Data analysis – Prof. Calogero

Lesson 4

In the slide there are the steps necessary for the bulk RNA-seq analysis: the fastq is evaluated by fastQC, then it is streamed by skewer. The streamed data are passed to STAR for mapping; the BAM file contains the position of the reads generated during the mapping and, finally, RSEM generated the count-table taking into consideration the GTF file for the annotation. For each sample, I get counts. On the slide there is a wapped function, that is present in the library "docker4seq". I am unable to use it, because it requires a lot of RAM. Once I pass all this information (the folder where the fastq is located, the scratch folder in which the temporary work is performed, the adapters that I which to trim, the type of sequencing, which can be paired-end or single-end), I go through all these steps. We will start from the count table that is generated from this workflow.

<u>scRNAseq</u>

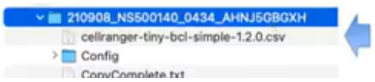
what can I do if I work at the level of single-cell? What changes comparing this analysis to the bulk one? The structure of FASTQ for single-cell is slightly different: in bulk I sequence different samples together, while in sc I have multiple transcripts associated to every sample. Fastq files are divided by samples on the bases of the index fastq file, called I1 (which contains the sample index). In the sample I am going to have 2 fastq files: R1 (the forward one) and R2 (the reverse one): R2 contains the sequence of the 3' end of each transcript present in each cell of the sample. With R2 I read the transcript associated to a specific piece of DNA. In R1 I have 2 information: one is the barcode of the cell present in the sample (if I have 2000 cells, the barcodes will differ in 2000 ways, because they refer to each cell), while the other identifier is the UMI, which is used to count the number of transcripts that are available.

In sc I divide samples with the index information (present in the I1 fastqc file), then I use R1 and R2 to collect the information of the sequence and of the cell to which the transcript of interest is associated.

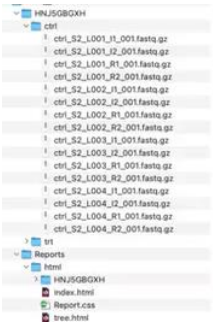
In the slide there are the steps related to the sc experiment, to which I am not interested. What I care about is the output of the sequencer, called BCL file; this file is a collection of images that contain all the information related to each nucleotide, that has been detected during the steps of sequencing. This file is converted in fastq with a specific program: with respect to what I normally get for the bulk (Illumina has a workflow that provides this transformation), in sc analysis I have a software called mkfastq, which gives me the counts. The BCL, with the information obtained from the samples, will generate the 3 fastq that I need (the index, R1 and R2). The program that allows me to manage the data coming from the sequencer is called cellranger. This program is located on the docker, and these are the steps I saw during the creation of the docker: docker run -i (interactive) -t (I can access my docker by typing) \ position of my BCL files \ genome \ docker container that I generated /bin/bash (bin/bash is not used). Once inside, there are some information, the most important of which is the comma-separated file, which describes the name of the experiment and the index associated to the experiment. Given these 2 elements, the software will organize everything in order to give me the 3 files that have been described before. In the example there are 2 samples, each one associated to an index. The code SI-TT-A2 is the index of the control, while SI-TT-B2 is the index of the treated sample, but the BCL file contains everything together. What does mkfastq do? It separates the control with respect to the treatment. In R1 there are different information: one of it is the code of the cell. The sequence that I read contains the transcript and other information, such as the cell from which the transcript derives. In R2 there is the transcript and its amount. If I read a transcript for 10 times, this means that the transcript is present for 10 times in the cell. Since the

material I work with is so tiny, there are a lot of PCR artifacts. In order to solve this problem, I have to look at the UMI (Unique Molecular identifier), which is made of 10, totally random nucleotides. They should be all different within the same cell and associated to a specific transcript. Example: my transcript is p53 and I have 9 p53 mapping on the R2 and 3 different UMI (each UMI is repeated 3 times) → each read of the transcript has 3 artifacts, and the real number of transcripts is 3. R1 and R2 contain all the information required for the analysis. On the other hand, if I read p53 for 9 times and each transcript has a different UMI, in the cell there are 9 p53 transcripts.

Structure of the BCL file



The first name is the name of the folder. There are 2 folders (ctrl and trt), and in ctrl there are R1, R2, I1 and I2 (I1 and I2 are the identifiers of R1 and R2). Given one index, I can run only 96 samples, but if I have 192 samples, the material on which I work is huge, so 2 indexes are generated. Index.html recalls the information related to the quality of the sequence: in this run and in the ctrl file there are 4 lengths (the flow cell is divided in 4 lines, and the information of the control sample are obtained from 4 lines).



Now, the fastq file must be converted into counts with some adaptations, because of some features of sc analysis. The RNA has a poly-A; during library construction, the oligo-dT is added on one side. After the first strand has been constructed, the second one must be built with an adapted called TSO, which allows to make a full cDNA. In fact, during retrotranscription, I have to force the retrotranscriptase to attach a set of C to the 3' end. The TSO is made of a stretch of G + a known oligo. During library preparation, the oligo-dT will be on one side, while the TSO will be on the other side. In sc many things can happens, and fragments can be small; frequently, the unknown piece of cDNA can have the 2 elements on both sides → the 1st thing I have to do is the trimming: I remove the poly-A and the TSO, in order to get the unknown piece of DNA. This is what is read by the R2 (that contains the transcript). Basically, I have to clean up all the information in order to align my transcript to the genome. Tagmentation uses yeast trasposome, which cuts the DNA in different positions. During trimming, I cut the DNA in a position and I attach the elements present in R1. If the fragment is too small, tagmentation will not occur and the R1 will be attached after the TSO. The reason why I clean the TSO and the poly-A is because I only want the 3'-end of the transcript.

After the clean-up, I take the cDNA and I use STAR to alignment. Alignments are "good" only when they drop in exonic regions at least for 50% of the sequence. Since the piece of DNA is small, it is possible that the fragments drop on exonic and non-exonic positions. If many fragments map on the non-coding region and only one maps on the coding region, this one is considered, while the other ones will be discarded. Multiple mapping works with mapping on exons. Once I know that my piece of cDNA is present on a coding area, the software checks if this area is annotated or not. General approach: I map the reads on the genome → I collect the reads that are associated to p53 in a specific cell → I wait the annotation of the transcripts to finish (example: I obtain 9 p53 fragments) → I move to the other side and check that p53 belongs to the same cell, so I have to look at the R1: I see how many UMIs are associated to p53.

Also on the R1 there is a control that must be done: I have to be sure that there are no nucleotide errors associated to my sequence. I have a barcode containing 2 sequences, and a barcode having 1 sequence only. These barcodes are characterized by similar nucleotides, except one → they are combined in one barcode, which is the most expressed one.

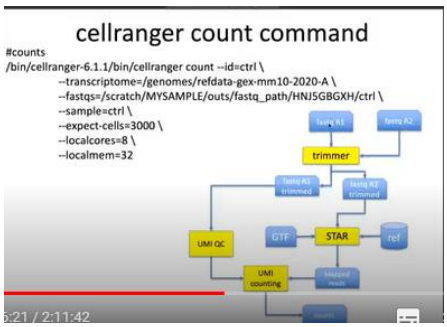
Example: I have 100 barcodes AATTCC and 1 AACTCC barcode. The first barcode, being very frequent, is going to be selected, because the C is an error. This way, I avoid to have an over-estimation of the expression. UMIs are taken together and sorted by similarity. If there is a mismatch over 10 nucleotides, the most frequent

group will be assign, without mis-counting the gene associated to the "wrong" barcode. This check is done by the UMI.

In 10XGenomics, barcodes are 10.000 thousands, and they are known; on the other hand, UMI are unknown and random, and mismatch can happen.

During mapping, samples are splitted and mapped one at a time: I map the control → results of the control; then I map the treated → results of the treated. During the mapping of the control I can get 3000 cells, while during the mapping of the treated I can get 5000 cells. 2 fastq files are generated → mapping with STAR → I get how many times p53 is detected → reads are paired, I take the UMI in the R1 → I align the fragments → if there are mismatches, I correct them to the most representative group. The R1 is modified to correct this error. After this kind of adjustment, R1 is attached to R2 → I count how many transcripts are associated to a specific cell.

Cellranger, which is designed to do the fastq, has another function called "count", which does the mapping. The counting element generates the table on which I will work. The main difference between bulk and sc analysis is that in bulk each sample is one sequence (it is a vector of counts: my sample 1, associated to a patient, has 20.000 expressed genes, but there are also multiple samples that must be analysed), while in sc analysis the situation is different: I cannot directly compare the control with the treatment because they derive from independent experiments. So, I analyse what is in the control, what is in the treatment group and what are the differences between the two. In general, in bulk I search for molecular events that discriminate my experimental condition (in a cell line with and without treatment, I look at the genes that change in the 2 conditions by analysing the transcriptome), while in sc analysis the situation is different: I have a heterogeneous population with the treatment and another one without the treatment. During treatment, a subset of cells is going to be killed. What I investigate is how the ratio between the 2 populations changes. So, in the bulk I look at genes changing (but I don't know if these genes belong to a specific subgroup of cells present in my sample), in sc I see if there is a changing in the ratio between the subgroups I have. Example: I take PBMCs, and I do a treatment on them → do I induce the proliferation of DCs? When I do the bulk, I will see an increased expression of genes associated to the activation of DCs, even though I don't know how many cells are changing. When I do sc, the population that can be labelled by the transcriptome of the DCs will increase after the treatment. Sc looks at the representation of populations in a sample, while bulk looks at differences in genes present in the bulk sample (which is a mix). What is the advantage of sc? If the event is related to a small population, this will not be seen in the bulk because the events happening in more numerous populations will mask this small event, but this does not happen in sc.



I will not run the command! This is a summary of the things that have been discussed. There are R1 and R2, that are trimmed. The trimmed R1 goes to the UMI QC, which is used for the counting. R2, instead, goes to STAR, I obtain the mapped read and the UMI counting is performed after the quality control of the UMI and, combined with the reads, counting can occur. This is done with the method "count". Mkfastq generates the separation between the control and treatment group + R1 and R2 for each sample. In bulk, mkfastq is not run because the fastq file is directly obtained. Sc is more tricky than Illumina, even though it allows to work with more than 3000 cells in an experiment; multiple samples can be mixed, and cellranger separates them into subgroups.

What is the output? What I am interested in is the filtered_feature_bc_matrix: this is the folder that contains 3 files → barcodes.tsv.gz (the file containing cell barcodes), features.tsv.gz (the name of the genes present in a table: given a table, barcodes are the columns, while the features are the rows) and matrix.mtx.gz. Why is everything organized this way? During sc analyses, only a few genes are detected as expressed in a cell,

and some are not detected because they did not attach to the poly-A or because they are below the limit of resolution of the method. If I have 3000 cells and in each cell there are only hundreds of genes that have an expression which is detected as different from 0, all the others have an expression equal to 0. Thus, the table that is generated is very large and it is made of 0 → single cell table have >90% of elements made by 0. So, there is the necessity to make this table more compressed, without 0: this is what is present in the file "matrix.mtx.gz". The compressed matrix is called "sparse matrix", because there are no 0 there.

Let's look at the organization of the file "matrix.mtx.gz": the first column is represented by the rows (which are 36601 in toto), then there is the total number of columns (585) and the counts (1044567). For example, the 36572nd row has 33 counts in column 1. In this format, 0 is never shown → the matrix is very tiny because all the zeroes are eliminated. At the end, this matrix file must be converted into a comma-separated file, which occupies more memory than the compressed file because the csv file contains the zeroes too.

The file "genes.tsv.gz" provides an old 10Xgenomics output: nowadays, 10X works with the "multimodal" technology, which means that also antibodies are included in the analysis (not only proteins are considered). These 3 files can be used by cellranger with a method called mat2csv, that converts the sparse metric into a sparse one (the csv).

What about bulk analysis? After the counting, I can obtain a multiQC information of the results. This is the output of the count analysis: when I run cellranger, I get an HTML file that contains this information. Let's have a look at it.

- The estimated number of cells detected (3623 in this case) depends on the cells that I analyse: if I work with 10000 cells, normally I get 2/3 of data. In this experiment, probably, they used 5000 cells.
- The number of reads per cell is the number of sequences associated to each barcode of the cell. It is a mean. The ideal condition for sc analyses is 50000 reads per cell, because it guarantees a good coverage. The detected genes are defined by the number of UMI associated to each gene: with 45000 reads, there are 1432 genes detected in a cell with at least 1 UMI.
- Median genes per cell: the expression of a gene is associated to one UMI.

I expect that the number of reads per cell is between 20000 and 50000: at this point, genes can be detected and their number is good.

Another information I get from the quality control is related to sequencing. The most important part is the "Q30 Bases in RNA Read": the number of barcodes in the RNA reads that have a good quality (Q30 is the score. The highest number is 40) is 91.6%. This number simply tells me that the quality of sequencing is good.

Another information is related to mapping: the 93% of reads is mapped to the genome (very good); the reads mapped confidently to exons are 67% (relatively good); in the non-exonic portion of the genome there is the 64.7% of reads.

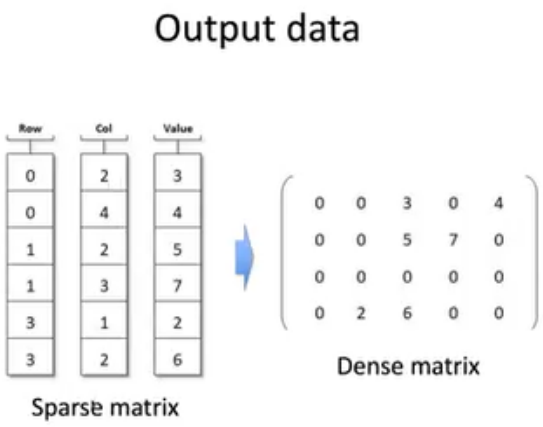
Opposite situation: an experiment that did not come well

The upper part is the good experiment, while the lower part is the bad experiment.

- The number of reads is not bad (21,625).
- There are 153 genes per cell: even given 21,625 reads per cell, only a few of them match the exonic part → the quality of the analysis was so bad that it did not allow me to catch a good number of genes. 153 genes are very few.
- The Q30 is 91% in the good experiment, while it is 59% in the bad experiment. This example comes from pancreatic samples, which are very difficult du study with sc analyses.
- Reads confidently mapped on the transcriptome are 64% in the good experiment, but only 38% in the bad one. Most of the reads do not map on the transcriptome and they are lost.

Differently from the bulk, I do not look at the experiment itself, but I look at how the cells behave and at the number of genes that are detected. What is the output? In the bulk, for each sample, only one row of counts is associated to the gene; in sc analysis, for each sample I have thousands of cells, and the expression of their genes is detected for each cell. In bulk I have a vector for each cell, while in sc I have a matrix for each cell. In the bulk I use multiple samples for my analysis, while in sc I focus on one experiment at a time.

Practice



The rows are the features, columns are the barcodes, and the matrix contains the value. On the right, there is the dense matrix (there are zeroes, the row is the cell and the column is the gene). I need to convert the files into the dense matrix to perform my analysis.

The exercise is a folder containing the files barcodes.tsv.gz, features.tsv.gz and matrix.mtx.gz

I have to use mat2csv to convert the sparse matrix into the dense one. I must have cellranger, which works on Unix only. So, I will have to download locally cellranger → I will put cellranger into my docker (the one I created, containing R inside), from which I will mount into the folder where I insert my files.

Cellranger is available as .tar.gz: "gz" is like "zip". In order to unzip the file, the command is gzip -d *name of the file*. Tar is a unique file containing all the folders that I want to unpack. While zip compresses all the files into a folder, tar creates a set of folders, that I have to locate in a specific position of the Unix system. Once the tar is created, the command tar xvf *name of the file* → I unpack everything and get a folder having this name. the .tar file disappears. What does "xvf" mean? X = execute; v = verbose (= it tells me everything it does on the terminal); f = file. The opposite command is xcf (where c means "compress"), which gives me the compressed material (.tar).

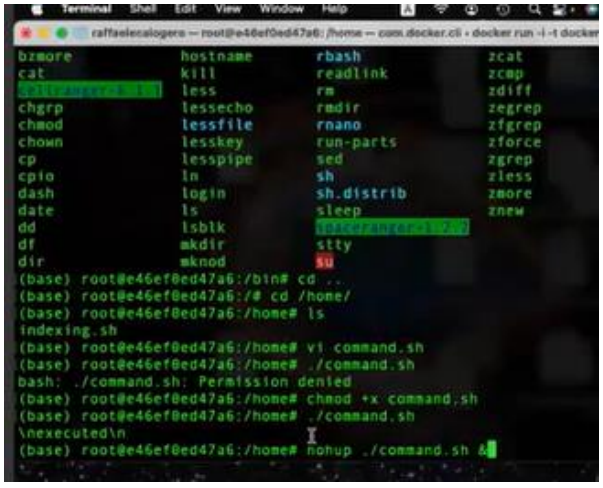
Homework 1st part

1) Click on the link where I can find cellranger: I select the whole command (the wget one) and cut and paste it on the docker environment.
2) Enter the docker: docker run -i -t *name of the docker* (in my case, r4:v.0.01)
3) Ls to know what is inside.
4) I have to put cellranger into bin: cd /bin → I run the wget command by pasting it. /bin always exists on dockers. In order to make wget run, I have to run apt-get!
5) Use gzip -d tp unfold the compressed file → it will take some time

6) Use tar xvf to unpack the folders
7) Commit the changes

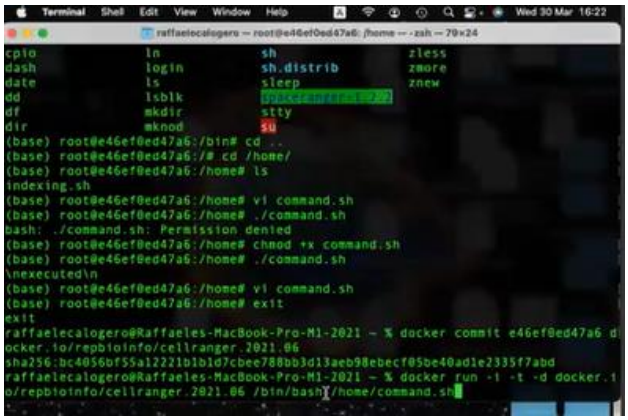
2nd part

1) Read the instructions present in the link for the command cellranger mat2csv folder where the barcode, matrix and feature files are located → I will get a dense matrix.
2) To do so, I have to mount a folder. I can use the -v (?) or, starting from the docker, I create a new Dockerfile and I copy the folder of interest into /home. This 2nd strategy, however, is not the best one.
   Example: my 3 files are in the "data" folder: I type /data/data.csv → I run cellranger with the method mat2csv, I know that my input material is in /data and I would like data.csv to be written into the data folder.
   This step may require some minutes.
3) Write a shell command to execute mat2csv: this can be done in background into the docker or using the docker run command.



Example: the prof moves to /home/ → with nano command.sh (this is an example of the name) I enter the Word interface → echo (=whatever is written here is shown on the screen) *\nexecuted\n* (\n indicates a character change)

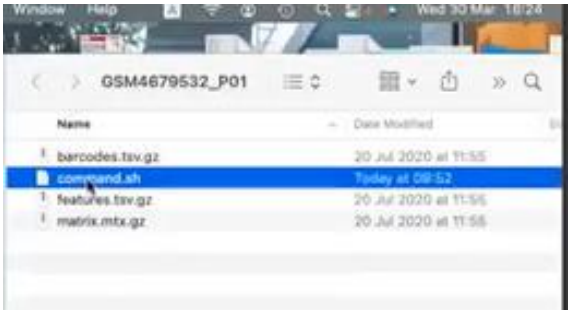
I exit from nano and, on the terminal, I write chmod -x command.sh → ./command.sh (even if the character is not changed). Problem: if command.sh is run directly, I cannot do anything before the run has finished. To do it in background, my terminal remains available. The command is nohup ./command.sh & (nohup = no hang up = execute this command even if I close the terminal). & is used to indicate the background (whatever information is running will be written on the file called nohup.out). 3rd possibility: I can use docker run to execute this file.



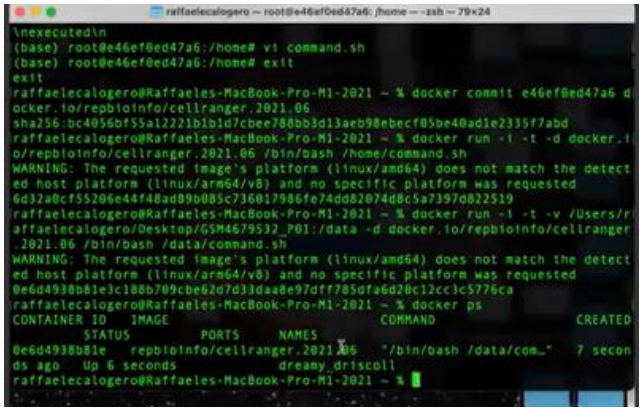
Example: I am in the docker container → exit → docker commit code of the docker → whatever is modified is now applied to this situation. -d = go in background. After the string I add /bin/bash /home/command.sh

I am outside of the docker, but using the bash interface I execute the command.sh that is present into /home/. This information run in background and data.csv will appear in the folder.

This is the organization of my folder (he does not show the content of command.sh because I have to do it by myself).



On the terminal, the prof mounts the folder data on the docker, I execute docker in background (-d) using the bash, and I execute /data/command.sh → docker ps shows me that is running in background:



at the end, results will be in the folder data, where there is also the command to generate these results.

Now, if I move to the folder data, I can see the difference in dimension of the 2 files.

In the bulk the question is “what are the genes that change between the treated and the untreated condition?”. The next step will be to evaluate the quality control of the experiment. My initial part answers to the question “is my experiment technically well done?” → yes: I have tables for the bulk and a unique table for the sc.