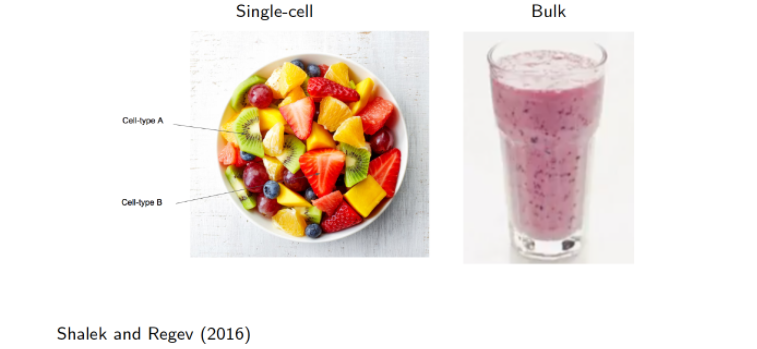
***[Single-Cell RNA-Seq with Bioconductor in R](https://www.datacamp.com/courses/single-cell-rna-seq-with-bioconductor-in-r/continue)***

# ***What is Single Cell RNA-Seq, and why is it useful?***

## **1. What is single-cell RNA-Seq, and why is it useful?**

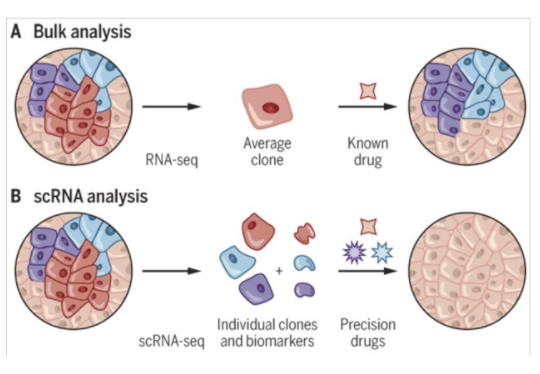
My name is Fanny Perraudeau and I'm your instructor for this course on single-cell RNA-Sequencing workflows. In this course, you'll understand what single-cell RNA-sequencing means and why it is useful for many applications in biology and medicine. You'll also learn how to analyze single cell data in R. And this first chapter is an introduction for the rest of the course.

## **2. Milkshake or fruit salad?.**



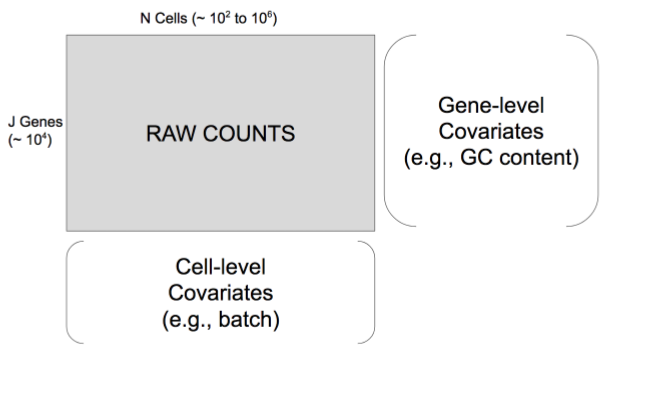
Today it is possible to obtain genome-wide transcriptome data from single cells using high-throughput sequencing. It is what we call single-cell RNA-Sequencing or scRNA-Seq. The main advantage of single-cell RNA-Sequencing is that it allows researchers to measure gene expression levels at the resolution of single cells. In that sense, single-cell RNA-Sequencing is analogous to a bowl of fruits, where each piece of fruit is a cell whose type can be identified. Why is it amazing? It is because the cellular resolution and the genome-wide scope offers the unprecedented opportunity to investigate at the cellular level fundamental biological questions, such as stem cell differentiation or the discovery and characterization of rare cell types. It makes it possible to address issues that are intractable using other methods, e.g. bulk RNA-sequencing. Using bulk sequencing, which is the technology that was developed before single-cell sequencing, you get an averaged gene expression profile of all the cells in the sample. Continuing the fruit analogy, bulk RNA-Seq could be viewed as a smoothie, where you sample a mixture of blended fruits which gives you an average signal of all cells in the sample. So, using bulk RNA-Sequencing, it is not possible to identify the different cell-types within one sample.

## **3. scRNA-Seq could revolutionize personalized medicine in cancer**



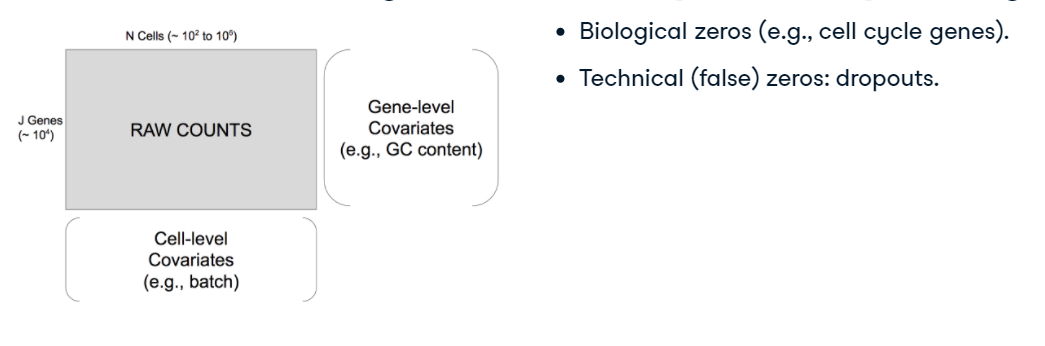
Why is single-cell RNA-Sequencing a revolution in biology? Well, it's because it has plenty of applications, especially in cancer, microbiology, and neurology. For example, in personalized medicine in cancer, it could enable researchers to identify individual clones and biomarkers in a tumor, and select precision drugs for each of them. This is not possible using bulk RNA-sequencing where you get an average gene expression profile of all the cells in the tumor. It's what's represented in the figure here where red, blue, and purple cells have been identified as different cell types and could be targeted separately by different drugs when single-cell RNA-sequencing is used.

## **4. Data structure**



With single-cell RNA-Seq, the data you get out from the lab after the preprocessing is this big matrix where you have the genes as the rows and the cells as the columns. Inside the matrix you have counts corresponding to the number of reads aligned to each gene and each cell where a read is a sequence of nucleotides (A,T,C,G). You also get two other matrices corresponding to the cell-level and gene-level covariates. For the gene-level covariates, you for example have the length of the genes or the GC content which is the percentage of nucleotides G and C compared to nucleotides A and T. For the cell-level covariates, you could have quality control measures of the cells, for example, the batches in which the cells have been sequenced.

## **5. Zero inflation in single-cell transcriptome sequencing**



Now, in the matrix of counts, there are many more zeros than in bulk RNA-sequencing. The zeros can be biological when a gene is simply not expressed in a cell. For example, genes involved in the division of the cell are not expressed at each step of the cell cycle. The zeros can also be technical when the sequencing machine fails to sequence reads from a gene and a cell. In that case, you observe a zero in the count matrix instead of an actual count. It's what people call dropouts.

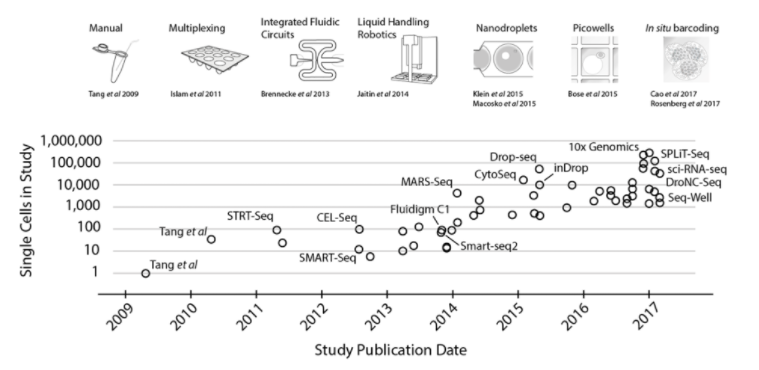
## **1. Typical workflow**

In this video, we are going to go over the typical steps to analyze single-cell RNA-seq data.

## **2. Exponential scaling of scRNA-Seq in the last decade**

The development of new methods and protocols for scRNA-Seq is currently a very active area of research, and several protocols have been published over the last few years. The image here taken from Svensson et al paper shows on the y-axis that the number of cells per dataset has increased from 1 cell for the very first dataset in 2009 to up to 1 million cells for datasets generated today.

1. 1 "Exponential scaling of single cell RNAseq in the last decade". Valentine Svensson, Roser Vento
2. 2 Tormo, Sarah A Teichmann



## **3. Aspects of scRNE-Seq methods**

The methods can be categorized in different ways, but the two most important aspects are quantification and capture. For quantification, there are two types of technologies, full-length and tag-based. The full-length protocols try to achieve a uniform coverage of each RNA sequence. By contrast, tag-based technologies only capture either the two ends of each RNA. The choice of quantification method has important implications for what types of analyses the data can be used for. Then, the strategy used for capture mostly determines throughput. The three most widely used options are microwell-, microfluidic- and droplet-based. For more details about the different technologies, you can go to the Hemberg lab website, it's the reference number 2 at the bottom of this slide and a great reference for analyzing scRNA-Seq.

## **4. scRNE-seq workflow**

After this brief overview of the different technologies, let's now get an overview of the different steps of a typical workflow to analyze single-cell RNA-seq. Each of these steps is actually a chapter of the course, so we won't go into details here, but just look at the big picture.

## **5. First step: quality control**

The very first step when working with scRNA-Seq data is to filter out low-quality cells to ensure that technical effects do not distort downstream analysis results. Two common measures of cell quality are the library size and cell coverage. The library size is defined as the total sum of counts across all genes, where here the word "library" refers to a cell. And the cell coverage is defined as the average number of genes with non-zero counts for that cell.

## **6. Typical workflow**

Once the problematic cells have been removed, a typical workflow to analyze scRNA-Seq data includes several steps. The first step is the normalization of cell- and gene- specific biases. It is a critical step in the analysis pipeline that adjusts for unwanted biological and technical effects that can mask the biological signal of interest.

1. 1 "Bioconductor workflow for single
2. 2 cell RNA sequencing". Perraudeau F, Risso D, Street K et al

## **7. Typical workflow**

Then, the large majority of scRNA-Seq analyses include a dimensionality reduction step where the number of dimensions goes from J (that is the number of genes) to K which is smaller than J. This step achieves a two-fold objectives: first the data become more tractable, and second noise can be reduced while preserving the signal of interest.

1. 1 "Bioconductor workflow for single
2. 2 cell RNA sequencing". Perraudeau F, Risso D, Street K et al

## **8. Typical workflow**

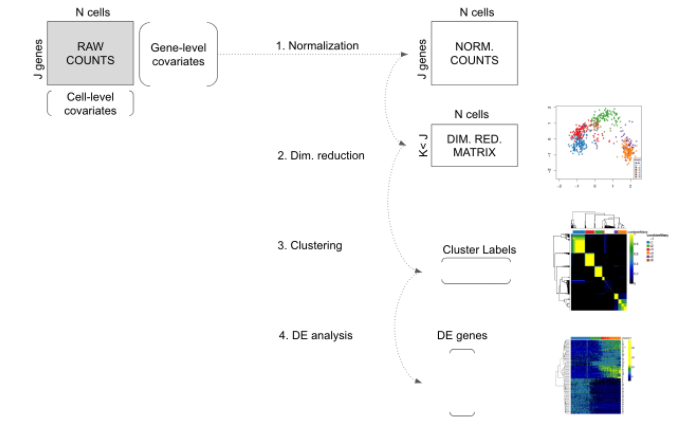
The third step, is to group the cells according to the low-dimensional matrix K by N computed in the previous step where N is the total number of cells in the dataset. From this step, you get a cluster label for each cell.

1. 1 "Bioconductor workflow for single
2. 2 cell RNA sequencing". Perraudeau F, Risso D, Street K et al

## **9. Typical workflow**

Finally, the last step is to find biomarkers between identified groups of cells, that is, find genes that are differentially expressed between groups of cells. This gives you an overview of the typical workflow used to analyze scRNA-Seq data.

1. 1 "Bioconductor workflow for single
2. 2 cell RNA sequencing". Perraudeau F, Risso D, Street K et al



# **GC content**

The GC content (or guanine-cytosine content) is the percentage of bases on a DNA or RNA molecule that are either guanine (G) or cytosine (C) out of four possible bases. In addition to guanine (G) and cytosine (C), the other bases are adenine (A), and thymine (T) in DNA or uracil (U) in RNA. We'll see later in the course that GC content could be a bias in scRNA-Seq.

# **Nesting between batches and biology**

In scRNA-Seq data analysis, we want to detect the biological signal of interest (e.g., difference of gene expression in cells from different patients) but remove the technical signals (e.g., difference of gene expression in cells sequenced from different technical batches). When biological (e.g., patients) and technical (e.g., batches) variables are confounded, it could be problematic because the two signals could be difficult to disentangle.

# **1. Load, create, and access single-cell datasets in R**

Let's learn how to load, create, and access single-cell data in R.

# **2. SingleCellExperiment class**

In this video, we focus on the SingleCellExperiment class. It’s a S4 class developed by Aaron Lun and Davide Risso. And it’s very useful to analyze single-cell data because it allows you to easily store and retrieve the matrix of counts but also information about the cells and the genes. You remember the three matrices (one with the raw counts, and the other two for gene and cell-level information) I showed in the previous video, the idea here is that we are going to use only one R object to store these three matrices.

1. 1 https://bioconductor.org/packages/3.9/bioc/html/SingleCellExperiment.html (by Aaron Lun and Davide Risso)

## **3. Load and install**

How does it work in practice? First, you need to install and load the SingleCellExperiment package using the usual biocLite and library functions.

## **4. SCE object from a counts matrix**

The next step is to create a SingleCellExperiment object. And, there are two ways to do this. The first way is to use the SingleCellExperiment function. For that, we'll create a small matrix counts with 4 genes and 2 cells where we simulate counts from a Poisson distribution using the rpois() function. Let's use the rownames() and colnames() functions to name the rows as the gene names and the columns as the cell names. Here, you see that gene names are Lamp5, Fam19a1, Cnr1, and RORB and cell names start with SRR and then a number.

## **5. SCE object from a counts matrix**

Using the matrix counts we just created, we create a sce object using the SingleCellExperiment() function. The first argument is assays which takes a list of count matrices. Here we'll only use our matrix counts with 4 genes and 2 cells. The second and third arguments are rowData and colData for data frames with information about the genes and cells. Here we only store the gene and cell names which are the row and column names of our matrix counts. You see that we have created the sce object with a dimension of 4 by 2 and you can see the gene and cell names.

## **6. SCE object from SummarizedExperiment**

The other way to create a SingleCellExperiment object is by transforming an existing SummarizedExperiment object which is commonly used when working with bulk RNA-Seq data. If you are not familiar with this class, you can look at the package documentation. We first need to create a SummarizedExperiment object using the same count matrix as in the previous slide and then use the `as()` function to transform the SummarizedExperiment into the SingleCellExperiment object. You see that the new object SCE is exactly the same as the SCE object we created in the previous slide.

1. 1 SummarizedExperiment package: https://bioconductor.org/packages/3.9/bioc/html/SummarizedExperiment.html

## **7. Real single-cell dataset**

We've created a fake dataset, but now let’s look at a real dataset. We can load the allen dataset from the package scRNA-Seq using the `data()` function. The dataset is a subset of the data published in Tasic et al in 2016, it contains about 400 cells from the visual cortex in male mice where cells were extracted from different layers of neurons. The dataset allen is a SummarizedExperiment object.

1. 1 Tasic et al "Adult mouse cortical cell taxonomy revealed by single cell transcriptomics"

## **8. Real single-cell dataset**

So as in the previous slides, we can create a SingleCellExperiment object from the object allen using the as() function and argument SingleCellExperiment. In the sce object, you can see the gene and cell names. And we also have access to cell quality measures. For example, the total number of reads aligned for each cell in the column naligned of the colData dataframe.

# **SCE object from counts matrix**

The SingleCellExperiment class is very useful for analyzing single-cell data sets. It defines a S4 class for storing data from single-cell experiments. This includes specialized methods to store and retrieve spike-in information, dimensionality reduction coordinates, and size factors for each cell, along with the usual metadata for genes and libraries. In this exercise, you will create a SingleCellExperiment object from a count matrix.

## **1. Quality Control**

In chapter 2, we go over the first steps of the workflow to analyze single-cell RNA-seq data, namely quality control and normalization. These two steps should get all the technical issues and biases out of the way so that in the next chapters we can focus on the biological signal of interest. In this video, we focus specifically on quality control with the goal to identify and remove bad quality cells and genes from our dataset. It is essential to remove these genes and cells since keeping them could bias our analysis. We explore different metrics and criteria to identify problematic genes and cells.

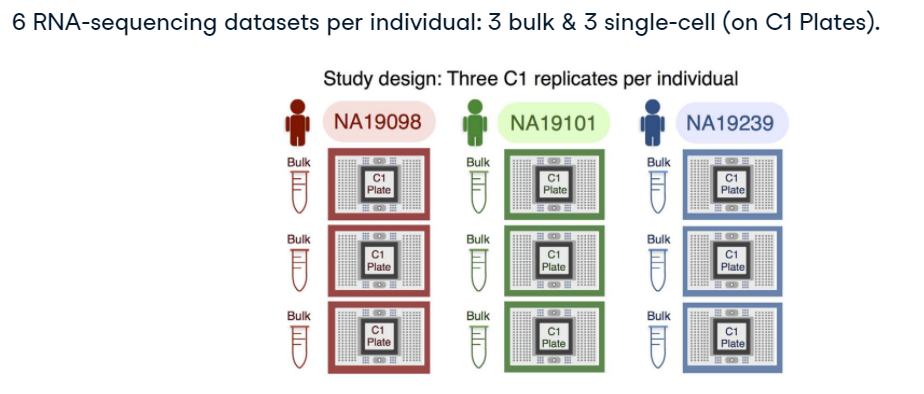
## **2. Tung dataset**

To illustrate quality control, we use a real dataset of induced pluripotent stem cells generated in Yoav Gilad’s lab. Samples were collected from 3 different individuals in different colors on the figure (red, green, blue). For each individual, 6 datasets were created: 3 using bulk and 3 using single-cell RNA-sequencing, and each dataset was sequenced in a different batch. You can get more details of the experiment in the referenced paper at the bottom of the slide.

1. 1 Batch effects and the effective design of single
2. 2 cell gene expression studies. Tung et al. Figure 1a.

## **3. Tung dataset**

The dataset generated by the experiment described in the previous slide has been stored as a SingleCellExperiment object called sce. You can see on the second line the dimensions of the count matrix. There are about 20000 genes and 900 cells. As in our previous examples, the line with the rownames shows the gene names starting with ENSG or ERCC and the line with the colnames shows the cell names starting with NA19. You also see that the colData matrix has columns with labels for the individual, replicate, well, batch, and sample\_id.



## **4. Calculate quality control measures**

In this dataset, we want to identify and filter out problematic genes and cells. To do so, let's calculate some quality control metrics using the package scater which is a useful package containing tools to analyze single-cell data. From the package scater, we use the function calculateQCMetrics() where the first argument should be an object of class sce. This function has many arguments and I encourage you to look at the documentation. In this example, we use the argument feature\_controls that takes a named list containing one or more vectors used to identify gene positive controls. Here, we use ERCC spike-in genes as positive control. ERCC spike-in genes are synthetic RNAs added to the sample in known quantity and they are used to filter out cells with small amount of endogenous RNAs by looking at the ratio between synthetic spike-in RNAs and endogenous RNAs. The rationale is the following: cells with a high level of spike-in RNAs had low starting amounts of RNA likely due to the cell being dead or stressed, so we want to filter out these cells with high ratio of synthetic spike-in RNAs compared to endogenous RNAs.

1. 1 Quality control with scater (Single
2. 2 Cell Analysis Toolkit for Gene Expression Data in R): https://bioconductor.org/packages/3.9/bioc/vignettes/scater/inst/doc/vignette
3. 3 qc.html

## **5. Functions used in exercises**

I'll stop here and let you play in the exercises where you'll see the following functions calculateQCMetrics() to calculate quality control measures. counts() of a SingleCellExperiment object to get the count matrix. rowSums() to calculate the sum across columns of a matrix. grepl() to get the items of a vector with a particular pattern, for example to get gene names starting with ERCC. isSpike() to identify spike-in genes. plot(density(x)) to plot the distribution of the x where x could for example be the library sizes. abline() to add a line to a plot.

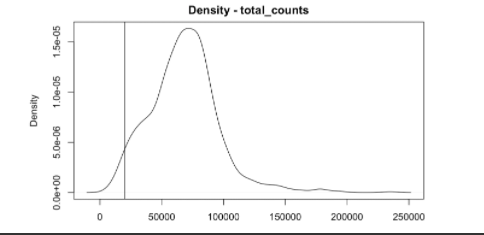
## **1. Quality Control (continued)**

## **2. Calculate quality control measures**

In the previous video, I showed you how to compute quality control metrics using the function calculateQCMetrics(). We now use these metrics to filter out bad quality cells and genes.

## **3. Cell filtering - Library size**

The first metrics we can look at is the library size. You remember from the previous chapter that the library size is the total number of reads for each cell. In the package scater, they use the term totalcounts instead of library size, but totalcounts and library size mean exactly the same. Our goal here is to remove cells with few reads because they are likely to have been lysed.



## **4. Cell filtering - Library size**

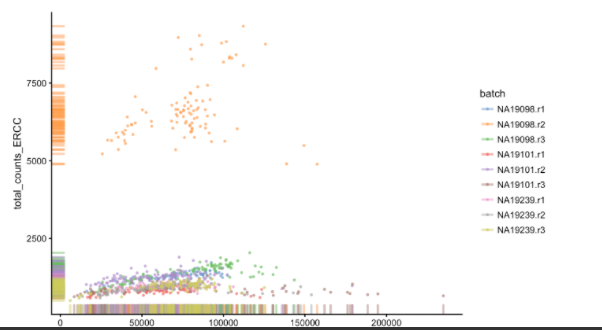
To do so, we can look at the distribution of the library size using functions plot and density and choose a threshold to filter out cells with library sizes smaller than our selected threshold. To plot the threshold on the graph, we use the function abline. The choice of the threshold depends on the dataset and should be data driven. Here a threshold of 20000 reads per cells seems reasonable since it does not remove too many cells but still removes cells likely to have been broken.

## **5. Cell filtering - Library size**

We can now use the function table to count the number of cells with a total count greater than our threshold. Here with a threshold of 20000, we remove 27 and keep about 900 cells.

## **6. Cell filtering - Batch**

The next metric we can look at is the quantity of ERCC versus endogenous RNA. To do so, we use the function plotPhenoData from the package scater. We plot the total counts on the x-axis compared to the total\_counts when we subset the dataset to keep only the ERCC genes. Each point on the graph is for a cell and cells are colored by batch.



## **7. Cell filtering - Batch**

You see that most of the orange cells from the batch NA19098.r2 have higher ratios between their total counts for ERCC and endogenous RNAs. In the Tung et al paper we talked about in the previous video, the authors have actually identified cells of smaller size in this batch. As these cells are very different from the other cells and have small quantity of endogenous RNAs, we want to remove these cells for the next steps of the analysis.

## **8. Cell filtering - Batch**

## **Again, we can use the functio**n table to count the number of cells not in batch NA19098.r2 that we keep for the analysis.

## **9. Gene filtering**

Finally, the next step is to remove genes that are mainly not expressed. The definition of "a gene is mainly not expressed" depends on the sequencing depth, but a reasonable criterium is usually to keep genes with counts of at least 2 in at least 2 cells. Note that gene filtering must be performed after cell filtering since some genes may only be detected in bad quality cells. Here, we would filter out about 4000 genes.

## **Filter cells by number of expressed genes**

To remove problematic cells, we want to filter out cells with small total number of genes with non-zero count (called total\_features in the package scater).

## **Use of positive controls**

To remove problematic cells, we can use the ratio between synthetic spike-in RNAs and endogenous RNAs. ERCC spike-in genes are synthetic RNAs added to the sample in known quantity that are used to filter out cells with small amount of endogenous RNAs. The rationale is that cells with a high level of spike-in RNAs had low starting amounts of RNA likely because they were stressed or dead, so we should filter them out.

## **Batch effect**

To observe the batch effect, we want to plot the first two principal components of the counts using plotPCA() from scater. Batch effects are common technical artifacts in single-cell data, as it means cells sequenced in the same batch are more similar than cells sequenced in different batches.

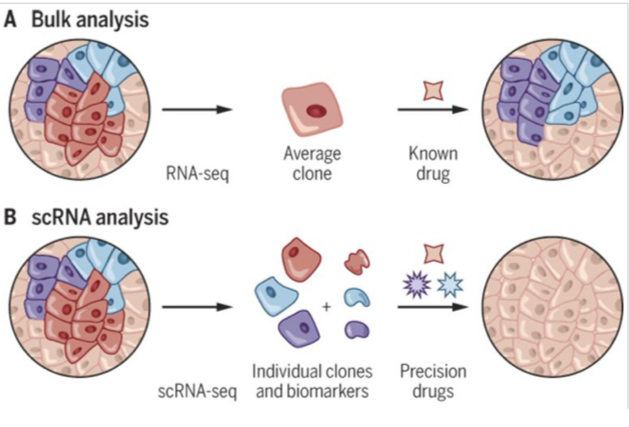
Batch effect is unwanted because we would rather detect the biological signal of interest (e.g., group cells according to their gene expression) than technical variation (e.g., batch).

## **1. Normalization**

In this video, you'll learn what normalization means and why it is needed when analyzing single-cell RNA-seq data.

## **2. Biological and technical variation**

## **When analyzing single-cell data, the goal** is to capture the biological signal of interest. For example, we want to be able to group the cells according to their gene expression profiles. Back to the example from the first chapter, we might want to group cells with different biomarkers to target each group of cells with different drugs. The challenge with single-cell data is that there are many technical artifacts that introduce variation in the data and mask the biological variation we want to observe.



## **3. Batch effect**

One of the common technical confounders is what people call the batch effect. On this slide, you see the Tung dataset we used in the previous video projected in two dimensions using Principal Component Analysis or PCA. You'll learn more about PCA in the next chapter but for the moment, let's say cells have been projected in two dimensions to capture the highest variation in the data. Each point on the plot is a cell, cells are colored by batch and shaped by individual. What you can see here is that cells are grouped by individual and batch. It's not surprising that cells cluster by individual because it means we capture an actual biological signal. However, it is unexpected that the cells cluster by batch because cells sequenced in different batches don't have different gene expression profiles, they've just been technically processed differently. So, here we just detect an unwanted technical artifact.

## **4. Goal of normalization**

So now you understand that the goal of normalization is to remove the unwanted technical variation (for example the batch effect) while preserving the biological signal of interest.

## **5. Normalization methods**

There are many other technical biases in single-cell data and you'll see some of them in the coding exercises. But for the moment let's just focus on how to correct for technical artifacts. Many normalization methods have been developed and here is just a short list of the most popular. The most straightforward way to normalize single-cell data is to divide each column of the matrix of counts by a normalization factor. This factor can simply be the library size or the library size divided by 1,000,000 to get counts per million or CPM. There are many variants of that: for example RPKM or TPM. Other methods divide each counts by a different scaling factor. Methods differ in the way they compute the scaling factors. The weighted Trimmed Mean of M-values or TMM is proposed in edgeR, the size factors proposed by DESeq is the geometric mean of each gene across all cells and in scran the size factors account for zero inflation. It's just a brief introduction to normalization and I encourage you to read more about it. The fourth reference at the bottom of the slide is a good start.

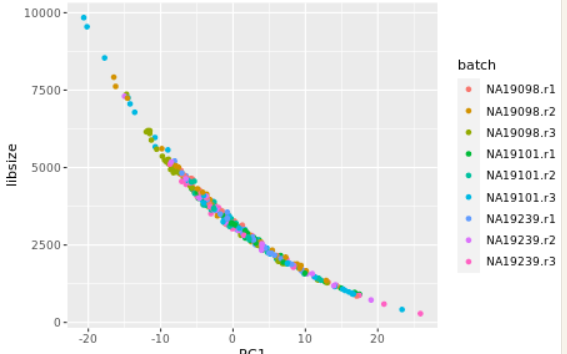
1. 1 "Normalizing single
2. 2 cell RNA sequencing data Challenges and opportunities" (Vallejos et al 2017)

## **6. Functions used in exercises**

In the exercises, you'll use the following functions plotPCA() to plot the first two principal components. reducedDim(sce, “PCA”)[, 1:2] to get the values of the first two principal components of the matrix of counts. computeSumFactors() and sizeFactors() to calculate and get the size factors. assays() to get the names of the matrices stored in a SingleCellExperiment object. Be careful here, function assays with an S et the end gives you the name of the assays whereas the function assay without an S at the end gives you the first assay of an sce object. normalize() to normalize counts of a sce object. And finally, a way to visualize if the normalization worked is to plot the relative log expression using function plotRLE().

## **Correlation between PC1 and library size**

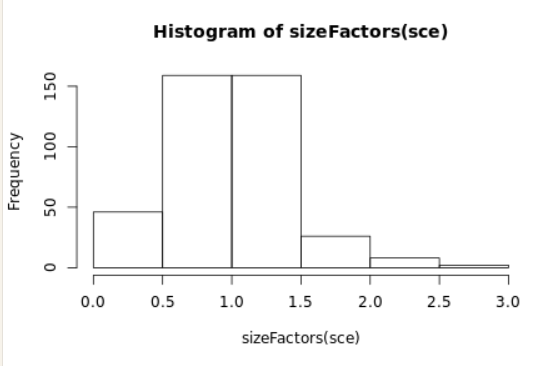
The first principal component captures the highest variation in the data and is often correlated with the library size, which means that the highest variation in the data is technical (total number of reads sequenced for each cell) instead of biological.



You now understand that there is a high correlation between PC1 and the library sizes. It means that the highest variation (PC1) in the data is technical (i.e., the total number of reads sequenced per cells) and not biological. It is the reason why we need to normalize the data: we want the highest variation in the data to be biological.

## **Compute size factors**

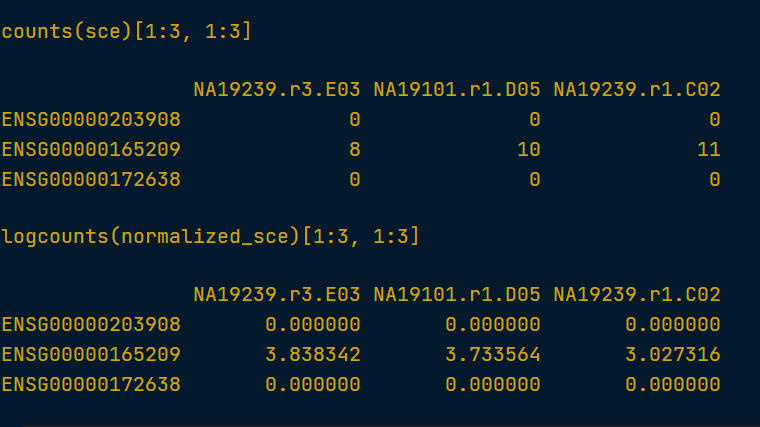
To remove technical artifacts, we want to compute size factors using the package scran.



You've computed size factors that you are going to use in the next exercise to normalize the data.

## **Normalize SCE object**

Recall that normalization can help us to remove technical artifacts. In this exercise, you will compute normalized log counts using the normalization method normalize() from the package scater.



You've performed normalization on the sce object to remove as much of technical variation as possible: the new normalized SCE object contains a new assay logcounts, containing normalized logcounts. In the next chapter, you'll visualize the data in two dimensions using dimensionality reduction techniques.

#### Visualization and Dimensionality Reduction

When studying single-cell data at the cellular level, the number of dimensions is the number of genes. The goal of dimensionality reduction is to reduce the number of dimensions to a smaller number either to visualize the data in 2 dimensions or to prepare the dataset for subsequent steps like clustering.

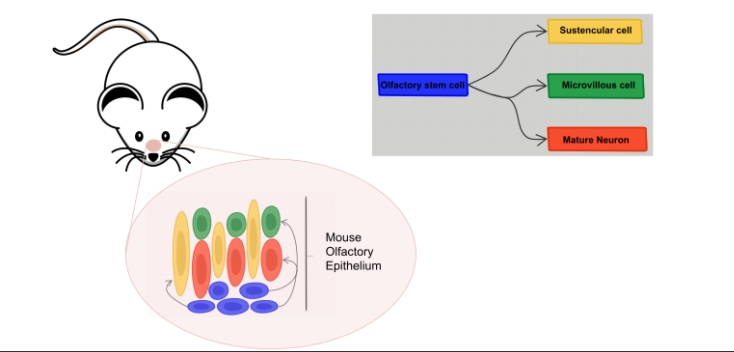
## **1. Mouse Epithelium Dataset**

We are now in Chapter 3 on visualization and dimensionality reduction. In scRNA-Seq, the number of dimensions is the number of genes. The goal of dimensionality reduction is to reduce the number of dimensions to a smaller number either to visualize the data in 2 or 3 dimensions or to prepare the dataset for subsequent steps like clustering.

## **2. Typical workflow**

## **Before we dive into the** details, let's take a step back and look again at our typical workflow from chapter 1. In the previous chapter, you learned about normalization which is the first step. Here, you'll learn about how to perform step 2 which is dimensionality reduction.

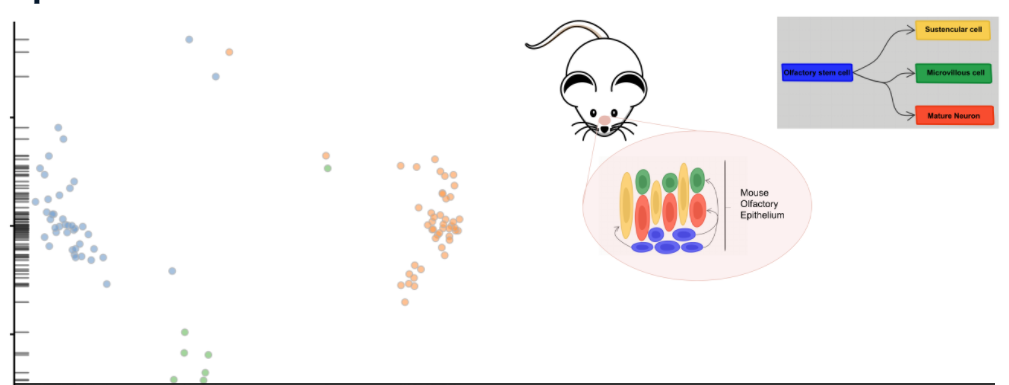
## **3. Stem Cell Differentiation in the Mouse Olfactory Epithelium**



To illustrate dimensionality reduction, we use a scRNA-Seq study of stem cell differentiation in the mouse olfactory epithelium. The olfactory epithelium contains mature sensory neurons, in orange here, that are continuously renewed in the epithelium through the differentiation of olfactory cells which are the actively proliferating cells in the epithelium. Now when a severe injury to the entire tissue happens, the olfactory epithelium can regenerate from stem cells, which become activated to differentiate and reconstitute all major cell types in the epithelium. This scRNA-Seq dataset was generated to study the differentiation of the stem cells shown in blue into the different cell types present in the olfactory epithelium. To map the developmental trajectories of the multiple cell lineages arising from the stem cells, cells were assigned to different lineages using an analysis pipeline similar to our typical workflow from the previous slide. Details on data generation and statistical methods are available in the referenced paper. As shown in the figure, it was found that the first major bifurcation in the stem cell lineage trajectory produce either mature sustentacular cells in yellow or mature neurons in orange and microvillus cells in green.

1. 1 Cell Stem Cell, Fletcher et al, Deconstructing Olfactory Stem Cell Trajectories at Single
2. 2 Cell Resolution (2017)

## **4. Stem Cell Differentiation in the Mouse Olfactory Epithelium**



In scRNA-seq analysis, dimensionality reduction is often used as a preliminary step prior to downstream analyses, such as clustering. The number of dimensions of a single cell dataset is the number of genes since it is the number of variables used to characterize the gene expression profile of a particular cell. Dimensionality reduction allows to reduce this number of dimensions to a smaller number than the number of genes. The advantages are that, first, the data becomes more tractable and, second, technical noise can be reduced while preserving the signal of interest. On the left side, you see the result of dimensionality reduction performed on a subset of the mouse epithelium dataset reducing the number of dimensions to two and visualizing the first two reduced dimensions. Each point represents a cell and cells are color coded by cell types inferred in the referenced paper for three cell types.

## **1. Visualization**

In the last video, I introduced the mouse epithelium dataset. In this video, we visualize this dataset using dimensionality reduction.

## **2. Dimensionality reduction**

In a single-cell dataset, each observation is a cell. For each cell, we observe its gene expression profile, meaning for each gene we know if it has a low or high number of detected RNAs which is a proxy for the gene expression. So, the number of dimensions for each cell is the number of genes. In the dimensionality reduction step of the single-cell workflow, we want to reduce the number of dimensions from the number of genes J to a smaller number, usually 2 if you want to visualize the dataset in 2 dimensions, like on the left side of this slide, or higher, like 50, if the goal is then to perform clustering.

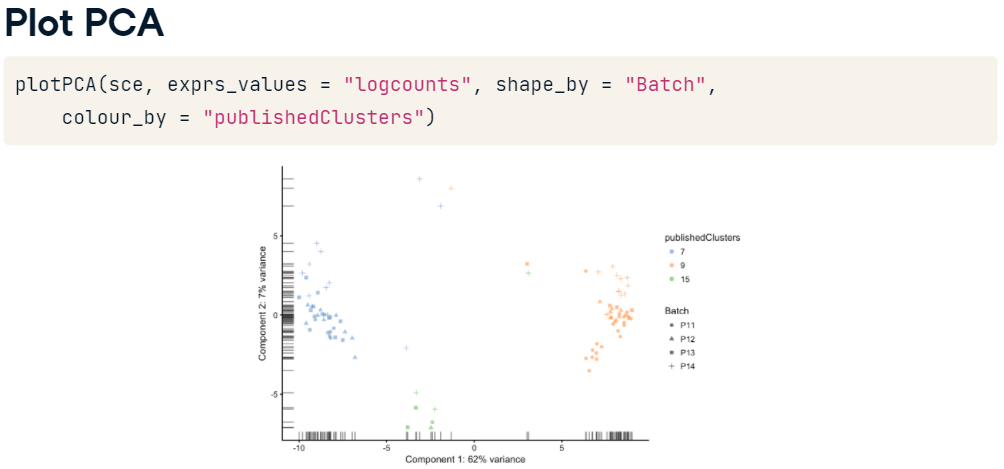
1. 1 Cell Stem Cell, Fletcher et al, Deconstructing Olfactory Stem Cell Trajectories at Single
2. 2 Cell Resolution (2017)

## **3. Dimensionality reduction methods**

Several methods can be used and here a few of them are listed. Principal component analysis (PCA) is probably the most popular method. It is a statistical procedure that uses a transformation to convert a set of observations into a set of values of linearly uncorrelated variables called principal components (PCs). The first PC accounts for as much of the variability in the data as possible, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. An alternative to PCA is tSNE which stands for t-Distributed Stochastic Neighbor Embedding. It combines dimensionality reduction (e.g. PCA) with random walks on the nearest-neighbor network to map high dimensional data to a 2-dimensional space while preserving local distances between cells. In addition to PCA and tSNE, other methods have specifically been designed for single-cell data like zifa and zinbwave. We'll look closer at the zinbwave method in the next lesson.

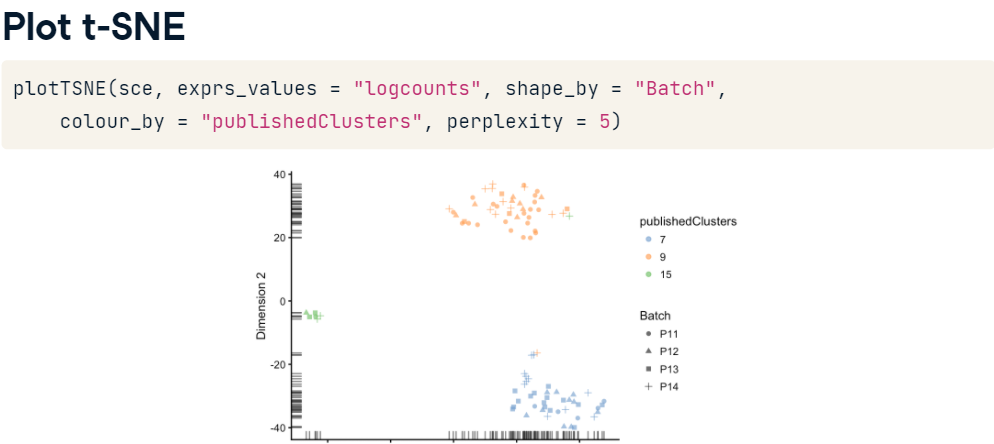
1. 1 ZIFA, Dimensionality reduction for zero
2. 2 inflated single
3. 3 cell gene expression analysis (Emma Pierson and Christopher Yau) Genome Biology
4. 4 A general and flexible method for signal extraction from single
5. 5 cell RNA
6. 6 seq data (Risso et al) Nature Communications

## **4. Plot PCA**

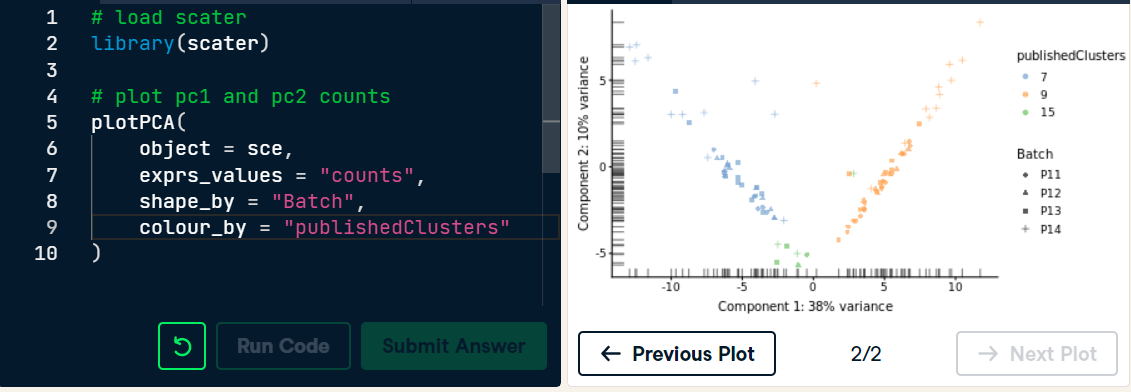


Before diving into dimensionality reduction theory and exploring the dimensionality reduction methods specific to scRNA-Seq, let's get familiar with PCA and t-SNE. The function plotPCA from the package scater can be used to look at the first and second principal components of a SingleCellExperiment object. Dimensionality reduction is usually performed on the log counts instead of the counts to reduce the bias towards highly expressed genes since higher counts could dominate the variation levels between the cells. To use the log of the counts, you'll use logcounts as expression values. On the plot here, cells were color coded by publishedClusters and shaped by batch.

## **5. Plot t-SNE**



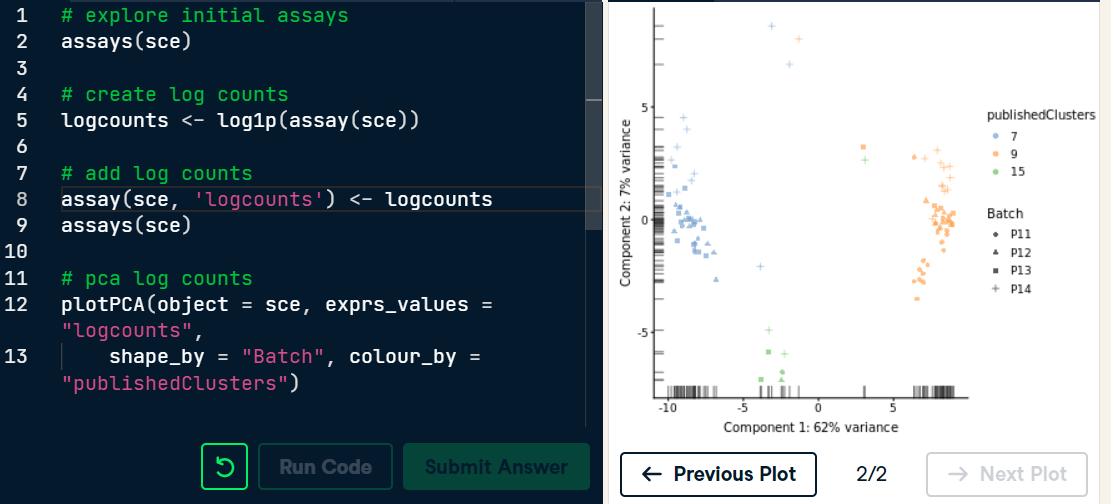
In the scater package, a similar function exists to plot tSNE. As you can see on this slide the call to the function plotTSNE is very similar to the one for the function plotPCA. There is an additional parameter called perplexity which is, in a sense, a guess about the number of close neighbors each point has. You'll be able to play with this parameter in the exercises.



Plotting PC1 and PC2 allows to take a first look at a scRNA-Seq dataset in two dimensions. Here you performed PCA on the raw counts. Usually, people prefer performing PCA on the log counts instead of the counts to reduce the bias towards highly expressed genes.

## **Plot PCA of log counts**

Dimensionality reduction is usually performed on the log counts instead of the counts to reduce the bias towards highly expressed genes (higher counts dominate the variation levels between the cells).

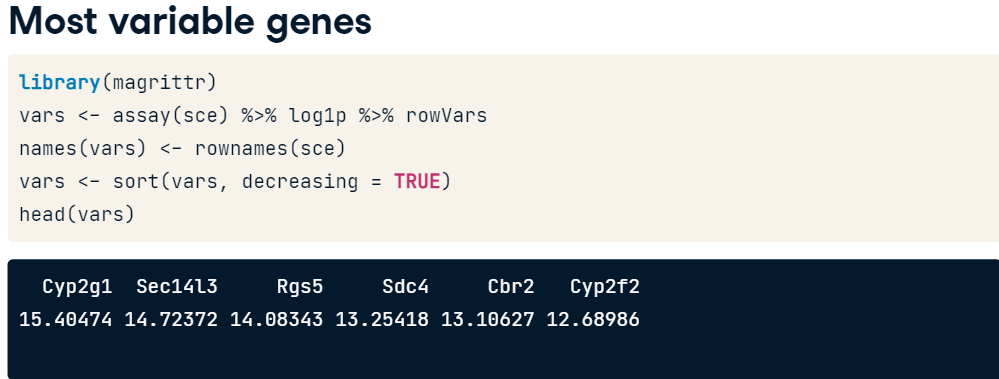


You can now see why it is preferable to perform PCA on the log counts than on the counts.

## **1. Dimensionality Reduction**

The goal of this video is to perform dimensionality reduction using PCA and base R functions.

## **2. Most variable genes**



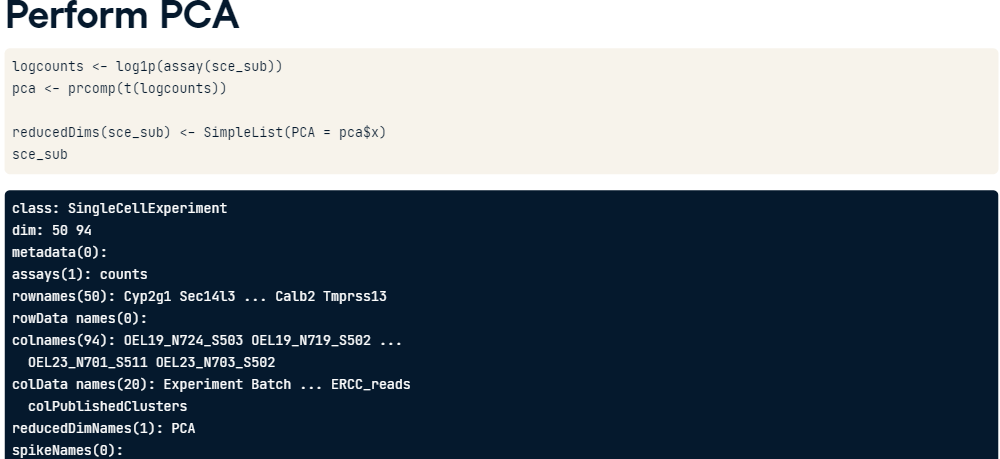
First, we want to subset the dataset and use only the most variable genes for computational efficiency. To do so, we use the package magrittr which provides a mechanism for chaining commands with a forward-pipe operator, % > %. This operator will forward a value, or the result of an expression, into the next function call. Here we use the matrix of counts of the object sce using the function assay, then take the log of the counts plus one using the function log1p and finally compute the variance for each gene using the function rowVars which calculates the variance across the rows of a matrix. We assign the result of this chain of operations to the variable vars. We then rename the vector vars with the same name as the genes and sort the genes according to their variance in a decreasing order. Looking at the first values of the variable vars, you can see the 6 most variable genes.

## **3. Subset sce**



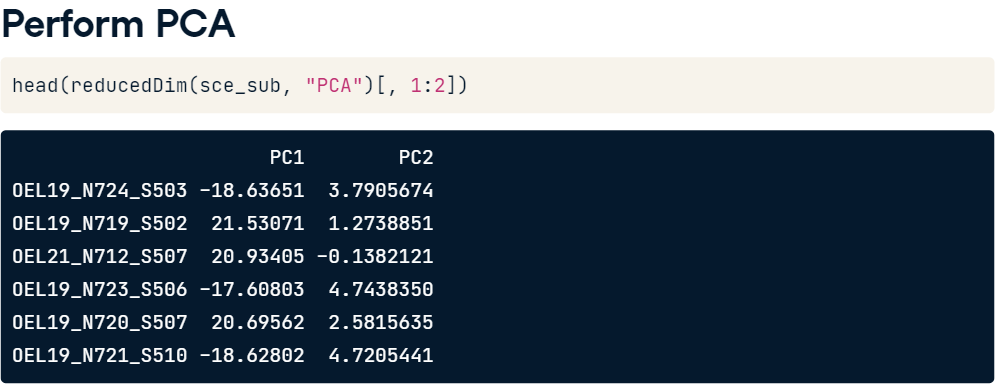
For computational efficiency, we can now subset the dataset to keep only the 50 most variable genes. You see now that the sce\_sub object has 50 genes and the same number of cells as before.

## **4. Perform PCA**



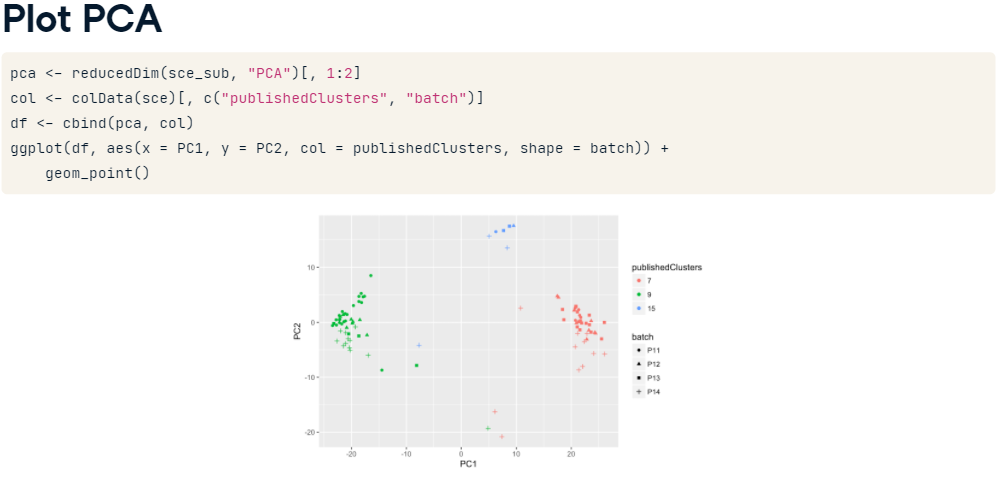
The next step is to perform dimensionality reduction using PCA. For that we use the prcomp function from base R. The PCA needs to be performed on the log of the counts so that the dimensionality reduction is not biased by the high count values. And here we need to use the transpose of the matrix logcounts to get the cells which are the observations here as the rows. The result is assigned to the variable pca. We can finally store the pca coordinates as a simple list into the sce\_sub object using the function reducedDims. You can see that the slot reducedDimNames of sce\_sub now contains the PCA coordinates.

## **5. Perform PCA**

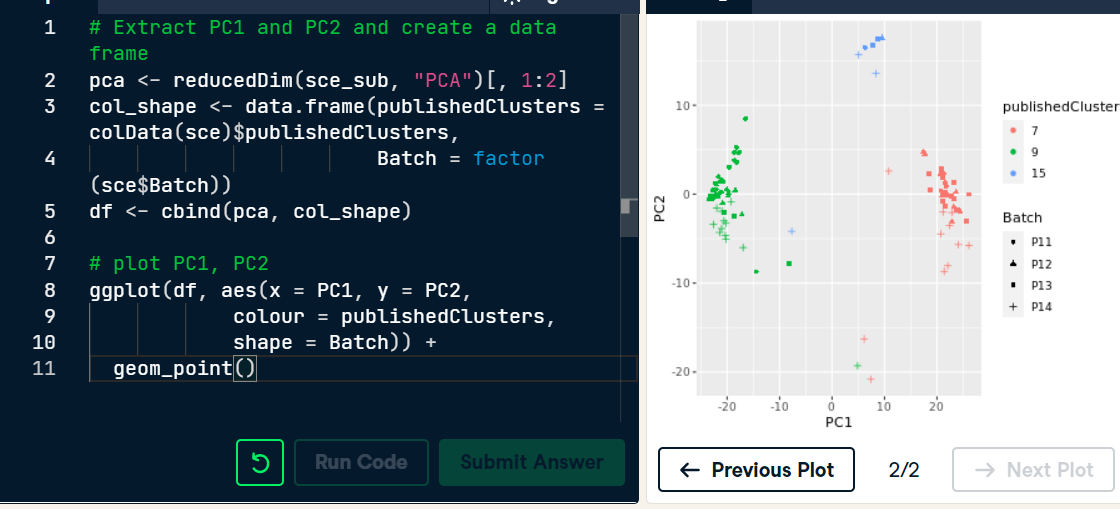


As shown on this slide, you can use the reduceDim function of sce\_sub to access the PCA coordinates.

## **6. Plot PCA**



Finally, you can plot principal components 1 and 2 of the sce\_sub object using the ggplot function. Each point on the graph represents a cell, cells are color-coded by clusters as published in the mouse epithelium paper and shaped by batch.



You've performed PCA and visualized a single cell data set in two dimensions. It is a great way to explore a single cell dataset.

## **1. Clustering methods for scRNA-Seq**

In this chapter, you are going to perform clustering and differential expression analysis. And in this lesson, we focus on clustering where the goal is to group cells with similar gene expression profiles.

## **2. Mouse epithelium dataset**

We'll use the same dataset as in the previous chapter, that is the mouse epithelium dataset. If you don't remember the details of this dataset, you can listen again to video one of chapter 3. In the plot on the left, each point represents a cell and cells have been color coded by the clusters found in the published paper on the mouse epithelium dataset referenced at the bottom of the slide. But, when you get the data from the lab, we don't know which cell is from which cell type or cluster. The goal of the clustering step is to group together cells with similar gene expression profiles. Once cells have been grouped by clusters, it is then easier to find a pattern in the gene expression profile of the cells from the same cluster. One question of interest for example could be to find the cell type of a particular cluster.

1. 1 Cell Stem Cell, Fletcher et al, Deconstructing Olfactory Stem Cell Trajectories at Single
2. 2 Cell Resolution

## **3. Clustering methods**

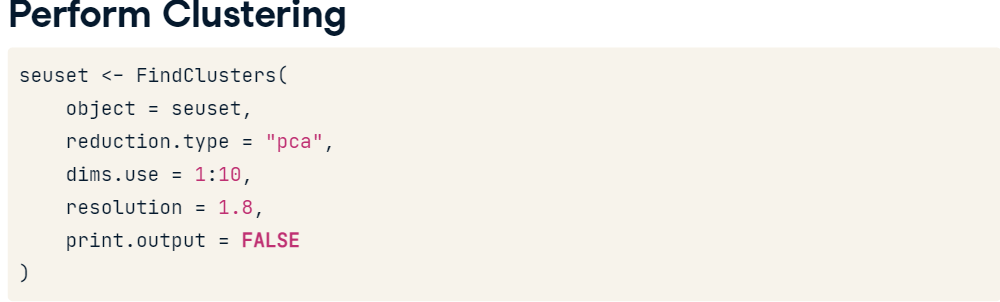
Unsupervised clustering is useful in many different applications and widely used in single cell analysis. Some of the most popular approaches are hierarchical and k-means clusterings. Briefly, in hierarchical clustering, each cell is initially assigned to its own cluster and then pairs of clusters are subsequently merged to create bigger clusters. In k-means clustering, the goal is to partition N cells into k different clusters in an iterative manner where at each step cluster centers are assigned and then each cell is assigned to its nearest cluster. The main challenges with all the clustering methods is to choose the number of clusters which often goes back to the question of the definition of a cell type. At which level do we want to cluster? Another challenge is the scalability. Clustering tools have been developed when the datasets were small, now that a dataset can contain millions of cells, tools don't scale well.

## **4. Create Seurat object**



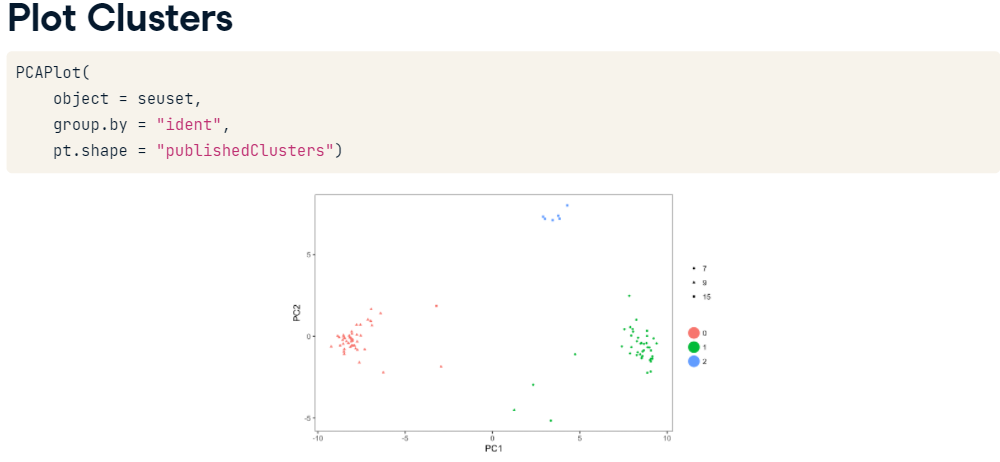
Let's now cluster cells from the mouse epithelium dataset. We'll use the package seurat, so we first need to create a Seurat object from the sce object using the function CreateSeuratObject(). The argument raw.data is the matrix of counts from the sce object and we want to keep the same meta.data as in the colData of our sce object. Following the steps of the workflow, we can normalize and scale the count matrix using the function scaleData(). You see that we've created a seuset object with 100 genes and 94 cells.

## **5. Perform Clustering**

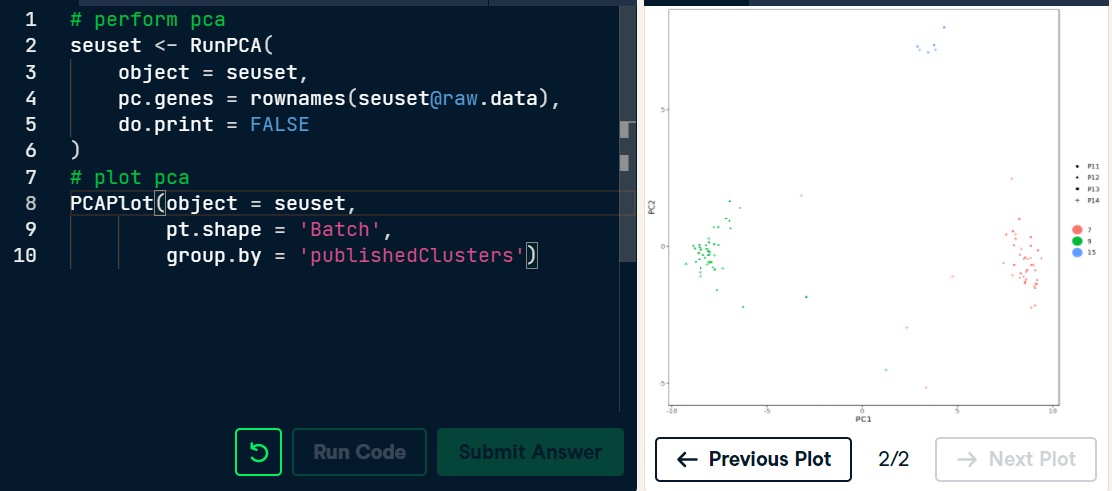


Now, to perform clustering on the seuset object, we use the function FindClusters() and then compare this clustering to the one from the published paper on the mouse epithelium dataset. Several clustering parameters can be tweaked. For example, the resolution parameter sets the granularity of the clustering, with increased values leading to a greater number of clusters. Here I chose 1.8 for the resolution. I encourage you to look at the documentation and explore the different parameters.

## **6. Plot Clusters**



We can now plot PC1 and PC2. Each dot on this plot is a cell, cells are shaped by the published clusters and grouped by clusters (called ident in the seurat object).



You ran PCA on the Seurat object and visualize the dataset in two dimensions.

## **1. Differential expression analysis**

The goal of this video is to perform differential expression analysis, that is, we want to find the genes that are differential expressed between known groups of cells.

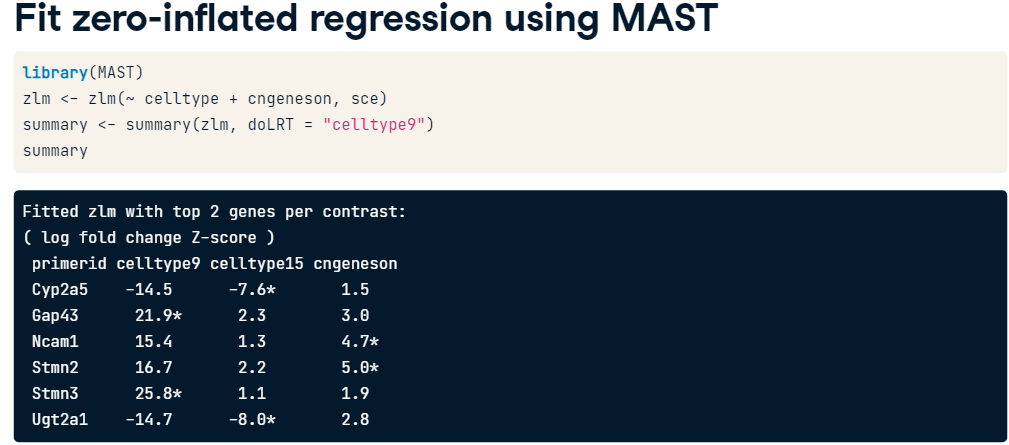
## **2. Typical workflow**

After assigning the cells to clusters, we are interested in finding genes that have non-constant expression patterns between cell-clusters. It's the last step of our typical workflow and it's usually called differential expression analysis or DE analysis.

## **3. DE methods**

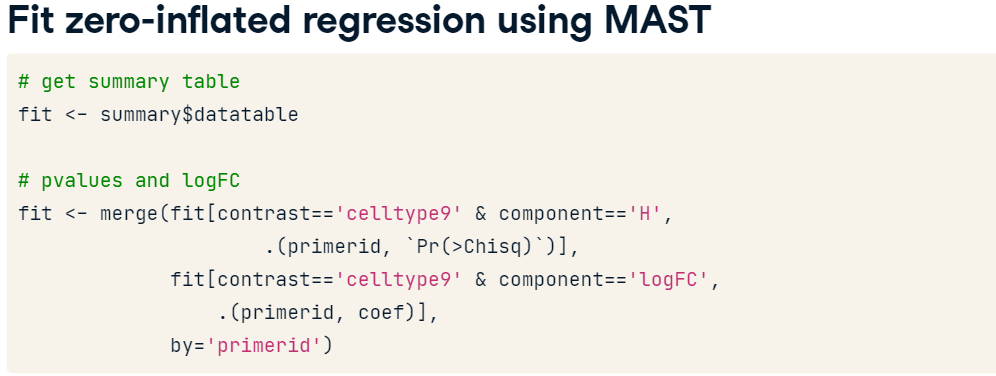
Several methods exist to perform DE analysis. Some methods were developed for bulk RNA-Sequencing data and are still used for single-cell data. This is the case of edgeR and DESeq2. If you've never heard of these methods, I encourage you to take the DataCamp course on bulk RNA sequencing analysis by Mary Piper. Other methods have been specifically developed for single-cell data and account for the zero-inflation in the data. SCDE was the first single-cell specific DE method. It fits a zero-inflated negative binomial model to gene expression data using Bayesian statistics. Another popular method is MAST which is also based on a zero-inflated negative binomial model but it uses a hurdle model. Again, I encourage you to look at the documentation for more details on these methods. In the following slides, we use MAST.

## **4. Fit zero-inflated regression using MAST**



To fit MAST zero-inflated negative binomial model for each gene across cells, we use the function zlm and regress the counts in the sce object on two variables: the cell-types found in the clustering step and the variable cngeneson which is the number of detected genes. Then, to compare cells in cell-type 9 and the cells in other clusters, we can use the function summary() and perform a likelihood Ratio Test using the argument doLRT equals cell-type9. You can see the log fold change Z-score for each gene and variable.

## **5. Fit zero-inflated regression using MAST**



Now, to get the p-values from the test, quite a bit of work is required. We want to get the p-values from the Hurdle model (so component H), the component log fold change and we store the result table in the variable fit.

## **6. Adjusted p-values**



The final step is to compute adjusted p-values. In the previous slide, we performed J separate hypothesis tests where J is the number of genes. In a typical single cell analysis where J is about 10,000 and if we use a standard p-value cut-off of .05, we'd expect about 500 genes to be deemed significant just by chance. So, to correct for multiple testing, we want to calculate the adjusted p-values. To do so, we use the function p.adjust. We can now create a table with for each gene, a p-value, an adjusted p-value and a log fold change to compare cells from cell type 9 to the other cells in the dataset.

# **Fit zero-inflated regression using MAST**

To identify genes that are differentially expressed (DE) between cell type 9 and the other cell types, we want to fit a zero-inflated regression using the function zlm() from the package MAST.

# **Create result table**

In the previous exercise, you fitted a zero-inflated regression to the mouse epithelium data set to compare the gene expression of cells from cell type nine to the other cells. Let's now process the summary table of the fitted model to get a p-value and log fold change (logFC) for each gene.

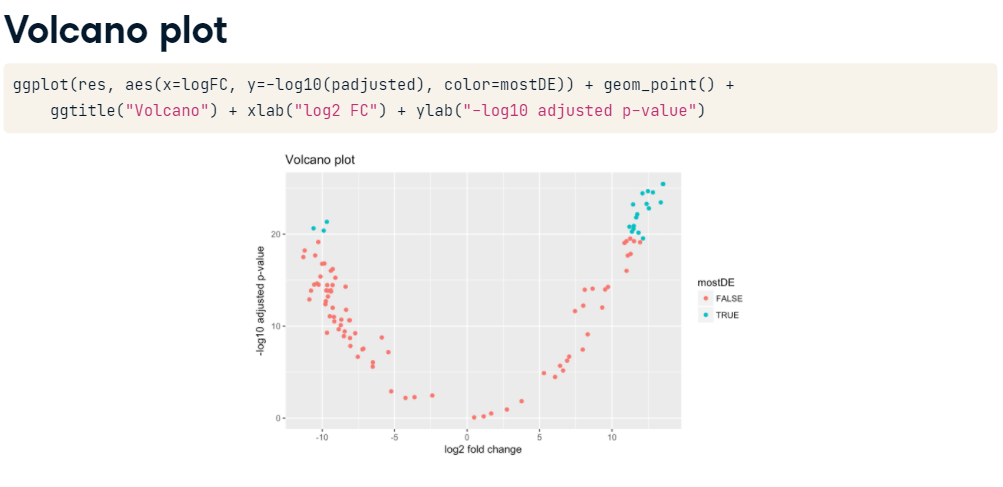
# **Compute adjusted p-values**

In the previous exercise, we performed J separate hypothesis tests where J is the number of genes. In a typical single cell analysis where J is about 10,000 and if we use a standard p-value cut-off of 0.05, we'd expect about 500 genes to be deemed significant just by chance. To correct for multiple testing, we want to calculate the adjusted p-values. To do so, we can to use the function p.adjust().

# **1. Visualization of DE genes**

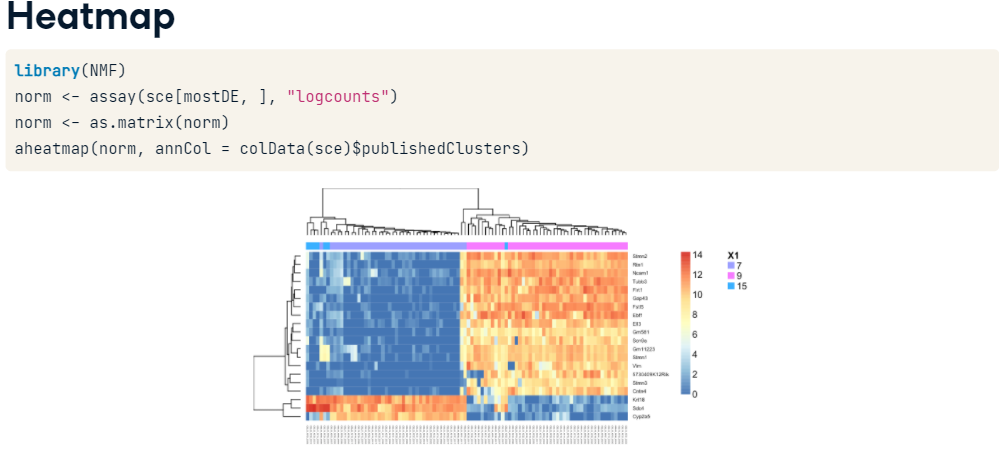
In this video, we visualize the result of the differential expression analysis using a volcano plot and a heatmap. These plots are usually the final plots of a single-cell analysis.

# **2. Volcano plot**



In single-cell analysis, a volcano plot is used to quickly identify the most meaningful changes in gene expression between different groups of cells. It combines a measure of statistical significance from a statistical test (e.g., a p-value ) on the x axis with the magnitude of the change or log fold change on the y axis. It is a great way to look at the results of a DE analysis since it allows quick visual identification of the genes that have both large magnitude changes and statistical significance. To plot a volcano plot, we can use ggplot2 with logFC on the x axis and -log10 of adjusted p-values on the y axis. The DE genes are the genes with high absolute log fold change and small p-values and are in blue on this plot.

# **3. Heatmap**



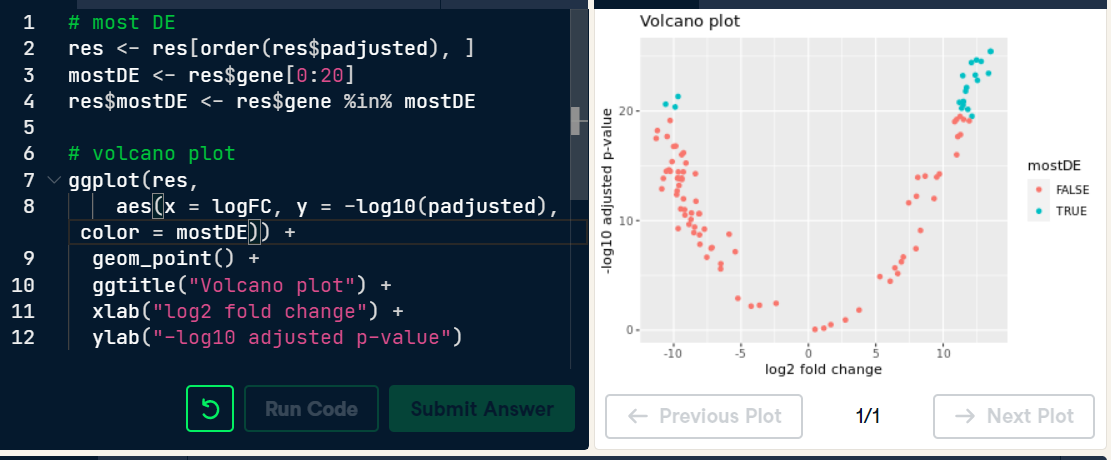
Another way to look at the results of the DE analysis is to plot a heatmap. Here we use the function aheatmap() from the package NMF with only the 20 most DE genes, that is the 20 genes with the smallest adjusted p-values. It shows the counts (normalized and on the log scale) with high and low gene expression in respectively red and blue. The purple, pink, and blue lines at the top of the heatmap indicates the clusters found by the published paper on the mouse epithelium dataset. As you can see here, cells in clusters 7 and 15 have a completely different gene expression profile compared to cells in cluster 9.

# **4. Typical workflow**

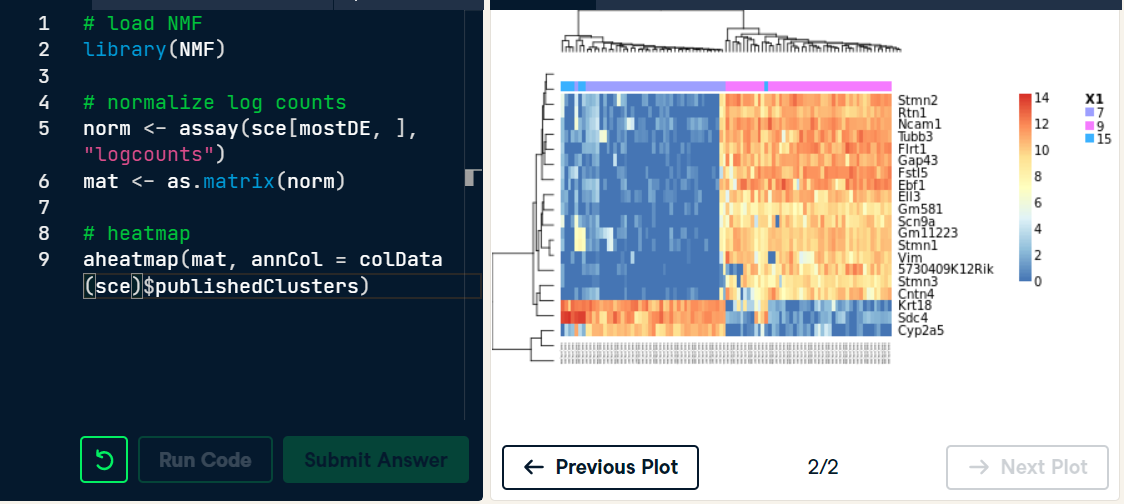
This course gave you an overview of a typical workflow for the analysis of scRNA-sequencing data in R. It covered four main steps: (1) normalization, (2) dimensionality reduction, (3) cell clustering and (4) DE analysis between the found clusters. I hope the proposed workflow will ease technical aspects of single cell data analysis and help you with the discovery of novel biological insights. To learn more about how to analyze single cell data, I highly recommend the material referenced below. The first reference has detailed information about each step of the workflow with different tools and it is very well explained. And the last two references are workflows to analyze single cell data.

# **Plot volcano plot**

To visualize the differentially expressed (DE) genes and choose threshold to identify DE genes, we want to plot a volcano plot. This plot has the log fold change (logFC) as the x-axis and -log10 of the adjusted p-values as the y-axis.



You've just plotted a volcano plot to visualize the most DE genes. This plot allows you to quickly identify interesting genes, that is genes with both small pvalues and high fold change between clusters.



Congratulations on successfully visualizing differentially expressed genes with a heatmap.