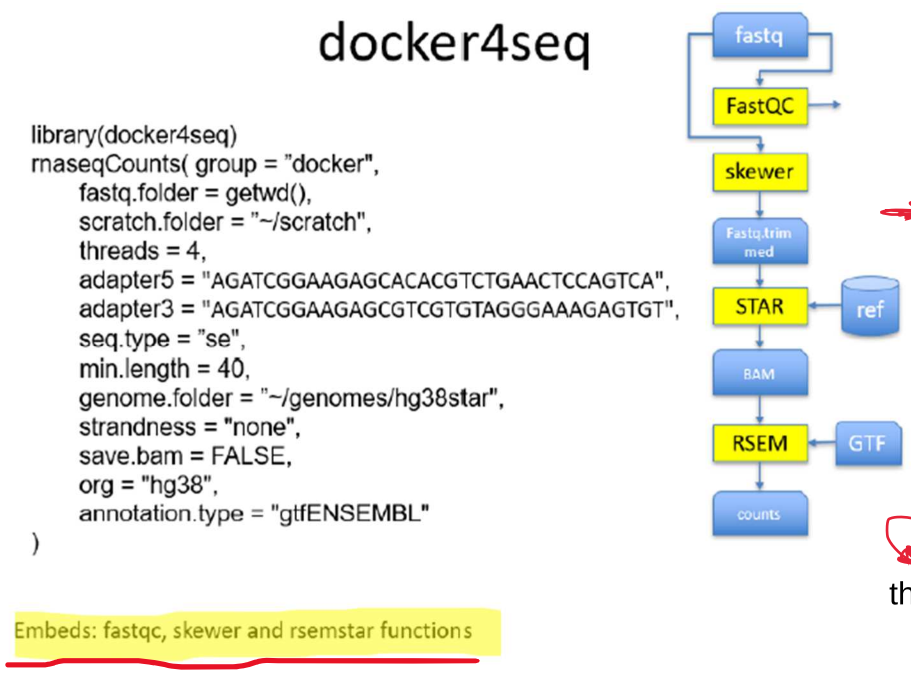
This is a summary of the code of last lecture. The **docker4seq** package allows to transform FATSQ files into actual RNA-seq counts. Overall the docker4seq package includes:



docker4seq

```
library(docker4seq)
rnaseqCounts( group = "docker",
      fastq.folder = getwd(),
      scratch.folder = "~/scratch",
      threads = 4,
      adapter5 = "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA",
      adapter3 = "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT",
      seq.type = "se",
      min.length = 40,
      genome.folder = "~/genomes/hg38star",
      strandness = "none",
      save.bam = FALSE,
      org = "hg38",
      annotation.type = "gtfENSEMBL"
)
```

Embeds: fastqc, skewer and rsemstar functions

1. FASTQ file analysed by **FastQC**;
2. Trimming by **Skewer**;
3. Genome mapping by **STAR**;
4. **BAM** file is generated (containing positions of reads on the reference);
5. The BAM file is processed by **RSEM** for annotation/count table generation (using a **.gtf file** for annotation). The final output is the count table.

the cpm file we normally start from?

This code requires a lot of RAM on you PRC in order to be executed, thus for the exercises we'll already have the final count table given.

SINGLE CELL RNA-SEQ: FASTQ 10x GENOMICS

Until now we referred to bulk RNA-seq analyses. What about scRNA-seq?

- **Bulk RNA-seq** → multiple samples are sequenced together (only distinction between different samples is made, not between single cells in each sample).
- **scRNA-seq** → single cells from multiple samples are sequenced and kept distinct.

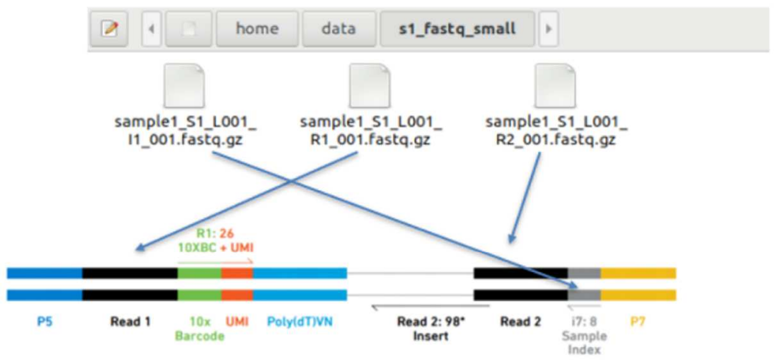
In order to distinguish transcripts based A) on the sample and B) on the single cell of origin there are specific tags:

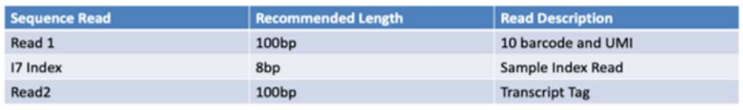
- Index identifier → to distinguish transcripts based on the sample
- 10x barcode → to distinguish transcripts coming from the same cell
- UMI → to distinguish each starting transcript from other ones and then count them

When doing scRNA-seq, 3 FASTQ files are obtained:

- I1 file → it contains the sample index sequence (read), which approximately is 6 nucleotides.
- R1 file → it contains the info related to the 10x barcode and the UMI (so both cell identifier and transcript identifier information). Basically, the info about the 5' of the sequence.
- R2 file → it contains info related to the mapping (to the unknown DNA sequence, the 3' of the transcript): it tells to the software which piece of cDNA is associated to a specific sequence.

In scRNA-seq the steps are:





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

1. Samples divided using the index information stored in the FASTQ file I1.

2. Using R1 and R2, the info related to the transcripts' sequence is connected to the different identifiers in order to univocally call each cell and transcript.

During the process there's also a PCR step as the starting amount of RNA is tiny, but this could generate artifacts and that's why the UMI exists (if a transcript is better retrotranscribed and amplified than others, you'll know thanks to the count of UMI and you don't consider it as originally more present within the cell): if the p53 transcripts mapping on the *TP53* gene are 9, but you find only 3 UMI associated to it, then the actual number of p53 transcripts is 3.

10x GENOMICS CELLRANGER

The output of the single cell sequencer is a **BCL file**. Then this file is converted into the 3 FASTQ files seen before by means of the **mkfastq** software. All this is done within the workflow of the sequencer (and it's sequencer-specific, in this case the 10x genomics is the most used).

The mkfastq program is wrapped, together with other functionalities, within the wrapper program **Cellranger**. The Cellranger mkfastq command line is the following:

```
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
```

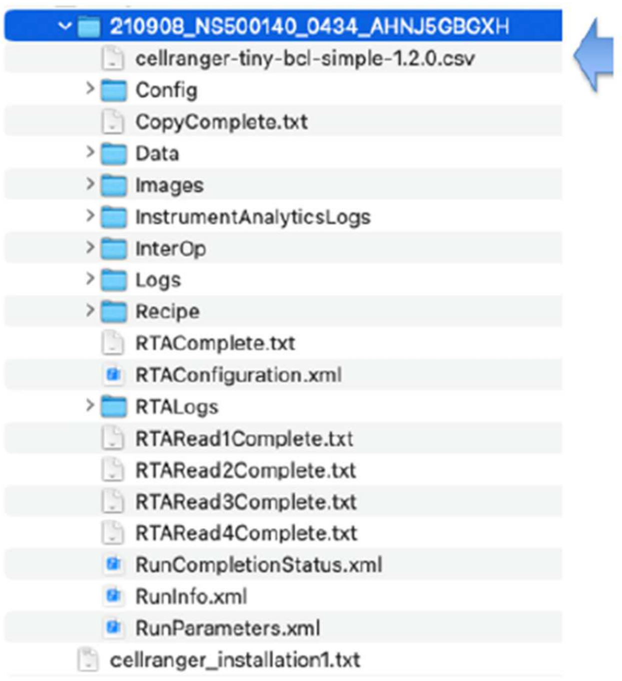
Cellranger has been located in a docker: this way it can be run from every PC. The first part of the command thus is needed to run the docker image and have the container with Cellranger. The `–v` option after `–i –t` is followed by the location of the BCL file and it's needed to write where the BCL file should be mounted in the Docker (for example, `/home`). The 4th row is the path of the docker that contains the Cellranger mkfastq command.

Once you've run the cellranger mkfastq command, a **Comma Separated File** (**.csv**) is obtained.

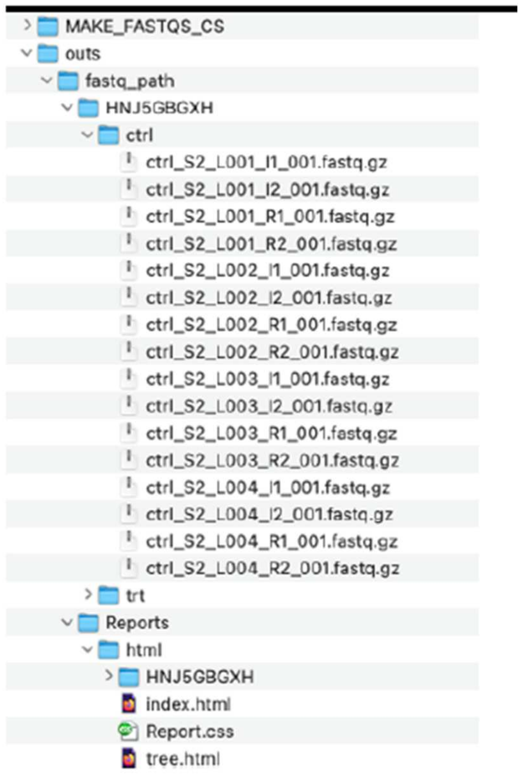
Remember that the BCL file given by the sequencer contains information of all the samples together. Each sample is associated to a specific code in an index file:

- *,trt,SI-TT-A2
- *,ctrl,SI-TT-B2

Where * is for the lane, trt/ctrl is the sample, and SI-TT-A2 is the index code. First, mkfastq separates treated (trt) from control (ctrl) samples: the output is the FASTQ file I1. Therefore, each sample will have a I1 file. Later on, each sample will also have a R1 and R2 file. All of the information is stored in the .csv file.



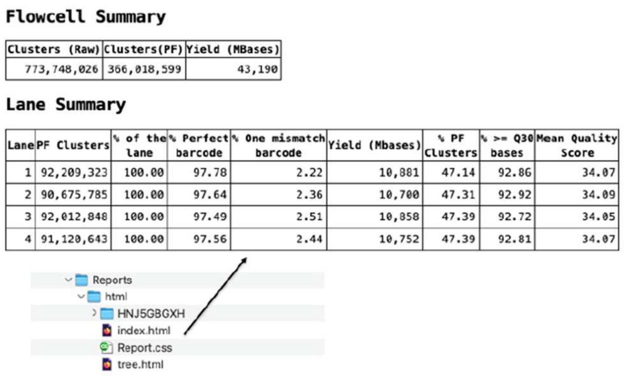
```
v 210908_NS500140_0434_AHNJ5GBGXH
    cellranger-tiny-bcl-simple-1.2.0.csv
  > Config
    CopyComplete.txt
  > Data
  > Images
  > InstrumentAnalyticsLogs
  > InterOp
  > Logs
  > Recipe
    RTAComplete.txt
    RTAConfiguration.xml
  > RTALogs
    RTARead1Complete.txt
    RTARead2Complete.txt
    RTARead3Complete.txt
    RTARead4Complete.txt
    RunCompletionStatus.xml
    RunInfo.xml
    RunParameters.xml
  cellranger_installation1.txt
```



```
> MAKE_FASTQS_CS
v outs
  v fastq_path
    v HNJ5GBGXH
      v ctrl
          ctrl_S2_L001_I1_001.fastq.gz
          ctrl_S2_L001_I2_001.fastq.gz
          ctrl_S2_L001_R1_001.fastq.gz
          ctrl_S2_L001_R2_001.fastq.gz
          ctrl_S2_L002_I1_001.fastq.gz
          ctrl_S2_L002_I2_001.fastq.gz
          ctrl_S2_L002_R1_001.fastq.gz
          ctrl_S2_L002_R2_001.fastq.gz
          ctrl_S2_L003_I1_001.fastq.gz
          ctrl_S2_L003_I2_001.fastq.gz
          ctrl_S2_L003_R1_001.fastq.gz
          ctrl_S2_L003_R2_001.fastq.gz
          ctrl_S2_L004_I1_001.fastq.gz
          ctrl_S2_L004_I2_001.fastq.gz
          ctrl_S2_L004_R1_001.fastq.gz
          ctrl_S2_L004_R2_001.fastq.gz
      > trt
    v Reports
      v html
        > HNJ5GBGXH
          index.html
          Report.css
          tree.html
```

On the right, I1 FASTQ files of ctrl samples are shown. In this case there are two indexes files: I1 and I2 → this is needed when dealing with a lot of samples. With only one index file, you can run 96 samples at max.

Other than the I1 file, the **index.html** file is created (which keeps record of the quality of the sequence). The L001-L004 refers to the lane: the flowcell is divided 4 lanes **(the transcripts from the same sample are spread in all the 4 lanes)**.



Flowcell Summary

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

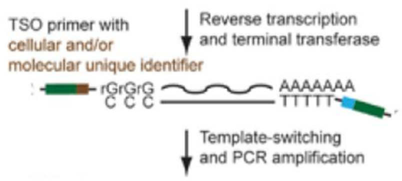
Lane Summary

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

```
v Reports
  v html
    > HNJ5GBGXH
      index.html
      Report.css
      tree.html
```

The FASTQ file should be then converted into counts (as with bulk RNA-seq analysis).

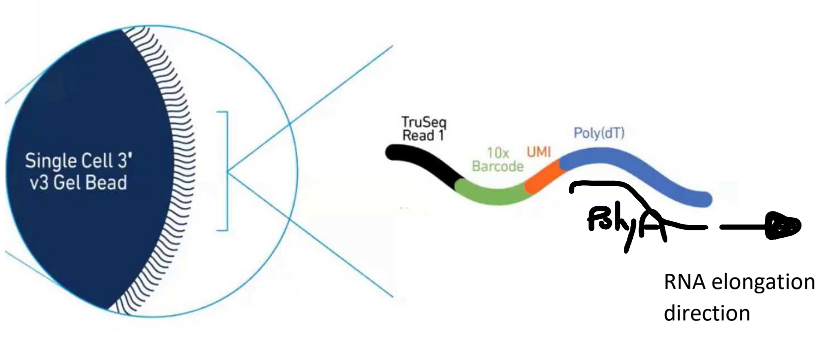
RNA libraries for scRNA-seq analysis, thus cDNAs, are built using an oligo-dT primer (to capture the polyA tail of mRNAs) and then a specific adapter called TSO to have also the second strand retrotranscribed → polyG + known oligo sequence (30 bp overall): this will anneal to a polyC attached to the 3' of the other strand (the one deriving from the oligo-dT primer). In libraries prepared this way, **each cDNA will have the oligo-dT sequence at one end and the TSO on the other end**. Short RNA fragments will likely end up with both these artificial pieces at their ends, that have to be removed before proceeding with the analysis by means of the trimming procedure.


TSO primer with cellular and/or molecular unique identifier
Reverse transcription and terminal transferase
rGrGrG
C C C
AAAAAAA
TTTTT
Template-switching and PCR amplification

After the trimming, both the TSO and oligo-dT are removed and the remaining unknown sequence is then stored in the R2 file. Also this step is done prior to the alignment.

mRNAs are captured by beads that have on their surface these elements (all together these go after the name of R1, indeed the R1 FASTQ file contains their related info):

- **Oligo(dT)** → to capture polyadenylated RNAs
- **UMI**
- **10x barcode** →
- **Sequencing primer (Primer Read 1)**
- P7 (for PCR step)



TSO is attached once the first strand (from the oligo-dT) has been retrotranscribed. **TSO should be removed prior to alignment but also to sequencing** → during the trimming, but the specific name is **tagmentation**. This way the 5' end is let free to attach to the R1. Every time tagmentation goes wrong (common with short trnascripts), the R1 attaches over the TSO.

STAR ALIGNMENT

Now that the trimming has been carried out, cDNA fragments are given as input to STAR and aligned (the software considers an alignment as good when at least 50% of a read is aligned). These cDNA reads obviously will only align to exonic regions. Each sample is mapped separately since it is distinguished based on its R1.

MAPQ ADJUSTEMENT

As cDNA fragments are generally short, they will likely align to multiple regions, however only one of these will fall on an exon (no multiple mapping on exons, problem solved).

ANNOTATION

Once the cDNA read has been aligned to an exonic area, that area will be assigned to an annotated gene. Counting can be now performed.

COUNTING

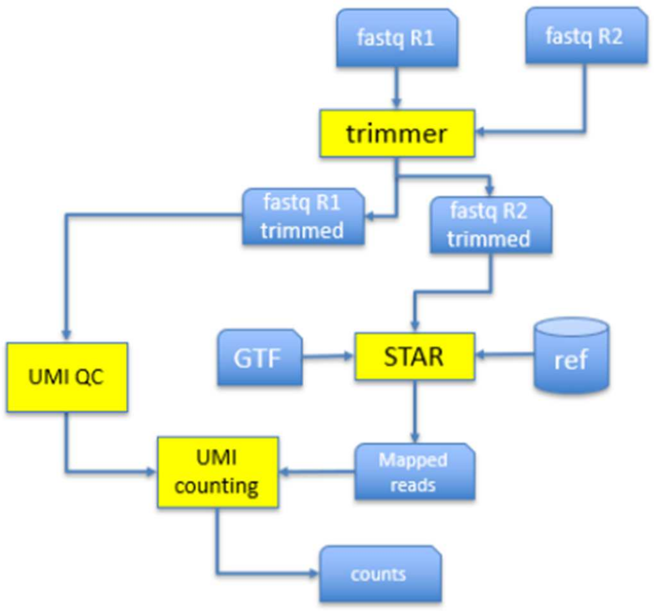
Considering one cell at a time, real reads (those that are counted) come from different transcripts within the same cell: same 10x barcode but different UMI. The count of course doesn't consider the cell or transcript of origin, but then you can retrieve these info by looking at the associated R1: based on this, transcripts from the same cell are pulled together.

What if the UMI sequence has been wrongly synthesized (since this is sequencing-by-synthesis)? UMI barcodes are grouped based on similarity before counting: if there is a UMI clearly less frequent than a very similar, different UMI, then the two are kept separated to avoid to count non-existent UMI (e.g., there are 100 AATTCC UMI and 1 AACTCC UMI: the second is a sequencing error: thus this point mutations are corrected and the UMI becomes as the actual one). Specifically, the threshold accepted for mismatches is 1/10 nucleotides. 10x barcodes (those for cells) are ~300.000 while UMIs are way more obviously (as UMIs are built as random sequences). Since you know the number of 10x barcodes, the procedure applied to the UMIs to avoid error while counting is not done.

Overall, let's assume to analyse 300 000 cells of a control sample. This sample will have its own I1 FASTQ file that distinguishes it: this is the first step of the analysis. Then, using the R2 file, the reads coming from this sample are mapped on the genome by STAR. Thanks to the UMI tag present in the R1 file it is possible to count the actual reads and remove the PCR artifacts.

With **bulk RNA-seq analysis**, each sample contains the expression of a certain number of genes (these expression values are an average that considers all the cells in the sample). Overall, there are several samples to analyse, for instance control and treated replicates of a certain cell line. The aim is to look for molecular events (e.g., different gene expression) that distinguish the different experimental conditions (ctrl *vs* treated) to highlight which genes do modify their expression upon a treatment.

With **scRNA-seq analysis**, you can't compare control and treated as they're analysed in different moments (2 independent experiment in 2 different times). In this case the aim is to highlight subset of cells with different response to a condition within the same sample (ctrl or treated), which would be masked in a bulk experiment. In bulk, changes in gene expression cannot be associated to a specific cell type (subset). In scRNA-seq, all the subpopulations within the samples are identified (clustering) and then the differences between same subpopulations of different samples are investigated. The relevance of scRNA-seq is the possibility to identify small sets of cells within a sample that are responsible for a particular phenotype and gene expression.



Pipeline to obtain counts for each gene

The **mkfastq** command allows discrimination of samples (the FASTQ file I1?) and the R1 and R2 files. Then the **cellranger count** command is used to count sequences. In bulk analysis, mkfastq is not necessary as the sequencer directly converts BCL to FASTQ files (while in scRNA-seq the output is a BCL file to be converted).

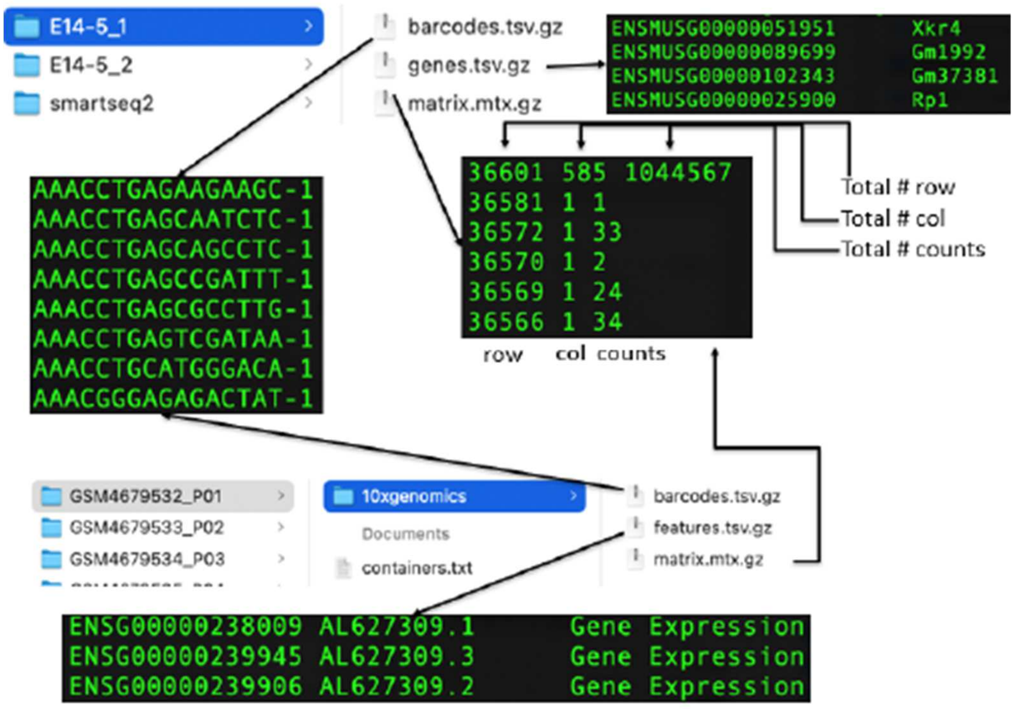
Nowadays scRNA-seq is done on up to 300 000 cells altogether, and on multiple samples at the same time (run simultaneously).

CELLRANGER OUTPUT

The output of a Cellranger analysis is a **filtered feature barcode matrix** (filtered_feature_bc_matrix) folder (one for each sample) that contains:

- barcodes.tsv.gz → containing cells' barcodes (column names: 1 column = 1 cell).
- features.tsv.gz → containing the names of the genes (row names).
- matrix.mtx.gz → sparse matrix with number of columns, rows, and counts.

In scRNA-seq analysis, only few genes are detected as expressed (there can be some issues that impede to detect most of transcripts over the sequencing process). For instance, let's consider 3000 cells that have been sequenced and 20.000 genes analysed: it could be that only 100 genes have an expression level different from 0. The table will have 3000 columns and 20.000 rows, but 90% of the table has 0 as gene expression value. We need a more compressed format, this table is too huge: a **sparse matrix** is generated, where all 0 values are removed (only non-zero values are present). To perform the actual analysis, the sparse matrix should be converted into a **dense matrix**: this is represented as a .csv file (bigger size than the sparse matrix, from MB to GB), which comprehends again the 0 values.



```
ENSMUSG00000051951    Xkr4
ENSMUSG00000089699    Gm1992
ENSMUSG00000102343    Gm37381
ENSMUSG00000025900    Rp1
```

```
AAACCTGAGAAGAAGC-1
AAACCTGAGCAATCTC-1
AAACCTGAGCAGCCTC-1
AAACCTGAGCCGATTT-1
AAACCTGAGCGCCTTG-1
AAACCTGAGTCGATAA-1
AAACCTGCATGGGACA-1
AAACGGGAGAGACTAT-1
```

```
36601 585 1044567
36581 1 1
36572 1 33
36570 1 2
36569 1 24
36566 1 34
```

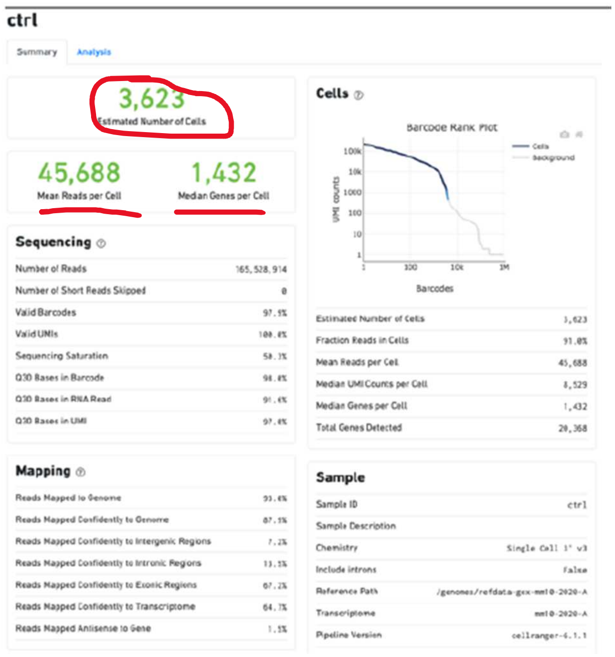
```
ENSG00000238009 AL627309.1      Gene Expression
ENSG00000239945 AL627309.3      Gene Expression
ENSG00000239906 AL627309.2      Gene Expression
```

The gene.tsv.gz contains the gene identifiers and symbols. The feature.tsv.gz is a more general name used nowadays since also other samples than genes can be included in this kind of analyses

The cellranger function **mat2csv** takes as input the folder filtered_feature_bc_matrix to make a .csv file (the dense matrix to perform the analysis).

We saw that for bulk RNA-seq it is possible to run MultiQC as a quality control of trimming and alignment. In scRNA-seq the situation is slightly different. In this case, the **cellranger count** generates a HTML file with info on the experiment quality, as with MultiQC:

The Estimated Number of Cells is lower (2/3 approx.) than the actual number of cells that were given as input. The Mean Reads per Cell indicates the average number of reads associated to a single barcode (the more reads you have, the more genes will likely be evaluated in their expression: between 20.000 and 50.000 is a good value to have enough gene coverage, while under 20.000 it is too low). The Median Genes per Cell corresponds to the number of detected genes within a cell with at least 1 UMI.



ctrl

3,623 Estimated Number of Cells

45,688 Mean Reads per Cell

1,432 Median Genes per Cell

Sequencing

Number of Reads	165,528,914
Number of Short Reads Skipped	0
Valid Barcodes	97.5%
Valid UMIs	100.0%
Sequencing Saturation	58.3%
Q30 Bases in Barcode	94.0%
Q30 Bases in RNA Read	91.6%
Q30 Bases in UMI	97.0%

Cells

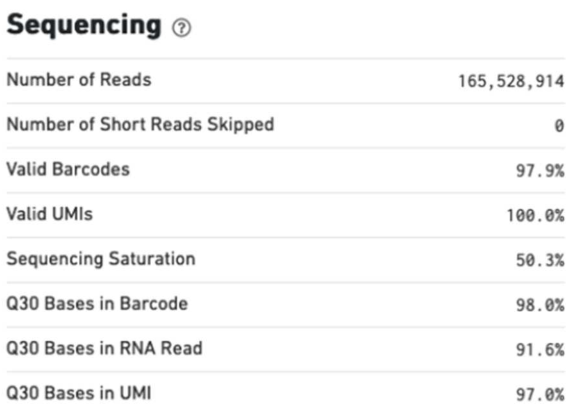
Estimated Number of Cells	3,623
Fraction Reads in Cells	91.0%
Mean Reads per Cell	45,688
Median UMI Counts per Cell	8,529
Median Genes per Cell	1,432
Total Genes Detected	20,368

Mapping

Reads Mapped to Genome	93.6%
Reads Mapped Confidently to Genome	87.5%
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%

Sample

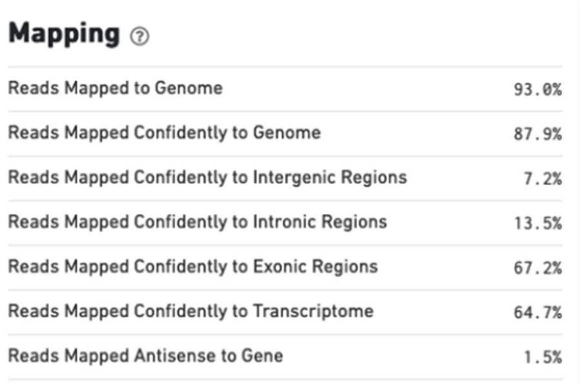
Sample ID	ctrl
Sample Description	
Chemistry	Single Cell 3' v3
Include introns	False
Reference Path	/genomes/refdata-gex-mm10-2020-A
Transcriptome	mm10-2020-A
Pipeline Version	cellranger-6.1.1



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%

Information about the quality of sequences are available. Q30 stands for quality (Phred) score higher than 30, that is the value from which a base is called with enough confidence (40 is the maximum value). In this instance, the quality of the experiment is pretty good in the barcodes, UMI, and actual read (RNA sequence).

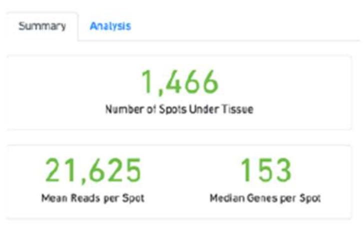


Mapping

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%

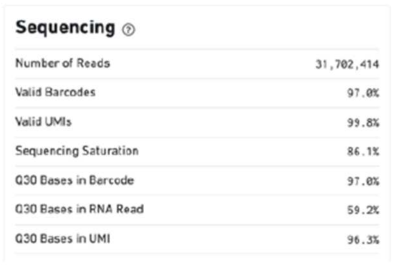
These instead are the info related to mapping quality. These are all good values except for the **Reads Mapped to Exonic Regions** (67% is quite low), but it's still a good result considering that this is a scRNA-seq experiment where the mapping is done only on reads of the 3' ends.

This below is an example of low quality outcome instead:



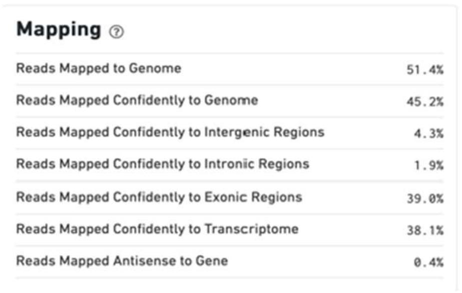
Summary Analysis

1,466	Number of Spots Under Tissue
21,625	Mean Reads per Spot
153	Median Genes per Spot



Sequencing

Number of Reads	31,702,414
Valid Barcodes	97.0%
Valid UMIs	99.8%
Sequencing Saturation	86.1%
Q30 Bases in Barcode	97.0%
Q30 Bases in RNA Read	59.2%
Q30 Bases in UMI	96.3%



Mapping

Reads Mapped to Genome	51.4%
Reads Mapped Confidently to Genome	45.2%
Reads Mapped Confidently to Intergenic Regions	4.3%
Reads Mapped Confidently to Intronic Regions	1.9%
Reads Mapped Confidently to Exonic Regions	39.0%
Reads Mapped Confidently to Transcriptome	38.1%
Reads Mapped Antisense to Gene	0.4%

The quality of the RNA sample is poor as it is demonstrated by the **Median Genes per Spot**. It is not by chance that this experiment is carried out on a tissue, where RNA extraction and analysis are very tricky indeed.

In scRNA-seq analysis compare different cells within the same sample, while in bulk RNA-seq you compare different samples.