Data analysis – lesson 3

Prof. Calogero

Alignment

What do we need to extract from the cells we analyse by the sequencer?

- Alignment of the reads on a reference genome (which provides more precise information than a transcriptome). This is done by genome mapping: I take the read and I use STAR → mapping → I use a software which evaluates the overall composition of isoforms that are present in the gene locus and converts this information in counts related to the isoforms and, then, in counts related to the gene (the counts related to the gene are nothing else than the counts related to single isoforms).

To do this analysis, I need a gtf file (annotation). The reference genome is already known, and the version used is 38 (mouse: version 11).

What does "assembly" mean? The complexity of the genome is very big: the dots indicate positions in which researchers found variations in the structure with respect to the original genome; they are spread all over the genome. Most of them are located in non-coding regions (so we do not care of them). The alignment to one genome only is not possible anymore, because all the variations must be considered. In which format are data present? They are present as fasta files (which contain the chromosome sequences) and annotation files. The 2 main repositories are Ensemble (which is the most used), whose database is gene-centric. The gene name is the reference, to which the isoforms are related. The repository is USFC, which is transctipt-centric: it does not consider the gene locus, but the transcripts located to a specific chromosomal position. A disadvantage of Ensembl is that it is highly dependent on the annotation: if it is lost, data are gone. However, an advantage of its first version, is that all the annotations are present in the Ensembl database: I am always able to reconstruct every type of annotation I need. Today, most of people work with gene annotations because they are easier to be used.

In the image I see the repository of Ensemble, and the data I am interested to do the mapping is the fasta file. Dna.toplevel file contains the standard genome + the alternative variants that have been detected over the years by genome sequences. I will not have to use it to do conventional data analysis, but it is sufficient to use the dna.primary_assembly. Another reason why I don't use the toplevel is because the generation of the reference that will be used by the aligner requires more than 32 G of RAM and contains more regions with respect to the primary_assembly. This file is enough because it contains all the coding information.

How do I do the mapping? I need an aligner that understands that a gene is split in many parts: the coding area that I see in the transcriptome is separated by many introns in the genome. A gap-aligner is able to recognise the introns. The reason why STAR became a gold standard is because of it and because it can map hundred million reads per hour in a fast and precise way; it is designed to work on gapped genes, so it is very fast when it has to map gapped genes. STARS takes together a certain number of quick methods and optimizes them in a fast way.

General idea of alignment: given a read, it is moved all along the genome until it is stopped on a specific position; (MMP = Maximal Mappable Prefixes) it cuts the read in many pieces and gives me the position of the genome in which it aligns. After the initial piece has been aligned, the other part of the read jumps on the introns and maps on another position of the genome. Then, if there is a mismatch, the second part of the read is cut in proximity of the mismatch region and maps after the mismatch: practically, the read aligns only in positions in which there is a perfect match. If the sequence is short and it has synthetic primers on the extremities, there will be no mapping: the read maps only in the position where it is aligned. The unmapped regions are discarded and not considered in the final mapping score. The number of mismatches could be very high; having reads separated in many pieces allows me to map a lot of regions of the genome.

Moreover, STAR is able to give extra information. Logfinal.out is a file of the format present in the slide, and it is read by multiqc. The file gives me the number of imput reads, their length and the fraction of reads that are managed to be mapped. Since I am performing an experiment in which I map on coding genes (exons only), I expect to have a high number of reads aligning (>90%). This file helps me in understanding some fraction lost. There are 2 important parts:

1) % of reads mapped to multiple loci: there is a fraction of reads missing the mapping position. Multiple loci mapping is related to pieces of RNA that map in different positions of the genome. Why does it happen? Because DNA contaminates RNA and it is redundant. Whenever a preparation of RNA is performed, DNA must be discarded.
2) Too short reads to be mapped: if the pieces of reads are too small, they cannot align on the genome; 3% of reads have these features, and they are mainly ribosomal RNA. For example, when I look at my whole transcript (and not only the poly-A one), I have to remove the ribosomal RNA to do the retrotranscription. If this passage does not work well, the reads that I have to map will finish in this region, since most of the RNA that will be retrotranscribed belongs to ribosomes. Let's imagine another situation: 30% of my reads are too short to be mapped on the transcript. This can be due to mycoplasma/yeast/bacteria contamination or to the contamination of another cell line. Mapping can help me in understanding the quality of my sample.

Once I have generated the mapping, this mapping goes in a BAM (?) file = binary version of a tab-delimited file in which all the information related to the mapping is stored. This information is not related to the name of the genome or of the transcript present in a specific location: BAM files are just an intermediate file containing the mapping information of the reads on specific portions of the genome. In order to associate them to different isoforms, I need another software, such as RSEM, which handles the fact that, among isoforms, there are common region. Looking at the slide, the red and yellow isoforms are the uncommon regions of the 2 transcripts. RSEM estimates the expression of the different isoforms on the basis of the specific isoforms. In the experiment, the red region has almost the double number of reads present on the yellow one → ratio = 2. Taking the blue reads, I assign them to the double number of red isoforms and the half number of the yellow ones. Basing on the specific isoform, I estimate the ratio of the isoforms, and then I assign the read ratio. Limit: the different level of coverage affects the quantification (the red and yellow reads can have a different length) → looking at isoforms implies to work at high coverage (I have to use 80 million reads per sample). Moreover, isoforms are difficult to handle: I have to demonstrate their existence and I have to demonstrate that there is a biological effect if I remove one specific isoform. The quantification is reasonable if the isoforms are known and reads are not too short.

For the annotation, in order to use RSEM, I need a gtf file. Annotation and assembly go together: in the slide there is the same location of chromosome 14; the upper one is MM9 (no genes), the bottom one is MM10 (genes present). The position of the gene present in MM10 could be a consequence of a displacement caused by a deletion; annotation should be associated to the same assembly. Files are organized in table, on the upper part of which there are UCFC files. geneID is the identifier of an isoform that is present in a specific locus. On the other hand, in the Ensemble there are more information and biotypes = coding genes (associated to proteins), long non-coding genes and short non-coding genes. The biotype allows me to identify sub-populations of RNA that have been annotated over the genome. All this information is summarized in a fastQC file.

Approach: fastQC → trimming with Skewer → mapping with STAR → counting with RSEM. Once this information is achieved, multiQC can be performed.

<u>Exercise comment</u>

- M aligned stands for million of reads aligned; ra, rb and rc have 1 million reads only. For ra there is ra1m.R1 (which is the name of the fastQC file). In all the files there are 1 million reads, so the "STAR:

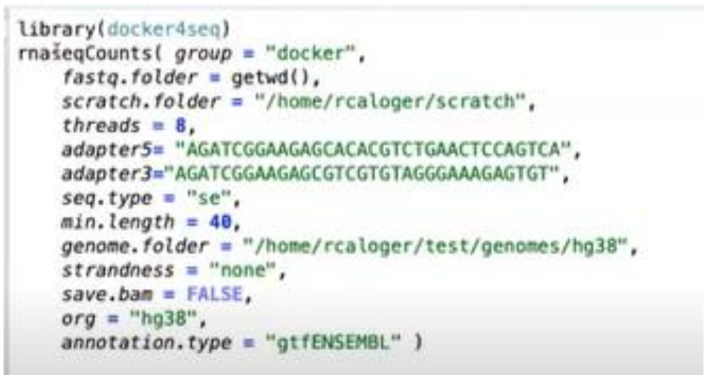
Alignment Scores" section is homogeneous. Blue = uniquely mapped; light blue = multiple loci; red = unmapped. The overall quality of the experiment is good.

- The Skewer trimming section tells me that I am trimming a few nucleotides: the insert was in the range of 95 nucleotides.
- In the "fastQC: Mean Quality Scores" I see that the reads all behave in the same way and have a high quality.
- The GC content has a strange behaviour but it is distributed homogeneously in all the samples.
- Duplication level: since I did the chemical fragmentation, some fragmentations can exist, because of the read cuts.
- Overrepresented sequences change sample-by-sample and they cannot be explained. Do not consider them!
- Adapter content: adapters are <1% and are ~75 nucleotides.

So, multiQC gives me an idea of all my samples at the same time.

This analysis was performed using the docker4seq package: it embeds in the docker tools that allow me to do sc analyses.

In R:



```
library(docker4seq)
rnaseqCounts( group = "docker",
    fastq.folder = getwd(),
    scratch.folder = "/home/rcaloger/scratch",
    threads = 8,
    adapter5= "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA",
    adapter3="AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT",
    seq.type = "se",
    min.length = 40,
    genome.folder = "/home/rcaloger/test/genomes/hg38",
    strandness = "none",
    save.bam = FALSE,
    org = "hg38",
    annotation.type = "gtfENSEMBL" )
```

rnaseqCounts is a wrap = it embeds other functions, each one doing something:

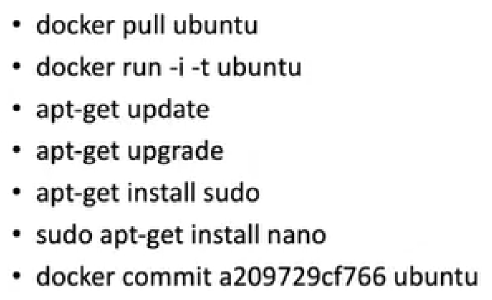
1) It runs fastqc
2) Trimming
3) Alignment and counting
4) Gives a final script.

- It runs with a docker with user permissions of my system.
- Fastq.folder = folder in which the fastq is located; if the command is run in the folder in which there are the samples, I can write getwd(): it takes the fill path in which I am located in a specific moment and in which there are my data.
- Scratch.folder = normally data are big, and they are located in low-performing disks. If the alignment is done on conventional disks, the time required is very long. Data must be copied on scratch, which must be created on the fast disk.
- Threds indicated the number of computing units that I use.
- Adapters can be found on Google, and they depend on the library I have to pair.
- Seq.type = "se" for single-end; "pe" for pair-end.
- Min.length = if reads are <40 nucleotides, they will be thrown away because they cannot be mapped on the human genome.
- Genome.folder = location where there is the reference aligner file that is ready for mapping; it is not a fasta, but it is reorganized in a binary form in order for it to be quickly accessible.
- Strandness = has 3 options: "none" (which means that I don't care if the strand is poly-A selected), "forward" and "reverse". Strandness is needed only when I don't know which strand I am studying (I put together the forward and the reverse one).
- Save.bam = the bam file is not needed: I am not interested in variants because I just count sequences present in different positions of the genome.

What is the output generated? It is gtf_annotated_gene.results: the 2nd column is the gene identifier in Ensembl = it tells me what is the code associated to a specific gene in Ensembl. 3rd column: biotype (protein_coding). TPM and FPKM are a sort of normalization, while expected_count is the output of RSEM:

RSEM gives the number of counts for each isoform, it sums them all together and gives the expected count for each gene, knowing the annotated transcript. Example: the transcript having 154 expected counts is referred to DPMI gene.

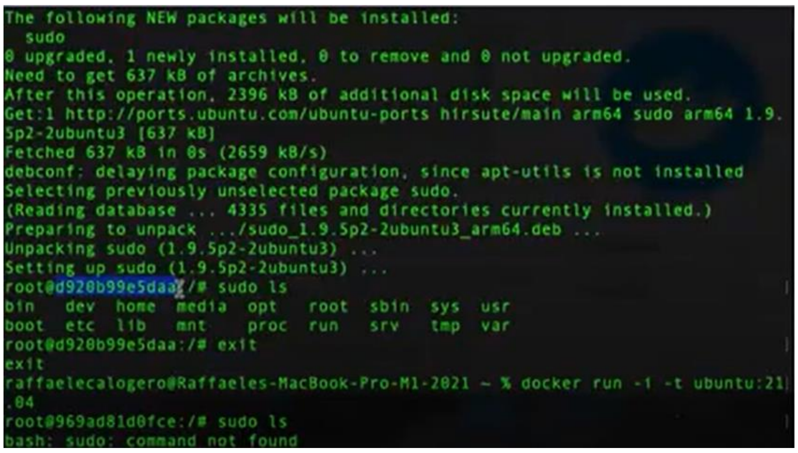
<u>How to build a docker container</u>



- docker pull ubuntu
- docker run -i -t ubuntu
- apt-get update
- apt-get upgrade
- apt-get install sudo
- sudo apt-get install nano
- docker commit a209729cf766 ubuntu

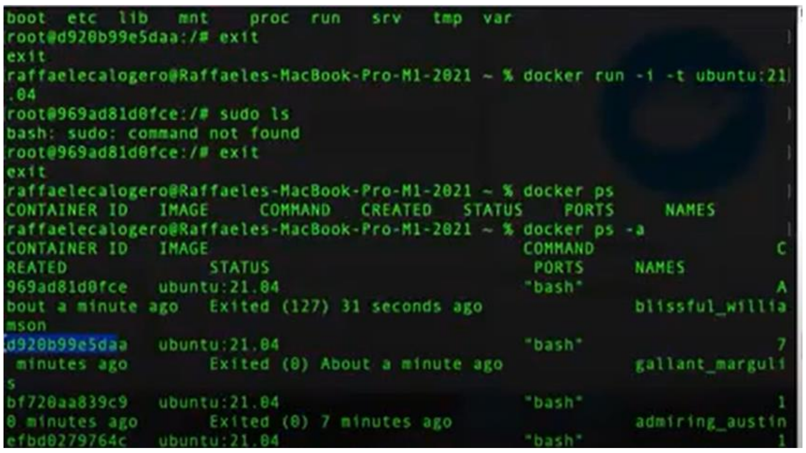
There is a repository, called "docker com", which is free to use for no-profit users. Search it on Google and try to set-up a free account there.

- Docker pull ubuntu:21.04 = I tell the computer "run my docker daemon". The computer searches a version that is already downloaded: if it is not present, it will be downloaded.
- Docker images allows me to see the images that are present in my local repository (for example there is an image called "ubuntu 21.04", even though the latest ubuntu version is 21.10; however, it is never good to work with a new version because there may be problems that people have not solved yet). Why do I create a docker? To make my life easier.
- /# gives me the docker name.
- Docker run is the command that runs the image that I want. -i = interactive; -t = I have a terminal that allows me to interact with it. Docker ps tells me which dockers are working: the one called bf…9 is the docker of the prof, which was run a few seconds ago. Ls = I see what is in my docker. A folder called "home" is where my user is. Cd /home allows me to move in home. So, docker is a virtual environment.
- Exit allows me to exit from the docker; now, if I type "docker ps" I do not see images anymore.
- Apt -get is used into ubuntu to install everything; apt-get update updates the paths of the latest versions of ubuntu.
- Apt-get upgrade is the command necessary to upgrade the version to the latest libraries. Update and upgrade have to be run the first time I start my environment, in order to make it "aligned" to the actual situation.
- The image becomes a container that is an instance of your image. The instance is a working representation of the image, which is in turn volatile. The docker is associated to a code.
- Sudo = I would like to run some commands that are applied as super user. When a software is installed on the computer, we prefer to work as sudo because any user is going to have the software installed in the right position. When I work as sudo, I can do everything (such as delete everything). What is the advantage? I still have the possibility to fix things.
- Apt-get install sudo
- Sudo ls shows me the list as super user



```
The following NEW packages will be installed:
  sudo
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 637 kB of archives.
After this operation, 2396 kB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports hirsute/main arm64 sudo arm64 1.9.
5p2-2ubuntu3 [637 kB]
Fetched 637 kB in 0s (2659 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package sudo.
(Reading database ... 4335 files and directories currently installed.)
Preparing to unpack .../sudo_1.9.5p2-2ubuntu3_arm64.deb ...
Unpacking sudo (1.9.5p2-2ubuntu3) ...
Setting up sudo (1.9.5p2-2ubuntu3) ...
root@d920b99e5daa:/# sudo ls
bin   dev  home  media  opt   root  sbin  sys  usr
boot  etc  lib   mnt    proc  run   srv   tmp  var
root@d920b99e5daa:/# exit
exit
raffaelecalogero@Raffaeles-MacBook-Pro-M1-2021 ~ % docker run -i -t ubuntu:21
.04
root@969ad81d0fce:/# sudo ls
bash: sudo: command not found
```

- Then I exit and re-enter again (docker run -i -t ubuntu:21.04) → sudo ls → output: command not found. Why? At the beginning, when I open an image, I get an instance, indicated by the code in the figure. Then I exit and open another instance (see the code before the command "sudo ls"). I exit.
- Docker ps → I don't see anything.

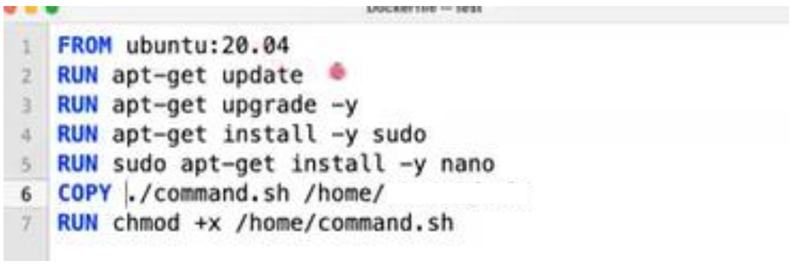


```
boot  etc  lib   mnt    proc  run   srv   tmp  var
root@d920b99e5daa:/# exit
exit
raffaelecalogero@Raffaeles-MacBook-Pro-M1-2021 ~ % docker run -i -t ubuntu:21
.04
root@969ad81d0fce:/# sudo ls
bash: sudo: command not found
root@969ad81d0fce:/# exit
exit
raffaelecalogero@Raffaeles-MacBook-Pro-M1-2021 ~ % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
raffaelecalogero@Raffaeles-MacBook-Pro-M1-2021 ~ % docker ps -a
CONTAINER ID   IMAGE                             COMMAND                 C
REATED              STATUS                        PORTS     NAMES
969ad81d0fce   ubuntu:21.04                      "bash"                  A
bout a minute ago   Exited (127) 31 seconds ago             blissful_willia
mson
d920b99e5daa   ubuntu:21.04                      "bash"                  7
 minutes ago        Exited (0) About a minute ago           gallant_marguli
s
bf720aa839c9   ubuntu:21.04                      "bash"                  1
0 minutes ago       Exited (0) 7 minutes ago                admiring_austin
efbd0279764c   ubuntu:21.04                      "bash"                  1
```

- Docker ps -a allows me to see all the instances that have been previously closed. The instance in the image is the one I am interested in, because it contains my image.
- The instance is volatile: when I close it, the information remains in the instance.
- Now I want to change the image created by my docker: docker commit *number of the instance in which there is sudo installed* (the one highlighted in the picture) ubuntu:21.04 (that is the image present in my computer)
- Now I apply what I changed to the image present on the computer: I run ubuntu → sudo ls
- Nano is a text editor that can be used without a graphical interface.
- Docker images → I copy the image ID of ubuntu 21.04 → docker tag *name of the image ID* ubuntu:21.04.raf this command will allow me to upload the new versions of ububtu on this tag. Whenever I change the tag, the tag allows me to select specific elements for the analysis. An image can have multiple names (that I chose when I use the tag). How can I remember what I have done? Using dockerfile = it is a way I can store the information I built from scratch in a docker file.

In order to create a docker, I have to create a folder in a place that does not require me a too long path (example: the folder created by the prof is called test); in the folder there is Dockerfile → open it with a text application → it is a summary of the information I did before.

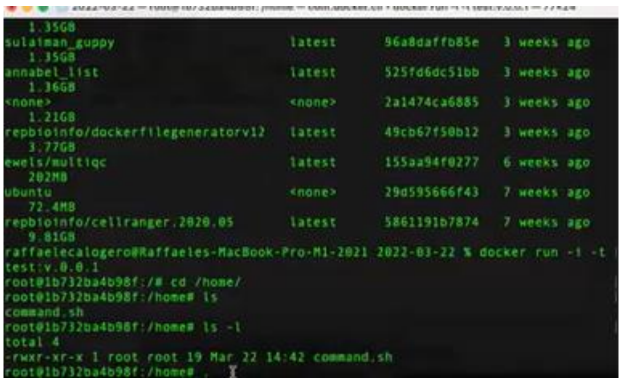
-y = if there is any new software that has been installed, it is going to be installed without asking me. Dockerfile does things without my intervention. In the folder there is a command.sh file, that I am going to copy in /home/ (where there is my ubuntu). This is a way I can create the script, which stays in the docker. So, dockerfile is a way to aggregate the information I want.



```
FROM ubuntu:20.04
RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y sudo
RUN sudo apt-get install -y nano
COPY ./command.sh /home/
RUN chmod +x /home/command.sh
```

now I move in the folder above the one where there is the command.sh → ls test → there are 3 files: Dockerfile, command.R and command.sh.

- There is a command that allows me to build a new image from scratch: docker build test -t test:v.0.0.1 (test:v.0.0.1 is the name that I want to give to my file; -t is the tagging). the output shows me what has been done.
- Docker images → there is the image test, with version v.0.0.1.



```
   1.35GB
sulaiman_guppy                   latest     96a8daffb85e   3 weeks ago
   1.35GB
annabel_list                     latest     525fd6dc51bb   3 weeks ago
   1.36GB
<none>                           <none>     2a1474ca6885   3 weeks ago
   1.21GB
repbioinfo/dockerfilegeneratorv12  latest   49cb67f50b12   3 weeks ago
   3.77GB
ewels/multiqc                    latest     155aa94f0277   6 weeks ago
   202MB
ubuntu                           <none>     29d595666f43   7 weeks ago
   72.4MB
repbioinfo/cellranger.2020.05    latest     5861191b7874   7 weeks ago
   9.81GB
raffaelecalogero@Raffaeles-MacBook-Pro-M1-2021 2022-03-22 % docker run -i -t
test:v.0.0.1
root@1b732ba4b98f:/# cd /home/
root@1b732ba4b98f:/home# ls
command.sh
root@1b732ba4b98f:/home# ls -l
total 4
-rwxr-xr-x 1 root root 19 Mar 22 14:42 command.sh
root@1b732ba4b98f:/home# 
```

- Docker run -i -t test:v.0.0.1
- Cd /home/
- Ls → I see the command.sh
- Ls -l

Command.sh is executable with chmod +x: if I type ./command.sh → output "Hello world"

- Nano command.sh allows me to modify the file (but the prof does not remember the commands anymore)

Exercise

1) Create a dockerfile using an R version that is available for R4.0 starting from rocker/r-ububtu → update → upgrade → install sudo and nano → find by yourself the R commands to install umap using the dockerfile: create the docker → go in → check that R starts → use the R command to install umap → search on Google what I can do to install a package in R.