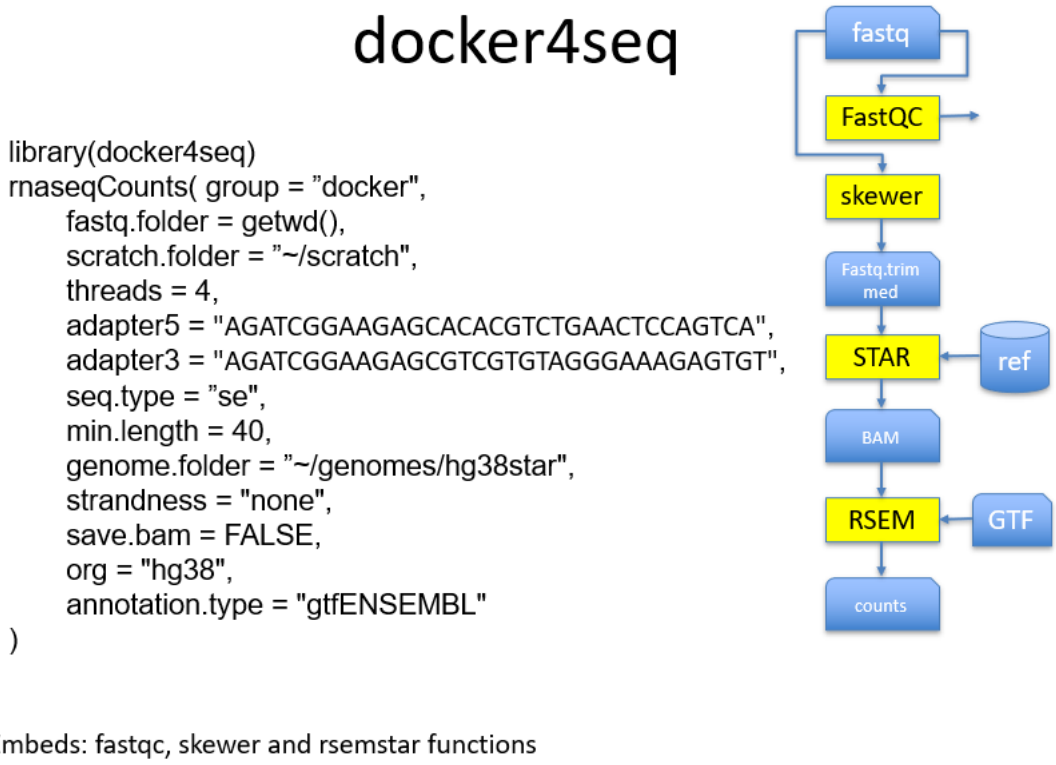
4 - scRNA 10x seq and Cellranger

TOPICS	
DATE	@March 30, 2022 2:00 PM
LAST EDITED BY	
LAST EDITED TIME	@May 4, 2022 2:00 PM
MADE BY	
PROF	Calogero
Recording	
STATUS	☑

In the following figure there is a summary of all the passages referring to the full set of steps to transform fastQ files to the actual counts of RNAseq.



Docker4seq is an R package featuring multiple functions able to recall different dockers to perform the alignment procedure seen in the previous lecture.

So the fastQ file is analyzed by FastQC, then trimmed by Skewer and lastly the trimmed data passes through STAR for the mapping to a reference genome.

A BAM file is generated, containing the position of the reads on the mapped genome and then this file is fed to RSEM that generates the count table using the .gtf file for the annotation.

In the exercise we will start from the count table created by these functions, otherwise we would need a lot of RAM.

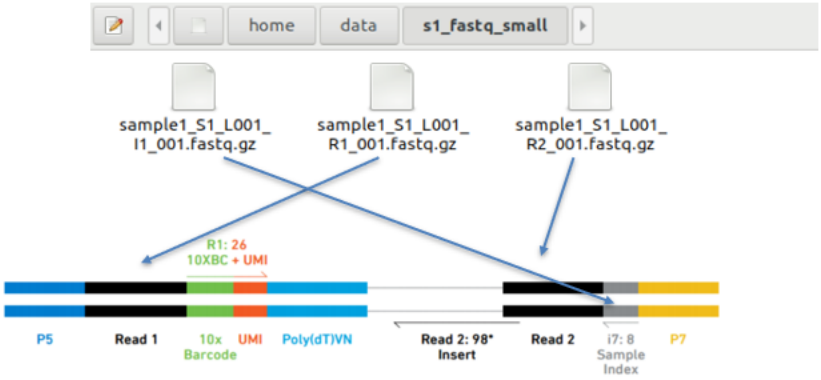
Docker4seq is a docker containing all these functions wrapped together. We can't use it on out PC because it requires a lot of RAM.

Single Cell RNA-seq: FastQ 10x genomics

What can we do if we work at scRNA-seq level? Is the procedure the same as bulk?

In bulk, multiple samples are sequenced together; in scRNA-seq, you sequence single cells from multiple samples. From each single sample, you will have the transcripts of multiple cells, thus you need multiple identifiers to differentiate those transcripts.

The sequences obtained are divided by sample by the **index** identifier, while the **barcode** defines the cell from which the transcript is coming from and the **UMI** is the identifier that is actually used to count the number of transcripts.



Sequence Read	Recommended Length	Read Description
Read 1	100bp	10 barcode and UMI
I7 Index	8bp	Sample Index Read
Read2	100bp	Transcript Tag

R1 forward and R2 the reverse

In scRNA-seq, three fastQ files are obtained:

- The I1 file contains the sample index sequence (six nucleotides), For example “exp1_ctrl12, “exp1_treated1” and so on.

Within each sample, the R1 and R2 files are obtained:

- The R1 contains the information related to the 10x barcode, that identifies the cell, and the UMI. R1 provides the barcode (cell identifier) and the UMI (transcript identifier) information. The UMI is used to count the transcripts, the unique molecules.

  In the single cell sequencing approach, each sample will contain the transcripts of multiple cells; this means that there will be other identifiers like the 10x barcodes. For example, for 2000 cells there will be 2000 different barcodes within the R1 file (10x barcode).

- R2 contains the information linked to the mapping, related to the unknown cDNA sequence, to the 3’ of the transcript. Basically it tells the software which piece of cDNA is associated to that specific sequence.

In the single cell sequencing, first you divide the samples with the index information in the specific fastQ file “Index1”; then, with the R1 and the R2 you connect the information of the sequence of the transcripts with all the identifiers you need to identify uniquely each cell and each transcript.

With R1 and R2, you are actually looking at the two sides of the same sequence; the R1 the 5’ end (forward sequence) and the R2 the 3’ end.

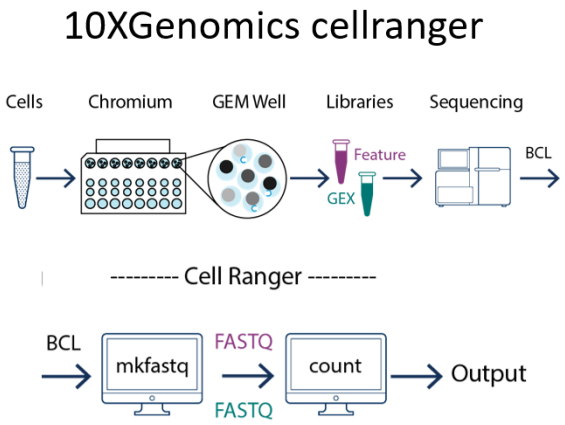
Since in scRNA-seq the amount of material is very tiny, the pipeline also includes a PCR step, which can generate a lot of artefacts. Thanks to the UMI (10 nt random sequence always different within the transcripts of the same cell), which is associated to a specific transcript, it is possible to precisely count the reads and remove any artefacts.

For example, the P53 transcripts mapping on the P53 gene in the R2 are 9, but only 3 UMI are present; this means that each read of the transcript has 3 artefacts. The real number to be associated to P53 is not 9 but 3.

to remove PCR artifacts

10x Genomics Cellranger

for bulk core facility after sequencing prodcue the BCL fioe for the fastq
for the single cell there is a specific sofytawre called mkfastq and from that fastq you get the counts

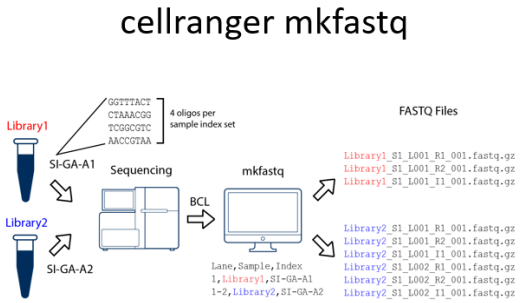


Here above are schematized the steps that are related to the single cell RNA sequencing; the upper part of the figure concerns procedures that won't be addressed in this course.

The output of the single cell sequencer is constituted by a **BCL file**, which is a collection of the images that contain all the info related to each single nucleotide detected over the various steps of the sequencer.

This BCL file is converted in a fastQ format (3 files) with a specific program, called **mkfastq**. This is different from what happens with the bulk sequencing, where the core facility produces the BCL and generates the fastQ file; in this case, the BCL to fastq conversion is done within the workflow of the sequencer; for example, Illumina has a specific workflow to provide this piece of information. The workflow is sequencer-specific, we will talk about the 10X genomics which is the most used.

The mkfastq software gives as output the three fastQ files described previously: I1, R1, R2.



The wrapper program that allows to do basic manipulations on the data coming from the sequencer is celled **cellranger**. This program is used in the 10x genomics single cell platform pipeline.

The cellranger mkfastq command line is reported below:

```
docker run  -i -t \
-v /somewhere/10Xgenomics:/scratch \
-v /somewhere/genomes:/genomes \
-d docker.io/repbioinfo/cellranger.2021.06 /bin/bash

/bin/cellranger-6.1.1/bin/cellranger mkfastq \
--id=MYSAMPLE\
--run=/scratch/210908_NS500140_0434_AHNJ5GBGXH \
--csv=/scratch/210908_NS500140_0434_AHNJ5GBGXH/cellranger-tiny-bcl-simple-1.2.0.csv
```

As you can see from the command, cellranger has been located in a docker; in this way it can be used by any PC; the first part of the command line corresponds to the command given to run the docker image.

In this case, after the docker run -t -i, there is the option -v followed by the path to the location of the BCL file. The -v option enables to mount on the docker the folder where the file you are analyzing is located; after the path, it is necessary to write where to mount the file in the docker. This is done by adding `:/location/in/the/docker`, for example: `/home`.

The last part of the first command is the name (or the path) of the docker that contains the cellranger mkfastq command.

This command is run directly from the terminal; once inside, there are a certain number of information that have to be passed and this is something that is normally found in the instruction of the methods.

Remember that the option -i means interactively and -t means that you have the possibility to access the root of the docker by typing.

The important thing is that, at the end, a **comma-separated-file (.csv)** is generated: it contains the name of the experiment and the index that is associated to that specific experiment. Even with this scarce data, the software is able to organise everything in order to give the operator the three files desired.

```
Lane,Sample,Index
*,trt,SI-TT-A2
*,ctrl,SI-TT-B2
```

index is the 6 nucleotide sequence

The Illumina sequencer generates a BCL file which contains the information of all the samples together. Each of the samples is associated to specific code in an index file. For example, the code in the image (SI-TT-A2) is the index code.

First, the program mkfastq is able to separate the ctrl and the treated samples; this is achieved by creating an **Index file I1**. The fastQ I1 file will be the same for all the samples of the same experiment, like the treated and the ctrl of this case, thus it is kind of useless.

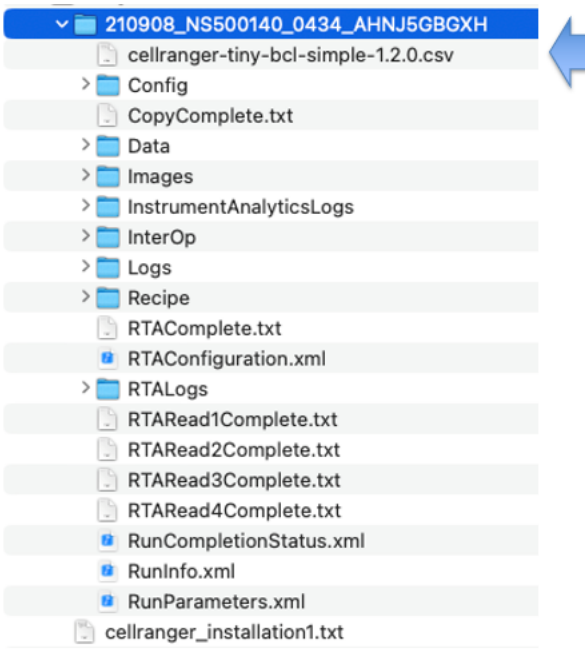
Each of the samples will have a I1 file (in this case it will be the same for the two), and then the fastQ R1 and R2 for each sample.

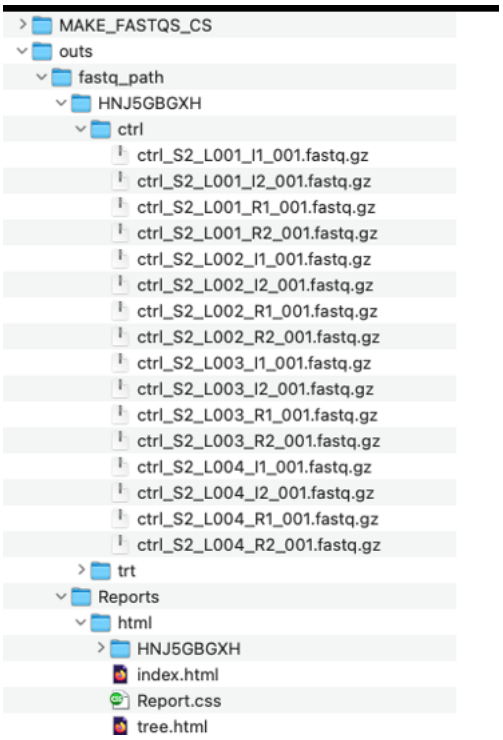
The UMI is a 10-base sequence that allows to solve the problem given by PCR artefacts, since all transcripts with the same UMI will be counted only once. It could be possible to have the same UMI in different cells, but in this case the barcode sequence will be different and thus there would be no problem in separating those reads.

these 10 nucleotides are totally random and will be differetn within the same cell and asssociated to a specific trasncript.

What's the structure of the FastQ files?

So in R1 there are various informations: the barcode for the cell. You are aactually reading the sequence, with R1 the forward of the same pieces od DNA and R" with the 3' end. Then also other informations are present apart from the rranscript.
In R2 the trascript sequence thta will be mapped to the genome adn then i will obtain the transcript and the cell to which the transcript belong. If i read multiple time the transcript this means thta is transcripted some times in the cell.





In the .csv file all the information is collected; this file is contained in the BCL folder (left picture). In the picture on the right some of the fastQ files of ctrl samples are shown; in this case there are two indexes files, I1 and I2. This means that in those samples there are two indexes, one on each side; they are needed when the number of samples is very big.
With only one index, you can run maximum 96 samples; while with two indexes the amount of samples that you are able to run is much, much bigger.

📌 Two indexes are necessary to distinguish all the samples in big runs

When the fastQ file is created, there is also another output which is the **index.html**, which records the information related to the quality of the sequence. Looking at the picture below, it is possible to see that the cell flow is divided into 4 lanes and each of them provides some information.

In fact, in the previous images, the ctrl samples had, in their name, L001 to L004, this means that t**he flowcell is divided in four lanes and the transcripts coming from the same sample are spread in all the four lanes**. Thus, files referring to the same sample can come from different “slots” of the sequencer.

beacuse everything is sequenced together but distributed on lanes

Flowcell Summary

Clusters (Raw)	Clusters(PF)	Yield (MBases)
773,748,026	366,018,599	43,190

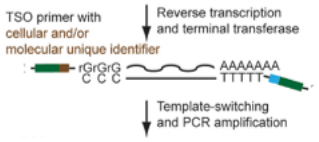
Lane Summary

Lane	PF Clusters	% of the lane	% Perfect barcode	% One mismatch barcode	Yield (Mbases)	% PF Clusters	% >= Q30 bases	Mean Quality Score
1	92,209,323	100.00	97.78	2.22	10,881	47.14	92.86	34.07
2	90,675,785	100.00	97.64	2.36	10,700	47.31	92.92	34.09
3	92,012,848	100.00	97.49	2.51	10,858	47.39	92.72	34.05
4	91,120,643	100.00	97.56	2.44	10,752	47.39	92.81	34.07

‼ Beware that transcripts from a single sample will be distributed randomly across all the lanes!

The first step of the analysis is the conversion of the fastQ files into counts, with an approach that is similar to the one used for the bulk analysis, but with some adaptations to the characteristics of the single cell analysis.

The first thing to notice is that the RNA library for the scRNA analysis has been built by exploiting the polyA of the transcripts. When constructing these libraries, normally the primer contains an oligodT useful to construct the first strand; in order to have the second strand, the longer transcript, a specific type of adapter is used and it's called TSO.

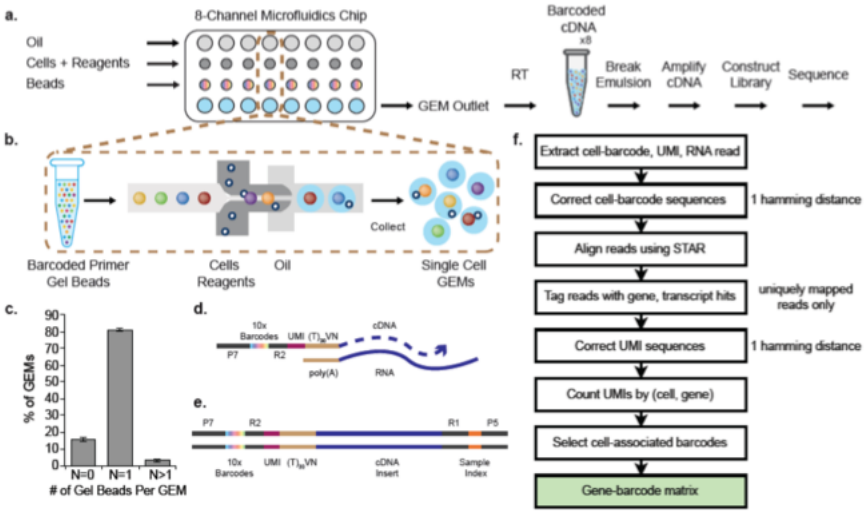


The TSO contains a polyG and a know oligo sequence and it's used to make a full cDNA by annealing to a set of C's that the retrotranscriptase has been forced to attach to the 3' end.
The TSO is **an oligo that hybridizes to untemplated C nucleotides added by the reverse transcriptase during reverse transcription**. The TSO adds a common 5' sequence to full length cDNA that is used for downstream cDNA amplification.

Practically, in the libraries prepared with this method, each cDNA strand will have a oligodT on one side and the TSO on the other. Meaning that a full length cDNA construct is flanked by the 30 bp **Template Switch Oligo (TSO)** sequence on the 5' end and a polyA on the 3' end.

If the unknown sequence in the middle is a short fragment, it is frequent that it contains both "extra" pieces in the sides; these fragments have to be removed to perform the analysis.

To remove the fragments, a **trimming** is performed: the polyA tail and the TSO sequence are removed to get only the unknown piece of DNA in the middle that is then stored in the R2 file. This step is necessary to clean up all the extra information before aligning to the genome.



The beads used in the scRNA sequencing contain an oligo with a PCR element (P7), the 10x barcode (the bead identifier), the sequencing primer, the UMI and the oligodT. The library creation starts from the pairing of the mRNA poly-A to the oligodT, the reverse transcription and then the interaction of the TSO with the C's at the endo of the first retrotranscribed strand. Then, the second strand is polymerised.

However, the TSO sequence shouldn't be there for the sequencing analysis and it is cut via the process of **tagmentation**. To do so, the yeast transposase TNA5 is used to cut the DNA in random positions depending on the concentration.

Basically, the trimming step is used to remove the TSO sequence and thus leave the 5' end free to attach the other sequences of the R1 needed for the scRNA sequencing. Sometimes the R1 can attach over the TSO and this causes a mess in that sample droplet; this happens often when the cDNA fragment is too small to be targeted in the tagmentation process.

After trimming: STAR alignment

The cDNA fragments are fed to the STAR software to align them; this program enables to align sequences considering that they should be only on exonic regions and that they should align at least for 50% of their sequence. This is what is called a “good alignment”.

MAPQ adjustement

Since the cDNA piece is little, it is possible that they will map to multiple regions; but usually only one of these multiple mapping positions considers an exonic region. This means that no multiple mapping on exon is observed and multiple mapping regions will map on introns rather than on other exonic regions.

Only the one mapping on the coding region will be considered correct, all the other mapping positions are discarded and that read is associated to that specific exonic position. By doing so, multiple mapping is pushed to have the reads associated to the exonic elements.

Transcriptome alignment and annotation

Once the cDNA read has been aligned to an exonic area, this area is assigned to an annotated gene. After this the reads are counted.

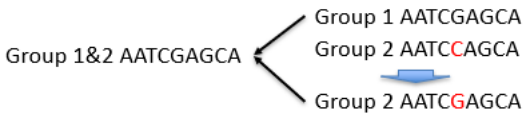
Counting

To perform the counting, it is necessary to check the R1 file of the specific reads; to be counted as real reads, meaning that they are coming from different transcripts of the same cells, the sequences have to contain the same 10x barcode but different UMI.

So basically the mapping is done to all the sequencing without considering their origin, then, by checking R1, the sequences coming from the same cell are pulled together.

Then the UMI is considered to differentiate PCR artefacts.

UMI counting



UMI’s are also checked to be sure that there are no single nucleotides errors occurred during sequencing (remember: we are talking about sequencing-by-synthesis technologies!). To do so, these barcodes are organized in terms of similarity to control and are regrouped before counting to eliminate artefacts that would lead to overcounting.

Since also errors in UMI’s can happen, if there is a UMI clearly less frequent than a very similar other UMI, the two are grouped to avoid losing reads or counting of non-existent UMI’s.

For example, there are 100 AATTCC barcodes and 1 AACTCC barcode; the second barcode is obviously a sequencing error for the UMI. To correct it, it’s just necessary to reaggregate and combine all those mismatching barcodes to the most sequenced UMI sequence and virtually remove these point mutations. This is done to avoid over-estimation of the RNA expression given by errors in the UMI sequencing.

In this process, the threshold accepted is of one mismatch over 10 nucleotides. This step can be done even before the association of the sequences to the gene.

The cell barcodes in 10x genomics are around 300.000, they are a fixed number, they are known prior to the experiment and they are much less than the UMI barcodes. While the UMI barcodes are much more since they are built as random sequences.

Because of this, this kind of check is not done on the 10x barcode sequences.

> ✅ With this process you are trying to avoid that an error is counted as a positive value

N.B. In the process of mapping, each sample (i.e. each type of treatment) is mapped alone since they are divided considering the R1.

LET'S SUMMARISE

Let's consider the analysis of a control sample, for example 300.000 cells, of an experiment. First of all, this sample has a fastq I1 file different form the other samples, thus it can be completely separated from them; this is the first thing done in the analysis.

With the fastq R2 files, first of all STAR is used to map the reads to the genome sequence, to get how many times a specific gene is detected. Then, since the reads are paired (i.e. you know the origin of each sequence by the association of R2 and R1 files), it is possible to align all the UMIs contained in all the R1s associated to the R2s mapping on the same region.

By doing this, it is possible to count the actual reads, removing the PCR artefacts and the possible mismatches to make them identical to the most expressed group.

After this kind of adjustments, it is possible to start counting how many transcripts are associated to the specific gene in the specific cell.

Calling cell barcodes

Cell Ranger is the tool designed to make the fastq file; the 3.0 version of this program has also the **count** option that enables the mapping. This function generates the matrix that has then to be converted into a dense matrix (this will be the exercise of this lesson).

With bulk RNAseq analysis, each sample has one sequence, for example, patient 1 sample contains the expression of 20000 genes (that corresponds to the average of multiple cells analysed in bulk). And then the bulk experiment involves multiple samples to analyse, like ctrl ad treated triplicates of the patient cell line.

For the single cell analysis, the situation is different: first of all, ctrl and treatment cannot be directly compared since they have been analysed in different moments; they are 2 independent experiments done in two different times.

Normally, the comparison is done within the ctrls and within the treated samples.

In the bulk analysis, the focus is on searching molecular events (like changes in gene expression) that are discriminating the experimental conditions, e.g. the cell line with and without treatment, thus the observations is on how the genes modify their expressio upon treatmentr.

While in scRNA sequence the samples are part of two heterogeneous populations of cells, characteristic that in the bulk cannot be assessed, and, since they are heterogenous, they may have different responses to the treatments. For exemple, one subset of the ctrl cells might be killed.

The advantage of making a single cell experiment is that the phenomena that are due to the change of gene expression in a small subset of the cell population can be analysed; this phenomena is masked when doing bulk analysis.

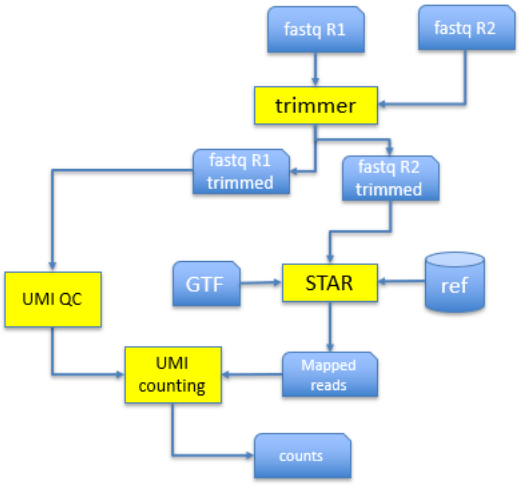
In the bulk, it's impossible to know if the change in gene expression is associated to a specific cell type.

In practice, in scRNA sequencing analysis, first the **clusters** of subpopulations within the samples are analysed, then the differences between the same cell subpopulations between the two samples are searched.

With bulk analysis, of course, this is not possible. The power of single cell sequencing is to detect small subpopulations of cells that may be driving important biological events but that could be masked by other cell types in bulk analysis.

Here below an example of the pipeline to obtain the counts for each gene, which was discussed up to now in the lessons.

We won't do this part as an exercise, since it needs a very big power of computation which is not present on normal PCs.



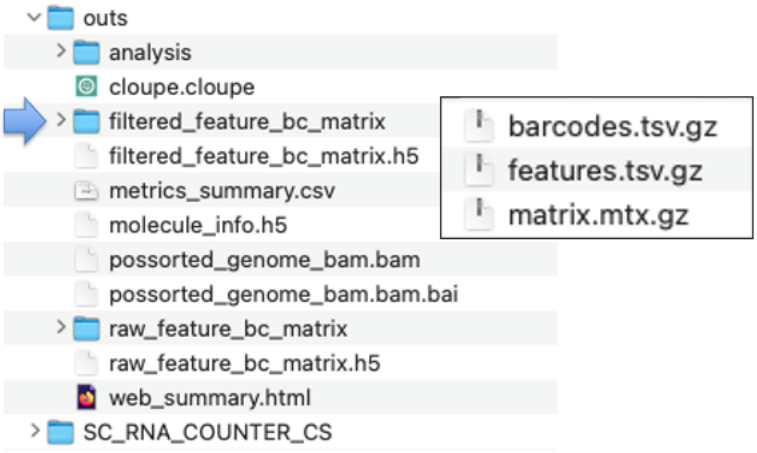
Basically, the mkfastq command (seen before) generates the separation between the samples and the R1 and R2 files, then the cellranger count is used to actually count the sequences.

In the bulk analysis, the mkfastq is not seen, since the fastq file is obtained directly. This is because the Illumina sequencer is able to do the BCL→fastq conversion directly. Thus the output of the sequencer is already a fastq file, not a BCL like it happens in the scRNAseq.

What we are seeing in the lessons is the simplified version of the scRNA sequencing approach. Today, in the scRNA sequencing, up to 300000 cells all together can be analysed; because of this multiple samples cam be mixed together and run simultaneously. Because of this, cellranger adapts specific procedures (not discussed in th elessons), thìo separate those samples in subgroups and analyse them one by one.

Cellranger output: Filtered features barcode matrix

the filtered feature barcode matrix is a file that contains the putput of the cellranger analysis; Indeed, it's a folder that contains three files: "barcodes.tsv.gz" (cell barcodes, column names), "features.tsv.gz" (name of the genes in the table, row name of the tables), "matrix.tsv.gz".



Why are they organised this way?

In the single cell analysis, only few genes are detected as expressed in the cells; indeed, some of them are not detected both because it is possible that stochastically some transcripts did not attach to the beads, because some genes are not expressed at all or because they are expressed below the threshold.

the point is the following: if there are 3000 cells, but in each cells only 100 genes have an expression level different from 0, the remaining 19.900 genes will have a 0 in the table. The same happens in each cell; this means that 90% of the table generated by the single cell sequencing has cells with zeros.

This table is very large and mostly made of 0 values; in order to have a more compressed table as output, the file is shrunk (around 1MB) to a **sparse matrix** in which all the 0 values have been eliminated. This table has a matrix containing only non-zeros values.

Then, to actually perform the analysis, the sparse matrix has to be converted into a **dense matrix** which can be represented as a comma separated file, .csv, that is much bigger in size, it has been expanded to almost a GB of size.

In the image above, the cell barcodes are represented in the upper left; they will give you basically the name of the column (?)

Then the genes.tsv.gz file contains the gene identifier (ENSMUSG....string) and the corresponding symbol; since a few years ago, only genes could be sequenced by 10x genomics, but now this technology is multimodal and enables to sequence also antibodies or other proteins.

Because of this, the actual name of the genes.tsv.gz file has been changed to features.tsv.gz (in the lower part of the image), to comprehend also these other sample types.

Lastly, the matrix file, which contains at the beginning values indicating the total number of rows, columns and counts. In the .mtx format (sparse matrix) the zeros are never shown, thus this is a tiny file.

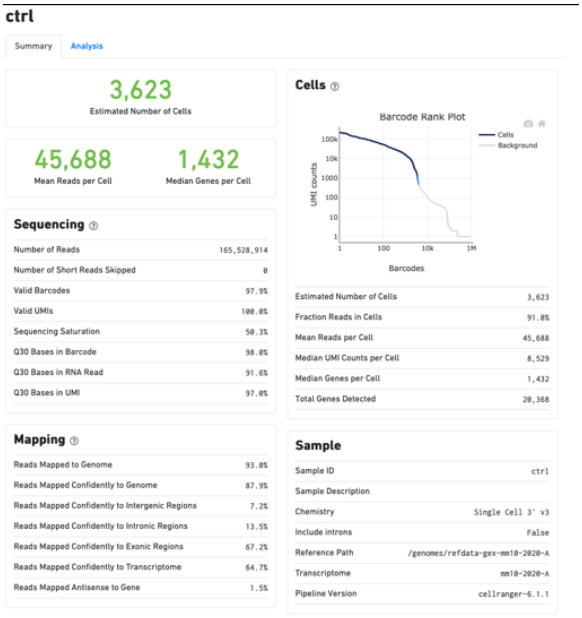
The output of the counting is a folder called **filtered_feature_bc_matrix** with the 3 files that are used by cellranger which uses the method **mat2csv** to make a csv file.

Thus the three files can be used by cellranger to reconvert the sparse matrix back to a dense matrix, the one comprising all the 0's.

The output HTML file

In the bulk analysis, after the counting, it is possible to do a multiQC analysis to evaluate the quality of the experiment. The same can be done in the scRNA sequencing, obviously in a slightly different way.

Here below there is the output that is given as part of the count analysis, meaning that the cellranger count method generates also an html file containing information related to the quality of the experiment.



The first part of info is the estimated number of cells that is detected, the number of reads per cell and the median number of genes detected per cell.

The first parameter depends on the type of experiment; if the starting samples contain 10.000 cells each, you will obtain a number which is smaller than the amount you actually put (2/3 approximately).

The mean number of reads per cells indicates the number of sequences detected associated to one barcode; the more reads you have, the more genes you can detect in each cell. The ideal condition for the single cell analysis is to have this value around 50.000 to have a good gene coverage.

Below 20.000 reads per cell, the resolution is too low and the number of genes detected per cell drops dramatically. The good range is between 20 to 50.000 per cell. to have a good coverage

The number of detected genes is defined by the number of UMI associated to each gene; practically, with 45000 reads, you can detect in one cell, at least 1.400 genes with [redacted] at least one UMI

Thus the median genes per cells corresponds to the # of genes detected in a cell with at least 1 UMI.

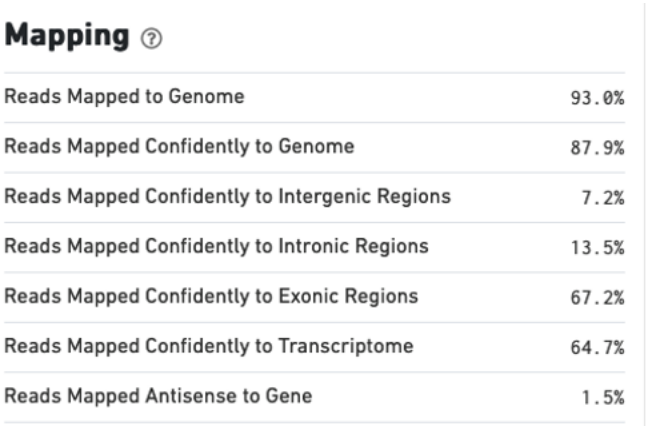
Sequencing

Number of Reads	165,528,914
Number of Short Reads Skipped	0
Valid Barcodes	97.9%
Valid UMIs	100.0%
Sequencing Saturation	50.3%
Q30 Bases in Barcode	98.0%
Q30 Bases in RNA Read	91.6%
Q30 Bases in UMI	97.0%

Another thing contained in this file are the information on the sequence.

The most important value is the number of barcodes and of the RNA reads with Q30; as seen in the previous lessons, Q30 indicates a quality score, the phred score. To have a good quality, this score has to be 30 at least (40 is the max value). In the case of the

image, the high percentage of Q30 barcodes and RNA reads tells us that the sequencing quality of the experiment is good.



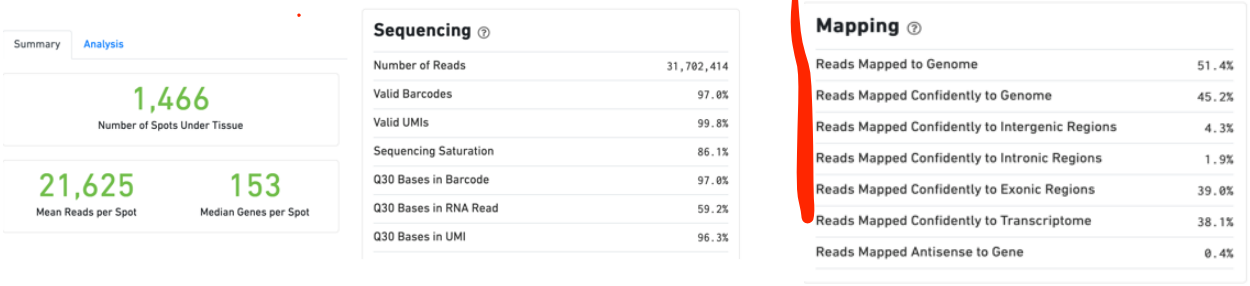
Mapping

Reads Mapped to Genome	93.0%
Reads Mapped Confidently to Genome	87.9%
Reads Mapped Confidently to Intergenic Regions	7.2%
Reads Mapped Confidently to Intronic Regions	13.5%
Reads Mapped Confidently to Exonic Regions	67.2%
Reads Mapped Confidently to Transcriptome	64.7%
Reads Mapped Antisense to Gene	1.5%

Some information, on the other hand, are related to the mapping; for example the number of reads mapped to the genome, or the reads mapped confidently to introns, exons, transcriptome and so on. In this case, all these values indicate a good quality of the experiment, even if some of the percentages are lower. For example, the reads mapped to the exonic regions are only 67%; however, this is a good value if we consider that in single cell sequencing, the mapping is only of the reads of the 3' end and thus it is possible that some of those reads are not counted because the program was not able to associate them to the 3' end of a transcript.

Out of the exonic, nearly 64% is associated to the transcriptome.

Here below, an example of a low quality result:



In this case, the number of mean reads it's not too bad, but considering the number of cells analysed, this is already an indicator that the experiment is not of super high quality.

The number of genes is the parameter that seals the deal: this is not a good experiment. In this case, even with 21.000 reads per cell, only 153 will be mapped on exonic regions.
This means that the quality of the RNA in the sample is really poor.

Also the other sequencing and mapping parameters show the bad quality of the experiment (which is from a pancreatic tissue extraction, which is a nightmare to do and obtain good quality RNA).

Overall, these information give the user a good overview of the general quality of the experiment.

With the bulk multiQC, you don't look at the experiment itself, but you observe how the various cells behave and the number of genes that are detected. Those are the information that make the user understand if the experiment is reasonable or not.

The output of the bulk analysis is just one row of counts associated with each gene for every sample (a vector); in scRNA sequencing you have a matrix for each sample (thousands of cells).

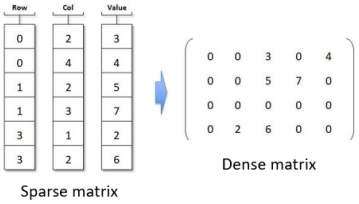
In bulk you compare the experiments and use multiple samples for the analysis, in single cell you compare the cells within one experiment (sample).

EXERCISE AT HOME

Output data

The aim of this exercise is to obtain a dense matrix (.csv file) from a sparse matrix that contains all the rows and columns of the input file, but without all the 0 values. This makes the dense matrix a much bigger file than the sparse matrix.

Row are the features
Col are the barcodes
MAtrix thta cintain the matrix
this is the input



each column is
the cell and the
row the genes

```
#link for the info to download cellranger: https://support.10xgenomics.com/single-cell-gene-expression/software/downloads/latest
0. log into the docker you have created as exercise for lesson 2
1. install wget using apt-get
2. download cellranger in /bin using wget
3. use gzip -d to unfold the compressed file
4. use tar xvf to unpack the folders
5. read the instruction to use cellranger mat2csv method: https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/
latest/output/matrices#mat2csv
6. write a bash command to execute mat2csv
7. Execute the command file in background within the docker container
8. Execute the command file in background using the docker run command
```

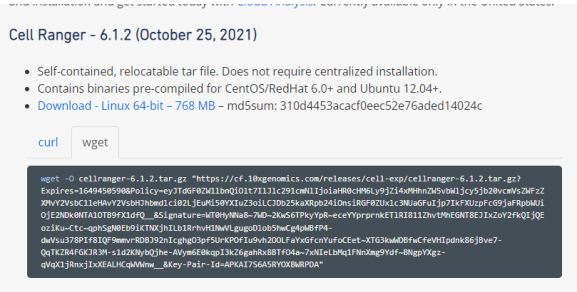
INSTALLING CELLRANGER

1. From the terminal run the docker you have created for the last exercise (r4:v.0.0.1)
2. Use the command below dockto install wget in your docker.

   ```
   apt-get install -y wget
   ```

3. KEEP THE TERMINAL OPEN AND DO NOT EXIT THE DOCKER IMAGE

1. Using the link in the exercise explanation above, find the cellranger software dowload page.
   You will need to fill in some of your information for the licence and then you'll be redirected to the page with the `wget` command to copy. ‼ copy the link in the big box ‼



2. In your terminal use the command below to enter the bin.

```
cd /bin
```

3. Paste the wget command in the terminal and press enter.
4. Cellranger will be downloaded locally in the bin. If you use “ls” in the bin, you will be able to see the cellranger as a .tar.gz file → -gz indicates that the file is compressed, like a zip file.
5. To unpack the cellranger use the command `gzip -d nameofthefile`. You will obtain a tar file; tar files are unique files containing all the folders packed together, that are then compressed with the gzip system.
6. To separate the folders use the command `tar xvf nameofthetarfile`. You will obtain a “cellranger” folder in the bin. The xvf options of this command stand for x=execute, v=verbose, f=file.

> 💡 To compress material the command is the same with c instead of x

6. EXIT THE DOCKER IMAGE AND COMMIT THE CHANGES
7. Now you will need to convert cellranger in an executable bash command (not explained in the lesson). You can read the instruction to do that in the same website as the one for the download, but basically you have to edit the .bashrc file with nano (that you have installed in your docker when you created it) and add at the end the command line that starts with export.
   https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/using/tutorial_in
   The part concerning this point is in the section “Add Cellranger to your path”.
8. Now you are ready to run cellranger
9. EXIT THE DOCKER AND COMMIT THE CHANGES

RUNNING CELLRANGER

There are 4 ways you can run the cellranger command:

A. From inside your docker writing down the command

B. From inside the docker using a file with a shell command

C. From inside the docker but in background

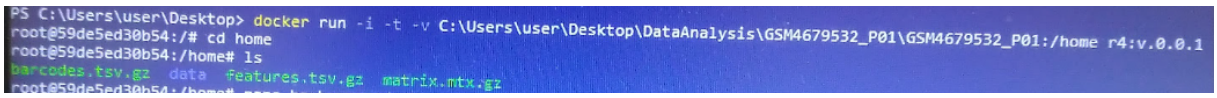
D. Directly from the terminal

A: running cellranger in the docker

1. Download the lesson 4 dataset from moodle. Unzip it and locate it in a folder of your choice, or the desktop.
2. Open the terminal.
3. Run the docker (always the same one, for me the name was r4:v.0.0.1) and mount your dataset folder in the docker home with the following command:

```
docker run -i -t -v path/of/the/folder :/home r4:v.0.0.1
```

4. Enter the home of your docker and do “ls”, you should see the three files of the dataset.
   In theory you could mount your folder in other parts of the docker, for simplicity put it in the home.



5. Create a new folder called “data” in the home of your docker with the command.

> 💡 It’s possible that this step is not necessary and that doing step 6 directly will work. In case it doesn’t, create the folder before going to step 6.

```
mkdir data
```

6. To run cellranger use the following command:

```
cellranger mat2csv /home /home/data/data.csv
```

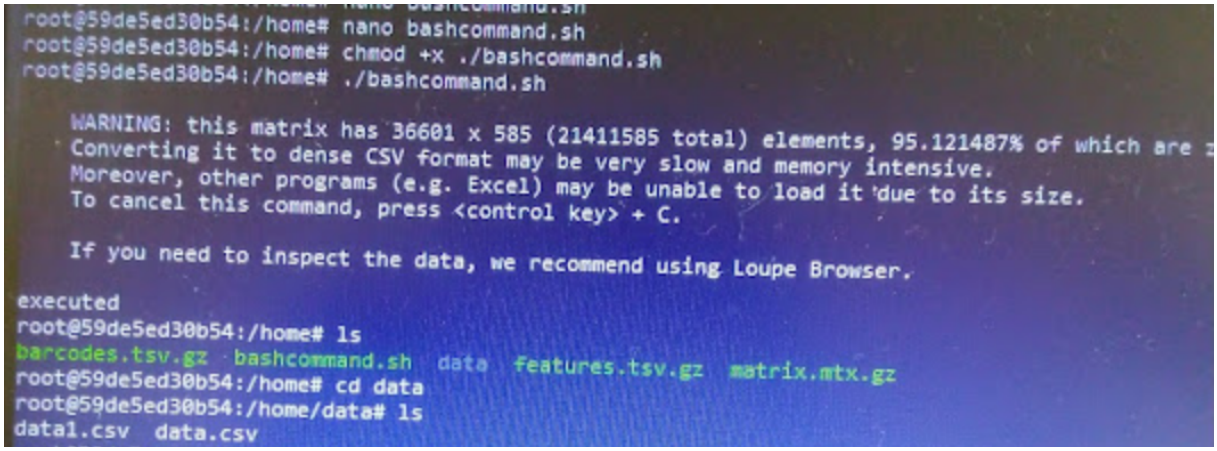
   This command basically orders to cellranger to convert the .mtx.gz file that you have mounted in the home (the /home part of the command) in a .csv file. The last part of the command line, `/home/data/data.csv` , is used to create a file in the data folder where the output will be written. Indeed, this last part of the command is the path of your output file, which in this case is called data.csv, but you can call it how you want.

7. The data.csv file is your dense matrix.
8. EXIT AND COMMIT

> ⛔ There is another way to mount the folder in your docker; basically you create a new docker image with a copy of the folder inside it.
>
> Note that THIS IS NOT IDEAL; it is always better to have the files locally and not in the image.
>
> To do this, you create a new dockerfile and copy the folder you want to mount into the /home/ of the new docker. You can start the docker file using the docker you already have; thus the first line of the dockerfile will be something like “FROM r4:v.0.0.1”.

B: Run cellranger with a shell command to execute mat2csv interactively

1. Run the docker mounting the file (point A.3)
2. Go to the home of the docker and use nano to edit a bashcommand.sh file. In this file you will write down the cellranger command of step A.5.
   It’s important to remember to change the name of the output file, otherwise it might overwrite the old data.csv file. In this case it’s not a problem, but it could be for other types of analysis.
3. Then run the `chmod +x ./bashcommand.sh` to make it executable. The ./ indicates the location of the file bashcommand; in this case the file is located in the home, thus ./ .
4. Then just type ./bashcommand.sh to run cellranger; you will get a warning, it’s normal.



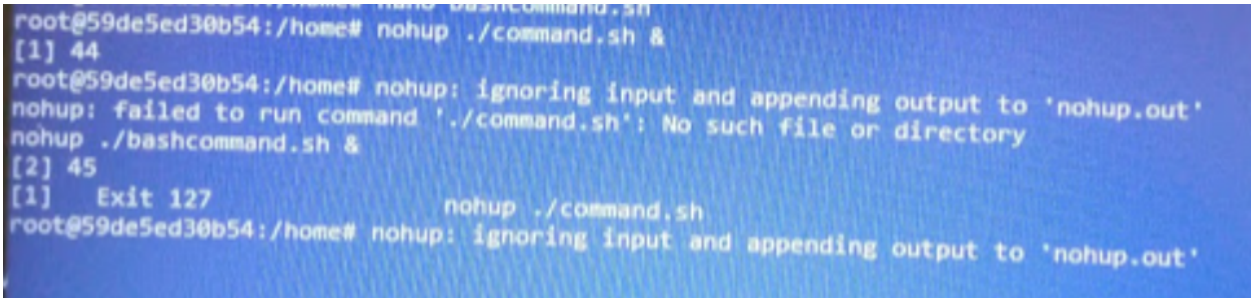
5. EXIT AND COMMIT

C: Run cellranger with a shell command to execute mat2csv in background

In point B, the cellranger was run in an interactive way, meaning that the terminal was frozen and no other command could be used. To avoid this, it is possible to run commands in background to make the terminal available while doing the process. This is done with the nohangup command.

Basically, you just add the nohup before the name of the bash command file as below; the & means “in the background”.

`nohup ./bashcommand.sh &`



Here I’m stupid and I wrote the name of the shell command file wrong 🙂

EXIT AND COMMIT

D: Run cellranger from the terminal, outside of the docker

You can also execute the cellranger command without entering the docker, meaning that you will run it while you run the docker. Basically you use the “docker run” command to execute the file.

First of all, modify your bashcommand.sh file like this:

```
#!/bin/bash
/bin/cellranger-6.1.2/cellranger mat2csv /home /home/data/data.csv
```

EXIT AND COMMIT THE CHANGES

Then use the following command from the terminal:

```
docker run -i -t -v  path/of/the/matrix/folder:/home r4:v.0.0.1 /bin/bash /path/of/the/bashfile.sh
```

The /bin/bash part basically “activates” the bash that searches for a shell command in the location you put after.

```
docker run -i -t r4:v.0.01
apt-get install wget
cd home
cd bin
wget -O cellranger-6.1.2.tar.gz "https://cf.10xgenomics.com/releases/cell-exp/cellranger-6.1.2.tar.gz?Expires=1649276952&Policy=eyJTdGF0ZW1
gzip -d cellranger-6.1.2.tar.gz
tar xvf cellranger-6.1.2.tar
cd home
nano ~/.bashrc
                #inside ~/.bashrc file at the end add
                export PATH=/bin/cellranger-6.1.2:$PATH
                #ctrl S to save and ctrl x to exit
exit
docker commit nameinstance r4:v.0.01

1. #RUN cellranger directly on the bash
docker run -i -t -v C:/Users/Utente/Desktop/GSM4679532_P01:/home r4:v.0.01 #to mount the folder with the data
cellranger mat2csv /home /home/data.csv

2. #RUN cellrager in background inside the docker
docker run -i -t -v C:/Users/Utente/Desktop/GSM4679532_P01:/home  #to mount the folder with the data
cd home
nano name.sh
       #inside nano
            #!/bin/bash
        cellranger mat2csv /home /home/data1.csv
        #ctrl S to save and ctrl x to exit
chmod +x ./name.sh
nohup ./name.sh &

3. #RUN cellranger in background outside of the docker
docker run -i -t -v C:/Users/Utente/Desktop/GSM4679532_P01:/home -d r4:v.0.01 /bin/bash /home/name.sh
```