA-seq data

The original researchers describe this step as follows:

> Raw counts were log- and size factor-normalized and scaled, followed by dimensional reduction via principal component analysis, as implemented in monocle3 (v0.2.3.0). Principal components were subsequently used as input for batch correction by matching mutual nearest neighbors, employing the function 'reducedMNN' in batchelor (v1.6.2). The Uniform Manifold Approximation and Projection (UMAP) method for dimensional reduction (uwot, v0.1.10) was applied to the resulting batch-corrected principal component scores. Finally, Leiden clustering was performed on UMAP coordinates. These steps were conducted via the monocle3 functions 'align_cds,', 'reduce_dimension,', and 'cluster_cells.'

However, I will be employing Seurat to do the same things. These are the steps:

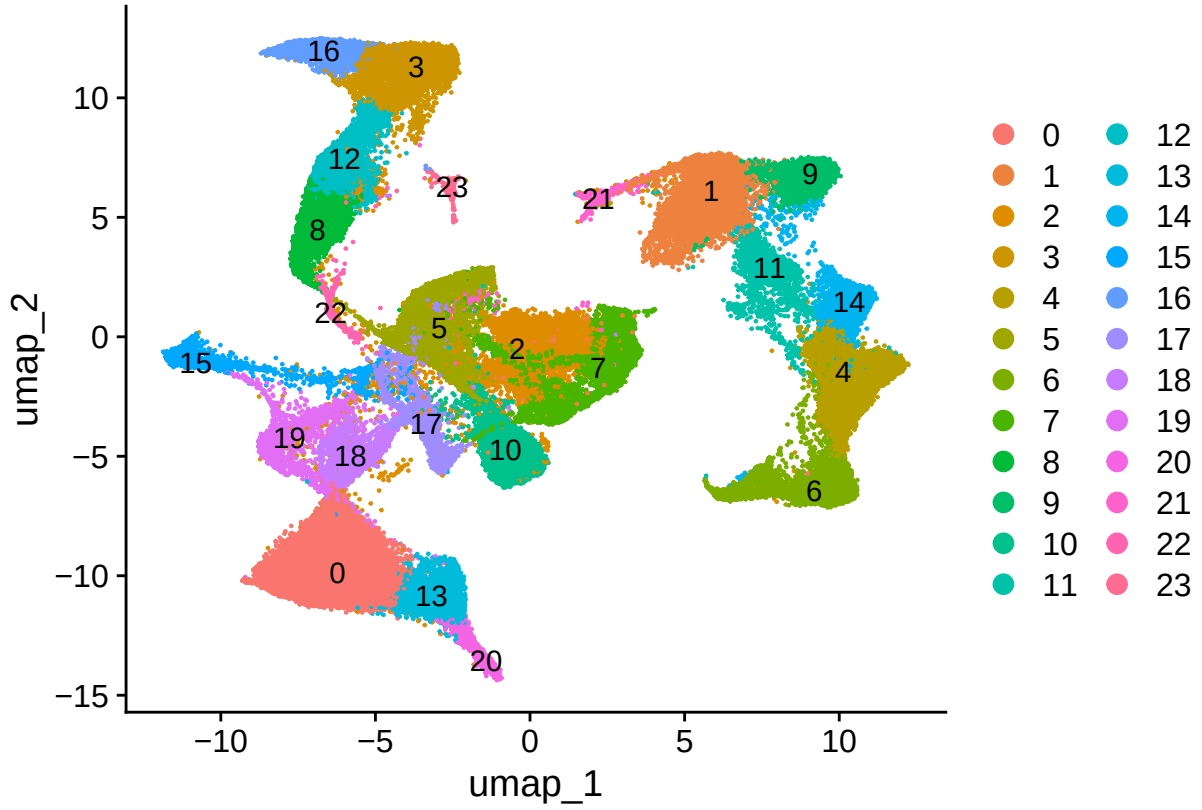
1. **Normalize**: Log-normalization of raw counts, typically to a fixed scale factor per cell. In Seurat, the function is `NormalizeData()`. In Monocle, that's `log1p(counts(cds))` or `preprocess_cds()`.
2. **Feature selection**: Identify the most variable genes for downstream dimensional reduction. In Seurat, the function is `FindVariableFeatures()`. In Monocle, that's `log1p(counts(cds))` or `preprocess_cds(method = 'PCA')`, internal.

(With the Seurat pipeline, these two were done at the beginning, immediately after loading the datasets individually. They must not be run again.)

3. **Scale Data**: Center and scale gene expression values per gene. In Seurat, the function is `ScaleData()`. In Monocle, that's `preprocess_cds()`.
4. **Batch Correction**: Correct for batch effects across samples using mutual nearest neighbors. In Seurat, the function is `RunFastMNN() [SeuratWrappers]`. In Monocle, that's `reducedMNN() [batchelor]`.
5. **Dimensionality Reduction (PCA)**: Reduce dimensionality using PCA on corrected/scaled data. In Seurat, the function is `RunPCA()`. In Monocle, that's `reduce_dimension(reduction_method = 'PCA')`.
6. **Embedding (UMAP)**: Visualize cells in 2D/3D space using UMAP on corrected PCs. In Seurat, the function is `RunUMAP()`. In Monocle, that's `reduce_dimension(reduction_method = 'UMAP')`.
7. **Clustering**: Cluster cells based on nearest neighbors (Leiden or Louvain). In Seurat, the function is `FindNeighbors() + FindClusters(algorithm = 4)`. In Monocle, that's `cluster_cells() [Leiden algorithm]`.

```
integrated <- ScaleData(integrated, verbose = FALSE)
integrated <- RunPCA(integrated, npcs = 30, verbose = FALSE)
integrated <- RunUMAP(integrated, reduction = "pca", dims = 1:30)
integrated <- FindNeighbors(integrated, reduction = "pca", dims = 1:30)
integrated <- FindClusters(integrated, resolution = 0.5, algorithm = 4)
```

```
DimPlot(integrated, reduction = "umap", label = TRUE)
```



After this, we can make a table to see what proportion of the total cells reside in each cluster. `Idents(integrated)` returns a vector of cell identities (in this case, the clusters for each cell), `table()`counts how many cells fall into each identity, and `prop.table()` converts raw counts into proportions (relative frequencies).

```
# Check what proportion of our total cells reside in each cluster:
prop.table(table(Idents(integrated)))
```

```
##
##           0           1           2           3           4           5
## 0.109507125 0.105223409 0.084064825 0.072142229 0.068601354 0.056715901
##           6           7           8           9          10          11
## 0.052605514 0.042106697 0.041735276 0.039259140 0.037934407 0.035557316
##          12          13          14          15          16          17
## 0.035247799 0.035148754 0.028450805 0.028438425 0.025838482 0.025714675
##          18          19          20          21          22          23
## 0.023882334 0.022966164 0.013148283 0.007527454 0.004593233 0.003590398
```

Integrate scRNA-seq samples with monocle, perform basic analyses and extract resulting metadata

`TODO`: Might remove this part, or make CDS with integrated Seurat object.

Right now, our Seurat objects are completely ignorant of our experimental design. We can inform every object to what phenotype or treatment they belong, and this information will be added in the object's `@meta.data`.

The library *monocle3* employs objects of a class called `cell_data_set`, which is derived from the Bioconductor `SingleCellExperiment` class. We need to convert our Seurat objects this class in order to work with *monocle3*,

providing the `@counts` and `@meta.data` of each one. This can be done with `map()`, to apply a function to each Seurat object in the list, which is more straightforward than the `for`loops I used in the code so far, but both do the job just fine! Then we can combine all the `cell_data_sets` into one in order to analyze them together, using the function `combine_cds()`.

TODO

Dimensional reduction and clustering of scRNA-seq data

> Raw counts were log- and size factor-normalized and scaled, followed by dimensional reduction via principal component analysis, as implemented in monocle3 (v0.2.3.0). Principal components were subsequently used as input for batch correction by matching mutual nearest neighbors, employing the function 'reducedMNN' in batchelor (v1.6.2). The Uniform Manifold Approximation and Projection (UMAP) method for dimensional reduction (uwot, v0.1.10) was applied to the resulting batch-corrected principal component scores. Finally, Leiden clustering was performed on UMAP coordinates. These steps were conducted via the monocle3 functions 'align_cds,', 'reduce_dimension,', and 'cluster_cells.'

TODO

Correct ambience: remove cell-free RNA contamination

TODO

Perform cell type classification via SingleR

TODO

Assemble metadata

TODO

Analyze differential gene expression using mixed models

TODO

Analyze copy number variations

TODO

Analyze cell-cell communication

TODO

Analyze myeloid subpopulation

TODO

Prepare dataset by Dong *et al* for analysis

TODO

Classify tumor cells as adrenal medullary cell types

TODO

Comparison of tumor samples via pseudobulk correlation

`TODO`

To be continued...